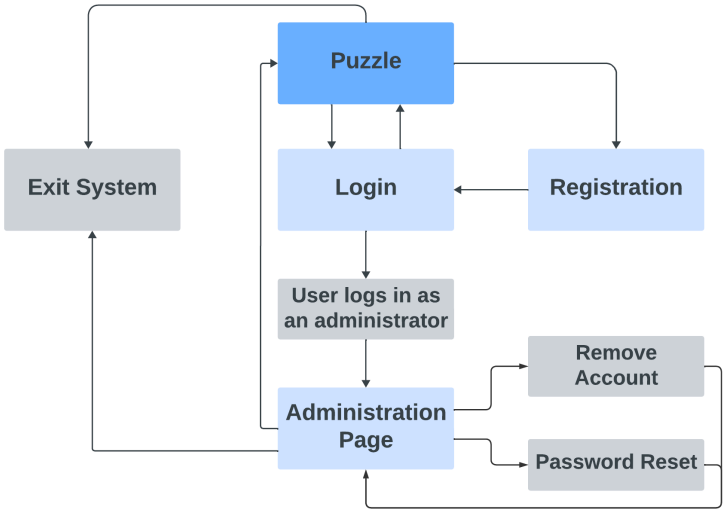
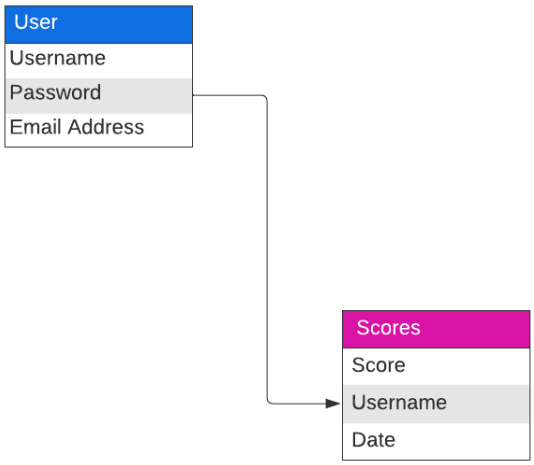
Project Description

- **Give a detailed description of the end product:**
    - The end product will be a Flask application that is deployed using Heroku. The Flask application will feature a puzzle game based on the Knapsack Problem. The user will be given a list of randomly generated items with associated weights and values, and be tasked with filling a backpack with the combination of items that yields the highest value without exceeding the maximum weight capacity. Based on the user's speed and overall correctness, a score will be given to them, which will be saved if they are logged in. A solution will be provided to the user if prompted, but they will not receive any points.
    - **Sitemap Flowchart**



    - **ERD**



- **The methods used in the puzzle solver:**
    - The 0/1 Knapsack Problem can be solved with a naive recursive method, where every subset of items is calculated, and the subset with the greatest value is the solution. This implementation has a time complexity of O(2^N).
    - The 0/1 Knapsack Problem can also be solved with dynamic programming, where the solution saves intermediate results in computing the solution. The time complexity of this is O(N * W).
        - https://www.geeksforgeeks.org/0-1-knapsack-problem-dp-10/
- **Describe the market space the application is related to and its selling points:**
    - This application can educate the user on the knapsack problem while providing a fun and interactive experience. It's simplicity can appeal to a wide audience of children and casual users, while more experienced users may test their speed and shoot for higher scores.

Functional Specifications

- **A complete list of product features, with a description for each feature:**
    - **Login and Registration:**
        - Users will be able to create an account with an email address, username, and password, which will be used to save their scores.
        - Users will also be able to play the puzzle game without logging in, but their progress will not be saved
        - Administrative users will:
            - Have all the functionality like the regular users.
            - Have additional privilege likes user account removals or password-reset.
    - **Puzzle Game and Solution:**
        - Users will be able to play the game and receive a score based on their speed and correctness.
            - A suboptimal solution can still receive points
            - The optimal solution will be provided after the users submits their solution
        - The user may ask for hints, which will detract from their score
        - The user may ask for the solution to the current puzzle, in which case they will not receive a score
        - Different puzzles can be randomly generated to give the user a near infinite amount of games

Deployment

- This application will be deployed using a combination of Heroku and Flask. Flask will be used in the development environment for the game logic, user interface, login and registration, and database. Heroku will be used in the staging and production environment of the deployment workflow, in order to be published as a web application. Heroku will also allow administrators to configure the environment in a secure way.
    - https://realpython.com/flask-by-example-part-1-project-setup/

Milestones

- **List of features that will be accomplished in the following (major) milestones:**
    - M1 (2/6 - 2/20): Create GitHub repository and setup Flask App
    - M2 (2/20 - 3/5): Begin rudimentary implementation of game in Flask
    - M3 (3/5 - 3/19): Create more detailed sprites in Aseprite and other GUI
    - M4 (3/19 - 4/2): Code the dynamic programming solution
    - M5 (4/2 - 4/16): Add Login/Signup to Flask App
    - M6 (4/16 - Finals): Final app cleanup and testing