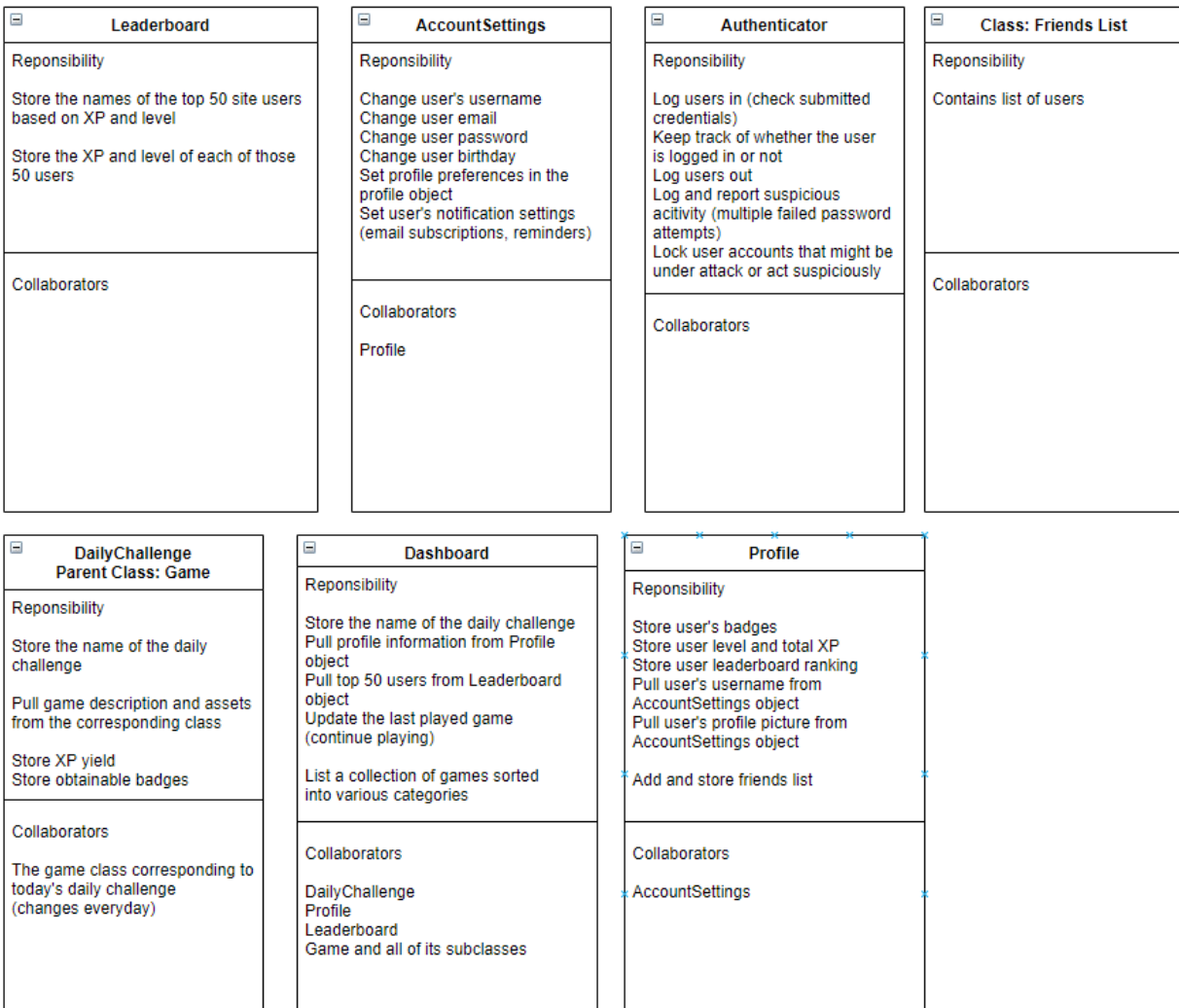


# DreamTeam System Design Document

## Table of Contents

CRC Cards.....	3
Software Architecture Diagram .....	7
Three-Tier Architecture.....	7

## CRC Cards





**Game (Abstract)**  
**Subclasses: DailyChallenge,**  
**StackGame, QueueGame, TreeTraversalGame,**  
**DictionaryGame, ArrayGame, IfStatementGame,**  
**HelloWorldGame**

Responsibility

Check game state (currently playing, loss, win)

Update user's XP count and level on game completion  
in Profile object

Update user's badge collection on achievement completion  
in Profile object

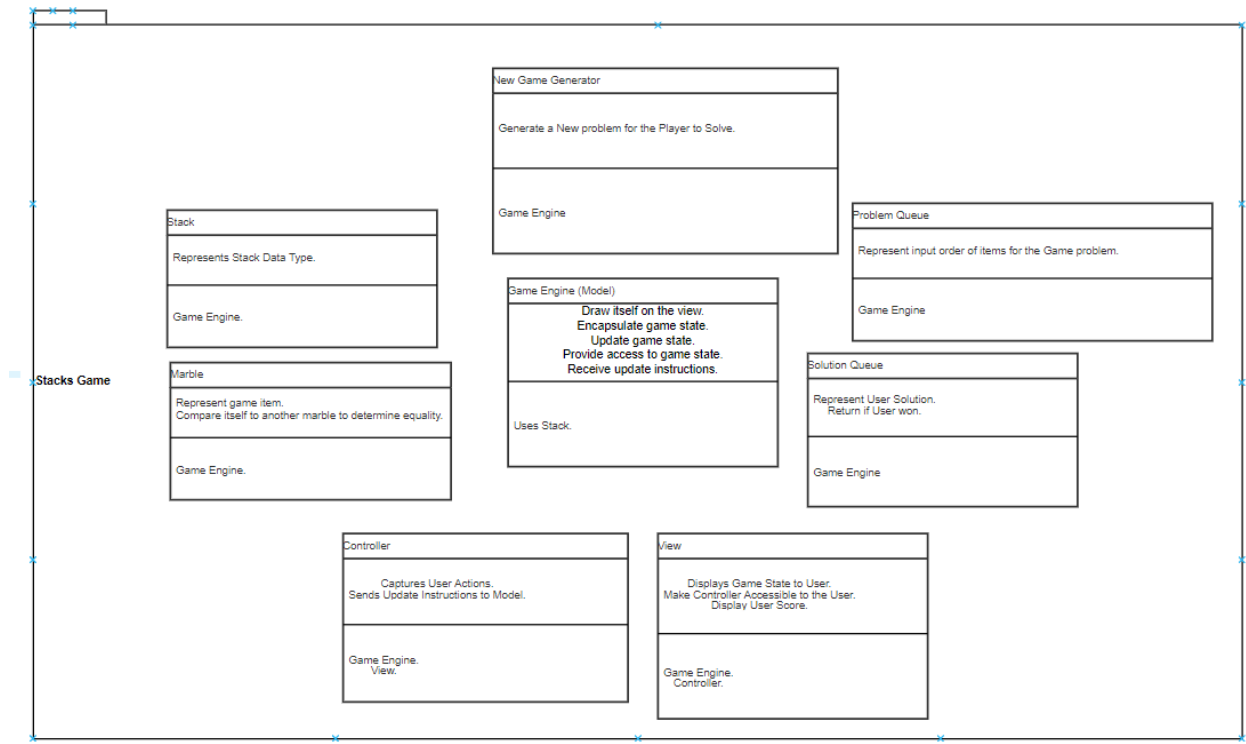
Validate user input

Update class attributes to reflect user's action in game  
Save Game state (if user leaves, doesn't have to start over )

Collaborators

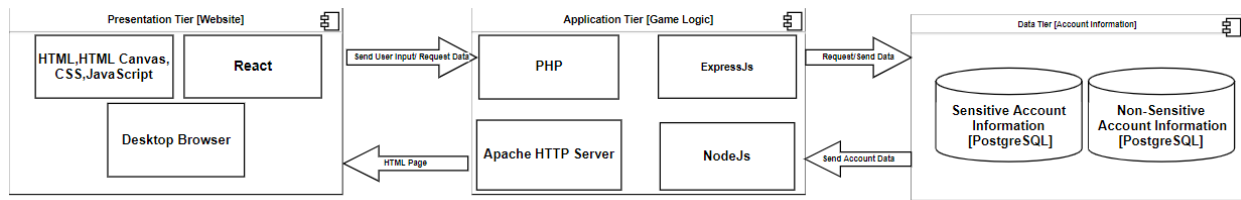
Profile

<div> <div></div> <div> <b>QueueGame</b>  Parent class: Game </div> </div> <div> <div>Responsibility</div> <div> Update XP yield based on user's score  Store obtainable badges  Update user's score after each marble action  Keep track of the order of marbles  Update game state to "won" when the amarbles are in the desired order  Record the user's action history (what moves have been made) </div> </div> <div> <div>Collaborators</div> </div>	<div> <div></div> <div> <b>IfStatementGame</b>  Parent class: Game </div> </div> <div> <div>Responsibility</div> <div> Update XP yield based on user's score  Store obtainable badges  Record the if conditions that the user submitted  Check that the user submitted valid if conditions through the dropdown list  Move the robot north, south, east, west depending on the if conditions specified by the user  Update the game state to "won" when the robot has reached the goal tile  Calculate the user's score based on the number of moves the robot took to reach the goal tile </div> </div> <div> <div>Collaborators</div> </div>	<div> <div></div> <div> <b>DictionaryGame</b>  Parent class: Game </div> </div> <div> <div>Responsibility</div> <div> Store XP yield  Store obtainable badges  Check whether the user answered the question correctly  Update the game state to "won" when the user has correctly answered all questions </div> </div> <div> <div>Collaborators</div> </div>
<div> <div></div> <div> <b>TreeTraversalGame</b>  Parent class: Game </div> </div> <div> <div>Responsibility</div> <div> Store XP yield  Store obtainable badges  Check if a user's traversal choice alerts the criminal  Update the game state to "won" if the user has chooses a traversal order that doesn't alert the criminal  Validate user input </div> </div> <div> <div>Collaborators</div> </div>	<div> <div></div> <div> <b>ArrayGame</b>  Parent class: Game </div> </div> <div> <div>Responsibility</div> <div> Store XP yield  Store obtainable badges  Check if the user dragged the shape to the correct array index  Update the game state to "won" when all questions have been answered correctly </div> </div> <div> <div>Collaborators</div> </div>	<div> <div></div> <div> <b>HelloWorldGame</b>  Parent class: Game </div> </div> <div> <div>Responsibility</div> <div> Store XP yield  Store obtainable badges  Check if the user has submitted their name through the text box  Check that the user has submitted "Hello World" through the text box  Update the game state to "won" when "Hello World" has been printed to the screen </div> </div> <div> <div>Collaborators</div> </div>



# Software Architecture Diagram

## Three-Tier Architecture



The three-tier architecture diagram is responsible for describing how our system is divided into components. Our system is divided into 3 tiers, those being the Presentation Tier (website), the Application Tier (web server), and the Data Tier (database).

The Presentation Tier contains the website layer. The website layer represents our website as it is displayed to user. The website layer includes all HTML, CSS, and JavaScript files using the React framework. The purpose of the website layer is to take user input and send it to the Application Tier for processing and validation. This includes usernames, emails, passwords, changes to other account information, and game inputs. The website layer is also responsible for requesting HTTP pages from the web server and displaying the HTML, CSS, and JavaScript that it receives from the Application Tier.

The Application Tier contains the web server layer. The web server layer includes the Apache HTTP Web Server that hosts our website and all PHP files and JavaScript files using the ExpressJs framework. The web server layer also includes the NodeJs environment required to execute the JavaScript files. The purpose of the web server layer is to process and validate all user input received from the website layer (Presentation Tier). The web server layer is responsible for handling all game logic. Therefore, all game inputs are processed, validated, and reflected in this layer. Additionally, the web server layer is responsible for querying and updating account data (e.g. usernames, emails, passwords) on the PostgreSQL databases in the Data Tier to perform password authentication and update user account information. Queries the database layer are made through prepared statements to avoid SQL injection attacks.

Within the Data Tier lives the database layer. The database layer includes the PostgreSQL databases responsible for storing usernames, passwords, emails, and other account information. The purpose of the database layer is to send non-sensitive

account information to the web server layer (Application Tier) for processing and to send sensitive account information to the web server layer for password authentication. The database layer is also responsible for reflecting any changes to user account information in the corresponding database.

The following is an example use case of our system. A user logs into our website with a correct username and password. The HTML file in our website layer sends the username and password to a PHP file in our web server layer by way of HTML form and POST request. The PHP file in our web server layer queries the row in the PostgreSQL database (sensitive account information) with the given username and password with a prepared statement. The database fulfills the query request by sending the row with matching username and password to the PHP file. The PHP file verifies that the query returned a non-empty output and then has the Apache HTTP Server send the “logged-in” HTML page to the website layer. The website layer displays the “logged-in” HTML page to the user. The user has now securely logged into our website.

If the user inputs an invalid password, the query to the PostgreSQL database returns an empty output. The PHP file would then log the failed password attempt and have the Apache HTTP server send an “error” HTML page to the website layer. After 3 failed password attempts, the user will be locked out of their account 5 minutes. The lockout times doubles with each subsequent failed attempt.