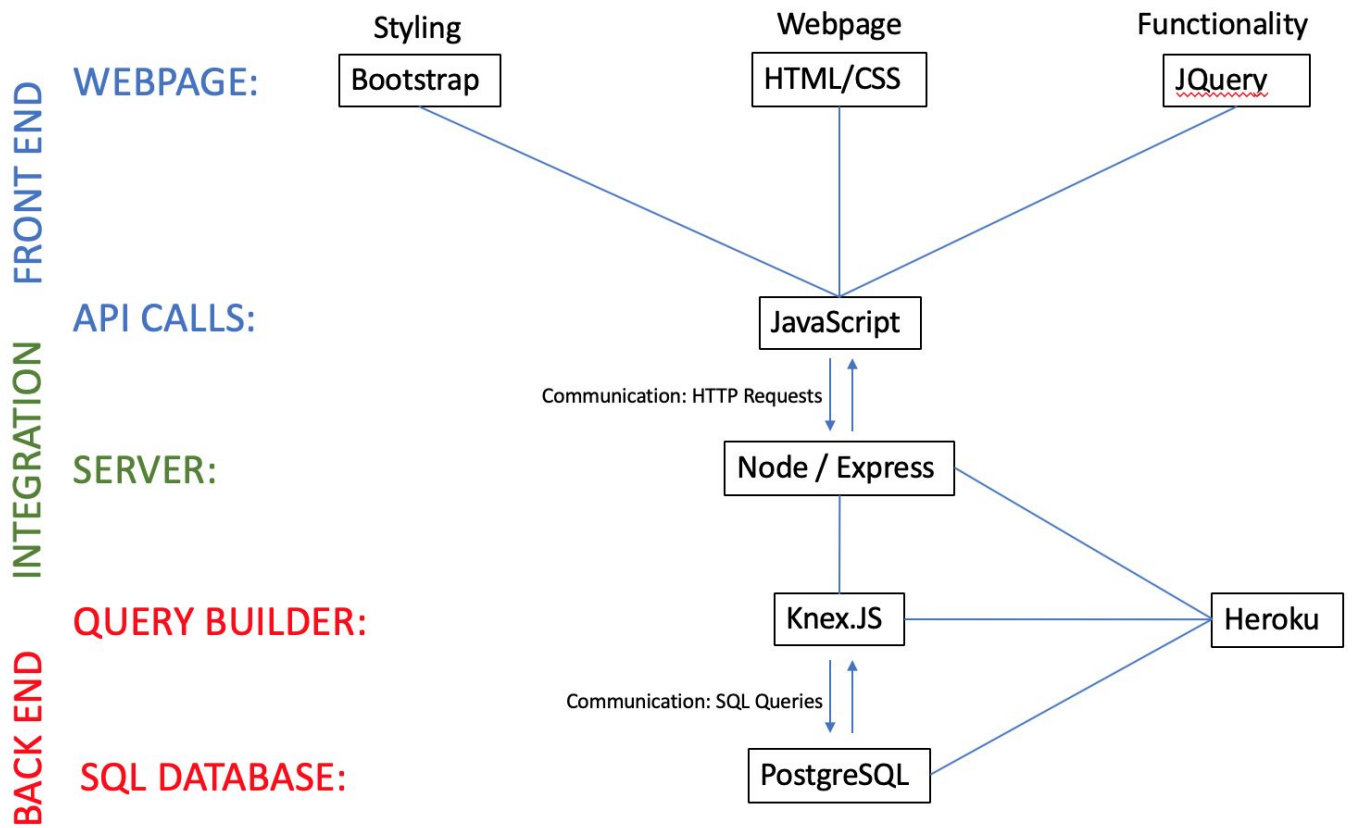
TEAM MEMBERS: Aidan Baack, Adam Spiers, Jaykob Velasquez, Yicheng Yi, Nicholas Lescanic

APPLICATION ARCHITECTURE:



The front end will be made using Bootstrap and JQuery. It will integrate with our server through JavaScript AJAX calls in order to communicate with the server through HTTP requests. The server will be made using Node with Express. The server will communicate with the database using SQL queries which will be built using the Knex.JS library. The integration layer and back end will be hosted on Heroku, and we already have this working. The front end files will be served by Heroku as well. The site can be viewed here.

FEATURES:

- LOGIN PAGE (Priority 1): The login page will allow a user to type in their username and password to quickly login to their account. The account holds information about preferences and statistics, so these preferences will then be applied to the relevant aspects of the application.
- REGISTRATION PAGE (Priority 1): The registration page will allow a user to create a new account through an intuitive table where they can input their name, email, and preferences for what they would like to see displayed on the information screen.
- SETTINGS PAGE (Priority 1): In the settings, a user can modify their preferences and view some of the statistics about their account. The information displayed on the info screen can be changed, and a user can find their average time to completing puzzles, how often they snooze the alarm, etc.
- GAME (Priority 2): a Hangman-style game that the user must complete in order to turn off the alarm. Each puzzle will be drawn from a database of old Wheel of Fortune puzzles, so there will be a clue and then a phrase that the user must populate with letters from an on-screen keyboard. A puzzle should take no more than 10-15 seconds to complete, turning off the alarm.

- INFO SCREEN (Priority 2): After completing the game and turning off the alarm, the user will be taken to an informational display that will pull in information from various sources via API calls and display it all in 4-6 tiles. Possible info sources include: Today's weather; top stories from Google News; top trending social media posts; stock ticker; Colorado mountain snow report; trivia; and more TBD. User will be able to customize what information appears.
- ALARM (Priority 2): a simple alarm clock that allows the user to set an alarm, establish standard snooze length, etc.
- ACCOUNTS (Priority 3): Each user will have an individualized account, accessed by username and password, that will store the user's preferences regarding snooze settings, informational display settings, and alarm settings.
- SCOREBOARD (Priority 3): A user will be able to see how well they do in comparison to all other users of the app through a scoreboard. The score will come from how quickly they solve the puzzle and how many wrong answers they have.

REQUIREMENTS: Please refer to the spreadsheet "ProjectMilestone4 Requirements Doc.xlsx" in this folder for our updated list of Functional and Non-Functional requirements.

Additionally, the information is included below. The features have the same priority as they do in the previous section. A further breakdown of the priority of sub features can be found on our Kanban board for the project, which we will gladly link upon request.

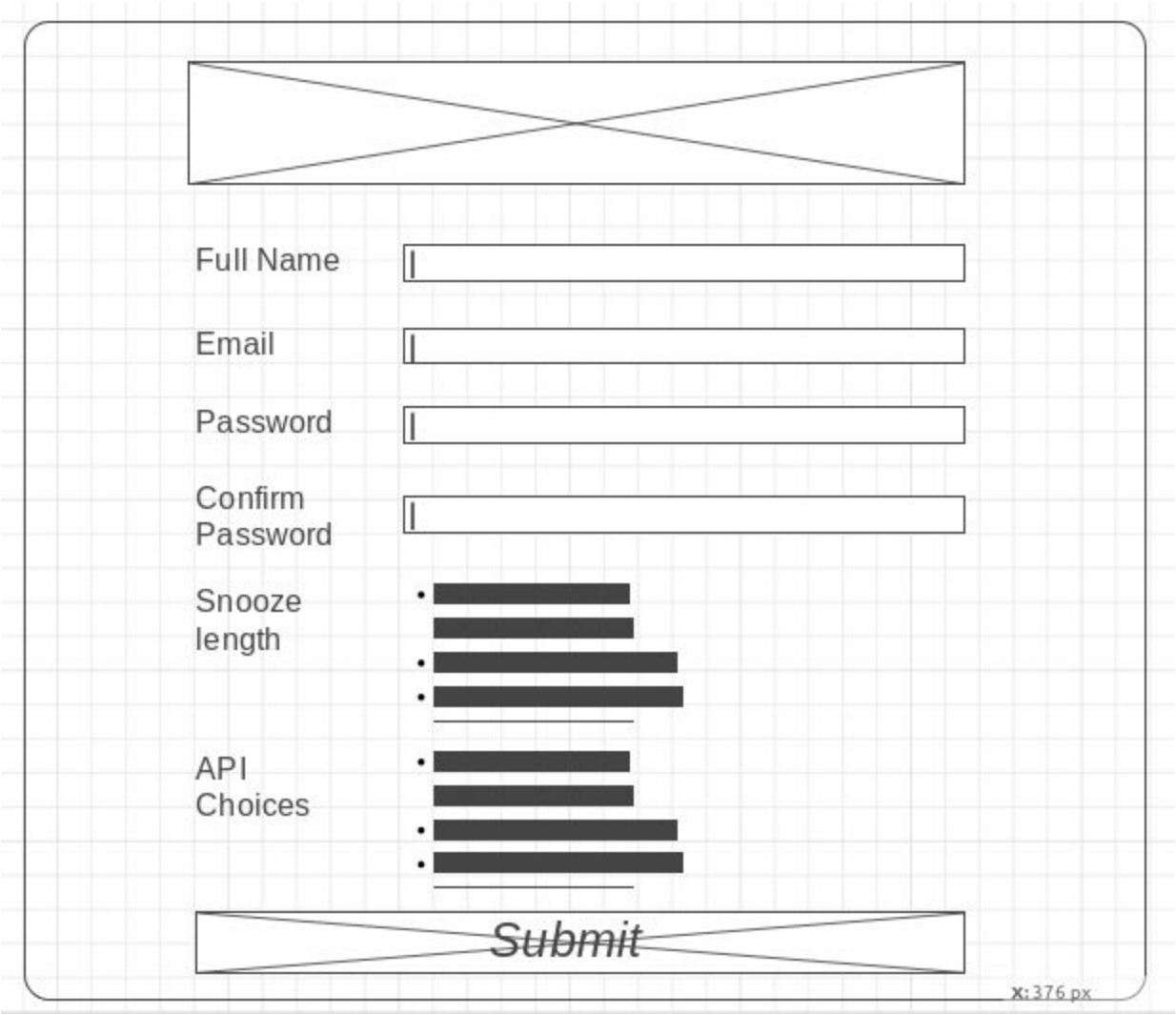
Feature	Functional Requirements	Non-Functional Requirements
ALARM	Ability to set the alarm	Digital representation of current time
	The alarm rings when it is supposed to	Sound of the alarm
	Customizable snooze length	Store user alarm in database
	Limit of 1 snooze per user per day	If user opts to play game, new game is loaded
	Alarm turns off when game is completed	Snooze activates timer, when timer ends alarm is reactivated and new game initiated
	Option to play hangman	
GAME	Clue displayed at top of screen	Database of Wheel of Fortune puzzles
	Blank puzzle boxes displayed beneath clue	Random selection of one puzzle per alarm, minimizing repeats
	Full keyboard displayed beneath blank puzzle boxes	Equation to compute user points base on puzzle completion time/wrong letters
	When incorrect letter selected, letter is disabled on keyboard and turned red	User point total stored in database
	When correct letter selected, letter is disabled on keyboard, turns lighter shade of green	
	Correct letter selections populate blank puzzle boxes for all matching letters	
	When puzzle is complete, alarm noise is turned off	
	When alarm noise turns off, user has option of using sole daily snooze, playing another game, or going to info screen	

	If user chooses to play another game, game restarts	
	If user chooses snooze, alarm is set again for X minutes in future, without snooze option at end	
	Otherwise, user is taken to info screen	
	User is awarded points based on puzzle completion time and number of wrong letters selected	
	User's total points are displayed on scoreboard	
INFO SCREEN	Attractive looking display of 4 to 6 tiles	Have all API calls pre-composed
	Each tile will hold up-to-date and relevant information chosen by user	Convert information from API calls to displayable text or pictures/GIFs
	User will have at least 10 info options from which to choose 4 to 6	Have API keys where necessary
	Options will include:	
	Weather.gov today's and tomorrow's weather	
	Google News top 4 stories	
	Social Media	
	Stocks (DJIA, S&P500, NASDAQ)	
	Snow report for 4 geographically dispersed Colorado ski resorts	
	This Day in History facts (events, births, deaths)	
	Famous Quote of the Day	
	NASA Astronomy Picture of the Day	
	Random GIFs (cat, dog, cute, funny, fail)	
	Two buttons at bottom of screen, Exit and Game:	
	Game button takes user back to play another game without the alarm sound	
	Exit button exits the application	
ACCOUNTS	Account accessed by username and password	Usernames and passwords stored on database, accessed through hash function
	User's preferences will be stored in their account and will be modifiable by the user	User preferences regarding the following will be stored in their user profile on database:
		Snooze length
		Info Screen choices
		Alarm settings
		and more TBD…

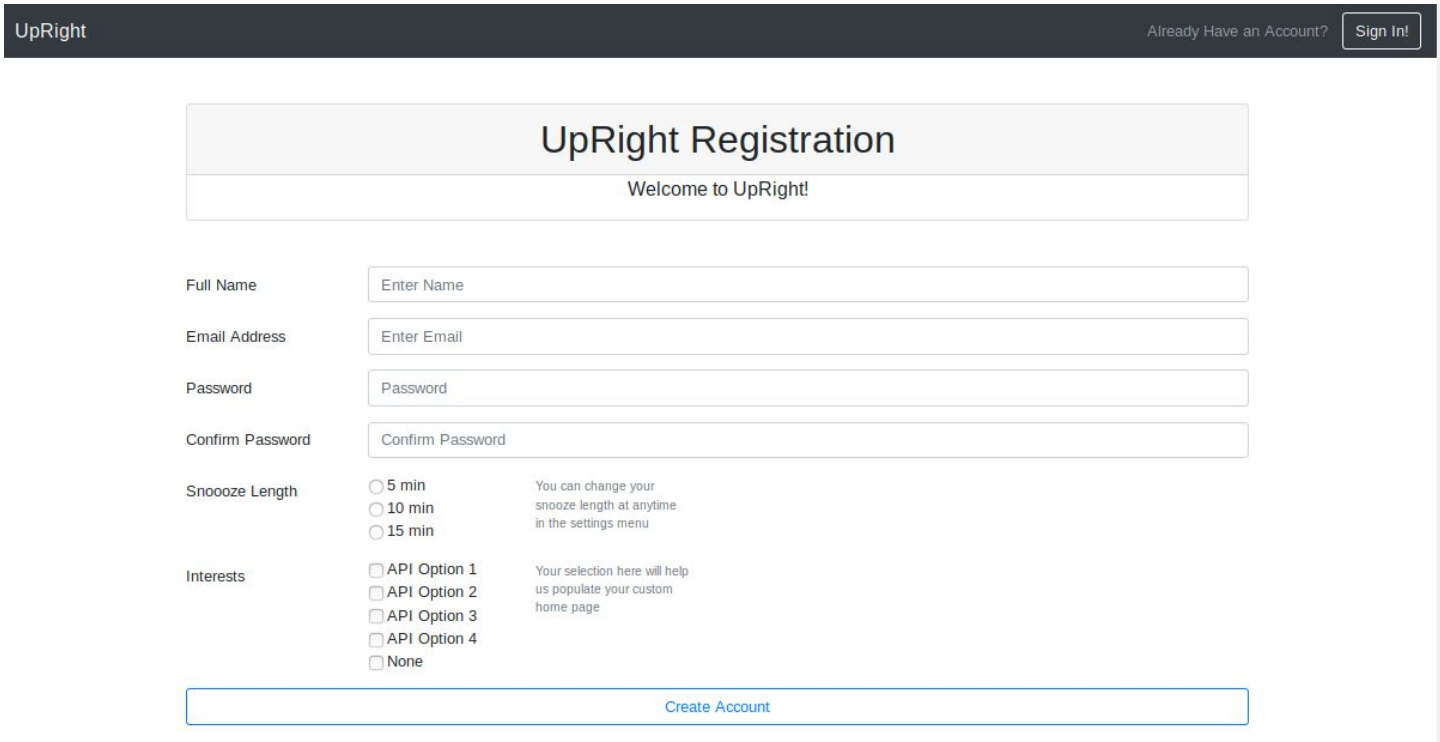
LOGIN PAGE	User can access the application by entering their username and password	Login information will need to be sent to the server
	New users can navigate to a registration page	If the login is correct then the server will need to reply that it is correct
		A cookie will need to be created storing that the login is correct
		The user should be redirected to an alarm page
REGISTRATION PAGE	Enter email and create password	Registration information will need to be sent to database
	Input boxes should be colored based on if they are correct (red for incorrect and green for correct)	Server will need to enter user information into the database and hash the password
	Choose snooze length and choose which information should be displayed on the information screen	Server will need to check that the email address is unique
SETTINGS PAGE	Change what information should be displayed on the information screen	App will need to send requests to the server to change the information
	Click a button to log out of the account	Cookie with login info will need to be deleted if user logs out
	Change password or account name	Server will need to input updated information into the database
SCOREBOARD	Displays user ranking, and the top 10 other users	App will need to make a request to the server to get the latest top scores
	Will Populate as user plays	Server will need to make a database request that will pull the top users
		Server will need to send information to the app and the information will need to be displayed on the page

FRONT END DESIGN:

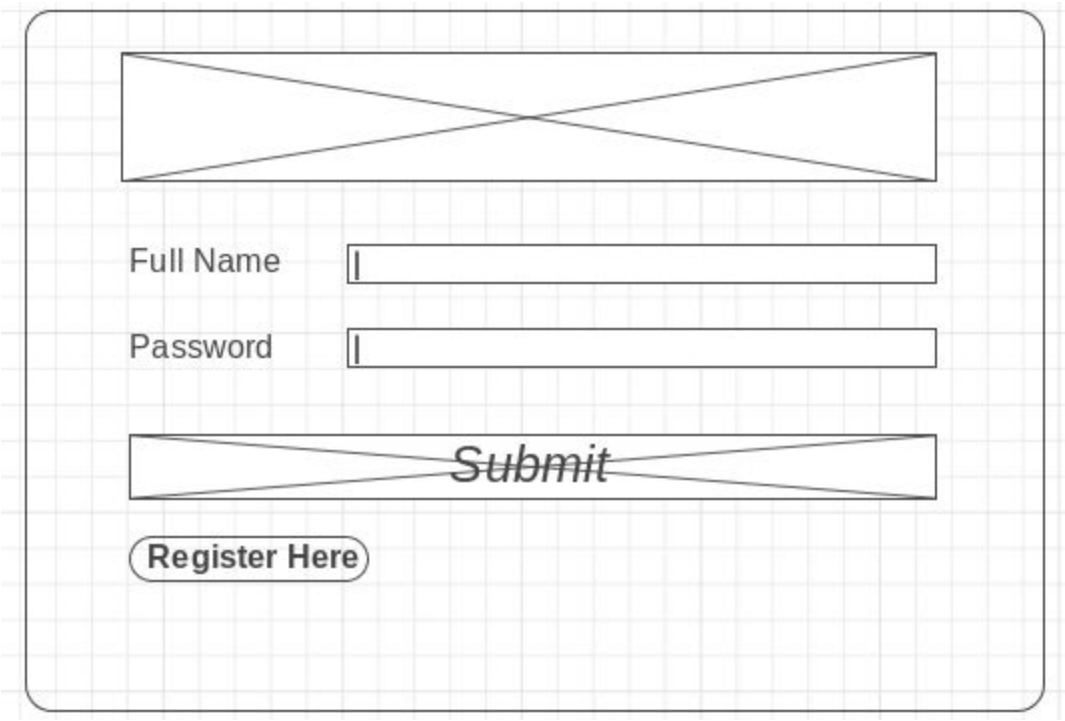
Registration Page Wireframe:



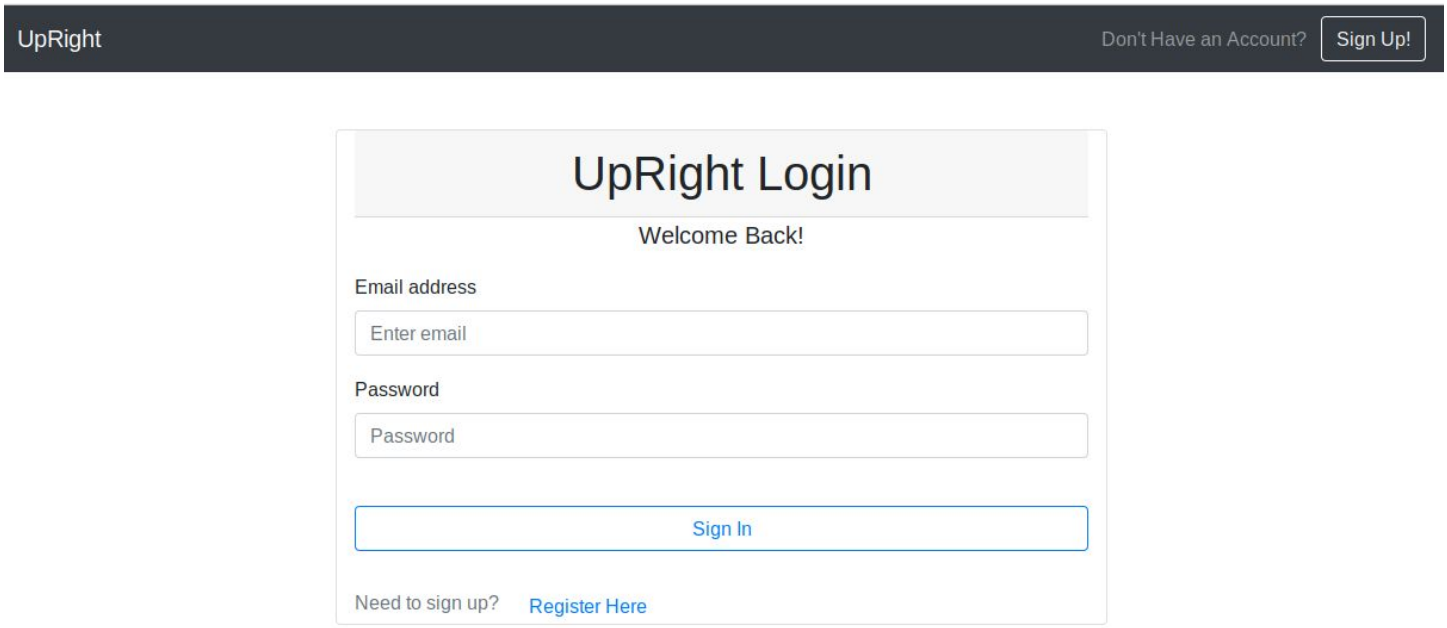
Current Progress on Implementation:



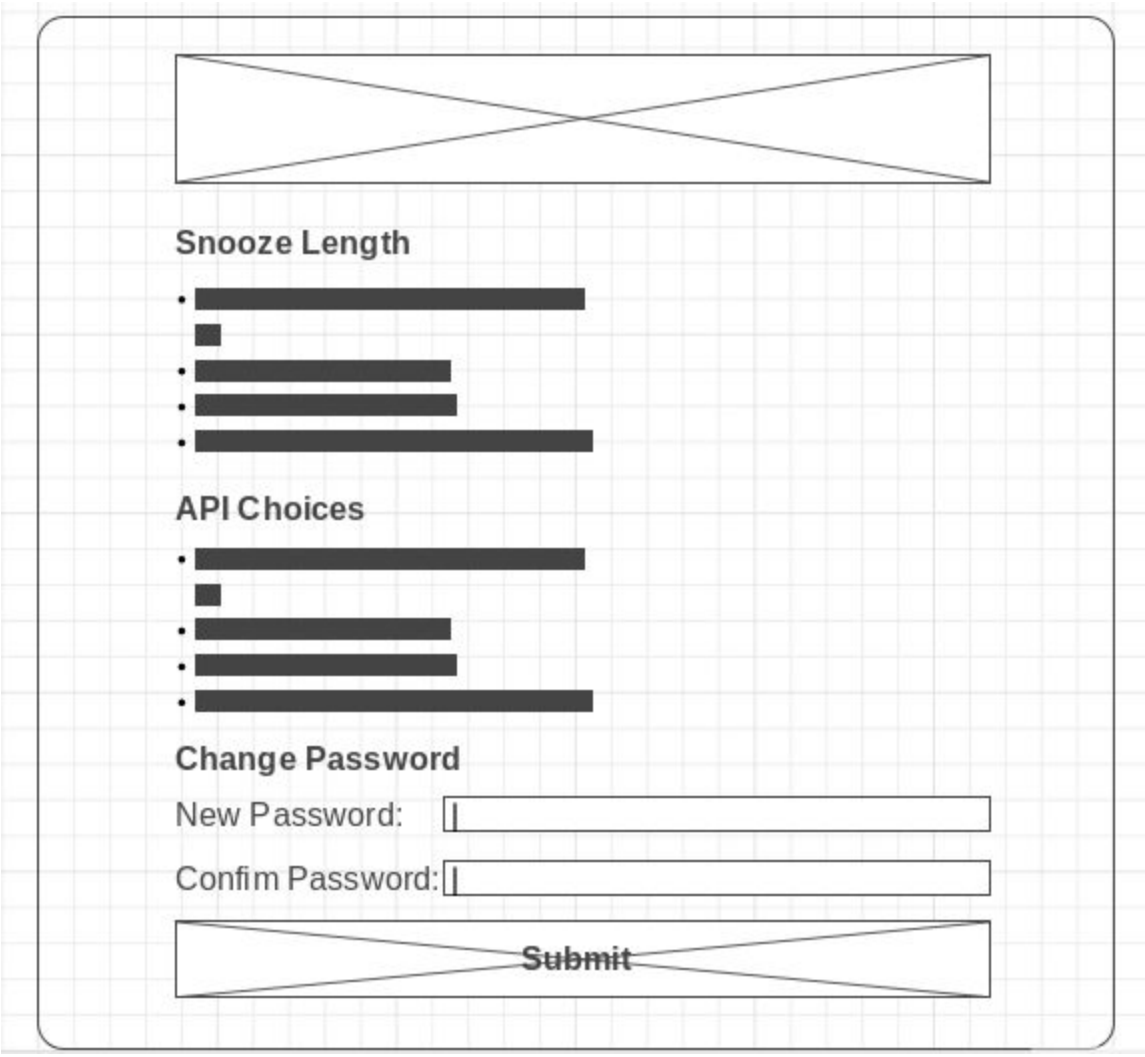
Login Page:



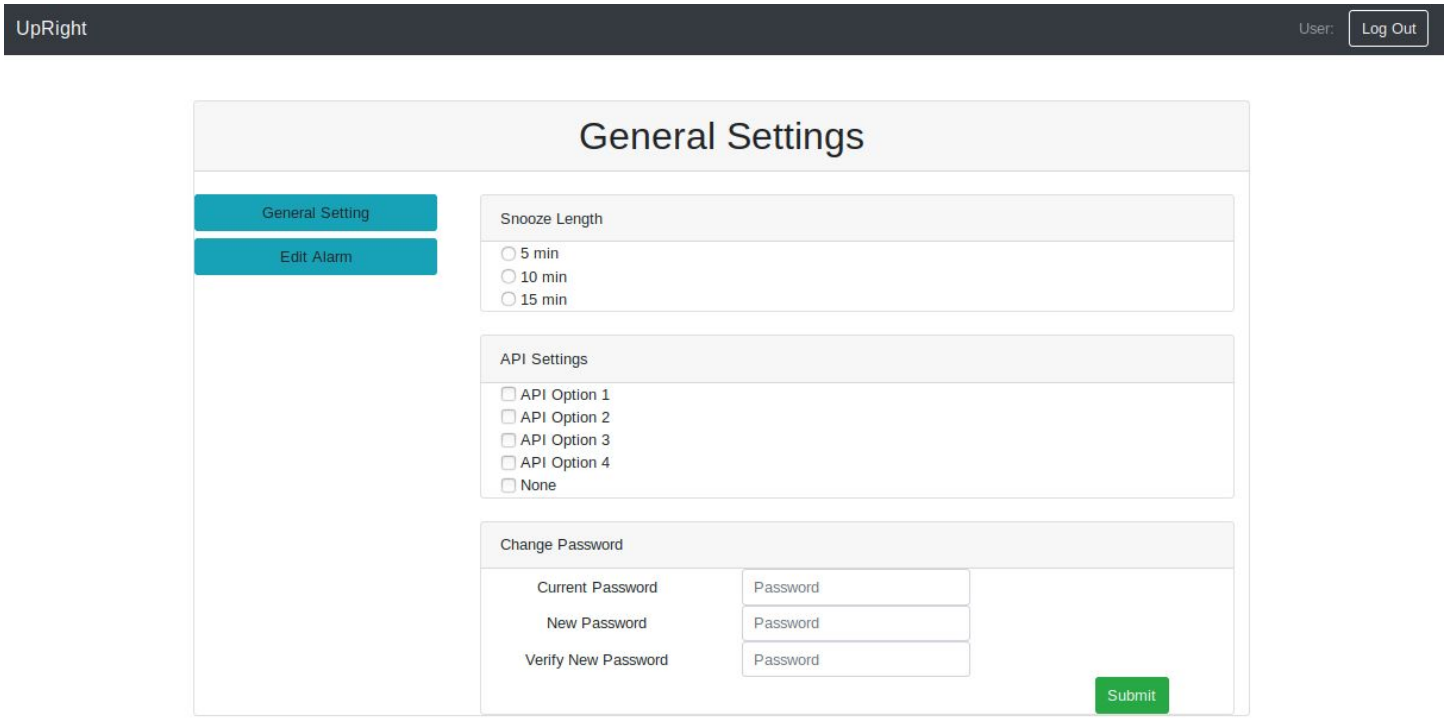
Current Progress on Implementation:



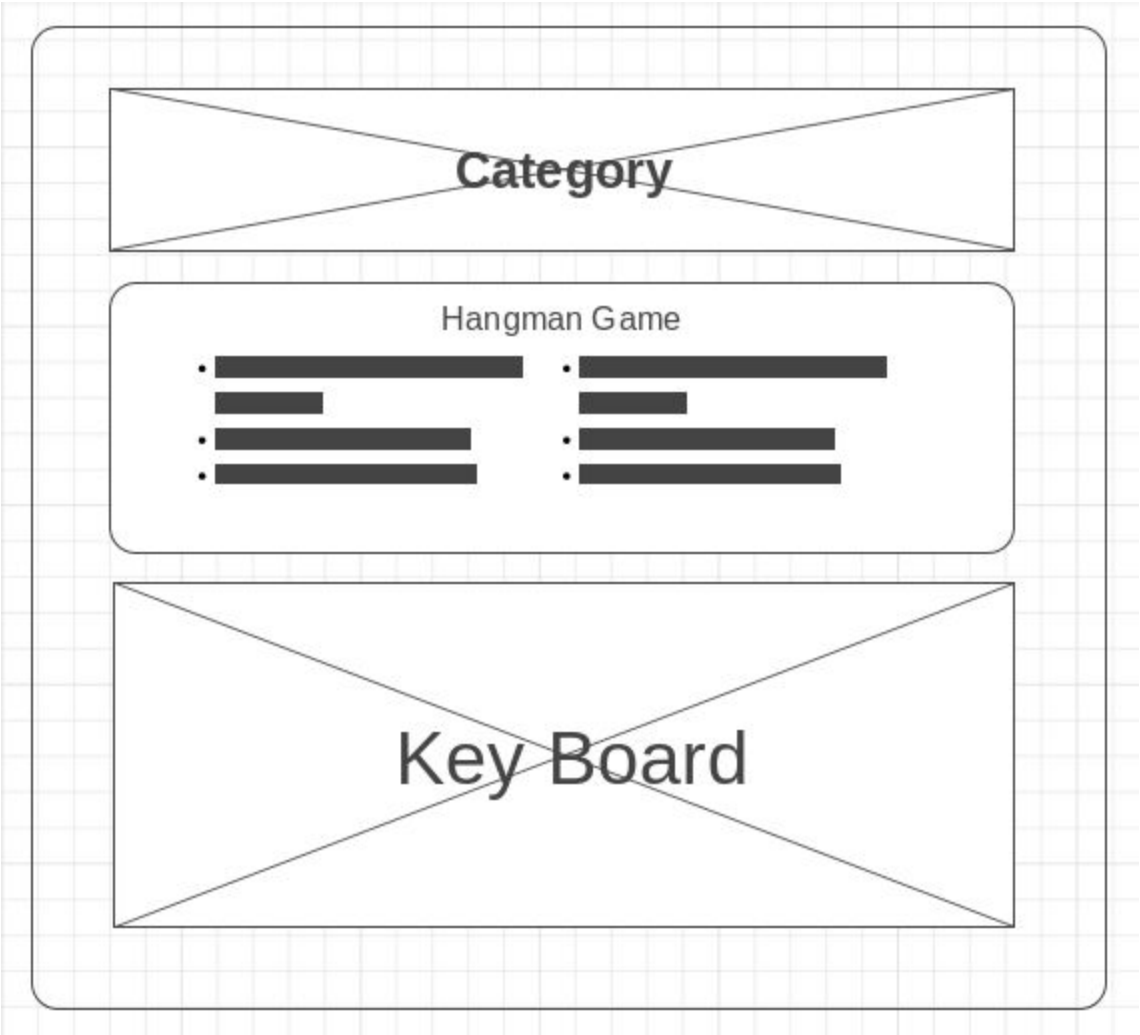
Settings Page:



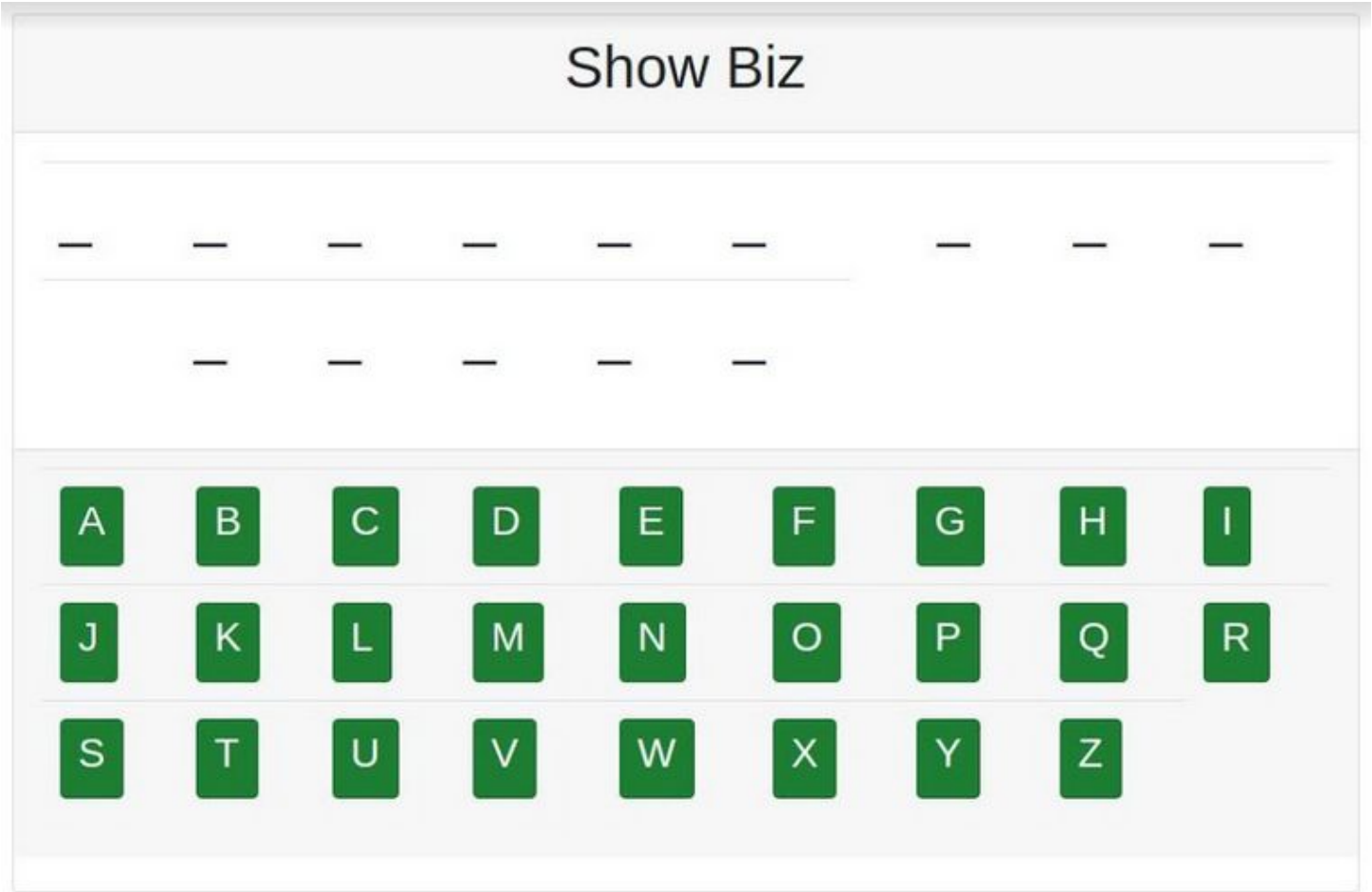
Current Progress on Implementation:



Hangman Game Page:



Current Progress on Implementation:



WEB SERVICES DESIGN:

Google News “newsapi.org” API:

- Key: 85582……..
- URL: https://newsapi.org/v2/top-headlines?country=us&apiKey=85582……..
- Returns JSON, will pull “title” and “url” from top 4 stories to create HTML links on Info Screen

Weather.gov API:

- Key: N/A
- URL: https://api.weather.gov/points/*latitude*,*longitude*/forecast
- *Latitude* and *longitude* will be populated from ZIP code database using user ZIP code. Returns JSON, will pull “temperature”, “shortForecast”, “name”, and “windSpeed” for today and tomorrow, display in card.

Quote of the Day API:

- Key: N/A
- URL: https://quotes.rest/qod
- Returns XML, will pull text from “quote” and “author” tags and form quotation in card.

This Day in History API:

- Key: N/A

- URL:
    - http://api.hiztory.org/date/event/"+mm+"/"+dd+"/api.xml
    - http://api.hiztory.org/date/birth/"+mm+"/"+dd+"/api.xml
    - http://api.hiztory.org/date/death/"+mm+"/"+dd+"/api.xml
- *mm* and *dd* will be populated with today's date using Date() in JS. Returns JSON with facts.

NASA Astronomy Picture of the Day:

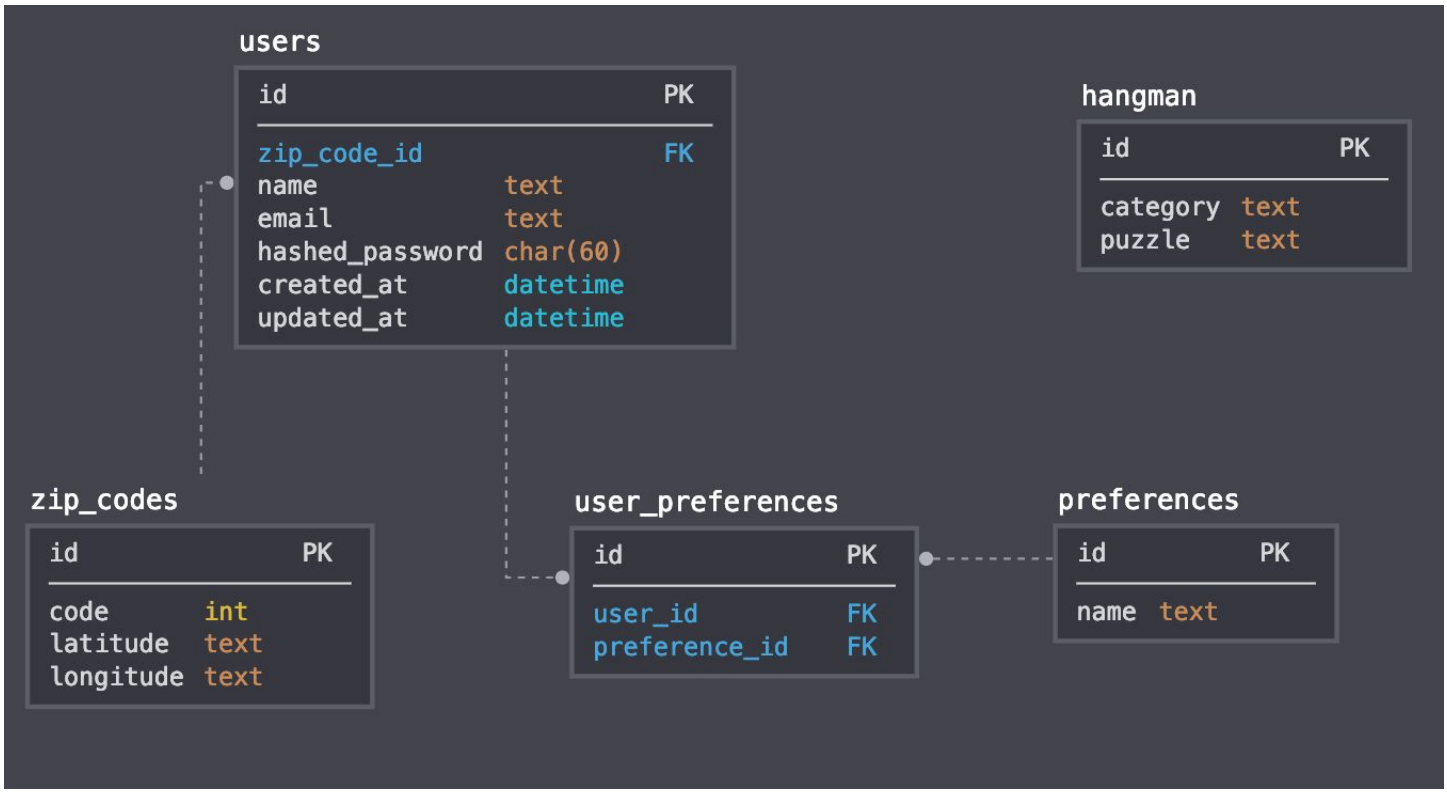
- Key: aOGxR.....
- URL: https://api.nasa.gov/planetary/apod?api_key=aOGxR.....
- Returns JSON with "url" that is link to image, "title" of image.

Tenor GIF Database API:

- Key: DUOD1.....
- URL: https://api.tenor.com/v1/random?q=" + *query term* + "&key=DUOD1.....&limit=1&ar_range=standard"
- *query term* is chosen by user to be either cat, dog, cute, funny, or fail. Returns JSON with link to random GIF, which will be displayed in card.

BACK END DESIGN: We will be using PostgreSQL along with Node.JS for creating the server. The database will need to store user information, user preferences for which data to display after winning the game, hangman puzzles for the game, and zip code information.

- The zip codes will be necessary for interfacing with the weather API we are using because the weather is pulled based on latitude and longitude. It will be better for the user experience if a user can simply enter their zip code when creating an account, and the latitude and longitude coordinates will automatically be acquired from our server.
- For preferences, there is a preferences table that will store which preferences are possible for a user to select. These preferences will be different types of information that will pull up after a user wins the game (Weather, GIFs, quotes, etc.). Each preference will have an id as well as the name of what it is.
- In the user_preferences table, the preferences that a user has selected will be saved. Each row will have a single user id and a single preference id, denoting that the user has selected the given preference. There will be a row for each preference that a user selects.
- The other information in the table can be found by examining the database diagram below.


users
id PK
zip_code_id FK
name text
email text
hashed_password char(60)
created_at datetime
updated_at datetime
hangman
id PK
category text
puzzle text
zip_codes
id PK
code int
latitude text
longitude text
user_preferences
id PK
user_id FK
preference_id FK
preferences
id PK
name text