SOFTWARE REQUIREMENTS SPECIFICATION

30.11.2014

Master King PROJECT

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

Table of Contents

1. INTRODUCTION

1.1 Problem Definition

That is actually not a problem. We don't think that selected project is a problem. That will be a card game which is “King” to entertain the people. In this project, we will implement an artificial intelligence and that is the main purpose of this project. After publishing this project, people will not spend time to find a person and they will play the game with good quality playing agents.

1.2.Purpose

The aim of this document is to present a detailed description of a project having a general name which is “Building Game Playing Agent”. Under this idea, we will implement the card game which is King. This document includes purpose, features and the interfaces of the application and it is prepared according to the “IEEE 830-1998” template. External and internal interfaces are explained. We also provide ER diagram, Class diagram, Use Case diagram and State diagram.

This project is aimed to reach all people who are interested in playing King which has better artificial intelligence than the previous released King games.

1.3 Scope

This product will be published for mobile phone having Android Operating System. The name of the product will be “Master King”.

The product records the scores that the players gain. According to the scores, the most scored player will be the winner of the month. The scores will be stored in the database. The users will not be able to play with real players but they will play against the agents. After clicking the cards on the interface, agents will play according to the chosen card by player. The selected cards by agents and players will be reflected on the interface.

The main purpose of this project is to publish better artificial intelligence. This will make the game harder so that the game will be more enjoyable for the players. The user interface will be user-friendly.

Minimax Algorithm

Minimax decision from the current state is computed by the minimax algorithm which uses a simple recursive computation of minimax values which each successor states have. Firstly, this recursive algorithm looks at decision tree from top to bottom and repeat this process backwards and return minimum value of tree. Minimax algorithm uses a complete depth first traversal of the game tree. Assume that depth of our game tree is equal to m and there are n possible moves at the each point, then time complexity of the algorithm is O(n^m) so that the result has impractical complexity.

This algorithm considers all nodes of tree and this causes large complexity for our project. Instead of looking all nodes of the tree, we prefer looking specific nodes and paths. Although we have researched this algorithm, we decided not to use it in our implementation.

Alpha-Beta Pruning

While minimax algorithm looks at all possibility of min value, this algorithm tries to reduce the number of nodes that will be searched. Unfortunately, we cannot eliminate the exponential computational complexity, but we can effectively cut it in half. We can borrow the idea of basic pruning (allows to ignore portions of the search tree that make no difference to the final choice) to eliminate some parts of the search tree. This technique is known as alpha- beta pruning in AI. In other words, Alpha-Beta pruning algorithm prevents higher cost of searching all nodes of the decision tree. Searching of the tree stops when the found node is worse than the previously found nodes. This is a good improvement over the minimax algorithm increasing the efficiency of this algorithm. Specifically, this algorithm is O(n ^(m/2))

β stands for the the minimum upper bound of possible solutions and and α stands for maximum lower bound of possible solutions. Algorithm works only if the current estimate of the value of the node is between the β and α values.

RULE-BASED SYSTEMS

In our project we have used rule based system approach. Thus, it will be suitable to mention this concept and the modifications we have done to this approach. Also, in this context we will mention our system's integration to this concept.

The simplest form of artificial intelligence which is generally used in industry is the rule-based system , also known as the expert system. Its main idea is simple. In these systems we have applied to the knowledge of humans in a specific domain to capture pieces of information. After this stage, developers take place and they embody these pieces of information within a software system. The knowledge is stored as a set of rules. These rules are of the form IF-THEN statements. To Illustrate the structure take the example into account. A typical rule for an automobile diagnosis system that we have investigated might look something like this:

```
RULE : [Is the car out of petrol ?]
IF

    (petrol tank fuel level < %20 of full level)
    OR (the car is on the way for 4 hours)

THEN

    refuel the fuel tank
```

Semantics of these statements are similar to the programming language ones. But unlike an if-then-else statement, it stands alone and does not fire in any predetermined order relative to other if-then-else statements. Simply, these systems are composed of five components which are knowledge base, database, inference engine, explanation facilities and the user interface.

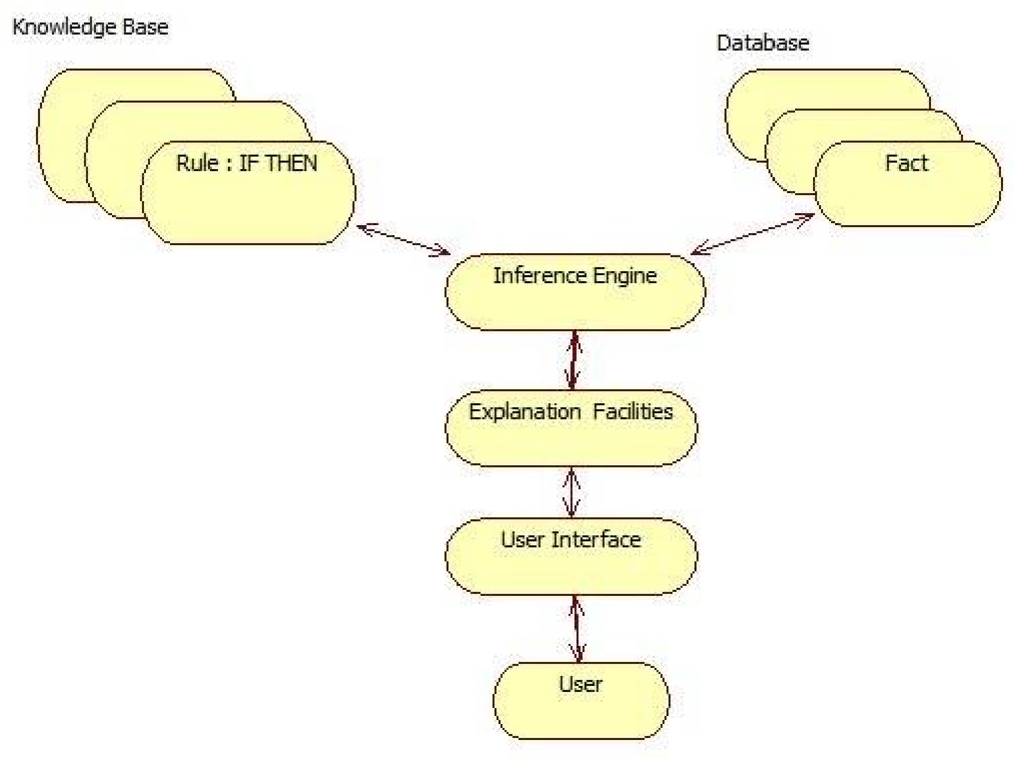


Figure 1: Architecture of a rule-based system

1- Knowledge Base

This part contains the rules embodying the knowledge of humans about a specific domain. Rules have priorities as expected. Each rule will be checked from higher to lower priority. They may consists of the facts and the rules recursively. Like in Prolog rules, if all of the requirements are satisfied (in our case facts and rules) its result will be true.

2- Database

This part of the system is responsible for storing the collection of the known facts which will be used by the rules. Inference engine uses this database to fetch some facts and match them against the rules. This is similar to what logic programming languages simply do.

3- Inference Engine

This part is responsible for carrying the reasoning process. It basically links the set of known facts and rules so as to find a solution to the current problem. There are two types of inference mechanism used in this engine which are forward and backward chaining. Forward chaining starts with the available data and uses rules to extract more data until a goal is reached. Backward chaining starts with set of goals and works backwards. The choice of these strategies are dependent on the design and the project itself.

4- Explanation Facilities

This part is responsible for explaining the user about the reasoning process of the system. By keeping track of the rules that are fired, an explanation facility presents a chain of reasoning that led to a certain conclusion. This feature makes a huge difference between expert systems and other conventional systems.

5- User Interface

This part is responsible for communication between the user and the system. The user interface of the design may be simple or sophisticated. Since there are various number of techniques to develop interfaces, it is easy to build user-friendly and simple interfaces.

OUR PROJECT INTEGRATION

After having these basic knowledge about rule-based systems, understanding the basic architecture of our project is straightforward which is shown on the second figure.

As known in the King Game there are two kinds of games which are positive and negative hands. We have seven games in total which are “No Tricks”, “No Hearts”, “No Queens”, “No Kings or Jacks”, “No King of Hearts”, “No Last 2 Tricks”, “Positive Hands”. For each type of the game we have different rules and strategies. These rules consist of the knowledge base component of our system. Since our project is in partial observability group of games, our knowledge of played cards increases dynamically. To play the game well, some pieces of information such as played cards array, type of cards that have already finished in a user's hand 2-d array, number of card type(e.g. diamonds) that have already played array should be stored. These inner storages consist of the database component of our system.

Inference engine contacts both with database and knowledge facts. Simply this is the actual AI that chooses the card which will be played in that specific trick. We are using the forward chaining strategy to evaluate the card result. Obviously, we can estimate the trick under consideration by our previous knowledge. For instance our engine will trace the arrays that we are storing in our database to eliminate the options that are unlikely to be played.

For the sake of simplicity we have skipped the explanation facilities. Note that by skipping this component we are assuming that all the players know at least basic rules of the game. Otherwise, for experienced users it will be boring to be informed for each trick.

The last component of our system is the user interface. We have mentioned all of the details in “User Requirements” part. For more information refer to section 3.1.

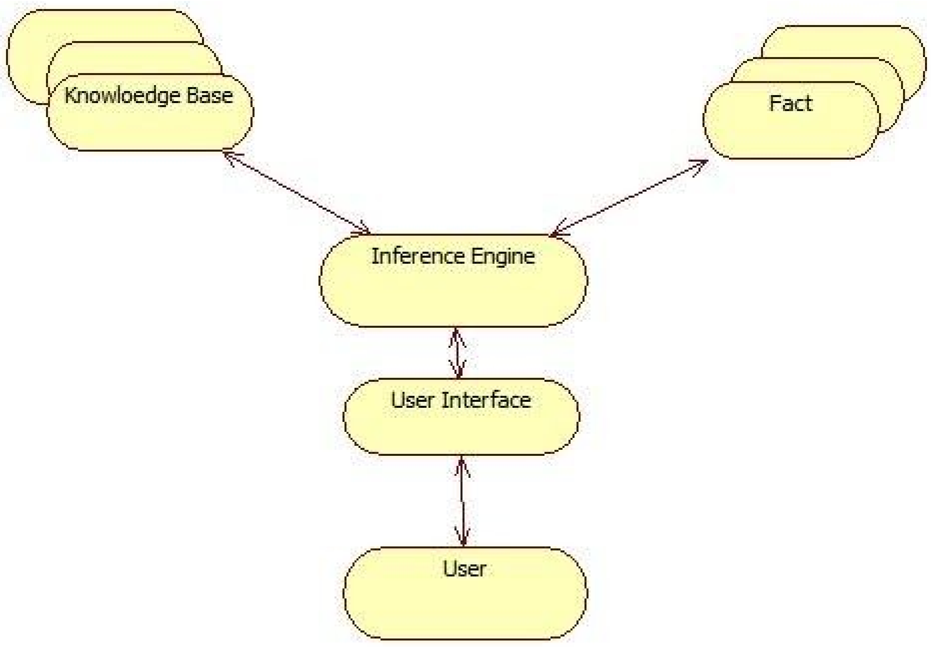


Figure 2: Architecture of our system

Since our system uses rule-based approach, we need to declare the rules explicitly. Because of several reasons we will embed all of the rules in an xml file by giving an index. After declaring the rules, if we want to add new ones, the place of the rules will be found easily using the indexes in the xml file. Also we want to have a very-well defined structure for all rules of the game. Since we are using object-oriented design and rule-based approach together, it will be convenient to have a project which satisfies the essential principles of software engineering such as maintainability.

Actually, there will be the piece of codes inside the xml tags for every rule. During the execution of the program,first of all we will parse the xml file and then build a tree which grows dynamically and uses the rules inside the xml file. For example the structure of the xml file will be as follows:

```
<0kupaçıkmış> “code will be placed here” </kupaçıkmış>
        <0.0kupasayısı> “code will be placed here” <0.0/kupasayısı>
                .
                .
                .
```

We will traverse the dynamically allocated tree in a depth-first manner.

1.4 Definitions, acronyms, and abbreviations

Term	**Definition or Abbreviation**
User	The person who is registered in system
Guest	The person who is not registered in system but can play the game
Admin	The person who controls database and system
SRS	Software Requirements Specification
UML	Unified Modeling Language
IEEE	Institute of Electrics and Electronics Engineers
Combo Box	An interface widget that allow users to pick a predefined value
DBMS	Database Management System
ER Diagram	Entity – Relationship Diagram
MySQL	An open source Database Management System
TCP/IP	A communication protocol for the Internet and similar networks
Bandwidth	The rate of data transfer measured in bits per second (bit/s)
JDBC	Java Database Connectivity

Table 1: Definitions, acronyms and abbreviations table

1.5. References

IEEE Standard Documents:

- IEEE Standard 830-1998, IEEE Recommended Practice for Software

For those who doesn't know how to play king refer to:

- http://en.wikipedia.org/wiki/King_%28card_game%29

- Artificial Intelligence: A Modern Approach (3rd Edition)

1.6. Overview

During the preparation of the this document, IEEE Std 830-1998 is followed.
External interfaces, functional requirements, use case diagrams, ER diagram and class diagrams are organized in the following sections.

2. Overall Description

2.1 Product perspective

This is system is not the part of any system. It is an independent mobile phone application.

There are a few number of King applications in the marketing area. The common issues of the previously published King applications are that they have playing agents that don't satisfy the users. Although users expect at least playing agents like them, agents only play with low level knowledge. Another problem is that user interfaces of the other King applications are not user friendly.

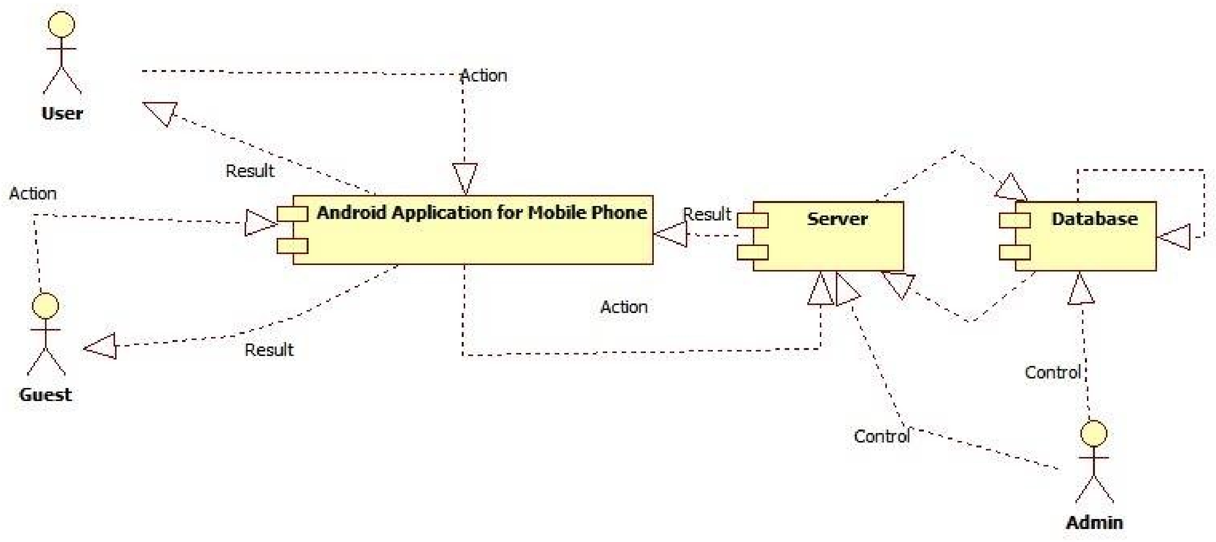


Figure 3: Block Diagram

Our project is a system that aims to entertain the people. It is a mobile phone application and it is server oriented. To be able to use this application, user needs an Internet connection to be able to take the responses from server and to download the application. Also, above block diagram Figure 3 shows interactions between user, guest, admin, user interface, server, databases.

On the server side, user sends actions like playing cards or choosing game type via interface and this actions are transmitted to server. Then, the server generates response which will be reflected to interface. These processes are repeated while user stays in the system.

Database server is used to store data about the game scores that user gained. Username, password and e-mail of the users are also stored. At the end of the each month, the total score of the users are sorted in descending order and winner of the that month is chosen.

Admin is responsible for the management of the server and database. He/she will control the system.

2.1.1 System interfaces

This system is an independent one and it is not related with any other system. So, there are no system interfaces.

2.1.2 User interfaces

Since our system is a mobile phone application, the interface of the system is not so complicated. It consists of green background, playing cards and score table. The players can only see their own cards. However, when players play their card, that card will move through the center of the interface and it will be visible to others.

When players launch the application, they will face with login screen. If they are already registered, they can enter the system using username and password. If they are not registered to system, they can either enter the system as guest without logging or they can register to system.

While playing the game, the user will be limited according the rules of the game. If player doesn't choose the valid card, system will not allow player to play that card. If user wants to see the score table, user can reach it using the D button on the main interface.

2.1.3 Hardware interfaces

The only hardware interface is the host platforms.

2.1.4 Software interfaces

The project will be deployed to server. It will run on all versions of the Android. It will be compatible with different size of the screens of mobile phones. There is also another software interface between server and database. Since we use Java to implement the project and we use MYSQL for the database, there is a connector which is JDBC to connect directly to MYSQL inside Java.

2.1.5 Communication interfaces

The server and DBMS communicate using the TCP/IP protocol. Also, mobile phone communicates with server using TCP/IP.

2.1.6 Memory

Since the project is a server oriented application, the application files will be stored on the server and main operations will be done on the server. Therefore; it will not occupy much space both on primary and secondary storage.

2.1.7 Operations

This part is already explained in section 2.1.2.

2.1.8 Site Adaptation Requirements

Since it is a mobile phone application, the only need for user is to have mobile phone with Android Operation System and Internet connection.

2.2 Product functions

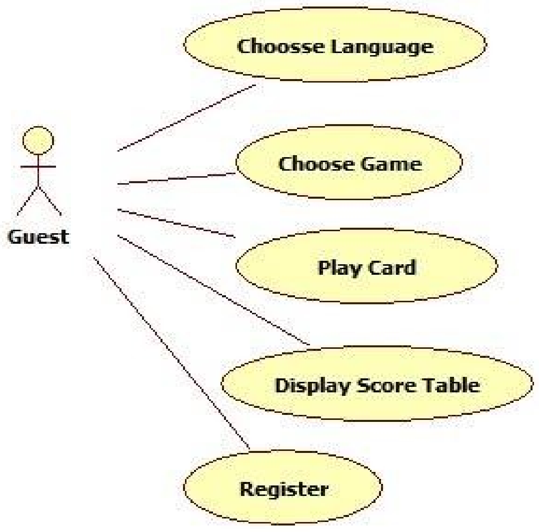


Figure 4: Use Case Diagram for Guest

No	Functionality	Short Description
1	Choose Language	Choosing the language of the system
2	Choose Game	Choosing the type of the game
3	Play Card	Playing the card
4	Display Score Table	Displaying the score table
5	Register	Registering the system

Table 2: Explanation for Guest Use Case Diagram

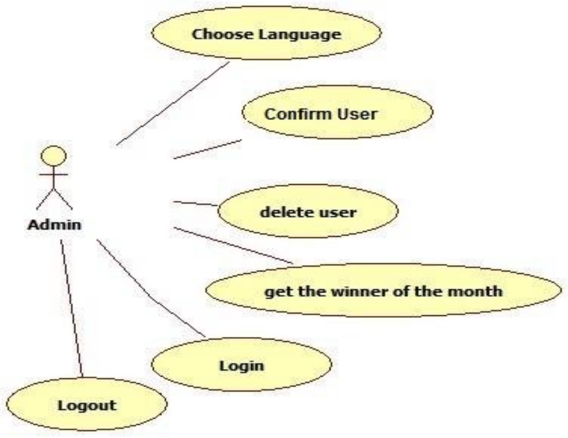


Figure 5: Use Case Diagram for Admin

No	Functionality	Short Description
1	Login	Logging in to system
2	Choose Language	Choosing the language of the system
3	Confirm User	Confirming the registered user
4	Delete User	Deleting the user from the system
5	Get the winner of the month	Getting the winner of the month
6	Logout	Logging out from the system

Table 3: Explanation for Admin Use Case Diagram

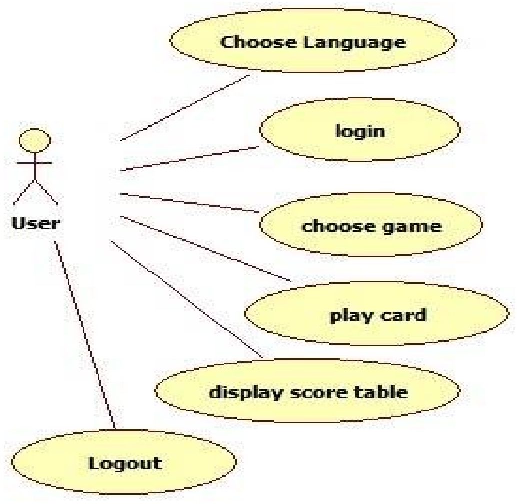


Figure 6: Use Case Diagram for User

No	Functionality	Short Description
1	Login	Logging in to system
2	Choose Language	Choosing the language of the system
2	Choose Game	Choosing the type of the game
3	Play Card	Playing the card
4	Display Score Table	Displaying the score table
6	Logout	Logging out from the system

Table 4: Explanation for User Use Case Diagram

The detailed explanation for the use case diagrams is given in section 3.2.

2.3 Constraints

Constraint 1

The username should consist of at most 15 characters.

Constraint 2

The username should be unique among the usernames in the database.

Constraint 3

The password should consist of at most 15 characters.

Constraint 4

The user should fill the required fields during the registration.

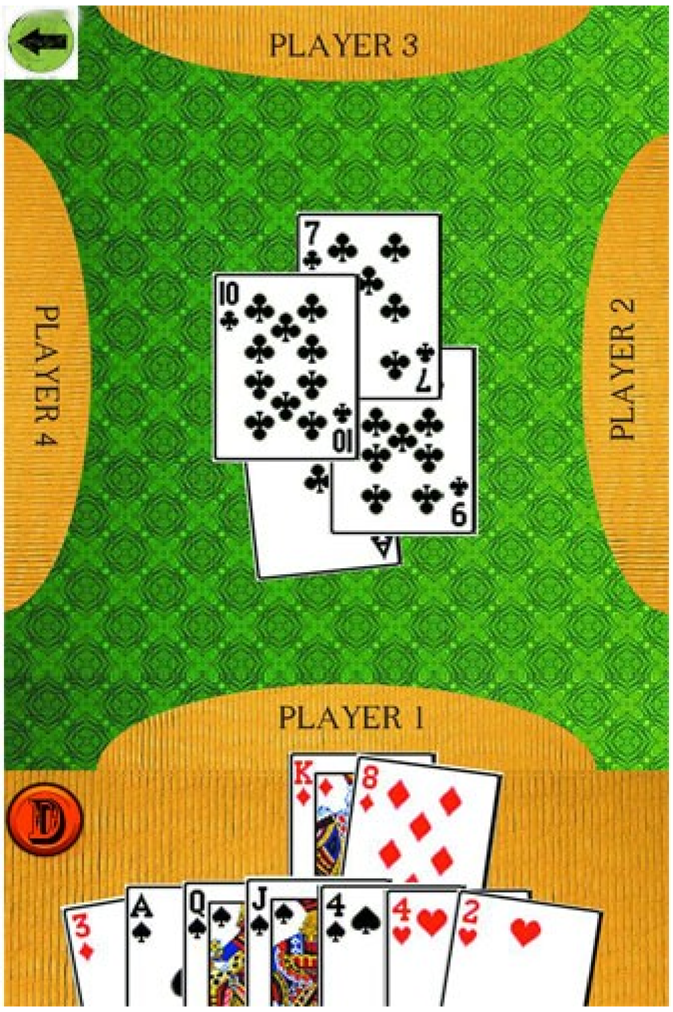
2.4 Assumptions and dependencies

There is only one assumption that user is able to play King game. Our system will run on Android Operating System. So, user interface and server communication will be dependent on Android Operating System.

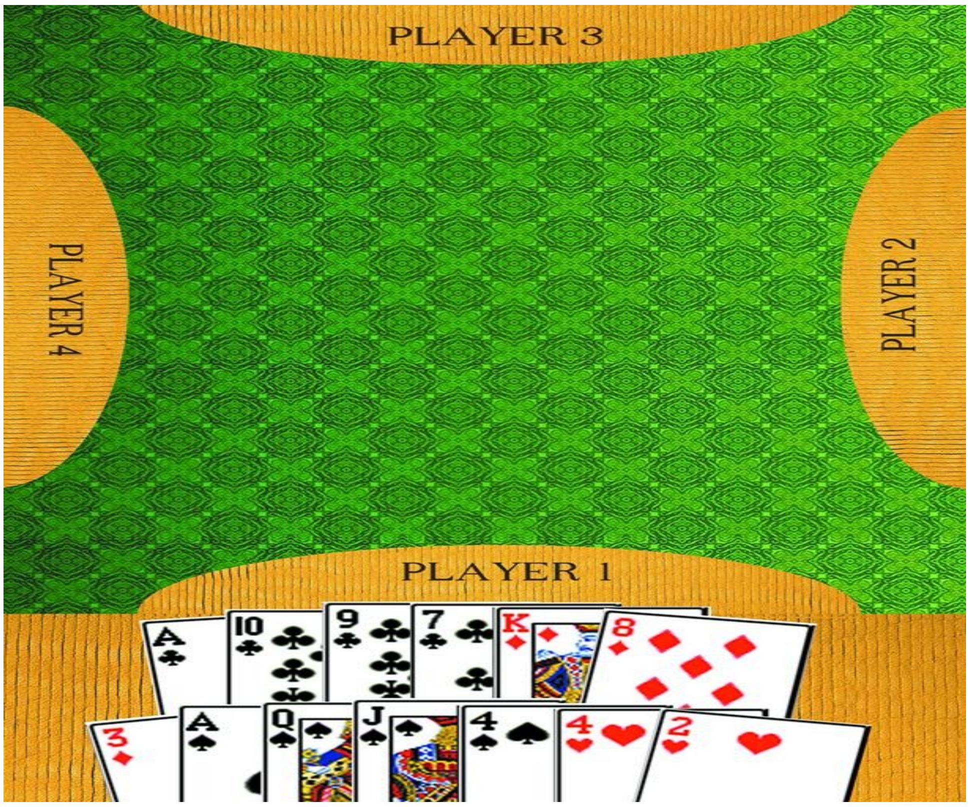
3. Specific requirements

The all types of requirements will be mentioned in this section.

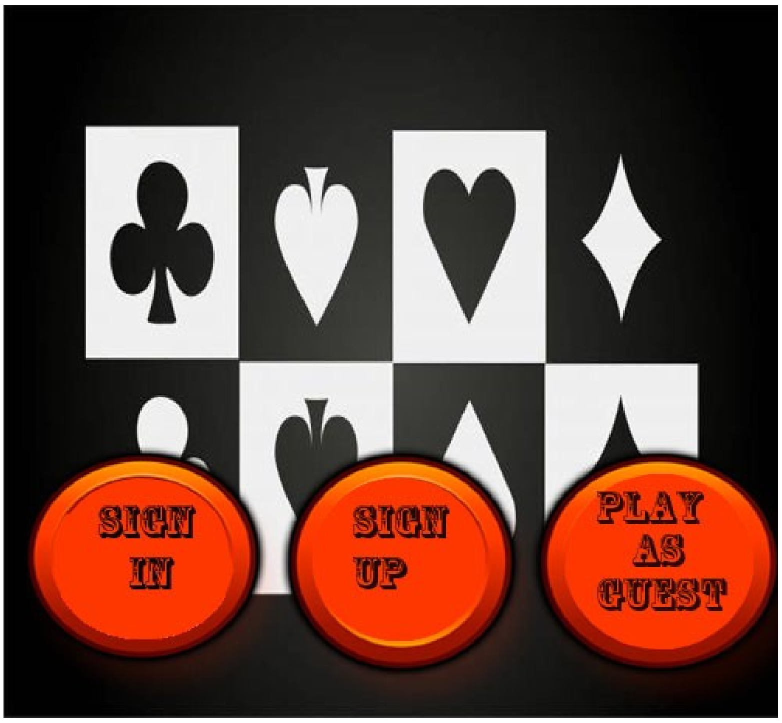
3.1 Interface Requirements



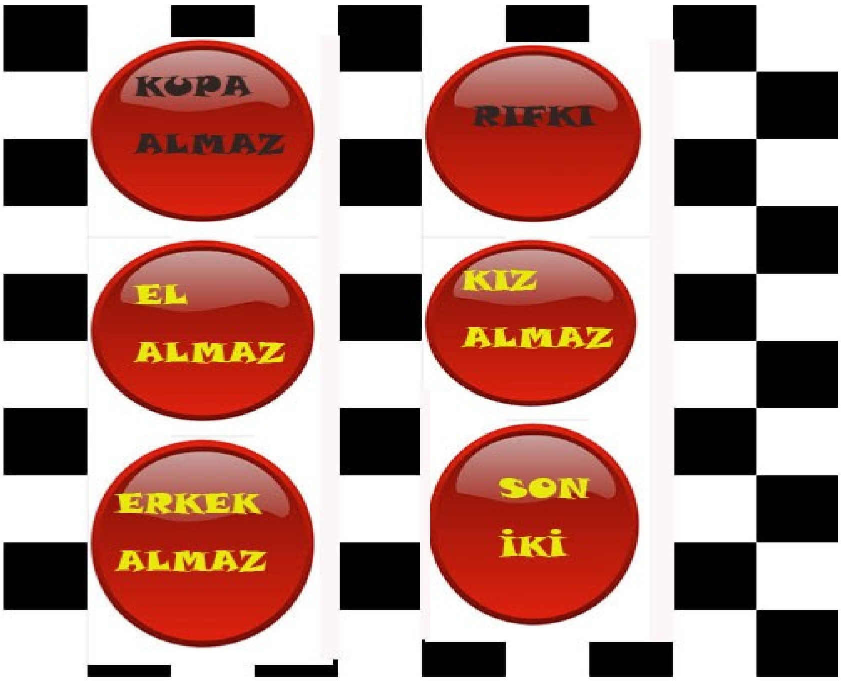
Interface 1: Main Playing Screen



Interface 2: Main Playing Screen 2



Interface 3: Login Screen



Interface 4: Choose Game Screen

User interface should be provided through mobile phone application. In main page, there exists sign up, sign in and play as a guest buttons.

- If user clicks sign up button, user register page should open. There exists username, password, e-mail fields. If username or e-mail already exists in database, system should give an error message. Then, user should try again new username or e-mail. If username and password fields are filled correctly, game page should open.
- If user clicks play as a guest button, game page should open.
- If user clicks sign in button, he/she should fill username and password fields correctly. If username or password is wrong, system should give relevant error message.
- In game page, there exists four seats that user or guest should choose. After this choice, game starts and thirteen cards should be appeared in front of user.
- During the game, if user clicks card that violates rules of king, this card shouldn't move.
- During the game, user can see score table by clicking D button.
- During the game, user can exit by clicking back button.
- During the game, color of finished game type, which was played two times, is gray and color of non-finished game type is yellow in game choosing screen.

3.2 Functional Requirements

3.2.1 'Choose Language' Use Case for Guest/User

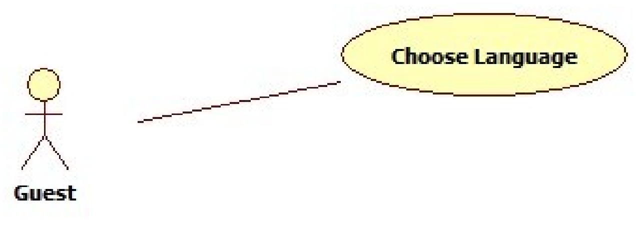


Figure 7: Choose Language Use Case Diagram for Guest

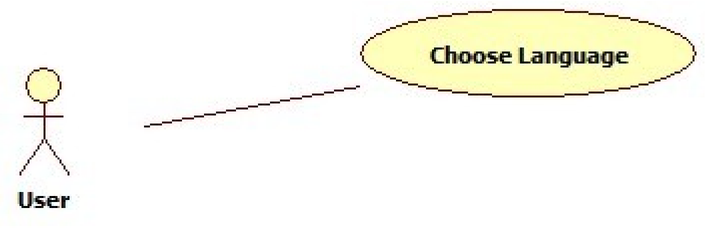


Figure 8: Choose Language Use Case Diagram for User

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. 2- The unregistered user/ user clicks the icon of the language that he/she wants to choose.
Exception Flow	-
Post Conditions	-

Table 5: Explanation for Choose Language Use Case Diagram

3.2.1.1. Functional Requirement 1.1

Language menu options shall be Turkish and English.

3.2.1.2. Functional Requirement 1.2

Language menu shall be in the upper right part of the main page.

3.2.1.3. Functional Requirement 1.3

Default language of the application shall be Turkish.

3.2.1.4. Functional Requirement 1.4

Language button shall be “Combo Box” button.

3.2.2 'Choose Game' Use Case for Guest/User

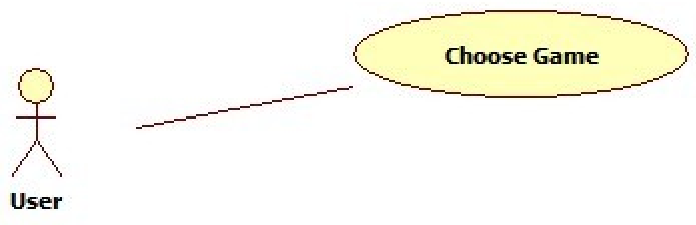


Figure 9: Choose Game Use Case Diagram for User

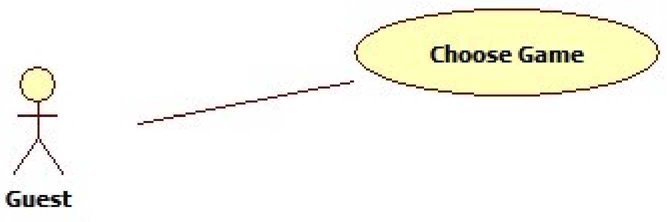


Figure 10: Choose Game Use Case Diagram for Guest

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. 2- The unregistered user / user clicks the icon of the language that he/she wants to choose. 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 6: Explanation for Choose Game Use Case Diagram

3.2.2.1. Functional Requirement 2.1

The player shall not choose the Auction game in the first round.

3.2.2.2. Functional Requirement 2.2

One negative hands game shall not be chosen more than twice.

3.2.2.3. Functional Requirement 2.3

The order for choosing game shall be in counter clockwise direction.

3.2.3 'Play Card' Use Case for Guest/User

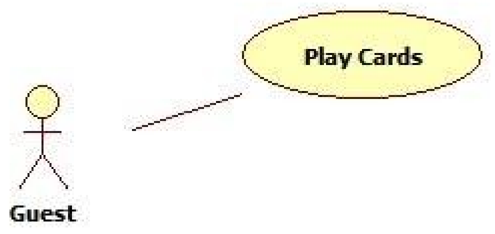


Figure 11: Play Card Use Case Diagram for Guest

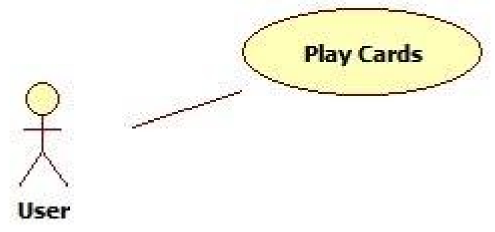


Figure 12: Play Card Use Case Diagram for User

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. 2- The unregistered user / user clicks the icon of the language that he/she wants to choose. 3- If it is guest's / user's turn, the guest clicks the card he/she wants to play.
Exception Flow	-
Post Conditions	-

Table 7: Explanation for Play Card Use Case Diagram

3.2.3.1. Functional Requirement 3.1

The player shall not play the card that violates rules of King.

3.2.3.2. Functional Requirement 3.2

In order to play the card, player shall wait until his/her turn comes.

3.2.3.3. Functional Requirement 3.3

Each player shall choose three negative hands and two positive hands.

3.2.4 'Display Score Table' Use Case for Guest/User

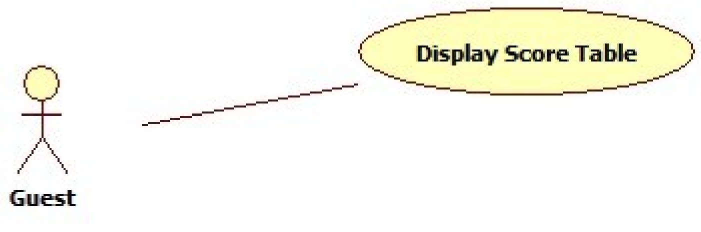


Figure 13: Display Score Table Use Case Diagram for Guest

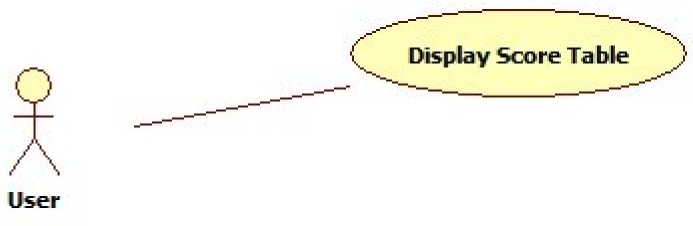


Figure 14: Display Score Table Use Case Diagram for User

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. 2- The unregistered user / user clicks the icon of the language that he/she wants to choose. 3- The unregistered user / user clicks the icon in order to see the current score situation.
Exception Flow	-
Post Conditions	-

Table 8: Explanation for Display Score Table Use Case Diagram

3.2.4.1. Functional Requirement 4.1

The system shall display total scores and individual score of each game.

3.2.4.2. Functional Requirement 4.2

The score table shall open in a new window.

3.2.4.3. Functional Requirement 4.3

The score table shall be displayed until user clicks anywhere of the screen.

3.2.5 'Register' Use Case for Guest

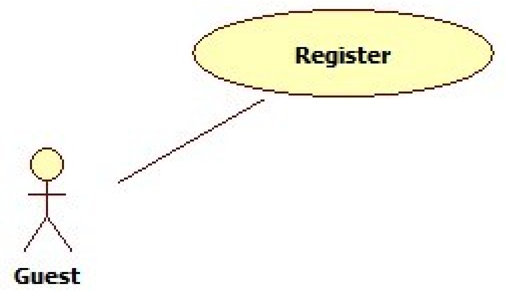


Figure 15: Register Use Case Diagram for Guest

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. 2- The unregistered user clicks the registration icon. 3- The unregistered user enters his/her user name, password and mail address. 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. -If unregistered user cannot be inserted due to system problems, an error message will be shown.
Post Conditions	-

Table 9: Explanation for Display Score Table Use Case Diagram

3.2.5.1. Functional Requirement 5.1

The user shall fill the required fields in registration form.

3.2.5.2. Functional Requirement 5.2

The system shall insert the data into the database.

3.2.5.3. Functional Requirement 5.3

The user shall click register button to finish the registration process.

3.2.6 'Login' Use Case for User

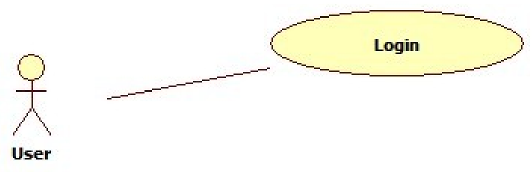


Figure 16: Register Use Case Diagram for User

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. 2- The registered user clicks the icon of the language that he/she wants to choose. 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 10: Explanation for Login Table Use Case Diagram

3.2.6.1. Functional Requirement 6.1

The system shall give an error message, when user enters wrong username or password.

3.2.6.2. Functional Requirement 6.2

The system shall make comparison between the entered values and database values.

3.2.6.3. Functional Requirement 6.3

The user shall click the Login button to enter the system.

3.2.7 'Logout' Use Case for User

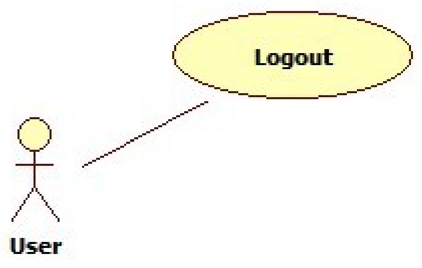


Figure 17: Logout Use Case Diagram for User

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. 2- The registered user clicks the icon of the language that he/she wants to choose. 3- The registered user enter his/her user name and password. Then, clicks the login button to proceed. 4- The register user clicks the logout button.
Exception Flow	-
Post Conditions	-

Table 11: Explanation for Logout Table Use Case Diagram

3.2.7.1. Functional Requirement 7.1

The user shall click the logout button to be able to exit the system.

3.2.8 'Confirm User' Use Case for Admin

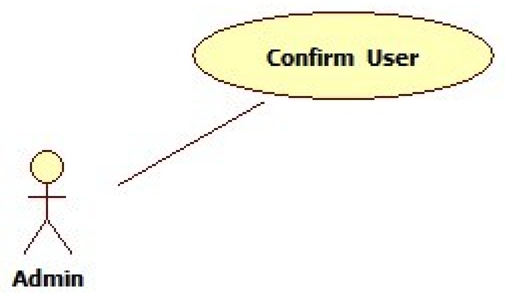


Figure 18: Confirm User Use Case Diagram for Admin

Use Case ID	UC8
Use Case Name	Confirm User
Description	This use case describes the event in which an admin confirms a user registration that is created before.
Actors	Admin
Preconditions	-
Trigger	The admin clicks the confirm button.
Basic Flow	1- The admin opens the admin panel. 2- The admin enter his/her user name and password. Then, clicks the login button to proceed. 3- If there are suspended registrations, admin clicks the confirm button and accepts the registrations.
Exception Flow	-
Post Conditions	-

Table 12: Explanation for Confirm User Table Use Case Diagram

3.2.8.1. Functional Requirement 8.1

The new registered users shall be sent to the admin panel.

3.2.8.2. Functional Requirement 8.2

The user shall not be able to enter the system, if registered account is not confirmed.

3.2.9 'Delete User' Use Case for Admin

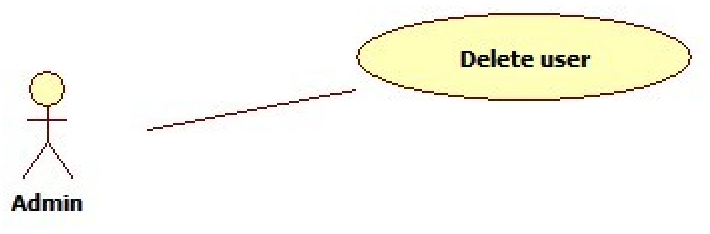


Figure 19: Delete User Use Case Diagram for Admin

Use Case ID	UC9
Use Case Name	Delete User
Description	This use case describes the event in which an admin deletes a user account that is created before.
Actors	Admin
Preconditions	-
Trigger	The admin clicks the delete button.
Basic Flow	1- The admin opens the admin panel. 2- The admin enter his/her user name and password. Then, clicks the login button to proceed. 3- The admin can delete a user account by clicking the delete button.
Exception Flow	-
Post Conditions	-

Table 13: Explanation for Delete User Table Use Case Diagram

3.2.9.1. Functional Requirement 9.1

The admin shall see the all registered users in the admin panel.

3.2.9.2. Functional Requirement 9.2

The registered user shall be deleted from database, when admin clicks the delete button.

3.2.10 'Get the Winner of the Month' Use Case for Admin

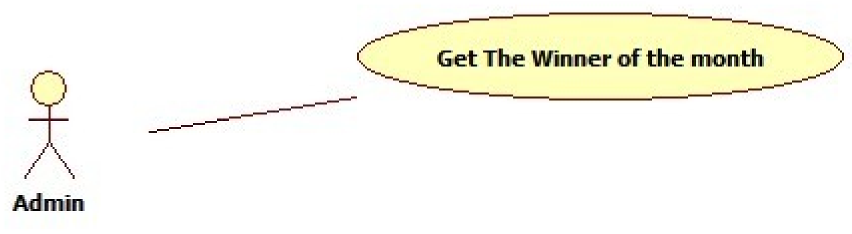


Figure 20: Get the Winner of the Month Use Case Diagram for Admin

XRef	
Use Case ID	UC10
Use Case Name	Get the Winner of the Month
Description	This use case describes the event in which an admin fetches and declares user who has achieved maximum score of that month.
Actors	Admin
Preconditions	-
Trigger	The admin clicks the winner button.
Basic Flow	1- The admin opens the admin panel. 2- The admin enter his/her user name and password. Then, clicks the login button to proceed. 3- The admin clicks the winner button and declares the winning user information of the month.
Exception Flow	-
Post Conditions	-

Table 14: Explanation for Get the Winner of the Month Table Use Case Diagram

3.2.10.1. Functional Requirement 10.1

The system shall display the most scored user.

3.2.10.2. Functional Requirement 10.2

The system shall initialize the scores of the users at the beginning of the month.

3.3 Non-functional Requirements

3.3.1 Performance Requirements

- Server Machine Bandwidth should meet at least 1.000.000 players that play simultaneously.
- Server Machine shall have low response time so that players can play simultaneously without any delay.
- During the game, card actions' time shall be at most 2 seconds.
- All of the functions described in Section 3.2 shall be completed within 3 seconds.

3.3.2 Design Constrains

3.3.2.1 Security

- The system should automatically logout when user exits the system.
- Access request of product shall be minimum.(e.g. product shall not violate privacy.)
- Password characters shall be displayed as black dots in login screen.
- If password is entered wrongly more than 5 times, an informative email shall be sent to the user.
- Password shall be at least 6 characters.
- Email addresses of the users shall be invisible to other users.

3.3.2.2 Portability

- Since it is server oriented product, the system will run on any platform that has Android Operating System.

3.3.2.3 Reliability

- If any component of the system does not response to user, the system shall display informative message about the error.
- During the game, each individual card shall be played only once.
- There shall be a backup system in order not to lose information of users of the system.
- The system shall play in accordance with its AI, that is, no cheating of rules is involved.

3.3.2.4 Availability

- The system shall be available 24 hours in a day unless there is an unexpected problem.

3.3.2.5 Maintainability

- System shall be implemented in such a way that it is not a problem to add a new system rule.

3.4 Logical Database Requirements

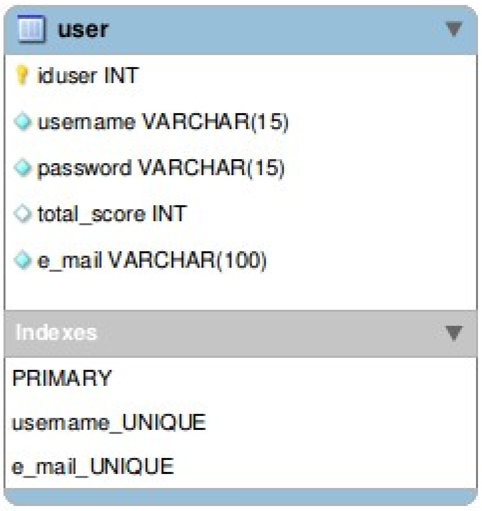


Figure 21: ER Diagram

The database consists of only one table which is named as user. It stores the required data to manage the registration of a player.

4 Data Model and Description

4.1 Data Description

4.1.1 Data Objects

Below class diagrams describes the system objects from an analysis perspective. It is drawn according to the UML standards.

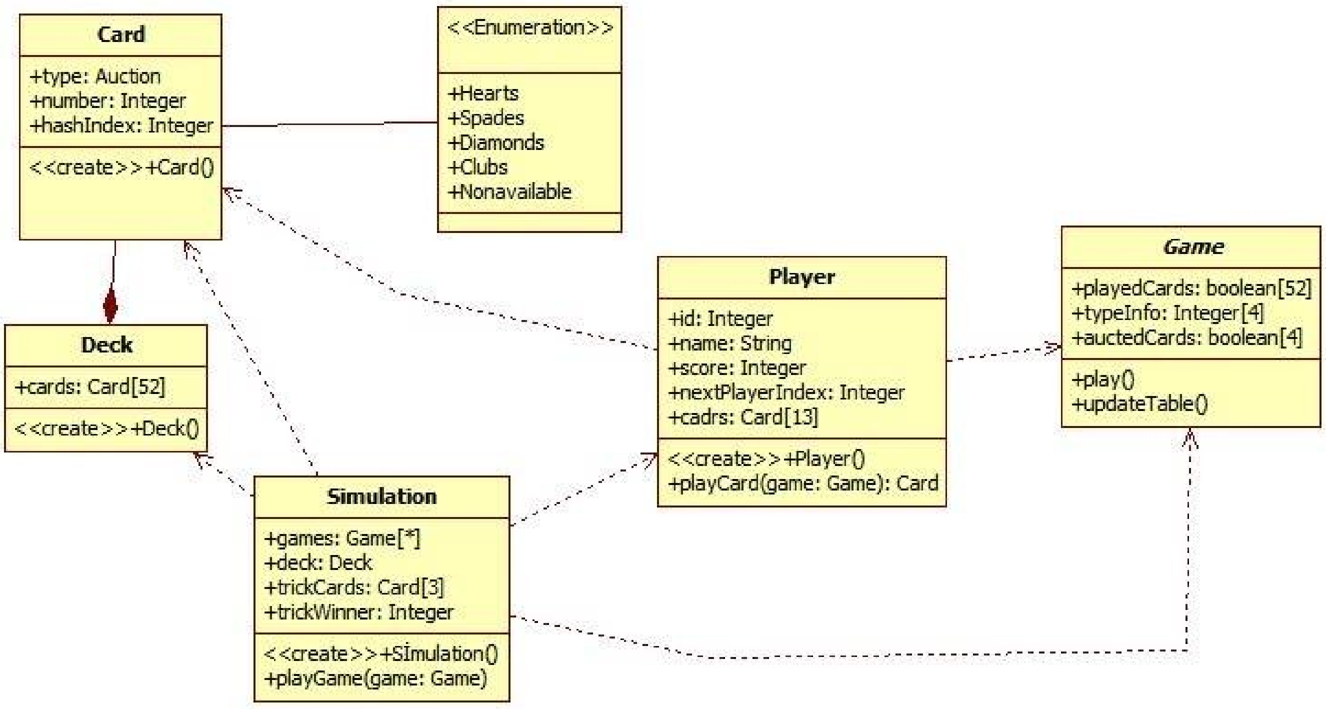


Figure 22: Analysis level Class Diagram- Part1

4.1.1.1 Card Object

This object represents just a single card in our King game. Each card has a type and number. As known there are 4 types of cards which are Diamonds, Spade, Clubs and Hearts. If the card is played, it is marked as Non-available. Public attribute hashIndex is used for indexing all of the cards. Instead of traversing all of the cards, it provides direct access to the current card.

4.1.1.2 Deck Object

This object represents the deck of cards. It composes of 52 cards as usual. By instantiating this class we create a card array whose size is 52.

4.1.1.3 Simulation Object

This is the class which simulates the game. It is dependent on the card, deck, player and game classes. Public attribute trickCards is responsible for keeping track of the current trick. Public attribute game array will store all 20 games of the King. Public attribute trickWinner will determine winner player id. PlayGame function will manage each trick accordingly.

4.1.1.4 Player Object

This object represents the players of the game. King game has always 4 players. Players have unique id, name, score , next player and array of cards whose size is 13. Its playGame function will determine and return a single card which is in players' cards array. The attribute nextPlayerIndex will be updated after each turn. The attribute score records players score.

4.1.1.5 Game Object

This is the abstract class which is the connector between simulation and types of games classes. (No Tricks, No Hearts, No Queens, No Kings or Jacks, No King of Hearts, No Last 2 Tricks, Auction.) The public attribute playedCards array represents played cards, the attribute typeInfo array represents the number of each card type that have already played, the attribute auctedCards 2-d array represents the type of cards that have already finished in a user's hand.

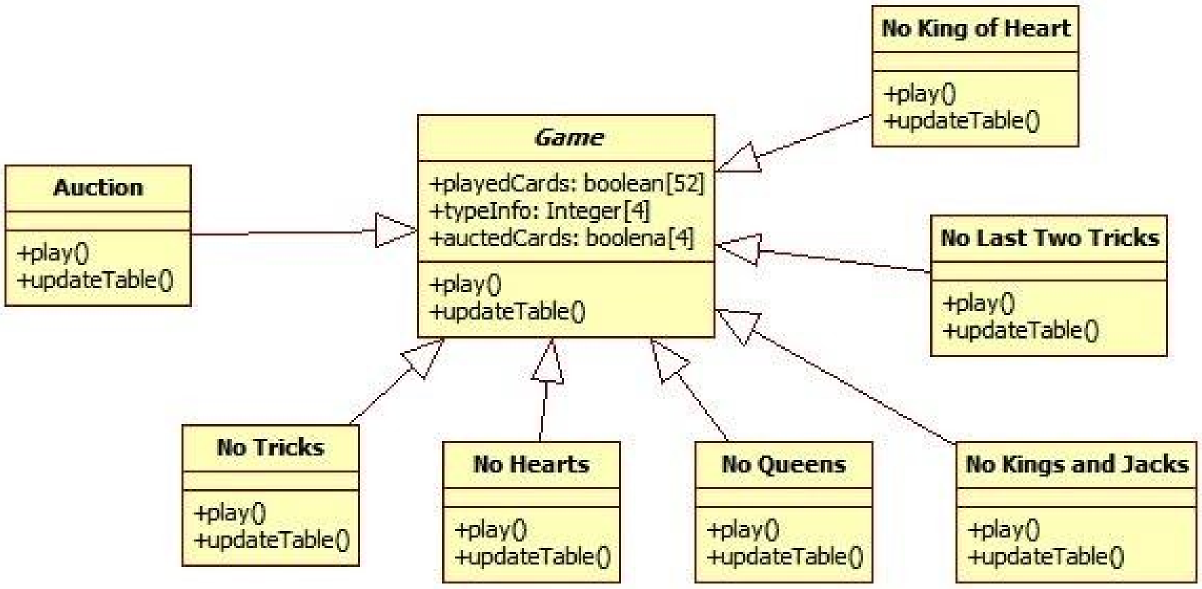


Figure 23: Analysis level Class Diagram- Part2

4.1.1.5 Auction, No Tricks, No Hearts, No Queens, No King of Hearts, No Last Two Tricks, No Kings or Jacks Objects

Each of these classes have overridden the play method of the abstract Game class accordingly. The public method updateTable will update the scores of player according to its own type.

4.1.2 Data dictionary

We have provided an Url for game rules in section 1.5 . Refer to that part.

5. Behavioral Model and Description

5.1 Description for software behavior

There are three actors for the system which are guest, user and admin.

Admin is responsible for database actions such as confirming and deleting user accounts and declaring most scored user of the month.

Guest is an unregistered user. He/she can directly enter the system and play the game without making any score.

User is a registered actor in the system. Firstly, he/she should sign in the system. If sign in process fails, he/she should try again. After sign in is successful, he/she will be directed to the game screen and start playing. Time of playing is up to user. Before the user exits from the system, score of the user is updated and the user will log out.

5.2 State Transition Diagram

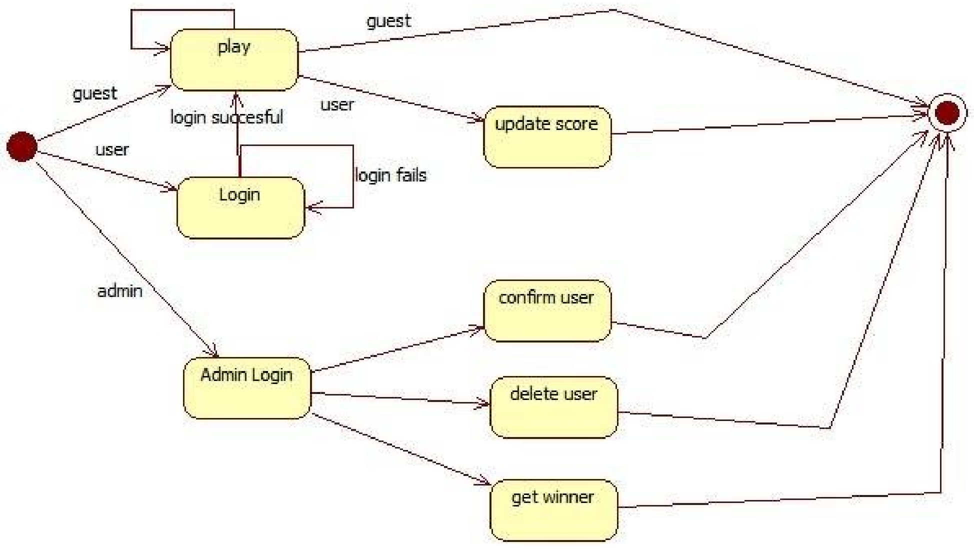


Figure 24: State Transition Diagram

6. Planning

6.1 Team Structure

All of the team members is able to work both individually and as a group. Before starting any part of the project, we decide what and how to do it together by meeting 2 – 3 times in a week. Later, we divide the work into small parts and each member works individually in his own part. After finishing the corresponding part, we repeat this process.

6.2 Estimation (Basic Schedule)

At the end of this semester, we are planning to complete positive hand and two negative hands. Also, we will show these games in prototype demonstration using user interface.

At the end of the academic year, we will complete the whole game and deployed it on a server. Database will be created to save users and people will start playing the game. Also, much more sophisticated interface will be developed.

6.3 Process Model

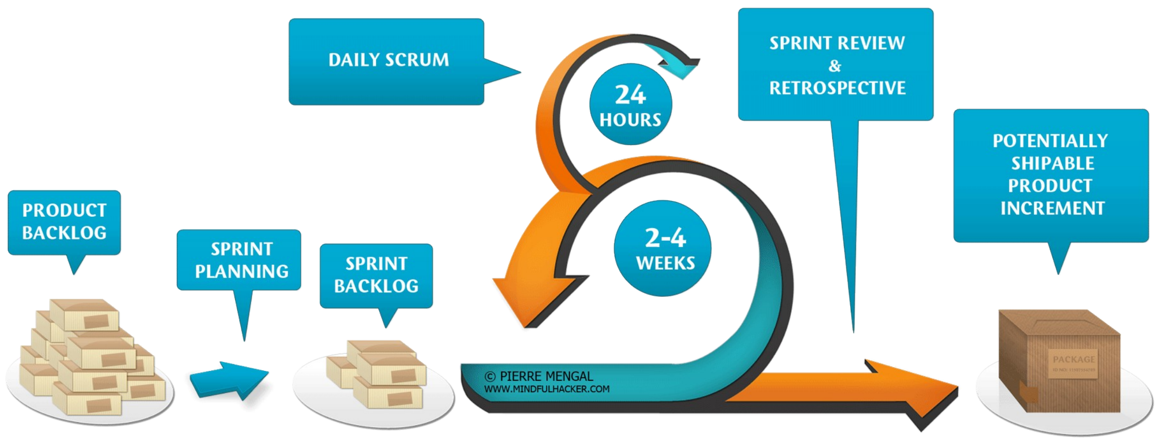


Figure 25: Process Model

In our project, we are following the below model which is Scrum Methodology.

7. Appendixes

7.1 Table of Figures

7.2 Table of Tables

7.3 Table of Interfaces