Haylie Carter – [htc8@pitt.edu](mailto:htc8@pitt.edu)

Joshua Sitler – [JJS271@pitt.edu](mailto:JJS271@pitt.edu)

INFSCI 0201 SEC1200 – Prof. Zheng

12/9/24

Contribution Document

**Haylie:**

1. Database (*models.py)*
   1. Created dependency and setup for SQLite
   2. Created *Account* model
      1. username, password, name;
      2. added methods for setting and checking passwords
   3. Created *Event* model
      1. Defines event details – event name, type, time, location, and date;
      2. Storing and retrieving event data
   4. Created *EventAttendance* model
      1. Keeps track of which users are attending which events
      2. Events can be stored by date for easier retrieval and display
      3. Acts as a link between users (Account) and events (Event); It records which users attend which events using two fields: account\_id (user) and event\_id (event). These fields together form a unique key to prevent duplicates. The setup allows a single user to attend multiple events and multiple users to attend the same event efficiently.
2. User Authentication (*views.py*, *models.py*, *login.html*)
   1. Integrated Flask-Login for user login and logout
   2. Created login and signup routes
      1. Login form and a signup form for creating new accounts;
      2. Added checks to prevent duplicate users by checking for the username in the database
   3. Password handling
      1. Used Flask-Bcrypt library to ensure that passwords are securely hashed and stored;
      2. Used Its-dangerous library to generate and check password hashes to prevent plain text password storage and improve security.
3. User Profile and Settings (*views.py*, *settings.html*)
   1. Organized profile information and account settings into separate sections
      1. Profile page – created a route to display the current logged-in user’s profile
      2. Settings page – users can update their personal details (name, description, hobbies, age) and change their password
4. Event Management (Backend) - (*views.py*, *models.py*)
   1. Created a backend routes for Event Creation, allowing event organizers to create new events and store them in the database (Events creation and List – Joshua)
   2. Added route to create a page dynamically based on event id’s
   3. Event organizers can edit or delete events they’ve created
5. Search Functionality (*views.py*, *Base.html)*
   1. Event search – implemented a search route where users can search for events by name
   2. Included a no-results message, providing more clear feedback to the user
6. Login Page Layout *(login.html)*
   1. Designed and implemented the layout for the login page using Bootstrap
   2. The page includes a form for logging in, with fields for the username and password.
   3. Incorporated flash messages to display feedback on login attempts (incorrect username/password or successful login).
   4. Added a checkbox to indicate if the user is an event organizer.
   5. Included two buttons for login and signup, with separate values to distinguish between them.
7. Navigation bar *(base.html)*
   1. Responsive navigation bar using Bootstrap – This includes links for the home, profile, settings, and logout pages, along with a search bar
   2. Prototype: Used a search input field and search button – when clicked, it fetches event search results dynamically;
      * + Included the feature of displaying suggestions when users type in the search box (after Events are being created).
      1. Search suggestions – Developed a dynamic suggestion box that appears when users input at least 3 characters into the search box
         * This box shows relevant event suggestions fetched from the server and allows users to click on them to be redirected to specific event pages.
      2. Feedback messages – displaying "No results found" when no events match the query.
      3. Enabled the search bar to appear and disappear on mobile screens when the search button is clicked.

List of Main Files Modified:

login.html

settings.html

\_\_init\_\_.py

models.py

views.py

base.html

**Joshua:**

**Event Creation – the Display**

* Designed and implemented the system for creating and displaying events.
* Integrated functionality for creating events through a web form submission.
* Configured event storage using a JSON file (events.json).

**Home Page Functionality**

* Designed the home page to adapt based on the logged-in user’s role:
  + **Event Organizer**: Displays events they are organizing and provides a button to create new events.
  + **General User**: Displays events they are attending.
  + **Featured Events**: Shows upcoming events visible to all users.
* Implemented back-end logic for the homepage route @main\_bp.route('/home').

**Event Listings and Search**

* Developed a page that lists all available events from events.json.
* Added search functionality allowing users to filter events by name or tags.
* Implemented search routes:
  + @main\_bp.route('/search') – Displays matching events.
  + @main\_bp.route('/event/<event\_id>') – Displays full event details.

**Event Details**

* Created a detailed event display page that shows:
  + Event name, type, date, time, location, description, and organizer details.
* Designed the event detail page using Bootstrap and Jinja2 templates.

**User Profile Management**

* Designed and implemented the user profile and settings system.
* Added profile and settings pages for:
  + Viewing personal information such as name, description, hobbies, and age.
  + Editing personal details and updating account settings.
* Back-end routes implemented:
  + @main\_bp.route('/profile') – Displays the user’s profile.
  + @main\_bp.route('/update\_settings') – Handles profile updates.

List of Files Responsible for:

* **Front-End Templates:**
* events\_details.html – Displays event-specific details.
* events.html – Lists all available events.
* home.html – Shows events tailored to the logged-in user’s role.
* profile.html – Displays personal user details and events associated with the user.
* create\_event.html – Contains the event creation form.
* settings.html – Displays user settings and account management options.
* **Back-End Code:**
* views.py – Implemented all core routes, including event creation, home page management, search, and user profiles.