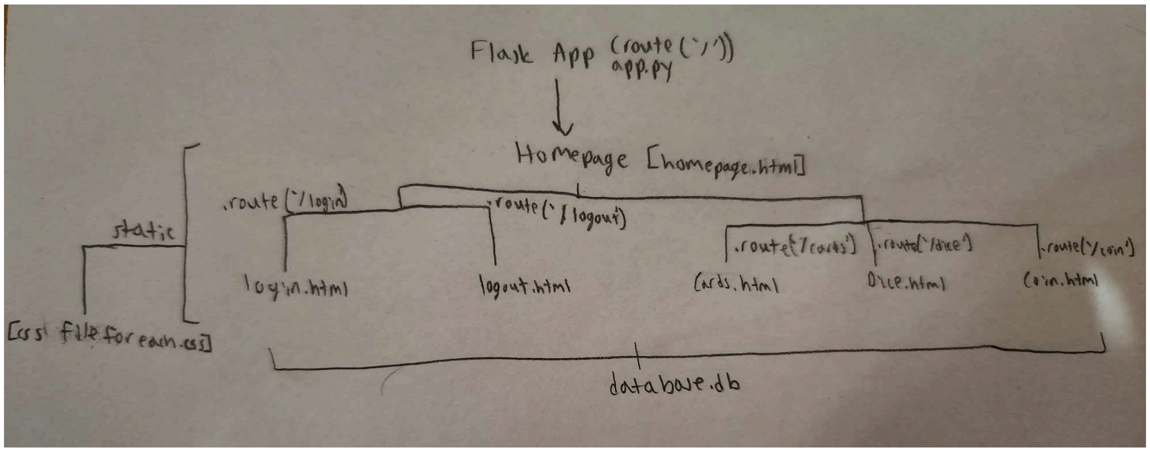
YACK - Aditya Anand, Suhana Kumar, Margie Cao, Stella Yampolsky
Software Development
P01 - Casino Simulator
Target Ship Date: 2024-12-17

- **Roles**
  - Flask - Aditya
    - Making routes and different functionalities for each
  - SQLite3 - Stella
    - Storing unique usernames and their respective passwords
    - Keeping a record of in-game currency for each user
  - API implementation - Margie
    - Retrieving data from APIs
    - Using API's to code specific games
  - HTML + CSS - Suhana
    - Creating templates for each HTML file
    - Using tailwind as the framework

- **Component Explanation**
  - **Flask**: Acts as the central hub connecting the frontend, backend, and database. It handles user requests and acts as a connector.
  - **SQLite3 Database**: Stores user accounts, game statistics, and balances. Flask interacts with the database to retrieve and update data based on user actions.
  - **Game Modules**: Each game module (Card Game, Coin Flip, Dice Simulator) interacts with its respective API to process game logic. The results are sent to Flask for storage in the database and displayed on the frontend.
  - **Front-End Templates**: Provides the interface for user interaction. Pages like home.html display personalized stats and links to games based on database queries handled by Flask.
  - **APIs**: Serves as the backbone for individual games, providing outcomes (e.g., card draws, coin toss results, dice rolls) to the game modules. These outcomes are used to update the database and user balances.



- **Database Organization (Casino.db)**
    - User table: unique username, password

Username	**Password**
bart	simpson
marge	simpson
homer	simpson

Disclaimer: these passwords are not very secure…

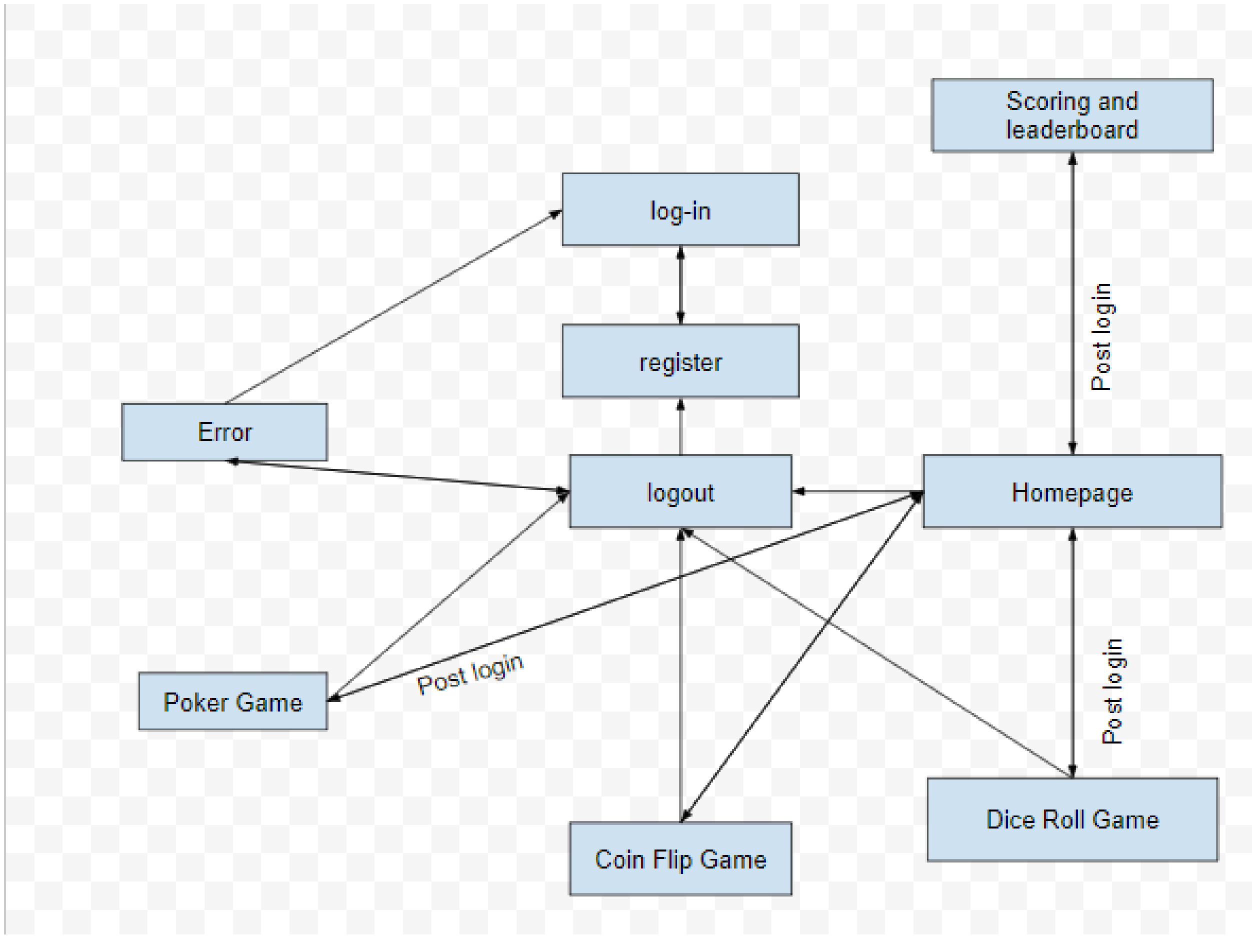
-
    - Currency table: username, currency
        - Retrieve currency by username
        - Currency is updated through playing games
    - When a user is added, they start with 100 (dollars! You're rich, just don't gamble)
    - Each daily login adds 50 dollars

Username	**Currency**
bart	100
marge	60
bart	150
homer	100
homer	0

- **Components**
    - login.html
        - Allows users to login
        - Redirects to register.html if user has no pre-existing account
    - logout.html
        - Logs users out of their account
        - Redirects to login.html
    - register.html
        - Allows users to create a new username and password
        - Username and password get stored in the database
        - Redirects to homepage.html after signing up
    - homepage.html
        - Displays the different games and personal bests
        - (Goal) Displays leaderboards across user
        - Can redirect to the different games

- blackjack.html
  - Used to play Blackjack
  - Uses deckofcards API
- dice.html
  - Used to play guess the number
  - Uses rpgdiceroller API
- coin.html
  - User is able to play heads or tails
  - Uses coinflip API
- blackjack.py
  - Functions necessary to play blackjack
- coin.py
  - Functions necessary to play coin flip
- dice.py
  - Functions necessary to play dice
- db.py
  - Functions necessary to manipulate database tables

<u>Site Map:</u>



- **API’s**
    - deckofcards API
        - Allows users to shuffle, draw, and create new decks freely
        - Will be used to facilitate games of poker or blackjack
        - No API key needed and no quotas
    - coinflip API
        - Returns a random result of heads or tails
        - Will be used to facilitate a probability game
        - Need to make a rapid account for an API key and 1000 requests per hour
    - rpgdiceroller API
        - Allows multiple die to be rolled simultaneously
        - The die can have a number of different faces (e.g. d4, d6, d20)
        - Will be used to facilitate a probability game
        - No API key needed and no quotas

- **FEF**
    - Tailwind
        - Very easy to add lots of customization to any element used, such as tweaking colors, position, padding,etc.
        - We want to add a theme to our website,related to ‘games and gambling’, such as using bright colors so such customization will help
        - Has built-in support for state variants (eg: hover, focus, active, etc) allowing for easy and clear game flow
        - Flexible for third-party integration since no specific structure or dependency is imposed. Easy to pair with backend system that consumes APIs