

4 Way Chess for Charity

Sonia Bush

Junjie Hao

Zachary Jordan

Austin Li

Rolando Martinez

Jim Moore

Kenneth Pham

Table of Contents

1. Delegation of Tasks	3
2. Deliverable 1	5
3. Presentation Choice	44
4. Project Scheduling and Cost Estimation	45
5. Software Testing Plan	51
6. Comparison of Work With Similar Designs	53
7. Conclusion	55
8. References	56

1. Delegation of Tasks

- Sonia Bush
 - Deliverable 1: Functional Requirements
 - Deliverable 1: Address Feedback from proposal
 - Deliverable 2: Cost, Effort, and Pricing estimation
- Junjie Hao
 - Deliverable 1: 1.2 - make github repository
 - Deliverable 1: 1.3 - add all team members and TA to github repository
 - Deliverable 1: Explain software process model used
 - Deliverable 2: Conclusion
 - Deliverable 2: References
- Zachary Jordan
 - Deliverable 1: 1.6 - Include Github url in deliverable 1
 - Deliverable 1: Use Case diagram
 - Deliverable 2: Project Scheduling
- Austin Li
 - Deliverable 1: Class diagram
 - Deliverable 2: Describe who did everything
 - Deliverable 2: User Interface demo on Presentation Slides
- Rolando Martinez
 - Deliverable 1: 1.5 - make “project_scope” file
 - Deliverable 1: Architectural Design
 - Deliverable 2: Estimated hardware, software and personnel costs.
- Jim Moore
 - Deliverable 1: 1.4 - make commit to github
 - Deliverable 1: Non-Functional Requirements
 - Deliverable 2: Develop a test plan
 - Deliverable 2: put together presentation clips and subtitles

- Kenneth Pham
 - Deliverable 1: Sequence Diagram design
 - Deliverable 2: Compare work w similar designs
- Everyone
 - Presentation Slides

2. Project Deliverable 1 Content

(See Next Page)

CS 3354 Software Engineering
Final Project Deliverable 1

4 Way Chess for Charity

Sonia Bush

Junjie Hao

Zachary Jordan

Austin Li

Rolando Martinez

Jim Moore

Kenneth Pham

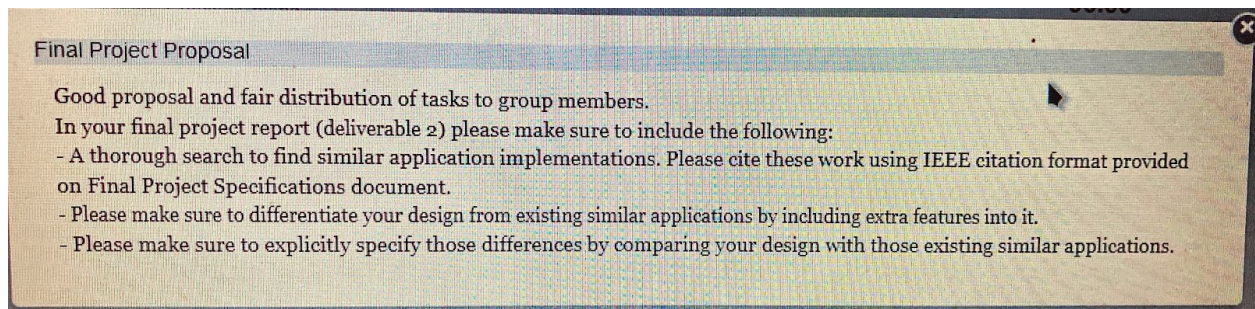
Table of Contents

• Feedback discussion	3
• GitHub repository information	5
• Delegation of tasks	6
• Software process model	7
• Functional requirements	8
• Non-functional requirements	10
• Use case diagram	12
• Sequence diagrams	14
• Class diagram	36
• Architectural design	37

1.

Final Project Draft Description:

Mobile fundraising 4 way chess game app. We can drive donations by having a highest donations leaderboard and a competitive chess leaderboard for bragging rights and fame. Users will also be able to watch ads or pay money to gain points that can be used to purchase cosmetic flairs and emotes for the game. The amount that users can play will be limited by an energy system. Users can purchase a premium account to get unlimited energy.



Please focus now on more charity based s/w for comparison, and the feedback we provided still applies to the new proposal.

Dropped Feature Ideas:

- Energy System:
 - We decided an energy system would likely irritate new players, so we removed it.
- Premium Subscription:
 - After we removed the energy system, the premium subscription no longer had a purpose, so we removed it as well.

Feedback

The purpose of this document is to address the feedback and to present a description of what our team is trying to accomplish by doing this software project. The 4 Way Chess Mobile Application is a turn-based multiplayer game that allows four opponents to test their chess skills. The winner will earn more points than their opponents. The more wins the player has, the better the reputation will be within the game as their name will show their ranking. The user can purchase or earn in-game currency and use it to unlock cosmetics that involves chess gear, flairs, emotes. In addition, all funds raised will go towards charity.

The following list includes research of similar application implementations. We will include them as IEEE citations and differentiate them from our game in the final project report. For example, none of the following applications involve charity:

1. <https://play.google.com/store/apps/details?id=de.j4velin.chess>

4-Player Chess is a multiplayer game made for android apps. 4-Player Chess offers a chess board with up to 64 pieces and four different game modes:

- 2-player standard mode-normal chess game.
- 2-player extended mode-extended board, each player has 32 pieces.
- 4-player team mode-each player has the standard 16 pieces, but is allied with another player against two opponents.
- 4-player mode, no teams- deathmatch, each player has the standard 16 pieces and fight against 3 other players.

2. <https://play.google.com/store/apps/details?id=com.merciari.chessx4>

Chess X4 is a multiplayer game made for android app. Chess X4 online allows users to create their game code in order to invite friends to play. The game allows the player to organize a 4-player game.

3. <https://play.google.com/store/apps/details?id=com.harmegedo>

Harmegedo 6 Player Chess is a multiplayer game made for android app. It allows users to play from 2 to 6 players. Players can play in teams of 2, 3, or everybody versus everybody.

4. <https://apps.apple.com/us/app/3-man-chess/id1438623432?ign-mpt=u%3D2>

3-man-chess a multiplayer game for IOS app. 3-Man Player allows the user to save their game and come back later, transfer a game in progress to other devices in multiplayer.

2. Github Repository

Link: <https://github.com/JunjieHao5/CS-3354-Final-Project-Group-3>

Team Members and their corresponding github account usernames:

Sonia Bush	SoniaYB3
Junjie Hao	JunjieHao5
Zachary Jordan	zachjordan16
Austin Li	austin-alt
Rolando Martinez	rolomart10
Jim Moore	xyag
Kenneth Pham	CloudByte10

3. Delegation of Tasks

- Sonia Bush
 - Deliverable 1: Functional Requirements
 - Deliverable 1: Address Feedback from proposal
 - Deliverable 2: Cost, Effort, and Pricing estimation
- Junjie Hao
 - Deliverable 1: 1.2 - make github repository
 - Deliverable 1: 1.3 - add all team members and TA to github repository
 - Deliverable 1: Explain software process model used
 - Deliverable 2: Conclusion
 - Deliverable 2: References
- Zachary Jordan
 - Deliverable 1: 1.6 - Include Github url in deliverable 1
 - Deliverable 1: Use Case diagram
 - Deliverable 2: Project Scheduling
- Austin Li
 - Deliverable 1: 1.4 - make commit to github
 - Deliverable 1: Class diagram
 - Deliverable 2: Describe who did everything
- Rolando Martinez
 - Deliverable 1: 1.5 - make “project_scope” file
 - Deliverable 1: Architectural Design
 - Deliverable 2: Compare work w similar designs
- Jim Moore
 - Deliverable 1: Non-Functional Requirements
 - Deliverable 2: Develop a test plan
- Kenneth Pham
 - Deliverable 1: Sequence Diagram design
 - Deliverable 2: Estimated hardware, software and personnel costs.
- Everyone
 - Presentation Slides

4. Software Process Model

Evolutionary models -Prototyping

Prototyping is employed in our project 4-way chess application. Prototyping, as a common evolutionary process model, satisfies the needs of developing different versions of the application. Iterative discussion for completing the original plan with more functions drives the process forward step by step.

A prototyping iteration perfectly fits our project. At the early stage of group discussion, all members agreed on creating a 4-players chess game for charity purposes. The structure of the application was founded. After further communication and diagraming on the primary user interface layout, the construction team would begin constructing a prototype. The prototype carrying two main functions, 4-ways chess and donation, would receive feedback from group members after its first deployment.

Further versions were completed with all essential functions and some advanced features, including login, password reset, general setting, reward, points, ELO system, leaderboard, and game store. All the new features would not be added to the prototype at one upgrade. Every one or two features are discussed and performed at one iteration. Our groups would continue creating new features and polishing existing functions.

The prototyping process model's iteration lets the developing team integrate more features into the original prototype in a more casual and more secure path. After team members touched the first prototype, we evolved more ideas of function that fit our projects. New features not only made the application more fun but also collaborated with the prototype with fewer bugs.

The prototyping process model also brings flexibility to the developing process. The group did not have to come up with a well-organized and completed plan when we started the project. Each iteration was also not time-consuming because specific separate tasks were focused. Our team constructed every step of the process fully discussed and efficiently with the prototyping model.

5a. Software Requirements

Functional Requirements

Login Access

1. The system shall allow users to login to the system or create an account with the system using a username and password.
2. The system shall allow users with their email address to recover their username/password by sending it to their email.

Leaderboard Management

1. The system shall allow users to select highly ranked players and display their information.
2. The selected player screen shall display donation amount, rank, and any events listed that involve that player.

Profile Management

1. The system shall allow the user to display their personal game record (total wins, total losses, and donations made).
2. The system shall allow the user to add/update in-game gear purchases made from the store, such as cosmetics (chess pieces, flairs, emotes), and view in game currency spent.

Play Match

1. The system shall support 4 player chess
2. The system shall support all chess moves.
3. The system shall support in game chat while playing.
4. The system shall allow the user to search for a match based on the ELO rating.

5. Selected opponents shall have their name, icons, and cosmetics displayed when the match starts.
6. The system shall display a timer for each player during the match. If the timer ends, the player's game is over, and the game grays out the timer and their chess pieces.
7. The system shall only count down the player's timer when it is their turn.
8. The system shall support and display the chess move history.
9. The system shall allow users to display emotes during the match.
10. The system shall display game results when the match is over and ELO rating progress.

In Game Store

1. The system shall allow users to watch ads for in game currency.
2. The system shall allow users to purchase in game currency.
3. The system shall allow all in game purchases made by users to go towards charity.

5b. Non-functional requirements

- *Product requirements*
 - **Usability**
 - The product should be available in at least one other language than English.
 - The product should not use color combinations that impair color-blind people's ability to use the app.
 - **Efficiency**
 - **Performance**
 - load time should be less than 1s on test phone
 - the game should run at 60fps without going above 50% cpu utilization on test phone
 - **Space**
 - the game should be less than 100 MB
 - **Dependability**
 - Mean time to failure should be greater than 20 hours of continuous play time.
 - Game should automatically handle errors and restart if needed.
 - Game servers should have >99% uptime.
 - **Security**
 - Server should verify all scores to prevent fake scores filling the leaderboard.
 - Same with donation perks.
- *Organizational*
 - **Environmental**
 - Server should be able to handle at least 1000 users on 500W of power
 - **Operational**
 - Administrators of the servers will have to have passwords longer than 12 characters that contain both letters and numbers. Passwords must be changed at least once every 6 months.
 - **Development**
 - Product should have at least 10 automatic unit tests and 10 integration tests that all pass before an update or the initial release can be deployed.
- *External*
 - **Regulatory.**
 - Any purchases that obtain in-game rewards must be deterministic (no random rewards) to avoid gambling laws.
 - Register the company as a LLC
 - **Ethical**
 - The app should not harass users with notifications or guilt them into donating.

- **Legislative**

- **Accounting**

- The app needs to log all donations in detail, including information about who donated, the amount, where the donation is going, and the date and time of the donation

- **Safety/Security**

- Payment information and records should be encrypted and not be visible to the application's servers (write-only)

6. Use case diagram

Used this on <https://yuml.me/diagram/usecase/draw> to make it:

[User]-(Login)

(Login) < (Create New Account)

(Login) < (Reset Password)

(Login) < (Reset Username)

[User]-(View Leaderboards)

(View Leaderboards) > (View Player Rank Leaderboard)

(View Leaderboards) > (View Donation Amount Leaderboard)

[User] - (View Personal Stats)

(View Personal Stats) > (View Past Donations)

(View Personal Stats) > (View Win/Loss Stats)

[User] - (Play Match)

(Play Match) > (Search for Match)

(Play Match) > (Give Post Match Rewards)

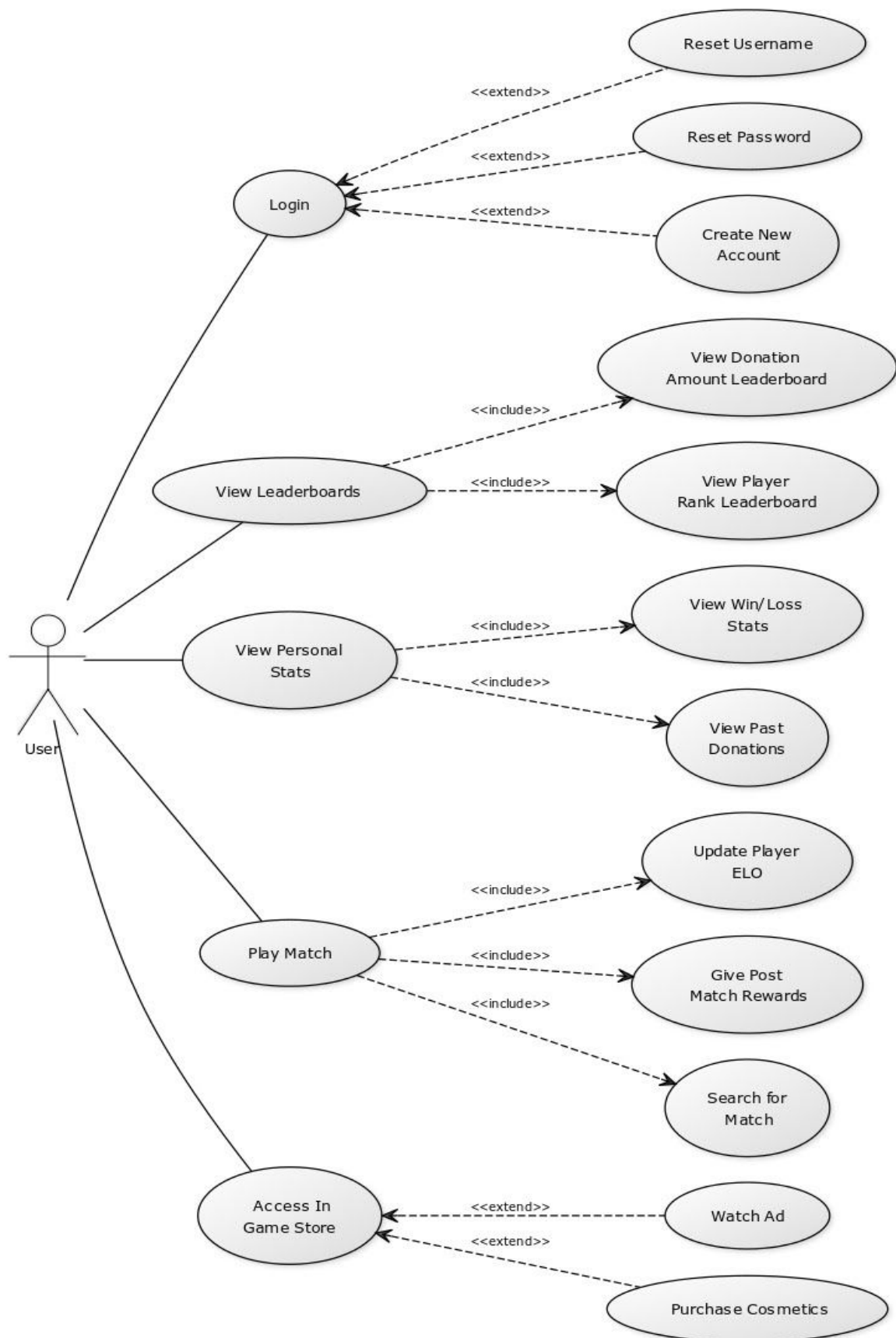
(Play Match) > (Update Player ELO)

[User] - (Access In Game Store)

(Access In Game Store) < (Purchase Cosmetics)

(Access In Game Store) < (Watch Ad)

Here is a jpeg:



7. Sequence Diagram

Created using <https://sequencediagram.org/>

//

title Reset Username

actor User

participant Mobile Application

participant Application Server

User -> Mobile Application: Click "Reset Username"

activate Mobile Application

Mobile Application -> User: Request email

User --> Mobile Application: Email entered

Mobile Application -> Application Server: Verify email

activate Application Server

alt If email is valid

Application Server --> Mobile Application: Email exists

Mobile Application -> User: Send username reset email

Mobile Application -> User: "Username reset email sent" message

else else

Application Server --> Mobile Application: Email does not exists

Mobile Application -> User: "Email not found" message

end

User -> Mobile Application: Enter new username via username reset email

Mobile Application -> Application Server: Update username

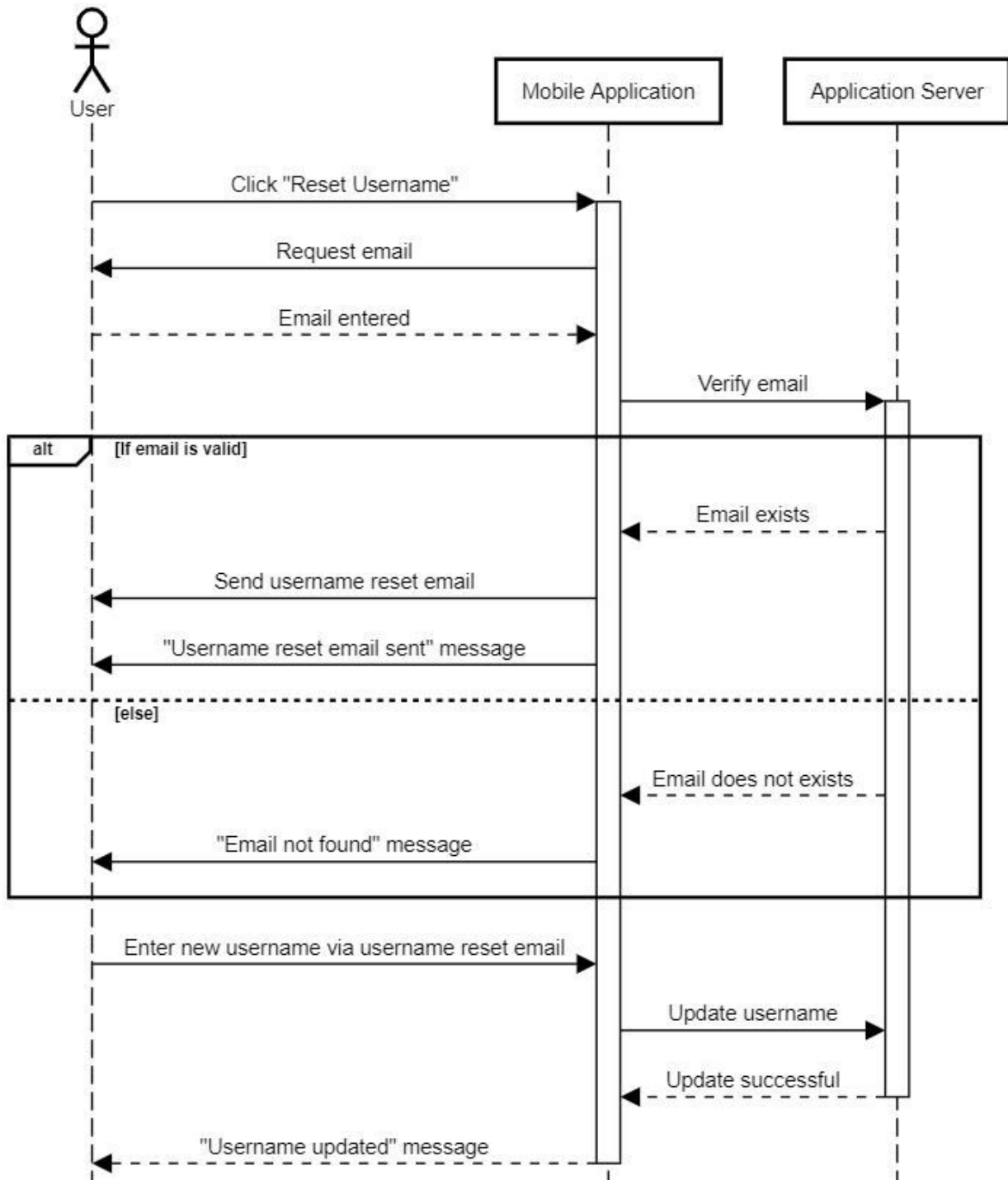
Application Server --> Mobile Application: Update successful

deactivate Application Server

Mobile Application --> User: "Username updated" message

deactivate Mobile Application

Reset Username



////////////////////////////////////

title Reset Password

actor User
participant Mobile Application
participant Application Server

entryspacing 0.9

User -> Mobile Application: Click "Reset Password"

activate Mobile Application

Mobile Application -> User: Request email

User --> Mobile Application: Email entered

Mobile Application -> Application Server: Verify email

activate Application Server

alt If email is valid

Application Server --> Mobile Application: Email exists

Mobile Application -> User: Send password reset email

Mobile Application -> User: "Password reset email sent" message

else else

Application Server --> Mobile Application: Email does not exists

Mobile Application -> User: "Email not found" message

end

User -> Mobile Application: Enter new password via password reset email

Mobile Application -> Application Server: Update password

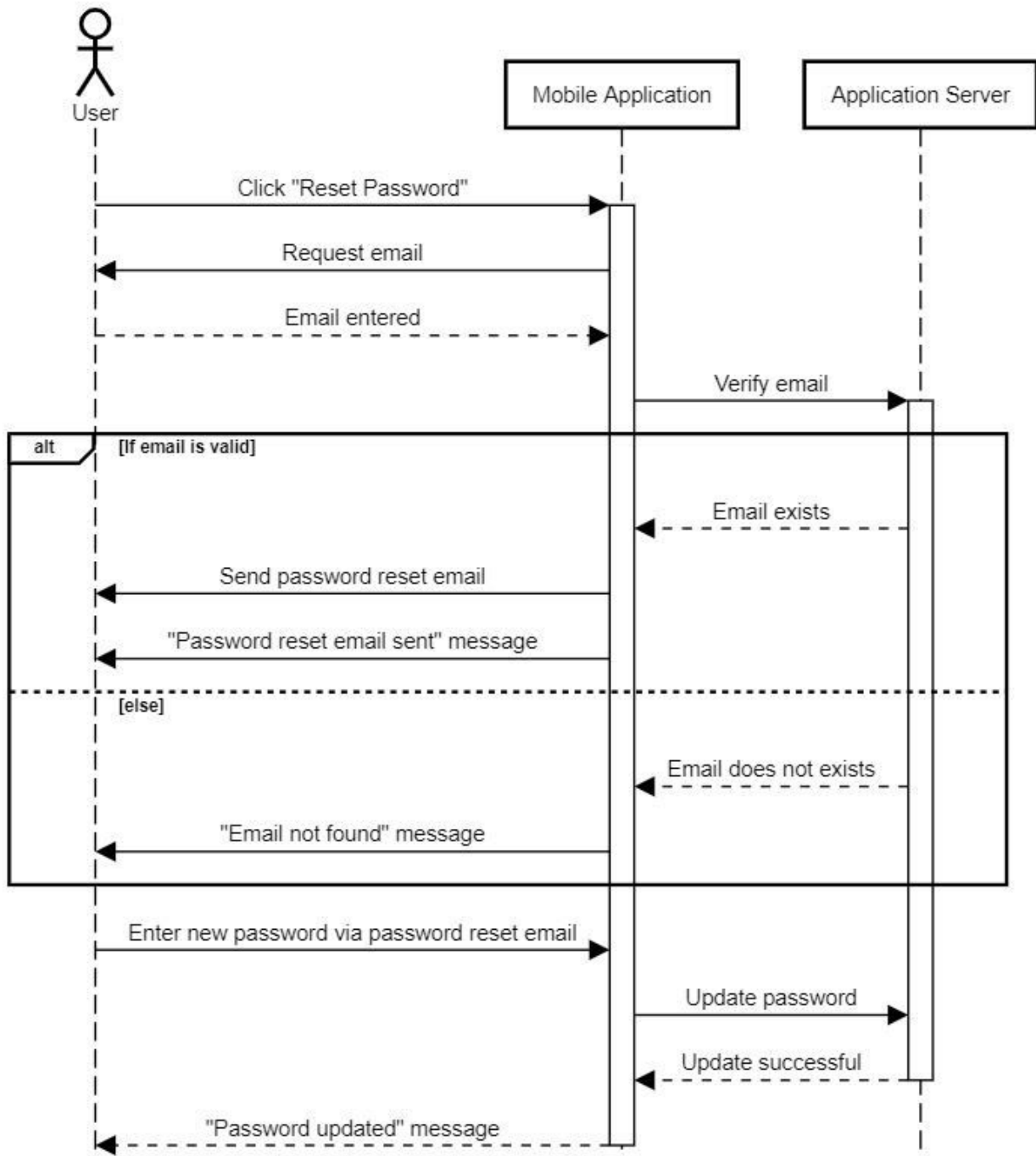
Application Server --> Mobile Application: Update successful

deactivate Application Server

Mobile Application --> User: "Password updated" message

deactivate Mobile Application

Reset Password



////////////////////////////////////

title Create New Account

actor User

participant Mobile Application

participant Application Server

entryspacing 0.9

User -> Mobile Application: Click "Create New Account"

activate Mobile Application

Mobile Application -> User: Request account details

User --> Mobile Application: Details entered

Mobile Application -> Application Server: Check for existing user

activate Application Server

alt If user already exists

Application Server --> Mobile Application: User exists

Mobile Application -> User: "User already exists" message

else else

Application Server --> Mobile Application: User does not exists

Mobile Application -> Application Server: Create new user

Application Server --> Mobile Application: User created successfully

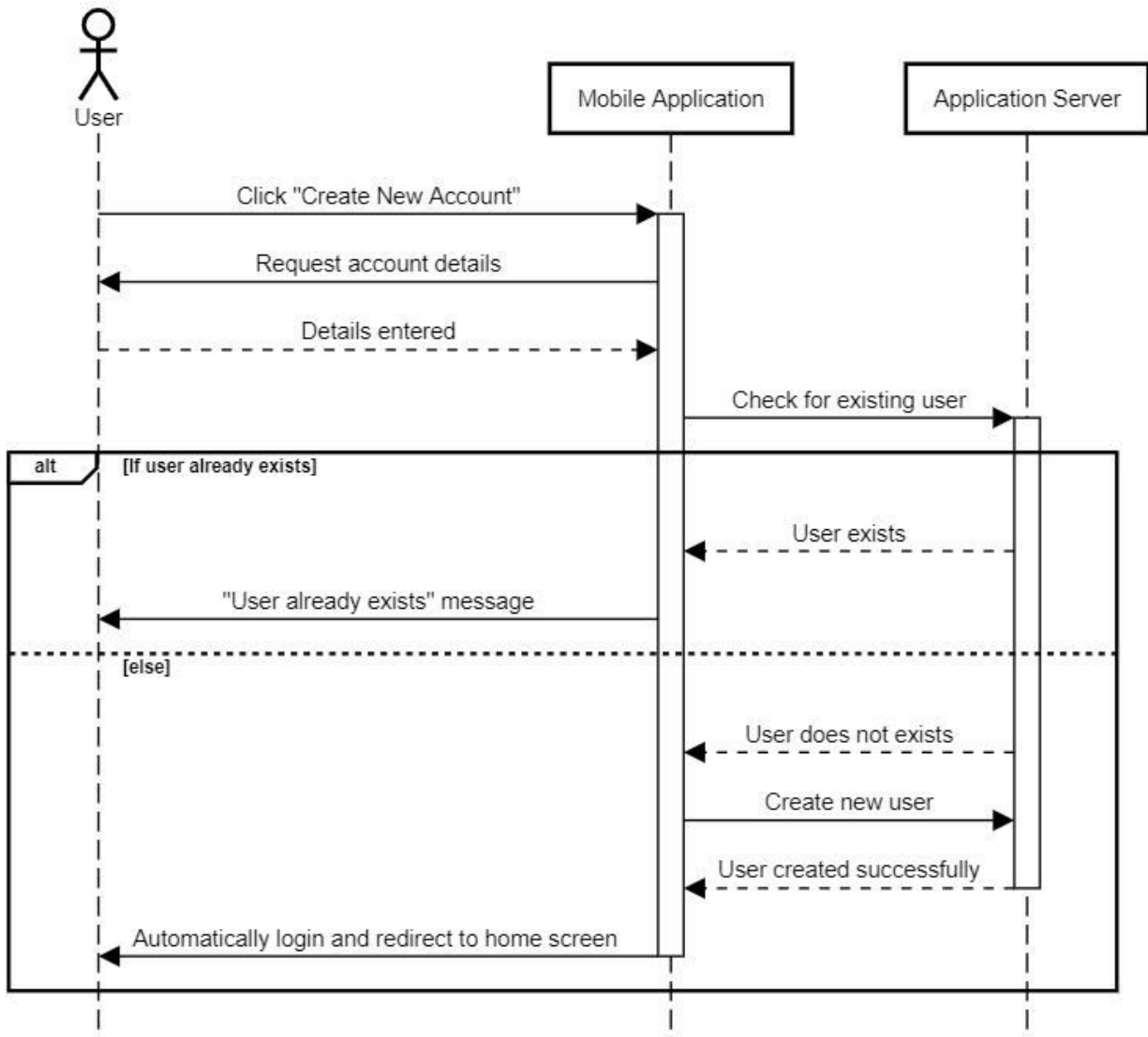
deactivate Application Server

Mobile Application -> User: Automatically login and redirect to home screen

deactivate Mobile Application

end

Create New Account



////////////////////////////////////

title Login

actor User

participant Mobile Application

participant Application Server

entryspacing 0.9

User -> Mobile Application: Enter username and password

activate Mobile Application

Mobile Application -> Application Server: Check for existing user with username and password
activate Application Server

alt If user already exists

Application Server --> Mobile Application: User exists

Mobile Application -> User: Redirect to home screen (user logged in)

else else

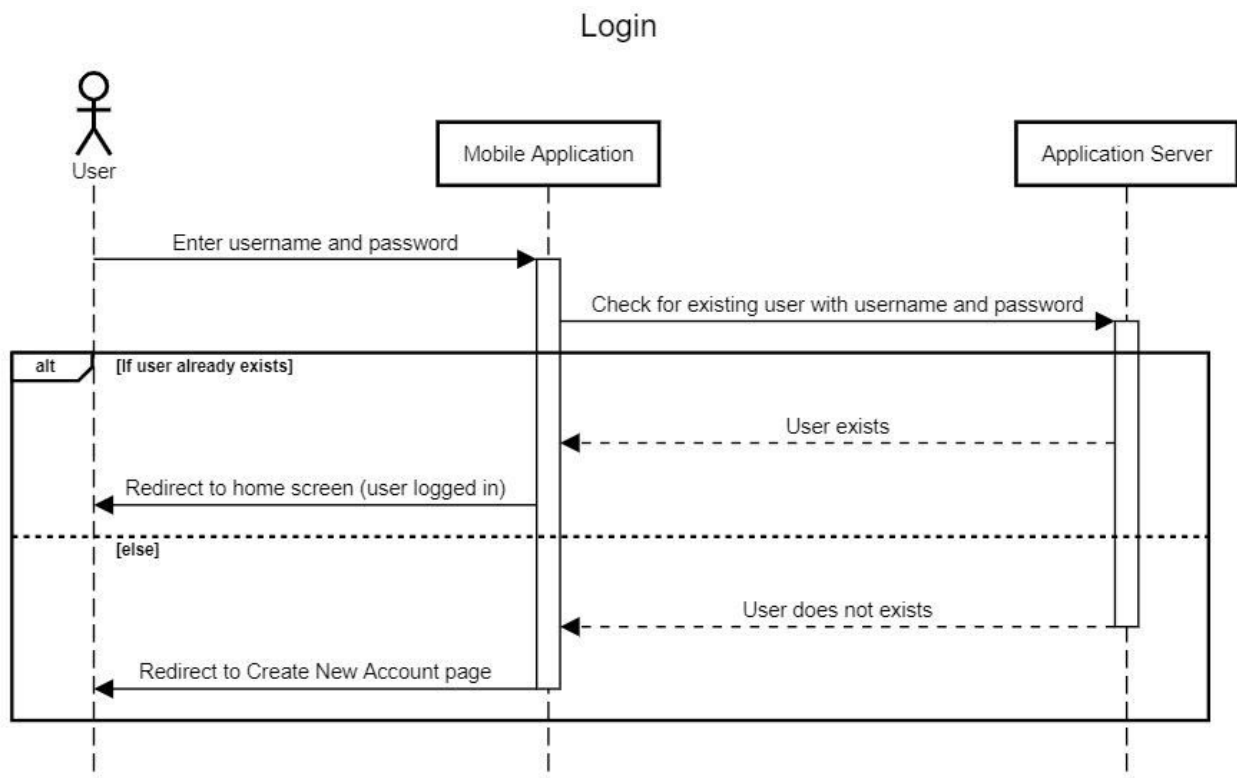
Application Server --> Mobile Application: User does not exists

deactivate Application Server

Mobile Application -> User: Redirect to Create New Account page

deactivate Mobile Application

end



////////////////////////////////////

title View Leaderboards

actor User

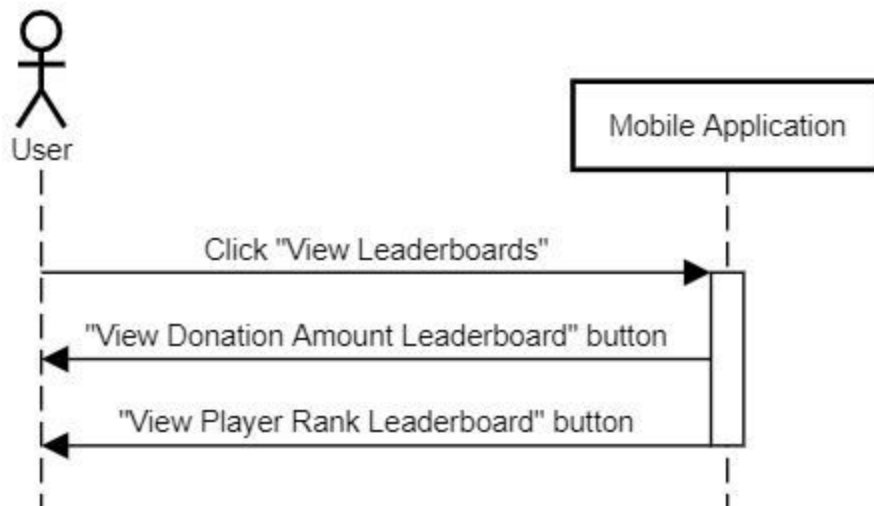
participant Mobile Application

entryspacing 0.9

User -> Mobile Application: Click "View Leaderboards"
activate Mobile Application

Mobile Application -> User: "View Donation Amount Leaderboard" button
Mobile Application -> User: "View Player Rank Leaderboard" button
deactivate Mobile Application

View Leaderboards



////////////////////////////////////

title View Donation Amount Leaderboard

actor User
participant Mobile Application
participant Application Server

entryspacing 0.9

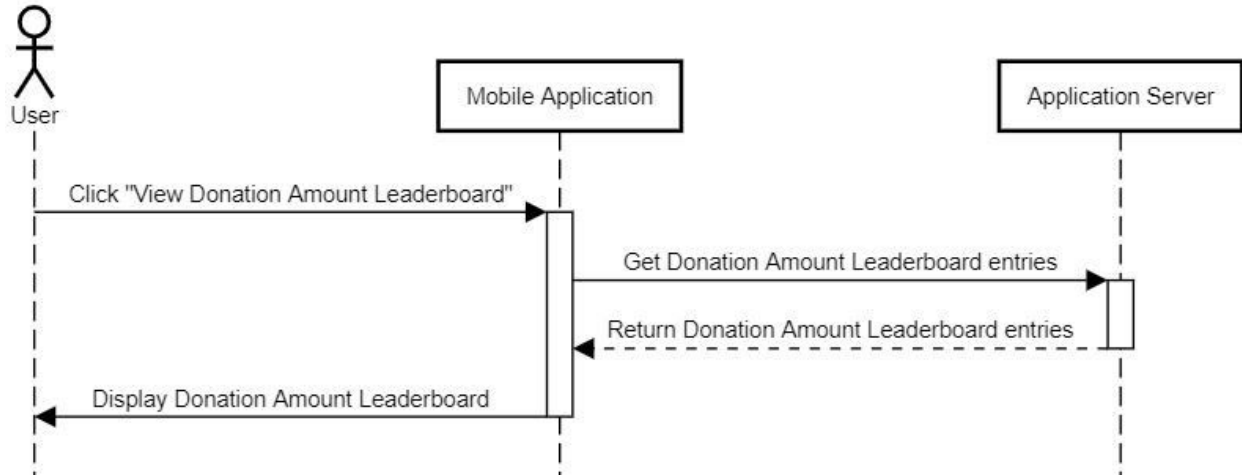
User -> Mobile Application: Click "View Donation Amount Leaderboard"
activate Mobile Application

Mobile Application -> Application Server: Get Donation Amount Leaderboard entries
activate Application Server

Application Server --> Mobile Application: Return Donation Amount Leaderboard entries
deactivate Application Server

Mobile Application -> User: Display Donation Amount Leaderboard
deactivate Mobile Application

View Donation Amount Leaderboard



////////////////////////////////////

title View Player Rank Leaderboard

actor User
participant Mobile Application
participant Application Server

entryspacing 0.9

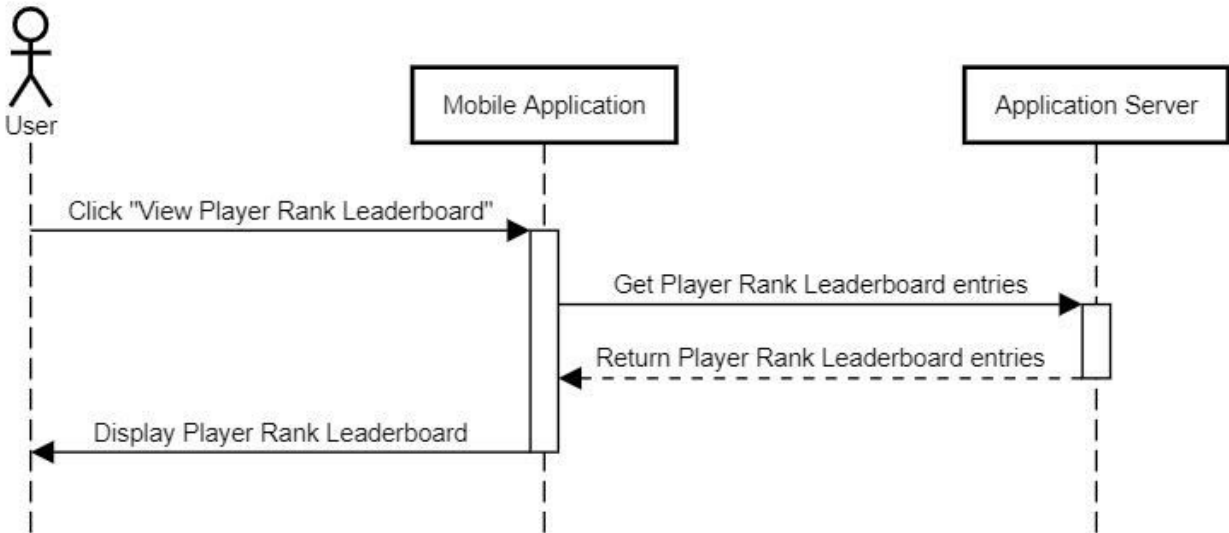
User -> Mobile Application: Click "View Player Rank Leaderboard"
activate Mobile Application

Mobile Application -> Application Server: Get Player Rank Leaderboard entries
activate Application Server

Application Server --> Mobile Application: Return Player Rank Leaderboard entries
deactivate Application Server

Mobile Application -> User: Display Player Rank Leaderboard
deactivate Mobile Application

View Player Rank Leaderboard



////////////////////////////////////

title View Personal Stats

actor User

participant Mobile Application

entryspacing 0.9

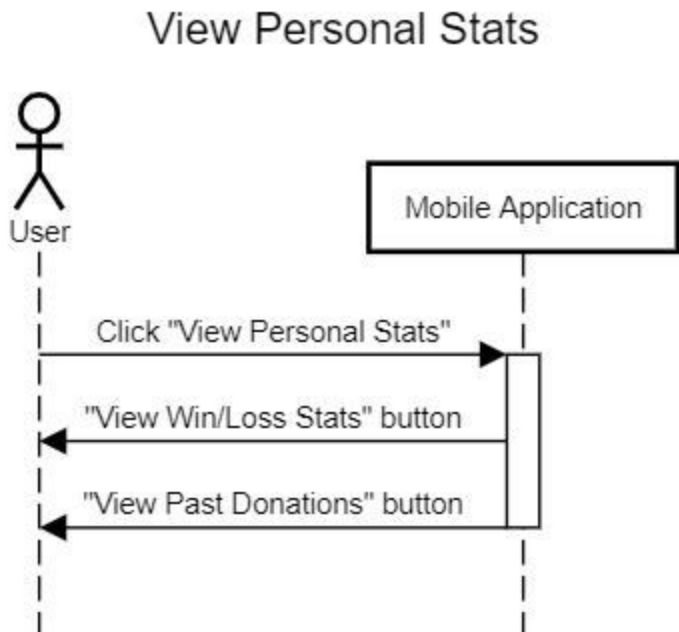
User -> Mobile Application: Click "View Personal Stats"

activate Mobile Application

Mobile Application -> User: "View Win/Loss Stats" button

Mobile Application -> User: "View Past Donations" button

deactivate Mobile Application



////////////////////////////////////

title View Win/Loss Stats

actor User

participant Mobile Application

participant Application Server

User -> Mobile Application: Click on "View Win/Loss Stats"

activate Mobile Application

Mobile Application ->Application Server: Get Win/Loss Stats

activate Application Server

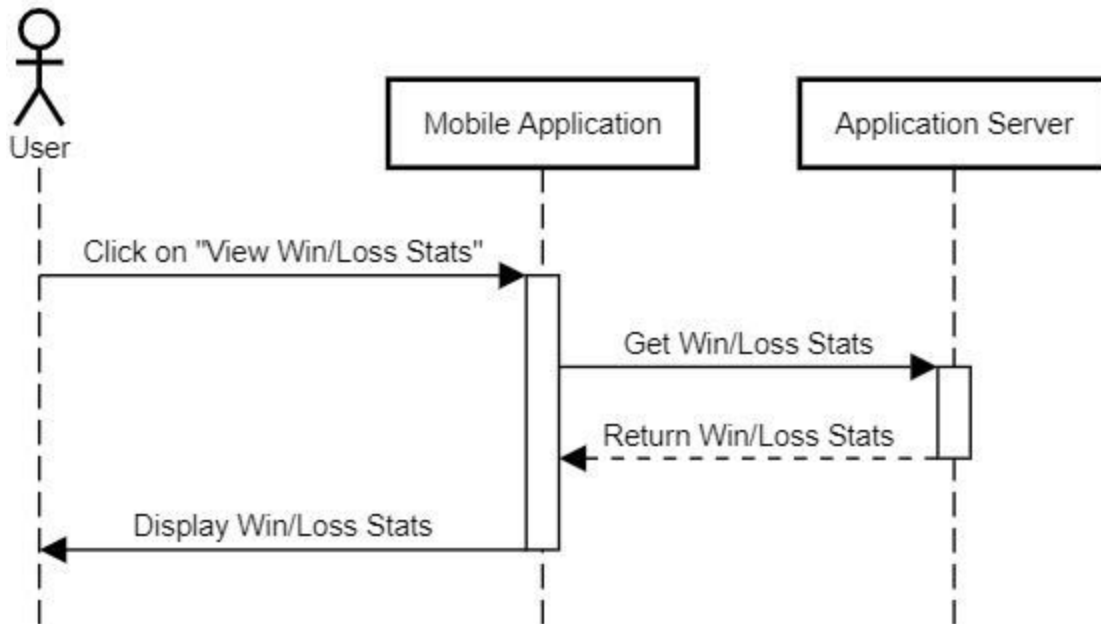
Application Server -->Mobile Application: Return Win/Loss Stats

deactivate Application Server

Mobile Application ->User: Display Win/Loss Stats

deactivate Mobile Application

View Win/Loss Stats



////////////////////////////////////

title View Past Donations

actor User

participant Mobile Application

participant Application Server

User -> Mobile Application: Click on "View Past Donations"

activate Mobile Application

Mobile Application ->Application Server: Get past donations

activate Application Server

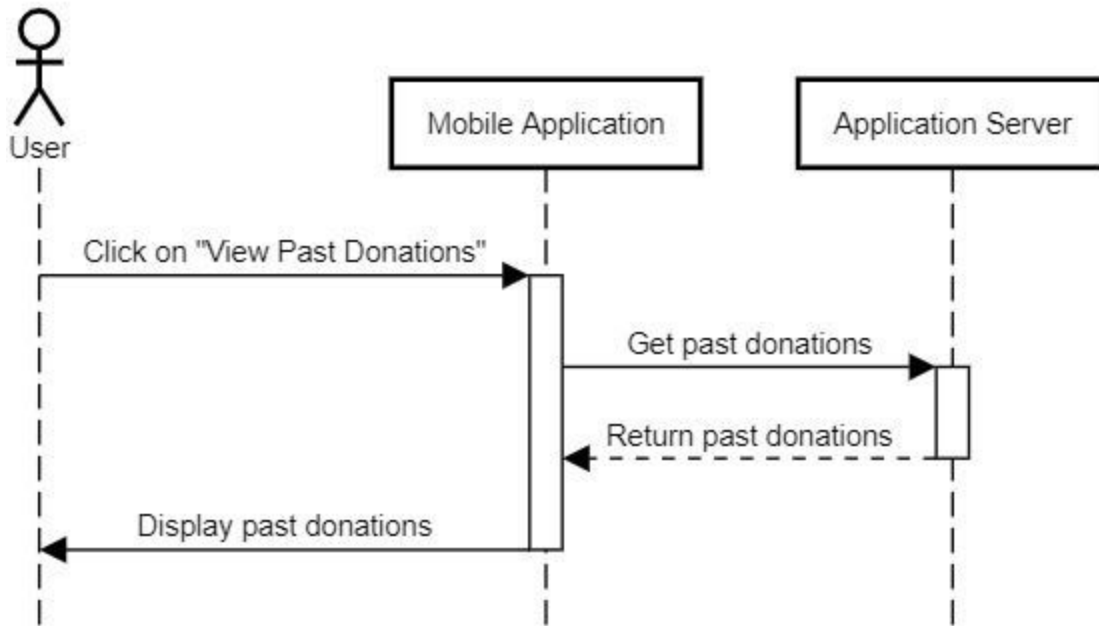
Application Server -->Mobile Application: Return past donations

deactivate Application Server

Mobile Application ->User: Display past donations

deactivate Mobile Application

View Past Donations



////////////////////////////////////

title Play Match

actor User

participant Mobile Application

participant Application Server

User -> Mobile Application: Click "Play Match"

activate Mobile Application

Mobile Application -> Application Server: Search for Match

activate Application Server

Application Server --> Mobile Application: Return match opponents

Mobile Application -> User: Start game for user

User -> Mobile Application: Play game

User -> Mobile Application: Finish game

Mobile Application -> Application Server: Calculate and store Post Match Rewards

Application Server --> Mobile Application: Return Rewards

Mobile Application -> Application Server: Update player ELO

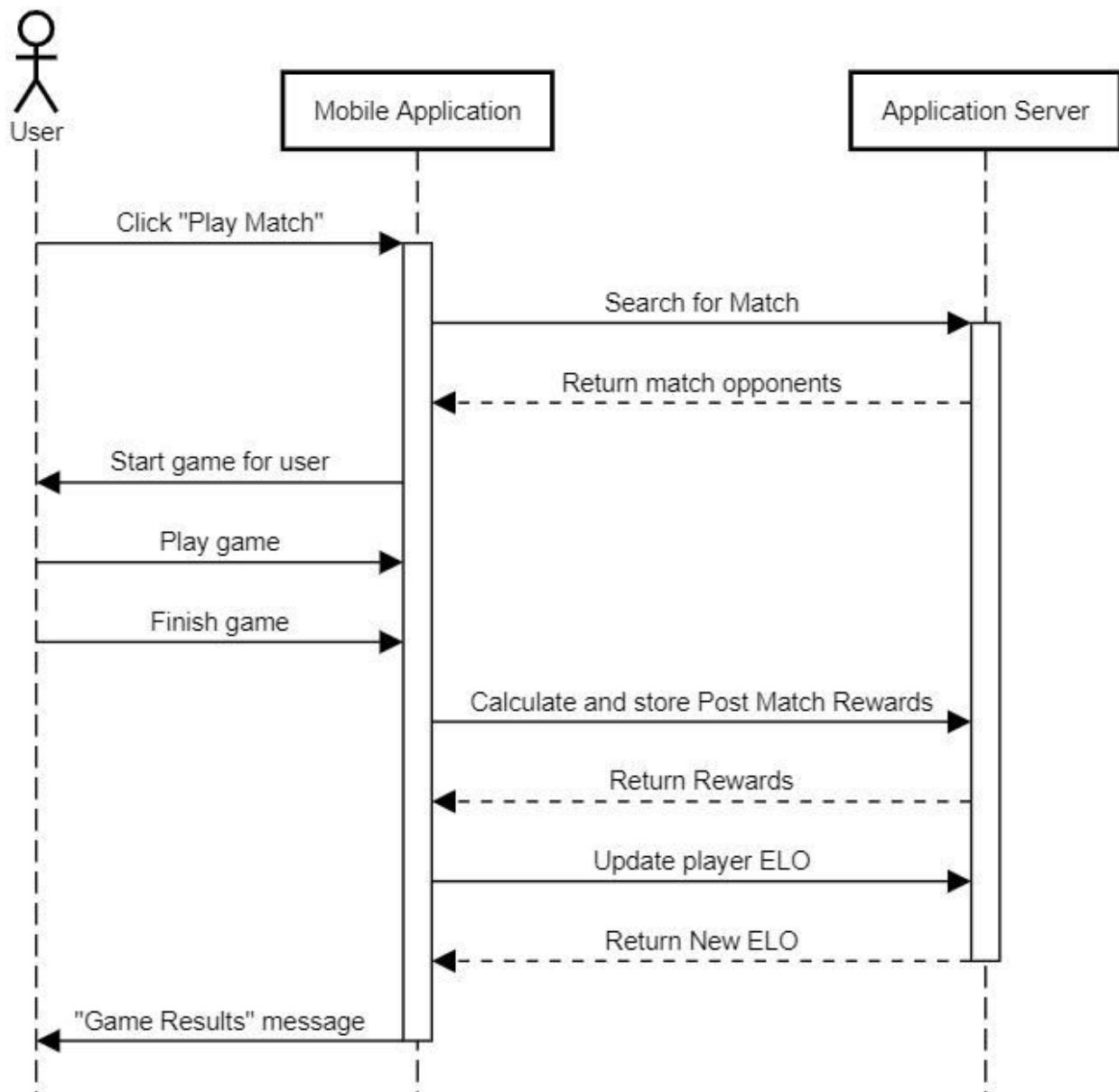
Application Server --> Mobile Application: Return New ELO

deactivate Application Server

Mobile Application -> User: "Game Results" message

deactivate Mobile Application

Play Match



//

title Update Player ELO

actor User

participant Mobile Application

participant Application Server

User -> Mobile Application: Complete Match

activate Mobile Application

alt User loses the match

Mobile Application -> Application Server: Update ELO (lower ELO)

activate Application Server

Application Server --> Mobile Application: Return new ELO

else else

Mobile Application -> Application Server: Update ELO (higher ELO)

Application Server --> Mobile Application: Return new ELO

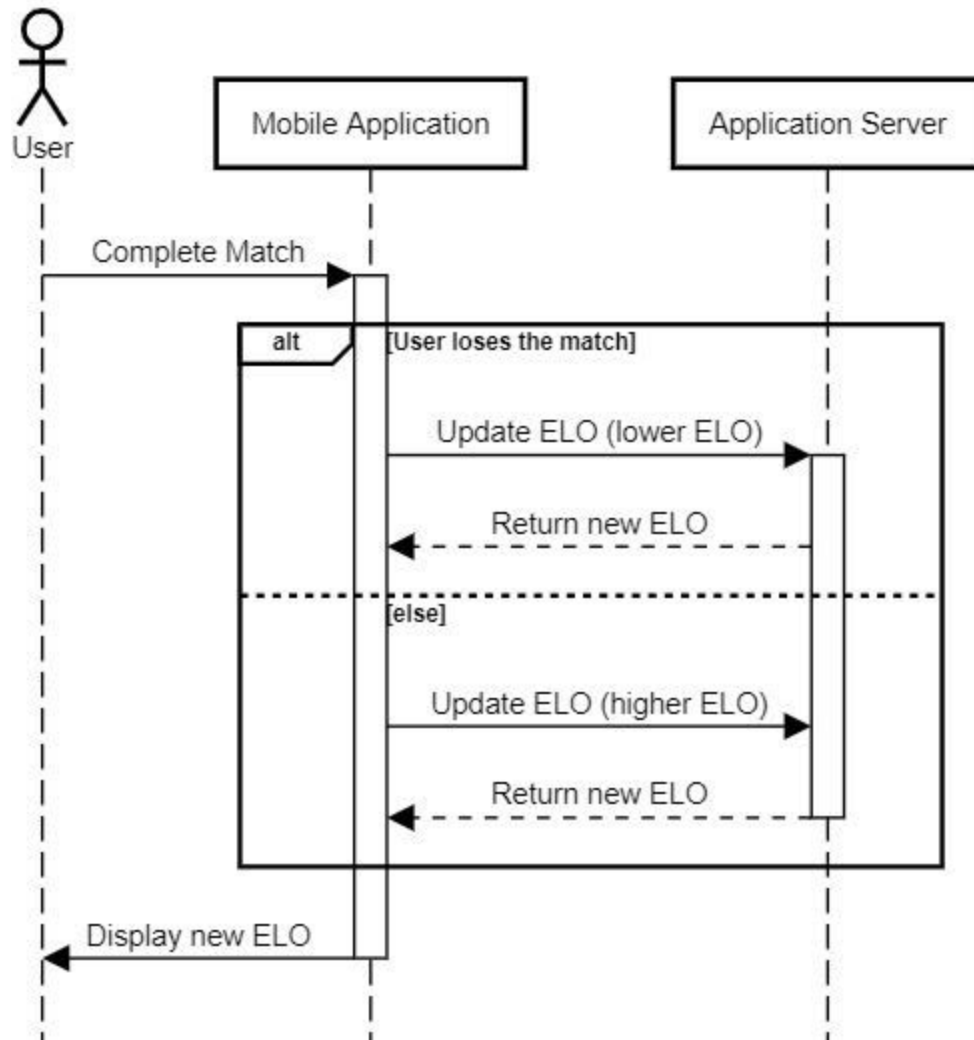
deactivate Application Server

end

Mobile Application -> User: Display new ELO

deactivate Mobile Application

Update Player ELO



////////////////////////////////////

title Give Post Match Rewards

actor User

participant Mobile Application

participant Application Server

User -> Mobile Application: Complete Match

activate Mobile Application

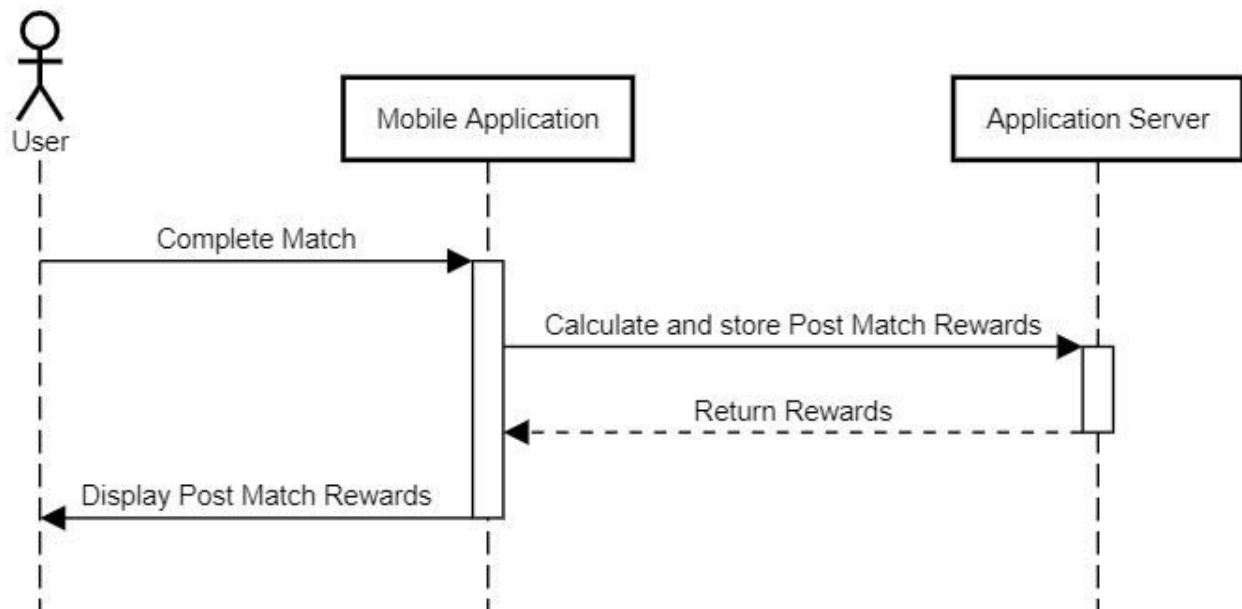
Mobile Application -> Application Server: Calculate and store Post Match Rewards

activate Application Server

Application Server --> Mobile Application: Return Rewards
deactivate Application Server

Mobile Application -> User: Display Post Match Rewards
deactivate Mobile Application

Give Post Match Rewards



////////////////////////////////////

title Search for Match

actor Actor
participant Mobile Application
participant Application Server

Actor -> Mobile Application: Click "Play Match"
activate Mobile Application

Mobile Application -> Application Server: Find players of similar ELO
activate Application Server

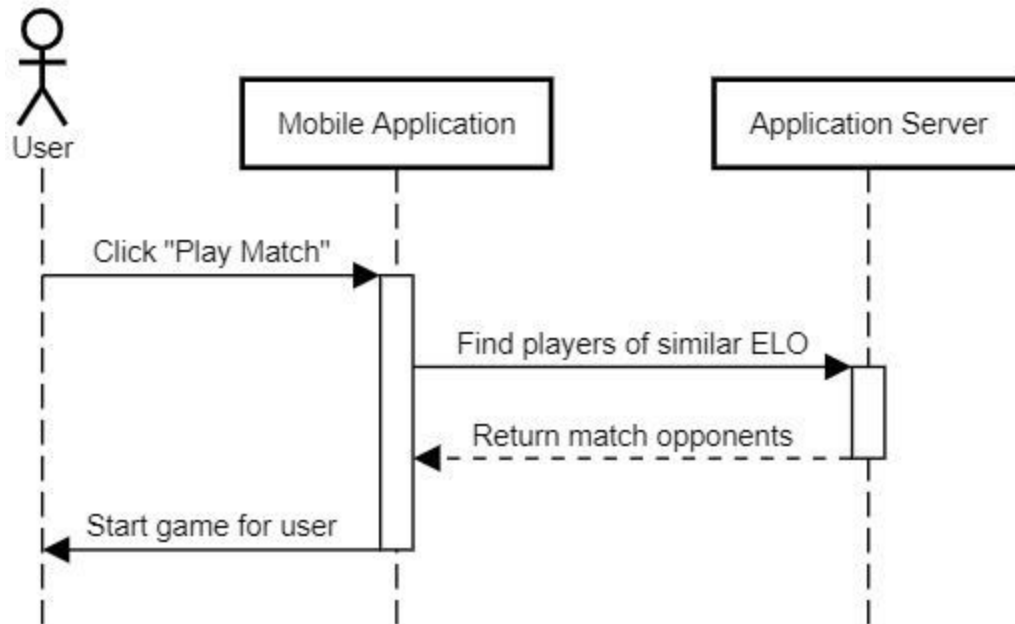
Application Server --> Mobile Application: Return match opponents

deactivate Application Server

Mobile Application -> Actor: Start game for user

deactivate Mobile Application

Search for Match



////////////////////////////////////

title Access In Game Store

actor User

participant Mobile Application

participant Application Server

User -> Mobile Application: Click "Store"

activate Mobile Application

Mobile Application -> User: Display "Watch Ad" button

Mobile Application -> Application Server: Get Cosmetics

activate Application Server

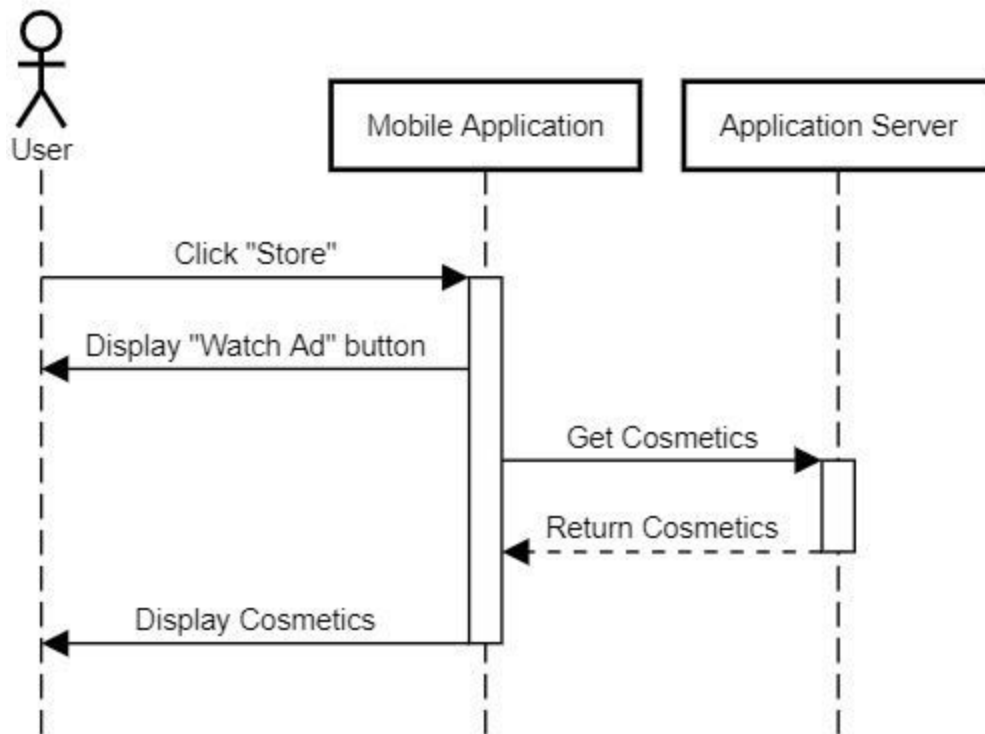
Application Server --> Mobile Application: Return Cosmetics

deactivate Application Server

Mobile Application -> User: Display Cosmetics

deactivate Mobile Application

Access In Game Store



////////////////////////////////////

title Watch Ad

actor User

participant Mobile Application

participant Application Server

User -> Mobile Application: Click "Watch Ad"

activate Mobile Application

Mobile Application -> Application Server: Get Ad

activate Application Server

Application Server --> Mobile Application: Return Ad

Mobile Application -> User: Display Ad

User -> Mobile Application: Exits Ad

alt If User watches full ad

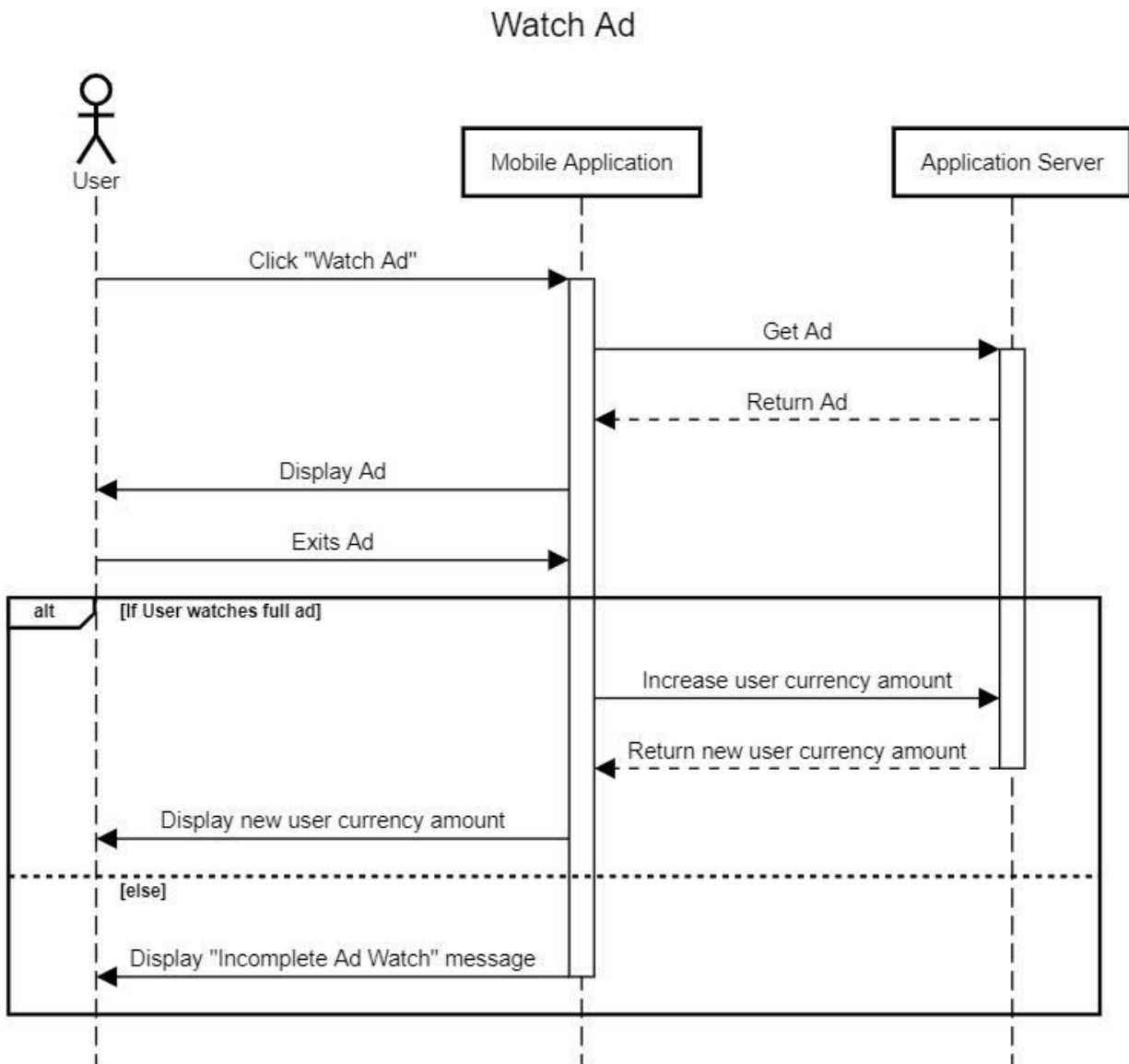
Mobile Application -> Application Server: Increase user currency amount

Application Server --> Mobile Application: Return new user currency amount
deactivate Application Server
Mobile Application -> User: Display new user currency amount

else else

Mobile Application -> User: Display "Incomplete Ad Watch" message
deactivate Mobile Application

deactivate Application Server
end



//

title Purchase Cosmetics

actor User

participant Mobile Application

participant Application Server

participant Financial services

User -> Mobile Application: Click on cosmetic item

activate Mobile Application

Mobile Application -> User: Get payment method

User --> Mobile Application: Payment method entered

Mobile Application ->Financial services: Verify payment and transaction

activate Financial services

alt If payment is valid

Financial services -->Mobile Application: Payment and transaction OK

Mobile Application -> Application Server: Update user inventory

activate Application Server

Application Server --> Mobile Application: Update inventory successful

deactivate Application Server

Mobile Application -> User: Display "Purchase Successful" message

else else

Financial services -->Mobile Application: Payment and transaction invalid

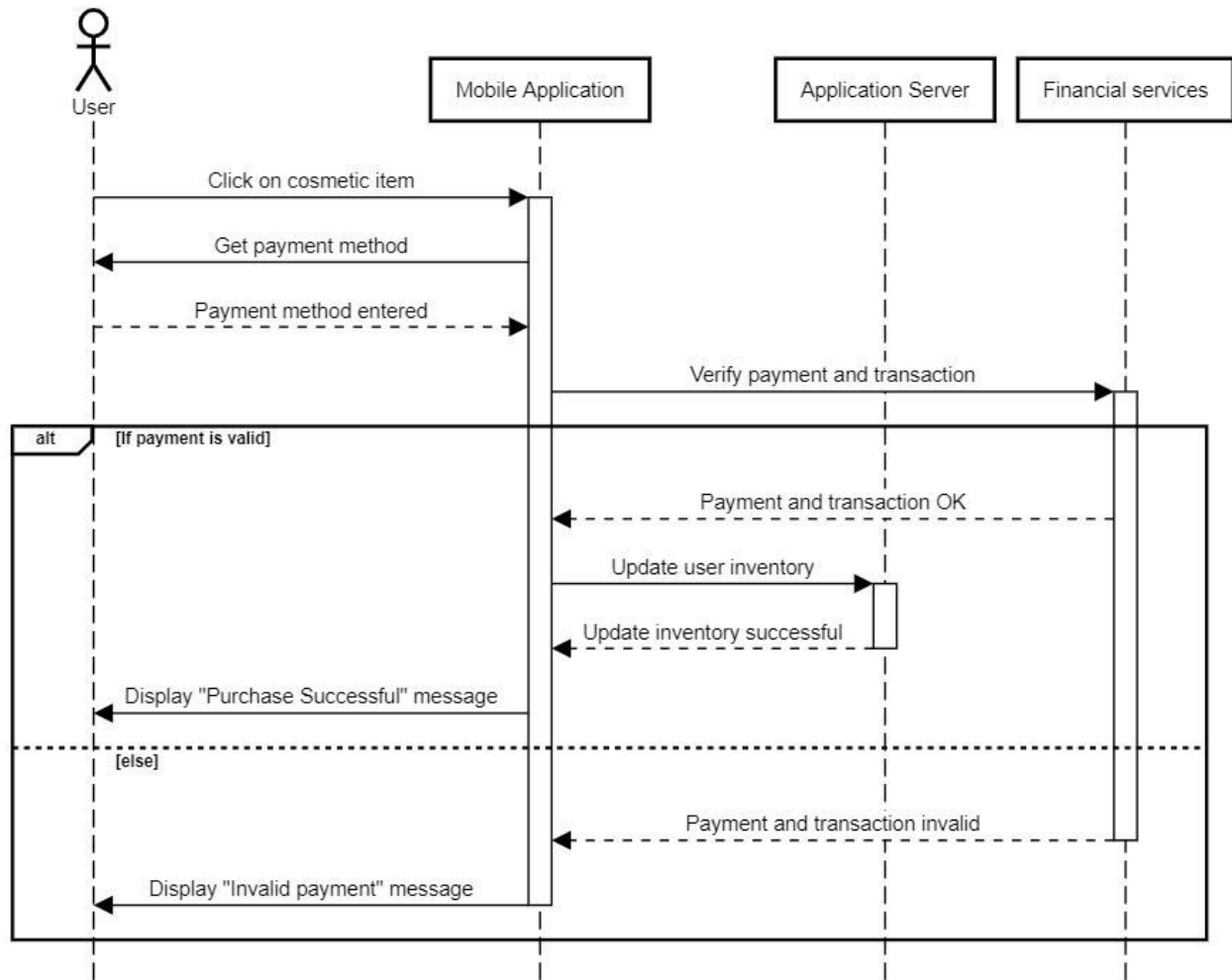
deactivate Financial services

Mobile Application ->User:Display "Invalid payment" message

deactivate Mobile Application

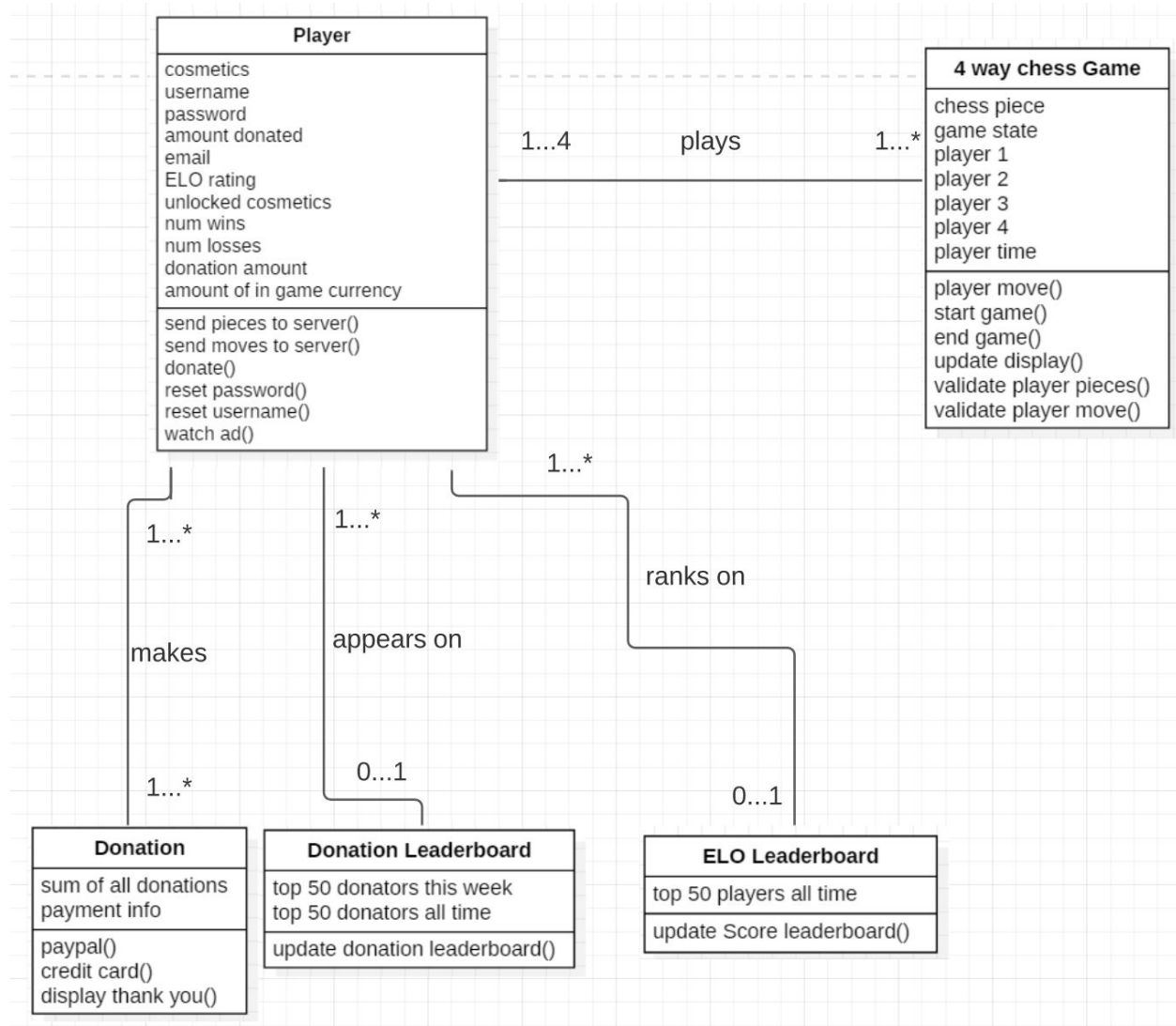
end

Purchase Cosmetics



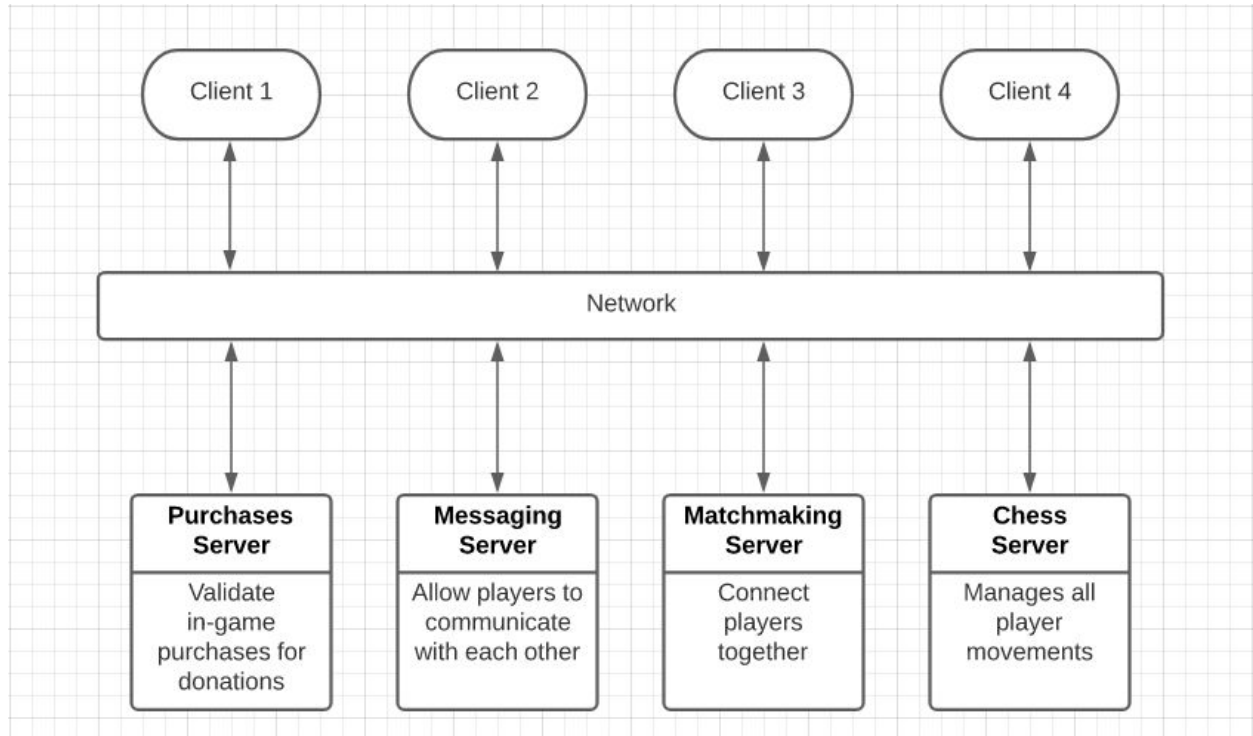
////////////////////////////////////

8. Class Diagram



9. Architectural Design:

Client-server architecture pattern:



--- End of Project Deliverable 1 Content ---

3. Presentation Option

We are going with option 2, the recorded presentation.

Here is a link to the videos

- No subtitles: <https://youtu.be/kyVme6OJ5OI>
- Subtitles: <https://youtu.be/rW3BrPl0Fno>

4. Project Scheduling, Cost, Effort and Pricing Estimation, Project duration and staffing

4.1 Project Scheduling

		Timeline (by week)							
		Dec 7	Dec 14	Dec 21	Dec 28	Jan 4	Jan 11	Jan 18	Jan 22
Tasks	Login Functionality								
	- Create Account								
	- Reset Username								
	- Reset Password								
	Account Stats Functionality								
	- Donation Stats								
	- Win/Loss Stats								
	- Player Rank Stats								
	Leaderboard Functionality								
	- Donation Leaderboard								
	- Player Rank Leaderboard								
	Play Game Functionality								
	- Play Game								
	- Search For Game								
	- Forfeit Game								
	- Win/Lose Game								
	- Update Player Stats								
	In-App Store Functionality								
	- Play Ad								
	- Display Items To User								
	- Process Purchase								

Project Scheduling Decisions:

Our project will be scheduled to begin on December 7, 2020. Our scheduling information is based on a team of 7 students, and assumes that each student is limited to 10 hours/week (2 hours/day), and cannot work on weekends. Based on these assumptions, shown below are the calculations for the estimated effort and length of the project, which gives a resulting estimated duration of 7 weeks. Based on this project duration estimate, the scheduled project end date is January 22, 2021.

4.2 Cost, Effort and Pricing Estimation.

For the cost, effort, and pricing estimation of the 4 Way Chess for Charity Software Project we decided to use the Function Point Method. A Function Point (FP) is a unit for which we measure the functional size of the software. For instance, while we are building our application we will be able to measure the functional size in terms of function points which is proportional to the effort.

The general approach for the Function Point Method is as follows:

1. Determine Function Category Count.

- The 4 way Chess for Charity Software is estimated to have the following function category counts:
 - Number of user inputs = 12
 - Number of user outputs = 6
 - Number of user queries = 15
 - Number of data files and relational tables = 25
 - Number of external interfaces = 7

2. Determine the Complexity.

- For each function type identified in the Function Category listed above is further classified as simple, average or complex and a weight is given to each:
 - Number of user inputs = 12, complexity = simple.
 - Number of user outputs = 6, complexity = average.
 - Number of user queries = 15, complexity = average.
 - Number of data files and relational tables = 25, complexity = average.
 - Number of external interfaces = 7, complexity = simple.

3. Compute gross function point (GFP).

- To calculate GFP is by multiplying each function type counts by its complexity weight.

Using the provided information above The Calculated Gross Function Point for the Project can be counted as follows:

	Function Category	Count	Complexity			Count x Complexity
			Simple	Average	Complex	
1	Number of user input	12	3	4	6	36
2	Number of user output	6	4	5	7	30
3	Number of user queries	15	3	4	6	60
4	Number of data files and relational tables	25	7	10	15	250
5	Number of external interfaces	7	5	7	10	35
Gross Function Point						411

4. Determine Processing Complexity (PC).

- The PC value is determined based on responses to questions called complexity weighting factors.
- Each complexity weighting factor is assigned a value that ranges between 0 and 5 as follows: 0 (no influence), 1 (incidental), 2 (moderate), 3 (average), 4 (significant), and 5 (essential).

The complexity weighting factor with the complexity adjustment value and its total shows in the table below:

Number	Complexity Weighting Factor	Value
1	Backup and Recovery	4
2	Data Communications	3
3	Distributed Processing	2
4	Performance Critical	3
5	Existing Operating Environment	2
6	Online Data Entry	3
7	Input Transaction over Multiple Screens or Operations	3
8	Master Files Updated Online	3
9	Information Domain Values Complex	3
10	Internal Processing Complex	3
11	Code Designed for Reuse	1
12	Conversion/Installation in Design	3
13	Multiple Installations	4
14	Application Designed for Change	4
Total Processing Complexity		41

5. Compute Processing Complexity Adjustment (PCA).

- PCA is given by using the formula:
 - $PCA = 0.65 + 0.01 \text{ (Total Processing Complexity)}$, where 0.65 is an empirical constant.

$$PCA = 0.65 + 0.01 (41) = 0.65 + 0.01 * 41 = 1.06$$

6. Compute Function Point (FP) using the formula: $FP = GFP * PCA$.

Using the values already calculated previously: $FP = 411 * 1.06 = 436$

The following is a summary of all the calculations:

The Gross Function Point:

$$GFP = 12 * 3 + 6 * 5 + 15 * 4 + 25 * 10 + 7 * 5 = 411 \text{ FP}$$

The Processing Complexity Adjustments:

$$PCA = 0.65 + 0.01(41) = 1.06$$

The Function Point Calculation:

$$FP = GFP * PCA = 411 * 1.06 = 436$$

The estimated effort is obtained as:

$$E = FP / \text{productivity} = 436 / 10 = 43.6 \approx 44 \text{ person-weeks}$$

If team size = 7, then project duration is:

$$D = E / \text{team size} = 44 / 7 = 6.28 \approx 7 \text{ weeks.}$$

4.3. Estimated cost of hardware products

The estimated cost of our hardware products which include the game server and server maintenance is \$3,600.

4.4. Estimated cost of software products

The estimated cost of our software products which include development tools, libraries used, and licensed software, data encryption tools, is \$1,139.

- The cost is reduced significantly considering the open source software available for chess applications.

4.5. Estimated cost of personnel

The estimated cost of personnel which include the employees that will code the final product and cost of training is \$50,252.

5. Testing Plan

- We will be using black-box testing with JUnit to unit test our application.
- We will also use JUnit to run a separate class as the driver for bottom-up integration testing.
- JUnit is a good choice for regression testing as well. Our previous tests are all written using JUnit, so we can just run them again to regression test.
- For validation testing: first, we will release the application to a hand-picked group of alpha testers of varying backgrounds and technology skill levels. Next, after increasing publicity as we approach launch, we will create a closed beta where testers are selected via an email lottery.
- After these phases of testing are complete, the application will be ready to launch.

Example Unit Test

- The example test is buying an item from the store. The store checks if the item is listed in the store, if it is in stock, and if the account balance is high enough to purchase it. If all these conditions are met, the item transfers from the store into the purchaser's account, and their balance and the store stock are updated.
 - There are two types of items: finite stock or infinite stock. Finite stock items eventually run out after a certain number are purchased and are removed from the store. This never happens with infinite stock.
- This test was conducted using equivalence partitioning.
 - Three main partitions that have subpartitions:
 - Invalid items (items that are not found in the store)
 - Fail when Item quantity = 0
 - Fail Item quantity < 0
 - Finite items
 - Fail with insufficient funds
 - Succeed with sufficient funds (make sure stock changes)
 - Fail when stock runs out
 - Succeed when purchasing Integer.MAX_VALUE items
 - Infinite items
 - Fail with insufficient funds
 - Succeed with sufficient funds (make sure stock doesn't change)

Results

- All tests successfully passed.
- This unit test caught an error I made when programming the store. Initially, infinite stock items were not being added to the store since their numerical stock was set to 0. The store thought that they were out of stock and immediately removed them.
 - The test made this bug easy to find and easy to fix. It was caused by a typo, but due to JUnit telling me which line of the test failed, I was able to immediately locate which class and method caused this logic error.
 - Without a test, I would have had to spend a significant amount of time debugging to find the cause of this error.
- With the note that this test will be able to stay in the codebase as a regression test, writing it and similar tests is definitely worth the time.

6. Comparison of work with similar designs.

COMPARATIVE ANALYSIS REPORT

Introduction

The purpose of this report is to compare our *4 Way Chess for Charity* application to similar designs in the field of our project domain. For our comparative analysis, we focused on mobile applications that made users perform certain tasks, such as playing a game, to donate to a certain charity. This analysis does not include charity apps that only allow the user to directly donate to a charity; there should be some sort of task that users can perform that indirectly gives to charities. In addition, charity games that have not been disturbed to mobile devices are excluded from our analysis.

Discussion

During our analysis, we found that the vast majority of charity-giving mobile applications were focused on tasks of self-improvement, such that both the user and the charity of choice are being helped by the usage of the app. Mobile applications such as *Impact*, *Sentiv*, and *Charity Miles* reward physical exercise (mainly by walking and running) with donations to charity [2, 5, 6]. It can be assumed that these applications are attempting to encourage regular exercise with donations to the user's favorite charity – along with the health benefits of exercising. A somewhat similar incentive is present [4] in *Freerice*, which allows users to answer educational trivia, such as “English Vocabulary” and “World Capitals,” to donate grains of rice and other food portions for each correct answer. Like the exercise for charity apps, it can be assumed that applications like *Freerice* are trying to incentivize trivial pursuit as a way to help families from around the world [4]. In comparison, our *4 Way Chess for Charity* application is largely reliant on its novelty as an alternative variant of chess. Although there is certainly an intellectual and strategic component to regular chess, 4-way chess is a game that is mainly played for entertainment purposes and lacks a competitive scene. This is due to the nature of 4-way chess, such that capturing pieces almost always puts the player in a disadvantaged state. This is because the chess piece is always at risk from the other three players. Therefore, with a game as niche as 4-way chess, the game may lack a steady player base and consistent charity donations.

Aside from the self-improvement apps, other charity apps employed other approaches. *Charity Truck* [3] is a clicker game that simulates a virtual charity truck that helps people in need; charity donations are made from microtransactions, such that 100% of all profits are given to charity. It should be noted that, although *Charity Truck* is similar to our application in terms of purpose, the game has a dismal download count (as of November of 2020), with slightly more than ten downloads since its release in July of 2020 [3]. Another charity game with a different approach is *GoodBookey*, which is a sports betting game, where the loser has to pay for the winner's choice of charity [1]. Here, instead of being motivated by self-improvement or gaming purposes,

the user is motivated by competition: the desire to win against others. On top of that, the winner's choice of charity gets donated to.

Conclusion

After our comparative analysis of several charity apps, we concluded that, due to the differing motivational factor in our application from other similar market applications, implementing and releasing the app will carry a certain degree of risk, such that the reliance on the application's novelty may result in an inconsistent player base and therefore a lack of charitable donations.

7. Conclusion

Overall, as a seven-member team, we have accomplished the most projected goals and constructed a workable plan for our 4-way Chess application. Every team member has fulfilled personal tasks commendably.

Prototyping has been applied for the whole software process. We had a group meeting every Friday after the semester. Even though the main purpose of raising charity has not been shifted, the final version of the project was totally different from the original one with more User Interface, excitement system, and revenue functions. Functional requirements and non-functional requirements were both well described and well thought. We came up with 19 functional requirements covering from login accessing to match play, which constructed every application component. We made 18 non-functional requirements, including usability, efficiency, dependability, security, environment, operation, development, regulatory, ethical, and legislative. The Case diagram, Sequence diagram, and Class diagram were all well-described and represented by accurate charts. Especially, 17 Sequence diagrams have been drawn by our group members.

After the phrase of deliverable 1, our group has worked on Similar Apps Comparison, Project Scheduling, Cost Estimation, and Testing Plan. Several mobile applications have been compared and analyzed, including Impact, Senti, Charity Miles, and Freerice. The comparison stated the advantages of our 4 ways chess for charity and the risk of releasing an application to a market carrying potential competitors. Based on some level of assumption and research, the project was scheduled to accomplish in 7 weeks, and the total cost would be 50,252 dollars. Function Point Method was applied to estimated effort. Parameters of GFP and PCA were discussed and verified by a group meeting. The final result of GFP and PCA are 411 and 1.06. After that, the Presentation was prepared by all group members together.

In conclusion, our group delivered a decent project of developing 4-way chess for charity applications. If we have to point out one aspect needed to improve, it must be building a path to cooperate with charitable organizations. Charity work is more than collecting donations and spending them out. A cooperation with partners in different fields and focusing on target groups require our application or our team to reach beyond software design.

8. References

References List

- [1] “About - GoodBookey,” *Goodbookey.com*, 25-Apr-2016. [Online]. Available: <http://goodbookey.com/about/>. [Accessed: 07-Nov-2020].
- [2] “Charity Miles app,” *Charitymiles.org*, 19-Mar-2018. [Online]. Available: <http://charitymiles.org/>. [Accessed: 07-Nov-2020].
- [3] “Charity truck press release,” *Starmetalgames.com*. [Online]. Available: <http://www.starmetalgames.com/charity-truck-press-release.html>. [Accessed: 07-Nov-2020].
- [4] “Freerice,” *Freerice.com*. [Online]. Available: <http://freerice.com/categories/english-vocabulary>. [Accessed: 07-Nov-2020].
- [5] A. Sengupta, “Generate charity by walking,” *Impactapp.in*. [Online]. Available: <http://impactapp.in/>. [Accessed: 07-Nov-2020].
- [6] “Sentiv,” *Makemydonation.org*. [Online]. Available: <http://makemydonation.org/portfolio/sentiv/>. [Accessed: 07-Nov-2020].