

# Humans vs Zombies Website Refactor

Team: Josh Brown

## Overview

The goal of this project is to refactor the already existing codebase for the HvZ website. The website is currently written primarily in PHP and embedded HTML, the refactor will migrating away from PHP to JavaScript using the React.js framework. The project will also include splitting the frontend and backend code into two different code bases.

## Project Requirements

The website should be able to:

- Allow new users to sign up on the website
- Allow admin users to create/manage events and other users
- Allow users to join events and play in events

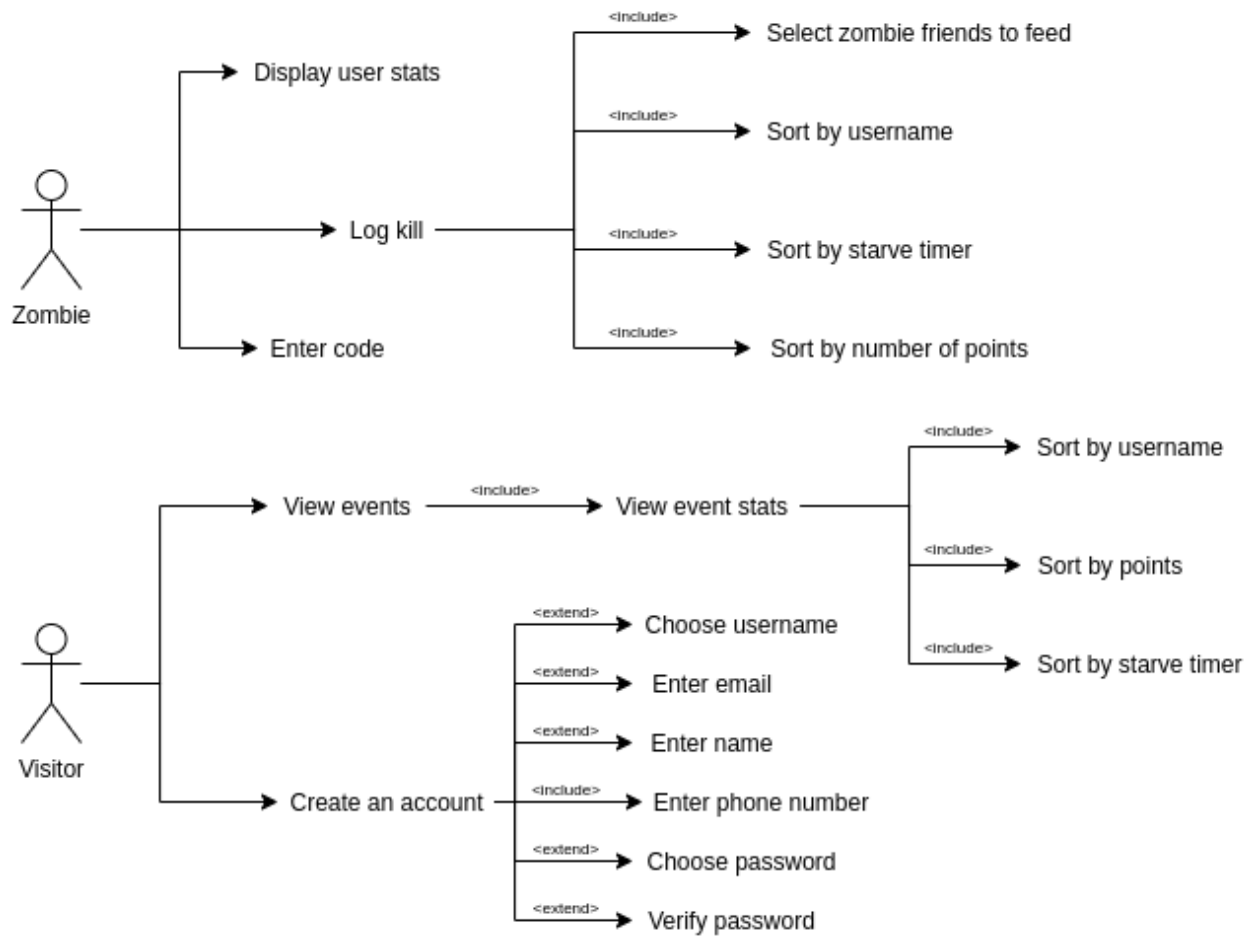
Stretch goals:

- Automatically update the website during events
- Automatically send emails

## Users and Tasks: Use Cases (Text or UML Diagrams)

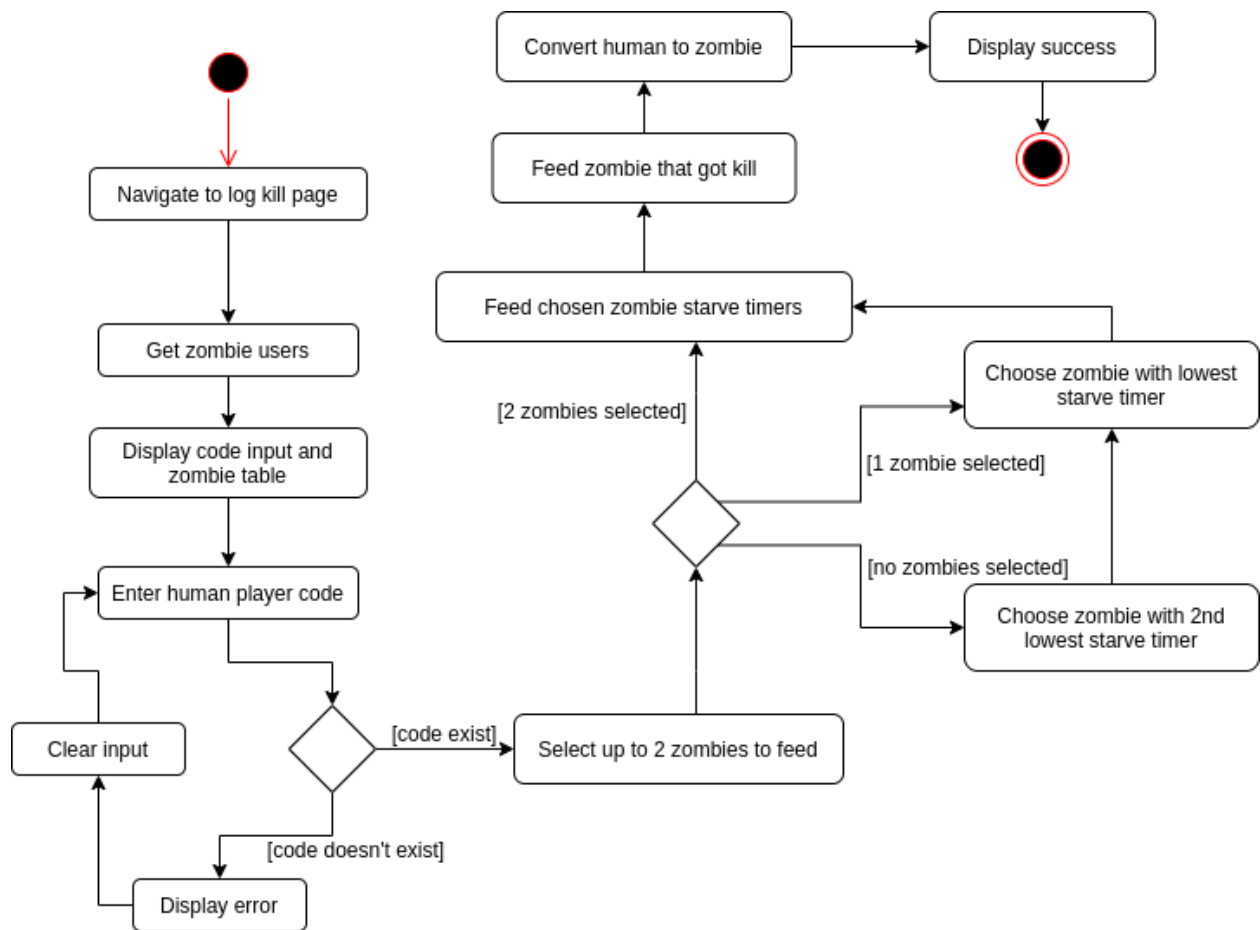
The system will consist of four main types of users:

1. **Visitor** - Anyone that visits the website
  - a. User can browse through each of the pages
  - b. User can sign up for the website
2. **Member** - A user that has signed up for an account
  - a. User can join Weeklong events
  - b. User can change account settings
3. **Player** - An account user that is part of a Weeklong event. There are three types of players:
  - a. Human:
    - i. Humans can input codes into the website for points
  - b. Zombie:
    - i. Zombies can input codes into the website for points
    - ii. Zombies can input Human player codes in order to zombify them
  - c. Deceased:
    - i. Deceased players can no longer participate in the event
4. **Moderator/Admin** - A moderator has special permissions to change some data in the database and manage events. Admins have a few higher-level permissions.
  - a. Moderators can create codes for weeklong events
  - b. Moderators can edit event details
  - c. Moderators can upload documents
  - d. Admins can edit users and change user clearance level (can make an account user into a moderator or admin)

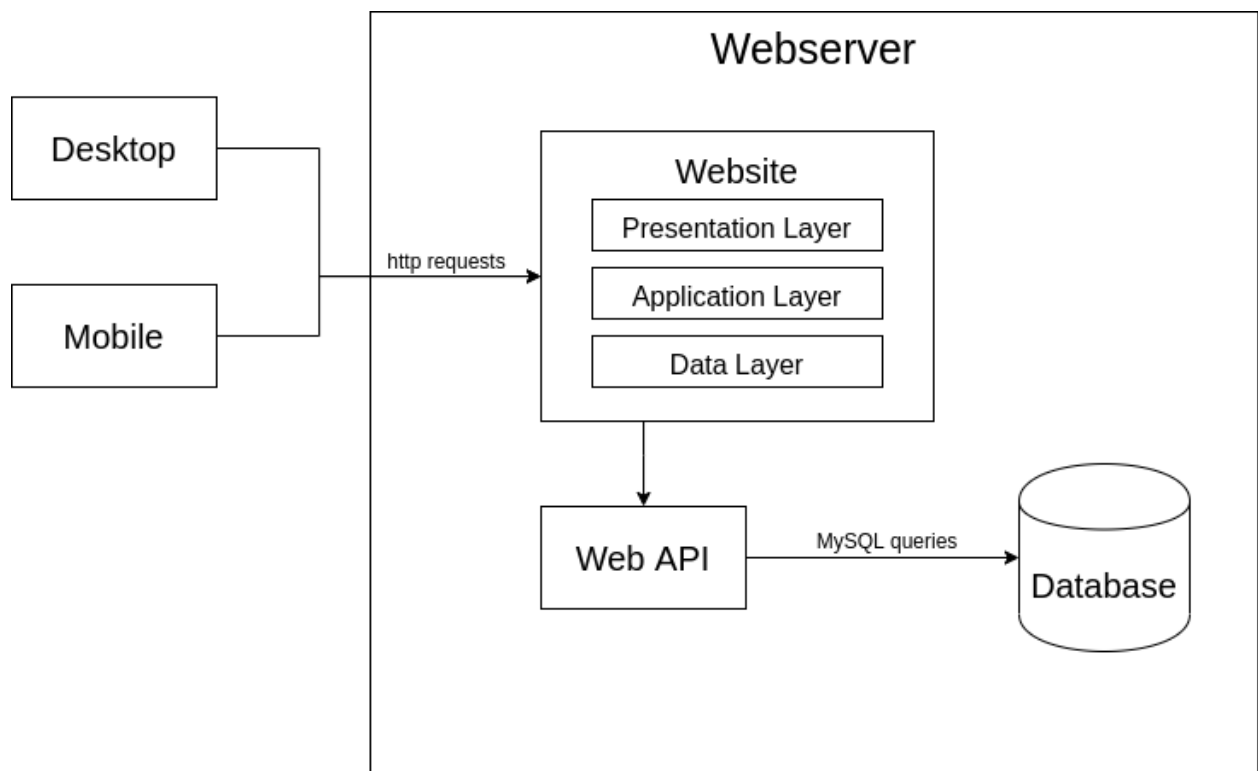


## UML Activity Diagram

Use case: Zombies can input Human player codes in order to zombify them



## Architecture Diagram



## Data Storage

The database will be a MySQL database hosted by Namecheap. The Database class in the backend portion of the code will be the class that actually interacts with the database. The frontend code will have an API class that makes REST calls to the backend in order to retrieve the data.

The Database class is the only class on the backend as the backend runs differently than the frontend. The backend has a single “routes” object that you use to map the url endpoints to functions. The different types of endpoints are sorted into files and then included into the main backend file.

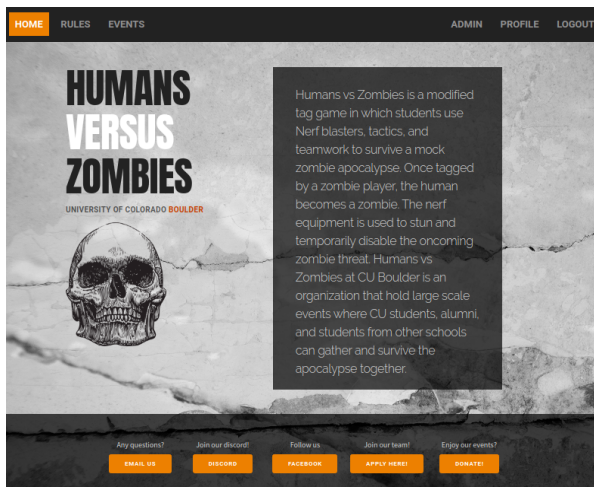
Database tables:

<b>Users</b> id username email firstName lastName password clearance	<b>UserDetails</b> user_id joinDate activated	<b>Tokens</b> id userID token type expiration	<b>Lockin</b> id title eventDate waiver eventbrite blasterEventbrite state
<b>LockinDetails</b> lockinID details	<b>Weeklongs</b> id title startDate endDate state stunTimer waiver	<b>WeeklongDetails</b> weeklongID details monday tuesday wednesday thursday friday	<b>WeeklongMissions</b> weeklongID day campus mission time location locationLink
<b>WeeklongPlayers</b> id userID weeklongID playerCode type status poisoned points kills starveDate	<b>Codes</b> id weeklongID code pointValue feedTime type numUses initNumUses activates expiration	<b>UsedCodes</b> id userID codeID timeUsed	<b>SupplyDrops</b> codeID location poisoned
<b>Activity</b> id weeklongID user1 user2 action			

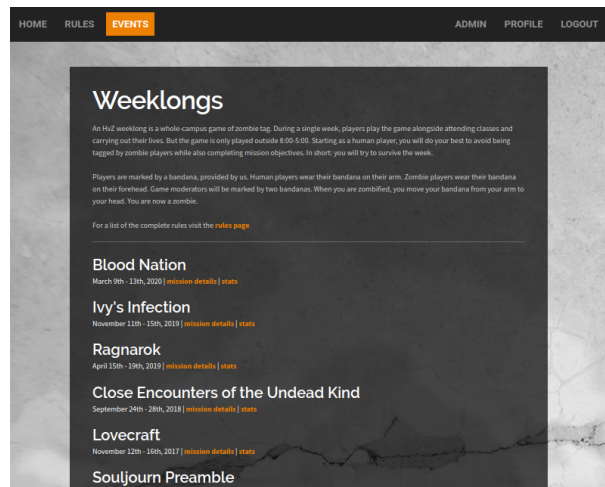
# UI Mockups/Sketches

As the website already exists the user interface will stay the same for the most part. The only pages that need to be redesigned are the moderator/admin pages and the profile page for members. The moderator/admin pages just need to be cleaned up with css. The profile page needs to have a side navigation bar or some sort of dropdown to access the settings page to change user information.

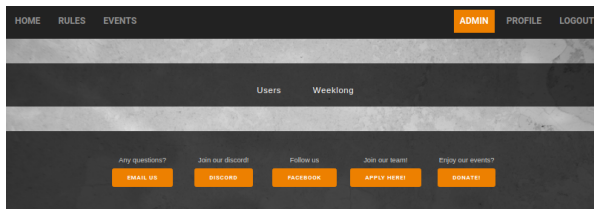
## Home page



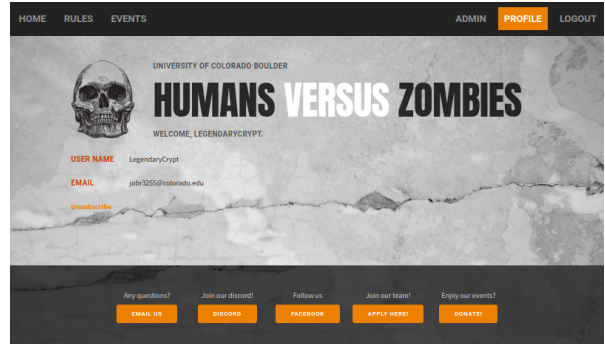
## Weeklong page



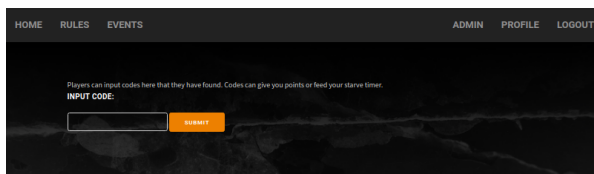
## Admin page



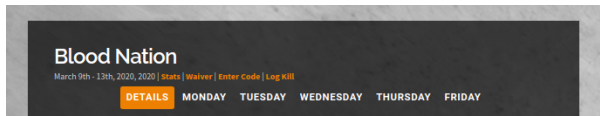
## Profile page



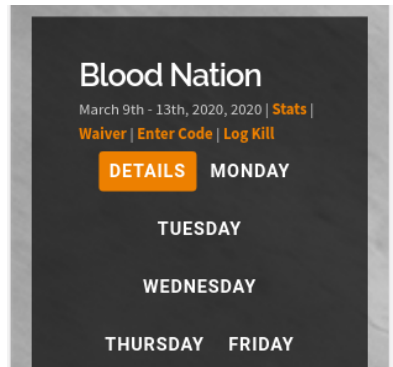
## Code entry page



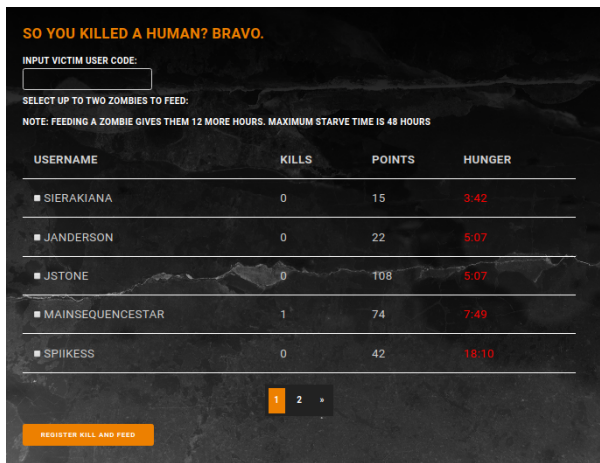
Weeklong details page (desktop)



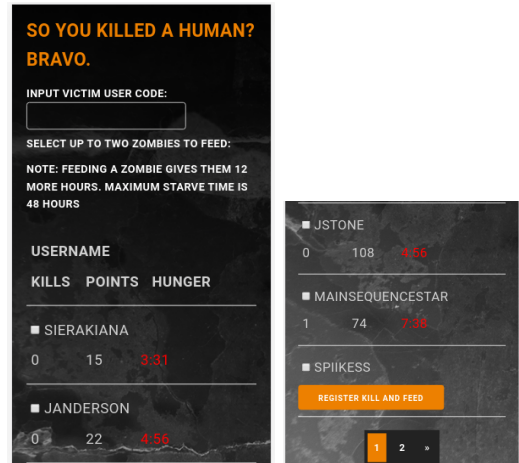
Weeklong details page (mobile)



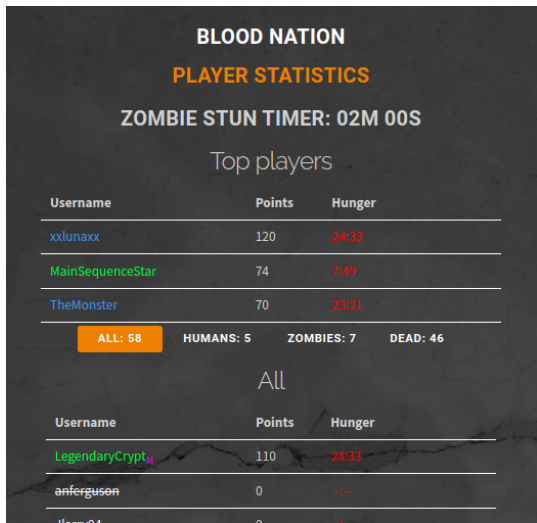
Zombie log kill page (desktop)



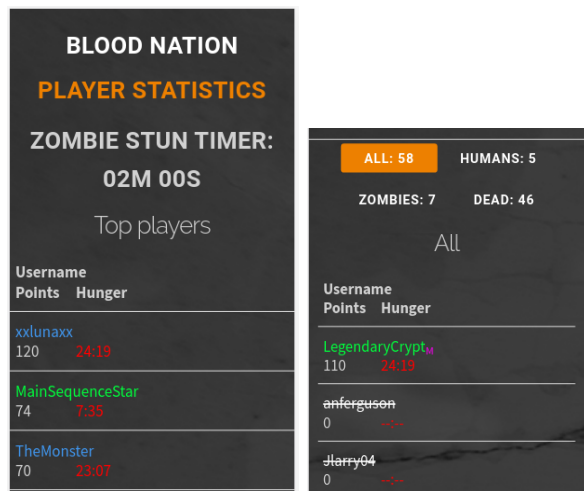
Zombie log kill page (mobile)



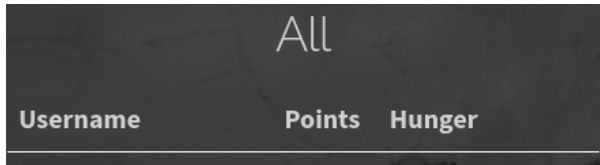
Weeklong stats page (desktop)



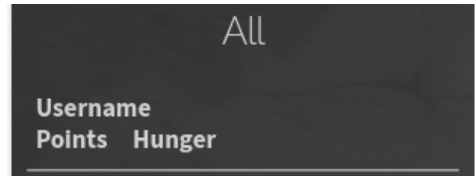
Weeklong stats page (mobile)



All players table (desktop)



All players table (mobile)



Human players table (desktop)

Humans			
Username	Points	Hunger	

Human players table (mobile)

Humans			
Username	Points	Hunger	

Zombie players table (desktop)

Zombies				
Username	Type	Kills	Points	Hunger

Zombie players table (mobile)

Zombies				
Username	Type	Kills	Points	Hunger

Dead players table (desktop)

Dead			
Username	Kills	Points	Starved

Dead players table (mobile)

Dead			
Username	Kills	Points	Starved

Nav bar (desktop)

HOME	RULES	EVENTS	ADMIN	PROFILE	LOGOUT
HOME	RULES	EVENTS	LOGIN	SIGN UP	

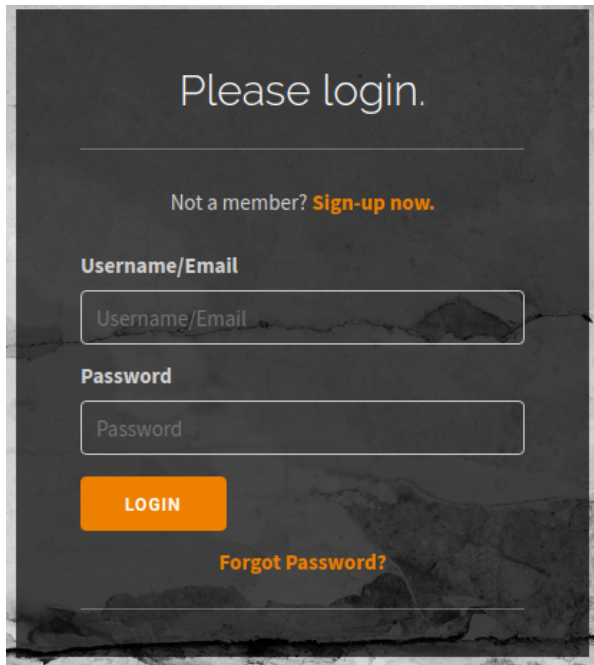
Nav bar (mobile)

☰	👤
☰	LOGIN SIGN UP

Nav bar mobile dropdowns

☰	👤
HOME	PROFILE
RULES	ADMIN
EVENTS	LOGOUT

Login page (desktop)



Please login.

---

Not a member? [Sign-up now.](#)

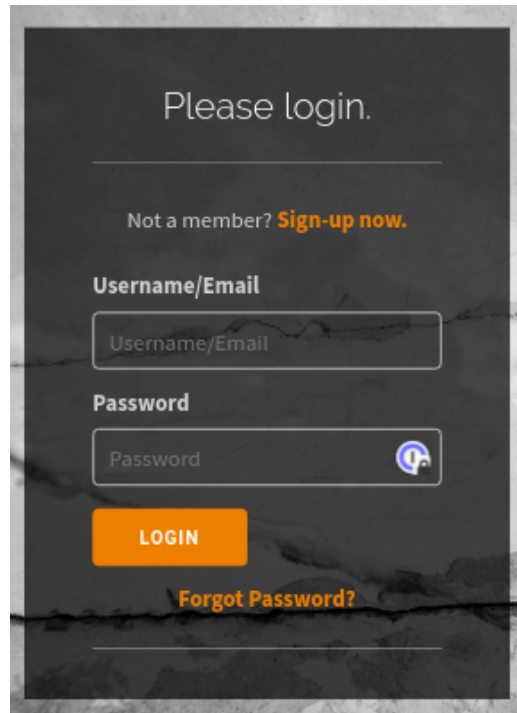
**Username/Email**

**Password**

**LOGIN**

[Forgot Password?](#)

Login page (mobile)




Please login.

---

Not a member? [Sign-up now.](#)

**Username/Email**

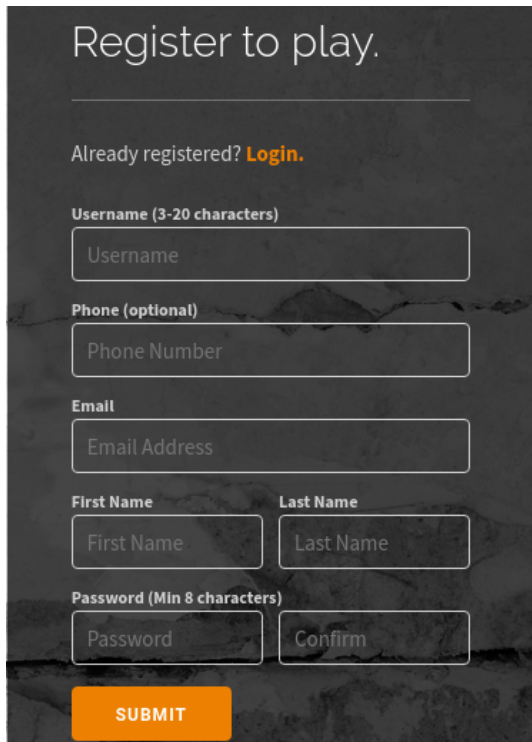
**Password**

**LOGIN**

[Forgot Password?](#)

Sign up page (desktop)



Register to play.

---

Already registered? [Login.](#)

**Username (3-20 characters)**

**Phone (optional)**

**Email**

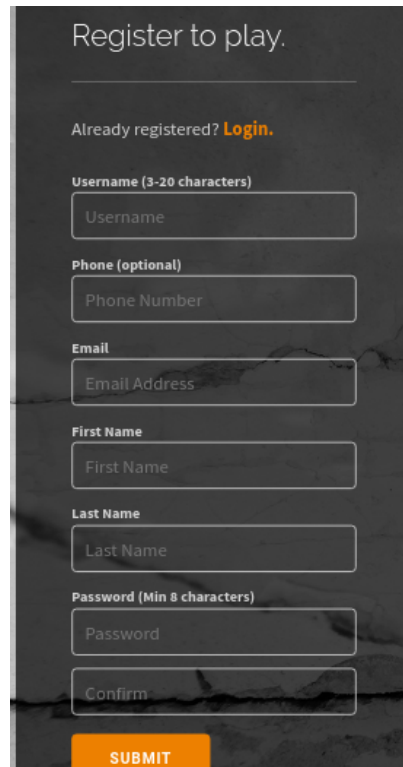
**First Name**

**Last Name**

**Password (Min 8 characters)**

**SUBMIT**

Sign up page (mobile)



Register to play.

---

Already registered? [Login.](#)

**Username (3-20 characters)**

**Phone (optional)**

**Email**

**First Name**

**Last Name**

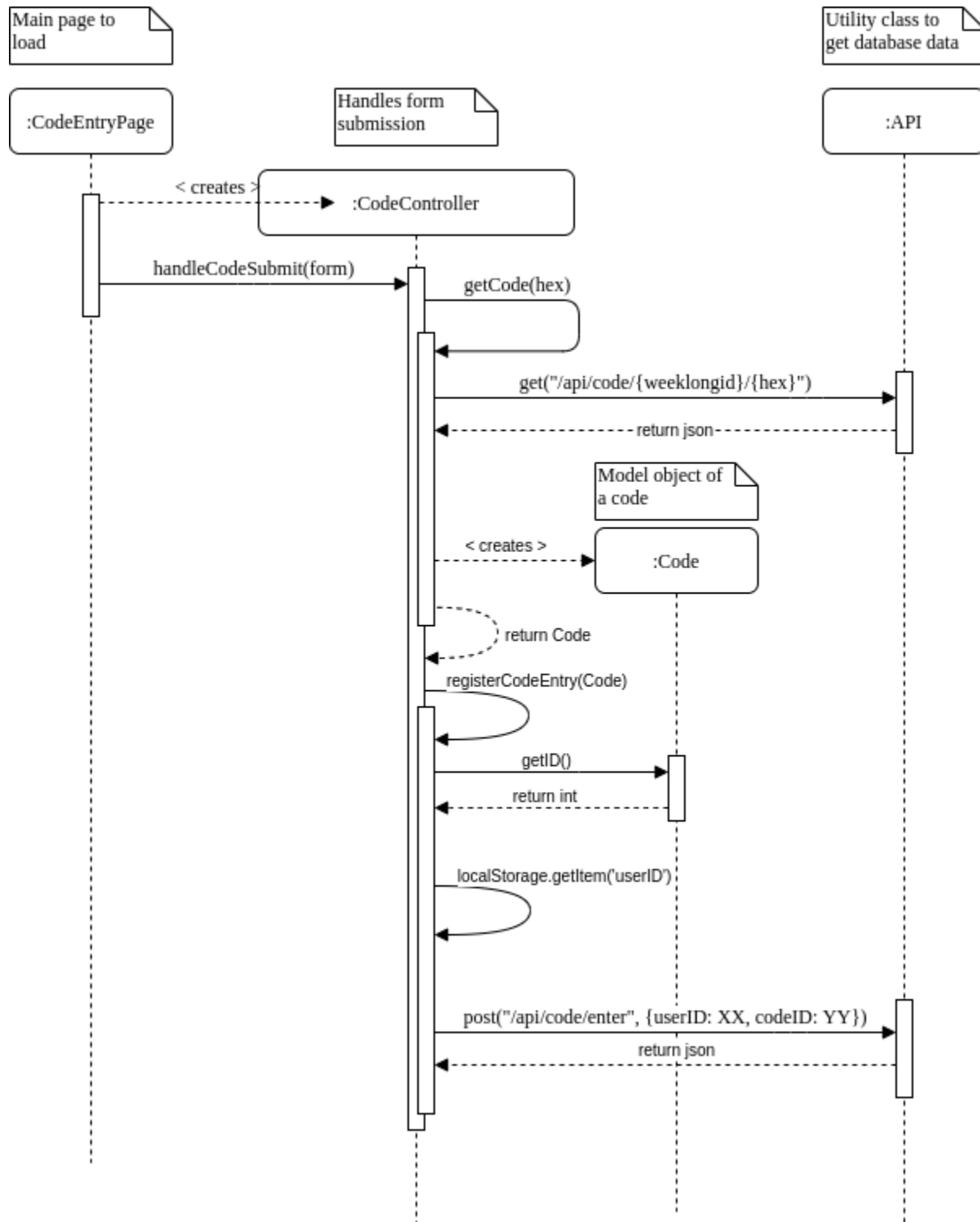
**Password (Min 8 characters)**

**SUBMIT**

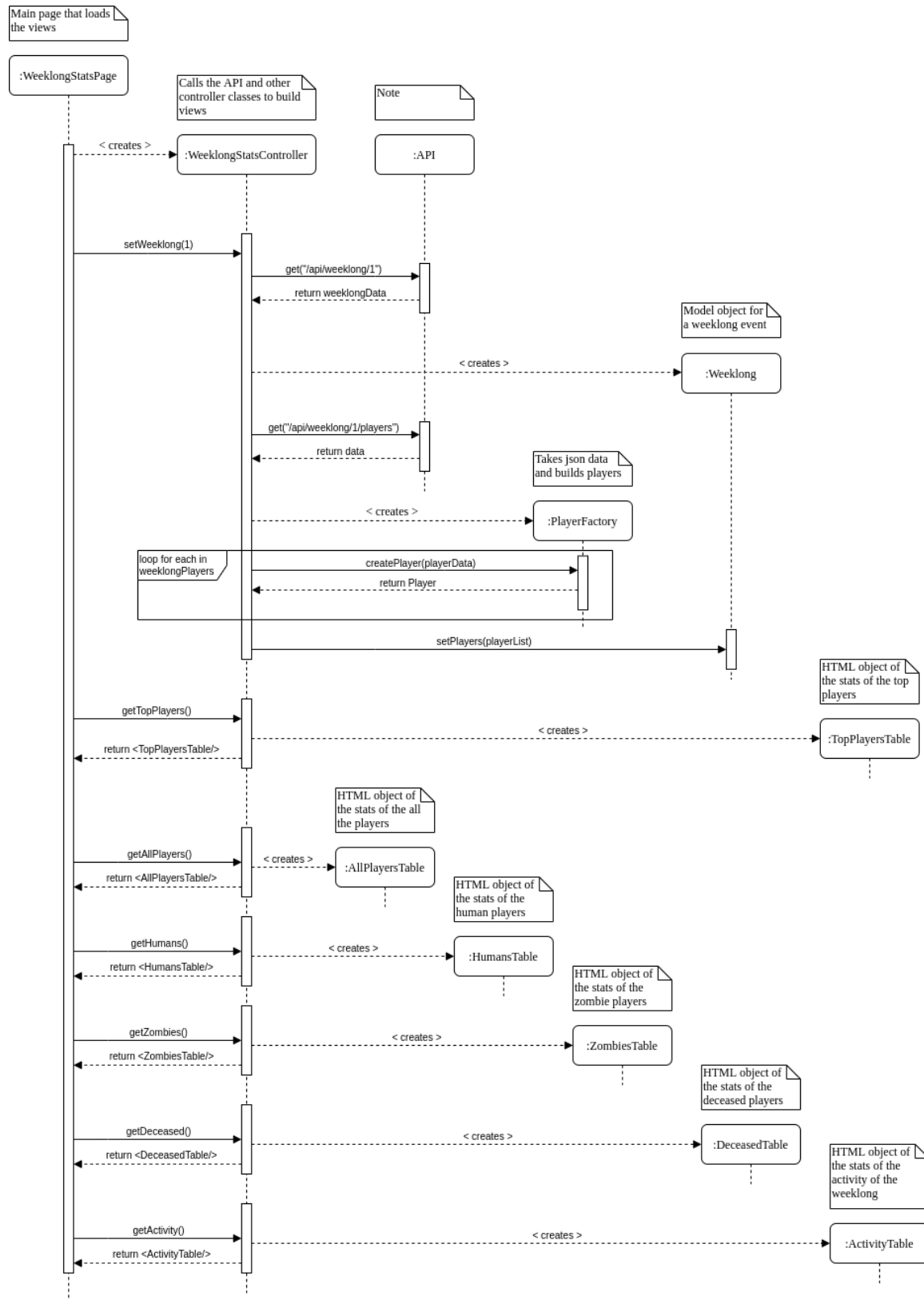


## User Interactions/UML Sequence Diagrams

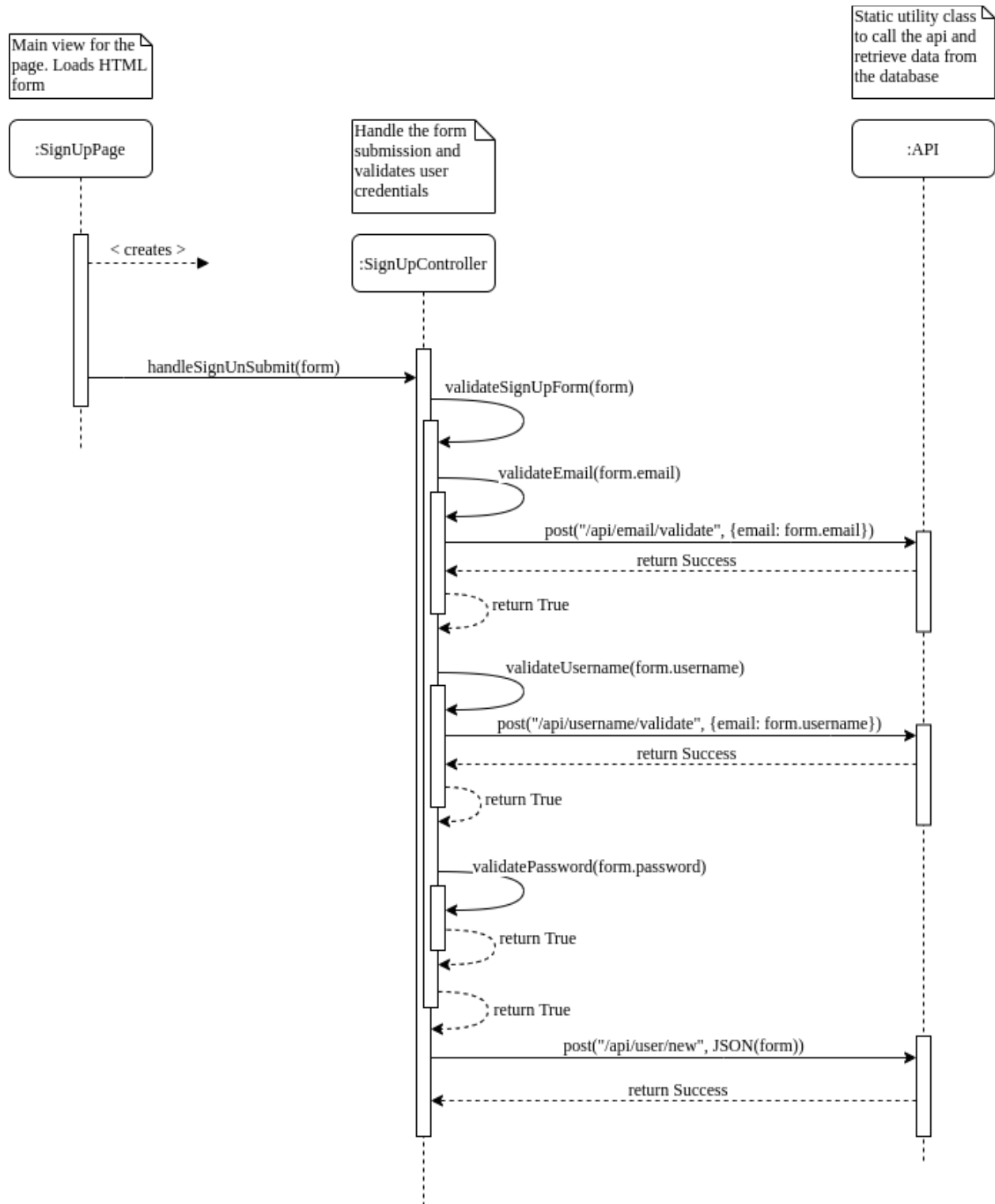
A player enters a valid code into the code entry page.



Visitors can visit the weeklong stats page and see all the different player table types and the player scores.



Visitors can signup for an account. This is assuming the visitor's inputs are all valid.



## UML Class Diagram

Color codes for UML diagram:

Green - Controller or Service class

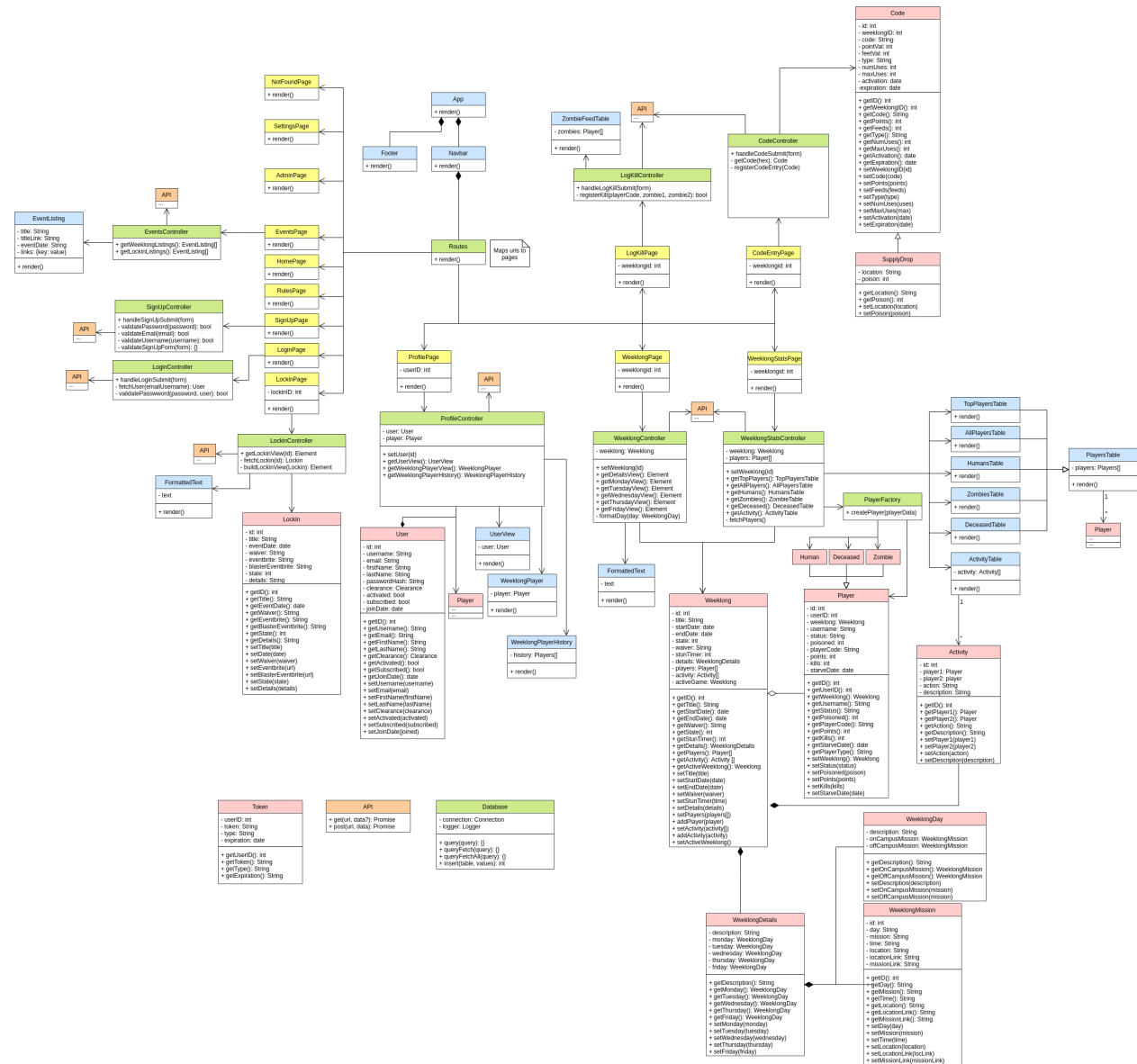
### Blue - View class

Yellow - Page view class

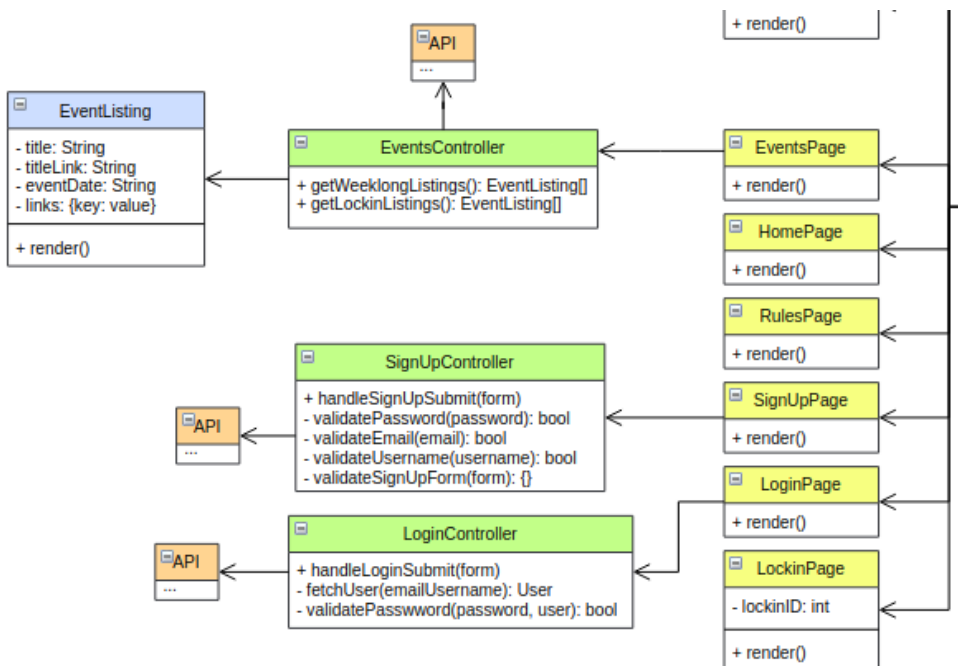
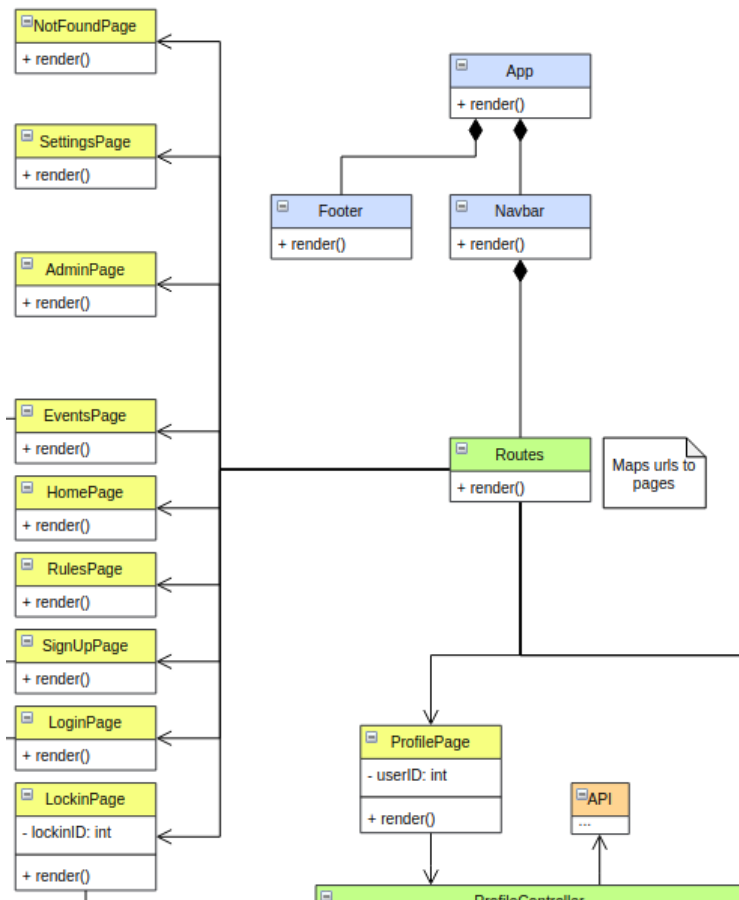
## Pink - Model class

## Orange - Static class

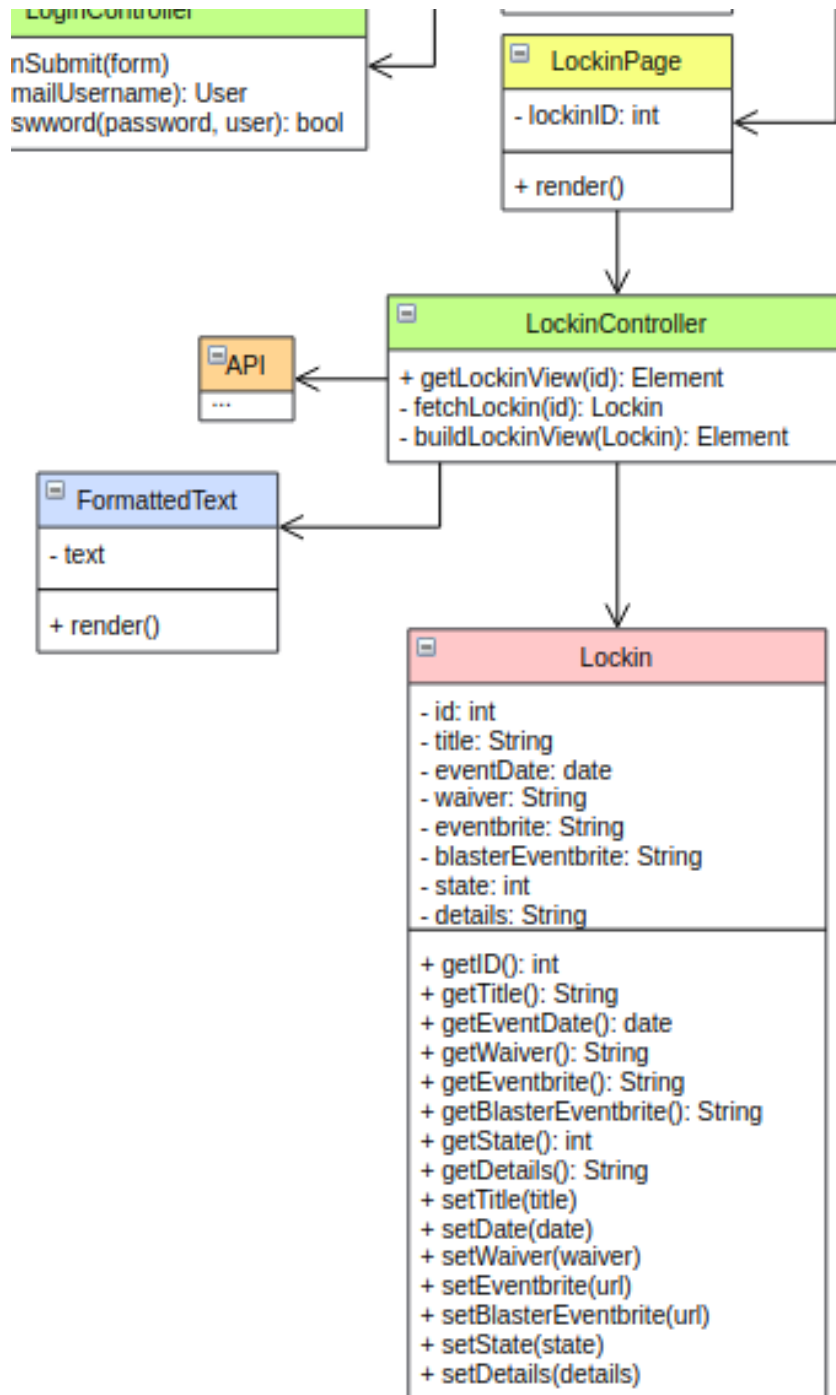
## Full Diagram



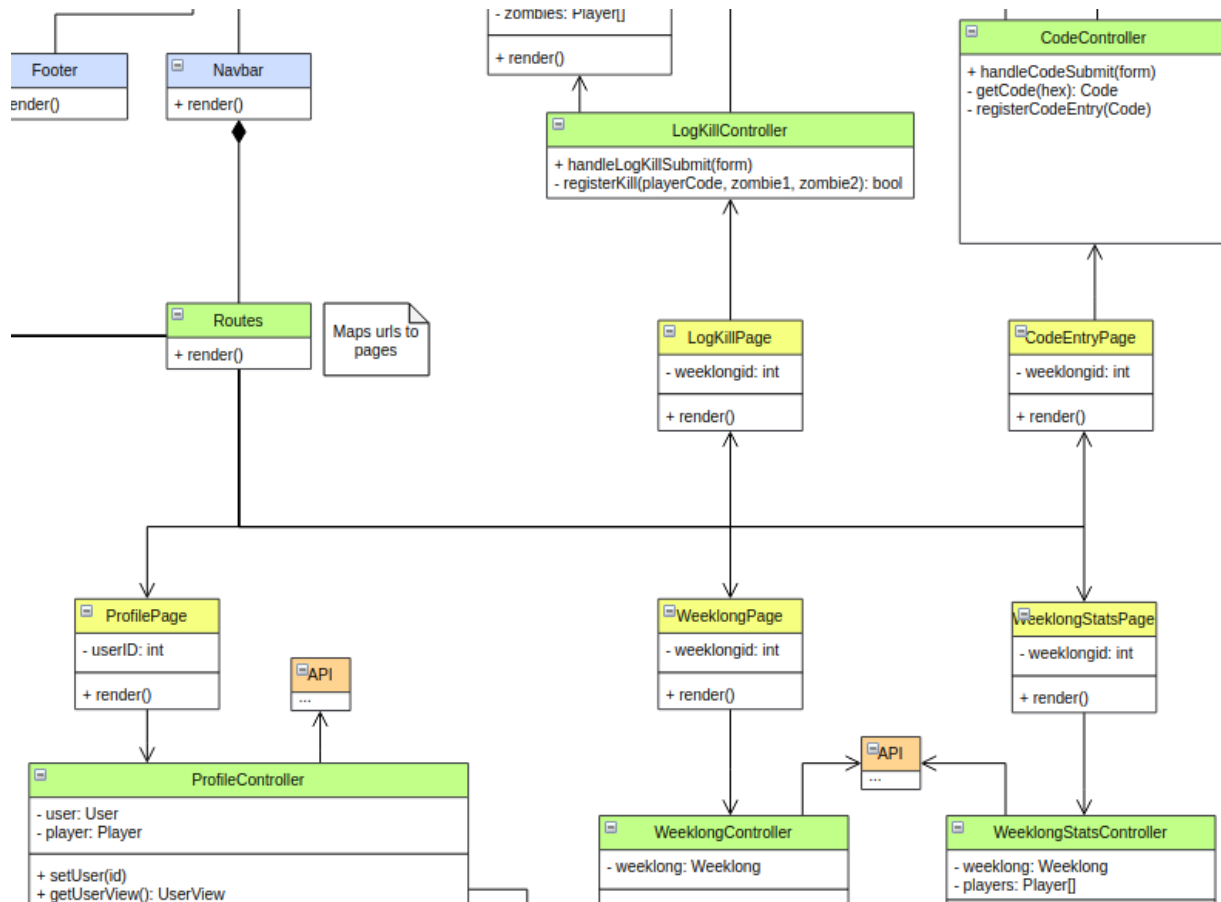
## Main App start and pages



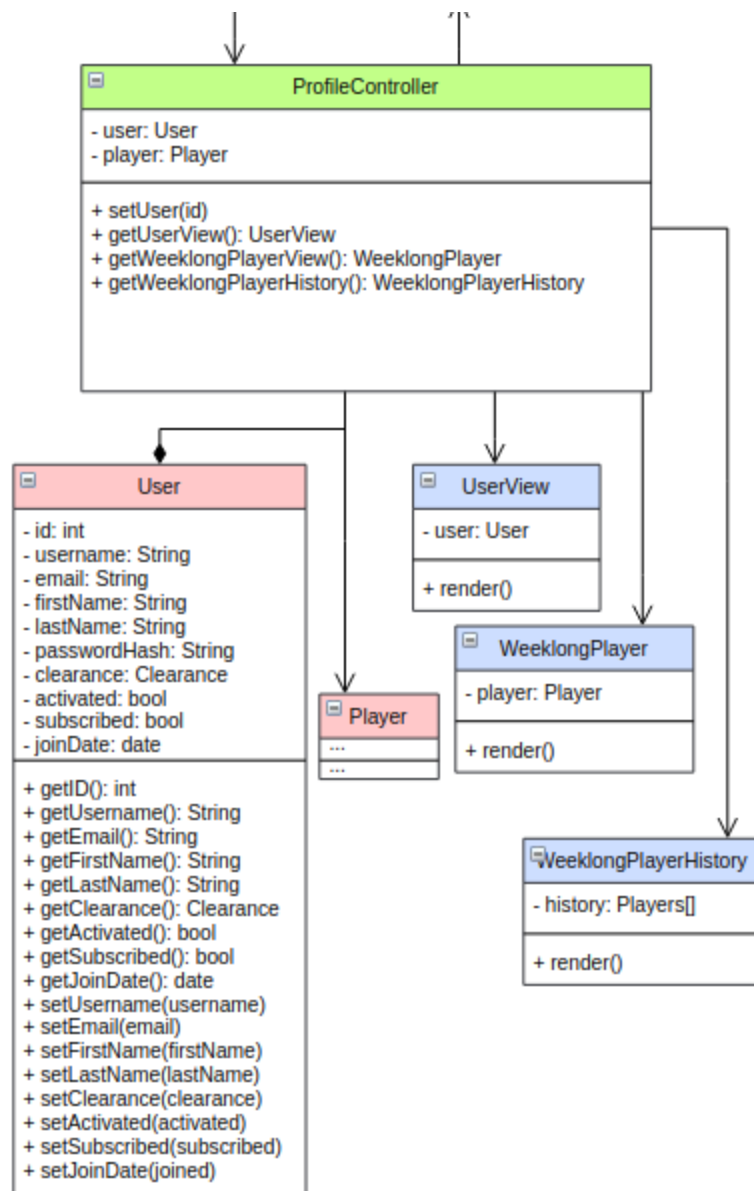
## Lockin classes



## Profile and weeklong pages

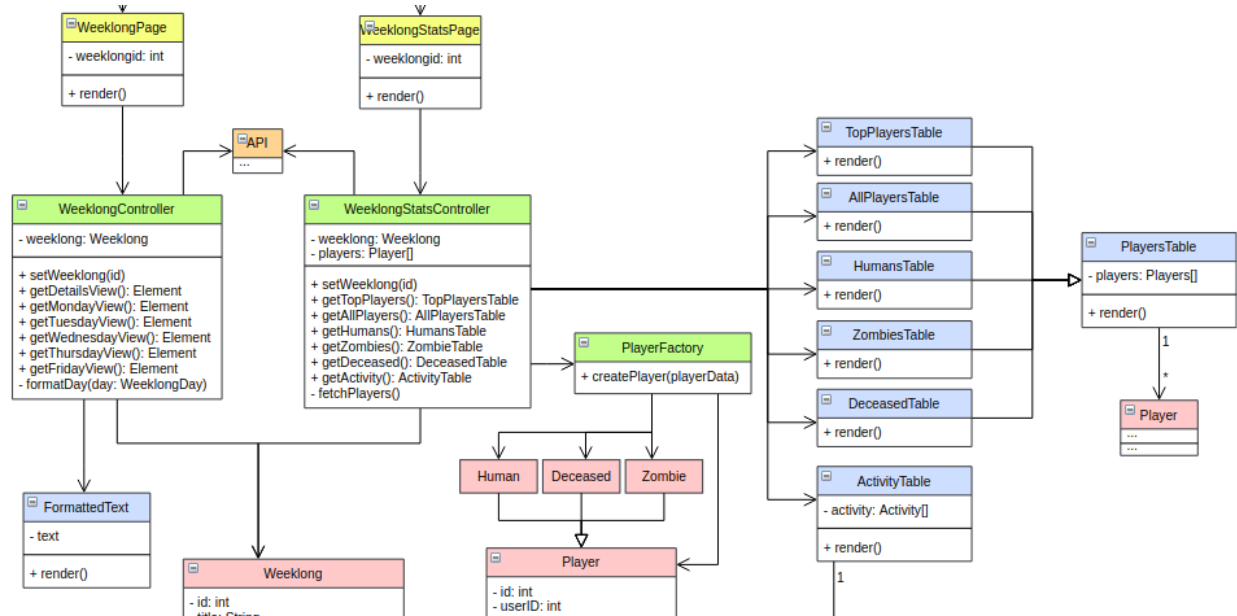


## Profile classes

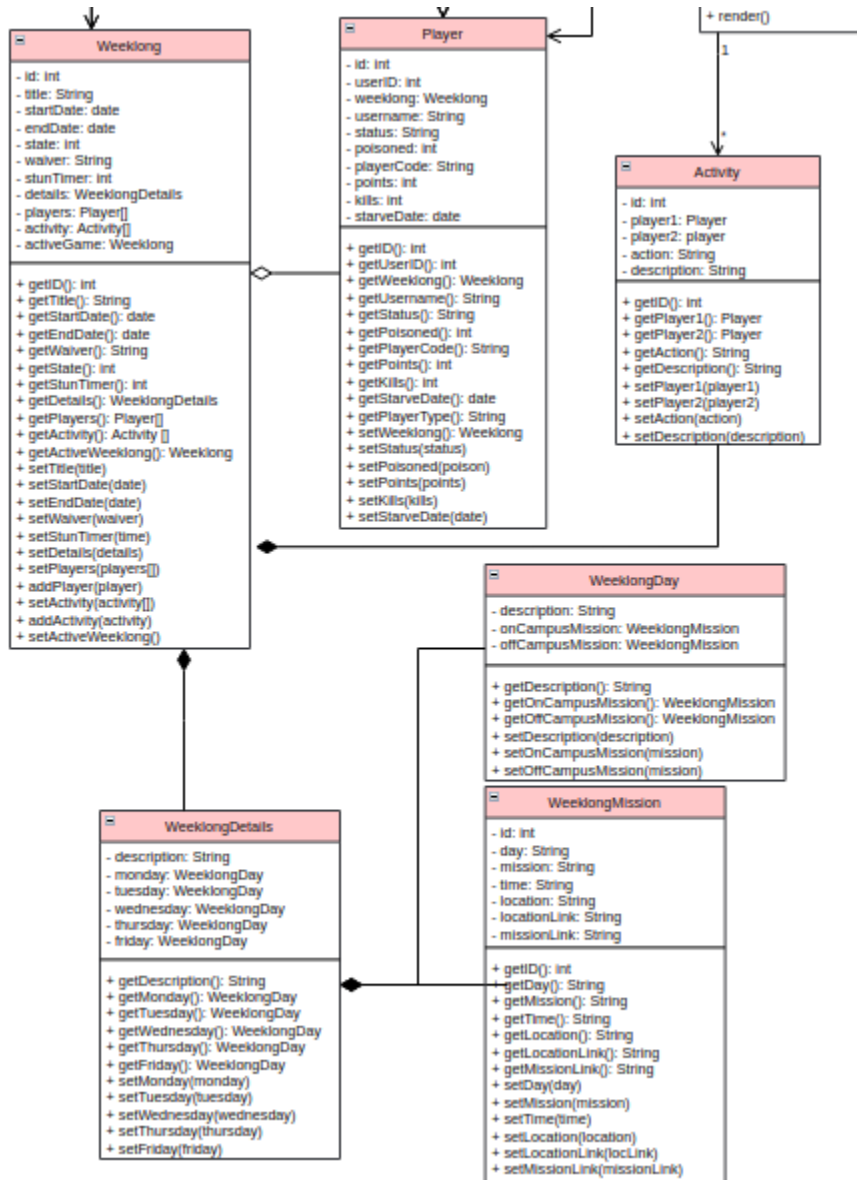




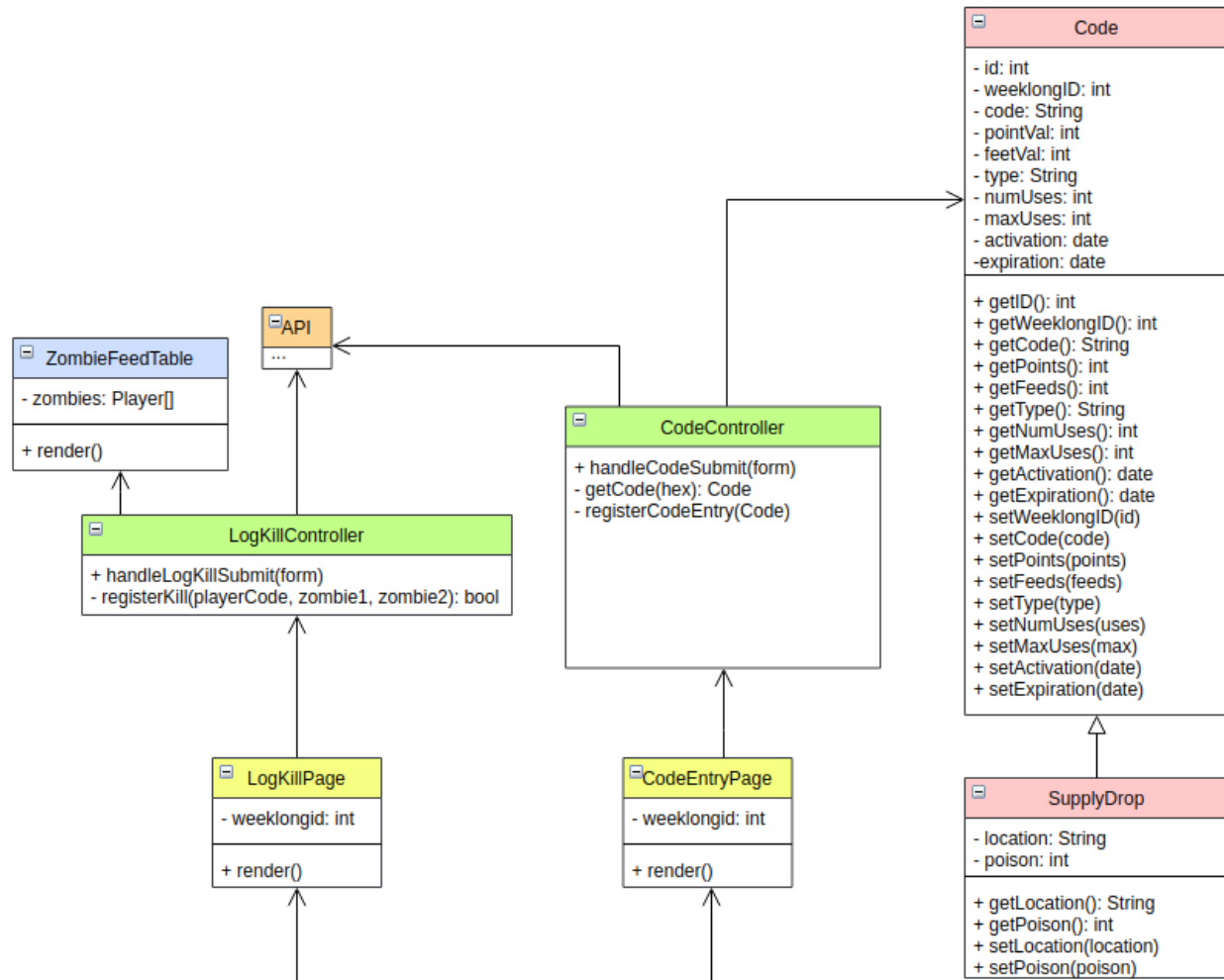
## Weeklong View and Controller classes



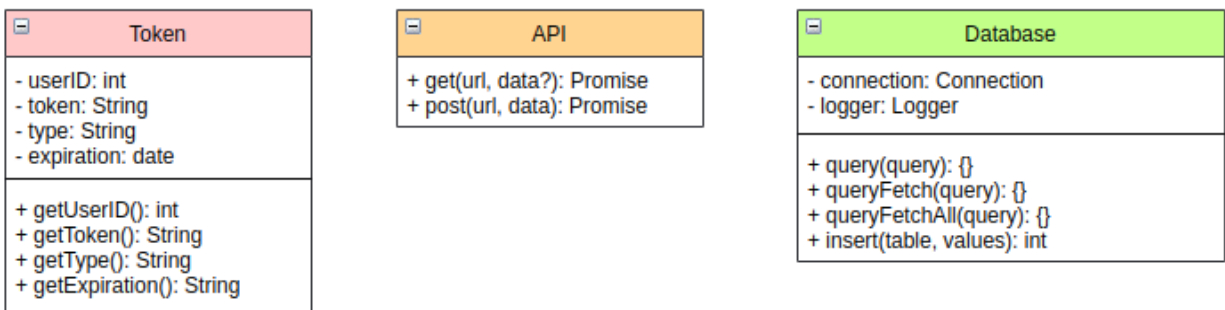
## Weeklong Model classes



## Code entry and log kill classes



## Backend and other classes



## Pattern Use

The three patterns used in this project are the Singleton pattern, the MVC pattern, and the Factory pattern.

### Singleton Pattern

The Database class on the backend of the application will be used for the singleton pattern because we only ever want to have one connection to the database.

### MVC Pattern

In the UML diagram, the colors show the model, view, and controller classes. The green classes are controllers, the blue and yellow classes are views, and the pink classes are the models.