SOFTWARE DESIGN DOCUMENT

04.01.2015

Master King PROJECT

BEDRETTİN ÇETİNKAYA -1819788
UĞUR ÇOĞALAN -1949825
OĞUZHAN YÜZER – 1819713
OĞUZ ÇAKMAK - 1819739

Table of Contents

1. Overview

This document contains the software design descriptions for Master King Project. It is prepared according to the “IEEE Standard for Information Technology – Systems Design – Software Design Descriptions – IEEE 1016 – 2009”.

This document provides the details of how the system should be built. The details are represented by using graphical notations such as some viewpoints (Context Logical, Dependency, Information , Pattern use, Interface, Interaction, and state dynamic viewpoint), use case models, sequence diagrams, class diagrams, object behavior models and other supporting design information.

1.1 Scope

This document describes the detailed structure of the components of the Master King Project and the implementation details needed to satisfy the requirements as specified in the Software Requirements Specification. A set of design views is presented in order to support the design and development process. This document will serve as a guideline through the implementation phase.

1.2 Purpose

The purpose of this document is to describe and visualize the design and architecture of Master King Project by using different viewpoints.

This document aims to describe the functional structure, data and algorithms to be implemented. Also, it will be the primary reference for the system resources and maintenance activities.

1.3 Intended Audience

The intended audience for this document is the design and development team of the Master King Project, supervisors and knowledgeable software designers. Also, users can use this document to understand this project.

2. Definitions

Term	**Definition or Abbreviation**
User	User The person who is registered in system
Guest	Guest The person who is not registered in system but can play the game
SRS	Software Requirements Specification
UML	Unified Modeling Language
IEEE	Institute of Electrics and Electronics Engineers
DBMS	Database Management System
ER Diagram	Entity – Relationship Diagram
JDBC	Java Database Connectivity
JDK	Java Development Kit

MySQL	An open source Database Management System

Table 1: Definitions Table

3. Conceptual Model for Software Design Descriptions

In this section, a conceptual model for the SDD is presented. This conceptual model mainly explains the context in which SDD is prepared and how it will be used.

3.1 Software Design in Context

System has three subsystems which are Server Engine, Client and Database. Also system has two types of actors which are Guest and User. If actor is guest, there is no need for database component because he/she is not registered in the system and only plays the game. On the other hand, user is registered in the system and some data about user like nickname, password, number of wins and points should be stored in database. Therefore, database is required if the actor is user.

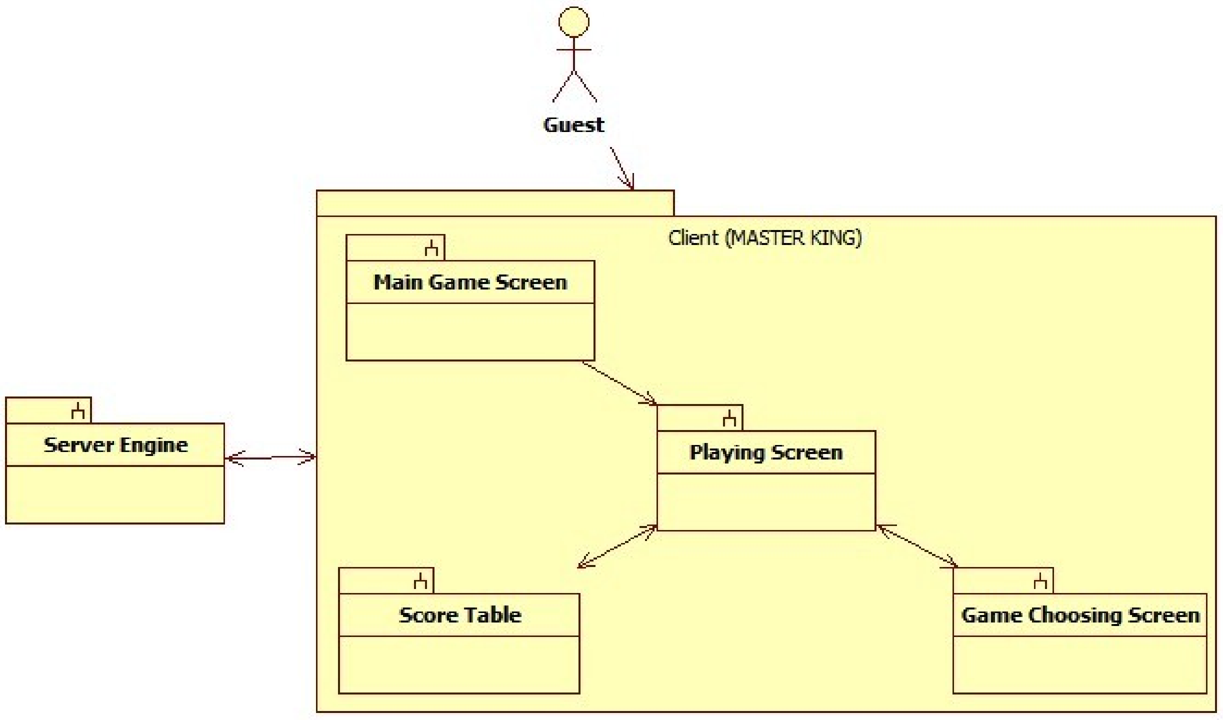


Figure 1: Guest Context Diagram

The system is an Android product .Therefore, Main Game Screen, Playing Screen, Score Table and Game Choosing Screen are different activities. This context diagram shows the interactions between guest and system. Guest enters the system and touches play as a guest button in Main Page Screen.

Because of the this choice, there is no need for database. Then, he/she is directed to Playing Screen, and game starts. During the game, guest can switch to Score Table or Game Choosing Screen by clicking the corresponding buttons.

After each kind of game, Score Table is updated. Also, there are interactions between Server Engine and all system components. Algorithms of agents are in the server engine and according to these algorithms, server engine returns data to system.

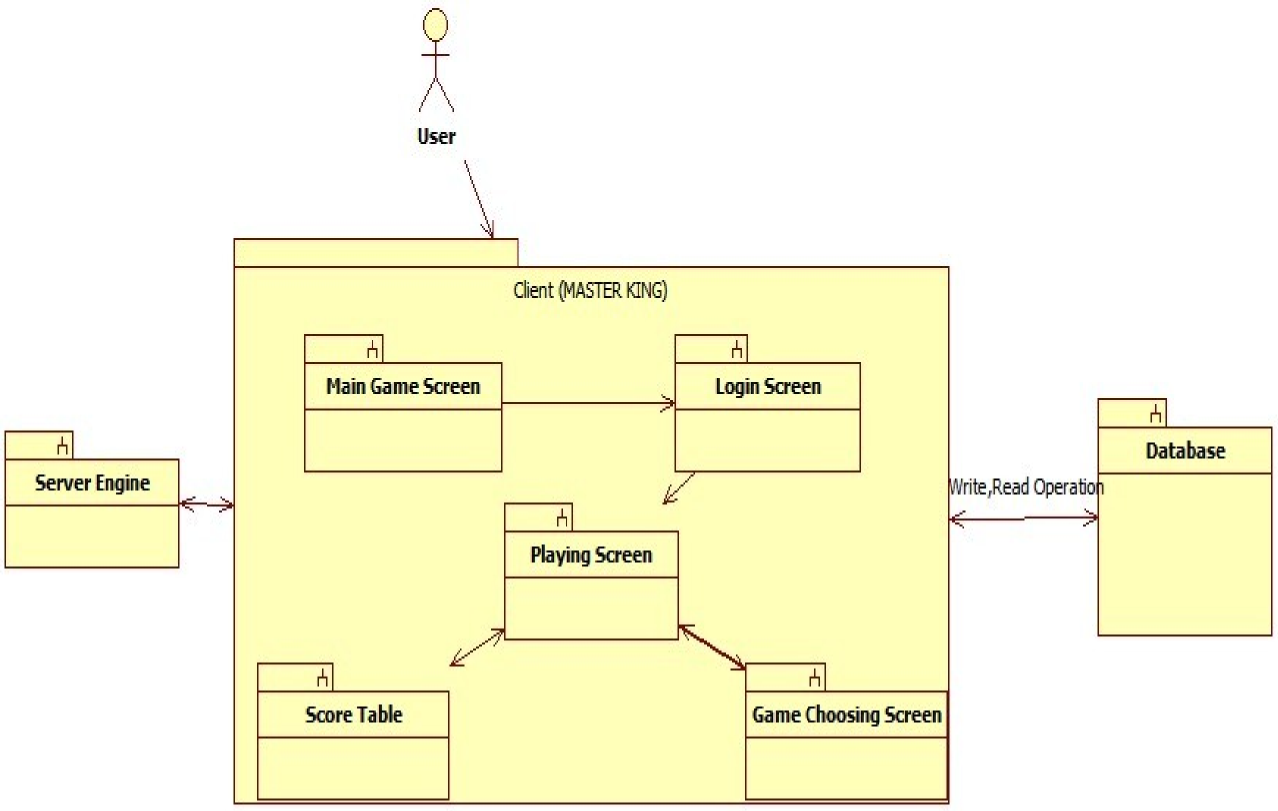


Figure 2: User Context Diagram

This context diagram shows the interactions between user and system. User enters the system, then he/she is directed to login screen. After a successful login operation, Playing Screen is shown up and the game starts. During the game, guest can switch to Score Table or Game Choosing Screen by clicking the corresponding buttons.

After each kind of game, Score Table is updated. When King game finishes, score of the user is sent to database. Also, there are interactions between Server Engine and all system components. Algorithms of agents are in the server engine and according to these algorithms, server engine returns data to system.

3.2 Software Design Descriptions within the Life Cycle

3.2.1 Influences on SDD Preparation

This document is prepared by considering the opinions of the King players and supervisors. SRS document is an important reference to this document.

3.2.2 Influences on Software Life Cycle Products

This project is a client-server application based on the Android. Therefore, all of the game algorithms will be in server side and all of the game steps will be evaluated on the server, and this evaluation will be reflected to the client side. Afterwards, corresponding card will be drawn on the Android screen. Network connection, server module and Android module play important role for this process. All of them should be synchronized.

3.2.3 Design Verification and Design Role in Validation

Since Master King is a server based single player card game, the server is responsible for running the agents according to users' moves. To get the same result in each execution we need to have reliable and stable server and network. After finishing the implementation of the entire system, entire system will be tested in detail to verify and validate.

4. Design Description Information Content

4.1 Introduction

The purpose of this section is to provide information about how the design description will be explained in the later sections of the document. In this section, we will give information about SDD identification, identified design stakeholders, identified design concerns, selected design viewpoints, design languages, design view, design overlays and design rationale.

4.2 SDD Identification

This is the initial SDD for the project Master King by team REBELLION. The date of release for this document is 04.01.2015. This is not the final design document for the project and is subject to change. The issuing organization for this document is team REBELLION whose members are Uğur Çoğalan, Bedrettin Çetinkaya, Oğuz Çakmak, Oğuzhan Yüzer. The supervisors of this project are Dr. Selim Temizer and Asst. Burak Velioğlu. The purpose of this document is to give design information about the project Master King. In this report, UML is main modeling language for explaining the design viewpoints. The intellectual property rights of the project belong to team. For the definitions and glossary refer to section 2.

4.3 Design Stakeholders and Their Concerns

The design stakeholders of this document are Dr. Selim Temizer, Asst. Burak Velioğlu and instructors of CENG491 course. Dr. Selim Temizer is the main supervisor of the team. Asst. Burak Velioğlu is one of the teaching assistants of the CENG491 course and he is responsible for managing the team. The team and assistant arrange weekly meetings about the project so that the team can get feedback from him.

4.4 Design Views

In the SRS Document, We already stated product context, system boundaries and other limitations. Also, detailed explations of design views which are used in system are in the section 4.5 and 5. One of the most important design concern is to be secure and reliable interactions between servers and other components. During a server operation, unrelated processes shouldn't interrupt this operation. The other design concern is flexibility. If new rules are added to the system, this shouldn't affect other unchanged components. Also, Context, Logical, Dependency, Information, Patern Use, Interface, interaction and State Dynamic Views are used while implementing the system. All of these views have different viewpoints and explation of these viewpoints are supported with diagrams.

4.5 Design Viewpoints

- Context Viewpoint explains what is expected from the user in the system and all roles of the user are clearly shown.
- Logical Viewpoint explains the logical class structure of the both server and android application and clearly shows functionalities of the classes.
- Dependency Viewpoint explains all of the component dependencies to each other.
- Information Viewpoint explains interaction between database and other system component.
- Pattern Use Viewpoint explains architectural pattern that are used in system.
- Interface Viewpoint explains system interfaces.
- Interaction Viewpoint explains all interaction beetwen system components.
- State Dynamic Viewpoint explains system's functionalites.

Also, detailed explanation of these Design Viewpoints are in section 5.

4.6 Design elements

4.6.1 Design Entities

- Player
- User
- Guest
- Points
- Game Type
- Client(Android)
- Server
- Database
- Network

4.6.2 Design Attributes

Name	**Description**
Player	Actor of the system
User	Registered actor
Guest	Unregistered actor
Points	Total score of user
Game Type	Positive or negative hands in King Game
Client(Android)	Transmits player's operation to server
Server	Process all algorithms
Database	Hold user's data
Network	Provides communication between server and client

Table 2: Design Attributes Table

4.6.3 Design Relationship

Relationship of all design attributes are illustrated in Section 5 Design Viewpoints.

4.6.4 Design Constrain

In the SRS document, design constrains are already stated.

4.7 Design overlays

There are no specific overlays with Design Viewpoints.

4.8 Design Rationale

Design choices are specified according to the some criterias such as maintainability, sustainability, relaibility, security and robustness. They could be updated according to the system requirements. Also, during the implementation process, all codes are commented. This makes the source code more understandable and easily changeable.

4.9 Design languages

Unified Modeling Language is used when design points are prepared.

5. Design Viewpoints

5.1 Introduction

In this section, 9 main viewpoints of the project will be explained:

- Context Viewpoint
- Composition Viewpoint
- Logical Viewpoint
- Dependency Viewpoint
- Information Viewpoint
- Pattern Use Viewpoint
- Interface Viewpoint
- Interaction Viewpoint
- State Dynamics Viewpoint

UML diagrams are used to visualize the system in order to increase understandability during the explanation of these viewpoints.

5.2 Context Viewpoint

This context viewpoint shows the functionalities provided by design. There are 3 main screens controlled by the user which are register screen, login screen and choose game screen. When the connection between client and server is declared, functionalities of the first two screens become active. Third screen will be available when the user starts playing.

5.2.1 Register Screen

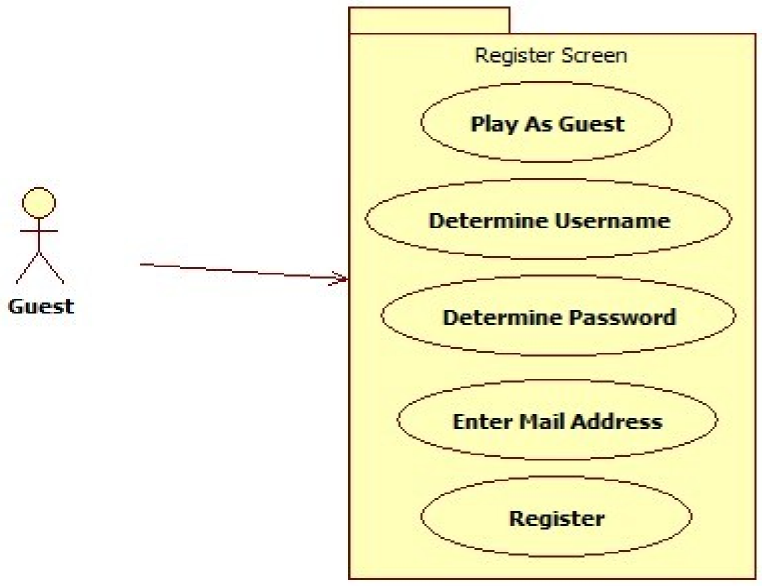


Figure 3: Use Case Diagram for Guest

Register screen can be used by unregistered users or by users who do not want to be registered in the system. Its functionalities are the following:

Play As Guest: If the person does not want to register but still wants to play the game, he/she can click this button and start playing directly.

Determine Username: If the person wants to register, he/she can create his/her user name in the specified area.

Determine Password: The unregistered user can determine his/her password in the specified area for the registration.

Enter Mail Address: The unregistered user can enter his/her e-mail address in the specified area for the registration.

Register: After all necessary fields are filled, user can click this button to register. The information will be saved on the database and an informing message about registration result will be sent back to the client.

5.2.2 Login Screen

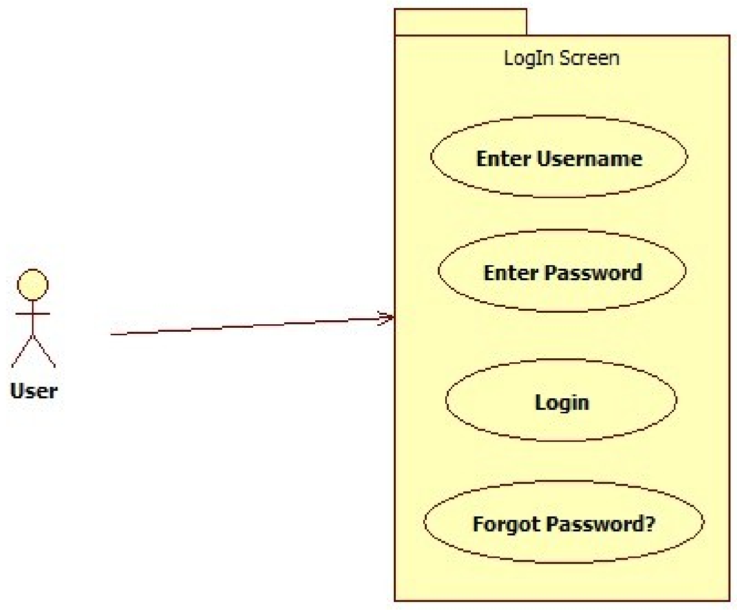


Figure 4: Use Case Diagram for User

Login Screen is for registered users who wants to login to the system. Its functionalities are the following:

Enter Username: The registered user can enter his/her user name in the specified area.

Enter Password: The registered user can enter his/her password in the specified area.

Login: After all necessary fields are filled, user can click this button to login. The information entered will be checked and an informing message about login result will be sent back to the client.

Forgot Password?: If the registered user does not remember his/her password, he/she can click this button. A temporary new password will be sent to the users e-mail address.

5.2.3 Choose Game Screen

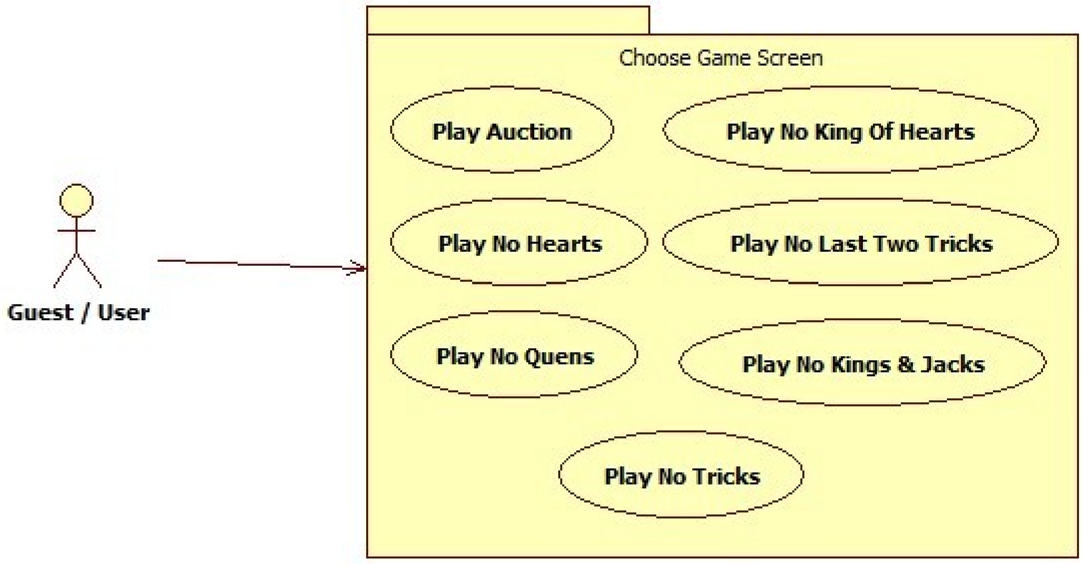


Figure 5: Use Case Diagram for Guest/User

Choose Game Screen is for all users who connected to the system. It becomes available when the users start playing. Its functionalities are the following:

When it is users turn to choose a game, he/she can click the button corresponding to the game he/she wants to play. After user clicks to a button, server engine will play the cards according to the chosen game and return the cards to the client.

5.3 Composition Viewpoint

Composition Viewpoint describes the system's submodules and their roles.

User connects the system through an Android device. After this point, the data flows between server and client during the game. Before user logs out, his/her score is updated on the database and user leaves.

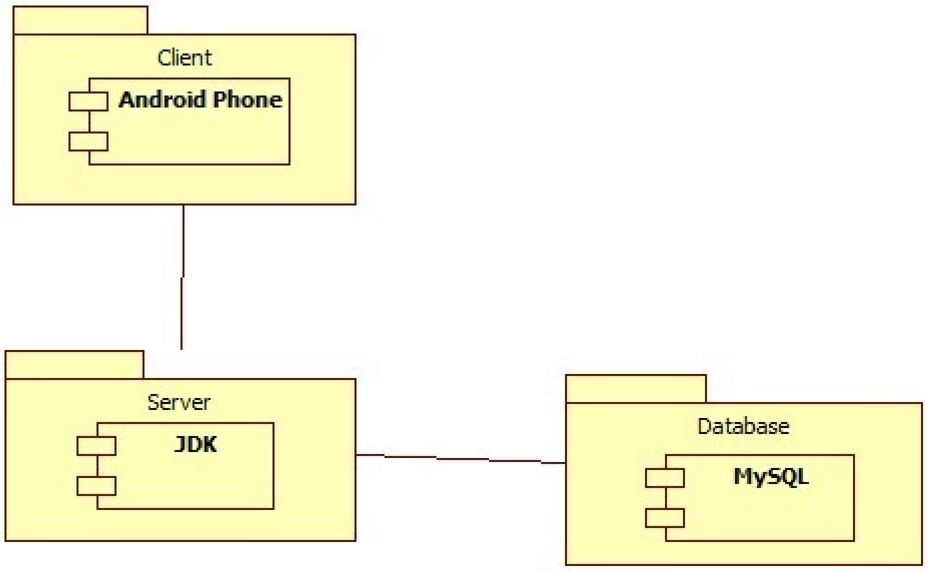


Figure 6: Component Diagram for Guest/User

5.4 Logical Viewpoint

This section contains the predefined classes, objects and the relationships between these entities. Logical Viewpoint is one of the most important viewpoints, because almost all design and implementation of the design is based on this logical concept. In the diagram below, all of the system entities and relationships between them is shown.

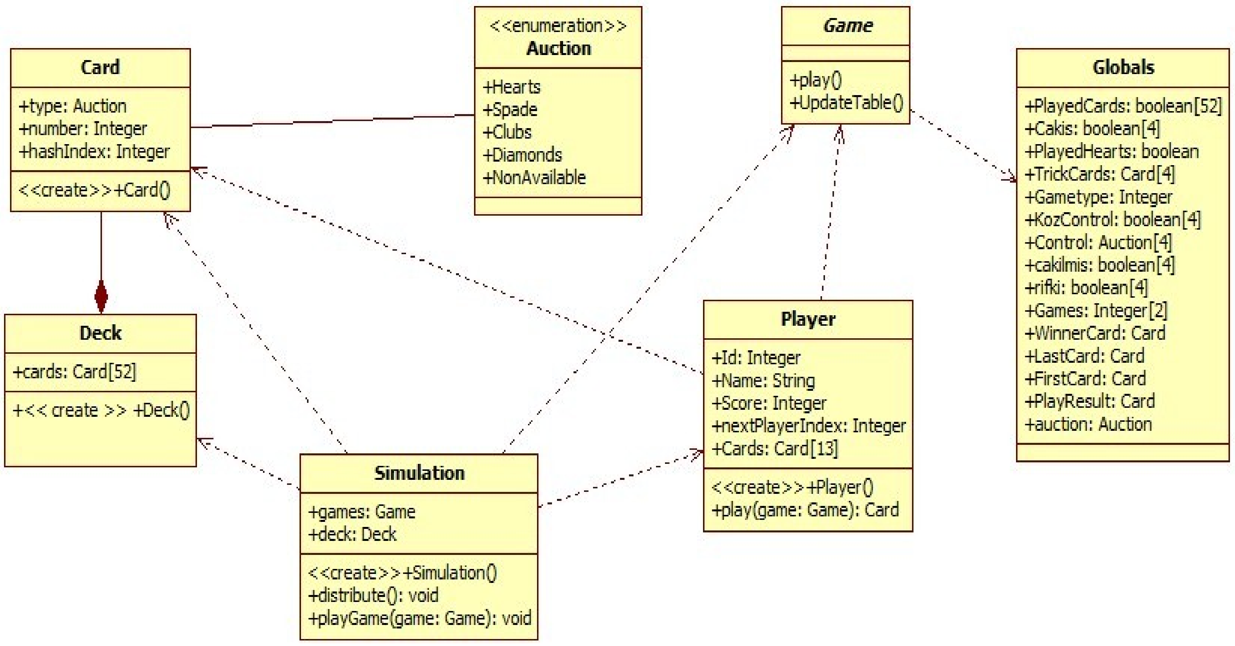


Figure 7: General Class Diagram of the System

5.4.1 Design Elements

<<Enumeration >> Auction: This enumeration is used to represent and separate the cards. Like in the real world, cards have 4 different types which are Hearts, Spade, Clubs and Diamonds. Also in the system, there is fifth type which is Non available. After a card is played, it is turned into Non available and never played again until chosen game ends.

Card Class: This class is used to represent cards. It is used by both player and simulation classes. After cards are distributed randomly by the Simulation Class to the players, players will make their move. Simulation will decide who wins the current trick according to card numbers and types. Also, hashIndex field is added to the Card Class. Since some information about the card is needed before playing it, these hash indexes will facilitate the search and speed up the process.

Deck Class: This class contains all the cards played in the King. Before a game begins, cards in the Deck Class are distributed to the players.

Player Class: Agents in the game are objects of Player Class. This class holds the object's cards and plays the cards when that objects' turn comes. In addition to the name and id, this class also has nextPlayerIndex attribute, which shows who will play after this agent plays. To be able to play a card, Player Class should call corresponding Game Class.

Simulation Class: This class is the main class of the system. At the beginning of each game, Simulation Class will distribute the cards. Later, it will run the next agent that is about to play. After all agents play their cards, Simulation will decide the winner of the trick. When the game is finished, it will update the scores of players accordingly.

Globals Class: All information that is necessary for decision classes is stored in this class. Globals Class does not contain any methods, it contains only player and card information. Before a player plays its card, Game Class will decide which card to play by considering the information on the Globals Class. After the agent plays the card, it will update the related fields of Globals Class.

Game Class: The decision trees are stored in Game Class. Before a player plays its card, it will call corresponding Game Class object. Game Class will consider all necessary fields in the Globals Class and will return a card accordingly. Also, it is Game Class' work to update Globals Class before returning the determined card. Since King consists of seven different games, seven different decision classes are needed. These classes and the inheritance relationship is shown in figure below.

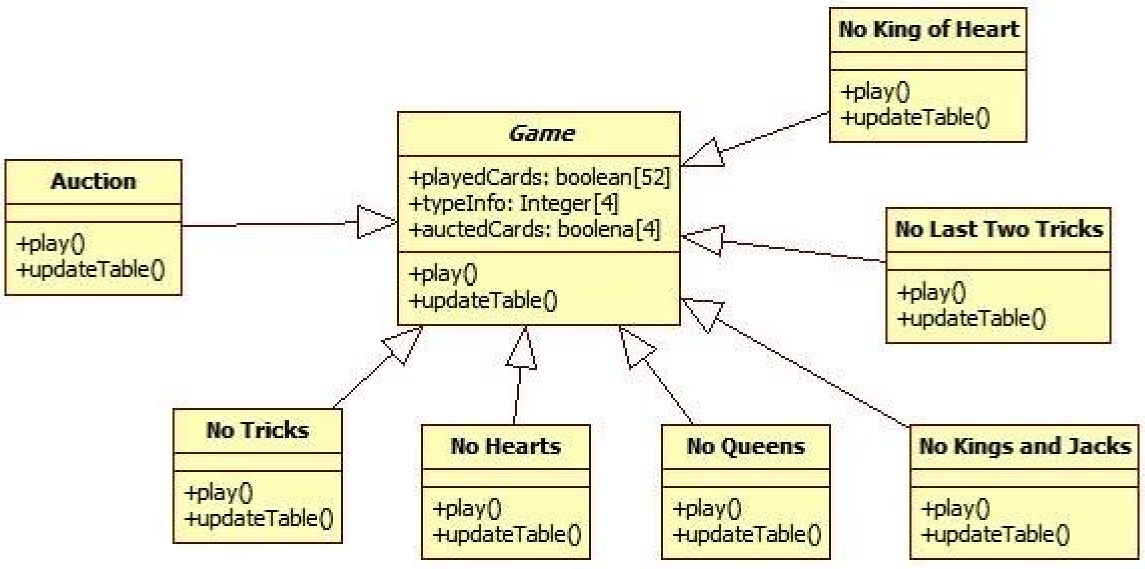


Figure 8: Class Diagram of Game Class

In order not to implement different play functions for different games in the Player Class, all decision classes are inherited from abstract Game Class. In this way, Player Class just calls the necessary Game Class and returns a card. Since different strategies are needed and different fields are considered for different games, play method of abstract Game Class is overridden by all decision classes. updateTable method is also overridden since each game changes the score differently.

5.5 Dependency Viewpoint

In this section, entities of the system and interactions between these entities are explained in detail.

Master King Project has 4 main entities:

- Server Database
- Server Engine
- Network
- Client

The diagram below shows the entities and the relations between them.

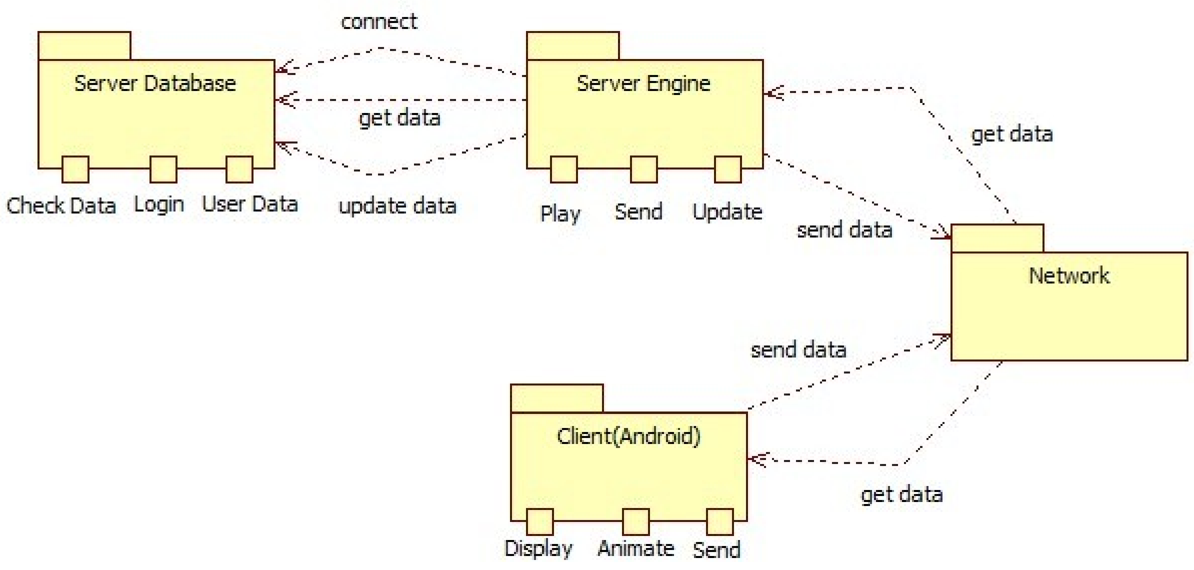


Figure 9: Entities of the System

5.5.1 Design Elements

Server Database: This entity is responsible for storing user data and assisting Server Engine to use this data. It checks the information sent by user before login or registration. Moreover, before a registered user leaves the system, Server Engine updates score of the user.

Server Engine: This entity runs the game. It gets the card played by the user and calls agents to maintain the game. After all agents play their cards, it sends the information to be displayed on the screen. This entity also updates the scores of users when the game is over.

Network Entity: Since the Master King project is a server based system, a network connection is needed. While Server Engine and Client sends information to each other, network will be mediator for the communication.

Client Entity: This entity is responsible for both handling user interaction and displaying game flow. After user clicks the card he/she wants to play, client sends this card to the Server Engine, animates the card and waits for Server Engine to run agents. After Server Engine sends the cards, Client gets, displays and animates them. Later on, it again waits for user interaction.

5.6 Information Viewpoint

In the system, only user information is stored on the database. This information will be used for score updates and login processes of the users. Since only user information is necessary, the database will have one table. The table is shown below.

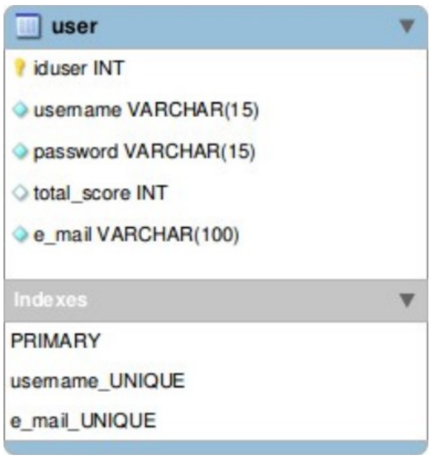


Figure 10: ER Diagram

The detailed information about the database is the following:

5.6.1 General Structure

In the database, there is one table which is “User”. Username and password is used to connect the user to the server. After the connection is successful, database will not be used until the game ends. When the game ends, database will be updated by adding current score of the user to the score column. After the score column is updated, user can start a new game or logout from the system.

5.6.2 Accessing to Database

When the client tries to login, it connects to database by using JDBC. After this point, when a user is connected, username and password will be checked. If the signing in is successful, the user will be directed to server engine and start playing the game.

5.6.3 Changing the Database

When the client application wants to log out, JDBC is used again and score is updated. The write operation will be done only when client ends the game and wants to logout.

5.6.4 Security of the Database

Since user passwords are stored in the database, there should be encrypted connection so that passwords and other data are stored securely. In order to protect the server and database, SSL will be used. Moreover, in order not to lose information in case of any problem, all user data will be backed up.

5.7 Patterns Use Viewpoint

In the Master King Project, mainly two architectural patterns are used. These are Client – Server Architectural Pattern and Object – Oriented Architectural Pattern.

5.7.1 Client – Server Architectural Pattern

The system will be implemented by using Client – Server Architectural Pattern. This pattern includes distributed systems that involve a separate client, server and connecting network. The server may send responses using a range of protocols and data formats to communicate with the client. The system will have a server containing the system logic and database. It will take the card played by the user, server will make its own moves and result will be sent back to the client.

Benefits of this architectural pattern are the following:

- **Multiple Access:** This architectural pattern supports many client simultaneously and it is the best pattern for on-line applications.
- **Ease of Maintenance:** Since responsibilities are distributed between server and clients, the client will not be affected by any update or repair.
- **Security:** All data is stored on the server, offering a greater control of security than the client machines.
- **Centralized Data Access:** Since data and logic is stored only on the server, accessing and updating is easier than the other architectural patterns.

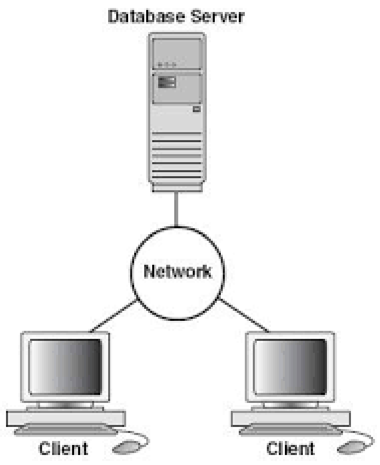


Figure 11: Client-Server Architecture

5.7.2 Object – Oriented Architectural Pattern

Other architectural pattern that Master King Project based on is object – oriented architectural pattern. In this pattern, responsibilities of a system is divided into cooperating individual objects, each containing its own data and behavior. The cooperation is done through interfaces, methods and messages.

Benefits of this architectural pattern are the following:

- **Testability:** Since an object - oriented system consists of individual objects, the entire system and each individual object can be tested, which leads to a more reliable application.
- **Re-usability:** Polymorphism and abstraction provides a reusable system.
- **Understandability:** Since objects in an object – oriented system is created by imitating real world objects, understanding the components and relations between them is much easier.
- **Extendibility:** Since an object – oriented system is a set of individual components, adding new functionality to the system is done by adding new components and relations to the system, without changing any previous component or relation.

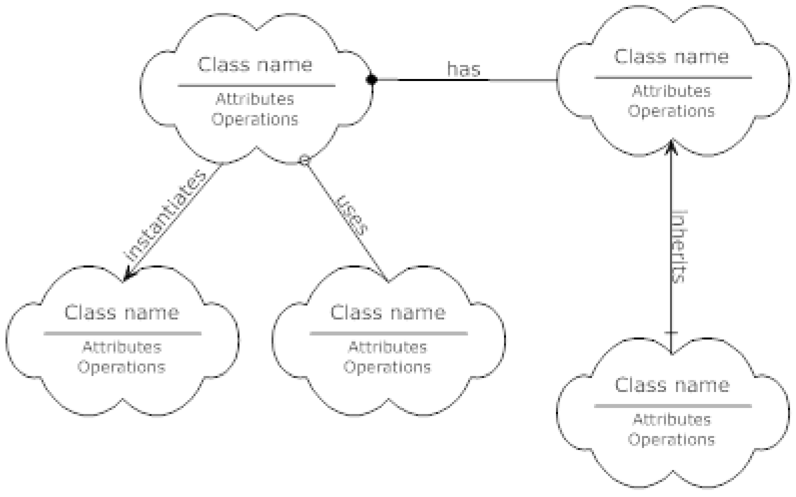


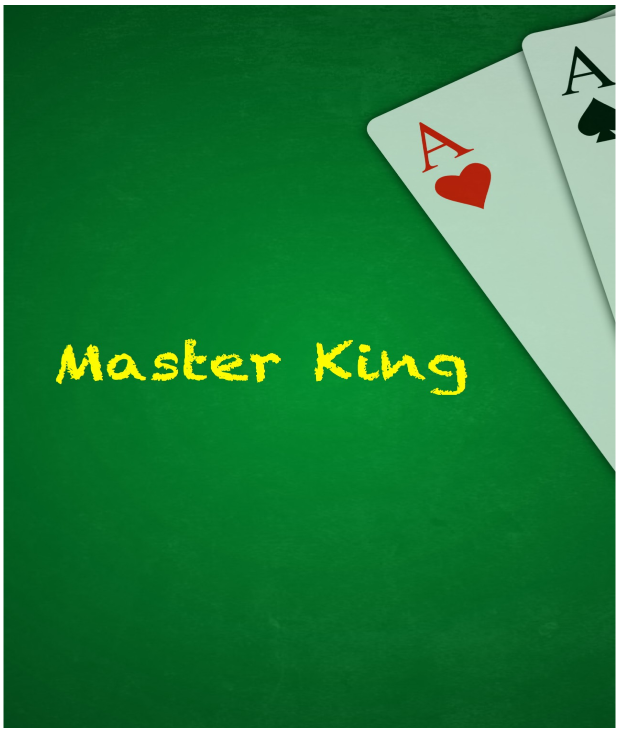
Figure 12: Object-Oriented Design Pattern

5.8 Interface Viewpoint

In order to visualize the project, Interface Viewpoint is necessary. In this section, some visual aspects of the system are explained.

5.8.1 Entrance Screen

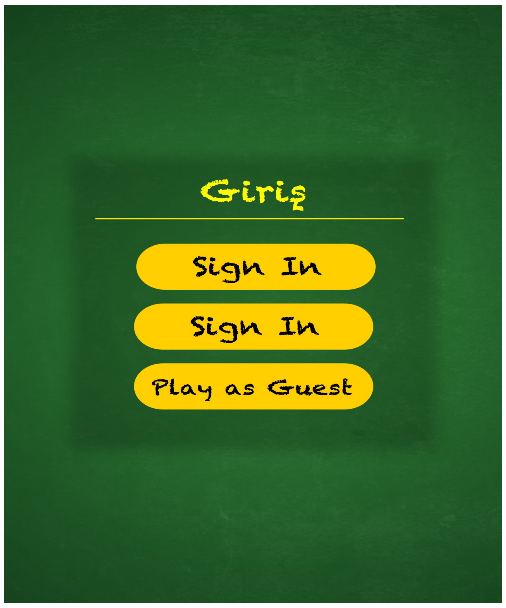
This is the initial screen that opens when user starts the application.



Interface 1 : Entrance Screen

5.8.2 Sign In Screen

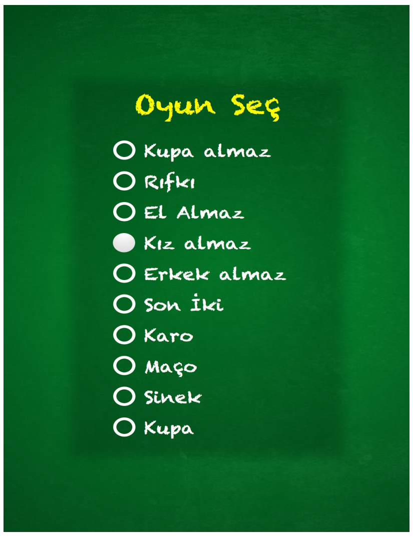
After user clicks anywhere on the first entrance screen, this screen pops up. In this screen, user can choose the operation he/she wants to do.



Interface 2 : Sing In Screen

5.8.3 Choose Game Screen

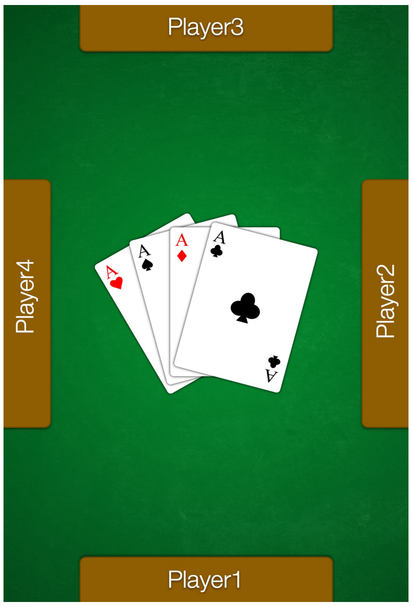
Before user starts playing a game, he/she can choose the game he/she wants to play.



Interface 3 : Choose Game Screen

5.8.4 Game Play Screen

This screen is the main play screen displayed while user is playing.



Interface 4 : Game Play Screen

5.9 Interaction Viewpoint

5.9.1 Design Concerns

Master King is a server based single player card game. In order for user to keep playing, client and server must interact continuously. Data flow should never stop until game ends and one component should wait for other components before playing. These interactions are explained in this section with details. UML diagrams are used during the explanation of these interactions between client and server.

5.9.2 Choose Language Interaction

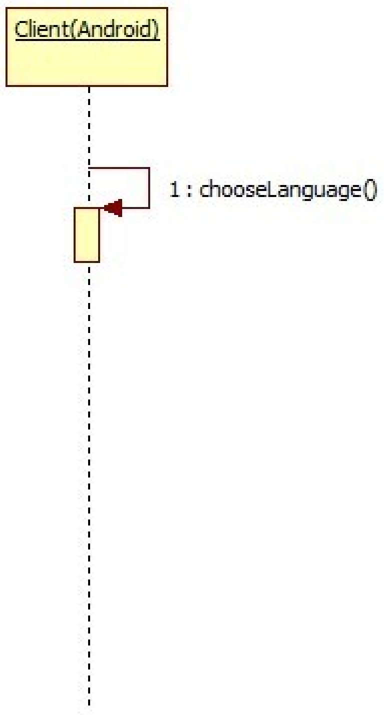


Figure 13: Choose Language Sequence Diagram

The diagram above explains Choose Language Interaction. When the user wants to change the language of the game, he/she clicks the related button. Since language is only clients concern related with Graphical User Interface, no database or server interaction is needed to change language. The screens, buttons and other visual means will be displayed accordingly.

5.9.3 Choose Game Interaction

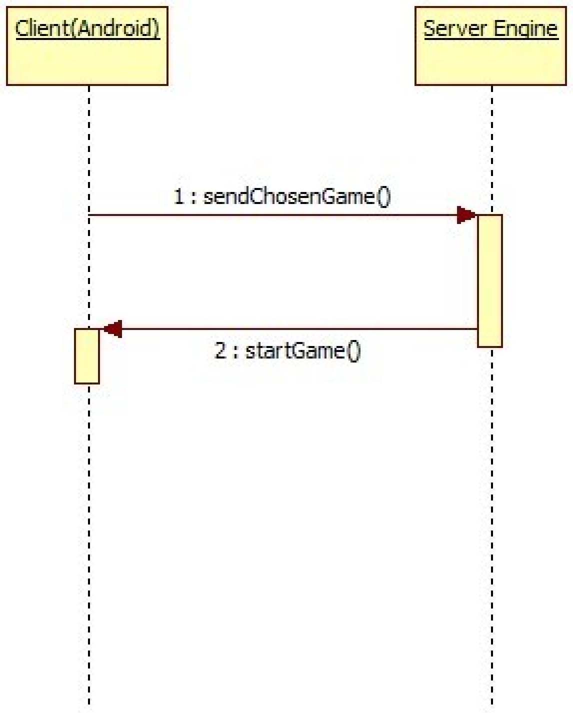


Figure 14: Choose Game Sequence Diagram

The diagram above explains Choose Game Interaction. When the user is to determine the game to be played, he/she clicks the related button. After the button is clicked, the request is sent to the server engine. Related decision and simulation class is called in the engine and informs back the client that engine is ready.

5.9.4 Play Card Interaction

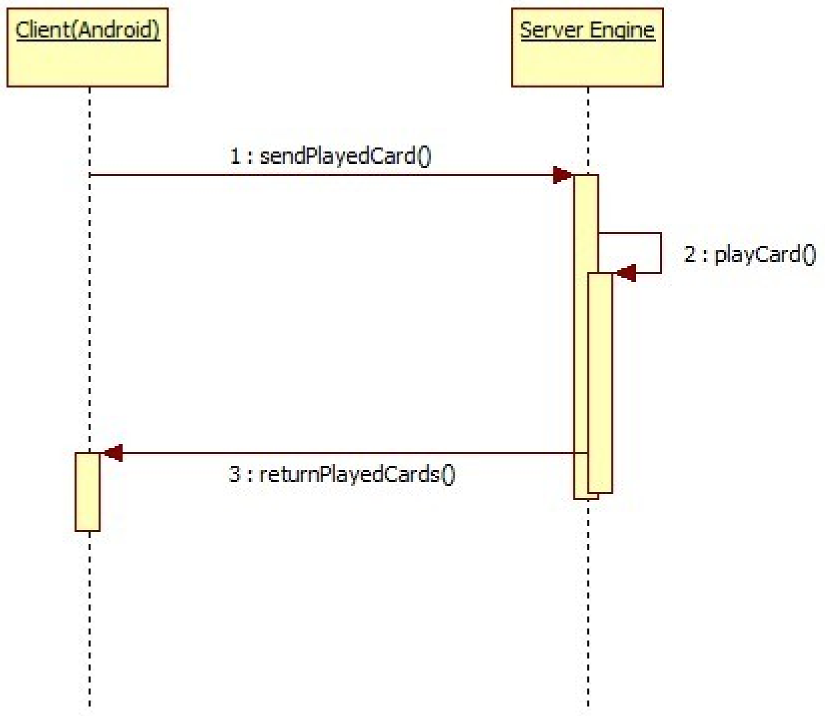


Figure 15: Play Card Sequence Diagram

This diagram above explains Play Card Interaction. Server Engine waits for player to play. When user clicks on the card that he/she wants to play, that card is sent to the Server Engine and client begins to wait. The server engine gets the card and run the agents accordingly. After all agents play, all cards played by the agents are sent to the client in order to draw and animate on the screen.

5.9.5 Display Score Table Interaction

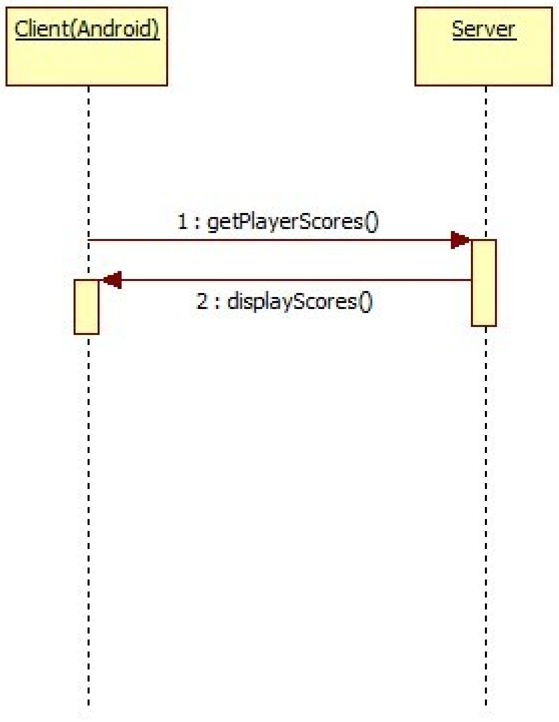


Figure 16: Display Score Sequence Diagram

This diagram above explains Display Score Table Interaction. When user wants to see the scores of the players, he/she clicks the related button. After display button is clicked, the request is sent to the Server Engine by the client. Server gets scores of the players and sends them to the client. Client displays the score table.

5.9.6 Register Interaction

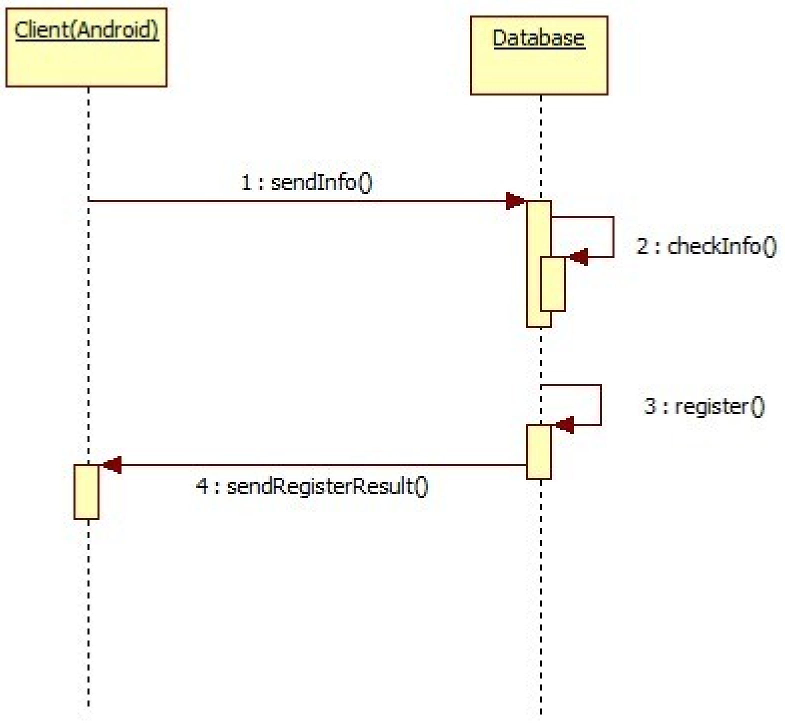


Figure 17: Register Sequence Diagram

This diagram above explains Register Interaction. When unregistered user fills all necessary fields in the Register Screen, clicks the register button. After this click, client gets the data and sends it to the database. Database checks the coming data whether or not it is proper for registration. If there is no problem with the data, database saves it. The registration result is sent back to client.

5.9.7 Login Interaction

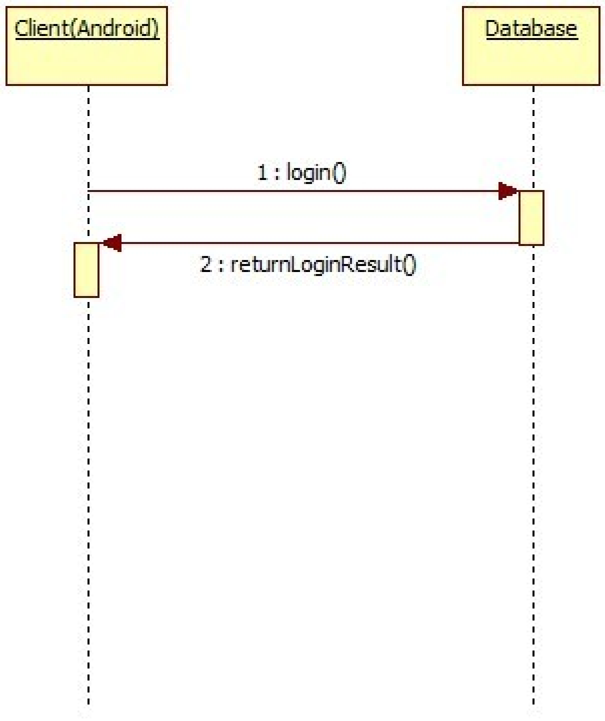


Figure 18: Login Sequence Diagram

This diagram above explains Login Interaction. When a registered user wants to login to the system, he/she fills all necessary fields in the Login Screen and clicks the button. After the click, the data is sent to the database. Database checks the user information and sends the login result to the client.

5.9.8 Logout Interaction

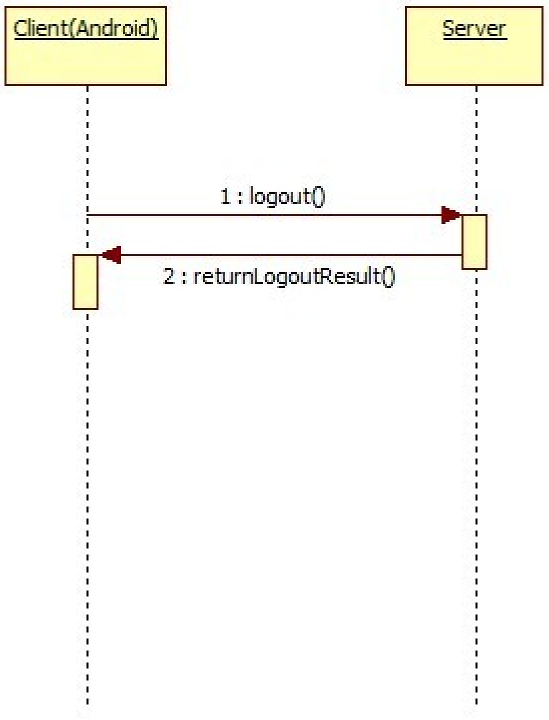


Figure 19: Logout Sequence Diagram

This diagram above explains Logout Interaction. When a registered user wants to logout from the system, he/she sends the request to the Server Engine. After request is received, Server Engine does necessary operations about user and user leaves securely.

5.10 State Dynamics Viewpoint

Since Master King Project is an user interactive system, State Dynamics is also important. State Dynamics Diagram divides the system into states and explains the functionality on each state. In this section, transitions between states and conditions leading to these transitions are explained. In the diagram below, all states of the system and interactions between them are shown.

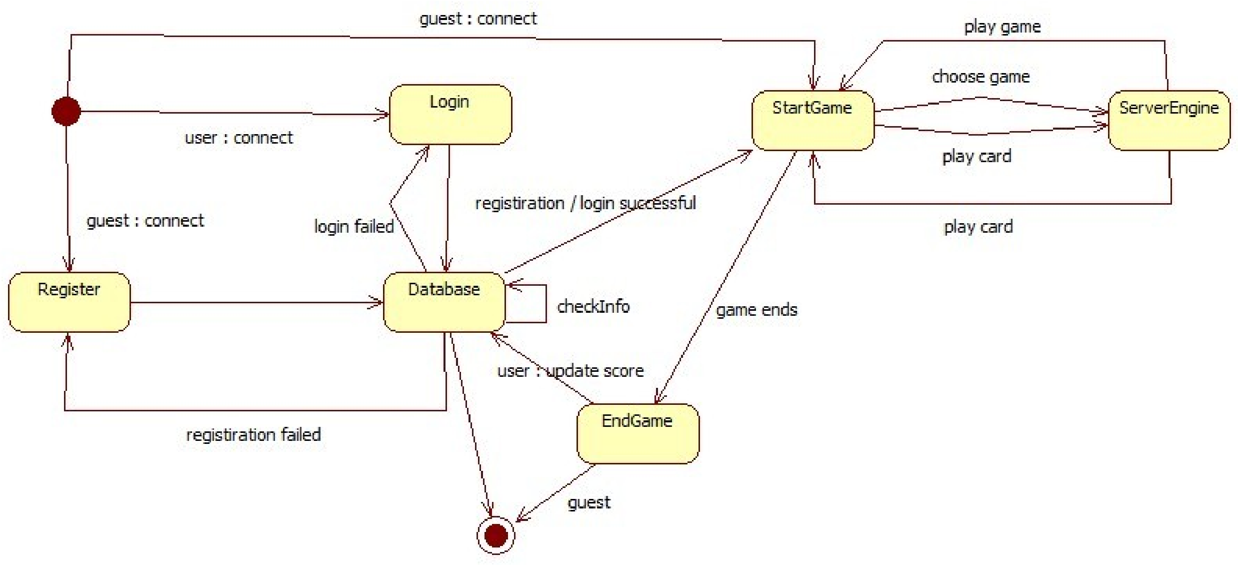


Figure 20: State Diagram

5.10.1 Design Elements

Initial state is waiting state. After user connected to the system and clicks a button from any initial screen, transitions begin.

Login State: If a registered user fills the necessary fields and clicks the Login button, the system passes to Login State. The database connection for Login process is handled here.

Register State: If an unregistered user fills the necessary fields and clicks the Register button, the system passes to the Register State. The database connection for Registration process is handled here.

Database State: All necessary procedures related to database are done here. When information is sent to the database from Login or Register state, database first checks the information whether or not it is proper for registration/login. If the information is proper for registration, the user is saved on the database and he/she is directed to the StartGame state. If it is not, he/she is directed to the Register state, expected to fill necessary fields again. Similarly, if the information is proper for Login, the user is directed to the StartGame state. If the information is not proper, the user is directed to the Login state, expected to fill necessary fields again. Moreover, before a registered user logs out from the system, he/she is directed to Database state and his/her score is updated.

StartGame State: Users are directed to this state when registration/login is successful. However, if a person does not want to register, he/she can click Play As Guest button and can directly start playing. Since Master King Project is server based system, this state should continuously interact with ServerEngine State. In this state, users can choose which game to play and after game starts, they can play their cards. The system passes to this state when it is users turn to choose a game or play card.

ServerEngine State: After a user chooses a game or plays his/her card, the system passes to this state. In this state, if user has chosen a game, related decision classes are called and clients move is waited. If user plays a card, this state runs the agents and returns played card to the client.

EndGame State: After a King game finishes and users want to leave, the system passes to EndGame state. In this state, if it is guest to leave the game, he/she leaves directly. If a registered user is about to logout, this state directs the user Database state to update score.

6. Appendixes

6.1 Table of Figures

6.2 Table of Tables

6.3 Table of Interfaces