Main Menu User Flow Diagram

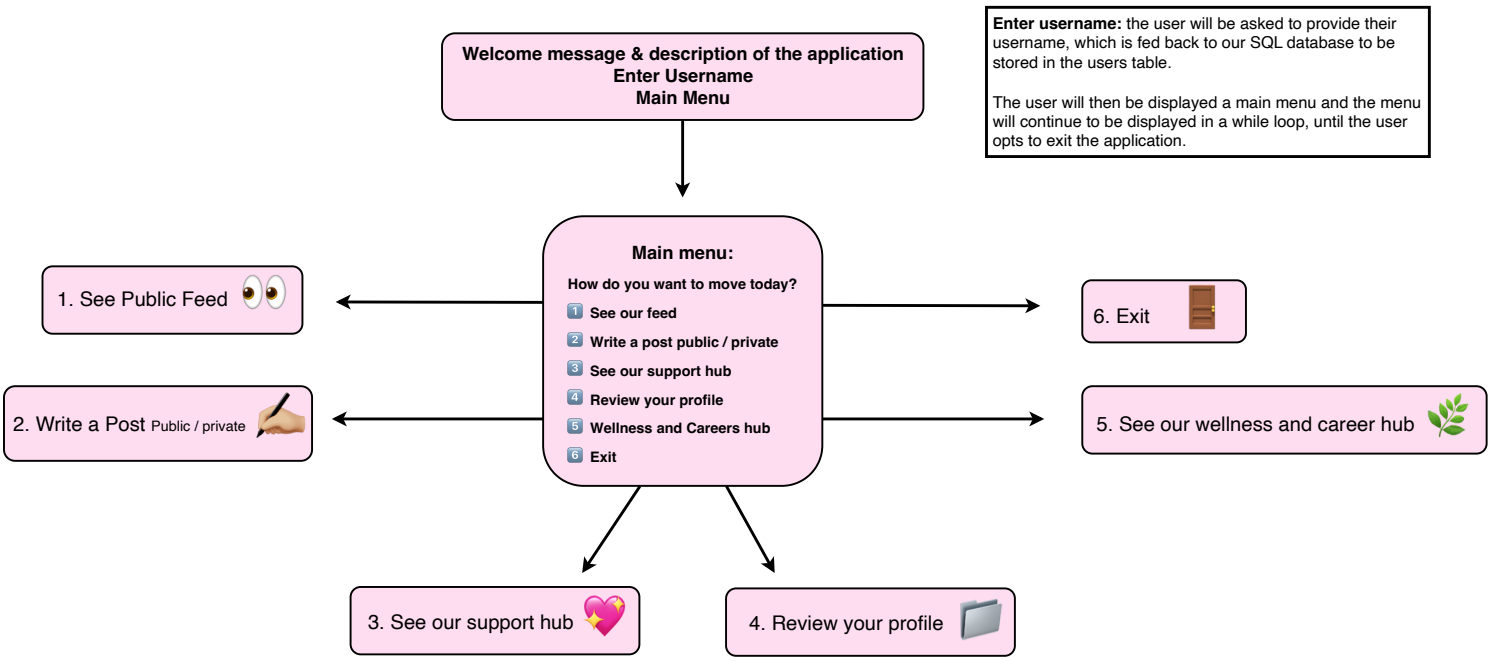

Figure 1. User Interaction Flow and Main Menu Navigation

This diagram illustrates the user journey as they interact with the application. After entering a username, the user is presented with a main menu that branches into six possible actions: viewing the public feed, writing a post, accessing the support hub, reviewing their profile, visiting the wellness and careers hub, or exiting the application.

Main Menu Options: Detailed Explanation

Option 1: See Public Feed

Posts that users have chosen to make public will display in the terminal and via a Flask route “/feed” in a json format. The posts are stored in and extracted from our SQL database in a table called journal entries using mysql connector.

The post information is converted into an iterable using a printer helper function in the clientside file. The information displayed is the:

1. post id
2. username of the poster
3. title of the post
4. post itself
5. status of the post (i.e. public or private)
6. likes
7. userlikes (i.e. users that have liked the post)
8. optional hashtags
9. affirmation

After the public feed is displayed to the user, they will be given the option to like a post. If they opt to like a post that has not been previously liked, then the “like count” will be incremented and the username of the user liking the post will also be stored as a “userlike”. Both the like and userlike are stored in SQL database.

Option 2: Create a post

The second option the user will be presented with is the choice to write a post, which will be stored in SQL and be displayed on the Flask routes “/feed” and “/username” if they choose to make the post public or “/username” only if they choose to make the post private.

Hashtag recommendation: after the user has written their post, a natural language python module scans the post, retrieves keywords from the post that are fed back to the user as recommended hashtags that they can choose to input as an add-on to the post. The natural language processing takes place in a class.

Keyword detection: if the user’s post contains certain keywords of concern, like “depression”, they will be asked if they want to see our support hub, which contains a database of resources stored in a python file. If they opt to go to the support hub, the resources they choose to see will be displayed in the terminal. The entire support hub can also be accessed via the Flask route “/support_hub”. If the user chooses to be rerouted to the support hub, they will not be provided with an affirmation. If they choose not to be rerouted to the support hub, they will be provided an affirmation.

Sentiment analysis: next, the post is analysed by the Vadersentiment python module to assess the sentiment of the post to produce a "positive", "neutral" or "negative" label.

Personalised affirmations: this label corresponds to curated lists of categories. RAPIDAPI, the external endpoint we have chosen to retrieve affirmations from, divides their affirmations into categories such as “love”, “blessings” etc. So, for example, if a user's sentiment is "positive", they may receive an affirmation in the "happiness" category. Affirmations are personalised to mood, not content. The sentiment analysis and affirmation generation takes place in a class, using class inheritance.

All the post information, along with hashtags and affirmations, is stored in the SQL database in a journal entries table.

Option 3: Access Our Support Hub:

The user will be asked if they want to see our support hub, which is a database of resources stored in a python file. The support hub can also be accessed via a Flask route /support_hub.

Option 4: Retrieve the user's post:

This is something of a personal profile to the user. Posts that the user has chosen to make public and private will display via a Flask route /username in a json format. The post information will also display in the terminal using a printer helper function in the clientside file and mirrors the format of the posts displayed in the public feed (option 1). The posts are stored and extracted from our SQL database in a table called journal entries using mysql connector.

Option 5: Wellness & Career hub:

The user will be asked if they want to see our wellness and careers hub, which is a database of resources stored in a python file. The wellness and career hub can also be accessed via a Flask route "/wellness_hub".

Option 6: Exit