Scheduling and Storage Program

Architecture/Design Document

Table of Contents

1. Introduction

This document describes the architecture and design for the scheduling and storage application being developed for Gina Campbell. The program is designed to allow entry and storage of schedule preferences and needs for any and all instructors in order to more effectively and efficiently generate schedules for each semester.

The purpose of this document is to describe the architecture and design of the scheduling and storage application in a way that addresses the interests and concerns of all major stakeholders. For this application the major stakeholders are:

- Gina Campbell - Sole/main user, client
- Kendall Bingham - Instructor, Project Quality Inspector
- Developers
- Project Manager
- Maintenance Programmer

The following sections will detail the architectural structures used and design decisions made to provide a product that meets the desired functional and non-functional requirements.

2. Design Goals

HTML/CSS formatting is being utilized to design the web pages and format them. The design of the interface will be focused on ease-of-use for the user. The program will utilize javascript to call stored information from, and send user input to, the incorporated databases to keep track of the past semester schedules and also view the list of courses being provided for the current semester alongside their instructors. If provided substantial time the scope could include focused views for different users by utilizing their respective logins and could provide the option of creating individualized schedules. Furthermore, the idea is to keep the design appealing and simplistic but also while meeting the client's needs.

3. System Behavior

The user can begin accessing the system by navigating to the webpage URL using a web browser. Upon entry the user will come across a login page. This page will require a username and password in order to proceed. Once a valid username and password have been inputted the user will be directed to the Home Page. This page includes all the necessary directory links:

- Scheduling Requests
- Instructor Database

- Generate Schedule
- Sign Out

Scheduling Requests -

- The user will have the ability to view, edit, create, delete, and approve/finalize requests from all instructors.
- Request options include, but are not limited to: classroom, date, time, course, ongoing/one-time, and have a preference weight.

Instructor Database -

- Instructor preferences and information are stored here after a request is submitted.
- The database is used to store this information and then feed the information to the scheduling algorithm.

Generate Schedule-

- Function triggered by user command
- reads and applies database information into a working schedule
- sends schedule to front end to be viewed by user

Sign Out -

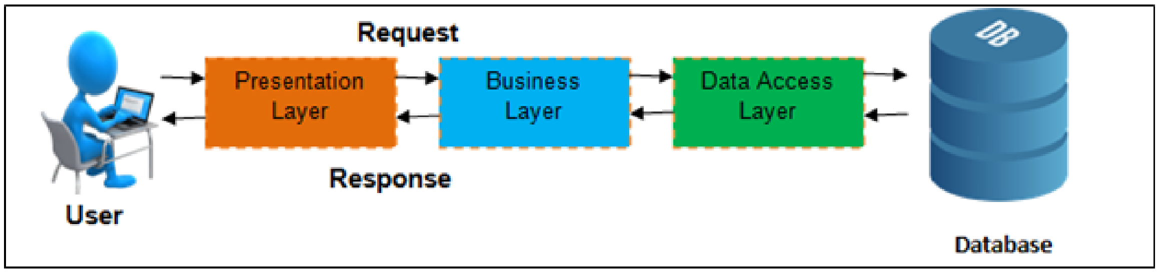
- User is redirected to login page and unable to navigate back
- Access approval is removed until correct username/pass is entered again

4. Logical View

This section is designed to show the architecture of the 3-tiered system this project has adapted. It will focus on the high-level modules and their relationship to one another, while the mid-level design shows the different code languages used and the way that each interacts with the other to perform the main functions of the program.

4.1 High-Level Design (Architecture)

The high-level view or architecture consists of <4> major components, 3 of which are hosted by Django (Presentation, Business, Data Access) and the other being hosted by MySQL:

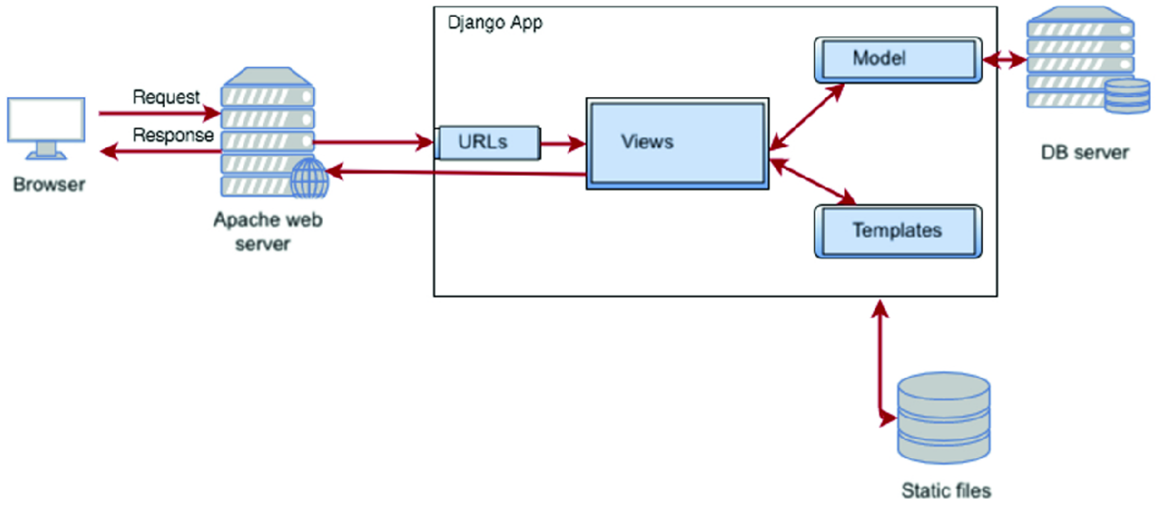


System Architecture

- The **Database** is a central repository for data on professors, classes, schedules, and rooms. Hosted by MySql.
- The **Data Access Layer** is the controller for querying data from the DB. Managed by Django in Python.
- The **Business Layer** is the main driver of the application. It presents information to the user and reacts to user inputs, as well as managing the scheduling algorithm. Currently locally hosted by Django, with hopes to migrate to an Apache server.
- The **Presentation Layer** is the part the user interacts with directly, written in HTML and CSS, and is stored statically in Django.

4.2 Mid-Level Design and Physical View

The mid-level design for the program can be broken down to each tier with the Django application being hosted -currently locally- on an Apache web server, the DB Server being MySQL, and the static files being the HTML/CSS for display to the user. The business logic receives requests from the user, then uses the Views controller to query data from the DB Server and format it to a template from the static files, to then send each response back to the user thru the web server.



4.3 Detailed Class Design

TBD

5. Process/Use Case View

- **Access Thread:** This thread is user-created and requires valid sign-in credentials. The thread is in charge of sending input information to the program database of valid credentials, verifying that both inputs exactly match available information in the database and returning a command to open the main program's home page

- **Application Thread:** This thread is created by the successful completion of the access thread and is not directly user-created. The application thread controls the navigation between pages, handling user input to return the display of whichever page is chosen

- **"View Database" Thread:** This thread is user-created with the action of clicking the link on the home page titled "Instructors"(subject to change). The thread redirects the user to a view of the table of instructors in the SQL database, enabling viewing access to individual instructor's information with another click from the user. This thread, along with the other functional pages of the program, also enables the user to return to the home page by clicking the link labeled "home."

- **Entry Form Thread:** This thread is user-created, again by clicking a link on the home page ("Enter Preferences", working title). This link redirects to a user-friendly entry form that includes input fields for the various categories of instructor information. Once input fields have been filled out, the user can trigger the next part of this thread by clicking "submit," which will create a new entry in the SQL database and store the information for viewing and use by the scheduling algorithm

- **Generate Schedule Thread:** This thread will be the most complicated as it incorporates the algorithm used to create these schedules. The user will generate this thread with the click of a link "Generate Schedule." This will trigger several automatic processes; the python function will pull instructor data from the SQL table to incorporate the scheduling restrictions into the semester calendar, repeating this process for each instructor's preferences as the algorithm works out a functional semester schedule. Once the schedule has been generated, the information will be sent back to the front end, displaying as a traditional calendar for the user to then print out and/or save to the local drive.