

Project00: Half-Quick

TNPG: coolbeans

Roster: Evan Chan, Amanda Tan, Danny Mok, Michelle Zhu

TARGET SHIP DATE: 2024-11-15

# DESIGN DOCUMENT (VERSION 1.0)

---

## I. Description

This project implements a web log hosting site, with the following features:

- Users will have to register to use the site.
- A logged-in user will be able to
  - Create a new blog
  - Update their blog by adding a new entry
  - View and edit their own past entries
  - View the blogs of other users

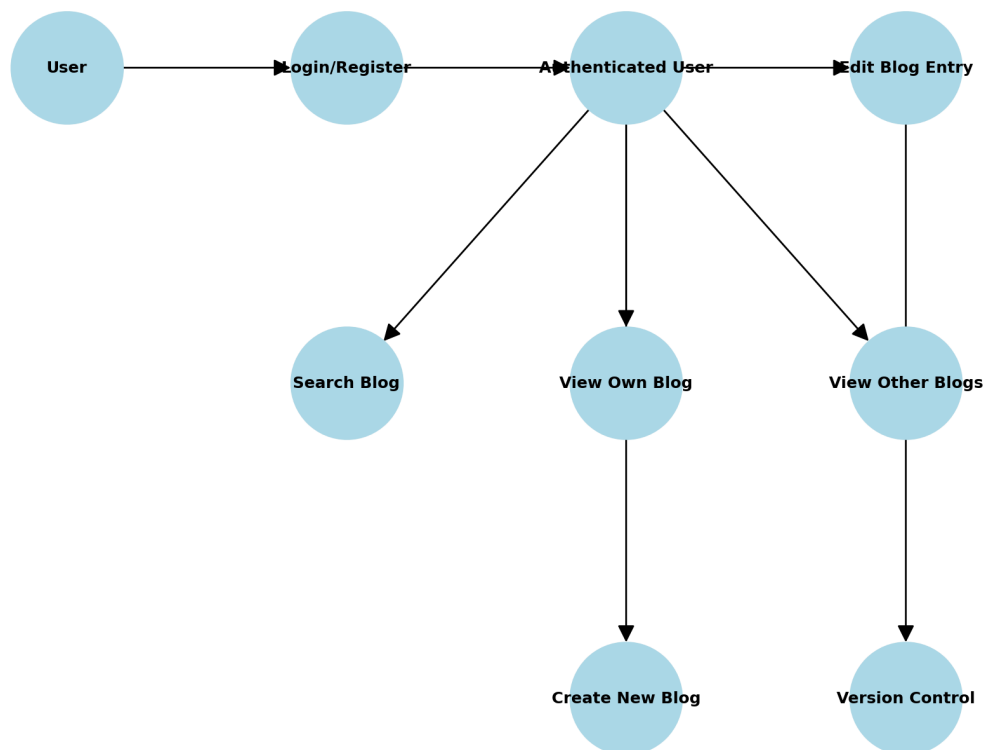
## A. Functionalities

- I. Management of Individual Blog Pages:
  - a. Ability to create blog pages and make further changes after publishing
  - b. Rich-text editor to create/format content: the form to submit/ make new text edits will have UI to help style the text
  - c. Version control: Edit history, store previous versions, rollback functionality (Start with only the version right before the current; Can expand to include all if time allows)
2. Search Functionality: Ability of users to search the website for content based on keywords/filters
  - a. Keywords/Filters are found by scanning each page of its contents
3. Home Page: Dynamically updating page with all the existing blog pages
  - a. Recent blogs show up on top
  - b. Uses links to bring to each page

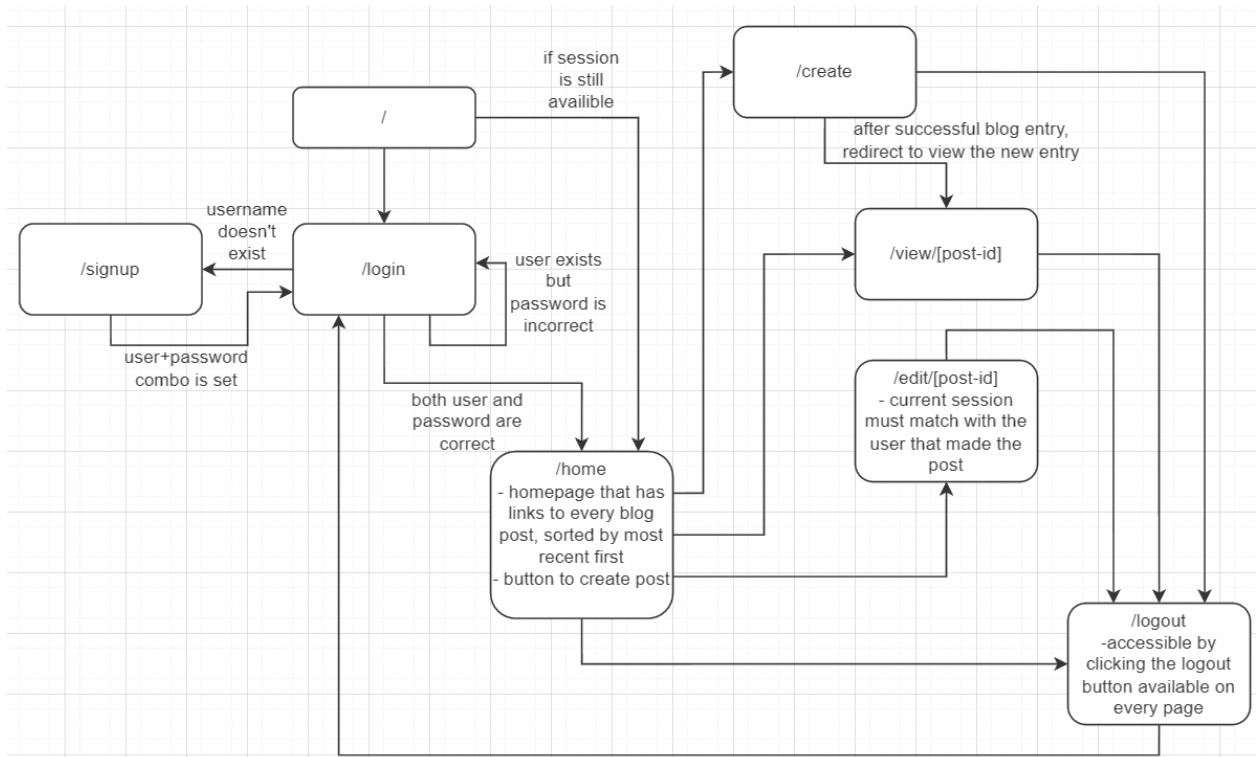
## B. Program Components

- I. Flask/Python:
  - a. Serves the websites, handles HTTP requests, manages user sessions and authentication (login), controls logic surrounding permissions
  - b. Interacts with various tables within the database file to fetch and store user and blog data

- c. Coordinates Flask routes (between HTML pages) and middleware functions (ex. Module for storing data into tables, creating user sessions, etc) to control application flow
- 2. SQLite3 (Database):
  - a. Stores user information (usernames, passwords, permissions, etc)
  - b. Stores data about each blog page in a unique table → Stores content such as titles, content, categories, author, etc
  - c. Stores version history in a separate table, but linked to tables for unique blog pages through a unique identifier such as page ID
  - d. Data used by middleware functions and HTML templates
- 3. HTML: Provides user interface with links to access different pages (ex. blog pages, blog creation pages, etc)
  - a. Renders content passed by Flask routes/database
  - b. Displays form for user interactions (ex. Creating blog pages, logging into an account, etc) and works with Python to create the various actions
- 4. Jinja: Create templates for blog pages with room for customization
  - a. Used by Flask routes to have blog pages with default formatting (keeps things consistent and makes it easy to create pages for new content)
  - b. More on jinja templating:  
<https://atufashireen.medium.com/creating-templates-with-jinja-in-python-3ff3b87d6740>
- 5. CSS: Create a rich-text editor and make the site look nice across the board
  - a. Works with HTML documents



## C. Site Map



## D. Database Organization

- I. User Table
  - a. User Id/Username (PK)
  - b. Password
  - c. Role (Site as a whole: admin/editor)
2. Blog Page Table
  - a. Page ID (PK)
  - b. Title
  - c. Content
  - d. Username of Creator (FK)
3. Version History Table (Only stores version before current)
  - a. Edit ID (PK)
  - b. Page ID (FK)
  - c. Content Before Edit
  - d. Content After Edit
  - e. Edited By Username (FK)

Note: FK stands for foreign key (to link tables), PK for primary key (each row value must be unique)

## E. Task Breakdown

- I. Evan Chan: Full-stack/Project Lead

- a. Work on version control system by creating any related Python modules and SQLite3 tables
  - b. Implement user authentication system and sessions
  - c. Manage overall system/file structure: Work with backend and frontend to make sure everything works smoothly
  - d. Pick up any work that isn't explicitly delegated/falling behind schedule
  - e. Assist with roadblocks in any area
2. Danny Mok: Backend/Database
  - a. Create SQLite3 database schema
  - b. Design database interaction modules (general operations, not specific; ex. Inserting data, Creating tables, etc)
  - c. Work with Amanda to create a module for search functionality/filtering (of the blog pages)
  - d. Work with Michelle to ensure database logic flows/works with front-end
3. Amanda Tan: Backend/Middleware
  - a. Create Python modules to handle logic relating to user actions (creating/editing/deleting blog pages)
  - b. Create middleware for user permissions and roles
  - c. Work with Danny to create a module for search functionality/filtering (of the blog pages)
  - d. Work with Michelle to ensure all related code flows with the front-end
4. Michelle Zhu: Frontend
  - a. Create HTML pages with Jinja templating for dynamic blog content
  - b. Design site style
  - c. Use CSS to make things look nice
  - d. Collaborate with the backend to ensure everything works with the frontend