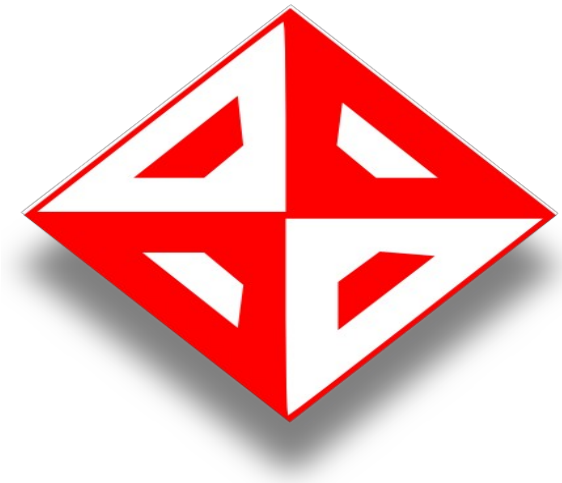
# SOFTWARE DESIGN DOCUMENT v1.1

**01.03.2015**

**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, Composition, 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 |
| TCP/IP | Transmission Control Protocol/Internet Protocol |
| 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 and scores should be stored in database. Therefore, database is required if the actor is user.

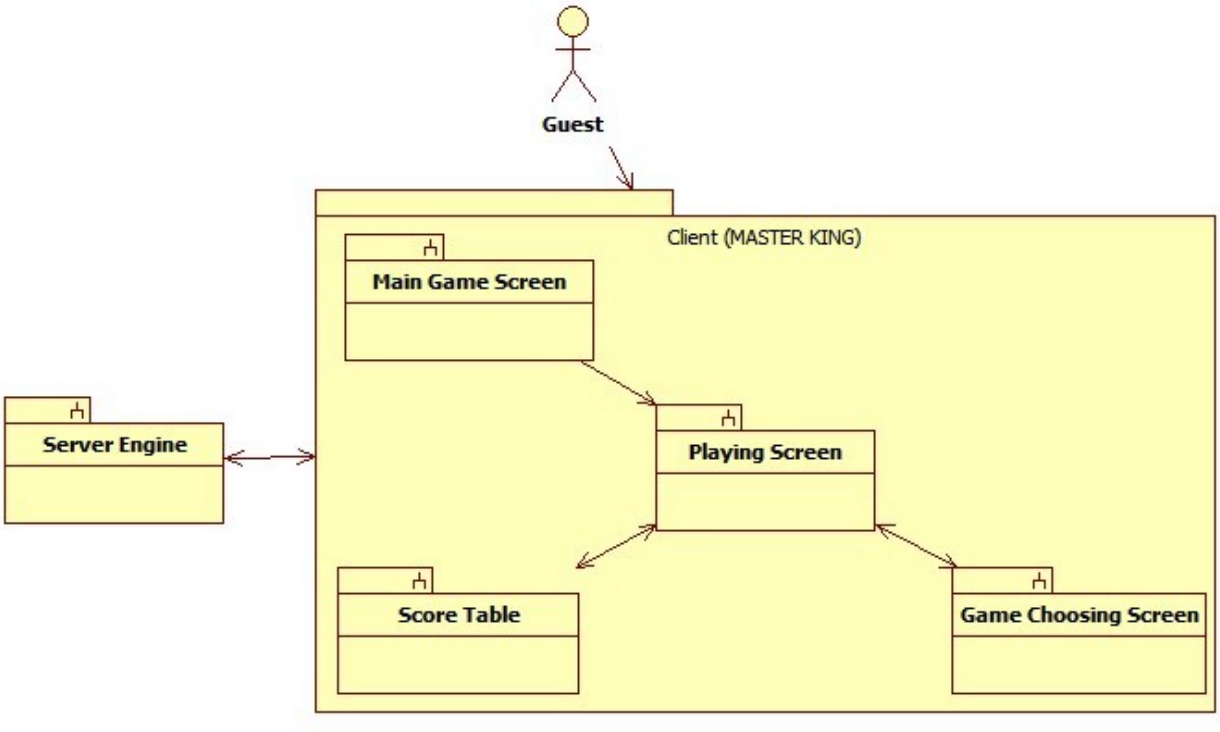



Figure 1: Guest Context Diagram

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 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.

## 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 updated SDD for the project Master King by team REBELLION. The date of release for this document is 01.03.2015. 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, product context, system boundaries and other limitations are already stated. Also, detailed explanations 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, Composition, 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 explanation 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.
- Composition Viewpoint describes the system's submodules and their roles.
- 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 between 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

In the project, Android Platform has been chosen because it is most widely used Operating System in Smart Phones, making easier to reach more people. Since Android Platform is compatible with

Java Programming Language, we shall develop in Java. The project requires some Object-Oriented paradigms so that we have used basic Object-Oriented architectural pattern which are explained in detail in the Section 5.7.2.

The reason why we have chosen server-client architectural pattern is to reduce the computational workload of the phones. In this way, the decision mechanisms by agents are done in the server. Client only sends the required data to the server and reflects cards on the screen. The details of this architecture is explained in detail in the section 5.7.1.

## 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. Conceptual flow and use cases are described here.

### 5.2.1 Conceptual Flow

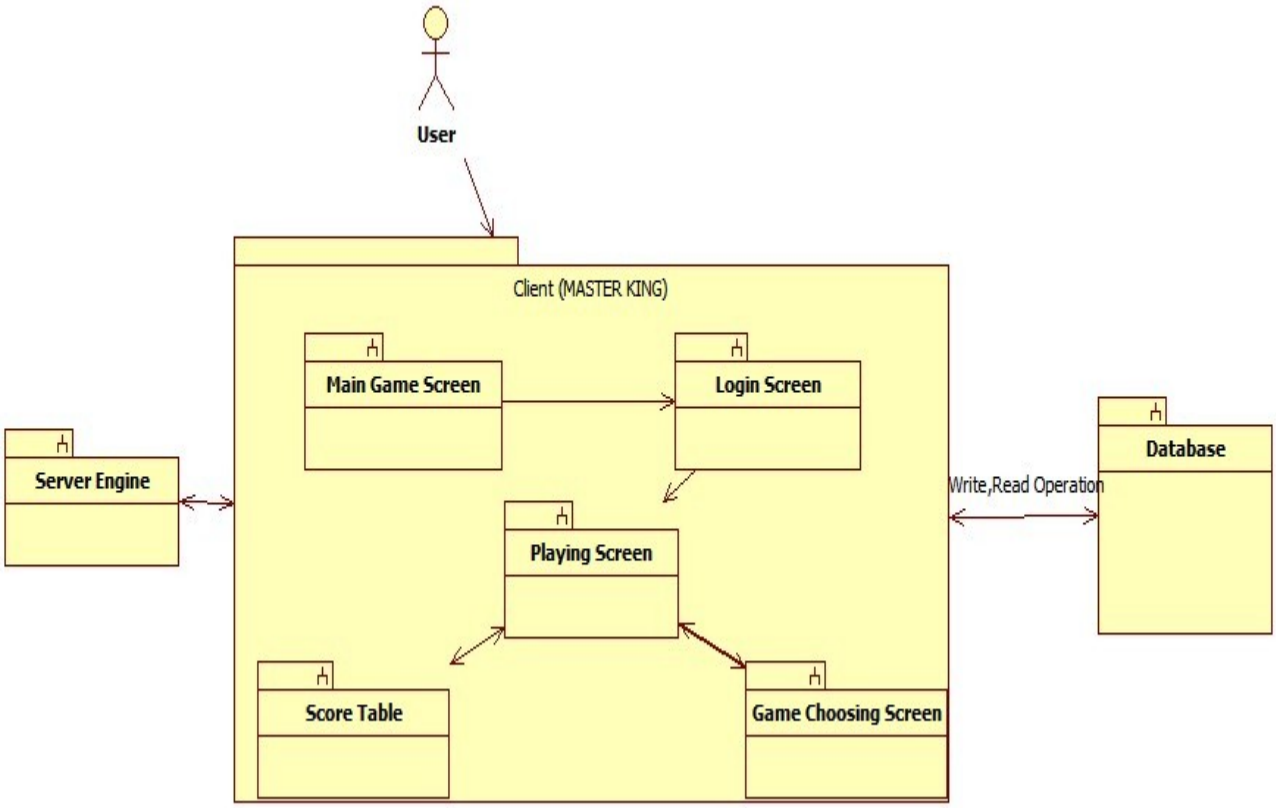



Figure 2: User Context Diagram

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 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.

## 5.2.2 Use Cases

### 5.2.2.1 'Choose Language' Use Case for Guest/User


Figure 3: Choose Language Use Case Diagram for Guest


Figure 4: Choose Language Use Case Diagram for User

| | |
|---|---|
| Use Case ID | UC1 |
| Use Case Name | Choose Language |
| Description | This use case describes the event in which an unregistered user / user wants to choose the language before starting the game. |
| Actors | Guest/User |
| Preconditions | - |
| Trigger | The unregistered user / user clicks on the corresponding icon of the one of two languages in the Choose Language section. |
| Basic Flow | 1- The unregistered user/ user opens the mobile phone application.<br>2- The unregistered user/ user clicks the icon of the language that he/she wants to choose. |
| Exception Flow | - |
| Post Conditions | - |

Table 3: Explanation for Choose Language Use Case Diagram

#### 5.2.2.2 'Choose Game' Use Case for Guest/User

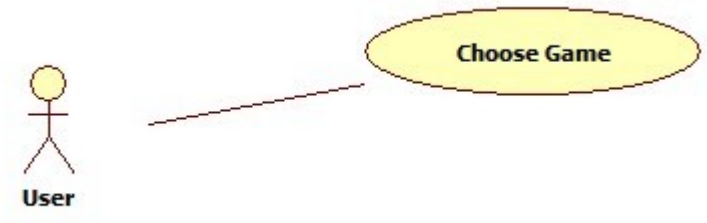



Figure 5: Choose Game Use Case Diagram for User

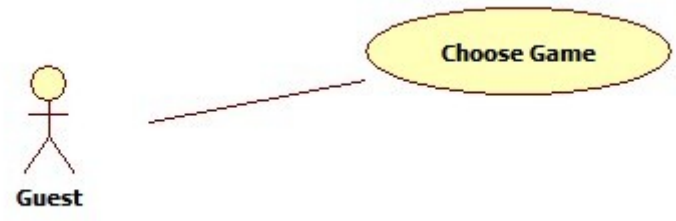



Figure 6: Choose Game Use Case Diagram for Guest

| | |
|---|---|
| Use Case ID | UC2 |
| Use Case Name | Choose Game |
| Description | This use case describes the event in which an unregistered user/ user chooses the King Game that he/she wants to play. |
| Actors | Guest/User |
| Preconditions | To be able to choose any game in the King, it must be guest's/user's turn. |
| Trigger | The unregistered user/ user clicks on the corresponding icon of the game that is allowed in that time. |
| Basic Flow | 1- The unregistered user / user opens the mobile phone application.<br>2- The unregistered user / user clicks the icon of the language that he/she wants to choose.<br>3- If it is guest's / user's turn, a window pops up in which player can choose one of allowed game he/ she desires. Otherwise, she/ he has to keep playing until his/her turn comes. |
| Exception Flow | - |
| Post Conditions | - |

Table 4: Explanation for Choose Game Use Case Diagram

#### 5.2.2.3 'Play Card' Use Case for Guest/User


Figure 7: Play Card Use Case Diagram for Guest


Figure 8: Play Card Use Case Diagram for User

| | |
|---|---|
| Use Case ID | UC3 |
| Use Case Name | Play Card |
| Description | This use case describes the event in which an unregistered user / user plays his/her trick in order to complete the trick under consideration. |
| Actors | Guest/ User |
| Preconditions | To be able to play the card of any game in the King, it must be guest's/ user's turn. |
| Trigger | The unregistered user / user clicks on the corresponding card of his/her hand. |
| Basic Flow | 1- The unregistered user / user opens the mobile phone application.<br>2- The unregistered user / user clicks the icon of the language that he/she wants to choose.<br>3- If it is guest's / user's turn, the guest clicks the card he/she wants to play. |
| Exception Flow | - |
| Post Conditions | - |

Table 5: Explanation for Play Card Use Case Diagram

#### 5.2.2.4 'Display Score Table' Use Case for Guest/User

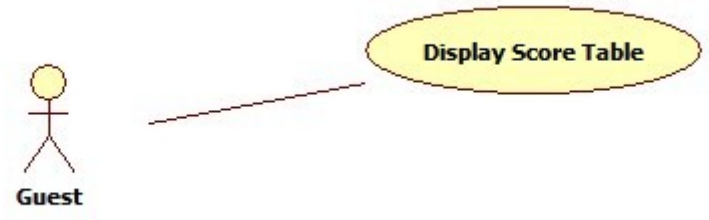



Figure 9: Display Score Table Use Case Diagram for Guest

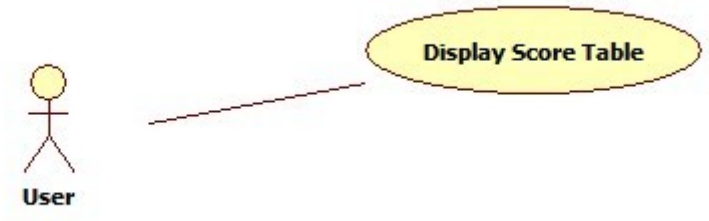



Figure 10: Display Score Table Use Case Diagram for User

| | |
|---|---|
| Use Case ID | UC4 |
| Use Case Name | Display the Score Table |
| Description | This use case describes the event in which an unregistered user / user displays the score table that contains the current scores of all players in the game. |
| Actors | Guest/ User |
| Preconditions | - |
| Trigger | The unregistered user / user clicks on the corresponding icon in order to see the score table. |
| Basic Flow | 1- The unregistered user / user opens the mobile phone application.<br>2- The unregistered user / user clicks the icon of the language that he/she wants to choose.<br>3- The unregistered user / user clicks the icon in order to see the current score situation. |
| Exception Flow | - |
| Post Conditions | - |

Table 6: Explanation for Display Score Table Use Case Diagram

#### 5.2.2.5 'Register' Use Case for Guest


Figure 11: Register Use Case Diagram for Guest

| | |
|---|---|
| Use Case ID | UC5 |
| Use Case Name | Register |
| Description | This use case describes the event in which a guest determines his/her user name and password and registers to the system |
| Actors | Guest |
| Preconditions | - |
| Trigger | The unregistered user clicks on the registration icon of the page in order to be inserted into the database by administrators. |
| Basic Flow | 1- The unregistered user opens the mobile phone application.<br>2- The unregistered user clicks the registration icon.<br>3- The unregistered user enters his/her user name, password and mail address.<br>4- The unregistered user clicks on the confirm button. |
| Exception Flow | -If the mail address or user name already exists in the database an error message will be shown.<br>-If unregistered user cannot be inserted due to system problems, an error message will be shown. |
| Post Conditions | - |

Table 7: Explanation for Display Score Table Use Case Diagram

#### 5.2.2.6 'Login' Use Case for User

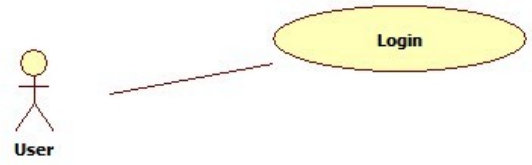



Figure 12: Register Use Case Diagram for User

| | |
|---|---|
| Use Case ID | UC6 |
| Use Case Name | Login |
| Description | This use case describes the event in which a registered user logins the system using his/her user name and password. |
| Actors | User |
| Preconditions | - |
| Trigger | The registered user clicks the login button. |
| Basic Flow | 1- The registered user opens the mobile phone application.<br>2- The registered user clicks the icon of the language that he/she wants to choose.<br>3- The registered user enter his/her user name and password. Then, clicks the login button to proceed. |
| Exception Flow | -If user / admin enters wrong password or user name, an error message will be shown. |
| Post Conditions | - |

Table 8: Explanation for Login Table Use Case Diagram

#### 5.2.2.7 'Logout' Use Case for User

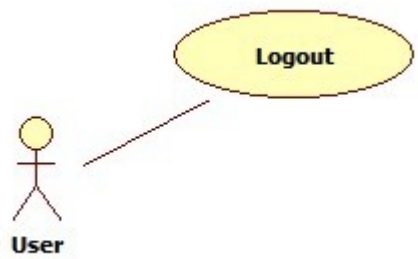



Figure 13: Logout Use Case Diagram for User

| | |
|---|---|
| Use Case ID | UC7 |
| Use Case Name | Logout |
| Description | This use case describes the event in which a registered user logout from the system by clicking the logout button. |
| Actors | User |
| Preconditions | - |
| Trigger | The registered user clicks the logout button. |
| Basic Flow | 1- The registered user opens the mobile phone application.<br>2- The registered user clicks the icon of the language that he/she wants to choose.<br>3- The registered user enter his/her user name and password. Then, clicks the login button to proceed.<br>4- The register user clicks the logout button. |
| Exception Flow | - |
| Post Conditions | - |

Table 9: Explanation for Logout Table Use Case Diagram

## 5.3 Composition Viewpoint

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

The component diagram shows the relationships between the components of the system. There are two main components which are Client and Server components.

The deployment diagram shows the relationships between the nodes of the system. Basically, the hardware components are user's smart phones and server side processing computers.

The following figures show the corresponding diagrams which are mentioned above.

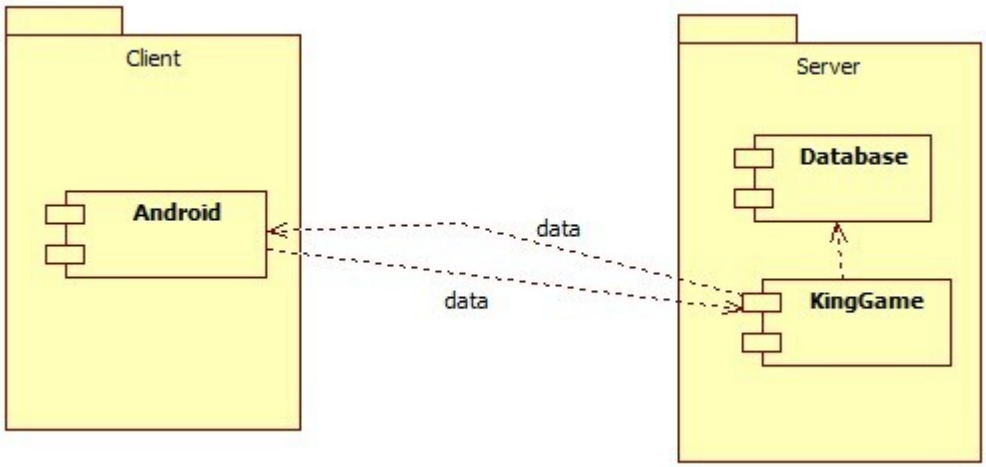



Figure 14: Component Diagram for Guest/User

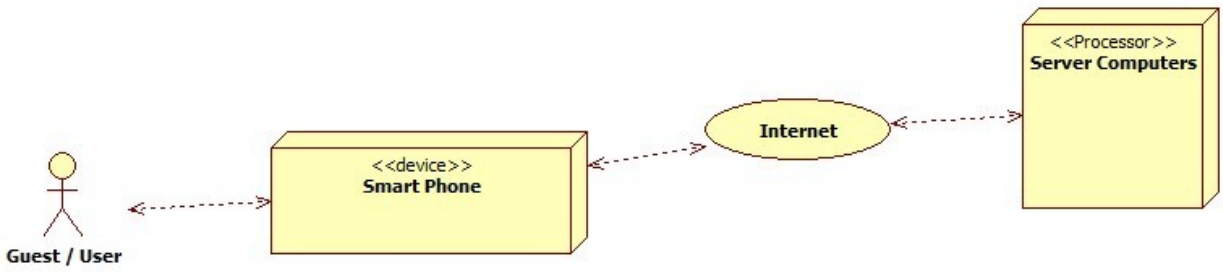



Figure 15: Deployment Diagram for the System

## 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, main system entities and relationships between them is shown.

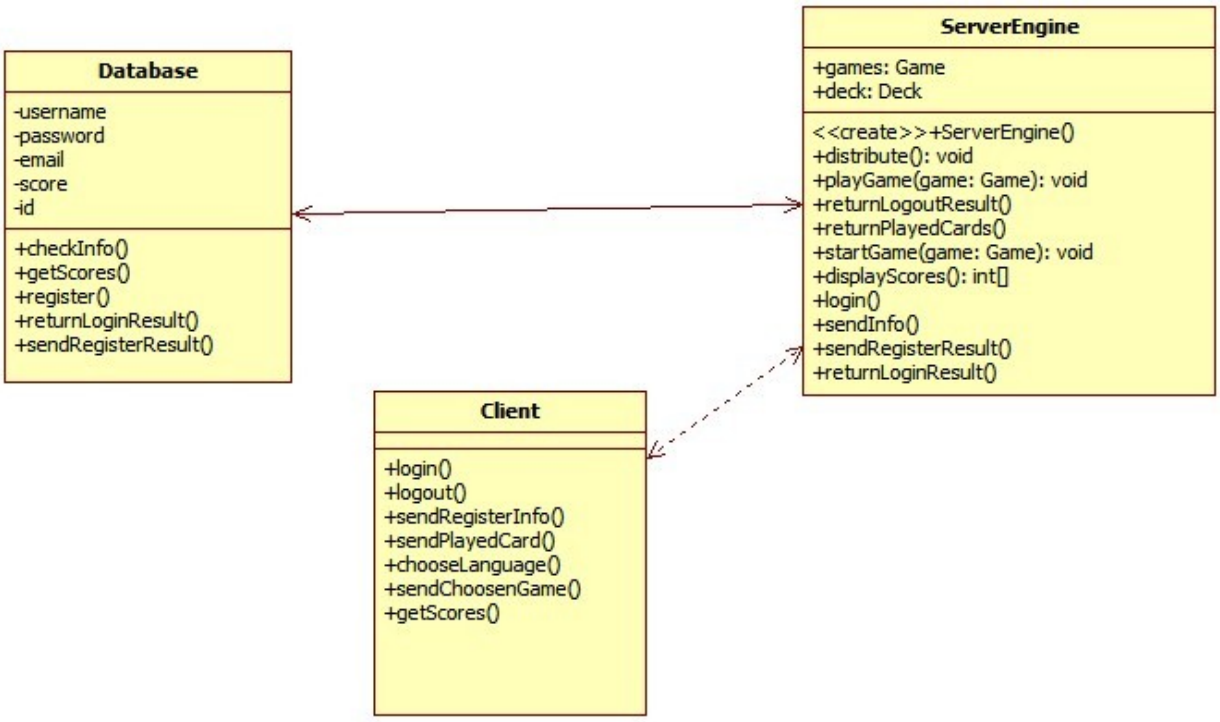



Figure 16: General Class Diagram of the System

For the decision mechanisms, server engine has the following class structures.

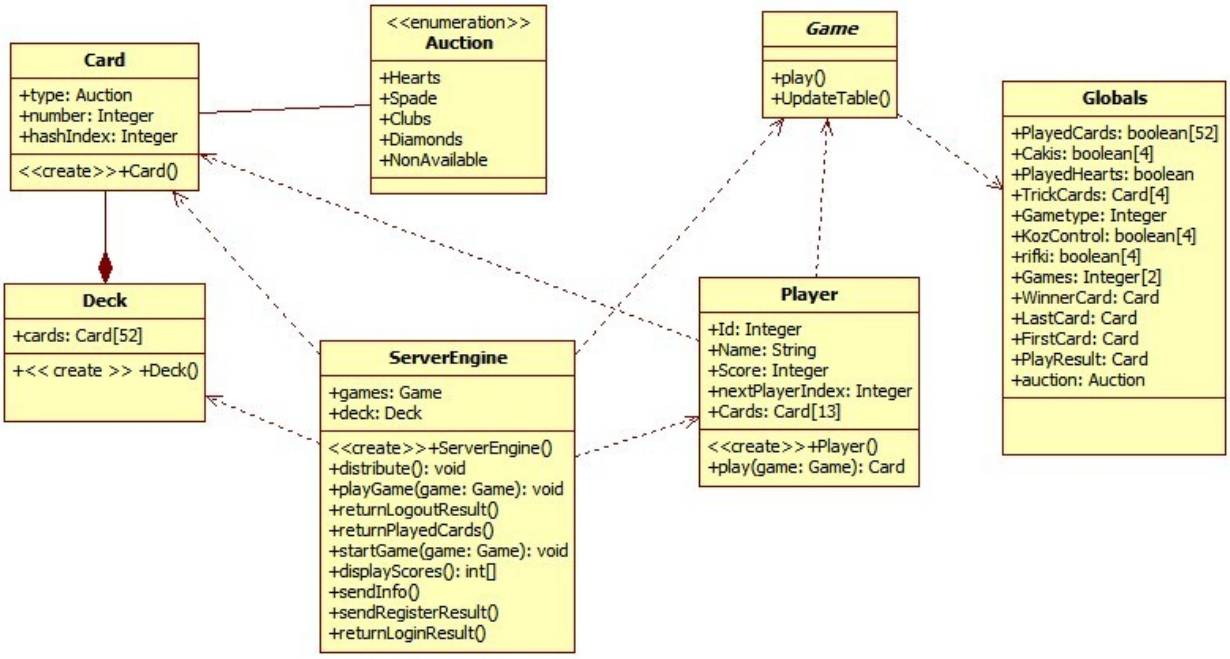



Figure 17: Class Diagram of the Server Side

Since King has seven different type of the games, inheritance of the game class has the following structure.

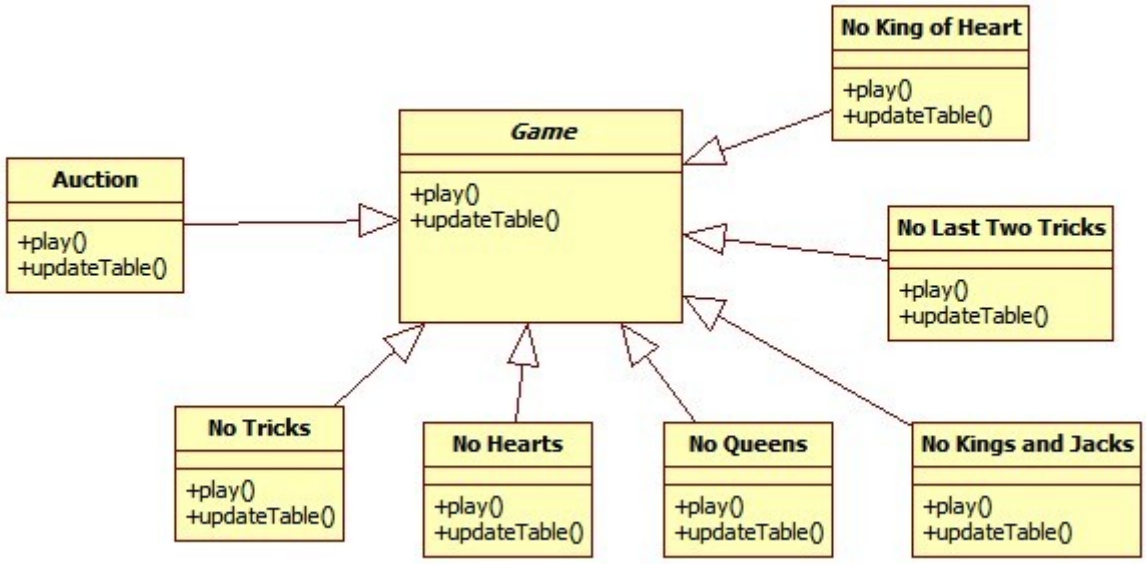



Figure 18: Class Diagram of Game Class

### 5.4.1 Design Elements

#### 5.4.1.1 Card Class

| Name | Method/ Field | Return Type/Identifier | Definition |
|---|---|---|---|
| type | F | Auction | Indicates type of an individual card |
| number | F | Integer | Indicates the value of an individual card. It ranges from 2 to 14. |
| hashIndex | F | Integer | Indicates hash value of an individual card. It ranges from 0 to 51. |
| Card() | M | - | Constructor of the Card Class. |

Table 10: Table of the Card Class

#### 5.4.1.2 Deck Class

| Name | Method/ Field | Return Type/Identifier | Definition |
|---|---|---|---|
| cards | F | Card[] | Represents the real deck which includes 52 cards. |
| Deck() | M | - | Constructor of the Deck Class. |

Table 11: Table of the Deck Class

#### 5.4.1.3 Auction Enumeration

| Name | Method/ Field | Return Type/Identifier | Definition |
|---|---|---|---|
| Hearts | F | Auction | Represents the Hearts in a real deck. |
| Spade | F | Auction | Represents the Spade in a real deck. |
| Clubs | F | Auction | Represents the Clubs in a real deck. |
| Diamonds | F | Auction | Represents the Diamonds in a real deck. |
| NonAvailable | F | Auction | Represents the played cards. |

Table 12: Table of the Auction Enumeration

#### 5.4.1.4 ServerEngine Class

| Name | Method/ Field | Return Type/Identifier | Definition |
|---|---|---|---|
| games | F | Game | Represents the each game in the King game. |
| deck | F | Deck | An Instance of a deck class |
| ServerEngine() | M | - | Constructor of the Simulation Class. |
| distribute() | M | void | Distributes the cards randomly to the players. Each player gets 13 cards. |

| | | | |
|---|---|---|---|
| returnLogoutResult() | M | boolean | Returns the success of the logout action |
| returnPlayedCards() | M | Card[] | Sends the cards played by the agents to the client |
| startGame(Game game) | M | void | Starts the game chosen by the user |
| displayScores() | M | Integer[] | Sends the scores of all players to the client |
| sendInfo() | M | void | Sends the user information to the database |
| login() | M | void | Directs login request from client to the database |
| sendRegisterResult() | M | boolean | Directs success of the registers action from database to the client |
| returnLoginResult() | M | boolean | Directs success of the login action from database to the client |
| playGame(Game game) | M | void | Simulates each King game according to the game rules. It determines the next player having the turn. |

Table 13: Table of the ServerEngine Class

#### 5.4.1.5 Player Class

| Name | Method/ Field | Return Type/Identifier | Definition |
|---|---|---|---|
| Id | F | Integer | Identifies the player uniquely. |
| Name | F | String | Identifies the name of the player. |
| Score | F | Integer | Holds the score of the player. |
| nextPlayerIndex | F | Integer | Holds the next player id. |
| Cards | F | Card[] | Represents the real cards of each player which includes 13 cards. |
| Player() | M | - | Constructor of the Player Class. |
| play(Game game) | M | Card | Determines the card which will be played. |

Table 14: Table of the Player Class

#### 5.4.1.6 Globals Class

| Name | Method/Field | Return Type/Identifier | Definition |
|---|---|---|---|
| PlayedCards | F | boolean[] | Indicates the played cards by players. |
| Cakis | F | boolean[] | Indicates whether a player has that type of card or not. |
| PlayedHearts | F | boolean | Indicates whether heart is played or not. |
| TrickCards | F | Card[] | Indicates the four cards played in one trick. |

| | | | |
|---|---|---|---|
| Gametype | F | Integer | Indicates type of game as an integer. |
| KozControl | F | boolean[] | Indicates whether a player has auction or not. |
| rifki | F | boolean[] | Indicates whether King of hearts is played or not. |
| Games | F | Integer[] | Indicates which king games can be selected. |
| WinnerCard | F | Card | Indicates the winner card in each trick. |
| LastCard | F | Card | Indicates the last card in each trick. |
| FirstCard | F | Card | Indicates the first card in each trick. |
| PlayResult | F | Card | Indicates the winner card in end of each trick. |
| auction | F | Auction | Indicates the auction type for Auction games. |

Table 15: Table of the Globals Class

#### 5.4.1.6 Game Class

| Name | Method/ Field | Return Type/Identifier | Definition |
|---|---|---|---|
| play() | M | Card | Returns a card according to game type and player's hand. |
| UpdateTable() | M | void | After each game,updates score table. |

Table 16: Table of the Game Class

#### 5.4.1.7 Client Class

| Name | Method/ Field | Return Type/Identifier | Definition |
|---|---|---|---|
| login() | M | void | Sends login request from client to the server |
| logout() | M | void | Sends logout request from client to the server |
| sendRegisterInfo() | M | void | Sends the information necessary for registration from client to the server |
| sendPlayedCard() | M | void | Sends the card played by the user to the server |
| chooseLanguage() | M | void | Sets the language of the interface according to user action |
| sendChoosenGame() | M | void | Sends the King game type chosen by the user to the server |
| getScores() | M | void | Sends a request from server to display scores on the screen |

Table 17: Table of the ServerEngine Class

### 5.4.1.8 Database Class

| Name | Method/ Field | Return Type/Identifier | Definition |
|---|---|---|---|
| username | F | String | Nickname of the registered users |
| password | F | String | Passwords of the registered users |
| email | F | String | Email addresses of the registered users |
| score | F | Integer | Total scores of the registered users |
| id | F | Integer | Unique ids of the registered users |
| checkInfo() | M | void | Checks the database whether the coming information is valid for registration/login |
| getScores() | M | void | Fetches the scores of the users and sends them to the server |
| register() | M | void | Saves the coming information to the database to register for the server |
| returnLoginResult() | M | boolean | Returns the success of login action to the server |
| sendRegisterResult() | M | boolean | Sends the success of login action to the server |

Table 18: Table of the Database Class

### 5.4.1.6 Child Classes of Game Class

Auction, No Tricks, No Hearts, No Queens, No Kings and Jacks, No Last Two Tricks, No King of Heart classes are child classes of game class. Like the game class, all of these classes have same methods. However, each of them implements according to its own rules.

play(): Returns a card according to game type and player's hand.

UpdateTable(): After each game, updates score table.

## 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.

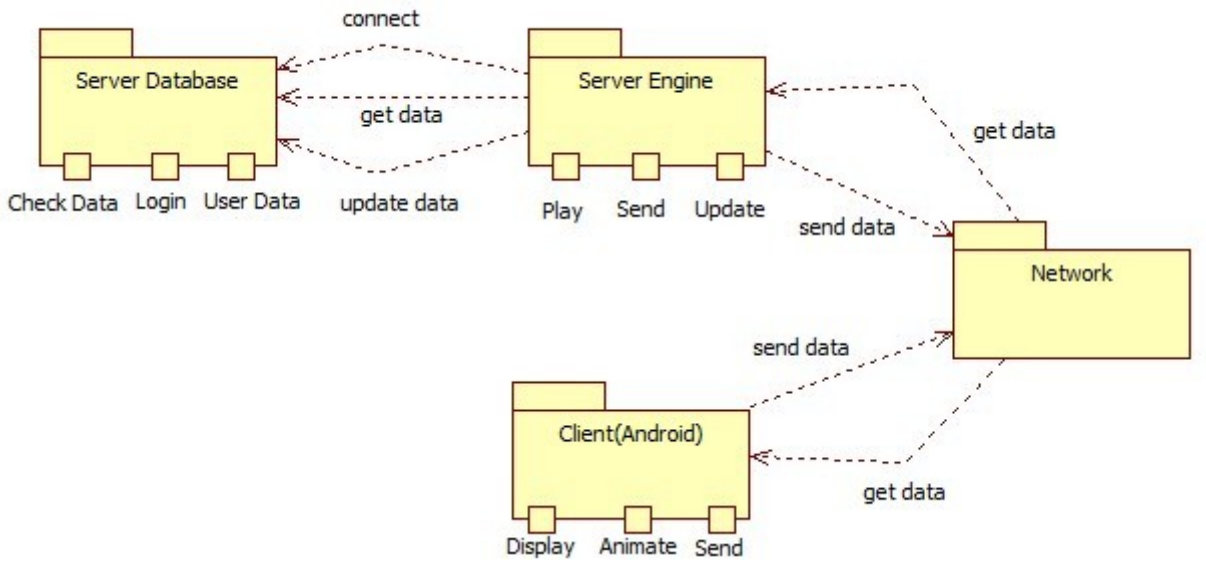



Figure 19: Entities of the System

### 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.

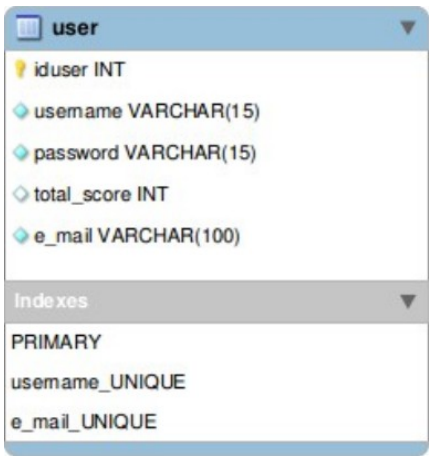



Figure 20: ER Diagram

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.

#### 5.6.1.1 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.

#### 5.6.1.2 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.

### 5.6.2 Information Flow

In this application, there is a data flow between client and server. To communicate with the server, Android application uses TCP/IP protocol. Android opens a socket to connect to the server. Data communication is supplied over this socket. Data are casted into the JSON objects before sending.

Data types that are flown between the server and client are listed below:

- Played cards played by agents are sent to client to be drawn.
- Scores of the players are sent to the client to be reflected.
- Played card by the user is sent to the server.
- The king game type chosen by the user is sent to the server.
- After the login operation of the user, username and password are sent to the server to check the success of the login operation.
- To check the registration operation, filled information is sent to the server to save the registered user.

This is the basic information flow.

Also, the agents use some data structures for their decision mechanisms. These data structures are explained in the Logical Viewpoint section 5.4.

## 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.

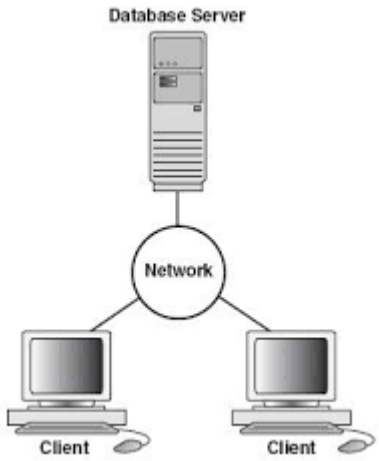



Figure 21: Client-Server Architecture

### 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.

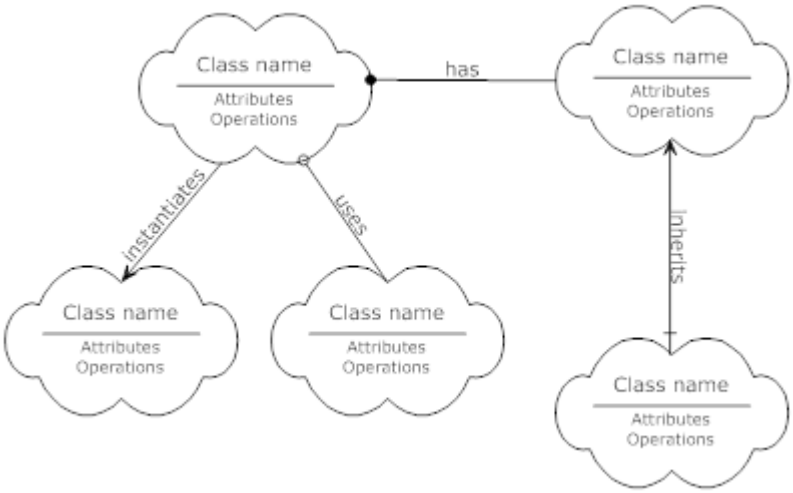



Figure 22: Object-Oriented Design Pattern

## 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: Sign in Screen

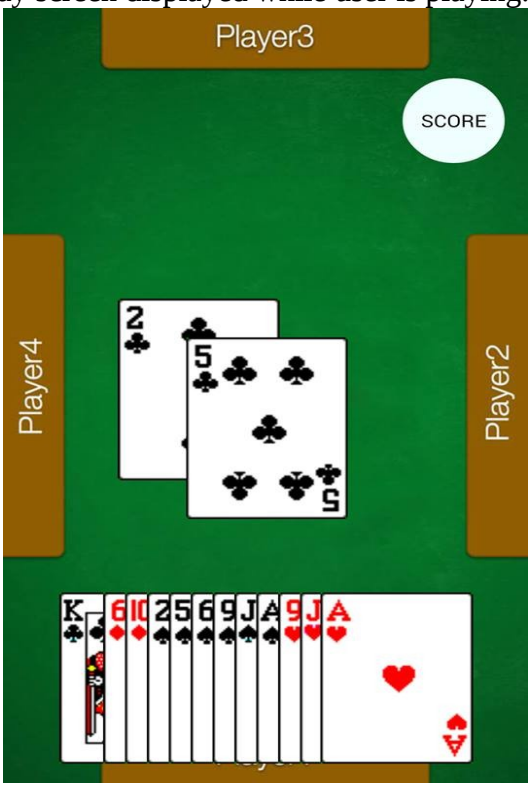
### 5.8.3 Choose Game Screen

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

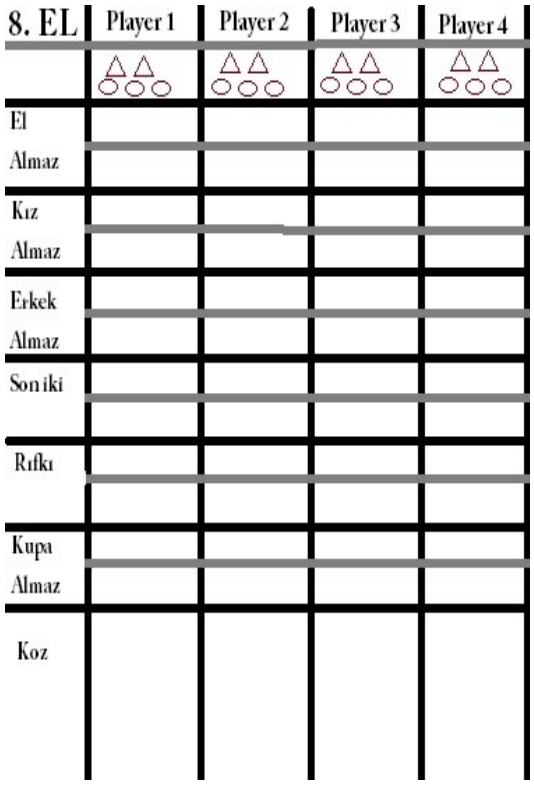
This screen is the main play screen displayed while user is playing.




Interface 4: Game Play Screen

### 5.8.5 Score Table Screen

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




Interface 5: Score Table 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

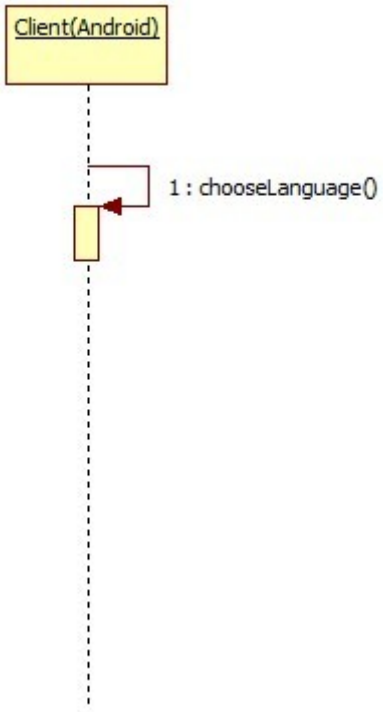



Figure 23: Choose Language Sequence Diagram

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

### 5.9.3 Choose Game Interaction

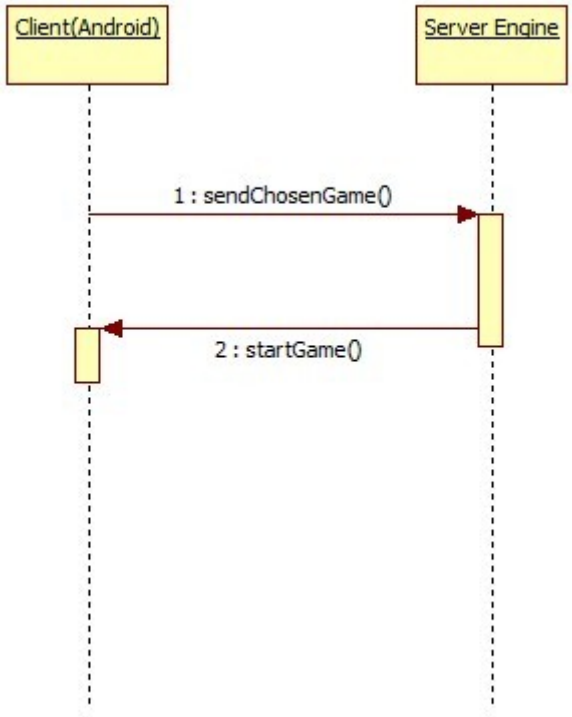



Figure 24: Choose Game Sequence Diagram

The diagram above explains Choose Game Interaction. When the user is to determine the game to be played, he/she clicks the 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

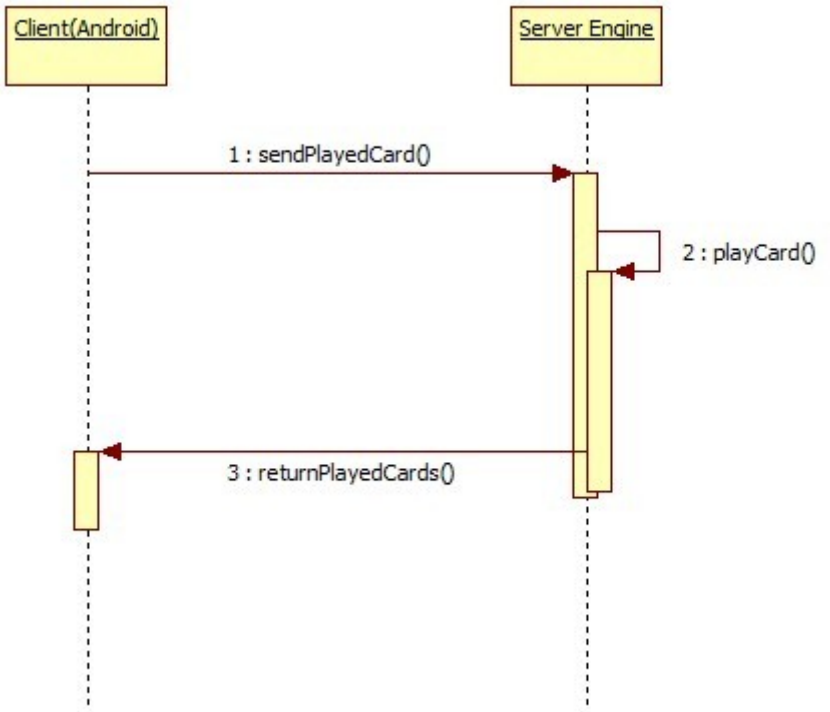



Figure 25: Play Card Sequence Diagram

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

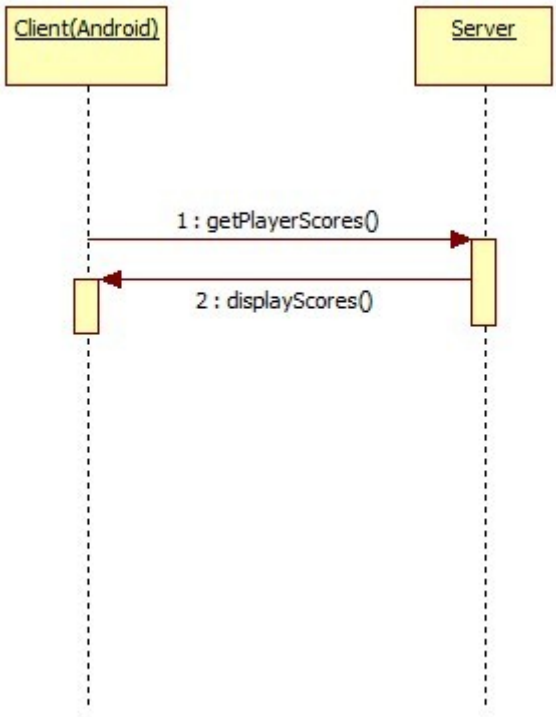



Figure 26: Display Score Sequence Diagram

This diagram above explains Display Score Table Interaction. When user wants to see the scores of the players, he/she clicks the button. After the 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

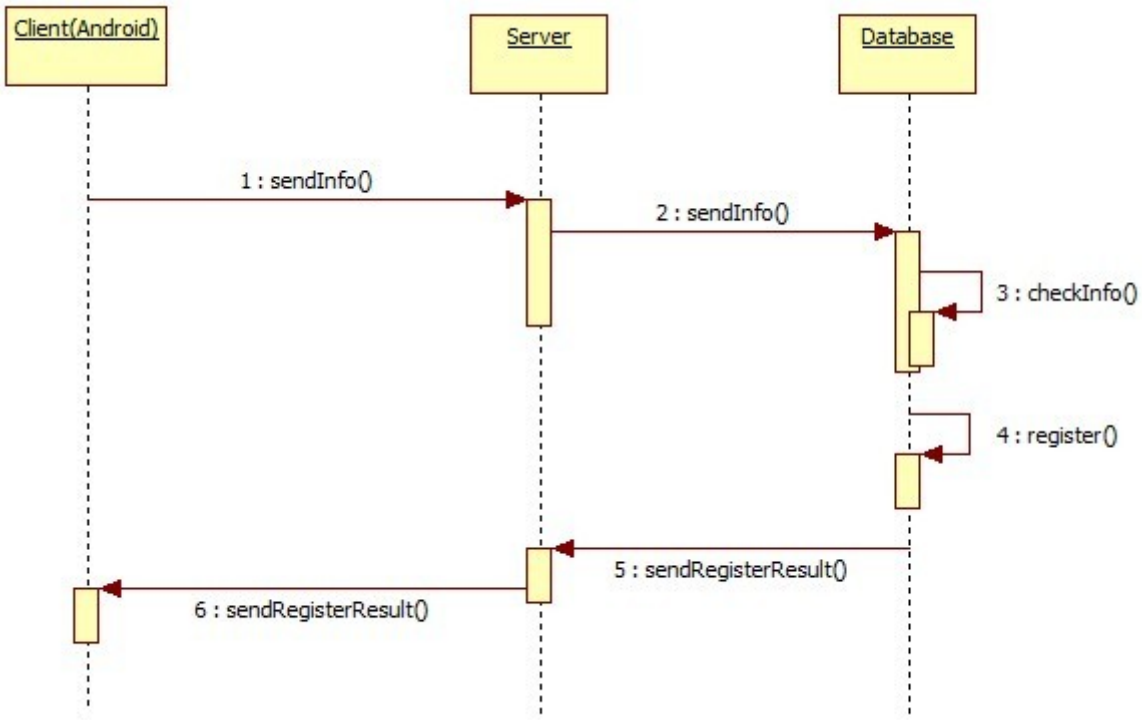



Figure 27: Register Sequence Diagram

This diagram above explains Register Interaction. When unregistered user fills all necessary fields in the Register Screen, clicks the button. After this click, client gets the data and sends it to the server. Afterwards, 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

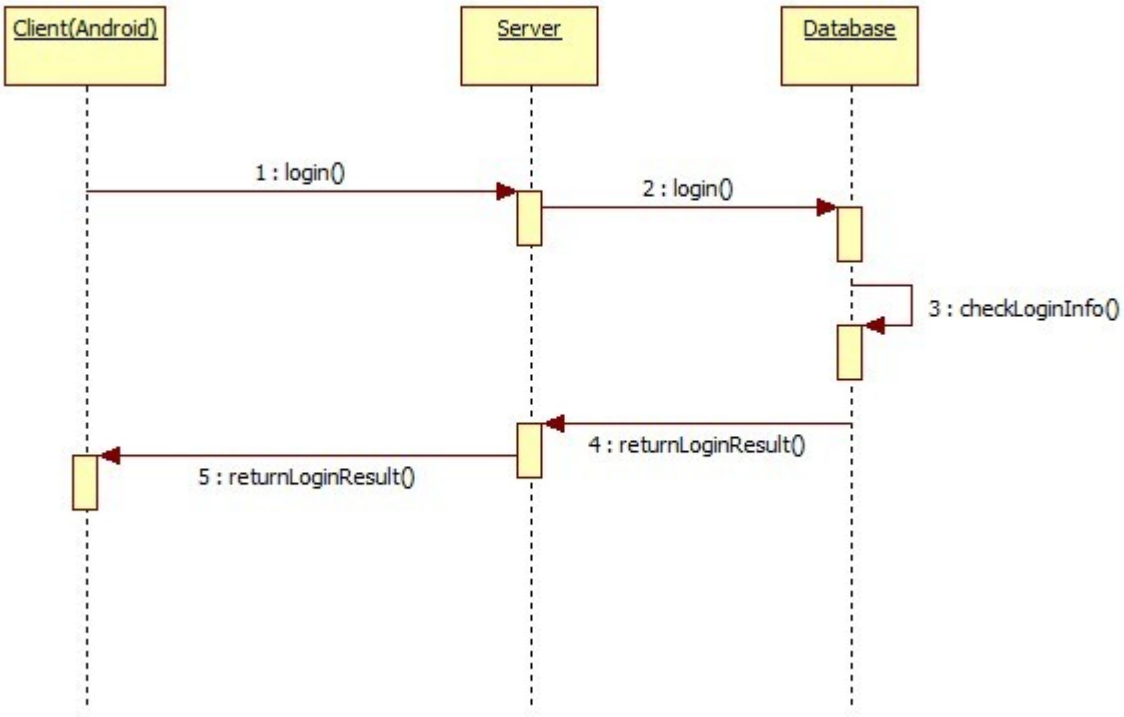



Figure 28: Login Sequence Diagram

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 server. After that, database checks the user information and sends the login result to the client.

### 5.9.8 Logout Interaction

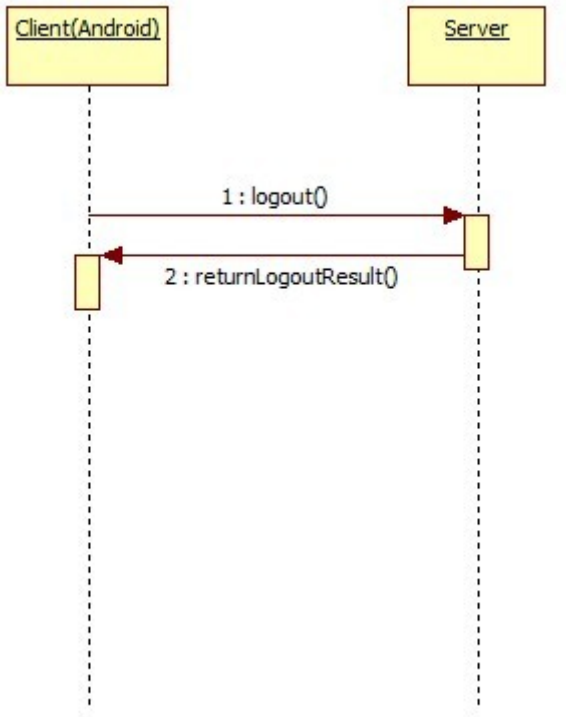



Figure 29: Logout Sequence Diagram

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.

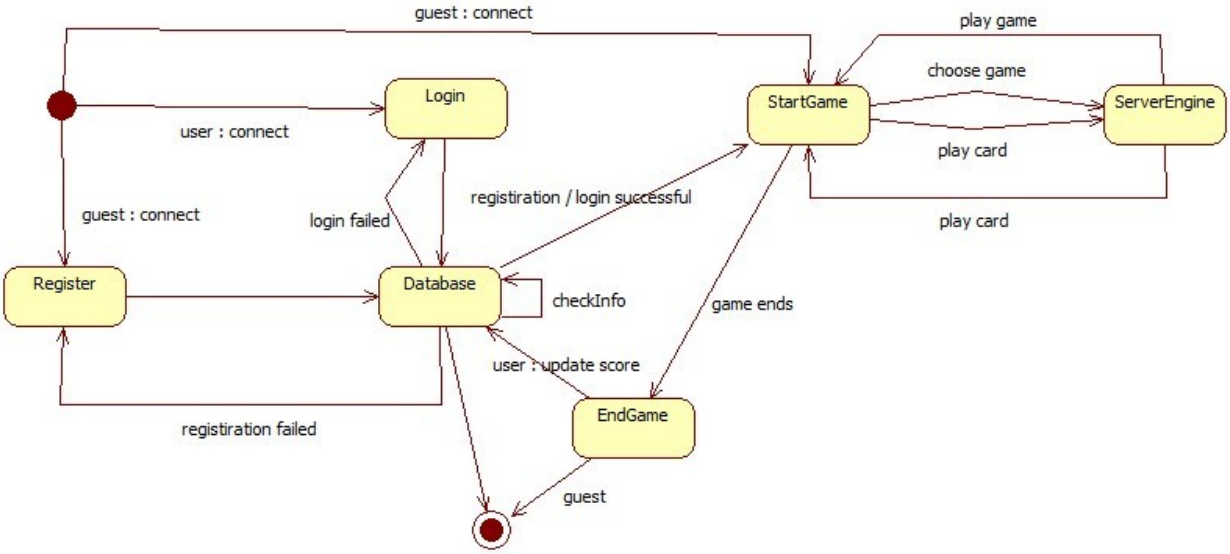



Figure 30: State Diagram

### 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