Software Design Document

Project Name: Ball on the Wall	Team ID: 30
Team Members: Jan van Zwol, Vo Nhat Minh, Tran Duc Duc, Marjolein Bolten, Ho Hoang Phuoc, Daan Velthuis	Mentor(s): Puru Vaish & Venelina Pocheva

1. Introduction

The product we are designing is an interactive augmented reality game where a virtual game is played using real world inputs. The game will consist of a projected image on a wall where there will be targets that the player needs to hit. The player can hit these targets by throwing a tennis ball at the wall that the image is projected on. This tennis ball will be tracked by two cameras. When the tennis ball hits the wall the Pi will use the cameras to pinpoint the location of impact (it also uses the cameras to notice when the ball has hit the wall) and use that information for the gamelogic.

2. Functional/Non Functional Requirements

Functional requirements

- Users can register a new account and log in to the system. *(1- Security by Design Checklist)*
- The system can detect when and where a tennis ball hits the wall and objectives.
- The system can control the projector to update the game situation correctly in real time
- Users are kept updated with the current score while playing.
- The system can update the online scoreboard through the Internet.

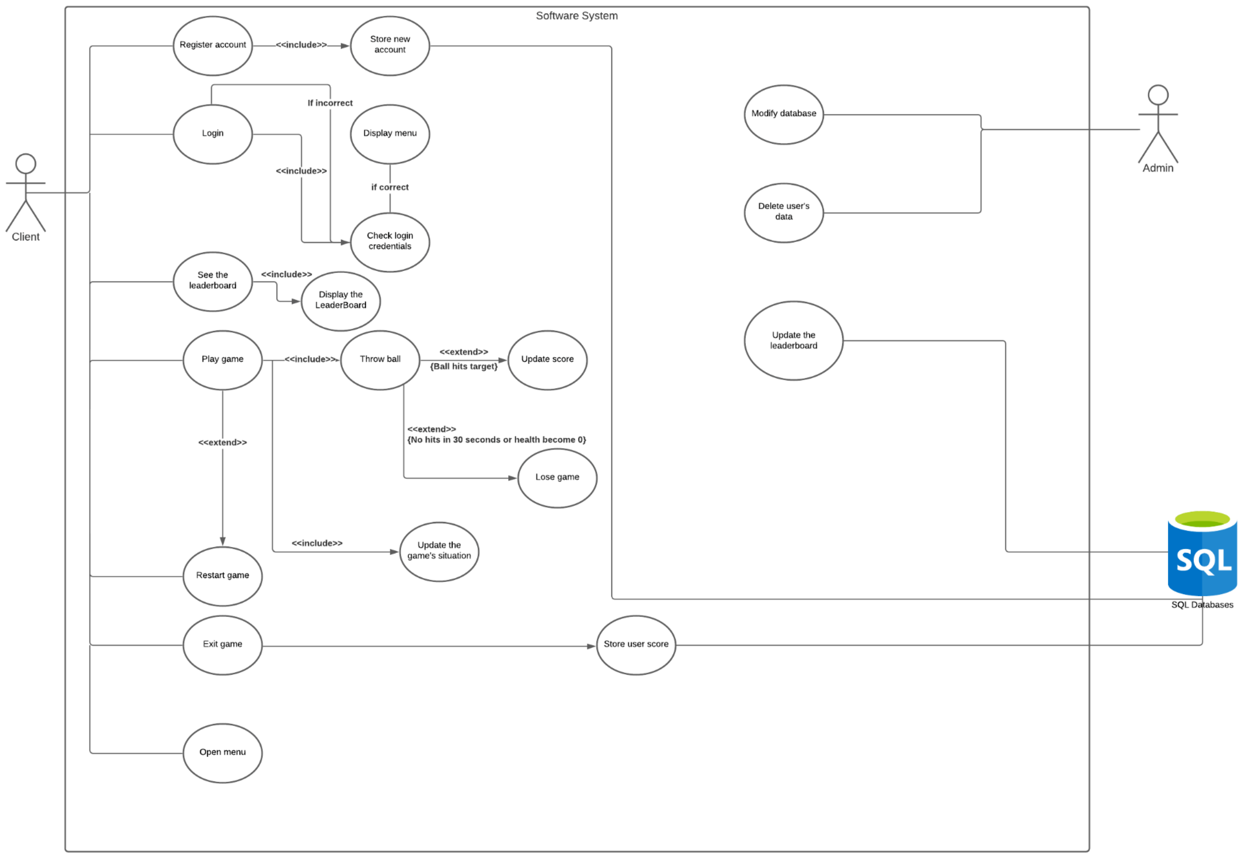
Non functional requirements

- The system will be able to process user data (login credentials, score) no longer than five seconds.
- The user interface should be user friendly with pleasing interaction.
- The updating game situation's delay should be less than half a second.

3. Architectural Design

In this section we will show our architectural design with the help of five UML diagrams: Use Case Diagram, Class Diagram for database, Class Diagram for game, Ball tracking activity diagram, Account creation activity diagram.

Use Case Diagram:

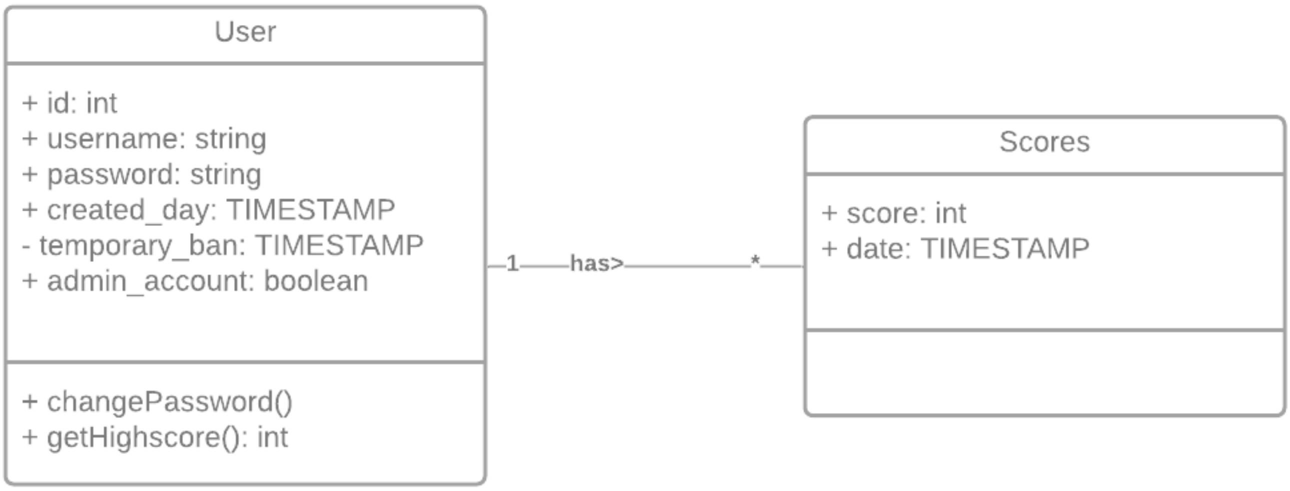


For the use case diagram, we have shown all basic use cases in our system presenting the relation between three object users, admin and database with the software system. Some notable use cases are:

+ Login/Registration: for users to create and use accounts to play games. User account and hashed password will be stored in Database
+ See/Update leaderboard: The list of user's top scores will be stored and updated constantly to the database and the user has an option to see them.
+ Play game: In play game option, the user will mostly interact with the ball and projector. Therefore the software needs to detect the hitted point of the ball to update the game's situation and score accordingly.
+ Exit game: Users have option to exit the game whenever they want and the highest score will be stored
+ Open menu: Users can open the menu to select the difficulty of the game, change landscape, time,...
+ Modify database: This option is only available to the admin user to reset or update the game

+ Delete user: This option is only available to the admin user to delete or ban invalid user accounts

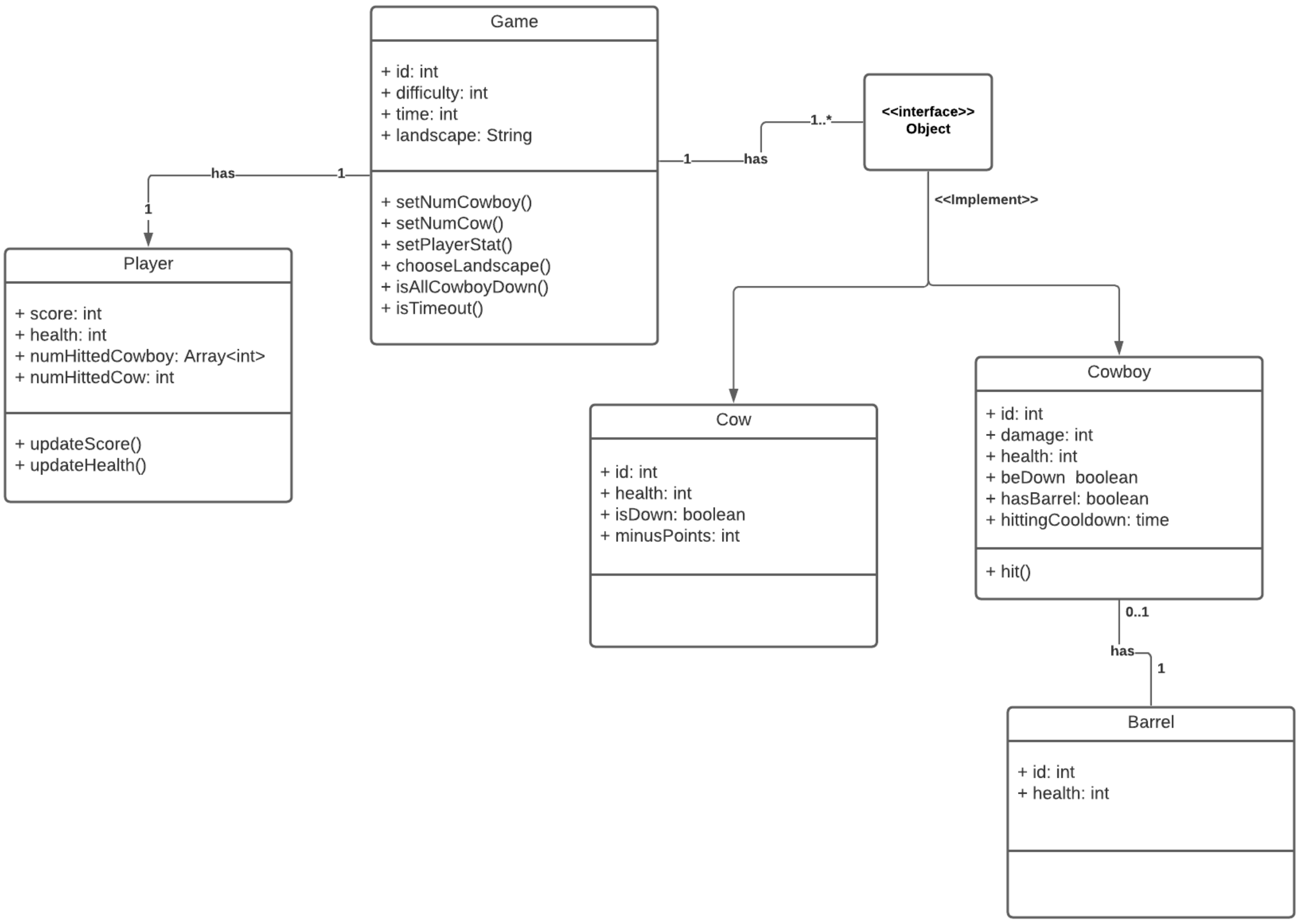
Class Diagram for database:



For the database class diagram, we have two classes:

+ Users: We will store the user's username and hashed password. Each user will have a unique id. Users will be banned for ten minutes if they type their password wrong three times, so we need to store when the temporary will end. Users will have the option to change their password.
+ Scores: There is a one-to-many relationship between the users and the scores. We store the value of the score as well as the date of the moment the score was acquired.

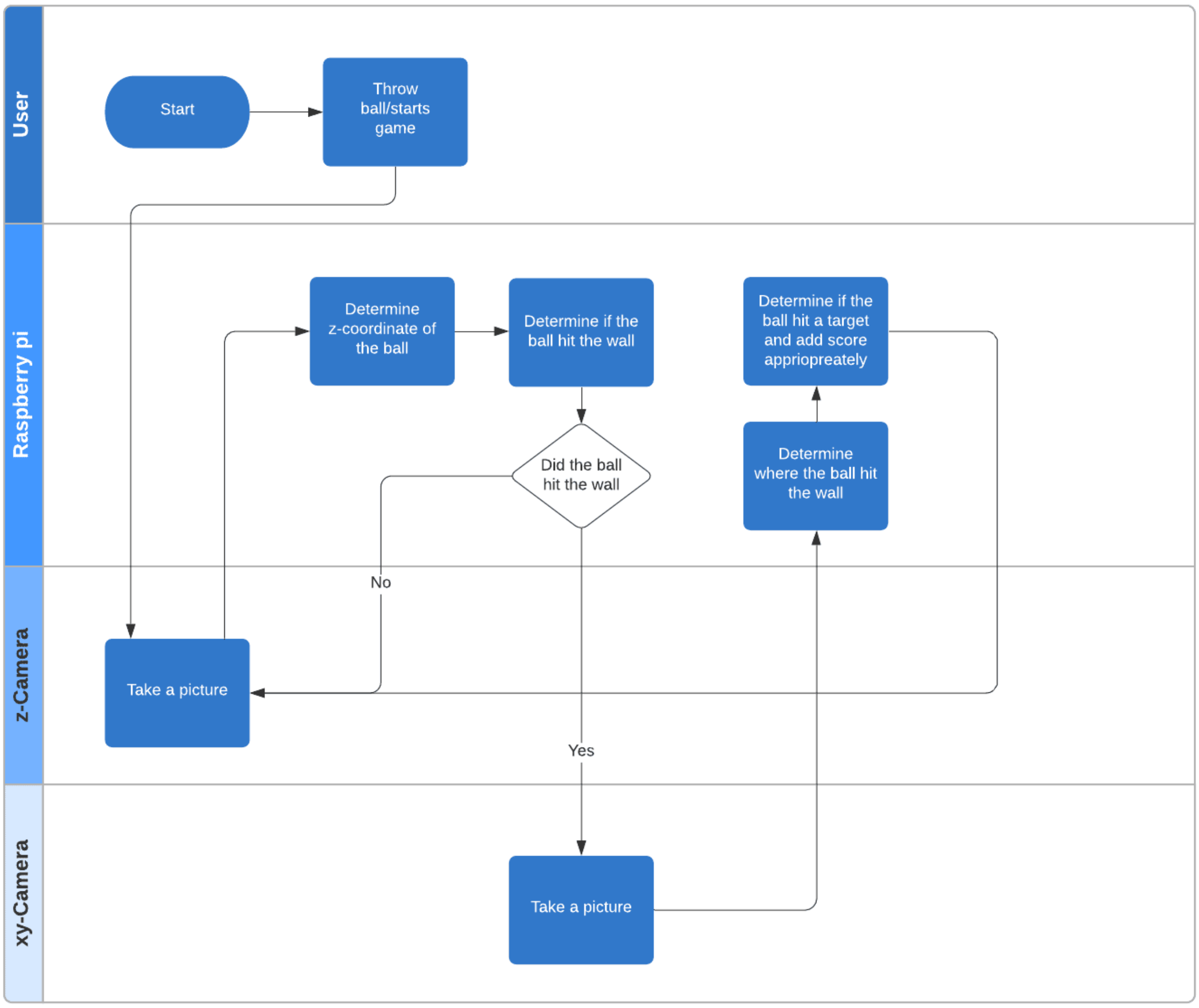
Class Diagram for game:



For the game class diagram, we have five classes:

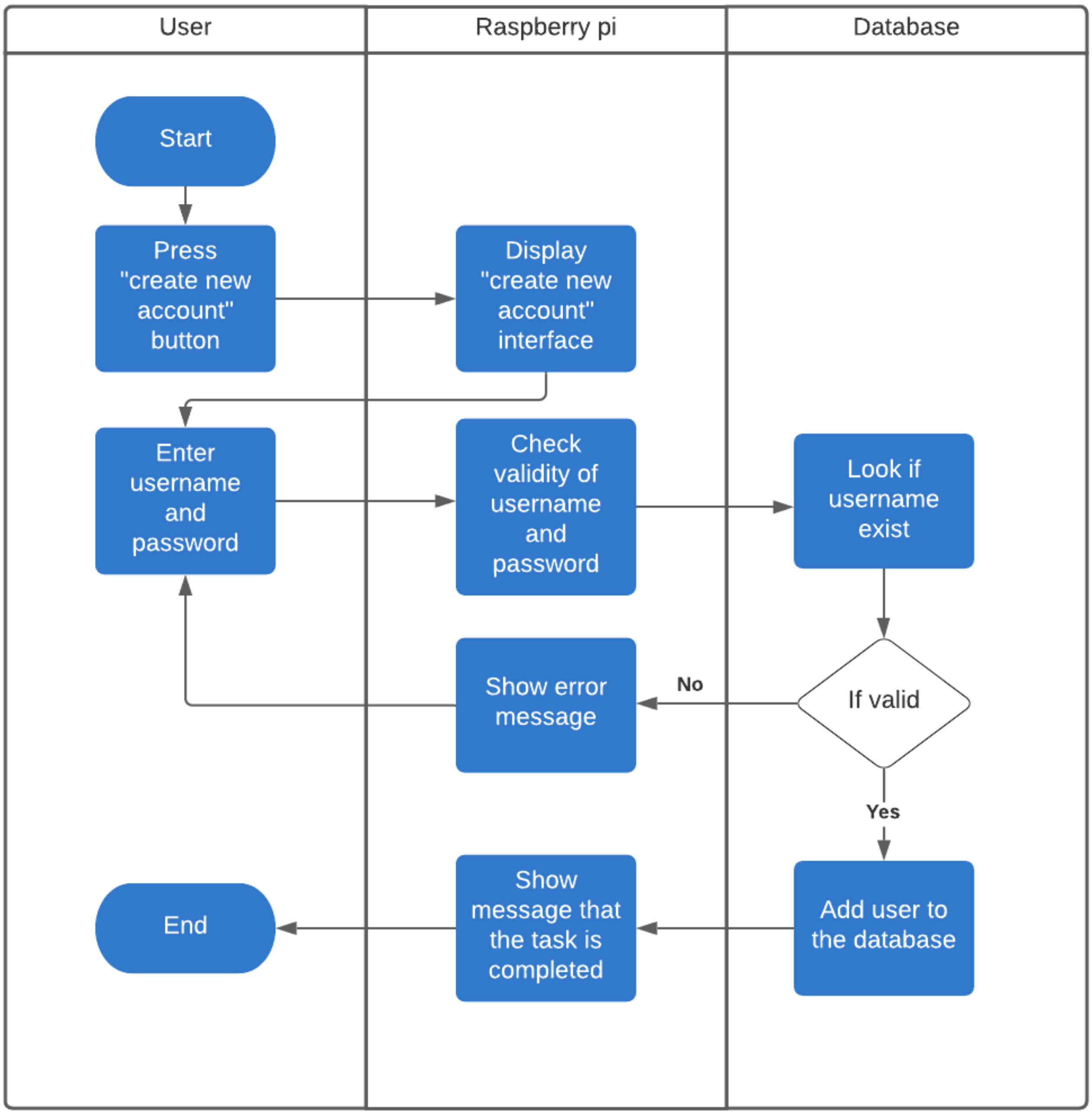
- Game: The game setting will have the level of difficulty, the landscape and time. The game ends whenever the user's health gets to zero.
- Player: The stats and scores of players.
- Cow: Object that players should not hit otherwise they will lose score.
- Cowboy: Object that players should hit. The main opponent of the player, they will shoot the user after a while.
- Barrel: Object to protect Cowboy.

Ball tracking activity diagram


User
Raspberry pi
z-Camera
xy-Camera
Start
Throw ball/starts game
Determine z-coordinate of the ball
Determine if the ball hit the wall
Did the ball hit the wall
No
Yes
Determine if the ball hit a target and add score approprately
Determine where the ball hit the wall
Take a picture
Take a picture

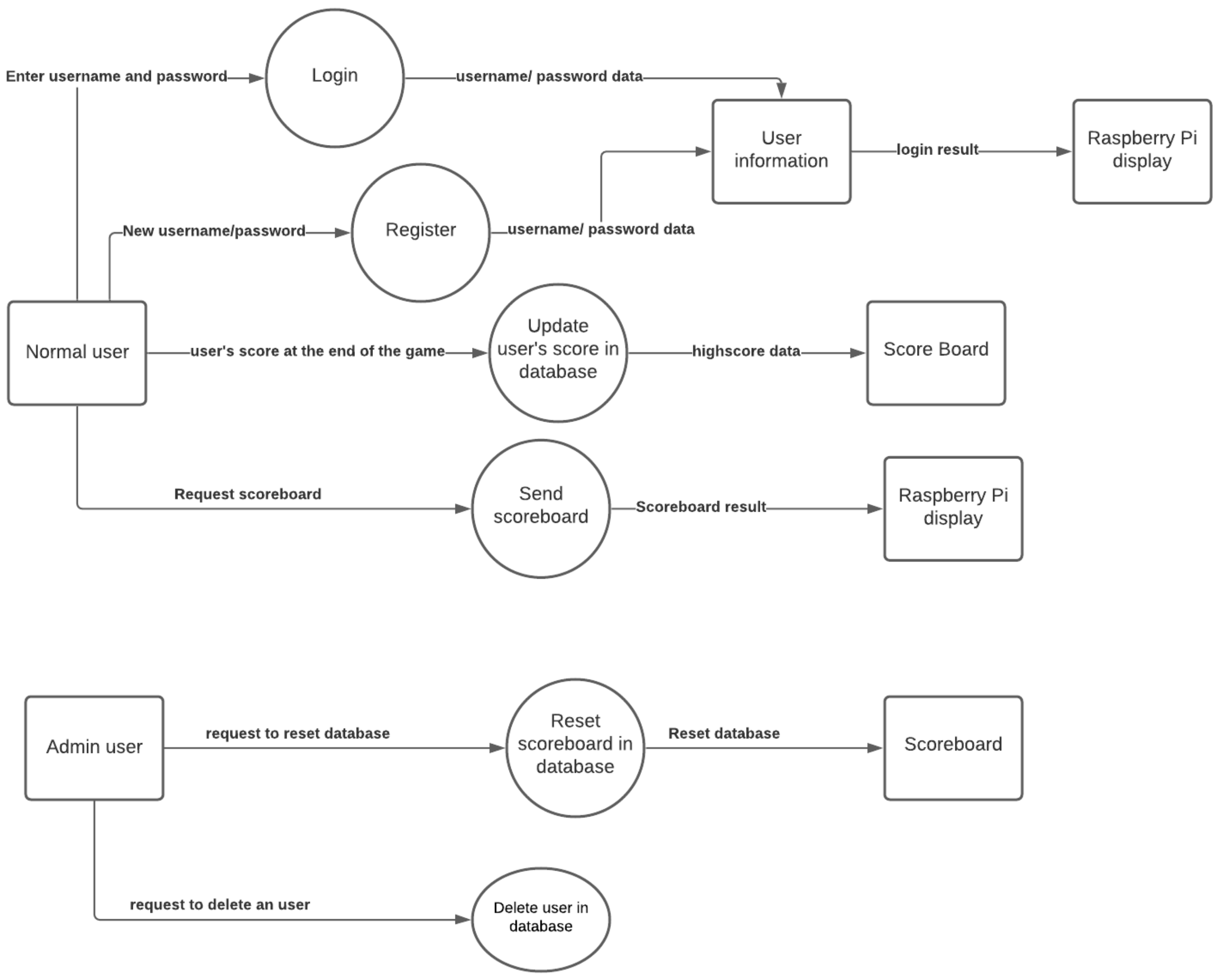
In this diagram the way the raspberry pi and the two cameras work together to determine where and when the ball hits the wall is shown. Here the z-camera is the camera that is pointed parallel to the wall and the xy-camera is pointed at the wall. The z-camera will take a picture of the ball and then the raspberry pi will determine whether the ball has hit the wall. If the ball has hit the wall the pi will make the xy-camera take a picture. It will use that information to determine where the ball hit the wall. And then see if the ball hit a target.

Account creation activity diagram


User
Raspberry pi
Database
Start
Press "create new account" button
Display "create new account" interface
Enter username and password
Check validity of username and password
Look if username exist
If valid
No
Show error message
Yes
Add user to the database
Show message that the task is completed
End

In this diagram the process of making a new account is shown. When a user presses the “create new account” button the raspberry pi will show the correct menu. There the user can input their username name and password. The raspberry pi will then check if the username and password are valid, if so it will send a query to the database. The database will then check if the username is not yet in the database. If the username is already in the database or the username and password are not valid the raspberry pi will show an error message and return to the input screen. If all the inputs were correct then the account will be added to the database and the raspberry pi will show a message to the user that the account was created successfully.

Data flow diagram



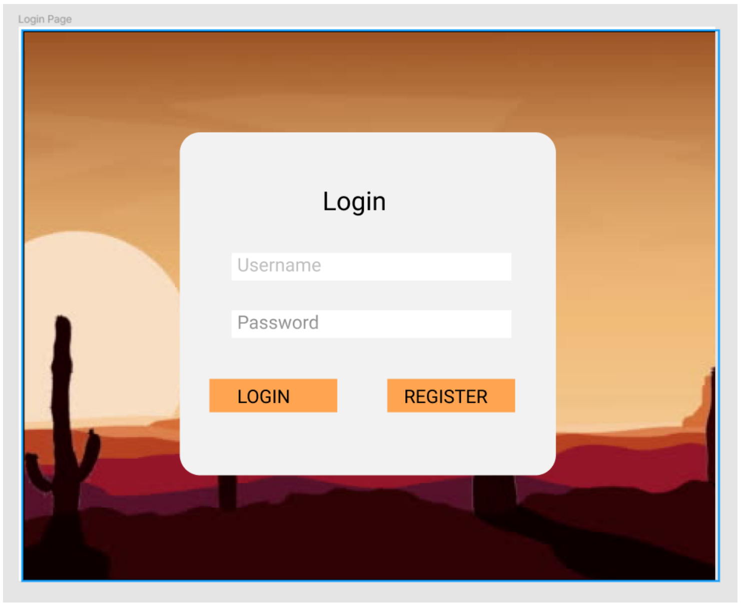
In this data flow diagram you can see 5 main flows:

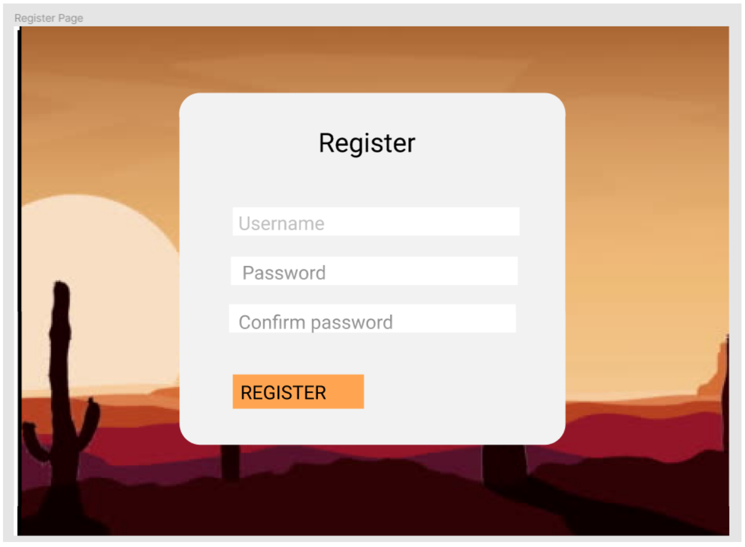
+ A normal user's registration information can come from the user's input and is sent to the database to create a new account.
+ A normal user's credentials can come from the user's input and are sent to the database to login and the result will be displayed back to the user.
+ A normal user can send a request to see the scoreboard and the database will send back the scoreboard to Raspberry Pi.
+ An admin user can request to reset the database and it will be updated in the database.
+ An admin user can request to delete a specific normal user's account and it will be updated in the database.

4. Product User Interface

We have designed an interface for the system. It includes different pages and each page has its own functionality which we will explain in detail:

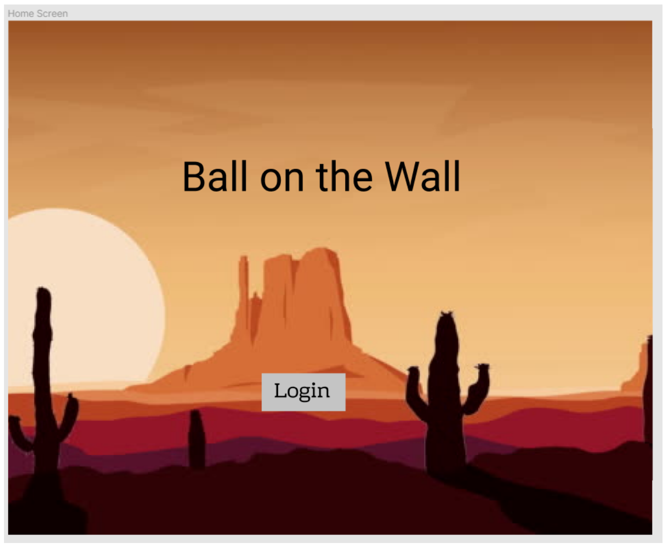
The login/register interface where players can login and create an account to play





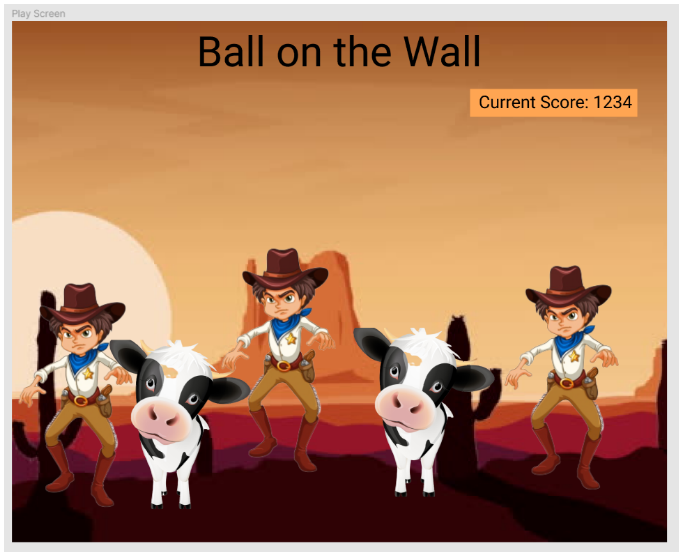
The login and register interface includes some personal information such as a username and password for playesr to fill in in order to create an account to play the game. This user data will be stored in the database.

The main screen



The main screen shows the main background of the game, game title "Ball on the Wall" and also a login button to redirect the player to the login screen.

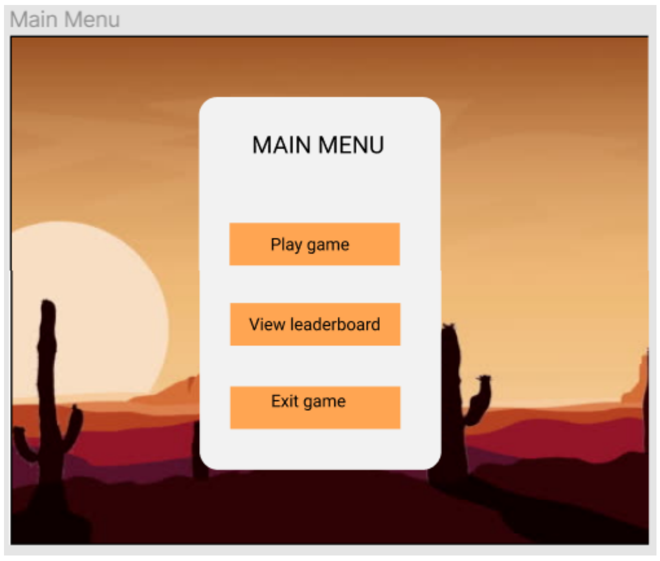
The play screen with all the game objects where the player can interact with in the game



The cowboy and the cow are two objects that will be shown up randomly on the screen, and these are the target objects for the player to throw the ball at. Once the ball hits the cowboy, the player will gain points, on the other hand, once the cow is hit, the player loses the point.

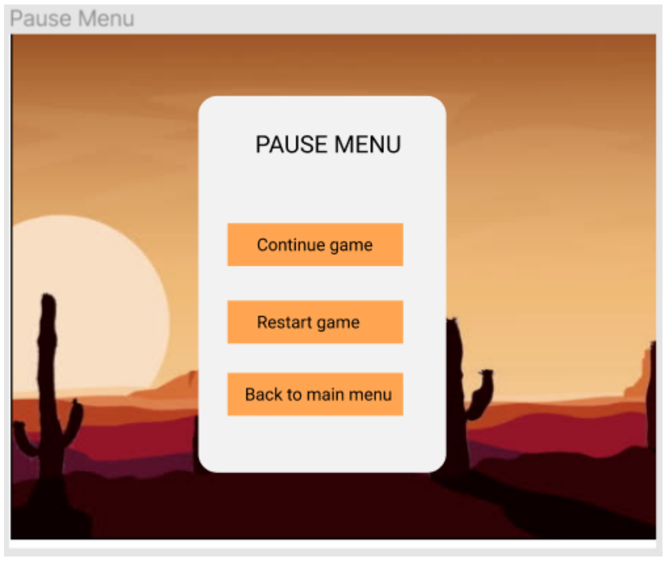
The play screen also shows the current score of the player, this will be updated every time the ball hits objects.

The main menu, pause menu and game over menu screen



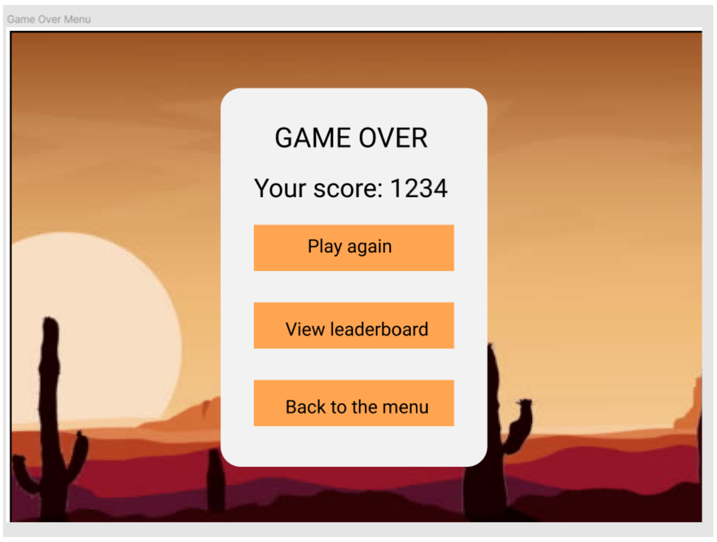
The main menu will show up once the player logs in successfully. The main menu contains three buttons:

+ *Play game:* A new game will start.
+ *View leaderboard:* The list with highscores will be displayed.
+ *Exit game:* The player will log out of the game and the main screen will be shown.



The pause menu will be shown when the player wants to pause the game by pressing the button(i.e Esc button in the keyboard). This menu contains 3 buttons:

+ *Continue game:* The current game will continue.
+ *Restart game:* The game will restart and also reset the score.
+ *Back to main menu:* The player returns to the Main Menu.



The game over menu will be shown when the game is over. This menu contains 3 buttons:

+ *Play again:* A new game will be started
+ *View leaderboard:* The list with highscores will be displayed.
+ *Back to main menu:* The player returns to the Main Menu.

5. Prevention/Mitigation Criteria (Security Controls)

+ Passwords must be sanitized and encoded before being pushed to the database.
+ Database may not be stored on the Raspberry Pi.
+ Database accesses must use prepared statements to prevent SQL-injections.
+ The Raspberry Pi must not be able to send environment data from the cameras.
+ Usernames and passwords must be forced to follow a specific format.
+ Users cannot enter the wrong password/username more than three times consecutively.
+ Admin functions must be granted to one and only one person.
+ ToS documents must be presented on the login screen.

6. The cost involved (numbering for requirements in SBD of sprint 1 is used)

The time investment for **requirement 1,2,3,5,7 and 8** will be all separate and therefore small code. It will not take much time to write, but it is still important to do.

Requirement 4 is about the GDPR, this will not be code we have to write. We just have to make sure our code is compliant with the GDPR.

The subscription for the online database (requirement 6) will cost money. As we are all students, we can use our student license to get free access to the database, but otherwise it will cost money.

7. Conclusion:

The theme of the game will be **cowboy's/western desert.** With this theme we designed mockups with the functions of the menus. We decided which **menus** we want when and which **buttons** on these menus we want. We changed nothing on the list of functional/non-functional requirements and the security by design checklist is now also implemented in this document. Most of the costs involved (for security requirements) in our application will be typing code, so **time related.**

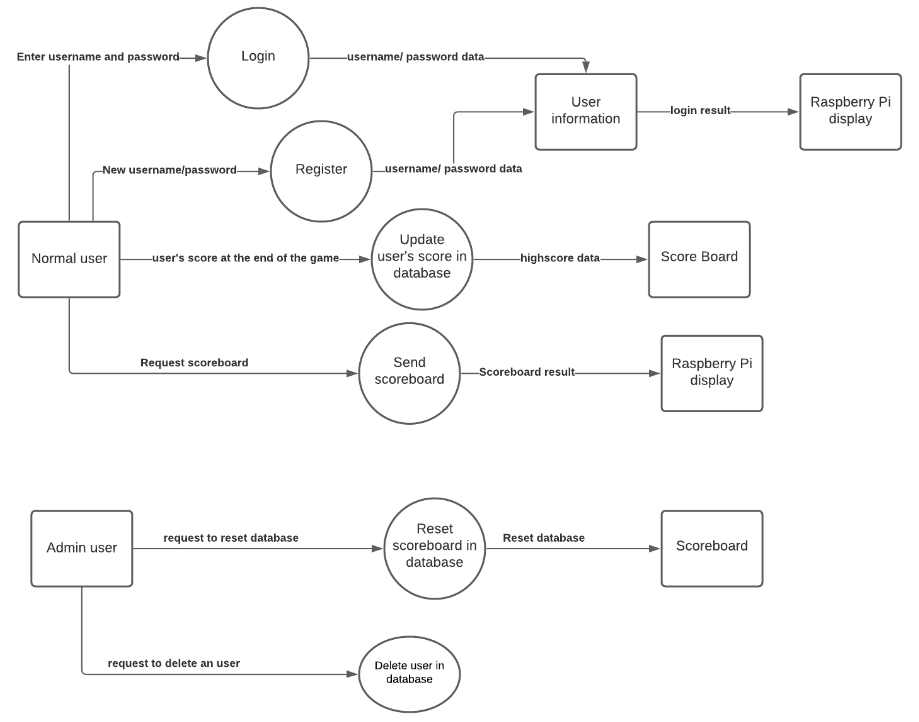
Module 5-Computer Systems (2021-22)
Project

UNIVERSITY OF TWENTE.

Security by Design Checklist
(Design Phase)

Team ID:30	**Team Members:** Jan van Zwol, Vo Nhat Minh, Tran Duc Duc, Marjolein Bolten, Ho Hoang Phuoc, Daan Velthuis
Project Name: Ball on the Wall	**Mentor(s):** Puru Vaish & Venelina Pocheva

Sr. No.	Review Security Architecture	Put checkmark ✓if you have completed the Review Security Architecture as suggested in the left column	Additional comments (If required)	Security Controls/Countermeasures	Put checkmark ✓if you have completed the Security controls points as suggested in the left column	Additional comments (if required)
1	Check Trust Boundaries, The trust boundaries in our system affect the Raspberry Pi and the database. + We do trust data coming from the database to the Raspberry Pi but we do not necessarily trust the data coming from the Raspberry Pi. + We also trust the requests that come from the admin account.		N/A	Check the prevention criteria, + All the data coming from the Raspberry Pi should be sanitized before being sent to the database. The software on the Pi should only make use of prepared statements when communicating with the database. + The admin account is a private account that can only be known by the staff and its password will be changed frequently to prevent being revealed.		N/A
2	Identify data flows + SQL injection through user's input can come from an untrusted source. + Credential input can come from an untrusted source. + The scores come from the user so it can also be from an untrusted source.		N/A	Check the mitigation criteria to reduce the impact of the risk/threat for the application + Salt password: If an attacker identifies the hash value of a password, he won't be able to login to the associated password because we'll use a random salt when hashing the user's password. We will use a library to hash the passwords. + Sanitize input: All user's input will be sanitized, i.e. remove white space, such that the attacker cannot input SQL injection through user's input.		N/A

				+ Checking invalid score: We will have some precalculated limited possible score that users can get from the game. If a user tries to input some fake score it will be checked and not accepted.		
3	Entry and Exit points of the system and its components. + Entry point: From user (normal and admin) input, data will be executed by the Raspberry Pi before pushed to the database. + Exit point: Raspberry Pi, data will be returned and displayed/ executed on the Raspberry Pi.		N/A	Make a data flow diagram to visualize and understand the data flow, input, output points, and trust boundaries .  Enter username and password Login username/ password data User information login result Raspberry Pi display New username/password Register username/ password data Normal user user's score at the end of the game Update user's score in database highscore data Score Board Request scoreboard Send scoreboard Scoreboard result Raspberry Pi display Admin user request to reset database Reset scoreboard in database Reset database Scoreboard request to delete an user Delete user in database		N/A
4	Write the complete architecture in the SDD template. Review and approve among yourselves and by your assigned mentor(s).		N/A	Analyze the cost involved to implement the security controls (if any).		N/A

Ho Hoang Phuoc: yes

Jan van Zwol: yes

Vo Nhat Minh: yes

Tran Duc Duc: yes

Daan Velthuis: yes

Marjolein Bolten: yes

Puru Vaish:
Venelina Pocheva: