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

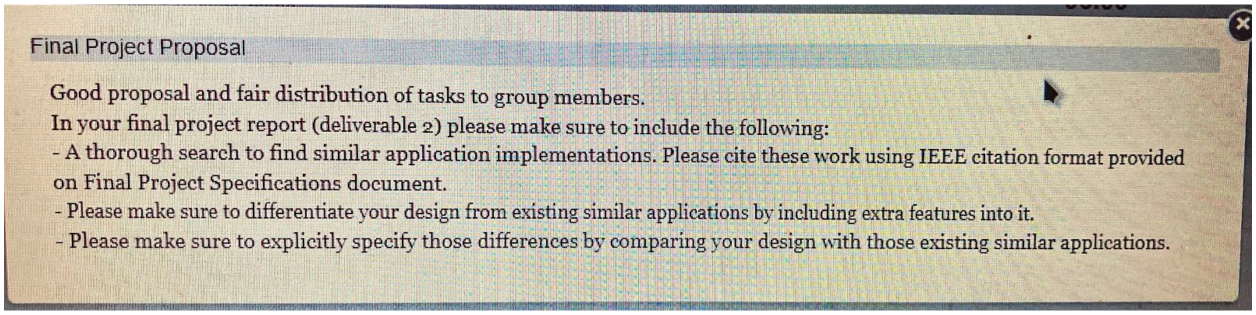
Kenneth Pham

Table of Contents

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.



Final Project Proposal

Good proposal and fair distribution of tasks to group members.
In your final project report (deliverable 2) please make sure to include the following:
- A thorough search to find similar application implementations. Please cite these work using IEEE citation format provided on Final Project Specifications document.
- Please make sure to differentiate your design from existing similar applications by including extra features into it.
- Please make sure to explicitly specify those differences by comparing your design with those existing similar applications.

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)

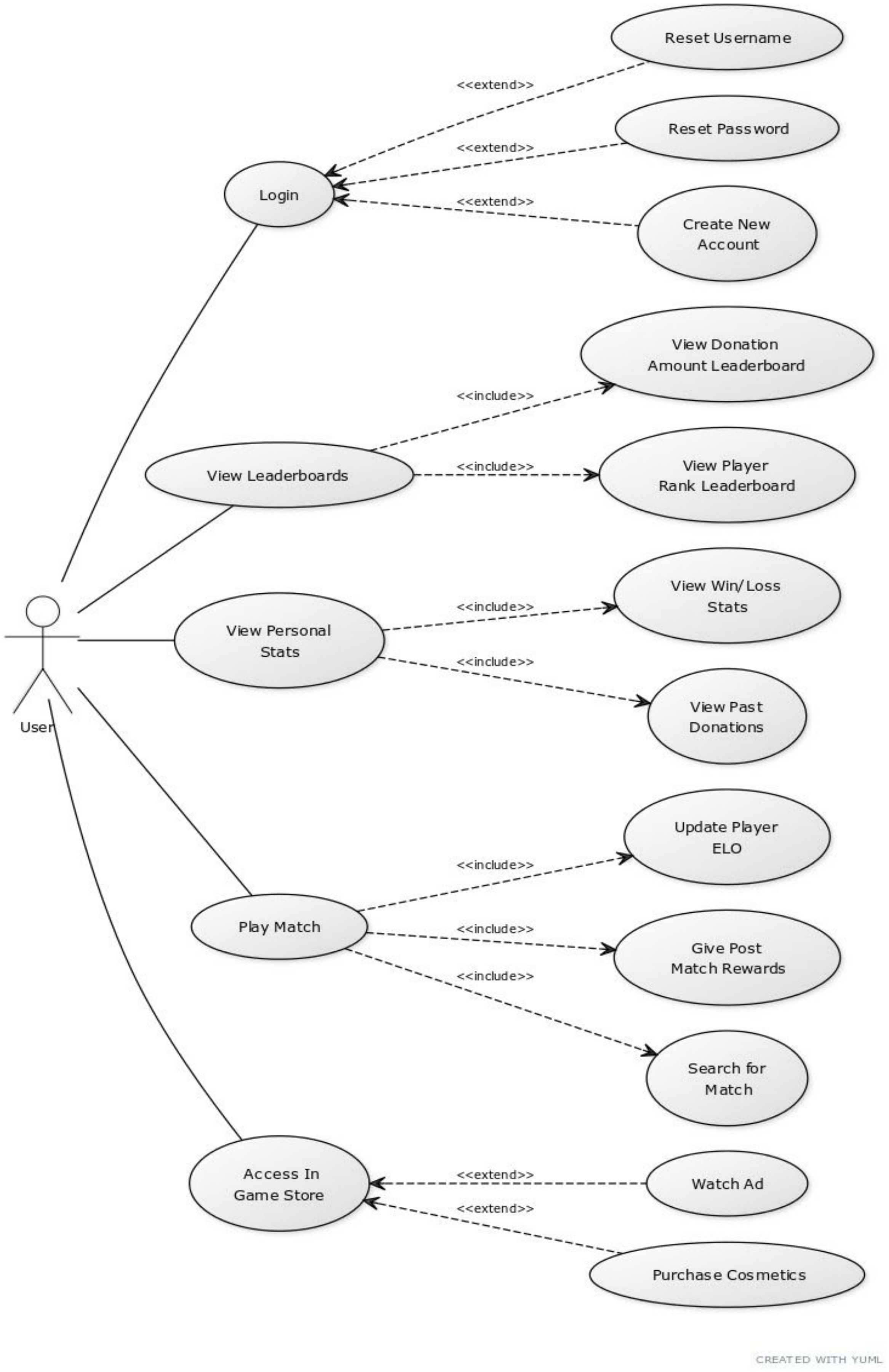
(Play Match) > (Update Player ELO)

[User] - (Access In Game Store)

(Access In Game Store) < (Purchase Cosmetics)

(Access In Game Store) < (Watch Ad)

Here is a jpeg:



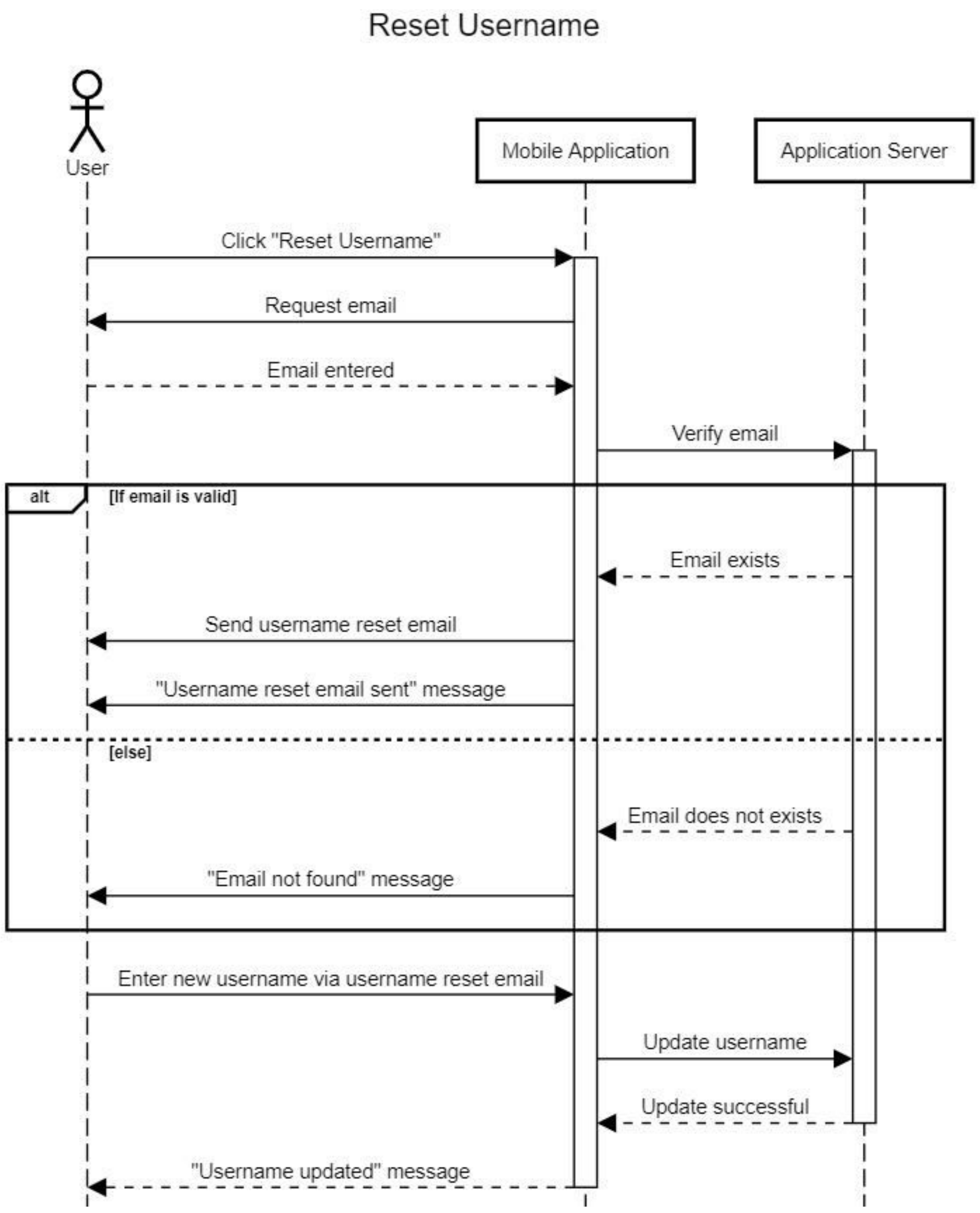
Reset Username
<<extend>>
Reset Password
<<extend>>
Login
<<extend>>
Create New Account
View Donation Amount Leaderboard
<<include>>
View Leaderboards
<<include>>
View Player Rank Leaderboard
View Win/Loss Stats
<<include>>
View Personal Stats
<<include>>
View Past Donations
User
Update Player ELO
<<include>>
Play Match
<<include>>
Give Post Match Rewards
<<include>>
Search for Match
Access In Game Store
<<extend>>
Watch Ad
<<extend>>
Purchase Cosmetics
CREATED WITH YUML

7. Sequence Diagram

Created using https://sequencediagram.org/

//

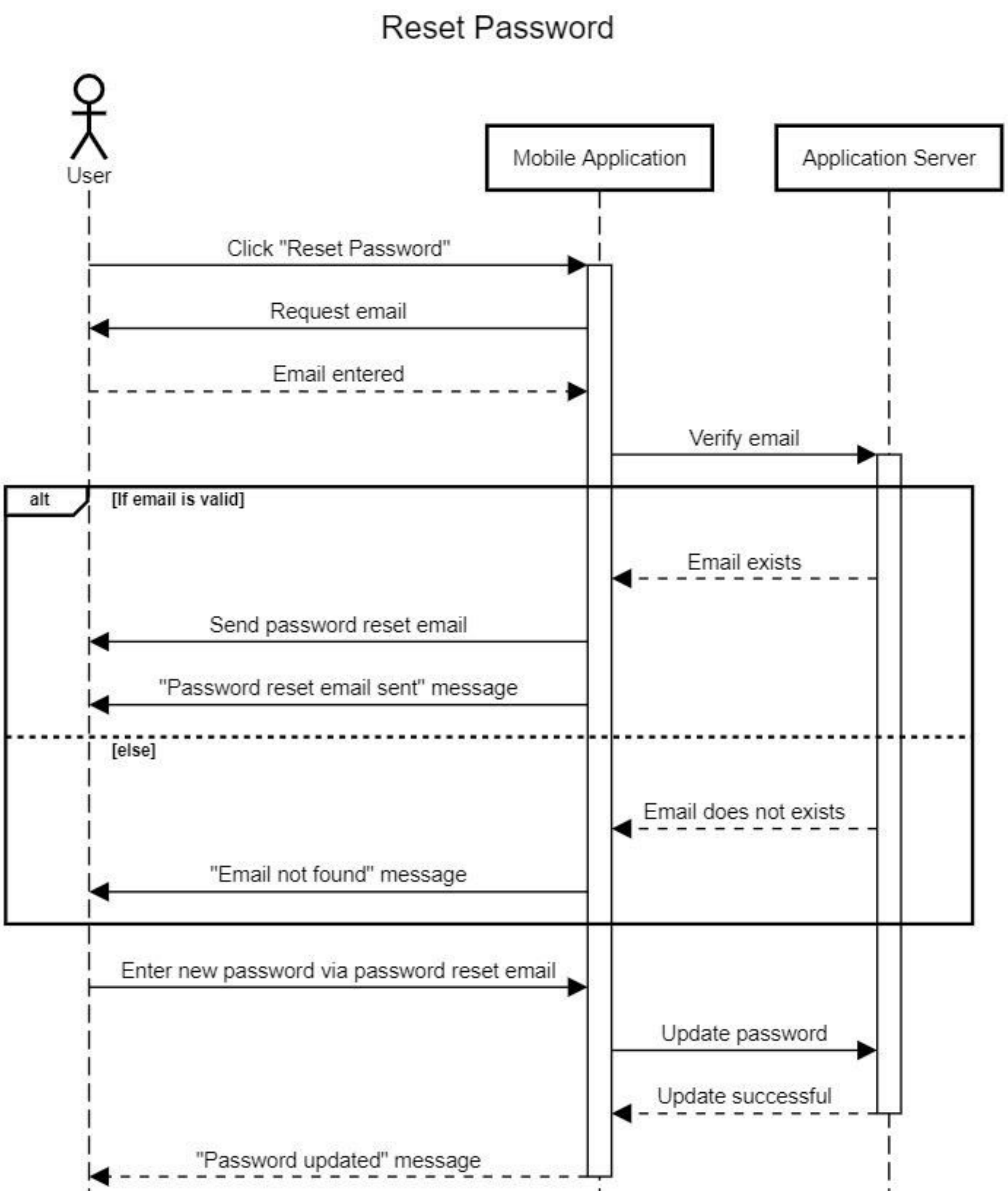
```
title Reset Username

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
```



//

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
```

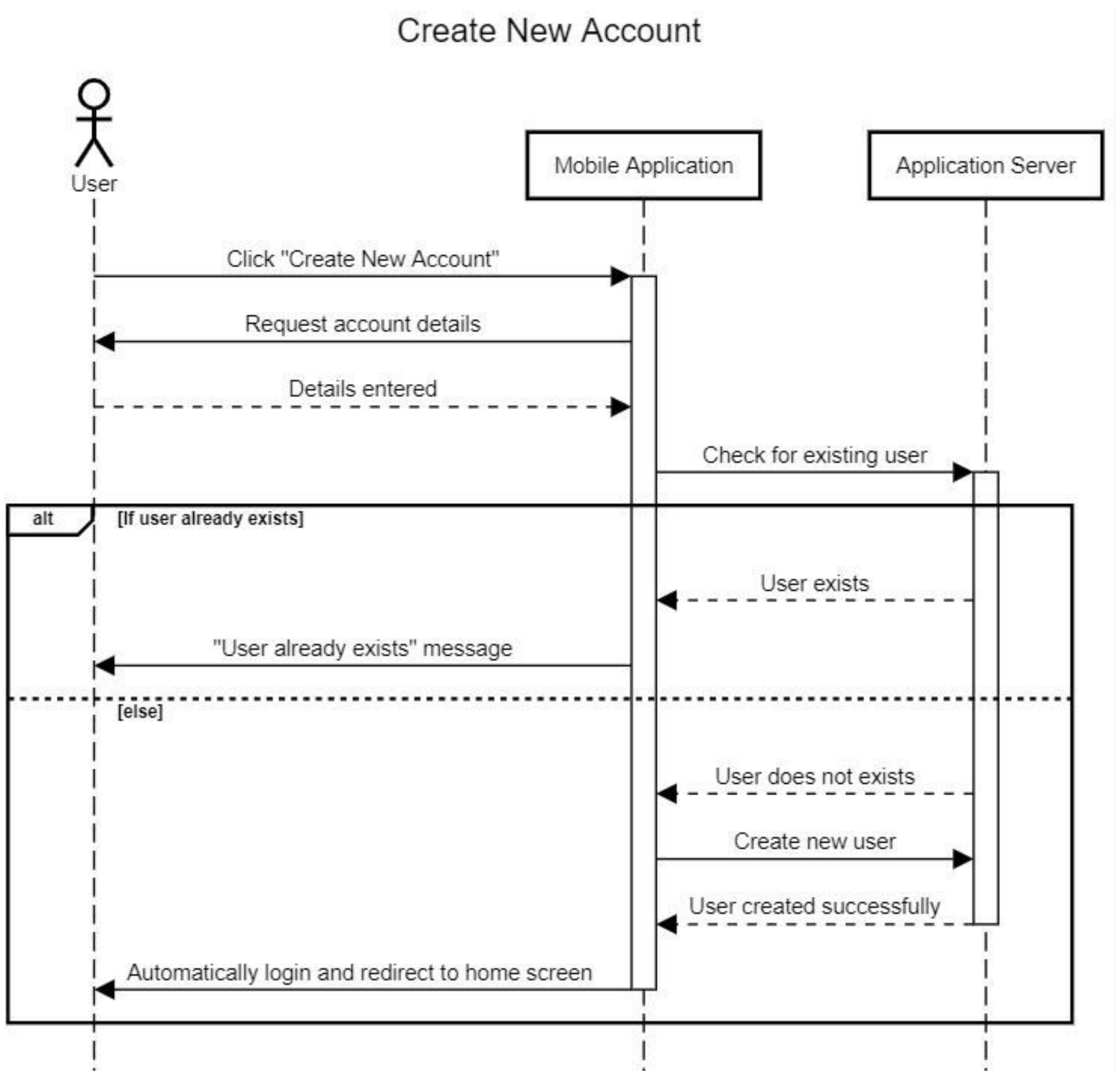


//

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
```



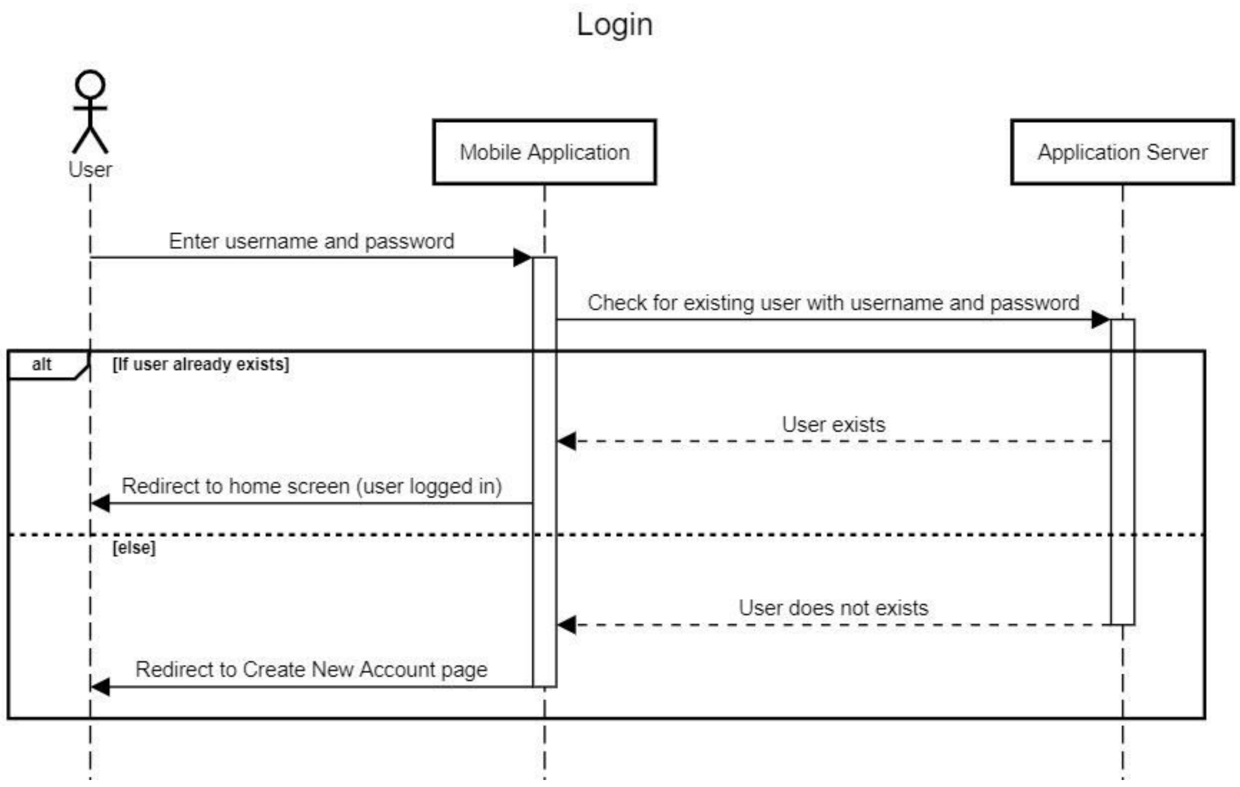
//

title Login

actor User
participant Mobile Application
participant Application Server

entryspacing 0.9
User -> Mobile Application: Enter username and password
activate Mobile Application

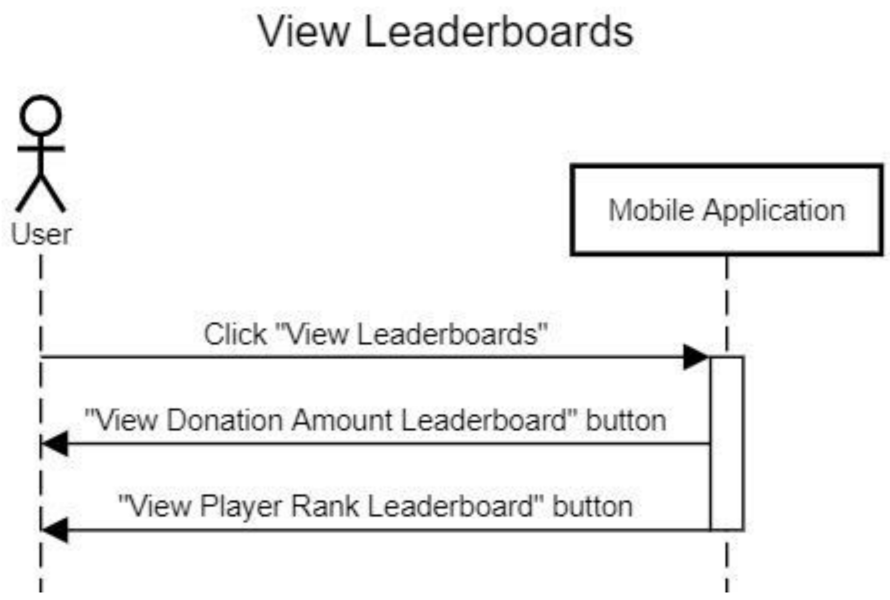
```
Mobile Application -> Application Server: Check for existing user with username and password
activate Application Server

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
```


Login
User
Mobile Application
Application Server
Enter username and password
Check for existing user with username and password
alt
[If user already exists]
User exists
Redirect to home screen (user logged in)
[else]
User does not exists
Redirect to Create New Account page

//

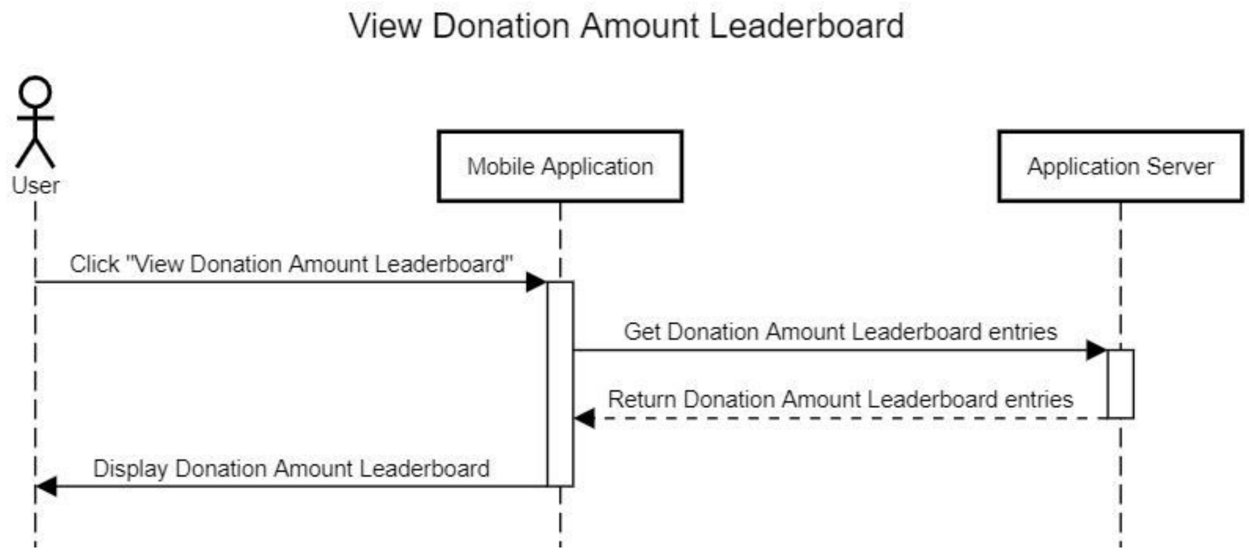
```
title View Leaderboards

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
User
Mobile Application
Click "View Leaderboards"
"View Donation Amount Leaderboard" button
"View Player Rank Leaderboard" button

```
/////////////////////////////////////////////////////////////////////////////////////

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
User
Mobile Application
Application Server
Click "View Donation Amount Leaderboard"
Get Donation Amount Leaderboard entries
Return Donation Amount Leaderboard entries
Display Donation Amount Leaderboard

//

title View Player Rank Leaderboard

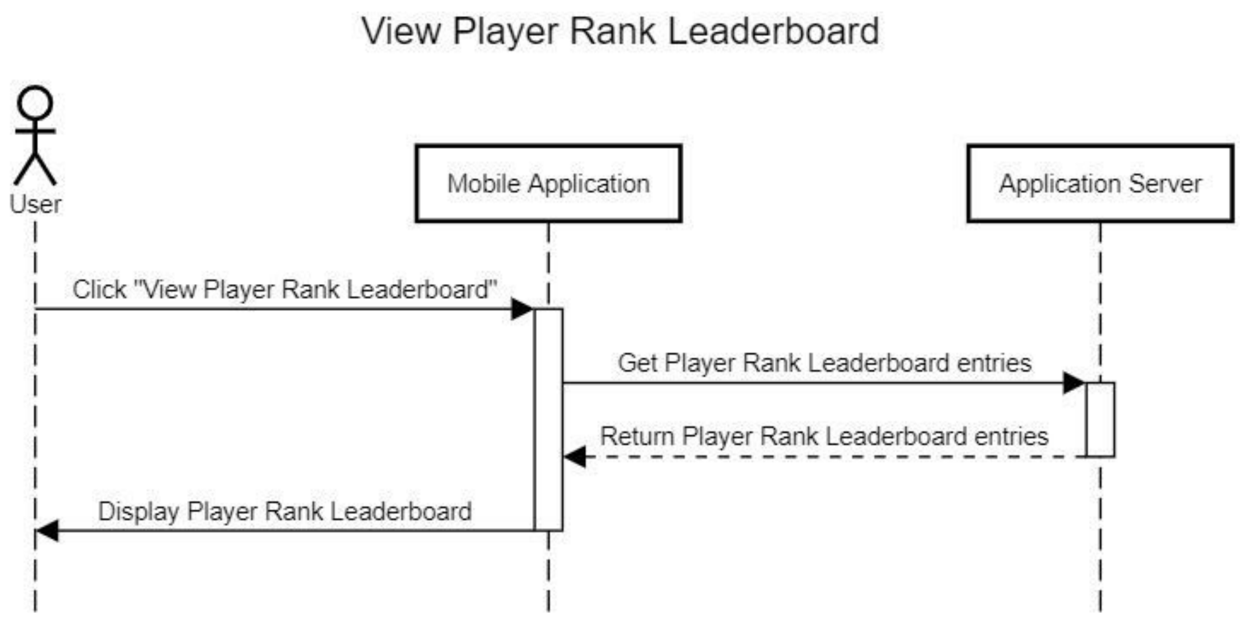
actor User
participant Mobile Application
participant Application Server

entryspacing 0.9
User -> Mobile Application: Click "View Player Rank Leaderboard"
activate Mobile Application

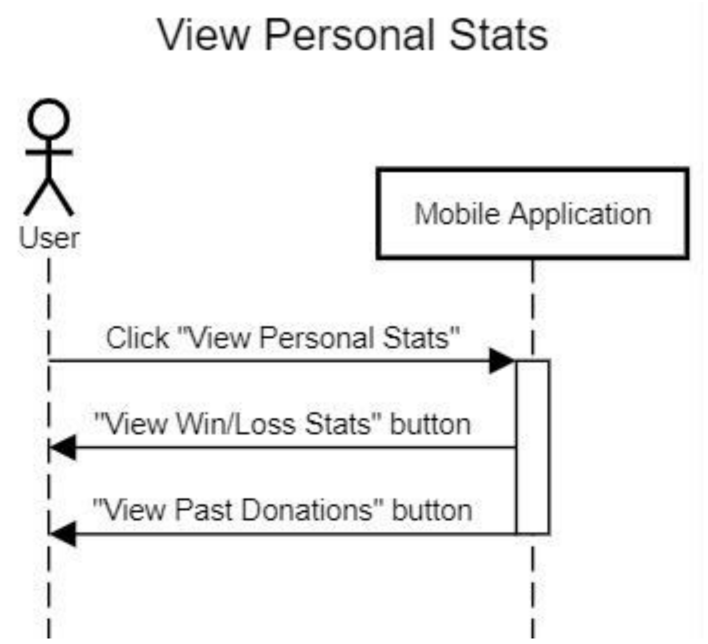
Mobile Application -> Application Server: Get Player Rank Leaderboard entries
activate Application Server

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

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

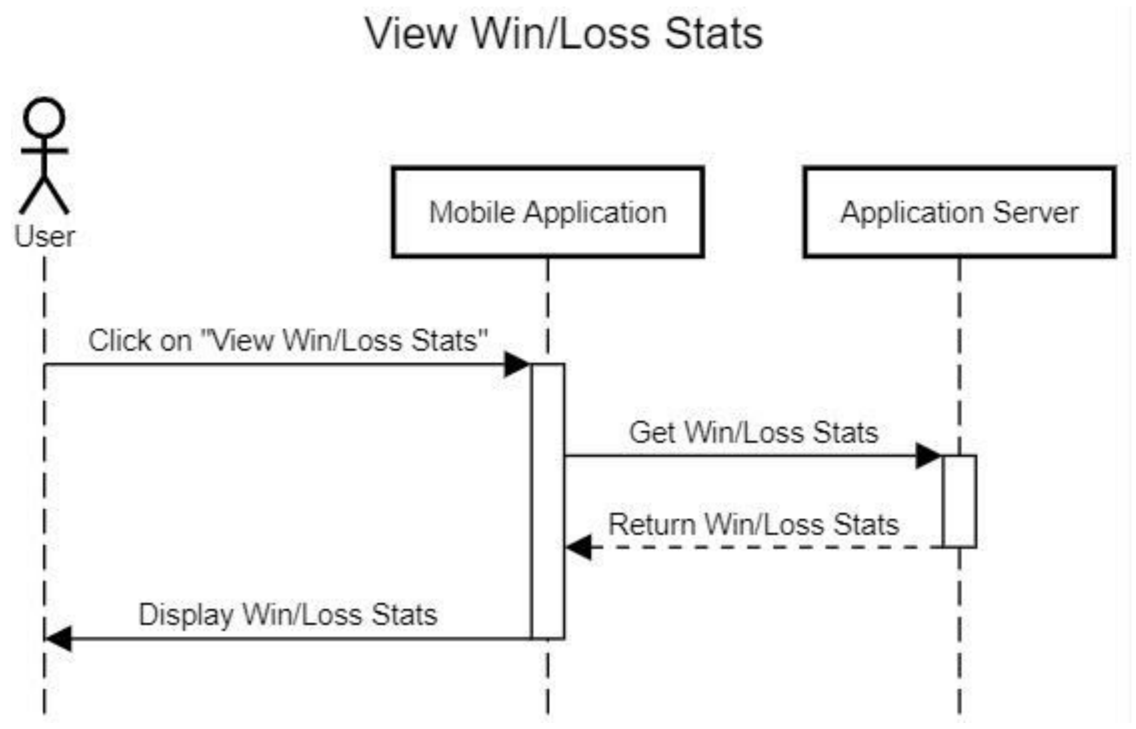

View Player Rank Leaderboard
User
Mobile Application
Application Server
Click "View Player Rank Leaderboard"
Get Player Rank Leaderboard entries
Return Player Rank Leaderboard entries
Display Player Rank Leaderboard

```
////////////////////////////////////////////////////////////////////////////////////////////

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
```

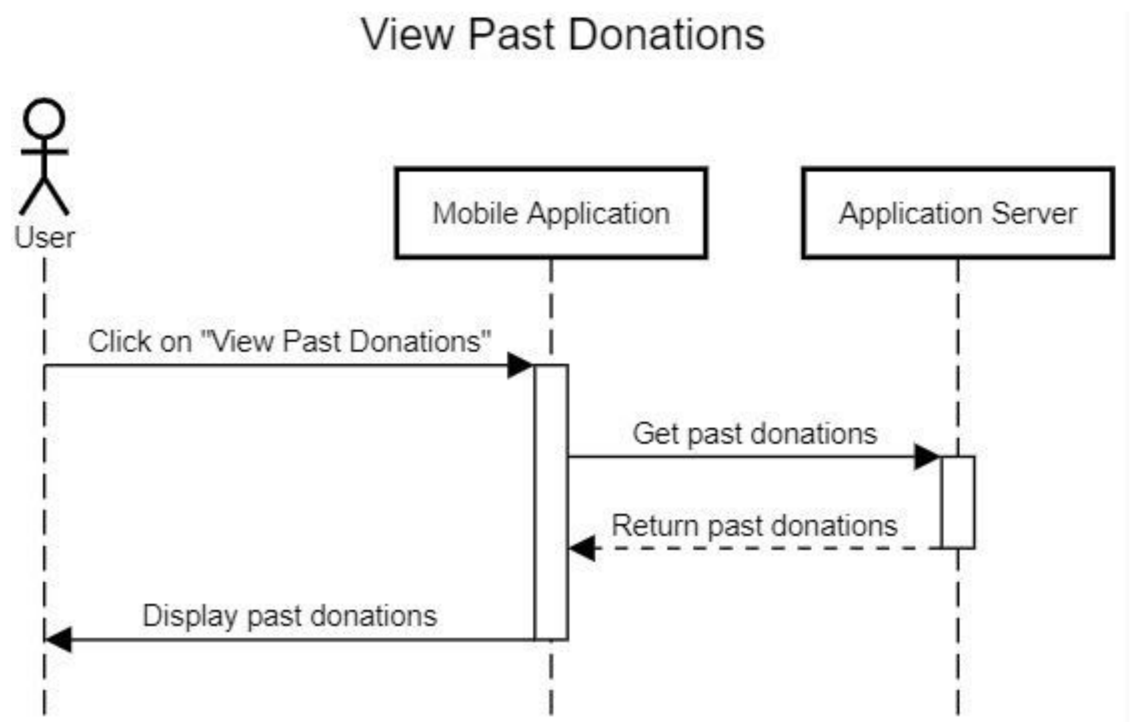

View Personal Stats
User
Mobile Application
Click "View Personal Stats"
"View Win/Loss Stats" button
"View Past Donations" button

///

```
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
```



```
///////////////////////////////////////////////////////////////////////////////////////////

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
```



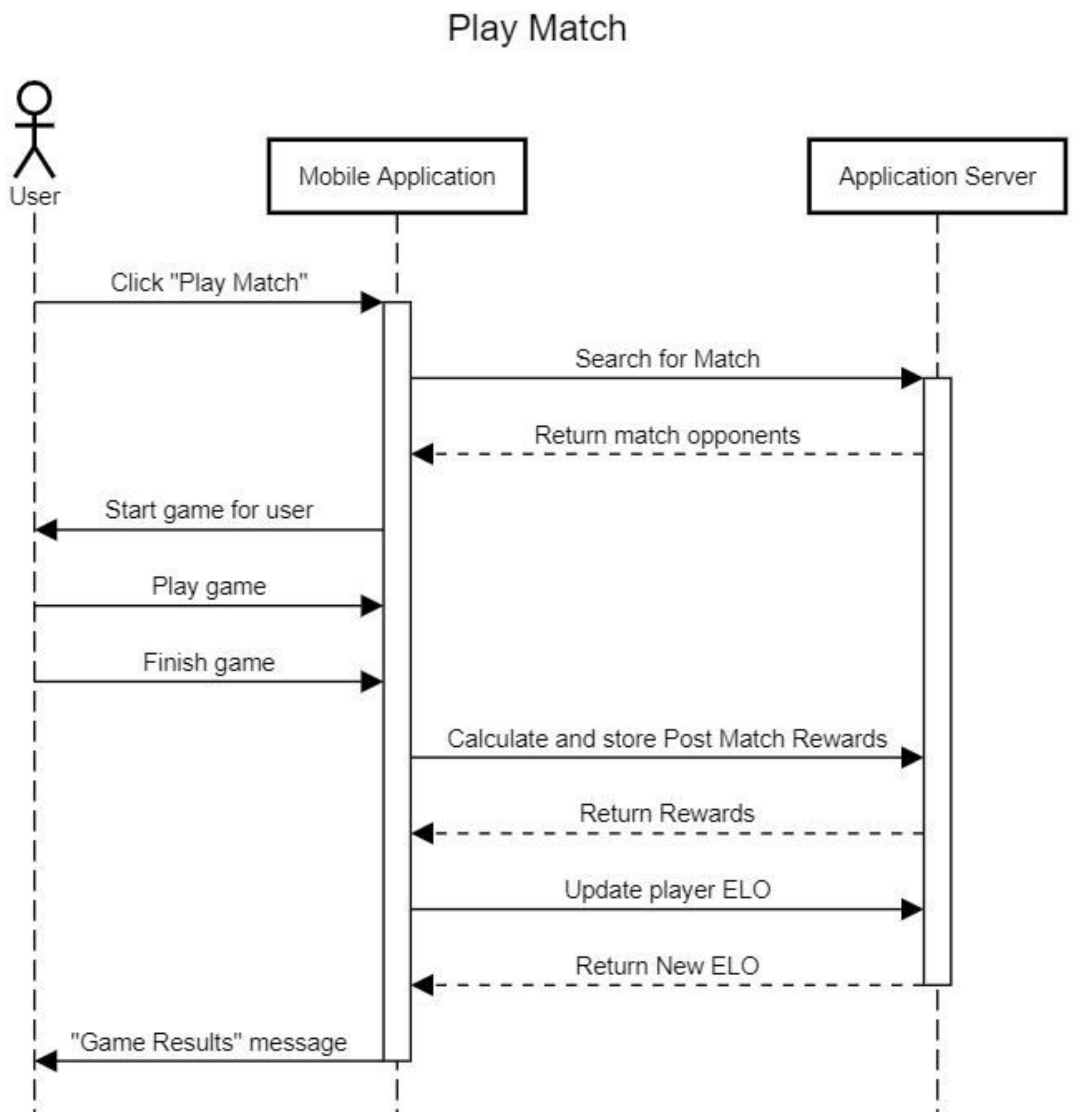
//

```
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
```

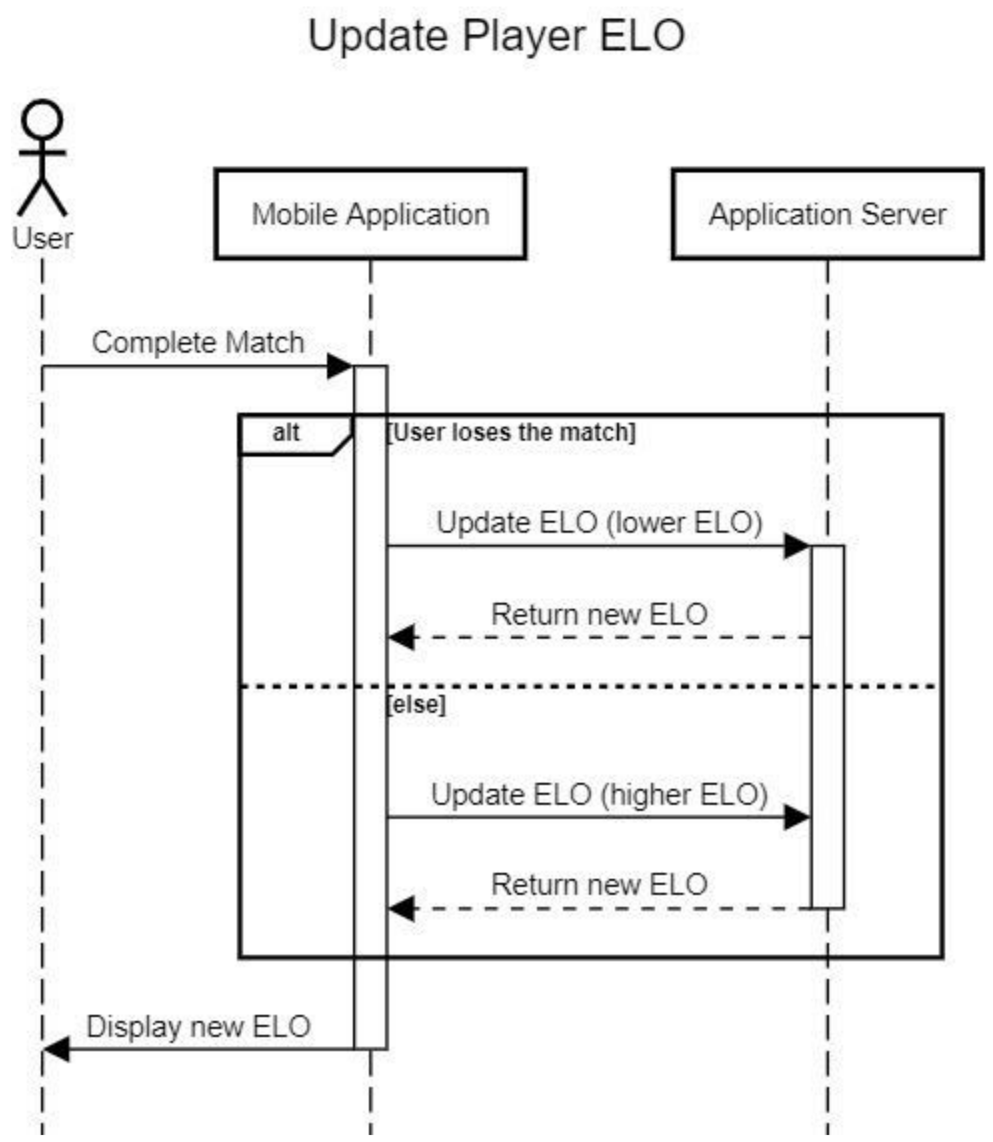
Mobile Application -> Application Server: Update player ELO

Application Server --> Mobile Application: Return New ELO
deactivate Application Server

Mobile Application -> User: "Game Results" message
deactivate Mobile Application



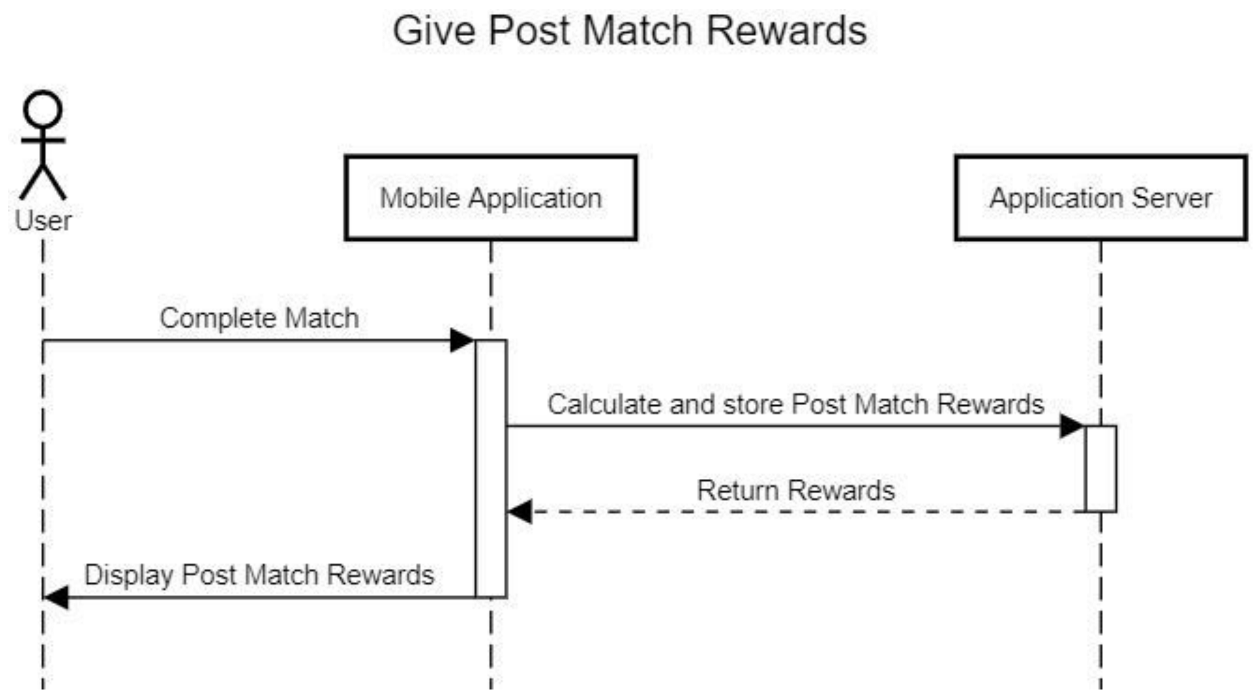
```
////////////////////////////////////////////////////////////////////////////////

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
```



```
////////////////////////////////////////////////////////////////////////////////

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



//

title Search for Match

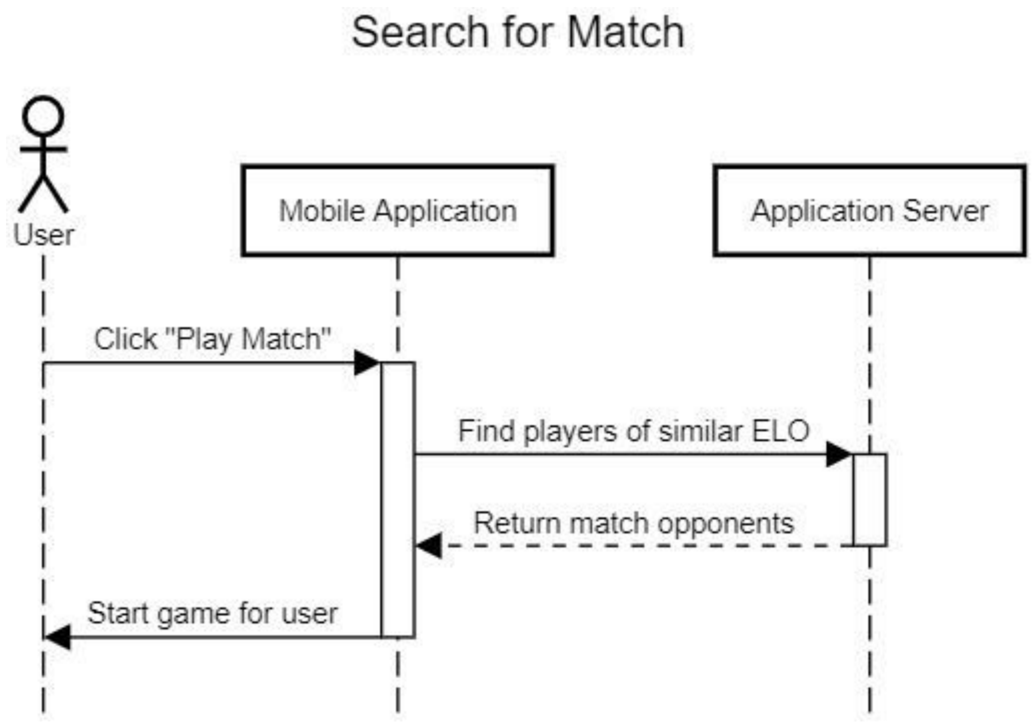
actor Actor
participant Mobile Application
participant Application Server

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



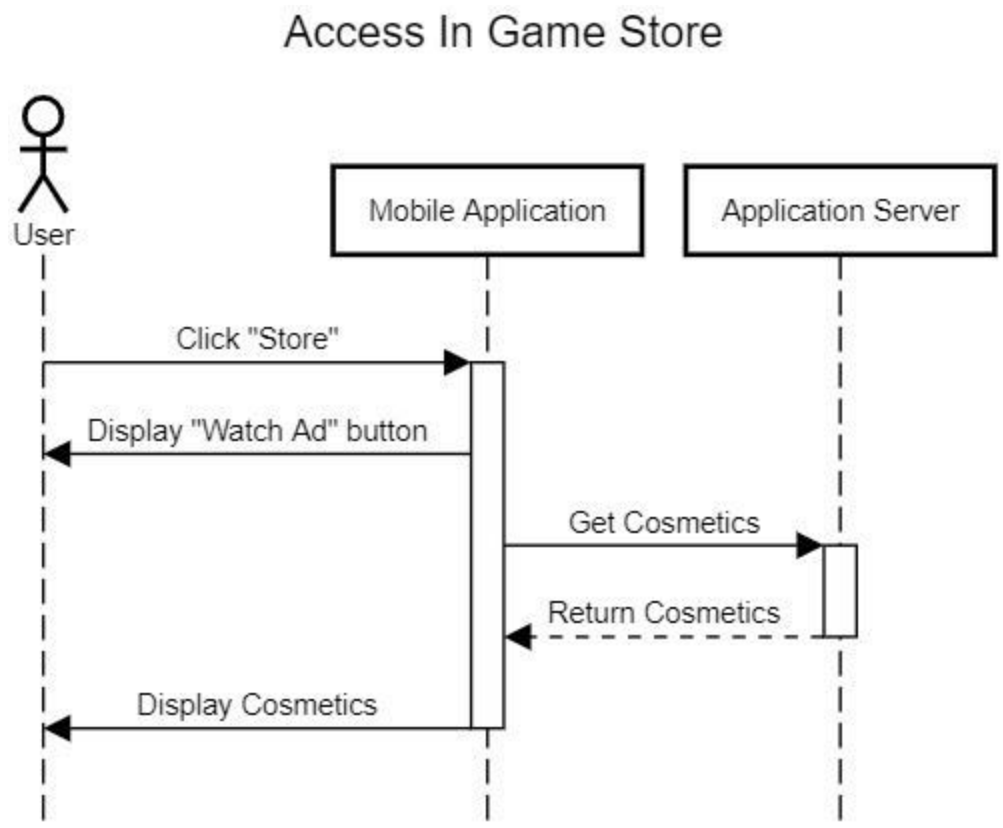
//

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



//

title Watch Ad

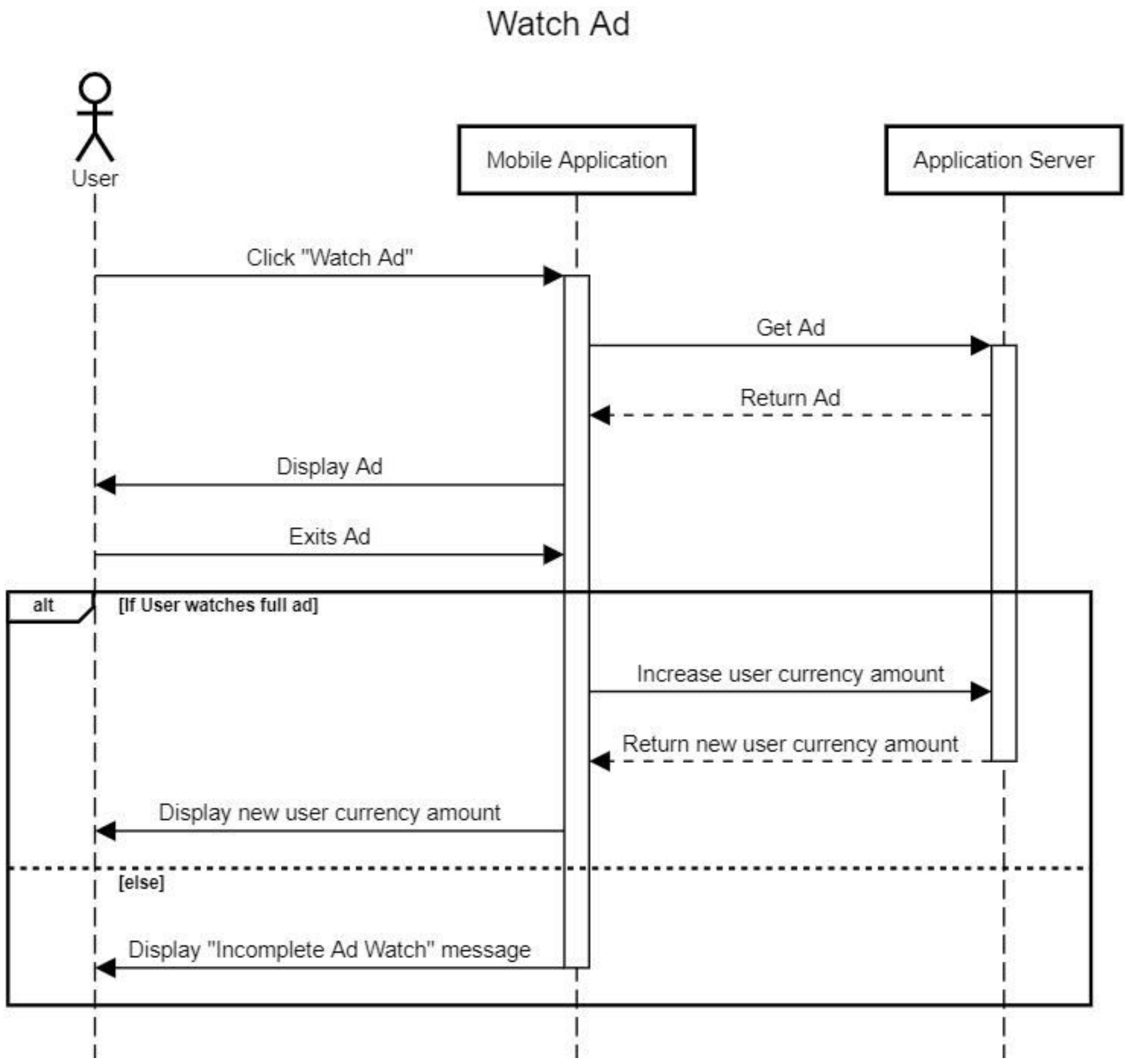
actor User
participant Mobile Application
participant Application Server

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

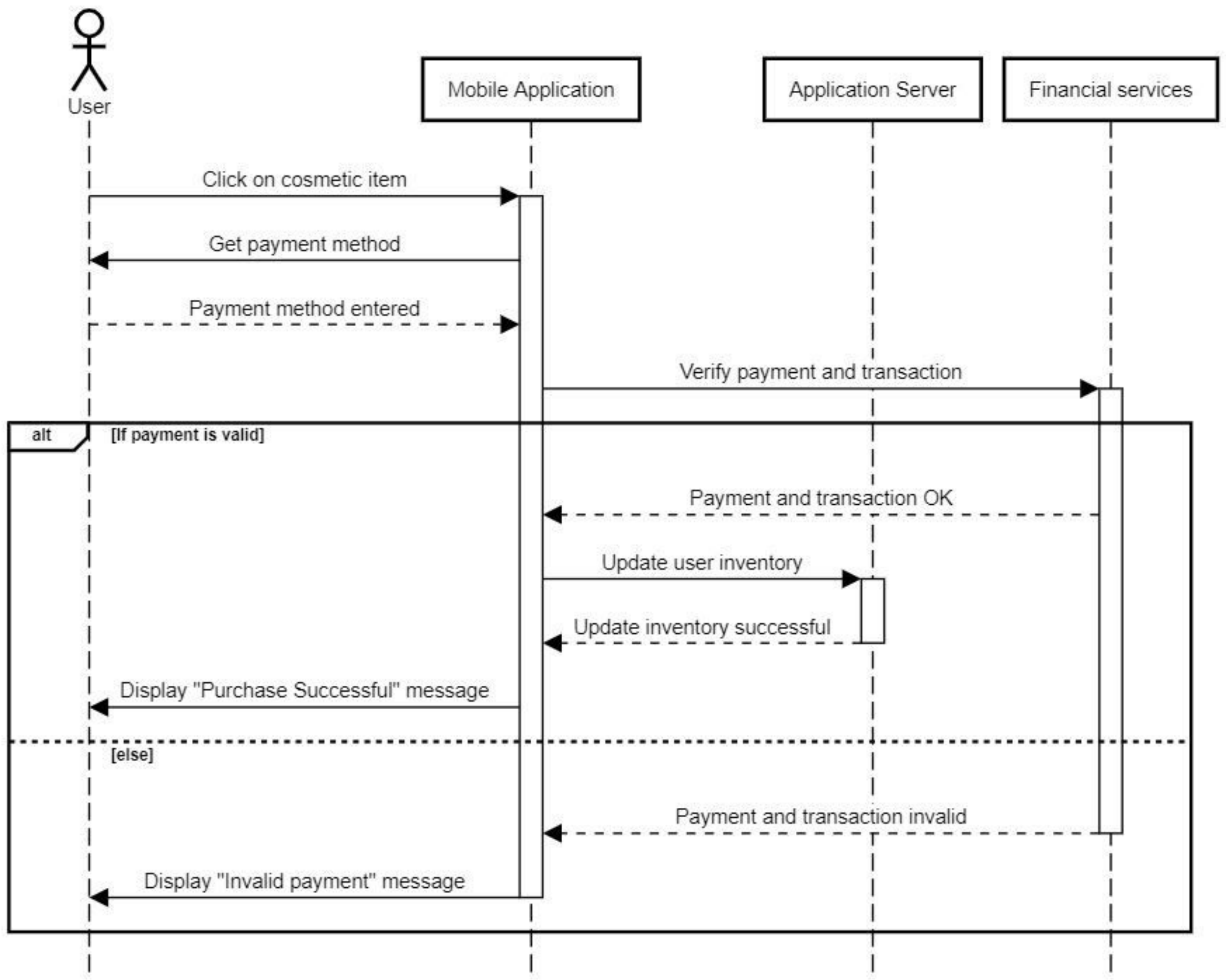
alt If User watches full ad
Mobile Application -> Application Server: Increase user currency amount

```
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
```


Watch Ad
User
Mobile Application
Application Server
Click "Watch Ad"
Get Ad
Return Ad
Display Ad
Exits Ad
alt
[If User watches full ad]
Increase user currency amount
Return new user currency amount
Display new user currency amount
[else]
Display "Incomplete Ad Watch" message

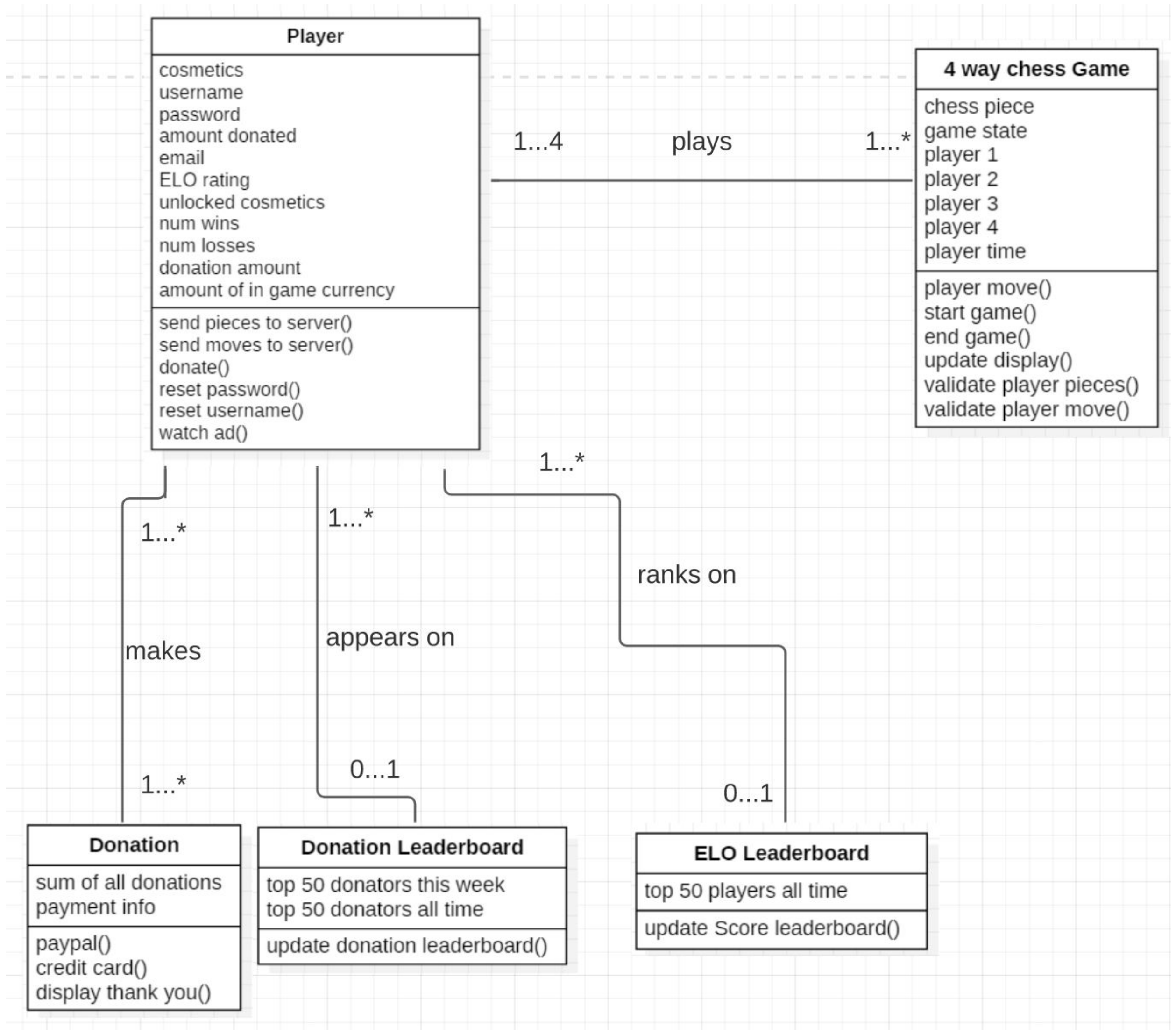
```
/////////////////////////////////////////////////////////////////////////////////////

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


User
Mobile Application
Application Server
Financial services
Click on cosmetic item
Get payment method
Payment method entered
Verify payment and transaction
alt
[If payment is valid]
Payment and transaction OK
Update user inventory
Update inventory successful
Display "Purchase Successful" message
[else]
Payment and transaction invalid
Display "Invalid payment" message

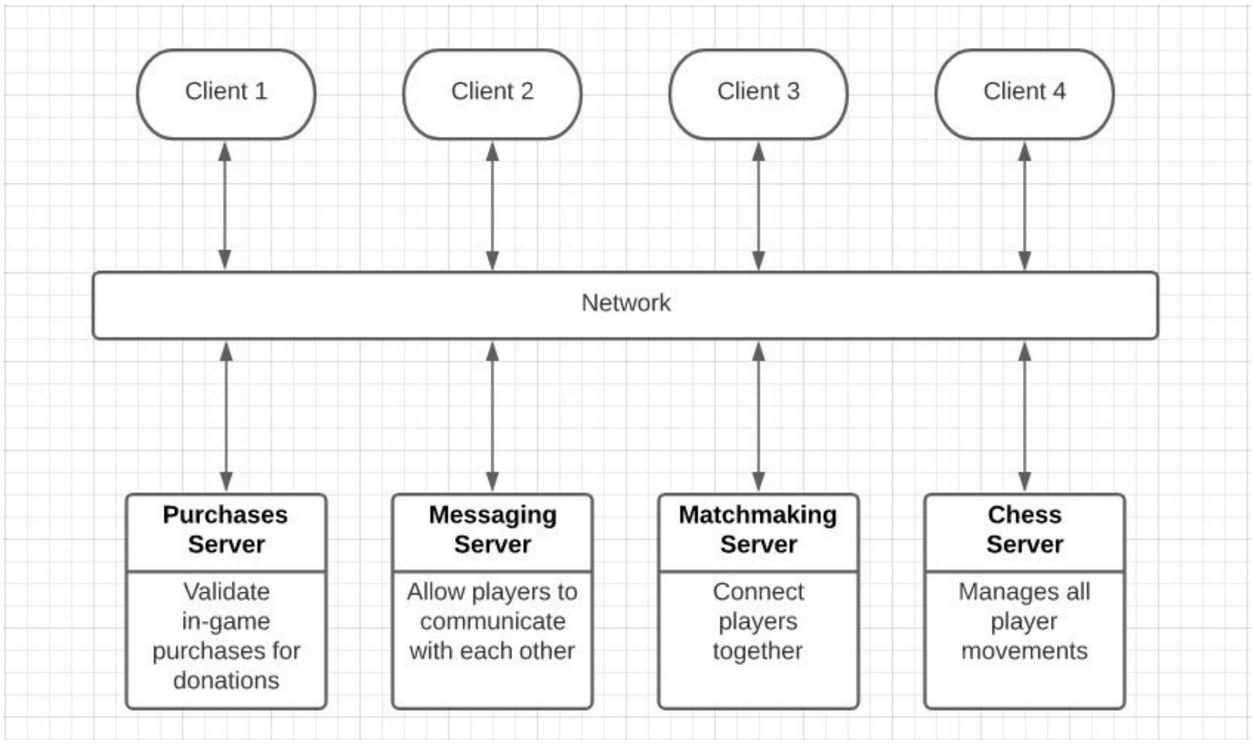
//

8. Class Diagram



9. Architectural Design:

Client-server architecture pattern:


Client 1
Client 2
Client 3
Client 4
Network
Purchases Server
Validate in-game purchases for donations
Messaging Server
Allow players to communicate with each other
Matchmaking Server
Connect players together
Chess Server
Manages all player movements