DESIGN DOCUMENT (VERSION 1.0)

I. Introduction

Project Name: PyWiki
TNPG: Boaas
Roster: Aidan Wong, Linda Zheng, Ryan Zhou, Michelle Zhu
TARGET SHIP DATE: 2024-II-04

II. Description

This project implements a collaborative web application called a “wiki,” where users can create, edit, and manage content regarding various topics.

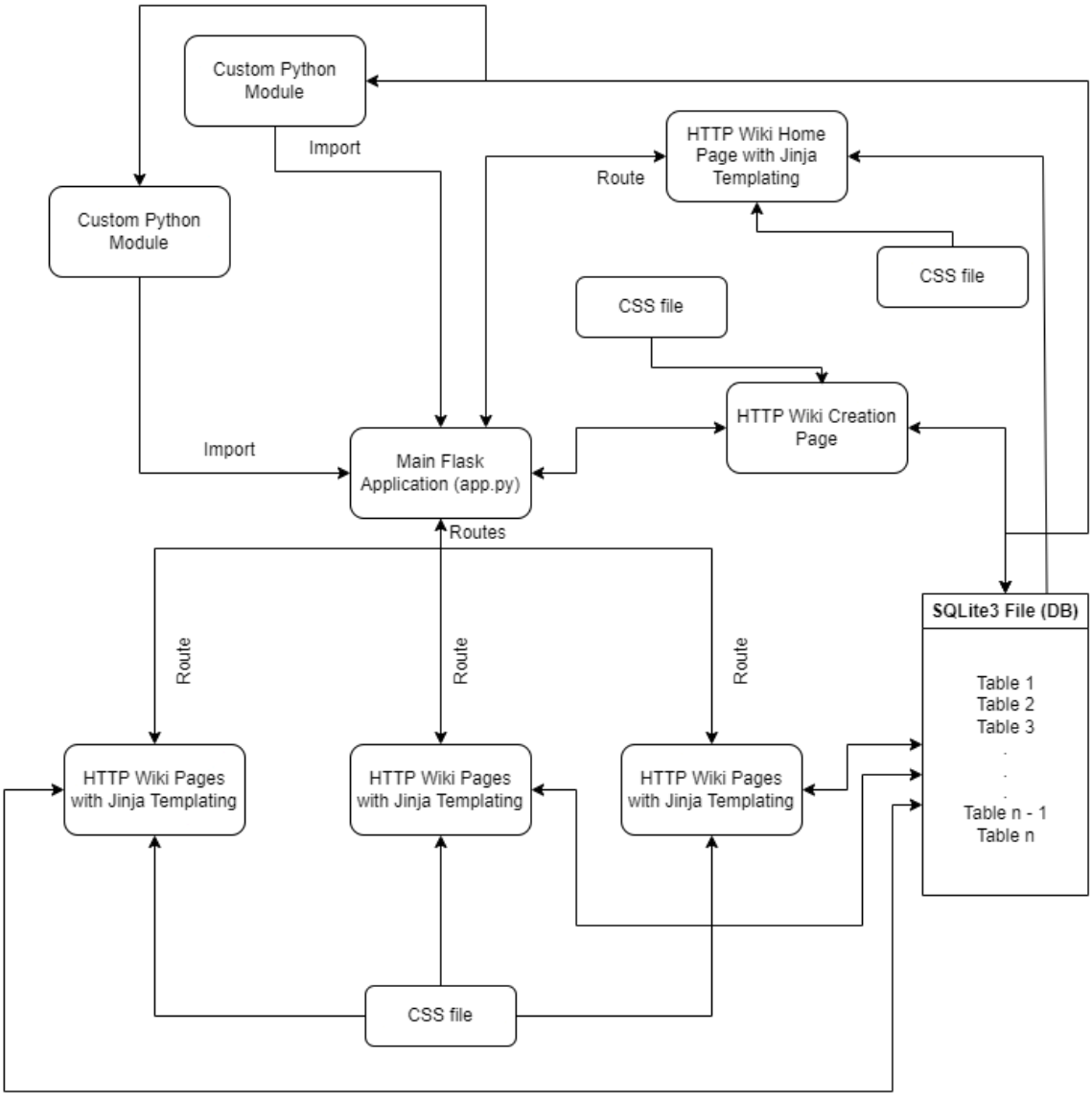
A. Functionalities

1. User accounts: Create customized experiences for different individuals
2. Management of Individual Wiki Pages:
    a. Appropriate permissions for various users - Specification of user roles (admin, editor, viewer, etc) within each wiki page and site as a whole
    b. Ability to create wiki pages and make further changes after publishing
    c. Rich-text editor to create/format content
    d. 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)
3. Search Functionality: Ability of users to search the website for content based on keywords/filters
4. Home Page: Dynamically updating page with all the existing wiki pages
    a. Uses links to bring to each page

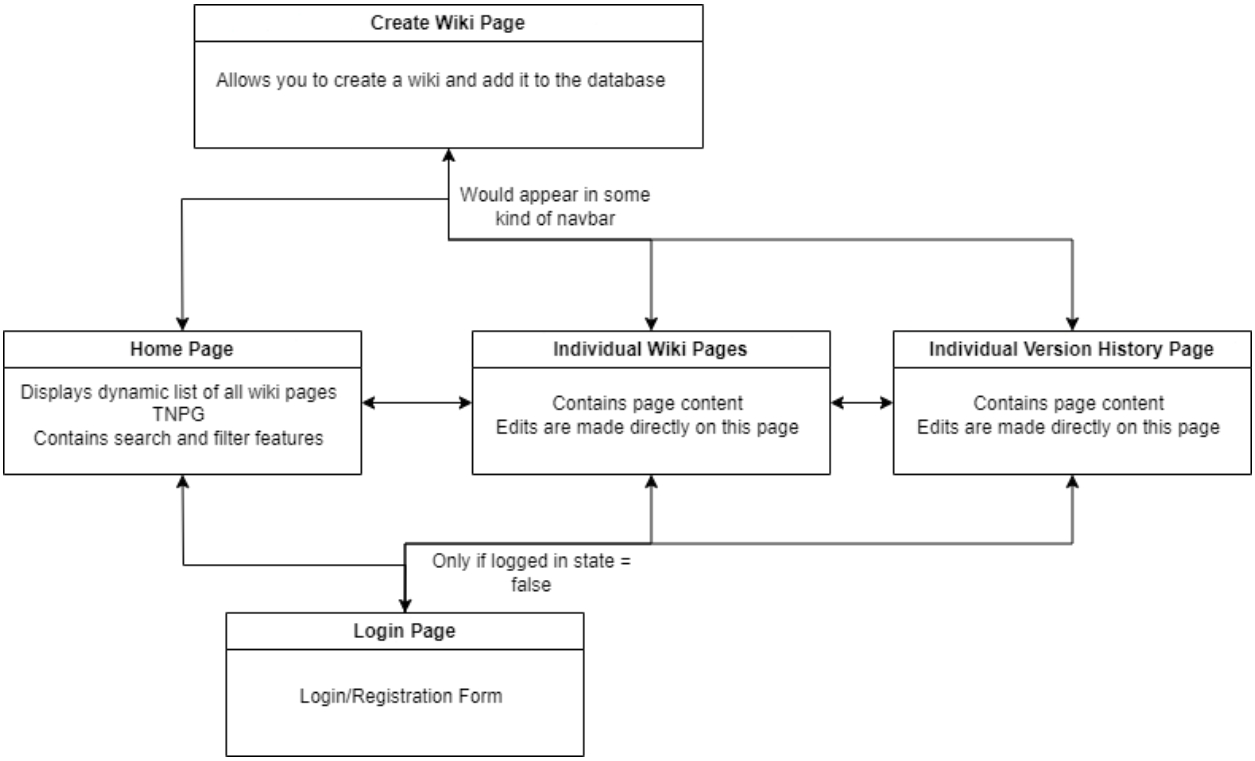
B. Program Components

1. 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 wiki 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 wiki 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 wiki 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. Wiki pages, wiki creation pages, etc)
    a. Renders content passed by Flask routes/database
    b. Displays form for user interactions (ex. Creating wiki pages, logging into an account, etc) and works with Python to create the various actions
4. Jinja: Create templates for wiki pages with room for customization
    a. Used by Flask routes to have wiki pages with similar formatting (keeps things consistent and makes it easy to create pages for new content)
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

1. User Table
    a. User Id/Username (PK)
    b. Password
    c. Role (Site as a whole: editor/viewer)
2. Wiki 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. Timestamp
    f. Edited By Username (FK)
4. Collaborators Table
    a. Page ID (FK)

    b. User ID/Username (FK)
    c. Permission Level (read/edit)

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

E. Task Breakdown

1. Aidan Wong: 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. Ryan Zhou: 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 Linda to create a module for search functionality/filtering (of the wiki pages)
    d. Work with Michelle to ensure database logic flows/works with front-end
3. Linda Zheng: Backend/Middleware
    a. Create Python modules to handle logic relating to user actions (creating/editing/deleting wiki pages)
    b. Create middleware for user permissions and roles
    c. Work with Ryan to create a module for search functionality/filtering (of the wiki 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 wiki 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