Implementasi Modul *House Editor* dan Fitur Sosial pada Aplikasi Permainan Sosial Card Warlock Saga pada Perangkat Android

Nama Mahasiswa	: Zendra Anugerah Andromedha
NRP	: 5110100219
Jurusan	: Teknik Informatika FTIf-ITS
Dosen Pembimbing 1	: Imam Kuswardayan, S.Kom., M.T.
Dosen Pembimbing 2	: Ridho Rahman Hariadi, S.Kom., M.Sc.

ABSTRAK

Card Warlock Saga adalah aplikasi permainan sosial yang juga mengandung unsur RPG, turn-based game dan card game, sehingga bisa dikatakan permainan ini bergenre turn-based collectible card yaitu bertema turn-based yang menggunakan kartu sebagai mekanisme pertarungan permainan. Pemain memiliki satu hero yang dikendalikan dan set kartu sebagai mekanisme pertarungan.

Dikarenakan permainan ini merupakan permainan sosial maka permainan ini dibuat agar antar pemain dapat terjadi interaksi, sehingga diimplementasikanlah fitur-fitur yang mendukung interaksi antar pemain seperti menambah teman, saling bertukar barang dan mengirim pesan. Dan permainan ini akan diintegrasikan dengan jejaring sosial Facebook.

Integrasi antara aplikasi permainan ini dengan jejaring sosial Facebook akan dibuat menggunakan kakas Facebook SDK for Unity dengan menggunakan layanan Facebook API. Dengan adanya integrasi dengan Facebook, diharapkan agar memudahkan interaksi yang terjadi antar pemain.

Kata kunci**: **Android Game-Based, Facebook, Game**, **Social Game, Unity.

Implementation of House Editor and Social Features on Card Warlock Saga Android Social Game-Based

Student Name	: Zendra Anugerah Andromedha
Student ID	: 5110100219
Major	: Teknik Informatika FTIf-ITS
Advisor 1	: Imam Kuswardayan, S.Kom., M.T.
Advisor 2	: Ridho Rahman Hariadi, S.Kom., M.Sc.

ABSTRACT

Card Warlock Saga is a social game application that also contains elements of RPG, turn-based games and card games, so this game's genre could be called turn-based collectible cards that are themed turn-based which use card as a battle mechanism. Players have a hero to control and set of cards as the fight mechanism.

Because of this game is a social game then this game is made so the player can interract with other player, so this game implements some features that support interaction between players like add friends, trading items and send message. And this game will be integrated with social networking site Facebook.

The integration between this game application and social networking site Facebook will developed with Facebook SDK for Unity Tools which use Facebook API service. The integration with Facebook is expected to facilitate the interaction between players.

Keyword**: **Android Game-Based, Facebook, Game**, **Social Game, Unity.

DAFTAR KODE SUMBER

BAB II
DASAR TEORI

Pada bab ini akan dibahas mengenai teori-teori yang menjadi dasar dari pembuatan tugas akhir. Teori-teori tersebut meliputi dasar game sosial, Unity, *Role Playing Game*, dan Facebook SDK for Unity.

2.1. *Game* Sosial

Game sosial adalah salah satu jenis permainan yang melibatkan interaksi antar pengguna atau pemain untuk melakukan aktivitas tertentu dalam *game* serta menggunakan jaringan sosial eksternal untuk memfasilitasi tujuan dari permainan ini [1].

Game sosial pada umumnya bersifat *asynchronous* di mana untuk memainkannya pemain tidak perlu *online* secara bersama–sama dan fitur *multiplayer*.

Fitur-fitur dalam permainan sosial yaitu antara lain adalah sebagai berikut [2].

- *Asynchronous gameplay*. Fitur yang memungkinkan permainan dapat diselesaikan tanpa perlu bantuan pemain lain yang berada dalam jaringan untuk bermain pada waktu yang sama.
- Komunitas. Salah satu fitur pembeda antara permainan sosial daring dan permainan daring lainnya. *Quest* atau objektif dari permainan hanya dapat dilakukan jika pemain membagi permainannya dengan teman-teman mereka (terhubung melalui jaringan sosial seperti Facebook) atau membuat pemain lain bermain sebagai tetangga atau sekutu.
- Tidak ada kondisi menang. Kebanyakan pengembang mengandalkan pengguna permainan mereka untuk sering bermain, biasanya tidak ada kondisi menang. Artinya, permainan tidak pernah berakhir dan tidak ada yang pernah dinyatakan menang. Sebaliknya, permainan memiliki banyak *quest* atau misi untuk diselesaikan.

- Mata uang virtual. Permainan sosial daring menggunakan mata uang virtual, di mana pemain harus membeli mata uang virtual dengan uang asli. Penggunaan mata uang virtual biasanya digunakan untuk membeli *upgrade* yang akan memakan waktu lebih lama jika didapatkan melalui *game achievement*. Dalam banyak kasus, beberapa *upgrade* hanya tersedia melalui mata uang virtual.

2.2. Unity

Unity merupakan sebuah ekosistem pengembangan permainan yang terintegrasi dan kaya akan alat atau perlengkapan yang sangat berguna untuk membangun permainan interaktif seperti pencahayaan, efek khusus, animasi dan mesin fisika. Unity dapat digunakan untuk membangun permainan dengan dukungan grafis tiga dimensi atau dua dimensi. Unity juga dapat digunakan untuk melakukan perubahan maupun menguji secara bersamaan pada permainan yang dibuat, dan ketika sudah siap permainan dapat dipublikasikan ke berbagai macam perangkat yang mendukung seperti Mac, PC, Linux, Windows Store, Windows Phone 8, Android, Blackberry 10, Wii U, PS3, dan Xbox 360 [3].

2.3. *Role Playing Game*

Role Playing Game (RPG) adalah sebuah *game* di mana pemain memainkan satu atau lebih karakter yang didesain oleh pemain dan menuntut mereka melakukan serangkaian misi yang dikelola oleh komputer. Kemenangan dicapai dengan menyelesaikan misi-misi yang tersedia. Pertumbuhan kekuatan dan kemampuan karakter merupakan kunci dalam genre ini. Umumnya tantangan yang muncul berupa pertarungan taktik, eksplorasi, dan *puzzle solving* [4].

2.4. Facebook SDK for Unity

Facebook SDK for Unity adalah *software development kit* yang dibuat oleh Facebook yang memungkinkan untuk mengintegrasikan *game* Unity dengan fitur-fitur sosial yang dimiliki Facebook dengan mudah [5].

2.5. Sistem Operasi Android

Android adalah sistem operasi ponsel berbasis Linux yang dikembangkan oleh Google. Android berbeda dengan sistem operasi ponsel lain karena Google secara aktif mengembangkan *platform* tetapi memberikan secara gratis untuk produsen *hardware* dan operator telepon yang ingin menggunakan Android pada perangkat mereka.

Android dikembangkan untuk perangkat bergerak layar sentuh. Walaupun dapat menggunakan *trackball* untuk beberapa navigasi, tapi hampir semuanya dilakukan melalui sentuhan. Android juga mendukung *multi-touch gestures* seperti *pinch-to-zoom* di versi 2.1 (Eclair) ke atas [6].

BAB III
ANALISIS DAN PERANCANGAN SISTEM

Bab ini membahas tahap analisis permasalahan dan perancangan dari sistem yang akan dibangun. Analisis permasalahan membahas permasalahan yang 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 berbasis komponen. Perancangan direpresentasikan dengan diagram UML (*Unified Modelling Language*).

3.1. Analisis

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

3.1.1. Analisis Permasalahan

Permasalahan utama yang diangkat dalam pembuatan tugas akhir ini adalah bagaimana menentukan fitur-fitur sosial yang sesuai dengan *game* Card Warlock Saga. Permasalahan kedua yaitu bagaimana mengintegrasikan *game* ini dengan Facebook.

3.1.2. Deskripsi Umum Sistem

Card Warlock Saga adalah aplikasi permainan *turn-based collectible card* yaitu bergenre RPG yang menggunakan kartu sebagai mekanisme pertarungan permainan. Pemain memiliki satu *hero* yang disebut *magician* dan set kartu sebagai mekanisme pertarungan.

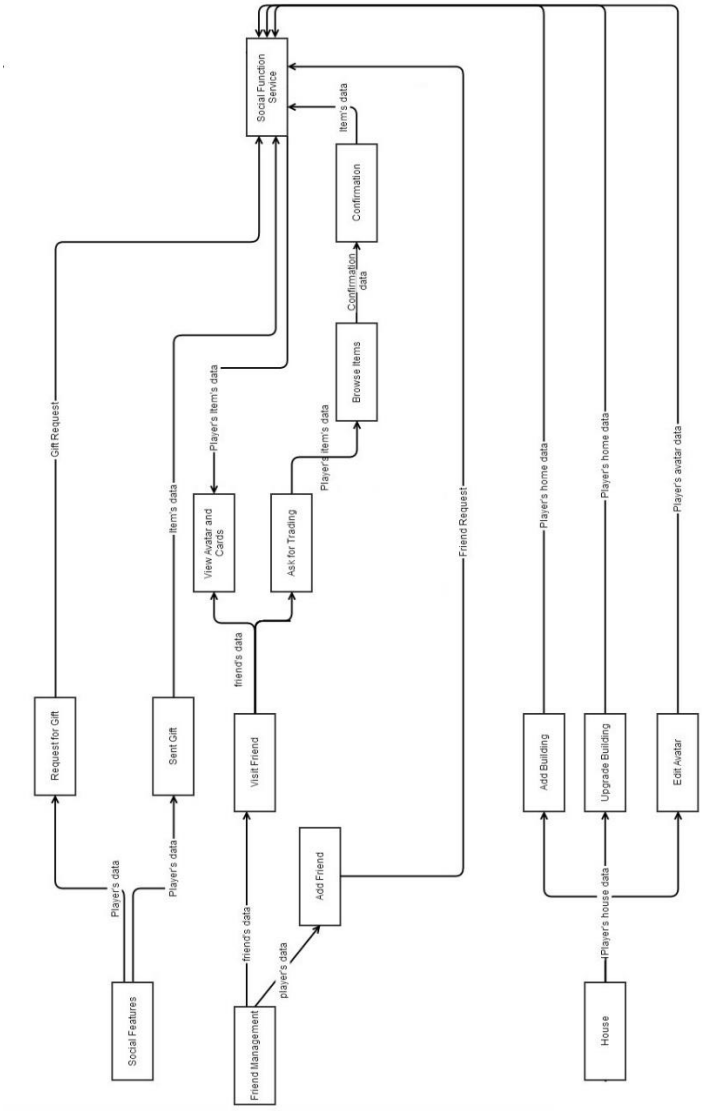
Pemain dapat mengunjungi teman pada menu *visit friend*, di mana pemain dapat mengakses beberapa fitur seperti sosial seperti *trading card*, *vs battle, request gift*, *invitation request*, dan *send gift*.

Game ini gabungan dari strategi, RPG, dan *card battle*, sehingga dengan penggabungan tersebut, memberikan suatu sensasi baru dalam permainan, di mana selain mengeksplor dunia *game*, pemain juga harus mengatur strategi kartu apa saja yang dimasukkan ke dalam *deck* dan dipakai untuk bertarung.

Seperti unsur *game* RPG lainnya, *game* ini tidak memiliki kondisi tamat. Dan agar tidak bosan, *game* ini dibekali dengan *avatar*, kartu, *dungeon*, dan *monster* yang beragam. Selain itu ditambahkan unsur sosial agar pemain dapat berinteraksi dengan pemain lain seperti *trading card*, *battle*, *invitation request*, serta *send gift*.

Untuk mempermudah antar pemain melakukan interaksi, maka dibangun fitur *house editor*, di mana pemain dapat melakukan banyak hal seperti mengatur *avatar*-nya sesuai keinginan dan mengatur kartu-kartu yang akan diperlihatkan saat pemain lain mengunjungi *house* pemain tersebut.

Untuk keseluruhan sistem akan dilampirkan pada *block diagram* di lampiran A pada Gambar 7.1. Dan untuk perancangan sistem fitur sosial digambarkan dengan *block diagram* seperti pada Gambar 3.1.



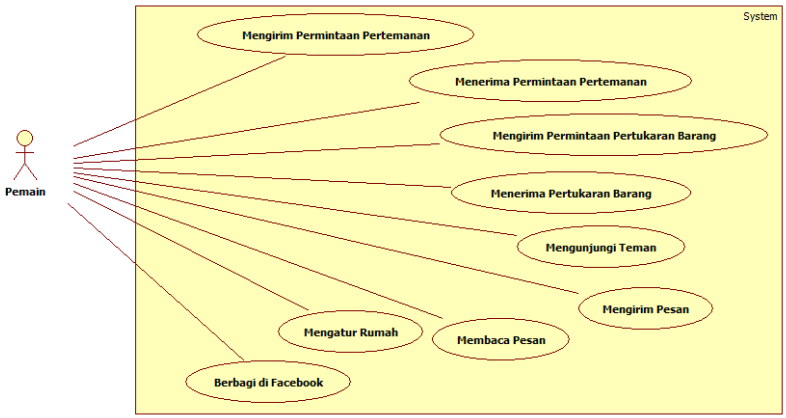
Gambar 3.1. Diagram Blok Fitur Sosial Permainan Card Warlock Saga

3.1.3. Aktor

Aktor mendefinisikan entitas-entitas yang terlibat dan berinteraksi langsung dengan sistem. Entitas ini bisa berupa manusia maupun sistem atau perangkat lunak yang lain. Aktor yang terdapat pada sistem ini hanya memiliki sebuah peran yaitu sebagai pengguna. Pengguna perangkat ini adalah pemain Card Warlock Saga.

3.1.4. Kasus Penggunaan

Berdasarkan analisis spesifikasi kebutuhan fungsional dan analisis aktor dari sistem dibuat kasus penggunaan sistem. Kasus-kasus penggunaan dalam sistem ini akan dijelaskan secara rinci pada subbab ini. Kasus penggunaan digambarkan dalam sebuah diagram kasus penggunaan. Diagram kasus penggunaan dapat dilihat pada Gambar 3.2. Tabel 3.1 berisi penjelasan dari setiap kasus penggunaan.



Gambar 3.2. Diagram Kasus Penggunaan

Tabel 3.1. Daftar Kode Diagram Kasus Penggunaan

Kode Kasus Penggunaan	Nama
UC-0001	Mengirim permintaan pertemanan

UC-0002	Menerima permintaan pertemanan
UC-0003	Mengatur rumah
UC-0004	Berbagi di Facebook
UC-0005	Mengunjungi teman
UC-0006	Mengirim permintaan pertukaran barang
UC-0007	Menerima pertukaran barang
UC-0008	Mengirim pesan
UC-0009	Membaca pesan

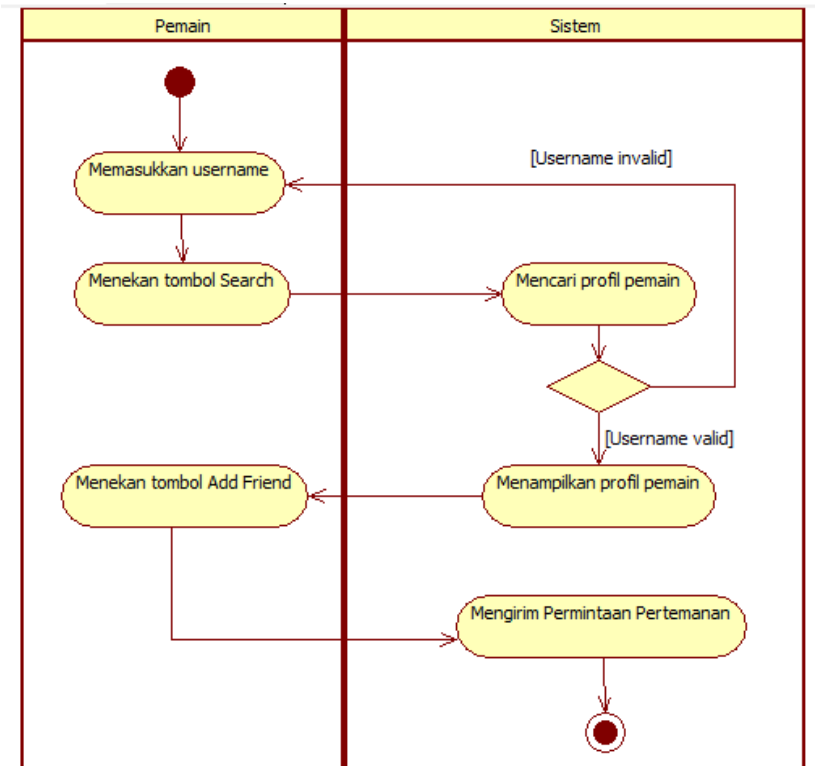
3.1.4.1. Mengirim Permintaaan Pertemanan

Pada kasus penggunaan ini, sistem menerima *input* berupa *username* pemain. Setelah itu, sistem akan menampilkan hasil berupa keterangan pemain tersebut. Setelah mengetahui bahwa hasil yang ditampilkan seperti yang dimaksud, sistem menerima *input* berupa perintah untuk mengeksekusi kode program untuk mengirim permintaan pertemanan. Spesifikasi kasus penggunaan ini dapat dilihat pada Tabel 3.2. Diagram aktivitas dari kasus penggunaan ini bisa dilihat pada Gambar 3.3.

Tabel 3.2. Spesifikasi Kasus Penggunaan Mengirim Permintaan Pertemanan

Nama	Mengirim Permintaan Pertemanan
Kode	UC-0001
Deskripsi	Implementasi fitur mengirim permintaan pertemanan di mana pemain dapat mengirim permintaan untuk dijadikan teman kepada pemain lain
Tipe	Fungsional

Pemicu	Pengguna menekan tombol *Add Friend* untuk memulai proses pada sistem.
Aktor	Pengguna
Kondisi Awal	Pengguna telah memasukkan *username* pemain yang dimaksud
Aliran: **- Kejadian Normal**	1. Pengguna memasukkan *username* pemain yang ingin dikirimi permintaan pertemanan. 2. Sistem menampilkan keterangan pemain. 3. Pengguna menekan tombol *Add Friend*. 4. Sistem mengirim permintaan pertemanan kepada pemain tersebut.
-Kejadian Alternatif	1. Pengguna memasukkan *username* pemain yang tidak valid 2. Kembali ke alur normal 1
Kondisi Akhir	Sistem mengirim permintaan pertemanan.



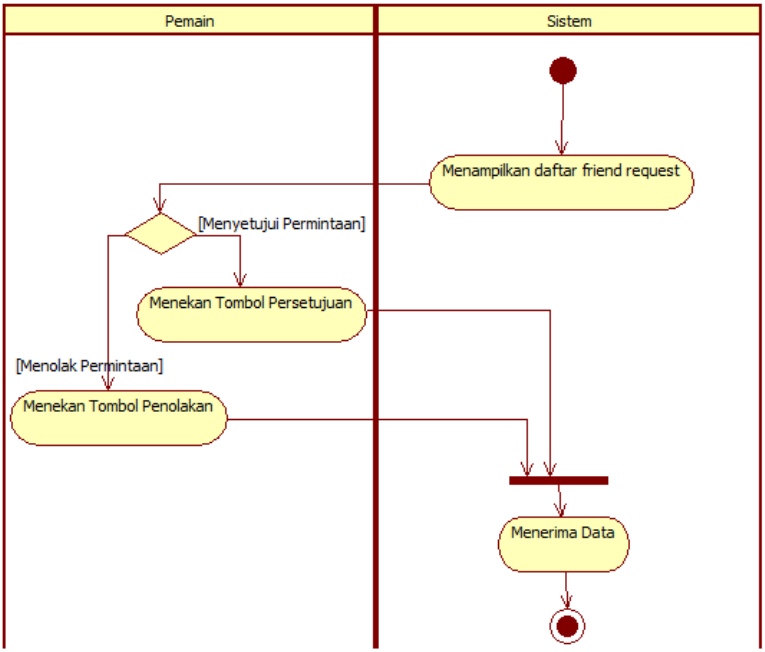
Gambar 3.3. Diagram Aktivitas Mengirim Permintaan Pertemanan

3.1.4.2. Menerima Permintaaan Pertemanan

Pada kasus penggunaan ini, sistem akan menampilkan tampilan daftar permintaan pertemanan, lalu sistem akan menerima *input* dari pemain berupa klik pada tombol persetujuan untuk mengeksekusi kode program untuk menyetujui permintaan pertemanan. Spesifikasi kasus penggunaan ini dapat dilihat pada Tabel 3.3. Diagram aktivitas dari kasus penggunaan ini bisa dilihat pada Gambar 3.4.

Tabel 3.3. Spesifikasi Kasus Penggunaan Menerima Permintaan Pertemanan

Nama	Menerima Permintaan Pertemanan
Kode	UC-0002
Deskripsi	Implementasi fitur menerima permintaan pertemanan di mana pemain dapat menyetujui maupun menolak permintaan untuk dijadikan teman oleh pemain lain
Tipe	Fungsional
Pemicu	Pengguna masuk ke menu *View Friend Request*
Aktor	Pengguna
Kondisi Awal	Pengguna telah masuk menu *View Friend Request*
Aliran: - **Kejadian Normal**	1. Sistem menampilkan daftar permintaan pertemanan 2. Pemain menyetujui permintaan pertemanan dengan menekan tombol persetujuan 3. Sistem menerima data masukan.
- **Kejadian Alternatif**	1. Pengguna menolak permintaan pertemanan dengan menekan tombol penolakan 2. Kembali ke alur normal 3
Kondisi Akhir	Sistem menerima data masukan



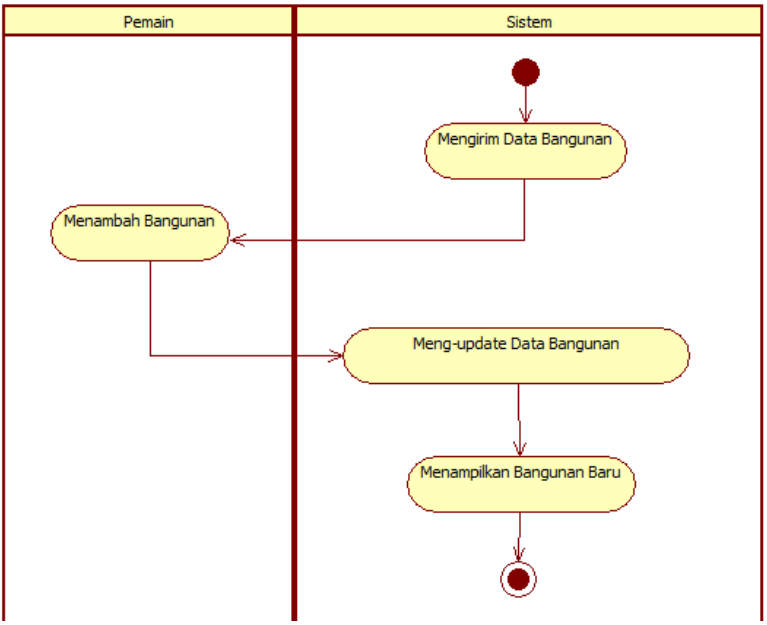
Gambar 3.4. Diagram Aktivitas Menerima Permintaan Pertemanan

3.1.4.3. Mengatur Rumah

Pada kasus penggunaan ini, pemain dapat menambah bangunan yang disediakan oleh sistem, dan dapat menaikkan level bangunan tersebut dengan cara meng-klik bangunan tersebut untuk mengambil *item* dan secara otomatis menaikkan *experience* bangunan tersebut. Sistem menerima masukan berupa klik pada tombol *Add Building* dan bangunan itu sendiri. Spesifikasi kasus penggunaan ini dapat dilihat pada Tabel 3.4. Diagram aktivitas dari kasus penggunaan ini bisa dilihat pada Gambar 3.5.

Tabel 3.4 Spesifikasi Kasus Penggunaan Mengatur Rumah

Nama	Mengatur rumah
Kode	UC-0003
Deskripsi	Pemain dapat menambah bangunan yang disediakan oleh sistem dan bangunan tersebut akan menghasilkan *item* yang dapat diambil oleh pemain
Tipe	Fungsional
Pemicu	Pengguna menekan tombol *Add Building* dan bangunan itu sendiri
Aktor	Pengguna
Kondisi Awal	Pengguna masuk ke menu *House Editor*
Aliran: **-Kejadian Normal**	1. Pengguna menekan tombol *Add Building* 2. Sistem menampilkan bangunan yang diinginkan pemain
-Kejadian Alternatif	Tidak ada
Kondisi Akhir	Sistem menampilkan bangunan baru
Kebutuhan Khusus	Tidak ada



Gambar 3.5. Diagram Aktivitas Mengatur Rumah

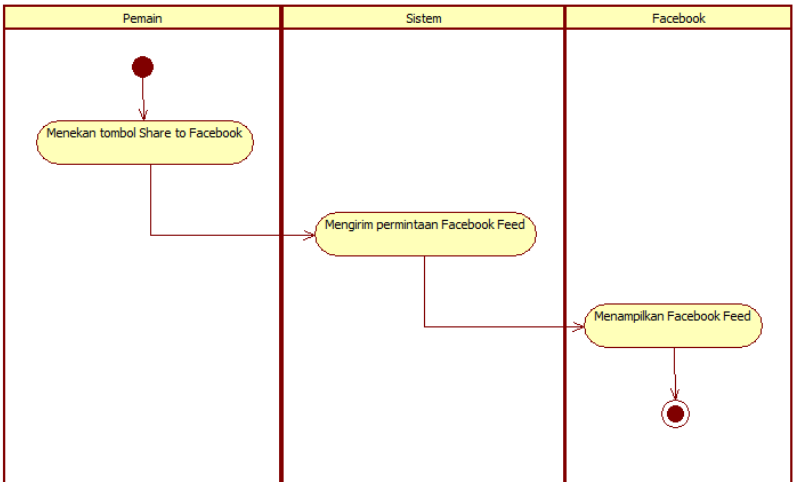
3.1.4.4. Berbagi di Facebook

Pada kasus penggunaan ini, pemain dapat berbagi tentang game Card Warlock Saga melalui *Facebook Feed*. Sistem menerima masukan berupa klik pada tombol *Share to Facebook*. Spesifikasi kasus penggunaan ini dapat dilihat pada Tabel 3.5. Diagram aktivitas dari kasus penggunaan ini bisa dilihat pada Gambar 3.6.

Tabel 3.5. Spesifikasi Kasus Penggunaan Berbagi di Facebook

Nama	Berbagi di Facebook
Kode	UC-0004

Deskripsi	Pemain dapat berbagi tentang game Card Warlock Saga melalui *Facebook Feed*
Tipe	Fungsional
Pemicu	Pengguna menekan tombol *Share to Facebook*
Aktor	Pengguna
Kondisi Awal	Pengguna sudah login melalui Facebook
Aliran: **-Kejadian Normal**	1. Pengguna menekan tombol *Share to* Facebook 2. Sistem mengirim permintaan ke Facebook 3. Facebook menampilkan status kepada pengguna lain
-Kejadian Alternatif	Tidak ada
Kondisi Akhir	Facebook menampilkan status
Kebutuhan Khusus	Tidak ada



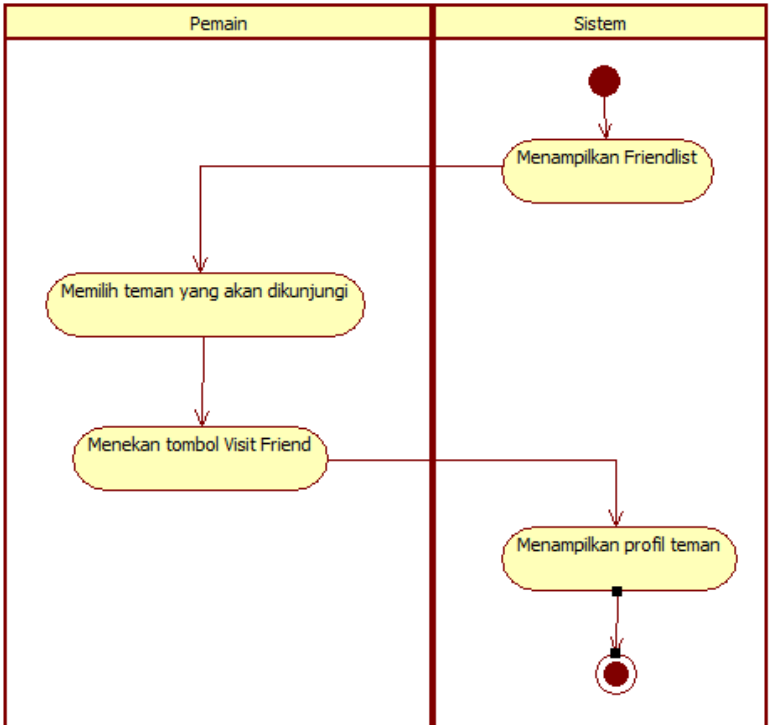
Gambar 3.6. Diagram Aktivitas Berbagi di Facebook

3.1.4.5. Mengunjungi Teman

Pada kasus penggunaan ini, sistem menerima *input* berupa klik pada tombol *Visit Friend*. Setelah itu, sistem akan menampilkan hasil berupa tampilan profil pemain, di antaranya *avatar* dan kartu yang dimiliki pemain tersebut. Spesifikasi kasus penggunaan ini dapat dilihat pada Tabel 3.6. Diagram aktivitas dari kasus penggunaan ini bisa dilihat pada Gambar 3.7.

Tabel 3.6. Spesifikasi Kasus Penggunaan Mengunjungi Teman

Nama	Mengunjungi Teman
Kode	UC-0005
Deskripsi	Implementasi fitur mengunjungi teman, di mana pemain atau aktor dapat mengunjungi dan melihat profil temannya.
Tipe	Fungsional
Pemicu	Pengguna menekan tombol *Visit Friend* untuk memulai proses pada sistem.
Aktor	Pengguna
Kondisi Awal	Pengguna memiliki teman yang akan dikunjungi
Aliran: - **Kejadian Normal**	1. Sistem menampilkan *Friendlist* pemain 2. Pengguna menekan tombol *Visit Friend* pada nama teman yang akan dikunjungi. 3. Sistem menampilkan profil teman tersebut termasuk *avatar* dan keterangan profil.
-Kejadian Alternatif	-
Kondisi Akhir	Sistem menampilkan profil teman.



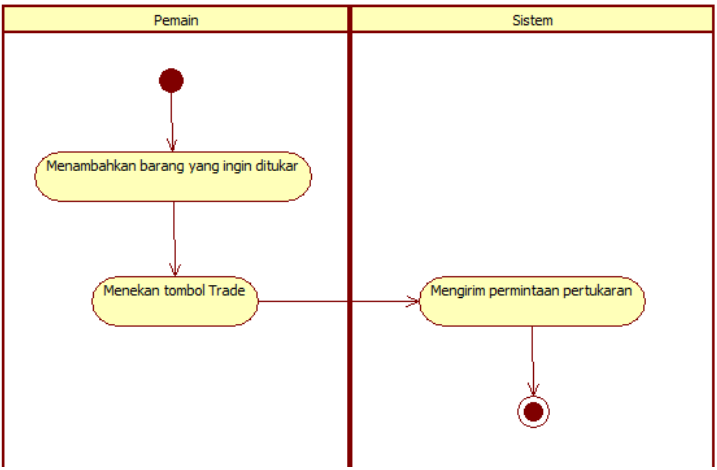
Gambar 3.7. Diagram Aktivitas Mengunjungi Teman

3.1.4.6. Mengirim Permintaan Pertukaran Barang

Pada kasus penggunaan ini, pemain dapat menawarkan atau mengirim permintaan pertukaran barang. Sistem menerima *input* berupa gambar barang yang akan ditukar dan klik pada tombol *Trade*. Setelah itu, sistem akan mengirim permintaan pertukaran barang kepada pemain yang dimaksud. Spesifikasi kasus penggunaan ini dapat dilihat pada Tabel 3.7 . Diagram aktivitas dari kasus penggunaan ini bisa dilihat pada Gambar 3.8.

Tabel 3.7. Spesifikasi Kasus Penggunaan Tukar Barang

Nama	Tukar Barang
Kode	UC-0006
Deskripsi	Implementasi fitur di mana pemain dapat mengirim permintaan pertukaran barang kepada pemain lain
Tipe	Fungsional
Pemicu	Pengguna menekan tombol *Trade*
Aktor	Pengguna
Kondisi Awal	Pengguna masuk ke profil teman untuk melihat barang yang ingin ditukar
Aliran: **- Kejadian Normal**	1. Pengguna menambahkan barang yang ingin ditukar 2. Pengguna menekan tombol *Trade*. 3. Sistem mengirim permintaan pertukaran.
-Kejadian Alternatif	Tidak ada
Kondisi Akhir	Sistem mengirim permintaan pertukaran



Gambar 3.8. Diagram Aktivitas Mengirim Permintaan Pertukaran

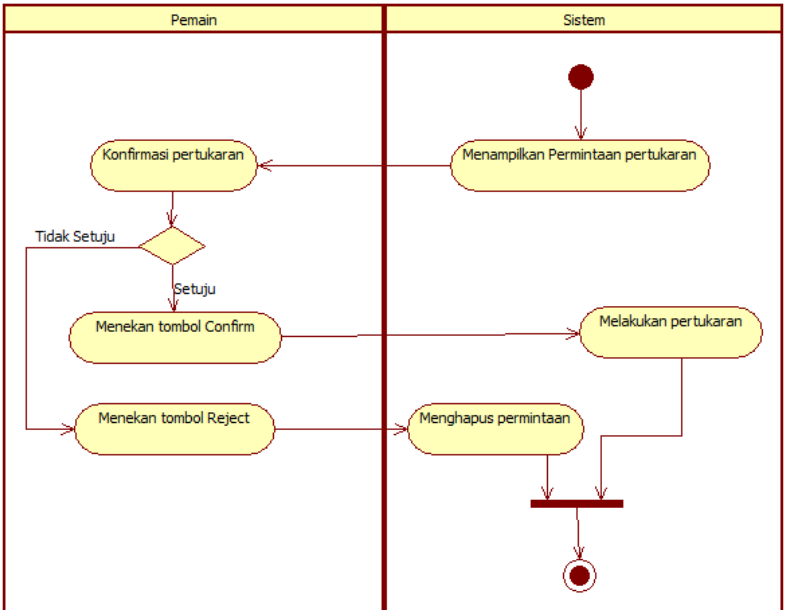
3.1.4.7. Menerima Pertukaran Barang

Pada kasus penggunaan ini, pemain dapat menerima atau menolak permintaan pertukaran barang yang dikirimkan oleh pemain lain. Sistem akan menampilkan daftar permintaan pertukaran. Sistem menerima *input* berupa klik pada tombol persetujuan atau penolakan pada permintaan pertukaran yang dimaksud. Setelah itu, sistem akan mengirim barang pada masing-masing pemain. Spesifikasi kasus penggunaan ini dapat dilihat pada Tabel 3.8. Diagram aktivitas dari kasus penggunaan ini bisa dilihat pada Gambar 3.9.

Tabel 3.8. Spesifikasi Kasus Penggunaan Menerima Pertukaran Barang

Nama	Tukar Barang
Kode	UC-0007
Deskripsi	Implementasi fitur di mana pemain dapat menyetujui atau menolak permintaan pertukaran barang dari pemain lain
Tipe	Fungsional
Pemicu	Pengguna menekan tombol menu permintaan pertukaran
Aktor	Pengguna
Kondisi Awal	Pengguna melihat daftar pertukaran barang pada menu
Aliran: - **Kejadian Normal**	1. Pengguna menekan tombol menu permintaan pertukaran 2. Sistem menampilkan daftar permintaan pertukaran 3. Pengguna melihat barang yang ditawarkan dan diinginkan oleh pemain lain dengan menekan tombol *view* pada daftar pertukaran barang yang dimaksud 4. Pengguna menyetujui permintaan pertukaran dengan cara menekan tombol *accept*

	5. Sistem melakukan pertukaran barang yang dimaksud
-Kejadian Alternatif	1. Pengguna menekan tombol *decline* 2. Sistem menghapus permintaan pertukaran
Kondisi Akhir	Sistem melakukan pertukaran barang



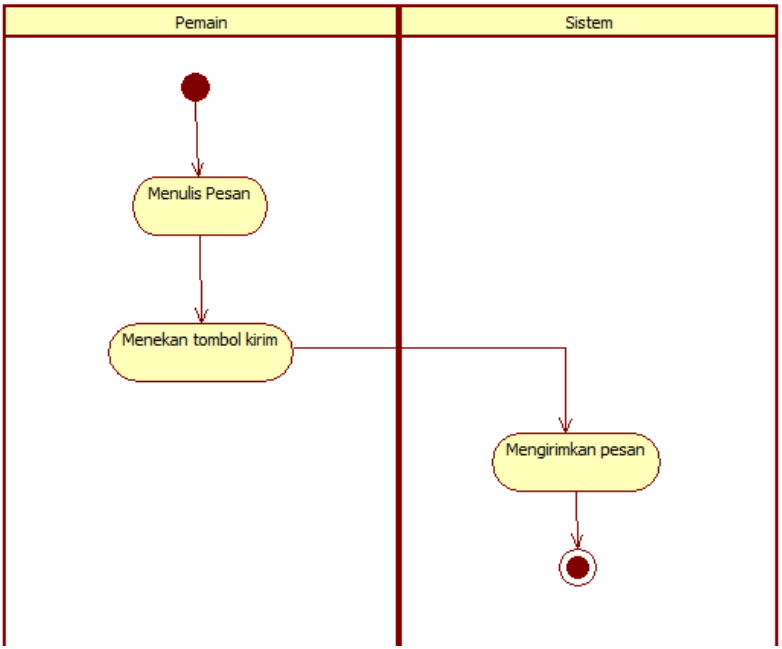
Gambar 3.9. Diagram Aktivitas Menerima Pertukaran Barang

3.1.4.8. Mengirim Pesan

Pada kasus penggunaan ini, pemain dapat mengirim pesan kepada pemain lain. Sistem menerima input berupa pesan dan klik pada tombol *Send*. Lalu sistem akan mengirim pesan yang dimaksud ke orang yang dituju. Spesifikasi kasus penggunaan ini dapat dilihat pada Tabel 3.9. Diagram aktivitas dari kasus penggunaan ini bisa dilihat pada Gambar 3.10.

Tabel 3.9. Spesifikasi Kasus Mengirim Pesan

Nama	Membaca Pesan
Kode	UC-0008
Deskripsi	Implementasi fitur di mana pemain dapat mengirim pesan kepada pemain lain
Tipe	Fungsional
Pemicu	Pengguna menekan tombol *send*
Aktor	Pengguna
Kondisi Awal	Pengguna mengunjungi profil pemain yang akan dikirimi pesan
Aliran: - **Kejadian Normal**	1. Pengguna menuliskan pesan yang akan dikirim 2. Pengguna menekan tombol *send* 3. Sistem akan mengirimkan pesan pada pemain yang dimaksud
-Kejadian Alternatif	Tidak ada
Kondisi Akhir	Sistem mengirim pesan



Gambar 3.10. Diagram Aktivitas Mengirim Pesan

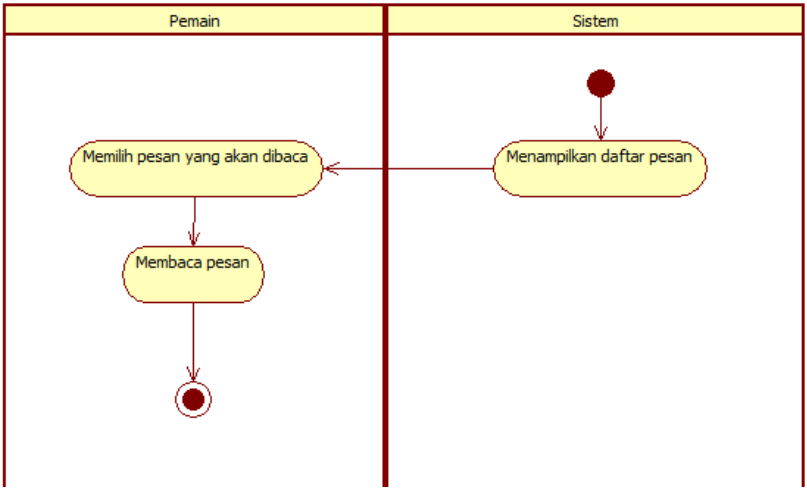
3.1.4.9. Membaca Pesan

Pada kasus penggunaan ini, pemain dapat membaca pesan yang dikirim oleh pemain lain. Sistem akan menampilkan pesan yang dikirim. Spesifikasi kasus penggunaan ini dapat dilihat pada Tabel 3.10. Diagram aktivitas dari kasus penggunaan ini bisa dilihat pada Gambar 3.11.

Tabel 3.10. Spesifikasi Kasus Membaca Pesan

Nama	Membaca Pesan
Kode	UC-0009
Deskripsi	Implementasi fitur di mana pemain dapat membaca pesan yang dikirimkan oleh pemain lain

Tipe	Fungsional
Pemicu	Pengguna masuk menu *Messaging*
Aktor	Pengguna
Kondisi Awal	Pengguna lain telah mengirim pesan
Aliran: **- Kejadian Normal**	1. Sistem menampilkan pesan. 2. Pengguna membaca pesan.
-Kejadian Alternatif	Tidak ada
Kondisi Akhir	Pengguna telah membaca pesan



Gambar 3.11. Diagram Aktivitas Membaca Pesan

3.1.5. 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 3.11.

Tabel 3.11. Daftar Kebutuhan Fungsional Perangkat Lunak

Kode Kebutuhan	Kebutuhan Fungsional	Deskripsi
F-0001	Mengirim permintaan pertemanan	Pengguna dapat mengirim permintaan pertemanan pada pengguna lain
F-0002	Menerima permintaan pertemanan	Pengguna dapat menerima maupun menolak permintaan pertemanan dari pengguna lain
F-0003	Mengatur rumah	Pengguna dapat menambah bangunan yang disediakan
F-0004	Berbagi di Facebook	Pengguna dapat berbagi tentang hal-hal seputar game melalui Facebook *Feed*
F-0005	Mengunjungi teman	Pengguna dapat melihat pengguna pemain lain
F-0006	Mengirim permintaan pertukaran barang	Pengguna dapat mengirim permintaan pertukaran barang kepada pengguna lain
F-0007	Menerima pertukaran barang	Pengguna dapat menerima maupun menolak permintaan pertukaran barang yang dikirim oleh pengguna lain
F-0008	Mengirim pesan	Pengguna dapat mengirim pesan pada pengguna lain
F-0009	Membaca pesan	Pengguna dapat membaca pesan yang dikirimkan oleh pengguna lain

3.2. Perancangan

Penjelasan tahap perancangan perangkat lunak dibagi menjadi beberapa bagian yaitu perancangan diagram kelas dan perancangan antarmuka.

3.2.1. Perancangan Diagram Kelas

Perancangan diagram kelas berisi rancangan dari kelas-kelas yang digunakan untuk membangun sistem. Pada subbab ini, akan dijelaskan rancangan diagram kelas untuk aplikasi permainan yang akan dibangun. Pada rancangan diagram kelas, selain beberapa kelas model, semua kelas yang dirancang akan ditempelkan pada masing-masing obyek di Unity dikarenakan Unity berbasis komponen.

Untuk rancangan diagram kelas yang akan dibangun, dapat dilihat di Lampiran B pada Gambar 8.1.

3.2.2. Perancangan Antarmuka Pengguna

Bagian ini membahas rancangan tampilan antar muka pada sistem. Pada sistem terdapat banyak tampilan yang digunakan pemain dari menu utama yang berupa *house editor* untuk masuk ke menu-menu lain seperti *friend management*, *visit friend's profile*, dan menu pertukaran barang.

3.2.2.1. Halaman Tampilan Menu Utama

Gambar 3.12. Rancangan Tampilan Utama House Editor

Halaman ini merupakan tampilan utama yang muncul setelah pengguna *login*. Pada halaman ini terdapat tombol-tombol yang berupa gambar atau *icon*, maupun tombol seperti pada umumnya. Selain itu, terdapat tombol sub-menu di mana tombol tersebut akan muncul saat tombol tertentu di klik. Rancangan tampilan dapat dilihat di Gambar 3.12.

Gambar 3.13. Rancangan Tampilan Tombol Sub-menu Add Building

Gambar 3.14. Rancangan Tampilan Tombol Sub-menu My Profile

Pengguna dapat memilih beberapa menu seperti yang terlihat pada Gambar 3.13 dan Gambar 3.14. Selain itu, terdapat beberapa panel yang akan muncul saat pengguna menekan tombol tertentu, seperti panel pesan pada Gambar 3.15 yang menampilkan pesan-pesan yang diterima pengguna, serta panel permintaan pertukaran pada Gambar 3.16 yang menampilkan nama-nama teman yang mengirimkan permintaan pertukaran barang pada pengguna.

Sedangkan pada Gambar 3.17, merupakan rancangan antarmuka panel *gift* di mana pengguna dapat melakukan *share daily gift to* Facebook dengan cara menekan tombol *share*.

Gambar 3.15. Rancangan Tampilan Panel Pesan

Gambar 3.16. Rancangan Tampilan Panel Permintaan Pertukaran

Gambar 3.17. Rancangan Tampilan Panel *Gift*

Spesifikasi atribut antarmuka tampilan utama dapat dilihat pada Tabel 3.12.

Tabel 3.12. Spesifikasi Atribut Antarmuka Tampilan Utama

No	Nama Atribut Antarmuka	Jenis Atribut	Kegunaan
1	*House_*	*Button*	Melakukan eksekusi untuk menampilkan tombol *Add Building Button*, *My Profile Button*, dan *My Friend Button*
2	*Colosseum_*	*Button*	Melakukan eksekusi untuk berpindah ke tampilan menu pemain melawan pemain
3	*Altar_*	*Button*	Melakukan eksekusi untuk menambahkan *magic dust* dan menambah *experience* dari bangunan *altar*
4	*Mine_*	*Button*	Melakukan eksekusi untuk menambahkan *gem* dan menambah *experience* dari bangunan *mine*

No	Nama Atribut Antarmuka	Jenis Atribut	Kegunaan
5	*Yggdrasil_*	*Button*	Melakukan eksekusi untuk menambahkan *berry* dan menambah *experience* dari bangunan *yggdrasil*
6	*Trade Button*	*Button*	Melakukan eksekusi untuk menampilkan *field Trade Panel*
7	*Dungeon Button*	*Button*	Melakukan eksekusi untuk berpindah ke menu *dungeon*
8	*Mail Button*	*Button*	Melakukan eksekusi untuk menampilkan *field Message Panel*
9	*Gift Button*	*Button*	Melakukan eksekusi untuk menampilkan panel *gift*
10	*Share Gift Button*	*Button*	Melakukan eksekusi untuk *share daily gift* melalui Facebook *Feed*
11	*Quit Button*	*Button*	Melakukan eksekusi untuk menutup aplikasi permainan
12	*Add Building Button*	*Button*	Melakukan eksekusi untuk menampilkan tombol *Altar Button*, *Mine Button*, dan *Yggdrasil Button*
13	*My Profile Button*	*Button*	Melakukan eksekusi untuk menampilkan tombol *Avatar Button, Edit Deck Button*, dan *Edit Party Button*
14	*My Friend Button*	*Button*	Melakukan eksekusi untuk berpindah ke menu *Friend Management*
15	*Altar Button*	*Button*	Melakukan eksekusi untuk menambahkan bangunan *Altar*
16	*Mine Button*	*Button*	Melakukan eksekusi untuk menambahkan bangunan *Mine*

No	Nama Atribut Antarmuka	Jenis Atribut	Kegunaan
17	*Yggdrasil Button*	*Button*	Melakukan eksekusi untuk menambahkan bangunan *Yggdrasil*
18	*Avatar Button*	*Button*	Melakukan eksekusi untuk berpindah ke menu *Avatar*
19	*Edit Deck Button*	*Button*	Melakukan eksekusi untuk berpindah ke menu *Edit Deck*
20	*Edit Party Button*	*Button*	Melakukan eksekusi untuk berpindah ke menu *Edit Party*
21	*View Request Button*	*Button*	Melakukan eksekusi untuk berpindah ke menu *View Trade Request*
22	*Message Panel*	Tabel	Menampilkan daftar pesan yang dimiliki pemain
23	*Trade Request Panel*	Tabel	Menampilkan daftar pemain yang mengirim permintaan pertukaran dan menampilkan tombol *View Request Button*
24	*Gift Panel*	Tabel	Menampilkan informasi *gift* yang didapat, serta terdapat tombol untuk *share gift*

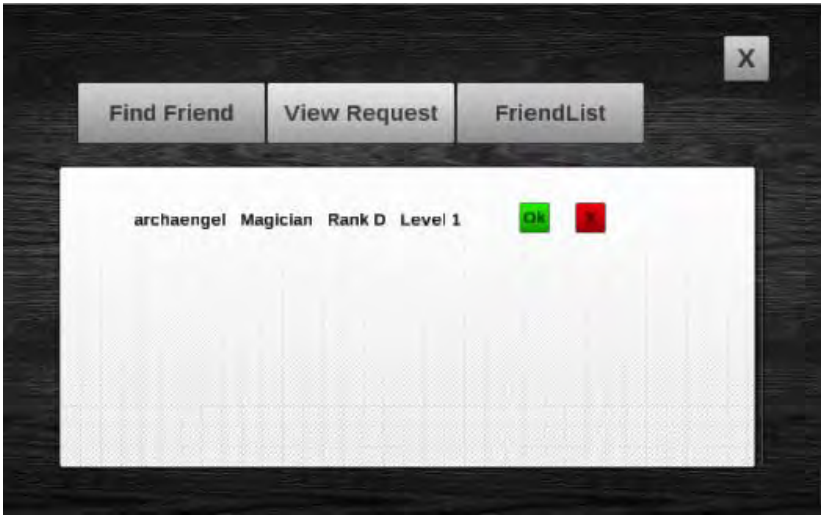
3.2.2.2. Halaman Tampilan Manajemen Pertemanan

Halaman ini merupakan tampilan yang muncul ketika pemain masuk menu manajemen pertemanan. Halaman ini bertujuan untuk menampilkan beberapa fitur, di antaranya menampilkan pencarian pemain yang akan dijadikan teman seperti pada Gambar 3.18. Selain

itu, halaman ini juga menampilkan daftar teman dan daftar permintaan pertemanan seperti pada Gambar 3.19 dan Gambar 3.20.

Gambar 3.18. Rancangan Tampilan Pencarian Pemain

Gambar 3.19. Rancangan Tampilan Daftar Teman



Gambar 3.20. Rancangan Tampilan Daftar Permintaan Teman

Spesifikasi atribut antarmuka tampilan manajemen pemain dapat dilihat pada Tabel 3.13.

Tabel 3.13. Spesifikasi Atribut Antarmuka Manajemen Pemain

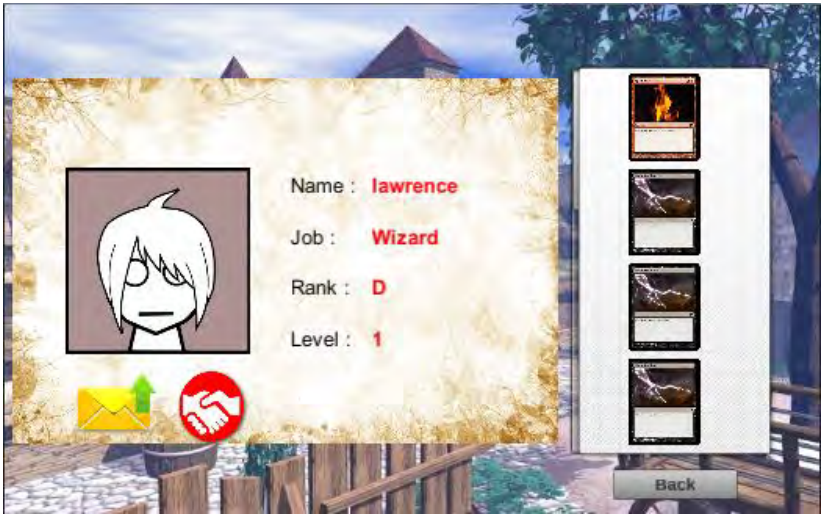
No	Nama Atribut Antarmuka	Jenis Atribut	Kegunaan
1	*Find Friend Button*	*Button*	Melakukan eksekusi untuk menampilkan tampilan pencarian pemain
2	*View Request Button*	*Button*	Melakukan eksekusi untuk menampilkan daftar permintaan pertemanan
3	*Friend List Button*	*Button*	Melakukan eksekusi untuk menampilkan daftar teman
4	*Search Player Button*	*Button*	Melakukan eksekusi untuk pencarian pemain
5	*Add Friend Button*	*Button*	Melakukan eksekusi untuk menambahkan teman dengan mengirimkan permintaan pertemanan ke pemain yang dimaksud
6	*Close Button*	*Button*	Melakukan eksekusi untuk kembali ke menu utama
7	*Visit Button*	*Button*	Melakukan eksekusi untuk mengunjungi profil teman dan berpindah ke menu profil teman
8	*Delete Friend Button*	*Button*	Melakukan eksekusi untuk menghapus pertemanan
9	*Accept Friend Request Button*	*Button*	Melakukan eksekusi untuk menerima permintaan pertemanan yang dikirimkan pemain lain
10	*Decline Friend Request Button*	*Button*	Melakukan eksekusi untuk menolak permintaan pertemanan yang dikirimkan pemain lain

No	Nama Atribut Antarmuka	Jenis Atribut	Kegunaan
11	*Search User Textbox*	*Textbox*	Menerima masukan berupa teks username suatu pemain yang akan dicari
12	*User Detail Textbox*	*Rich Textbox*	Menampilkan detail pemain setelah menekan tombol Search Player Button
13	*Friend List Panel*	*Grid*	Menampilkan daftar teman yang dimiliki dan tombol untuk mengunjungi maupun menghapus pertemanan
14	*Friend Request Panel*	*Grid*	Menampilkan daftar permintaan pertemanan yang dikirim pemain lain dan tombol untuk menolak maupun menyetujui permintaan

3.2.2.3. Halaman Tampilan Mengunjungi Profil Teman

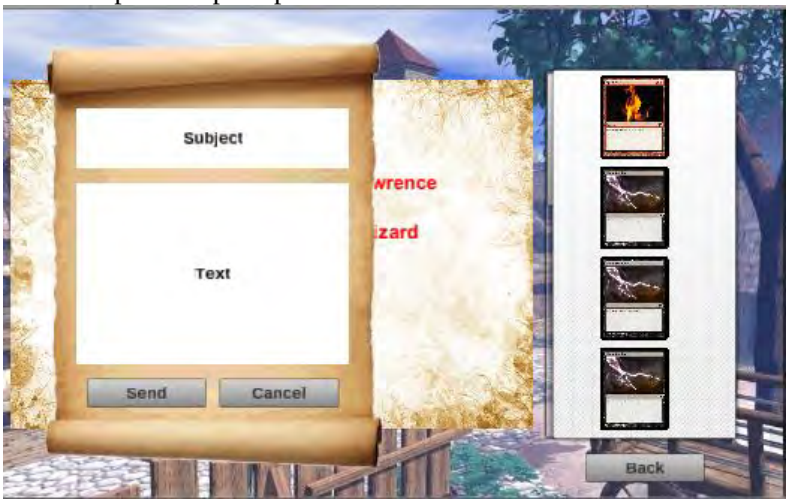
Halaman ini merupakan tampilan yang muncul ketika pemain masuk menu manajemen pertemanan dan menekan tombol *visit* pada salah satu teman di daftar teman. Halaman ini bertujuan untuk menampilkan profil teman yang dikunjungi. Pada halaman ini, selain pemain bisa melihat informasi profil teman, juga bisa mengakses fitur mengirim pesan maupun ajakan untuk pertukaran barang.

Informasi profil pemain yang ditampilkan contohnya adalah *avatar* teman, *deck* yang berisi kartu-kartu yang saat ini digunakan teman, dan keterangan-keterangan seperti *job*, *level*, maupun *rank*. Terlihat seperti tampilan pada Gambar 3.21.



Gambar 3.21. Rancangan Tampilan Mengunjungi Profil Teman

Pada rancangan tampilan tersebut, terdapat tombol atau *icon* berbentuk amplop. *Icon* tersebut berfungsi menampilkan panel masukan pesan seperti pada Gambar 3.22.



Gambar 3.22. Rancangan Tampilan Panel Mengirim Pesan

Spesifikasi atribut antarmuka tampilan mengunjungi profil teman dapat dilihat pada Tabel 3.14.

Tabel 3.14. Spesifikasi Atribut Antarmuka Mengunjungi Profil Teman

No	Nama Atribut Antarmuka	Jenis Atribut	Kegunaan
1	*View Message Panel Button*	*Button*	Melakukan eksekusi untuk menampilkan *Messaging Panel*
2	*Send Trade Request Button*	*Button*	Melakukan eksekusi untuk berpindah ke halaman *Trade Request*
3	*Back Button*	*Button*	Melakukan eksekusi untuk kembali ke halaman utama
4	*Send Message Button*	*Button*	Melakukan eksekusi untuk mengirim pesan ke teman yang dikunjungi profilnya
5	*Cancel Button*	*Button*	Melakukan eksekusi untuk menutup *Messaging Panel*
6	*Friend Avatar*	*Picture*	Menampilkan *avatar* teman yang dikunjungi saat itu
7	*Messaging Panel*	*Field*	Menampilkan *field* yang berisi textbox yang menerima masukan subyek dan isi pesan, juga tombol mengirim maupun batal.
8	*Friend Deck Panel*	*Grid*	Menampilkan kartu-kartu yang dimiliki teman yang saat ini berada pada *deck*
9	*Mail Subject Textbox*	*Textbox*	Menerima masukan berupa tulisan yang merupakan subyek pesan yang akan dikirim
10	*Mail Textbox*	*Textbox*	Menerima masukan berupa tulisan yang merupakan isi pesan yang akan dikirim

No	Nama Atribut Antarmuka	Jenis Atribut	Kegunaan
11	*Friend Name Label*	Label	Menampilkan tulisan yang berisi keterangan nama teman
12	*Friend Job Label*	Label	Menampilkan tulisan yang berisi keterangan *job* teman
13	*Friend Rank Label*	Label	Menampilkan tulisan yang berisi keterangan *rank* teman
14	*Friend Level Label*	Label	Menampilkan tulisan yang berisi keterangan *level* teman

3.2.2.4. Halaman Tampilan Mengirim Permintaan Pertukaran Barang

Halaman ini merupakan tampilan yang muncul ketika pemain mengunjungi profil salah satu teman dan menekan tombol permintaan pertukaran barang. Halaman ini bertujuan agar pemain dapat mengirim permintaan pertukaran pada teman yang dimaksud.

Sebelum mengirim pertukaran barang, pemain terlebih dahulu memindahkan kartu yang ada di masing-masing panel *trunk* pemain ke panel permintaan dan penawaran dengan cara *drag and drop*.

Rancangan tampilan antarmuka mengirim permintaan pertukaran barang ditunjukkan pada Gambar 3.23.

Gambar 3.23. Rancangan Tampilan Mengirim Permintaan Pertukaran Barang

Spesifikasi atribut antarmuka tampilan mengirim permintaan pertukaran barang dapat dilihat pada Tabel 3.15.

Tabel 3.15. Spesifikasi Atribut Antarmuka Mengirim Permintaan Pertukaran Barang

No	Nama Atribut Antarmuka	Jenis Atribut	Kegunaan
1	*My Trunk Panel*	*Grid*	Menampilkan kartu-kartu pengguna yang tersedia untuk ditukar
2	*My Friend Trunk Panel*	*Grid*	Menampilkan kartu-kartu teman yang tersedia untuk ditukar
3	*My Offer Panel*	*Grid*	Menerima masukan berupa perpindahan kartu yang ingin ditawarkan dengan cara *drag and drop* dari *My Trunk Panel*
4	*Requested Panel*	*Grid*	Menerima masukan berupa perpindahan kartu milik teman yang diinginkan dengan cara *drag and drop* dari *My Friend Trunk Panel*

No	Nama Atribut Antarmuka	Jenis Atribut	Kegunaan
5	*Send Button*	*Button*	Melakukan eksekusi untuk mengirim permintaan pertukaran barang kepada teman yang dituju
6	*Cancel Button*	*Button*	Melakukan eksekusi untuk kembali ke halaman profil teman

3.2.2.5. Halaman Tampilan Persetujuan Permintaan Pertukaran Barang

Halaman ini merupakan tampilan yang muncul ketika pemain menekan tombol *view* pada panel permintaan pertukaran barang di menu utama seperti pada Gambar 3.16. Halaman ini bertujuan agar pemain dapat melihat, menerima, maupun menolak permintaan pertukaran yang dikirim oleh teman.

Rancangan tampilan antarmuka mengirim permintaan pertukaran barang ditunjukkan pada Gambar 3.24.

Gambar 3.24. Rancangan Tampilan Persetujuan Permintaan Pertukaran Barang

Spesifikasi atribut antarmuka tampilan mengirim permintaan pertukaran barang dapat dilihat pada Tabel 3.16.

Tabel 3.16. Spesifikasi Atribut Antarmuka Mengirim Permintaan Pertukaran Barang

No	Nama Atribut Antarmuka	Jenis Atribut	Kegunaan
1	*Requested Panel*	*Grid*	Menampilkan kartu-kartu pengguna diinginkan oleh pengirim permintaan
2	*Offered Panel*	*Grid*	Menampilkan kartu-kartu pengirim permintaan yang ditawarkan pada pengguna
3	*Accept Button*	*Button*	Melakukan eksekusi untuk menerima permintaan pertukaran dan sistem akan melakukan pertukaran barang tersebut

No	Nama Atribut Antarmuka	Jenis Atribut	Kegunaan
4	*Decline Button*	*Button*	Melakukan eksekusi untuk menolak permintaan pertukaran dan sistem akan menghapus permintaan pertukaran tersebut
5	*Cancel Button*	*Button*	Melakukan eksekusi untuk kembali ke halaman utama

BAB IV
IMPLEMENTASI

Bab ini membahas tentang implementasi dari perancangan aplikasi permainan. Bab ini berisi proses implementasi dari setiap kelas pada semua modul. Dan akan dibagi berdasarkan masing-masing halaman tampilan atau *scene*.

4.1. Implementasi Halaman Tampilan Menu Utama

Pada bagian ini, akan dijelaskan kelas-kelas yang diimplementasikan pada halaman tampilan menu utama. Kelas-kelas yang diimplementasikan adalah kelas-kelas pada kebutuhan fungsionalitas mengatur rumah dan membaca pesan.

4.1.1. Kelas `ModelBuilding`

Kelas ini merupakan kelas *abstract* yang akan diturunkan ke model-model building di antaranya `MineModel`, `YggdrasilModel`, dan `AltarModel`. Di mana kelas ini menyimpan atribut-atribut dan fungsi yang akan digunakan oleh kelas model turunannya. Detail dapat dilihat pada Kode Sumber 9.1. Fungsi yang akan dipakai oleh kelas turunannya adalah fungsi untuk membaca data building dari *file* xml seperti terlihat pada Kode Sumber 4.1.

```
public void getDatabaseBuilding()
    {
        try
        {
            XmlSerializer deserializer = new
XmlSerializer(typeof(PlayerBuildingFromService));
            TextReader textReader = new
StreamReader(Application.persistentDataPath +
"/building_of_" + GameManager.Instance().PlayerId
+ ".xml");
            object obj =
deserializer.Deserialize(textReader);
```

```
            building =
(PlayerBuildingFromService)obj;
            textReader.Close();
        }
        catch (Exception e)
        {
            Debug.Log(e);
        }
                        }
```

Kode Sumber 4.1. Fungsi Membaca Data *Building* dari *File* Xml

4.1.2. Kelas `ModelAltar`

Kelas ini mengimplementasi fungsi pada kelas abstrak `BuildingModel`. Detail dapat dilihat pada Kode Sumber 9.2. Fungsi yang diimplementasi adalah fungsi membaca data *building*. Selain mengimplementasi fungsi dari kelas abstrak, kelas `AltarModel` juga memiliki fungsi yang akan dipanggil oleh kelas `AltarManagement` yang berguna untuk mendapatkan data *altar*.

```
public void getAltarData()
    {
        name = "Altar";
        about = "An Altar that used to perform
some magic, and sometimes drop some Magic Dust";
        level = Building.buildings[0].Level;
        productClaimed =
Building.buildings[0].ProductClaimed;
        productMax =
Building.buildings[0].ProductionQuantity;
        expAltar =
Building.buildings[0].CurrentXP;
        maxAltarExp =
Building.buildings[0].MaxXP;
    }
```

Kode Sumber 4.2. Fungsi Mengambil Data *Altar*

Seperti yang tertera pada Kode Sumber 4.2, keterangan-keterangan tentang bangunan *altar* yang akan dikirimkan ke kelas

kontrolnya adalah keterangan nama bangunan, deskripsi bangunan, level bangunan, produk yang sudah diklaim, maksimal produk yang dapat diklaim, *experience altar* saat ini, dan maksimal *experience altar*.

4.1.3. Kelas `ModelMine`

Kelas ini mengimplementasi fungsi pada kelas abstrak `BuildingModel`. Detail dapat dilihat pada Kode Sumber 9.3. Fungsi yang diimplementasi adalah fungsi membaca data *building*. Selain mengimplementasi fungsi dari kelas abstrak, kelas `MineModel` juga memiliki fungsi yang akan dipanggil oleh kelas `MineManagement` yang berguna untuk mendapatkan data *mine*.

```
public void getMineData()
    {
        name = "Mine";
        about = "A Dwarf's Mine that not used
again, but still produce some Oridecon";
        level = Building.buildings[1].Level;
        productClaimed =
Building.buildings[1].ProductClaimed;
        productMax =
Building.buildings[1].ProductionQuantity;
        expMine =
Building.buildings[1].CurrentXP;
        maxMineExp = Building.buildings[1].MaxXP;
    }
```

Kode Sumber 4.3. Fungsi Mengambil Data *Mine*

Seperti yang tertera pada Kode Sumber 4.3, keterangan-keterangan tentang bangunan *mine* yang akan dikirimkan ke kelas kontrolnya adalah keterangan nama bangunan, deskripsi bangunan, level bangunan, produk yang sudah diklaim, maksimal produk yang dapat diklaim, *experience mine* saat ini, dan maksimal *experience mine*.

4.1.4. Kelas `ModelYggdrasil`

Kelas ini mengimplementasi fungsi pada kelas abstrak `BuildingModel`. Detail dapat dilihat pada Kode Sumber 9.4. Fungsi yang diimplementasi adalah fungsi membaca data *building*. Selain mengimplementasi fungsi dari kelas abstrak, kelas `YggdrasilModel` juga memiliki fungsi yang akan dipanggil oleh kelas `YggdrasilManagement` yang berguna untuk mendapatkan data *yggdrasil*.

```
public void getYggdrasilData()
    {
        name = "Yggdrasil";
        about = "The Tree of Life. Its leaf which
have some magic power, can be used to craft a
Card";
        level = Building.buildings[2].Level;
        productClaimed =
Building.buildings[2].ProductClaimed;
        productMax =
Building.buildings[2].ProductionQuantity;
        expYggdrasil =
Building.buildings[2].CurrentXP;
        maxYggdrasilExp =
Building.buildings[2].MaxXP;
    }
```

Kode Sumber 4.4. Fungsi Mengambil Data *Yggdrasil*

Seperti yang tertera pada Kode Sumber 4.4, keterangan-keterangan tentang bangunan *yggdrasil* yang akan dikirimkan ke kelas kontrolnya adalah keterangan nama bangunan, deskripsi bangunan, level bangunan, produk yang sudah diklaim, maksimal produk yang dapat diklaim, *experience yggdrasil* saat ini, dan maksimal *experience yggdrasil*.

4.1.5. Kelas `AltarManagement`

Kelas ini merupakan kelas kontrol yang akan ditempelkan pada obyek *Altar* di Unity. Kelas ini berfungsi untuk melakukan eksekusi fungsi untuk menambah *experience* maupun level *altar* serta menambah barang pengguna yang bernama *magic dust* ketika

bangunan *altar* di klik oleh pengguna. Detail dapat dilihat pada Kode Sumber 9.5.

4.1.6. Kelas `MineManagement`

Kelas ini merupakan kelas kontrol yang akan ditempelkan pada obyek *Mine* di Unity. Kelas ini berfungsi untuk melakukan eksekusi fungsi untuk menambah *experience* maupun level *mine* serta menambah barang pengguna yang bernama *gem* ketika bangunan *mine* di klik oleh pengguna. Detail dapat dilihat Kode Sumber 9.6.

4.1.7. Kelas `YggdrasilManagement`

Kelas ini merupakan kelas kontrol yang akan ditempelkan pada obyek *Yggdrasil* di Unity. Kelas ini berfungsi untuk melakukan eksekusi fungsi untuk menambah *experience* maupun level *yggdrasil* serta menambah barang pengguna yang bernama *berry* ketika bangunan *yggdrasil* di klik oleh pengguna. Detail dapat dilihat pada Kode Sumber 9.7.

4.1.8. Kelas `BuildingClickManagement`

Kelas ini merupakan kelas kontrol yang akan ditempelkan pada suatu obyek kosong di Unity. Kelas ini berfungsi untuk melakukan eksekusi fungsi ketika bangunan tertentu seperti *house* dan *Colosseum* di klik oleh pengguna. Detail dapat dilihat pada Kode Sumber 9.8. Proses eksekusi dapat dilihat pada potongan kode yang tertera pada Kode Sumber 4.5.

```
void OnClick()
      {
            RaycastHit2D hit =
Physics2D.Raycast(Camera.main.ScreenToWorldPoint(
Input.mousePosition), Vector2.zero);
            if (Input.GetMouseButtonUp(0))
            {
                  if (hit.collider != null)
                  {
```

```
                            if
(!hit.collider.gameObject.name.ToLower().Contains
("buttonaddbuilding") &&

!hit.collider.gameObject.name.ToLower().Contains(
"buttonmyprofile")) Destroy(menuList);

                if
(hit.collider.gameObject.name.ToLower().Contains(
"house_") &&
!hit.collider.gameObject.name.ToLower().Contains(
"button"))
                            {
                                    GameObject obj =
hit.collider.gameObject as GameObject;

      Debug.Log(obj.name);
                                    var currentPos =
this.transform.position;
                                    var subMenuList =
Instantiate(SubMenuHouse, currentPos,
Quaternion.identity);
                                    menuList =
subMenuList;

      clickedBuildingName = obj.name;
                } else
                if
(hit.collider.gameObject.name.ToLower().Contains(
"colosseum"))
                {

GameManager.Instance().GameMode = "pvp";

Application.LoadLevel("PVPlogin");
                }
                   } else Destroy(menuList);
            }
      }
```

Kode Sumber 4.5. Fungsi `OnClick` pada Kelas `BuildingClickManagement`

Seperti yang tertera pada Kode Sumber 4.5, terdapat dua kondisi, yang pertama, ketika bangunan *house* ditekan oleh pengguna, maka akan memunculkan sub-menu dari bangunan tersebut. Sedangkan kondisi kedua adalah saat bangunan *Colosseum* ditekan oleh pengguna, maka akan mengaktifkan mode pemain lawan pemain, dan akan berpindah ke halaman PVP atau *Player Versus Player*.

4.1.9. Kelas `AddBuildingMenu`

Kelas ini merupakan kelas kontrol yang akan ditempelkan pada suatu obyek kosong di Unity. Kelas ini berfungsi untuk melakukan eksekusi fungsi ketika pengguna menekan sub-menu yang berada pada menu utama seperti sub-menu *deck editor*, *party editor*, dan sebagainya. Detail dapat dilihat pada Kode Sumber 9.9. Proses eksekusi dapat dilihat pada potongan kode yang tertera pada Kode Sumber 4.6.

```
void SubMenuClick()
    {
        RaycastHit2D hit =
Physics2D.Raycast(Camera.main.ScreenToWorldPoint(
Input.mousePosition), Vector2.zero);
        if (Input.GetMouseButtonUp(0))
        {
            if (hit.collider != null)
            {
                if
(hit.collider.gameObject.name.ToLower().Contains(
"buttonavatar"))
                {

Application.LoadLevel("AvatarCostumization");
                }
                else if
(hit.collider.gameObject.name.ToLower().Contains(
"buttoneditdeck"))
                {
```

```
Application.LoadLevel("DeckEditor");
                }
                else if
(hit.collider.gameObject.name.ToLower().Contains(
"buttoneditparty"))
                {

Application.LoadLevel("PartyEditor");
                }
            }
            else
            {
                Destroy(subMenu);
            }
        }
    }
```

Kode Sumber 4.6. Fungsi `SubMenuClick` pada Kelas `AddBuildingMenu`

Seperti yang tertera pada Kode Sumber 4.6, terdapat tiga pilihan sub-menu. Yang pertama adalah kostumisasi *avatar*, di mana ketika menu tersebut ditekan, maka tampilan akan berpindah ke halaman kostumisasi *avatar*. Menu kedua adalah *Deck Editor*, ketika menu tersebut ditekan, maka tampilan akan berpindah ke halaman *Deck Editor*. Menu ketiga adalah *Party Editor*, ketika menu tersebut ditekan, maka tampilan akan berpindah ke halaman *Party Editor*.

4.1.10. Kelas `MessageManager`

Kelas ini merupakan kelas kontrol yang akan ditempelkan pada suatu obyek kosong di Unity. Kelas ini berfungsi untuk melakukan beberapa eksekusi fungsi. Detail dapat dilihat pada Kode Sumber 9.10. Yang pertama adalah pada saat tampilan menu utama pertama kali berjalan, maka kelas ini akan mengeksekusi fungsi `ViewMessage` seperti yang tertera pada Kode Sumber 4.7 untuk menampilkan pesan pada panel pesan dengan cara men-

download file xml dari *web service* lalu dibaca menggunakan `XmlSerializer` dan keterangan pesan yang diambil akan ditampilkan.

```
void ViewMessage()
    {

WebServiceSingleton.GetInstance().ProcessRequest(
"get_messages", GameManager.Instance().PlayerId);
        if
(WebServiceSingleton.GetInstance().queryResult >
0)
        {

Debug.Log(WebServiceSingleton.GetInstance().Downl
oadFile("get_messages",
GameManager.Instance().PlayerId));
            try
            {
                XmlSerializer deserializer = new
XmlSerializer(typeof(MessagesFromService));
                textReader = new
StreamReader(Application.persistentDataPath +
"/messages_of_" + GameManager.Instance().PlayerId
+ ".xml");
                object obj =
deserializer.Deserialize(textReader);
                MessagesFromService messagesList
= (MessagesFromService)obj;
                int i = 1;
                foreach (var vmessage in
messagesList.friendMessage)
                {
                    newMessage.name = "Message_"
+ i;

newMessage.transform.GetChild(0).GetComponent<UIL
abel>().name = "Message_Title_" + i;

newMessage.transform.GetChild(1).transform.GetChi
ld(0).name = "Message_Text_" + i;
```

```
newMessage.transform.GetChild(0).GetComponent<UIL
abel>().text = vmessage.Subject;

newMessage.transform.GetChild(1).transform.GetChi
ld(0).GetComponent<UILabel>().text = "From : " +
vmessage.Sender + "\n \n";

newMessage.transform.GetChild(1).transform.GetChi
ld(0).GetComponent<UILabel>().text +=
"Message:\n" + vmessage.Message;
                    i++;
                    var newMes =
NGUITools.AddChild(messageTable, newMessage);
                    Debug.Log(vmessage.Message);
                }
                textReader.Close();

messageTable.GetComponent<UITable>().Reposition()
;
            }
            catch (Exception e)
            {
                Debug.Log(e);
            }
        }
    }
```

Kode Sumber 4.7. Fungsi `ViewMessage` pada Kelas `MessageManager`

Selain itu, kelas ini juga menangani ketika pengguna menekan tombol atau ikon pesan di bagian kanan atas menu utama. Maka kelas ini akan melakukan eksekusi fungsi seperti yang tertera pada Kode Sumber 4.8 untuk memunculkan panel pesan.

```
void ViewMessagePanel()
    {
        if (!GameManager.Instance().UpdatePaused)
        {
```

```
            TweenObjectIn(messagePanel,
messagePanelPosition);
            GameManager.Instance().UpdatePaused =
true;
        }
    }
```

Kode Sumber 4.8. Fungsi `ViewMessagePanel` pada Kelas `MessageManager`

Pada saat pengguna menekan tombol *close* pada panel pesan, maka kelas ini akan mengeksekusi fungsi `MessageTweenOut` seperti pada Kode Sumber 4.9 untuk menghilangkan panel pesan.

```
void MessageTweenOut()
    {
        TweenObjectOut(messagePanel,
messagePanelPositionStart);
        GameManager.Instance().UpdatePaused =
false;
    }
```

Kode Sumber 4.9. Fungsi `MessageTweenOut` pada Kelas `MessageManager`

4.1.11. Kelas `MyTradeRequestManager`

Kelas ini merupakan kelas kontrol yang akan ditempelkan pada suatu obyek kosong di Unity. Kelas ini berfungsi untuk melakukan beberapa eksekusi fungsi. Detail dapat dilihat pada Kode Sumber 9.11. Yang pertama adalah pada saat tampilan menu utama pertama kali berjalan, maka kelas ini akan mengeksekusi fungsi `ViewTradeRequest` seperti yang tertera pada Kode Sumber 4.10 untuk menampilkan nama teman yang mengirim permintaan pertukaran dan tombol untuk melihat permintaan pada panel pesan dengan cara men-*download file* xml dari *web service* lalu dibaca menggunakan `XmlSerializer` dan nama pengirim akan akan ditampilkan.

```
void ViewTradeRequest()
    {

WebServiceSingleton.GetInstance().ProcessRequest(
"get_trade_request_list",
GameManager.Instance().PlayerId);
        if
(WebServiceSingleton.GetInstance().queryResult >
0)
        {

Debug.Log(WebServiceSingleton.GetInstance().Downl
oadFile("get_trade_request_list",
GameManager.Instance().PlayerId));
            try
            {
                XmlSerializer deserializer = new
XmlSerializer(typeof(TradeRequestFromService));
                textReader = new
StreamReader(Application.persistentDataPath +
"/trade_request_list_of_" +
GameManager.Instance().PlayerId + ".xml");
                object obj =
deserializer.Deserialize(textReader);
                TradeRequestFromService tradeList
= (TradeRequestFromService)obj;
                foreach (var from in
tradeList.players)
                {
                    newTradeRequest.name =
from.ID + "_" + "tradeFrom_";
                    viewTradeButton.name =
from.ID + "_" + "tradeID_" ;

newTradeRequest.GetComponent<UILabel>().text =
"trade request from : " + from.Name;
                    var newFromButton =
NGUITools.AddChild(tradeRequestTable,
viewTradeButton);
```

```
                    var newFrom =
NGUITools.AddChild(tradeRequestTable,
newTradeRequest);
                }
                textReader.Close();

tradeRequestTable.GetComponent<UITable>().Reposit
ion();
            }
            catch (Exception e)
            {
                Debug.Log(e);
            }
        }
    }
```

Kode Sumber 4.10. Fungsi `ViewTradeRequest` pada Kelas `MyTradeRequestManager`

Selain itu, kelas ini juga menangani ketika pengguna menekan tombol atau ikon *trade* di bagian atas menu utama. Maka kelas ini akan melakukan eksekusi fungsi seperti yang tertera pada Kode Sumber 4.11 untuk memunculkan panel pesan.

```
void ViewTradeRequestPanel()
    {
        if (!GameManager.Instance().UpdatePaused)
        {
            TweenObjectIn(tradeRequestPanel,
messagePanelPosition);
            GameManager.Instance().UpdatePaused =
true;
        }
    }
```

Kode Sumber 4.11. Fungsi `ViewTradeRequestPanel` pada Kelas `MyTradeRequestManager`

Pada saat pengguna menekan tombol *close* pada panel permintaan pertukaran, maka kelas ini akan mengeksekusi fungsi

`TradeRequestTweenOut` seperti pada Kode Sumber 4.12 untuk menghilangkan panel pesan.

```
void TradeRequestTweenOut()
    {
        TweenObjectOut(tradeRequestPanel,
messagePanelPositionStart);
        GameManager.Instance().UpdatePaused =
false;
    }
```

Kode Sumber 4.12. Fungsi `TradeRequestTweenOut` pada Kelas `MyTradeRequestManager`

Jika terdapat permintaan pertukaran barang pada panel permintaan pertukaran, dan pengguna menekan tombol *view* pada permintaan pertukaran barang yang dimaksud, maka kelas ini akan mengeksekusi fungsi yang tertera pada Kode Sumber 4.13 untuk berpindah ke halaman *trade confirmation*.

```
void GoToTradeRequest(object value)
    {
        tradeID = value as string;
        GameManager.Instance().TradeID        =
tradeID;

WebServiceSingleton.GetInstance().ProcessReque
st("get_card_request_list",
GameManager.Instance().PlayerId    +    "|"    +
tradeID);
        if
(WebServiceSingleton.GetInstance().queryResult
> 0)
        {

Debug.Log(WebServiceSingleton.GetInstance().Do
wnloadFile("get_card_request_list",
GameManager.Instance().PlayerId));
        }
```

```
Application.LoadLevel("TradingConfirmationScen
e");
    }
```

Kode Sumber 4.13. Fungsi `GoToTradeRequest` pada Kelas `MyTradeRequestManager`

4.1.12. Kelas GiftManager

Kelas ini merupakan kelas kontrol yang akan ditempelkan pada suatu obyek kosong di Unity. Kelas ini berfungsi untuk melakukan beberapa eksekusi fungsi, di antaranya adalah `ShowGiftPanel`, `ShowGiftInfo`, dan `ShareGift`. Detail dapat dilihat pada Kode Sumber 9.12.

Yang pertama adalah fungsi `ShowGiftPanel` yang dieksekusi pada saat pengguna menekan tombol atau *icon gift*. Fungsi ini berguna untuk menampilkan panel *gift* serta informasi tentang gift jika sebelumnya pengguna telah mengklaim *gift*. Potongan kode dapat dilihat pada Kode Sumber 4.14.

```
void ShowGiftPanel()
    {
        var parms = new TweenParms();
        parms.Prop("position", new Vector3(0f,
0.5f, 0f));
        HOTween.To(giftPanel.transform, 1f,
parms);
        startInfoPos =
giftInfo.transform.position;
        GameManager.Instance().UpdatePaused =
true;
WebServiceSingleton.GetInstance().ProcessRequest(
"get_gift_notif",
GameManager.Instance().PlayerId);
        if
(WebServiceSingleton.GetInstance().queryResult ==
1)
        {
```

```
giftReceived.GetComponent<UILabel>().text =
WebServiceSingleton.GetInstance().queryInfo;
            ShowGiftInfo();
        }
    }
```

Kode Sumber 4.14. Fungsi `ShowGiftPanel` pada Kelas `GiftManager`

Yang kedua adalah fungsi `ShareGift` yang dieksekusi pada saat pengguna menekan tombol *share*. Fungsi ini berguna untuk mengirimkan request ke Facebook untuk menampilkan *Feed* di Facebook. Potongan kode dapat dilihat pada Kode Sumber 4.15.

Integrasi dengan Facebook dilakukan menggunakan Facebook SDK for Unity. Di mana kakas tersebut akan melakukan pemanggilan fungsi Facebook API dengan menambahkan *query* Facebook API untuk mendapat informasi tertentu. Kemudian *service* Facebook API melakukan permintaan data ke *database* server Facebook. Selanjutnya *database* server Facebook merespon dan oleh *service* API Facebook dikembalikan ke aplikasi sebagai sebuah data JSON.

```
void ShareGift()
    {
        failedLabel.GetComponent<UILabel>().text
= "";
WebServiceSingleton.GetInstance().ProcessRequest(
"share_gift", GameManager.Instance().PlayerId);
        if
(WebServiceSingleton.GetInstance().queryResult ==
1)
        {
            FH.FeedLink =
"cws.yowanda.com/G?name=" +
GameManager.Instance().PlayerId;
            FH.FeedPicture =
"http://cws.yowanda.com/images/img.png";
            FH.feedLinkDescription = "A Gift from
the Night!";
```

```
            FH.CallFBFeed();
        }
        else
        {
failedLabel.GetComponent<UILabel>().text =
WebServiceSingleton.GetInstance().queryInfo;
        }
    }
```

Kode Sumber 4.15. Fungsi `ShareGift` pada Kelas `GiftManager`

4.2. Implementasi Halaman Tampilan Manajemen Pertemanan

Pada bagian ini, akan dijelaskan kelas-kelas yang diimplementasikan pada halaman tampilan manajemen pertemanan. Kelas-kelas yang diimplementasikan adalah kelas-kelas pada kebutuhan fungsionalitas tentang pertemanan.

4.2.1. Kelas `FriendClickManager`

Kelas ini merupakan kelas kontrol yang akan ditempelkan pada suatu obyek kosong di Unity. Kelas ini berfungsi untuk melakukan beberapa eksekusi fungsi, di antaranya adalah `DownloadXML`, `SearchByNickname`, `AddFriend`, `ViewFriendRequest`, `ViewFriendList`, `VisitFriend`, `AcceptFriendRequest`, dan `IgnoreFriendRequest`. Detail dapat dilihat pada Kode Sumber 9.13.

Yang pertama adalah fungsi `DownloadXML` yang dieksekusi pada saat tampilan manejemen pertemanan pertama kali berjalan. Fungsi ini berguna untuk mengambil data dari web service untuk disimpan di dalam bentuk *file* xml. Potongan kode dapat dilihat pada Kode Sumber 4.16.

```
void DownloadXML()
    {

WebServiceSingleton.GetInstance().ProcessRequest(
"get_partial_profile",
```

```
friendSearchInputLabel.GetComponent<UILabel>().te
xt);

Debug.Log(WebServiceSingleton.GetInstance().Downl
oadFile("get_partial_profile",
friendSearchInputLabel.GetComponent<UILabel>().te
xt));
    }
```

Kode Sumber 4.16. Fungsi `DownloadXML` pada Kelas `FriendClickManager`

Informasi yang diambil pada fungsi tersebut adalah keterangan profil yang parsial yang akan dibaca oleh `SearchByNickname` untuk kemudian ditampilkan saat pengguna menekan tombol *search*. Potongan kode fungsi `SearchByNickname` dapat dilihat pada Kode Sumber 4.17.

```
void SearchByNickname()
    {

Debug.Log(friendSearchInputLabel.GetComponent<UIL
abel>().text);

WebServiceSingleton.GetInstance().ProcessRequest(
"get_partial_profile",
friendSearchInputLabel.GetComponent<UILabel>().te
xt);

        if
(WebServiceSingleton.GetInstance().queryResult >
0)
        {
            DownloadXML();
            try
            {
                XmlSerializer deserializer = new
XmlSerializer(typeof(PartialProfileFromService));
                textReader = new
StreamReader(Application.persistentDataPath +
"/partial_profile_of_" +
```

```
friendSearchInputLabel.GetComponent<UILabel>().te
xt + ".xml");
                object obj =
deserializer.Deserialize(textReader);
                players =
(PartialProfileFromService)obj;

friendSearchResultLabel.GetComponent<UILabel>().t
ext = "Nickname: " + players.Name + "\nJob: " +
players.Job + "\nRank: " + players.Rank +
"\nLevel: " + players.Level;
                textReader.Dispose();
                Debug.Log("bisa");
            }
            catch (Exception e)
            {
                Debug.Log(e);
            }
        }
        else
        {

friendSearchResultLabel.GetComponent<UILabel>().t
ext = "Player doesn't exist";
        }
    }
```

Kode Sumber 4.17. Fungsi `SearchByNickname` pada Kelas `FriendProfileManager`

Pada kode sumber diatas, *file* xml akan dibaca dan ditampilkan pada label, sehingga pengguna dapat melihat apakah *username* yang dimasukkan pengguna merupakan *user* yang valid. Setelah mengetahui bahwa user valid, maka pengguna dapat mengirim permintaan pertemanan, dan fungsi `AddFriend` seperti tertera pada Kode Sumber 4.18 akan mengirimkan permintaan pertemanan tersebut dengan mengirimkan *request* melalui *web service*.

```
void AddFriend()
```

```
    {
WebServiceSingleton.GetInstance().ProcessRequest(
"send_friend_request",
GameManager.Instance().PlayerId + "|" +
friendSearchInputLabel.GetComponent<UILabel>().te
xt);

Debug.Log(WebServiceSingleton.GetInstance().query
Info);
        if
(WebServiceSingleton.GetInstance().queryResult >
0)
        {

friendSearchResultLabel.GetComponent<UILabel>().t
ext += "\nStatus: " +
WebServiceSingleton.GetInstance().queryInfo;
        }
        else
        {
friendSearchResultLabel.GetComponent<UILabel>().t
ext = "Player doesn't exist";
        }
    }
```

Kode Sumber 4.18. Fungsi `AddFriend` pada Kelas `FriendClickManager`

4.3. Implementasi Halaman Tampilan Mengunjungi Profil Teman

Pada bagian ini, akan dijelaskan kelas-kelas yang diimplementasikan pada halaman tampilan mengunjungi profil teman. Kelas-kelas yang diimplementasikan adalah kelas-kelas pada kebutuhan fungsionalitas mengunjungi profil teman.

4.3.1. Kelas `FriendProfileManager`

Kelas ini merupakan kelas kontrol yang akan ditempelkan pada suatu obyek kosong di Unity. Kelas ini berfungsi untuk melakukan beberapa eksekusi fungsi, di antaranya adalah

`ViewFriendProfile` untuk menampilkan profil teman yang sedang dikunjungi, `LoadCardFromService` untuk menampilkan daftar kartu yang digunakan teman di dalam *deck*, dan `SendMessage` untuk mengirimkan pesan pada teman yang sedang dikunjungi. Detail dapat dilihat pada Kode Sumber 9.14.

4.3.2. Kelas `AvatarAttachment`

Kelas ini merupakan kelas kontrol yang akan ditempelkan pada suatu obyek kosong di Unity. Di mana ketika fungsi ini dijalankan, maka akan di-*instance* obyek-obyek yang merupakan bagian dari avatar seperti rambut, mata, dan mulut pada obyek kosong di mana kelas ini tertempel.

Kelas ini berfungsi untuk melakukan beberapa eksekusi fungsi, di antaranya adalah `GetDatabaseAvatar`, `AttachHair`, `AttachEye`, dan `AttachMouth`. Detail dapat dilihat pada Kode Sumber 9.15.

`GetDatabaseAvatar` adalah fungsi untuk mendapatkan *list* bagian-bagian *avatar* seperti mata, mulut, dan rambut dari membaca *file* xml yang diambil dari *database server* melalui *web service*. Potongan kode dapat dilihat pada Kode Sumber 4.19.

```
void getDatabaseAvatar()
    {
        try
        {
            XmlSerializer deserializer = new
XmlSerializer(typeof(AvatarFromService));
            TextReader textReader = new
StreamReader(Application.persistentDataPath +
"/player_avatar_of_" + friendName + ".xml");
            object obj =
deserializer.Deserialize(textReader);
            var avi = (AvatarFromService)obj;
            foreach (var s in avi.aviDetail)
            {
                avatarList.Add(s.Name);
                Debug.Log(s.Name);
```

```
            }
            textReader.Close();
        }
        catch (Exception e)
        {
            Debug.Log(e);
        }
    }
```

Kode Sumber 4.19. Fungsi `GetDatabaseAvatar` pada Kelas `AvatarAttachment`

Setelah mendapatkan *list* yang berisi bagian-bagian *avatar*, maka masing-masing bagian akan dimunculkan dalam bentuk obyek pada Unity oleh fungsi `AttachHair`, `AttachEye`, dan `AttachMouth`.

4.4. Implementasi Halaman Tampilan Mengirim Permintaan Pertukaran Barang

Pada bagian ini, akan dijelaskan kelas-kelas yang diimplementasikan pada halaman tampilan permintaan pertukaran barang. Kelas-kelas yang diimplementasikan adalah kelas-kelas pada kebutuhan fungsionalitas mengirim permintaan pertukaran barang.

4.4.1. Kelas `TradingCardLoader`

Kelas ini merupakan kelas kontrol yang akan ditempelkan pada suatu obyek kosong di Unity. Kelas ini berfungsi untuk melakukan beberapa eksekusi fungsi, di antaranya adalah `LoadTrunk` untuk menampilkan kartu-kartu yang ada pada *trunk* teman yang dapat ditukar dan `SendTradingRequest` untuk mengirimkan permintaan pertukaran melalui *web service*. Detail dapat dilihat pada Kode Sumber 9.16. Fungsi `LoadTrunk` dapat dilihat pada Kode Sumber 4.20.

```
public void LoadTrunk(string method, string name,
GameObject grid)
```

```
        {

WebServiceSingleton.GetInstance().ProcessRequest(
"get_player_trunk", name);

WebServiceSingleton.GetInstance().DownloadFile("g
et_player_trunk", name);
            List<string> list = new
List<string>();
            Boolean _isEmpty = false;
            try
            {

                TextReader textReader = new
StreamReader(Application.persistentDataPath + "/"
+ method + name + ".xml");
                _xmlDoc.Load(textReader);
                _nameNodes =
_xmlDoc.GetElementsByTagName("Name");
                _quantityNodes =
_xmlDoc.GetElementsByTagName("Quantity");
                for (int i = 0; i <
_nameNodes.Count; i++)
                {
                    for (int j = 0; j <
int.Parse(_quantityNodes[i].InnerXml); j++)
                    {
list.Add(_nameNodes[i].InnerXml);
                    }
                }
            }
            catch
            {
                _isEmpty = true;
            }

            if (!_isEmpty) AddToGrid(grid, list);
        }
```

Kode Sumber 4.20. Fungsi `LoadTrunk` pada kelas `TradingCardLoader`

4.5. Implementasi Halaman Tampilan Persetujuan Permintaan Pertukaran Barang

Pada bagian ini, akan dijelaskan kelas-kelas yang diimplementasikan pada halaman tampilan persetujuan pertukaran barang. Kelas-kelas yang diimplementasikan adalah kelas-kelas pada kebutuhan fungsionalitas menerima permintaan pertukaran barang.

4.5.1. Kelas `TradingConfirmation`

Kelas ini merupakan kelas kontrol yang akan ditempelkan pada suatu obyek kosong di Unity. Kelas ini berfungsi untuk melakukan beberapa eksekusi fungsi, di antaranya adalah `LoadOffer` untuk menampilkan kartu-kartu yang ingin ditukar dan yang ditawarkan oleh pengirim, `AcceptTradeRequest` untuk menyetujui permintaan pertukaran, dan `DeclineTradeRequest` untuk menolak atau membatalkan permintaan pertukaran. Detail dapat dilihat pada Kode Sumber 9.17.

Cara kerja fungsi `LoadOffer` kurang lebih sama seperti fungsi `LoadTrunk` pada kelas `TradingCardLoader`. Sedangkan cara kerja `AcceptTradeRequest` hampir sama dengan `DeclineTradeRequest` yaitu sama-sama melakukan *request* ke *web service* untuk menyetujui atau menolak permintaan. Perbedaan dari kedua fungsi itu adalah pada `AcceptTradeRequest`, setelah mengirim *request* ke *web service*, fungsi ini mengirim request kembali ke *web service* untuk men-*download* dan menambahkan kartu yang baru saja masuk dari hasil pertukaran seperti tertera pada Kode Sumber 4.21 dan Kode Sumber 4.22.

```
public void AcceptTradeRequest()
    {

WebServiceSingleton.GetInstance().ProcessRequest(
```

```
"accept_trade_request",
GameManager.Instance().TradeID);

Debug.Log(WebServiceSingleton.GetInstance().Downl
oadFile("get_player_trunk",
GameManager.Instance().PlayerId));
        Application.LoadLevel("HouseEditor");
    }
```

Kode Sumber 4.21. Fungsi `AcceptTradeRequest` pada Kelas `TradingConfirmation`

```
public void DeclineTradeRequest()
    {

WebServiceSingleton.GetInstance().ProcessRequest(
"decline_trade_request",
GameManager.Instance().TradeID);
        Application.LoadLevel("HouseEditor");
    }
```

Kode Sumber 4.22. Fungsi `DeclineTradeRequest` pada Kelas `TradingConfirmation`

BAB V
PENGUJIAN DAN EVALUASI

Bab ini membahas pengujian dan evaluasi pada modul yang dikembangkan. Pengujian yang dilakukan adalah secara *black box* yaitu pengujian fungsionalitas dan pengujian integrasi untuk menguji integrasi modul ini dengan modul-modul lain. Pengujian dilakukan untuk mengetahui apakah modul sudah sesuai dengan kasus penggunaan pada bab 3.

5.1. Lingkungan Pengujian

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

Jenis Device :Lenovo IdeaTab S600
Prosesor :Quad-core 1.2 GHz.
Sistem Operasi :Android Jelly Bean 4.2.
Memori :1.02 GB.

Jenis Device :HandPhone Andromax Z
Prosesor :Quad-core 1.5 GHz MT6589E.
Sistem Operasi :Android Jelly Bean 4.2.1.
Memori :1.00 GB

5.2. Skenario Pengujian

Pada bagian ini akan dijelaskan tentang skenario pengujian yang dilakukan. Pengujian yang dilakukan adalah pengujian fungsionalitas berdasarkan masing-masing kebutuhan untuk menguji apakah modul yang dibuat sudah berjalan dengan benar.

Selain itu akan dilakukan pengujian untuk integrasi antar modul, yaitu modul sosial ini dengan modul-modul lain yang dikerjakan oleh anggota tim.

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 pada kebutuhan fungsionalitas dapat dijabarkan pada subbab berikut.

5.2.1.1. Pengujian Fitur Mengirim Permintaan Pertemanan

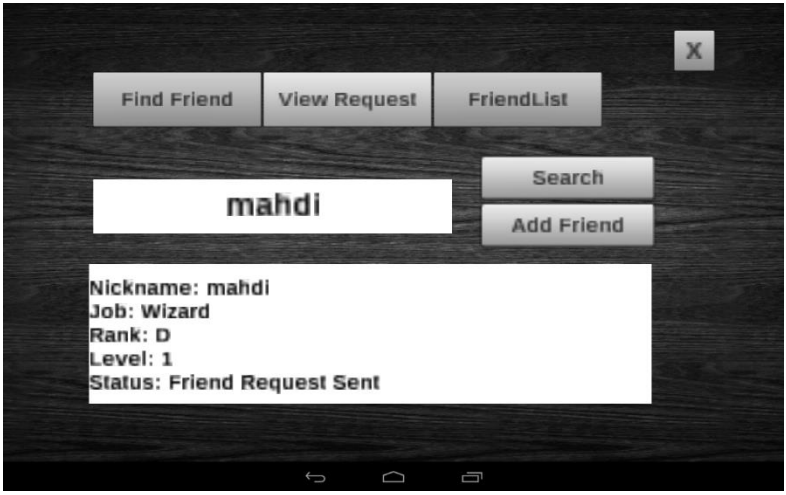
Pengujian fitur mengirim permintaan pertemanan bertujuan untuk menguji apakah aplikasi dapat mengirim permintaan pertemanan yang dikirimkan oleh pengguna ke pengguna lain.

Tabel 5.1. Pengujian Fitur Mengirim Permintaan Pertemanan

ID	UJ.UC-0001
Referensi Kasus Penggunaan	UC-0001
Nama	Pengujian fitur mengirim permintaan pertemanan
Tujuan Pengujian	Menguji apakah pemain dapat mengirimkan permintaan pertemanan kepada pemain lain
Skenario	Pengguna menekan tombol *Add Friend*
Kondisi Awal	Pengguna sudah masuk menu manajemen pertemanan
Masukan	Sentuhan layar
Hasil Yang Diharapkan	Sistem mengirimkan permintaan pertemanan dan menampilkan status permintaan pertemanan pada pengguna

Hasil Yang Didapat	Sistem mengirimkan permintaan pertemanan dan menampilkan status permintaan pertemanan pada pengguna
Hasil Pengujian	Berhasil
Kondisi Akhir	Sistem menampilkan teks "*Friend Request Sent*"

Gambar 5.1. Kondisi Awal Setelah Pengguna Menemukan Pemain yang Dimaksud



Gambar 5.2. Kondisi Setelah Pengguna Menekan Tombol Add Friend

Detail dari skenario dapat dilihat pada Tabel 5.1, sedangkan hasil uji dapat dilihat pada Gambar 5.1 dan Gambar 5.2. Berdasarkan pengujian, aplikasi menampilkan teks status yang berada di bagian paling bawah, bahwa permintaan pertemanan telah terkirim. Sehingga menunjukkan bahwa pengujian berhasil.

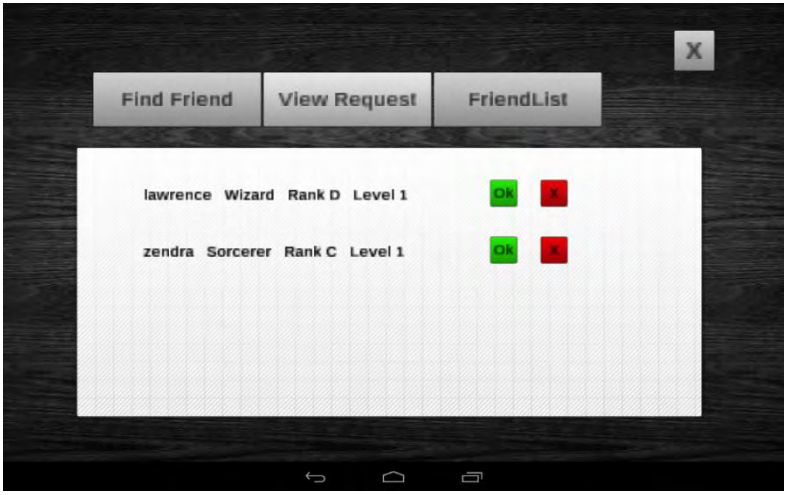
5.2.1.2. Pengujian Fitur Menerima Permintaan Pertemanan

Pengujian fitur menerima permintaan pertemanan bertujuan untuk menguji apakah aplikasi dapat menampilkan, menerima maupun menolak permintaan pertemanan yang dikirimkan oleh pengguna ke pengguna lain.

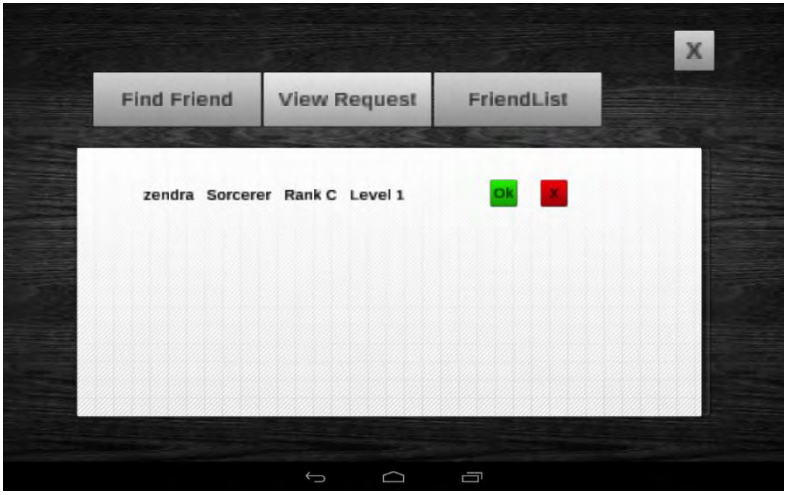
Tabel 5.2. Pengujian Fitur Menerima Permintaan Pertemanan

ID	UJ.UC-0002
Referensi Kasus Penggunaan	UC-0002

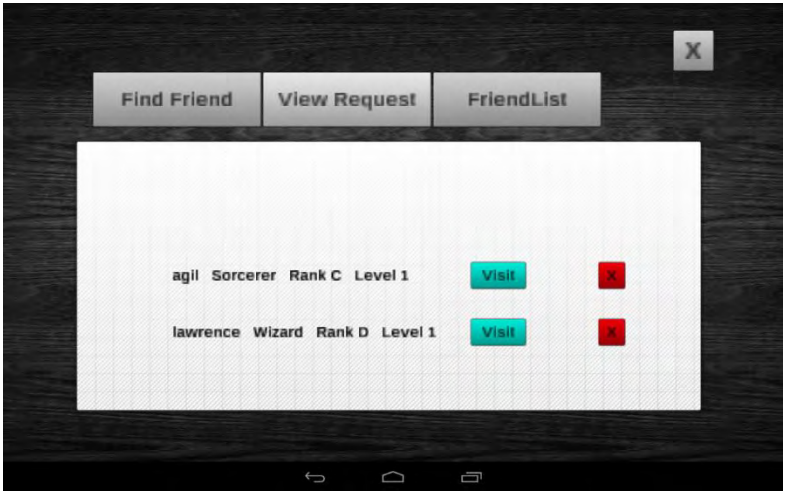
Nama	Pengujian fitur menerima permintaan pertemanan
Tujuan Pengujian	Menguji apakah pemain dapat menerima permintaan pertemanan dari pemain lain
Skenario 1	Pengguna menekan tombol *Accept Friend*
Skenario 2	Pengguna menekan tombol *Decline Friend*
Kondisi Awal	Pengguna lain sudah mengirim permintaan pertemanan
Masukan	Sentuhan layar
Hasil Yang Diharapkan	Sistem menambahkan nama pengguna ke daftar teman
Hasil Yang Didapat	Sistem menambahkan nama pengguna ke daftar teman
Hasil Pengujian	Berhasil
Kondisi Akhir	Sistem menampilkan nama pengguna di daftar teman



Gambar 5.3. Kondisi Awal Saat Pemain Melihat Friend Request



Gambar 5.4. Tampilan View Friend Request Setelah Pemain Menerima Request dari Salah Satu Pemain



Gambar 5.5. Tampilan View FriendList Setelah Pemain Menerima Request dari Salah Satu Pemain

Detail dari skenario dapat dilihat pada Tabel 5.2 sedangkan hasil uji dapat dilihat pada Gambar 5.3, Gambar 5.4 dan Gambar 5.5. Berdasarkan pengujian aplikasi dapat menerima dan menolak permintaan pertemanan. Permintaan yang diterima, akan secara otomatis masuk ke daftar teman pengguna.

5.2.1.3. Pengujian Fitur Mengatur Rumah

Pengujian fitur menerima permintaan pertemanan bertujuan untuk menguji apakah aplikasi dapat menampilkan, menerima maupun menolak permintaan pertemanan yang dikirimkan oleh pengguna ke pengguna lain.

Tabel 5.3. Pengujian Fitur Mengatur Rumah

ID	UJ.UC-0003
Referensi Kasus Penggunaan	UC-0003
Nama	Pengujian fitur mengatur rumah

Tujuan Pengujian	Menguji apakah pemain dapat menambahkan bangunan baru
Skenario	Pengguna menekan tombol *Add Building* dan menambahkan building tertentu
Kondisi Awal	Pengguna sudah berada di halaman menu utama
Masukan	Sentuhan layar
Hasil Yang Diharapkan	Sistem menampilkan bangunan yang ditambahkan
Hasil Yang Didapat	Sistem menampilkan bangunan yang ditambahkan
Hasil Pengujian	Berhasil
Kondisi Akhir	Sistem menampilkan bangunan yang ditambahkan

Gambar 5.6. Kondisi Awal Sebelum Pemain Memiliki Bangunan *Yggdrasil*

Gambar 5.7. Kondisi Setelah Pemain Menambahkan Bangunan *Yggdrasil*

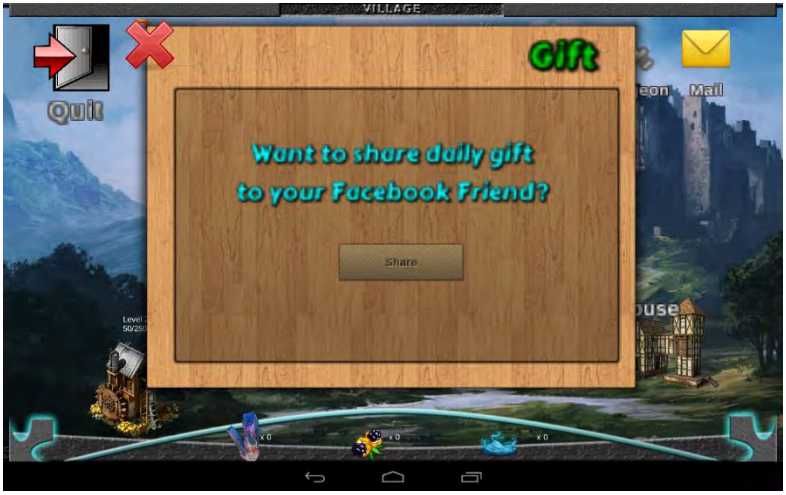
Detail dari skenario dapat dilihat pada Tabel 5.3 sedangkan hasil uji dapat dilihat pada Gambar 5.6 dan Gambar 5.7. Berdasarkan pengujian aplikasi dapat menampilkan bangunan yang baru ditambahkan.

5.2.1.4. Pengujian Fitur Berbagi di Facebook

Pengujian fitur berbagi di Facebook bertujuan untuk menguji apakah aplikasi dapat menampilkan Feed di halaman Facebook melalui Aplikasi Game Card Warlock Saga.

Tabel 5.4. Pengujian Fitur Berbagi di Facebook

ID	UJ.UC-0004
Referensi Kasus Penggunaan	UC-0004
Nama	Pengujian fitur berbagi di Facebook
Tujuan Pengujian	Menguji apakah sistem dapat terintegrasi dengan Facebook, dan Facebook menampilkan Feed melalui aplikasi Card Warlock Saga
Skenario	Pengguna menekan tombol *Share to Facebook*
Kondisi Awal	Pengguna masuk panel *gift*
Masukan	Sentuhan layar
Hasil Yang Diharapkan	Facebook menampilkan Facebook Feed melalui Card Warlock Saga
Hasil Yang Didapat	Facebook menampilkan Facebook Feed melalui Card Warlock Saga
Hasil Pengujian	Berhasil
Kondisi Akhir	Facebook menampilkan Facebook Feed melalui Card Warlock Saga



Gambar 5.8. Kondisi Awal Saat Pemain Berada Pada Panel *Gift*

Gambar 5.9. Tampilan Ketika Pengguna Berbagi di Facebook

Gambar 5.10. Tampilan *Feed* atau Status pada Facebook

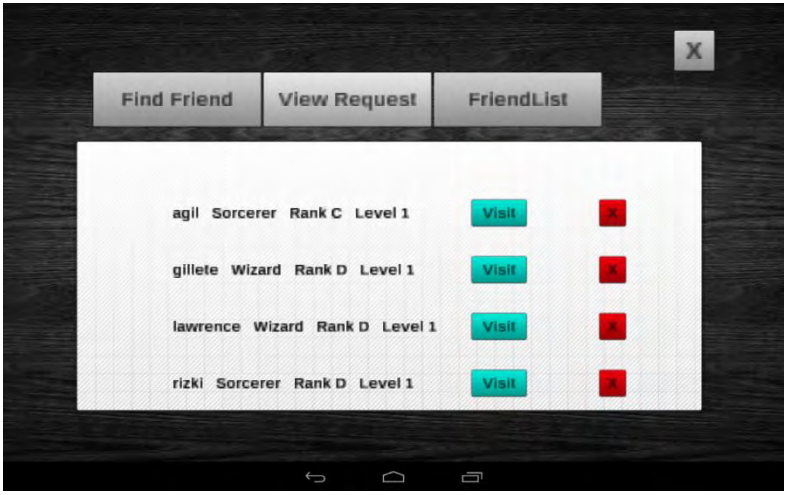
Detail dari skenario dapat dilihat pada Tabel 5.4 sedangkan hasil uji dapat dilihat pada Gambar 5.8, Gambar 5.9, dan Gambar 5.10. Berdasarkan pengujian, sistem dapat terintegrasi dengan Facebook, dan Facebook menampilkan *feed* yang di-*request* oleh sistem.

5.2.1.5. Pengujian Fitur Mengunjungi Teman

Pengujian fitur mengunjungi teman bertujuan untuk menguji apakah pengguna dapat melihat profil teman maupun hal-hal yang berhubungan dengan profil teman, seperti kartu-kartu yang digunakan dalam *deck*, level pemain, rank pemain, dan sebagainya.

Tabel 5.5. Pengujian Fitur Mengunjungi Teman

ID	UJ.UC-0005
Referensi Kasus Penggunaan	UC-0005
Nama	Pengujian fitur Mengunjungi Teman
Tujuan Pengujian	Menguji apakah pengguna dapat melihat profil teman maupun hal-hal yang berhubungan dengan profil teman
Skenario	Pengguna menekan tombol *Visit Friend*
Kondisi Awal	Pengguna memiliki teman yang terdaftar di daftar teman
Masukan	Sentuhan layar
Hasil Yang Diharapkan	Sistem dapat menampilkan profil teman
Hasil Yang Didapat	Sistem dapat menampilkan profil teman
Hasil Pengujian	Berhasil
Kondisi Akhir	Sistem dapat menampilkan profil teman pada halaman mengunjungi teman



Gambar 5.11. Kondisi Awal Ketika Pemain Memilih Teman yang Dikunjungi

Gambar 5.12. Tampilan Ketika Mengunjungi Profil Teman

Detail dari skenario dapat dilihat pada Tabel 5.5 sedangkan hasil uji dapat dilihat pada Gambar 5.11 dan Gambar 5.12.

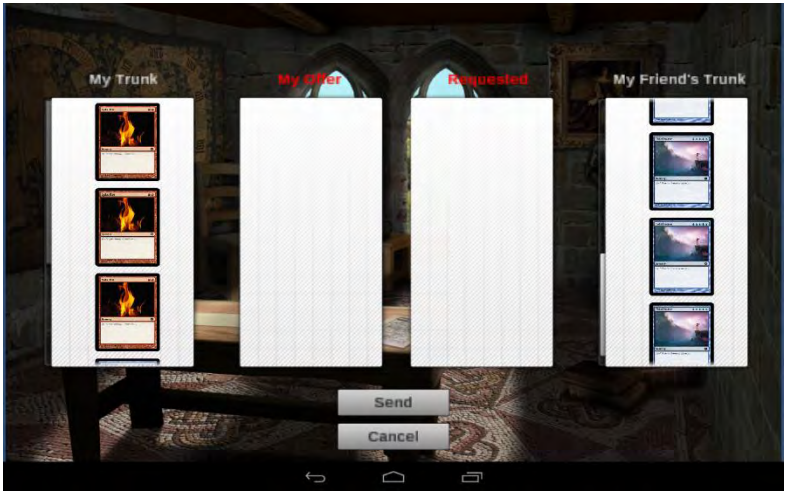
Berdasarkan pengujian aplikasi dapat menampilkan profil teman yang dikunjungi.

5.2.1.6. Pengujian Fitur Mengirim Permintaan Pertukaran Barang

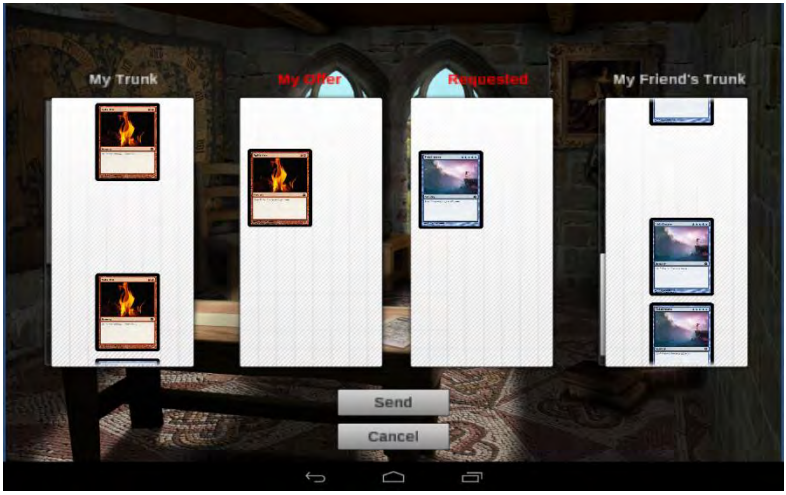
Pengujian fitur mengirim permintaan pertukaran barang bertujuan untuk menguji apakah pengguna dapat mengirim permintaan barang kepada pemain lain.

Tabel 5.6. Pengujian Fitur Mengirim Permintaan Pertukaran Barang

ID	UJ.UC-0006
Referensi Kasus Penggunaan	UC-0006
Nama	Pengujian fitur mengirim permintaan pertukaran barang
Tujuan Pengujian	Menguji apakah pengguna dapat mengirim permintaan pertukaran barang pada pengguna lain
Skenario	Pengguna menekan tombol *Send Trade Request*
Kondisi Awal	Pengguna masuk ke halaman permintaan pertukaran barang, dan memasukkan kartu yang diinginkan dan ditawarkan di *field* yang disediakan
Masukan	Sentuhan layar
Hasil Yang Diharapkan	Sistem mengirim permintaan pertukaran sehingga pengguna lain dapat menerima
Hasil Yang Didapat	Sistem mengirim permintaan pertukaran sehingga pengguna lain dapat menerima
Hasil Pengujian	Berhasil
Kondisi Akhir	Sistem mengirim permintaan pertukaran



Gambar 5.13. Kondisi Awal Ketika Pemain Melihat Kartu yang Tersedia



Gambar 5.14. Kondisi Ketika Pemain Mengirim Permintaan Pertukaran

Detail dari skenario dapat dilihat pada Tabel 5.6 sedangkan hasil uji dapat dilihat pada Gambar 5.13 dan Gambar 5.14. Berdasarkan pengujian aplikasi dapat menampilkan permintaan pertukaran barang dengan kartu apa saja yang ditawarkan dan diinginkan.

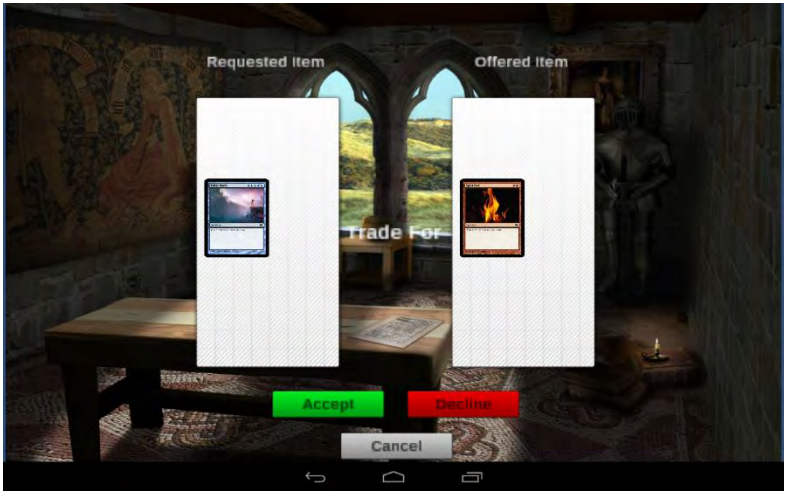
5.2.1.7. Pengujian Fitur Menerima Permintaan Pertukaran Barang

Pengujian fitur menerima permintaan pertukaran barang bertujuan untuk menguji apakah pengguna dapat menerima permintaan pertukaran barang yang dikirim oleh pemain lain.

Tabel 5.7. Pengujian Fitur Menerima Permintaan Pertukaran Barang

ID	UJ.UC-0007
Referensi Kasus Penggunaan	UC-0007
Nama	Pengujian fitur menerima permintaan pertukaran barang
Tujuan Pengujian	Menguji apakah pengguna dapat menerima permintaan pertukaran barang yang dikirim oleh pemain lain
Skenario 1	Pengguna menekan tombol *Accept Trade Request*
Skenario 2	Pengguna menekan tombol *Decline Trade Request*
Kondisi Awal	Pengguna telah mendapat permintaan pertukaran barang yang dikirim pemain lain
Masukan	Sentuhan layar
Hasil Yang Diharapkan	Kartu yang diinginkan dan ditawarkan dikirim ke masing-masing pengguna
Hasil Yang Didapat	Kartu yang diinginkan dan ditawarkan dikirim ke masing-masing pengguna
Hasil Pengujian	Berhasil
Kondisi Akhir	Kartu berhasil ditukar dan masuk ke profil masing-masing pengguna

Gambar 5.15. Kondisi Awal Ketika Pemain Memilih Melihat *Trade Request*



Gambar 5.16. Tampilan Ketika Melihat Trade Request dari Pemain Lain

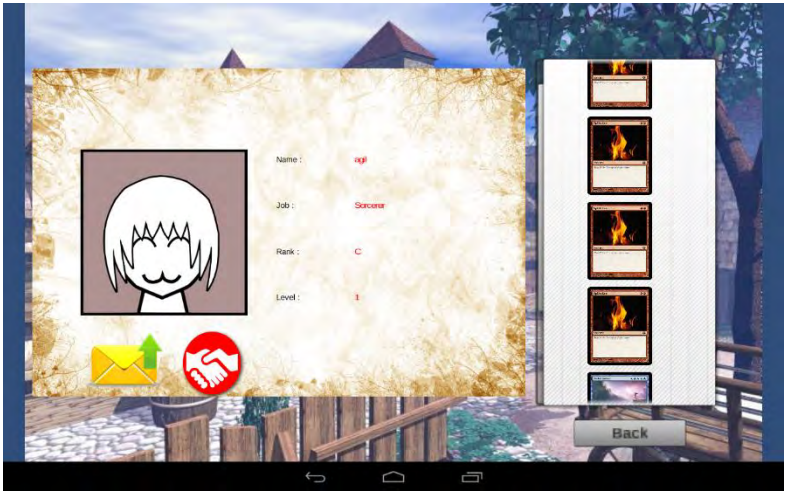
Detail dari skenario dapat dilihat pada Tabel 5.7 sedangkan hasil uji dapat dilihat pada Gambar 5.15 dan Gambar 5.16. Berdasarkan pengujian aplikasi dapat menampilkan permintaan pertukaran kartu untuk kemudian disetujui atau ditolak.

5.2.1.8. Pengujian Fitur Mengirim Pesan

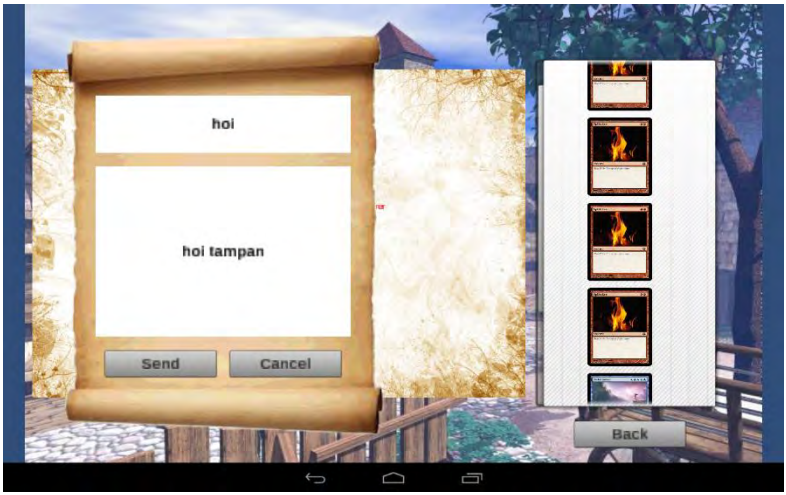
Pengujian fitur mengirim pesan bertujuan untuk menguji apakah pengguna dapat mengirim pesan ke pengguna lain.

Tabel 5.8. Pengujian Fitur Mengirim Pesan

ID	UJ.UC-0008
Referensi Kasus Penggunaan	UC-0008
Nama	Pengujian fitur mengirim pesan
Tujuan Pengujian	Menguji apakah pengguna dapat mengirim pesan kepada pengguna lain
Skenario	Pengguna menekan tombol *Send Message*
Kondisi Awal	Pengguna mengunjungi teman yang akan dikirimi pesan dan menuliskan pesan pada *field* yang disediakan
Masukan	Sentuhan layar
Hasil Yang Diharapkan	Sistem berhasil mengirim pesan
Hasil Yang Didapat	Sistem berhasil mengirim pesan
Hasil Pengujian	Berhasil
Kondisi Akhir	Sistem berhasil mengirim pesan kepada pengguna lain.



Gambar 5.17. Kondisi Awal Ketika Pemain Mengunjungi Teman Untuk Mengirim Pesan



Gambar 5.18. Kondisi Ketika Mengirim Pesan

Detail dari skenario dapat dilihat pada Tabel 5.8 sedangkan hasil uji dapat dilihat pada Gambar 5.17 dan Gambar 5.18.

Berdasarkan pengujian aplikasi dapat menampilkan pengiriman pesan.

5.2.1.9. Pengujian Fitur Membaca Pesan

Pengujian fitur mengunjungi teman bertujuan untuk menguji apakah pengguna dapat membaca pesan yang sebelumya telah dikirim oleh pengguna lain.

Tabel 5.9. Pengujian Fitur Membaca Pesan

ID	UJ.UC-0009
Referensi Kasus Penggunaan	UC-0009
Nama	Pengujian fitur membaca pesan
Tujuan Pengujian	Menguji apakah pengguna dapat membaca pesan yang telah dikirimkan pengguna lain
Skenario	Pengguna menekan tombol *Mail*
Kondisi Awal	Pengguna lain telah mengirim pesan
Masukan	Sentuhan layar
Hasil Yang Diharapkan	Sistem menampilkan pesan beserta nama pengirimnya
Hasil Yang Didapat	Sistem menampilkan pesan beserta nama pengirimnya
Hasil Pengujian	Berhasil
Kondisi Akhir	Sistem menampilkan pesan beserta nama pengirimnya

Gambar 5.19. Kondisi Awal Ketika Pemain Berada di Halaman Utama

Gambar 5.20. Kondisi Ketika Pemain Membaca Pesan

Detail dari skenario dapat dilihat pada Tabel 5.9 sedangkan hasil uji dapat dilihat pada Gambar 5.19 dan Gambar 5.20. Berdasarkan pengujian aplikasi dapat menampilkan pesan yang dikirimkan pemain lain.

5.2.2. Pengujian Integrasi dengan Modul Lain

Pengujian integrasi dengan modul lain dilakukan dengan menyiapkan skenario sebagai tolak ukur keberhasilan pengujian dan setiap hasil dari skenario akan dilihat dari modul lain untuk melihat apakah integrasi antar modul telah tereksekusi dengan benar.

5.2.2.1. Pengujian Integrasi dengan Modul *Server Side*

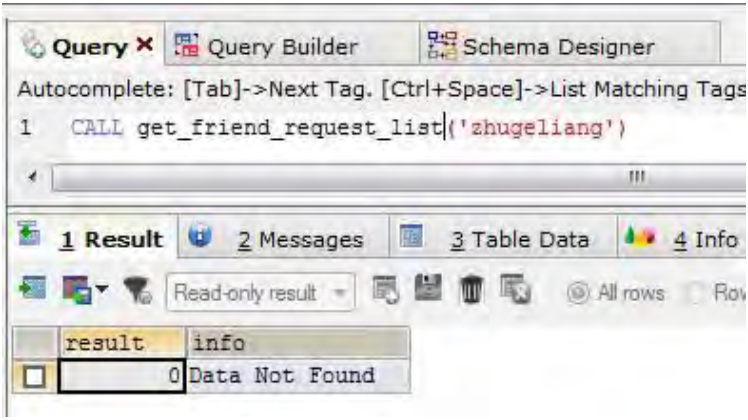
Pengujian integrasi dengan modul *server side* dilakukan untuk menguji apakah kegiatan yang dilakukan pada modul sosial mempengaruhi modul *server side*. Yang diuji adalah *database* yang terdapat pada modul *server side* dan yang diuji pada modul sosial adalah fitur mengirim permintaan pertemanan. Detail skenario pengujian dapat dilihat pada Tabel 5.10.

Tabel 5.10. Pengujian Integrasi dengan Modul *Server Side*

ID	UJI.UC-0001
Referensi Kasus Penggunaan	UC-0001
Nama	Pengujian integrasi dengan modul *server side*
Tujuan Pengujian	Menguji apakah salah satu fitur pada modul sosial yaitu fitur mengirim permintaan pertemanan mempengaruhi *database* pada modul *server side*
Skenario	Pengguna menekan tombol *Add Friend*
Kondisi Awal	Pengguna sudah masuk menu manajemen pertemanan dan menemukan username pemain yang dimaksud
Masukan	Sentuhan layar
Hasil Yang Diharapkan	*Database* pada modul *server side* menyimpan permintaan pertemanan yang telah dieksekusi pada modul sosial
Hasil Yang Didapat	*Database* pada modul *server side* menyimpan permintaan pertemanan yang telah dieksekusi pada modul sosial

Hasil Pengujian	Berhasil
Kondisi Akhir	*Database* pada modul *server side* ter-*update*

Gambar 5.21. Kondisi Awal Pengujian Integrasi dengan Modul *Server Side*

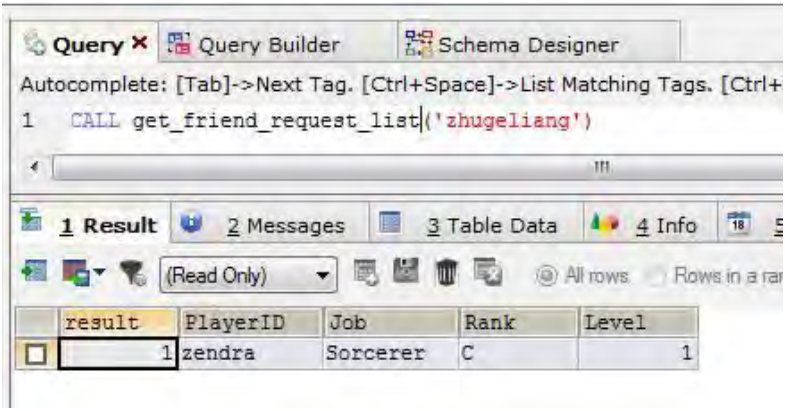


Gambar 5.22. Kondisi Awal *Database* pada Modul *Server Side*

Kondisi awal pengujian adalah ketika pemain telah mencari *username* yang ingin dijadikan teman seperti dapat dilihat pada tampilan modul sosial pada Gambar 5.21. Sedangkan kondisi awal *database* pada modul *server side* sebelum modul sosial mengirimkan permintaan pertemanan, terlihat seperti pada Gambar 5.22 di mana hasil *query* pada permintaan pertemanan yang dimaksud masih kosong.

Saat skenario pengujian dilakukan, maka tampilan pada modul sosial akan tampak seperti Gambar 5.23. Setelah permintaan pertemanan dikirim pada modul sosial, maka *database* pada modul *server side* akan berubah dan terlihat seperti pada Gambar 5.24. Dari hasil pengujian, disimpulkan bahwa pengujian integrasi modul sosial dengan modul *server side* berhasil.

Gambar 5.23. Tampilan Modul Sosial pada Skenario Pengujian UJI.UC-0001



Gambar 5.24. Tampilan Hasil Pengujian pada *Database* Modul *Server Side*

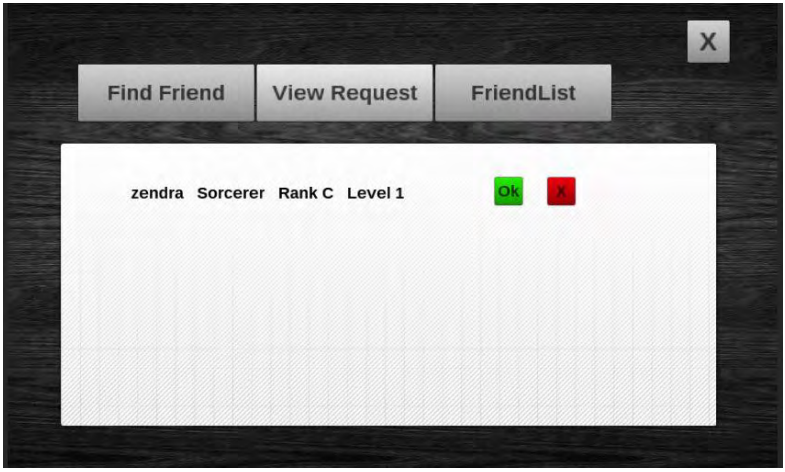
5.2.2.2. Pengujian Integrasi dengan Modul Pertarungan

Pengujian integrasi dengan modul pertarungan dilakukan untuk menguji apakah kegiatan yang dilakukan pada modul sosial mempengaruhi modul pertarungan. Yang diuji adalah fitur *party editor* yang terdapat pada modul pertarungan yang terintegrasi dengan daftar teman pada modul sosial. Detail skenario pengujian dapat dilihat pada Tabel 5.11.

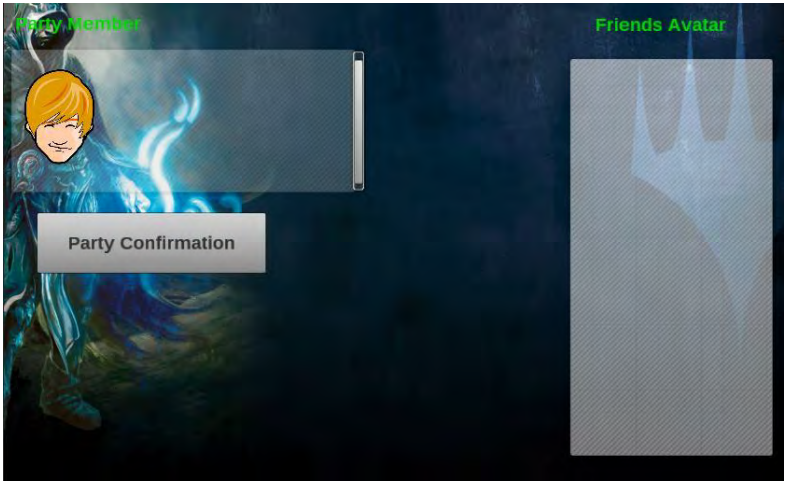
Tabel 5.11. Pengujian Integrasi dengan Modul Pertarungan

ID	UJI.UC-0002
Referensi Kasus Penggunaan	UC-0002
Nama	Pengujian integrasi dengan modul pertarungan
Tujuan Pengujian	Menguji apakah salah satu fitur pada modul sosial yaitu fitur menerima permintaan pertemanan mempengaruhi daftar teman pada fitur *party editor* di modul pertarungan
Skenario	Pengguna menerima permintaan pertemanan
Kondisi Awal	Pengguna sudah mendapat permintaan pertemanan dari pengguna lain

Masukan	Sentuhan layar
Hasil Yang Diharapkan	Daftar teman pada fitur *party editor* di modul pertarungan terintegrasi dengan daftar teman pada modul sosial
Hasil Yang Didapat	Daftar teman pada fitur *party editor* di modul pertarungan terintegrasi dengan daftar teman pada modul sosial
Hasil Pengujian	Berhasil
Kondisi Akhir	*Database* pada fitur *party editor* modul pertarungan ter-*update*



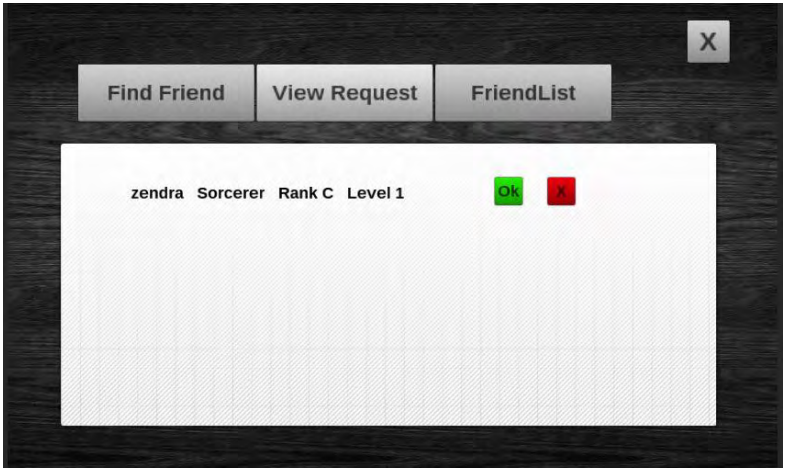
Gambar 5.25. Kondisi Awal Pengujian Integrasi dengan Modul Pertarungan



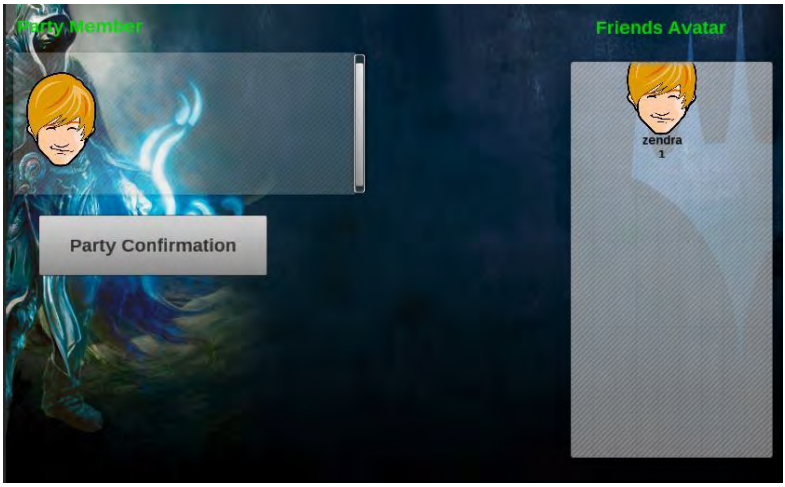
Gambar 5.26. Kondisi Awal Daftar Teman pada *Party Editor* di Modul Pertarungan

Kondisi awal pengujian adalah ketika pemain telah mendapat permintaan pertemanan seperti dapat dilihat pada tampilan modul sosial pada Gambar 5.25. Sedangkan kondisi awal *party editor* pada modul pertarungan sebelum modul sosial menerima permintaan pertemanan, terlihat seperti pada Gambar 5.26 di mana panel daftar teman di sebelah kanan tampilan masih kosong.

Saat skenario pengujian dilakukan yaitu permintaan diterima, maka daftar teman akan bertambah dan terlihat seperti tampilan pada modul sosial di Gambar 5.27. Setelah permintaan pertemanan diterima pada modul sosial, maka daftar teman pada modul sosial dan modul pertarungan akan bertambah seperti terlihat seperti pada Gambar 5.28. Dari hasil pengujian, disimpulkan bahwa pengujian integrasi modul sosial dengan modul pertarungan berhasil.



Gambar 5.27. Tampilan Modul Sosial pada Skenario Pengujian UJI.UC-0002



Gambar 5.28. Tampilan Hasil Pengujian pada Daftar Teman di Modul Pertarungan

BAB VI
KESIMPULAN DAN SARAN

Pada bab ini akan diberikan kesimpulan yang diambil selama pengerjaan tugas akhir serta saran-saran tentang pengembangan yang dapat dilakukan terhadap tugas akhir ini di masa yang akan datang.

6.1. Kesimpulan

Dari hasil selama proses perancangan, implementasi, serta pengujian dapat diambil kesimpulan sebagai berikut:

1. Aplikasi permainan Card Warlock Saga terintegrasi dengan Facebook dengan menggunakan Facebook SDK for Unity seperti pada implementasi di subbab 4.1.12, dimana kakas tersebut akan melakukan pemanggilan fungsi Facebook API dengan menambahkan *query* Facebook API untuk mendapat informasi tertentu. Kemudian *service* Facebook API melakukan permintaan data ke *database* server Facebook. Selanjutnya *database* server Facebook merespon dan oleh *service* API Facebook dikembalikan ke aplikasi sebagai sebuah data JSON.
2. Berdasarkan kebutuhan fungsional pada analisis dan perancangan, maka fitur sosial yang diimplementasikan pada aplikasi permainan Card Warlock Saga yaitu mengirim permintaan pertemanan, menerima permintaan pertemanan, mengatur rumah, berbagi di Facebook, mengunjungi teman, mengirim permintaan pertukaran, menerima pertukaran, mengirim pesan, dan membaca pesan.
3. Sesuai implementasi dan pengujian integrasi yang telah dilakukan pada subbab 5.2.2, integrasi dengan database pada modul *server side* dilakukan dengan pemanggilan *web service* pada setiap fungsionalitas yang diimplementasikan. Sedangkan integrasi dengan modul pertarungan dilakukan

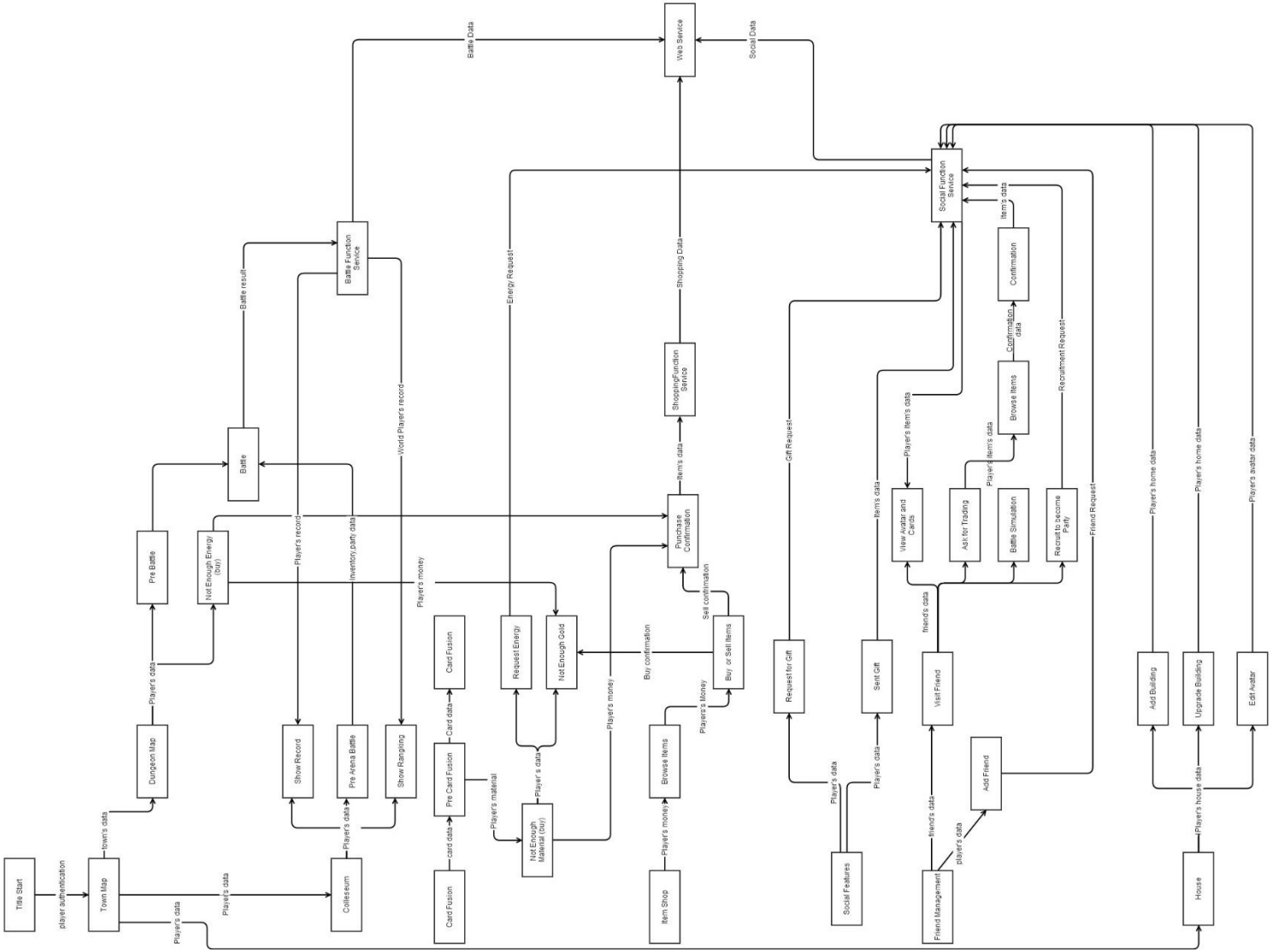
melalui database pada modul *server side* agar data antara modul sosial dan modul pertarungan sama.

6.2. Saran

Berikut saran-saran untuk pengembangan dan perbaikan sistem di masa yang akan datang. Di antaranya adalah sebagai berikut:

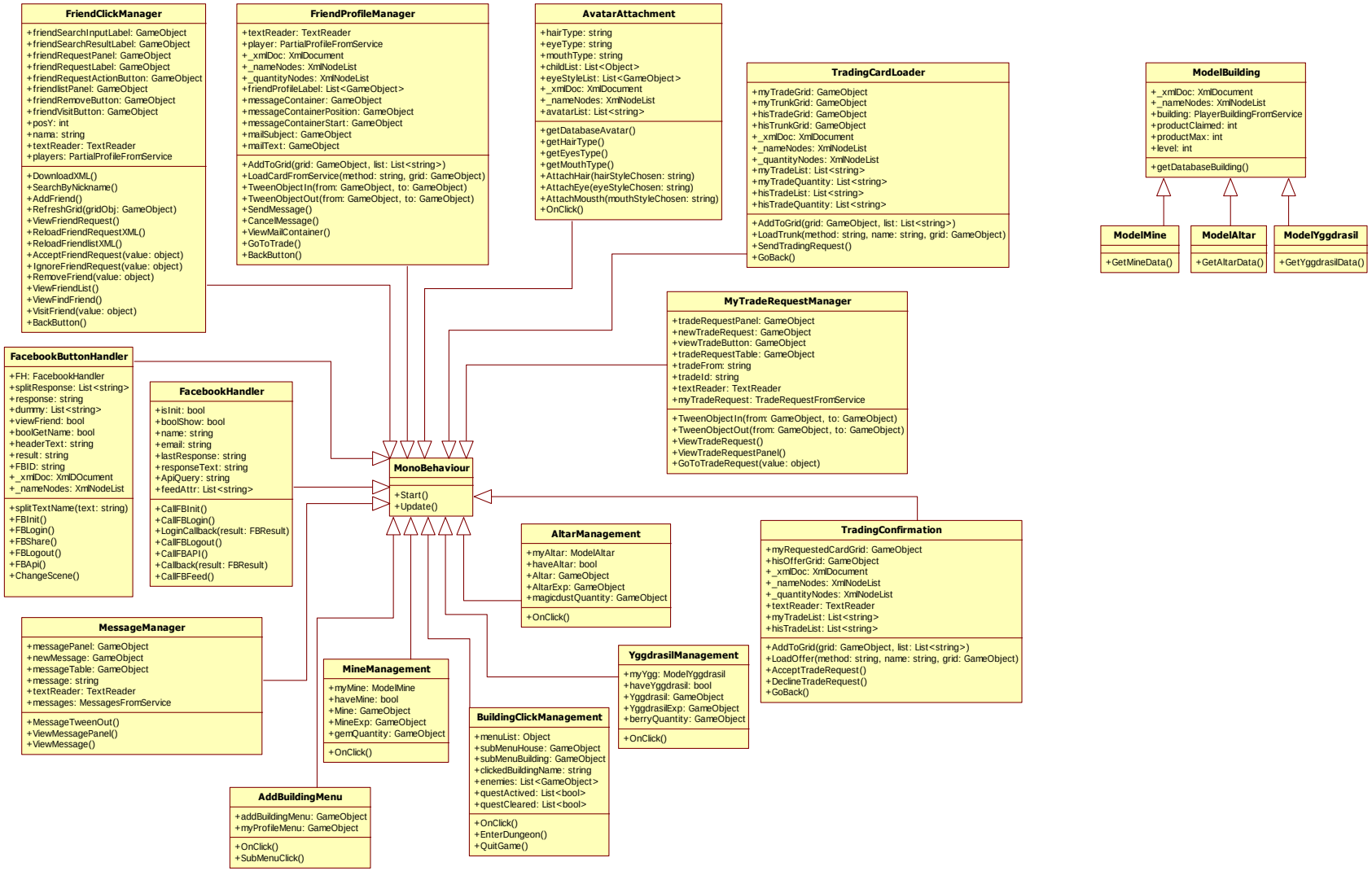
1. Integrasi dengan media sosial lain, tidak hanya dengan Facebook.

LAMPIRAN A. DIAGRAM BLOK PERMAINAN CARD WARLOCK SAGA



Gambar 7.1. Diagram Blok Keseluruhan Permainan Card Warlock Saga

LAMPIRAN B. DIAGRAM KELAS FITUR SOSIAL



Gambar 8.1. Diagram Kelas Fitur Sosial Pada Permainan Card Warlock Saga

LAMPIRAN C.KODE SUMBER

```
using UnityEngine;
using System.Collections;
using System.Net;
using ModulPertarungan;
using System;
using System.IO;
using System.Xml;
using System.Collections.Generic;
using System.Xml.Serialization;

public class ModelBuilding{

    private XmlDocument _xmlDoc;
    private XmlNodeList _nameNodes;
    List<string> avatarList;
    string playerName = "";
    PlayerBuildingFromService building;

    public PlayerBuildingFromService Building
    {
        get { return building; }
        set { building = value; }
    }

    public ModelBuilding()
    {

    }

      int charLevel;
      string playerID;

      public int getCharLevel(string playerID)
      {
            charLevel = 15;
            return charLevel;
      }
```

```
        public int getBuildingData(string
playerId, string buildingName)
        {
                int buildingLevel = 1;
                return buildingLevel;
        }

        public int levelUp(string buildingName)
        {
                int levelNow = 1;
                return levelNow;
        }

    public void getDatabaseBuilding()
    {
        try
        {
            XmlSerializer deserializer = new
XmlSerializer(typeof(PlayerBuildingFromService));
            TextReader textReader = new
StreamReader(Application.persistentDataPath +
"/building_of_" + GameManager.Instance().PlayerId
+ ".xml");
            object obj =
deserializer.Deserialize(textReader);
            building =
(PlayerBuildingFromService)obj;
            textReader.Close();
        }
        catch (Exception e)
        {
            Debug.Log(e);
        }
    }
}
```

Kode Sumber 9.1. Kelas `ModelBuilding`

```
using UnityEngine;
using System.Collections;
```

```
public class ModelAltar : ModelBuilding {

      void Start () {
      }

      void Update () {

      }

      int charLevel;
    int productClaimed, productMax, level;
      string name, about;
      string playerID;
      int expAltar = 0;
      int maxAltarExp = 100;

      public ModelAltar()
    {

    }

    public void getAltarData()
    {
        name = "Altar";
             about = "An Altar that used to
perform some magic, and sometimes drop some Magic
Dust";
             level = Building.buildings[0].Level;
             productClaimed =
Building.buildings[0].ProductClaimed;
        productMax =
Building.buildings[0].ProductionQuantity;
        expAltar =
Building.buildings[0].CurrentXP;
        maxAltarExp =
Building.buildings[0].MaxXP;

             charLevel = getCharLevel (playerID);
    }

    public int ProductMax
    {
```

```
        get { return productMax; }
        set { productMax = value; }
    }

    public int ProductClaimed
    {
        get { return productClaimed; }
        set { productClaimed = value; }
    }

      public int ExpAltar {
            get {
                  return expAltar;
            }
            set {
                  expAltar = value;
            }
      }

      public int MaxAltarExp {
            get {
                  return maxAltarExp;
            }
            set {
                  maxAltarExp = value;
            }
      }

      public int Level {
            get {
                  return level;
            }
            set {
                  level = value;
            }
      }
}
```

Kode Sumber 9.2. Kelas `ModelAltar`

```
using UnityEngine;
```

```
using System.Collections;

public class ModelMine :  ModelBuilding {

       void Start () {

       }

       void Update () {

       }

    int charLevel;
    int productClaimed, productMax, level;
    string name, about;
    string playerID;
    int expMine = 0;
    int maxMineExp = 100;

       public ModelMine()
       {
       }

    public void getMineData()
    {
        name = "Mine";
        about = "A Dwarf's Mine that not used
again, but still produce some Oridecon";
        level = Building.buildings[1].Level;
        productClaimed =
Building.buildings[1].ProductClaimed;
        productMax =
Building.buildings[1].ProductionQuantity;
        expMine =
Building.buildings[1].CurrentXP;
        maxMineExp = Building.buildings[1].MaxXP;

        charLevel = getCharLevel(playerID);
    }

    public int ProductMax
    {
```

```
        get { return productMax; }
        set { productMax = value; }
    }

    public int ProductClaimed
    {
        get { return productClaimed; }
        set { productClaimed = value; }
    }


      public int ExpMine {
            get {
                  return expMine;
            }
            set {
                  expMine = value;
            }
      }

      public int MaxMineExp {
            get {
                  return maxMineExp;
            }
            set {
                  maxMineExp = value;
            }
      }

      public int Level {
            get {
                  return level;
            }
            set {
                  level = value;
            }
      }
}
```

Kode Sumber 9.3. Kelas `ModelMine`

```
using UnityEngine;
using System.Collections;

public class ModelYggdrasil : ModelBuilding {

       void Start () {

       }

       void Update () {

       }

    int charLevel;
    int productClaimed, productMax, level;
    string name, about;
    string playerID;
      int expYggdrasil = 0;
      int maxYggdrasilExp = 100;

      public ModelYggdrasil()
      {
      }

    public void getYggdrasilData()
    {
        name = "Yggdrasil";
        about = "The Tree of Life. Its leaf which
have some magic power, can be used to craft a
Card";
        level = Building.buildings[2].Level;
        productClaimed =
Building.buildings[2].ProductClaimed;
        productMax =
Building.buildings[2].ProductionQuantity;
        expYggdrasil =
Building.buildings[2].CurrentXP;
        maxYggdrasilExp =
Building.buildings[2].MaxXP;

        charLevel = getCharLevel(playerID);
    }
```

```
    public int ProductMax
    {
        get { return productMax; }
        set { productMax = value; }
    }

    public int ProductClaimed
    {
        get { return productClaimed; }
        set { productClaimed = value; }
    }

      public int ExpYggdrasil {
            get {
                  return expYggdrasil;
            }
            set {
                  expYggdrasil = value;
            }
      }

      public int MaxYggdrasilExp {
            get {
                  return maxYggdrasilExp;
            }
            set {
                  maxYggdrasilExp = value;
            }
      }

      public int Level {
            get {
                  return level;
            }
            set {
                  level = value;
            }
      }
}
```

Kode Sumber 9.4. Kelas `ModelYggdrasil`

```
using UnityEngine;
using System.Collections;
using ModulPertarungan;

public class AltarManagement : MonoBehaviour {

    ModelAltar myAltar;
    QuantityManagement quantity;
    bool haveAltar = false;
    public GameObject Altar;
    public GameObject AltarExp;
    public GameObject magicdustQuantity;

    void Start()
    {
        myAltar = new ModelAltar();
        myAltar.getDatabaseBuilding();
        myAltar.getAltarData();
        quantity = new QuantityManagement();
        if (myAltar.Level > 0)
        {
            haveAltar = true;
            var currentPos =
this.transform.position;
            var altar = Instantiate(Altar,
currentPos, Quaternion.identity);
            AltarExp.GetComponent<GUIText>().text
= "Level " + myAltar.Level + "\n" +
myAltar.ExpAltar + "/" + myAltar.MaxAltarExp;
        }
      }

      void Update () {
        if (!GameManager.Instance().UpdatePaused)
        {
            OnClick();
        }
      }
```

```
	void OnClick()
	{
		RaycastHit2D hit =
Physics2D.Raycast(Camera.main.ScreenToWorldPoint(
Input.mousePosition), Vector2.zero);
		if (Input.GetMouseButtonUp(0))
		{
			if (hit.collider != null)
			{
				if
(hit.collider.gameObject.name.ToLower().Contains(
"addaltar") && haveAltar == false)
				{
					GameObject obj =
hit.collider.gameObject as GameObject;

	Debug.Log(obj.name);
					var currentPos =
this.transform.position;
					var altar =
Instantiate(Altar, currentPos,
Quaternion.identity);
					Debug.Log ("add
altar");
					haveAltar = true;

	AltarExp.GetComponent<GUIText>().text =
"Level " + myAltar.Level + "\n" +
myAltar.ExpAltar + "/" + myAltar.MaxAltarExp;
				}
				else
				if
(hit.collider.gameObject.name.ToLower().Contains(
"altar_"))
				{

	quantity.TotalMagicDust++;
					myAltar.ExpAltar
+= 30;

	Debug.Log(myAltar.ExpAltar);
```

```
        if(myAltar.ExpAltar>myAltar.MaxAltarExp)
                                        {

        myAltar.ExpAltar = 0;

        myAltar.Level++;

        myAltar.MaxAltarExp = myAltar.Level * 100;
                                        }

        AltarExp.GetComponent<GUIText>().text =
"Level " + myAltar.Level + "\n" +
myAltar.ExpAltar + "/" + myAltar.MaxAltarExp;

        magicdustQuantity.GetComponent<GUIText>().
text = "x "+ quantity.TotalMagicDust;
                                }
                        }
                }
        }
}
```

Kode Sumber 9.5. Kelas `AltarManagement`

```
using UnityEngine;
using System.Collections;
using ModulPertarungan;

public class MineManagement : MonoBehaviour {

    ModelMine myMine = new ModelMine();
    QuantityManagement quantity = new
QuantityManagement();
    bool haveMine;
    public GameObject Mine;
    public GameObject MineExp;
    public GameObject gemQuantity;

      void Start () {
        myMine = new ModelMine();
```

```
        myMine.getDatabaseBuilding();
        myMine.getMineData();
        quantity = new QuantityManagement();
        if (myMine.Level > 0)
        {
            haveMine = true;
            var currentPos =
this.transform.position;
            var mine = Instantiate(Mine,
currentPos, Quaternion.identity);
            MineExp.GetComponent<GUIText>().text
= "Level " + myMine.Level + "\n" + myMine.ExpMine
+ "/" + myMine.MaxMineExp;
        }
      }

      void Update () {
        if (!GameManager.Instance().UpdatePaused)
        {
            OnClick();
        }
      }


      void OnClick()
      {
            RaycastHit2D hit =
Physics2D.Raycast(Camera.main.ScreenToWorldPoint(
Input.mousePosition), Vector2.zero);
            if (Input.GetMouseButtonUp(0))
            {
                  if (hit.collider != null)
                  {
                        if
(hit.collider.gameObject.name.ToLower().Contains(
"addmine") && haveMine == false)
                        {
                              GameObject obj =
hit.collider.gameObject as GameObject;

      Debug.Log(obj.name);
```

```
                                    var currentPos =
this.transform.position;
                                    var mine =
Instantiate(Mine, currentPos,
Quaternion.identity);
                                    Debug.Log ("add
mine");
                                    haveMine = true;

      MineExp.GetComponent<GUIText>().text =
"Level " + myMine.Level + "\n" + myMine.ExpMine +
"/" + myMine.MaxMineExp;
                              }
                              else
                              if
(hit.collider.gameObject.name.ToLower().Contains(
"mine_"))
                              {

      quantity.TotalGem++;
                                    myMine.ExpMine +=
30;

      Debug.Log(myMine.ExpMine);

      if(myMine.ExpMine>myMine.MaxMineExp)
                                    {

      myMine.ExpMine = 0;

      myMine.Level++;

      myMine.MaxMineExp = myMine.Level * 100;
                                    }

      MineExp.GetComponent<GUIText>().text =
"Level " + myMine.Level + "\n" + myMine.ExpMine +
"/" + myMine.MaxMineExp;

      gemQuantity.GetComponent<GUIText>().text =
"x "+ quantity.TotalGem;
                              }
```

```
                    }
                }
            }
}
```

Kode Sumber 9.6. Kelas `MineManagement`

```
using UnityEngine;
using System.Collections;
using ModulPertarungan;

public class YggdrasilManagement : MonoBehaviour
{

    ModelYggdrasil myYgg = new ModelYggdrasil();
    QuantityManagement quantity = new
QuantityManagement();
    bool haveYggdrasil = false;
    public GameObject Yggdrasil;
    public GameObject YggdrasilExp;
    public GameObject berryQuantity;

       void Start () {
         myYgg = new ModelYggdrasil();
         myYgg.getDatabaseBuilding();
         myYgg.getYggdrasilData();
         quantity = new QuantityManagement();
         if (myYgg.Level > 0)
         {
             haveYggdrasil = true;
             var currentPos =
this.transform.position;
             var yggdrasil =
Instantiate(Yggdrasil, currentPos,
Quaternion.identity);

YggdrasilExp.GetComponent<GUIText>().text =
"Level " + myYgg.Level + "\n" +
myYgg.ExpYggdrasil + "/" + myYgg.MaxYggdrasilExp;
         }
       }
```

```
      void Update () {
        if (!GameManager.Instance().UpdatePaused)
        {
            OnClick();
        }
      }

      void OnClick()
      {
            RaycastHit2D hit =
Physics2D.Raycast(Camera.main.ScreenToWorldPoint(
Input.mousePosition), Vector2.zero);
            if (Input.GetMouseButtonUp(0))
            {
                  if (hit.collider != null)
                  {
                        if
(hit.collider.gameObject.name.ToLower().Contains(
"addyggdrasil") && haveYggdrasil == false)
                        {
                              GameObject obj =
hit.collider.gameObject as GameObject;

      Debug.Log(obj.name);
                              var currentPos =
this.transform.position;
                              var yggdrasil =
Instantiate(Yggdrasil, currentPos,
Quaternion.identity);
                              Debug.Log ("add
yggdrasil");
                              haveYggdrasil =
true;

      YggdrasilExp.GetComponent<GUIText>().text
= "Level " + myYgg.Level + "\n" +
myYgg.ExpYggdrasil + "/" + myYgg.MaxYggdrasilExp;
                        }
                        else
```

```
                            if
(hit.collider.gameObject.name.ToLower().Contains(
"yggdrasil_"))
                            {

        quantity.TotalBerry++;

        myYgg.ExpYggdrasil += 30;


        Debug.Log(myYgg.ExpYggdrasil);

        if(myYgg.ExpYggdrasil>myYgg.MaxYggdrasilEx
p)
                                    {

        myYgg.ExpYggdrasil = 0;
                                            myYgg.Level
++;

        myYgg.MaxYggdrasilExp = myYgg.Level * 100;
                                    }

        YggdrasilExp.GetComponent<GUIText>().text
= "Level " + myYgg.Level + "\n" +
myYgg.ExpYggdrasil + "/" + myYgg.MaxYggdrasilExp;

        berryQuantity.GetComponent<GUIText>().text
= "x "+ quantity.TotalBerry;
                            }
                    }
                }
        }
}
```

Kode Sumber 9.7. Kelas `YggdrasilManagement`

```
using UnityEngine;
using System.Collections;
using ModulPertarungan;
using System.Collections.Generic;
```

```
public class BuildingClickManagement :
MonoBehaviour {
    public List<GameObject> enemies;
    private bool[] questActived;
    private bool[] questCleared;
    private Dictionary<string, bool>
buttonElemental;
    private ShopButtonManager tutorial;

      void Start () {
        GameManager.Instance().GameMode = "";

WebServiceSingleton.GetInstance().isLoadingScreen
= false;
        GameManager.Instance().UpdatePaused =
false;
        tutorial = GameObject.Find("Tutorial
Box").GetComponent<ShopButtonManager>();
        tutorial.SendMessage("ShowShop");
      }

      void Update () {
        if (!GameManager.Instance().UpdatePaused)
        {
            OnClick();
        }
      }

      Object menuList;
      public GameObject SubMenuHouse,
SubMenuBuilding;
      public string clickedBuildingName;

      public string ClickedBuildingName {
            get {
                  return clickedBuildingName;
            }
      }

      void OnClick()
      {
```

```
            RaycastHit2D hit =
Physics2D.Raycast(Camera.main.ScreenToWorldPoint(
Input.mousePosition), Vector2.zero);
            if (Input.GetMouseButtonUp(0))
            {
                  if (hit.collider != null)
                  {
                        if
(!hit.collider.gameObject.name.ToLower().Contains
("buttonaddbuilding") &&

!hit.collider.gameObject.name.ToLower().Contains(
"buttonmyprofile")) Destroy(menuList);

                if
(hit.collider.gameObject.name.ToLower().Contains(
"house_") &&
!hit.collider.gameObject.name.ToLower().Contains(
"button"))
                        {
                              GameObject obj =
hit.collider.gameObject as GameObject;

      Debug.Log(obj.name);
                              var currentPos =
this.transform.position;
                              var subMenuList =
Instantiate(SubMenuHouse, currentPos,
Quaternion.identity);
                              menuList =
subMenuList;

      clickedBuildingName = obj.name;
                } else
                if
(hit.collider.gameObject.name.ToLower().Contains(
"colosseum"))
                {

GameManager.Instance().GameMode = "pvp";

Application.LoadLevel("PVPlogin");
```

```
                }
                    } else Destroy(menuList);
                }
            }
        public void EnterDungeon()
        {
            questActived = new bool[] { true, false,
false, false, false, false, false, false };
            questCleared = new bool[] { false, false,
false, false, false, false, false, false };
            buttonElemental = new Dictionary<string,
bool>()
            {
                {"@Fire",true},
                {"@Water",false},
                {"@Earth",false},
                {"@Thunder",false},
                {"@Wind",false},
            };
            TextureSingleton.Instance().QuestActive =
questActived;
            TextureSingleton.Instance().QuestCleared
= questCleared;
            TextureSingleton.Instance().ElementButton
= buttonElemental;
            TextureSingleton.Instance().BackScene =
Application.loadedLevelName;
            Application.LoadLevel("Dungeon_00");
        }
        public void QuitGame()
        {
            Application.Quit();
        }
}
```

Kode Sumber 9.8. Kelas `BuildingClickManagement`

```
using UnityEngine;
using System.Collections;

public class AddBuildingMenu : MonoBehaviour {
```

```
      void Start () {

      }

      void Update () {
            OnClick ();
        SubMenuClick();
      }

      Object subMenu;
      public GameObject addBuildingMenu;
    public GameObject myProfileMenu;

      void OnClick()
      {
            RaycastHit2D hit =
Physics2D.Raycast(Camera.main.ScreenToWorldPoint(
Input.mousePosition), Vector2.zero);
            if (Input.GetMouseButtonUp(0))
            {
            if (hit.collider != null)
            {
                if
(hit.collider.gameObject.name.ToLower().Contains(
"buttonaddbuilding"))
                {
                    Destroy(subMenu);
                    GameObject obj =
hit.collider.gameObject as GameObject;
                    Debug.Log(obj.name);
                    var currentPos =
this.transform.position;
                    var subMenuList =
Instantiate(addBuildingMenu, currentPos,
Quaternion.identity);
                    subMenu = subMenuList;
                    Debug.Log("addbuilding");
                }
                else if
(hit.collider.gameObject.name.ToLower().Contains(
"buttonmyprofile"))
```

```
                {
                    Destroy(subMenu);
                    GameObject obj =
hit.collider.gameObject as GameObject;
                    Debug.Log(obj.name);
                    var currentPos =
this.transform.position;
                    var subMenuList =
Instantiate(myProfileMenu, currentPos,
Quaternion.identity);
                    subMenu = subMenuList;
                    Debug.Log("myprofile");
                }
                else if
(hit.collider.gameObject.name.ToLower().Contains(
"buttonmyfriend"))
                {

Application.LoadLevel("FriendManagementNew");
                }
            }
            else
            {
                Destroy(subMenu);
            }
             }
      }

    void SubMenuClick()
    {
        RaycastHit2D hit =
Physics2D.Raycast(Camera.main.ScreenToWorldPoint(
Input.mousePosition), Vector2.zero);
        if (Input.GetMouseButtonUp(0))
        {
            if (hit.collider != null)
            {
                if
(hit.collider.gameObject.name.ToLower().Contains(
"buttonavatar"))
                {
```

```
Application.LoadLevel("AvatarCostumization");
                }
                else if
(hit.collider.gameObject.name.ToLower().Contains(
"buttoneditdeck"))
                {

Application.LoadLevel("DeckEditor");
                }
                else if
(hit.collider.gameObject.name.ToLower().Contains(
"buttoneditparty"))
                {

Application.LoadLevel("PartyEditor");
                }
            }
            else
            {
                Destroy(subMenu);
            }
        }
    }
}
```

Kode Sumber 9.9. Kelas `AddBuildingMenu`

```
using UnityEngine;
using System.Collections;
using Holoville.HOTween;
using ModulPertarungan;
using System.Xml.Serialization;
using System.IO;
using System;
using System.Collections.Generic;

public class MessageManager : MonoBehaviour {

    public GameObject messagePanel;
    public GameObject messagePanelPosition;
```

```
    public GameObject messagePanelPositionStart;
    public GameObject newMessage;
    public GameObject messageTable;
    string message;

    TextReader textReader;
    MessagesFromService messages;

      void Start () {
        ViewMessage();
      }

      void Update () {
      }

    void TweenObjectIn(GameObject from,
GameObject to)
    {
        var parms = new TweenParms();
        parms.Prop("position",
to.transform.position);
        HOTween.To(from.transform, 1f, parms);
    }


    void TweenObjectOut(GameObject from,
GameObject to)
    {
        from.transform.position =
to.transform.position;
    }

    void ViewMessagePanel()
    {
        ViewMessage();
        if (!GameManager.Instance().UpdatePaused)
        {
            TweenObjectIn(messagePanel,
messagePanelPosition);
            GameManager.Instance().UpdatePaused =
true;
        }
```

```
    }

    void ReloadGrid()
    {
        List<GameObject> messageChild = new
List<GameObject>();
        foreach (Transform mChild in
messageTable.transform)
        {
            Destroy(mChild.gameObject);
        }
    }

    void MessageTweenOut()
    {
        TweenObjectOut(messagePanel,
messagePanelPositionStart);
        GameManager.Instance().UpdatePaused =
false;
    }

    void ViewMessage()
    {
        ReloadGrid();

WebServiceSingleton.GetInstance().ProcessRequest(
"get_messages", GameManager.Instance().PlayerId);
        if
(WebServiceSingleton.GetInstance().queryResult >
0)
        {

Debug.Log(WebServiceSingleton.GetInstance().Downl
oadFile("get_messages",
GameManager.Instance().PlayerId));
            try
            {
                XmlSerializer deserializer = new
XmlSerializer(typeof(MessagesFromService));
                textReader = new
StreamReader(Application.persistentDataPath +
```

```
"/messages_of_" + GameManager.Instance().PlayerId
+ ".xml");
                object obj =
deserializer.Deserialize(textReader);
                MessagesFromService messagesList
= (MessagesFromService)obj;
                int i = 1;
                foreach (var vmessage in
messagesList.friendMessage)
                {
                    newMessage.name = "Message_"
+ i;

newMessage.transform.GetChild(0).GetComponent<UIL
abel>().name = "Message_Title_" + i;

newMessage.transform.GetChild(1).transform.GetChi
ld(0).name = "Message_Text_" + i;

newMessage.transform.GetChild(0).GetComponent<UIL
abel>().text = vmessage.Subject;

newMessage.transform.GetChild(1).transform.GetChi
ld(0).GetComponent<UILabel>().text = "From : " +
vmessage.Sender + "\n \n";

newMessage.transform.GetChild(1).transform.GetChi
ld(0).GetComponent<UILabel>().text +=
"Message:\n" + vmessage.Message;
                    i++;
                    var newMes =
NGUITools.AddChild(messageTable, newMessage);
                    Debug.Log(vmessage.Message);
                }
                textReader.Close();

messageTable.GetComponent<UITable>().Reposition()
;
            }
            catch (Exception e)
            {
                Debug.Log(e);
```

```
            }
        }
    }
}
```

Kode Sumber 9.10. Kelas `MessageManager`

```
using UnityEngine;
using System.Collections;
using Holoville.HOTween;
using ModulPertarungan;
using System.Xml.Serialization;
using System.IO;
using System;
using System.Collections.Generic;

public class MyTradeRequestManager :
MonoBehaviour {

    public GameObject tradeRequestPanel;
    public GameObject messagePanelPosition;
    public GameObject messagePanelPositionStart;
    public GameObject newTradeRequest;
    public GameObject viewTradeButton;
    public GameObject tradeRequestTable;
    string tradeFrom;
    string tradeID;

    TextReader textReader;
    TradeRequestFromService myTradeRequest;
      void Start () {

WebServiceSingleton.GetInstance().isLoadingScreen
= false;
        ViewTradeRequest();
      }

      void Update () {
      }
```

```
    void TweenObjectIn(GameObject from,
GameObject to)
    {
        var parms = new TweenParms();
        parms.Prop("position",
to.transform.position);
        HOTween.To(from.transform, 1f, parms);
    }

    void ReloadGrid()
    {
        List<GameObject> messageChild = new
List<GameObject>();
        foreach (Transform mChild in
tradeRequestTable.transform)
        {
            Destroy(mChild.gameObject);
        }
    }

    void TweenObjectOut(GameObject from,
GameObject to)
    {
        from.transform.position =
to.transform.position;
    }

    void MessageTweenOut()
    {
        TweenObjectOut(tradeRequestPanel,
messagePanelPositionStart);
        GameManager.Instance().UpdatePaused =
false;
    }

    void ViewTradeRequest()
    {
        ReloadGrid();

WebServiceSingleton.GetInstance().ProcessRequest(
"get_trade_request_list",
GameManager.Instance().PlayerId);
```

```
        if
(WebServiceSingleton.GetInstance().queryResult >
0)
        {

Debug.Log(WebServiceSingleton.GetInstance().Downl
oadFile("get_trade_request_list",
GameManager.Instance().PlayerId));
            try
            {
                XmlSerializer deserializer = new
XmlSerializer(typeof(TradeRequestFromService));
                textReader = new
StreamReader(Application.persistentDataPath +
"/trade_request_list_of_" +
GameManager.Instance().PlayerId + ".xml");
                object obj =
deserializer.Deserialize(textReader);
                TradeRequestFromService tradeList
= (TradeRequestFromService)obj;
                foreach (var from in
tradeList.players)
                {
                    newTradeRequest.name =
from.ID + "_" + "tradeFrom_";
                    viewTradeButton.name =
from.ID + "_" + "tradeID_" ;

newTradeRequest.GetComponent<UILabel>().text =
"trade request from : " + from.Name;
                    var newFromButton =
NGUITools.AddChild(tradeRequestTable,
viewTradeButton);
                    var newFrom =
NGUITools.AddChild(tradeRequestTable,
newTradeRequest);
                }
                textReader.Close();

tradeRequestTable.GetComponent<UITable>().Reposit
ion();
            }
```

```
            catch (Exception e)
            {
                Debug.Log(e);
            }
        }
    }
    void ViewTradeRequestPanel()
    {
        ViewTradeRequest();
        if (!GameManager.Instance().UpdatePaused)
        {
            TweenObjectIn(tradeRequestPanel,
messagePanelPosition);
            GameManager.Instance().UpdatePaused =
true;
        }
    }
    void GoToTradeRequest(object value)
    {
        tradeID = value as string;
        GameManager.Instance().TradeID = tradeID;

WebServiceSingleton.GetInstance().ProcessRequest(
"get_card_request_list",
GameManager.Instance().PlayerId + "|" + tradeID);
        if
(WebServiceSingleton.GetInstance().queryResult >
0)
        {

Debug.Log(WebServiceSingleton.GetInstance().Downl
oadFile("get_card_request_list",
GameManager.Instance().PlayerId));
        }

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

Kode Sumber 9.11. Kelas `MyTradeRequestManager`

```
using UnityEngine;
using System.Collections;
using ModulPertarungan;
using Holoville.HOTween;

public class GiftManager : MonoBehaviour {

    Vector3 startPosition;
    Vector3 startInfoPos;
    public GameObject giftPanel;
    public GameObject failedLabel;
    public GameObject giftInfo;
    public GameObject giftReceived;
    FacebookHandler FH = new FacebookHandler();

       void Start () {
         startPosition =
giftPanel.transform.position;
         failedLabel.GetComponent<UILabel>().text
= "";
       }

       void Update () {

       }

    void ShowGiftPanel()
    {
        var parms = new TweenParms();
        parms.Prop("position", new Vector3(0f,
0.5f, 0f));
        HOTween.To(giftPanel.transform, 1f,
parms);
        startInfoPos =
giftInfo.transform.position;
        GameManager.Instance().UpdatePaused =
true;


WebServiceSingleton.GetInstance().ProcessRequest(
"get_gift_notif",
GameManager.Instance().PlayerId);
```

```
        if
(WebServiceSingleton.GetInstance().queryResult ==
1)
        {

giftReceived.GetComponent<UILabel>().text =
WebServiceSingleton.GetInstance().queryInfo;
            ShowGiftInfo();
        }
    }
    void CloseGiftPanel()
    {
        failedLabel.GetComponent<UILabel>().text
= "";
        var parms = new TweenParms();
        parms.Prop("position", startPosition);
        HOTween.To(giftPanel.transform, 1f,
parms);
        startInfoPos =
giftInfo.transform.position;
        GameManager.Instance().UpdatePaused =
false;
    }
    void ShowGiftInfo()
    {
        var parms = new TweenParms();
        parms.Prop("position", new Vector3(0f,
0.5f, 0f));
        HOTween.To(giftInfo.transform, 1f,
parms);
    }
    void CloseGiftInfo()
    {
        failedLabel.GetComponent<UILabel>().text
= "";
        var parms = new TweenParms();
        parms.Prop("position", startInfoPos);
        HOTween.To(giftInfo.transform, 1f,
parms);
    }

    void ShareGift()
```

```
    {
        failedLabel.GetComponent<UILabel>().text
= "";

WebServiceSingleton.GetInstance().ProcessRequest(
"share_gift", GameManager.Instance().PlayerId);
        if
(WebServiceSingleton.GetInstance().queryResult ==
1)
        {
            FH.FeedLink =
"cws.yowanda.com/G?name=" +
GameManager.Instance().PlayerId;
            FH.FeedPicture =
"http://cws.yowanda.com/images/img.png";
            FH.feedLinkDescription = "A Gift from
the Night!";
            FH.CallFBFeed();
        }
        else
        {

failedLabel.GetComponent<UILabel>().text =
WebServiceSingleton.GetInstance().queryInfo;
        }
    }
}
```

Kode Sumber 9.12. Kelas `GiftManager`

```
using UnityEngine;
using System.Collections;
using ModulPertarungan;
using System.Xml.Serialization;
using System.IO;
using System;

public class FriendClickManager : MonoBehaviour {

    public GameObject friendSearchResultLabel;
    public GameObject friendSearchInputLabel;
```

```
    public GameObject friendRequestPanel;
    public GameObject friendRequestLabel;
    public GameObject friendRequestActionButton;
    public GameObject friendlistPanel;
    public GameObject friendRemoveButton;
    public GameObject friendVisitButton;

    public GameObject friendlistTab,
viewFriendRequestTab, findFriendTab;
    int posY;
    string nama;
    TextReader textReader;
    PartialProfileFromService players;

      void Start () {

      }

      void Update () {

friendRequestPanel.GetComponent<UIGrid>().Reposit
ion();

friendlistPanel.GetComponent<UIGrid>().Reposition
();
      }

    void DownloadXML()
    {

WebServiceSingleton.GetInstance().ProcessRequest(
"get_partial_profile",
friendSearchInputLabel.GetComponent<UILabel>().te
xt);

Debug.Log(WebServiceSingleton.GetInstance().Downl
oadFile("get_partial_profile",
friendSearchInputLabel.GetComponent<UILabel>().te
xt));
    }

    void SearchByNickname()
```

```
    {

Debug.Log(friendSearchInputLabel.GetComponent<UIL
abel>().text);

WebServiceSingleton.GetInstance().ProcessRequest(
"get_partial_profile",
friendSearchInputLabel.GetComponent<UILabel>().te
xt);

        if
(WebServiceSingleton.GetInstance().queryResult >
0)
        {
            DownloadXML();
            try
            {
                XmlSerializer deserializer = new
XmlSerializer(typeof(PartialProfileFromService));
                textReader = new
StreamReader(Application.persistentDataPath +
"/partial_profile_of_" +
friendSearchInputLabel.GetComponent<UILabel>().te
xt + ".xml");
                object obj =
deserializer.Deserialize(textReader);
                players =
(PartialProfileFromService)obj;

friendSearchResultLabel.GetComponent<UILabel>().t
ext = "Nickname: " + players.Name + "\nJob: " +
players.Job + "\nRank: " + players.Rank +
"\nLevel: " + players.Level;
                textReader.Dispose();
                Debug.Log("bisa");
            }
            catch (Exception e)
            {
                Debug.Log(e);
            }
        }
        else
```

```
        {

friendSearchResultLabel.GetComponent<UILabel>().t
ext = "Player doesn't exist";
        }
    }

    void AddFriend()
    {

WebServiceSingleton.GetInstance().ProcessRequest(
"send_friend_request",
GameManager.Instance().PlayerId + "|" +
friendSearchInputLabel.GetComponent<UILabel>().te
xt);

Debug.Log(WebServiceSingleton.GetInstance().query
Info);
        if
(WebServiceSingleton.GetInstance().queryResult >
0)
        {

friendSearchResultLabel.GetComponent<UILabel>().t
ext += "\nStatus: " +
WebServiceSingleton.GetInstance().queryInfo;
        }
        else
        {

friendSearchResultLabel.GetComponent<UILabel>().t
ext = "Player doesn't exist";

        }
    }

    void RefreshGrid(GameObject gridObj)
    {
        foreach (Transform objGrid in
gridObj.transform)
        {
            Destroy(objGrid.gameObject);
```

```
        }
    }

    public void ViewFriendRequest()
    {
        viewFriendRequestTab.SetActive(true);
        friendlistTab.SetActive(false);
        findFriendTab.SetActive(false);
        RefreshGrid(friendRequestPanel);
        ReloadFriendRequestXML();
        if
(WebServiceSingleton.GetInstance().queryResult >
0)
        {
            try
            {
                XmlSerializer deserializer = new
XmlSerializer(typeof(RequestFromService));
                textReader = new
StreamReader(Application.persistentDataPath +
"/friend_request_of_" +
GameManager.Instance().PlayerId + ".xml");
                object obj =
deserializer.Deserialize(textReader);
                RequestFromService friendRequest
= (RequestFromService)obj;
                foreach (var player in
friendRequest.players)
                {

friendRequestLabel.GetComponent<UILabel>().text =
player.Name + "    " + player.Job + "    Rank " +
player.Rank + "    Level " + player.Level;
                    var objL =
NGUITools.AddChild(friendRequestPanel,
friendRequestLabel);
                    objL.name = player.Name +
"_label";
                    var objB =
NGUITools.AddChild(friendRequestPanel,
friendRequestActionButton);
```

```
                    objB.name = player.Name +
"_request_button";
                }
                textReader.Close();
            }
            catch (Exception e)
            {
                Debug.Log(e);
            }
        }
    }

    void ReloadFriendRequestXML()
    {

WebServiceSingleton.GetInstance().ProcessRequest(
"get_friend_request",
GameManager.Instance().PlayerId);

Debug.Log(WebServiceSingleton.GetInstance().Downl
oadFile("get_friend_request",
GameManager.Instance().PlayerId));
    }

    void ReloadFriendlistXML()
    {

WebServiceSingleton.GetInstance().ProcessRequest(
"get_friend_list",
GameManager.Instance().PlayerId);

Debug.Log(WebServiceSingleton.GetInstance().Downl
oadFile("get_friend_list",
GameManager.Instance().PlayerId));
    }

    void AcceptFriendRequest(object value)
    {
        nama = value as string;
        Debug.Log(nama);

WebServiceSingleton.GetInstance().ProcessRequest(
```

```
"accept_friend_request",
GameManager.Instance().PlayerId + "|" + nama);
        ViewFriendRequest();
    }

    void IgnoreFriendRequest(object value)
    {
        nama = value as string;
        Debug.Log(nama);

WebServiceSingleton.GetInstance().ProcessRequest(
"ignore_friend_request",
GameManager.Instance().PlayerId + "|" + nama);
        ViewFriendRequest();
    }

    void RemoveFriend(object value)
    {
        nama = value as string;
        Debug.Log(nama);

WebServiceSingleton.GetInstance().ProcessRequest(
"remove_friend", GameManager.Instance().PlayerId
+ "|" + nama);
        ViewFriendList();
    }

    void ViewFriendList()
    {
        friendlistTab.SetActive(true);
        viewFriendRequestTab.SetActive(false);
        findFriendTab.SetActive(false);

        RefreshGrid(friendlistPanel);
        ReloadFriendlistXML();
        if
(WebServiceSingleton.GetInstance().queryResult >
0)
        {
            try
            {
```

```
                XmlSerializer deserializer = new
XmlSerializer(typeof(FriendListFromService));
                textReader = new
StreamReader(Application.persistentDataPath +
"/friends_of_" + GameManager.Instance().PlayerId
+ ".xml");
                object obj =
deserializer.Deserialize(textReader);
                FriendListFromService friendlist
= (FriendListFromService)obj;
                foreach (var player in
friendlist.players)
                {

friendRequestLabel.GetComponent<UILabel>().text =
player.Name + "    " + player.Job + "    Rank " +
player.Rank + "    Level " + player.Level;
                    var objL =
NGUITools.AddChild(friendlistPanel,
friendRequestLabel);
                    objL.name = player.Name +
"_label";
                    var objV =
NGUITools.AddChild(friendlistPanel,
friendVisitButton);
                    objV.name = player.Name +
"_nvisit_button";
                    var objB =
NGUITools.AddChild(friendlistPanel,
friendRemoveButton);
                    objB.name = player.Name +
"_remove_button";
                }
                textReader.Close();
            }
            catch (Exception e)
            {
                Debug.Log(e);
            }
        }
    }
```

```
    void ViewFindFriend()
    {
        findFriendTab.SetActive(true);
        friendlistTab.SetActive(false);
        viewFriendRequestTab.SetActive(false);
    }

    void VisitFriend(object value)
    {
        nama = value as string;
        Debug.Log("visit " + nama);
        GameManager.Instance().FriendName = nama;

WebServiceSingleton.GetInstance().ProcessRequest(
"get_player_avatar", nama);

Debug.Log(WebServiceSingleton.GetInstance().Downl
oadFile("get_player_avatar", nama));

WebServiceSingleton.GetInstance().ProcessRequest(
"get_profile", nama);

Debug.Log(WebServiceSingleton.GetInstance().Downl
oadFile("get_partial_profile", nama));

WebServiceSingleton.GetInstance().ProcessRequest(
"get_player_deck", nama);

Debug.Log(WebServiceSingleton.GetInstance().Downl
oadFile("get_player_deck", nama));
        Application.LoadLevel("FriendProfile");
    }

    void BackButton()
    {
        Application.LoadLevel("HouseEditor");
    }
}
```

Kode Sumber 9.13. Kelas `FriendClickManager`

```
using UnityEngine;
using System.Collections;
using ModulPertarungan;
using System.Xml.Serialization;
using System.IO;
using System;
using System.Collections.Generic;
using System.Xml;
using Holoville.HOTween;

public class FriendProfileManager : MonoBehaviour
{

    TextReader textReader;
    PartialProfileFromService player;
    private XmlDocument _xmlDoc;
    private XmlNodeList _nameNodes;
    private XmlNodeList _quantityNodes;
    public GameObject friendNameLabel;
    public GameObject friendJobLabel;
    public GameObject friendRankLabel;
    public GameObject friendLevelLabel;
    public GameObject friendCardGrid;

    public GameObject messageContainer;
    public GameObject messageContainerPosition;
    public GameObject messageContainerStart;
    public GameObject mailSubject;
    public GameObject mailText;

      void Start () {

viewFriendProfile(GameManager.Instance().FriendNa
me);
        _xmlDoc = new XmlDocument();
        LoadCardFromService("deck_of_",
friendCardGrid);
      }

      void Update () {

      }
```

```
    void viewFriendProfile(string name)
    {

WebServiceSingleton.GetInstance().ProcessRequest(
"get_partial_profile",
GameManager.Instance().FriendName);
         if
(WebServiceSingleton.GetInstance().queryResult >
0)
         {
             try
             {
                 XmlSerializer deserializer = new
XmlSerializer(typeof(PartialProfileFromService));
                 textReader = new
StreamReader(Application.persistentDataPath +
"/partial_profile_of_" + name + ".xml");
                 object obj =
deserializer.Deserialize(textReader);
                 player =
(PartialProfileFromService)obj;

friendNameLabel.GetComponent<UILabel>().text =
player.Name;

friendJobLabel.GetComponent<UILabel>().text =
player.Job;

friendRankLabel.GetComponent<UILabel>().text =
player.Rank;

friendLevelLabel.GetComponent<UILabel>().text =
player.Level.ToString();
                 textReader.Close();
             }
             catch (Exception e)
             {
                 Debug.Log(e);
             }
         }
    }
```

```
    void AddToGrid(GameObject grid, List<string>
list)
    {
        foreach (string s in list)
        {
            NGUITools.AddChild(grid,
(GameObject)Resources.Load("DisplayCards/" + s,
typeof(GameObject)));
        }
        grid.GetComponent<UIGrid>().Reposition();
    }

    void LoadCardFromService(string method,
GameObject grid)
    {
        List<string> list = new List<string>();
        Boolean _isEmpty = false;
        try
        {
            TextReader textReader = new
StreamReader(Application.persistentDataPath + "/"
+ method + GameManager.Instance().FriendName +
".xml");
            _xmlDoc.Load(textReader);
            _nameNodes =
_xmlDoc.GetElementsByTagName("Name");
            _quantityNodes =
_xmlDoc.GetElementsByTagName("Quantity");

            for (int i = 0; i < _nameNodes.Count;
i++)
            {
                for (int j = 0; j <
int.Parse(_quantityNodes[i].InnerXml); j++)
                {

list.Add(_nameNodes[i].InnerXml);
                }
            }

IntegrationTest.Pass(this.gameObject);
```

```
        }
        catch
        {
            _isEmpty = true;
        }

        if (!_isEmpty) AddToGrid(grid, list);
    }

    void BackButton()
    {

Application.LoadLevel("FriendManagementNew");
    }

    void TweenObjectIn(GameObject from,
GameObject to)
    {
        var parms = new TweenParms();
        parms.Prop("position",
to.transform.position);
        HOTween.To(from.transform, 1f, parms);
    }


    void TweenObjectOut(GameObject from,
GameObject to)
    {
        from.transform.position =
to.transform.position;
    }

    void SendMessage()
    {
        var encoded_subject =
System.Text.Encoding.UTF8.GetBytes(mailSubject.Ge
tComponent<UILabel>().text);
        var encoded_text =
System.Text.Encoding.UTF8.GetBytes(mailText.GetCo
mponent<UILabel>().text);
```

```
WebServiceSingleton.GetInstance().ProcessRequest(
"send_message", GameManager.Instance().PlayerId +
"-" + GameManager.Instance().FriendName + "|" +
System.Convert.ToBase64String(encoded_subject) +
"|" +
System.Convert.ToBase64String(encoded_text));

Debug.Log(WebServiceSingleton.GetInstance().respo
nseFromServer);
        TweenObjectIn(messageContainer,
messageContainerStart);
    }

    void CancelMessage()
    {
        TweenObjectIn(messageContainer,
messageContainerStart);
        mailSubject.GetComponent<UILabel>().text
= "Subject";
        mailText.GetComponent<UILabel>().text =
"Text";
    }

    void ViewMailContainer()
    {
        TweenObjectIn(messageContainer,
messageContainerPosition);
    }

    void GoToTrade()
    {
        Application.LoadLevel("TradingScene");
    }
}
```

Kode Sumber 9.14. Kelas `FriendProfileManager`

```
using UnityEngine;
using System.Collections;
using System.Net;
```

```
using ModulPertarungan;
using System;
using System.IO;
using System.Xml;
using System.Collections.Generic;
using System.Xml.Serialization;

public class AvatarAttachment : MonoBehaviour {

       string hairType, eyesType, mouthType;
    public GameObject hairStyle1, hairStyle2,
hairStyle3;
    public GameObject eyeStyle1, eyeStyle2,
eyeStyle3;
    public GameObject mouthStyle1, mouthStyle2,
mouthStyle3;
    UnityEngine.Object hChild, mChild, eChild;
    string playerName = "";
    private XmlDocument _xmlDoc;
    private XmlNodeList _nameNodes;
    List<string> avatarList;

    public List<string> AvatarList
    {
        get { return avatarList; }
        set { avatarList = value; }
    }

       void Start () {
         playerName =
GameManager.Instance().PlayerId;
         _xmlDoc = new XmlDocument();
         avatarList = new List<string>();
         getDatabaseAvatar();
              AttachHair("menu"+getHairType());
              AttachEye("menu"+getEyesType());
              AttachMouth("menu"+getMouthType());
       }

       void Update () {
         OnClick();
       }
```

```
    void getDatabaseAvatar()
    {
        try
        {
            XmlSerializer deserializer = new
XmlSerializer(typeof(AvatarFromService));
            TextReader textReader = new
StreamReader(Application.persistentDataPath +
"/player_avatar_of_" +
GameManager.Instance().PlayerId + ".xml");
            object obj =
deserializer.Deserialize(textReader);
            var avi = (AvatarFromService)obj;
            foreach (var s in avi.aviDetail)
            {
                avatarList.Add(s.Name);
                Debug.Log(s.Name);
            }
            textReader.Close();
        }
        catch (Exception e)
        {
            Debug.Log(e);
        }
    }

      public string getHairType()
      {
            hairType = avatarList[0];
            return hairType;
      }

      public string getEyesType()
      {
            eyesType = avatarList[1];
            return eyesType;
      }

      public string getMouthType()
      {
            mouthType = avatarList[2];
```

```
            return mouthType;
        }

    void AttachHair(string hairStyleChosen)
    {
        Destroy(hChild);
        var currentPos = this.transform.position;
        if (hairStyleChosen == "menurambut1")
        {
            var hairStyle =
Instantiate(hairStyle1, currentPos,
Quaternion.identity);
            hChild = hairStyle;
            avatarList[0] = "rambut1";
            Debug.Log("hairstyle1");
        }
        else
            if (hairStyleChosen == "menurambut2")
            {
                var hairStyle =
Instantiate(hairStyle2, currentPos,
Quaternion.identity);
                hChild = hairStyle;
                avatarList[0] = "rambut2";
                Debug.Log("hairstyle2");
            }
            else
                if (hairStyleChosen ==
"menurambut3")
                {
                    var hairStyle =
Instantiate(hairStyle3, currentPos,
Quaternion.identity);
                    hChild = hairStyle;
                    avatarList[0] = "rambut3";
                    Debug.Log("hairstyle3");
                }
    }

    void AttachEye(string eyeStyleChosen)
    {
        Destroy(eChild);
```

```
        var currentPos = this.transform.position;
        if (eyeStyleChosen == "menumata1")
        {
            var eyeStyle = Instantiate(eyeStyle1,
currentPos, Quaternion.identity);
            eChild = eyeStyle;
            avatarList[1] = "mata1";
            Debug.Log("eyestyle1");
        }
        else
            if (eyeStyleChosen == "menumata2")
            {
                var eyeStyle =
Instantiate(eyeStyle2, currentPos,
Quaternion.identity);
                eChild = eyeStyle;
                avatarList[1] = "mata2";
                Debug.Log("eyestyle2");
            }
            else
                if (eyeStyleChosen ==
"menumata3")
                {
                    var eyeStyle =
Instantiate(eyeStyle3, currentPos,
Quaternion.identity);
                    eChild = eyeStyle;
                    avatarList[1] = "mata3";
                    Debug.Log("eyestyle3");
                }
    }

    void AttachMouth(string mouthStyleChosen)
    {
        Destroy(mChild);
        var currentPos = this.transform.position;
        if (mouthStyleChosen == "menumulut1")
        {
            var mouthStyle =
Instantiate(mouthStyle1, currentPos,
Quaternion.identity);
            mChild = mouthStyle;
```

```
            avatarList[2] = "mulut1";
            Debug.Log("mouthstyle1");
        }
        else
            if (mouthStyleChosen == "menumulut2")
            {
                var mouthStyle =
Instantiate(mouthStyle2, currentPos,
Quaternion.identity);
                mChild = mouthStyle;
                avatarList[2] = "mulut2";
                Debug.Log("mouthstyle2");
            }
            else
                if (mouthStyleChosen ==
"menumulut3")
                {
                    var mouthStyle =
Instantiate(mouthStyle3, currentPos,
Quaternion.identity);
                    mChild = mouthStyle;
                    avatarList[2] = "mulut3";
                    Debug.Log("mouthstyle3");
                }
    }

    void OnClick()
    {
        RaycastHit2D hit =
Physics2D.Raycast(Camera.main.ScreenToWorldPoint(
Input.mousePosition), Vector2.zero);
        if (Input.GetMouseButtonUp(0))
        {
            if (hit.collider != null)
            {
                if
(hit.collider.gameObject.name.ToLower().Contains(
"menumulut"))
                {
                    GameObject objM =
hit.collider.gameObject as GameObject;
                    Debug.Log(objM.name);
```

```
                    AttachMouth(objM.name);
                }
                else if
(hit.collider.gameObject.name.ToLower().Contains(
"menumata"))
                {
                    GameObject objE =
hit.collider.gameObject as GameObject;
                    Debug.Log(objE.name);
                    AttachEye(objE.name);
                }
                else if
(hit.collider.gameObject.name.ToLower().Contains(
"menurambut"))
                {
                    GameObject objH =
hit.collider.gameObject as GameObject;
                    Debug.Log(objH.name);
                    AttachHair(objH.name);
                }
                else if
(hit.collider.gameObject.name.ToLower().Contains(
"backbutton"))
                {

Application.LoadLevel("BeforeBattle");
                }
                else if
(hit.collider.gameObject.name.ToLower().Contains(
"savebutton"))
                {

WebServiceSingleton.GetInstance().ProcessRequest(
"edit_avatar", playerName + "|" + avatarList[0] +
"-" + avatarList[1] + "-" + avatarList[2]);

Debug.Log(WebServiceSingleton.GetInstance().query
Info);

WebServiceSingleton.GetInstance().ProcessRequest(
"get_player_avatar", playerName);
```

```
Debug.Log(WebServiceSingleton.GetInstance().Downl
oadFile("get_player_avatar", playerName));
                }
            }
        }
    }
}
```

Kode Sumber 9.15. Kelas `AvatarAttachment`

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using UnityEngine;
using System.Xml;
using System.IO;
using System.Xml.Serialization;

namespace ModulPertarungan
{
    public class TradingCardLoader :
MonoBehaviour
    {
        public GameObject myTradeGrid;
        public GameObject myTrunkGrid;
        public GameObject hisTradeGrid;
        public GameObject hisTrunkGrid;
        public GameObject myName;
        public GameObject hisName;
        private int totalDeckCostSize;
        private XmlDocument _xmlDoc;
        private XmlNodeList _nameNodes;
        private XmlNodeList _quantityNodes;
        private List<string> myTradeList;
        private List<int> myTradeQuantity;
        private List<string> hisTradeList;
        private List<int> hisTradeQuantity;

        void Start()
```

```
        {
            _xmlDoc = new XmlDocument();
            LoadTrunk("trunk_of_",
GameManager.Instance().PlayerId, myTrunkGrid);
            LoadTrunk("trunk_of_",
GameManager.Instance().FriendName, hisTrunkGrid);
        }

        void Update()
        {
        }

        public void AddToGrid(GameObject grid,
List<string> list)
        {
            foreach (string s in list)
            {
                NGUITools.AddChild(grid,
(GameObject)Resources.Load("DisplayCards/" + s,
typeof(GameObject)));
            }

grid.GetComponent<UIGrid>().Reposition();
        }

        public void LoadTrunk(string method,
string name, GameObject grid)
        {

WebServiceSingleton.GetInstance().ProcessRequest(
"get_player_trunk", name);

WebServiceSingleton.GetInstance().DownloadFile("g
et_player_trunk", name);
            List<string> list = new
List<string>();
            Boolean _isEmpty = false;
            try
            {
                TextReader textReader = new
StreamReader(Application.persistentDataPath + "/"
+ method + name + ".xml");
```

```
                _xmlDoc.Load(textReader);
                _nameNodes =
_xmlDoc.GetElementsByTagName("Name");
                _quantityNodes =
_xmlDoc.GetElementsByTagName("Quantity");
                for (int i = 0; i <
_nameNodes.Count; i++)
                {
                    for (int j = 0; j <
int.Parse(_quantityNodes[i].InnerXml); j++)
                    {

list.Add(_nameNodes[i].InnerXml);
                    }
                }
            }
            catch
            {
                _isEmpty = true;
            }

            if (!_isEmpty) AddToGrid(grid, list);
        }

        public void SendTradingRequest()
        {
            myTradeList = new List<string>();
            myTradeQuantity = new List<int>();
            hisTradeList = new List<string>();
            hisTradeQuantity = new List<int>();
            //MY CARD LIST FOR TRADING
            foreach (Transform t in
myTradeGrid.transform)
            {
                string s = t.name.Split('(')[0];


                bool is_distinguish = true;
                for (int i = 0; i <
myTradeList.Count; i++)
                {
                    if (myTradeList[i] == s)
```

```
                    {
                        is_distinguish = false;
                        myTradeQuantity[i]++;
                        break;
                    }
                }
                if (is_distinguish)
                {
                    myTradeList.Add(s);
                    myTradeQuantity.Add(1);
                }
            }

            //HIS CARD LIST FOR TRADING
            foreach (Transform t in
hisTradeGrid.transform)
            {
                string s = t.name.Split('(')[0];


                bool is_distinguish = true;
                for (int i = 0; i <
hisTradeList.Count; i++)
                {
                    if (hisTradeList[i] == s)
                    {
                        is_distinguish = false;
                        hisTradeQuantity[i]++;
                        break;
                    }
                }
                if (is_distinguish)
                {
                    hisTradeList.Add(s);
                    hisTradeQuantity.Add(1);
                }
            }
            TradeRequest tradingRequest = new
TradeRequest();
            tradingRequest.RequestedPlayer =
GameManager.Instance().FriendName;
```

```
            tradingRequest.SenderPlayer =
GameManager.Instance().PlayerId;
            tradingRequest.senderCards = new
List<CardRequest>();
            tradingRequest.requestedCards = new
List<CardRequest>();
            for (int i = 0; i <
myTradeList.Count; i++)
            {
                CardRequest c = new
CardRequest();
                c.Name = myTradeList[i];
                c.Quantity = myTradeQuantity[i];

tradingRequest.senderCards.Add(c);
            }
            for (int i = 0; i <
hisTradeList.Count; i++)
            {
                CardRequest c = new
CardRequest();
                c.Name = hisTradeList[i];
                c.Quantity = hisTradeQuantity[i];

tradingRequest.requestedCards.Add(c);
            }
            XmlSerializer serializer = new
XmlSerializer(typeof(TradeRequest));
            using (TextWriter writer = new
StreamWriter(Application.persistentDataPath +
"/trading_detail.xml"))
            {
                serializer.Serialize(writer,
tradingRequest);
            }
            try
            {
                string text =
System.IO.File.ReadAllText(Application.persistent
DataPath + "/trading_detail.xml");
                var encoded_string =
System.Text.Encoding.UTF8.GetBytes(text);
```

```
WebServiceSingleton.GetInstance().ProcessRequest(
"send_trade_request",
System.Convert.ToBase64String(encoded_string));

Debug.Log(WebServiceSingleton.GetInstance().respo
nseFromServer);
            }
            catch (Exception e)
            {
                Debug.Log(e);
            }

            Application.LoadLevel("HouseEditor");
        }

        void GoBack()
        {

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

Kode Sumber 9.16. Kelas `TradingCardLoader`

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using UnityEngine;
using System.Xml;
using System.IO;
using System.Xml.Serialization;
using ModulPertarungan;

public class TradingConfirmation : MonoBehaviour
{
    public GameObject myRequestedCardGrid;
    public GameObject hisOfferGrid;
    private XmlDocument _xmlDoc;
    private XmlNodeList _nameNodes;
```

```
    private XmlNodeList _quantityNodes;
    TextReader textReader;
    private List<string> myTradeList;
    private List<string> hisTradeList;

      void Start ()
    {
        _xmlDoc = new XmlDocument();

        LoadOffer("card_request_list_of_",
GameManager.Instance().PlayerId);
      }

      void Update ()
    {

      }

    public void AddToGrid(GameObject grid,
List<string> list)
    {
        foreach (string s in list)
        {
            NGUITools.AddChild(grid,
(GameObject)Resources.Load("DisplayCards/" + s,
typeof(GameObject)));
        }
        grid.GetComponent<UIGrid>().Reposition();
    }

    public void LoadOffer(string method, string
name)
    {
        myTradeList = new List<string>();
        hisTradeList = new List<string>();
        Boolean _isEmpty = false;
        try
        {
            XmlSerializer deserializer = new
XmlSerializer(typeof(CardRequestFromService));
            textReader = new
StreamReader(Application.persistentDataPath +
```

```
"/card_request_list_of_" +
GameManager.Instance().PlayerId + ".xml");
            object obj =
deserializer.Deserialize(textReader);
            CardRequestFromService cardReqList =
(CardRequestFromService)obj;
            foreach (var _card in
cardReqList.requestedCards)
            {
                for (int i = 0; i <
_card.Quantity; i++)
                {
                    myTradeList.Add(_card.Name);
                }
            }
            foreach (var _card in
cardReqList.offeredCards)
            {
                for (int i = 0; i <
_card.Quantity; i++)
                {
                    hisTradeList.Add(_card.Name);
                }
            }
            textReader.Close();
        }
        catch (Exception e)
        {
            Debug.Log(e);
        }

        if (!_isEmpty)
        {
            AddToGrid(myRequestedCardGrid,
myTradeList);
            AddToGrid(hisOfferGrid,
hisTradeList);
        }


myRequestedCardGrid.GetComponent<UIGrid>().Reposi
tion();
```

```
hisOfferGrid.GetComponent<UIGrid>().Reposition();

    }

    public void AcceptTradeRequest()
    {

WebServiceSingleton.GetInstance().ProcessRequest(
"accept_trade_request",
GameManager.Instance().TradeID);

Debug.Log(WebServiceSingleton.GetInstance().Downl
oadFile("get_player_trunk",
GameManager.Instance().PlayerId));
        Application.LoadLevel("HouseEditor");
    }

    public void DeclineTradeRequest()
    {

WebServiceSingleton.GetInstance().ProcessRequest(
"decline_trade_request",
GameManager.Instance().TradeID);
        Application.LoadLevel("HouseEditor");
    }

    void GoBack()
    {
        Application.LoadLevel("HouseEditor");
    }
}
```

Kode Sumber 9.17. Kelas `TradingConfirmation`

BIODATA PENULIS

Penulis, Zendra Anugerah Andromedha, lahir di kota Surabaya pada tanggal 25 Juli 1992. Penulis adalah anak pertama dari dua bersaudara dan dibesarkan di kota Surabaya, Jawa Timur.

Penulis menempuh pendidikan formal di SDN Kaliasin I Surabaya (1998-2004), SMPN 1 Surabaya (2004-2007), SMAN 2 Surabaya (2007-2010). Pada tahun 2010, penulis memulai pendidikan S1 jurusan Teknik Informatika Fakultas Teknologi Informasi di Institut Teknologi Sepuluh Nopember Surabaya, Jawa Timur.

Di jurusan Teknik Informatika, penulis mengambil bidang minat Rekayasa Perangkat Lunak dan memiliki ketertarikan di bidang pengembangan game, animasi, *software engineering*, dan *software maintenance*. Penulis dapat dihubungi melalui alamat email zendraanugerah@gmail.com