

CENG 491

Software Requirements Specification Report

Frog on Fire

Onur Dilek

Ahmet Furkan Üstün

Erkan Yeşilbaş

Table of Contents

1. Introduction

This document contains software requirements for the prospective “blöflü pişti” project of our team “Frog on Fire”. The document indicates descriptions of functions and specifications of the project with its all aspects and possible functionalities that is gradually to be developed. The approach used in this specification is adapted from IEEE recommended practices and also it abides the standards presented. We, “Frog on Fire”, assume full responsibility of the requirements presented in this document. Our team is composed of by three members; Onur Dilek, Erkan Yeşilbaş and Ahmet Furkan Üstün. We all are willing for the aforementioned project that will be available in mobile game market and hope the majority of pişti addicts will enjoy the game.

We will have external meetings with our teaching assistant Mine Yoldaş and Ali Tamur; from our project sponsor Usta Yazılım. The topic of the meetings with the student assistant is mostly about the planning of project. The content of the meetings with our sponsor is the details of the design and development process.

1.1. Problem Definition

In spite of a deep probe of online game market in order to be able to get sight of an exciting version among famous card games, there is not any extraordinary version of such a classical game, pişti. The possible demand for “blöflü pişti” leads us to develop it for mobile platforms.

Blöflü Pişti will be available and in use for Iphone game market. Beacause of the fact that the game will use a learning artificial intelligence system, it is going to play better in the process of gaining experience by practising.

1.2. Purpose

The purpose of this report is specification of requirements and preparation of the fundamentals and basic framework for project before getting into its subsequent design issues. Audience for this report is our sponsor company, Usta Yazılım and teaching staff of the course Computer Engineering Design. The requirements suggested in this document will serve as a guideline throughout the development process of this project. As a final step, the end-product will be tested against the requirements to ensure the quality of the software produced.

1.3. Scope

This document includes the requirements of our project, namely "blöflü pişti".Also, this document addresses the functionality, external interfaces,attributes,design constraints of the project to be developed.

1.4. User and Literature Survey

Although there are some open-source card game projects available, our project will be the first in this area. Because we will develop for iPhone and there is no available "blöflü pişti" project in this platform. This also can be understood that there is no English response for "blöflü pişti". In iPhone platform there is a "pişti" game, but our project will differ from existing one with AI agent and the main logic of the game

since our game has some different rules with its bluff part.
Our market researches show that card games have still their popularity since online playing option never uses its attraction.Actually,players do not want extreme functionalities from the games since their main need is just killing time with other users. However, game should offer not only main functionalities but also some features, so that it shall not bore the players.

1.5. Definitions and Abbreviations

AI: Artificial Intelligence
Http: Hypertext Transfer Protocol
GUI: Graphical User Interface
IEEE: Institute of Electrical and Electronics Engineering
UML: Unified Modelling Language

1.6. References

IEEE, IEEE STD 830-1998 IEEE Recommended Practice for Software Requirements Specifications. IEEE Computer Society.

CSS.06 - Yazılım Gereksinimleri Belirtimi Standardı. Rev.7.0

http://toucharcade.com/category/games/card/

http://itunes.apple.com/us/app/pisti-ii/id424594136?mt=8

1.7. Overview

The document consists of seven distinct chapters. First section introduces the project “blöflü pişti”. In the second chapter the game is explained with a high-level perspective. The details of the project is started to be given in chapter 3. This section explains the software requirements to a degree that enables designers to design this system, and testers to test it. Domain for the software and behavioral model is described in chapter 4 and 5, respectively. Finally, planning of the project is presented and the report is concluded.

2. Overall Description

2.1. Product Perspective

"Blöflü Pişti" is web-based software which can be used by the registered users and administrators. In order to use the system, users must login with entering their username and passwords which they got when they signed up to system.This user information held by database.

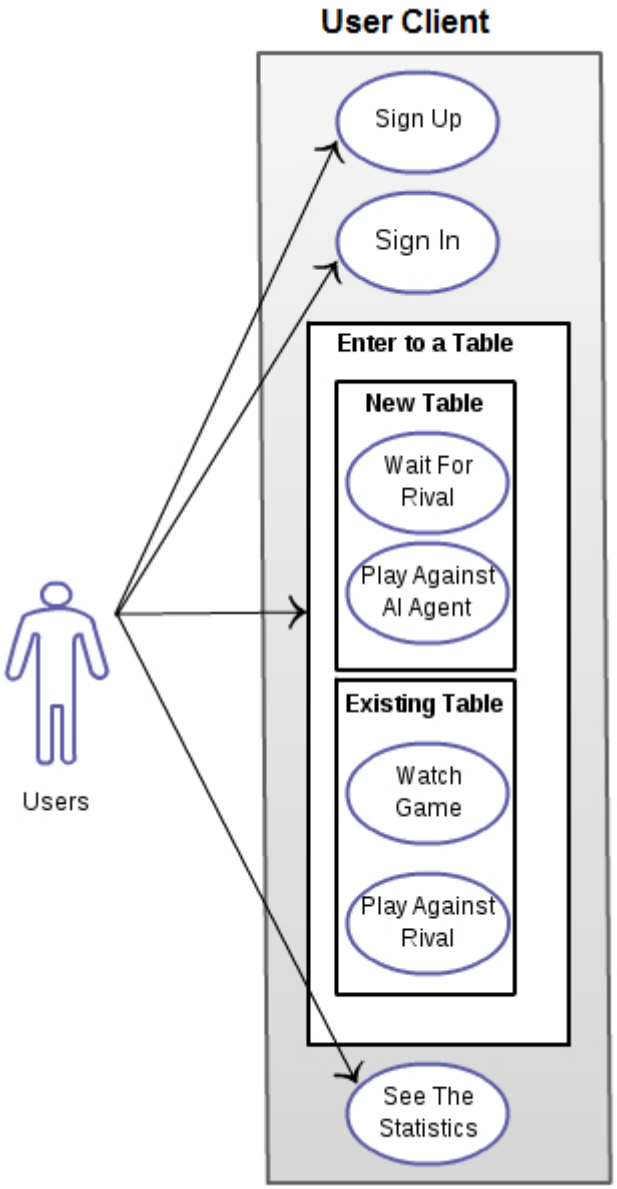
The diagram below shows the general structure and main components of the product.The database which holds the user information also starts to keep records of users' all statistics when users sign in to the system. This statistics database is updated with all new records when users signed out and they can see their all updated statistics in their next entrance to the system.

The game has two different interfaces, namely user interface and administrator

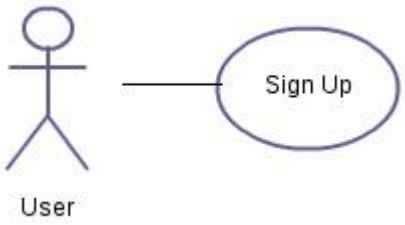
interface. As mentioned before, users will see their all information and statistics in their own interface while administrators can manipulate these statistics with their different interface.

2.2. Product Functions

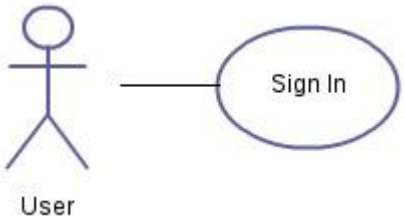
2.2.1. Users Functions



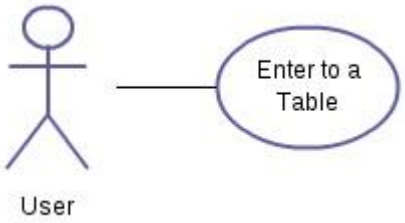
Sign-up: Users should register to system before they can play the game.



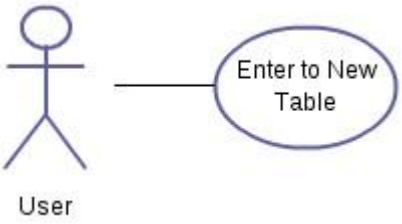
Sign-in: After signing up, users must sign in with their usernames and passwords provided by the system enter to a table.



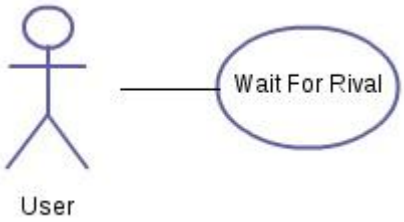
Enter to a table: When users sign in to system, they have to enter to a table to play the game. This can be in two ways.



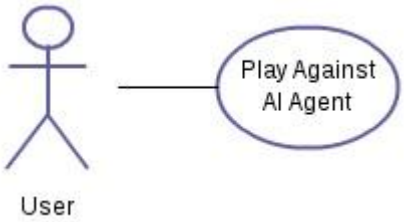
New table: Users can create a new table.



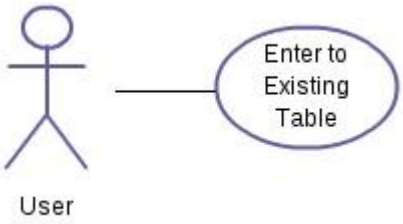
Wait for rival: After creating the new table they can wait for a rival to enter their table.



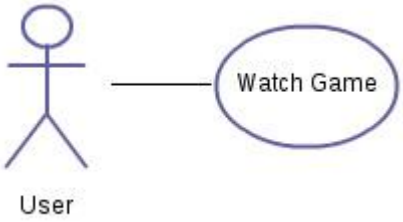
Play against AI agent: If users do not want to wait for a rival, they can just start playing with AI agent.



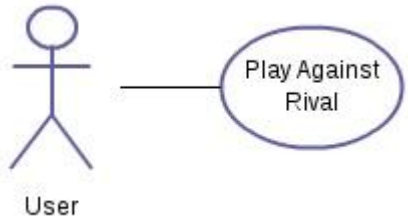
Existing table: Users can join to a table that is created by another user.



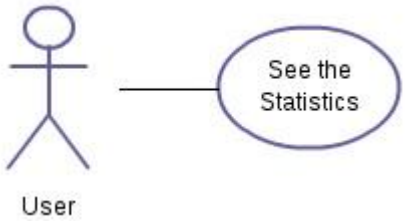
Watch game: When users join to a existing table, if the table is full, they can watch the game that is played by other users.



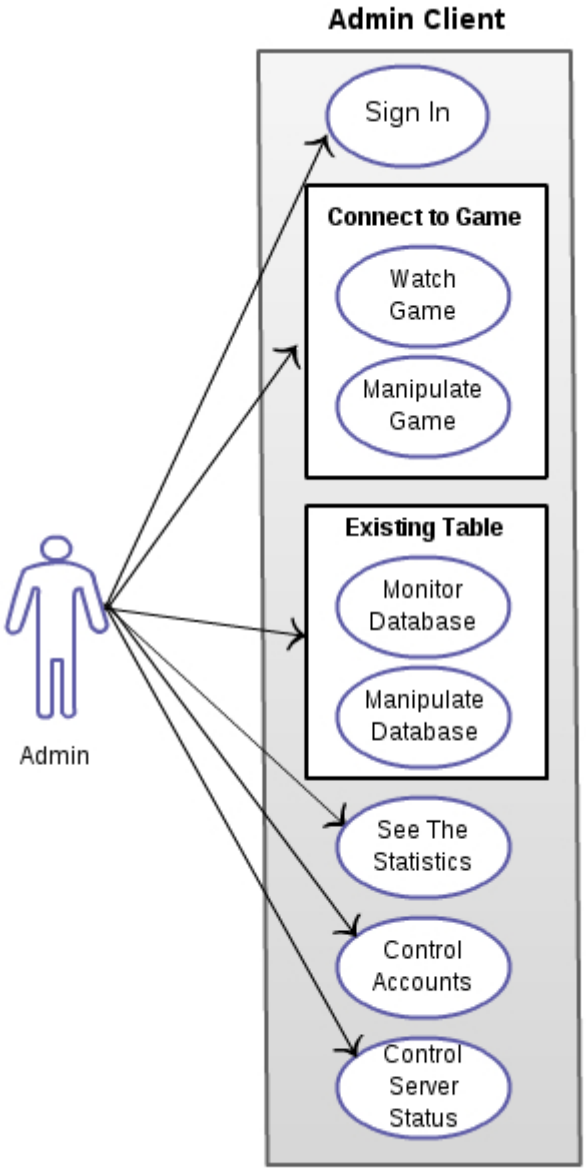
Play against rival: If the table owner has waiting for a rival and the table is not full, users can play against these users.



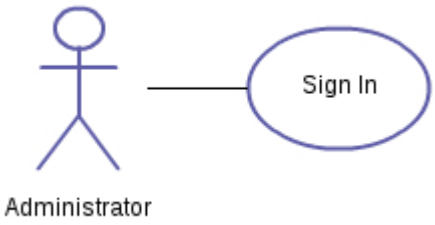
See the statistics: Users can see their all old statistics when they sign in to system.



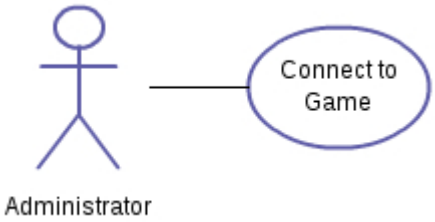
2.2.2. Administrator Functions



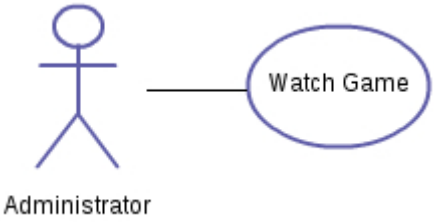
Sign-in: Administrators have to sign in to system before they can use.



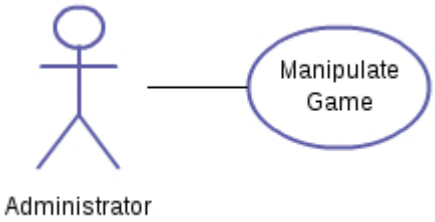
Connect to game: After signing in, they can connect to game.



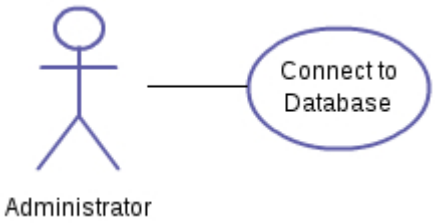
Watch game: If they connect to users' existing table, they can watch the game.



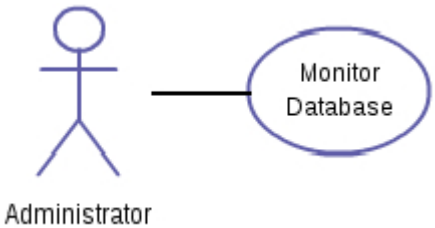
Manipulate game: Administrators can manipulate the users' game according to their wills.



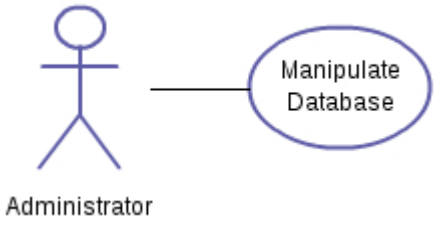
Connect to database: Administrators can connect to database where all information about the game is hold.



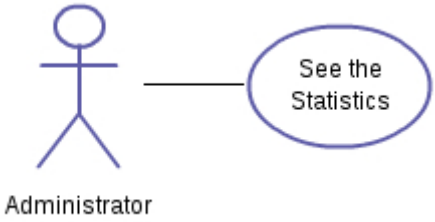
Monitor database: They can monitor the information in database.



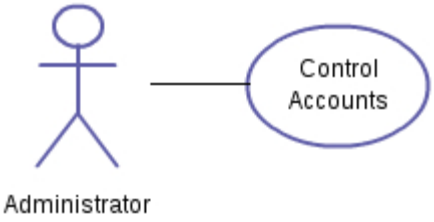
Manipulate database: They can manipulate database and can change the information on it.



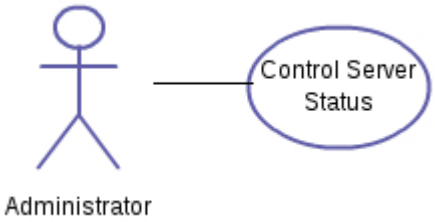
See the statistics: As users also have, administrators can also see the users' recorded statistics.



Control accounts: Administrators can control accounts.In another words they can delete or update user accounts.



Control server status: Administrators can also control server status, as natural.



2.3. Constraints, Assumptions and Dependencies

3. Specific Requirements

This section specifies and describes the software requirements of the project with enough details to allow programmers to design and test a system that satisfies the pre-defined specification and requirements.

3.1. Interface Requirements

3.1.1. User Interfaces

There are two user classes and therefor two interfaces for “blöflü pişti”, namely “players” and “administrators”. While interfaces allow players to play and interact with the game, administrators monitor and manage the game and databases. Following sections describe detailed characteristics of the said interfaces.

User-Client Interface: Client will be installed on a mobile phone. After installation, client software will be able to interact with the system via web services. User will be able to connect to the server, monitor own statistics, register a user, log in as a user, create a table or join an existing tables and play the game.

Admin-Client Interface: Client will be installed on a computer. After installation, client software will be able to interact with, change or update the system via web services. Admin will be able to connect to the server, monitor and manipulate the game and databases, view user statistics, control player accounts and server status.

3.1.2. Hardware Interfaces

The client and server have different hardware interfaces. Following sections describe detailed characteristics of the said interfaces.

User-Client Hardware Interface: The client component of the game will rely on GPU and CPU of iPhone's SoC's to compute and process the network operations, game graphics and player interactions. CPU will be of the “Arm Cortex” family and GPU will be of “PowerVR” family products. Clients will have at least 512(2x256) RAM and runs on a 1GHz device. For the optimal network load actions will be responded by client software then get confirmed by the server and become true. Client will also require a “wifi adapter” or “gprs/edge/3g supported Cellular data transfer arrays” to connect to server online.

Admin-Client Interface: The client component for administrators will rely on GPU and CPU of a computer to compute and process the network operation, game graphics, and server or database interactions. CPU will belong to x86 family of instruction set architecture so that it can operate on 64 bit processors with 32 bit backward compatibility. Client software will also require an Ethernet card in order to communicate with the server.

Server Interface: The server component of the system will be interfacing with an x86 processor to process database and network tasks. For numerous clients to connect at the same time internet connection with 100MB/sec or faster is required. A medium sized storage for the database is also required.

3.1.3. Software Interfaces

User-Client software will be an application of iOS whereas Admin-Client software and Server softwares will be an application of NT family operating systems. User-Client will be developed in Objective C, whereas Admin-Client and Server softwares will be java with C++ native language support. As different programming languages will be used for different components, Microsoft .Net 4.0 will be used for interoperability. XNA 4.0 runtime libraries will be used for client side graphics computations. For network communication, the client and the server software will communicate through a .Net web service.

3.1.4. Communications Interfaces

The interaction between server and client will be maintained with a web service implementation. No interaction between clients will be possible. Only way to interact between clients will be through the server.

3.2. Functional Requirements

Each major function with its data flow and requirements will be presented in this section.

3.2.1. Sign-Up

Description: This function specifies how users sign-up for the game.

Basic Data Flow:

1. User runs the client component of the project.
2. User selects “Sign-Up” button from the client.
3. User fills the form for user-name, password and e-mail.
4. The user account is created.

Alternative Data Flow 1:

1. User runs the client component of the project.
2. User selects “Sign-Up” button from the client.
3. User fills the form for user-name, password and e-mail.
4. The user-name exists in the database. Server denies the client request.
5. The GUI shows an error message asking for user to select different user-name.
6. The sign-up user interface goes back to (3).

Alternative Data Flow 2:

1. User runs the client component of the project.
2. User selects “Sign-Up” button from the client.
3. User fills the form for user-name, password and e-mail.
4. Password is too short (6 characters) or contains illegal characters. Server denies the client request.
5. The GUI shows an error message asking for user to select different password.
6. The sign-up user interface goes back to (3).

Alternative Data Flow 3:

1. User runs the client component of the project.
2. User selects “Sign-Up” button from the client.

3. User fills the form for user-name, password and e-mail.

4. The email adresse exists in the database. Server denies the client request.

5. The GUI shows an error message asking for user to select different email adresse.

6. The sign-up user interface goes back to (3).

Functional requirements:

1. REQ-1: Server should not accept the same user-name twice when signing-up users.

2. REQ-2: Server should not accept passwords shorter than 6 characters. Also passwords should not contain illegal characters.

3. REQ-3: Server should not accept the same email adresse twice when signing-up users.

3.2.2. Sign-in

Description: This function specifies the signing-in process of the game.

Basic Data Flow:

1. User runs the client component of the game.

2. User selects “Sign-in” button from the client.

3. User fills the form for user-name and password.

4. Server autenticates the user-id and password, and the user is granted access to the game.

Alternative Data Flow 1:

1. User runs the client component of the game.
2. User selects “Sign-in” button from the client.
3. User fills the form for user-name and password.
4. Server does not autenticate the user-name and password entered.
5. The signing-in user interface goes back to (3).

Functional Requirements:

1. REQ-4: Server should check the user-name and password entered by the user.

3.2.3. Create a Table

Description: This function specifies the “creating a table” process.

Basic Data Flow:

1. User runs the client component of the game.
2. User selects “Sign-in” button from the client.
3. User fills the form for user-name and password.
4. Server autenticates the user-id and password, and the user is granted access to the game.
5. User selects “Create a new table” button from client.
6. Server accepts the request. A new table is created on the server.
7. User is automatically joined newly created table.
8. Table joins “existing tables” list for new users.

Alternative Data Flow 1:

1. User runs the client component of the game.

2. User selects “Sign-in” button from the client.

3. User fills the form for user-name and password.

4. Server autenticates the user-id and password, and the user is granted access to the game.

5. User selects “Create a new table” button from client.

6. Server doesn't accept the request, showing the user the reason(max number of tables reached).

Functional Requirements:

1. REQ-5: Server should check the number of tables to see if it reached the maximum possible.

3.2.4. Join an Existing Table

Description: This function specifies the “joining an existing table” process.

Basic Data Flow:

1. User runs the client component of the game.

2. User selects “Sign-in” button from the client.

3. User fills the form for user-name and password.

4. Server autenticates the user-id and password, and the user is granted access to the game.

5. User selects “Join an Existing Table” button from client.

6. Server accepts the request. A list of existing tables shown to the user.

7. User selects a table to join from the list.

8. Server accepts the request. User enters selected table.

Alternative Data Flow 1:

1. User runs the client component of the game.

2. User selects “Sign-in” button from the client.

3. User fills the form for user-name and password.

4. Server autenticates the user-id and password, and the user is granted access to the game.

5. User selects “Join an Existing Table” button from client.

6. Server denies the request. (No available table found).

7. User redirected to “Create a table” function.

Alternative Data Flow 2:

1. User runs the client component of the game.

2. User selects “Sign-in” button from the client.

3. User fills the form for user-name and password.

4. Server autenticates the user-id and password, and the user is granted access to the game.

5. User selects “Join an Existing Table” button from client.

6. Server accepts the request. A list of existing tables shown to the user.

7. User selects a table to join from the list.

8. Server declines the request. (table already full).

9. Server returns fresh list of existing tables returning to (6).

Functional Requirements:

1. REQ-6: Server should check if a table has been filled before allowing another user to join that table.

3.2.5. Leave Table

Description: This function describes the process of leaving a table.

Basic Data Flow:

1. User runs the client component of the game.
2. User selects “Sign-in” button from the client.
3. User fills the form for user-name and password.
4. Server autenticates the user-id and password, and the user is granted access to the game.
5. User selects “Create a new table” button from client.
6. Server accepts the request. A new table is created on the server.
7. User is automatically joined newly created table.
8. User selects “Leave Table” from the client.
9. Server accepts the request. User is removed from the table.
10. No users in table. Table is deleted.

Alternative Data Flow 1:

1. User runs the client component of the game.
2. User selects “Sign-in” button from the client.
3. User fills the form for user-name and password.
4. Server autenticates the user-id and password, and the user is granted access to the game.
5. User selects “Join an Existing Table” button from client.
6. Server accepts the request. A list of existing tables shown to the user.
7. User selects a table to join from the list.

8. Server accepts the request. User enters selected table.

9. User selects “Leave Table” from the client.

10. Server accepts the request. User is removed from the table.

11. Users in table. Table joins “existing tables” list for new users.

Functional Requirements:

1. REQ-7: Server should check the table if there are any users left on a table before removing the table.

3.2.6. Add AI agent

Description: This function specifies the AI agent creation. It does not however specify any content related to logic and actions of the agent.

Basic Data Flow 1:

1. User runs the client component of the game.

2. User selects “Sign-in” button from the client.

3. User fills the form for user-name and password.

4. Server autenticates the user-id and password, and the user are granted access to the game.

5. User selects “Create a new table” button from client.

6. Server accepts the request. A new table is created on the server.

7. User is automatically joined newly created table.

8. User selects “Add AI Agent”.

9. Server accepts the request. An AI agent is created and joined the table.

Alternative Data Flow 1:

1. User runs the client component of the game.
2. User selects “Sign-in” button from the client.
3. User fills the form for user-name and password.
4. Server autenticates the user-id and password, and the user is granted access to the game.
5. User selects “Join an Existing Table” button from client.
6. Server accepts the request. A list of existing tables shown to the user.
7. User selects a table to join from the list.
8. Server accepts the request. User enters selected table.
9. Other player leaves the table.
10. User selects “Add AI Agent”.
11. Server accepts the request. An AI agent is created and joined the table.

3.2.7. Play Game

Description: This function specifies the playing the game process. It does not however, specify any content within the game logic, nor does it specify the graphics within the game.

Basic Data Flow:

1. User runs the client component of the game.
2. User selects “Sign-in” button from the client.
3. User fills the form for user-name and password.
4. Server autenticates the user-id and password, and the user are granted access to the game.

5. User selects “Create a new table” button from client.

6. Server accepts the request. A new table is created on the server.

7. User is automatically joined newly created table.

8. User waits for another player to join the table.

9. Another player joins the table.

10. Server removes the table from “existing tables” list.

11. Users play the game.

Alternative Data Flow 1:

1. User runs the client component of the game.

2. User selects “Sign-in” button from the client.

3. User fills the form for user-name and password.

4. Server autenticates the user-id and password, and the user is granted access to the game.

5. User selects “Join an Existing Table” button from client.

6. Server accepts the request. A list of existing tables shown to the user.

7. User selects a table to join from the list.

8. Server accepts the request. User enters selected table.

9. Server removes the table from “existing tables” list.

10. Users play the game.

Alternative Data Flow 2:

1. User runs the client component of the game.

2. User selects “Sign-in” button from the client.

3. User fills the form for user-name and password.

4. Server autenticates the user-id and password, and the user is granted access to the game.

5. User selects “Create a new table” button from client.

6. Server accepts the request. A new table is created on the server.

7. User is automatically joined newly created table.

8. User selects “Add AI Agent”.

9. Server accepts the request. An AI agent is created and joined the table.

10. User plays the game.

Alternative Data Flow 3:

1. User runs the client component of the game.

2. User selects “Sign-in” button from the client.

3. User fills the form for user-name and password.

4. Server autenticates the user-id and password, and the user is granted access to the game.

5. User selects “Join an Existing Table” button from client.

6. Server accepts the request. A list of existing tables shown to the user.

7. User selects a table to join from the list.

8. Server accepts the request. User enters selected table.

9. Other player leaves the table.

10. User selects “Add AI Agent”.

11. Server accepts the request. An AI agent is created and joined the table.

12. User plays the game.

3.2.8. Signing-out

Description: This function specifies the logging out process.

Basic Data Flow:

1. User runs the client component of the game.
2. User selects “Sign-in” button from the client.
3. User fills the form for user-name and password.
4. Server autenticates the user-id and password, and the user is granted access to the game.
5. User selects “Leave the game” button from client.
6. Server accepts the request. User is removed from the server.
7. Client closes itself.

Alternative Data Flow 1:

1. User runs the client component of the game.
2. User selects “Sign-in” button from the client.
3. User fills the form for user-name and password.
4. Server autenticates the user-id and password, and the user is granted access to the game.
5. User selects “Create a new table” button from client.
6. Server accepts the request. A new table is created on the server.
7. User is automatically joined newly created table.
8. User selects “Leave the game” button from client.
9. Server accepts the request. User is removed from table, then user is removed from the server.
10. Server removes the empty table.
11. Client closes itself.

Alternative Data Flow 2:

1. User runs the client component of the game.

2. User selects “Sign-in” button from the client.

3. User fills the form for user-name and password.

4. Server autenticates the user-id and password, and the user is granted access to the game.

5. User selects “Join an Existing Table” button from client.

6. Server accepts the request. A list of existing tables shown to the user.

7. User selects a table to join from the list.

8. Server accepts the request. User enters selected table.

9. Server removes the table from “existing tables” list.

10. User selects “Leave the game” button from client.

11. Server accepts the request. User is removed from table, then user is removed from the server.

12. Client closes itself.

3.2.9. Monitor Statistics

Description: This function specifies the logging out process.

Basic Data Flow:

1. User runs the client component of the game.

2. User selects “Sign-in” button from the client.

3. User fills the form for user-name and password.

4. Server autenticates the user-id and password, and the user is granted access to the game.

5. User selects “View my stats” button from client.

6. Server accepts the request.

7. List of statistics is shown to user.(win/loss ratio, bluff/real ratio, etc.)

3.2.10. Manage Accounts

Description: This function describes how administrators manage accounts of users.

Basic Data Flow:

1. Administrator logs in to the server computer.
2. Administrator selects “accounts” button from server GUI.
3. GUI shows accounts in tabular form with associated actions.
4. Administrator views and takes actions on managing accounts.
5. Administrator updates the accounts and selects “submit” button.
6. Administrator log out from the server computer.

Alternative Data Flow 1:

1. Administrator logs in to the server computer.
2. Administrator selects “accounts” button from server GUI.
3. GUI shows accounts in tabular form with associated actions.
4. Administrator views and takes actions on managing accounts.
5. Administrator updates the accounts and selects “submit” button.
6. Server rejects the changes explaning the administrator the error occurred.
7. The GUI goes back to (2).

Functional Requirements:

1. REQ-8: Server GUI should not accept multiple changes at one submit to prevent malicious errors

3.3. Non-Functional Requirements

3.3.1. Performance Requirements

In order for “blöflü pişti” to be massively mutiplayer, the server should support over 1000 users concurrently. Although it is turn based and not “real time” delay of communications between server and user should not exceed 1 second.

In order for this game to be maintainable, the GUI of the server component should be able to accept administrator connections esspecially when overloaded. The administrator should be able to log on to server within 5 seconds even when server supports 1000 players. When administrator selects an operation on server GUI it shoul be available in 10 seconds and be actualized by the server within 1 minute. Should server fail to accommodate users succesfully and need a reboot, it should be available in 5 minutes of getting the command from an administrator.

3.3.2. Design Constraints

1. **Security:** The communications between the server and client will use WS-Security in Web Services Enhancements for Microsoft .Net (WSE). Secure communication will be provided by XML digital signatures to ensure message integrity and XML encryption for message confidentiality. In order to prevent abuse of the server component, software on the server should block non-ending recurring requests from the same IP adress. Server should also log common proxy addresses as well to check if the client is performing DoS attack.

2. **Reliability:** User account informations will be collected in the database, so if an error were to occur, users should not feel any consequences. There should always be backups at regular intervals. User statistics will also change how AI agent will play therefore it is of great importance to keep the data consistent.
3. **Usability:** Client side of "Blöflü Pişti" should be very easy to use with a friendly GUI. As the game is pretty easy so should be the client usage. Users should be able to learn and play the game in minimal time while agents will be of enough challange for users who wish to improve their game. Also agents can be changed into newcomer "mode" to make it easier for new players.
4. **Availability:** All online game applications have the risk of server failure due to excessive number of users. Having regular backups is also important here since it will allow the server to go back to available mode in minimal time. That a server failure was to happen, a reboot and system restoration should commence in a matter of minutes.

Software System Attributes:

1. Maintainability: The modification of the source code should be disabled. The extensions should be applicable directly to the back end, without modification. The patching service should be regarded as a different component in case extensions would be off-loaded to a different server.
2. Portability: Client component of the software should be portable. Target hardware platform for the client is smart mobile phones namely of Apple'e iPhone family. Therefore the client software will be deployed onto iOS operating systems with version 4 and above. However the server side of the game should be independent of specific hardware or software configuration.

3. Scalability: Most important system attribute of the game should be its scalability. Deploying additional server and dividing the number of players among themselves. And these operations should not require additional extension on the client software.

4. Data Model and Description

4.1. Data Description

The complete data model is given in Figure 4.1.a. The focus of the model is on Table, User and AI agent objects. Account and statistics are associated with User object. AI agent however will be able to access oppositions data and will be recording its own data, along with determining game situation and change personality according to it. Both Users and AI agents will be able to play only when they are in a Table object. Following sections describes all the data objects and their major attributes in more detail.

4.1.1. Data Objects

In this section a brief description of each daha object is given. For each data object, function and semantics associated with it are summarized. This section also describes majer attributes of data objects.

Table: This data object represents a table created by users. Database needs to store the identifier of the table which is its major attribute. Other attributes of a table is user-id's which are also stored in database.

Id: Unique identifier of a table data object.

User_id_1: an attribute of table data object.(never null)

User_id_2: an attribute of table data object.(default=null)

User: The user data object -as its name suggests- represents a player character in the game. Besides its unique identifier id, it also has 4 major attributes namely account, online, points, stats.

Id: The unique identifier of the User data objective.

Account: References corresponding Account data object.

Online: Boolean attribute showing whether user is connected to server or not.

Points: Points accumulated through playing with other users.

Stats: References corresponding Statistics data object.

AI Agent: This data object represents an artifical intelligence agent, simulating a user in any table. Id is its unique identifier; it also has other major attributes, table_id, user_id, tendancy, and mode.

Id: The unique identifier of the AI Agent data objective.

table_id: The id of the table agent is currently located.

user_id: The id of the user, who summoned the agent.(for referencing statistics).

Tendancy: derived from oppositions statistics.

Mode: predefined playing styles. (Default: null)

Account: This data object stores the account information of users. The account data

object does not store all character information. However it is referenced by a user object. Email is unique identifier; account name is also unique but not identifier. The rest is only major attributes.

Email: The email address provided when creating the account.

Name: the username chosen by account owner.

Password: the password of the account owner.

Online: Boolean attribute showing either the account owner is connected to server.

Confirmed: Boolean attribute showing whether an account has been confirmed via e-mail activation or not.

Statistics: This data object represents the statistical data stored on the database pes user basis. Id is unique identifier same as user id and referenced by user data object. Other data is statistical.

Id: The id of statistics, referenced by user.id as they are equal.

Win: number of games won.

Total_game: number of games played.

Bluff: number of times bluffed.

Total_pisti: number of times made a “pisti”.

4.1.2. Relationships

This section describes the relationships between the data objects described in the previous section.

Table - User: Each table must have at least 1 user in it or they are decommissioned. However a user does not need to be in a table at all times. Table can hold multiple users or one user and an AI agent.

Table – AI agent: Table must have 1 user already in before an AI agent can be summoned to a table. AI agent keeps track of in which table it is summoned.

User – AI agent: AI agent must be summoned by a User to exist. Each AI agent therefore must play against a user; however users don't need an AI agent to play a game.

User – Account: Each user must be associated with at most one account. Account attributes can be changed later where user cannot.

User – Statistics: Each statistic must belong to a user with the same id. Some of the user statistics may not exist at creation but later on incremented.

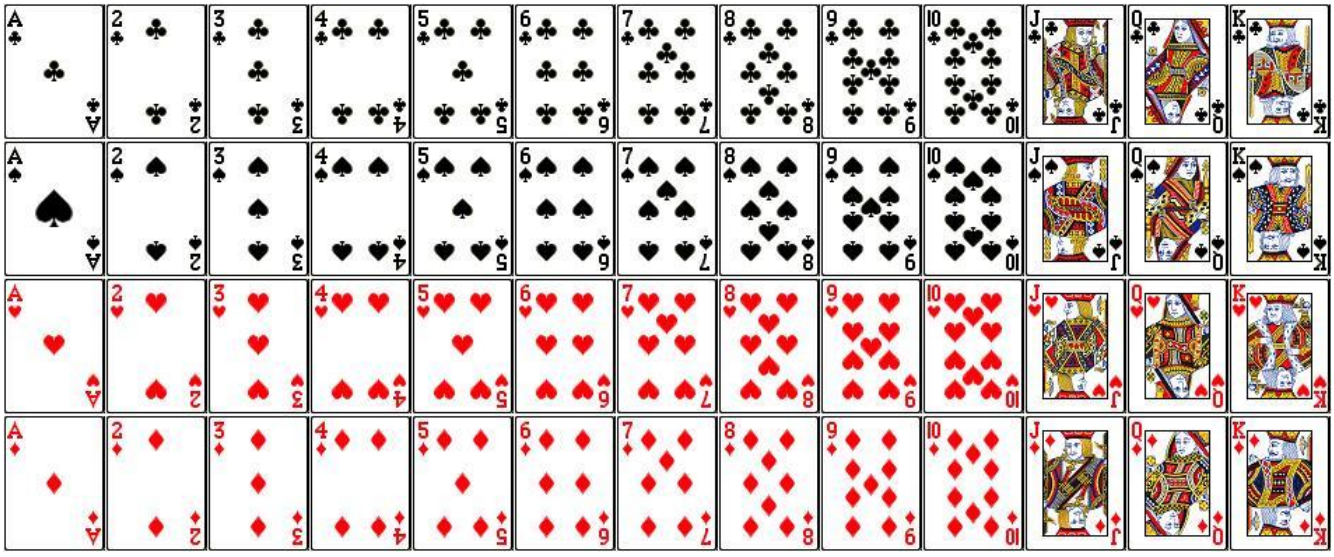
AI agent – Statistics: AI agent will try to check opposition's statistics and develop a play style. AI agent can also change behaviour depending on derived data from statistics.

4.1.3. Complete Data Model

This section describes a complete data model merging data object descriptions with relationships explained in previous sections. Figure 4.1.3.a shows the entiti relationship diagram of the data model which provides a conceptual representation of data. Major objects are User, AI agent, Table, Statistics, Account.

4.1.4. Data Dictionary

Standard 52-card deck: The primary deck of fifty-two playing cards in use today, usually known as the French deck, includes thirteen ranks of each of the four French suits, clubs (♣), diamonds (♦), hearts (♥) and spades (♠), with reversible Rouennais "court" or face cards. Some modern designs, however, have done away with reversible face cards. Each suit includes an ace, depicting a single symbol of its suit; a king, queen, and jack, each depicted with a symbol of its suit; and ranks two through ten, with each card depicting that many symbols (*pips*) of its suit. Two (sometimes one or four) Jokers, often distinguishable with one being more colorful than the other, are included in commercial decks but many games require one or both to be removed before play. Modern playing cards carry index labels on opposite corners (rarely, all four corners) to facilitate identifying the cards when they overlap and so that they appear identical for players on opposite sides.



Bluff: (or deception, beguilement, deceit, mystification bad-faith, and sub-terfuge) are acts to propagate beliefs that are not true, or not the whole

truth (as in half-truths or omission). Bluff can involve dissimulation, propaganda and sleight of hand. It can employ distraction, camouflage or concealment.

Bluff is a major relational transgression that often leads to feelings of betrayal and distrust between relational partners. Bluff violates relational rules and is considered to be a negative violation of expectations. Most people expect friends, relational partners, and even strangers to be truthful most of the time. If people expected most conversations to be untruthful, talking and communicating with others would require distraction and misdirection to acquire reliable information. On a given day, it is likely that most human beings will either deceive or be deceived by another person. A significant amount of deception occurs between romantic and relational partners.

5. Behavioral Model and Description

5.1. Description for Software Behavior

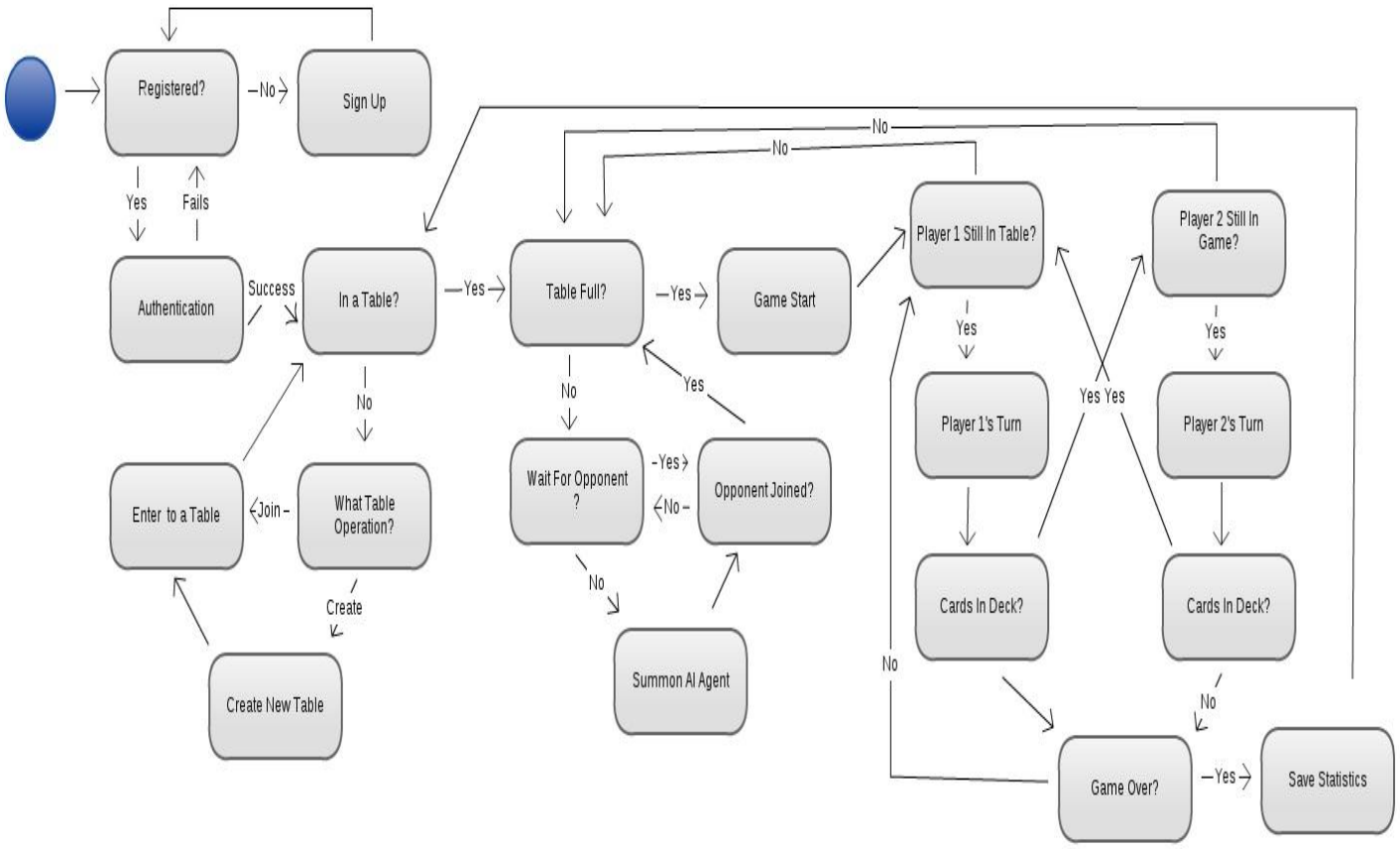
The game has two main parts, client and server. The relationship between these two components is that they communicate by accomplishing a package transfer between corresponding part's communication layer.

Game loop, which belongs to client's component, interacts with Tables on the server component and during that interaction; information is passed to server, therefore to tables via communication layers. Information can be deck composition, played cards, current score or a bluff. It also initiates the game to start.

Database inside the server component stores the information such as account

information of the users, their statistics etc. it also allows AI agents to derive their play style according to these values.

5.2. State Transition Diagrams



6. Planning

6.1. Team Structure

"Blöflü Pişti" project does not have any team leader but processes with sharing of the work to be done. Each member of the team puts his effort to the project.

Team has weekly meeting with department appointed assistant. Team also

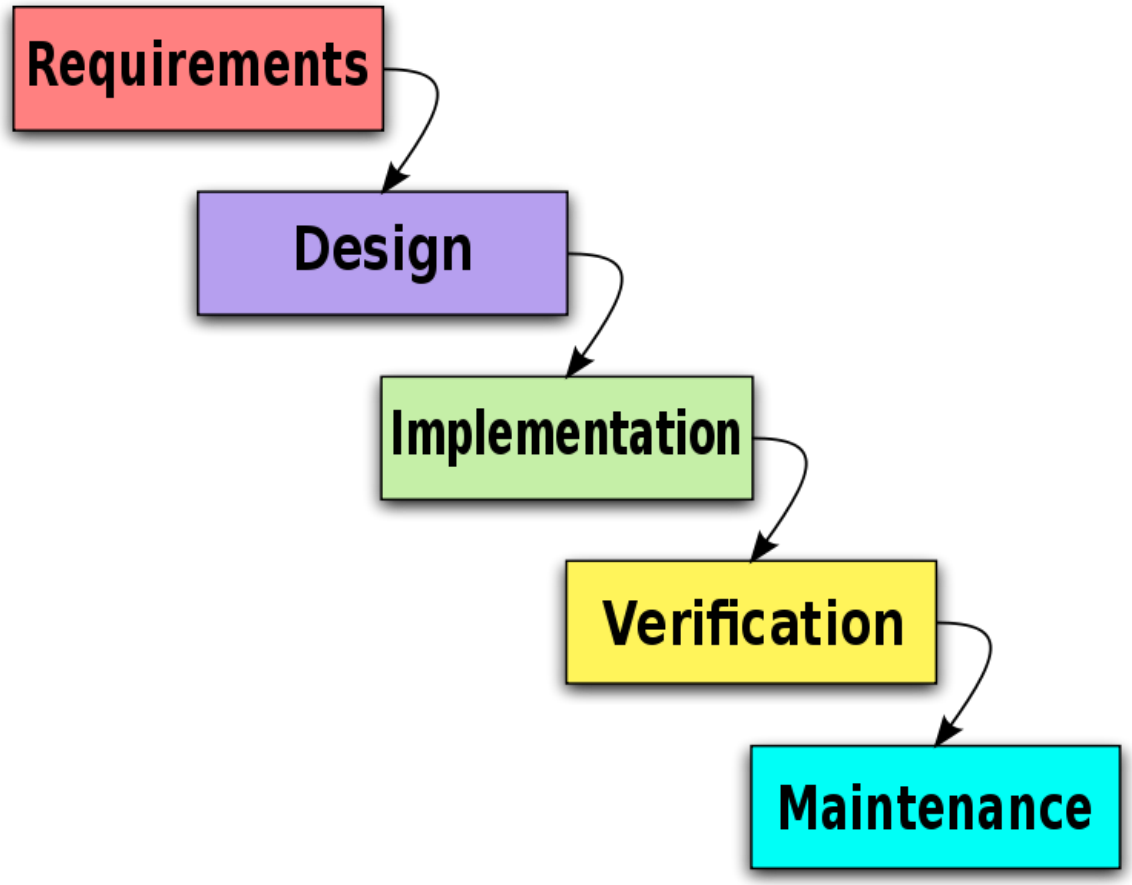
meets with Usta Yazılım for progress reports and decision making. Members of the team study the topics that have been determined before with fully participation every Tuesday and Thursday.

6.2. Estimation

The plan is to prepare the detailed design report and prototype demo till the end of the semester. Second semester will mainly base of configuration management, system design development and system testing.

The detailed schedule of project is given in figure 23.

6.3. Process Model



7. Conclusion

This report shows Frog on Fire group's approach at creating a game with AI agents, namely "blöflü pişti". All system components, interactions among them and relations between different data objects are defined in this report. Market research presented in this document also covers related other games since no currently available game is same with this projects specifications. Scheduling and timeline is given a little loosely than actual planned timeline to make room for future possibilities of delay. This specification will constitute the basis for design, development and testing of the project however significant changes can be made with Usta Yazılım in the process.