BabyX: Vedant Kothari, Tahmim Hassan, Benjamin Rudinski, Yinwei Zhang
SoftDev
P05: Le Fin
2025-03-25
Time Spent:
TARGET SHIP DATE: 5/30/25

DESIGN DOCUMENT (PRELIMINARY)

I. Description

At a school like Stuyvesant, it's fair to say academic rigor is at the forefront of our expectations. Exams and quizzes take up a big proportion of one's time. One thing we've all learned is that the key to studying is studying with quantity rather than quality. Oftentimes, we are forced to cram studying in a short time due to accumulation of other things. Our goal is to fix this issue. We want to provide a centralized location for students to find resources. They can do anything study-related on this site, from finding study guides for the most challenging courses (physics, chemistry, calc, etc.), a free-to-use whiteboard for students to write work out, and practice quizzes with data visualization features.

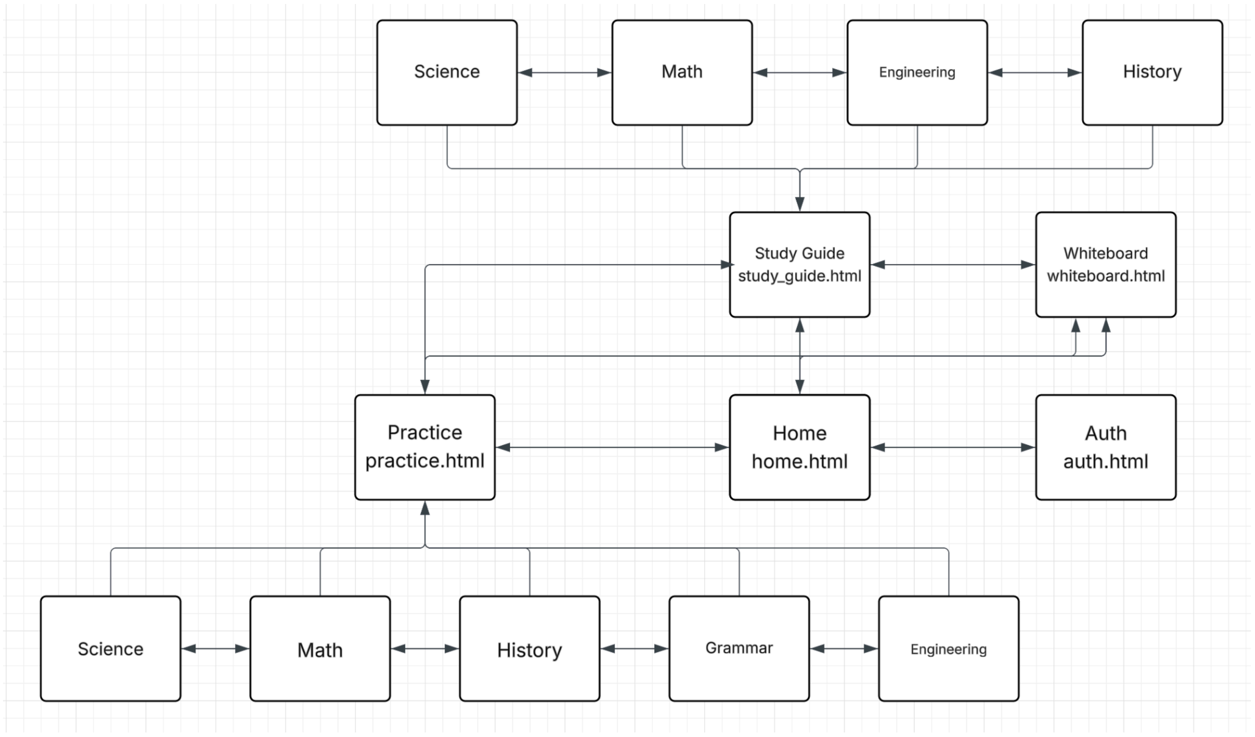
A. Program Components

1. Flask/Middleware
    a. Handles URL routes and serves web app while managing session flow.
    b. Session Management: Keeps track of logged-in states and user data inputs for obesity predictions
2. SQLite Database
    a. User information: stores username, password, and history.
    b. Study guide dataset: stores study guides for different topics or maybe from other users as well.
3. Data Processing
    a. Python scripts to clean and format study guides and quiz datasets
    b. Ensure uniform structure for easy querying and visualization
4. Obesity Prediction Module (Deprecated)
    a. Removed; this component was relevant only under the previous MongoDB/obesity-related scope.
5. Apex
    a. Bar charts, line charts, pie charts to reflect trends in scores and practice questions

    b. User-driven filtering to update charts based on specific parameters
6. Front End
    a. HTML templates to display data from Flask routes
    b. Foundation provides grids and styling for multi-screen UI
    c. Integrates with D3 such that data visualization is well structured within the UI
    d. FEF: Foundation

Program Component Connections

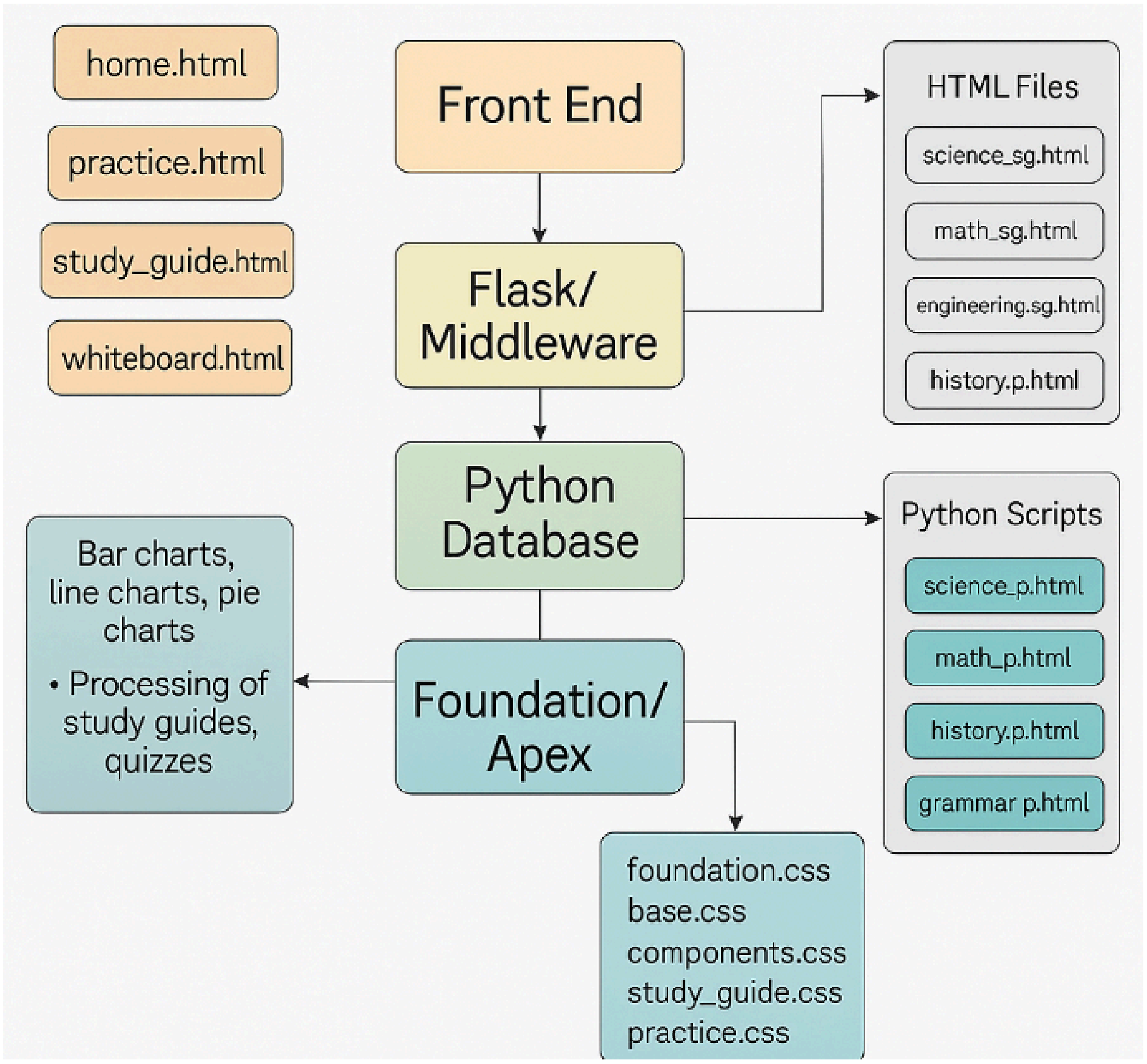
- Flask ↔ SQLite
    - User login, registration, data queries
    - Study guide retrieval and quiz result updates
- Flask ↔ Templates
    - Serves dynamic content such as guides, quizzes, graphs
    - Handles user inputs and navigation
- Templates ↔ Apex
    - Embeds charts into user dashboard and quiz results
    - Receives data from Flask endpoints via templating or JS fetch



B. Site Map + Descriptions

1. Home (home.html)
    a. Home page for website

2. Practice (practice.html)
    a. Page to find practice questions for each of the following subjects: Science, Math, History, Grammar, & Engineering
3. Study Guide (study_guide.html)
    a. Page to find study guides for classes in each of the following subject categories: Science, Math, Engineering, & History
4. Whiteboard (whiteboard.html)
    a. Page where users can draw notes, do scrap work, etc. on a whiteboard



C. Component Map

1. Front End

    a. Handles all user-facing pages and UI components. Built with HTML templates styled using Foundation/Apex for responsiveness and visual polish.
    b. Communicates with Flask to display dynamic data like study guides, quizzes, and visualizations.
    c. Connected to: Flask, Apex
2. Flask / Middleware
    a. Acts as the backbone of the application, managing routing, sessions, and logic for requests and responses.
    b. It serves all HTML pages, handles user login/registration, and processes form inputs.
    c. Connected to: SQLite , Front End
3. SQLite Database
    a. Stores user credentials, quiz scores, and study guide metadata.
    b. Lightweight and embedded, this component is used for fast reads/writes by Flask when users log in or interact with content.
    c. Connected to: Flask
4. Python Scripts
    a. Used to preprocess and normalize uploaded study guides and quiz data. These scripts ensure the datasets are clean and structured for efficient use in both the frontend and graphs.
    b. Connected to: SQLite, Flask
5. Foundation / Apex
    a. Dual-purpose UI and visualization toolkit. Foundation provides CSS grid layout, buttons, and forms for UI consistency.
    b. Apex handles rendering charts (bar, line, pie) that reflect quiz data, accuracy breakdowns, and subject trends.
    c. Connected to: Front End, Flask

D. Frontend Framework: Foundation

1. Why Foundation?
    a. Easy-to-use and easy-to-follow tutorials
    b. A vast amount of CSS and JS components that will make things look and feel good
2. How Foundation?
    a. Grid System: Structure layout of pages like the dashboard and graph visualizers (Website will be able to adapt to size changes)
    b. Pre-designed Components: Buttons, Forms, etc
    c. JS Plugins: Modal Windows for confirmation messages

E. APIS

    a. None

F. Data Visualization Library: Apex

1. Why Apex?
    a. Flexibility - Precise control over various parts of the graph, enabling highly customizable visualizations
    b. Data Binding - Binds data to DOM elements, allowing real-time updates
    c. Appearance - Looks extremely clean and has many built-in animations/interactable elements
    d. Declarative Syntax - Simplifies mapping data to graphical elements
2. How Apex?
    a. Bar charts - Compare test scores in certain subjects over time
    b. Line charts - Track how test scores have changed over time
    c. Pie charts - Show the distribution of accuracy in different types of question per subject (ex. Integrals, series, derivatives, etc.)

G. Datasets

1. SAT Scores by state (1999) https://www.kaggle.com/datasets/billbasener/sat-score-data-by-state?
    a. Display SAT score data by state in 1999 to show data from then.

User	**Example**
Username	”Vedant”
Password	********
Study Guides Viewed	2
Quizzes Completed	6
Whiteboard Saves	3
Login Timestamps (in Unix Epoch)	Table(145624542, …)

H. Task Breakdown

1. Frontend Lead - Vedant Kothari
    a. Frontend - HTML templates, CSS, and FEF implementation
    b. Creating study guides for courses
2. Graph Lead - Yinwei (Will) Zhang

    a. Graph Visualization → Work with Database and Middleware leaders for implementation
    b. User authentication functionality
3. Middleware Lead + PM - Tahmim Hassan
    a. Setting up the Flask environment and creating functionality for the website
    b. Working with Database and Graph leaders to develop specialized functions
    c. Hosting website on droplet
    d. Facilitating clear communication
4. Database Lead - Benjamin Rudinski
    a. Create a database using SQL that holds user information
    b. Working with Middleware and Graph leaders to create necessary general functions to interact with the database