EE/CprE/SE 491
Iowa State A.I. Schedule Companion
Design Document

Saturday, April 27, 2024

Project Title: Iowa State A.I. Schedule Companion
Group Number: sddec24-08
Client: Koby Fowler
Faculty Advisor: Abraham Aldaco

Team Members:

- *Koby Fowler* - Overall Application Leader
- *Chandrashekar Tirunagiri* - Frontend Design Leader
- *Raghuram Guddati* - Backend Design Leader
- *Jacob Paustian* - Artificial Intelligence Research Leader
- *Christian Deam* - Meeting Manager
- *Anna Huggins* - Team Manager

Executive Summary

Development Standards & Practices Used

Practices:

- Project Management Style: Waterfall
- Deployment: Continuous Integration / Continuous Deployment (CI/CD)
- Code Review
- Code Commenting

Standards:

- IEEE/ISO/IEC 29119-4-2021 - International Software Test Design Techniques
- ISO/IEC 14519: 1999 (IEEE Std 1003.5b) - Information technology--POSIX(TM) Ada Language Interfaces--Binding for System Application Program Interface (API)

For additional information, please see section 2.3 Engineering Standards.

Summary of Requirements

- Frontend:
    - Modern, easy-to-read user-interface
    - Prompt-style communication between the user and artificial intelligence
    - Clear page navigation
    - Users can import class/degree documents
    - Users can create/sign into an account
- Backend:
    - SQL Database
        - Stores user information, class preferences, and generated schedules for each user
    - Hosting Service
        - To allow users to connect, we will host our website through a server.
    - ChatGPT
        - Takes in user input via prompts or document upload to generate schedules.

For additional information, please see section 2.1 Requirements.

Applicable Courses from Iowa State University Curriculum

- Com S 309 - Software Development Practices
- SE 422 - Cloud Computing
- SE 317 - Software Testing
- Com S 319 - Construction of User Interfaces

New Skills & Knowledge Acquired

- Artificial Intelligence API training, implementation, and application
- Amazon Web Services (AWS)
- Next.JS abstraction practices

Table of Contents

1. Introduction

1.1 Problem Statement

As fellow Iowa State students, we know very well how difficult it can be to sign up for classes each semester. Especially as incoming freshmen, it feels like an impossible task to try and lay out a four-year plan. For most, it is infeasible for them to do. Thus, students rely heavily on their academic advisors to guide them. Academic advisors here at Iowa State are so crucial that meetings with your advisor can become a hot commodity come class sign-up time. Ultimately, the issue can be summed up as the following:

- There is an overwhelming amount of information to process.
- Information is scattered throughout all of Iowa State's numerous web domains.
- Checking your schedule's correctness is challenging and incredibly error-prone.
- As a result of all the previous points, students have low confidence in their schedule.

We will address these issues through a new app, code-named the *Schedule Companion.* While we have set no limits to what this app can do, our primary goal is to address the issues mentioned previously by creating an AI-driven system that students can interact with. We want to provide tools to the user that allow them to give an AI assistant information about themselves, their academic goals, and their interests, who can also give the user back information that is correct and suitable for them.

1.2 Intended Users

The primary target audience of our project will be Iowa State students, and they will be catered to the most. And although students will heavily benefit from our project, so will advisors and perhaps even professors. Students will benefit the most since the whole project revolves around them and ensuring they are satisfied with our product. By proxy, advisors should benefit from this because they often figure out schedules for the students, so with our application acting as a medium between both sides, this will allow advisors to have an easier job advising students with their classes.

1.2.1 Students

User Description

A current Iowa State student must determine their best class options for their upcoming semester. Although many students meet with their academic advisors to get an idea of what courses they should take, students still must manually determine what courses to take and the specific class times and sections.

User Needs

- A way to generate class schedule options, ideally one that considers their preferences and course requirements
- An easy way to narrow down schedule options without having to sort through hundreds of options

User Results from Project Solution

With our solution, students can input their specific user information and class preferences and allow artificial intelligence to do the heavy lifting; AI will be able to consider student input and help create preferred class schedules rather than forcing students to sort through or create their schedule options manually.

As a result, using our application will allow students to spend less time worrying about their upcoming semester and more time focusing on their current classes. AI Schedule Companion will give students more power, efficiency, and ease of class scheduling.

1.2.2 Advisors

User Description

This user is an academic advisor at Iowa State University who wants to help students determine their upcoming semester class schedule. An academic advisor would benefit from AI Schedule Companion because the application will speed up the scheduling process and allow for the quick creation of a baseline schedule that can be shared with or discussed with students.

User Needs

- An easy-to-follow, intuitive web application for schedule generation
- A way to share and store previously created schedules
- The ability to generate multiple schedules with different criteria
- A quick way of generating schedule options for students

User Results from Project Solution

For academic advisors, our web application, AI Schedule Companion, will allow for the easy generation of semester schedules. One of the primary focuses of our solution has been to maintain easy-to-follow, uncluttered user interfaces such that students and advisors alike can easily navigate through the pages. Additionally, our application will allow academic advisors to create, store, and share various schedules such that they can help multiple students. Lastly, our solution will provide fast, customizable schedule creation- all scheduling considerations are handled by A.I., reducing the workload on the advisor.

1.2.3 User Needs Conclusion

While advisors and students may differ in various ways, both have similar issues: difficulties making or narrowing down schedule options, determining the best fit, and ease of access to a scheduling tool. With our application, thinking is only needed for the initial user information and class preferences input, and the actual schedule generation is passed on to artificial intelligence. By allowing artificial intelligence to create class schedules, human error is reduced by a considerable degree, allowing both advisors and students to spend less time thinking of schedule options.

2. Requirements, Constraints, and Standards

2.1 Requirements & Constraints

2.1.1 Functional Requirements

- **Frontend**
    - User Interface Functionality
        - Users can create an account to store user information and previously generated schedules.
        - Users can generate class schedules by inputting student-specific information and pressing a “generate” button.
            - Student information is made up of user information as well as their class preferences:
                - *User information*: courses previously taken, their major, number of credits they intend to take in a semester, current student year, etc.
                - *Class preferences*: class preferences regarding time of day, location on campus, conflicts with external obligations, etc.
        - Allows users to see class statistics
            - Class statistics: information about class availability, professor, locations, class times, and other class information.
    - User Interface Design
        - Website Theme
            - Iowa State University-themed
                - Primary colors: Red, Yellow, White, and Black.
            - Modern styling
                - Follow design trends in new websites to keep our user interfaces modernized and intuitive.
            - Simplistic UI design
                - Maintain sleek, simple, and readable web pages to reduce confusion among new users.
                - Limit the number of colors to prevent the website from being distracting.

- **Backend**
    - Database
        - Application creates an SQL database of both users (ISU student accounts) and student information (user input)
            - These are linked such that the student account and user information can generate user-specific schedules using that information.
- **Artificial Intelligence**
    - ChatGPT-4.0 Turbo
        - Takes in user input about class requirements to generate schedules.
        - Can take in user documents (like a PDF) to generate student-specific context.
        - Can use existing ISU websites as a source of schedule planning information.
- **Hosting Service**
    - Amazon Web Services (AWS)
        - To allow users to connect, we will host our website through a server.

2.1.2 Resource Requirements

- **Frontend**
    - Scripting
        - *Next.JS 14.1:* A React framework for web application development. This resource allows for access to React Components - premade components that can be added into our webpages without forcing us to manually do in-depth component building. This resource also automatically configures tools necessary for React.
    - Libraries
        - *React:* A JavaScript library for building user interfaces. A tool for easier user interface development.
        - *Material UI (MUI):* A React component library that allows for simply, customizable access to React components.
- **Backend**
    - Scripting
        - *Terraform:* An “infrastructure as code” tool that allows developers to build, change, and version cloud resources efficiently.

    - Database
        - *PostgreSQL:* An advanced open-source database- allows usto store data for our web application.
- **Artificial Intelligence**
    - OpenAI API
        - *ChatGPT-4.0 Turbo:* An artificial intelligence model that can be trained with schedule information and used to generate schedules.
- **Hosting**
    - Amazon Web Services (AWS)
        - *Amplify:* A fully-managed continuous integration/continuous deployment and hosting service. This resource allows for fast, secure, and reliable rendering for our web application.
        - *Elastic Compute Cloud (EC2):* A web service that provides on-demand computing and scalability capabilities for our application. This tool allows us to build and host our web application.
        - *Amazon Relational Database (RDS):* A web service that provides database automation; RDS allows for easier set up, operation, and scaling for our relational database in the AWS Cloud.

2.1.3 Additional Requirements

- **Physical**
    - Accessibility
        - The application will be hosted online and can be accessed via a website on a mobile device or computer.

2.1.4 Constraints

- **Time**
    - Deadlines
        - For this project, our final deliverable deadline is the end of CPRE/EE/SE/CYBE 492
        - To accommodate our final deadline, we have a team deadline for the end of April for a working prototype application that can take user input and generate accurate schedule options.
- **Scope**
    - Project Deliverables
        - The application is accessible via mobile device or computer.

        - The website allows students to create an account.
        - Student accounts store their user information, preferences, and previously generated schedules.
        - The application combines ChatGPT with student input to generate custom, accurate class schedules.

2.2 Engineering Standards

- **IEEE/ISO/IEC 29119-4-2021 - International Software Test Design Techniques**
    - Equivalence Partitioning:
        - Categorize student information inputs into different equivalence classes. Test each partition to ensure proper handling of various scenarios. It will be beneficial for testing consistency in AI-generated output.
    - Boundary Value Analysis:
        - Identify maximum and minimum values for credit hours, GPA requirements, etc.
        - Test with boundary values to verify correct behavior at the edges of acceptable ranges, particularly regarding user input and class information.
    - Decision Table Testing:
        - Define rules for course prerequisites, credit hour requirements, and constraints using decision tables.
        - Test different combinations of input conditions against expected outcomes.
    - State Transition Testing:
        - Model various states of a student's class schedule (e.g., planning, finalized, in progress).
        - Test transitions between states (e.g., adding and dropping courses) to ensure appropriate handling.
- **ISO/IEC 14519: 1999 (IEEE Std 1003.5b) - Information technology--POSIX(TM) Ada Language Interfaces--Binding for System Application Program Interface (API)**
    - Interoperability Testing:
        - Confirm our application works seamlessly with other systems or tools (such as OpenAI or AWS).

        - Test communication and exchanging data with external systems. This will be important to manage our resources efficiently.
    - Conformance Testing:
        - Ensure our application meets the defined specifications. Given inputs should not yield somewhat random outputs.
        - Test various features and functionalities to ensure they adhere to the specified criteria. Given a scenario, our application should not produce schedules with errors.
    - Robustness Testing:
        - Ensure our application can handle unexpected situations or inputs.
        - Test error-handling mechanisms and boundary cases to ensure the application remains stable and reliable.

3. Project Plan

3.1 Project Management & Tracking Procedures

3.1.1 Project Management

Our team has adopted a more traditional waterfall approach to our project management. Waterfall development was selected to allow us, as developers, greater flexibility between project development and external facts, like school and work. Additionally, a waterfall approach makes more sense in the scope of our project. Our project requirements are clear, detailed, and unlikely to change. With concrete project requirements, this stability allows for comprehensive planning and design before much of the core development has even begun.

3.1.2 Tracking Procedures

For tracking procedures, our team utilizes team-based communication and accountability in addition to virtual tools for project development.

We use Snapchat and weekly team meetings to track progress between teammates for group communication and relaying updates. We use full-team meetings (in-person and virtual) for overall project updates and small-group meetings for specific development teams to focus on end-specific issues, task decomposition, and discussions. For example, a frontend team meeting would require the attendance of only frontend team members, and likewise, a backend meeting only requires backend developers.

Additionally, we utilize virtual tools for tracking progress, such as Git for version control and GitLab for source control. These tools allow our team to track individual progress as well as maintain a clear understanding of what features have been implemented.

3.2 Task Decomposition

3.2.1 Primary Tasks

- Frontend Design - Website design and user interfaces.
- Backend Infrastructure - Backend code for databases.
- Artificial Intelligence - Used for schedule generation.

3.2.2 Task Relationship Overview

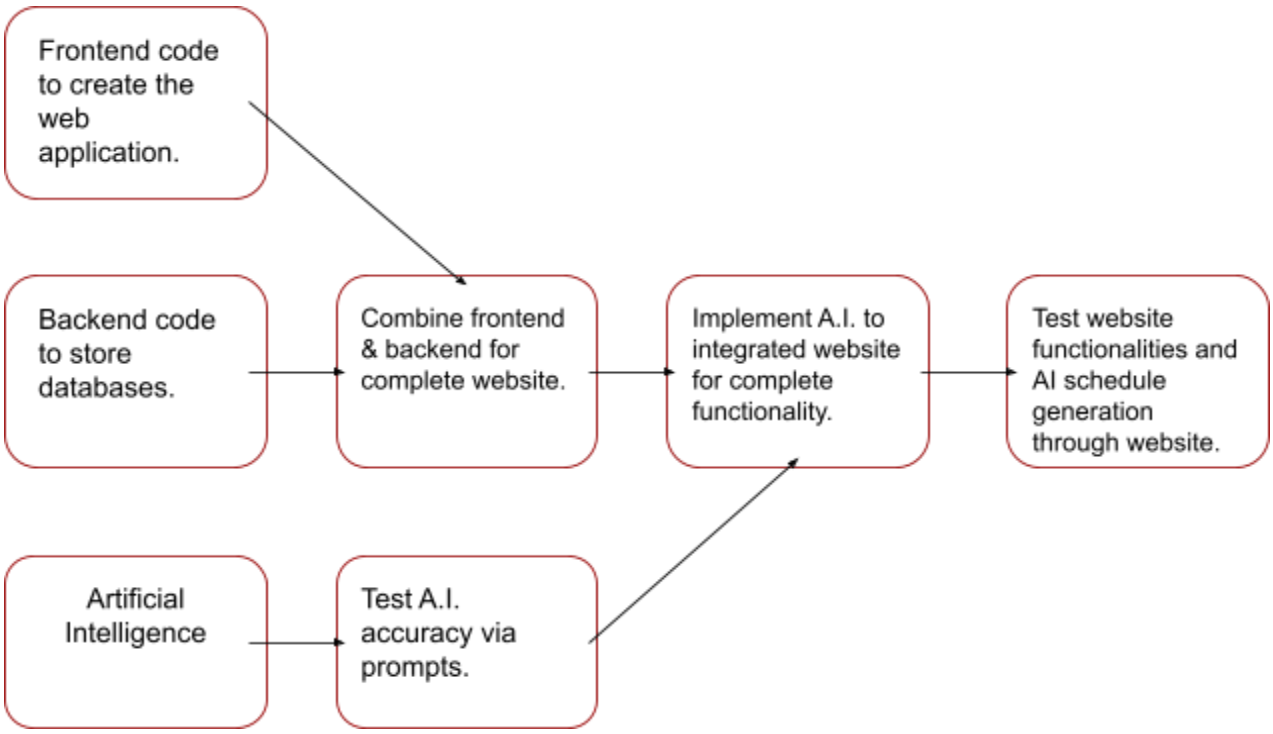

Figure 1.0: *Task Relationship Overview*

The above graphic visualizes our task decomposition through our three main application components: frontend, backend, and artificial intelligence. The frontend will focus on website and user interface design in our project. The backend will focus on website infrastructure and database design. Artificial intelligence refers to using OpenAI API to generate schedules and interact with user data.

While some development of these components may be conducted separately, each of these three components will be integrated into our application for full functionality.

3.3 Project Proposed Milestones and Evaluation Criteria

Iowa State A.I. Schedule Companion v1.0 Completion - Deadline: March 29, 2024

This milestone agglomerates the development of all application components: frontend, backend, and artificial intelligence. Throughout the first phase of development for our application, each component will initially be developed independently until we reach the integration phase.

Each component will undergo research and introductory development during independent component development, specifically:

- *Frontend*: webpage design, creation, and modification.
- *Backend*: initial database infrastructure setup and code testing.
- *Artificial Intelligence*: A.I. research to understand its capabilities and how it will interact with our application later in development.

Once we enter the integration phase, we will begin creating and testing interactions between different components. First, we will test frontend access to the backend databases, i.e.) accessing class information and storing user input in the database. Next, we will integrate artificial intelligence into our web application for initial integration testing. This phase will be a stepping stone before additional test runs and deployment is completed.

To evaluate this milestone, we will analyze our project's existing code and what is accessible and possible when hosted locally.

Evaluation of this milestone entails:

- *Baseline user interface design*: Majority, if not all, web pages are created, and page navigation is set up. Some webpages may undergo updates to theme design or layout, but all pages and navigation should be working for v1.0.
    - Pages: Home, Login, Sign-Up, User Information, A.I. Scheduler, Previous Schedules, and Class Preferences.
- *Database accessibility*: The frontend user interfaces can access and display database information and store information from user input.
    - Data: user information, account information, and class preferences.

- *Able to be locally hosted without errors*: The website is able to be hosted locally on a team member's machine. When locally hosted, our front and backend code should compile and run without errors.

Conduct User Test Run - Deadline: April 5, 2024

This milestone is the next step after completing *Iowa State A.I. Schedule Companion v1.0* and acts as the completion of our first significant trial of application testing. In this milestone, we will test the primary function of our application: artificial intelligence schedule generation.

To evaluate this milestone, we will do a full run-through of the application's functionalities, namely the criteria listed below.

Evaluation of this milestone entails:

- *User Login/Sign-Up:* A student can log in and/or create an account with their email and specified password.
- *Ability to Enter User Info and Preferences:* Once logged in, the user can answer prompts on the "User Info" page to give specific answers regarding year classification, major, and previous grades. This information will be stored in our database and visible to and editable by the user. On the Class Preferences page, a student will be able to answer additional questions regarding specific class preferences, i.e.) class time, location, in-person vs. online, etc.
- *Artificial Intelligence Schedule Generation:* Once users have submitted their information and preferences, they can access the "A.I. Scheduler" page. In this state, the A.I. scheduler will be tested to create a schedule based on the user information and preferences.
- *Artificial Intelligence Schedule Storage:* Once the A.I. has generated a schedule, this schedule should be visible to the user on the webpage, saved in the database, and accessible on the "Previous Schedules" page.

3.4 Project Timeline

3.4.1 Timeline Overview

Below is our project timeline for the first development semester. In our timeline, we divided our project into 5 phases: planning, design, development, testing, and deployment. For this project, each phase corresponds to three main components of our application: frontend, backend, and artificial intelligence.

3.4.2 Timeline Table

Phase	**State Date**	**End Date**
Planning	March 1, 2024	March 7, 2024
Design	March 8, 2024	March 14, 2024
Development	March 15, 2024	March 28, 2024
Testing	March 29, 2024	April 4, 2024
Deployment	April 5, 2024	April 11, 2024

Table 1.0: *Timeline*

3.4.3 Target Breakdown

Target 1: Service Building & Core Features - March 1, 2024 - March 14, 2024

- ***Backend*** - Developers begin database infrastructure setup, awaiting integration with frontend user interfaces.
- ***Frontend*** - Core webpages are created, and user interface designs are updated to work with backend infrastructure.
- ***Artificial Intelligence Service*** - Undergoes initial design, data modeling, and information training.

Target 2: Implementation & Integration - March 15, 2024 - March 28, 2024

- ***Frontend and Backend Integration*** - Connect frontend and backend, and begin integration testing.
    - Link backend databases to frontend user interfaces, namely allowing frontend to access information already in the database, i.e.) class information.
    - Link frontend prompts to backend databases to store user information, preferences, and eventually generated schedules.
- ***A.I. Integration*** - connect backend databases and frontend to artificial intelligence.

Target 3: Testing & Application Polishing - March 29, 2024 - April 4, 2024

- ***Web Application Analysis*** - Compare the A.I. Scheduler web application to existing Iowa State University applications to verify our website is: similar to use, utilizes similar page designs and themes, and allows Iowa State students easy access to schedule generation.
- ***Prototype Testing*** - Begin testing of all integrated features, i.e.) the frontend can access backend databases, user information is correctly stored in the databases, and both frontend and backend can access artificial intelligence for schedule generation.
- ***Confirmation of Schedule Integrity*** - Verify that generated test schedules are sufficient without significant errors.
- ***Security Considerations*** - Ensure our application is secure regarding user accounts, information, and personalized class schedules.

3.5 Risks and Mitigation

Unable to complete Milestone: *Iowa State A.I. Schedule Companion v1.0*

Completion - Deadline: March 29, 2024

Failure to achieve this milestone will result in our project being behind our preferred schedule, requiring additional hours to be put in this semester and early next semester. However, at our current pace, we will be on track for a complete and functional web application before the final deadline next semester.

To mitigate this risk, we meet weekly to hold members accountable and discuss any issues that may need to be addressed throughout the next development cycle. Additionally, since our team has been cohesively in this project's development and research phase, we have already made great strides to stay caught up before our final deadline.

3.6 Personnel Effort Requirements

Below is *Table 3.0: Personnel Effort Requirements*, which lists various tasks necessary to complete our project and the estimated amount of time put forth and yet to be completed.

Task #	**Task**	**Description**	**Projected Number of Hours to Complete**
1	Initial repository setup	Implement database and frontend resources into GitHub and AWS setup for hosting.	12 hours
2	Initial Backend Database Set up	Creation of MySQL database and group discussion regarding database design.	8 hours
3	A.I. Research	Introductory research conducted regarding OpenAI capabilities.	8 hours
4	Complete initial frontend web application design	For our website, we estimate there to be approximately 7-10 pages; each of these pages will need to be designed by a member of the frontend team.	35 hours split among the frontend development team *I estimate about 5 hours/page, as each page will have an initial design plan and be modified during implementation.
5	Frontend & backend integration	Implement backend accessibility logic in the frontend code. Verify front-end connection to databases via storing and accessing.	12 hours
6	A.I. Integration	Implement artificial intelligence logic into the frontend and give it access to the backend database for schedule storage.	8 hours
7	Version 1.0 User Testing (Local)	Conduct user test run locally of our web application; verify frontend can access backend databases and that A.I. is accessible and can generate schedules.	2 hours

8	Continued Frontend Design Updates	Based on the testing results, update webpages as needed to improve: ease of use, readability, and functionality.	12 hours
9	Continued Backend Updates	Based on the testing results, modify backend databases as needed to work efficiently.	12 hours
10	Web Application Deployment	After successful local testing, we will begin the online deployment of our website for additional testing and development.	8 hours
11	Consequent Testing	Based on frontend and backend changes, conduct another user test and obtain feedback from test users.	2 hours
12	Final frontend web application modifications	Based on later web application testing, clean up user interfaces and forms.	6 hours
13	Final backend database code modifications	Based on later application testing, modify database infrastructure as needed.	6 hours
14	Final Complete Application Testing	Confirm that all application functionalities are working as best as possible and that the application is easy to use.	1 hour

Table 2.0: *Personnel Effort Requirements*

In the above table, various efforts are described regarding our project. Most project tasks are divided into 3 categories: frontend, backend, and artificial intelligence. Additionally, there is a focus on initial design and development, integration, testing, and web application deployment. The tasks listed are the primary tasks that must be executed for completion of our project.

3.7 Other Resource Requirements

ETG Correspondence for Backend Resources

Some additional resource requirements are: access to Amazon Web Services (AWS), OpenAI, and GitLab for this project. In the initial construction phase of this project, we faced some issues regarding permissions in GitLab during setup in GitLab when trying to connect AWS and OpenAI services.

4. Design

4.1 Design Exploration

4.1.1 Broader Context

Our application is designed for Iowa State University students and advisors to generate class schedules more efficiently, accurately, and easily. Our project addresses the pending needs of students to have an efficient way to generate schedule options and add additional functionality to consider user preferences. For more information on user preferences, please see section 2.1.1 Functional Requirements - Frontend. Below, more contextual considerations are listed in *Table 3.0: Broader Context Considerations*.

Area	Description	Examples
Public health, safety, and welfare	Our application is intended to positively impact users, primarily Iowa State University affiliated students and advisors, as addressed in section 1.2 Intended Users. The impact of our application is to improve the efficiency and accuracy of schedule generation for our intended users, overall increasing user well-being. Indirectly, there will be little to no impact on external public welfare.	- Reducing the workload and time students and advisors must manually spend creating schedules. - Increasing the quality of generated schedules by considering more specific user needs. - Reducing frustration when narrowing down schedule options by providing more specific, user-oriented options.
Environmental	Overall, this web application will be sustainable and have minimal environmental impact, primarily due to the fact that this application is Iowa State University-oriented rather than extending to various universities across the United States. This application will not require manufacturing or extensive use of environmental resources.	- The goal of this application is to utilize as few resources as possible, but the application will ultimately require hosting functionalities for accessibility and database storage.

Economic	Overall, this project will have some economic costs for our team/Iowa State University but will be free to users. Costs for our application include: tokens spent training A.I. with user information and hosting our application on a server.	- No cost for users to utilize this web application. - Some cost for utilizing A.I. to generate schedule options. - Additional cost for hosting and maintanence of web application.

Table 3.0: *Broader Context Considerations.*

4.1.2 Prior Work/Solutions

Iowa State University Schedule Planner

The previous solution for class schedule generation was the Iowa State University Schedule Planner [4]. This tool allows students to directly access class information from Iowa State, ensuring accuracy of section times and course availability. In this application, schedules can be automatically generated based on the courses selected manually by the user, providing numerous schedule possibilities based on various class sections. The below figure is a screenshot of the schedule planner.

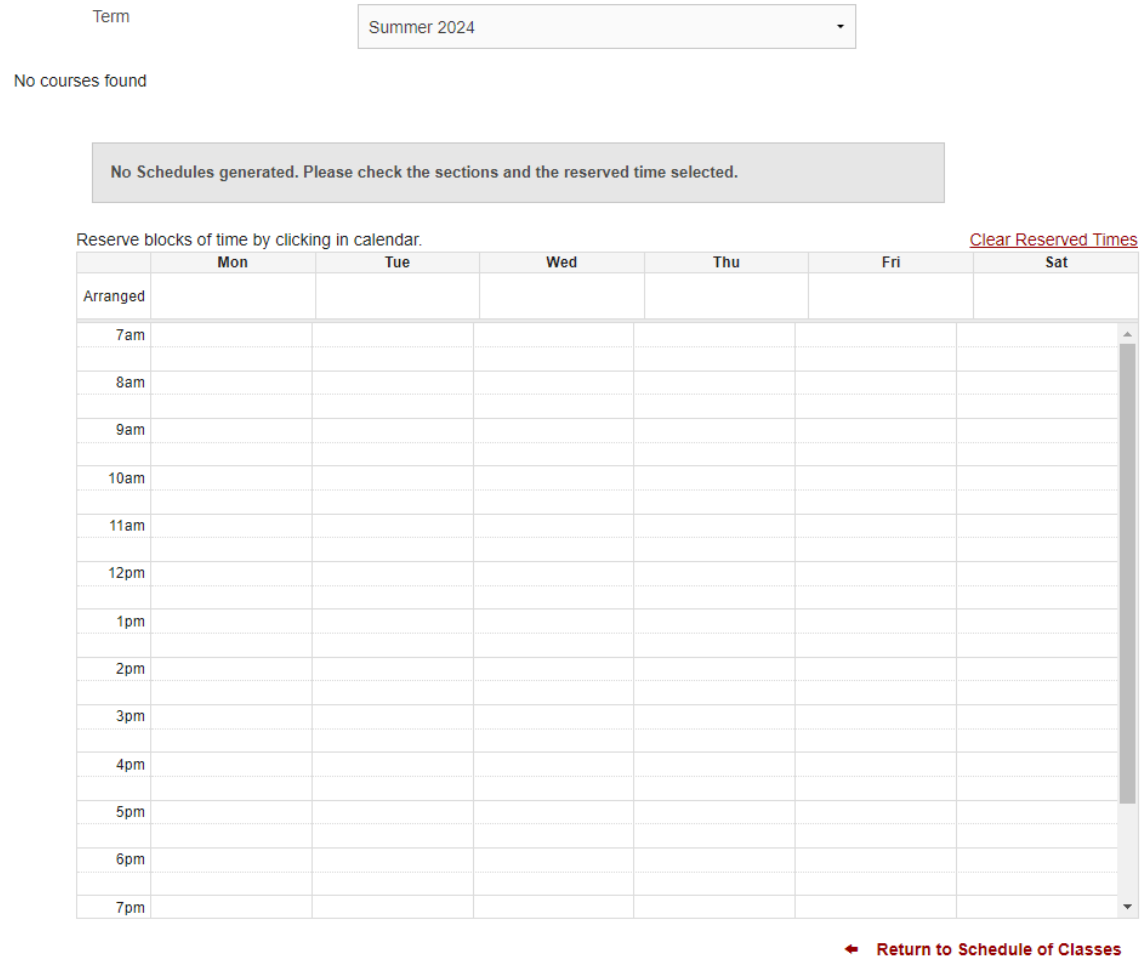

Figure 3.0: *Iowa State Schedule of Classes Schedule Planner.*

Pros:

- Users can directly access Iowa State class information
- Can accurately generate schedules based on selected courses manually added by the user

Cons:

- This application will be discontinued by Fall of 2024
- Students must manually weed through generated schedules to meet their preferences
- Users must manually input specific course requirements for the given semester
- Cannot account for user preferences

Our solution, A.I. Schedule Companion, is important for two primary reasons:

1.) The current online schedule planner will be discontinued following Iowa State's switch to Workday, so there will be a need for a new application.

2.) Our application will go beyond the previous site's functionalities.

A.I. Schedule Companion will utilize a similar foundation of taking Iowa State class information, but our solution differs in schedule generation. In our solution, user information and artificial intelligence will be used to create user-specific schedules that cater to user preferences, reducing the hassle of manually sorting through generated schedule options.

Workday

As of Spring 2024, Workday is the new system that Iowa State utilizes for student information and course scheduling [5]. Accordingly, Workday has a system to create schedules; this system allows students to manually select courses and add them to a “Saved Schedule” which can later be used to register for courses. In the figure below, a screenshot of Workday’s schedule creation system is shown [6].

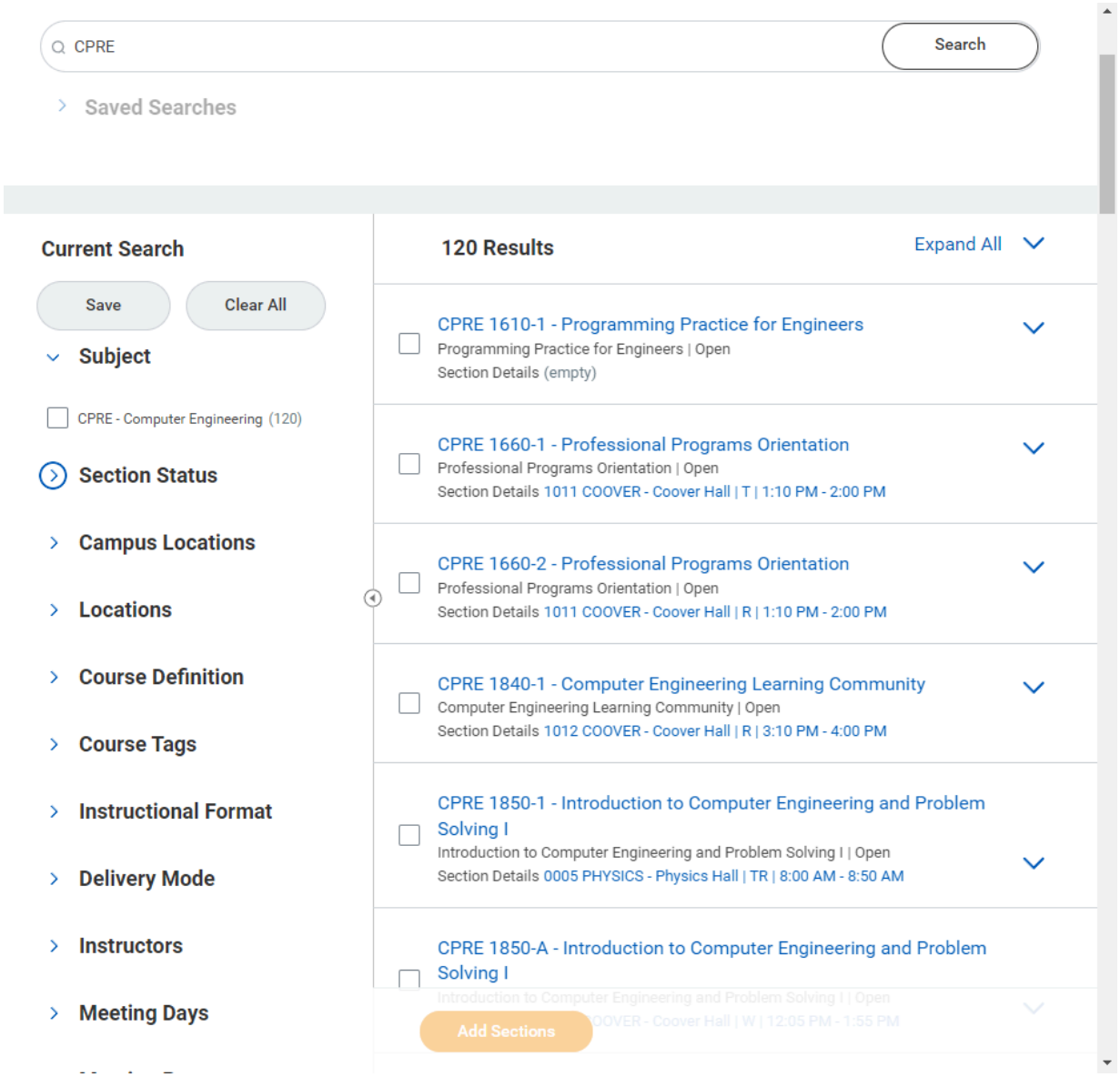

Figure 3.1: *Workday Course Adding.*

In Figure 3.1, on the left side there are various search options that can be used to narrow down classes via locations, instructional formats, delivery, and more. These are options allow students to toggle which types of courses they are to see, and then can manually add them to a “Saved Schedule” accordingly.

Pros:

- Direct access to Iowa State class information
- Can detect course time conflicts and reports them to the user

Cons:

- User interfaces aren’t very intuitive
- Cannot account for user preferences
- Users must manually select courses and determine which sections to take (to prevent conflicts)

Overall, the “Saved Schedules” within Workday are helpful for class registration, but ultimately there isn’t a robust means of generating schedules that cater to student preferences. In A.I. Schedule Companion, clean, easy-to-follow user interfaces will be utilized to help make the schedule generation process seamless.

4.1.3 Technical Complexity

Our project, A.I. Schedule Companion, is made up of 3 primary components- frontend user interfaces, backend databases, and artificial intelligence. For a more detailed overview of the project structure and components, please refer to section 4.3 Proposed Design.

Frontend

On the frontend, development efforts are being put forth in designing sleek, easy-to-follow user interfaces that provide clear guidance for users and display relevant information. This development is being conducted Next.JS and utilizes library components from Material UI (MUI) and React. Even with these tools, frontend development has provided a moderate level of difficulty, requiring developers to gain new skills and insights when using resources they have not previously, and when working to cater our designs to students.

The frontend focuses heavily on

Backend

On the backend, development is focused on providing efficient, secure storage of user information and accessibility of Iowa State class information. The backend is utilizing a PostgreSQL database to store student account information, preferences, and created schedules. Backend development requires connection to the frontend to obtain user information as well as allow the frontend to display stored preferences, information, and schedules. The backend must also be connected to the artificial intelligence such that when the A.I. generates a schedule, the schedule can be stored by the user.

Artificial Intelligence

Another major component of this project is artificial intelligence. A.I. is the driving force behind schedule generation in this project. Where frontend is designed to make the schedule creation process easier, A.I. does the heavy-lifting of actually generating schedules that caters to users' class preferences, current user information, and course availability. Utilizing A.I. makes the process easier for the user, but behind the scenes, the A.I. must be trained using Iowa State course information and 4-year plans for various majors. The A.I. component must be able to effectively build accurate schedules and allow functionality such that a user can modify their preferences and course criteria.

Integration & Hosting

A significant piece of this project is integration and hosting. In order to combine the aforementioned project components above, all components must be integrated and function cohesively. Independently each component may be function, but in order for this project to be effective, all components must be integrated and our web application must be deployed online for user access. Integration of this project is primarily done utilizing Amazon Web Services (AWS) tools and GitHub for Continuous Integration/Continuous Deployment (CI/CD). As our project is a web application designed to make students' lives easier, our website must be hosted reliably and securely so that students can generate schedules.

Summary

Overall, the project complexity is driven by the integration of various components that must all work together in order to effectively assist students in schedule generation. This project is intended to replace and improve upon what Iowa State Schedule Planner was (please refer to section 4.1.2 Prior Work/Solutions for more information regarding the schedule planner.) In order to be an effective solution, all components of this web application must be fully functioning and integrated, and tested to allow for effective schedule generation. Our solution will allow students an easier time generating schedules by utilizing sleek user interfaces, robust database handling, reliable hosting, and trained artificial intelligence.

4.2 Design Exploration

4.2.1 Design Decisions

Initial Set Up - Infrastructure and Resources Selected for Development

Our solution's first major design decision was regarding what tools and resources would be utilized for development, infrastructure, and hosting. Our first team meeting discussed the various APIs and resources required for frontend and backend development, artificial intelligence access, and website hosting. When deciding which resources to use, we considered: familiarity, available documentation, integration capabilities, and cost.

To view the specific resources selected from this decision, please see section 2.1.2 Resource Requirements.

User-Base

In the early stages of development for this solution, we had to identify the user scope of our project.

Our primary question was:

- Do we want to cater only to Iowa State Students, or do we want to allow students from other universities to use this application?

Regarding the user base for this project, we decided to limit our project to Iowa State University students because: we are most familiar with the Iowa State student base, we have flexibility in accessing Iowa State University information and resources, and we want to keep the capabilities of our application reasonable, considering available hosting and database constraints.

Through our own Iowa State student experiences, our team is aware of the schedule generation issues and needs of other students, specifically at Iowa State. As our team is a part of the user base, we can reach out to other students for specific feedback, allowing Iowa State University-specific schedule generation and accommodations to provide the best possible user experience.

Regarding resources, we will be able to more easily access information related to Iowa State University, like class information, current registration capabilities, Iowa State application designs, etc. Additionally, for web application assistance and funding, we will utilize the Electronics and Technology Group (ETG) for assistance throughout this project.

Lastly, another big consideration when making this decision was the capabilities and resources needed for our application. Broadening our web application to more universities would be cumbersome for our database, hosting, and the artificial intelligence training process. If our application were to be utilized by various other universities, we would need extensive resources to maintain our database, train our artificial intelligence, and allow many users to access the application. Regarding resource consumption, limiting this project to Iowa State University allows us to generate ISU-specific schedules without exceeding our application's baseline capabilities.

Application Features

For our project's scope, we also had to consider how many features we wanted to include in our web application. Namely, we considered: Do we want to provide additional features, beyond schedule generation?

Ultimately, for this decision, we opted to limit our project to artificial intelligence schedule generation and schedule storage. For this decision, we considered the timeline for our project-additional, unnecessary features would considerably increase the workload. Another consideration was that if we included too many additional features, our application could be incomplete and jumbled with unnecessary capabilities.

By reducing the scope of our application's features to schedule generation, we will be able to focus more time on the quality of our application's primary goal rather than the quantity of additional features.

4.2.2 Ideation

Design Decision

One of our main design decisions was "simplicity in getting schedules." From that, we were able to form exactly five options for solving this decision: clean user interfaces, clear, concise user input, ease of understanding schedule, compare/contrast with known classes, and additional features. The last option would be better implemented when the project base is completed, but it was an option nonetheless.

Clean User Interfaces

User interface, or UI for short, is the point where man and machine communicate with each other. It is difficult to say what is *not* a UI when dealing with computers; every icon is one, web browsers are one, and even the computer mouse is UI. If building a project involving computers and applications that actually work is the main goal of a team, then creating a legible UI is the second most important. The reason for this is because it is the language barrier. In order to properly tell your users where to go, you have to illustrate through the UI. Unlike yellow-paint design*****, the UI should not baby the user in any way on where to go. Instead, it should make all, if not most, of its functions available from the screen they start on. They can be hidden in separate tabs such as the Store tab having "for you" and "special" options underneath it. With our design, we devised some ideas on how to do this properly.

- Utilize page organization (i.e., drawers/side tabs for navigation)
    - This is precisely the same as the Store example from the prior paragraph.
- Allow scrolling on pages or have it easy to store more information on the same page
    - The idea is to eliminate the number of pages one can go to, thus allowing them to wait less time to traverse to a page they think might have the answer they are searching for.
- Create simple prompts that give clear spaces for user input/answers
    - If the user has to input something, what we are asking/prompting them to do should be clear.

    - A pseudo-example of bad user input would be having a prompt for “Name” and then “Last Name.” When “Name” is present, most people assume that it means their first and last name, but the website meant for only the first name.
- Reduce page clutter - only have the necessary information
    - If the information presented before someone only has fifty percent of what they want/need, eliminate the other fifty as it is distracting.
    - This is no excuse for poor grammar or cavespeak, though. Much like the presentations, use bullet points and sentences to illustrate the point.
- Use asset color scheme to avoid distracting web pages
    - An overuse of colors or misuse of them can cause the eyes to struggle to read or comprehend the page. It is good to remember that humans were not made to look at electronic screens, so if they should do so, they should look at something pleasant or at least not damning to the eyes.

Clean User Input

For websites/applications that demand user input, the most important aspect of this is being clear with your wording. Not being specific in even a single word can cause confusion, including the incredibly common “a” and “the” as both preface different uses of the singular tense. The question “point at *a* square” vs. “point at *the* square” differs in the fact that with “a”, it appears there is more than one square, and they are tasked with only pointing at one. Meanwhile, “the” indicates there is only one in this picture. With that example aside, we had further ideas to emphasize this point.

- Edit button - allows users to change their input as needed
    - This option is if the user clicks to the next page when answering prompts and realizes they got incorrect information beforehand.
- Simple user prompts for initial input
    - No paragraph-based questions should be given to users for questioning, just simple sentences. That is acceptable if an example is needed to illustrate what is being appropriately asked.
- Avoid open-ended questions
    - With simplicity, it should also be specific. Look back to the square example, for instance.
- Allow AI to ask specific follow-up questions to clarify schedule criteria

    - This would be given to the user if some of their information conflicts which then confuses the AI.

Ease of Understanding Schedules

As with most things, schedules are best understood via visuals. However, as with UI, schedules can get confusing if not properly explained. The greatest aspect of a schedule is how it can be tailored to someone personally which is usually just caused by them understanding their own schedules. With our program, we hope for that personalization as it will encourage students to use it more often if they can relate. To ensure this happens, we had some ideas to help.

- Create a visual/chart of generated schedule options
    - Instead of visuals of just selected classes, it would show the schedules with all of your personalized options.
    - To avoid the clutter of time slots, it would most likely be organized to have the AI's highest accuracy-based classes appear in front of the others (which are desaturated until specified by the user, which then desaturates the rest so only one can be seen).
- Provide simple text-based generated schedule
    - This option would be best used if the user was in a hurry, a simple text document that shows the assumed classes they want to take.
    - It is stated in the sample that it is "not as helpful."
- Provide an image of the ISU 4-year plan and the generated semester schedule
    - This helps the users and the AI understand the schedule they are/will be given. Essentially, a template for both parties.
- Ability to indicate which classes count for which requirement (i.e., tech elective, computation elective)
    - This would help the user understand why that class was picked over other options, which would also aid them in understanding their own audit.

Compare/Contrast Known Classes

The last relevant option refers to the ability of the AI to look at every single ISU class. With a lot of information, it can correctly assign person X with their preferred and required classes. There are some ideas to help explain further.

- Take into account classes previously vs ones still required to take.
    - This would take in the audit of the student to properly give them what they still need to take instead of accidentally recommending already completed courses.
- Compare required classes for students with known and available classes.

- When somebody activates the scheduler, it will create the accommodated schedule and discard any full classes afterward. It would also state that X class was full and thus not on the options.

*: “Yellow-paint design” refers to illustrating where to go in a video game via yellow paint. This is done to ensure a player does not get stuck in a location. Although seemingly harmless design, it has been overused in that even easy-to-see paths have that paint which is seen as “babying” the player.

4.2.3 Decision-Making & Trade-Offs

For clean user interfaces, the main pro would be that the website itself would look clean and well-made. This would ensure the user traverses the correct site to get what they want. The only con would be the difficulty of making the perfect website. Since website design could be classified as art, only a few engineers have experience in that field and thus would struggle.

With clean user input, users would always be clear when prompted to type something in. They would be able to understand what is being asked and why clearly. However, the English language is a difficult line to tread upon. Unless one has spent *days* writing and understanding how to make proper questions, a simple slip in the crack could cause confusion.
Given we are working on making a scheduler, an important aspect would be allowing everyone to understand the schedules we are creating. Doing so would encourage them to use our website more as it is easy to understand. Unlike the other options, there is no real downside other than making it too simple for somebody. The idea is to have it personalized; having it too simple can drain the soul.

Finally, with the AI looking at every class, it can easily come to quick and concise conclusions on making the perfect schedule. It can look at which classes are needed for somebody’s education, which classes they want, and which classes are full to make a great schedule for them. The issue lies in giving the AI all the data and hoping it can accurately make a schedule to suit their needs. With such pros and cons, we are doing all the options. That’s because each option feeds into each other, and only focusing on one would reduce the quality of the product. Along the way, we will focus on one *at a time,* but we will still focus on all of them.

4.3 Proposed Design

4.3.1 Overview

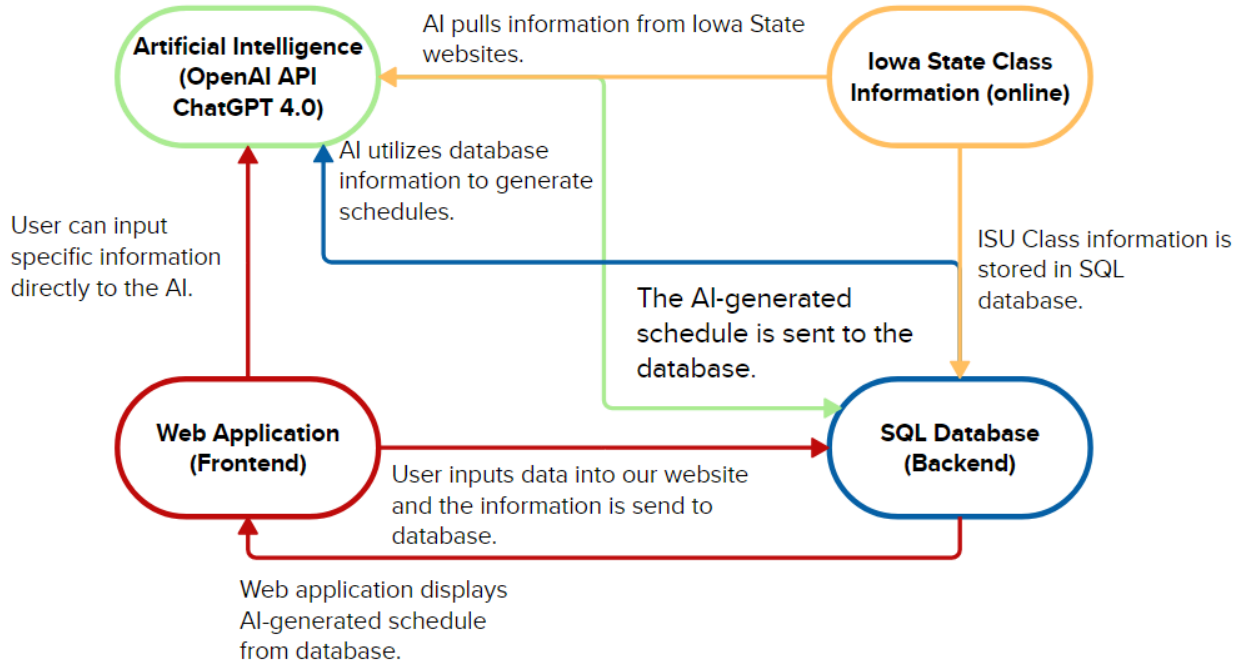

Figure 4.0: *Design Overview.*

Per Figure 3.0 shown above, our web application can be broken down into 3 primary pieces, with two components accessing an online resource (Iowa State class information online.)

In the above graphic, it is essential to note that users will only directly interact with the “Web Application” node, which visually all users will be able to interact with. The “Artificial Intelligence” and “SQL Database” nodes will be behind the scenes, allowing for efficient schedule generation or storage of user input into our frontend application.

Frontend (User-side)

In our project, users will access a website that will allow them to sign in using their Iowa State credentials and then allow them to enter preferences and requirements for their class schedules. Once the information has been input, users will be able to access the A.I. Scheduler page, which will allow them to generate schedules based on their input.

Backend

On the backend, our use of OpenAI API will allow us to utilize artificial intelligence to generate schedules based on the user information stored in our database and store the generated schedules in the database to be presented on the frontend.

External Resources

For AI training purposes and ease of access, our database and artificial intelligence will access Iowa State websites for class information. For our artificial intelligence, the Iowa State Online class information will be used for training, as the AI will need a basis for how 4-year plans for various majors work and what classes are available and when. For our database, more specific information may be stored regarding classes.

4.3.2 Detailed Design & Visuals

4.3.2.1 Detailed Overview

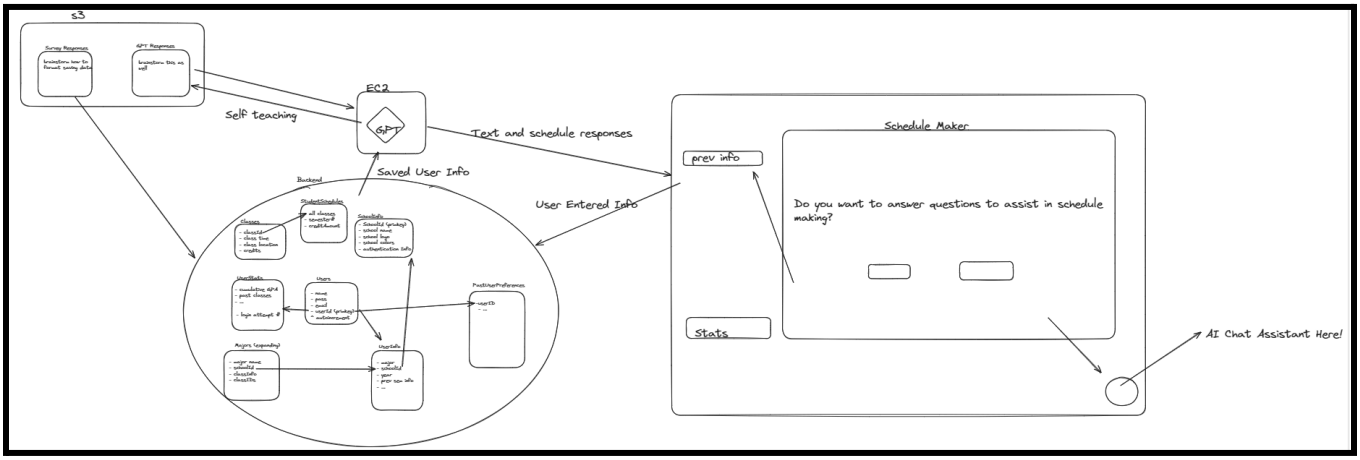

Figure 4.1: *Detailed Design Overview.*

Frontend Overview

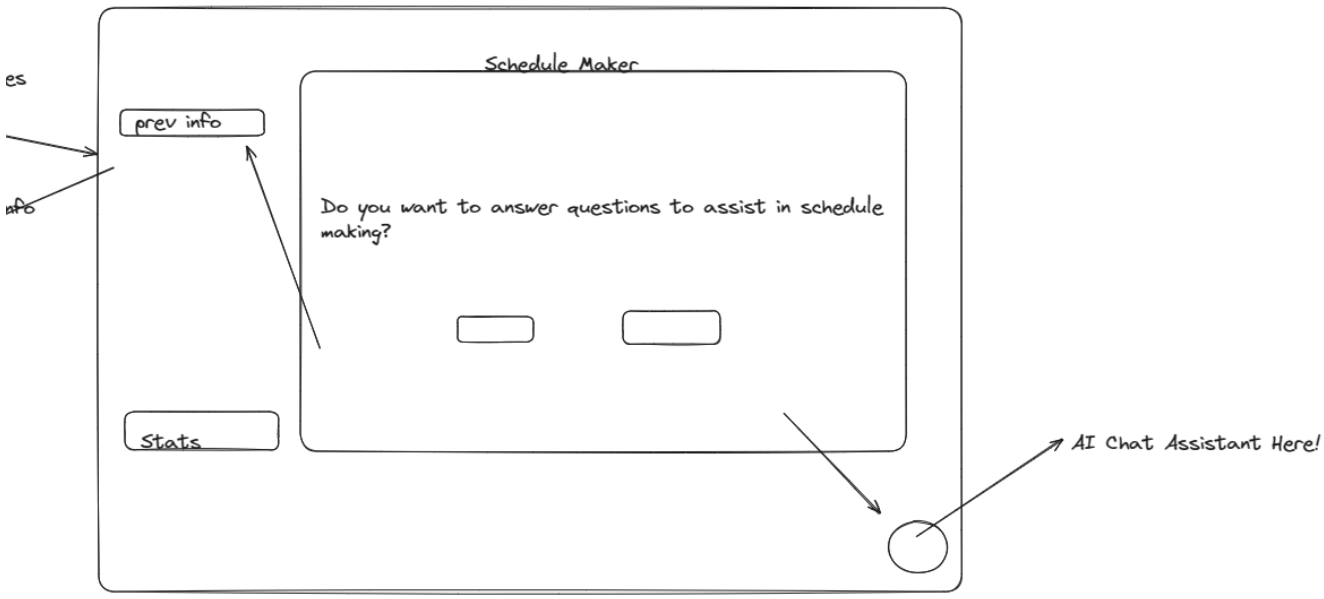

Figure 4.2: *Schedule Maker Page.*

In **Figure 4.1**, an example AI Scheduler webpage is drawn. This drawing is a snippet from **Figure 4.0**, which indicates its relation to other components of our application. The scheduler page allows users to answer prompts so that artificial intelligence can generate a more specific schedule for their needs. This webpage allows user information to be stored in backend databases so the AI can access it later, if necessary.

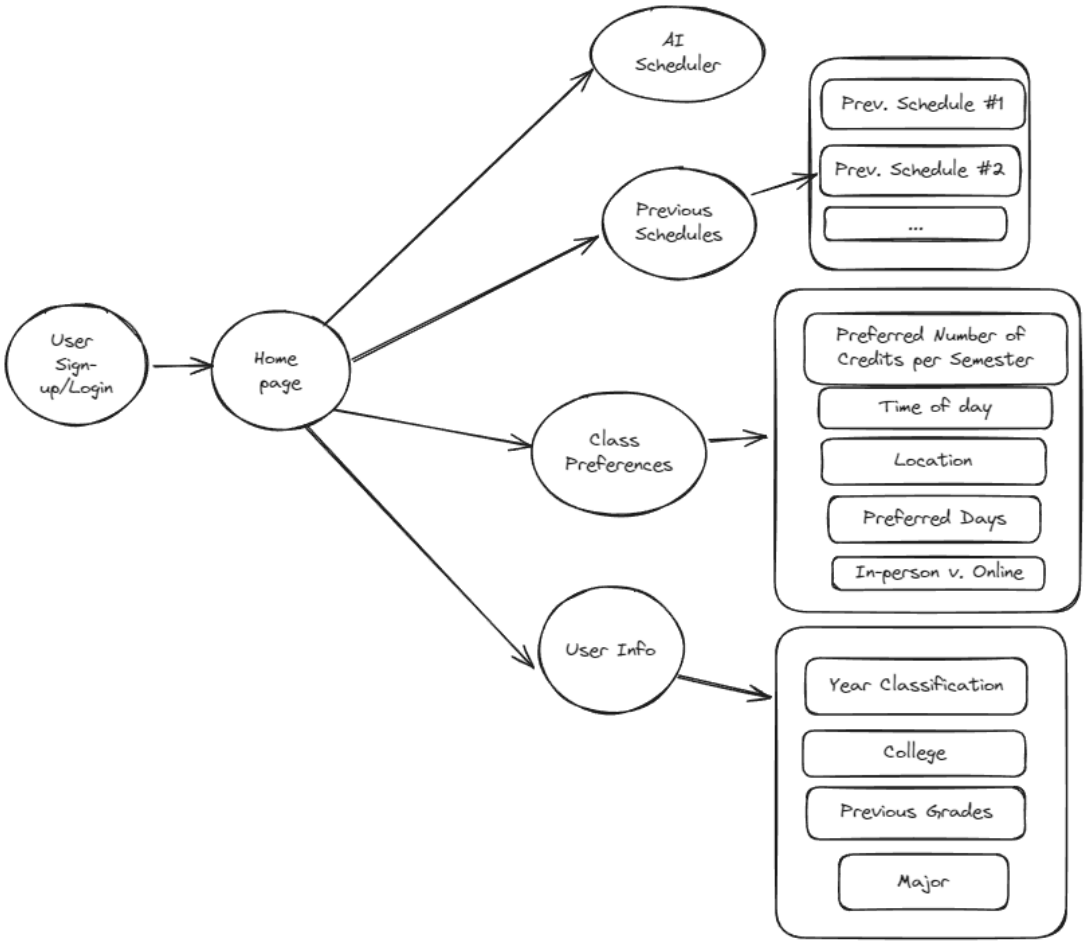

Figure 4.3: *Frontend Webpage and Contents Overview.*

In the figure above, the general webpage navigation is indicated. Circles with labels represent webpages: Login/sign-up, home, user information, class preferences, previous schedules, and the AI Scheduler page. In the rounded rectangles, the various information accessible by each webpage is indicated.

Backend Overview

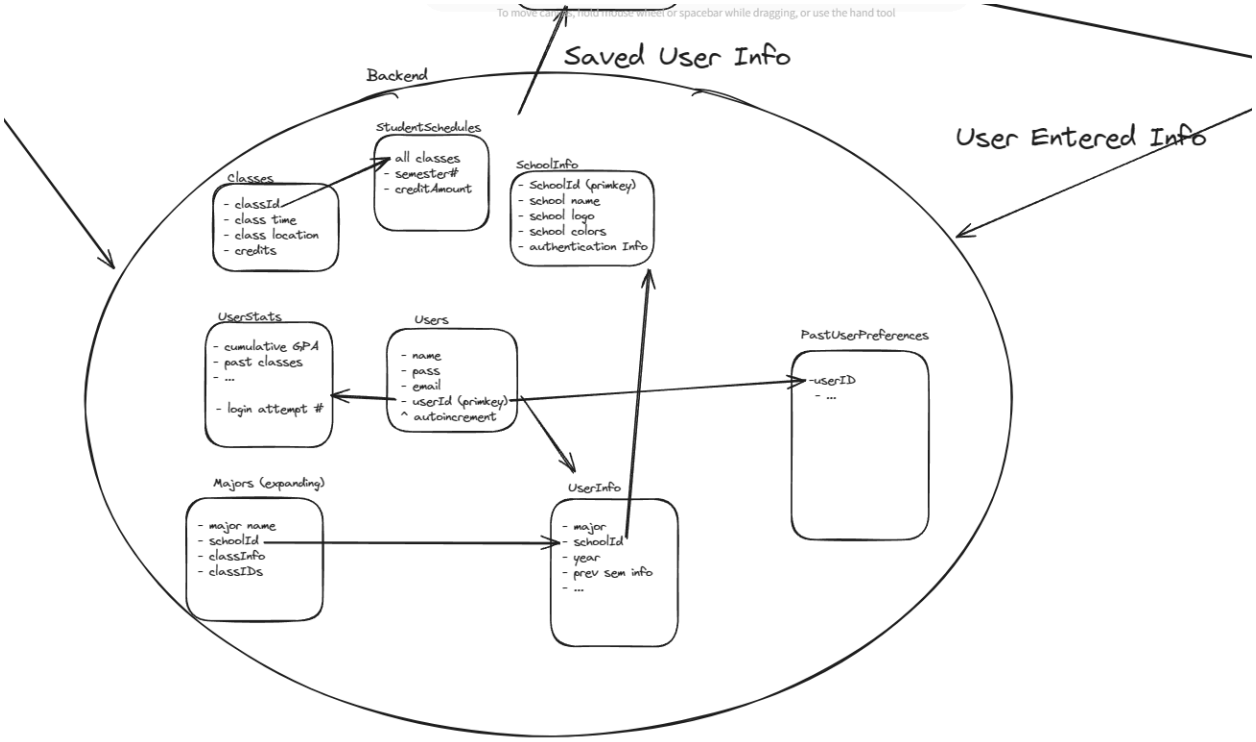

Figure 4.4: *Backend Overview (PostgreSQL Tables).*

For our backend infrastructure, we have opted to utilize Terraform scripts to streamline the provisioning and management of our PostgreSQL tables. These tables will be the backbone for storing correlated student information, which is essential for user registration within our application. In addition to PostgreSQL, we'll leverage the power of OpenAI's API to enhance functionalities within our platform, enabling advanced features and intelligent interactions. To accommodate the storage of multimedia assets, such as images, we will employ Amazon S3, providing scalable and durable object storage capabilities. Complementing these services, we'll also incorporate Amazon EC2 instances to support the computational requirements of our application, ensuring reliable and efficient performance across all aspects of our backend architecture. Through this strategic selection of technologies and services, we aim to build a robust and scalable backend infrastructure that forms the foundation for a seamless and feature-rich user experience.

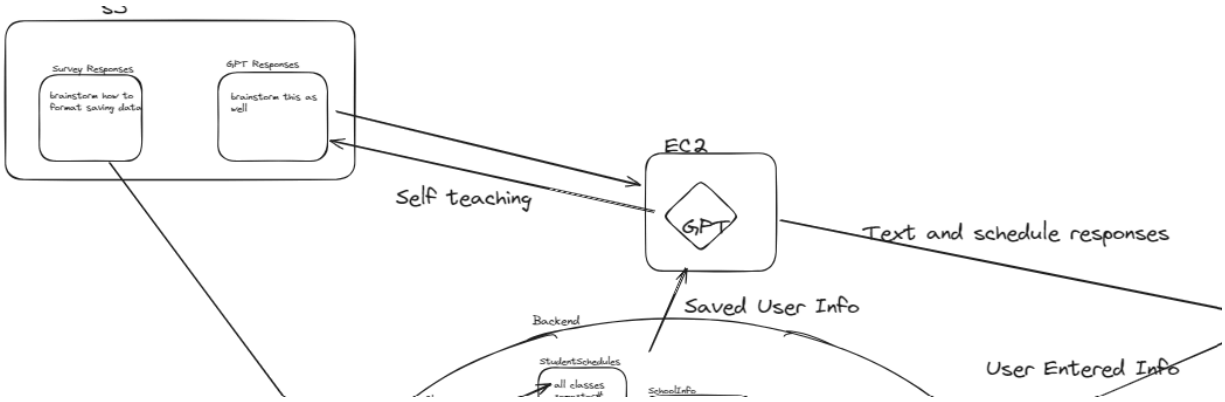

Figure 4.5: *Backend Overview (implementation of EC2, OpenAI API and S3).*

Figure 4.4 presents our initial implementation showcasing the connectivity between OpenAI, Amazon S3, and EC2 instances within our backend architecture. This diagram illustrates the fundamental setup for our backend tables, where PostgreSQL serves as the primary database for storing student information essential for user registration. OpenAI's API integration enriches our application's capabilities, providing advanced functionalities and intelligent interactions for enhanced user experiences. Additionally, Amazon S3 facilitates seamless storage of multimedia assets, such as images, ensuring scalability and durability. Complementing these services, EC2 instances are strategically deployed to meet computational demands, ensuring optimal performance and reliability across our backend infrastructure.

4.3.3 Functionality

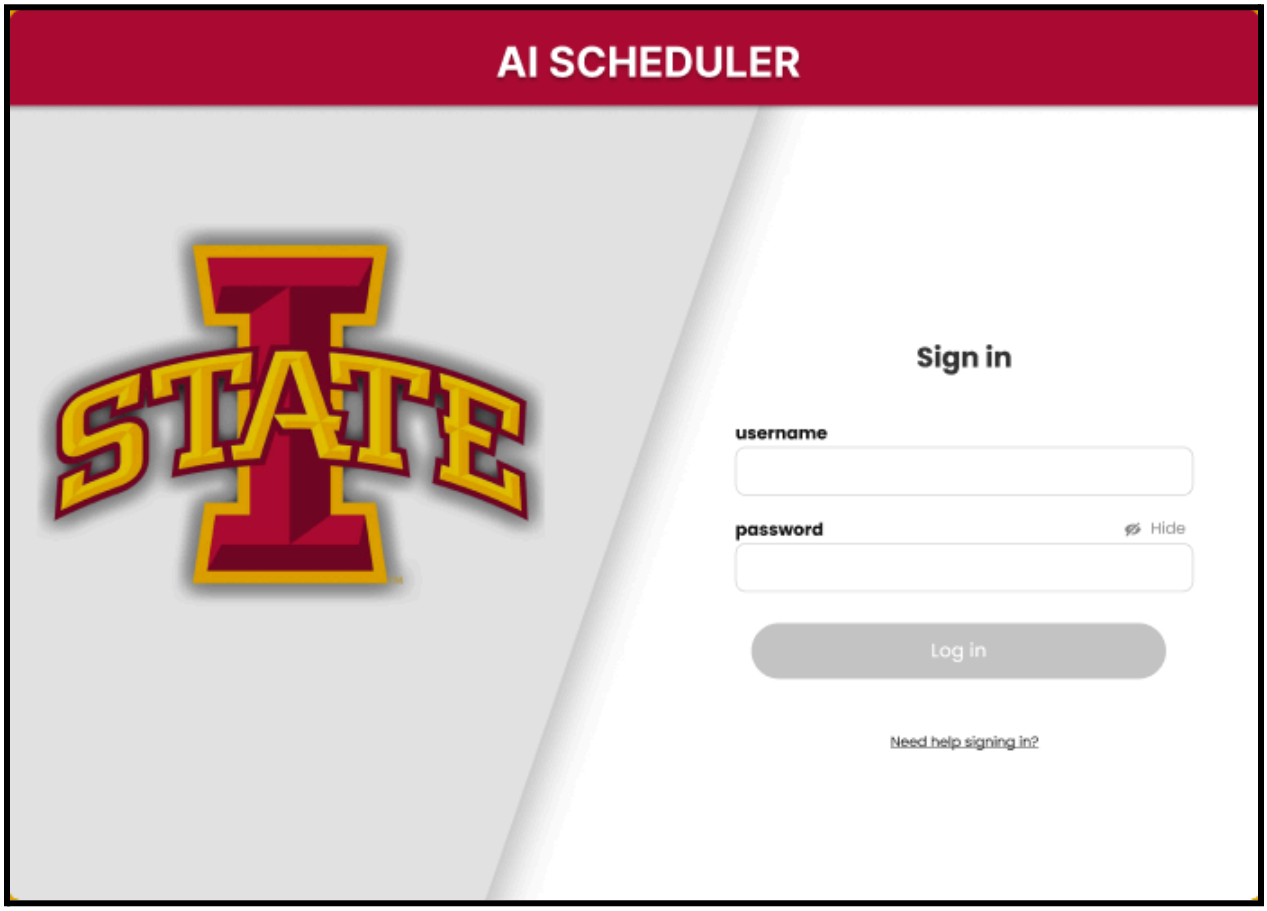

Figure 5.0: *AI Scheduler Sign-In page.*

Our design is intended to be operated by users on the frontend side, via web application access. Users who access our website will be prompted to log in or sign up. Once the user is signed in, they'll be presented with a homepage.

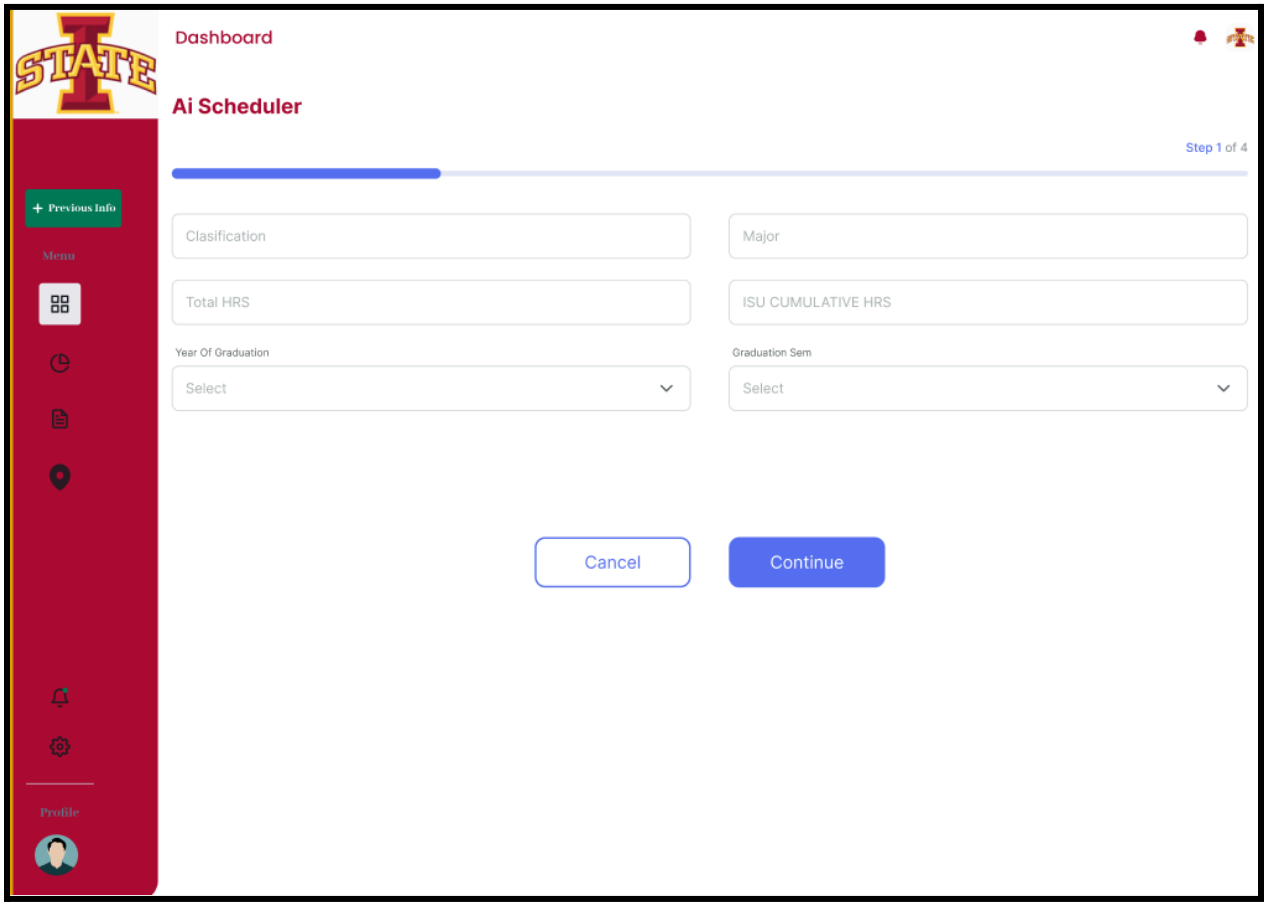

Figure 5.1: *AI Scheduler page.*

If the user still needs to answer the class preferences and user information prompts, they will be prompted to provide their information. This is to better clarify to the AI where to start looking, which will cause it to give even more accurate information.

After entering the user information, they will be brought to the pages that ask questions about their preferences. The questions will look similar, albeit in the last picture's format. After answering all the questions, the AI will start creating the schedule, which might take a bit. When created, the user will be prompted to a different page showing the results. This results page shows the schedule that the AI came up with. At this point, the user can save the schedule, which will input it into Workday, or tweak it a little if the results aren't quite what they wanted.

4.3.4 Areas of Concern & Development

Our proposed solution will satisfy the requirements: to create an easy, effective, and reliable schedule-generation experience for users. In the scope of our project, A.I. Schedule Companion is inherently designed to cater to users preferences, and that focus is reflected not only in the class preferences for schedule generation, but also when designing user interfaces; this application is intended to be for Iowa State students and advisors by Iowa State students. Given our background in the class scheduling process, we are able to use this experience to help make the schedule creation process more efficient and refined, reducing the manual effort required by users.

One of the primary concerns as of now is utilizing the Iowa State GitLab for project efforts. Since this project is a web application, we must host the site so that it is accessible on the internet. In order to have the correct permissions for hosting via Amazon Web Services (AWS), our team has had to create a separate GitHub for development efforts. The issue with this stems from ETG funding being distributed to the Iowa State-affilied GitLab, and that GitLab account is unable to provide us with the permissions required for hosting. In order to mediate this issue, our team has been coordinating with ETG to find a work around so that our site may be hosted and also receive proper funding.

4.3 Technology Considerations

4.3.1 Frontend

- **Next.JS**
    - ***Strengths:*** Next.js offers server-side rendering, which is excellent for SEO and initial page load performance. It supports static site generation, dynamic routing, and has a great developer experience with hot reloading.
    - **Weaknesses:** Next.js applications can be heavier than single-page applications due to server-side rendering and the initial download size. It also locks you into its ecosystem, which can limit flexibility in some scenarios.
    - ***Trade-offs:*** Choosing Next.js means optimizing for performance and SEO at the potential cost of flexibility regarding infrastructure and deployment options.
    - ***Alternatives:*** React with Create React App for more flexibility, Gatsby for static site generation focused projects.
- **Material-UI (MUI)**
    - ***Strengths:*** MUI provides a robust set of React components that follow Material Design principles, allowing for the quick development of attractive, consistent UIs.
    - ***Weaknesses:*** The comprehensive nature of MUI can lead to bloated bundle sizes if not properly tree-shaken. Some customization may require overriding styles, which can get complex.
    - ***Trade-offs:*** MUI accelerates development with a ready-made component library at the expense of potentially larger bundle sizes and less customization flexibility.
    - ***Alternatives:*** Tailwind CSS for utility-first CSS, allowing for more custom design with fewer predefined components, and Bootstrap for a different set of design conventions.

4.3.2 Backend

- **PostgreSQL**
    - ***Strengths:*** PostgreSQL is an advanced, open-source relational database with strong ACID compliance, extensive features like complex queries, JSON support, and reliability.
    - ***Weaknesses:*** It can have a steeper learning curve for unfamiliar SQL users. Management and scaling can become complex with large datasets.
    - ***Trade-offs****:* Opting for PostgreSQL means gaining a powerful, reliable database system at the cost of potentially complex management and optimization requirements.
    - ***Alternatives:*** MySQL for a more widely used SQL option with a larger community, MongoDB for a NoSQL option that might scale more easily with unstructured data.
- **Terraform**
    - ***Strengths:*** Terraform allows you to define infrastructure as code, making it easy to create, manage, and update infrastructure resources across multiple service providers with a consistent workflow.
    - ***Weaknesses:*** The learning curve can be steep. Debugging errors in Terraform scripts can be challenging due to the abstraction layer over cloud resources.
    - ***Trade-offs:*** Terraform provides infrastructure management consistency and efficiency at the cost of initial complexity and learning.
    - ***Alternatives:*** AWS CloudFormation for closer integration with AWS services, Ansible for more general IT automation beyond infrastructure as code.

4.3.3 Cloud & Hosting

- **AWS EC2**
    - ***Strengths:*** EC2 provides scalable computing capacity in the AWS cloud, allowing you to develop and deploy applications rapidly without hardware constraints.
    - ***Weaknesses:*** Cost management can be complex, and performance optimization requires understanding the right instance types for your workload.
    - ***Trade-offs:*** EC2 offers flexibility and scalability for your computing needs at the potential cost of complex cost and performance management.

    - ***Alternatives:*** Google Cloud Compute Engine for a different cloud provider option, AWS Lambda for serverless computing that can scale automatically.
- **Planned Cloud Database Storages (MongoDB Atlas, RDS, DynamoDB)**
    - ***Strengths:*** MongoDB Atlas offers flexibility with unstructured data. RDS provides ease of use with relational databases. DynamoDB's gives seamless scalability and performance.
    - Each of these databases offers unique strengths, and the choice would depend on specific use cases, data structures, and scalability needs.

4.3.4 AI Integration

- **OpenAI API**
    - ***Strengths:*** Provides access to powerful AI models for various tasks, including text generation, analysis, and more, allowing for innovative features in your application.
    - ***Weaknesses:*** Dependency on an external service for core functionality can introduce cost, latency, and availability risks.
    - ***Trade-offs:*** Leveraging the OpenAI API enables cutting-edge AI features at the expense of increased operational complexity and reliance on a third-party service.
    - ***Alternatives:*** Developing in-house AI capabilities (requires significant investment in resources and expertise), using other AI APIs like Google Cloud AI for different integration options and capabilities.

4.4 Design Analysis

Thus far, the design of our project has been working well for our team. We have been building out each core section of our application independently. Our front end, back end, and AI Service are not functioning in tandem. This will likely be one of the next steps in our project's development, as the front end has been looking for the ability to make REST API calls for a little while now. Similarly, the AI Service does not have a path forward that does not depend on the actions of a user and the tools provided to them. The AI Service can have basic interactions with trained content given a POST request; however, having a selected training set, having user-submitted documents, etc., will require the additional backend infrastructure. Thus, arguably, our backend is the most crucial part of our design that needs development. Until then, we cannot be sure if all systems and their current implementations are viable for the final product.

5. Testing

Testing is integral to ensuring the reliability, functionality, and security of our web app designed to assist Iowa State University students in building their class schedules and more. Our testing plan is tailored specifically to our project, addressing its unique requirements, interfaces, and potential challenges.

5. Unit Testing

Unit testing focuses on testing individual components or units of our system. In our case, units include various modules such as the chat interface, schedule generation algorithm, user authentication, and database operations. Each unit will be tested in isolation to ensure its functionality meets the specified requirements. We will employ testing frameworks like Jest for JavaScript-based components.

5.2 Interface Testing

Interface testing involves testing the interactions between different units or components of our system. For example, we will test the integration between the chat interface and the schedule generation algorithm to ensure smooth communication and data exchange. Tools like Selenium WebDriver will be used for automated interface testing, simulating user interactions and validating responses.

5.3 Integration Testing

Integration testing focuses on verifying the interactions and data flow between interconnected modules or subsystems. Critical integration paths in our design include the communication between the front-end interface, back-end server, and database. We will conduct integration tests to validate the functionality of these paths, ensuring seamless operation of the entire system. Postman will be used for API testing, while Docker Compose will facilitate testing in containerized environments.

5.4 System Testing

System testing involves testing the system as a whole to validate its compliance with the specified requirements. This includes executing a comprehensive set of unit tests, interface tests, and integration tests. Our system testing strategy will be closely tied to the functional and non-functional requirements outlined in the project scope. Continuous integration and

deployment (CI/CD) pipelines, such as GitLab CI, will automate the execution of tests and ensure consistent performance across different environments.

5.5 Regression Testing

Regression testing ensures that new additions or modifications do not introduce defects or regressions in existing functionality. We will maintain a suite of regression tests covering critical features such as class schedule generation, user authentication, and data persistence. These tests will be driven by requirements and executed automatically as part of our CI/CD pipeline. Tools like Git version control and GitHub Actions will aid in managing and running regression tests efficiently.

5.6 Acceptance Testing

Acceptance testing involves validating whether the design requirements, both functional and non-functional, are met from the end user's perspective. We will collaborate closely with our client, involving them in acceptance testing to gather feedback and ensure alignment with their expectations. User acceptance testing (UAT) sessions will be conducted, where students interact with the system to evaluate its usability, performance, and overall satisfaction.

5.7 Security Testing

While security testing is not currently applicable to our project, it will become crucial in future phases when handling sensitive information and deploying the application on cloud platforms like Amazon Web Services (AWS). During the implementation phase, we will perform security assessments and penetration testing to identify and mitigate potential vulnerabilities. Tools like OWASP ZAP and Burp Suite will be utilized for security testing, ensuring user data's confidentiality, integrity, and availability.

5.8 Results

Once testing is complete, the results will demonstrate how well our design aligns with the specified requirements and standards. Through a meticulous process encompassing unit, interface, integration, system, regression, and acceptance testing, we aim to validate our web application's functionality, performance, and security.

To provide concrete evidence of our testing efforts, we will employ various metrics and techniques, including:

1) Code Coverage Analysis: We will measure the percentage of code executed during testing to ensure that all parts of the codebase are adequately exercised- tools like Istanbul for JavaScript or JaCoCo.
2) Test Case Documentation: Each test case, including its purpose, inputs, expected outcomes, and actual results, will be meticulously documented. This documentation will serve as a reference for future testing cycles and provide insight into the thoroughness of our testing process.
3) Test Automation Reports: For automated tests, we will generate detailed reports outlining test execution results, including pass/fail statuses, execution times, and any encountered errors or exceptions. These reports will facilitate quick identification of issues and enable timely remediation.
4) Performance Metrics: In addition to functional testing, we will assess the performance of our web application under various load conditions. Metrics such as response times, throughput, and resource utilization will be monitored and analyzed to ensure optimal performance and scalability.
5) Security Assessment Findings: If applicable, security assessment and penetration testing results will be documented, highlighting any identified vulnerabilities or weaknesses in our application's security posture. Remediation plans will be developed to address these findings and enhance the application's resilience against cyber threats. This section primarily relates to Amazon Web Services and hosting, but this issue is not as problematic due to the low amount of personal information stored within our databases.

We aim to provide transparent and verifiable evidence of our testing functionality by leveraging these techniques and metrics. This comprehensive approach will instill confidence in the reliability and robustness of our web application, ultimately benefiting the Iowa State University student community.

6. Implementation

As we move from development to deployment, our attention turns to improving, testing, and finally making the Iowa State A.I. Schedule Companion available to all students. This important

phase includes many important jobs, such as making the design more user-centered, testing it thoroughly, and making the big switch to public hosting. As we bring together new technologies and user needs, our goal is not only to provide a smooth schedule solution but also to build a flexible platform that gives students the tools they need to manage their schoolwork.

6.1 Hosting and Deployment:

- Procure hosting services from a reliable provider capable of handling anticipated user traffic.
- Configure servers and databases for deployment, ensuring scalability and reliability.
- Deploy the updated version of the Iowa State A.I. Schedule Companion to the public hosting environment.

6.2 Testing and Quality Assurance:

- Organize testing groups consisting of students from various academic backgrounds to gather diverse feedback.
- Conduct iterative testing cycles to identify and address any remaining issues or bugs.
- Fine-tune the application based on feedback from testing groups to ensure usability and effectiveness.

6.3 Security and Privacy Enhancements:

- Conduct a security audit to identify potential vulnerabilities in the application.
- Implement encryption mechanisms to secure user data stored in the database.
- Update privacy policies and user agreements to comply with data protection regulations.

6.4 Design Refinement:

- Review user feedback from the test run and identify areas for improvement in the UI/UX design.
- Refine web page layouts, color schemes, and navigation to enhance user experience.
- Update design documentation to reflect any changes.

6.5 Backend Enhancements:

- Address any performance issues or bugs identified during the test run.
- Optimize database queries and backend code for improved efficiency.

- Implement additional features requested by users or stakeholders, such as class search functionality or schedule sharing options.

6.6 Public Release:

- Announce the public availability of the application to the student body through various channels, such as university newsletters, social media, and campus events.
- Provide instructions for accessing and using the application, including account creation and schedule generation.
- Monitor user adoption and address any issues or concerns raised by early adopters.

6.7 Post-Release Support and Monitoring:

- Offer ongoing technical support to users and promptly address any reported issues or inquiries.
- Monitor application performance and server uptime to ensure a seamless user experience.
- Gather feedback from users through surveys or feedback forms to inform future updates and enhancements.

7. Professional Responsibility

7.1 Areas of Responsibility

Area of Professional Responsibility	NSPE Code of Ethics	ACM Code of Ethics
1. Public Safety, Health, and Welfare	Engineers shall hold paramount the safety, health, and welfare of the public.	1.1 Contribute to society and to human well-being, acknowledging that all people are stakeholders in computing.
2. Competence and Accountability	Engineers shall perform services only in areas of their competence.	2.2 Maintain high standards of professional competence, conduct, and ethical practice.
3. Honesty, Integrity, Fairness	Engineers shall act with honesty, integrity, and fairness.	1.3 Be honest and trustworthy.
4. Conflict of Interest	Engineers shall act in professional matters for each employer or client as faithful agents or trustees, and shall avoid conflicts of interest.	1.3 Avoid deception and be honest and trustworthy.
5. Confidentiality and Proprietary Information	Engineers shall not disclose, without consent, confidential information concerning the business affairs or technical processes of any present or former client or employer, or public body on which they serve.	1.8 Honor confidentiality and privacy.
6. Competence and Quality of Work	Engineers shall strive to serve the public interest and to improve their competence and the quality of their work.	2.2 Maintain high standards of professional competence, conduct, and ethical practice.

7. Professional Development	Engineers shall continue their professional development throughout their careers and should keep current in their specialty fields by engaging in professional practice, participating in continuing education courses, reading in technical literature, and attending professional meetings and seminars.	2.2 Maintain high standards of professional competence, conduct, and ethical practice.

Table 4.0: *Area of Responsibility.*

The ACM Code emphasizes contributing to society and human well-being, recognizing that all people are stakeholders in computing, while the NSPE Code focuses explicitly on the safety, health, and welfare of the public.

Both codes stress competence, but the ACM Code also includes conduct and ethical practice, whereas the NSPE Code focuses on performing services only in competence areas.

Both codes emphasize honesty and integrity, but the NSPE Code also includes fairness, while the ACM Code focuses on trustworthiness.

The ACM Code emphasizes avoiding deception and being honest and trustworthy, while the NSPE Code specifically mentions avoiding conflicts of interest and acting as faithful agents or trustees.

Both codes address confidentiality, but the NSPE Code also mentions proprietary information and consent, while the ACM Code focuses on honoring privacy.

The ACM Code stresses maintaining high standards of competence, conduct, and ethical practice, while the NSPE Code emphasizes serving the public interest and improving competence and quality of work.

Both codes emphasize continuing professional development, but the NSPE Code provides specific examples, such as engaging in professional practice, participating in continuing

education courses, reading technical literature, and attending professional meetings and seminars.

7.2 Project-Specific Professional Responsibility Areas

1) Public Safety, Health, and Welfare

- ***Application:*** The AI Class Scheduler should ensure fair and non-discriminatory schedules for all students, contributing to their well-being.
- ***Performance:*** Medium - The team should develop an algorithm prioritizing fairness and equity.

2) Competence and Accountability

- ***Application:*** As students, the team should strive to develop their skills and knowledge in AI, software development, and ethical practices throughout the project.
- ***Performance:*** High - The team should actively seek learning opportunities and take responsibility for the project's outcomes.

3) Honesty, Integrity, Fairness

- ***Application:*** The team should be transparent about the capabilities and limitations of the AI Class Scheduler and ensure that it treats all students fairly.
- ***Performance:*** High - The team should prioritize transparency and fairness in the development and presentation of the project.

4) Conflict of Interest

- ***Application:*** The team should disclose any potential conflicts of interest within the team or with external stakeholders.
- ***Performance:*** High - The team should maintain open communication and address any potential conflicts of interest promptly.

5) Confidentiality and Proprietary Information

- ***Application:*** The team should protect the privacy and confidentiality of student data used in the AI Class Scheduler.

- ***Performance:*** High - The team should implement appropriate security measures and adhere to data protection regulations.

6) Competence and Quality of Work

- ***Application:*** The team should continuously improve their skills and the quality of the AI Class Scheduler throughout the project.
- ***Performance:*** High - The team should set goals for improvement and regularly assess the project's quality.

7) Professional Development

- ***Application:*** The team should engage in learning opportunities related to AI, software development, and ethical practices during the project.
- ***Performance:*** High - The team should actively seek resources and participate in relevant training, seminars, or courses.

7.3 Most Applicable Professional Responsibility Area

Public Safety, Health, and Welfare

The most applicable professional responsibility area for the AI Class Scheduler project is "Public Safety, Health, and Welfare." As the project directly impacts the well-being of students by generating their class schedules, it is crucial to ensure that the AI algorithm is designed to be fair, non-discriminatory, and equitable.

8. Closing Material

8.1 Discussion

The main goal of our project is to create an application that utilizes user-input to create a semester-based schedule that is fine-tuned to the user's preferences. The requirements for a complete project are that we have a web application that uses AI to create a schedule automatically, and that these schedules can be easily created and stored by students.

Given that our project is not complete yet, the requirements have yet to be met but that does mean they will *eventually* be met. That said, we do still have the main goal in mind of making a schedule generation web application oriented toward students. So long as we prioritize (a.) the automatic creation of a schedule and (b.) the fact that students are our main users, then the requirements for our project will be met.

8.2 Conclusion

Current Solution Evaluation

At this point in the development process, our team has created a web application prototype, implemented backend infrastructure for hosting, database storage, and artificial intelligence access, and began training and testing with ChatGPT-4.0 artificial intelligence. In our current state, most developmental testing has been conducted locally, and has not yet utilized the cloud.

Milestone Evaluation

Iowa State A.I. Schedule Companion v1.0 Completion - Deadline: March 29, 2024

Regarding this milestone, our team has completed this milestone. Our team has successfully met the criteria listed, namely: majority of the anticipated webpages have been initialized (though the designs may not be finalized,) our web application can be compiled locally and display webpages without errors, and our backend infrastructure has been setup and is accessible.

Conduct User Test Run - Deadline: April 5, 2024

Overall, this milestone is intended to act as the first significant trial of our application's primary functionalities, namely that our web application can utilize artificial intelligence to generate a class schedule.

In our current state, our application has not met this milestone as integration has not yet been fully completed. The various components of our application (frontend user interfaces, backend databases, and artificial intelligence) work independently, but integrated testing has not been conducted.

One major influence that prevented this milestone from being completed successfully was deployment and integration set backs when trying to use AWS in GitLab. This was an issue we faced initially, and were able to mitigate by using a student-owned GitHub repository, but in order to use ETG-provided funding for artificial intelligence training, hosting, etc., we needed to utilize the designated senior design GitLab. However, using this GitLab led to difficult issues regarding AWS deployment.

An additional factor on this milestone was the impact of other coursework for all team members; as the semester progressed, time that could be designated toward web development and database structuring fluctuated, reducing the overall amount of time being devoted the integration effort as a whole.

Please refer to section 3.3 Project Proposed Milestones and Evaluation Criteria for more information about the above milestones.

The Future of Iowa State A.I. Schedule Companion

Although our second milestone was not achieved this semester, our team has put in a great amount of effort and the current state of our application is not far behind. Our biggest pushes coming forward will be to finish integration for all components of our web application, and try complete a user test run prior to the end of the semester. Luckily for this project, we still have more time to put into development, as our application's objectives are still within reach.

8.3 References

[1] Vercel, “Docs | Next.js,” nextjs.org. https://nextjs.org/docs

[2] MUI, “Material UI Components,” MUI. https://mui.com/material-ui/all-components/

[3] “OpenAI API,” *platform.openai.com*. https://platform.openai.com/docs/api-reference

[4] “Course Planner: Iowa State University,” *classes.iastate.edu*. https://classes.iastate.edu/planner (accessed Mar. 19, 2024).

[5] I. S. U. Office of the Registrar, “Workday FAQ | The Office of the Registrar | Iowa State University,” *www.registrar.iastate.edu*. https://www.registrar.iastate.edu/resources/workday-faq (accessed Apr. 26, 2024).

[6] Iowa State University Workday. *Workday, Inc.,* 2024. [Online].

[7] LangChain, “Quickstart,” js.langchain.com. https://js.langchain.com/docs/get_started/quickstart/

8.4 Appendices

For 4.2.3 Functionality, there were two additional figures which were created but not used.

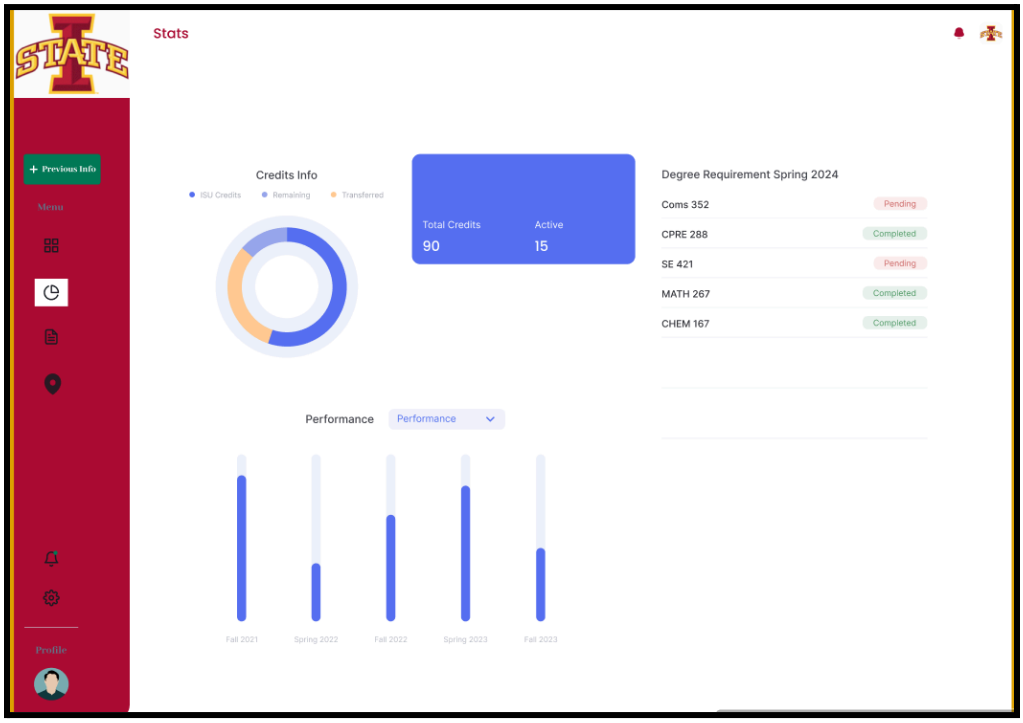

Figure 6.0: *Hypothetical User Statistics Webpage Design.*

This was a hypothesized page from our website, it would display the stats of the student. This would allow them to much more easily see their own information, such as GPA, current credits, needed credits, and current classes. The reason this is simpler is because with AccessPlus, one had to undergo an audit and scroll through to find the wanted information. But with this tab of the website, all the wanted (or at least, what *we* thought they wanted) information would be displayed.

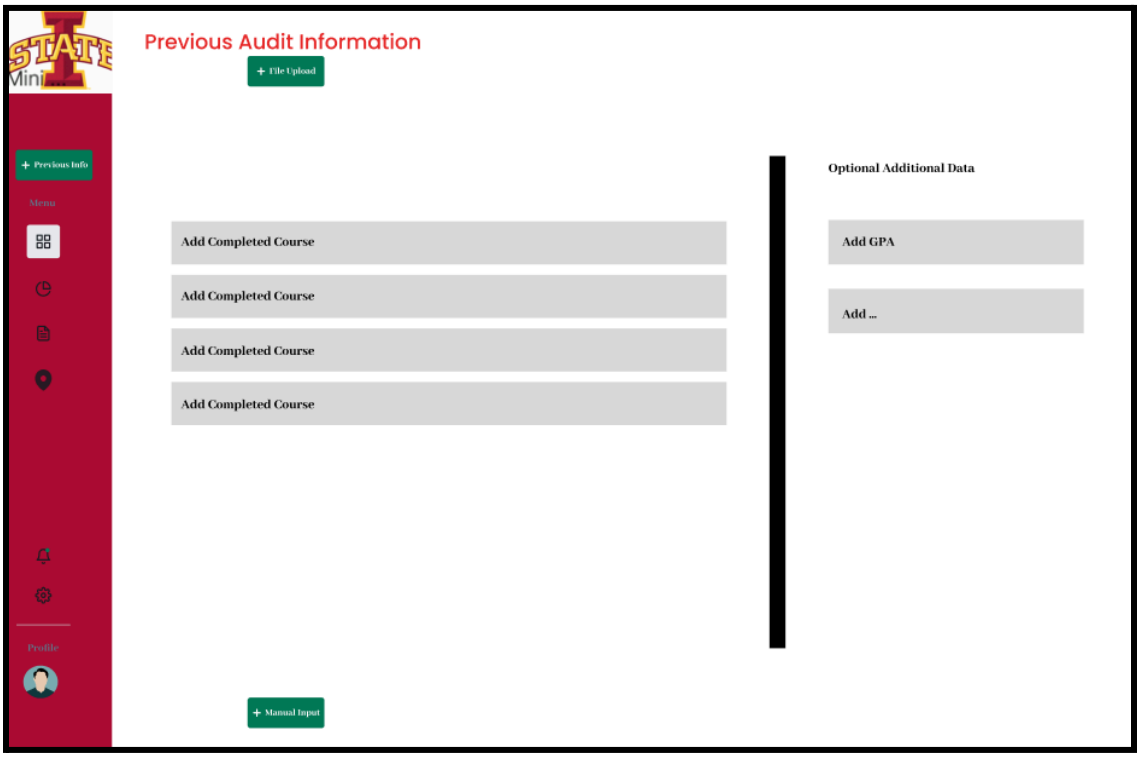

Figure 6.1: *Hypothetical Audit Information Webpage.*

Another hypothesized page of the website, this one had all the audit information. While the stats page had specific information, this had all of it to access. As well, the thought that one could manually input their own audit was implemented here.

Although these two pages are less guaranteed than those already displayed in Functionality, the Stats page seems like a great idea and would definitely make it more convenient for students if they wanted their own information. As with the audit, there would be some struggle in the manual input of audit info, but access to audits would still be likely.

9. Team

9.1 Team Members

Team Members:

- *Koby Fowler*
- *Chandrashekar Tirunagiri*
- *Raghuram Guddati*
- *Jacob Paustian*
- *Christian Deam*
- *Anna Huggins*

9.2 Required Skill Sets

Frontend Skills

- NextJS abstraction and programming abilities
- General user interface design knowledge (navigation, page layouts, etc.)
- Integration to backend database capabilities

Backend Skills

- **Database**
    - Understanding PostgreSQL database tables and linking
    - Integration to frontend capabilities
- **Artificial Intelligence**
    - Understanding of OpenAI API integration, training, and usage
- **Hosting Skills**
    - Knowledge of Amazon Web Services (AWS) deployment setup and website hosting

9.3 Skill Sets Covered by the Team

Frontend Skills

- Covered by:
    - Chandrashekar Tirunagiri
    - Anna Huggins

Backend Skills

- Covered by:
    - *Raghuram Guddati*
    - *Jacob Paustian*
    - *Koby Fowler*
    - *Christian Deam*

9.4 Project Management Style Adopted by the Team

For this project, our team is utilizing a Waterfall project management style. Please refer to section 3.1.1 Project Management, for additional information on this process.

9.5 Initial Project Management Roles

- *Koby Fowler* - Overall Application Leader
- *Chandrashekar Tirunagiri* - Frontend Design Leader
- *Raghuram Guddati* - Backend Design Leader
- *Jacob Paustian* - Artificial Intelligence Research Leader
- *Christian Deam* - Meeting Manager
- *Anna Huggins* - Team Manager

9.6 Team Contract

Team Procedures

Communication Preferences

Regular Team Meetings:

- **In-Person: (preferred)**
    - *Location*: Parks Library or Student Innovation Center
    - *Time*: Monday, 4:30pm
- **Virtual:**
    - *Location*: Microsoft Teams
    - *Time*: Monday, 4:30pm

**Note*: Our preferred method of team meetings is in-person, but we will allow the virtual option to give all team members some added flexibility.

General Communication:

- **Primary Communication:**
    - Microsoft Teams & Snapchat
- **Teaching Assistant Communication:**
    - Discord

General Policies

- **Decision-Making:**
    - Majority vote; group discussions for major decisions during team meetings
- **Record-Keeping:**
    - Weekly meetings as needed
        - Christian Deam (Meeting Manager) will manage the documentation and scheduling of meetings

Collaboration & Inclusion

- **Identifying & resolving collaboration or inclusion issues:**
    - During team meetings, reserve time for everyone to bring up any additional comments and concerns
    - Utilize daily communication for any questions throughout the week (i.e. Microsoft Teams & Snapchat)
    - If issues cannot be resolved within our group, escalate issues to discussion with our TA

Participation Expectations

Individual Expectations

- **Attendance, punctuality, and participation at all team meetings:**
    - All team members must at least attempt to attend meetings that pertain to them
    - i.e.) Front-end developers will attend front-end meetings, and all team members will attend weekly full-team meetings as frequently as possible

- **Level of responsibility for fulfilling team assignments, timelines, and deadlines:**
    - All team members must at least attempt all assigned tasks by the next meeting

- **Level of communication with other team members:**
    - Be in contact at least once a week
    - Respond to direct team messages as soon as possible

- **Level of commitment to team decisions and tasks:**
    - All team members should work toward base project completion by the final deadline

Leadership

Assigned Roles

Overall Application Leader: Koby Fowler
Frontend Design Leader: Chandrashekar Tirunagiri
Backend Design Leader: Raghumram Guddati
A.I. Research Leader: Jacob Paustian
Meeting Manager: Christian Deam
Team Manager: Anna Huggins

**Note*: These assigned roles do not limit team members' responsibilities but act as guidelines for primary tasks for each member.

Individual Skills & Expertise

- **Jacob Paustian**
    - For my skills and expertise, I have front/backend dev experience along with some low-level programming
- **Chandrashekar Tirunagiri**
    - Experienced in backend development:
        - Some experience from internships
    - Would like to work on frontend for new experiences
- **Raghuram Guddati**
    - Experienced in backend and frontend development:
        - Higher proficiency in backend development
- **Christian Deam**
    - Moderate backend experience and minor front-end knowledge
    - Extensive programming knowledge and experience
- **Anna Huggins**
    - Experienced in frontend design
        - Preference toward frontend development
        - Android Studio development
    - General programming experience:
        - Java, C, C#
    - No recent web design experience, but open to learning throughout the project.
- **Koby Fowler**
    - Experienced in full stack development from a job and leading a team of engineers to build an application.
        - Preference is infrastructure or backend, but can do frontend as well

Strategies for Accountability & Responsibility

- **Supporting and guiding team members' work:**
    - Weekly meetings
    - Consistent communication (weekly, daily, as needed)
    - Positive feedback

- **Recognizing the contributions of team members:**
    - Use of GitHub activity heatmap to check code contributions
    - Use of attendance log of team meetings for group attendance
    - Discussion of individual progress during team meetings

Goal-Setting, Planning, & Execution

Overall Team Goal for this Semester (Spring 2024)

Completion of a working baseline application

- Completion of an application that can assist students and academic advisors in creating schedules that best fit the students' needs. This application will use ChatGPT 4.0 API to help decide the best schedules.

Strategies for Maintaining Our Goal

- **Individual & Team Assignments:**
    - Work will be assigned based on leadership roles and individual expertise listed
    - Git Issues will be utilized to handle specific assignments
- **Strategies for Keeping on Task**
    - Weekly SCRUM-style meetings will be held to ensure everyone is completing their assigned tasks and avoid procrastination

Consequences for Not Adhering to Team Contract

Handling Infractions of Team Contract

- **Initial Infractions:**
    - Discuss contract infractions in team meetings with all group members
    - Goal: determine if there is additional assistance that can be provided to prevent future infractions

- **Continual Infractions:**
    - Inform our TA of our team's current infractions and status, as well as any previous mitigation techniques previously used

Team Signatures

Contract Agreement

a) I participated in formulating the standards, roles, and procedures as stated in this contract.
b) I understand that I am obligated to abide by these terms and conditions.
c) I understand that if I do not abide by these terms and conditions, I will suffer the consequences as stated in this contract.

---

1) Koby Fowler	DATE: 1/29/2024
2) Chandrashekar Tirunagiri	DATE: 1/29/2024
3) Anna Huggins	DATE: 1/29/2024
4) Raghuram Guddati	DATE: 1/29/2024
5) Jacob Paustian	DATE: 1/29/2024
6) Christian Deam	DATE: 1/29/2024
