TUGAS AKHIR - KI091391

RANCANG BANGUN MODUL EDITOR RUANGAN DAN FITUR SOSIAL PADA APLIKASI *GAME* SOSIAL FOOD MERCHANT SAGA PADA PERANGKAT ANDROID

Muamar Agus Salim
NRP 5110100 148

Dosen Pembimbing
Imam Kuswardayan, S.Kom., M.T.
Dwi Sunaryono, S.Kom., M.Kom.

JURUSAN TEKNIK INFORMATIKA
Fakultas Teknologi Informasi
Institut Teknologi Sepuluh Nopember
Surabaya 2014

FINAL PROJECT - KI091391

DESIGN AND IMPLEMENTATION OF ROOM EDITOR AND SOCIAL FEATURE MODULE IN SOCIAL GAME FOOD MERCHANT SAGA ON ANDROID DEVICE

Muamar Agus Salim
NRP 5110100 148

Advisor
Imam Kuswardayan, S.Kom., M.T.
Dwi Sunaryono, S.Kom., M.Kom.

DEPARTMENT OF INFORMATICS
Faculty of Information Technology
Institut Teknologi Sepuluh Nopember
Surabaya 2014

Rancang Bangun Modul Editor Ruangan dan Fitur Sosial pada Aplikasi *Game* Sosial Food Merchant Saga pada Perangkat Android

Nama Mahasiswa	: Muamar Agus Salim
NRP	: 5110100148
Jurusan	: Teknik Informatika FTIf-ITS
Dosen Pembimbing 1	: Imam Kuswardayan, S.Kom., M.T.
Dosen Pembimbing 2	: Dwi Sunaryono, S.Kom., M.Kom.

ABSTRAK

Game sosial adalah game yang memasukkan aktivitas-aktivitas sosial dalam permainan, sehingga pemain dapat saling berinteraksi satu sama lain dalam berbagai bentuk sesuai dengan karakteristik game yang dimainkan. Perkembangan game sosial mengalami peningkatan yang cukup pesat dalam dekade terakhir ini. Hampir semua jenis aliran game yang dipublikasikan kepada pengguna memiliki fitur sosial yang diterapkan dalam alur permainan. Mulai dari fitur sosial yang minor hingga dalam skala yang besar dapat ditemukan pada game yang beredar saat ini. Perkembangan dalam aspek fitur sosial dalam pengembangan game tidak lepas dari peran perkembangan pesat teknologi internet. Peningkatan perkembangan yang sangat pesat pada teknologi internet mendorong kebutuhan manusia untuk lebih dapat berinteraksi dengan sesama tanpa dibatasi jarak dan waktu. Hal ini melatarbelakangi perkembangan game sosial yang cukup pesat pada dekade terakhir. Sehingga salah satu yang menjadi kebutuhan dasar sebuah game yang memiliki fitur sosial adalah kebutuhan koneksi terhadap internet karena fitur-fitur sosial tersebut diakses melalui koneksi internet.

Modul editor ruangan dan fitur sosial pada game Food Merchant Saga mengimplementasikan pola permainan sosial dalam bentuk simulasi berbisnis pada kerangka pengembangan game dengan kakas Unity. Pola permainan simulasi dilakukan dengan aktivitas mengelola pujasera, menata tata letak ruangan pujasera yang dapat dikunjungi satu sama lain oleh pemain yang saling berteman. Kerangka ruangan pujasera dibangun dalam posisi penampang isometric dengan penerapan berbasis ubin untuk menghemat alokasi pemrosesan permainan. Didukung pula dengan integrasi pada media sosial Facebook melalui layanan API Facebook untuk

menerapkan fitur-fitur sosial yang lebih mudah dan dapat dijangkau oleh setiap orang yang memiliki akun media sosial Facebook. Selain itu karena memang layanan API Facebook ditujukan untuk mengembangkan fitur sosial game yang terintegrasi dengan Facebook sebagai salah satu media sosial terbesar di dunia.

Modul editor ruangan dan fitur sosial pada game Food Merchant Saga diuji dengan pengujian fungsionalitas dan pengujian integritas. Pengujian fungsionalitas dilakukan untuk menguji bahwa modul dapat diimplementasikan secara terpisah pada kerangka pengembangan mandiri. Pengujian fungsionalitas dilakukan untuk menguji bahwa modul dapat diimplementasikan secara terintegrasi pada kerangka pengembangan game Food Merchant Saga bersama dengan modul-modul lainnya. Semua skenario pada pengujian fungsionalitas maupun pengujian pengujian integritas memberikan hasil sesuai dengan keluaran yang diharapkan.

Kata kunci**: **API Facebook, Game Berbasis Ubin, Game Sosial, Isometric, Unity.

Design and Implementation of Room Editor and Social Feature Module in Social *Game* Food Merchant Saga on Android Device

Student Name	: Muamar Agus Salim
Student ID	: 5110100148
Major	: Teknik Informatika FTIf-ITS
Advisor 1	: Imam Kuswardayan, S.Kom., M.T.
Advisor 2	: Dwi Sunaryono, S.Kom., M.Kom.

ABSTRACT

Social game is game that incorporating social activities in the game, so that players can interact with each other in accordance with the characteristics of the various forms of the game being played. The development of social games has increased tremendeously in the last decade. Almost all types of games that published to the users have social features implemented in the groove of the game. Ranging from minor to social features in a large scale can be found in the game currently available. Developments in the aspect of social features in game development can not be separated from the role of the rapid development of internet technology. Improved rapid growth in internet technology encourages the human need to be able to interact with each other without being limited by distance and time. It is behind the development of social games quite rapidly in the last decade. Thus becoming one of the basic requirements of a game that has social features is a need for internet connection for social features are accessed via an internet connection.

Module of room editor and social features in Food Merchant Saga implements a pattern in the form of social games business simulations in the framework of game development with Unity. Pattern simulation implemented with activities managing food court, organizing food court room layout that can be visited each other by mutual friends of players. The frame of food court room developed in cross-section isometric with tile-based implementation for processing allocation save in the game. Also with the integration of social media named Facebook through the Facebook API services to implement social features that easier and can be reached by

everyone who has a Facebook social media accounts. Moreover, because the Facebook API service is aimed at developing social gaming features that integrates with Facebook as one of the biggest social media in the world.

Module of room editor and social features in Food Merchant Saga are tested with functionality testing and integrity testing. Functionality testing carried out to test that the module can be implemented separately in the self-development framework. Integrity testing carried out to test that the module can be implemented as an integrated module in the framework Food Merchant Saga development along with other modules. All scenarios in functionality and integrity testing gives results in accordance with the expected output planned.

Keyword**: **Facebook API, Isometric, Social Games, Tile-based Games, Unity.

LEMBAR PENGESAHAN

Rancang Bangun Modul Editor Ruangan dan Fitur Sosial pada Aplikasi *Game* Sosial Food Merchant Saga pada Perangkat Android

TUGAS AKHIR

Diajukan Guna Memenuhi Salah Satu Syarat
Memperoleh Gelar Sarjana Komputer
pada
Bidang Studi Rekayasa Perangkat Lunak
Program Studi S-1 Jurusan Teknik Informatika
Fakultas Teknologi Informasi
Institut Teknologi Sepuluh Nopember

Oleh :
MUAMAR AGUS SALIM
NRP : 5110 100 148

Disetujui oleh Dosen Pembimbing tugas akhir :

IMAM KUSWARDAYAN, S.Kom., M.T. .....
NIP: 197612152003121001 (Pembimbing 1)

DWI SUNARYONO, S.Kom., M.Kom. .....
NIP: 197205281997021001 (Pembimbing 2)

SURABAYA
JULI 2014

KATA PENGANTAR

Puji syukur kepada Allah Yang Maha Esa atas segala karunia dan rahmat-Nya penulis dapat menyelesaikan tugas akhir yang berjudul :

"Rancang Bangun Modul Editor Ruangan dan Fitur Sosial Pada Aplikasi Game Sosial Food Merchant Saga Pada Perangkat Android"

Harapan dari penulis semoga apa yang tertulis di dalam buku tugas akhir ini dapat bermanfaat bagi pengembangan ilmu pengetahuan saat ini, serta dapat memberikan kontribusi yang nyata.

Dalam pelaksanaan dan pembuatan tugas akhir ini tentunya sangat banyak bantuan yang penulis terima dari berbagai pihak, tanpa mengurangi rasa hormat penulis ingin mengucapkan terima kasih sebesar-besarnya kepada:

1. Ibu Mimik Mariani, Bapak Abdul Shomad, dan keluarga yang selalu memberikan dukungan penuh untuk menyelesaikan tugas akhir ini.
2. Bapak Imam Kuswardayan dan Bapak Dwi Sunaryono selaku dosen pembimbing yang telah bersedia meluangkan waktu untuk memberikan petunjuk selama proses pengerjaan tugas akhir ini.
3. Bapak dan Ibu dosen Jurusan Teknik Informatika ITS yang telah banyak memberikan ilmu dan bimbingan yang tak ternilai harganya bagi penulis.
4. Seluruh staf dan karyawan FTIf ITS yang banyak memberikan kelancaran administrasi akademik kepada penulis.
5. Rekan-rekan Square Enix Smilework Surabaya, Mas Michael, Mas Mahdi, Mas Aries, dan Mas Junian yang selalu siap mendorong dan memberikan bantuan ketika penulis mengalami kesulitan.
6. Teman-teman penghuni Lab Pengembangan *Game* selama proses pengerjaan tugas akhir dan teman-teman lain yang telah memberikan banyak dukungan dan semangat kepada penulis.

7. Rekan satu tim *game* sosial *Food Merchant Saga*, Aminudin, Fadjar, Riduwan dan Yasin yang telah banyak berjuang bersama penulis mengerjakan tugas akhir.
8. Teman-teman sepermainan selama kuliah, Fadlika, Hani, Fahmi, Agus Tri, Agus, Iqbal, Ika, Luluk, Anita, Ervina, Dila, Ninis, dan semua yang tidak tersebutkan satu persatu, terima kasih telah berteman dengan saya.
9. Teman-teman angkatan 2010 jurusan Teknik Informatika ITS yang telah menemani perjuangan selama 4 tahun ini atas saran, masukan, dan dukungan terhadap pengerjaan tugas akhir ini.
10. Rekan-rekan dakwah di Yayasan Uswah Student Center yang telah memberikan banyak pelajaran hidup dan kemakluman yang tinggi agar penulis dapat menyelesaikan tugas akhir ini.
11. Serta pihak-pihak lain yang tidak dapat disebutkan disini yang telah banyak membantu penulis dalam penyusunan tugas akhir ini.

Penulis telah berusaha sebaik-baiknya dalam menyusun tugas akhir ini, namun penulis mohon maaf apabila terdapat kekurangan, kesalahan maupun kelalaian yang telah penulis lakukan. Kritik dan saran yang membangun dapat disampaikan sebagai bahan perbaikan selanjutnya.

Surabaya, Juni 2014

Muamar Agus Salim

DAFTAR ISI

DAFTAR GAMBAR

DAFTAR KODE SUMBER

DAFTAR TABEL

BAB I
PENDAHULUAN

Pada bab ini akan dipaparkan mengenai garis besar tugas akhir yang meliputi latar belakang, tujuan, rumusan dan batasan permasalahan, metodologi pembuatan tugas akhir, dan sistematika penulisan.

1.1. Latar Belakang

Sebagai makhluk sosial, manusia saling berinteraksi dengan manusia lainnya. Teknologi yang berkembang pesat, membuat cara manusia dalam berinteraksi dengan manusia lain menjadi berkembang. Mulai dari interaksi secara langsung, surat-menyurat, sambungan telepon, surat elektronik, pesan singkat (SMS), *chatting*, situs media sosial, hingga interaksi menggunakan media *game*.

Dalam beberapa dekade terakhir perkembangan bentuk *game* berlangsung cukup pesat. Perkembangan ini terjadi pada berbagai aspek pada *game*, seperti cara permainan, media permainan, dan bentuk permainan. Saat ini *game* tidak sebatas permainan yang dimainkan melalui perangkat *console* yang hanya dapat dimainkan satu atau dua orang dalam waktu yang sama, tetapi banyak pemain dapat bermain *game* yang sama tanpa harus di waktu yang sama dan dengan perangkat yang sama. Fenomena ini terjadi karena selain didorong oleh pesatnya perkembangan teknologi di dunia *digital*, juga secara karakteristik bermain dapat mendorong terbentuknya lingkungan sosial [1]. Hal ini mendorong perkembangan *game* menjadi media berinteraksi bagi para pemainnya. Sehingga berkembang secara menjamur berbagai *game* yang menawarkan model permainan secara masif dengan berbagai bentuk, baik secara langsung (*synchronous*) maupun tidak langsung (*asynchronous*) kepada para pemainnya. *Game* yang menyediakan fungsi interaksi sosial seperti ini disebut *Game* Sosial.

Selain itu perkembangan bentuk permainan *game* juga terjadi secara cepat. Saat ini *game* yang banyak beredar lebih banyak menggunakan unsur-unsur realitas interaksi sosial pada kehidupan

nyata untuk diterapkan dalam *game* tersebut. Aktivitas zaman kerajaan, aktivitas perdagangan, aktivitas bercocok tanam dan berbagai aktivitas sehari-hari manusia telah banyak diterapkan menjadi *game*. Hingga saat ini *game* bertema simulasi realitas masih menjadi salah satu tema yang cukup populer.

Untuk itu, muncul sebuah gagasan dalam proyek tugas akhir ini untuk mengembangkan aplikasi *game* sosial bertema simulasi bisnis kuliner yang berjudul Food Merchant Saga yang berjalan pada perangkat *smartphone* dengan sistem operasi Android. Dengan *game* ini diharapkan menjadi sarana interaksi sosial yang baik bagi para pemain *game* dan memberikan nilai edukasi *entrepreneurship* melalui simulasi bisnis kuliner.

1.2. Tujuan

Tujuan dari penyusunan tugas akhir ini adalah untuk menghasilkan modul editor ruangan dan fitur sosial yang merupakan bagian dari *game* sosial Food Merchant Saga yang berjalan pada platform Android. Selain itu pembuatan *game* sosial ini juga dikarenakan adanya kerjasama dengan SquareEnix yang merupakan *developer* dan *publisher game.*

1.3. Rumusan Permasalahan

Rumusan masalah yang diangkat dalam tugas akhir ini adalah sebagai berikut.

1. Bagaimana rancangan dan implementasi objek ruangan dalam lingkungan pengembangan Unity.
2. Bagaimana rancangan dan implementasi *tile-based isometric interface* pada *game* sosial Food Merchant Saga.
3. Bagaimana implementasi editor ruangan pada penampang muka isometrik pada *game* sosial Food Merchant Saga.
4. Bagaimana menerapkan fitur sosial menggunakan API Facebook pada lingkungan pengembangan Unity.

1.4. Batasan Permasalahan

Lingkup masalah yang dibahas dalam tugas akhir ini memiliki beberapa batasan, di antaranya sebagai berikut.

1. Tugas akhir ini dibangun menggunakan *tools* Unity versi 4.3 dengan bahasa pemrograman C#.
2. Media sosial yang diintegrasikan dengan aplikasi *game* Food Merchant Saga adalah Facebook menggunakan API Facebook.
3. Sistem operasi Android yang digunakan adalah versi 4.0 Ice Cream Sandwich atau versi yang lebih baru.

1.5. Metodologi

Langkah-langkah yang ditempuh dalam pengerjaan tugas akhir ini yaitu:

1. **Studi literatur**

   Mengumpulkan dan menggali informasi dan literatur yang diperlukan dalam proses perancangan dan implementasi sistem yang akan dibangun. Literatur yang digunakan adalah sebagai berikut.

   a. Teknik pemrograman untuk pengembangan *game tile-based* dengan antarmuka *isometric* menggunakan Unity dengan bahasa pemrograman C#.
   b. Penggunaan Unity untuk menciptakan objek/model serta memanipulasi objek/model dalam *game*.
   c. Konsep pengaksesan informasi Facebook menggunakan Graph API Facebook.

2. **Analisis dan Perancangan Sistem**

   Pada tahap ini dilakukan analisis dan pendefinisian kebutuhan sistem untuk mengetahui permasalahan yang sedang dihadapi. Selanjutnya, dirumuskan rancangan sistem yang dapat memberi solusi terhadap permasalahan tersebut. Langkah yang akan digunakan pada tahap ini sebagai berikut:

   a. Analisis kebutuhan fungsional dan nonfungsional pada modul editor ruangan dan fitur sosial yang akan dibangun.

b. Perancangan integrasi dengan media sosial Facebook menggunakan Graph API Facebook.

3. Implementasi

Pada tahap ini dilakukan pembuatan perangkat lunak yang merupakan implementasi dari rancangan yang telah dibuat sebelumnya. Perincian tahap ini adalah sebagai berikut:

a. Implementasi penampang muka *isometric* pada *game* Food Merchant Saga.
b. Implementasi modul editor ruangan untuk *game* Food Merchant Saga pada Unity.
c. Implementasi integrasi dengan media sosial Facebook menggunakan Unity Facebook SDK.

4. Pengujian dan evaluasi

Pada tahap ini akan dilakukan pengujian terhadap perangkat lunak menggunakan skenario yang telah disiapkan sebelumnya. Uji coba dan evaluasi dilakukan untuk mencari masalah yang mungkin timbul, mengevaluasi jalannya program, dan mengadakan perbaikan jika ada kekurangan. Pengujian dilakukan menggunakan pengujian *blackbox* dengan jenis pengujian yaitu pengujian fungsional dan sistem (*fungsional and system testing*), yaitu pengujian terhadap kesesuaian modul yang dikerjakan dengan spesifikasi fungsional serta terhadap lingkungan sistem yang dikehendaki atau lebih dari yang dikehendaki [2]. Aspek-aspek yang diuji dengan pengujian fungsional dan sistem mencakup antara lain:

a. Pengujian kebenaran (*correctness testing*). Pengujian ini dilakukan untuk mengetahui apakah fungsi-fungsi atau fitur-fitur modul atau program telah sesuai dengan spesifikasi kebutuhan fungsional.
b. Pengujian modularitas dan integritas (*modularity & integrity testing*). Pengujian ini dilakukan untuk mengetahui apakah modul dapat diuji dan dijalankan secara terpisah tanpa terintegrasikan dengan modul yang lainnya maupun ketika

diintegrasikan dengan modul lainnya untuk membentuk kesatuan *game* Food Merchant Saga. Hal ini diperlukan untuk kemungkinan jika modul-modul lain belum selesai dikembangkan.

5. **Penyusunan buku tugas akhir**
   Pada tahap ini dilakukan pendokumentasian dan pelaporan dari seluruh konsep, dasar teori, implementasi, proses yang telah dilakukan, dan hasil-hasil yang telah didapatkan selama pengerjaan tugas akhir.

1.6. Sistematika Penulisan

Buku tugas akhir ini bertujuan untuk mendapatkan gambaran dari pengerjaan tugas akhir ini. Selain itu, diharapkan dapat berguna untuk pembaca yang tertarik untuk melakukan pengembangan lebih lanjut. Secara garis besar, buku tugas akhir terdiri atas beberapa bagian seperti berikut ini.

Bab I Pendahuluan
Bab ini berisi latar belakang masalah, tujuan dan manfaat pembuatan tugas akhir, permasalahan, batasan masalah, metodologi yang digunakan, dan sistematika penyusunan tugas akhir.

Bab II Dasar Teori
Bab ini membahas beberapa teori penunjang yang berhubungan dengan pokok pembahasan dan mendasari pembuatan tugas akhir ini.

Bab III Analisis dan Perancangan Sistem
Bab ini membahas mengenai perancangan perangkat lunak yang meliputi perancangan data, arsitektur dan proses dan antarmuka pada perangkat lunak yang dibangun.

Bab IV Implementasi
Bab ini berisi implementasi dari perancangan perangkat lunak.

Bab V Pengujian dan Evaluasi

Bab ini membahas pengujian dengan metode pengujian *blackbox* untuk mengetahui penilaian pada aspek kegunaan (*usability*), kebenaran (*correctness*), tekanan serta performa (*stress & performance testing*), dan modularitas serta integritas (*modularity & integrity*) dari perangkat lunak.

Bab VI Kesimpulan

Bab ini berisi kesimpulan dari hasil pengujian yang dilakukan. Bab ini membahas saran-saran untuk pengembangan perangkat lunak lebih lanjut.

Daftar Pustaka

Merupakan daftar referensi yang digunakan untuk mengembangkan tugas akhir.

Lampiran

Merupakan bab tambahan yang berisi daftar istilah yang penting pada aplikasi ini.

BAB II
DASAR TEORI

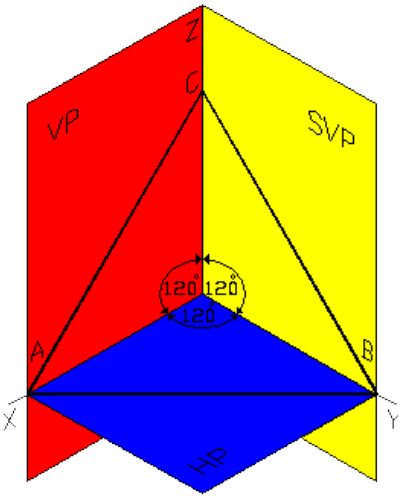
Pada bab ini akan dibahas mengenai teori-teori yang menjadi dasar dari pembuatan tugas akhir. Teori-teori tersebut adalah *Social Game*, Tampilan *Isometric*, API Facebook, *Tile-based Games*, dan Unity3D.

2.1. *Social Game*

Social Game adalah *game* yang dimainkan lebih dari satu orang yang berada pada suatu jaringan sosial tertentu, sehingga pemain bisa mengundang atau mengajak teman untuk melakukan aktivitas tertentu dalam *game*. *Game* sosial memiliki dua sifat, yaitu *synchronous*, yang mengharuskan para pemain untuk bermain bersamaan dalam waktu yang sama. Kemudian *asynchronous*, yang memungkinkan para pemain untuk bermain *game* yang sama tanpa harus dalam waktu yang bersamaan [3].

2.2. Tampilan *Isometric*

Tampilan *isometric* menggunakan dasar dari proyeksi *ortographic*, yaitu cara menampilkan objek 3 dimensi dari arah yang berbeda. Biasanya ditampilkan dari arah depan (front view), samping (side view) dan atas (plan view) sehingga semua bagian penting dari objek tersebut dapat terlihat [4]. Ilustrasi tampilan *isometric* seperti pada Gambar 2.1.

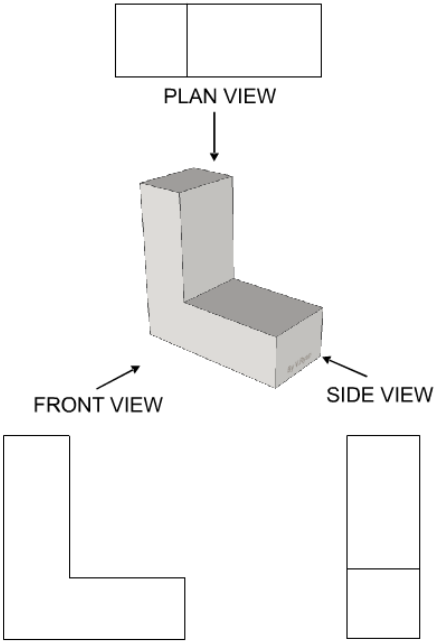


Gambar 2.1 Konsep Tampilan *Isometric*

Pada penerapan dalam dunia pembuatan *game*, tampilan *isometric* adalah cara menampilkan tiga sisi objek 3 dimensi kedalam tampilan 2 dimensi dengan memberikan tiga sumbu koordinatnya sudut yang sama, yaitu 120 derajat, di mana secara sesungguhnya sudut antara ketiga sumbu koordinat adalah 90 derajat [5]. Dengan begitu tampilan *isometric* memberikan cita rasa 3 dimensi pada gambar yang sesungguhnya hanya 2 dimensi.

2.2.1. Proyeksi *Orthographic*

Proyeksi *orthographic* diperoleh apabila sinar proyeksi tegak lurus dengan bidang proyeksi. Proyeksi *orthographic* sering digunakan untuk menghasilkan tampak depan, tampak belakang, tampak samping, dan tampak atas dari sebuah benda atau disebut sebagai *Multiview Orthographic*. Tampak atas, tampak belakang dan tampak dari samping sebuah benda sering disebut sebagai *elevation.* Sedangkan tampak dari atas disebut sebagai *plan view.*



Gambar 2.2 Konsep Tampilan *Orthographic* Jenis *Axonometric*

Proyeksi *orthographic* yang menampakkan lebih dari satu permukaan disebut juga dengan *axonometric* seperti pada Gambar 2.2. Terdapat tiga jenis proyeksi *orthographic* yang bersifat *axonometric*, yaitu *isometric*, *diametric*, dan *trimetric*.

2.3. API Facebook

API Facebook adalah sarana utama yang disediakan oleh Facebook bagi pengembang perangkat lunak untuk mengakses data ke dalam dan ke luar Facebook *Social Graph*. API Facebook merupakan API berbasis HTTP tingkat rendah yang dapat digunakan untuk melakukan pemrosesan data, dan berbagai fitur Facebook lainnya.

API Facebook dibuat untuk pengaksesan data '*Social Graph'* yang dimiliki Facebook. Data *'Social Graph'* tersebut terdiri dari *nodes* (representasi objek dalam Facebook, seperti pengguna, foto,

dan halaman), *edges* (hubungan antara satu objek dengan objek lainnya, seperti foto milik pengguna, dan komentar dalam status), *fields* (informasi yang dimiliki oleh suatu objek, seperti nama pengguna, tanggal lahir pengguna) [6].

2.3.1. API

API atau *Application Programming Interfaces* adalah sekumpulan perintah, fungsi, aturan, dan protokol yang digunakan untuk menjembatani komponen suatu perangkat lunak berinteraksi dengan komponen perangkat lunak lainnya. API seringkali diterapkan oleh pengembang software sebagai sebuah tata cara bagi seseorang untuk mengakses suatu komponen tertentu dari perangkat lunak. Pada umumnya, API diterapkan dalam bentuk *library* yang mencakup *routines*, struktur data, objek kelas, dan variabel. Pada beberapa kasus penerapan API hanya berupa protokol atau spesifikasi yang diterapkan dalam suatu kerangka kerja [7].

2.3.2. Open Graph Facebook

Open Graph Facebook adalah fitur yang disediakan oleh Facebook untuk para *developer* di halaman *Developer* Facebook untuk secara lebih mudah membuat berbagai skenario cerita ketika berbagi sesuatu ke Facebook. Fitur ini dahulunya adalah fitur *Feed Dialog*, dan dikembangkan oleh Facebook supaya lebih terstruktur dengan menerapkan Open Graph. Misalnya *developer* aplikasi GoodReads untuk memenuhi kebutuhan berbagi aktivitas sesuai dengan fungsi-fungsi aplikasinya, membuat sebuah skenario cerita *Like a Book*. Untuk melakukannya, *developer* perlu membuat pada halaman *dashboard* aplikasi Facebook sebuah *Object* dan sebuah *Action*. *Like* adalah *Action* dan *Book* adalah *Object*. Kemudian dibuat skenario cerita dengan menggabungkan *Action* dengan *Object* [6].

2.3.3. Unity Facebook SDK

Unity Facebook SDK adalah sebuah SDK(*Software Development Kit*) yang disediakan oleh Facebook untuk pengembangan API Facebook dalam kerangka kerja Unity. Unity Facebook SDK menyediakan koleksi komprehensif berbagai fitur

sosial Facebook yang diakses menggunakan API Facebook. Facebook menyediakan SDK ini untuk meningkatkan integrasi antara Facebook dengan Unity. SDK ini berjalan pada API versi 2.0 [8].

2.4. Sistem Operasi Android

Android adalah sebuah sistem operasi yang dikembangkan secara khusus untuk perangkat bergerak dan layar sentuh. Sistem operasi ini berbasis Linux. Sistem operasi ini dikembangkan pertama kali oleh Android, Inc. dengan dukungan dana dari Google. Hingga pada tahun 2005 Android, Inc. dibeli oleh Google dan pengembangan sistem operasi Android berada di bawah Google.

Android menjadi sistem operasi yang paling banyak digunakan pada perangkat bergerak dan layar sentuh di seluruh dunia saat ini. Fakta ini terjadi karena Google menjadikan Android sebagai sistem operasi open source di bawah lisensi Apache sehingga dapat digunakan dan dimodifikasi secara bebas oleh para pengembang dan pemilik merek perangkat keras [9].

2.5. *Tile-based Games*

Tile-based games adalah bentuk permainan dengan merepresentasikan lingkungan *game* dalam pola persegi yang berfungsi sebagai *map*. Penerapan *tile-based* pada pengembangan *game* yang berjalan melalui komputer terinspirasi dari permainan-permainan tradisional yang menggunakan balok-balok persegi. Penggunaan konsep *tile-based* pada pengembangan *game* yang berjalan melalui komputer ini telah dimulai cukup lama, yaitu saat komputer belum memiliki kecepatan GHz dan memori ratusan MB. Kebutuhan untuk meningkatkan performa *game* dengan keterbatasan kemampuan komputer mendorong *developer game* untuk menerapkan konsep *tile-based*. Dengan konsep *tile-based developer* dapat memotong suatu gambar menjadi potongan-potongan persegi yang lebih kecil sehingga tidak memakan memori terlalu banyak [10].

2.6. Unity

Unity merupakan sebuah perangkat pengembangan *game* yang terpadu dan memiliki banyak fungsionalitas untuk membangun *game* atau objek 3 dimensi. Unity menyediakan berbagai fungsionalitas siap pakai seperti pencahayaan, efek khusus, animasi, dan aspek fisika. Unity dapat digunakan untuk mengembangkan baik *game* 3 dimensi maupun *game* 2 dimensi. Selain itu *game* yang dikembangkan menggunakan Unity dapat dipublikasikan dalam berbagai perangkat seperti Mac, PC, Linux, Windows Store, Windows Phone 8, Android, Blackberry 10, Wii U, PS3 dan Xbox 360. Versi terbaru dari Unity saat ini adalah Unity 4.3 [11].

BAB III
ANALISIS DAN PERANCANGAN SISTEM

Bab ini membahas tahap analisis permasalahan dan perancangan dari sistem yang akan dibangun. Analisis permasalahan membahas permasalahan yang diangkat dalam pengerjaan tugas akhir. Analisis kebutuhan mencantumkan kebutuhan-kebutuhan yang diperlukan perangkat lunak. Selanjutnya dibahas mengenai perancangan sistem yang dibuat. Pendekatan yang dibuat dalam perancangan ini adalah pendekatan berorientasi objek. Perancangan direpresentasikan dengan diagram UML (*Unified Modelling Language*).

3.1. Analisis

Tahap analisis dibagi menjadi beberapa bagian antara lain analisis permasalahan, deskripsi umum sistem, analisis aktor, spesifikasi kebutuhan perangkat lunak, dan skenario kasus penggunaan.

3.1.1. Analisis Permasalahan

Permasalahan utama yang diangkat dalam pembuatan tugas akhir ini adalah bagaimana merancang halaman utama permainan yang berbentuk *isometric* dan *tile-based* untuk *game* Food Merchant Saga yang berjalan pada perangkat Android. Permasalahan kedua adalah bagaimana menerapkan fitur sosial pada *game* Food Merchant Saga yang terhubung dengan media sosial Facebook.

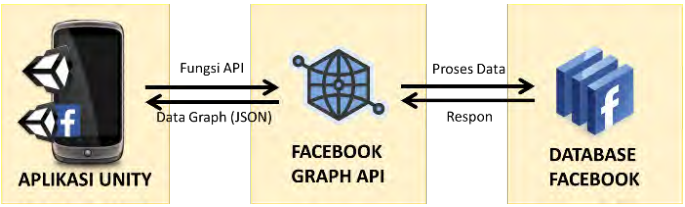
Perancangan penampang *isometric* dapat diterapkan dengan menentukan posisi konstan perbedaan setiap tingkat *tile* yang tersusun merepresentasikan sebuah bidang datar persegi. Unity dengan fungsi utamanya sebagai pengembangan berbasis 3 dimensi memberikan kemudahan dengan fungsi proyeksi kamera yang dimilikinya memungkinkan untuk menerapkan proyeksi *orthogonal*. Tetapi karena pengembangan *game* Food Merchant Saga ini berbasis 2 dimensi sehingga tidak memungkinkan untuk menerapkan fitur tersebut. Oleh karena itu cara yang memungkinkan untuk dilakukan

adalah mensiasati objek *tile* yang seharusnya berbentuk persegi menjadi bentuk belah ketupat untuk memenuhi kriteria proyeksi *isometric*.

Sebagai *game* bertipe sosial maka pemain *game* Food Merchant Saga bisa melakukan interaksi dengan pemain lainnya yang juga bermain *game* Food Merchant Saga. Oleh karena itu perancangan fitur sosial akan dibangun dengan terintegrasi pada media sosial yang telah dikenal secara umum oleh masyarakat, yaitu Facebook. Integrasi dilakukan dengan memanfaatkan API yang telah dibangun oleh Facebook menggunakan Unity Facebook SDK. Dengan integrasi pada Facebook, memungkinkan pemain untuk berbagi aktivitas permainan melalui Facebook, mengirim pesan kepada teman melalui Facebook dan fitur-fitur lainnya.

3.1.2. Arsitektur Sistem

Modul yang dibangun pada penelitian ini memanfaatkan Unity Facebook SDK sebagai kakas bantu untuk berkomunikasi dengan API Facebook. Arsitektur sistem dari modul yang dibangun melibatkan tiga lapisan sistem, yaitu lapisan aplikasi Unity pada perangkat Android, lapisan API Facebook, dan lapisan *database* Facebook. Seperti terlihat pada Gambar 3.1, pada lapisan aplikasi dimana modul dimplementasikan menggunakan Unity Facebook SDK, melakukan pemanggilan fungsi API. Kemudian pada *service* API Facebook melakukan permintaan data sesuai pemanggilan fungsi dari aplikasi kepada *database server* Facebook. Selanjutnya *database server* Facebook merespon dan oleh *service* API Facebook dikembalikan ke aplikasi sebagai sebuah data JSON.



Gambar 3.1 Arsitektur Sistem

3.1.3. Deskripsi Umum Sistem

Food Merchant Saga adalah aplikasi permainan simulasi berbisnis makanan nusantara dengan desain yang menarik di mana pemain harus mengatur berjalannya toko makanan yang berbentuk pujasera. Pemain akan diberi modal berupa sebuah pujasera kecil yang hanya berisi kedai bakso dan kedai nasi pecel. Kemudian seiring dengan meningkatnya level pujasera, maka luas bangunan pujasera juga akan bertambah. Pemain juga dapat membeli dan menambahkan kedai baru pada toko mereka dengan syarat luas toko dan uang untuk membeli kedai masih mencukupi. Pemain dapat meng-*upgrade* rombong yang mereka miliki sehingga akan meningkatkan kecepatan produksi serta akan membuka jenis menu makanan baru yang dapat diproduksi oleh rombong tersebut.

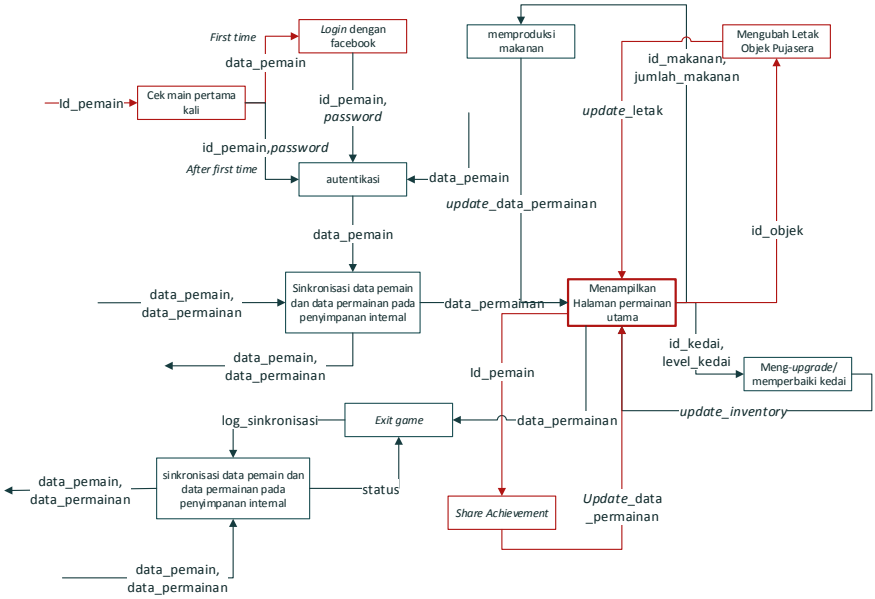
Permainan ini memiliki *genre* simulasi yang ditampilkan dengan tampilan *isometric*. Jenis makanan yang dapat dijual dalam *game* ini merupakan makanan-makanan khas yang berasal dari Indonesia.

Aplikasi yang dibangun pada Tugas Akhir ini adalah modul editor ruangan dan fitur sosial. Modul editor ruangan menampilkan antarmuka *game* menggunakan penampang *isometric*. Penampang *isometric* memungkinkan pemain untuk melihat objek dari 3 sudut koordinat sekaligus sehingga tampak seperti 3 dimensi. Dengan antarmuka tersebut, pemain bisa mengatur penempatan posisi objek-objek dalam pujasera yang dikelolanya.

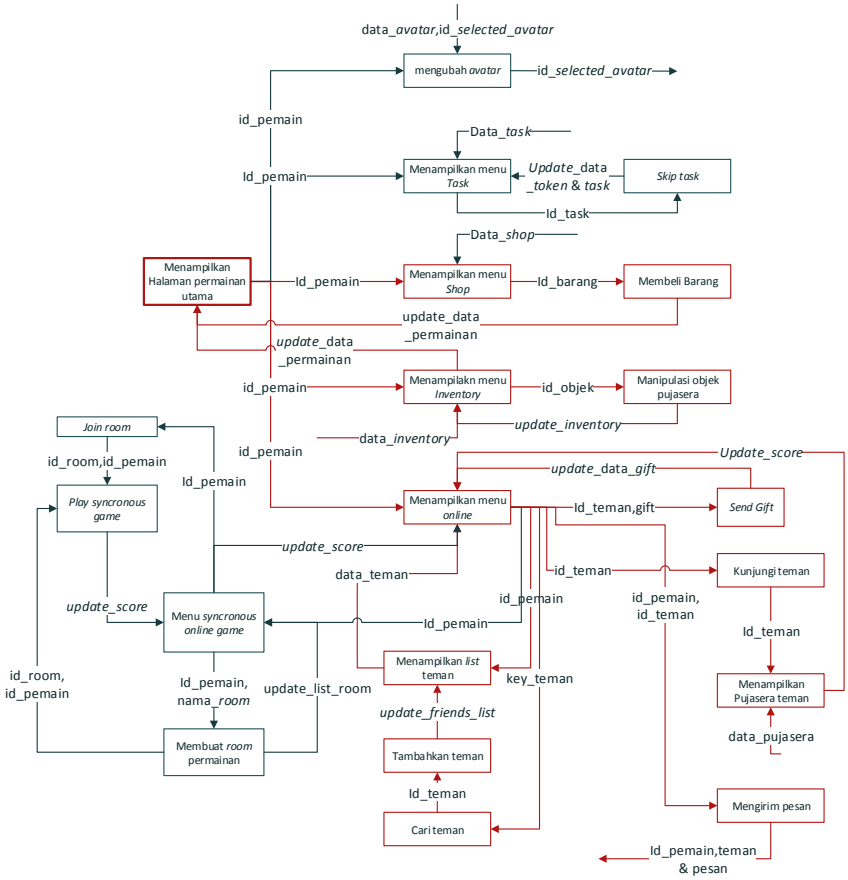
Pengembangan *game* ini menggunakan konsep berbasis *tile* atau *tile-based*. *Tile-based* adalah bentuk pengembangan *game* dengan menginterpretasikan lingkungan *game* dalam pola persegi yang dijadikan *map*. Dengan digambarkan dalam bentuk ubin-ubin, penempatan posisi objek-objek oleh pemain bisa lebih teratur karena telah ditentukan posisinya pasa setiap kotak ubin-ubin yang ada.

Sedangkan fitur sosial yang dikembangkan adalah dengan mengintegrasikan *game* Food Merchant Saga dengan media sosial Facebook. Facebook menyediakan API untuk menjembatani kebutuhan *developer* dalam memanfaatkan jaringan pertemanan seseorang yang terdaftar di Facebook dalam aplikasi atau perangkat

lunak yang sedang dikembangkan. Pada pengembangan *game* Food Merchant Saga penggunaan API Facebook juga mendukung fungsi sosial yang ingin ditawarkan dari *game* Food Merchant Saga.



Gambar 3.2 Diagram Balok Permainan Food Merchant Saga Bagian 1



Gambar 3.3 Diagram Balok Permainan Food Merchant Saga Bagian 2

Gambar 3.2 dan Gambar 3.3 merupakan potongan dari *block diagram* dari system yang akan diimplementasikan. *Block diagram* yang ditandai dengan warna merah adalah representasi dari modul editor ruangan dan fitur sosial yang akan diimplementasikan di dalam permainan.

3.1.4. Analisis Aktor

Aktor adalah entitas-entitas yang terlibat dan berinteraksi secara langsung dengan sistem. Dalam perangkat lunak, entitas bisa berupa manusia maupun sistem perangkat lunak lainnya. Pada permainan Food Merchant Saga, aktor yang berperan pada permainan ini hanya memiliki peran sebagai pengguna yang memainkan permainan Food Merchant Saga.

3.1.5. Spesifikasi Kebutuhan Perangkat Lunak

Bagian ini berisi kebutuhan perangkat lunak yang diuraikan secara rinci seperti kebutuhan perangkat lunak dalam sistem yang mencakup kebutuhan fungsional dan kebutuhan non fungsional. Pada penerapan modul yang dikerjakan hanya terdapat kebutuhan fungsional.

3.1.5.1. Kebutuhan Fungsional

Kebutuhan fungsional berisi proses-proses yang harus dimiliki sistem. Kebutuhan fungsional mendefinisikan layanan yang harus disediakan dan reaksi sistem terhadap masukan atau pada situasi tertentu. Daftar kebutuhan fungsional dapat dilihat pada Tabel 3.1

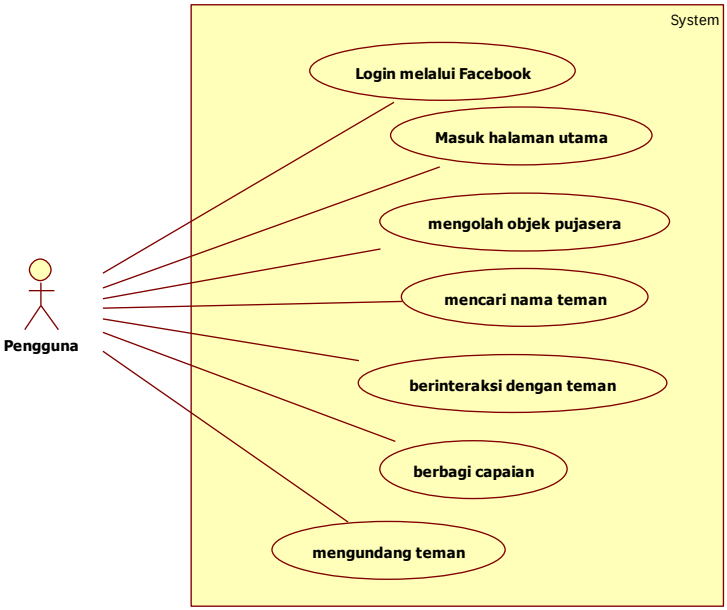
Tabel 3.1 Daftar Kebutuhan Fungsional Perangkat Lunak

Kode Kebutuhan	Kebutuhan Fungsional	Deskripsi
F-01	Melakukan pemeriksaan *login* yang terintegrasi dengan Facebook	Pengguna masuk permainan dengan menggunakan akun Facebook.
F-02	Menampilkan halaman permainan utama	Pengguna dapat memasuki halaman permainan utama yang berisi pujasera yang dikelolanya
F-03	Memanipulasi objek pada pujasera	Pengguna dapat mengatur dan mengolah objek-objek pada pujasera

Kode Kebutuhan	Kebutuhan Fungsional	Deskripsi
F-04	Menampilkan daftar teman	Pengguna dapat melihat daftar teman sesuai kata kunci yang diberikan
F-05	Mengirim pesan kepada teman	Pengguna dapat mengirim pesan kepada teman melalui Facebook
F-06	Mengunjungi pujasera teman	Pengguna dapat mengunjungi pujasera teman
F-07	Mengirim hadiah kepada teman	Pengguna dapat mengirim hadiah kepada teman
F-08	Berbagi capaian	Pengguna dapat berbagi capaian-capaian permainan melalui Facebook
F-09	Mengundang teman untuk bermain	Pengguna dapat mengundang teman melalui Facebook untuk bermain *game* ini

3.1.6. Skenario Kasus Penggunaan

Pada sub bab ini akan dijelaskan mengenai skenario kasus penggunaan yang terdapat pada sistem. Diagram kasus penggunaan dapat dilihat pada Gambar 3.4. Penjelasan dari masing-masing kasus dapat dilihat pada Tabel 3.2.



Gambar 3.4 Diagram Kasus Penggunaan

Tabel 3.2 Deskripsi Kasus Penggunaan

Kode Kebutuhan	Nama Kasus Penggunaan	Deskripsi
UC-01	*Login* melalui Facebook	Pengguna masuk permainan dengan menggunakan akun Facebook.
UC-02	Masuk halaman utama	Pengguna dapat memasuki halaman permainan utama yang berisi pujasera yang dikelolanya
UC-03	Mengolah objek pujasera	Pengguna dapat mengatur dan mengolah objek-objek pada pujasera
UC-04	Mencari nama teman	Mencari nama temannya yang juga bermain *game* ini

Kode Kebutuhan	Nama Kasus Penggunaan	Deskripsi
UC-05	Berinteraksi dengan teman	Pengguna dapat melakukan interaksi dengan teman, seperti mengirim pesan dan mengunjungi
UC-06	Berbari capaian	Pengguna dapat berbagi capaian-capaian permainan melalui Facebook
UC-07	Mengundang teman	Pengguna dapat mengundang teman Facebook yang belum pernah bermain *game* ini

3.1.6.1. Kasus Penggunaan *Login* dengan Facebook (UC-01)

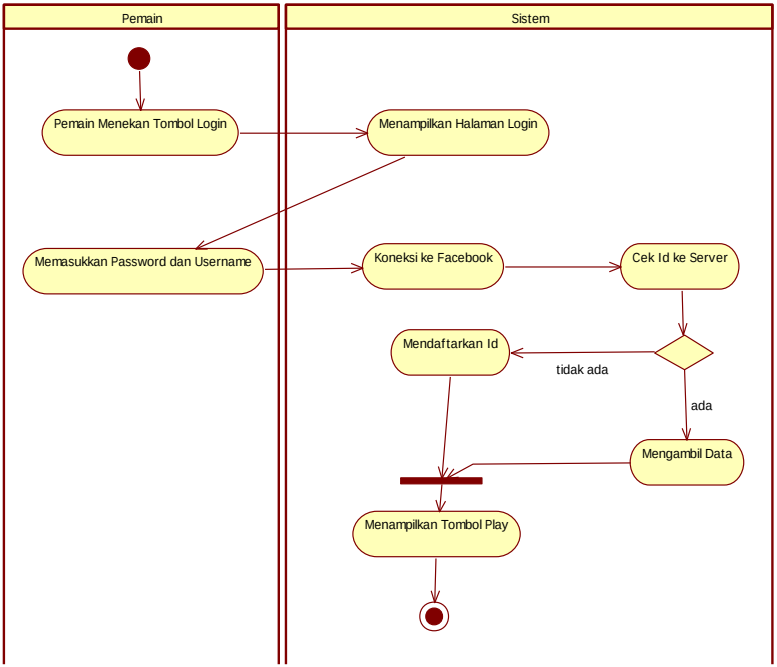
Kasus penggunaan *login* dengan Facebook ini dilakukan ketika pengguna memulai permainan di perangkat mereka. Untuk dapat bermain pertama kali, pemain harus login menggunakan *username* dan *password* Facebook mereka. Kemudian sistem akan mengolah apakah *id* Facebook pemain pernah memainkan *game* ini atau belum. Jika sudah, maka sistem akan menampilkan data permainan pengguna terakhir kali, jika belum sistem akan mendaftar *id* pengguna ke server. Spesifikasi kasus penggunaan *login* dengan Facebook dijelaskan dalam Tabel 3.3.

Tabel 3.3 Spesifikasi Kasus Penggunaan UC-01

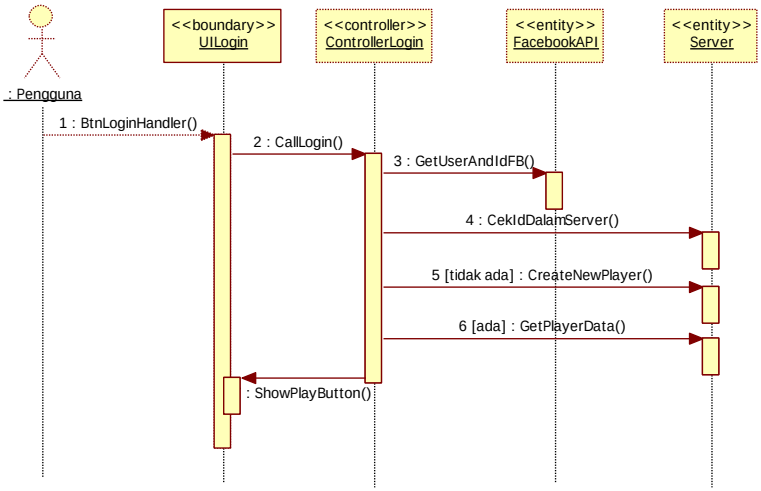
Nama Kasus Penggunaan	*Login* dengan Facebook
Kode	UC-01
Deskripsi	Pengguna masuk permainan dengan menggunakan akun Facebook.
Aktor	Pengguna *game* Food Merchant Saga
Kondisi Awal	-
Pemicu	Pengguna menekan tombol *login* yang terlihat pada layar untuk melakukan verifikasi *login*
Alur Normal	1. Pengguna menekan tombol *login* pada layar. 2. Sistem menampilkan halaman *login* Facebook.

	3. Pengguna memasukkan *username* dan *password* Facebook. 4. Sistem melakukan koneksi dengan Facebook untuk mendapatkan *id* Facebook pengguna. A1. Koneksi internet terputus. 5. Sistem melakukan koneksi dengan server untuk memeriksa apakah ada *id* pengguna di server. A1. Koneksi internet terputus. 6. Jika *id* pengguna ada, mengambil data permainan terakhir. 7. Jika *id* pengguna belum ada, mendaftarkan *id* ke server. 8. Tombol *login* berubah menjadi tombol *play* 9. Selesai.
Alur Alternatif	A1. Koneksi internet terputus saat melakukan *login* A1.1. Sistem memberi peringatan bahwa proses *login* terhenti A1.2. Menuju alur normal nomor 9.
Kondisi Akhir	Pengguna menerima respon hasil proses *login*.

Berdasarkan pada spesifikasi kasus penggunaan pada Tabel 3.3 yang melibatkan aktor dan sistem, maka dapat digambarkan diagram aktivitas untuk kasus penggunaan *login* dengan Facebook seperti pada Gambar 3.5 dan diagram alir pada Gambar 3.6.



Gambar 3.5 Diagram Aktivitas Kasus Penggunaan UC-01



Gambar 3.6 Diagram Alir Kasus Penggunaan UC-01

3.1.6.2. Kasus Penggunaan Masuk Halaman Utama (UC-02)

Kasus penggunaan masuk halaman utama terjadi setelah pengguna berhasil *login* ke dalam permainan. Terdapat tombol *play* yang jika ditekan pengguna akan mengarahkan ke halaman utama. Halaman utama menampilkan pujasera dan objek-objek di dalamnya. Jika pemain pernah *login* sebelumnya, data pujasera diambil dari penyimpanan internal. Penyimpanan internal menyimpan data permainan pemain antara lain.

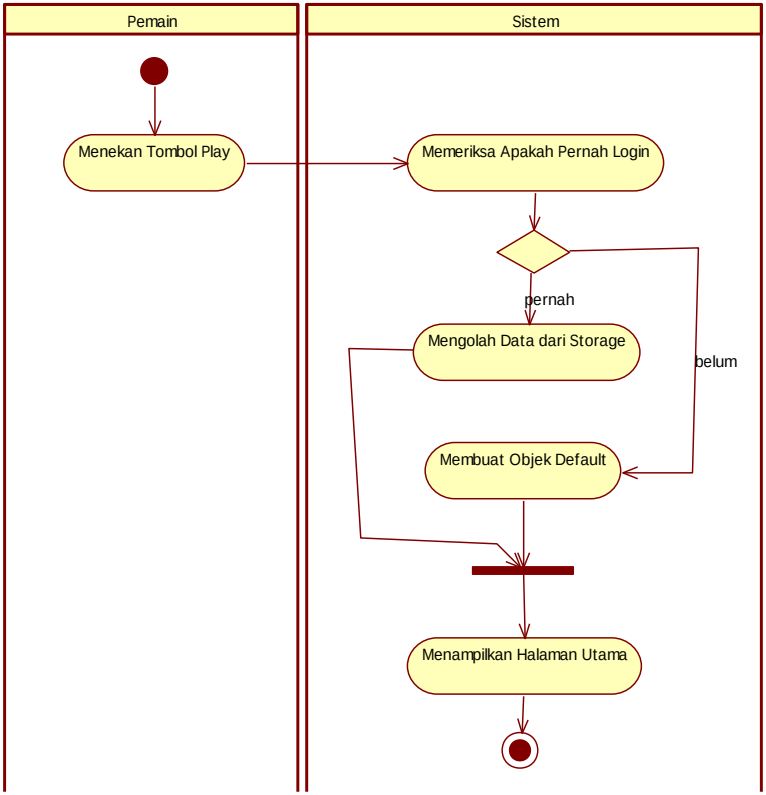
1. *PlayerData*, data informasi pemain seperti nama, nomor identitas, jumlah poin, level, dan sebagainya.
2. *ListOfObject*, data daftar objek-objek yang tersedia pada menu *shop*.
3. *ListOfUsedObject*, data daftar objek-objek yang digunakan pemain dalam permainan Food Merchant Saga.

Jika pemain baru pertama kali *login* maka data pujasera yang ditampilkan adalah data *default* yang dihasilkan oleh sistem. Spesifikasi kasus penggunaan masuk halaman utama dijelaskan pada Tabel 3.4.

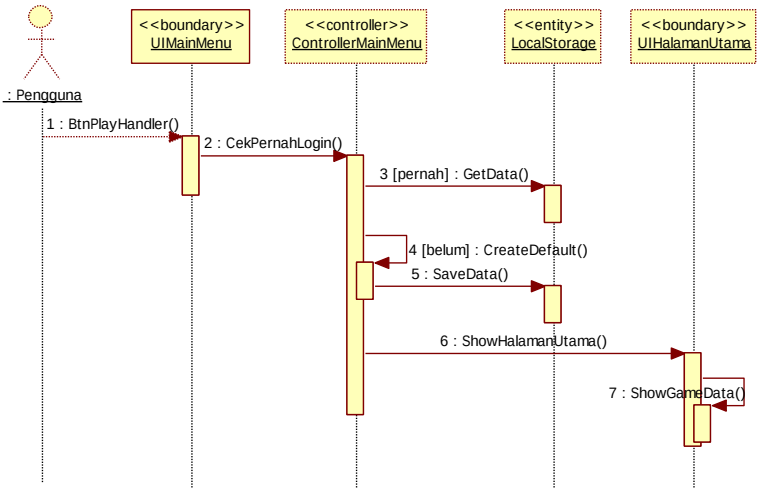
Tabel 3.4 Spesifikasi Kasus Penggunaan UC-02

Nama Kasus Penggunaan	Masuk Halaman Utama
Kode	UC-02
Deskripsi	Pengguna dapat memasuki halaman permainan utama yang berisi pujasera yang dikelolanya
Aktor	Pengguna *game* Food Merchant Saga
Kondisi Awal	Berhasil *login* ke dalam permainan
Pemicu	Pengguna menekan tombol *play* yang terlihat pada layar untuk memulai permainan
Alur Normal	1. Pengguna menekan tombol *play* pada layar. 2. Sistem memeriksa apakah pengguna pernah *login* dengan perangkat tersebut atau baru pertama kali *login*. 3. Jika pernah, maka sistem mengolah data dari penyimpanan internal untuk ditampilkan. 4. Jika belum pernah, maka sistem akan mengolah data *default* untuk ditampilkan. 5. Sistem menampilkan data pujasera ke halaman utama pemain. 6. Selesai.
Alur Alternatif	-
Kondisi Akhir	Pengguna mendapatkan tampilan halaman utama berisi pujasera dan objek-objek di dalamnya.

Berdasarkan pada spesifikasi kasus penggunaan pada Tabel 3.4 yang melibatkan aktor dan sistem, maka dapat digambarkan diagram aktivitas untuk kasus penggunaan masuk halaman utama seperti pada Gambar 3.7 dan diagram alir pada Gambar 3.8 .



Gambar 3.7 Diagram Aktivitas Kasus Penggunaan UC-02



Gambar 3.8 Diagram Alir Kasus Penggunaan UC-02

3.1.6.3. Kasus Penggunaan Mengolah Objek Pujasera (UC-03)

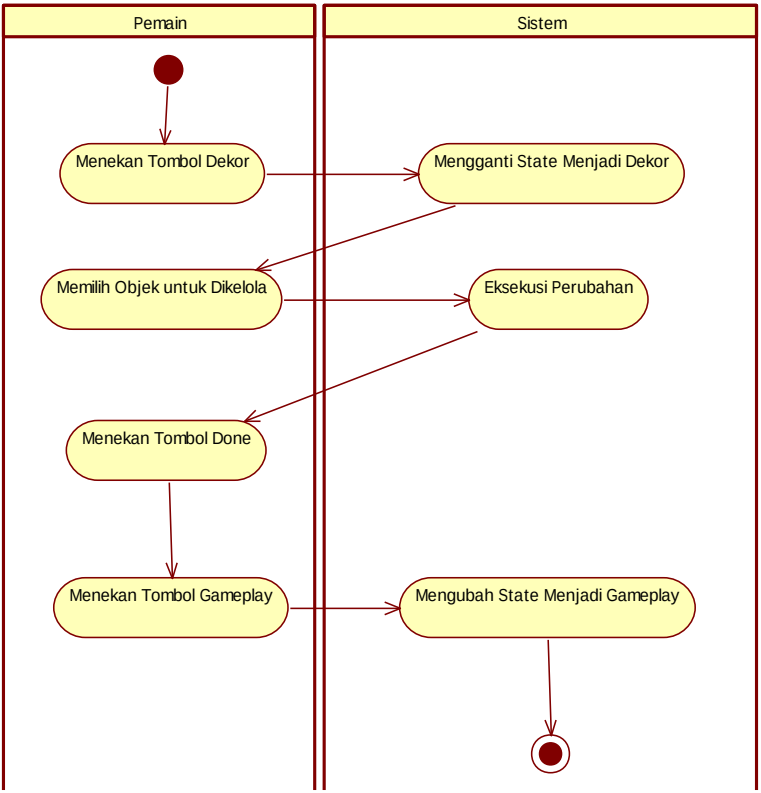
Kasus penggunaan mengolah objek pujasera dapat dilakukan oleh pengguna setelah berada di dalam pujasera. Pengguna dapat mengolah objek-objek pada pujasera dengan menekan tombol mode dekor, kemudian memilih salah satu objek untuk diubah posisi atau arahnya atau menghapus objek dari pujasera. Spesifikasi kasus penggunaan mengolah objek pujasera dijelaskan pada Tabel 3.5.

Tabel 3.5 Spesifikasi Kasus Penggunaan UC-03

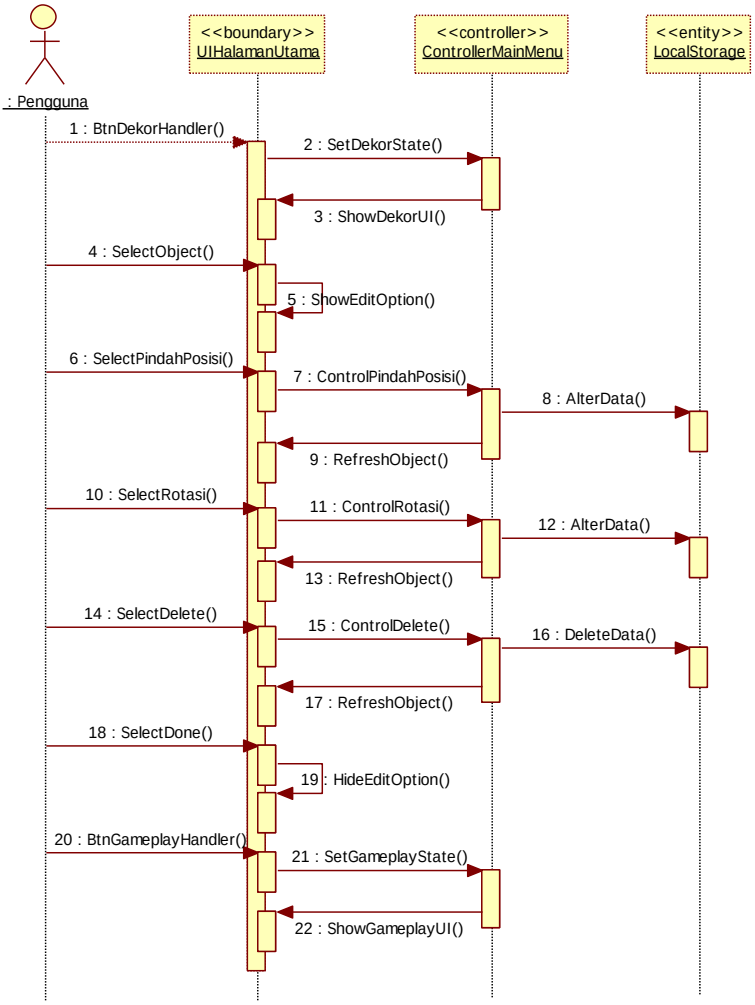
Nama Kasus Penggunaan	Mengolah Objek Pujasera
Kode	UC-03
Deskripsi	Pengguna dapat mengatur dan mengolah objek-objek pada pujasera
Aktor	Pengguna *game* Food Merchant Saga
Kondisi Awal	-

Pemicu	Pengguna menekan tombol mode dekor yang terlihat pada layar untuk memulai mengelola objek
Alur Normal	1. Pengguna menekan tombol mode dekor pada layar. 2. Sistem mengganti *state* permainan menjadi *state* dekor. 3. Pemain memilih sebuah objek untuk dikelola. 4. Pemain menekan tombol *done* untuk mengakhiri mengelola suatu objek. 5. Pemain menekan tombol mode *gameplay* untuk keluar dari mode dekor 6. Selesai.
Alur Alternatif	-
Kondisi Akhir	Pengguna mendapatkan tampilan halaman utama dengan kondisi objek sesuai dengan pengelolaan.

Berdasarkan pada spesifikasi kasus penggunaan pada Tabel 3.5 yang melibatkan aktor dan sistem, maka dapat digambarkan diagram aktivitas untuk kasus penggunaan mengolah objek pujasera seperti pada Gambar 3.9 dan diagram alir pada Gambar 3.10.



Gambar 3.9 Diagram Aktivitas Kasus Penggunaan UC-03



Gambar 3.10 Diagram Alir Kasus Penggunaan UC-03

3.1.6.4. Kasus Penggunaan Mencari Nama Teman (UC-04)

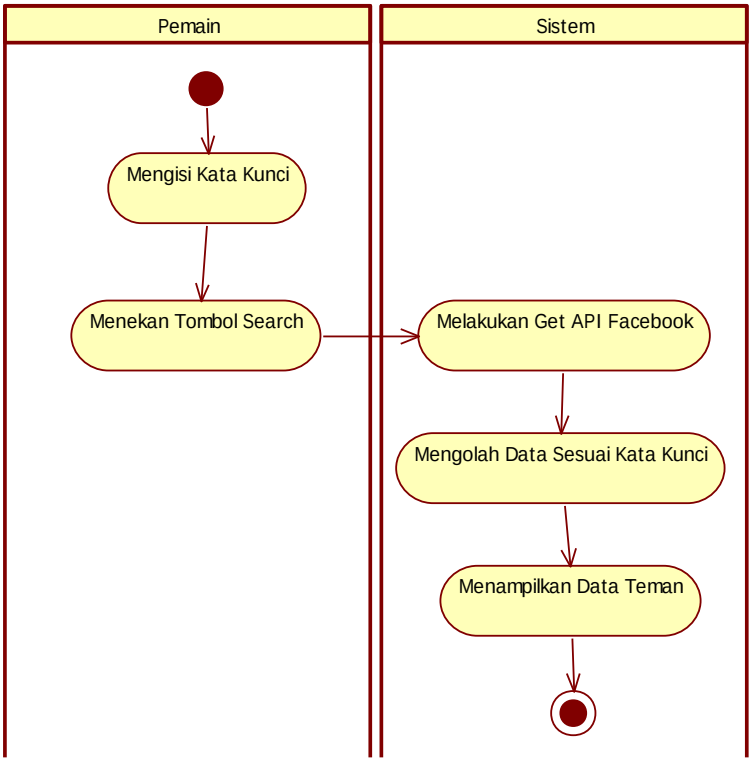
Kasus penggunaan mencari nama teman dilakukan oleh pengguna untuk mendapatkan data teman Facebook pengguna yang juga memainkan *game* Food Merchant Saga. Pemain mengisi kata kunci pencarian pada kotak tulisan sesuai dengan teman Facebook yang ingin dicari. Spesifikasi kasus penggunaan mencari nama teman dijelaskan pada Tabel 3.6.

Tabel 3.6 Spesifikasi Kasus Penggunan UC-04

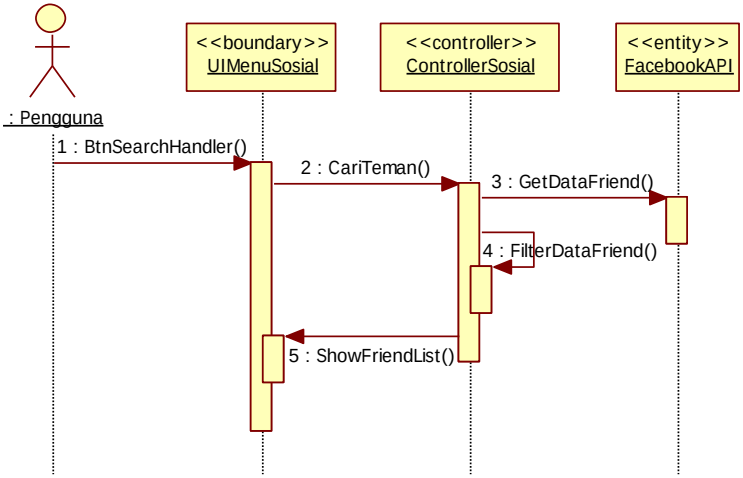
Nama Kasus Penggunaan	Mencari Nama Teman
Kode	UC-04
Deskripsi	Mencari nama temannya yang juga bermain *game* ini
Aktor	Pengguna *game* Food Merchant Saga
Kondisi Awal	-
Pemicu	Pengguna menekan tombol *search* yang terlihat pada layar untuk memulai pencarian teman
Alur Normal	1. Pengguna mengisi kata kunci pencarian pada kotak tulisan yang ada pada layar. 2. Pengguna menekan tombol *search* untuk memulai pencarian teman. 3. Sistem melakukan permintaan data API pada Facebook untuk berupa data teman yang bermain *game* Food Merchant Saga 4. Sistem mengolah data untuk mengambil nama-nama teman yang memenuhi kriteria kata kunci pencarian 5. Sistem menampilkan data teman yang sesuai kriteria kata kunci pencarian ke layar pengguna. 6. Selesai.
Alur Alternatif	-

Kondisi Akhir	Pengguna mendapatkan daftar nama teman Facebook yang juga bermain *game* ini yang sesuai kriteria pencarian.

Berdasarkan pada spesifikasi kasus penggunaan pada Tabel 3.6 yang melibatkan aktor dan sistem, maka dapat digambarkan diagram aktivitas untuk kasus penggunaan mencari nama teman seperti pada Gambar 3.11 dan diagram alir pada Gambar 3.12.



Gambar 3.11 Diagram Aktivitas Kasus Penggunaan UC-04



Gambar 3.12 Diagram Alir Kasus Penggunaan UC-04

3.1.6.5. Kasus Penggunaan Berinteraksi dengan Teman (UC-05)

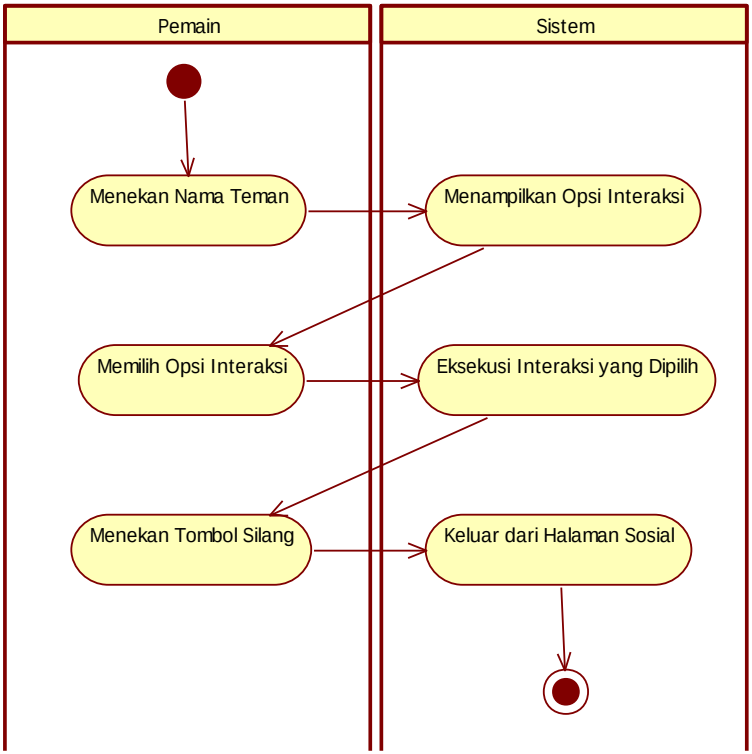
Kasus penggunaan berinteraksi dengan teman dijalankan oleh pengguna setelah melakukan pencarian nama teman dan memilih teman yang ingin dieksekusi. Pemain menekan pada nama teman yang dikehendaki, kemudian akan muncul tiga tombol yang merupakan opsi interaksi dengan teman, yaitu kirim hadiah, mengunjungi, dan kirim pesan. Spesifikasi kasus kegunaan berinteraksi dengan teman dijelaskan pada Tabel 3.7.

Tabel 3.7 Spesifikasi Kasus Penggunan UC-05

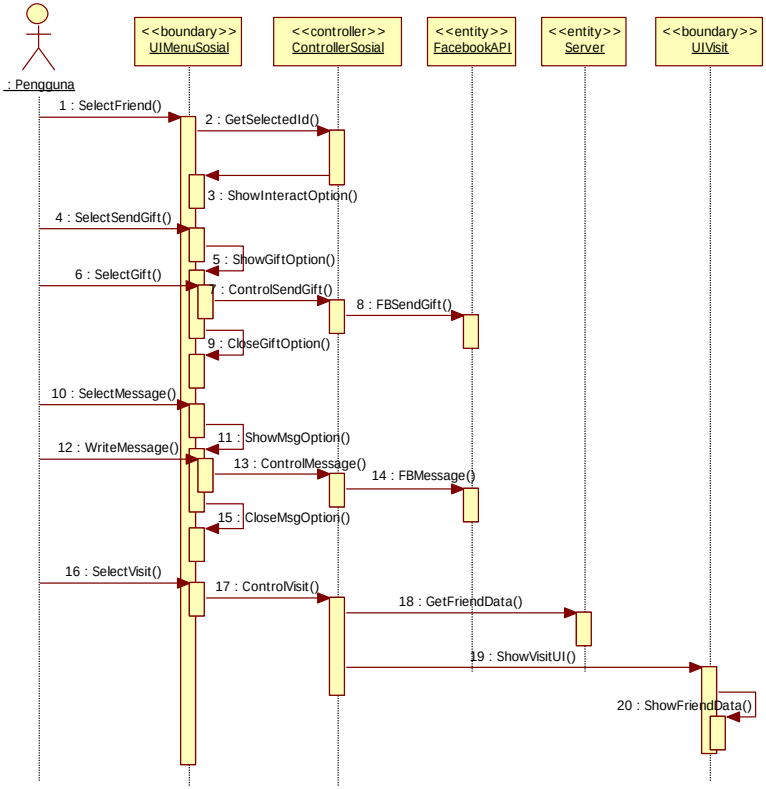
Nama Kasus Penggunaan	Berinteraksi dengan Teman
Kode	UC-05
Deskripsi	Pengguna dapat melakukan interaksi dengan teman, seperti mengirim pesan dan mengunjungi

Aktor	Pengguna *game* Food Merchant Saga
Kondisi Awal	-
Pemicu	Pengguna menekan salah satu nama teman dari hasil pencarian untuk dieksekusi
Alur Normal	1. Pengguna menekan salah satu nama teman dari hasil pencarian untuk memulai interaksi. 2. Sistem menampilkan tiga pilihan opsi interaksi, yaitu mengirim hadiah, mengunjungi, dan mengirim pesan. 3. Pengguna memilih salah satu opsi interaksi 4. Sistem memproses opsi interaksi pada teman yang dipilih. 5. Pengguna menekan tombol silang untuk keluar dari menu sosial. 6. Selesai.
Alur Alternatif	-
Kondisi Akhir	Sistem mengirimkan proses interaksi yang dipilih pengguna terhadap teman yang dipilih ke akun Facebook teman yang dipilih.

Berdasarkan pada spesifikasi kasus penggunaan pada Tabel 3.7 yang melibatkan aktor dan sistem, maka dapat digambarkan diagram aktivitas untuk kasus penggunaan berinteraksi dengan teman seperti pada Gambar 3.13 dan diagram alir pada Gambar 3.14.



Gambar 3.13 Diagram Aktivitas Kasus Penggunaan UC-05



Gambar 3.14 Diagram Alir Kasus Penggunaan UC-05

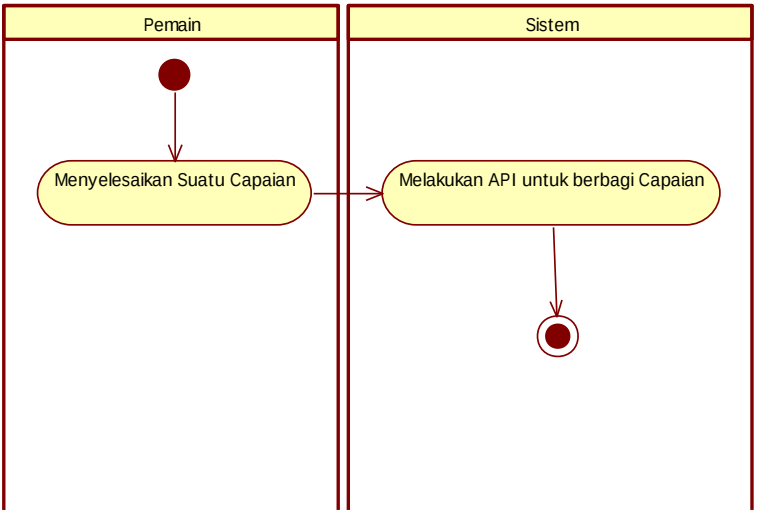
3.1.6.6. Kasus Penggunaan Berbagi Capaian (UC-06)

Kasus penggunaan berbagi capaian akan dilakukan setelah pemain berhasil melakukan suatu capaian tertentu pada permainan Food Merchant Saga. Sistem akan mengirim data pemberitahuan capaian pengguna ke akun Facebook pengguna untuk diterbitkan ke Facebook sehingga dapat disaksikan oleh teman-teman Facebook pengguna. Spesifikasi kasus penggunaan berbagi capaian dijelaskan pada Tabel 3.8.

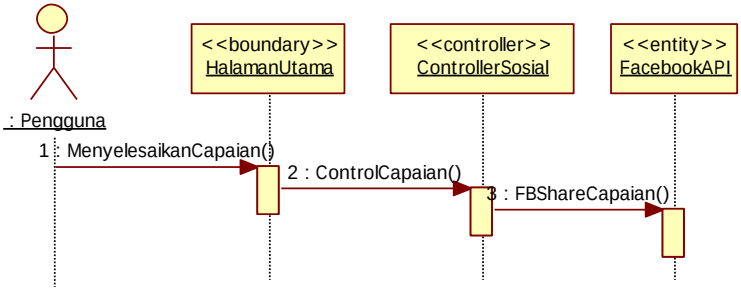
Tabel 3.8 Spesifikasi Kasus Penggunaan UC-06

Nama Kasus Penggunaan	Berbagi Capaian
Kode	UC-06
Deskripsi	Pengguna dapat berbagi capaian-capaian permainan melalui Facebook
Aktor	Pengguna *game* Food Merchant Saga
Kondisi Awal	-
Pemicu	Pengguna menyelesaikan suatu capaian pada *game* Food Merchant Saga
Alur Normal	1. Pengguna menyelesaikan suatu capaian pada *game* Food Merchant Saga. 2. Sistem mengirim data informasi capaian ke Facebook untuk diterbitkan ke halaman Facebook pengguna 3. Selesai.
Alur Alternatif	-
Kondisi Akhir	Pemberitahuan bahwa sistem telah membagikan capaian permainan pengguna ke halaman Facebook.

Berdasarkan pada spesifikasi kasus penggunaan pada Tabel 3.8 yang melibatkan aktor dan sistem, maka dapat digambarkan diagram aktivitas untuk kasus penggunaan berinteraksi dengan teman seperti pada Gambar 3.15 dan diagram alir pada Gambar 3.16.



Gambar 3.15 Diagram Aktivitas Kasus Penggunaan UC-06



Gambar 3.16 Diagram Alir Kasus Penggunaan UC-06

3.1.6.7. Kasus Penggunaan Mengundang Teman (UC-07)

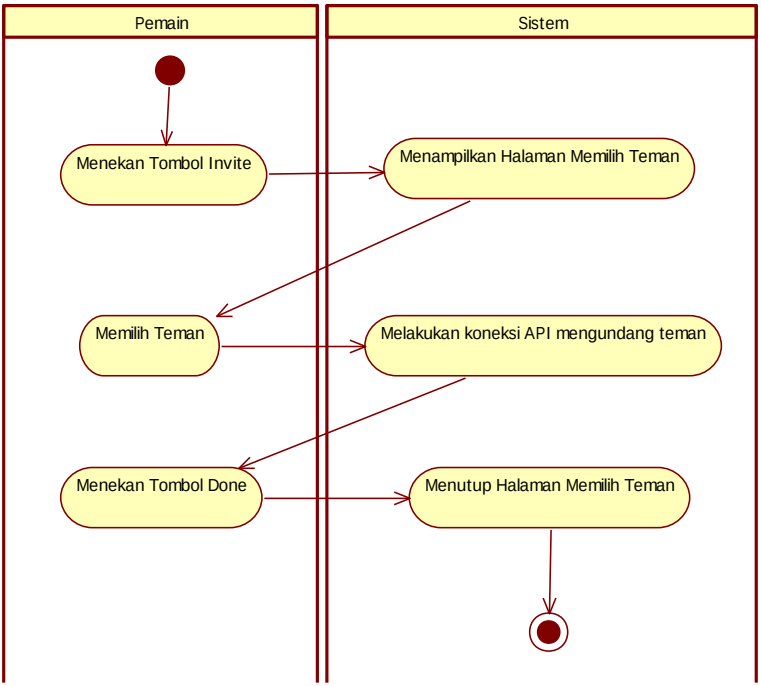
Kasus penggunaan mengundang teman terdapat di halaman menu sosial. Pengguna dapat mengundang teman Facebook yang tidak

bermain *game* Food Merchant Saga untuk bermain *game* Food Merchant Saga dengan menekan tombol *invite*. Sistem akan menampilkan halaman dialog untuk memilih teman Facebook mana saja yang akan dikirimi pesan undangan oleh pengguna. Spesifikasi kasus penggunaan mengundang teman dijelaskan pada Tabel 3.9.

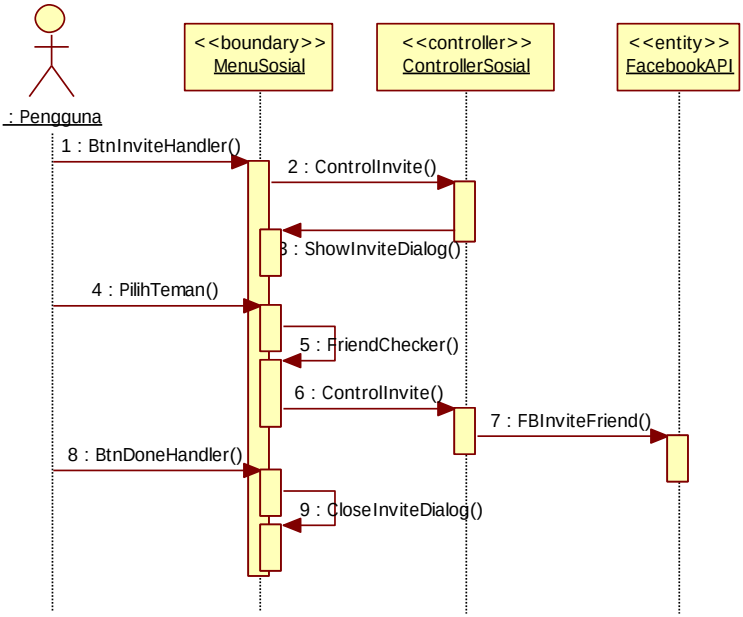
Tabel 3.9 Spesifikasi Kasus Penggunaan UC-07

Nama Kasus Penggunaan	**Mengundang Teman**
Kode	UC-07
Deskripsi	Pengguna dapat mengundang teman Facebook yang belum pernah bermain *game* ini
Aktor	Pengguna *game* Food Merchant Saga
Kondisi Awal	-
Pemicu	Pengguna menekan tombol *invite* untuk memulai mengundang teman bermain Food Merchant Saga
Alur Normal	1. Pengguna menekan tombol *invite* untuk memulai mengundang teman bermain Food Merchant Saga. 2. Sistem menampilkan halaman dialog berisi daftar teman Facebook untuk dipilih oleh pengguna 3. Pengguna memilih teman Facebook untuk dikirimi pesan undangan 4. Sistem melakukan koneksi API ke Facebook untuk mengirim data teman yang diundang oleh pengguna. 5. Pengguna menekan tombol *done* pada halaman dialog. 6. Selesai.
Alur Alternatif	-
Kondisi Akhir	Pengguna keluar dari halaman dialog mengundang teman.

Berdasarkan pada spesifikasi kasus penggunaan pada Tabel 3.9 yang melibatkan aktor dan sistem, maka dapat digambarkan diagram aktivitas untuk kasus penggunaan mengundang teman seperti pada Gambar 3.17 dan diagram alir pada Gambar 3.18.



Gambar 3.17 Diagram Aktivitas Kasus Penggunaan UC-07



Gambar 3.18 Diagram Alir Kasus Penggunaan UC-07

3.2. Perancangan Sistem

Penjelasan tahap perancangan perangkat lunak dibagi menjasi beberapa bagian yaitu perancangan aturan main, perancangan komponen objek, perancangan diagram kelas, dan perancangan antarmuka

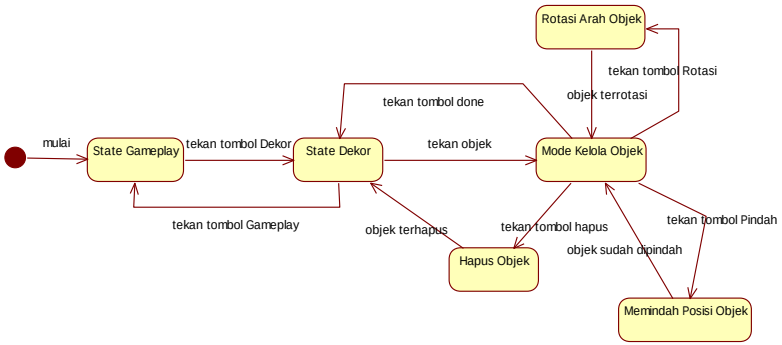
3.2.1. Perancangan Alur Main

Sub bab ini membahas bagaimana perancangan alur main bagi pengguna dalam melakukan fungsionalistas dari modul yang dikerjakan. Alur main dibagi menjad dua, alur main editor ruangan, dan alur main fitur sosial.

3.2.1.1. Alur Main Editor Ruangan

Pada editor ruangan, terdapat dua *state* yang digunakan, yaitu *state* Dekor dan *state Gameplay*. Untuk dapat mengelola objek-objek pada ruangan pujasera, pemain harus berada pada *state* Dekor. Sedangkan *state Gameplay* adalah untuk alur permainan utama. Untuk masuk pada *state* Dekor, pengguna menekan tombol Dekor yang ada pada layar. Pada *state* Dekor, pengguna bisa memilih objek pada pujasera yang ingin dikelola dengan cara menekan pada objek tersebut. Setelah menekan suatu objek, akan muncul kotak dialog berisi tombol-tombol opsi pengelolaan. Tombol-tombol tersebut yaitu tombol Pindah Posisi, tombol Rotasi, tombol Hapus, dan tombol Selesai. Setiap tombol memiliki fungsi sesuai namanya. Tombol Pindah Posisi untuk memindah posisi objek dari satu lantai ke lantai lain. Tombol Rotasi untuk memutar arah objek pada 4 arah ruangan. Tombol Hapus untuk menghapus objek dari ruangan. Sedangkan tombol Selesai untuk mengakhiri pengelolaan pada objek tersebut.

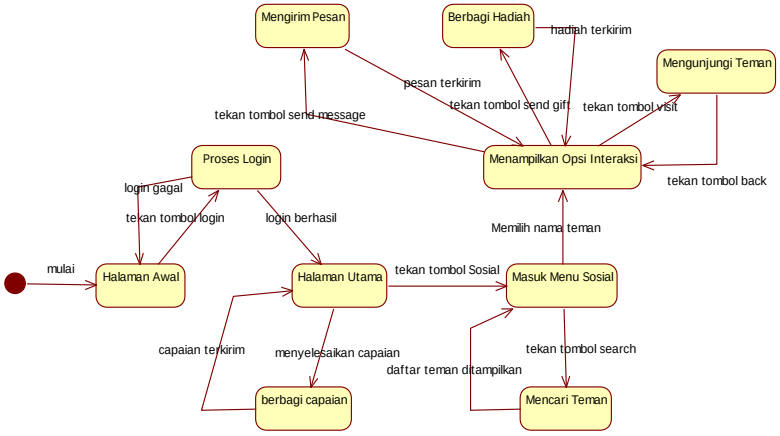
Selain itu pada *state* Dekor, terdapat tombol *Upgrade*. Tombol *Upgrade* digunakan untuk menambah luas ruangan pujasera yang dimiliki pemain. Dengan menekan tombol *Upgrade*, luas ruangan pujasera pemain akan bertambah 2 lantai ke samping dan ke depan. Alur main editor ruangan sebagaimana pada Gambar 3.19.



Gambar 3.19 Diagram *State* Editor Ruangan

3.2.1.2. Alur Main Fitur Sosial

Pada fitur sosial, terdapat tiga area penggunaan fungsionalitas modul fitur sosial. Pertama pada saat pemeriksaan *login* menggunakan Facebook di awal memulai permainan. Kedua pada halaman utama yaitu pada berlangsungnya permainan utama, dimana ketika pemain menyelesaikan suatu capaian, maka sistem akan melakukan *share achievement* melalui akun Facebook pemain. Ketiga adalah pada halaman Menu Sosial, dimana pemain dapat melakukan interaksi dengan teman Facebook yang juga bermain *game* Food Merchant Saga, dan mengundang teman Facebook lainnya untuk bermain *game* ini. Alur main fitur sosial sebagaimana diilustrasikan pada Gambar 3.20.



Gambar 3.20 Diagram *State* Fitur Sosial

Fitur sosial yang dikembangkan pada modul ini antara lain.

1. Fitur Mengirim Pesan. Pemain dapat mengirim pesan dari halaman permainan kepada teman yang dikehendaki dimana pesan tersebut dikirim melalui media sosial Facebook.
2. Fitur Mengirim Hadiah. Pemain dapat memberikan hadiah kepada teman yang dikehendaki dengan memilih objek hadiah yang dikehendaki melalui halaman permainan, kemudian pada akun

Facebook pemain akan ditampilkan sebuah *custom stories* dengan tautan kepada teman yang dipilih bahwa pemain mengirimkan hadiah.

3. Fitur Mengunjungi Teman. Pemain dapat mengunjungi pujasera dari teman Facebook yang bermain permainan ini. Saat pemain mengunjungi salah satu teman, maka pada halaman permainan akan ditampilkan tata letak objek-objek pujasera teman yang dipilih yang mana diambil dari *database server*.
4. Fitur Berbagi Capaian. Pemain dapat berbagi capaian-capaian permainan yang telah dilakukan oleh pemain ke media sosial Facebook.

3.2.2. Perancangan Komponen Objek

Sub bab ini menjelaskan mengenai perancangan komponen objek yang diterapkan pada ruangan pujasera di halaman utama permainan. Komponen objek ini dibagi menjadi empat, yaitu komponen objek Kedai, komponen objek Hiasan, komponen objek Properti, dan komponen objek Hadiah.

3.2.2.1. Komponen Objek Kedai

Komponen objek Kedai merupakan objek yang penting dalam permainan Food Merchant Saga. Setiap objek Kedai mewakili suatu jenis masakan nusantara dimana NPC pengunjung pujasera membeli makanan di setiap kedai tersebut. Spesifikasi komponen objek kedai sebagaimana pada Tabel 3.10.

Tabel 3.10 Spesifikasi Komponen Objek Kedai

Nama Objek	Gambar Objek	Keterangan
Kedai Bakso		Harga Beli: 100 koin Harga Jual: 90 koin Kebutuhan Level: 1 Popularitas: 10

Kedai Soto		Harga Beli: 200 koin Harga Jual: 100 koin Kebutuhan Level: 1 Popularitas: 15
Kedai Sate		Harga Beli: 250 koin Harga Jual: 200 koin Kebutuhan Level: 4 Popularitas: 20
Kedai Gulai		Harga Beli: 400 koin Harga Jual: 300 koin Kebutuhan Level: 4 Popularitas: 30
Kedai Mie		Harga Beli: 400 koin Harga Jual: 300 koin Kebutuhan Level: 7 Popularitas: 40
Kedai Ikan		Harga Beli: 400 koin Harga Jual: 300 koin Kebutuhan Level: 9 Popularitas: 50

Kedai Rujak		Harga Beli: 400 koin Harga Jual: 300 koin Kebutuhan Level: 10 Popularitas: 50
Kedai Sop		Harga Beli: 400 koin Harga Jual: 300 koin Kebutuhan Level: 11 Popularitas:70

3.2.2.2. Komponen Objek Hiasan

Komponen objek kedai adalah objek yang berisi hiasan-hiasan ruangan untuk diterapkan pada ruangan pujasera. Spesifikasi komponen objek hiasan sebagaimana Tabel 3.11.

Tabel 3.11 Spesifikasi Komponen Objek Hiasan

Nama Objek		Keterangan
Pot Bunga Kecil 1		Harga Beli: 15 koin Harga Jual: 10 koin Kebutuhan Level: 1 Popularitas: 10
Pot Bunga Kecil 2		Harga Beli: 20 koin Harga Jual: 10 koin Kebutuhan Level: 1 Popularitas: 15

Pot Bunga Besar 1		Harga Beli: 40 koin Harga Jual: 30 koin Kebutuhan Level: 2 Popularitas: 20
Pot Bunga Besar 2		Harga Beli: 45 koin Harga Jual: 30 koin Kebutuhan Level: 3 Popularitas: 25
Pot Bunga Besar 2		Harga Beli: 50 koin Harga Jual: 30 koin Kebutuhan Level: 4 Popularitas: 30
Hiasan Dinding 1		Harga Beli: 15 koin Harga Jual: 10 koin Kebutuhan Level: 1 Popularitas: 10
Hiasan Dinding 2		Harga Beli: 20 koin Harga Jual: 10 koin Kebutuhan Level: 3 Popularitas: 15

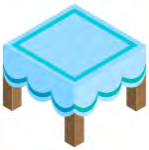
Hiasan Dinding 3		Harga Beli: 25 koin Harga Jual: 15 koin Kebutuhan Level: 5 Popularitas: 20
Balon Hias		Harga Beli: 30 koin Harga Jual: 20 koin Kebutuhan Level: 2 Popularitas: 30

3.2.2.3. Komponen Objek Properti

Komponen objek properti adalah objek-objek perabotan yang terkait erat dengan pujasera, seperti meja dan kursi. Spesifikasi komponen objek properti sebagaimana Tabel 3.12.

Tabel 3.12 Spesifikasi Komponen Objek Properti

Nama Objek		Keterangan
Kursi Plastik		Harga Beli: 30 koin Harga Jual: 10 koin Kebutuhan Level: 1 Popularitas: 5
Kursi Kayu		Harga Beli: 60 koin Harga Jual: 40 koin Kebutuhan Level: 4 Popularitas: 10

Kursi Eksklusif		Harga Beli: 150 koin Harga Jual: 100 koin Kebutuhan Level: 10 Popularitas: 20
Meja Plastik		Harga Beli: 70 koin Harga Jual: 30 koin Kebutuhan Level: 1 Popularitas: 5
Meja Kayu		Harga Beli: 100 koin Harga Jual: 60 koin Kebutuhan Level: 4 Popularitas: 10
Meja Eksklusif		Harga Beli: 300 koin Harga Jual: 250 koin Kebutuhan Level: 10 Popularitas: 20
Vending Machine		Harga Beli: 100 koin Harga Jual: 60 koin Kebutuhan Level: 1 Popularitas: 20

3.2.2.4. Komponen Objek Hadiah

Komponen objek hadiah adalah objek-objek yang digunakan untuk berkirim hadiah kepada sesama pemain. Spesifikasi objek hadiah sebagaimana pada Tabel 3.13.

Tabel 3.13 Spesifikasi Komponen Objek Hadiah

Nama Objek		Keterangan
Kartu Pos Mini		Harga Beli: 20 koin Kebutuhan Level: 1 Popularitas: 10
Kartu Pos *Deluxe*		Harga Beli: 30 koin Kebutuhan Level: 2 Popularitas: 15
Album Foto Mini		Harga Beli: 25 koin Kebutuhan Level: 2 Popularitas: 20
Album Foto *Deluxe*		Harga Beli: 45 koin Kebutuhan Level: 3 Popularitas: 25

Action Figure		Harga Beli: 50 koin Kebutuhan Level: 3 Popularitas: 30
Lukisan Kecil		Harga Beli: 65 koin Kebutuhan Level: 4 Popularitas: 40
Lukisan Besar		Harga Beli: 80 koin Kebutuhan Level: 5 Popularitas: 45
Kotak Musik Mini		Harga Beli: 60 koin Kebutuhan Level: 6 Popularitas: 50
Kotak Musik Mewah		Harga Beli: 90 koin Kebutuhan Level: 8 Popularitas: 60

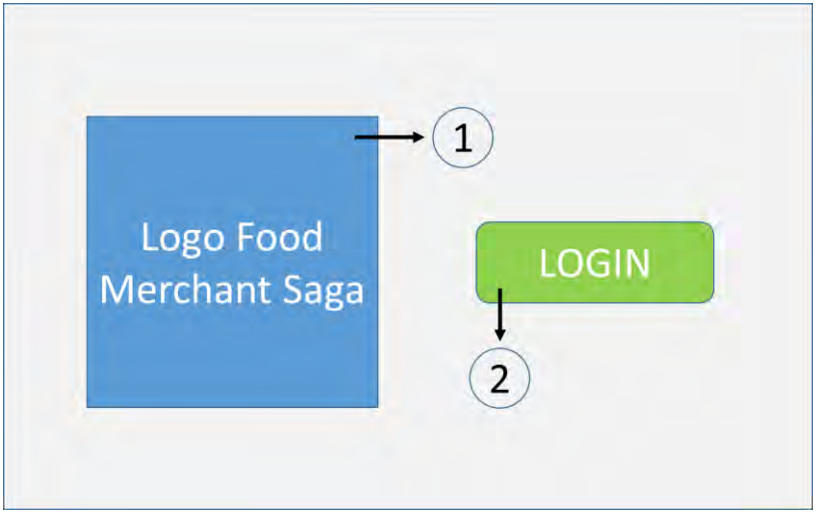
3.2.3. Perancangan Antarmuka

Bagian ini membahas rancangan tampilan antar muka pada sistem. Terdapat beberapa tampilan antarmuka yang berkaitan dengan

modul editor ruangan dan fitur sosial, yaitu tampilan *login*, tampilan permainan utama, tampilan menu sosial, tampilan *send gift*, dan tampilan *send message*.

3.2.3.1. Tampilan Halaman *Login*

Halaman ini merupakan halaman yang muncul di awal setiap kali pemain menjalankan aplikasi permainan Food Merchant Saga. Pada halaman ini terdapat satu tombol untuk *login*. Tombol ini akan berubah menjadi tombol *play* saat pemain berhasil *login*.



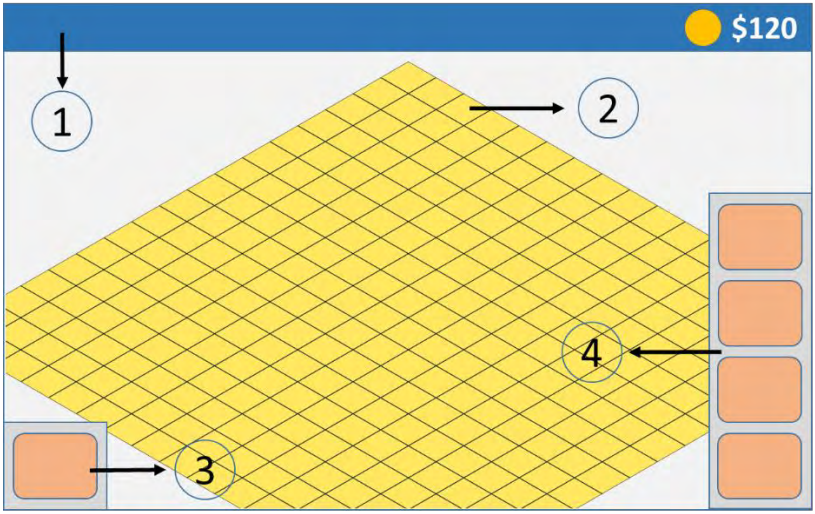
Gambar 3.21 Rancangan Antarmuka Halaman *Login*

Gambar 3.21 merupakan rancangan antarmuka tampilan halaman *login*. Berikut penjelasan masing-masing nomor yang tertera dalam gambar.

1. Logo permainan Food Merchant Saga yang ditampilkan sebagai pengaya halaman *login*.
2. Tombol *login* untuk pemain dapat melakukan proses *login* saat menekan tombol tersebut. Ketika proses *login* berhasil, antarmuka tombol akan berubah menjadi tombol *play*.

3.2.3.2. Tampilan Halaman Permainan Utama

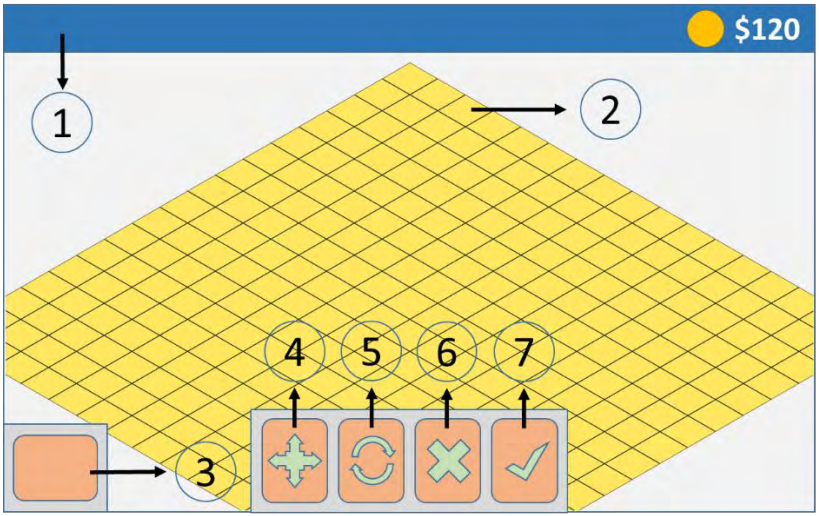
Halaman ini merupakan halaman tempat berlangsungnya permainan utama dari *game* Food Merchant Saga. Tampilan ini memiliki dua *state*, yaitu *state Gameplay* dan *state* Dekor. Pada saat *state Gameplay*, tampilan berisi beberapa tombol diantaranya, tombol Menu *Shop*, tombol *Setting*, tombol Menu Sosial, tombol *Inventory*, tombol Dekor, dan tombol *Close Application*. Sedangkan pada saat *state* Dekor, tampilan berisi papan kelola yang didalamnya terdapat tombol Pinda Posisi, tombol Rotasi, tombol Hapus, dan tombol Selesai. Selain itu terdapat juga tombol *Gameplay* dan tombol *Upgrade*. Perbedaan tampilan saat berada di *state* Dekor dan *Gameplay* sebagaimana Gambar 3.22 dan Gambar 3.23.



Gambar 3.22 Rancangan Antarmuka Halaman Permainan Utama *State Gameplay*

Gambar 3.22 merupakan rancangan antarmuka tampilan halaman permainan utama pada saat *state Gameplay*. Berikut penjelasan masing-masing nomor yang tertera dalam gambar.

1. Panel status HUD (*Head-up Display*), untuk menampilkan status-status permainan yang umum, seperti jumlah uang dan waktu.
2. Penampang ruangan *isometric*, dimana pada penampang tersebut terdapat objek-objek pujasera untuk dimainkan.
3. Tombol Dekor, digunakan untuk mengubah *state* dari *state Gameplay* ke *state* Dekor.
4. Panel menu utama, berisi tombol-tombol untuk menampilkan menu-menu utama, seperti menu *Shop*, *Inventory*, *Sosial*, dan pengaturan.



Gambar 3.23 Rancangan Antarmuka Halaman Permainan Utama *State* Dekor

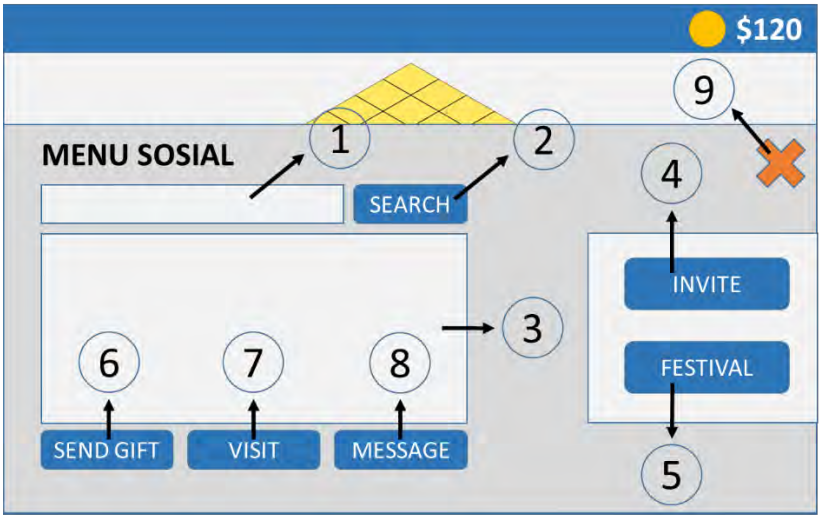
Gambar 3.23 merupakan rancangan antarmuka tampilan halaman permainan utama pada saat *state* Dekor. Berikut penjelasan masing-masing nomor yang tertera dalam gambar.

1. Panel status HUD (*Head-up Display*), untuk menampilkan status-status permainan yang umum, seperti jumlah uang dan waktu.
2. Penampang ruangan *isometric*, dimana pada penampang tersebut terdapat objek-objek pujasera untuk dimainkan.

3. Tombol *Gameplay*, digunakan untuk beralih dari *state* Dekor ke *state Gameplay*.
4. Tombol pindah posisi, digunakan untuk mengubah posisi objek pada penampang ruangan *isometric*.
5. Tombol rotasi, digunakan untuk mengubah arah objek terhadap penampang ruangan *isometric*.
6. Tombol hapus, digunakan untuk menghapus objek dari penampang ruangan *isometric*.
7. Tombol selesai, digunakan untuk mengakhiri pengaturan pada suatu objek.

3.2.3.3. Tampilan Halaman Menu Sosial

Halaman ini merupakan halaman untuk sebagian besar fungsi sosial, yaitu mencari teman, interaksi dengan teman, dan mengundang teman. Pada halaman ini terdapat kotak tulisan dan tombol *search* untuk mencari teman yang diinginkan. Kemudian terdapat tombol *send gift*, tombol *visit*, dan tombol *send message* untuk berinteraksi dengan teman, dan tombol *invite* untuk mengundang teman.



Gambar 3.24 Rancangan Antarmuka Halaman Menu Sosial

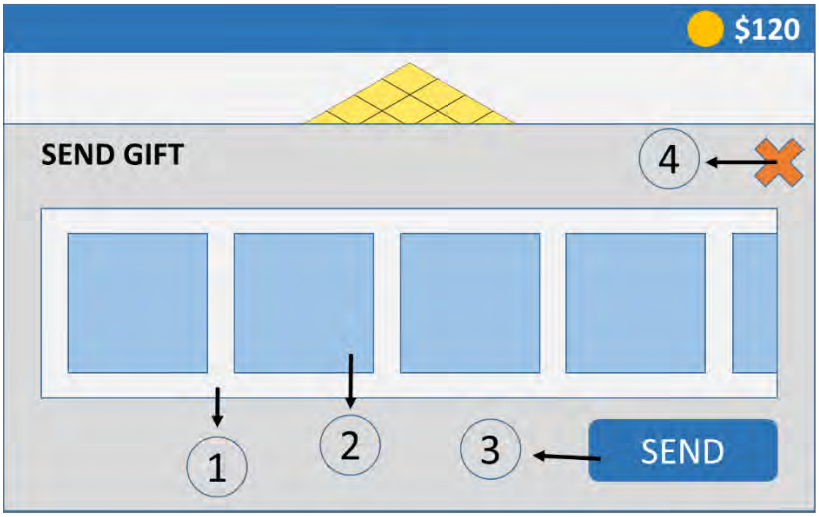
Gambar 3.24 merupakan rancangan antarmuka tampilan halaman menu sosial. Berikut penjelasan masing-masing nomor yang tertera dalam gambar.

1. Kotak tulisan pencarian, untuk memasukkan kata kunci pencarian teman.
2. Tombol *Search*, untuk melakukan pencarian sesuai dengan kata kunci yang terdapat pada kotak tulisan pencarian.
3. Panel hasil pencarian, untuk menampilkan daftar nama-nama teman hasil pencarian.
4. Tombol *Invite*, untuk melakukan proses pengundangan teman Facebook untuk bermain *game* Food Merchant Saga.
5. Tombol Festival, untuk menuju ke mode permainan festival dari *game* Food Merchant Saga.
6. Tombol *Send Gift*, untuk melakukan proses pemberian hadiah pada teman yang dipilih pada panel hasil pencarian teman.
7. Tombol *Visit*, untuk mengunjungi pujasera teman yang dipilih pada panel hasil pencarian teman.

8. Tombol *Message*, untuk mengirimkan pesan pada teman yang dipilih pada panel hasil pencarian teman.
9. Tombol keluar, untuk menutup halaman menu sosial.

3.2.3.4. Tampilan Halaman *Send Gift*

Halaman ini merupakan halaman untuk memilih hadiah yang akan dikirim kepada teman. Halaman ini muncul setelah pemain menekan tombol *send gift*. Pada halaman ini terdapat kotak *scroll view* untuk memilih hadiah yang ingin dikirim, dan tombol *send* untuk mengirim hadiah.



Gambar 3.25 Rancangan Antarmuka Halaman *Send Gift*

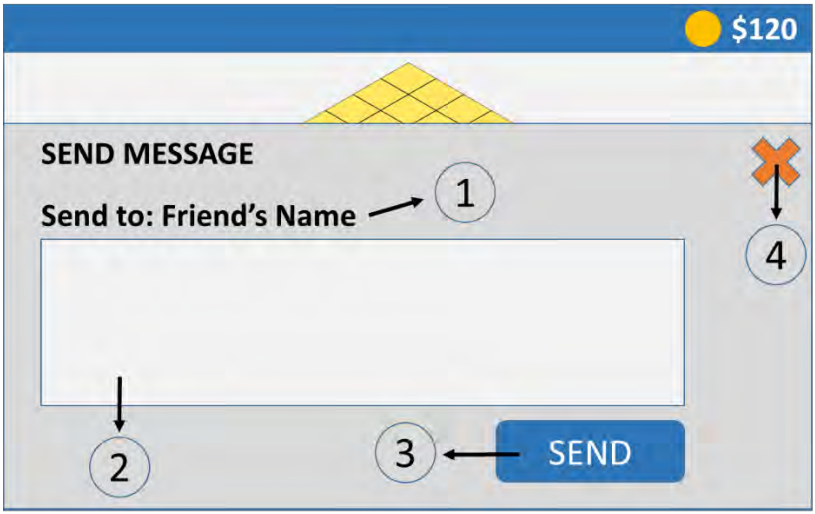
Gambar 3.25 merupakan rancangan tampilan halaman untuk mengirim hadiah. Berikut penjelasan masing-masing nomor yang tertera pada gambar.

1. Panel daftar hadiah yang bisa dipilih oleh pemain.
2. Panel satuan hadiah berisi gambar hadiah, nama hadiah, dan harga beli hadiah.

3. Tombol *Send*, untuk mengirimkan hadiah yang telah dipilih oleh pemain dari panel daftar hadiah.
4. Tombol keluar, untuk menutup halaman *Send Gift*.

3.2.3.5. Tampilan Halaman *Send Message*

Halaman ini merupakan halaman untuk mengirim pesan kepada teman yang telah dipilih. Halaman ini muncul setelah pemain menekan tombol *send message*. Pada halaman ini terdapat kotak tulisan untuk menulis isi pesan yang ingin dikirim dan tombol *send message* untuk mengirim pesan.



Gambar 3.26 Rancangan Antarmuka Halaman *Send Message*

Gambar 3.26 merupakan rancangan tampilan halaman untuk mengirimkan pesan kepada teman. Berikut penjelasan masing-masing nomor yang tertera dalam gambar.

1. Nama teman yang akan dikirim pesan. Nama ini berdasarkan pada teman yang dipilih dari daftar pencarian teman.
2. Kotak tulisan pesan, untuk menuliskan pesan yang akan dikirimkan kepada teman yang dipilih.

3. Tombol *Send*, digunakan untuk mengirim pesan dari kotak tulisan kepada teman yang dipilih.
4. Tombol keluar, untuk menutup halaman *Send Message*.

BAB IV
IMPLEMENTASI

Bab ini membahas tentang implementasi dari perancangan sistem. Bab ini berisi proses implementasi dari setiap kelas pada semua modul. Bahasa pemrograman yang digunakan adalah bahasa pemrograman C#.

4.1. Implementasi Kelas Komponen

Lapisan komponen merupakan lapisan untuk menampung variabel-variabel dan fungsi yang dimiliki sebuah *game object* pada Unity. Kelas-kelas ini disisipkan sebagai komponen pada *game object* sesuai dengan kebutuhan.

4.1.1. Kelas BaseScript

Kelas ini merupakan kelas utama untuk penerapan berbagai fungsi *gameplay*. Kelas ini berjalan saat menjalankan halaman utama permainan. Kelas ini disisipkan pada *game object* Scripts pada *scene* *Game*PlayTest. Fungsi kelas ini diantaranya untuk melakukan *instantiate* objek *tile* dan *wall* dari pujasera dan objek objek aktif pengguna. Impelementasi fungsi *instantiate tile* dan objek aktif pengguna sebagaimana pada Kode Sumber 4.1 dan Kode Sumber 4.2.

```
1   if(i == 0)
2   {
3       var wallInstant = Instantiate(wallPrefab2) as
4   Transform;
5       wallInstant.position = new
6   Vector3((float)arrayX[i,j], (float)arrayY[i,j],
7   baseTile.transform.position.z);
8       wallInstant.parent = baseTile.transform;
9       TileScript tilePos2 =
10  wallInstant.GetComponent<TileScript>();
11      tilePos2.posX = i;
12      tilePos2.posY = j;
13
14  wallInstant.GetChild(0).GetComponent<SpriteRender
15
```

```
16  er>().sortingOrder = ((int)tilePos2.posY * 10) +
17  (int)tilePos2.posX;
18  }
```

Kode Sumber 4.1 Potongan Fungsi *Instantiate Tile*

```
1   posX = k.GridPosition.XPos;
2   posY = k.GridPosition.YPos;
3   clone.transform.parent = baseObject.transform;
4   clone.GetComponentInChildren<SpriteRenderer>().so
5   rtingOrder = (posY * 10) + posX;
6   FMSView kv = clone.GetComponent<FMSView>();
7   kv.index = UsedFMS.IndexOf(k);
```

Kode Sumber 4.2 Potongan Fungsi *Draw Object*

4.1.2. Kelas TileScript

Kelas ini untuk menyimpan variabel posisi *tile* yang bernama GridPosition. Kelas ini disisipkan pada *game object* Tile dan ObjWall yang merepresentasikan *tile* dan *wall* dari ruangan pujasera. Variabel GridPosition digunakan untuk menyimpan posisi x dan y dari objek Tile dan ObjWall terhadap bidang *isometric*.

4.1.3. Kelas ObjScript

Kelas ini digunakan untuk menangani *event* ketika objek yang tersisipi kelas ini mendapatkan masukan sentuhan. Kelas ini disisipkan pada objek-objek yang dimainkan pada pujasera. Ketika object disentuh maka kelas ObjScript menjalankan event sesuai dengan *state* yang sedang berjalan. Jika *state* yang aktif adalah Dekor, maka fungsi *handler* akan menyimpan status objek sebagai objek yang sedang dipilih pada variabel objOnPicked.

4.1.4. Kelas PickScript

Kelas ini disisipkan pada *game object* Tile. Kelas PickScript ini untuk menangani event ketika *game object* Tile mendapatkan masukan sentuhan. Saat *game object* Tile tersentuh, kelas PickScript akan menjalankan fungsi untuk mendapatkan posisi x dan y dari *game object* Tile yang tersentuh. Berfungsi ketika pengguna melakukan pemindahan posisi objek pujasera.

4.1.5. Kelas SpriteScript

Kelas ini menyimpan variabel bertipe Sprite yang disisipkan pada *game object* yang bisa dirotasi, seperti kursi dan kedai.

4.1.6. Kelas CameraScript

Kelas ini disisipkan pada *game object* MainCamera. Kelas ini berfungsi untuk *event handler* ketika player menyentuh layar untuk menggeser kamera. Ketika sentuhan diterima, fungsi akan memperhitungkan perbedaan waktu posisi sentuhan dan memindah posisi kamera sesuai dengan kembalian dari perhitungan tersebut.

4.1.7. Kelas MainMenu

Kelas ini disisipkan pada *game object* Scripts pada *scene* MainMenu. Kelas ini berjalan saat pemain membuka halaman *Login* dari *game* Food Merchant Saga. Kelas ini berperan untuk menangani inisialisasi komponen Facebook SDK dan melakukan proses *login*.

4.1.8. Kelas VisitView

Kelas ini disisipkan pada *game object* Scripts pada *scene Game*PlayVisit. Kelas ini berjalan saat pemain berada pada halaman mengunjungi teman. Kelas ini berfungsi untuk menangani *instantiate* objek *Tile* dan *wall* pada penampang ruangan *isometric* dari halaman mengunjungi teman. Kelas ini juga berfungsi untuk melakukan *instantiate* objek-objek pujasera dari teman yang dikunjungi.

4.1.9. Kelas SocialView

Kelas ini disisipkan pada *game object* MenuOnline pada *scene Game*PlayTest. Kelas ini untuk menangani proses-proses pada menu sosial, seperti pencarian teman, mengundang teman, dan berinteraksi dengan teman yang dipilih.

4.1.10. Kelas GiftView

Kelas ini disisipkan pada *game object* MenuGift pada *scene Game*PlayTest. Kelas ini untuk menangani proses pengiriman hadiah pada halaman pengiriman hadiah setelah pemain menekan tombol *Send Gift* pada menu sosial.

4.1.11. Kelas MessageView

Kelas ini disisipkan pada *game object* MenuMessage pada *scene Game*PlayTest. Kelas ini untuk menangani proses mengirim pesan pada halaman pengiriman pesan setelah pemain menekan tombol *Send Message* pada menu sosial.

4.2. Implementasi Antarmuka

Pada bagian ini dijelaskan mengenai implementasi antarmuka dari modul yang dikerjakan. Implementasi antarmuka ini berdasarkan pada perancangan antarmuka pada perancangan sistem.

4.2.1. Implementasi Antarmuka Halaman *Login*

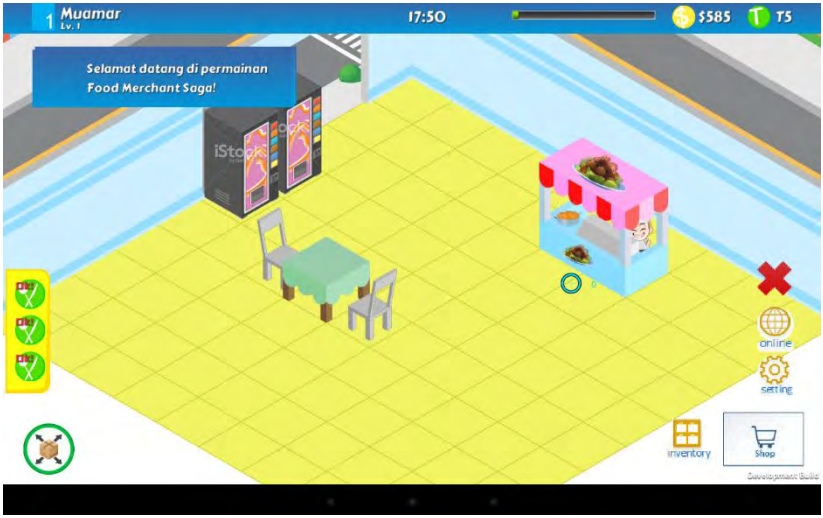
Halaman *login* adalah halaman pertama yang ditampilkan kepada pengguna saat memulai *game* Food Merchant Saga. Implementasi antarmuka halaman *login* sebagaimana ditunjukkan pada Gambar 4.1.

Gambar 4.1 Tampilan Halaman *Login*

Pada Gambar 4.1 yang merupakan tampilan antarmuka halaman *login*, terdapat tombol untuk melakukan proses *login*. Saat pertama kali memasang aplikasi pada perangkat, pemain perlu melakukan proses *login* terlebih dahulu. Proses *login* dilakukan dengan integrasi terhadap Facebook. Setelah proses *login* selesai, tombol akan berubah menjadi tombol *play* untuk memulai permainan. Selanjutnya ketika pemain membuka aplikasi di lain kesempatan, langsung muncul tombol *play*. Pemain tidak perlu *login* kembali karena data identitas pemain telah tersimpan pada penyimpanan internal aplikasi.

4.2.2. Implementasi Antarmuka Halaman Permainan Utama

Halaman permainan utama adalah tempat dimana sebagian besar proses permainan dilakukan. Pada halaman ini terdapat dua macam *state* permainan yang bisa diubah setiap saat. *State Gameplay* adalah *state* dimana pemain melakukan proses bisnis pujasera yang dimiliki. Sedangkan *state* Dekor adalah *state* dimana pemain dapat mengelola objek-objek pada ruangan pujaseranya. Implementasi antar muka halaman permainan utama pada saat *state Gameplay* sebagaimana pada Gambar 4.2 , dan pada saat *state* Dekor sebagaimana pada Gambar 4.3.



Gambar 4.2 Tampilan Halaman Permainan Utama *State Gameplay*

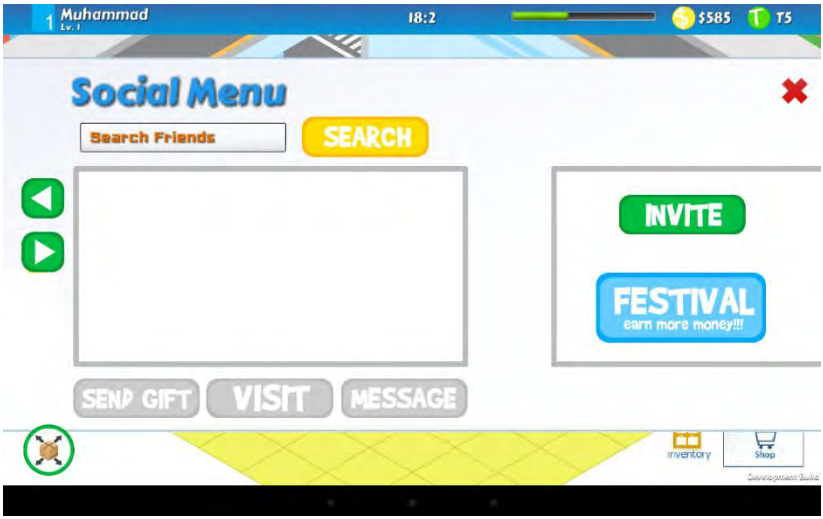
Gambar 4.3 Tampilan Halaman Permainan Utama *State* Dekor

Pada Gambar 4.2 saat *state* permainan berada pada *state Gameplay*, pada sisi kanan terdapat tombol-tombol menu antara lain menu *shop*, menu *inventory*, menu sosial, menu pengaturan, dan tombol keluar aplikasi. Sedangkan pada sisi kiri terdapat tombol untuk mengubah *state* menjadi *state* Dekor.

Pada Gambar 4.3 saat *state* permainan berada pada *state* Dekor, terdapat tombol *upgrade* untuk melakukan penambahan luas pujasera. Pada *state* ini, ketika pemain menekan salah satu objek yang terdapat pada pujasera, akan muncul kotak berisi tombol-tombol untuk melakukan pengelolaan objek, diantaranya tombol pindah posisi, tombol rotasi, tombol hapus, dan tombol selesai. Untuk kembali ke *state Gameplay*, dengan menekan tombol pada sisi kiri bawah, yaitu tombol untuk mengubah *state*.

4.2.3. Implementasi Antarmuka Halaman Menu Sosial

Halamn menu sosial akan muncul saat pemain menekan tombol menu sosial yang terdapat pada halaman permainan utama. Implementasi antarmuka halaman menu sosial sebagaimana pada Gambar 4.4.

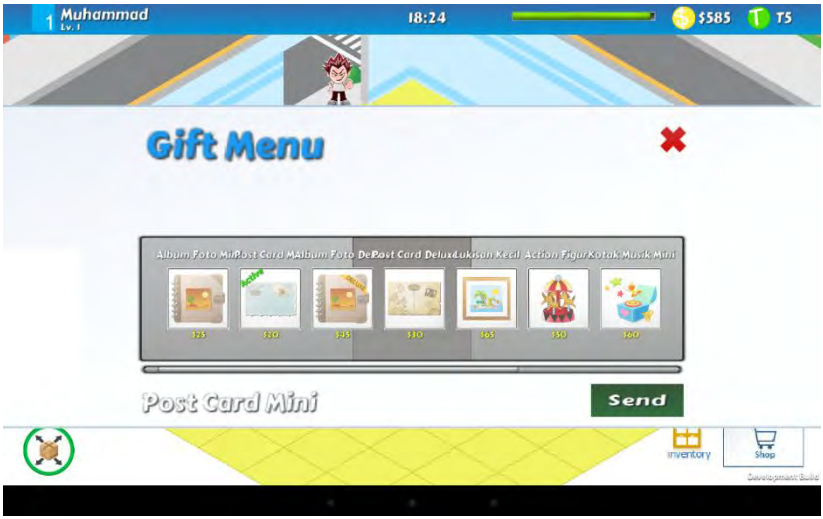


Gambar 4.4 Tampilan Halaman Menu Sosial

Pada Gambar 4.4 terdapat kotak tulisan untuk mencari nama teman beserta dengan tombol pencariannya. Kemudian terdapat kotak untuk menampilkan hasil pencarian teman. Kemudian terdapat tiga tombol opsi interaksi dengan teman yang menjadi aktif ketika pemain menekan salah satu nama teman hasil pencarian, antara lain tombol *send gift*, tombol *visit*, dan tombol *send message*. Selain itu di sisi kanan terdapat tombol *invite* untuk melakukan pengundangan kepada teman Facebook, dan tombol untuk menuju menu festival.

4.2.4. Implementasi Antarmuka Halaman *Send Gift*

Halaman *send gift* muncul ketika pemain menekan tombol *send gift* setelah pemain memilih nama teman dari daftar hasil pencarian. Implementasi antarmuka halaman *send gift* sebagaimana pada Gambar 4.5.

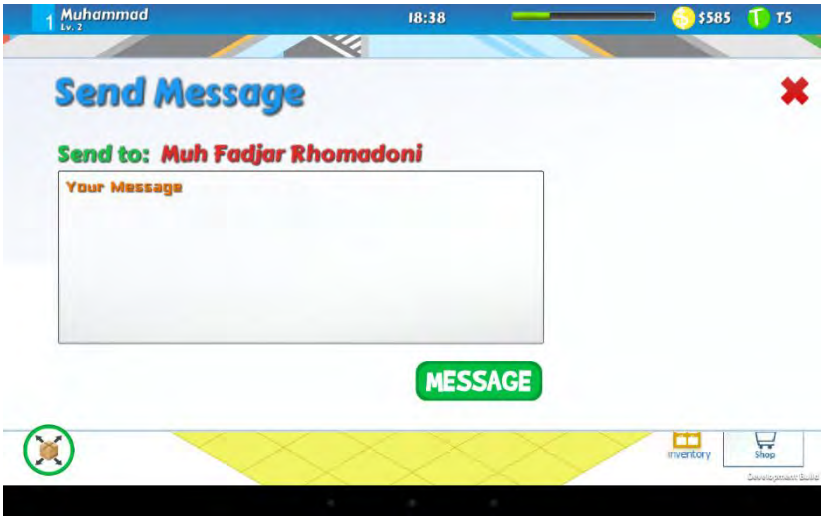


Gambar 4.5 Tampilan Halaman *Send Gift*

Pada Gambar 4.5 terdapat kotak yang berisi objek-objek hadiah yang bisi dikirim kepada teman. Ketika memilih salah satu hadiah, kemudian akan muncul tombol *send* untuk mengirim hadiah yang telah dipilih kepada teman yang dituju.

4.2.5. Implementasi Antarmuka Halaman *Send Message*

Halaman *send message* muncul ketika pemain menekan tombol *send message* pada halaman menu sosial setelah pemain memilih nama teman dari daftar hasil pencarian teman. Implementasi antarmuka halaman *send message* sebagaimana pada Gambar 4.6.



Gambar 4.6 Tampilan Halaman *Send Message*

Pada Gambar 4.6 terdapat kotak tulisan untuk mengisi pesan yang akan dikirim kepada teman yang dituju yang tertera diatas kotak tulisan. Kemudian terdapat tombol *message* untuk mengirim pesan yang telah ditulis tersebut.

4.3. Implementasi Kasus Penggunaan

Pada bagian ini dijelaskan mengenai implementasi kasus penggunaan dalam modul editor ruangan dan fitur sosial.

4.3.1. Implementasi *Login* dengan Facebook

Pada saat memulai permainan, akan ditampilkan halaman depan untuk melakukan proses *login* terlebih dahulu. Untuk dapat melakukan *login*, pemain perlu menekan tombol *Login*. Saat tombol ditekan, maka kelas *event handler* pada tombol Login yaitu BtLoginScript akan memanggil fungsi CallFBLogin yang terdapat pada kelas MainMenu. Kode Sumber 4.3 menunjukkan potongan fungsi CallFBLogin.

```
1   public void CallFBLogin()
2   {
3
4   FB.Login("email,publish_actions,user_friends",
5   LoginCallback);
6   }
7
8   private void LoginCallback(FBResult result)
9   {
10      if (result.Error != null)
11      {
12          status.GetComponent<TextMesh>().text =
13  "Error Response:\n" + result.Error;
14      }
15      else if (!FB.IsLoggedIn)
16      {
17          status.GetComponent<TextMesh>().text =
18  "Login cancelled by Player";
19      }
20      else
21      {
22          status.GetComponent<TextMesh>().text =
23  "Login was successful!";
24          PlayerData.playerid = FB.UserId;
25          PlayerData.haslogin = true;
26          Managers.Debug.print ("userid :" +
27  PlayerData.playerid);
28          Managers.Debug.print ("accestoken: " +
29  FB.AccessToken);
30          isLogin = true;
31          GetAPIData();
32      }
33  }
```

Kode Sumber 4.3 Potongan Fungsi CallFBLogin

Setelah mendapatkan data identitas Facebook, sistem memeriksa data identitas tersebut dengan data yang terdapat pada *server*. Jika data sudah terdapat pada *server*, maka sistem akan mengambil data objek pemain yang terdapat dalam server. Jika data tidak ditemukan, maka sistem akan membuat objek *default* untuk pemain. Kode Sumber 4.4 menunjukkan potongan fungsi proses pemeriksaan identitas ke *server*.

```
1   if(Managers.Service.CheckPlayer
2   (PlayerData.playerid))
3   {
4       status.GetComponent<TextMesh>().text =
5   "loading player data";
6
7       WWW www = new
8   WWW("http://199.175.51.79:8000/FM/TActiveItems/"
9   + PlayerData.playerid);
10      while(!www.isDone)
11      {
12      }
13
14      WWW www2 = new
15  WWW("http://199.175.51.79:8000/FM/TActiveStalls/"
16  + PlayerData.playerid);
17      while(!www2.isDone)
18      {
19      }
20
21      if (www.isDone &&
22  string.IsNullOrEmpty(www.error) && www2.isDone &&
23  string.IsNullOrEmpty(www2.error))
24      {
25          strItemUrl = www.text;
26          strKedaiUrl = www2.text;
27          listItemUrl =
28  (List<object>)Json.Deserialize (strItemUrl);
29          listKedaiUrl =
30  (List<object>)Json.Deserialize (strKedaiUrl);
31
32          GenerateObject();
33      }
34      else
35      {
36          Managers.Debug.print("ambil data gagal");
37      }
38  }
39  else
40  {
41      if(Managers.Service.Login(PlayerData.name,
42  PlayerData.playerid))
43      {
```

```
44          status.GetComponent<TextMesh>().text =
45  "you are registered now!";
46
47  btLogin.GetComponent<BtLoginScript>().IsLogin();
48      }
49      else
50      {
51          if(Managers.Service.CheckPlayer
52  (PlayerData.playerid))
53          {
54              status.GetComponent<TextMesh>().text
55  = "you are registered now!";
56
57  btLogin.GetComponent<BtLoginScript>().IsLogin();
58          }
59          else
60          {
61              status.GetComponent<TextMesh>().text
62  = "can't create player, check your internet
63  connection";
64
65  btLogin.GetComponent<BtLoginScript>().IsNotLogin(
67  );
68          }
69      }
70  }
```

Kode Sumber 4.4 Potongan Fungsi Pemeriksaan ke Server

4.3.2. Implementasi Masuk Halaman Utama

Saat pemain telah berhasil *login*, pada layar akan terdapat tombol *play*. Untuk masuk ke halaman utama, maka pemain akan menekan tombol tersebut. Ketika tombol ditekan maka proses yang berjalan adalah menampilkan level yang bernama *Game*PlayTest. Sedangkan level saat masih di halaman login bernama MainMenu. Saat menekan tombol *play* maka sistem akan menutup level MainMenu dan menampilkan level *Game*PlayTest. Saat menampilkan level *Game*PlayTest, sistem menjalankan kelas BaseScript yang berisi fungsi melakukan *instantiate Tile* dan objek-objek pujasera pemain. Kode Sumber 4.5 menunjukkan potongan fungsi untuk memindah level.

```
1   void HandleStateChanged(object sender,
2   TouchScript.Events.GestureStateChangeEventArgs e)
3   {
4       if (e.State ==
5   Gesture.GestureState.Recognized)
6       {
7           Managers.Debug.print("button clicked");
8           if(isLogin == false && stateButton ==
9   true)
10          {
11
12  transform.GetComponent<SpriteRenderer>().sprite =
13  spLoading;
14
15  scriptObj.GetComponent<MainMenu>().CallFBLogin();
16              stateButton = false;
17          }
18          else if(isLogin == true && stateButton ==
19  true)
20          {
21
22  transform.GetComponent<SpriteRenderer>().sprite =
23  spLoading;
24
25  Application.LoadLevel("GamePlayTest");
26          }
27      }
28  }
```

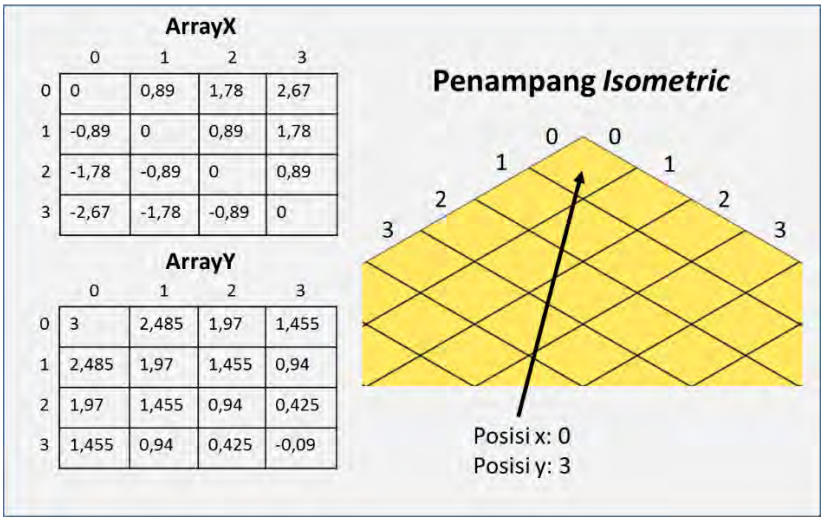
Kode Sumber 4.5 Potongan Fungsi Memindah Level/*Scene*

4.3.2.1. Penampang Ruangan *Isometric*

Pada saat pemain memasuki halaman permainan utama, maka akan ditampilkan ruangan pujasera dalam bentuk *isometric*. Pembuatan penampang ruangan *isometric* ini dilakukan dengan melakukan representasi *array* dua dimensi yang menyimpan posisi aktual dalam kerangka Unity ke dalam kerangka posisi *isometric*. Terdapat dua *array* yang menyimpan posisi aktual setiap *tile* dari kerangka Unity terhadap derajat x dan y, yaitu arrayX dan arrayY. Kemudian objek *Tile* digambar ke permukaan dengan posisi x terhadap arrayX dan posisi y terhadap arrayY pada indeks *array* yang

sama untuk setiap objek. Setiap objek *Tile* memiliki komponen TileScript untuk menyimpan posisi *grid isometric* setiap *Tile* dimana posisi *grid* x diambil dari indeks *array* dimensi pertama, dan posisi *grid* y diambil dari indeks *array* dimensi kedua.

Penyimpanan posisi *grid* pada setiap objek *Tile* digunakan untuk memandu objek-objek yang akan diletakkan pada suatu posisi tertentu dimana terlebih dahulu akan mengambil posisi *grid* sesuai *Tile* yang dituju sehingga posisi objek berada tepat didalam suatu *Tile* dari penampang *isometric*. Gambar 4.7 menunjukkan ilustrasi penerapan posisi *grid* dari *array* yang menyimpan posisi aktual. Sedangkan implementasinya seperti ditunjukkan pada Kode Sumber 4.6.



	0	1	2	3
0	0	0,89	1,78	2,67
1	-0,89	0	0,89	1,78
2	-1,78	-0,89	0	0,89
3	-2,67	-1,78	-0,89	0

	0	1	2	3
0	3	2,485	1,97	1,455
1	2,485	1,97	1,455	0,94
2	1,97	1,455	0,94	0,425
3	1,455	0,94	0,425	-0,09

Gambar 4.7 Ilustrasi Implementasi Penampang *Isometric*

```
1  void TileInstantiate ()
2  {
3      arrayX = new double[100,100];
4      arrayY = new double[100,100];
5
```

```
6       for(int i = 0; i<100; i++)
7       {
8           for(int j = 0; j<100; j++)
9           {
10              arrayX[i,j] = (0.89 * j) - (0.89 *
11      i);
12              arrayY[i,j] = 3 - (0.515 * j) -
13      (0.515 * i);
14          }
15      }
16
17      curTile = Managers.Data.PlayerData.curTile;
18      for (int i = 0; i<curTile; i++)
19      {
20          for (int j = 0; j<curTile; j++)
21          {
22              if(i == 0)
23              {
24                  var wallInstant =
25      Instantiate(wallPrefab2) as Transform;
26                  wallInstant.position = new
27      Vector3((float)arrayX[i,j], (float)arrayY[i,j],
28      baseTile.transform.position.z);
29                  wallInstant.parent =
30      baseTile.transform;
31                  TileScript tilePos2 =
32      wallInstant.GetComponent<TileScript>();
33                  tilePos2.posX = i;
34                  tilePos2.posY = j;
35
36      wallInstant.GetChild(0).GetComponent<SpriteRender
37      er>().sortingOrder = ((int)tilePos2.posY * 10) +
38      (int)tilePos2.posX;
39              }
40
41              if(j == 0)
42              {
43                  var wallInstant =
44      Instantiate(wallPrefab1) as Transform;
45                  wallInstant.position = new
46      Vector3((float)arrayX[i,j], (float)arrayY[i,j],
47      baseTile.transform.position.z);
48                  wallInstant.parent =
49      baseTile.transform;
```

```
50                  TileScript tilePos1 =
51  wallInstant.GetComponent<TileScript>();
52                  tilePos1.posX = i;
53                  tilePos1.posY = j;
54
55  wallInstant.GetChild(0).GetComponent<SpriteRender
56  er>().sortingOrder = ((int)tilePos1.posY * 10) +
57  (int)tilePos1.posX;
58
59                  if(i>0 && i<3)
60                  {
61
62  wallInstant.GetChild(0).GetComponent<SpriteRender
63  er>().sprite =
64  wallInstant.GetChild(0).GetComponent<SpriteScript
65  >().sprite1;
66                  }
67              }
68              var tileInstant =
69  Instantiate(tilePrefab) as Transform;
70              tileInstant.position = new
71  Vector3((float)arrayX[i,j], (float)arrayY[i,j],
72  baseTile.transform.position.z);
73              tileInstant.parent =
74  baseTile.transform;
75              TileScript tilePos =
76  tileInstant.GetComponent<TileScript>();
77              tilePos.posX = i;
78              tilePos.posY = j;
79              tilePos.gridPosition = new
80  GridPosition(i, j);
81          }
82      }
83  }
```

Kode Sumber 4.6 Potongan Fungsi Membuat Penampang *Isometric*

4.3.3. Implementasi Mengolah Objek Pujasera

Pada saat berada di halaman utama, pemain dapat melakukan pengelolaan objek pujasera. Untuk dapat mengelola objek, pemain perlu menekan tombol Dekor untuk beralih ke *state* Dekor. Saat berada pada *mode* Dekor, pemain dapat menekan pada objek

tertentu untuk melakukan pengelolaan. Pengelolaan objek terdapat beberapa opsi, yaitu memindah posisi, merotasi arah, dan menghapus objek. Kode Sumber 4.7 menunjukkan potongan fungsi untuk mengubah *state* permainan.

```
1   public void OnClickAction(GameObject go)
2   {
3       if (StateManager.currentState ==
4   GameState.GAMEPLAY)
5       {
6           if (!isDecor)
7           {
8
9   scriptObj.GetComponent<BaseScript>().isDecormode
10  = true;
11
12  usedSprite.GetComponent<SpriteRenderer>().sprite
13  = homeSprite;
14              isDecor = true;
15              StateManager.currentState =
16  GameState.DEKOR;
17              print("Dekor Active");
18          }
19      }
20      else if (StateManager.currentState ==
21  GameState.DEKOR)
22      {
23          if (isDecor &&
24  !scriptObj.GetComponent<BaseScript>().isPickObj)
25          {
26
27  scriptObj.GetComponent<BaseScript>().isDecormode
28  = false;
29
30  usedSprite.GetComponent<SpriteRenderer>().sprite
31  = decorSprite;
32              isDecor = false;
33              StateManager.currentState =
34  GameState.GAMEPLAY;
35              print("Gameplay Active");
36          }
37      }
38  }
```

Kode Sumber 4.7 Potongan Fungsi Mengubah *State* Permainan

Setelah berada pada *state* Dekor, jika pemain ingin memindahkan objek, pemain akan menekan tombol Pindah Posisi. Ketika menekan tombol posisi, *event handler* pada kelas BtMoveScript akan mengisi nilai variabel commandCode = 1. Setelah itu pemain memilih posisi dimana objek akan dipindah dengan menekan ke area *tile* yang kosong. Saat menekan area *tile* yang kosong, *event handler* pada kelas PickScript akan melakukan fungsi memindah posisi objek. Kode Sumber 4.8 menunjukkan potongan fungsi untuk memindah posisi objek

```
1   if(objScript.GetComponent<BaseScript>().isPickObj
2   &&
3   objScript.GetComponent<BaseScript>().objOnPicked
4   &&
5   objScript.GetComponent<BaseScript>().commandCode
6   == 1)
7   {
8       objOnPicked =
9   objScript.GetComponent<BaseScript>().objOnPicked;
10      objOnPicked.position = transform.position;
11      int posX =
12  (int)transform.GetComponent<TileScript>().posX;
13      int posY =
14  (int)transform.GetComponent<TileScript>().posY;
15
16      objOnPickedChild = objOnPicked.GetChild(0);
17
18  objOnPickedChild.GetComponent<SpriteRenderer>().s
19  ortingOrder = (posY * 10) + posX;
20
21  objOnPicked.GetComponent<FMSView>().DataFMS.GridP
22  osition =
23  transform.GetComponent<TileScript>().gridPosition
24  ;
25  }
```

Kode Sumber 4.8 Potongan Fungsi Memindah Posisi Objek

Pemain juga bisa memutar arah objek pada pujasera dengan menekan tombol Rotasi. Saat tombol ditekan, *event handler* pada kelas BtRotateScript akan menjalankan fungsi memutar arah objek. Kode Sumber 4.9 menunjukkan potongan fungsi untuk memutar arah objek.

```
objName = objOnRotated.name;
curSprite =
objOnRotated.GetComponent<ObjScript>().curSprite;
if (curSprite + 1 < maxRotate)
{
    objOnRotated.GetComponent<SpriteRenderer>().sprite=
objOnRotated.GetComponent<SpriteScript>().GetSprite(cur
Sprite + 1);
    objOnRotated.GetComponent<ObjScript>().curSprite =
curSprite + 1;
}
else
{
    objOnRotated.GetComponent<SpriteRenderer>().sprite=
objOnRotated.GetComponent<SpriteScript>().GetSprite(0);
    objOnRotated.GetComponent<ObjScript>().curSprite =
0;
}
```

Kode Sumber 4.9 Potongan Fungsi Memutar Arah Objek

Untuk menghapus objek, pemain akan menekan tombol Hapus. Saat tombol ditekan, *event handler* pada kelas BtDeleteScript akan menjalankan fungsi untuk menghapus objek. Kode Sumber 4.10 menunjukkan potongan fungsi untuk menghapus objek.

```
if (baseScript.isPickObj &&
baseScript.objOnPicked)
{
    baseScript.commandCode = 3;

    objOnDeleted = baseScript.objOnPicked;
    Destroy(objOnDeleted.gameObject);
    baseScript.isPickObj = false;

Inventory.Add(objOnDeleted.GetComponent<FMSView>(
).DataFMS);


```

```
14
15  baseScript.DeleteFMS(objOnDeleted.GetComponent<FM
16  SView>().index);
17
18  Managers.Debug.print(objOnDeleted.GetComponent<FM
19  SView>().index.ToString());
20      Managers.Debug.print("Deleting: " +
21  objOnDeleted.GetComponent<FMSView>().DataFMS.Name
22  );
23
24  inventory.GetComponent<InventoryView>().UpdateLis
    tInventory();
    }
```

Kode Sumber 4.10 Potongan Fungsi Menghapus Objek

4.3.4. Implementasi Mencari Nama Teman

Pemain dapat melakukan pencarian teman Facebook yang juga bermain *game* Food Merchant Saga. Terlebih dahulu pemain harus masuk ke menu sosial. Di halaman menu sosial terdapat kotak tulisan untuk diisi dengan kata kunci pencarian yang diinginkan. Kemudian setelah itu pemain menekan tombol *search*. Pada saat tombol ditekan, *event handler* pada kelas SocialView akan menjalankan fungsi untuk memanggil fungsi CallFBSearch pada kelas SocialFB. Lalu melakukan *deserialize* terhadap kembalian fungsi yang bertipe JSON dan diisikan pada variable bertipe *list* yang bernama friends pada kelas SocialView. Kode Sumber 4.11 menunjukkan potongan fungsi untuk melakukan pencarian teman.

```
1   public void CallFBSearch(string query)
2   {
3       searchQuery = query;
4
5   FB.API("/me?fields=id,first_name,friends.fields(n
6   ame,id)", Facebook.HttpMethod.GET,
7   SearchCallback);
8   }
9
10  private void SearchCallback(FBResult result)
```

```
{
    if(result.Error != null)
    {
        FB.API("/me?fields=friends.limit(100))",
Facebook.HttpMethod.GET, SearchCallback);
        return;
    }

    Managers.Debug.print("callback: " +
result.Text);
    SocialView.friends =
FBUtil.DeserializeJSONFriends (result.Text);
    transform.GetComponent<SocialView>
().ShowFriends ();
}
```

Kode Sumber 4.11 Potongan Fungsi Pencarian Teman

4.3.5. Implementasi Berinteraksi dengan Teman

Pada menu sosial, setelah mencari teman dan mendapatkan daftar teman yang memenuhi kriteria, pemain dapat melakukan interaksi dengan teman yang diinginkan. Pertama pemain memilih teman untuk berinteraksi dengan menekan pada teman yang diinginkan. Kemudian tombol opsi interaksi menjadi aktif. Terdapat tiga tipe interaksi, yaitu mengirim hadiah, mengunjungi, dan mengirim pesan.

Saat pemain menekan tombol *send gift*, maka sistem akan melakukan proses mengirimkan hadiah. Sistem menangkap objek teman yang terpilih lalu menampilkan halaman untuk Menu *Gift.* Kode Sumber 4.12 menunjukkan potongan fungsi untuk menampilkan Menu *Gift*

```
private void OnClickGift(GameObject go)
{
    StateManager.AnimateIn (menuGift);
    gameObject.SetActive (false);
}
```

Kode Sumber 4.12 Potongan Fungsi Menampilkan Menu *Gift*

Pada halaman menu *Gift*, pemain memilih hadiah yang akan diberikan lalu menekan tombol *send* untuk mengeksekusi fungsi pengiriman hadiah. *Event handler* pada kelas GiftView akan melakukan pemanggilan fungsi CallFBGift yang terdapat pada kelas SocialFB. Kode Sumber 4.13 menunjukkan potongan fungsi CallFBGift.

```
1  public void CallFBGift(string idFriend)
2  {
3      string giftMessage = "I've sent you a gift!";
4      string giftToId = idFriend;
5
6      var queryGift = new Dictionary<string,
7  string>();
8      queryGift["gift"] =
9  "http://samples.ogp.me/314820825348714";
10     queryGift["message"] = giftMessage;
11     queryGift ["tags"] = giftToId;
12     FB.API ("/me/foodmerchantsaga:send",
13 Facebook.HttpMethod.POST, Callback, queryGift);
14 }
```

Kode Sumber 4.13 Potongan Fungsi CallFBGift

Jika pemain menekan tombol *send message* maka sistem akan menampilkan halaman Menu *Message*. Pada halaman ini terdapat kotak tulisan untuk mengisi pesan yang akan dikirim, lalu ketika pemain menekan tombol *send message*, maka *event handler* pada kelas MessageView akan menjalankan fungsi CallFBMessage yang terdapat pada kelas SocialFB. Kode Sumber 4.14 menunjukkan potongan fungsi CallFbMessage.

```
1  public void CallFBMessage(string title, string
2  message, string idFriend)
3  {
4      string tellMessage = message;
5      string tellToId = idFriend;
6
7      var queryTell = new Dictionary<string,
8  string>();
9      queryTell["profile"] = tellToId;
```

```
10      queryTell["message"] = tellMessage;
11      queryTell ["tags"] = tellToId;
12      FB.API ("/me/foodmerchantsaga:tell",
13  Facebook.HttpMethod.POST, Callback, queryTell);
14  }
15
```

Kode Sumber 4.14 Potongan Fungsi CallFBMessage

Jika pemain menekan tombol *visit*, sistem akan melakukan koneksi ke server untuk mengambil data objek dari teman yang dipilih. Kemudian sistem akan memindah level ke level *Game*PlayVisit. Pada level *Game*PlayVisit sistem menampilkan susunan tata letak objek pujasera teman yang dikunjungi. Kode Sumber 4.15 menunjukkan potongan fungsi menampilkan objek pujasera teman.

```
1   private void InitData()
2   {
3       try
4       {
5           listItem = (List<object>)Json.Deserialize
6   (dataCarrier.GetComponent<DataScript>().strItemFr
7   iend);
8           listKedai =
9   (List<object>)Json.Deserialize
10  (dataCarrier.GetComponent<DataScript>().strKedaiF
11  riend);
12      }
13      catch(Exception e)
14      {
15          Managers.Debug.print(e.Message);
16      }
17
18      GenerateObject (listItem);
19      GenerateObject (listKedai);
20  }
```

Kode Sumber 4.15 Potongan Fungsi Menampilkan Objek Teman

4.3.6. Implementasi Berbagi Capaian

Pada saat bermain *game* Food Merchant Saga, pemain akan mendapatkan suatu capaian tertentu, misalnya, kenaikan level,

pertambahan luas ruangan pujasera, dan lain sebagainya. Pada saat pemain mendapatkan suatu capaian tersebut, sistem akan melakukan koneksi dengan Facebook untuk mengirimkan data capaian dan menrbitkannya melalui akun Facebook pemain sehingga dapat dibaca oleh teman-teman Facebook pemain. Kode Sumber 4.16 menunjukkan potongan fungsi untuk melakukan berbagi capaian tersebut.

```
1  public void CallFBShare(string link, string
2  linkName, string linkCaption, string
3  linkDescription)
4  {
5      string FeedLink = link;
6      string FeedLinkName = linkName;
7      string FeedLinkCaption = linkCaption;
8      string FeedLinkDescription = linkDescription;
9      string FeedReference = "";
10
11     FB.Feed(
12         link: FeedLink,
13         linkName: FeedLinkName,
14         linkCaption: FeedLinkCaption,
15         linkDescription: FeedLinkDescription,
16         reference: FeedReference,
17         callback: Callback
18     );
19 }
```

Kode Sumber 4.16 Potongan Fungsi Berbagi Capaian

4.3.7. Implementasi Mengundang Teman

Pada halaman menu sosial, pemain dapat mengundang teman-teman Facebook pemain yang belum bermain Food Merchant Saga. Undangan akan dikirim dalam bentuk notifikasi di halaman Facebook teman yang diundang. Untuk dapat mengundang teman, pemain menekan tombol *invite* yang berada pada menu sosial. Kemudian *event handler* pada kelas SocialView akan memanggil fungsi CallFBInvite yang berada pada kelas SocialFB. Kode Sumber 4.17 menunjukkan potongan fungsi untuk mengundang teman.

```
1  public void CallFBInvite(string title,
2  string message)
3  {
4      string FriendSelectorTitle = title;
5      string FriendSelectorMessage = message;
6      string FriendSelectorFilters =
7  "[\"all\",\"app_users\",\"app_non_users\"]";
8
9      FB.AppRequest(
10             message: FriendSelectorMessage,
11             filters: FriendSelectorFilters,
12             title: FriendSelectorTitle,
13             callback: Callback
14     );
15 }
16
17
```

Kode Sumber 4.17 Potongan Fungsi mengundang Teman

BAB V
PENGUJIAN DAN EVALUASI

Bab ini membahas pengujian dan evaluasi pada modul yang dikembangkan. Pengujian yang dilakukan adalah pengujian terhadap kebutuhan fungsionalitas sistem dan kegunaan sistem. Pengujian fungsionalitas mengacu pada kasus penggunaan pada bab tiga. Pengujian kegunaan program dilakukan dengan mengetahui tanggapan dari pengguna terhadap sistem. Hasil evaluasi menjabarkan tentang rangkuman hasil pengujian pada bagian akhir bab ini.

5.1. Lingkungan Pengujian

Lingkungan pengujian sistem pada pengerjaan tugas akhir ini dilakukan pada lingkungan dan alat kakas sebagai berikut:

Prosesor	: Quad-core 1.2 GHz Cortex-A7
Memori	: 1 GB
Jenis Device	: *Smartphone*
Sistem Operasi	: Android 4.2.2
Ukuran Layar	: 1280 x 800 pixels, 10.1 inches.
Game Engine	: Unity

5.2. Skenario Pengujian

Pada bagian ini akan dijelaskan tentang skenario pengujian yang dilakukan. Pengujian yang dilakukan adalah pengujian kebutuhan fungsionalitas dan pengujian integritas. Pengujian fungsionalitas menggunakan metode kotak hitam (*black box*). Metode ini menekankan pada kesesuaian hasil keluaran sistem secara modular dan dijalankan secara terpisah dari modul lainnya. Sedangkan pengujian integritas untuk mengetahui apakah modul dapat berjalan ketika diintegrasikan dengan modul lainnya dalam *game* Food Merchant Saga.

5.2.1. Pengujian Fungsionalitas

Pengujian fungsionalitas sistem dilakukan dengan menyiapkan sejumlah skenario sebagai tolak ukur keberhasilan pengujian.

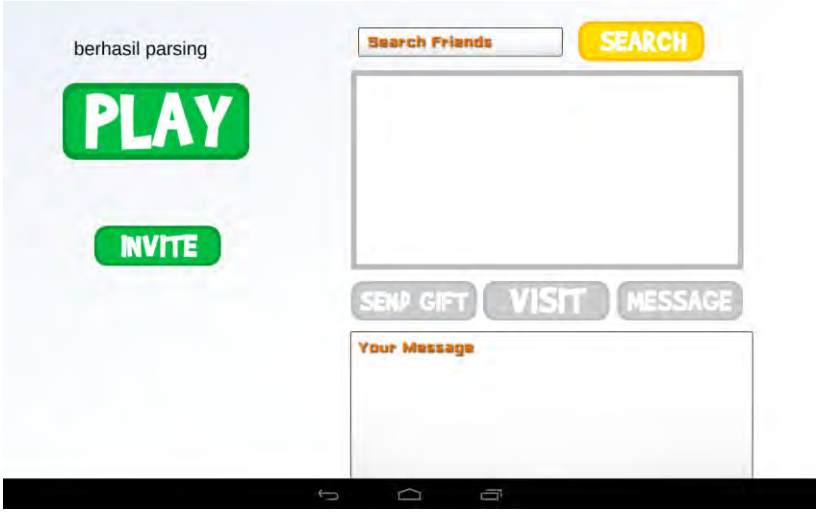
Pengujian fungsionalitas dilakukan dengan mengacu pada kasus penggunaan yang telah dijelaskan pada subbab 3.1.5. Pengujian ini bertujuan untuk menguji apakah modul yang dibuat dapat diterapkan pada berkas pengembangan Unity secara modular atau terpisah dari berkas pengembangan *game* Food Merchant Saga. Pada pengujian ini menggunakan lingkungan pengujian berupa berkas proyek Unity yang berbeda dari berkas pengembangan *game* Food Merchant Saga. Pengujian pada kebutuhan fungsionalitas dijabarkan pada subbab selanjutnya.

5.2.1.1. Pengujian *Login* dengan Facebook

Berikut ini merupakan pembahasan pengujian proses *login* melalui Facebook. Skenario pengujian dari pengujian *login* dengan Facebook ini sebagaimana pada Tabel 5.1. Dokumentasi pengujian sebagaimana pada Gambar 5.1.

Tabel 5.1 Skenario Pengujian *Login* dengan Facebook

Nama Pengujian	Pengujian *Login* dengan Facebook
Kode	PF-01
Use case	UC-01
Tujuan	Memeriksa fungsi untuk melakukan proses *login* ke Facebook berjalan atau tidak
Kondisi awal	Pengguna belum melakukan proses *login*
Skenario	1. Pengguna menekan tombol *login* yang terdapat pada layar.
Masukan	Nama pengguna dan *password* Facebook
Keluaran yang diharapkan	Tombol *login* berubah menjadi tombol *play*
Hasil pengujian	Berhasil



Gambar 5.1 Dokumentasi Pengujian PF-01

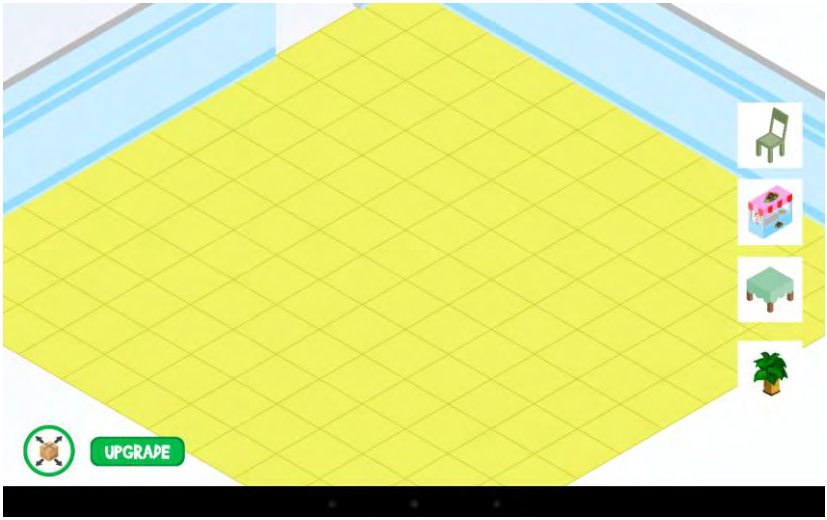
5.2.1.2. Pengujian Membuat Penampang *Isometric*

Berikut ini merupakan pembahasan pengujian membuat penampang *isometric*. Skenario pengujian dari pengujian membuat penampang *isometric* ini adalah sebagaimana ditunjukkan pada Tabel 5.2. Dokumentasi pengujian sebagaimana pada Gambar 5.2.

Tabel 5.2 Skenario Pengujian Membuat Penampang *Isometric*

Nama Pengujian	Pengujian Membuat Penampang *Isometric*
Kode	PF-02
Use case	UC-02
Tujuan	Memeriksa fungsi membuat penampang *isometric* berhasil atau tidak
Kondisi awal	Pengguna membuka halaman permainan utama
Skenario	1. Pengguna telah berhasil melakukan proses *login*. 2. Pengguna menekan tombol *play* yang terdapat pada layar.
Masukan	-

Keluaran yang diharapkan	Halaman permainan utama tertampilkan ruangan pujasera dengan penampang *isometric*
Hasil pengujian	Berhasil



Gambar 5.2 Dokumentasi Pengujian PF-02

5.2.1.3. Pengujian Mengelola Objek pada Penampang *Isometric*

Berikut ini merupakan pembahasan pengujian mengelola objek pada penampang *isometric*. Skenario pengujian pada pengujian mengelola objek pada penampang *isometric* ini terdapat tiga jenis. Skenario pertama adalah melakukan pemindahan posisi objek, sebagaimana dijelaskan pada Tabel 5.3. Skenario kedua adalah melakukan perubahan arah objek, sebagaimana dijelaskan pada Tabel 5.4. Skenario ketiga adalah melakukan penambahan dan penghapusan objek pada penampang *isometric*, sebagaimana dijelaskan pada Tabel 5.5. Dokumentasi pengujian sebagaimana pada Gambar 5.3 dan Gambar 5.4.

Tabel 5.3 Skenario Pengujian Memindah Posisi Objek

Nama Pengujian	Pengujian Memindah Posisi Objek
Kode	PF-03
Use case	UC-03
Tujuan	Memeriksa fungsi memindah objek dapat berjalan atau tidak
Kondisi awal	Pengguna berada pada halaman permainan utama
Skenario	1. Pengguna menekan salah satu objek yang akan dikelola. 2. Pengguna menekan tombol pindah posisi yang terdapat pada layar. 3. Pengguna menekan pada sembarang *tile* sebagai lokasi tujuan objek.
Masukan	-
Keluaran yang diharapkan	Objek yang dipilih berpindah posisi sesuai dengan pilihan *tile* yang dipilih pengguna
Hasil pengujian	Berhasil

Tabel 5.4 Skenario Pengujian Mengubah Arah Objek

Nama Pengujian	Pengujian Mengubah Arah Objek
Kode	PF-04
Use case	UC-03
Tujuan	Memeriksa fungsi mengubah arah objek dapat berjalan atau tidak
Kondisi awal	Pengguna berada pada halaman permainan utama
Skenario	1. Pengguna menekan salah satu objek yang akan dikelola. 2. Pengguna menekan tombol rotasi yang terdapat pada layar sebanyak yang dibutuhkan untuk mendapatkan arah yang diinginkan.
Masukan	-
Keluaran yang diharapkan	Objek yang dipilih berubah arah terhadap penampang *isometric* searah jarum jam setiap kali pengguna menekan tombol rotasi
Hasil pengujian	Berhasil

Tabel 5.5 Skenario Pengujian Menambah dan Menghapus Objek

Nama Pengujian	Pengujian Menambah dan Menghapus Objek
Kode	PF-05
Use case	UC-03
Tujuan	Memeriksa fungsi menambah dan menghapus objek dapat berjalan atau tidak
Kondisi awal	Pengguna berada pada halaman permainan utama
Skenario	1. Pengguna menekan salah satu gambar objek untuk menambah objek pada penampang *isometric.* 2. Pengguna menekan objek yang telah ditambah. 3. Pengguna menekan tombol hapus untuk menghapus objek yang dipilih.
Masukan	-
Keluaran yang diharapkan	Objek dapat ditambah ketika menekan gambar objek yang diinginkan, dan dapat dihapus ketika menekan tombol hapus.
Hasil pengujian	Berhasil

Gambar 5.3 Dokumentasi Pengujian Mengelola Objek pada Penampang *Isometric* Pertama

Gambar 5.4 Dokumentasi Pengujian Mengelola Objek pada Penampang *Isometric* Kedua

5.2.1.4. Pengujian Menampilkan Daftar Teman

Berikut ini merupakan pembahasan pengujian menampilkan daftar teman. Skenario pengujian dari pengujian menampilkan daftar teman ini sebagaimana ditunjukkan pada Tabel 5.6. Dokumentasi pengujian sebagaimana pada Gambar 5.5.

Tabel 5.6 Skenario Pengujian Menampilkan Daftar Teman

Nama Pengujian	Pengujian Menampilkan Daftar Teman
Kode	PF-06
Use case	UC-04
Tujuan	Memeriksa fungsi menampilkan daftar teman dapat berjalan atau tidak
Kondisi awal	Pengguna telah melakukan proses *login*
Skenario	1. Pengguna mengisi kata kunci pencarian pada kotak tulisan pencarian. 2. Pengguna menekan tombol *search* untuk melakukan pencarian
Masukan	Kata kunci pencarian
Keluaran yang diharapkan	Pada kotak hasil pencarian tertampilkan nama-nama teman sesuai kata kunci pencarian.
Hasil pengujian	Berhasil

Gambar 5.5 Dokumentasi Pengujian PF-06

5.2.1.5. Pengujian Berinteraksi dengan Teman

Berikut ini merupakan pembahasan pengujian berinteraksi dengan teman. Skenario pengujian pada pengujian mengelola objek pada penampang *isometric* ini terdapat tiga jenis. Skenario pertama adalah mengirimkan hadiah, sebagaimana dijelaskan pada Tabel 5.7. Dokumentasi pengujian dari skenario ini sebagaimana pada Gambar 5.6. Skenario kedua adalah mengunjungi pujasera teman, sebagaimana dijelaskan pada Tabel 5.8. Dokumentasi pengujian dari skenario ini sebagaimana pada Gambar 5.7. Skenario ketiga adalah mengirim pesan, sebagaimana dijelaskan pada Tabel 5.9. Dokumentasi pengujian dari skenario ini sebagaimana pada Gambar 5.8.

Tabel 5.7 Skenario Pengujian Mengirimkan Hadiah

Nama Pengujian	Pengujian Mengirimkan Hadiah
Kode	PF-07
Use case	UC-05
Tujuan	Memeriksa fungsi mengirimkan hadiah kepada teman dapat berjalan atau tidak
Kondisi awal	Pengguna memilih salah satu nama teman

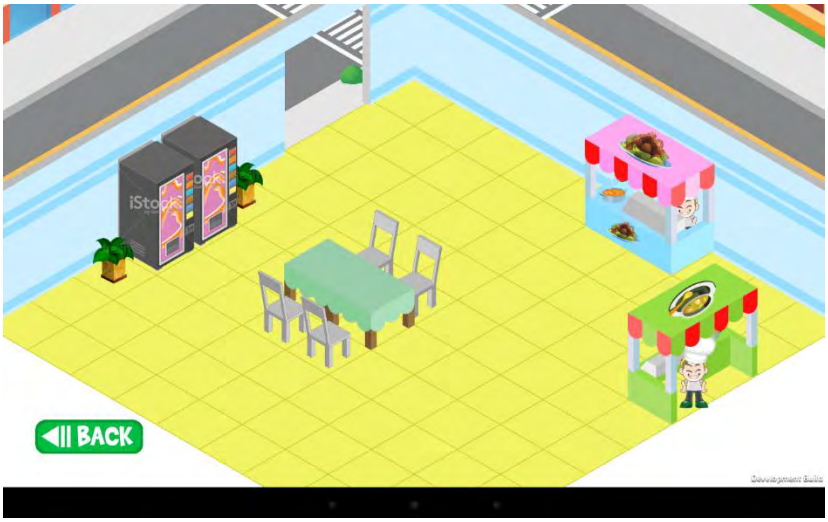
Skenario	1. Pengguna memilih salah satu nama teman dari hasil pencarian teman. 2. Pengguna menekan tombol *send gift*.
Masukan	-
Keluaran yang diharapkan	Teman mendapat notifikasi di akun Facebook bahwa telah dikirimi hadiah oleh pengguna.
Hasil pengujian	Berhasil

Gambar 5.6 Dokumentasi Pengujian PF-07

Tabel 5.8 Skenario Pengujian mengunjungi Pujasera Teman

Nama Pengujian	Pengujian Mengunjungi Pujasera Teman
Kode	PF-08
Use case	UC-05
Tujuan	Memeriksa fungsi mengunjungi pujasera teman dapat berjalan atau tidak
Kondisi awal	Pengguna memilih salah satu nama teman
Skenario	1. Pengguna memilih salah satu nama teman dari hasil pencarian teman. 2. Pengguna menekan tombol *visit*.
Masukan	-
Keluaran yang diharapkan	Sistem menampilkan halaman berisi teta letak objek-objek pujasera teman yang dipilih.

Hasil pengujian	Berhasil



Gambar 5.7 Dokumentasi Pengujian PF-08

Tabel 5.9 Skenario Pengujian Mengirimkan Pesan

Nama Pengujian	Pengujian Mengirimkan Pesan
Kode	PF-09
Use case	UC-05
Tujuan	Memeriksa fungsi mengirimkan pesan kepada teman dapat berjalan atau tidak
Kondisi awal	Pengguna memilih salah satu nama teman
Skenario	1. Pengguna memilih salah satu nama teman dari hasil pencarian teman. 2. Pengguna mengisi kotak tulisan pesan sesuai dengan pesan yang ingin ditulis. 3. Pengguna menekan tombol *message*.
Masukan	Kalimat pesan
Keluaran yang diharapkan	Teman mendapat notifikasi di akun Facebook bahwa telah dikirimi pesan oleh pengguna.
Hasil pengujian	Berhasil

Gambar 5.8 Dokumentasi Pengujian PF-09

5.2.1.6. Pengujian Berbagi Capaian

Berikut ini merupakan pembahasan pengujian berbagi capaian. Skenario pengujian dari pengujian berbagi capaian ini sebagaimana ditunjukkan pada Tabel 5.10. Dokumentasi pengujian sebagaimana pada Gambar 5.9.

Tabel 5.10 Skenario Pengujian Berbagi Capaian

Nama Pengujian	Pengujian Berbagi Capaian
Kode	PF-10
Use case	UC-06
Tujuan	Memeriksa fungsi berbagi capaian dapat berjalan atau tidak
Kondisi awal	Pengguna berada pada halaman permainan utama

Skenario	1. Pengguna menekan tombol *upgrade* untuk menambah luas pujasera.
Masukan	-
Keluaran yang diharapkan	Akun Facebook pengguna membagikan aktivitas pencapaian yang dilakukan pengguna dalam permainan.
Hasil pengujian	Berhasil

Gambar 5.9 Dokumentasi Pengujian PF-10

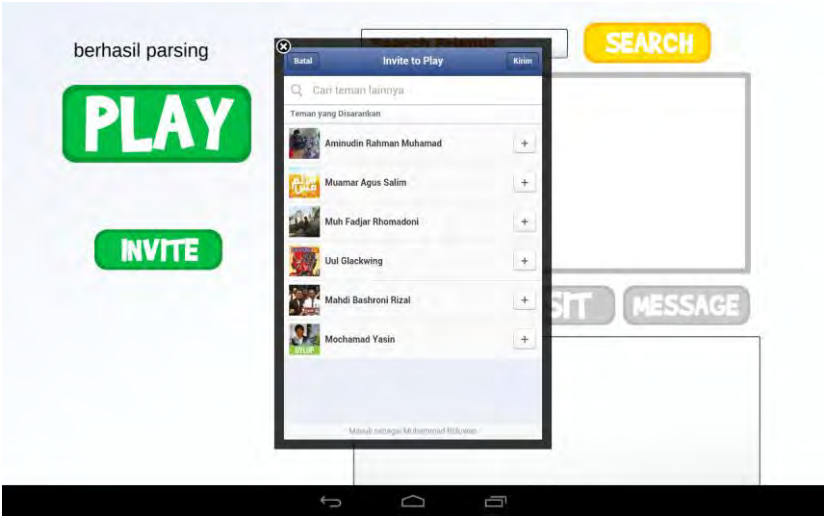
5.2.1.7. Pengujian Mengundang Teman

Berikut ini merupakan pembahasan pengujian mengundang teman. Skenario pengujian sebagaimana ditunjukkan pada Tabel 5.11. Dokumentasi pengujian sebagaimana pada Gambar 5.10.

Tabel 5.11 Skenario Pengujian Mengundang Teman

Nama Pengujian	Pengujian Mengundang Teman
Kode	PF-11
Use case	UC-07
Tujuan	Memeriksa fungsi mengundang teman dapat berjalan atau tidak
Kondisi awal	Pengguna telah berhasil melakukan *login*.

Skenario	1. Pengguna menekan tombol *invite* untuk mengundang teman bermain Food Merchant Saga melalui Facebook. 2. Pengguna memilih teman yang akan dikirimi undangan. 3. Pengguna menekan tombol *done*.
Masukan	-
Keluaran yang diharapkan	Akun Facebook teman yang dipilih akan mendapatkan notifikasi pengundangan dari pengguna
Hasil pengujian	Berhasil



Gambar 5.10 Dokumentasi Pengujian PF-12

5.2.2. Pengujian Integritas

Pengujian integritas sistem dilakukan dengan menyiapkan sejumlah skenario sebagai tolak ukur keberhasilan pengujian. Pengujian integritas dilakukan dengan mengacu pada kasus penggunaan yang telah dijelaskan pada subbab 3.1.5. Pengujian ini bertujuan untuk menguji apakah modul yang dibuat dapat diterapkan

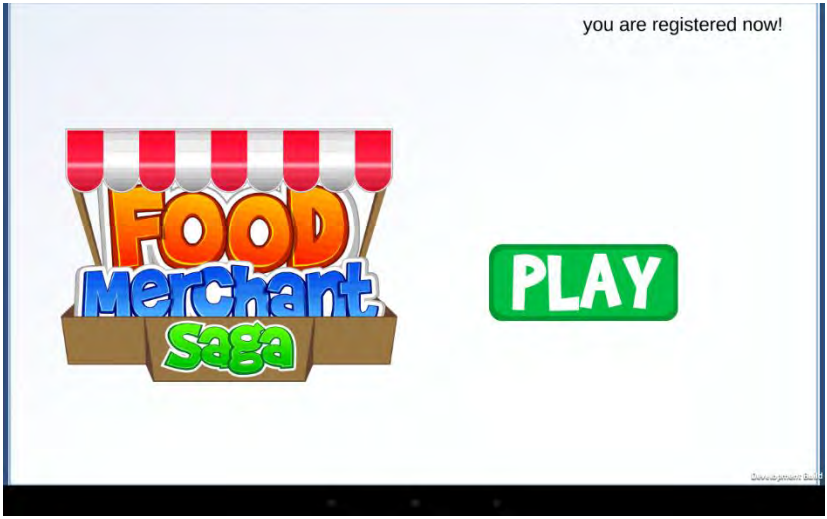
terintegrasi dengan modul lainnya pada pengembangan permainan Food Merchant Saga. Pada pengujian ini menggunakan lingkungan pengujian berupa berkas pengembangan *game* Food Merchant Saga. Macam-macam skenario pengujian integritas dijabarkan pada subbab selanjutnya.

5.2.2.1. Pengujian Login dengan Facebook

Berikut ini merupakan pembahasan pengujian integritas pada kasus penggunaan *login* dengan Facebook (UC-01). Terdapat dua skenario pengujian yang diterapkan pada pengujian ini. Skenario pertama adalah proses *login* dengan pengguna belum terdaftar dalam *database server*, sebagaimana dijelaskan pada Tabel 5.12. Dokumentasi pengujian dari skenario pengujian ini sebagaimana pada Gambar 5.11. Skenario kedua adalah proses *login* dengan pengguna telah terdaftar dalam *database server*, sebagaimana dijelaskan pada Tabel 5.13. Dokumentasi pengujian dari skenario pengujian ini sebagaimana pada Gambar 5.12.

Tabel 5.12 Skenario Pengujian Integritas Proses *Login* Pengguna Baru

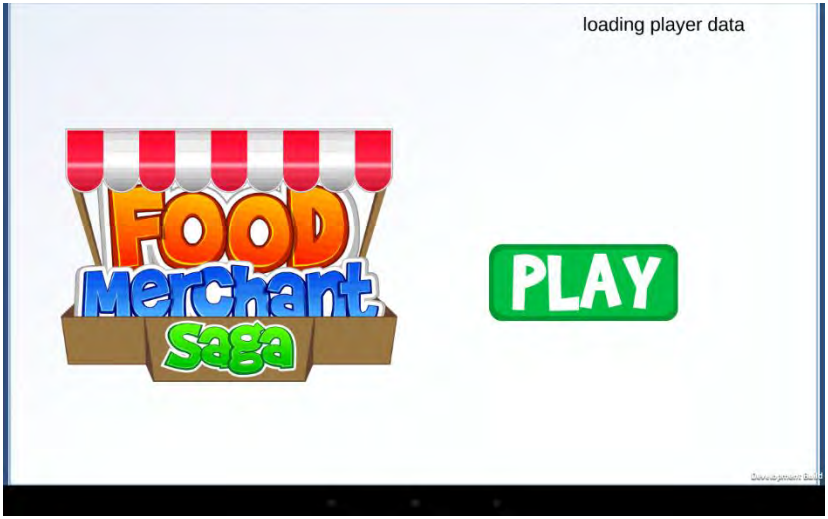
Nama Pengujian	Pengujian Integritas Proses *Login* Pengguna Baru
Kode	PI-01
Use case	UC-01
Tujuan	Memeriksa fungsi *login* pengguna baru dapat berjalan atau tidak
Kondisi awal	Pengguna berada pada halaman *login*.
Skenario	1. Pengguna menekan tombol *login* yang terdapat pada layar.
Masukan	Nama pengguna dan *password* Facebook
Keluaran yang diharapkan	Muncul pemberitahuan bahwa registrasi berhasil, dan tombol *login* berubah menjadi tombol *play*.
Hasil pengujian	Berhasil



Gambar 5.11 Dokumentasi Pengujian PI-01

Tabel 5.13 Skenario Pengujian Integritas Proses *Login* Pengguna Terdaftar

Nama Pengujian	Pengujian Integritas Proses *Login* Pengguna Terdaftar
Kode	PI-02
Use case	UC-01
Tujuan	Memeriksa fungsi *login* pengguna terdaftar dapat berjalan atau tidak
Kondisi awal	Pengguna berada pada halaman *login*.
Skenario	1. Pengguna menekan tombol *login* yang terdapat pada layar.
Masukan	Nama pengguna dan *password* Facebook
Keluaran yang diharapkan	Muncul pemberitahuan bahwa sedang mengambil data permainan, dan tombol *login* berubah menjadi tombol *play*.
Hasil pengujian	Berhasil



Gambar 5.12 Dokumentasi Pengujian PI-02

5.2.2.2. Pengujian Menampilkan Penampang *Isometric* pada Halaman Permainan Utama

Berikut ini merupakan pembahasan pengujian integritas pada kasus penggunaan masuk halaman utama (UC-02). Skenario pengujian pada pengujian menampilkan penampang *isometric* pada halaman permainan utama ini sebagaimana ditunjukkan pada Tabel 5.14. Dokumentasi pengujian sebagaimana pada Gambar 5.13.

Tabel 5.14 Skenario Pengujian Integritas Menampilkan Penampang *Isometric* pada Halaman Permainan Utama

Nama Pengujian	Pengujian Integritas Menampilkan Penampang *Isometric* pada Halaman Permainan Utama
Kode	PI-03
Use case	UC-02
Tujuan	Memeriksa fungsi menampilkan penampang *isometric* dapat berjalan atau tidak
Kondisi awal	Pengguna telah berhasil melakukan *login*

Skenario	1. Pengguna menekan tombol *play* yang terdapat pada layar.
Masukan	-
Keluaran yang diharapkan	Muncul penampang ruangan *isometric* beserta objek-objek pujasera pemain.
Hasil pengujian	Berhasil

Gambar 5.13 Dokumentasi Pengujian PI-03

5.2.2.3. Pengujian Mengolah Objek Pujasera

Berikut ini merupakan pembahasan pengujian integritas pada kasus penggunaan mengolah objek pujasera (UC-03). Skenario pengujian pada pengujian mengolah objek pujasera ini sebagaimana ditunjukkan pada Tabel 5.15. Dokumentasi pengujian sebagaimana pada Gambar 5.14.

Tabel 5.15 Skenario Pengujian Integritas Mengolah Objek Pujasera

Nama Pengujian	Pengujian Integritas Mengolah Objek Pujasera
Kode	PI-04
Use case	UC-03

Tujuan	Memeriksa fungsi mengolah objek pujasera dapat berjalan atau tidak
Kondisi awal	Pengguna berada pada halaman permainan utama
Skenario	1. Pengguna menekan tombol menu *shop*. 2. Pengguna memilih *item* pada menu *shop*. 3. Pengguna menekan tombol *buy*. 4. Pengguna menekan tombol Dekor untuk mengubah menjadi *state* Dekor. 5. Pengguna menekan pada suatu objek. 6. Pengguna menekan tombol pindah posisi untuk memindah posisi 7. Pengguna menekan pada *tile* tertentu untuk menempatkan objek. 8. Pengguna menekan tombol rotasi untuk mengubah arah objek. 9. Pengguna menekan tombol hapus untuk menghapus objek 10. Pengguna menekan tombol selesai untuk mengakhiri mengelola objek
Masukan	-
Keluaran yang diharapkan	Menampilkan tata letak objek pujasera sesuai dengan pengelolaan pengguna
Hasil pengujian	Berhasil

Gambar 5.14 Dokumentasi Pengujian PI-04

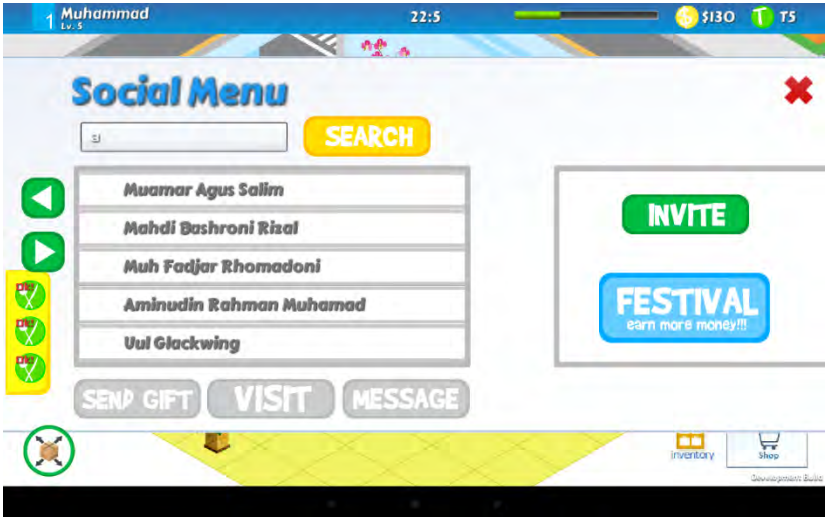
5.2.2.4. Pengujian Mencari Nama Teman

Berikut ini merupakan pembahasan pengujian integritas pada kasus penggunaan mencari nama teman (UC-04). Skenario pengujian pada pengujian mencari nama teman ini adalah sebagaimana ditunjukkan pada Tabel 5.16. Dokumentasi pengujian sebagaimana pada Gambar 5.15.

Tabel 5.16 Skenario Pengujian Integritas Mencari Nama Teman

Nama Pengujian	Pengujian Integritas Mencari Nama Teman
Kode	PI-05
Use case	UC-04
Tujuan	Memeriksa fungsi mencari nama teman dapat berjalan atau tidak
Kondisi awal	Pengguna berada pada halaman menu sosial
Skenario	1. Pengguna menekan tombol menu sosial. 2. Pengguna mengisi kata kunci pencarian pada kotak tulisan pencarian. 3. Pengguna menekan tombol *search* untuk melakukan pencarian sesuai kata kunci.

Masukan	Kata kunci pencarian
Keluaran yang diharapkan	Pada kotak hasil pencarian tertampilkan nama-nama teman sesuai kata kunci pencarian.
Hasil pengujian	Berhasil



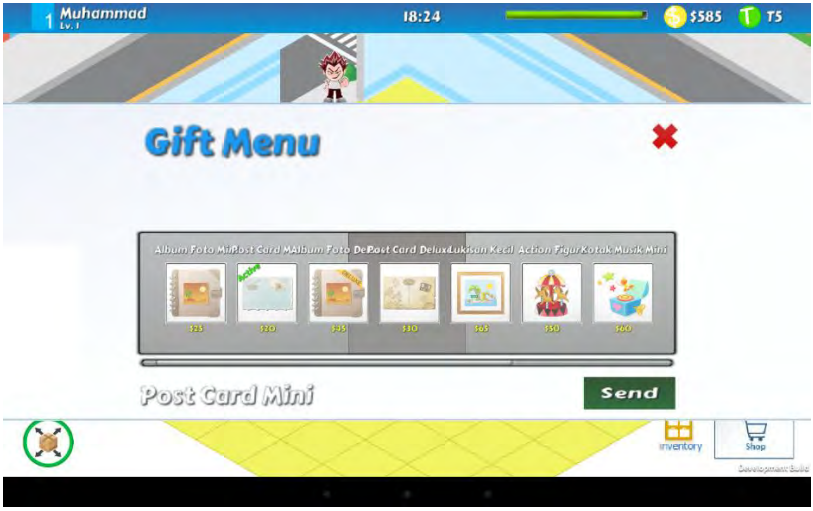
Gambar 5.15 Dokumentasi Pengujian PI-05

5.2.2.5. Pengujian Berinteraksi dengan Teman

Berikut ini merpakan pembahasan pengujian integritas pada kasus penggunaan berinteraksi dengan teman(UC-05). Skenario pengujian pada pengujian berinteraksi dengan teman ini terdapat tiga jenis. Skenario pertama adalah mengirimkan hadiah, sebagaimana dijelaskan pada Tabel 5.17. Dokumentasi pengujian dari skenario ini sebagaimana pada Gambar 5.16. Skenario kedua adalah mengunjungi pujasera teman, sebagaimana dijelaskan pada Tabel 5.18. Dokumentasi pengujian dari skenario ini sebagaimana pada Gambar 5.17. Skenario ketiga adalah mengirim pesan, sebagaimana dijelaskan pada Tabel 5.19. Dokumentasi pengujian dari skenario ini sebagaimana pada Gambar 5.18.

Tabel 5.17 Skenario Pengujian Integritas Mengirimkan Hadiah

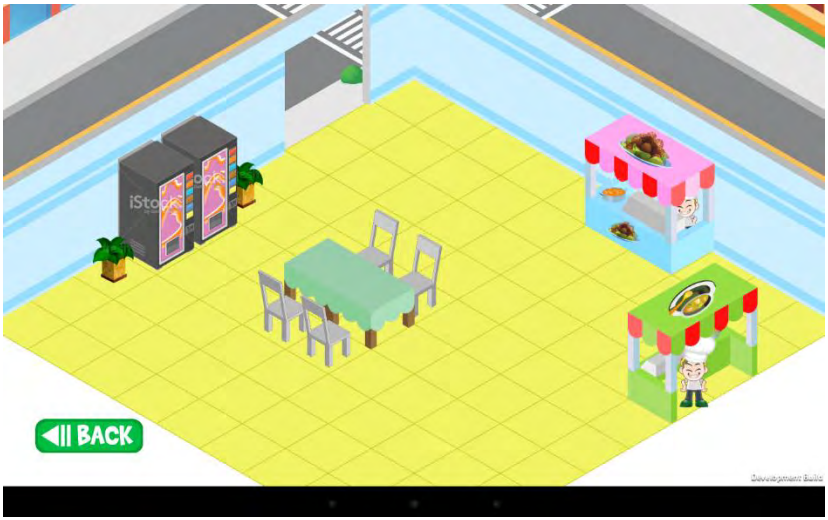
Nama Pengujian	Pengujian Integritas Mengirimkan Hadiah
Kode	PI-06
Use case	UC-05
Tujuan	Memeriksa fungsi mengirim hadiah dapat berjalan atau tidak
Kondisi awal	Pengguna memilih salah satu teman dari hasil pencarian
Skenario	1. Pengguna memilih salah satu teman dari hasil pencarian 2. Pengguna menekan tombol *send gift* untuk membuka halaman menu *gift*. 3. Pengguna memilih item hadiah yang akan dikirim. 4. Pengguna menekan tombol *send* untuk mengirim hadiah.
Masukan	-
Keluaran yang diharapkan	Teman mendapat notifikasi di akun Facebook bahwa telah dikirimi hadiah oleh pengguna.
Hasil pengujian	Berhasil



Gambar 5.16 Dokumentasi Pengujian PI-06

Tabel 5.18 Skenario Pengujian Integritas Mengunjungi Pujasera Teman

Nama Pengujian	Pengujian Integritas Mengunjungi Pujasera Teman
Kode	PI-07
Use case	UC-05
Tujuan	Memeriksa fungsi mengunjungi pujasera teman dapat berjalan atau tidak
Kondisi awal	Pengguna memilih salah satu teman dari hasil pencarian
Skenario	5. Pengguna memilih salah satu teman dari hasil pencarian 6. Pengguna menekan tombol *visit* untuk membuka halaman kunjungan teman
Masukan	-
Keluaran yang diharapkan	Sistem menampilkan halaman berisi tata letak objek-objek pujasera milik teman
Hasil pengujian	Berhasil



Gambar 5.17 Dokumentasi Pengujian PI-07

Tabel 5.19 Skenario Pengujian Integritas Mengirim Pesan

Nama Pengujian	Pengujian Integritas Mengirim Pesan
Kode	PI-08
Use case	UC-05
Tujuan	Memeriksa fungsi mengirim pesan kepada teman dapat berjalan atau tidak
Kondisi awal	Pengguna memilih salah satu teman dari hasil pencarian
Skenario	7. Pengguna memilih salah satu teman dari hasil pencarian. 8. Pengguna menekan tombol *message* untuk membuka halaman menu *message*. 9. Pengguna menulis pesan yang akan dikirim. 10. Pengguna menekan tombol *message* pada layar untuk mengirim pesan.
Masukan	Kalimat pesan
Keluaran yang diharapkan	Teman mendapat notifikasi di akun Facebook bahwa telah dikirimi pesan oleh pengguna.

Hasil pengujian	Berhasil

Gambar 5.18 Dokumentasi Pengujian PI-08

5.2.2.6. Pengujian Berbagi Capaian

Berikut ini merupakan penjelasan pengujian integritas pada kasus penggunaan berbagi capaian (UC-06). Skenario pengujian pada pengujian berbagi capaian ini adalah sebagaimana ditunjukkan pada Tabel 5.20. Dokumentasi pengujian sebagaimana pada Gambar 5.19.

Tabel 5.20 Skenario Pengujian Integritas Berbagi Capaian

Nama Pengujian	Pengujian Integritas Berbagi Capaian
Kode	PI-09
Use case	UC-06
Tujuan	Memeriksa fungsi berbagi capaian dapat berjalan atau tidak
Kondisi awal	Pengguna akan melakukan capaian berupa penambahan luas ruangan

Skenario	11. Pengguna menekan tombol *upgrade* untuk menambah luas ruangan
Masukan	-
Keluaran yang diharapkan	Akun Facebook pengguna membagikan aktivitas pencapaian yang dilakukan pengguna dalam permainan.
Hasil pengujian	Berhasil

Gambar 5.19 Dokumentasi Pengujian PI-09

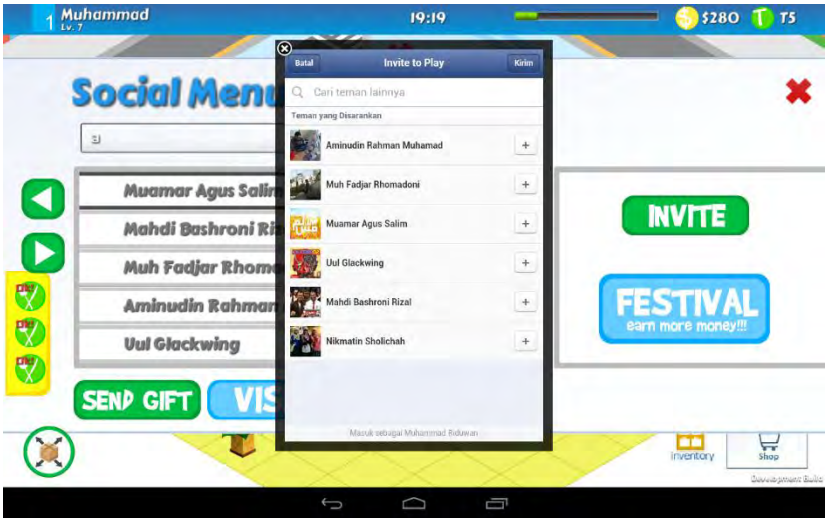
5.2.2.7. Pengujian Mengundang Teman

Berikut ini merupakan pembahasan pengujian integritas pada kasus penggunaan mengundang teman (UC-07). Skenario pengujian pada pengujian mengundang teman ini adalah sebagaimana ditunjukkan pada Tabel 5.21. Dokumentasi pengujian sebagaimana pada Gambar 5.20.

Tabel 5.21 Skenario Pengujian Integritas Mengundang Teman

Nama Pengujian	Pengujian Integritas Mengundang Teman
Kode	PI-10
Use case	UC-07
Tujuan	Memeriksa fungsi mengundang teman dapat berjalan atau tidak
Kondisi awal	Pengguna berada pada halaman menu sosial

Skenario	12. Pengguna menekan tombol *invite* untuk mengundang teman melalui Facebook. 13. Pengguna memilih teman yang akan diundang. 14. Pengguna menekan tombol *done*.
Masukan	-
Keluaran yang diharapkan	Akun Facebook teman mendapat notifikasi pengundangan dari pengguna
Hasil pengujian	Berhasil



Gambar 5.20 Dokumentasi Pengujian PI-10

5.3. Evaluasi Pengujian

Berdasarkan hasil pengujian fungsionalitas dan pengujian integritas, dilakukan evaluasi terhadap hasil pengujian tersebut sebagai berikut.

5.3.1. Evaluasi Pengujian Fungsionalitas

Pengujian fungsionalitas yang telah dilakukan memberikan hasil yang sesuai dengan skenario yang telah direncanakan. Evaluasi

pengujian pada masing-masing fungsionalitas dijelaskan sebagai berikut.

1. Kasus penggunaan UC-01 *login* dengan Facebook, yang diuji dengan pengujian fungsionalitas modular telah sesuai dengan yang diharapkan. Kondisi ini diperlihatkan pada pengujian PF-01 yang menunjukkan bahwa proses *login* menggunakan Facebook berjalan dengan benar pada berkas pengembangan yang terpisah.
2. Kasus penggunaan UC-02 masuk halaman utama, yang diuji dengan pengujian fungsionalitas modular untuk menampilkan penampang *isometric* telah sesuai dengan yang diharapkan. Kondisi ini diperlihatkan pada pengujian PF-02 yang menunjukkan bahwa proses membuat penampang *isometric* berjalan dengan benar pada berkas pengembangan yang terpisah..
3. Kasus penggunaan UC-03 mengolah objek pujasera, yang diuji dengan pengujian fungsionalitas modular telah sesuai dengan yang diharapkan. Kondisi ini diperlihatkan pada pengujian PF-03, PF-04, dan PF-05 yang menunjukkan bahwa proses mengolah objek pujasera berjalan dengan benar pada berkas pengembangan yang terpisah..
4. Kasus penggunaan UC-04 mencari nama teman, yang diuji dengan pengujian fungsionalitas modular untuk menampilkan daftar nama teman hasil pencarian telah sesuai dengan yang diharapkan. Kondisi ini diperlihatkan pada pengujian PF-06 yang menunjukkan bahwa proses menampilkan nama teman berjalan dengan benar pada berkas pengembangan yang terpisah..
5. Kasus penggunaan UC-05 berinteraksi dengan teman, yang diuji dengan pengujian fungsionalitas modular telah sesuai dengan yang diharapkan. Kondisi ini diperlihatkan pada pengujian PF-07, PF-08, dan PF-09 yang menunjukkan bahwa proses interaksi dengan teman seperti mengirim hadiah, mengunjungi, dan mengirim pesan berjalan dengan benar pada berkas pengembangan yang terpisah..

6. Kasus penggunaan UC-06 berbagi capaian, yang diuji dengan pengujian fungsionalitas modular telah sesuai dengan yang diharapkan. Kondisi ini diperlihatkan pada pengujian PF-10 yang menunjukkan bahwa proses berbagi capaian berjalan dengan benar pada berkas pengembangan yang terpisah..
7. Kasus penggunaan UC-07 mengundang teman, yang diuji dengan pengujian fungsionalitas modular telah sesuai dengan yang diharapkan. Kondisi ini diperlihatkan pada pengujian PF-11 yang menunjukkan bahwa proses mengundang teman berjalan dengan benar pada berkas pengembangan yang terpisah..

Dari evaluasi di atas didapatkan bahwa semua kebutuhan fungsional pada kasus penggunaan yang diuji untuk mengetahui kebenaran dan modularitas modul yang dikerjakan berhasil dilakukan. Tabel 5.22 menunjukkan rangkuman hasil pengujian berdasarkan kasus penggunaannya.

Tabel 5.22 Rekapitulasi Hasil Pengujian Fungsionalitas

Kasus Penggunaan	Pengujian Fungsionalitas	Hasil Pengujian
UC-01	PF-01	Berhasil
UC-02	PF-02	Berhasil
UC-03	PF-03	Berhasil
	PF-04	Berhasil
	PF-05	Berhasil
UC-04	PF-06	Berhasil
UC-05	PF-07	Berhasil
	PF-08	Berhasil
	PF-09	Berhasil
UC-06	PF-10	Berhasil
UC-07	PF-11	Berhasil

5.3.2. Evaluasi Pengujian Integritas

Pengujian integritas yang telah dilakukan memberikan hasil yang sesuai dengan skenario yang telah direncanakan. Evaluasi pengujian pada masing-masing fungsionalitas pada kasus penggunaan dijelaskan sebagai berikut.

1. Kasus penggunaan UC-01 *login* dengan Facebook, yang diuji dengan pengujian integritas telah sesuai dengan yang diharapkan. Kondisi ini diperlihatkan pada pengujian PI-01 dan PI-02 yang menunjukkan bahwa proses *login* menggunakan Facebook untuk pemain baru dan pemain terdaftar berjalan dengan benar pada berkas pengembangan yang terintegrasi.
2. Kasus penggunaan UC-02 menampilkan halaman utama, yang diuji dengan pengujian integritas telah sesuai dengan yang diharapkan. Kondisi ini diperlihatkan dengan pengujian PI-03 yang menunjukkan bahwa proses menampilkan penampang *isometric* dan objek pujasera pemain pada halaman permainan utama berjalan dengan benar pada berkas pengembangan yang terintegrasi.
3. Kasus penggunaan UC-03 mengolah objek pujasera, yang diuji dengan pengujian integritas telah sesuai dengan yang diharapkan. Kondisi ini diperlihatkan dengan pengujian PI-04 yang menunjukkan bahwa proses mengolah objek pujasera berjalan dengan benar pada berkas pengembangan yang terintegrasi.
4. Kasus penggunaan UC-04 mencari nama teman, yang diuji dengan pengujian integritas telah sesuai dengan yang diharapkan. Kondisi ini diperlihatkan dengan pengujian PI-05 yang menunjukkan bahwa proses mencari nama teman dan menampilkan hasilnya dapat berjalan dengan benar pada berkas pengembangan yang terintegrasi.
5. Kasus penggunaan UC-05 berinteraksi dengan teman, yang diuji dengan pengujian integritas telah sesuai dengan yang diharapkan. Kondisi ini diperlihatkan dengan pengujian PI-06, PI-07, dan PI-08 yang menunjukkan bahwa proses interaksi seperti mengirim hadiah, mengunjungi teman, dan

mengirim pesan dapat berjalan dengan benar pada berkas pengembangan yang terintegrasi.

6. Kasus penggunaan UC-06 berbagi capaian, yang diuji dengan pengujian integritas telah sesuai dengan yang diharapkan. Kondisi ini diperlihatkan dengan pengujian PI-09 yang menunjukkan bahwa proses berbagi capaian dapat berjalan dengan benar pada berkas pengembangan yang terintegrasi.
7. Kasus penggunaan UC-07 mengundang teman, yang diuji dengan pengujian integritas telah sesuai dengan yang diharapkan. Kondisi ini diperlihatkan dengan pengujian PI-10 yang menunjukkan bahwa proses mengundang teman dapat berjalan dengan benar pada berkas pengembangan yang terintegrasi.

Dari evaluasi di atas didapatkan bahwa semua kebutuhan fungsional pada kasus penggunaan yang diuji untuk mengetahui kebenaran dan integritas modul yang dikerjakan berhasil dilakukan. Tabel 5.23 menunjukkan rangkuman hasil pengujian berdasarkan kasus penggunaannya.

Tabel 5.23 Rekapitulasi Hasil Pengujian Integritas

Kasus Penggunaan	Pengujian Fungsionalitas	Hasil Pengujian
UC-01	PI-01	Berhasil
	PI-02	Berhasil
UC-02	PI-03	Berhasil
UC-03	PI-04	Berhasil
UC-04	PI-05	Berhasil
UC-05	PI-06	Berhasil
	PI-07	Berhasil
	PI-08	Berhasil
UC-06	PI-09	Berhasil
UC-07	PI-10	Berhasil

BAB VI
PENUTUP

Pada bab ini dijelaskan kesimpulan yang dapat diambil dalam pengerjaan Tugas Akhir dan saran untuk pengembangan lebih lanjut dari modul editor ruangan dan fitur sosial pada pada *game* Food Merchant Saga ini.

6.1. Kesimpulan

Dalam proses pengerjaan Tugas Akhir dari tahap pendahuluan, kajian pustaka, analisis, perancangan, implementasi dan pengujian modul editor ruangan dan fitur sosial pada *game* Food Merchant Saga diperoleh kesimpulan sebagai berikut.

1. Tugas akhir yang dikerjakan dapat menghasilkan modul untuk editor ruangan dan fitur sosial yang menjadi salah satu kebutuhan fungsionalitas pada *game* Food Merchant Saga.
2. Modul editor ruangan dan fitur sosial pada *game* Food Merchant Saga dapat mengimplementasikan manajemen objek-objek ruangan pujasera dalam kerangka pengembangan *game* berbasis objek pada kakas Unity.
3. Modul editor ruangan dan fitur sosial dapat menerapkan antarmuka berbasis *tile* dalam bentuk tampilan *isometric* pada *game* Food Merchant Saga.
4. Modul editor ruangan dan fitur sosial pada *game* Food Merchant Saga dapat melakukan pengelolaan ruangan pujasera yang berbentuk penampang *isometric* dengan fungsi-fungsi utama seperti memindah letak objek, mengubah arah objek, menambah dan menghapus objek, serta penambahan luas pujasera.
5. Modul editor ruangan dan fitur sosial pada *game* Food Merchant Saga dapat menjalankan proses fitur sosial yang dibutuhkan pada *game* Food Merchant Saga dengan menerapkan layanan API Facebook dengan modul bantu Unity Facebook SDK.

6. Modul editor ruangan dan fitur sosial dapat diimplementasikan secara terpisah pada berkas pengembangan Unity yang mandiri berdasarkan hasil pengujian fungsionalitas.
7. Modul editor ruangan dan fitur sosial dapat diimplementasikan secara terintegrasi dengan modul lainnya pada berkas pengembangan Unity *game* Food Merchant Saga berdasarkan hasil pengujian integritas.

6.2. Saran

Berikut ini merupakan saran mengenai pengembangan lebih lanjut pada modul editor ruangan dan fitur sosial pada *game* Food merchant Saga berdasarkan hasil rancangan, implementasi dan uji coba yang telah dilakukan.

1. Menyempurnakan proses alur main pada fitur pengelolaan objek pada ruangan pujasera yang berbasis *isometric* sehingga lebih intuitif dan mudah dioperasikan oleh pengguna.
2. Meningkatkan rasa sosial yang lebih pada *game* Food Merchant Saga dengan mempermudah alur pengoperasian fitur sosial agar mudah digunakan oleh pengguna.
3. Meningkatkan integrasi sosial pada sosial media Facebook dengan memperbanyak aktivitas-aktvitas sosial yang dapat dilakukan pemain dengan menerapkan lebih jauh penggunaan fitur Open Graph yang terdapat pada layanan API Faebook.
4. Meningkatkan desain tampilan antarmuka yang lebih menarik dan memberikan suasana permainan yang menyenangkan bagi pengguna *game* Food Merchant Saga.

DAFTAR PUSTAKA

[1] M. Prensky, Digital *Game*-Based Learning, Oakcrest Avenue: Paragon House, 2012.

[2] L. Williams, "Testing Overview and Black-Box Testing Techniques," 2006. [Online]. Available: http://agile.csc.ncsu.edu/SEMaterials/BlackBox.pdf. [Accessed 12 Maret 2014].

[3] N. Lovell, "What is a social *game*?," *Game*s Brief, 2011. [Online]. Available: http://www.*game*sbrief.com/2011/01/what-is-a-social-*game*/. [Accessed 20 Februari 2014].

[4] V. Ryan, "First Angle - Orthographic Projection," www.technologystudent.com, 2010. [Online]. Available: http://www.technologystudent.com/designpro/ortho1.htm. [Accessed 16 Februari 2014].

[5] A. Keane, "Isometric Projection," University of Limerick, 1998. [Online]. Available: http://www.ul.ie/~rynnet/keanea/isometri.htm. [Accessed 20 Februari 2014].

[6] Facebook, "The Graph API," Facebook Inc., April 2014. [Online]. Available: https://developers.facebook.com/docs/graph-api. [Accessed 10 Mei 2014].

[7] D. Orenstein, "QuickStudy: Application Programming Interface (API)," International Data Group, 10 Januari 2000. [Online]. Available: http://www.computerworld.com/s/article/43487/Application_Programming_Interface. [Accessed 17 April 2014].

[8] Facebook, "Facebook SDK for Unity," Facebook Inc., April 2014. [Online]. Available:

https://developers.facebook.com/docs/unity. [Accessed 10 Mei 2014].

[9] M. Karch, "What is Google Android?," about.com, 2014. [Online]. Available: http://google.about.com/od/socialtoolsfromgoogle/p/android_what_is.htm. [Accessed 20 Februari 2014].

[10] P. Tony, "Tile Based *Game*s - Why Tiles?," www.tonypa.pri.ee, 2005. [Online]. Available: http://www.tonypa.pri.ee/tbw/tut00.html. [Accessed 20 Februari 2014].

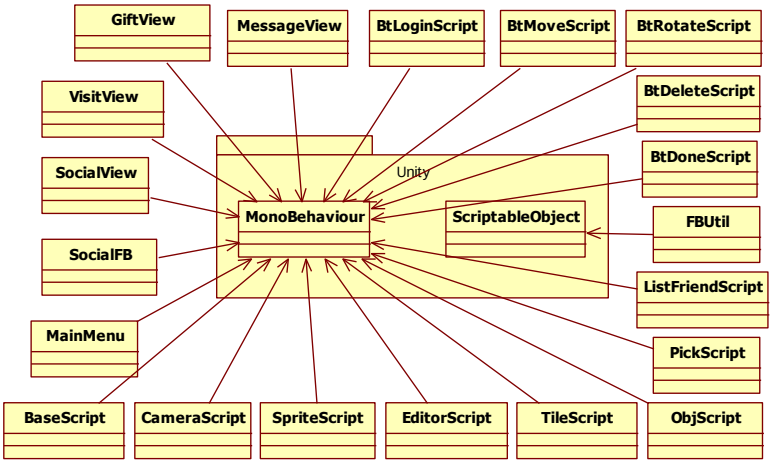
[11] Unity, "Unity - Create *Game*s with Unity," Unity3D, 2013. [Online]. Available: http://unity3d.com/pages/create-*game*s. [Accessed 16 Februari 2014].

BIODATA PENULIS

Penulis lahir di Surabaya, 11 Agustus 1992. Penulis menempuh pendidikan di SD Negeri Gundih 1 Surabaya, SMP Muhammadiyah 2 Surabaya, dan SMA Negeri 5 Surabaya. Penulis melanjutkan pendidikan sarjana di Jurusan Teknik Informatika Institut Teknologi Sepuluh Nopember Surabaya. Dalam menyelesaikan pendidikan S1, penulis mengambil bidang minat Rekayasa Perangkat Lunak (*Sofware Engineering*). Selama kuliah penulis aktif dalam berbagai kegiatan dan organisasi baik itu akademik maupun nonakademik. Pada dunia akademik, penulis aktif sebagai asisten dosen mata kuliah Desain Web (PIKTI) dan Pemrograman Web (PIKTI).

Pada dunia nonakademik, penulis diberi amanah sebagai Ketua Departemen Syiar Keluarga Muslim Informatika (KMI) pada periode kepengurusan 2012-2013. Penulis juga aktif mengikuti kegiatan di luar organisasi kampus dengan menjadi Ketua Divisi Media Yayasan Uswah Student Center Surabaya. Penulis memiliki ketertarikan pada pengembangan *game*, desain web, dan desain grafis. Penulis dapat dihubungi melalui alamat email muamaragussalim@gmail.com.

LAMPIRAN A – DIAGRAM KELAS



Gambar 8.1 Diagram Kelas Modul Editor Ruangan dan Fitur Sosial

LAMPIRAN B – KODE SUMBER

```
public class BaseScript : MonoBehaviour {

    public double[,] arrayX;
    public double[,] arrayY;
    private GameObject baseTile;
    public GameObject baseObject;
    private GameObject mainCamera;
    public Transform tilePrefab;
    public Transform wallPrefab1;
    public Transform wallPrefab2;
    public Transform boxPrefab;
    public bool upgradeFlag;
    public int curTile;
    public bool isPickObj;
    public Transform objOnPicked;
    public int commandCode;
    public float maxPanX = 1.78f;
    public float maxPanY = 2.06f;
    public bool isPanActive = true;
    public bool isDecormode = false;
    GameObject textStatus;

    public List<FMSObject> UsedFMS
    {
        get { return Managers.Data.ListUsedFMS; }
        set { Managers.Data.ListUsedFMS = value; }
    }

    private int[,] box = {
        {1, 1, 2, 3, 4, 4},
        {2, 3, 1, 3, 0, 3}
    };


    void Start () {
        upgradeFlag = false;
        isPickObj = false;
        baseTile = GameObject.Find ("1BaseTile");
        textStatus = GameObject.Find("Status");
        mainCamera =
GameObject.FindWithTag("MainCamera");
        TileInstantiate ();
```

```
        DrawUsedFMSObject();
    }

    void Update () {
        if (upgradeFlag)
        {
            LevelUpgrade();

Managers.Debug.print(StateManager.currentState.ToStrin
g());
        }
    }

    void LevelUpgrade () {
        int prevCurTile = curTile;
        curTile = prevCurTile + 2;

        Managers.Debug.print(prevCurTile + " " +
curTile);

        for (int i = 0; i<curTile; i++)
        {
            if(i<prevCurTile)
            {
                for (int j = prevCurTile; j<curTile;
j++)
                {
                    var tileInstant =
Instantiate(tilePrefab) as Transform;
                    tileInstant.position = new
Vector3((float)arrayX[i,j], (float)arrayY[i,j],
baseTile.transform.position.z);
                    tileInstant.parent =
baseTile.transform;
                    TileScript tilePos =
tileInstant.GetComponent<TileScript>();
                    tilePos.posX = i;
                    tilePos.posY = j;


                    if(i == 0)
                    {
```

```
                        var wallInstant =
Instantiate(wallPrefab2) as Transform;
                        wallInstant.position = new
Vector3((float)arrayX[i,j], (float)arrayY[i,j],
baseTile.transform.position.z);
                        wallInstant.parent =
baseTile.transform;
                        TileScript tilePos2 =
wallInstant.GetComponent<TileScript>();
                        tilePos2.posX = i;
                        tilePos2.posY = j;

wallInstant.GetChild(0).GetComponent<SpriteRenderer>()
.sortingOrder = ((int)tilePos2.posY * 10) +
(int)tilePos2.posX;
                    }
                }
            }

            else
            {
                for (int j = 0; j<curTile; j++)
                {
                    var tileInstant =
Instantiate(tilePrefab) as Transform;
                    tileInstant.position = new
Vector3((float)arrayX[i,j], (float)arrayY[i,j],
baseTile.transform.position.z);
                    tileInstant.parent =
baseTile.transform;
                    TileScript tilePos =
tileInstant.GetComponent<TileScript>();
                    tilePos.posX = i;
                    tilePos.posY = j;


                    if(j == 0)
                    {
                        var wallInstant =
Instantiate(wallPrefab1) as Transform;
                        wallInstant.position = new
Vector3((float)arrayX[i,j], (float)arrayY[i,j],
baseTile.transform.position.z);
```

```
                        wallInstant.parent =
baseTile.transform;
                        TileScript tilePos1 =
wallInstant.GetComponent<TileScript>();
                        tilePos1.posX = i;
                        tilePos1.posY = j;

wallInstant.GetChild(0).GetComponent<SpriteRenderer>()
.sortingOrder = ((int)tilePos1.posY * 10) +
(int)tilePos1.posX;
                    }
                }
            }

        }

        maxPanX = maxPanX + (float)(0.89 * 2);
        maxPanY = maxPanY + (float)(0.515 * 2);

        upgradeFlag = false;
    }

    void TileInstantiate (){
        arrayX = new double[100,100];
        arrayY = new double[100,100];

        for(int i = 0; i<100; i++)
        {
            for(int j = 0; j<100; j++)
            {
                arrayX[i,j] = (0.89 * j) - (0.89 * i);
                arrayY[i,j] = 3 - (0.515 * j) - (0.515
* i);
            }
        }

        if(Managers.Data.PlayerData.curTile != null)
        {
            curTile =
Managers.Data.PlayerData.curTile;
        }
        else
        {
            curTile = 10;
```

```
        }
        for (int i = 0; i<curTile; i++)
        {
            for (int j = 0; j<curTile; j++)
            {
                if(i == 0)
                {
                    var wallInstant =
Instantiate(wallPrefab2) as Transform;
                    wallInstant.position = new
Vector3((float)arrayX[i,j], (float)arrayY[i,j],
baseTile.transform.position.z);
                    wallInstant.parent =
baseTile.transform;
                    TileScript tilePos2 =
wallInstant.GetComponent<TileScript>();
                    tilePos2.posX = i;
                    tilePos2.posY = j;

wallInstant.GetChild(0).GetComponent<SpriteRenderer>()
.sortingOrder = ((int)tilePos2.posY * 10) +
(int)tilePos2.posX;
                }

                if(j == 0)
                {
                    var wallInstant =
Instantiate(wallPrefab1) as Transform;
                    wallInstant.position = new
Vector3((float)arrayX[i,j], (float)arrayY[i,j],
baseTile.transform.position.z);
                    wallInstant.parent =
baseTile.transform;
                    TileScript tilePos1 =
wallInstant.GetComponent<TileScript>();
                    tilePos1.posX = i;
                    tilePos1.posY = j;

wallInstant.GetChild(0).GetComponent<SpriteRenderer>()
.sortingOrder = ((int)tilePos1.posY * 10) +
(int)tilePos1.posX;

                    if(i>0 && i<3)
                    {
```

```
wallInstant.GetChild(0).GetComponent<SpriteRenderer>()
.sprite =
wallInstant.GetChild(0).GetComponent<SpriteScript>().s
prite1;
                    }
                }
                var tileInstant =
Instantiate(tilePrefab) as Transform;
                tileInstant.position = new
Vector3((float)arrayX[i,j], (float)arrayY[i,j],
baseTile.transform.position.z);
                tileInstant.parent =
baseTile.transform;
                TileScript tilePos =
tileInstant.GetComponent<TileScript>();
                tilePos.posX = i;
                tilePos.posY = j;
                tilePos.gridPosition = new
GridPosition(i, j);
            }
        }
    }

    private GameObject clone;
    public void DrawUsedFMSObject()
    {
        Managers.Debug.print("Jumlah fmsObject: " +
UsedFMS.Count.ToString());
        int posX = 0;
        int posY = 0;
        Property prop;
        foreach (FMSObject k in UsedFMS)
        {
            clone =
(GameObject)Instantiate(Loader.Load(k.PrefabName));
            clone.transform.position =
GridPositionToVector3(k.GridPosition);

            posX = k.GridPosition.XPos;
            posY = k.GridPosition.YPos;

            CheckRotation(k);
```

```
            clone.transform.parent =
baseObject.transform;

clone.GetComponentInChildren<SpriteRenderer>().sorting
Order = (posY * 10) + posX;

clone.GetComponentInChildren<ObjScript>().posX = posX;

clone.GetComponentInChildren<ObjScript>().posY = posY;
            FMSView kv =
clone.GetComponent<FMSView>();
            kv.index = UsedFMS.IndexOf(k);
        }
    }

    public void CheckRotation(FMSObject k)
    {
        if(k is Property)
        {
            Property cloneProp = (Property)k;
            if(cloneProp.GridRotation ==
GridRotation.Rotate90)
            {

clone.GetComponentInChildren<ObjScript>().curSprite =
1;
            }
            else if(cloneProp.GridRotation ==
GridRotation.Rotate180)
            {

clone.GetComponentInChildren<ObjScript>().curSprite =
2;
            }
            else if(cloneProp.GridRotation ==
GridRotation.Rotate270)
            {

clone.GetComponentInChildren<ObjScript>().curSprite =
3;
            }
        }
        else if(k is Stall)
        {
```

```
            Stall cloneStall = (Stall)k;
            if(cloneStall.GridRotation ==
GridRotation.Rotate90)
            {

clone.GetComponentInChildren<ObjScript>().curSprite =
1;
            }
            else if(cloneStall.GridRotation ==
GridRotation.Rotate180)
            {

clone.GetComponentInChildren<ObjScript>().curSprite =
2;
            }
            else if(cloneStall.GridRotation ==
GridRotation.Rotate270)
            {

clone.GetComponentInChildren<ObjScript>().curSprite =
3;
            }
        }
        else if(k is Accessory)
        {
            Accessory cloneAccessory = (Accessory)k;
            if(cloneAccessory.GridRotation ==
GridRotation.Rotate90)
            {

clone.GetComponentInChildren<ObjScript>().curSprite =
1;
            }
            else if(cloneAccessory.GridRotation ==
GridRotation.Rotate180)
            {

clone.GetComponentInChildren<ObjScript>().curSprite =
2;
            }
            else if(cloneAccessory.GridRotation ==
GridRotation.Rotate270)
            {
```

```
clone.GetComponentInChildren<ObjScript>().curSprite =
3;
            }
        }
    }

    public void DrawUsedFMSObject(FMSObject k)
    {
        GridPosition pos =
gameObject.GetComponent<ObjectHandler>
().gridPosition;
        k.GridPosition = pos;
        UsedFMS.Add(k);
        ReloadView();
    }

    private Vector3 GridPositionToVector3(GridPosition
position) {
        return new
Vector3((float)arrayX[position.XPos, position.YPos],
(float)arrayY[position.XPos, position.YPos],
baseObject.transform.position.z);
    }

    public void DeleteFMS(int index) {
        Managers.Data.RemoveUsedFMS(index);
        ReloadView();
    }

    public void ReloadView(){
        RemoveFMSObject();
        DrawUsedFMSObject();
    }

    public void RemoveFMSObject()
    {
        foreach (Transform child in
baseObject.transform)
            Destroy(child.gameObject);
    }
}
```

Kode Sumber 9.1 Kelas BaseScript untuk Membuat Penampang *Isometric* dan Menggambar Objek

```
public class ObjScript : MonoBehaviour {

    GameObject scriptObj;
    public int maxSprite;
    public int curSprite;
    public float posX;
    public float posY;
    public bool isFestival;

    // Use this for initialization
    void Start () {
        scriptObj = GameObject.Find("Scripts");
        GetComponent<TapGesture>().StateChanged +=
HandleStateChanged;
    }

    GameObject popUp;
    void HandleStateChanged(object sender,
TouchScript.Events.GestureStateChangeEventArgs e)
    {
        if(e.State == Gesture.GestureState.Recognized)
        {
            if (StateManager.currentState ==
GameState.DEKOR)
            {

if(!scriptObj.GetComponent<BaseScript>().isPickObj &&
scriptObj.GetComponent<BaseScript>().isDecormode)
                {

transform.GetComponent<SpriteRenderer>().color = new
Color(1f, 1f, 1f, 0.5f);

scriptObj.GetComponent<BaseScript>().isPickObj = true;

scriptObj.GetComponent<BaseScript>().objOnPicked =
transform.parent;
                }
            }
            if (StateManager.currentState ==
GameState.GAMEPLAY)
            {
```

```
                FMSView vdata =
transform.parent.GetComponent<FMSView>();

                if (vdata.DataFMS is Stall)
                {
                    Stall k = (Stall)vdata.DataFMS;
                    if (!k.isUpgrade && !isFestival)
                    {
                        popUp =
(GameObject)Instantiate(Loader.Load("PopUpMenu"));
                        popUp.transform.position =
transform.parent.position;
                        popUp.transform.parent =
transform.parent.parent;

popUp.GetComponent<PopUpMenuView>().icon =
transform.parent.GetComponent<FMSView>().rotate1;

popUp.GetComponent<PopUpMenuView>().kedaiIndex =
transform.parent.GetComponent<FMSView>().index;
                        HOTween.To(popUp.transform,
0.3f, new TweenParms().Prop("localScale", new
Vector3(2f, 2f, 1f)).Ease(EaseType.EaseInElastic));
                        StateManager.currentState =
GameState.POPUPMENU;
                        Managers.Debug.print("click at
" + transform.parent.name);
                    }
                }
            }
        }
    }

}
```

Kode Sumber 9.2 Kelas ObjScript untuk Menangani *Behaviour* Objek Ketika Ditekan

```
public class CameraScript : MonoBehaviour {

    public float speed = 0.05f;
    GameObject background;
    GameObject foreground;
    float xMax = 19.0f;
```

```
    float yDownMax = -24.5f;
    float yUpMax = 12.0f;

    void Start () {
        background = GameObject.Find("0 -
Background");
        foreground = GameObject.Find ("2 -
Foreground");
    }

    // Update is called once per frame
    void Update () {
        if (StateManager.currentState ==
GameState.GAMEPLAY || StateManager.currentState ==
GameState.DEKOR || StateManager.currentState ==
GameState.POPUPMENU)
        {
            if (Input.touchCount > 0 &&
Input.GetTouch(0).phase == TouchPhase.Moved)
            {
                Vector2 touchDeltaPosition =
Input.GetTouch(0).deltaPosition;

                if(transform.position.x < xMax &&
transform.position.x > -xMax && transform.position.y <
yUpMax && transform.position.y > yDownMax)
                {
                transform.Translate(-
touchDeltaPosition.x * speed / 3, -
touchDeltaPosition.y * speed / 3, 0);

                background.transform.position = new
Vector3(transform.position.x, transform.position.y,
background.transform.position.z);
                foreground.transform.position = new
Vector3(transform.position.x, transform.position.y,
foreground.transform.position.z);

                }

                if(transform.position.x > xMax)
                {
                    float diffX = transform.position.x
- xMax;
```

```
                    transform.Translate(-diffX * 2, 0,
0);
                }
                else if(transform.position.x < -xMax)
                {
                    float diffX =
(transform.position.x * -1.0f) - xMax;
                    transform.Translate(diffX * 2, 0,
0);
                }
                else if(transform.position.y > yUpMax)
                {
                    float diffY = transform.position.y
- yUpMax;
                    transform.Translate(0, -diffY * 2
, 0);
                }
                else if(transform.position.y <
yDownMax)
                {
                    float diffY =
(transform.position.y * -1.0f) - (yDownMax * -1.0f);
                    transform.Translate(0, diffY * 2,
0);
                }

                background.transform.position = new
Vector3(transform.position.x, transform.position.y,
background.transform.position.z);
                foreground.transform.position = new
Vector3(transform.position.x, transform.position.y,
foreground.transform.position.z);

            }
        }
    }

}
```

Kode Sumber 9.3 Kelas CameraScript untuk Menangani *Behaviour* Objek Kamera Ketika Ditekan

```
public class BtMoveScript : MonoBehaviour {
```

```
    public GameObject scriptObj;

    // Use this for initialization
    void Start () {
        UIEventListener.Get(gameObject).onClick +=
this.HandleStateChanged;
    }

    private void HandleStateChanged(GameObject go)
    {
        print("Hay");
        if (StateManager.currentState ==
GameState.DEKOR)
        {
            if
(scriptObj.GetComponent<BaseScript>().isPickObj &&
scriptObj.GetComponent<BaseScript>().objOnPicked)
            {
                print("Move Command Active");

scriptObj.GetComponent<BaseScript>().commandCode = 1;
            }
        }
    }
}
```

Kode Sumber 9.4 Kelas BtMoveScript untuk Menangani *Event Handling* pada Tombol Pindah Posisi

```
public class BtRotateScript : MonoBehaviour {

    GameObject scriptObj;
    Transform objOnRotated;
    Transform objOnReset;
    int maxRotate;
    int curSprite;
    string objName;

    void Start () {
        scriptObj = GameObject.Find ("Scripts");
        UIEventListener.Get(gameObject).onClick +=
this.HandleStateChanged;
    }
```

```
    private void HandleStateChanged(GameObject go)
    {

        if
(scriptObj.GetComponent<BaseScript>().isPickObj &&
scriptObj.GetComponent<BaseScript>().objOnPicked)
        {

scriptObj.GetComponent<BaseScript>().commandCode = 2;
        }

        if
(scriptObj.GetComponent<BaseScript>().commandCode ==
2)
        {
            objOnRotated =
scriptObj.GetComponent<BaseScript>().objOnPicked.GetCh
ild(0);
            objOnReset =
scriptObj.GetComponent<BaseScript>().objOnPicked;
            maxRotate =
objOnRotated.GetComponent<ObjScript>().maxSprite;
            if (maxRotate > 1)
            {
                objName = objOnRotated.name;
                curSprite =
objOnRotated.GetComponent<ObjScript>().curSprite;

                if (curSprite + 1 < maxRotate)
                {

objOnRotated.GetComponent<SpriteRenderer>().sprite =
objOnRotated.GetComponent<SpriteScript>().GetSprite(cu
rSprite + 1);

objOnRotated.GetComponent<ObjScript>().curSprite =
curSprite + 1;
                }
                else
                {

objOnRotated.GetComponent<SpriteRenderer>().sprite =
objOnRotated.GetComponent<SpriteScript>().GetSprite(0)
;
```

```
objOnRotated.GetComponent<ObjScript>().curSprite = 0;
                }


if(objOnReset.GetComponent<FMSView>().DataFMS.Id ==
"acc_06" ||

objOnReset.GetComponent<FMSView>().DataFMS.Id ==
"acc_07" ||

objOnReset.GetComponent<FMSView>().DataFMS.Id ==
"acc_08")
                {

objOnRotated.GetComponent<PolygonCollider2D>().CreateP
rimitive(6);
                }

            }
        }
    }

}
```

Kode Sumber 9.5 Kelas BtRotateScript untuk Menangani *Event Handling* pada Tombol Rotasi

```
public class BtDeleteScript : MonoBehaviour {

    public GameObject inventory;
    public BaseScript baseScript;
    Transform objOnDeleted;
    FMSObject fmsObject;

    public List<FMSObject> Inventory
    {
        get { return Managers.Data.ListInventory; }
        set { Managers.Data.ListInventory = value; }
    }

    void Start () {
```

```
        UIEventListener.Get(gameObject).onClick +=
this.HandleStateChanged;
    }

    private void HandleStateChanged(GameObject go)
    {
        if (StateManager.currentState ==
GameState.DEKOR)
        {
            if (baseScript.isPickObj &&
baseScript.objOnPicked)
            {
                baseScript.commandCode = 3;

                objOnDeleted = baseScript.objOnPicked;
                Destroy(objOnDeleted.gameObject);
                baseScript.isPickObj = false;

Inventory.Add(objOnDeleted.GetComponent<FMSView>().Dat
aFMS);

baseScript.DeleteFMS(objOnDeleted.GetComponent<FMSView
>().index);

Managers.Debug.print(objOnDeleted.GetComponent<FMSView
>().index.ToString());
                Managers.Debug.print("Deleting: " +
objOnDeleted.GetComponent<FMSView>().DataFMS.Name);


inventory.GetComponent<InventoryView>().UpdateListInve
ntory();
            }

        }
    }

}
```

Kode Sumber 9.6 Kelas BtDeleteScript untuk Menangani *Event Handling* pada Tombol Hapus Objek

```
public class SocialFB : MonoBehaviour {
```

```
    private int flagFunc = 0;
    private string searchQuery;
    private int flagAction;

    public void CallFBInvite(string title, string
message)
    {
        string FriendSelectorTitle = title;
        string FriendSelectorMessage = message;
        string FriendSelectorFilters =
"[\"all\",\"app_users\",\"app_non_users\"]";

        FB.AppRequest(
            message: FriendSelectorMessage,
            filters: FriendSelectorFilters,
            title: FriendSelectorTitle,
            callback: Callback
            );
    }

    public void CallFBMessage(string title, string
message, string idFriend)
    {
        string tellMessage = message;
        string tellToId = idFriend;

        var queryTell = new Dictionary<string,
string>();
        queryTell["profile"] = tellToId;
        queryTell["message"] = tellMessage;
        queryTell ["tags"] = tellToId;
        flagAction = 1;
        FB.API ("/me/foodmerchantsaga:tell",
Facebook.HttpMethod.POST, Callback, queryTell);
    }

    public void CallFBGift(string idFriend)
    {
        string giftMessage = "I've sent you a gift!";
        string giftToId = idFriend;

        var queryGift = new Dictionary<string,
string>();
```

```
        queryGift["gift"] =
"http://samples.ogp.me/314820825348714";
        queryGift["message"] = giftMessage;
        queryGift ["tags"] = giftToId;
        flagAction = 2;
        FB.API ("/me/foodmerchantsaga:send",
Facebook.HttpMethod.POST, Callback, queryGift);
    }

    public void CallFBSearch(string query)
    {
        searchQuery = query;


FB.API("/me?fields=id,first_name,friends.fields(name,i
d)", Facebook.HttpMethod.GET, SearchCallback);
    }

    public void CallFBShare(string link, string
linkName, string linkCaption, string linkDescription)
    {
        Managers.Debug.print ("login: " +
FB.IsLoggedIn);

        string FeedLink = link;
        string FeedLinkName = linkName;
        string FeedLinkCaption = linkCaption;
        string FeedLinkDescription = linkDescription;
        string FeedReference = "";

        FB.Feed(
            link: FeedLink,
            linkName: FeedLinkName,
            linkCaption: FeedLinkCaption,
            linkDescription: FeedLinkDescription,
            reference: FeedReference,
            callback: Callback
            );
    }

    public void CallFBVisit(string idFriend)
    {
        string visitMessage = "I've visited your
Pujasera";
```

```
        string visitToId = idFriend;

        var queryVisit = new Dictionary<string,
string>();
        queryVisit["pujasera"] =
"http://samples.ogp.me/314753028688827";
        queryVisit["message"] = visitMessage;
        queryVisit["tags"] = visitToId;
        flagAction = 3;
        FB.API ("/me/foodmerchantsaga:visit",
Facebook.HttpMethod.POST, Callback, queryVisit);
    }

    private void Callback(FBResult result)
    {
        if (result.Error != null)
        {
            Managers.Debug.print("Error Response: \n"
+ result.Error);

            if(flagAction == 1)
            {

gameObject.GetComponent<SocialView>().loadingWindow.Se
tActive(false);
                StateManager.currentState =
GameState.MENU_SOCIAL;
                flagAction = 0;
            }
            else if(flagAction == 2)
            {

gameObject.GetComponent<SocialView>().loadingWindow.Se
tActive(false);
                StateManager.currentState =
GameState.MENU_SOCIAL;
                flagAction = 0;
            }
        }
        else
        {
            Managers.Debug.print("Success Response:
\n" + result.Text);
```

```
            if(flagAction == 1)
            {

gameObject.GetComponent<SocialView>().loadingWindow.Se
tActive(false);
                StateManager.currentState =
GameState.MENU_SOCIAL;
                flagAction = 0;

StateManager.AnimateOut(gameObject.GetComponent<Social
View>().menuMessage);
                gameObject.SetActive(true);
            }
            else if(flagAction == 2)
            {

gameObject.GetComponent<SocialView>().loadingWindow.Se
tActive(false);
                StateManager.currentState =
GameState.MENU_SOCIAL;
                flagAction = 0;

StateManager.AnimateOut(gameObject.GetComponent<Social
View>().menuGift);
                gameObject.SetActive(true);
            }
            else if(flagAction == 3)
            {

            }
        }
    }

    private void LoginCallback(FBResult result)
    {
        if (result.Error != null)
        {
            Managers.Debug.print("Error Response: \n"
+ result.Error);
        }
        else
        {
            Managers.Debug.print("Success Response:
\n" + result.Text);
```

```
        }
    }

    private void SearchCallback(FBResult result)
    {
        if(result.Error != null)
        {
            FB.API("/me?fields=friends.limit(100))",
Facebook.HttpMethod.GET, SearchCallback);
            return;
        }

        Managers.Debug.print("callback: " +
result.Text);
        SocialView.friends =
FBUtil.DeserializeJSONFriends (result.Text);

        transform.GetComponent<SocialView>
().ShowFriends ();

    }
}
```

Kode Sumber 9.7 Kelas SocialFB untuk Melakukan Proses-Proses Sosial

```
public class GiftView : MonoBehaviour {
    public HeadUpDView hud;
    public BaseScript baseScript;
    public ObjectHandler objectHandler;
    public BtDecorScript btDekor;
    public GameObject giftPanel;
    private GameObject giftTemplate;
    private GameObject rootShop;
    private GameObject clone;
    public GameObject giftItem, selectedItemText;
    public GameObject btnSendObj;
    public GameObject btnClose;
    public GameObject objScript;
    public GameObject menuSocial;
    public GameObject loadingWindow;

    private InternalStorageUtil objectStorage;
```

```
    private Vector3 rootPos = new Vector3();
    private Vector3 kPos = new Vector3();

    public List<FMSObject> listOfObject = new
List<FMSObject>();
    public FMSObject selectedItem = new FMSObject();
    public FMSObject selectedItemTemp = new
FMSObject();

    //Data
    public PlayerData PlayerData
    {
        get { return Managers.Data.PlayerData; }
        set { Managers.Data.PlayerData = value; }
    }

    public List<FMSObject> UsedFMS
    {
        get { return Managers.Data.ListUsedFMS; }
        set { Managers.Data.ListUsedFMS = value; }
    }

    float xDif = 1.5f;

    void Start () {
        Init();
        InitGameObject();
        BindDataToGameObject();

    }

    private void Init()
    {
        Managers.Debug.print("masuk init data");
        objectStorage = new
InternalStorageUtil(Storage.FMSOBJECT);
        objScript = GameObject.Find("Scripts");

        DataGenerator dg = new DataGenerator();
        dg.Generate();

        listOfObject =
objectStorage.Load<FMSObject>();
```

```
        print(PlayerData.pujasera);
    }

    private void BindDataToGameObject()
    {
        foreach (FMSObject k in listOfObject) {
            if (k is Gift) {
                Transform gtrans =
giftTemplate.transform;
                kPos = gtrans.position;
                int g = 1;

                clone =
(GameObject)Instantiate(giftItem,
giftTemplate.transform.position, Quaternion.identity);
                clone.transform.parent =
giftPanel.transform;

clone.transform.Find("UISprite").GetComponent<UISprite
>().spriteName = k.Id;
                gtrans.position = new Vector3(kPos.x +
g * xDif, kPos.y, kPos.z);
                g++;

                ItemView iv =
clone.GetComponent<ItemView>();
                iv.itemDeail = k;
                int level = PlayerData.level;
                if (k.CekActive(level))
                {
                    UIEventListener.Get(clone).onClick
+= this.OnClickAction;
                    GameObject activeLable =
clone.transform.FindChild("UILabel - New").gameObject;
                    activeLable.SetActive(true);
                }
            }
        }
    }

    private void OnClickAction(GameObject go)
    {
        selectedItem =
go.GetComponent<ItemView>().itemDeail;
```

```
        UILabel selectTed =
selectedItemText.GetComponent<UILabel>();
        selectTed.text = selectedItem.Name;

        if (selectedItem.Active != true)
        {
            btnSendObj.SetActive(true);
            UIEventListener.Get(btnSendObj).onClick +=
this.OnClickSend;
        }
    }

    private void OnClickSend(GameObject go)
    {
        if(StateManager.currentState ==
GameState.MENU_SOCIAL){
            if (PlayerData.money <
selectedItem.BuyCost)
            {
                Managers.Debug.print("Uang tidak
mencukupi!");
            }
            else
            {
                if( selectedItem is Gift)
                {
                    Managers.Debug.print("Masuk: " +
selectedItem.Name);
                    PlayerData.money -=
selectedItem.BuyCost;

menuSocial.GetComponent<SocialFB>().CallFBGift(menuSoc
ial.GetComponent<SocialView>().selectedFriend.GetCompo
nent<ListFriendScript>().friendId);

                    PlayerData.jmlTransaksi += 1;
                    Managers.Debug.print("Use " +
selectedItem.Name);
                    Managers.Debug.print("Send to " +
menuSocial.GetComponent<SocialView>().selectedFriend.G
etComponent<ListFriendScript>().friendName);
```

```
                        StateManager.currentState =
GameState.LOADING;
                        loadingWindow.SetActive (true);
                    }

                }
            }
        }

        void InitGameObject()
        {
            Managers.Debug.print("masuk init gameobject");
            giftPanel = GameObject.Find("UIGrid - Gift");

            giftTemplate = GameObject.Find("ItemTemplate -
Gift");

            rootShop = GameObject.Find("RootShopGift");
            rootPos = rootShop.transform.position;
            rootPos.y = rootPos.y + 8f;
            btnSendObj.SetActive(false);
            Managers.Debug.print("keluar init
gameobject");

            UIEventListener.Get(GameObject.Find("Button -
CloseGift")).onClick += this.OnClickCloseGift;
        }

        private void OnClickCloseGift(GameObject go)
        {
            if(StateManager.currentState ==
GameState.MENU_SOCIAL)
            {
                StateManager.AnimateOut(gameObject);
                menuSocial.SetActive (true);
            }
        }
}
```

Kode Sumber 9.8 Kelas GiftView untuk Menangani Proses pada Halaman Menu *Gift*

```
public class MessageView : MonoBehaviour {
    public UILabel messageText;
```

```
    public GameObject btnMessage;
    public UILabel sendtoText;
    public GameObject menuSocial;
    public GameObject btnClose;
    public GameObject loadingWindow;

    void Start () {
        UIEventListener.Get(btnClose).onClick +=
this.OnClickCloseMessage;
        UIEventListener.Get(btnMessage).onClick +=
this.OnClickMessage;
    }

    private void OnClickMessage(GameObject go)
    {
        if(StateManager.currentState ==
GameState.MENU_SOCIAL)
        {
            menuSocial.GetComponent<SocialFB>
().CallFBMessage ("Send Message", messageText.text,
menuSocial.GetComponent<SocialView>
().selectedFriend.GetComponent<ListFriendScript>
().friendId);
            messageText.text = "Your Message";

            StateManager.currentState =
GameState.LOADING;
            loadingWindow.SetActive (true);
        }

    }

    private void OnClickCloseMessage(GameObject go)
    {
        if(StateManager.currentState ==
GameState.MENU_SOCIAL)
        {
            StateManager.AnimateOut(gameObject);
            menuSocial.SetActive (true);
        }
    }
}
```

Kode Sumber 9.9 Kelas MessageView untuk Menangani Proses pada Halaman Mengirim Pesan

```
public class VisitView : MonoBehaviour {

    public double[,] arrayX;
    public double[,] arrayY;
    public string dataUrl;
    List<object> listItem = new List<object>();
    List<object> listKedai = new List<object>();
    List<object> listStatus = new List<object>();
    private GameObject clone;
    public GameObject baseObject;
    public GameObject baseTile;
    public GameObject btnBack;
    public GameObject dataCarrier;
    public Transform tilePrefab;
    public Transform wallPrefab1;
    public Transform wallPrefab2;
    public int curTile;
    public GameObject loadingWindow;
    public string strItemFriend, strKedaiFriend,
strStatusFriend;

    void Start () {
        UIEventListener.Get (btnBack).onClick +=
this.OnClickBack;

        dataCarrier = GameObject.Find("DataCarrier");

        Managers.Debug.print ("item: " +
dataCarrier.GetComponent<DataScript>().strItemFriend);
        Managers.Debug.print ("status: " +
dataCarrier.GetComponent<DataScript>().strStatusFriend
);

        TileInstantiate ();
        InitData ();
    }

    private void OnClickBack(GameObject go)
    {
        loadingWindow.SetActive (true);

        Application.LoadLevel("GamePlayTest");
    }
```

```
    void TileInstantiate (){
        arrayX = new double[100,100];
        arrayY = new double[100,100];

        for(int i = 0; i<100; i++)
        {
            for(int j = 0; j<100; j++)
            {
                arrayX[i,j] = (0.89 * j) - (0.89 * i);
                arrayY[i,j] = 3 - (0.515 * j) - (0.515
* i);
            }
        }

        listStatus = (List<object>)Json.Deserialize
(dataCarrier.GetComponent<DataScript>().strStatusFrien
d);
        curTile = Int32.Parse
((string)(((IDictionary)listStatus[0])["Tile"]));
        for (int i = 0; i<curTile; i++)
        {
            for (int j = 0; j<curTile; j++)
            {
                if(i == 0)
                {
                    var wallInstant =
Instantiate(wallPrefab2) as Transform;
                    wallInstant.position = new
Vector3((float)arrayX[i,j], (float)arrayY[i,j],
baseTile.transform.position.z);
                    wallInstant.parent =
baseTile.transform;
                    TileScript tilePos2 =
wallInstant.GetComponent<TileScript>();
                    tilePos2.posX = i;
                    tilePos2.posY = j;

wallInstant.GetChild(0).GetComponent<SpriteRenderer>()
.sortingOrder = ((int)tilePos2.posY * 10) +
(int)tilePos2.posX;
                }

                if(j == 0)
```

```
                {
                    var wallInstant =
Instantiate(wallPrefab1) as Transform;
                    wallInstant.position = new
Vector3((float)arrayX[i,j], (float)arrayY[i,j],
baseTile.transform.position.z);
                    wallInstant.parent =
baseTile.transform;
                    TileScript tilePos1 =
wallInstant.GetComponent<TileScript>();
                    tilePos1.posX = i;
                    tilePos1.posY = j;

wallInstant.GetChild(0).GetComponent<SpriteRenderer>()
.sortingOrder = ((int)tilePos1.posY * 10) +
(int)tilePos1.posX;

                    if(i>0 && i<3)
                    {

wallInstant.GetChild(0).GetComponent<SpriteRenderer>()
.sprite =
wallInstant.GetChild(0).GetComponent<SpriteScript>().s
prite1;
                    }
                }
                var tileInstant =
Instantiate(tilePrefab) as Transform;
                tileInstant.position = new
Vector3((float)arrayX[i,j], (float)arrayY[i,j],
baseTile.transform.position.z);
                tileInstant.parent =
baseTile.transform;
                TileScript tilePos =
tileInstant.GetComponent<TileScript>();
                tilePos.posX = i;
                tilePos.posY = j;
                tilePos.gridPosition = new
GridPosition(i, j);
            }
        }
    }

    private void InitData()
```

```
    {
        try{
            listItem = (List<object>)Json.Deserialize
(dataCarrier.GetComponent<DataScript>().strItemFriend)
;
            listKedai = (List<object>)Json.Deserialize
(dataCarrier.GetComponent<DataScript>().strKedaiFriend
);
        }catch(Exception e){
            Managers.Debug.print(e.Message);
        }

        GenerateObject (listItem);
        GenerateObject (listKedai);

    }

    private void GenerateObject(List<object> listData)
    {
        int posX = 0;
        int posY = 0;
        IDictionary dictData;

        for(int i = 0; i<listData.Count; i++)
        {
            dictData = (IDictionary)listData[i];

            posX =
Int32.Parse((string)dictData["POSISI_X"]);
            posY =
Int32.Parse((string)dictData["POSISI_Y"]);

            clone =
(GameObject)Instantiate(Loader.Load((string)dictData["
PREFAB"]));
            clone.transform.position = new
Vector3((float)arrayX[posX, posY], (float)arrayY[posX,
posY], baseObject.transform.position.z);

            clone.transform.parent =
baseObject.transform;
            clone.GetComponent<FMSView>().enabled =
false;
```

```
clone.GetComponentInChildren<SpriteRenderer>().sorting
Order = (posY * 10) + posX;

clone.GetComponentInChildren<ObjScript>().posX = posX;
clone.GetComponentInChildren<ObjScript>().posY = posY;

            if((string)dictData["ROTASI"] == "1")
            {

clone.GetComponentInChildren<SpriteRenderer>().sprite
=
clone.GetComponentInChildren<SpriteScript>().sprite0;
            }
            else if((string)dictData["ROTASI"] == "2")
            {

clone.GetComponentInChildren<SpriteRenderer>().sprite
=
clone.GetComponentInChildren<SpriteScript>().sprite1;
            }
            else if((string)dictData["ROTASI"] == "3")
            {

clone.GetComponentInChildren<SpriteRenderer>().sprite
=
clone.GetComponentInChildren<SpriteScript>().sprite2;
            }
            else if((string)dictData["ROTASI"] == "4")
            {

clone.GetComponentInChildren<SpriteRenderer>().sprite
=
clone.GetComponentInChildren<SpriteScript>().sprite3;
            }
        }
    }
}
```

Kode Sumber 9.10 Kelas VisitView untuk Menampilkan Objek Pujasera Saat Mengunjungi Teman