CHALMERS

Internet Board Game Server

Design and implementation of a Correspondence Board Game Server

Master of Science Thesis in Software Engineering

Golnaz Seyrafi

Department of Computer Science

Division of Software Engineering

CHALMERS UNIVERSITY OF TECHNOLOGY

Göteborg, Sweden, 2009

Internet Board Game Server

Golnaz Seyrafi

Examiner: Sven Arne Adreasson

Chalmers University of Technology
University of Gothenburg
Department of Computer Science and Engineering
SE-412 96 Göteborg
Sweden
Telephone + 46 (0)31-772 1000

Department of Computer Science and Engineering
Göteborg, Sweden January 2010

Abstract

Like in many other industries, to get quality product it is necessary to define what is going to be made, design and plan the process before actual building is started. On the other hand attempting to design everything perfectly can result in “analysis paralysis”, a situation with long phases of planning and designing with little or no value achieved. In this master thesis it is tried to compare different solutions for developing a web application and practice the different stages of developing a web application using the selected solution. Although the first intention for developing this website was to play Shogi (Japanese chess), the design makes it easy to add any two player board game to the server. Code reuse and using appropriate development process and development tools are practices that can improve both productivity and time to market. The development process in this project is a variant of FDD, which benefits from advantages of agile and traditional development processes. Testing is done according to agile testing philosophy; Usability testing started after making the simplest prototype.

Contents

1. Introduction

A web application is an application that is accessed via a web browser over Internet. Although the focus of the project was to develop a website for playing the game Shogi, it is designed in a way that other two player board games can be added easily. The development process which is followed is a variant of Feature Driven Development. Among many different development processes, Feature Driven Development fulfils the agile goals, and still emphasis on up-front design and documentation. FDD starts by defining the domain model, but in this process the project is first defined using use cases and use case diagrams, then the domain object model is defined, then the class diagrams, containing classes, methods and attributes, are defined. The development is planned based on the methods of the classes (features). Testing is done to ensure that both use cases and classes are implemented as expected.

The whole project is done using open source development tools. ArgoUML and StarUML are used for the design and LAMP for implementation. Java script is used at client side for best compatibility (for example with mobile devices).

Following is a brief introduction to Japanese chess, Shogi, and later about correspondence board games.

1.1. Shogi

The old Indian game Chartunga is said to be the common ancestor of most modern chess games, like Xiangqi (Chinese Chess), western chess and Shogi (Japanese Chess). They are all abstract two player board games. Shogi is by far the most complex chess game that is actually played. Shogi is mainly played in Japan, with about one million players. It has in recent years met a big popularity in Shanghai, where it perhaps is more common than Xiangqi today. Figure 1 shows the start position of pieces on a Shogi board.

The rules are much the same as western chess, except two things. The pieces are flat and have same color for both opponnents. They have five sides and the direction decides which player side they belong to. The other difference is that captured pieces are kept by the player who captured it, and can be ‘dropped’ into the board as a move later. A piece that goes into or leaves the opponents zone (the first three ranks) can be upgraded by ‘promotion’. This is done by flipping the piece so that you see its backside. Once promoted it cannot be transformed back. The king and the gold cannot be promoted.

The goal of the game is to take the opponents king. The player who does that wins the game. In practice one never takes the king, instead winning is claimed when that is possible.

1.1.1. Special rules

Shogi has complicated rules that designate it from any other board games. The most important rules are:

- Two pawns cannot be placed at the same file unless they are promoted. It is an illegal move and result in a loss.

- You cannot win by dropping a pawn. It will result in a loss.

- If a position is repeated three times in a row the game is draw, and should be played again. A draw is not a legal result in Shogi. If the repetition is caused by repeated check, the player who causes the check will lose the game.

- If both players' kings reach the opponents back rank the game is judged by counting pieces in a special way.

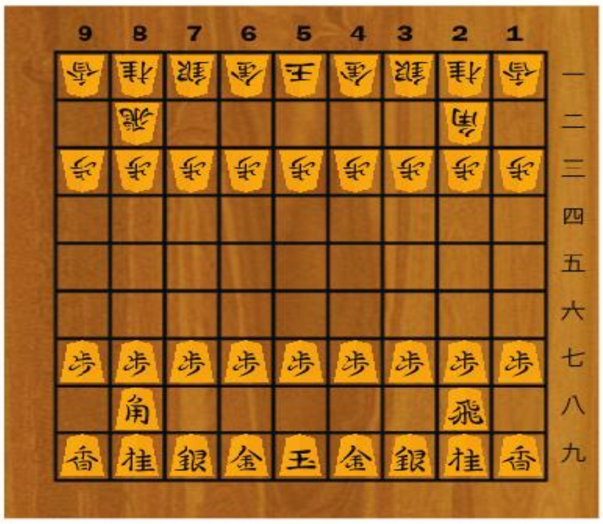


Figure1. Start position

Figure 2, shows the movement of the Shogi pieces, both as default and when they are promoted.

1.2. Server-based correspondence board games

Correspondence playing of board games has been done by post for long time. The email was a natural follow up and today most correspondence games are handled by game servers. Traditionally correspondence playing was for deeper analyzes of the game. Today it is popular since people have limited time for playing.

Old time correspondence games were very slow. East Germany won an international team game in western chess four years after their country did not longer exist. Today games are often finished within a few days.

Correspondence players often have several games going simultaneously. Tournament games can be played concurrently, and some players may have more than one hundred games continuing at the same time. The opponent doesn't need to be on line; a player makes a move and submits it. Legal move check ability excludes errors like mistyping. Most servers use some variation of Elo rating system to define player's strength.

Correspondence games are traditionally social type of playing. Each move usually includes a personal note. Today the game servers usually have functionalities such as social networking sites, such as forum, personal page, chat et cetera.

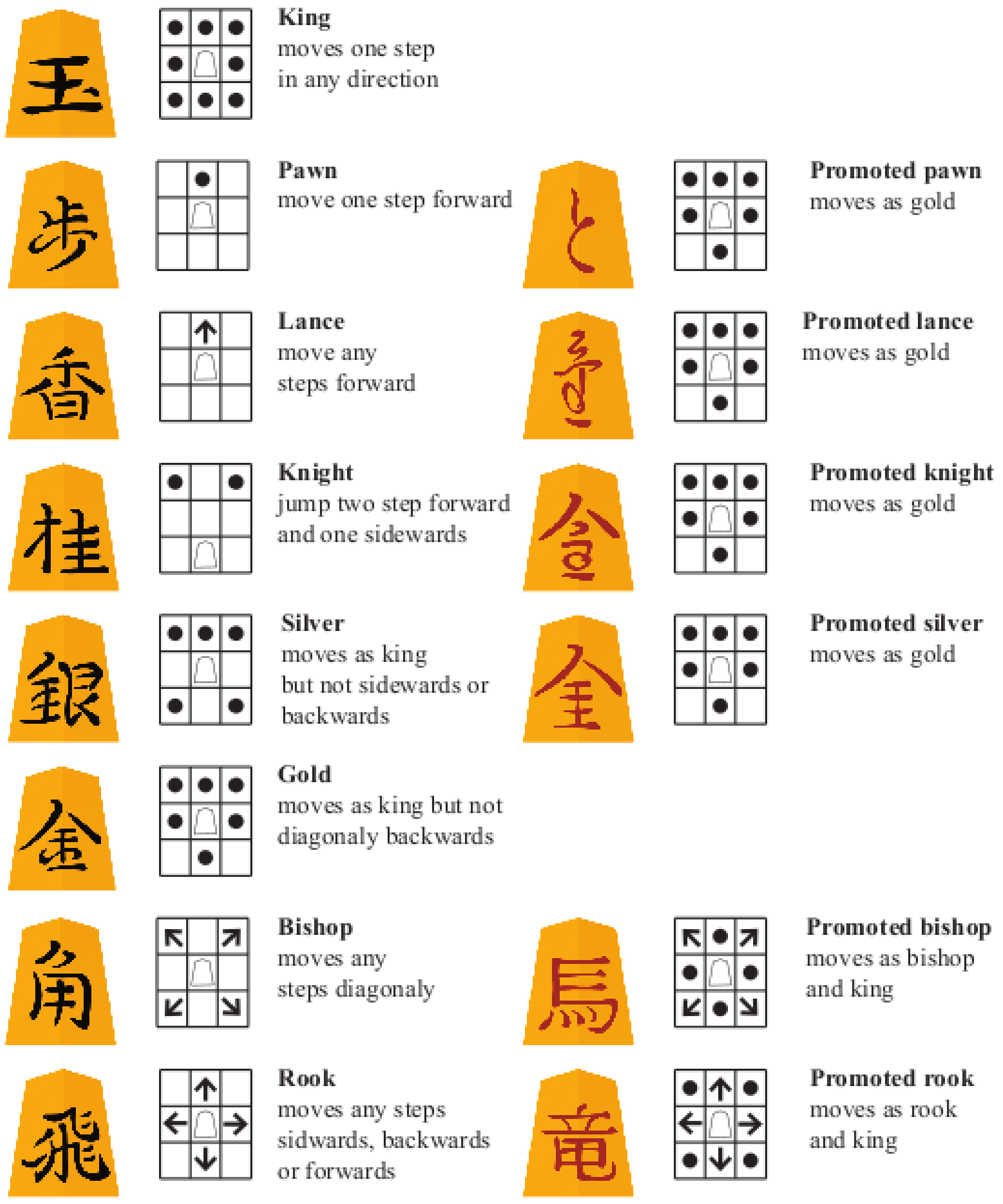


Figure2. Shogi pieces

2. Development process

A development process must be appropriate to the project at hand. In case of this project there were the time limit and the fact that all the roles were done by one person. For the purpose of having a thorough report for the thesis documents had to be provided to make the process clear. On the other hand the rigid methods which spend a lot of time on the planning part were not desirable as the time for planning was limited. Among agile methods Feature Driven Development focuses on both delivering documents and finishing the project in a timely manner. The steps in FDD according to [6] are:

- Develop an overall model
- Build a feature list
- Plan by feature
- Design by feature
- Build by feature

FDD is different from all other agile methods. For example, FDD values design over "the code is the design", FDD values design first. The unit of development in an FDD project is called a feature. Features (tiny client-valued function) are being completed every week in an FDD project. The process followed in this project is a variant of FDD. FDD starts with defining the domain model. Here it is tried to have a combination of RUP and FDD. The steps of FDD are covered in the same order in the development process. Although use cases are not a part of FDD, they are used in this development process. The process is started with use case diagrams. Finding actors and usecases is a great tool to get familiar with the domain. After that the domain model will be much clearer. The next step in FDD is listing the features. Here the features are listed using UML class diagrams. Class diagrams will make the implementation much easier; on the other hand the features will simply be the methods of the classes. Sequence diagrams are also provided to clarify the interaction between objects. After that the coding is started.

The testing is done according to agile testing, from the user perspective, as early as a primitive prototype becomes available. The feedback that is provided by the users in this way is the most valuable source to improve the application.

What The FDD "Meta-Process" basically says is, think a little bit before doing it, then detail it and do it a little bit at a time[6].

3. Requirements

Requirements are the base of a software application. The implementation will be based on this requirements and testing will verify that the requirements are satisfied. Requirements are divided into two categories, functional requirements and non-functional requirements. Functional requirements describe the behaviors (functions or services) of the system that support user tasks or activities. Non-functional requirements include constraints and qualities. Qualities are properties or characteristics of the system that will affect the degree of satisfaction of stakeholders with the system. Constraints are the limitations that should be considered during the development process. We can think of functional requirements capturing *what* the system must do, and the run-time qualities as describing *how well* these functional requirements are satisfied [3].

3.1. Functional requirements

The website should provide the users the possibility to play correspondence Shogi. A user should register in the site to be able to play in the site. After signing up, the player should be able to log in to the site. To play in a game there should be several choices; the player can start a game, other players should be able to see the list of the games which are started by others so that they can join them if they want. Another option is to search for a specific player, based on different factors such as rating, nationality, etc and challenge that player. The challenged player can accept or deny the challenge. If the challenged player accepts the challenge the game is started, otherwise a message is sent to the challenge initiative saying that the challenge is not accepted. Another option for playing Shogi on the site is to sign up for tournaments. The tournament starts automatically after six players have joined to play in the tournament. All the players in the tournament will play with each other which will make 15 games. When a tournament is started another tournament is created so that other people can join and play (Sit'n go). Players have a graphical board interface, where they can drag and drop the pieces and then submit their move. They can also analyze the game checking which moves are possible later. When a player submits a move, an email is sent to the opponent that now it is his turn to play. The other player logs in and he can see the list of the games he is playing. He can then continue playing. If a player feels that he cannot win the game he can resign the game, and the other player wins. After a game ends the ratings will be adjusted according to the result.

3.1.1. Use cases

Use cases are very useful in determining the classes, and they are especially important in testing the product. The actors considered for the website are Non-registered user, Registered user (Player) and Administrator. Following, the actors and their corresponding use cases are described.

Actor: Non-registered user

- **Sign up:** The players provide information in a form. After submitting, the server checks that the nickname is not taken and the email is not used before. It also checks that all other information is given correct. Then the server stores the user in a temporary table and generates a password. Then it sends a confirmation email to the player. When the player clicks the link with a confirmation password, the account will be activated.

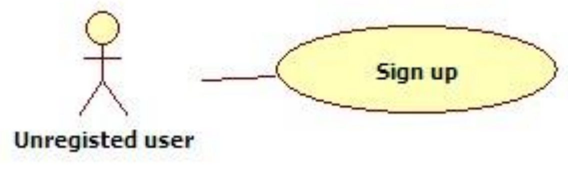


Figure3. Use case diagram for non-registered user

Actor: Registered user (Player)

- **Login:** The user enters user name and a password. If they match the player will be logged in.
- **Start game:** The player can start a game, other players should be able to see the game and join it if they want. The game will start when the second player joins the game.
- **Join a game:** Any player can join a game that is started by another player. The games start directly when both seat are taken.
- **Challenge a player:** A player can challenge another player. If the other player accepts, the game will start. The player can find other players by a search function.
- **Join a tournament:** All players can join the tournament. The games start when all seats are taken. A new tournament starts directly on a Sit 'n go basis. If the players can't be separated by points the winner is decided by some kind of tie break.
- **Resign a game:** A player can any time resign a game.
- **Play game:** A player knows that it is his turn to play, either by logging in and checking the game list or by the optional of receive an email. When a player is about to time out, he always receives a notification email.
- **Post comment:** A player should be able to chat with the opponent during the game. Other community functions like a forum may be implemented.
- **Edit personal page:** Nicknames, ratings and played games, are always public. Email addresses should not be public. Players can add other information such as real name and nationality to their page which will be public.

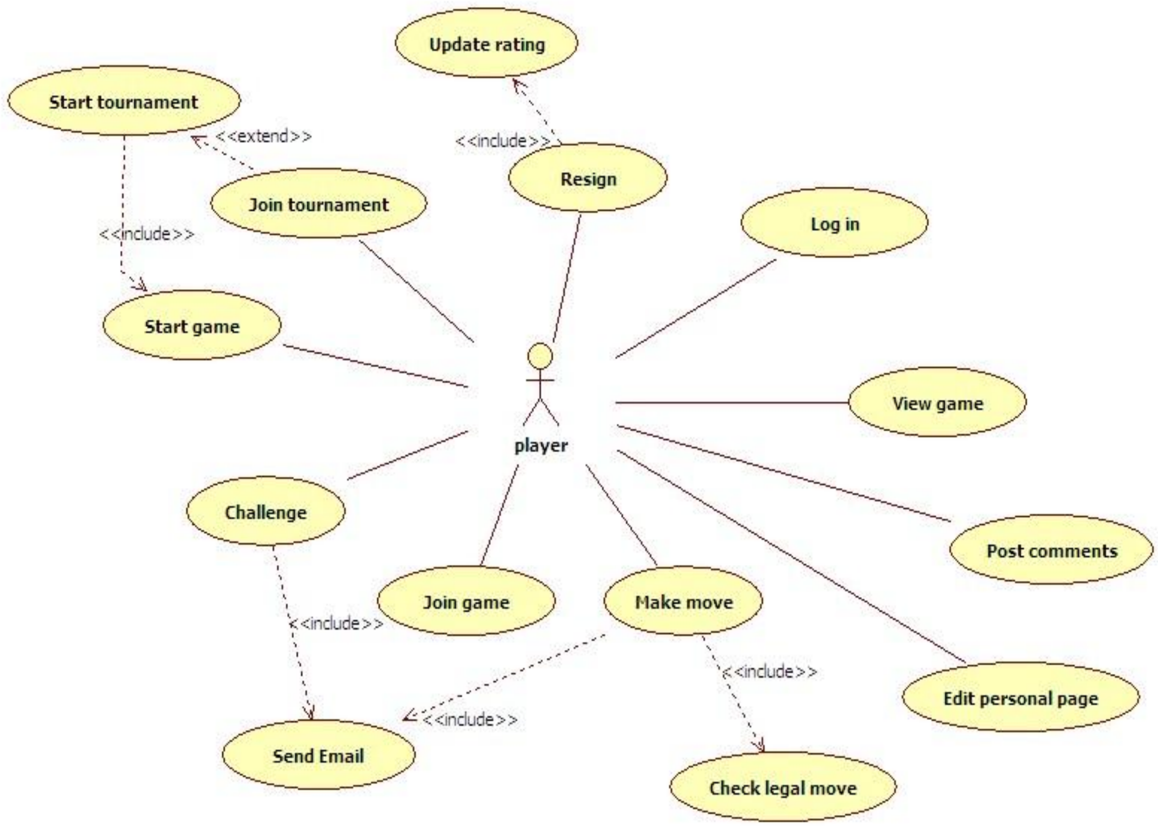


Figure4. Use case diagram for player

Actor: Administrator

- **Judge games:** If a player thinks that the opponent has made an illegal move and should lose the game, there should be a button that they can claim a winning situation. The administrator should then check the game and make a decision. This option can be omitted later when the ckecking illegal move algorithm is 100% reliable.
- **Manage users:** If there is any problem with creating or editing a user the administrator should be able to handle it. If any player violates the rules of the site, or a user wants to delete his account, the admin should be able to delete that user.
- **Manage games:** The administrator should be able to edit the moves or other information related to the game, he should also be able to delete a game in special cases.
- **Send email:** The admin should be able to send email to all or selected group of players, to keep them informed about the server

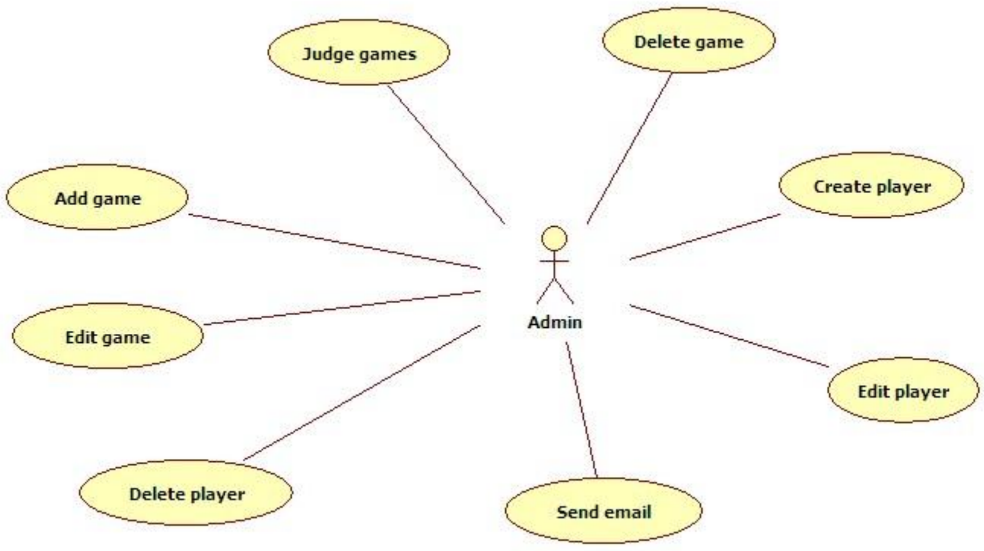


Figure5. Use case diagram for Admin actor

3.2. Non functional requirements

NFRs do not express any functionality to be implemented in the future system directly. They express behavioral conditions and constraints that must exist in system development and operation. In Web systems development, web developers are usually more concerned with functional aspects of the systems, such as the layout and structure of the pages, the navigability of the pages, and the various functionalities that the Web systems need to provide, Although, NFRs such as security, data and system integration, and system performance are crucial to the development of Web systems. Open source softwares are well known for paying attention to technical requirements because of that open source softwares have high security and high quality.

3.2.1. Usability

The website should allow Shogi players (players should have basic knowledge about Shogi) to play Shogi. The website should be self instructing and any player should be able to start their first game within a few minutes.

3.2.2. Availability

As it's not a real time playing server, availability is not a vital requirement. Players should be able to access the server whenever they want. Short down periods (a few minutes) is no problem. But if there is longer period in a way that is annoying for the users, it is not acceptable.

3.2.3. Reliability

Reliability is the probability that the software performs its intended functions correctly in a specified period of time under stated operation conditions. There should be appropriate error handling to prevent unforeseen problems.

3.2.4. Performance

All Web pages must download within three seconds during an average load, and five seconds during a peak load.

3.2.5. Supportability

The players should be able to report the problem by email when there is a failure. The website should portable in the sense of well known browsers.

3.2.6. Security

The security measures that should be considered:

- Password should be stored in encrypted format.
- Personal information is only available to other registered players. Although email is considered personal information and is not revealed. The games can be reviewed by all the registered players.
- There should be measures to stop injection attacks.
- A back up routine should run on the server.

3.2.7. Minimum system requirements

To be able to use the website a user needs to have a computer with Internet connection which can run any web browser that accepts Java script and session cookies.

4. Analysis

To move on to design it is essential to understand how the product will be used and how it must perform. After eliciting the use cases, the requirements can be clarified more with a detailed domain model. In the second part of the analysis open source methods that were analyzed for the implementation are discussed.

4.1. Domain object model

Domain object model or conceptual model describes the various entities involved in the system and their relationships. As part of the domain model, domain objects must be created that describe all the objects mentioned in use cases.

The main domain objects which are identified are player, game, tournament, board, move and pieces. Figure 6 shows the domain object and their relationships.

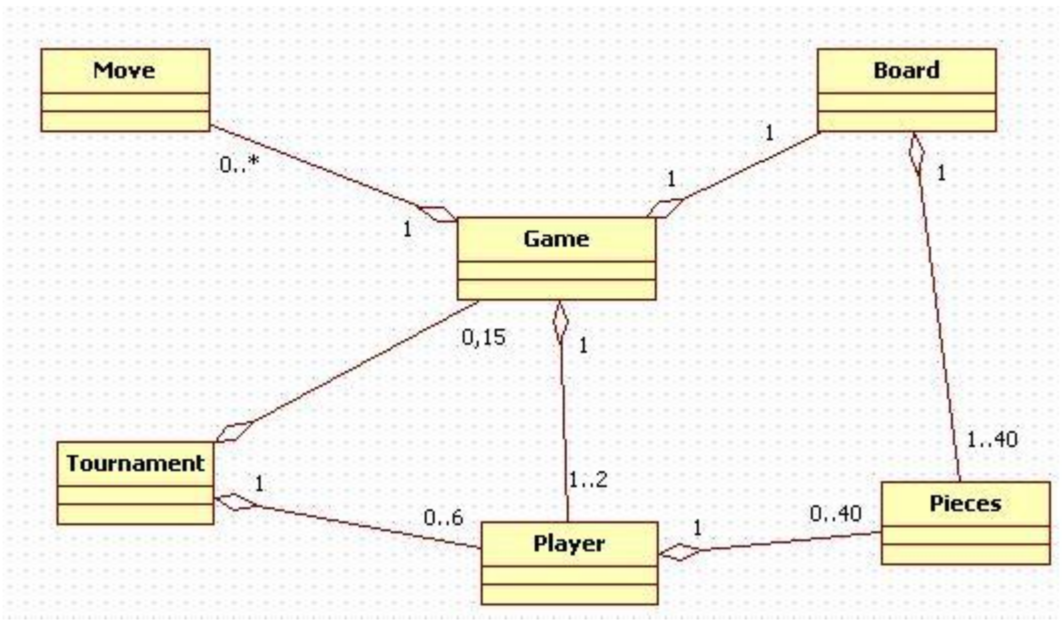


Figure6. Domain model

Every game has one board and one or two players. If a game is started by a player and no one has joined it yet there will be one player. Each player in the beginning has twenty pieces, but the pieces can be captured by the opponent or opponent pieces can be captured. When all the pieces of the opponent are captured the opponent has zero pieces and is lost. When the game is not started yet there are zero moves, after the game is started the number of moves is infinite. The tournament will start when six players have joined for that tournament. Each player then plays with all other players, which will be fifteen games.

4.2. Open source development tools

Although the first thing that comes to mind when talking about open source softwares is that they have no cost, the free software foundation intends the word 'free' to mean "free as in free speech" and not "free as in free beer" with emphasis on the positive freedom to distribute rather than a negative freedom from cost. Open source is not only about software. It is ranging from the recipes of beverages and beers to presidents and governments.

Open source programs are the product of collaboration among a large number of different programmers. Open source solutions are said to be more reliable since they typically have thousands of independent programmers testing and fixing bugs of the softwares. They are flexible also as programmers can customize them and add new abilities to them. Moreover free software pays special attention to technical (non-functional) requirements. It does not require thinking about commercial pressure that often degrades the quality of the software. Commercial pressures make traditional software developers pay more attention to customers' requirements than to security requirements, since such features are somewhat invisible. Using Open source softwares improves security, the quality of the software is higher than the costs are and there is higher reliability.

The three major web development platforms are LAMP, ASP.NET and J2EE. LAMP uses PHP. PHP is more portable than Java, since it's an interpreted language and nearly all hosts support it. To support Java, a JVM needs to be running and that adds extra complexity that many hosts don't want. That is why almost all shared hosting websites support PHP with affordable price, and it is more expensive to have Java hosting. On the other hand, since PHP code is interpreted each time it loads it will never perform as well as Java and it's not possible to catch error at compile time (there is no compile time). PHP is the only open-source server-side scripting language that's both fun and easy to learn. Anything that can be done using regular programming languages can also be done by writing PHP scripts. It is free and available for Windows, Unix, and Linux with the Apache and IIS web servers (and possibly others). Microsoft's ASP.NET is free, but has the platform costs of Windows and security weakness of IIS[8]. Recent surveys show that more than 16,000,000 Web sites use PHP as a server side scripting language, although PHP is mostly used for smaller projects, which perhaps suits its initial acronym which was Personal Home Page, but it was changed in PHP3 to stand for Hypertext Preprocessor. It is easier and faster to code in php. Java on the other hand is used for more complex or "critical" projects, generally. Using PHP when some features of Java are not needed reduces development cost, development turnaround time, and increases enhancements and maintenance.

One thing that makes PHP easy to use is that it is loosely typed i.e. variables types don't need to be declared. PHP automatically determines variables types by the context in which they are being used.

PHP can be used in LAMP framework which stands for the first letter of Linux (operating system), Apache HTTP Server, MySQL (database software), and PHP or Perl. Java based servers were also considered. The benefit with using LAMP is that it is easily configured and robust.

5. Design

High level design describes the architecture that is used. Sequence diagrams are attached as appendix. Low level design is depicted using class diagram.

In the previous section different tools that can be used for implementation were analyzed. In the following the tools that were used to design the project are described.

5.1. Different techniques

5.1.1. Implementation tools

As it is no real time and complicated activity involved, PHP is preferred before Java, even if Java might give more features and libraries. The database and web server were chosen according to LAMP solution stack. The actual server is hosted at a web hotel which supports LAMP. The main focus for the server is to make it as compatible as possible with most devices. To serve this purpose the tools that are used must be portable. It is discussed in the previous part; how portability can be achieved by using PHP and LAMP in general. PHP version 5 is used in this project. It is the newest version and it fully supports object-oriented syntax. PHP is used in combination with Apache. Requests for PHP scripts are received by the Web server, and are handled by the PHP interpreter. The results obtained after execution are returned to the Web server, which takes care of transmitting them to the client browser. MySQL 5.1 runs the database.

5.1.2. Find appropriate IDE

The project was first developed using Gedit, a text editor for Linux which supports PHP and many other languages, Gedit works pretty well. However other IDEs were also tried to find an IDE which has more programming features. The first choice was Zend studio, Zend Studio 7.0, for Linux (64 bit), was downloaded first. It was very time consuming to download and it crashed while installing, then the 32 bit package was downloaded .It was no problem to install(except that it was huge). But when the already existing project was opened in Zend, errors popped up, saying the configurations need to be done. It should also be mentioned that it was the trial version and it is not free.

The next IDE to be tried was Netbeans. It is free, download and installation took much less time than Zend and the project executed as expected in the first try. Everything worked and features such as code completing and debugging make programming much easier. It is a java application so it is more bulky than Gedit, but feature wise its better than any IDE that has been tried. It is not at all as large as eclipse and runs much faster than eclipse. The only thing that can be faster is Gedit, but once the useful feature set of Netbeans are realized it is much more desirable than Gedit.

5.2. High level design

Three tier architecture is used for the architecture of the web application. At the base of an application is the database tier, consisting of the database management system that manages the database containing the data users create, delete, modify, and query. Built on top of the database tier is the complex middle tier, which contains most of the application logic and communicates data between the other tiers. On top is the client tier, usually web browser software that interacts with the application.

- GUI(Client tier)
- Buisness layer(Middle tier)
- Database tier

Figure 7 shows the order of the layers and how they interact[9].

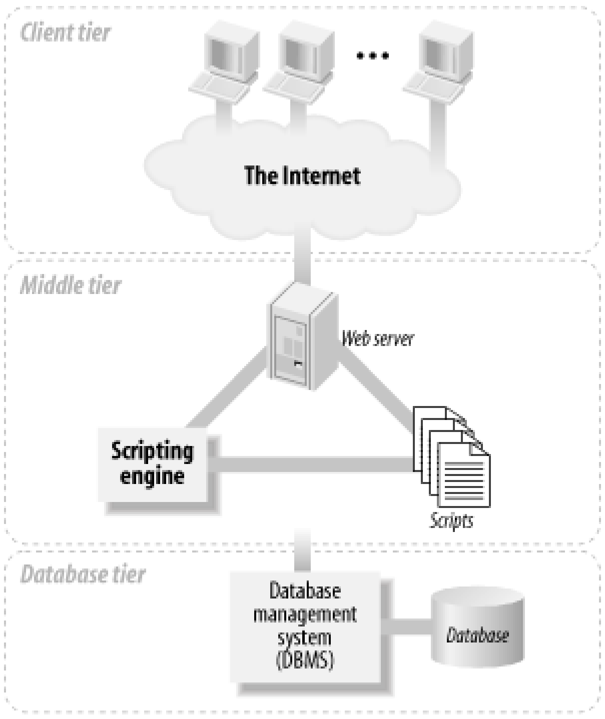


Figure7. Three tier architecture

5.3. Low level design

Low level design consists of two parts, Static Model, which will result in Class diagrams, and Dynamic Model which results in Sequence diagrams.

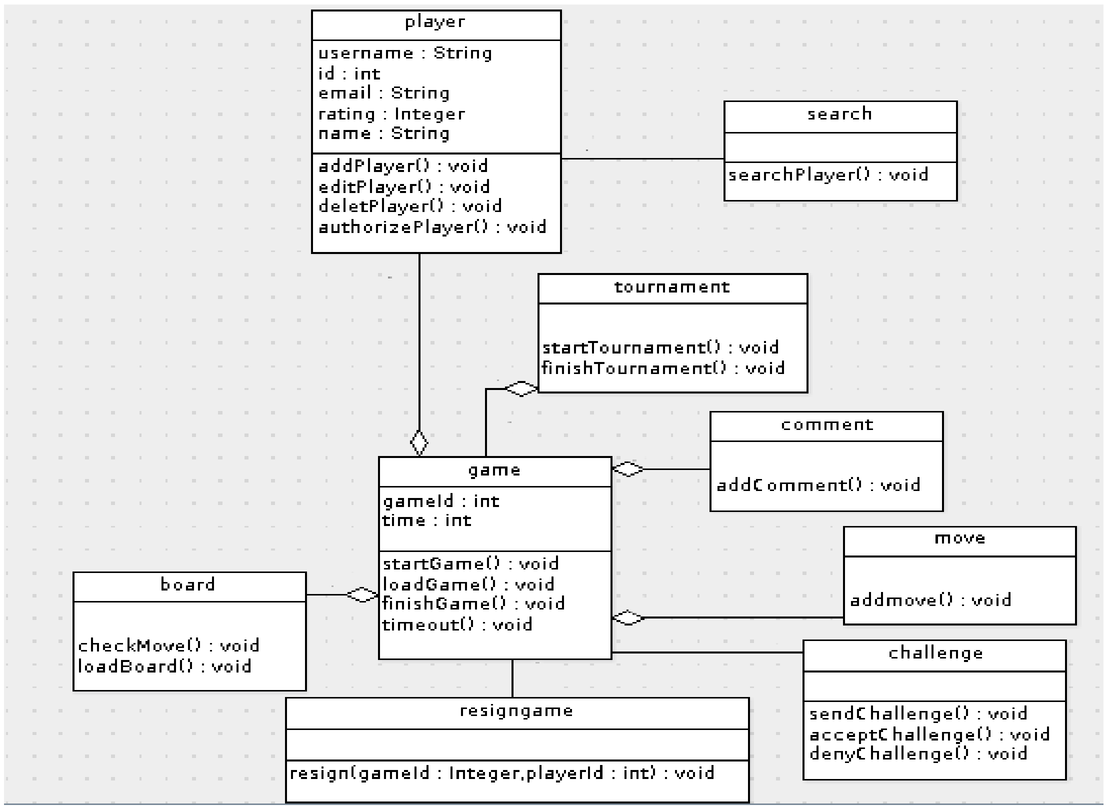


Figure8. Class diagram

5.3.1. Static model

To get the static model we should find responsibilities, attributes, and relationships for each class. Figure 8 shows the classes, their attributes and methods and their relationships. The main classes are:

- **Game**

  Attributes: Players, Time per move, Status, Type

  Responsibilities: initialize, update, authenticate

- **Player**

  Attributes: ID, Facebook ID, user name, rating, personal data

  Responsibilities: authenticate, update

- **Tournament**

  Attributes: Games number

  Responsibilities: Start tournament, Finish tournament

- **Facebook**

Attributes: Facebook ID (mostly is handled by FBML, Facebook Markup Language)

Responsibilities: communicate with Facebook

- **GUI**

  Attributes: visualize data from the board class

  Responsibilities: show board, drag and drop

5.3.2. Dynamic model

Sequence diagrams are used to show the interaction between objects. The sequence diagrams are in Appendix.

5.4. Database structure

Database tables, the fields and keys are shown in figure. The database consists of five main tables which are Users, Games, Moves, game types and Tournaments and two temporary tables, tmp_users and Ladder. For the purpose of having several type of games, game_type table was added, with a foreign key in 'games' table. As the ratings in different game should be separate a separate table for user rating was created with a foreign key from users table. Language and caption table are added for making the website multilingual, which was not in the time plan of this thesis. Figure 9 shows the tables of the database and their relationships.

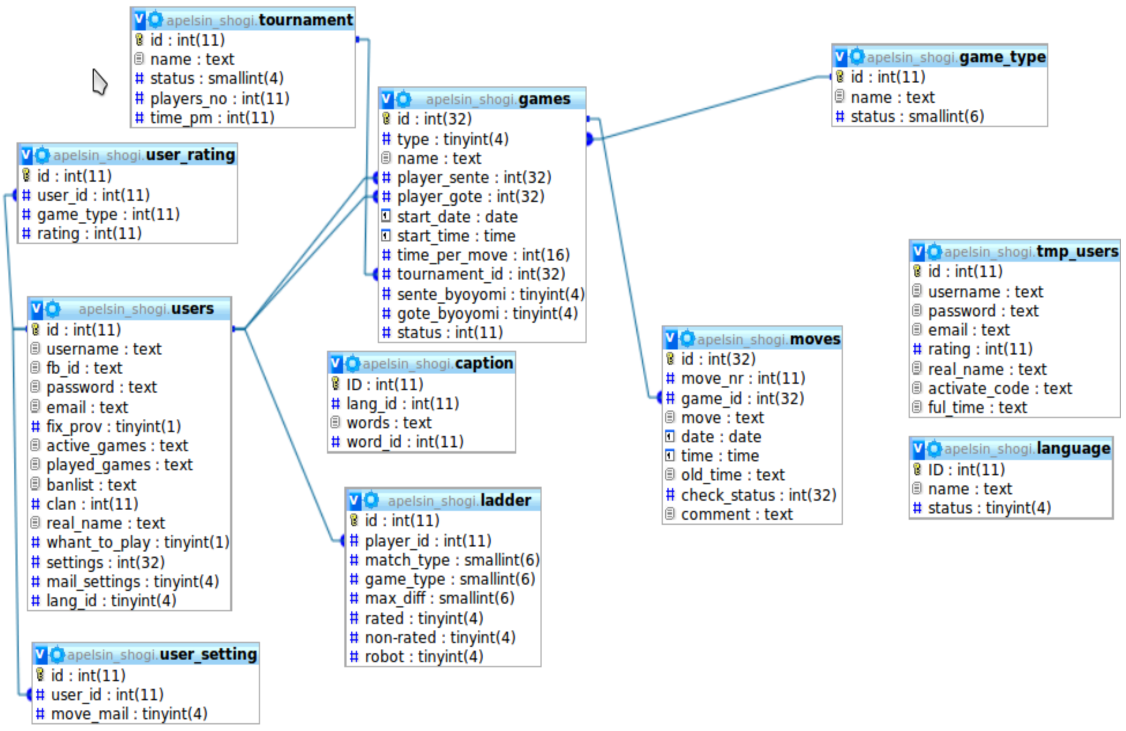


Figure9. Database design

6. Implementation

The Implementation was done using LAMP solution stack. The classes were made according to class diagram and the features were implemented as class functions. After performing acceptance test the code was reviewed to make optimizations. Following some challenges in the implementation are described.

6.1. Double submit problem

Submission of the same data more than once in a POST request is undesirable, it can happen after submitting a move (or any page that has accepted post data) if the user refreshes the page a message appears on the browser saying that by doing that the information will be resent. If the user accepts that without noticing what will happen it can cause undesired results, like multiple entries of previously entered data in database. A common way to avoid that is using Post/Redirect/Get (PRG) design pattern which avoids certain duplicate form submissions. The solution is to redirect the user to a new page after posting the data.

In PHP it is done by using the header function:

```
<? header('Location: /message.php'); ?>
```

The user is redirected from the page which contains post to a page which shows a confirmation of the submitted data. In this case by pressing refresh or back button, no harm can be done.

Another good thing with using header function is that it prevents header injection attacks. Header injection attacks happen when users receive an invalid redirect, which is made by a malformed HTTP request, when accessing the web site. The header function prevents more than one header to be sent at once as a protection against header injection attacks.

There are some difficulties in using header function. The most common is to get the warning 'header already sent'. To prevent that one must make sure no outputs are sent before header function. It can sometimes be tricky to find what's wrong. It may take time to realize the warning is for a seemingly innocent space or blank line before the PHP start tag or in this case after the end tag.[4]

6.2. Drag and drop

The GUI is made in Javascript, not so difficult but it took a while before it worked well.

6.3. Make legal move check

Here some standard techniques from western chess software are used. The drop pieces rule adds to the complexity of legal move checking. Taking the king and resigning are the game ending conditions.

6.4. Facebook application

The application was added as a Facebook application using IFrames. Facebook has pretty well documented classes for that purpose.

6.5. Implement security

The following practices were implemented to establish an acceptable level of security:

- HTML- and SQL-injection, are pretty common problems, so I just did implement some well proven classes. The most important is to check every data that can be entered by the user.
- prevent session hijacking by using session_regenerate_id.
- Store the hash sum of password rather than clear text.
- Prevent cross site scripting attack using PHP's html Special Char function which checks user-submitted data for HTML tags.

6.6. Implement cookie

After a successful login a session is set for that user, the user can visit different pages of the website with that session and a PHP class checks before opening every personal page that the session is set for the user. After clicking on Logout button the session is unset. If the user closes the browser without logging out, the session will expire after a certain time. If the user is idle on the site after a certain time the session will expire. But for saving the user data between different visits, cookies can be used. Cookies are pieces of data which are saved on the user's computer. If the user checks the 'remember me' option a cookie will be set for her. If she restarts the browser without logging out, she doesn't need to enter user name and password for the next visit, the website will redirect the user to her index page. However if the user presses logout, both the session and the cookie will expire.

6.7. Sending email when little time left

Gameknot[6], a correspondence Chess server, has a service which sends email when there is little time left for making a move. A possible way is to call a function that checks for the time left, from many places in the code. If there is no activity on the server, no code will be executed and no message will be sent.

7. Testing

In agile development acceptance testing is started from early stages. It should be repeated at the end of each iteration. This helps to reduce the cost of fixing the defects. For functional testing the test was mainly done by the users. It was tried to have a working prototype so that the players can continue playing and give feedback. If a use case was fulfilled without problems new use case development started. As there was only one person for developing the project, the input from users was valuable and helped a lot to determine a functional design.

Non-functional testing, such as testing the security, performance and usability of the system should be done by a specialist. For security testing appropriate hacks should be tested. The system was not vulnerable to SQL injection attack. Performance can be tested by checking how many users can send request to the website at the same time. Usability can be tested by observing the users while they are using the website. Observing how they navigate to different pages and how much time does it take for them to find the link they are looking for, can be a great feedback.

8. Time plan

After eliciting the requirements and gathering the features that should be implemented the coding should be started. The first version was very simple user interface, people could manually type the move in a box, like entering 7776, which is similar to the way people played correspondence chess before using Internet. No legal move check was implemented. This was the first prototype. In this way ten players played on the site and finished about 20 games. The feedbacks were gathered. Next step was to add a drag and drop, as this was the most asked function from the users. According to FDD time plan there should be a time limit to implement each feature. In this project it was one week for each feature. The features were then tested and if the users weren’t satisfied the feedback should be considered in the next iteration. Otherwise the feature is added to the final build.

9. Conclusion

Making decisions is an inevitable part of a software development process. Decisions should be made to choose appropriate development process. There is no right development process that can be used for all projects. The right process should be determined based on the characteristics of the project. For this project a balanced solution was needed that has agile process characteristics specially for testing but also focuses on documentation. FDD fits well for the project but still some changes had to be done to tailor it for this specific project. It is the same when it comes to deciding about choosing framework, programming language, database etc. Although having knowledge about different solutions makes it easier to make a decision, often a thorough search is needed before choosing one solution.

It was decided to use open source development tools in this project as they are more trusted and of course free. Many people contribute to make quality open source softwares and there are many communities and documents available to help. Furthermore open source softwares focus more on non functional requirements which means the quality of the software is considered highly as well as functional requirements.

For the coding phase PHP was chosen to be the programming language. It is part of the LAMP solution stack. It is fun and easy to learn. It is fast to develop with PHP as most functions which are needed to develop a web application are already made. Therefore it is faster and cheaper to use PHP.

Testing the application according to agile testing, start acceptance testing from early stages of development, gives valuable feedbacks which help to improve the software according to the users' needs.

References

1. Atzeni et al, 1998, Baresi et al, 2000, Bonifati et al, 2000 and Rossi et al, 1999

2. http://www.redhat.com/magazine/003jan05/features/lamp/

3. http://www.bredemeyer.com/

4. http://www.tech-recipes.com/

5. http://gameknot.com/

6. http://www.featuredrivendevelopment.com/

7.http://www.syllogisticsoftware.com/papers/Web_Development_Technology_Comparison.html

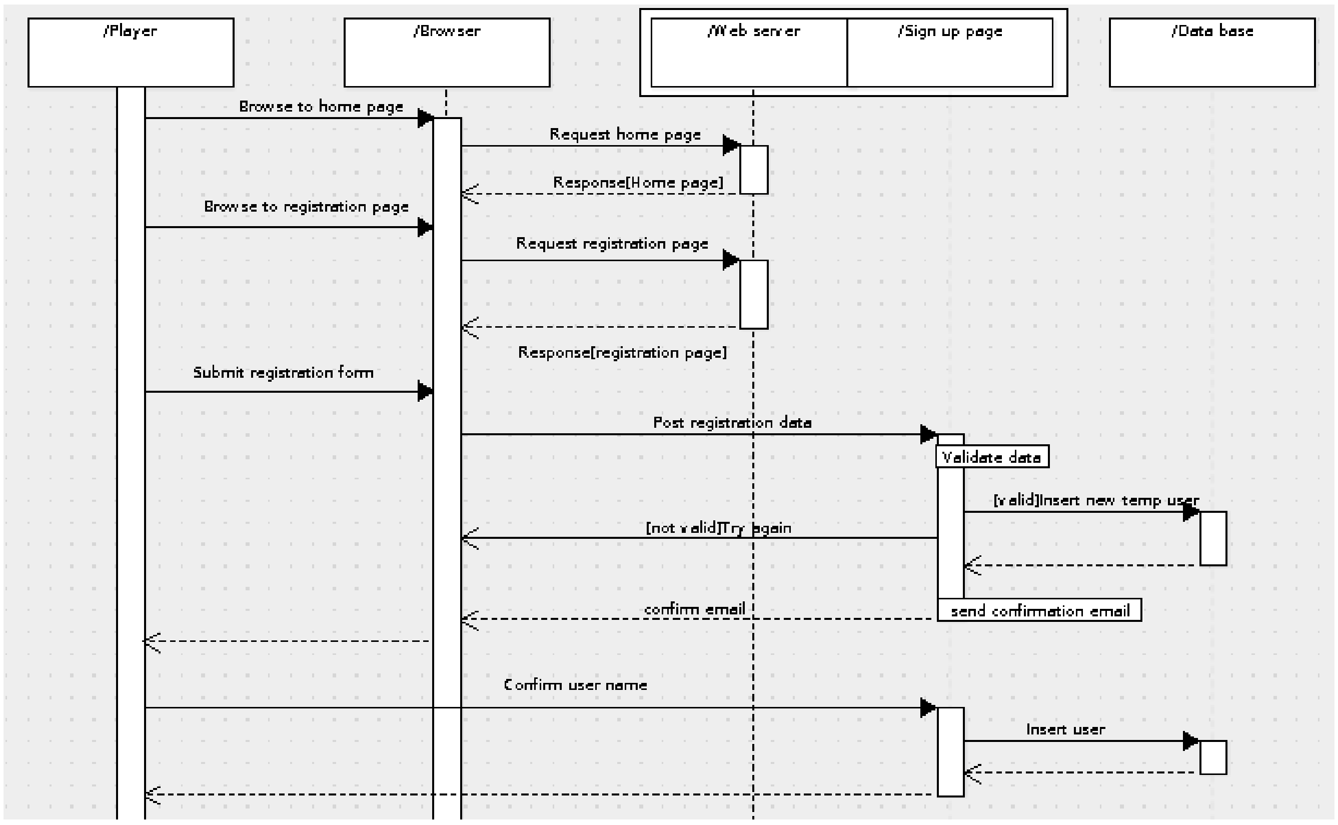
8. http://www.msversus.org/microsoft-net.html

9. http://book.opensourceproject.org.cn/lamp/mysql/phpmysql/opensource/webdbapps_snode13.html

Appendix A. Sequence diagrams

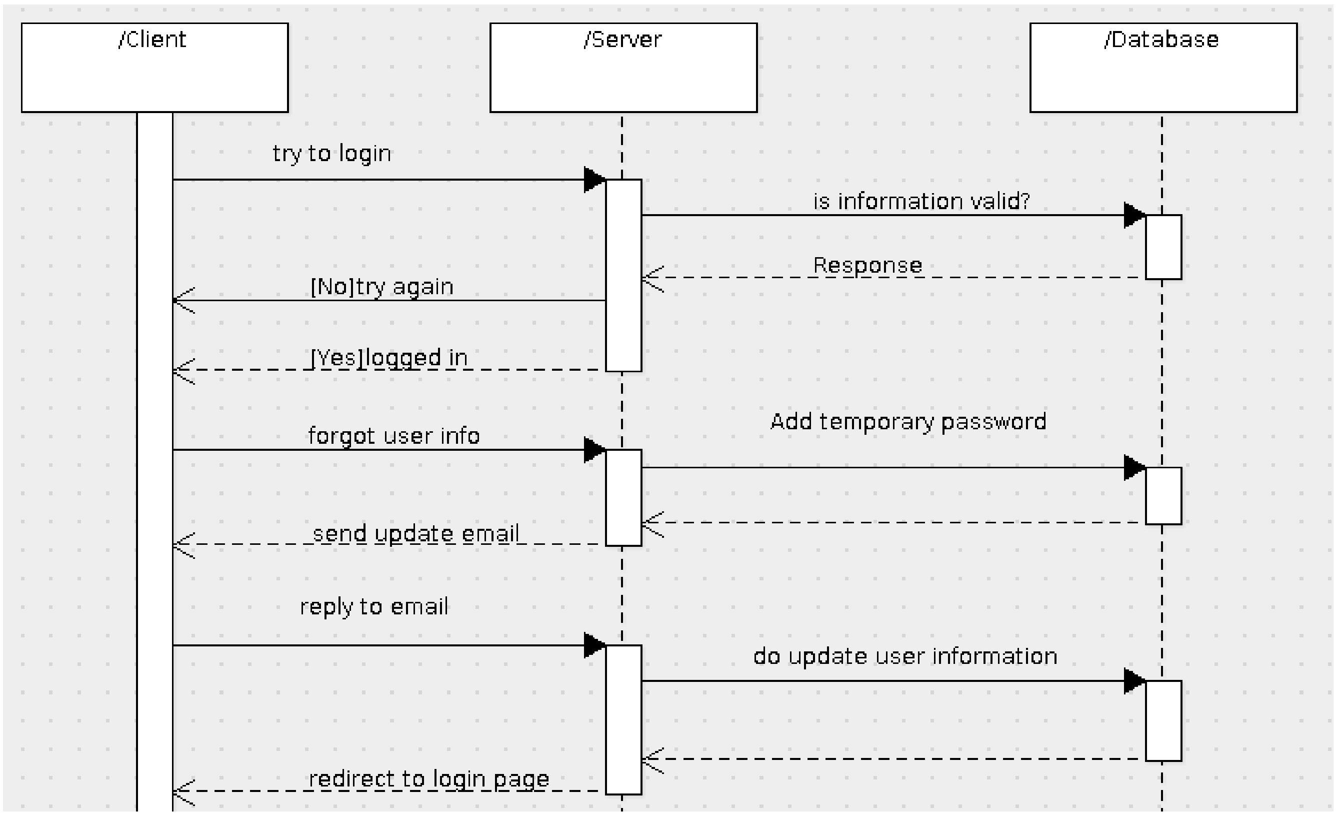
Sign up

Non-registered user opens the registration page and fills the data and submits it. The server validates the data. If the data is valid a confirmation email is sent to the user. After the user confirms the registration, the user is added to the users table



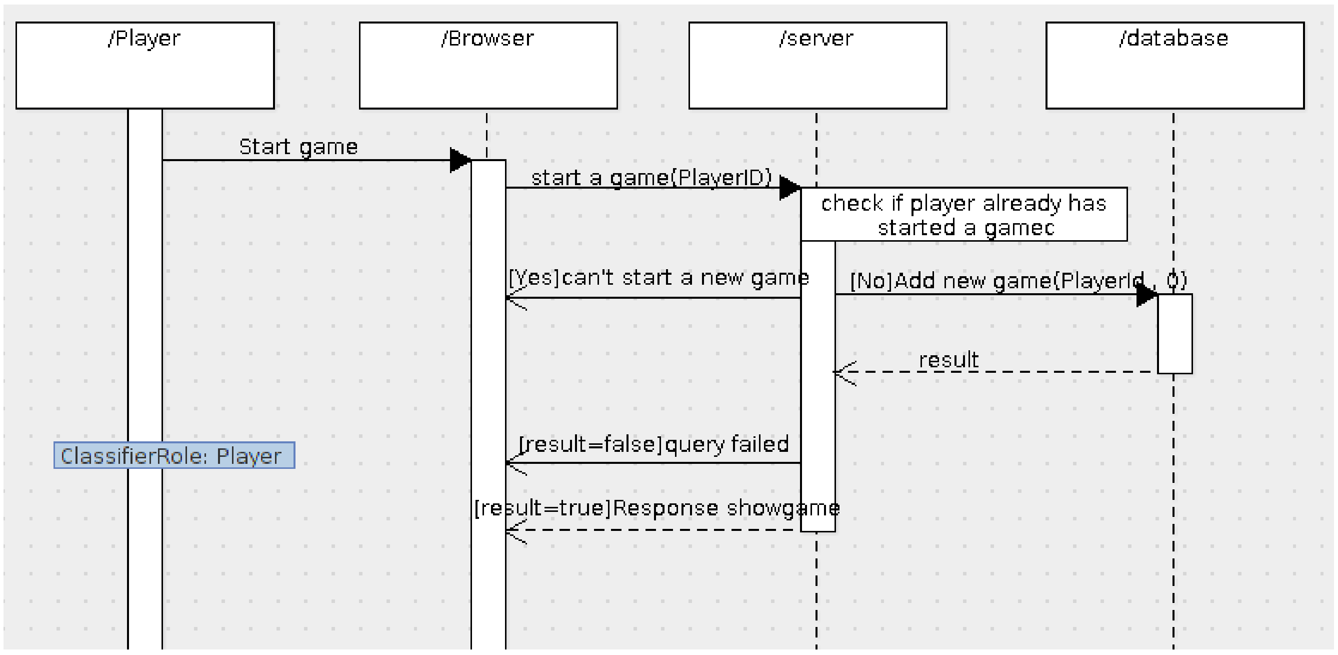
Login

A player tries to login by entering username and password. The server checks with the database to authenticate the user. If the user is authenticated he is logged in.



Start game

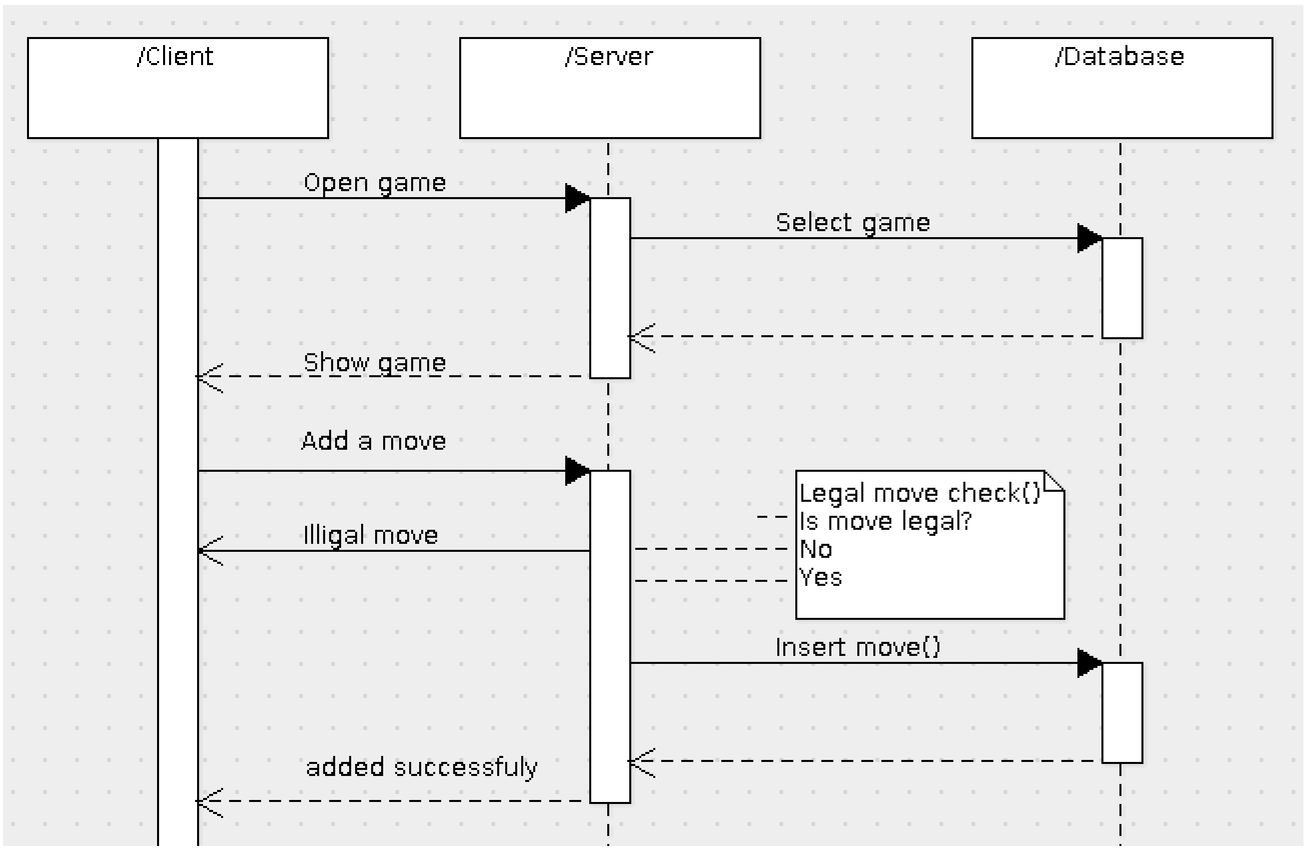
The player submits a ‘start game’ request. The server checks if the player has already started a game. If not a game is added to the database.



/Player
/Browser
/server
/database
Start game
start a game(PlayerID)
check if player already has started a gamec
[Yes]can't start a new game
[No]Add new game(PlayerId , 0)
result
ClassifierRole: Player
[result=false]query failed
[result=true]Response showgame

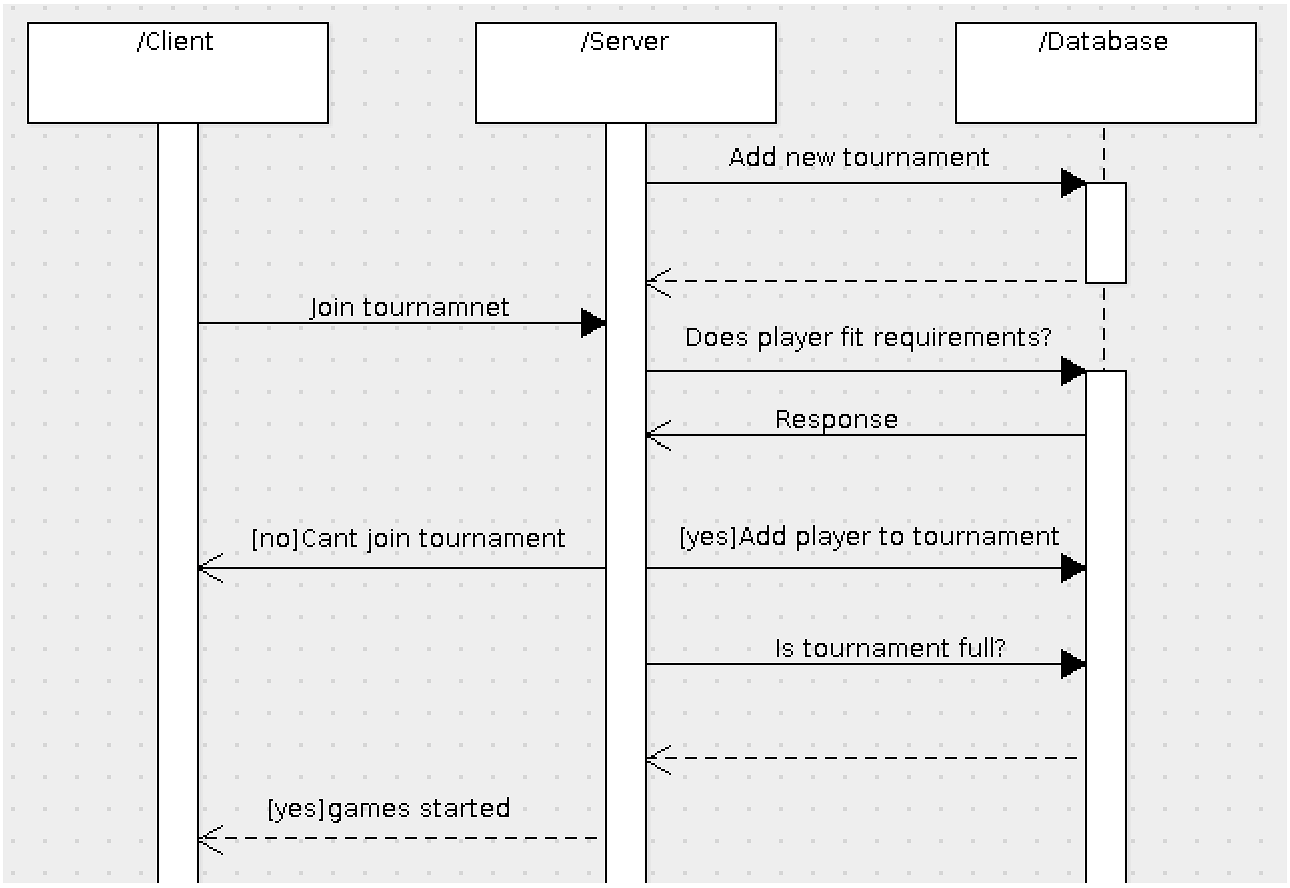
Add move

The player makes a move using the GUI. Server checks if the move is legal. If it is the move is added to the database.



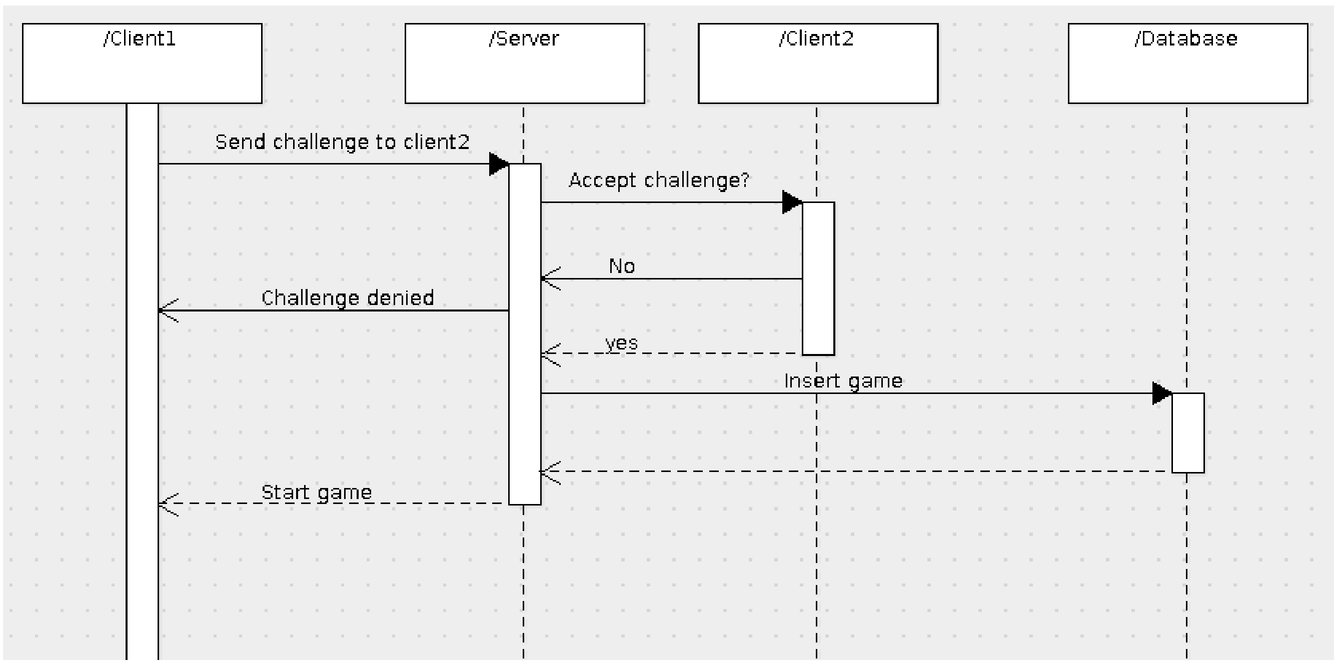
Join tournament

A new tournament is added to the server. Players submit that they want to join the tournament. If the player fits the rating limitation of the tournament he is added to the tournament. If the tournament is full the games will be started.



Challenge

Player1 submits a challenge with player2. The server sends an email to client2. If player2 accepts the challenge the game is started.


/Client1
/Server
/Client2
/Database
Send challenge to client2
Accept challenge?
No
Challenge denied
yes
Insert game
Start game