Fast Food Wars

A Computer Game Developed for Melange Computing Services

Prepared by

Abhishek Singh Rathore

Zichen Wang

Jeff Grandt

William Montgomery

CS 440 - Fall 2013

The University of Illinois at Chicago

Table of Contents

List of Tables

List of Figures

I. Project Description

1. Project Overview

Fast Food Wars is a turn-based strategy board game for 2 to 6 players that is played on a computer. Age recommendation for this game is ages 11 and up. This game places fast food franchise owners against each other in a war to conquer the city and its suburbs. Each player owns a certain franchise and battles their opponents to become the last one standing. This fast paced strategy game requires luck as well as intelligence to become the winner who will be named the most successful businessman.

Each player starts their board piece on their respective end of the hexagonal board and is given a certain amount of seed money. The board is a hexagonal game board that is modeled after the layout of a city. The center of the board represents high-priced, downtown locations and the area near the boundaries represent the suburbs where land is cheaper. Board pieces are designed to model a popular type of fast food restaurant chain, for example a burger or taco place. The player advances their piece one board block at a time. Along the way the player may choose to purchase the board spaces they land on or decline to buy them. Purchasing a board space represents the opening of a franchise. At the beginning of each turn, the player is given a “payout” (i.e. given game money) proportional to the number of franchises owned and the location of those franchises. Properties in the center of the city have higher payouts, but come at a higher initial cost and suffer from more competition from other players. This simulates real world competition between franchise owners.

Players can either move to an empty space and purchase it for face value or try to conquer an opponent’s space. When facing off with an opponent’s franchise, it is both luck and cash that determines the players fate. The storyline is as follows. The player who is attempting to conquer a board space sets up a franchise in that area. The more money invested in this particular franchise the more chance that this franchise will succeed. This is modeled by the player laying down a dollar amount, which is comparable to the price of real estate in that area. To be given an extra advantage, the player may add to this dollar amount up to 6 times a required investment amount specified on the space. The fate is decided by the roll of the die, depending on how many times the required investment the player has paid. For each payment of the specified investment amount, the player gets an extra side on the die. For example, if the player pays 3 times the investment amount specified on the board space and then rolls a one, two, or three, they will win the board space being battled over. In real terms, this represents the player’s franchise outperforming the other player’s franchise, causing the defending player’s franchise to shut down. If the player rolls a four, five, or a six, then the offensive attacking player loses the investment money he or she put down for the property and then the franchise closes, leaving the

defending player's franchise in place.

Gameplay continues until one player has completely vanquished all of their opposing players' franchises.

This game will be implemented using the Java programming language and will be initially developed for the Android operating system, with the intention that versions for other operating systems will be available in the future. The game can be played as a single player versus one or several AI players, as well as multiplayer versus up to 5 other players over the Internet.

2. The Purpose of the Project

The main focus of the project is to attempt to capture some of the gaming market that has been identified as being underserved. Melange must be able to develop a product that engages its users with competitive and lively gameplay and provide hours of entertainment to its end users. This game models the competitive edge that exists in the fast food franchise world. There is a constant struggle to be the most popular restaurant, open more and more businesses, and make the most money. This board game takes all the competitive edge of this aforementioned franchise world, boxes it up, and brings it to the user.

2a The User Business or Background of the Project Effort

This project is to create a strategy board game with the fast food franchise world as its model. The user will be provided with hours of entertainment and fun gameplay.

The situation of the fast food franchise world today is of great ubiquity and importance. The competition is fierce and only a few great restaurants can survive. This competition is the backbone of Capitalism and the central motivation in this game. Our proposed client, Funskool, Inc. believes that this type of competitive gameplay will draw their customers to purchase this computer game.

Considering the ubiquity of the computer-based game industry, this project will strive to gain great profits for our client. It is our clients understanding that a game of this type will hit the market with a heavy momentum and will achieve great sales. It is our prerogative then to make this product as user friendly as possible so that our client can achieve this goal.

2b Goals of the Project

The product being developed is a computer-based video game based on a fast food franchise competitive board game. The client knows that this game is unique and will accommodate their customers' desire to play fun and strategic board games on their computers.

The business being described in this game is that of the fast food industry. Fast food chains have turned to franchising to gain greater profit margins and expand territory. This game takes this principle and creates a fun strategy game based on it. The motivation behind this game is for the end user to be interested in this type of competitive game play in a knowledge domain each one of them has some familiarity with.

People want to be able to play fun games with their friends and families and would be interested to play as fun characters like tacos or a burger and fries.

2c Measurement

We expect to develop a product that is above a 90% satisfaction rate from end users, as measured by an independent third party.

We expect to develop a product that is relatively bug-free. A product will be considered relatively bug-free if a maximum of 1 high-priority bug fix is submitted in the month after the initial development phase.

We expect to develop a product that meets or exceeds all client expectations and eventually purchased.

3. The Scope of the Work

The initial release will consist of an Android client and a HTTP-based back end.

3a The Current Situation

For the past 15 years, Melange Computing Services has focused entirely on developing 3d Games. While successful, development costs are high and the market is both limited and saturated. There are currently no products that the company has developed that are playable on mobile devices or are developed with networked play as a primary objective. The development of Fast Food Wars would signal an initial foray into the market of cross-platform games. As such, while the company has a wealth of traditional software development resources, the company has few web resources available. It will be critical that the company devote some resources to developing this area.

3b The Context of the Work

The development of Fast Food Wars will require knowledge in two specific areas, fast-food restaurant industry and computer systems. The fast-food industry should be understood at a high level to be able to simulate what makes a fast-food chain successful and at the micro-scale to be able to simulate a particular location. In terms of computer systems, it is necessary to investigate the development of hardware and software that will be required to run the back end web services. This might include web servers, database server, load

balancers, and routing hardware. Mobile and some desktop clients (most notably Mac OS X) clients will have to interface with push notification systems. Experts with experience interfacing with these systems should be consulted. Additionally, since a goal of this project is to develop cross-platform services, a development framework must be investigated to minimize development duplication.

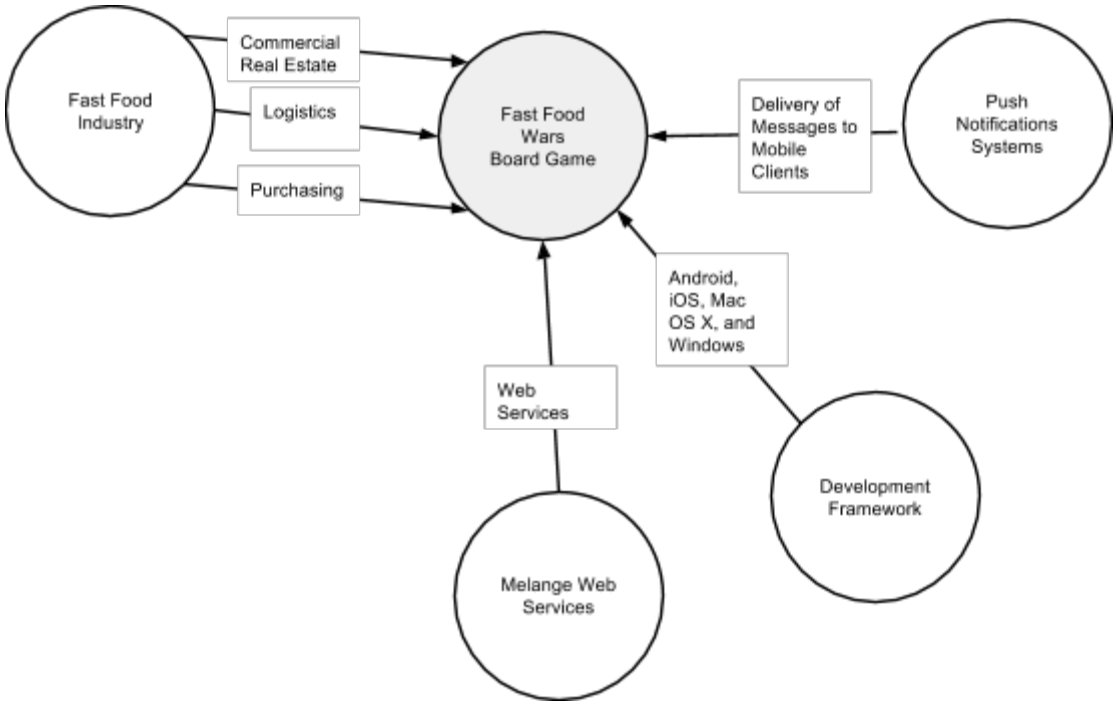

Figure 1 - Work Context

3c Work Partitioning

Event Name	Input and Output	Summary
Player Starts Game	User Event (In)	Displays splash screen.
Player Chooses Single Player Game	Game Selection (In)	A new game is started.
Player Chooses Multiple Player Game	Game Selection (In)	The web services are queried and user is added to active game or to new game.
Player moves in Single Player Game	User Move (In)	The player moves and the AI moves for computer players.
Player moves in Multiple Player Game	User More (In)	The player move and sends the move to the web services.

Player Wins Game in Single Player Mode or Multiple Player Mode	Scoreboard (Out)	The scores are displayed after each game.

Table 1 - Business Event List

3d Competing Products

There are many examples of competing products. In the realm of traditional board games, games as Monopoly© and Stratego© offer similar elements to Fast Food Wars. There are web versions of board games as well (i.e. Words With Friends© is a web version of Scrabble©). There is a smaller niche for this game, as it uses the paradigm of a board game, but is solely computer based.

4. The Scope of the Product

4a Product Boundary

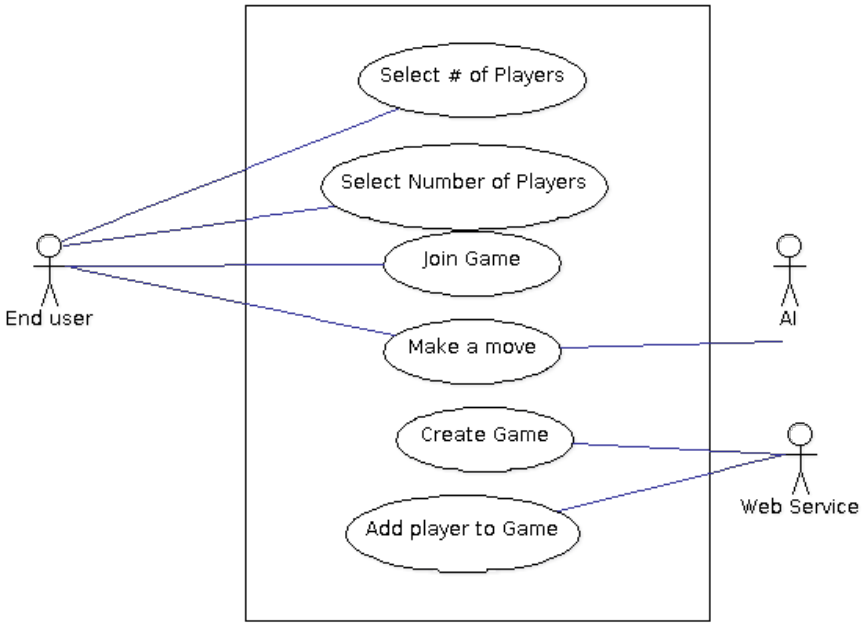

Figure 2 - Product Boundary

5. Stakeholders

5a The Client

Initially, the client will be developed in-house, for Melange, but the goal is to develop a relationship with an outside entity so that Melange can shift its focus to the development of software, rather than marketing and distribution.

5b The Customer

We have identified an interested party for the project, Funskool, Inc. Funskool is a joint venture between Indian tire giant, MRF, and Hasbro, Inc., the worldwide leader in children's and family leisure time products. After more than thirty years effort, Funskool has emerged as the largest toy company in India. This project will be successful only if we meet the needs of Funskool or similar company. The rapid rise of mobile devices has piqued the interest of Funskool.

5c Domain Expert

Fast Food Industry Experts

As this game is based on the theme of fast food industry, fast food industry experts are invited to participate in the whole process of game design and competition strategies design. Experts are expected to ensure the correctness and rationality from general ideas to specific implementation. All items presented in this game should originate from real life, and the design should take this into account. Furthermore, in-depth knowledge of fast food industry should be provided to the user in the game guides and helps.

To this end, we have identified George Forrest, who owns five McDonald's franchises in Charlotte, North Carolina. He can be contacted via LinkedIn at http://www.linkedin.com/pub/george-forrest/8/6b8/454 .

Software Interface Specialists

As this game is designed to interact with other software on mobile devices, software interface specialists will be consulted to ensure a smooth interaction between different software interfaces. For example, when users want to solve problems by phone calls, this game needs to interact with low level interfaces of mobile devices. Interaction with browser interfaces are needed when customers prefer online solutions. And this game will interact with network interfaces when customers are playing online games.

Network Experts

Because this game provides interaction between customers, network experts are invited to participate in the development of this game. They provide

optimized solutions adapting to different network infrastructures, which ensure best user experience. They also are responsible for providing optimized server side network solutions.

3D Engine Experts

Although this game is a tabletop game, strategies are main features other than fancy 3D effects. 3D effects allow this game to take advantage of the latest displaying technologies such as the overall quality and performance of the game.

Artificial Intelligence Experts

This game has player vs. player model and player vs. AI model. The AI system is designed to learn and estimate players' behavior with empirical learning algorithm. AI experts are invited to provide statistic based learning algorithm which enables the AI system can learn from experience as human do.

5d Hands-On Users of the Product

First Time User

First time user will be prompted with guides and helps. Helps will be provided at key steps of all procedures. The first time running this game, the user will be asked if the guides and helps should be turned on or off. User might choose to turn it off and and turn it on later in the game settings. If user guide and help information fails to provide an answer, users can click on a link to the Funskool online customer help center to access additional help online.

Once the user clicks the Funskool online customer help center link, the game will be temporarily suspended, and the user will be asked to choose if they want to solve their problem with phone call or online. If user prefers phone call and their devices are cell phones, phone calls will be made automatically. If users play this game on tablets or touch screen computers, or are not using their cell phones, only the phone number will be provided, and there will be no attempt to automate the phone call.

If users choose to solve their problems online, they will be redirected to the Funskool online help center using their default Internet browser on their device.

Users can also find help and guide information in Q&A sections of this game or refer to online discussion forum or game masters supported by Funskool.

Multi-Language Support

In order to fully exploit international game market, Funskool decided to provide different language versions of this game. Once installed, this game will automatically detect the language packages installed on devices (Language

Packages for Operating System of that device), and prompt users with options to download languages for this game according to installed language packages list.

Users also could download language packages not in that list. If there is no language package available for some users, they are encouraged to ask Funskool for language packages.

5e Maintenance Users

The maintenance users will be broken down into three categories. Users who select the phone option from the Funskool online customer help center link will be greeted by a Customer Support Representative (CSR). The CSR will be responsible for front line technical and user support through email, Twitter©, and chat rooms. Issues in either the web platform or the client software that cannot be resolved will be escalated to a Web Support Technician (WST) or a Client Support Technician (CST). Issues that cannot be resolved will be turned over to Support Developers.

6. Mandated Constraints

There are certain constraints that must be satisfied, although we intend to leave as much to the developers as possible.

6a Solution Constraints

Description: The product should run on any Android device version that runs version 4.0 (Ice Cream Sandwich) or later.

Rationale: While prior versions are prevalent, devices with version 4.0 or greater make up 65% of the Android marketplace and is rapidly becoming the baseline for devices.

Fit criterion: The application successfully runs on an Android device 4.0.x, 4.1.x, and 4.2.x.

6b Implementation Environment of the Current System

As this is a new application, there is no system in place. Melange does host servers in two location in the United States, in Salt Lake City, Utah and in Alexandria, Virginia. It is expected that these facilities will host the web portion of the solution. These services have redundant power systems with battery and off-grid, diesel backup with fuel to supply the facility for 7 days in the event of a disaster. The facilities also have redundant backup Internet access from Tier-1 providers. Security, cooling, and cable management are state of the art.

End users will obtain the application through the Google Play Store, accessible through all Android devices. Guidelines can be found in the bibliography.

6c Partner or Collaborative Applications

The system will interact with the Google Cloud Messaging system (GCM). The GCM service allows messages to be sent to the application when the application is in the background or not running. In the future, iOS and Mac OS X clients will use the Apple Push Notifications Service, while Windows clients will use Windows Azure.

6d Off-the-Shelf Software

For the development of the Android client, it is suggested that a framework that is capable of generating code for multiple operating systems be utilized. It is recommended that QT and Processing be considered, although this is by no means an exhaustive list of frameworks that could be useful.

For the web API, it is suggested that a combination of Apache web server and the mySQL database be used, with a suitable open-sourced framework for the HTTP connections. Possible frameworks, such as Rails-API (Ruby), Node.js, or Slim (PHP), should be considered. Since Web technologies change so fast, it is imperative that the skillset of the team that is tasked with building this product be taken into consideration.

In both instances, Git should be used for source control management. Melange currently has a Gitolite server for hosting Git repositories. It is anticipated that this can be used with little to no configuration.

Redundant, commodity servers should be sufficient for the initial release. Additionally, sufficient workstations or laptops should be provided to the entire development and support team.

6e Anticipated Environment

This product will be used on a wide variety of devices so the user interface must be fairly high contrast.

Mobile devices can lose and regain Internet connectivity often. To this end, any loss of Internet connectivity should not change the state of the game.

Push notifications should not be considered reliable. The system should provide a mechanism to sync game state.

6f Schedule Constraints

The initial phase of this project should be completed within 14 weeks, as it is anticipated that the window of student developers is a single semester. If the project is not finished within this time period, the project should be considered a complete loss as it is unlikely that future students will take on the project.

From a business standpoint, the market is already crowded, and Melange is late to the game. Thus a marketing campaign has been linked to the successful release of the project. Melange has committed to investing significant resources to this campaign. Certain parts of the campaign are not time critical, but the loss from other parts can be significant. To this end, it is imperative that a customer release be ready within 20 weeks of project start.

6g Budget Constraints

There is currently no budget for the development of the project, as student knowledge and labor will be utilized throughout the project. However, hosting and supporting the web services has be estimated as follows:

Description	One Time Price	Monthly Price	First Year Total
Hardware	$5,000.00		$5,000.00
Connectivity		$100.00	$100.00
Technical Support		$1,000.00	$12,000.00
		Total	$17,000.00

Table 2 - Initial Budget

7. Naming Conventions and Definitions

7a Definitions of Key Terms

board - The hexagonal board where all play takes place.

computer player - A player controlled by the computer AI.

HTTP - Hypertext Transfer Protocol. A protocol for sending data over the Internet.

human player - A player controlled by an end-user.

JSON - Javascript Object Notation. A method of encoding data commonly used for transmitting over HTTP.

MWS - Melange Web Services - refers to any web services hosted by Melange.

MWS team - The web team responsible for development, deployment, troubleshooting, and maintenance of MWS.

mobile device - Any small, handheld computing device, typically having a display screen with touch input and/or a miniature keyboard and weighing less than 2 pounds. For the first release, Fast Food Wars software will specifically be developed for Android mobile devices.

Outage Level - An outage level corresponds to any situation wherein the system is not able to perform as intended either through hardware, software, or configuration faults. The system outage will be divided into four levels ranging from A to D, where A is the highest level of fault and D is the lowest level.

player - A user of the software.

push notification - Any message sent through a third-party server that can be delivered to the device when the software is either running or not running.

7b UML and Other Notation Used in This Document

This document uses the ISO standard UML version 2.4.1. A link to this specification is listed in the bibliography. Any departure from this specification will be listed here.

7c Data Dictionary for Any Included Models

As models have not been developed, this section is a work in progress.

8. Relevant Facts and Assumptions

8a Facts

85% of Americans play video games.

The average age of the most frequent video game buyer is 35 years old.

36% of gamers play on their smartphones.

Purchases of mobile games represented 40% of the video game market in 2012.

8b Assumptions

Mobile games will become an increasingly important area for game development.

Tabletop games are a natural fit for mobile devices.

II. Requirements

9. Functional Requirements

The system shall have the ability to create a single player game.

The system shall have the ability to create a multiplayer game with players across the Internet.

The system shall have the ability to load user profiles from a web service.

The system shall have the ability to save an ongoing war.

The game shall have the ability to accept input from a touchscreen.

To maintain a whimsical aspect amidst the strategic game play, the game shall show an animation whenever a user gets a good deal or makes a reasonably beneficial bargain to acquire assets.

As most online games, a limitation that users of Fast Food Wars will face is that an ongoing war can only be saved after a checkpoint. If a player opts to quit the game before reaching a checkpoint, all their money and acquisitions will be lost, forcing them to start over.

The system shall offer the ability to download and install upgrades.

10. Data Requirements

The focus of this product is entirely on the Fast Food Market and is implemented as an online game giving a general sense of how the real world market operates.

The business subject matter here is mainly how to maximize daily profit of various fast food chains when opening one or several locations. Users must research the feasibility of opening in a new location by identify potential customers based on the profile of the neighborhood, and by evaluating the competition.

The game extrapolates all the data related to how fast food operations are carried out to lead to a successful business venture by recognizing a good opportunity to buy a piece of land (a spot on the board), making best use of monetary assets, and being constantly aware of the rival fast food chains operating around the area.

11. Performance Requirements

11a Speed and Latency Requirements

Any interface between a user and the automated system shall have a maximum response time of 3 seconds.

The response shall be fast enough to avoid interrupting the user's flow of thought.

Newly acquired assets including money should be shown in the users profile within two seconds of the acquisition.

11b Precision or Accuracy Requirements

All currency should be represented with two decimal places.

11c Capacity Requirements

This system should allow a maximum number of ten users per game for the off-line model.

The system should allow a maximum current gaming session of 10,000. Each gaming session contains only one game and up to ten users. A gaming session could be either in pre-gaming stage, gaming stage, or post-gaming stage.

End-User response time should be less than 100 microseconds when there are 1,000 gaming sessions, 200 microseconds when there are 5,000 sessions, and 300 microseconds when there are 10,000 sessions.

This system should allow a concurrent logon request at a minimum of 5,000 per minute, and a concurrent gaming session creation request at a minimum of 1,000 per minute.

12. Dependability Requirements

12a Reliability Requirements

An synchronization mechanism should protect the data consistency between end user system and remote server end system. In the event of discrepancy between client and server, the server should be considered authoritative.

A server side archive mechanism should protect gaming data and user data during system outage and exception.

A backup system should automatically switch on-line when the major system is experiencing unexpected outage or high system delay.

Backup power should be provisioned at all data centers to run servers for up to 36 hours in the event of extended power outage.

Backup Internet connectivity should be provisioned to accommodate traffic in the event of loss of connectivity from primary provider. The link should be able to handle a load of at least 110% of the maximum traffic seen in the prior month of service.

12b Availability Requirements

System outage Level A describes a major failure of the core system. The entire system is affected at this level and the system is unavailable to end users. Level A outage should occur less than 1 time per year and for less than 15 minutes per occurrence.

System outage Level B describes a high system delay of the core system. The entire system is affected at this level. Any degradation of average response time to a level of 400% of the average response time over the past month should be considered a Level B outage. Level B outage should occur less than 10 times per year and for less than 1 hour per occurrence.

System outage Level C describes a major failure of one or several subsystems. System outage level C should be less than 100 times for each year and for less than 1 hour per occurrence.

System outage level D describes a high system delay of one or several subsystems. System outage level D should be less than 1000 times for each year and for less than 1 hour per occurrence.

12c Robustness or Fault-Tolerance Requirements

When the end user software loses connectivity to the remote server, it should inform users that connection was lost and automatically reconnect to the remote server when Internet connectivity is restored.

As data is distributed, data inconsistency may occur. The system should automatically correct data inconsistency according to archived data at server end.

Repeated failure to synchronize is an indication of data corruption in the data store. The system should inform system operators and should provide a method for restoring known good data.

13. Maintainability and Supportability Requirements

13a Maintenance Requirements

Maintenance activities are divided into three sub-categories, i.e., corrective maintenance, adaptive maintenance, and perfective maintenance. Corrective maintenance is needed when a specific function model in the system does not

work. Adaptive maintenance is needed when external changes require a change to the system. Perfective maintenance iis the 'maintenance' phase of the system. There are many reasons for maintaining a system that fall into the categories given above.

- An error / bug is serious enough to need fixing.
- A new business process needs to be incorporated.
- A security vulnerability in the system has been found and needs patching .
- A user has identified how the system could be improved.
- The hardware or network is being improved and so the system should take advantage of that.

These activities should only require system outage of less than two hours for applying update patches to the server end system. The scope that is affected by updating should be restricted to necessary sub-models or sub-regions to fix existing system defects. The changes should only affect other sub-system or sub-regions that are necessary to apply update patches. The code that is related to a specific problem or change should be documented and easy to understand and locate. The changes should be quickly verified by system operator, preferably in isolation. The changes also should be at low risk of breaking existing features. And if update patches unexpectedly affect other sub-systems, they should be easy to detect and diagnose.

These activities should only include changes for external influence or strategic changes within the company; e.g, the government recently changed the tax rate, a bank decides to offer a new mortgage product, which will have to be included in the system so that mortgage interest and payments can be calculated; the company has introduced an online system for customers to place orders, which needs to be integrated into their normal ordering system. The code that is related to a specific problem or change should be documented and easy to understand and locate. The changes should be quickly verified by system operator, preferably in isolation. The changes also should be at low risk of breaking existing features.

The system should be adapted to these types of changes within one month, and scheduled system outage should be less than 12 hours.

These activities should include minor changes to improve the system, and require no scheduled system outage and only cause a slight increase in system delay. Those changes are needed when the end user find tweaks or minor improvements that could be made to improve the way the system works. Examples of making the system more perfect includes a better data input screen or form, a more advanced help system, tweaks to the code so it is more responsive, reorganising data sets within a database so they can be searched faster or use less storage, and providing shortcut commands that experts can use instead of the slower standard menu system. These tweaks are not major enough to prompt a major enhancement, so the maintenance team can improve

the system to suit.

The Maintenance Document should be updated and delivered before each version of system release. The document should be in easy understanding fashion.

The maintenance team should perform the day-to-day maintenance of both end user system and remote server system, which includes performing backups, upgrading obsolete systems (both hardware and software), among other activities.

There should be system administrators who are in charge of duties including managing user access to the system, managing the network and operating system, and other system related duties.

There should be system analysts that will investigate the system for any issues. When there are maintenance requests, the system analysts are the persons who will research the request to see if it is feasible.

There should be system developers that will perform any development duties, i.e., programming, for any maintenance task.

The system design should follow the object oriented programming paradigms. The maintenance index of this system should be lower than 10 and WMC (Weighted Methods Per Class) >100, CBO (Coupling Between Object Classes) > 5, RFC (Response for a class) > 100, RFC > 5*NOM (Number of Methods for a Class), NOM > 40 .

13b Supportability Requirements

Technical support group should be in three levels, A, B, and C. Level A support group should log calls and provide basic troubleshooting. Level B support group should be composed of system experts who are able to solve functional issues. Level C support group should be composed of engineers of the system, programmer, and administrator of the system.

Level A support are call center personnel,who ask the user basic symptom questions in order to determine the user's issue. These support personnel concentrate on issues regarding desktop computers, laptops, routers, etc. , and should have A+ certification. If the level A support group cannot fix the issue, the issue should be escalated to level B support group.

Level B support are system experts who possess complete knowledge of usage and purpose of sub-systems and models. They are responsible for providing usage guide and for pinpointing problems caused by misusing or misunderstanding of the system.

Level C support are the engineers of the software. If a database issue cannot be

solved by the first two levels of support, the issue will be given to the administrator of the database to solve the issue.

13c Adaptability Requirements

The user end system should be compatible with mainstream operating systems such as Windows, Mac OS, Ubuntu, and Chrome OS. System updates and patches should be automatically downloaded and installed.

The installer of user end system should be able to detect the default language of operating system and automatically switch to that language. It also should allow user to reset language options. If required software, operating system component/ service, common language runtime library is missing or turned off, the installer should notify user where and how to download and install updating packages, or turn up system service.

End user system should be compatible with non-English input methods.

13d Scalability and Extensibility Requirements

System capacity will need to be increased when adding new features. When adding new features, existing models that have no interaction with new models should not be affected, no latency increment should be seen during and after the updating stage. When increasing system capacity, core system should not experience heavy latency increment, and literal subsystems should not be affected.

The product shall be capable of processing the existing 100,000 customers. This number is expected to grow to 500,000 customers within three years.

The product shall be able to process 5,000 gaming sessions concurrently.

14. Security Requirements

14a Access Requirements

Only the product developers have the rights to make any kind of changes or modifications in the gameplay and the database.

The network administrator is the only one responsible for the maintenance and with modification permissions of the Server.

A user can make changes allowed by the settings menu to fit their gameplay needs.

Any user can update the Product version on their device by downloading the most recent release with a high speed internet connection.

All data included in the gameplay shall be public to anyone who choses to play the game but should be restricted to the rest of the public.

Only users can access the support option to register themselves online with a username, password and other information and ask the tech support team for online help.

To maintain security, the database with saved game data and user profiles is kept only on the server side with limited access to the users.

A user can view other user profiles but cannot access them to make changes in any way.

14b Integrity Requirements

A data protection and synchronization sub-system should ensure the data is not modified, altered, or deleted without authorization in either storage or in transit. This sub-system should be able to detect any unauthorized modification of data and must yield a security-related event. It also should be able to identify the originator of any information before that information is used in any restricted function of the information resource. Plus, this sub-system must be able to log any attempt by the administrator to authorize any user to bypass the administrator-configured data integrity controls. It also performs data integrity checks to protect data consistency. When data integrity checks fail, this sub-system must reject the end user data.

Maximum percentage of data files/records corrupted per unit time is 10MB.

Maximum percentage of messages corrupted 1%.

Maximum percentage of programs corrupted per unit time 1%.

14c Privacy Requirements

Before installation, the installer must display and require users' consent of end user privacy policies agreement. The installation should be stopped if end users disagree with those policies. The agreement must inform users the nature and purpose of the data will be collected by the system. It also needs to ask users to comply with privacy policies.

This system should be in accordance with laws related to privacy regulations. All end users' private data should not be used for any commercial purposes. A privacy protection sub-system must provide protection to end users data from risk of theft and third-party vulnerabilities. This sub-system should be able to identify privacy risk at different levels. A list of potential risks should be generated using available project information and requirements. It also should provide basic privacy risk analysis tools. After risks have been identified, system

data will be need to be evaluated for their probability of occurrence and their potential impact on the project or system and then be classified into different risk levels ranked by importance based on the probability of occurrence and degree of impact.

14d Immunity Requirements

The system should have an immunity mechanism or a sub-system protecting itself from malicious programs or sabotage activities. This sub-system should prevent malicious programs from destroying or damaging data and applications, unauthorized users or programs from accessing restricted data/services and automatic player agent program or robotic player from participating in the game.

The minimum percentage of malicious programs identified should be equal to or greater than 80%.

The minimum percentage of malicious programs prevented from causing infection should be equal to or greater than 90%.

The minimum percentage of malicious programs cured should be equal to or greater than 90%.

15. Usability and Humanity Requirements

15a Ease of Use Requirements

The game shall be fairly easy yet competitive for anyone over 15 years of age.

The product does not require special market analysis or is in no way intended just for Fast Food chain owners / professionals.

The product, once downloaded. should be ready to use after just one read of the “About The Game” page.

Any user should be able to continue from their last saved checkpoint without remembering anything about their previous gameplay session.

The game should be understandable without any specific detailed knowledge about business management.

Any user familiar with smartphone/ PC/ tabletop gaming should be able to play the game carrying out basic operations, which make up for more than 60 percent of the game.

Some users might have to spend a little time reading or watching videos related how the market works in order to be successful, and should in no way be cumbersome for the users.

15b Personalization and Internationalization Requirements

The product shall comply with the buyer's cultural and lingual standards.

The game installation should allow users to choose a language that best suits their cultural aspects.

The UI of the game should be optimized for spelling preferences and translated common idioms and phrases to appeal to the international buyer market.

The game should support a customizable interface with the option to select different icons and colours.

15c Learning Requirements

There is a slight learning curve to the game. The user must learn how to operate within the game (where and how to move and various rules) and have a general knowledge of how retail chains function and a basic understanding of the concept of business management to be a successful player.

For a user with basic knowledge of the domain, the game should be easily playable after a cursory reading of the "About The Game" page to get familiarized with the rules.

For a user with minimum or no knowledge of the domain whatsoever, the product should provide tutorials showing how to proceed.

The product shall have an option to enable novice user to read an article written just for the purpose of getting an understanding of what the fast food market is all about and view special videos pertaining to the gameplay and basic information provided by actual domain experts who are not a part of the development team.

The users with marketing/management experience should be able to start playing the game almost immediately.

The users who are not familiar with the domain shall be patient and spend a little time viewing the videos and reading the articles.

15d Understandability and Politeness Requirements

The game should use the language of operation in such a way that the words, phrases, buttons and the various symbols are intelligible not only to users with domain knowledge but also those without any knowledge.

The development details and design constructs should be hidden from the users in a refined manner.

The distinction between the icons, gameplay options and other in game

functionalities should be made absolutely clear.

The product should not have any ambiguities.

15e Accessibility Requirements

The product shall be usable by visually impaired people by incorporating a fully functional voice prompt option that may guide the disabled.

The product shall conform to any Disability acts prevailing in the regions recognized as the potential target market.

15f User Documentation Requirements

The Document Editor is responsible for keeping all the documents up to date and check if they conform to the standards.

The documentation for the product shall include a help manual apart from the technical specification.

The technical specification should specify the technical requirements for the game to be used by the users.

15g Training Requirements

The training plan will cover who will be trained, what will be covered, and how it will be covered. The environments are classes that contain 15 to 25 trainees, one-on-one training, and self-guided training. The environment also includes who will be conducting the training. The trainers should be able to answer questions about the system. Specify the delivery method of the training is the last part of training plan. It defines how the training will be delivered, i.e. lecturing to the trainees or hands-on laboratory exercises. If users have questions or an issue with the system, the user needs to contact their technical support staff, which should be a standard section of this lesson. The technical support staff helps users and makes sure the system runs the way it was designed.

16. Look and Feel Requirements

16a Appearance Requirements

The user interface shall utilize bright colors that resemble those of fast food chains. Heavy use of bright yellow, orange, and red is encouraged.

Icons used in the game should be similar to, but not exact copies of, actual chain logos.

Inclusion of a character resembling the Hamburglar is highly encouraged.

16b Style Requirements

The product shall evince the experience of dining at a fast-food restaurant.

The user interface shall appear like a fast food menu board.

17. Operational and Environmental Requirements

17a Expected Physical Environment

The client application will be used on mobile applications with possibly intermittent Internet connectivity.

The web server will be located in geographically distinct data centers to provide redundancy. As such, provisions should be made for the maintenance and service of remote servers.

17b Requirements for Interfacing with Adjacent Systems

The product should support push notifications via the Apple Push Notifications Service.

The product should support push notifications via the Google Cloud Messaging service.

The product should support push notifications via the Windows Azure system.

In order to interface with these systems, the system should be able to issue HTTP requests with a JSON payload.

17c Productization Requirements

The product will be distributed by the Google Play web-based store. As such, the product should adhere to all Google Play rules and regulations including and such good practices.

While the initial product will only be offered on Android, the long-term goals include the release of an iOS client. As such, the development of the Android client should be led with an eye on the iOS market. This will allow for smoother roll-out of clients.

17d Release Requirements

Critical bug fixes should be released within 72 hours of confirmation.

Minor bug fixes and minor enhancements should be released on a bi-weekly schedule.

Major improvements and features should be released on a quarterly basis.

18. Cultural and Political Requirements

18a Cultural Requirements

The product should be allow for customization to allow for cultural specific franchises. For example, a hamburger franchise might be offensive to certain groups.

The product should allow for regional customization to allow for different cultural practices in terms of money lending and interest.

The product should support worldwide currencies.

18b Political Requirements

The product shall utilize as much open-source software as possible.

The product shall not be customized for countries on the United States State Department's list of State Sponsors of Terrorism.

19. Legal Requirements

19a Compliance Requirements

The product should be implemented so as not to use imagery, logos, or names that might infringe against an existing company.

The product should safeguard any identifiable personal information.

All personal information should protected in accordance with data security laws.

19b Standards Requirements

The product shall comply with the W3C Web Content Accessibility Guidelines.

III. System Models

In this section, we present the system design of Fast Food War. At the requirement stage, we describe the system in terms of functional and nonfunctional requirements, scenarios, and use cases. Our goal during requirements was to present the system in a way that our clients would be able to understand and depend on, which further facilitated the communication between developer and clients. In this section, we focus on the solution domain, which is the concern of the program developers. Hence, we adopt class diagram, sequence diagram, design object model, and package diagram, which

can better describe the inner structure of our system requiring high level programming knowledge. Now that the requirements pertaining to this project are solidified the focus moves on to the inner workings of the system that will actually be implemented.

We adopted refined scenario and use case models, class diagrams, and refined sequence diagrams, since we believe this combination is sufficient to guide developer to implement the whole system.

20. Scenarios

Use Case	InstallApplication
Actors	Initiated by End User Google Play store
Flow of Events	1. The End User selects install. 2. The system provides an executable to download. 3. The End User downloads and installs the executable.
Entry Condition	The End User has opened the Google Play application and navigated to the Fast Food Wars page.
Exit Condition	The End User has installed the Fast Food Wars client.

Table 3 - InstallApplication Scenario

Use Case	ShowMenu
Actors	Initiated by End User
Flow of Events	1. The End User opens the application. 2. The System displays a menu with options to play a new single player game or new multiplayer game, manage the user account, or exit the application.
Entry Condition	The End User has downloaded and installed the application from the Google Play store.
Exit Condition	The End User has viewed the available options.

Table 4 - ShowMenu Scenario

Use Case	CreateAccount
Actors	Initiated by End User
Flow of Events	1. The end user chooses to create an account from the Manage Account menu. 2. The system provides a form for the End User to fill out. 3. The End User fills out the form. 4. The system processes the form. [optional] 5. The system rejects the form if there are any errors. [optional] 6. The system displays a partially filled out form. [optional] 7. The user corrects any errors and resubmits. 8. The system creates an account within the system. 9. The system displays a message indicating the account was successfully created.
Entry Condition	The End User has opened the application and has selected “Manage Account” from the menu.
Exit Condition	The End User has an account in the system and has viewed a confirmation.

Table 5 - CreateAccount Scenario

Use Case	ManageAccount
Actors	Initiated by End User
Flow of Events	1. The end user chooses to manage the account from the Manage Account menu. 2. The system provides a form for the End User to fill out with the user’s current information filled out. 3. The End User fills out the form. 4. The system processes the form. [optional] 5. The system rejects the form if there are any errors. [optional] 6. The system displays a partially filled out form. [optional] 7. The user corrects any errors and resubmits. 8. The system updates the account within the system. 9. The system displays a message indicating the account

	was successfully updated.
Entry Condition	The End User has opened the application and has selected “Manage Account” from the menu.
Exit Condition	The End User has an updated account in the system and has viewed a confirmation.

Table 6 - ManageAccount Scenario

Use Case	DeleteAccount
Actors	Initiated by End User
Flow of Events	1. The end user chooses to delete an account from the Manage Account menu. 2. The system asks for confirmation with a warning that the action is irrevocable. 3. The End User clicks the button. 4. The system processes the account and removes any personally identifying information from the database. 5. The system displays a confirmation that the account has been deleted.
Entry Condition	The End User has opened the application and has selected “Manage Account” from the menu.
Exit Condition	The End User has an account in the system and has viewed a confirmation.

Table 7 - Delete Account Scenario

Use Case	CreateSinglePlayerGame
Actors	Initiated by End User
Flow of Events	1. The End User selects “Play Single Player Game”. 2. The system presents the End User with game options, including the number of AI opponents the End User wishes to play against. 3. The End User selects the number of AI opponents. 4. The system creates a game and waits for opponents to join. 5. The system creates AI opponents for this game and

	automatically joins those opponents into the game. 6. The system displays the playing board.
Entry Condition	The End User has opened the application and is viewing the main menu.
Exit Condition	The End User has started a new game with AI opponents.

Table 8 - CreateSinglePlayerGame

Use Case	CreateMultiplePlayerGame
Actors	Initiated by End User Web Player
Flow of Events	1. The End User selects “Play Multiple Player Game”. 2. The system presents the End User with game options, including the number of opponents the End User wishes to play against. 3. The End User selects the number of opponents. 4. The system creates a game and waits for opponents to join. 5. A number of Web Players join the game. 6. The system displays the playing board.
Entry Condition	The End User has opened the application and is viewing the main menu.
Exit Condition	The End User has started a new game with web opponents.

Table 9 - CreateMultiplayePlayerGame Scenario

Use Case	JoinMultiplePlayerGame
Actors	Initiated by End User Web Player
Flow of Events	1. The End User selects “Join Multiple Player Game”. 2. The system searches for games that need additional players. 3. The system displays a list of those games to the user. 4. The user selects a game to join. 5. The system joins the End User to the game. [optional]

	6. The system waits for additional players to join. 7. The system displays the board.
Entry Condition	The End User has opened the application and is viewing the main menu.
Exit Condition	The End User has started a new game with web opponents.
Notes	Includes login and load user data use cases

Table 10 - JoinMultiplePlayerGame Scenario

Use Case	ShowScoresSinglePlayer
Actors	Initiated by End User
Flow of Events	1. The End User or some other user performs an action that terminates the game. 2. The system displays a list of the scores from the previous game.
Entry Condition	The End User is in a single player game
Exit Condition	The End User has viewed the scores from the game.

Table 11 - ShowScoresSinglePlayer Scenario

Use Case	ShowScoresMultiplayer
Actors	Initiated by End User
Flow of Events	1. The End User or some other user performs an action that terminates the game. 2. The system displays a list of the scores from the previous game.
Entry Condition	The End User is in a multiplayer game
Exit Condition	The End User has viewed the scores from the game.

Table 12 - ShowScoresMultiplayer

21. Use Case Model

21a Overall Use Case

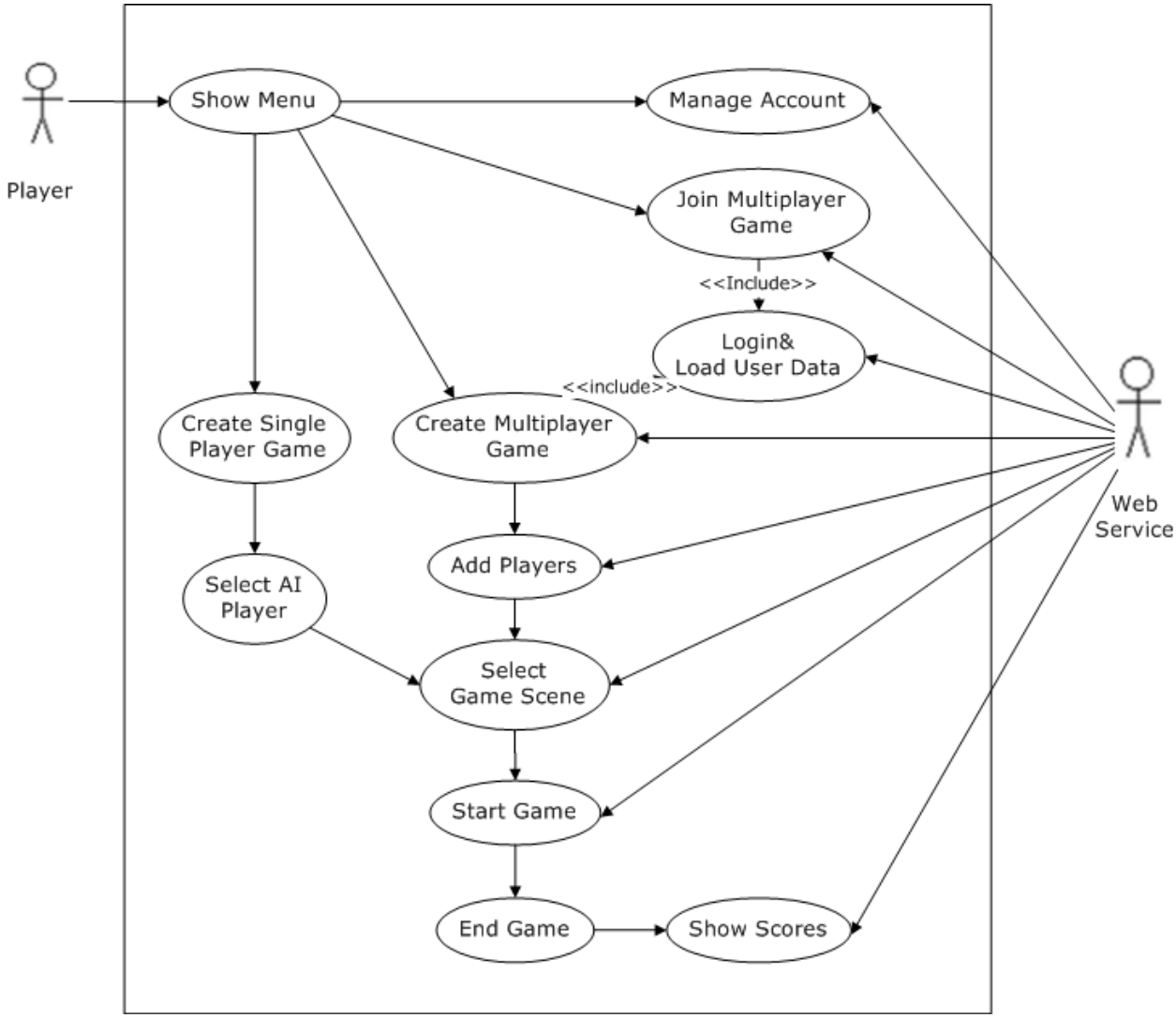

Figure 3 - Overall Use Case Diagram

21b Game Play Use Case

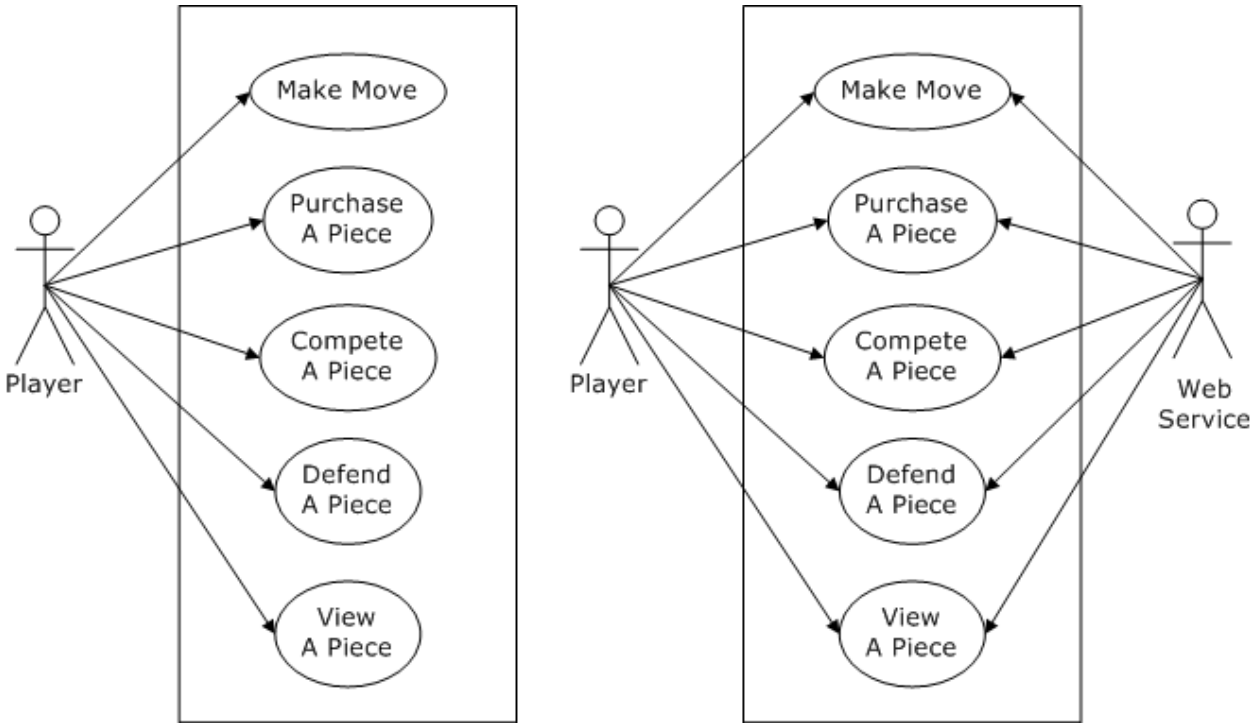

Figure 4 - Game Play Use Case

22. Object Model

22a AndroidClient Model

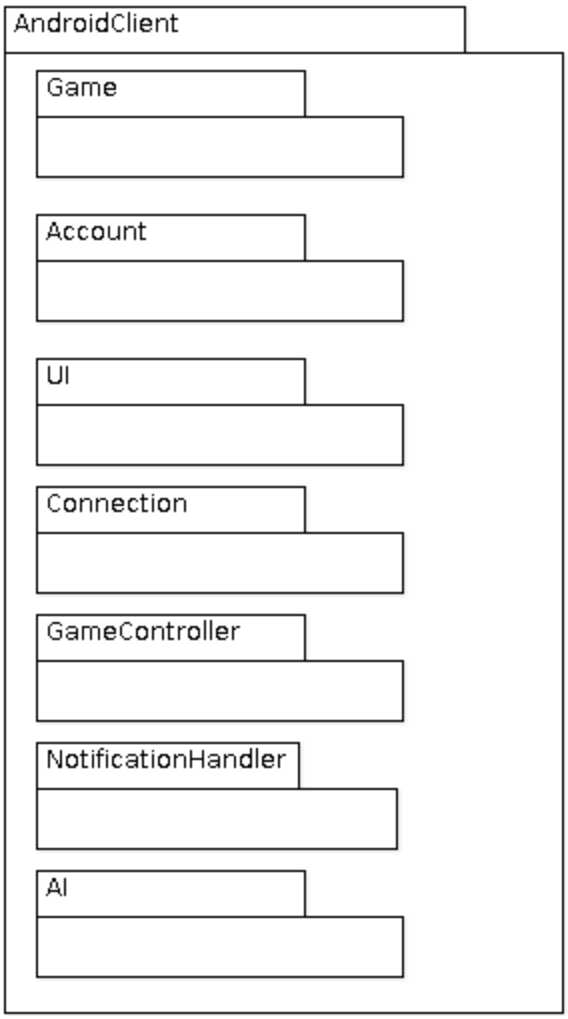

Figure 5 - AndroidClient Model

The Android client will be divided into the following subsystems:

- **Game** - The Game subsystem is responsible for the handling the representation of the current game state. This subsystem is analogous to the Model layer in MVC.
- **Account** - The Account subsystem is responsible for maintaining local account information and communicating through the Connection subsystem to verify and create user accounts.
- **UI** - The UI subsystem is responsible for displaying the current game state to the user and interacting with the user. This subsystem is analogous to the view layer in MVC.
- **Connection** - The connection subsystem is responsible for establishing a

connection to the web server, as well as packaging data for transmittal and unpacking results.

- **Game Controller** - The Game Controller subsystem provides a layer of abstraction between the UI and the model of the Game. This layer is analogous to the Controller layer in MVC.
- **Notification Handler** - The Notification Handler subsystem controls events triggered from the reception of push notifications. When receiving a push notification, the notification handler will send a request for the Game to sync with the web server to obtain new plays.
- **AI** - The AI subsystem is responsible for analyzing the current state of the board and making plays on behalf of AI users.

22b WebServer Model

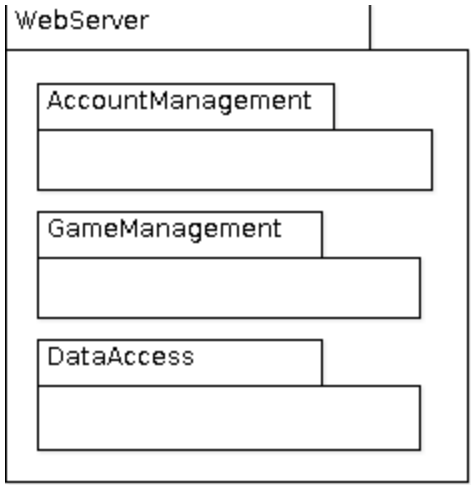

Figure 6 - WebServer Model

The WebServer subsystem is composed of three subsystems:

- **AccountManagement** - The AccountManagement subsystem is responsible for creating, updating, verifying, and deleting user accounts.
- **GameManagement** - The GameManagement subsystem is responsible for maintaining and updating game state for online games.
- **DataAccess** - The DataAccess subsystem is responsible for translating updates and data accesses into valid requests and maintaining a connection with the database.

22c Refined Object Model

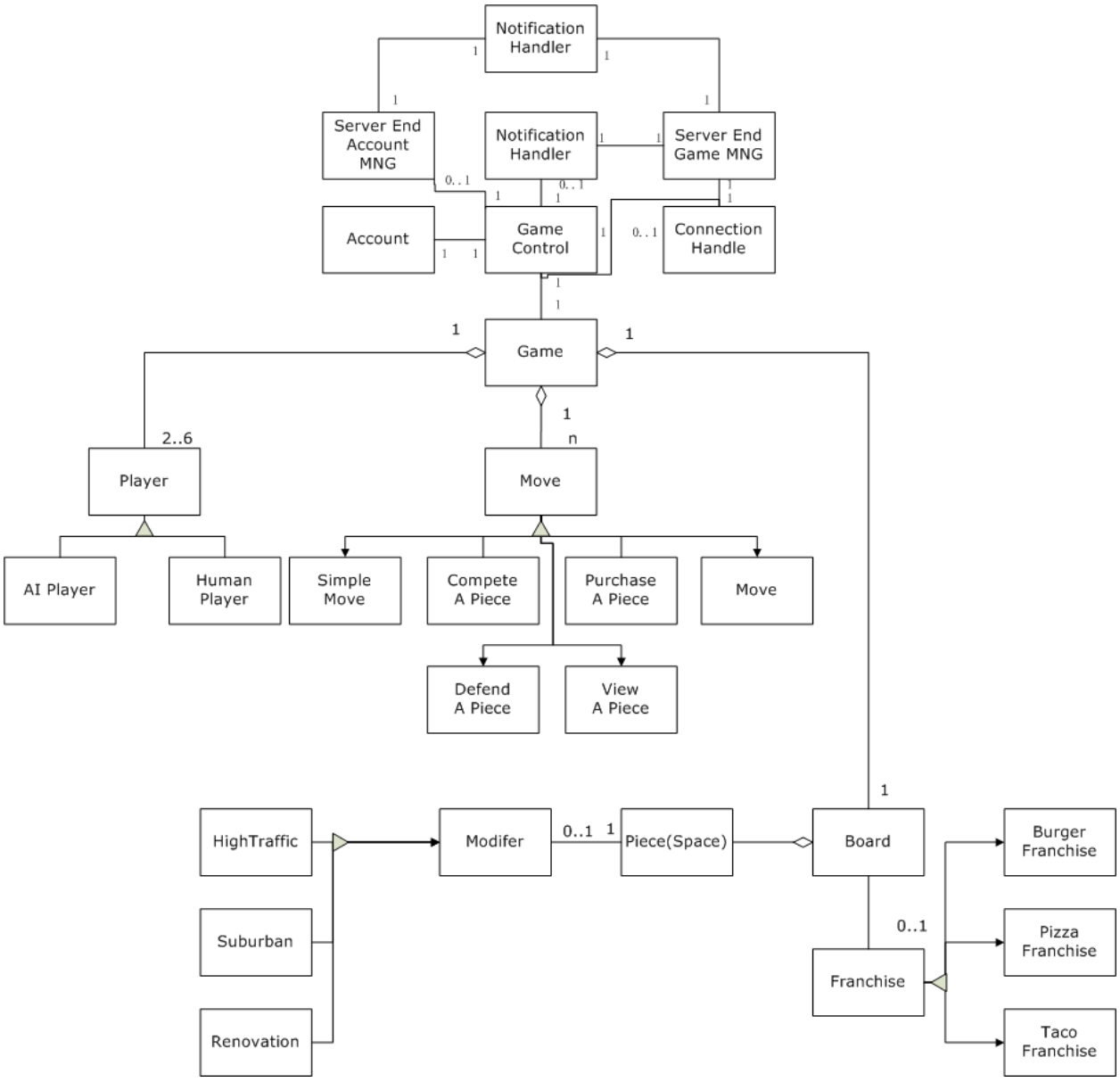

Figure 7 - Refined Object Model

23. Class Diagrams

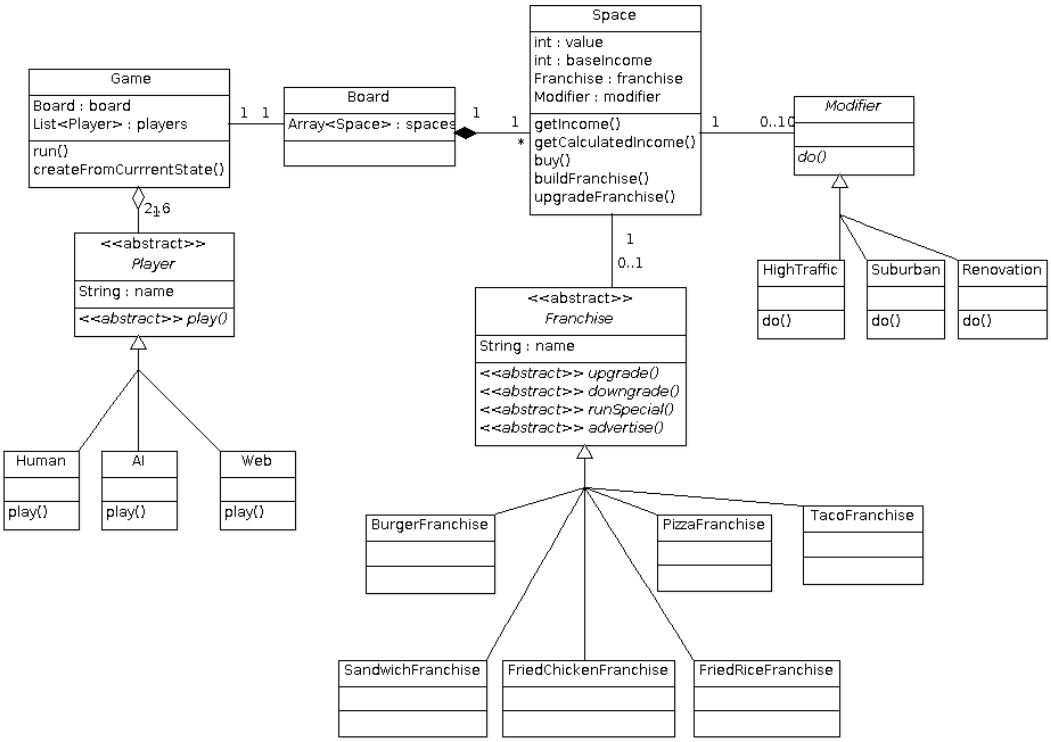

Figure 8 - Class Diagram

24. Dynamic Model

24a InstallApplication Sequence Diagram

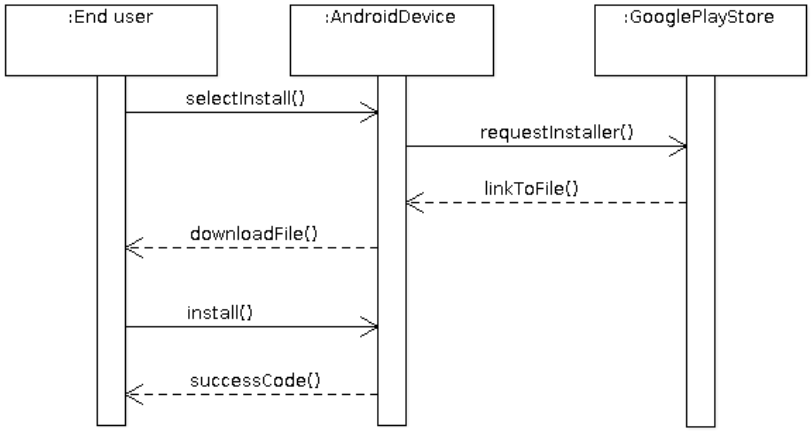

Figure 9 - InstallApplication Sequence Diagram

24b CreateAccount Sequence Diagram

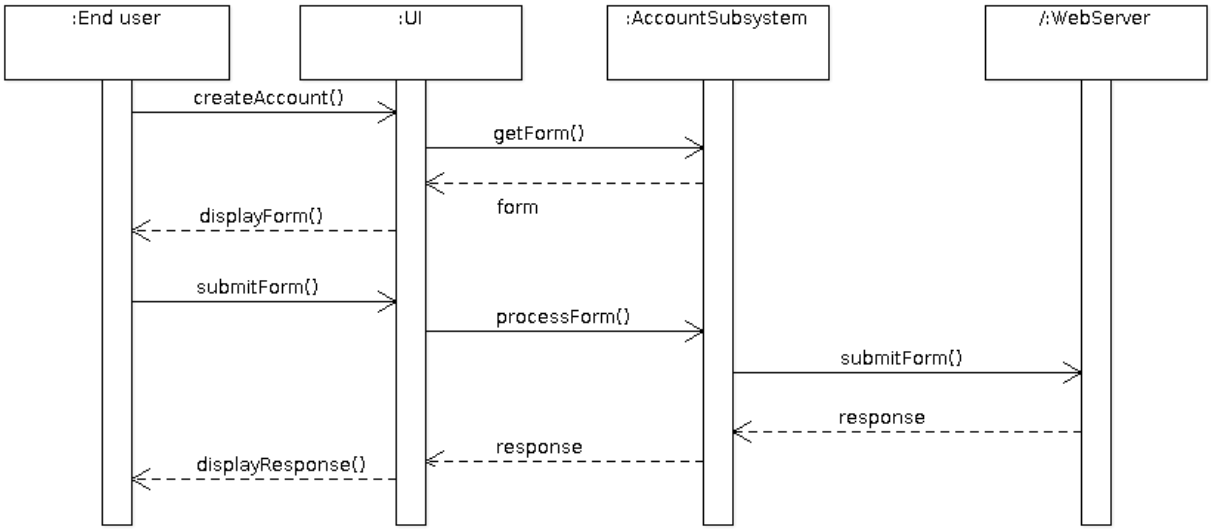

Figure 10 - CreateAccount Sequence Diagram

24c ShowMenu Sequence Diagram

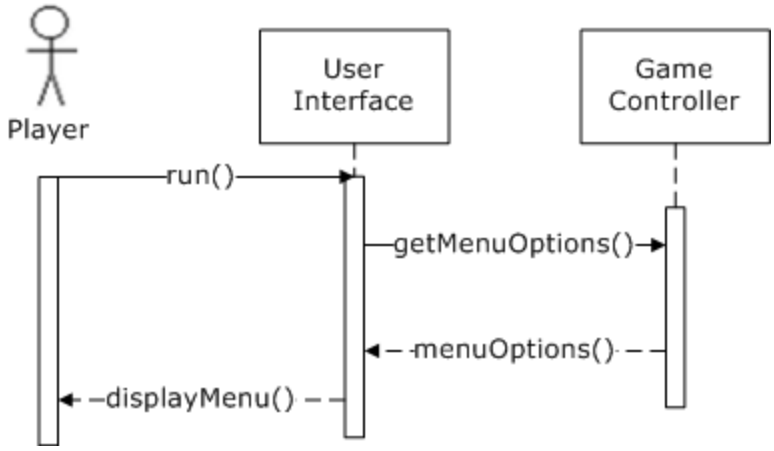

Figure 11 - ShowMenu Sequence Diagram

24d User Manage Account Sequence Diagram

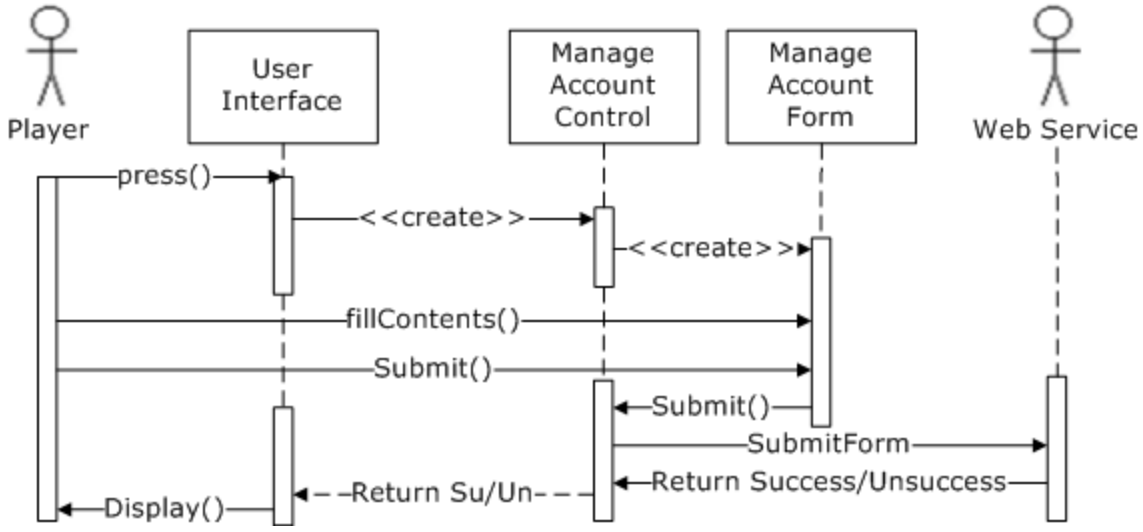

Figure 12 - User Manage Account Sequence Diagram

24e User Play Single Game Sequence Diagram

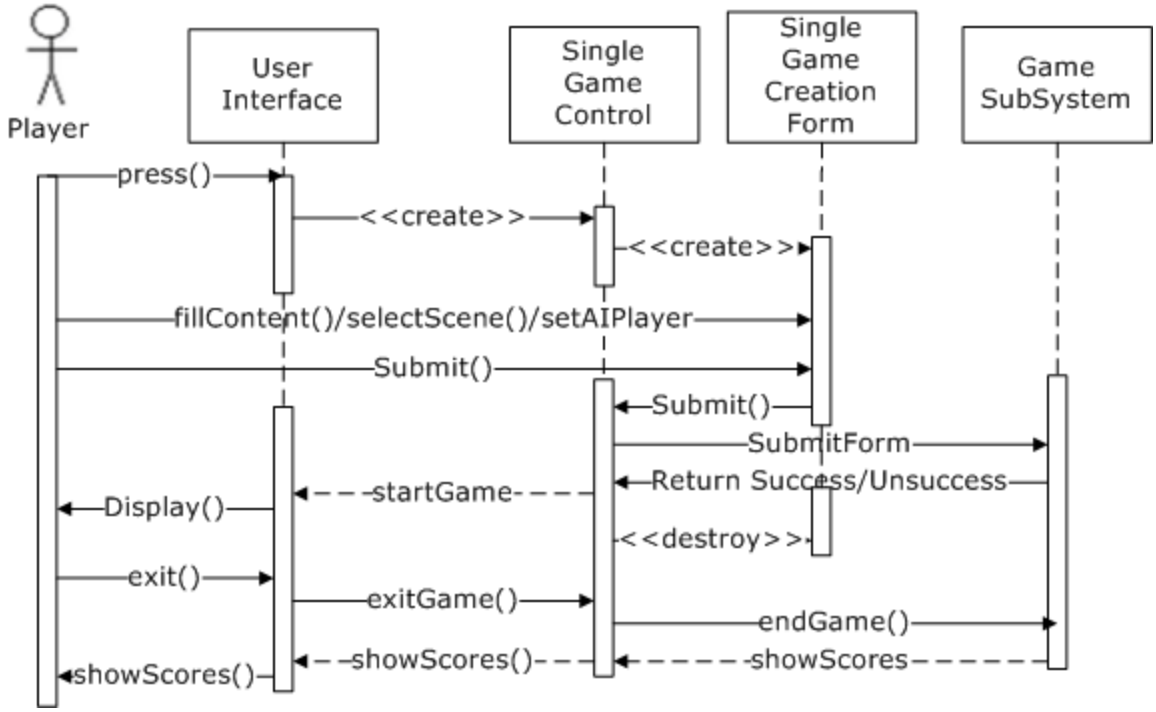

Figure 13 - User Play Single Game Sequence Diagram

24f User Play Multiplayer Game Sequence Diagram

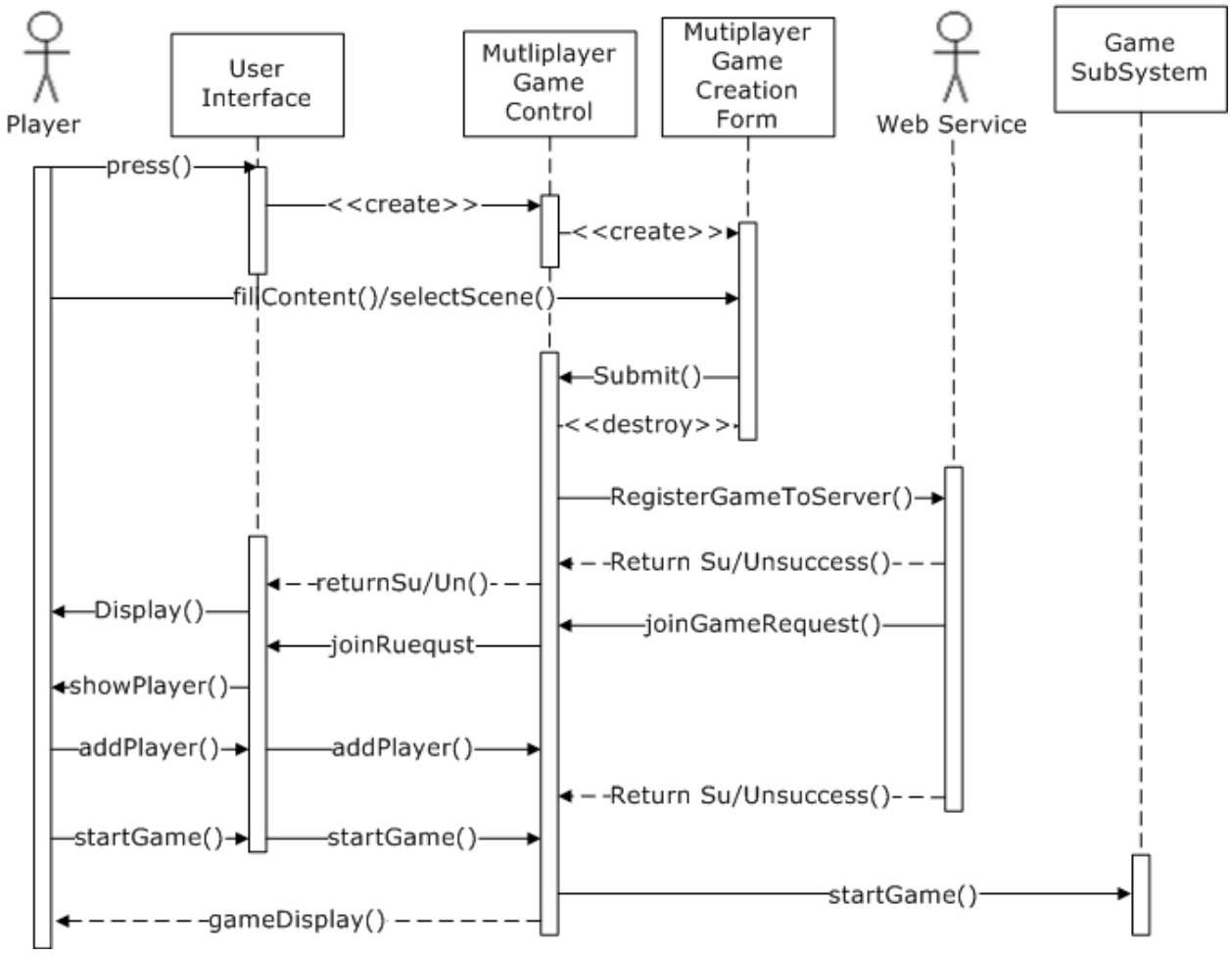

Figure 14 - User Play Multiplayer Game Sequence Diagram

24g User Join Multiplayer Game Sequence Diagram

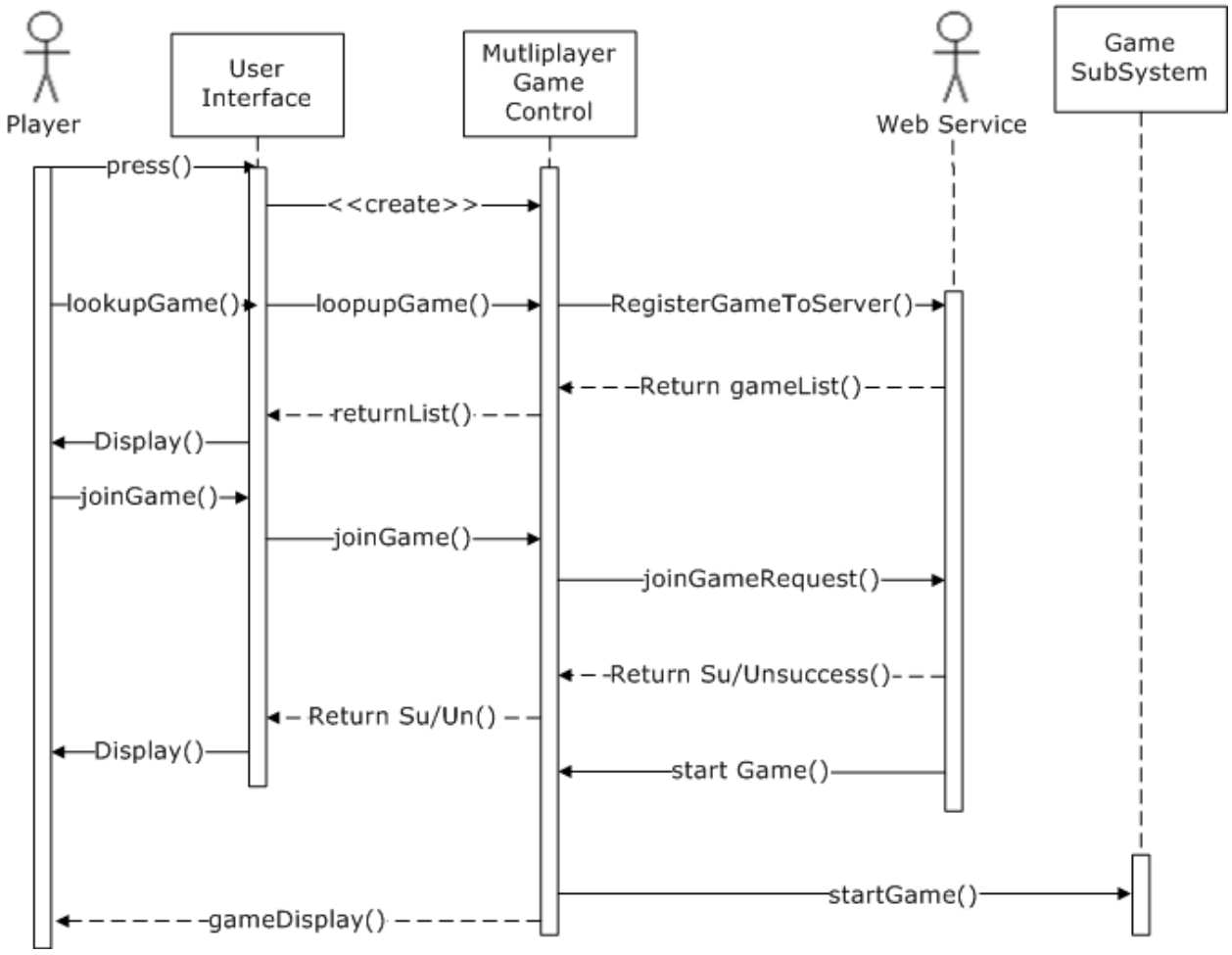

Figure 15 - User Join Multiplayer Game Sequence Diagram

25. User Interface

25a Game-Play

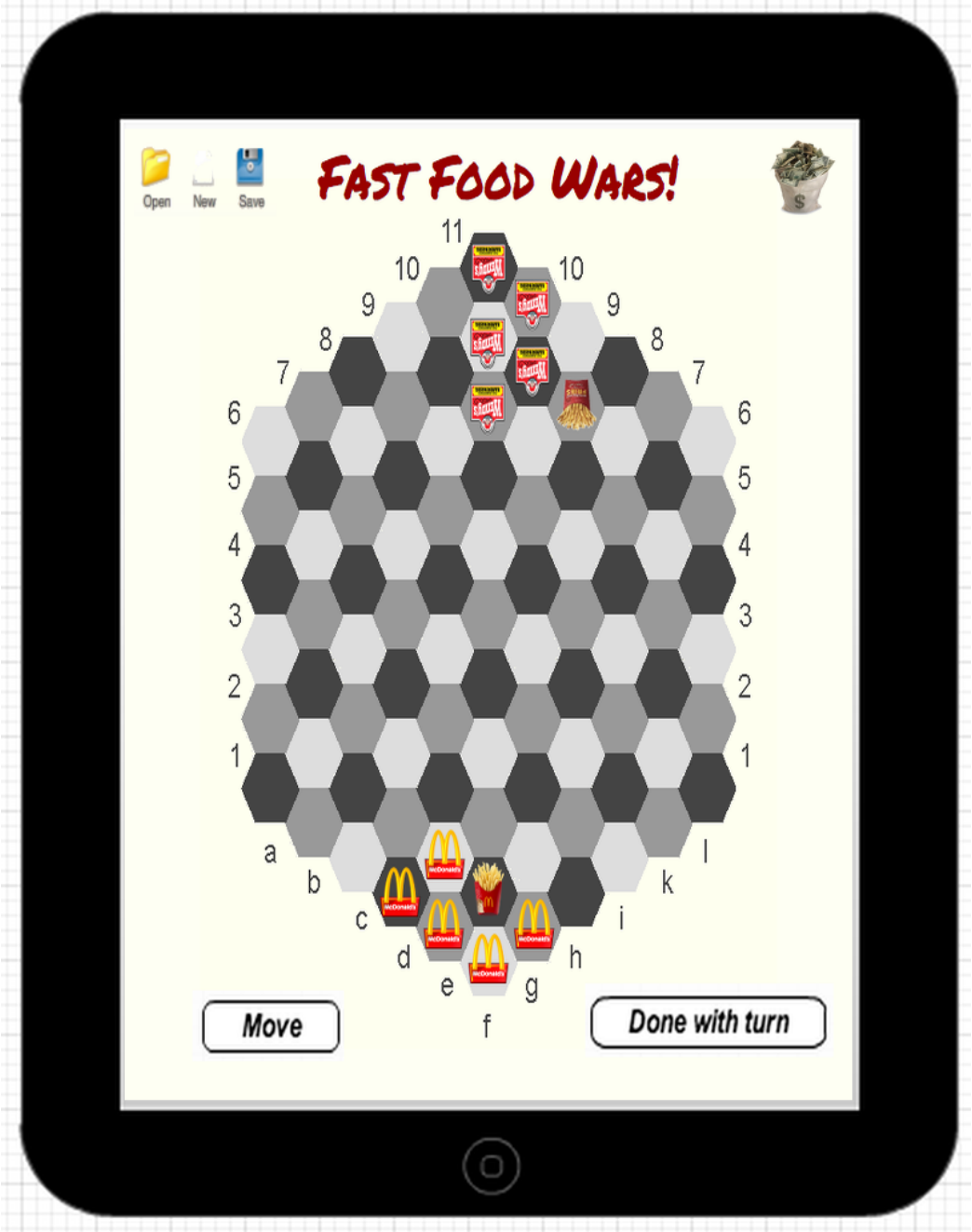

Figure 16 - Game Play

Open: opens a saved game.

Save: saves the current game state.

New: creates a new game.

Bag of money: represents the assets involved in the gameplay. In the place or in addition to a money bag or alike icon will be the information corresponding to the players assets with respect to the current game state.

Move: initiates the players next move. After this button is clicked the user will be able to “attack” (i.e. “purchase”) one of their pieces surrounding spaces and mark it as their own fast food franchise. This implementation will be left up to the design / development team to determine how this exact part of gameplay will be conceived. The main feature that must be abided by is that the player must be able to buy franchises on surrounding spaces and the assets used will be reflected in the “bag of money” asset information area.

Done with turn: initiates the next players turn. This locks gameplay for this user (i.e. they will not be able to move or change the game state in any way until the other players turn has finished).

25b Signing Up / Creating User Profile

Figure 17 - Signing Up / Creating User Profile

This is a simple sign-up screen that just provides the backend system with required information to build a user profile that is required to start a network connection for a multiplayer game.

25c Error Messages

Figure 18 - Error Messages

A simple error handling screen that will be displayed to the user if the server cannot be reached for whatever reason. All other error messages will fall under a similar construction so they are omitted from the mockups.

25d Menu

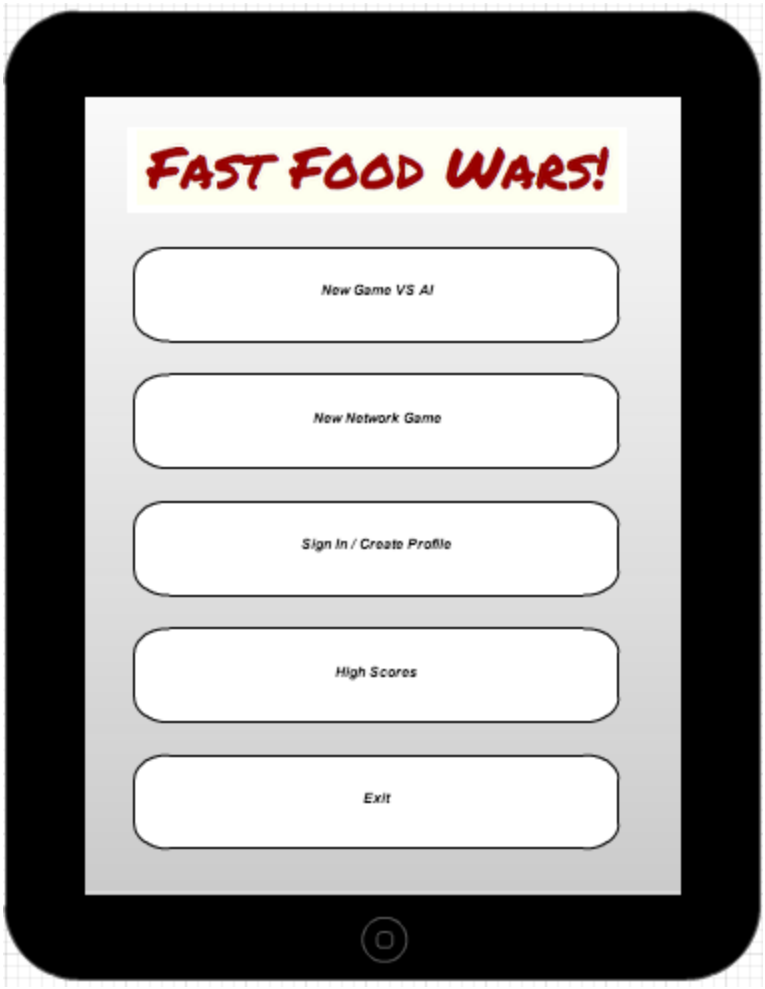

Figure 19 - Menu

The main menu for choosing gameplay options. This is the central screen where the user will either navigate to:

New Game VS AI: play a new game vs the computer AI; initiating the CreateSinglePlayerGame use case.

New Network Game: creates / plays a new multiplayer (network) game; initiating either the CreateMultiplePlayerGame or JoinMultiplePlayerGame use cases.

Sign In / Create Profile: let the user sign in if they are an existing user or create a profile if they do not have one already: initiating either the CreateAccount, DeleteAccount, or ManageAccount use cases.

High Scores: lets the player see the high scores for single player (vs AI) games; initiating the ShowScoresSinglePlayer use case.

Exit: lets the user exit the application.

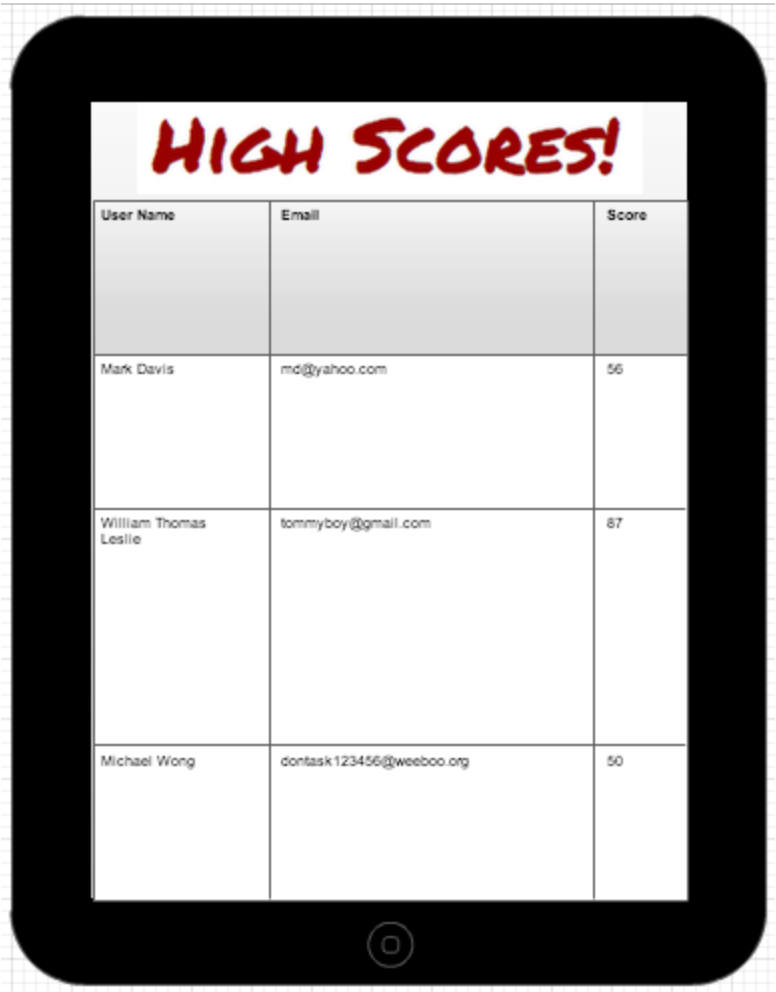

Figure 20 - High Scores

High scores screen shows the users high scores. We will leave up the decision of whether or not this information is too personal to display on this screen to the implementors.

IV. Design

26. System Design

26a Design goals

Performance criteria

Response time

Any interface between a user and the game should have maximum response time of 0.5 second.

The response time should be consistent at all points during the game in order for the users to feel the fluent and smooth working of the software product.

Moving a player from one spot on the board to another should not take more than 3 seconds after the user has decided where to move.

Graphics performance

The system should be able to detect any hardware changes and automatically optimize the in game display.

Users shall be allowed to set display and other settings according to their preferences regardless of the hardware and OS the game is running on.

Memory

By default, maximum memory allotted for the game to run smoothly should be 1.5 GB.

Users shall be able to change the configuration for more space to use according to their operational environment.

Reliability and Availability

The game should be readily available to download and/or play for users who have systems that meet the hardware and software requirements.

The game should not show any kind of lag or jittery behavior whatsoever.

Robustness/Fault-Tolerance

While downloading the game on an Android device, the progress shall be saved in the event of a connection failure or network outage and be resumed once the device is back online.

While playing, the system should prompt the user to save his/her progress (assets) after every sale or purchase.

In case of an unexpected system hardware or software failure, the game should be able to start and run once the system is back up without any special effort.

The database should be consistent and uncorrupt in case the game shuts down abruptly.

In case of any unexpected game crash, the user shall be able to review a report and send it to the maintenance team for future upgrades.

Security

Users should only have access to their data and should not be allowed to tamper with other users' data. All user and system files should be maintained securely.

End user criteria

Utility

The interface and the gameplay should provide an overall challenging and strategic tabletop gaming experience.

Fast Food Wars should implicitly give a real world perspective to the user about how the Fast Food business is implemented and what the basic mind set must be like to manage resources.

Usability

Users should be able to download and play the game easily on either a PC, a laptop or an Android device as long as it meets the software requirements and has a working internet connection (for download).

The game should be easier for people familiar with the business market as compared to the users completely unaware of the domain who might have to go through the tutorials.

The system shall provide videos by domain experts and users who test the game before its release telling novice users about the fast food industry and giving verbal instructions on how to play apart from and an instructions manual.

Users shall be able to select the level of difficulty while playing both the AI and online to match their level of knowhow.

Maintenance Criteria

Scalability and Extensibility Requirements

The game shall be either a single player in which case the AI would be the opponent or a multiplayer game in which case users may play each other to become the Fast Food champion.

Without any modification to the rest of the system, the development team shall be able to introduce new assets and incentives.

The development team shall be able to add new services to the Game subsystem without having to modify the rest of the system with it.

Modifiability

The organization and documentation of the the product should be unambiguous to enable developers to add new features to the code.

The source code should also support low-level changes without having to make any changes to the rest of the system.

The game should be easily upgradable to keep up with upcoming technologies and to treat any bugs reported by users.

The source code should be in accordance with the design pattern used.

Low coupling and high cohesion should be given serious consideration while subsystem decomposition.

Functionality of the system should not be changed if a new sub system is added or new class is implemented.

Readability and Traceability

The code should not be hard to read but it shall be obfuscated to prevent changes by anyone other than the developers. Each section of the implementation should be documented with strict adherence to the guidelines and the source code also should be structured professionally.

The implementers should not take any more than two minutes to map the code to the corresponding requirement.

Adaptability

The entire game shall be written in Java and compatible with any desktop, laptop, or smartphone with Java runtime environment and Android (only applicable to smart phones) .

OpenGL is an essential component to get the graphical display as intended.

Cost criteria

Development cost

The first release of the game should take around $6,500 and should be finished within a time period of 5 months. Any conflicts regarding the deadline shall be reviewed and discussed.

Deployment cost

Since the game mainly targets the open source android market, there should be minimum deployment cost but for other OS like Windows, the deployment cost

should be whatever the vendor sees fit ,which usually is not obscenely high.

It shall not take more than 15 minutes to install the game even on the lowest configuration.

Maintenance cost

The developers should address user complaints in a timely manner and release bug fixes on a regular basis.

The development team shall give out new releases of the software once every three months to keep the application updated and even notify the registered users by email.

Administration cost

No additional costs except for the one by the administrative team shall be incurred.

The cost of the administrative team should be kept as minimal as possible.

Upgrade cost

No additional costs shall be incurred for users to update the game to the latest versions developed by the design and / or development teams.

Current Software Architecture

The game incorporates the Repository architectural style as it shall be capable of handling a large number of users and all their data and also be able to process all of that data in an ordered manner.
Fast Food Wars uses a centralized repository where all functions a user might use are stored and any user that interacts with the system can access the repository to utilize the public functions.
As the game is also an online multiplayer game, the development team realises that the major flaw here is the risk of a bottleneck and proposes to address the issue.

27. Proposed Software Architecture

The proposed architecture is a four-tier architectural style, with aspects of other styles. The Android client will serve as the presentation client, with the web server fulfilling both the interface and application logic. A database will complete the final tier and provide persistent storage. The notifications subsystem should function in more of a client-server architecture, with the web server acting as the client.

27a Subsystem Decomposition

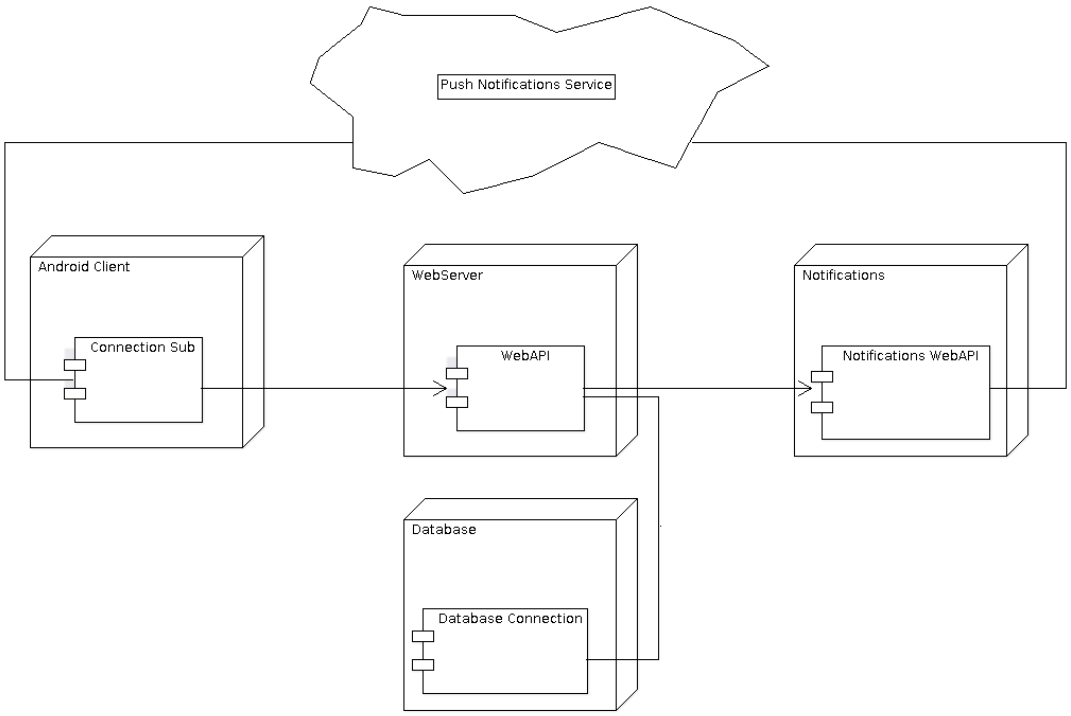

Figure 21 - Subsystem Decomposition

The system will be decomposed into four subsystems. These are:

- **Android Client** - The Android Client will be installed on user's Android devices. The Android Client will communicate with the WebServer over HTTPS. The Android Client will also receive messages from the Push Notification Server using built-in Android functionality.
- **WebServer** - The WebServer will be on a device running Apache with the lightweight RailsAPI serving as an interface between web requests and the business logic. The WebServer will fulfill requests from Android Clients. It will maintain a connection to the database and to the Notifications server.
- **Database** - The Database will fulfill requests for data from the WebServer.
- **Notifications** - The Notifications subsystem is an existing service. Minor modifications are necessary to support Fast Food Wars. It will receive requests from the WebServer, and if applicable, attempt to fulfil the request through the Push Notifications Service.

27b Hardware / Software mapping

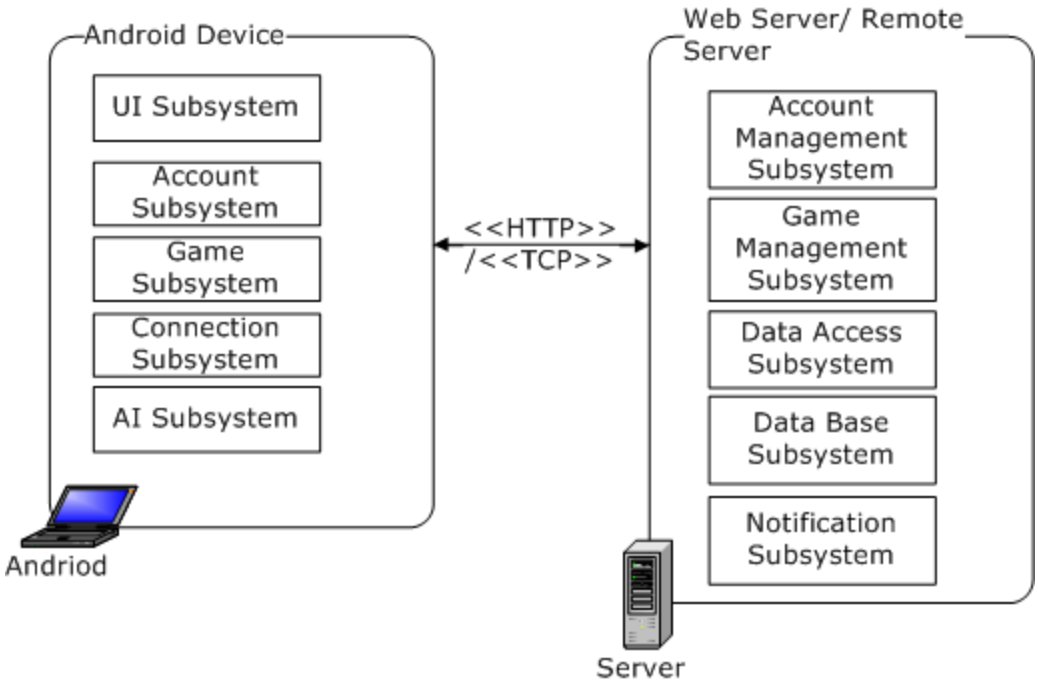

Figure 22 - Hardware/Software Mapping

Fast Food Wars shall be written entirely in Java. It would be fully compatible with any PC, Laptop or Handheld device that runs Java on it or has the ability to do so. More recent versions of Operating Systems like Win 8, Win 8.1, Mac OS X, Android 4.x shall get a slightly graphically enhanced version of the product. Also, the game uses Java OpenGL to display the hexagonal playing platform as well as the User Interface. Java OpenGL is an Open Source Library which makes is fairly easy to utilize as well as meets the design goal of minimizing operational cost.

Java is used extensively to implement the application logic as well, which meets the development team's design goal of minimizing deployment and development cost.

Once downloaded, Fast Food Wars will run directly from the user's computer system with without any special aiding softwares. The only requirement for successfully running the game is that the user shall have Java runtime environment, which is easily downloadable, installed on their machine.

The data in the game is both low abstraction and high, which is the reason the development team has chosen to use relational database and flat files to store user data, game data, assets, store data etc.

27c Persistent Data Management

For this product, the only persistent data is the user information and saved

progress (state of game) once the player starts playing the game. Storage of a user's information including any data related to gameplay must be done in an orderly fashion. Provisions for users to create new game files, load previously saved files from the database or locally and overwrite previously saved files with new saves shall be made. The game state a user chooses to save shall depend on the checkpoints the user has reached. There is no minimum asset limit for a user to save any game data that can be loaded later. Flat files may be used for data stored locally on a user's computer and since Fast Food Wars is also an online multiplayer game, it may require concurrent access to a large data set for which a relational database shall be used as the storage that the users can access to retrieve their progress and all accumulated assets.

To meet the design goals, the development team makes sure that system storage method chosen while allowing concurrent access to data, maintains a certain level of consistency to minimize the risk of failure during a system crash and low level of abstraction wherever possible.

Users are not required to install separate database management systems to access their data.

The system stores all the game data of all it's users in a single database.

With low level file abstraction, developers can perform various speed optimizations easily, which in turns leads to accomplishing the above design goals.

Aware of the risk of system crashes, the development team takes into serious consideration the trade off between operation cost and estimated maintenance cost in such an event.

27d Access control and security

The user being one of the actors that can directly interact with the system shall have limited access to the system. Once downloaded and installed, a user shall be able to view the in game menus, play the game, create a profile to use the multiplayer feature, view previously saved games, view assets accumulated, load saved games and also view their profiles and progress. The user shall not be allowed access to the database to make changes to other's profiles. The user will not be able to add/modify any classes or interfaces in the system or any of the design features except through the game's preferences and settings.

The database manager will be the only one with full administrative access to the database.

Security, being one of the key design goals of the product ,shall be given top priority. No user can access the system in any way other than the ones outlined above.

The development team has full access to every functionality related to the game and is the only actor that can develop updates and issue them for the users to download from the respective app stores (OS dependent).

27e Global software control

The global control mechanism chosen for Fast Food Wars is event-driven as the development team believes that it is the optimal path to take for the actual game play essentially so that all inputs are centralized by the system and put into a loop. The PlayGame class shall be used to realize the event-driven control that will include user command events as well as the asset sale/purchase calculation based events during active game-play. However, the UI Subsystem, Asset Subsystem, Connection Subsystem and User data subsystem shall have procedure-driven control mechanisms as operations needs single inputs from users to carry out one thing at a time in real time. The UI class includes user input, waits for it, and performs the necessary execution.

27f Boundary conditions

After being downloaded and installed, the system startup occurs through interaction between the user and the user interface. After system startup, the user initializes the Game play subsystem. A user can either start a new game or load one up from a previous session having all the parameters restored. The Asset Subsystem shall be accessed both by the UI and Gameplay subsystem but with the difference of what operations a user can perform. The User Interface Subsystem shall initiate with the startup of the system. To begin with, the user would not have any data stored whatsoever. The first data stored would be the user's user profile followed by any game session the user decides to save along with the assets accumulated up to that point. In the event of crashes, users would not lose any data previously saved and shall be able to retrieve it once the system starts back up.The users shall also have an option to retrieve any game play sessions played in the past by using their passwords to access the database. All specific save operations will be made from cached files in order to protect the original data files from getting corrupted in the event of a failure. The UI would initiate saving upon user action. The gameplay subsystem shall check for user inputs and all operations before displaying an error message. In case restoring the game is not an option, the system shall stop the current on going war and return the user to the User Interface with the choice to load the last saved state. The system should be able to copy the byte sequence before saving it, which can later be used for recovery. This will ensure a seamless recovery.

The Game Subsystem is always initiated by the user through the UI and shut down either when a player becomes the Fast Food Champion or when the user decides to save a game and quit. The asset subsystem shares exception handling methods with the Game subsystem by making use of cached files. The

asset subsystem shuts down upon user request through the UI subsystem.

28. Subsystem services

28a Android Client

Account

The Account subsystem will implement the verify(), create(), and delete() methods.

Game

The Game subsystem will implement the play(), sync(), and submitMove() methods.

UI

The UI subsystem will implement the display() method.

PushNotifications

The PushNotifications subsystem will not expose any public methods. This class will receive events directly from the Android operating system.

GameController

The GameController subsystem will not implement any public methods. The controller is responsible for handling events and serves as a bridge between the Game object and the UI.

Connection

The Connection subsystem will implement the send() method. To allow for asynchronous sending, it will accept a callback.

AI

The AI subsystem will implement the play() method.

29. Object Design

29a Object Design trade-offs

The Space class maintains a list of Modifier objects. This is likely to become cumbersome if later iterations of the game require additional or more complicated modifiers. It is possible that the Decorator Design Pattern would be of use. The simple list of abstract object is easier to implement and has the benefit of less obfuscated code.

The Franchise class is an abstract class, which subclasses, such as BurgerFranchise inherit. It appears at this time there is not much difference between the classes. It would seem that it would make more sense to just bring any attributes into the Franchise class. This proposed class structure does offer significant more flexibility for future development. It is envisioned that in future releases each chain will have special “skills”.

29b Interface Documentation guidelines

It is vital that all interfaces be documented thoroughly. To this end, during all phases of development, the development team should employ Doxygen tags, so that documentation can be automatically generated. This documentation will be available to all members of the team.

29c Packages

The system shall be divided into packages that correspond to the subsystems. All efforts should be made to use off-the-shelf software to satisfy the requirements. For example, the RailsAPI package provides much of the functionality of the interface to the WebServer application logic out of the box. Likewise, the ActiveRecord package provides functionality to perform database queries with little to no configuration. All effort should be extended to adapt to new technologies.

29d Class Interfaces

The system uses the traditional implementer, extender and user method to specify interfaces.

While starting to write classes, the development team was broken into specific roles. The implementer shall be the one who develops initial classes which may lead to realising specializations by the extender while the user can invoke any operation from the classes already created.
For all the classes i.e Space, Game and Board, there shall be interfaces specified that make use of the internal operations and public methods of the classes.

The Game class has the operations to run() and createfromcurrentstate() and has a relationship with the Board class with a multiplicity of 1-1. The Board class has a composition relationship with the Space class with multiplicity of 1-1. The space class has a 1-0..1 relationship with an abstract Franchise class a well as a 1-0..10 relationship with a modifier class. There is an abstract Player class, which has an aggregation relationship with the Game class mentioned above. The Player class depicts inheritance by having three subclasses Human, AI and Web.

BurgerFranchise, PizzaFranchise, SandwichFranchise, TacoFranchise, FriedChickenFranchise and FriedRiceFranchise are all classes that inherit methods from Franchise.

V. Test Plans

30. Test Methodology

Fast Food Wars is design to be a multiplayer game targeting Android devices. The first design goal of this game is to create a game that complies with the functional requirements. The second goal is to create a game with high reliability, usability, performance and low maintenance cost. Given these two goals, a multi-stage test procedure will be necessary to validate and verify the functional requirements and design goals.

Traditional test methodology dictates a testing phase at the end of the product development. This runs the risk of developing a product that is hard-to-use or unusable. To mitigate these issues, it is imperative that we integrate testing with the implementation.

Initially, there are two testing plans, one for the Android client and one for the WebServer. These can be performed in parallel. Once the tests are completed for these subsystems, integration testing can be performed between the two systems. Finally the system can be tested as a whole.

30a Android Client Testing

Traditionally the first phase in testing is code inspection. New technology has allowed for earlier usability testing, as mockups can be developed with little to no existing code. Initially, the application will be prototypes will be developed with pen and paper. By taking photos of the sketches and importing the images into a program such as Prototyping on Paper (https://popapp.in/), an interactive mockup of the user interface can be built. The prototype can be sent to Funskool for preliminary testing. This early feedback should be invaluable for further development.

Once Funskool is satisfied with the initial design, coding can begin and additional usability testing can occur. The user interface should be mocked with a tool such as Fluid UI (https://fluidui.com/demos). Fluid UI allows the user to build interactive demos that the user can interact with in a more realistic way than pen and paper or even a Wizard of Oz prototype. The simulation can include swipes, double taps, or other gestures on mobile device with little to no coding.

While the UI is being solidified, the coding should progress. It is recommended that the developers use a Test Driven Development methodology, developing

unit tests concurrently with the coding of the client. A suitable framework is recommended, such as JUnit. Obviously, this decision for the testing framework should be delayed until the final decision for the development environment is decided. A goal of 80% code coverage for unit tests should be met or exceeded.

Once a week during development, the development team should meet for code inspection for any code that has been produced that week. A Code Inspection Checklist will be filled out by a team member, and potential defect will be discussed as a team. The inspection should cover the unit tests to verify the tests are sufficient. Additionally:

- All methods that are considered to be critical and involving complex internal state changes should be tested with state-based testing.
- All polymorphic classes should be tested with Polymorphism testing method.
- All methods with complex control flow should be tested with path-based testing.

Within the Android client, integration testing should be devised to identify deviations from the specified interface. When all of these tests are passing, the Android client subsystem should be tested as a whole.

30b WebServer Testing

The testing of the WebServer should mirror the testing of the Android client with the following exceptions.

- Usability testing will not be performed on this subsystem. The primary user interface is the Android client, and it will be assumed that the preliminary usability testing has been performed on it before commencing development of the WebServer subsystem.
- Since the WebServer will be written in Ruby and not Java, the unit testing framework should be RSpec and possibly Cucumber.
- During development, the web API should be documented with Swagger (https://developers.helloreverb.com/swagger/), so that the client developers are able to better test the interface.

30c Integration Testing

Upon successful completion of the Android Client and the WebServer, the team can begin final integration testing. Since there are really only two subsystems, and test methodology should be equivalent. During this phase, it is important that the interfaces be continually tested. To this end, a server running the Jenkins CI software (http://jenkins-ci.org/) should be utilized. This server will run the entire suite of tests at every commit to the Git repository. The software will send notifications to the Project Manager and the original author of the commit whenever a test fails, allowing the team to address issues immediately instead of finding the issues later in testing.

30d System Testing

Upon passing of all integration tests, the system should be tested as a whole. Initially, the system will be deployed internally, with users manually reporting bugs or issues with the company's issue tracking software. Once the bugs have stabilized, the game will be released as a beta to select customers. Concurrently, the system will be released to Funskool for Acceptance Testing. Issues will be tracked, prioritized, and resolved before final Acceptance Testing.

31. Testing Materials

The testing and development teams will require a range of Android 4.x devices, including tablets and smartphones.

In addition, server hardware with Jenkins CI should be provided.

32. Testing Plan Example Form

Function Model Name	Feature Name	Function Model Owner	If Tested	Testing Approch	Test Case ID	Test Schedule
Logon Model	User Authentication	N/A	No	Unit Test	User Authentication	To be decided
a	a	a		a	a	
a	a	a		a	a	
a	a	a		a	a	
a	a	a		a	a	

Table 13 - Testing Plan Example Form

33. Sample Test cases

Test-Case identifier	User Authentication
Test location	http://cs440g10.uic.edu/testcase/userAuthentication
Feature to be Tested	User logon to the system
Feature Pass/Fail Criteria	The test passes if the user with correct ID and combination is accepted by the system

Means of control	The logon() method is called via a test driver UserLogon
Data	1.The device is properly connected to internet 2.The debug model is on for console output 3.Sets of user ID and combination correct and incorrect
Test Procedure	The test is started by user input double-click the user case at the specific location.

VI. Project Issues

34. Open Issues

The decision on whether our game would be compatible with older versions of android is still pending.

The client after having met with the development team has discussed some changes that might modify the functionality of the entire game in the future but these changes have not yet been officially recognized.

The word around the online game market is that the product would not be compatible with Windows devices which the development has not yet responded to.

35. Off-the-Shelf Solutions

35a Ready-Made Products

Products that have been thoroughly studied as are somewhat likely in functionality than our Game are listed below:

- Monopoly
- Risk

35b Reusable Components

There are several pieces of code that can be used while implementation so as to save programming time. The development team has made sure that the reuse of code in certain situations would not alter the rest of the system.

35c Products That Can Be Copied

Funtastic and Kadiko Enterprises published a game called Kaliko which used a hexagonal game board. Our development team acquired that idea and by

reusing the hexagonal board for our game, saved around 30% of the development time that would've come from creating our own game board.

36. New Problems

36a Effects on the Current Environment

Although, it has been considered with caution, the change in the classes of the Game subsystem may lead to modifications of the functionality of the UI subsystem.

A domain change request by the client could lead to the entire project being built again from square one.

Even the slightest change in the deadline would affect the work of the development team including managers.

36b Potential User Problems

The impressionable users of the product might have a negative impact by playing the game too much in a way that they might be swayed by the allure of all the acquisitions and pay less attention to other important aspects of their lives.

The development team realizes the above issue and shall make the client aware of the issue but it ultimately be the clients call.

There would also be a precautionary warning on the screens on the users before they download the game off the internet.

36c Limitations in the Anticipated Implementation Environment That May Inhibit the New Product

The budget allocated to implement the design of the game and make it available for different platforms over different operating systems might not be sufficient.

If the client decided to pull a turn around and change the functionality, the development team suspects there won't be enough resources to make those changes.

The product might be prone to reverse engineering in the open source market since the code would not be obfuscated as the development team would have to allocate budget for the purpose which is not feasible

36d Follow-Up Problems

The product might not be available for all the platforms the client intends in the amount of time desired as the budget seems insufficient.

37. Tasks

37a Project Planning

After being given the problem statement by the client, our development team sat down to discuss it and came to the conclusion that, it would be most feasible for the development of Fast Food Wars if we adopted the traditional software development life cycle approach. This decision was made after an initial consideration of the resources available.

The diagram below taken from the third edition object oriented software engineering by Bern Bruegge and Allen H. Duttoit, illustrates the activities involved in the object oriented software engineering

The approach consists of the following major phases:

Application Domain:

- Requirements elicitation
- Analysis

Solution Domain:

- System design
- Object design
- Implementation
- Testing

Our approach focusses on Requirements Engineering, System Design and Object Design while also, developing test plans to be implemented after the code is written to make sure every requirement has been met before the actual delivery of the product.

The development team after the first few meetings devised an initial plan to lay out the development process for Fast Food Wars with the estimated time it would take to finish the phase. The table below illustrates the same.

	Activity	Assigned Role	Input	Output	Duration (weeks)
1	Requirements Elicitation	System Architect	Problem statement	RAD	2
2	Analysis		RAD	Object Model	3

3	System Design	System designer	Object Model	UML Class diagrams; Design document	4
4	Object Design		Design Document; Class diagrams	Refined design document; Detailed diagrams	3
5	Test planning	Tester	RAD	Test Plans	3

Table 14 - Project Planning Durations

The development team decided to carry on with the above approach as it provides for iterative development which means more than one process can be carried out simultaneously.

38. Risks

Our development team has made every effort possible to work around problems encountered while designing the product. Although, the development was able to find alternatives to most problems , some areas remain that the team fears might pose threats to the product.

The major risks identified throughout the course of development are listed below:

- Client might want to change functionality before delivery.
- Inadequate resources for extending functionality to other operating systems.
- Excessive schedule pressure
- Silver bullet syndrome: Management's belief that negotiating for increasing the budget might solve the problem of extending the Games functionality to other platforms.
- Increasing end user's needs and demands for online tabletop games might affect the client's needs and inturn the product.
- Low productivity introduced due to tools like ArgoUML.

39. Costs

Event	Cost estimate (USD $)
Initial setup	500-700
Team meetings	600-850
Software tools used	900-1100

Software development	700-1400
Implementation	650-1000
Testing	975-1250
Presentations to client	700-850
Installation guide on different platforms	600-780
Managing (updates etc.)	700-1000 quarterly
Modifications in functionality after the RAD is written	500-800

Table 15 - Costs

The table above is a representation of the entire cost estimates for developing Fast Food Wars.

40. Waiting Room

Throughout the process of developing Fast Food Wars, during meetings, the development team came up with ideas that fit the domain perfectly but were not completely pragmatic to actually realize due to lack of sufficient resources. These ideas were considered as requirements that shall be realized in the future releases:

1. The game shall run on other Windows 7 and 8/8.1, ios, OS X and also on linux based operating systems and not just android which is the major focus for the current release.
2. The game play shall be 3D
3. Enhancements shall be made to the create user profile menu. Users shall be asked for more details to generate a fully illustrative database.
4. A power up component shall be added in the future such that players can buy power ups with game money to gain advantages in the game.
5. Automatic detection of a player type to generate new game suitable for the user's level i.e if the player creates a user profile stating that he/she is an expert in business marketing strategies, the game system shall generate a game of difficulty high automatically which can later be switched to an easier level by the user before the game starts .

VII. Project Retrospective

41. Summary

Looking back at the work our team put in towards developing the product throughout the entire time period, there are numerous things that proved positive in the most beneficial of ways while there were also a few methods that failed to provide any significant results.

42. Tools Used

1. Git: Git proved to be a frequently used Case tool for collaborative development in that it allowed for each team member to upload his work to the repository and share it with the others. Also, each team members contribution was merged with the existing documents by Git easily.
   A slight downside to using Git was that, at times it became confusing and inconvenient when two members were working on the same phase simultaneously and the last one to save the document to Git would have his document pushed negating all the changes made by the other team member regardless of their significance. Thus, it was necessary to be in constant communication with everybody while working on the documents.
2. **Google Drive**: After working on Git the first two weeks, our team decided to set up Google drive for the development of Fast Food Wars which has till now proved to be highly efficient in every way. anybody can upload documents and share it with others anytime. The people the document is shared with can edit it with ease. Not only could we collaborate effortlessly, asynchronously and productively but also view each team members work in real time and be notified of any changes that were made so that there would be no confusion whatsoever.
3. **ArgoUML**: Since,software development demands representation, our development team used ArgoUML to draw UML diagrams. In hindsight it wasn't the best choice but it is the one we had to go with as we did not have any other resource to perform the representation tasks. Using rational rose was our first thought but the expense was out of our budget.
   ArgoUML has a poor user interface which slowed down the development at times leading to low productivity.

Besides using tools to accomplish tasks of generating the documents, our development team found that having regular meetings where each member is present and contributes to the ideas and provide criticism to the other member to better their idea is by far the most useful and highly productive. Meetings can be as short as an hour considering everybodys time constraints but coming together to work synchronously is of utmost importance.

VIII. Sprint Goals

43. First Sprint Goals

We expect and hope to see a hexagonal game board with a player and all the ways he can move in. The board should be coloured Blue as it is important to the client. Alongwith the previous goal, there shall also be a user interface with a new game, save game, load game and create profile buttons. Functionality for new game and save game must be provided.

44. Second Sprint Goals

We expect to see refinement of the previous design with the introduction of an AI which plays against a human player after the player has started a new game. Each move the player makes should have a monetary outcome. The player should be able to save the game after a checkpoint and retrieve all the assets accumulated to that point upon loading the saved game.

We do not expect any back end services to be implemented in the time allowed to implement our design. The people implementing should just focus on the Android client and all the services provided by the sub systems mentioned in the design section.

IX. References / Bibliography

J. Bell, "Underwater Archaeological Survey Report Template: A Sample Document for Generating Consistent Professional Reports," Underwater Archaeological Society of Chicago, Chicago, 2012.

M. Fowler, UML Distilled, Third Edition, Boston: Pearson Education, 2004.

Information technology - Object Management Group Unified Modeling Language (OMG UML), Infrastructure, version 2.4.1, http://www.omg.org/spec/UML/ISO/19505-1/PDF

Launch Checklist - Android Developers, http://developer.android.com/distribute/googleplay/publish/preparing.html

The Entertainment Software Association, Industry Facts. http://www.theesa.com/facts/

McDonald's Franchise Owner - George Forrest. WTVI PBS Charlotte. http://www.wtvi.org/csbs/2012/04/10/mcdonalds-franchise-owner-george-forrest/

Web Content Accessibility Guidelines (WCAG). W3C. http://www.w3.org/WAI/intro/wcag.php

Doxygen: Main Page. http://www.stack.nl/~dimitri/doxygen/