Duck Planner: An Improved Schedule Builder for University of Oregon Students

Annika Huston, Ethan Dinh, Nathan Gong
Web Feet Team
J. Flores (JJF) – 10-10-23 – v1.0

1 Table of Contents

2 SDS Revision History

This lists every modification to the document and the respective authors. Entries are chronologically ordered.

Date	Author	Description
11-20-2023	AH	Added initial document to repository
11-22-2023	AH, ED	Filled in Sections 3, 4.1 – 4.3
11-22-2023	AH, NG	Filled in Sections 5.0 – 5.3
11-30-2023	AH, NG	Filled in Section 6, 7
12-1-2023	ED, AH, NG	Revised the SDS to prep for submission

3 System Overview

For students at the University of Oregon, the process of selecting and scheduling classes each term can be complicated and time consuming. With the large amount of class options, times offered, and personal scheduling preferences, students need a tool that will simplify the process and cater to their needs. Our intention behind the system is to create a personalized scheduling aid, allowing students to input their desired classes, apply filters, and view schedule variations in a simplified manner.

Duck Planner consists of two main components: a login screen and a scheduling page. The login screen allows users to create a new account or facilitates users access to their personalized dashboard containing previously entered classes and saved schedules. The scheduling and filtering features fetch real-time class data where students can select or modify the classes they plan to take. Once a schedule is chosen, it is saved to their account for future reference. Our system integrates class data with personal preferences while providing an interface between the schedule selection and schedule visualization that saves students time and simplifies their academic planning process.

4 Software Architecture

The following section effectively details and conveys the key design decisions, which include how the system is broken down into components and the relationships between these components (Faulk, 2017).

4.1 Components and Their Functionality

Client Side

4.1.1. Create Account/ Login Page

Allows students to create a new account or log into an existing account. By creating and logging into an account, the user will be allowed to save their schedules and view them whenever they desire. To do so, the user will provide login information which is then sent to an AWS lambda function for processing on the server side.

4.1.2. Scheduling Dashboard

Class Selection
Interface lets users select classes they are interested in taking. Real-time data is fetched from the University of Oregon's DuckWeb to provide students with the latest information. After a student selects a schedule, an API post request is made to an AWS Lambda function in order for the server to compute and determine the possible schedules the user can take. The results are then sent back to the client side where the visuals are then rendered for the user.

Dynamic Sorting
Users can apply various filters to view tailored schedules. This is help aid in simplifying the scheduling process. The sorting process is computed on the client side to aid in the rendering speed of both the previews and the calendar view.

Schedule Visualization
Once classes are selected and filters are applied, students are able to view potential schedules on the same dashboard. These visuals are computed on the client side to aid in rendering speed. They also have the option to save schedules to their account.

4.1.3. Map Dashboard

In addition to viewing previews of their schedules, the user-selected schedule contains building codes which are then sent via a post request to an AWS API that returns the latitude and longitude of the building. That data is then supplied to MapBox's API which displays a visual marker on a dynamic map of the University of

Oregon to indicate where the students' classes are located.

Server Side

Database

- *Account Information*
  - 95 number – Proprietary UO ID
  - Password – User Defined
- *Class Information*
  - Time – 24hr Format
  - Professor – First and last name
  - Location – Building Codes
  - Requirements – Per University Codes
- *Saved Schedules*
  - Stores schedules that users saved for future use.

Web Scraper

Periodically scrapes class data from the university's website and updates the class information within the database. This ensures that the most current class data is available to users on the client side. The web scraper is to be hosted on an AWS lambda function which is linked with an interval timing function that allows for automatic periodic scraping. The scraper links directly to the database and asynchronously updated the data.

4.2 Module Interaction

In our system, every module has a distinct role, yet they all interact to provide an effective scheduling experience. This modularity allows for increased bug resistance and easier debugging. The modules include:

4.2.1. Authentication Module

This module handles the creation of new user account and verification of existing users. It directly interacts with the Database Module. When a user creates a new account, their information is stored in the database, and when a user logs in, their information fetches and verifies their 95 number and password. After they are verified, students are directed to the Dashboard Module. The verification is hosted on the server side which aids in providing security to the user. Furthermore, by implementing a hidden hash function on the server side, the user's data is hidden to both other users and the developers.

Dashboard Module

This is the central hub where users interact with class data and visualize schedules. It communicates with the Database Module to retrieve real-time data and have options to view schedules. It also must communicate with the Database Module when the student wasn't to save a chosen schedule.

4.2.2. Database Module

The module stores and manages all the system data. It serves as the primary data access layer, responding to queries from the Authentication Module and the Dashboard Module. The WebScraping Module updates the Database Module to keep the data up to date.

4.2.3. Scheduling Module

The scheduling module directly interacts with both the dashboard and database modules. In specific, when a user selects a class via the dashboard module, a post request is sent to the backend where the scheduling module requests data from the database to calculate and determine all possible schedules.

4.3 Rationale for Architecture

The primary objective is to provide students with a platform where they can seamlessly create their academic schedules. The architecture was designed with this objective in mind, with each module serving a specific purpose. The decision to join the log on and create profile on one module was due to the shared common goal of user verification. The unified approach allows us to have a smooth transition from creating an account and logging in.

The Dashboard Module is separate as it is a central hub for all user functionalities. It holds all the class selection, dynamic filtering, and schedule visualization functionalities. Keeping it distinct allows us to easily adjust for enhancements in the future. The Database Module and WebScraping Module are critical back-end components. The Database Module acts as a storage and retrieval system and the WebScraping Module ensure the Database is up to date. Keeping them separate allows us to easily integrate other data sources in the future without any major changes, for example, if we extended our services to other universities.

One of the vital interactions allowing our software to function is the interaction between the frontend and the backend. To do so, we implemented the following AWS services: Lambda, CloudFront, Route 53, S3 Buckets, and API gateway. These services allow for static hosting of our main webpage and interactions between our frontend and API calls to the backend.

This integration strategy allows for more secure queries. Especially when users are providing their ID number and creating passwords, we did not want to expose our hash functions to the user. Furthermore, AWS Lambda is charged by a per use basis. As such, we can scale the software as more users begin to utilize our services.

5 Software Modules

5.1 Authentication Module

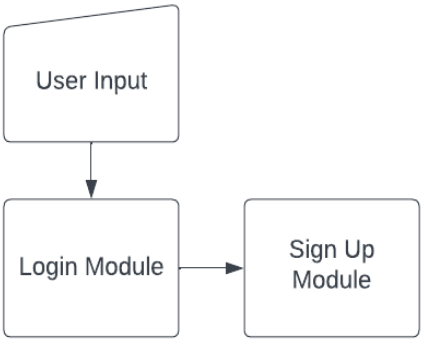
5.1.1. Primary Role and Function

This module allows users to create a new account or log in to their existing account. New users are prompted to enter their ID (95 number), full name, and a password. Returning users are prompted to enter their ID and their password. Correct credentials will allow students access to their account, while incorrect credentials will display a message explaining the system is unable to verify their credentials.

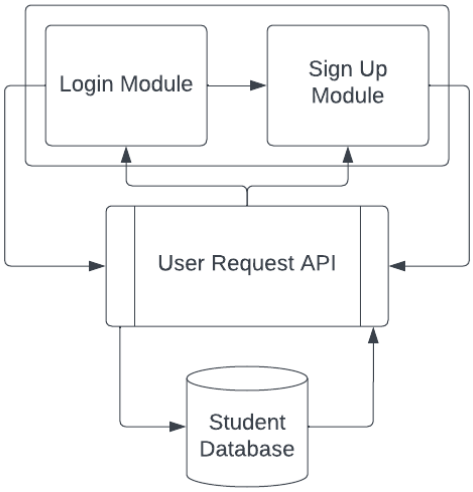
5.1.2. Interface

The webpage is engineered to be functional and user-friendly, irrespective of a user's login status. Without logging in, users can interact with the site, though their schedules won't be preserved. Conversely, account creation unlocks additional features, such as the ability to save and revisit schedules. Initial interaction is streamlined with a prominent login button at the top-right corner of the page, accompanied by an option to register for first-time users at the pop-up's base. Successful login or account creation seamlessly redirects users to the Dashboard Module, ensuring a fluid and uninterrupted user experience

5.1.3. Static Model



5.1.4. Dynamic Model



5.1.5. Design Rationale

The design of the Authentication Module is centered on security and user convenience. By structuring the frontend to issue POST requests to a serverless Lambda function, we ensure a robust separation between the client and the backend processing. This architecture choice prevents users from having direct exposure to sensitive hash functions and the underlying authentication mechanisms. The Lambda function acts as an intermediary, which verifies credentials securely while maintaining the integrity and confidentiality of the user's data. The design also focuses on user experience, allowing for flexibility in schedule management with added benefits for logged-in users.

5.1.6. Alternative Designs

One alternative design was to have separate modules for the login and account creation features. However with the shared goal and overlap in access to the Database Module, it seemed more efficient to combine them into one.

5.2 Dashboard Module

5.2.1. Primary Role and Function

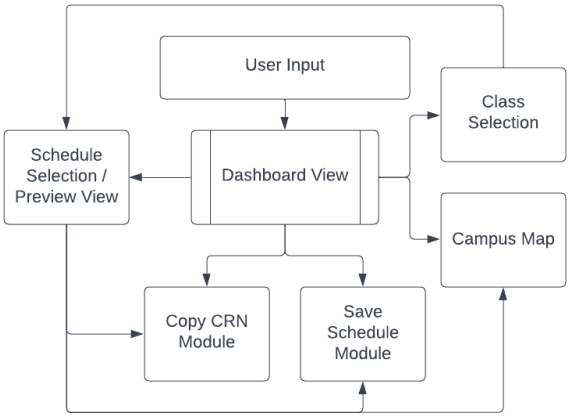
This module is the main module that users will interact with. New users and returning users that don’t have an existing schedule or selected classes with be provided and landscape to search for their preferred classes and select a schedule with these classes based on a personalized filtered sort. Returning users will be able to delete and change an existing schedule that they

previously selected. A map will also be present to locate where their selected classes are on campus.

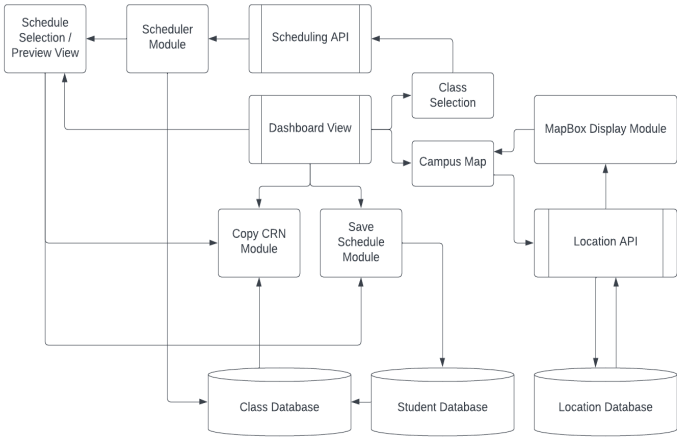
5.2.2. Interface

The user interface is compact and easy to understand, with three columns separated for the usage of class selection, schedule selection, and schedule image from left to right. The class selection row contains a search bar with fuzzy search which will provide the user with the most similar class results to their input. Clicking on a class that is returned from this search will add it to their schedule, and all possible combinations of the chosen classes will populate the schedule selection column. A filter selector at the top of the schedule selection column will sort the possible schedules by the chosen filter, and clicking on any given schedule preview will create a larger image in the largest and rightmost schedule. A button at the top of the page will lead the user to a campus map that will show the location of all selected classes.

5.2.3. Static Model



5.2.4. Dynamic Model



5.2.5. Design Rationale

We wanted the main Module to be easy to navigate/understand but also compact and simplified. The design we came up with has all the main components of signing up for classes and creating a schedule while not cutting corners and excluding features we thought would make our website stand out/compete compared to the original.

5.2.6. Alternative Designs

One alternative design that was briefly looked at had the three main sections of class selection, schedule filters, and class view as slightly separated/disconnected. It was initially doubted that all three would fit on a screen at the same time without sacrificing features or size of the main schedule view, but we managed to fit everything into the simplified design without excluding anything important.

5.3 Database Module

5.3.1. Primary Role and Function

The database module is the central repository for the class info and locations, student information, and the saved schedule. After the we scrape DuckWeb, class information is inserted into the table to be used in scheduling. The database in the foundational backbone of the DuckPlanner system. It ensures consistency in data among all the systems and makes it easy for the backend and frontend to access. This organization aids in efficiently retrieving complex queries. The way the database was created also was designed for increased scalability.

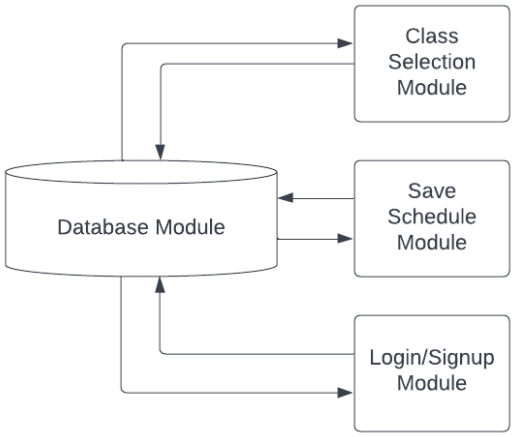
5.3.2. Interface

The user interacts with the database through the log-in/sign up page, the class selection page, and the saved schedule pages. When creating or accessing their accounts, they enter their personal information which is stored in the database. New account information is stored and encrypted for security. Choosing classes is also a dynamically populated list that is from the stored class data. Lastly, when viewing and managing their saved schedules it also accesses this module. In addition, updating and deleting are immediately updated in the database as well.

5.3.3. Static Model

This is not applicable for the database module.

5.3.4. *Dynamic Model*



5.3.5. *Design Rationale*

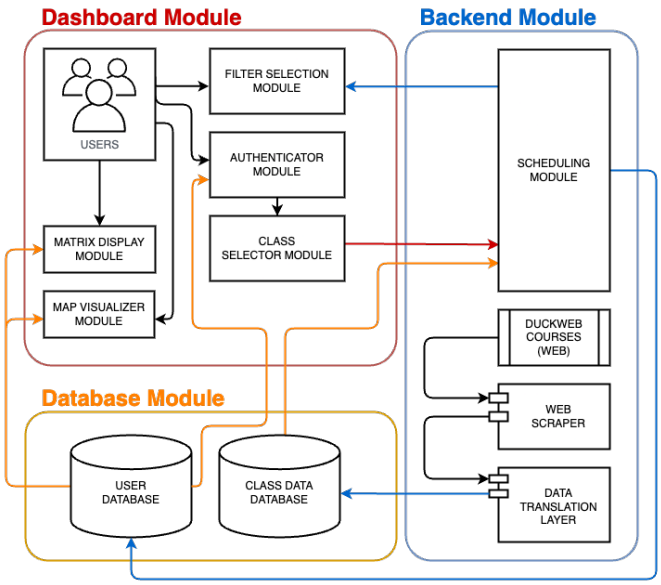
We designed our database due to security considerations, scalability, and performance. Student data is very sensitive information. Using UUID() to protect student IDs and student saved schedules, we were able to prevent any unwanted access into our database and information. The database architecture was also designed to enable scalability in terms of more servers being added or adding more power to the existing servers to be capable of processing more intensive tasks. Finally, we designed it with efficient querying in mind to ensure we are capable of rapid data retrieval. Due to all schedule combination possible, it was important to keep performance in mind for optimized usability.

5.3.6. *Alternative Designs*

Instead of hosting it on the UO CS server, we can host the database on AWS. AWS offer a broad range of capabilities that can enhance our scalability and reliability. By moving our information to a more reliable service, we can ensure that our port connection will not fail and will increase the speed of our performance. However, since the data is hosted on the UO CS server, we are ensuring that student data is secure and private.

6 Architectural Design

The figure provided represents DuckPlanner's overarching architectural framework, meticulously segmented into distinct modules that facilitate interaction among the frontend, database, and backend layers. Each module has been crafted to perform specific roles within the system, ensuring a streamlined workflow and robust data management. The frontend serves as the user interface, providing an intuitive and interactive experience for users, while the backend handles the core logic, processing, and security protocols. The database is the central repository for data storage and retrieval, acting as the backbone for data persistence. This modular approach not only simplifies maintenance and scalability but also enhances security by clearly defining the boundaries and responsibilities of each component in the system.



7 Acknowledgements

We would like to thank Professor Juan Flores for all of the support and wisdom he has provided to our team.