Project planning

Individual project

Sam Philipsen | S3-CB02

Versioning table

V0.1	Creation of document. Added title, versioning table and table of contents.
V0.2	Added backlog which includes user stories 1-6 and some user requirements.
V0.3	Added more user stories and divided them into smaller stories.
V0.4	Divided big user stories into smaller ones.
V1.0	Added test cases and introduction.
V1.1	Added C4 diagram (to C3).
V1.2	Added applied research section.
V2.0	Finished applied research section, updated C4 diagrams, added design decisions.

Table of contents

Introduction

Online gaming has taken the world by storm in the past decade. New games seem to be created out of thin air, and the consumer can be overwhelmed by the wide variety and complexity of these games. In times like these, some prefer to step away from the mainstream gaming market and go to a more familiar space. This space being card games. The familiarity and easy to learn rules of card games are sure to bring in tons of users.

On top of that, the convenience of a website-based card game service makes this one of the easiest games to get into.

Backlog

User stories

User story 1 (Story points: 1 week) – Priority 9
As a player
I can play a game of coin toss
So that I can have fun

Acceptance criteria:

- The game starts when two players are in it.
- The game randomly assigns a player to one of the two sides of the coin.
- Once the coin has been tossed, the player who was assigned to the winning side wins the game.
- The game will not start until two players are in.

User story 2 (Story points: 1 week) – Priority 5
As a player
I can bet on myself
So that I can potentially earn more points

Acceptance criteria:

- The user can not bet any more points than he/she owns.
- If the game has a minimum bet set in place, the user has to at least bet that amount of points.
- The user's points that they bet are deducted from the account.

User story 3 (Story points: 1 week) – Priority 5
As a player
I can win a game
So that I can win more points

Acceptance criteria:

- The amount of points the user has bet, and the bet of any other user that lost is added to the winning user's account.
- Only the winning user gets any points.
- After the winner gets the points, every user is thrown out of the game and the game is deleted.

User story 4 (Story points: 1 week) – Priority 5
As a player
I can lose a game
When this happens, I lose points

Acceptance criteria:

- Any points that the player has bet is gone.

- The player, and any other players in the game are thrown out of the game and the game is deleted.

User story 5 (Story points: 3 weeks) – Priority: 9
As a website member
I can join a game
So that I can play the game

Acceptance criteria:
- The member can not join the game if it has already started.
- The member joins the game.
- The user count in the game goes up.

User story 6 (Story points: 1 week) – Priority: 7
As a person visiting the website
I can see a list of available games and their status (open/closed)
So that I can look for a game to join.

Acceptance criteria:
- A list of games is displayed with the amount of people, the type of game and the host name.
- A game is displayed as ‘blocked’ if it is full.
- A guest (not logged in user) can not interact with any of the games.

User story 7 (Story points: 3 weeks) – Priority: 5
As a website member
I can host a game
So that I can play a game with my own preferred settings
Acceptance criteria:
- The website displays a menu where the user can choose their own settings, such as the type of game, number of players and the minimum bet.
- The host can see the players in the game and remove them if they want.
- The website opens a game with the chosen settings.

User story 8 (Story points: 5 days) – Priority: 5
As a person visiting the website
I can register an account
So that I can use the full functions of the website

Acceptance criteria:
- The website checks if the user can create an account by checking e-mail availability.
- The website adds an account to the database with relevant information, like e-mail, username and password.
- The person gets logged in with their new account.
- The new member can now log in/log out of the website.

User story 9 (Story points: 2 weeks) – Priority: 4
As a member
I can edit my account information
So that any information that needs to be changed gets changed

Acceptance criteria:

- The website checks if the chosen information can be changed.
- The website changes the information in the database.
- The website displays the new information on the account page.

User requirements

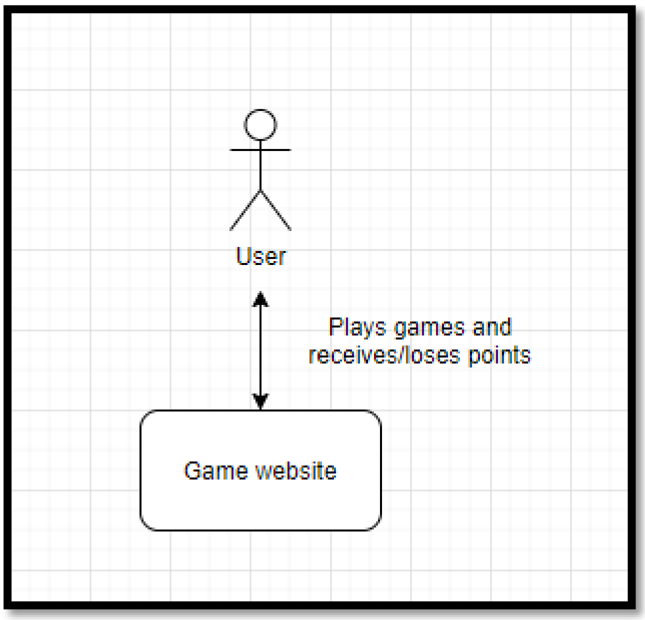
Number	Details	Priority
UR-1	**Users** can register an account to the website	100
UR-2	**Users** can log in to their accounts on the website	100
UR-3	**Users** can view their account information on an account page	25
UR-4	**Users** can change their account information on an account page	25
UR-5	**Users** can participate in a game on the website	100
UR-6	**Users** can create a game	50
UR-7	**Users** can change certain settings inside a game they have created	25
UR-8	**Users** can place bets inside the game	25
UR-9	**Users** can view their scores on a leaderboard page	10
-----------	**--**	------------
UR-10	**Administrators** can delete games	45
UR-11	**Administrators** can edit game settings	20
UR-12	**Administrators** can retrieve data like the amount of points earned on each game	75
-----------	**--**	------------
UR-13	**Guests** can view games	35
UR-14	**Guests** can register an account	50
UR-15	**Guests** can view other people's account information (limited to non-sensitive details)	10

Test plan

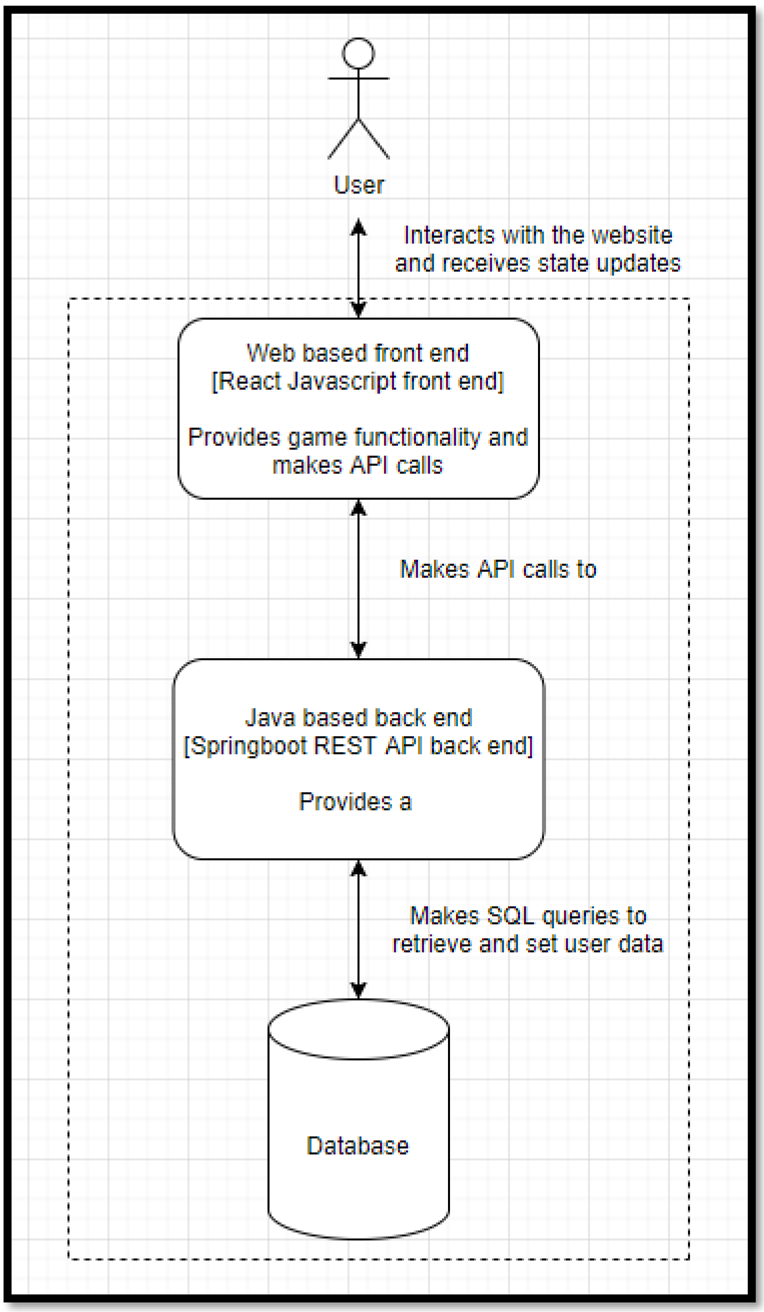
ID	US	Name	Pre-condition	Test data	Expected result
TC-1	US-1	Playing a game of coin toss	User is in the game and has decided to play	User's name: Peter Password: 123 ID: 0 Points: 100	User choses a side of the coin and waits until the game starts.
TC-2	US-2	Betting on myself to win a game	User has bet 50 points on himself to win.	User's name: Peter Password: 123 ID: 0 Points: 100	User either receives double his bet back, or loses his entire bet based on if he won or lost.
TC-3	US-3	Winning a game of coin toss	User has won the game.	User's name: Peter Password: 123 ID: 0 Points: 100	User receives double their bet back, if they put a bet in, and is then thrown out of the game.
TC-4	US-4	Losing a game	User has lost the game	User's name: Peter Password: 123 ID: 0 Points: 100	User loses their bet and gets thrown out of the game.
TC-5	US-5	Joining a game	User is logged in and on the game-list screen	User's name: Peter Password: 123 ID: 0 Points: 100	User has clicked on a game that they want to play, and the website is now changed for them to be in the game.
TC-6	US-6	Looking for an available game to play	User is on the game-list screen		User or guest can look through the list and see what kind of games are offered.
TC-7	US-7	Hosting a game for people to join	User is in the host settings menu screen	User's name: Peter Password: 123 ID: 0 Points: 100	The user is presented with options for the session they want to start. These options are: type of game, maximum number of players and the minimum bet required. After choosing the preferred settings, the user can start the game.
TC-8	US-8	Registering an account	Guest is on the register page		The guest has to fill in the options of e-mail, password and name. The system will check if the user does not exist already. If not, an account is created.
TC-9	US-9	Editing account information	User is on the account information page	User's name: Peter Password: 123 ID: 0 Points: 100	The user sees the editable fields (name, password and email) and can change them. Upon clicking 'save' these changes are saved.

C4 diagrams

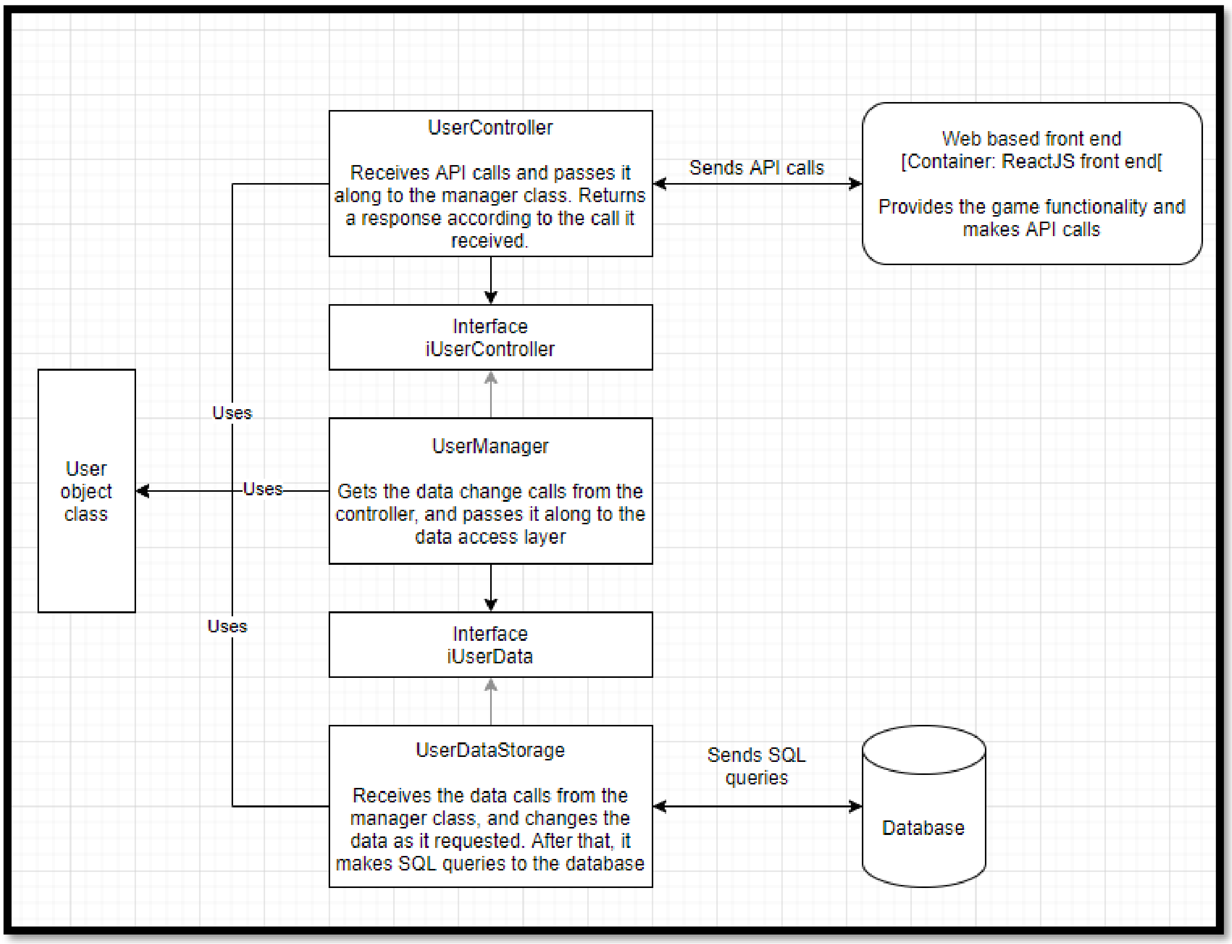
C1



C2


User
Interacts with the website and receives state updates
Web based front end
[React Javascript front end]
Provides game functionality and makes API calls
Makes API calls to
Java based back end
[Springboot REST API back end]
Provides a
Makes SQL queries to retrieve and set user data
Database

C3


UserController
Receives API calls and passes it along to the manager class. Returns a response according to the call it received.
Sends API calls
Web based front end
[Container: ReactJS front end[
Provides the game functionality and makes API calls
Interface
iUserController
Uses
User object class
Uses
UserManager
Gets the data change calls from the controller, and passes it along to the data access layer
Uses
Interface
iUserData
UserDataStorage
Receives the data calls from the manager class, and changes the data as it requested. After that, it makes SQL queries to the database
Sends SQL queries
Database

Design decisions

As seen in the C4 diagrams above, the system is currently built up like the following:

The user can interact with the web based front end service. These interactions are planned to be logging in and playing games, though more might be added in the future. Once the user tries to log in to the system, a HTTP request with the filled in user details is created. This request is sent to the Spring boot API. The request is received by the controller class, which then tries to get the required information from the manager class. It does this through the iUserManager interface class, as to avoid every program in the system from depending on each other.
The manager class then tries to get the requested data from the data access layer, also going through an interface in the process. The data access class gets the requested data (the user's information) and it all gets passed back to the API controller class, which sends the data to the web service. Once the web service has this information, it can now display it and store it for further HTTP requests.

Why dependency injection?
The reason the manager and database access layer have interfaces is because of dependency injection. If I'd want to test a single method in one of the classes, it would need to load every class just for testing a single method. To prevent this, these classes are connected to an interface. This means that I can put a fake data class instead of a real one so it does not have to load the real one each time.

Why spring boot?
Spring boot is used because it is an easy to use, but still very functional way to make the web API work. The required setup and learning curve is minimal compared to other popular web frameworks. It is also very easy to connect to the rest of my application.

Applied research

Problem:
How do I plan on storing sensitive data about users to easily retrieve from the API?

Main question:
What database programming language is best used for storing standard user information (name, email, password, id etc.) in my project.

Sub questions:

1. Does storing user data need a relational database, or non-relational database?
   Research methods:
   - Community research
     By looking at what other people have done already, I can see what they did and compare it to what I'm doing. This way I know the advantages and disadvantages of the different types of relational databases with the datasets that look like mine.
   - Data analytics
     Measuring and researching my dataset will give me more information on what it needs and doesn't need. If the things it needs match the advantages of a certain database type, then it gets me closer to my decision.
   - Domain modeling
     Similar to the previous method, this will give me more information about my dataset and see what database type gives the most advantages.

2. What kind of tables does a database need for storing user data?
   Research methods:
   - Decomposition
     By breaking my software system and data down into smaller pieces, it gives me a clearer overview of what I'm going to have to do with my data. This way I can already start on the normalizing of the dataset and determine the tables faster.
   - Domain modeling
     This will make me get a better understanding of how the database should look like, which in turn gives me a good overview of what tables the database will need.

3. What is the easiest and most secure database language to transfer data from?
   Research methods:
   - Best good and bad practices.
     Looking at what other people have done gives me a good idea on how the methods they used work. Using this knowledge, I should get a good idea on what the better database languages are.
   - Document analysis
     To get a better understanding of the most popular database access languages, I can read through their documentation. Once I know more about these languages I can make a well built up reason for picking the right one.