UNO Classroom Scheduler Application

Capsoft Solutions

Tyler Porter
Trevor Ahlers
Fara Abdulloev
Honora Shanno
Haresh Chandrasekar

Sponsor:

Dr. Siy

Project Report

1. Introduction 5
1.1 Motivation 5
1.2 Similar Applications 6
1.3 Theoretical Foundations 8
1.4 Software Engineering Challenges 9
1.5 Context 10
2. Requirements 13
2.1 Goals 13
2.2 Functional Requirements 14
2.2.1 BDD Scenarios 14
2.2.2 Use Cases 15
2.2.2.1 Use case 1. 15
2.2.2.2 Use case 2. 16
2.2.2.3 Use case 3. 16
2.2.2.4 Use case 4. 17
2.3 Wireframes 18
2.4 Object Model 20
2.5 Non-Functional Requirements 20
2.6 External System Interfaces 21
2.6.1 File Format Interfaces 22
2.6.1.1 Input File Requirements 22
2.6.1.2 Output File Specifications 22
2.6.2 REST API Endpoints 23
2.6.2.1 Authentication & User Management 23
2.6.2.2 Uploads & Assignment Initialization 23
2.6.2.3 Data Retrieval and Editing 24
2.6.2.4 Conflict Management 24
2.6.2.5 Assignment File State Management 24
2.7 Requirements Not Implemented 25

3. Architecture and Design 26
3.1 Logical Decomposition 26
3.2 Development 26
3.2.1 Python Backend 26
3.2.1.1 Flask 27
3.2.1.2 Python Libraries 28
3.2.1.3 CourseSection Model 28
2.3.1.4 Classroom Model 29
2.3.1.5 Conflict Model 30
2.3.1.6 AssignmentFile Model 30
2.3.1.7 Room Scoring Module 31
2.3.1.8 Assigner Module 31
2.3.1.9 Exporter Module 32
2.3.1.11 The Problem of Parsing Meeting Times with Assigned Rooms 32
3.2.2 Angular Frontend 34
3.2.2.1 Request and Data Flow 35
3.2.2.2 Angular Libraries 35
3.2.2.3 Enforcing Authentication 35
3.2.2.4 Environment Configuration 36
3.3 Data 36
3.4 Execution Flow View 37
3.5 Screenshots 39
4. Implementation 42
4.1 Design Changes 42
4.2 Deployment 43
4.2.1 Directory Structure 43
4.3 Technical Issue Encountered 44
5. Testing 45
5.1 Overall Testing Strategy 45

5.1.1 What We Planned to Use 45
5.1.2 What We Ended Up Using 46
5.2 Realistic Test Data 46
5.3 Tests Conducted 47
5.4 Tests Results 47
5.5 Tests Not Conducted 48
6. Summary 49
6.1 Outcome of Risks 49
6.2 Milestone Summary 50
6.3 Future Extensions 51
6.4 What we Learned 51
Tyler 51
Trevor 52
Fara 53
Honora 53
Haresh 54
7. Local and Global Impacts 55
8. Appendices 57
8.1 Relevant URLs 57
8.2 Setup/build/installation/deployment instructions 57
8.3 Team Weekly Meeting Notes 58
8.4 Individual Journals 70
8.4.1 Tyler Porter 70
8.4.2 Trevor Ahlers 80
8.4.3 Fara Abdulloev 84
8.4.4 Honora Shanno 88
8.4.5 Haresh Chandrasekar 99
8.5 Email Transcripts of Discussions with Clients 101
9. References 109

1. Introduction

1.1 Motivation

This web application was developed to reduce the workload of the Classroom Coordinator responsible for assigning courses to rooms in the Peter Kiewit Institute (PKI) building at the University of Nebraska at Omaha.

Each semester, the Classroom Coordinator receives a CSV file containing all course sections that need room assignments in PKI. Difficulties arise when conflicts occur and managing both scheduled and emergent conflicts has been shown to contribute to the high turnover in this role. We aimed to automate as much of this process as possible and reduce the mental load on the user who utilizes our application to produce a finalized output CSV that is either conflict-free or allows conflicts to be ignored. This solution CSV or Change Report can be provided to the registrar for approval and any emergent changes or conflicts raised from the registrar can be addressed by re-entering the solution CSV into the platform for further iteration. This should help reduce training time for classroom coordinators, help retain employees in this position that would otherwise quit due to task difficulty and speed up the process of assigning classes to their respective rooms at the PKI building.

The final product is a responsive, browser-based web application accessible from any computer with a stable internet connection. While the platform is not currently optimized for mobile devices and may experience display issues on screens with very low resolutions or uncommon aspect ratios, it has been tested across a range of modern desktop and laptop environments. Although server access restrictions prevented us from fully integrating the system with UNO's internal intranet, the application has been designed with future adaptability in mind and can be configured for deployment within the university's internal network infrastructure as needed.

User access is protected through a secure authentication process that begins on the landing page, where individuals can either log in with existing credentials or register a new account. Once authenticated, users are directed to the main dashboard, where they are prompted to upload the CSV file containing that term's course data. The term is defined by a combination of semester and year (e.g., Fall 2025), and this information is used to scope the scheduling operation.

Within the platform, users can choose from multiple automated assignment methodologies to best align with their specific needs. Upon upload, the system immediately analyzes the course data for potential scheduling conflicts—such as double-booked rooms, overlapping class times, or capacity mismatches—and presents any issues to the user in a clearly itemized format. Users can then resolve these conflicts using the platform's built-in tools or allow them to persist when preparing the output, depending on the context and urgency.

In addition to conflict resolution, we have implemented several quality-of-life features to enhance the user experience and improve operational efficiency. These include real-time search functionality for locating classes quickly, customizable table columns for a personalized data view, and in-app editing capabilities that enable users to adjust assignments directly within the interface. To maintain integrity and prevent miscommunication, the ability to create entirely new classes within the platform is intentionally restricted. This measure ensures that all structural scheduling changes remain coordinated through the registrar's office.

1.2 Similar Applications

Our project focuses on developing a classroom scheduling web platform that automates the assignment of course sections to classrooms based on constraints such as capacity, equipment requirements, meeting patterns, and instructor preferences. The goal is to streamline the scheduling process for university coordinators by automating room assignments, detecting conflicts, and enabling manual overrides when necessary.

Several existing applications offer functionality that overlaps with or complements key aspects of our classroom scheduling platform.

- **Google Calendar** is widely used for time and event scheduling. It supports interactive calendar views, drag-and-drop scheduling, and color-coded events that help users visually interpret large volumes of overlapping data (Google, 2025). Our application uses a similar interface design in the calendar view, enabling coordinators to view, identify, and resolve classroom conflicts effectively.

- **Salesforce** is a powerful enterprise-level productivity platform. While its core purpose is customer relationship management (CRM), it excels in managing large datasets, supporting robust user authentication, and offering highly customizable dashboards (Salesforce, 2025). We drew inspiration from its ability to filter, categorize, and sort data for our section views and conflict resolution tools.

- **Photopea** is a web-based photo editing tool that includes non-linear editing features, such as undo/redo support and session-based state management (Photopea, 2025). Though its purpose differs, our application parallels this functionality through undo/redo capabilities for classroom assignments and schedule changes, enhancing user control and reducing risk during scheduling.

- **Skedda** is a space management and booking platform used for scheduling shared spaces such as meeting rooms and classrooms. It integrates seamlessly with tools like Microsoft 365 and allows for visualization of bookings via calendar views, similar to our UI design (Skedda, 2025). However, Skedda lacks the domain-specific logic we've implemented, such as cross-listing recognition and automatic assignment of classes based on department usage history.

Feature	Google Calendar	Salesforce	Photopea	Skedda	Our Scheduler App
Interactive Calendar View	✅	❌	❌	✅	✅
Automatic Scheduling with Constraints	❌	❌	❌	❌	✅
Cross-listing Handling	❌	❌	❌	❌	✅
Custom Filters and Sorting	❌	✅	❌	Limited	✅
CSV Import/Export	✅ (limited)	✅	✅	✅	✅
Domain-Specific Room	❌	❌	❌	❌	✅

Assignment Logic					

Figure 1-0: Above is a comparison of similar applications and their key features compared to our custom scheduling platform.

1.3 Theoretical Foundations

Our project is grounded in several key theoretical and architectural principles that guided the system's design, development, and testing:

- **Model-View-Controller (MVC) Architecture:**
  The application is built using the MVC design pattern, promoting separation of concerns by dividing the system into three distinct layers—Model (data and business logic), View (user interface), and Controller (request handling and coordination). This structure improves maintainability, scalability, and parallel development.
- **Object-Oriented Programming (OOP):**
  Core components such as CourseSection, Classroom, and Conflict are implemented as object-oriented classes, enabling encapsulation, reusability, and clear abstraction of system entities and behaviors.
- **Relational Database for Data Persistence:**
  The system utilizes a relational database managed through Amazon RDS to persistently store user accounts, session data, and classroom assignment files. This ensures data integrity, consistency, and long-term accessibility.
- **Static Frontend Hosting on AWS S3:**
  The frontend is built with Angular and hosted as a static site via Amazon S3. The use of a reliable cloud-based storage and delivery system guarantees high availability, low latency, and seamless integration with custom domain routing via AWS Route 53.
- **Automated Testing with Pytest:**
  Backend logic and object model interactions are validated using pytest, an industry-standard testing framework. This ensures accuracy in data handling, route responses, and assignment logic through continuous verification during development.
- **Prototyping and Agile Development Methodology:**
  The team adopted an Agile mindset, using iterative prototyping to develop and refine features based on stakeholder feedback. Regular team standups, sponsor

meetings, and short development cycles allowed for rapid adaptation to changing requirements.

1.4 Software Engineering Challenges

We faced many Computer science challenges while working on this project. To begin with our group was trying to create a web application using a front-end framework no one in the group has experience working with which could cause issues if the group is unable to find solutions to problems that arise when working with this new framework. Another challenge was managing the complexity of the system without knowing specific client requirements because there was no employed Classroom Coordinator at project inception. This became clearer once we begin meeting with the sponsor, but even then, we were missing key information such as an example spreadsheet to represent data as it would be when it was provided to our platform, rather than the spreadsheets we had that were already finalized. At the time, it posed an issue for development by delaying decisions in planning and the steps we could take in our implementations. Our team also faced the challenge of finding the right vendor to work with to host our web platform. Our sponsor suggested deploying with Google Cloud but no one on the team had prior experience with this. We eventually opted for Amazon Web Services due to its scalability, prior experience our team has had with it, and it's flexbility with free tier options. Another major consideration was how we would handle data consistency with multiple users interacting with one database. We were concerned about addressing race conditions or invalidation of sessions due to conflicts with a single relational database. Some of the greatest difficulty was found when analyzing the input spreadsheet sample data. Our analysis revealed there was an overwhelming amount of inconsistency in how the data was formatted. These inconsistencies included how a field was utilized by various parties, irrelevant data in fields that were otherwise consistent, different standards for how the same information was presented between fields, introduction of new attributes in future versions of spreadsheets, the unknown intention of interactions between multiple data fields, the limited amount of sample we had, the lack of data relevant to assignment decisions, the pre-existing conflicts on partially assigned spreadsheets and how to unwind them, and the decisions about which classes were in our scope to be included for consideration of automatic assignment.

Our team also faced a number of software engineering challenges that tested out understanding of requirements and required us to compromise the utility of our skillsets in favor of a plan that was more aligned with the vision of the sponsor and key stakeholders. In the planning stages of the project, many of these requirements were only partially defined, which introduced ambiguity into our design and development process. As we

continued to meet with our sponsor, their objectives and expectations gradually became clearer, allowing us to better tailor our development priorities to their vision.

However, even with improved communication and clarity, we recognized early on that it might not be feasible to meet every single request or desired feature within the limited timeframe and resources available. Some requirements presented technical or logistical barriers that could have impacted areas of the application or development schedule. During our initial meetings, our sponsor emphasized that several institutional and practical constraints would need to be considered during development, some of which could not be resolved directly by our team.

This required us to adopt a strategic approach to project management, where we had to carefully evaluate and prioritize feature implementation based on feasibility, impact, and alignment with the core goals of the platform. This meant making tough decisions about which features to develop fully, which to simplify, and which to postpone or omit altogether. Our guiding principle became maximizing value within our constraints and delivering a functional, scalable, and user-friendly application that addressed the most critical needs of the sponsor while leaving room for future enhancements and iterations. We wanted this program to be used in production without being abandoned for not conforming the real-world needs of a classroom coordinator.

These software engineering challenges taught our team valuable lessons in managing stakeholder expectations, balancing scope with available resources, and maintaining flexibility in a dynamic development environment. They reinforced the importance of iterative development, clear communication, and the need for regular feedback loops when working on real-world software solutions with external partners.

1.5 Context

The goal of our application is to design a web-based scheduling platform that automates the assignment of course sections to classrooms in the Peter Kiewit Institute (PKI) building at the University of Nebraska at Omaha. The subject facet of our project focuses on streamlining and replacing the current manual scheduling process handled by a single classroom coordinator. Each semester, the coordinator must interpret and manage large amounts of spreadsheet data to assign classes to rooms while resolving a variety of conflicts related to time, capacity, and room requirements. This process is tedious, mentally taxing, and has contributed to high turnover in the position. Our solution

produces an automated Course Section Assignment Draft and integrates conflict detection, resolution tools, and an exportable CSV format for submission to the registrar. Automating this process increases efficiency, accuracy, and consistency—ultimately improving employee retention and reducing scheduling bottlenecks.

The usage facet addresses the real-world workflow at UNO, which includes four distinct phases: Planning, Proof, Assignment, and Publish. Our application supports the Proof and Assignment phases, during which coordinators finalize data and produce usable classroom assignments. The input data remains in spreadsheet (CSV) format to align with current practices. Our tool accounts for several recurring challenges:

- Assigning cross-listed undergraduate/graduate courses to a shared room,
- Accommodating hybrid in-person/remote classes without consistent time data,
- Satisfying special classroom constraints like equipment needs (e.g., projectors, HDMI, lab computers),
- Ensuring departments with recurring room usage trends are matched to consistent room assignments where possible.

Once conflicts are resolved or acknowledged, coordinators can generate a finalized CSV to share with the registrar. The Publish phase, which involves campus-wide schedule distribution, remains beyond the scope of this application but may benefit from future integration.

From the IT system facet, our application is designed as a cross-platform web tool hosted on cloud infrastructure, using Amazon Web Services (AWS) for deployment. Services include S3 for static content hosting, EC2 for backend application logic, and RDS for persistent database storage. The system could also be migrated to Google Cloud depending on future stakeholder preferences. While future integration with UNO's internal systems is anticipated, the current version remains self-contained, using static file uploads and outputs. This ensures compatibility with current workflows while maintaining the flexibility to scale and evolve.

The development facet involves a modular design using the Angular framework for the frontend and Flask (Python) for the backend. Object-oriented programming is used to model course sections, classroom objects, conflict logic, and assignment history. The backend uses RESTful APIs to provide secure and consistent communication between system components. Development followed Agile methodology with iterative releases, continuous testing, and GitHub-managed branching. Automated testing using Pytest and Cypress ensures code integrity. Additionally, documentation is automatically generated with Sphinx for the backend and Compodoc for the frontend. All frontend assets are

hosted statically and served through an S3 bucket, enabling smooth, fast rendering for users.

The legal and ethical facet of our project is grounded in three core principles: privacy, fairness, and control. The system handles sensitive data including instructor names, room assignments, and instructional methods, so user authentication and secure access are critical. Our scheduling algorithm avoids bias by relying on objective factors like room capacity, class size, and equipment requirements while allowing for manual overrides where appropriate. Role-based access ensures decisions are made by authorized personnel. The system preserves a clear audit trail for changes, allowing users to track adjustments and maintain transparency with the registrar. Ultimately, our design ensures that users retain control over scheduling decisions, supporting a fair and ethical approach to academic operations.

2. Requirements

2.1 Goals

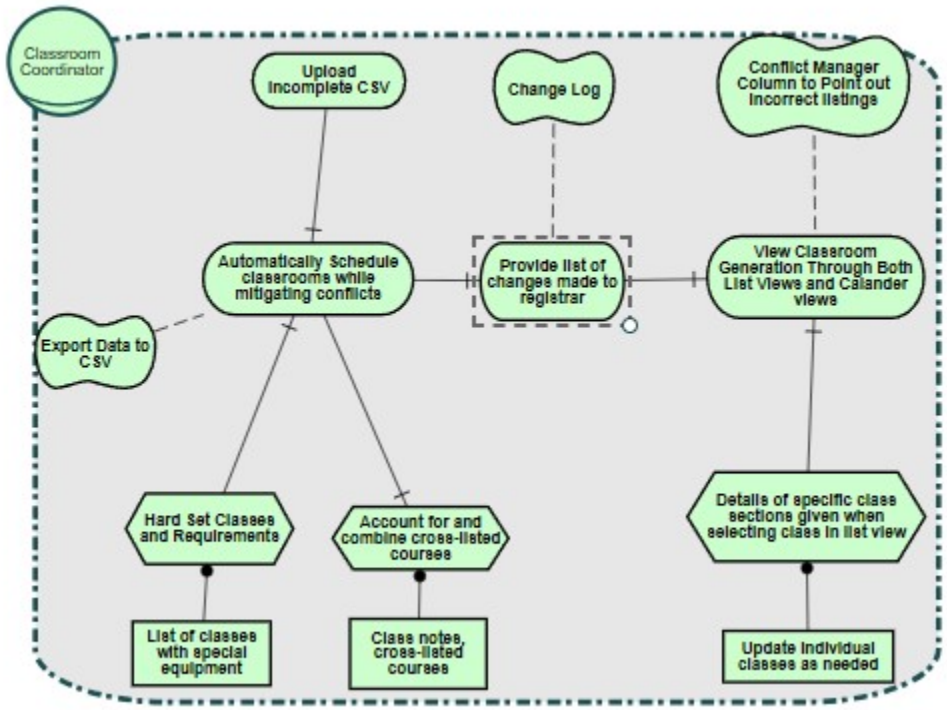


Figure 2-1: Goal Model Diagram

For our goal model, our overarching goal is to create a way that the classroom coordinator is able to automate their task of setting classroom schedules for a semester. This is accomplished by providing a list of changes made and a destination for each assigned classroom. This main goal can be expanded apon and will be linked to automating the scheduling of classes into PKI classrooms, and to provide different views for the classroom coordinator to be able to accurately and efficiently modify and confirm the classroom schedule These will serve as the top points in our model. Branching off the web application goal we have soft goals of creating multiple interactive schedule views, as well as making these views easy to use and visually appealing. Additionally, we are striving to create the scheduler in such a way that little to no manual changes will be needed after the schedule is set but will contain a built-in conflict manager that shows any conflicts that the process was unable to catch and makes note of that in the changes log.
Branching off the scheduling goal we have a series of tasks that break down the scheduling process, like importing data from the Excel file, filling in hard-set classes, accounting for

and combining cross-listed courses, filling in the remaining classes, prompting the user/coordinator for conflicts, and finally exporting the completed schedule back into an Excel file.
Regarding prompting the user for conflicts, we will need to make sure that all classes are assigned to a room that doesn't conflict with other classes or teacher's schedules. The user will be able to access the original file conversion that contains teacher, class section, special requests, and cross-listing information to help mitigate these conflicts themselves. They will also have access to the department chair and registrar as resources for any additional conflict mitigation needs that they can't meet on their own.

2.2 Functional Requirements

Must Have – UI, Scheduling Algorithm, manual overrides
Should Have – Fancy UI
Could Have – User Authentication
Won't have – App, LLM for accurately scheduler classes, integration with UNO database, support for classes that don't conform to the regular session class calendar, such as J-term classes.

2.2.1 BDD Scenarios

As a classroom coordinator so that I can correct errors efficiently, I want to have an undo/redo feature.

As a classroom coordinator so that I can have a finalized schedule without conflicts, I want to have all of the conflicts presented to me.

As a classroom coordinator so that I can easily communicate changes with the registrar, I want to have an automatic exporter for the solution CSV.

As a classroom coordinator, so that I can prevent any delays with scheduling, I want the scheduler app to automatically assign every class it can — subject to the following constraints:

- The classroom must have capacity for the expected class size.
- Classes cannot overlap in the same classroom.
- Classrooms must meet technical requirements (e.g., projector, HDMI support, or computer lab if needed).
- Sections with cross listings must be assigned to the same room.
- Assignments should avoid overfitting (e.g., avoid placing 5-student classes in 60-seat rooms if smaller rooms are available).

- Historical and department-based room usage should be respected to promote scheduling consistency.

As a classroom coordinator so that I can locate conflicts for a specific section, I want to be able to search for sections by key-phrase.

As a classroom coordinator so that I can get arbitrary information about sections, I want a view that shows attributes of a section.

As a classroom coordinator so that I can answer questions from the chair about distribution of assigned sections by criteria such as department, class seniority, and class enrollment count, I want to be able to color code classes in the available views.

As a classroom coordinator, so that I can understand room availability and usage, I want to be able to view the full schedule for each classroom, including which sections are scheduled and when.

As a classroom coordinator, so that cross listed sections are scheduled together, I want the scheduler to automatically detect cross listings and assign the same room to all sections in the group.

As a classroom coordinator, so that I can clearly communicate finalized assignments to the registrar, I want the application to generate a complete log or report of all classroom assignments, showing details for each course section.

As a classroom coordinator, so that I can track modifications to section assignments, I want a change log to show edits made since the initial import — including what changed, who made the change, and when.

As a classroom coordinator, so that I can review detailed course information, I want to be able to view all section attributes (e.g., instructor, meeting pattern, equipment needs, max enrollment) in a centralized panel.

2.2.2 Use Cases

2.2.2.1 Use case 1. Assign Class based on capacity and schedule.

Actors: Coordinator
Pre-conditions: Coordinator logged into system. A class has been scheduled, with defined size, required room type, and timing.
Post-conditions: The system has assigned a classroom that fits capacity and schedule.

Flow of events:
- The administrator initiates the scheduling process by running the automatic scheduling algorithm.
- The system checks available classrooms based on the class size, room type, and scheduling conflicts.
- The system assigns the class to an available classroom.
- If no room can accommodate the class, the system flags the class as unassigned and notifies the administrator.

Remarks: This use case ensures that classrooms are assigned according to predefined constraints like room size, schedule, and class type, preventing scheduling conflicts and ensuring efficient use of available resources.

2.2.2.2 Use case 2. Manual Override

Actors: Coordinator
Pre-conditions: A class has been automatically assigned to a room. The coordinator is logged into the system.
Post-conditions: The system updates the classroom assignment with the manual selection. The overridden room assignment remains in place for future scheduling.
Flow of events:
- The administrator views the classroom assignment for a scheduled class.
- The administrator selects the option to manually override the classroom assignment.
- The administrator chooses a new classroom.
- The system confirms the manual override and updates the classroom assignment.
- The system ensures that the new manual assignment is respected in future scheduling attempts unless manually changed.

Remarks: This use case allows the administrator to correct automatic assignments if needed and ensures that the manual override is respected in the future.

2.2.2.3 Use case 3. Login to Scheduler system

Actors: User (coordinator)
Pre-conditions: The user has valid login credentials. The user is on the login page of the system.
Post-conditions: The user is logged in and redirected to the scheduling dashboard. If the login fails, the system displays an error message.

Flow of events:

- The user enters their username and password on the login page.
- The system validates the credentials.
- If credentials are correct, the user is logged in and directed to the scheduling dashboard.
- If credentials are incorrect, the system displays an error message prompting the user to try again.

Remarks: This use case ensures that only authorized users can access and make changes to the scheduling system.

2.2.2.4 Use case 4. Detect scheduling conflicts

Actors: Coordinator

Pre-conditions: The administrator is logged in. The system has classrooms with scheduled classes.

Post-conditions: The system identifies and flags any scheduling conflicts. The administrator is notified of the conflicts.

Flow of events:

- The administrator initiates the scheduling process or checks existing schedules.
- The system scans for potential conflicts (e.g., overlapping classes in the same room at the same time).
- The system flags any conflicts found and notifies the administrator.
- The administrator resolves the conflicts manually by reassigning classes as necessary.

Remarks: This use case helps administrators quickly detect and resolve conflicts to ensure that no classroom is double-booked.

2.3 Wireframes

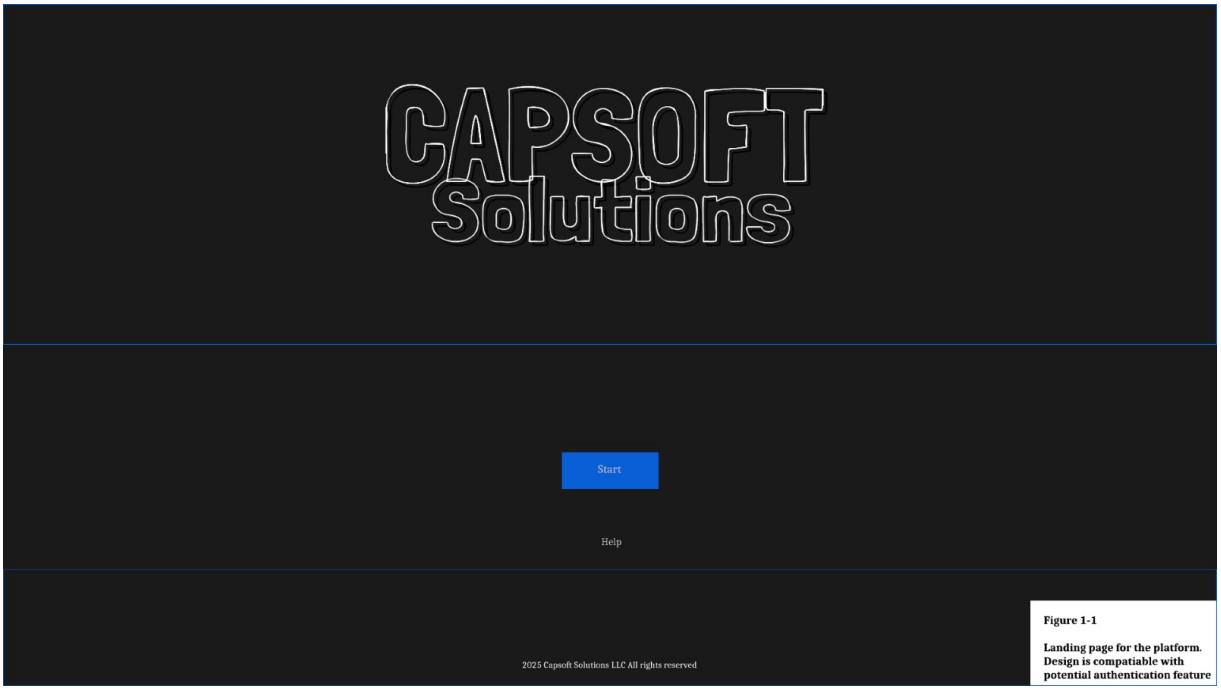


Figure 1-2: Landing page for the platform. Design is compatible with potential authentication feature.

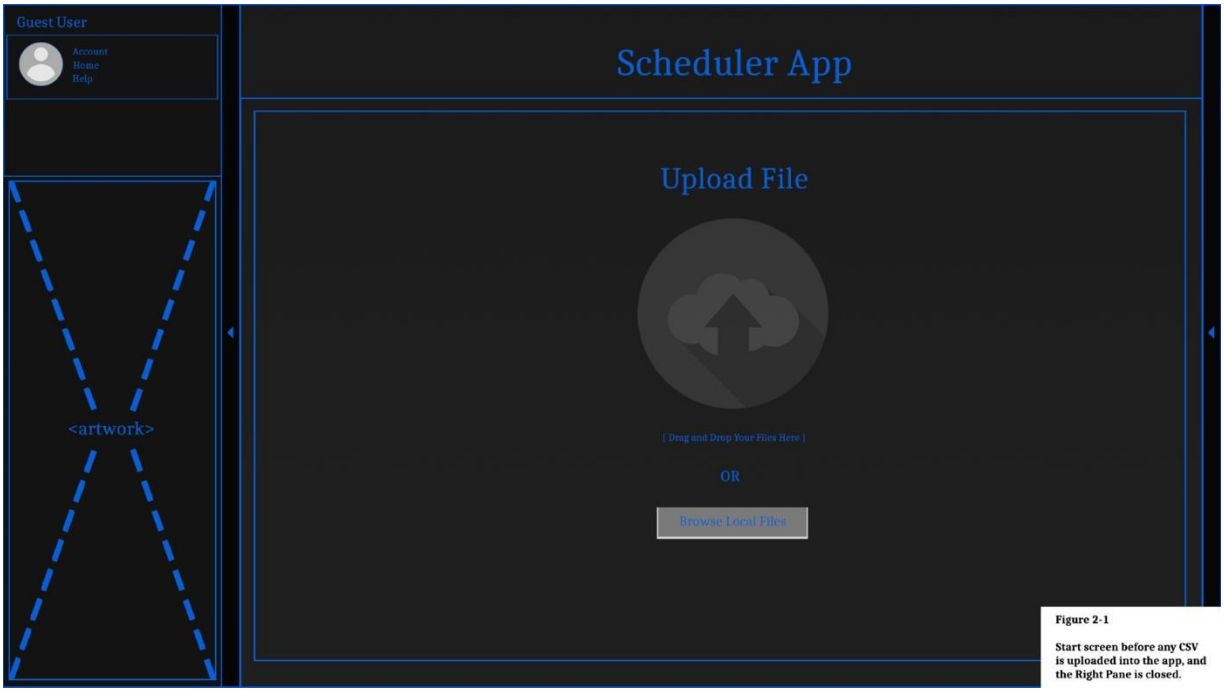


Figure 2-1: Start screen before any CSV is uploaded into the app, and the Right Pane is closed.

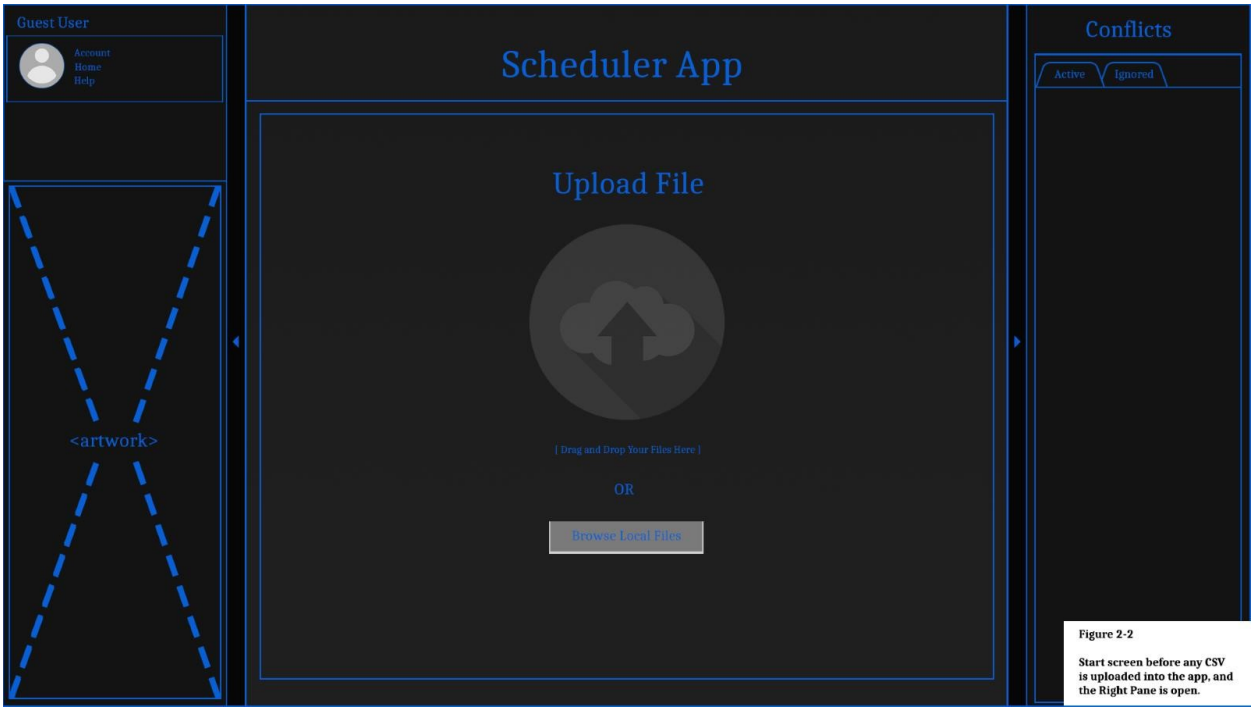


Figure 2-2: Start screen before any CSV is uploaded into the app, and the Right Pane is open.

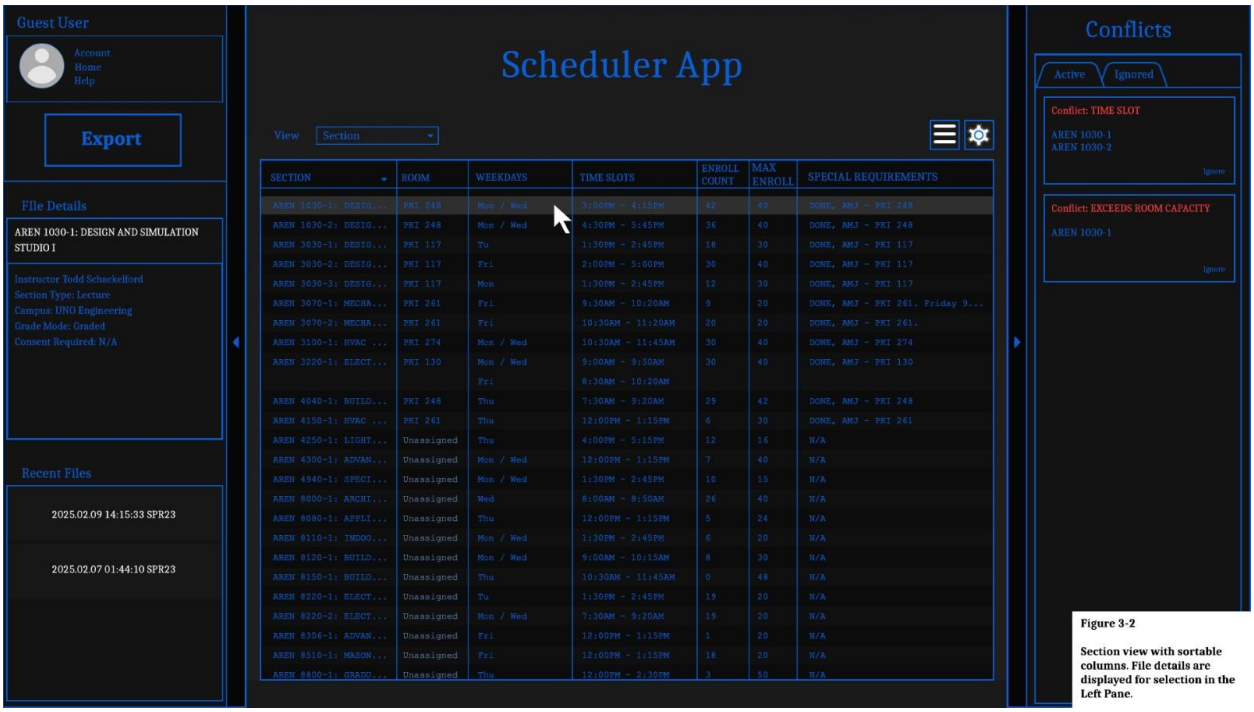
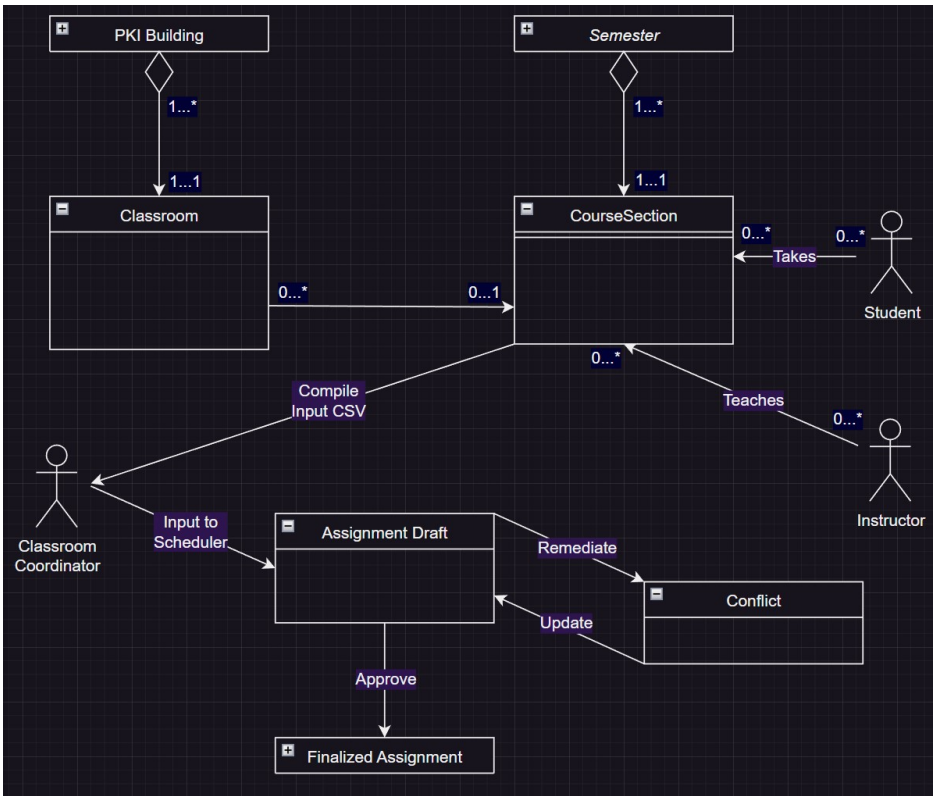


Figure 3-2: Section view with sortable columns. File details are displayed for selection in the Left Pane

2.4 Object Model



3-3: Object Model

Optional entities include Conflicts and Assignment Drafts. Conflicts might not occur if the input CSV has nothing that needs to be addressed after the automatic scheduling is applied. The Assignment Draft could also be redundant if the automatic scheduler output needs no revision by the Classroom Coordinator, although it would still need to be approved by the Classroom Coordinator.

The remaining entities are required: PKI Building, Semester, Classroom, CourseSection, Finalized Assignment, Classroom Coordinator, Student, Instructor. All of these entities are a necessary component to produce the solution CSV and model would not work without them.

2.5 Non-Functional Requirements

1. Performance
Goal: Ensure the application processes and assigns all classroom schedules within a reasonable time.

- Question: How quickly can the system process a CSV file with 500+ course sections and return classroom assignments?
- Metric: System should complete scheduling and conflict analysis within 30 seconds for up to 500 course sections on a standard AWS EC2 instance.

2. Reliability
Goal: Provide a stable system with minimal crashes or failed processes.

- Question: How often does the system fail during file upload, assignment generation, or conflict detection?
- Metric: System uptime should exceed 99% during testing and deployment; error logs should show no more than 1 critical failure per 100 sessions.

3. Usability
Goal: Make the application intuitive and user-friendly for non-technical users.

- Question: Can a new user complete a basic scheduling task (upload, assign, export) without training?
- Metric: At least 80% of test users should be able to complete these tasks within 10 minutes with no prior training.

4. Portability
Goal: Ensure compatibility across different devices and browsers.

- Question: Does the application function properly on all major browsers (Chrome, Firefox, Edge, Safari)?
- Metric: The UI and key features must work on 100% of tested browsers on both Windows and macOS platforms.

5. Scalability
Goal: Support growth in dataset size or number of users without major performance degradation.

- Question: How does the system handle large class schedules or multiple concurrent users?
- Metric: System must support at least 3 concurrent users and datasets of up to 1,000 sections with <10% increase in response time.

6. Security
Goal: Protect sensitive user and scheduling data.

- Question: Is access restricted to authenticated users? Are passwords securely stored?
- Metric: All endpoints that expose sensitive data must be protected with JWT authentication; passwords must be hashed and not stored in plain text.

2.6 External System Interfaces

Our web application interacts with external systems primarily through file-based data exchange and a set of RESTful APIs. These interfaces are designed to accommodate the

current operational infrastructure at the University of Nebraska at Omaha (UNO), with future extensibility for direct database integrations.

2.6.1 File Format Interfaces

2.6.1.1 Input File Requirements

- Course Sections CSV (INPUT_CSV)
    - Required for initializing section data each semester
    - Must include fields such as:
    - Course ID
    - Section Type
    - Meeting Pattern
    - Instructor
    - Room
    - Enrollment, Max Enrollment, etc.
    - File Location: uploads/Fall2025.csv (configurable via upload endpoint)
- Classroom Resources CSV (ROOMS_CSV)
- Used to populate available classroom data for assignment
    - Required columns:
    - Room Number
    - Seats
    - Displays
    - Computer Count
    - Information and Connectivity
- File Location: Files/PKIRooms.csv
- Training Data Files
    - CSV files used to generate frequency maps for predictive assignment
    - Managed via /training/upload and /training/reset
    - Default: Fall2022.csv, Fall2025.csv

2.6.1.2 Output File Specifications

- Output CSV (OUTPUT_CSV)
    - Updated course section file with room assignments filled in
    - Output file: Files/Exports/room_assignments.csv
- Downloadable via /api/download
    - Maintains original column structure with an updated Room field
- Change Log Text File (CHANGELOG_FILE)

    - Human-readable summary of changes from the original CSV input
    - Output: Files/Exports/change_log.txt
    - Downloadable via /api/change-log

2.6.2 REST API Endpoints

2.6.2.1 Authentication & User Management

- POST /api/login
  Authenticates user via username and password, returns JWT token.
  Input: { "username": "user", "password": "pass" }
  Output: { "token": "JWT-token" }
- POST /api/register
  Registers a new user.
  Input: { "username": "user", "email": "email", "password": "pass" }
  Output: { "msg": "User created" }
- GET /api/current-user
  Returns the username from the current session (if JWT exists).
  Output: { "username": "admin" }

2.6.2.2 Uploads & Assignment Initialization

- POST /upload
  Uploads the input CSV and triggers assignment logic.
  Form-data:
    - file: the input CSV
    - logic: JSON list of assignment modes (["historical", "predictive"])
      Output: { "message": "File uploaded and memory updated", "filename": "Fall2025.csv" }
- GET /training/files
  Lists current training CSVs
  Output: { "files": ["Fall2022.csv", "Fall2025.csv"] }
- POST /training/upload
  Uploads a new CSV for training frequency map
  Input: form with file field
- POST /training/reset
  Resets training list to default set

2.6.2.3 Data Retrieval and Editing

- GET /api/data
  Returns updated assignment data (requires authentication)
- POST /api/data/columns
  Filters response columns for the table
  Input: { "columns": ["ROOM", "MEETING_PATTERN"] }
- GET /details/<id>
  Returns HTML-formatted detail panel for a section
- GET /edit/<id>
  Returns editable data for a section
- POST /edit/save/<id>
  Updates a section's room, meeting, instructor, etc.

2.6.2.4 Conflict Management

- GET /conflicts/active
  Returns all active conflicts
- GET /conflicts/ignored
  Returns ignored conflicts
- PUT /conflict/ignore/<id>
  Marks a conflict as ignored
- PUT /conflict/activate/<id>
  Reactivates an ignored conflict

2.6.2.5 Assignment File State Management

- GET /assignment-files
  Returns a list of available assignment file snapshots from the DB
- GET /assignment-file/<fid>
  Loads a specific assignment file into memory
- POST /assignment-file/init/<fid>
  Sets a given file as the active memory state

2.7 Requirements Not Implemented

Persistent Assignment File Browser:

The scheduler should maintain a built-in file management system that allows users to save and load assignment files with custom filenames. These files would appear in a user-friendly browser panel within the platform, making it easy for coordinators to retrieve previous work or distribute drafts to stakeholders. Each saved file should preserve the full internal state of the system, including the uploaded input CSV, all CourseSection, Classroom, and Conflict objects. This feature was planned to rely on a relational database for secure, persistent storage but was not completed in the final version.

User Settings Memory:

The platform was intended to store user preferences to improve workflow continuity. These preferences would include column layout customization, selected assignment logic (e.g., predictive vs. historical), and a record of training datasets used in prior sessions. Upon logging in, each user would have their personalized workspace automatically restored. This feature was partially designed but not fully integrated into the system.

Role-Based Access Control:

Although the current application supports basic authentication, a planned feature was to implement role-based access control to differentiate between user types—such as administrators, coordinators, and viewers. Each role would have specific privileges regarding data editing, exporting, and access to advanced settings. This was not implemented due to time constraints.

Full Integration with UNO Systems:

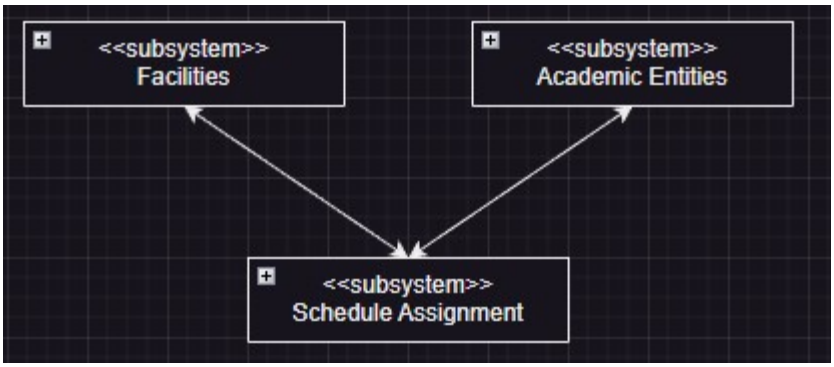
While CSV import/export is supported, full integration with UNO's internal databases and registrar APIs was planned as a long-term goal. This would allow the platform to directly ingest official course schedules and push finalized assignments to registrar systems. However, this integration was not feasible without server access and was out of scope for this project phase.

3. Architecture and Design

The frontend is hosted statically on an S3 bucket, which has its endpoint URL converted to our registered domain: capsoftsolutions.com using AWS's Route 53.

Our users access the platform through their own computers, using a compatible browser that renders the frontend, sends user requests to the Elastic Beanstalk server(s), and displays server responses.

3.1 Logical Decomposition



3-4: Logical Decomposition

3.2 Development

The project is structured with a Python backend and Angular frontend. Here we will detail the functionality of the individual components of each.

3.2.1 Python Backend

The backend of our application is built using a structured, object-oriented design that aligns with the Model-View-Controller (MVC) paradigm commonly used in industry-grade software development. This approach divides the system responsibilities into three main layers: Models (data structure and business logic), Controllers (application logic and API endpoints), and Views (represented by the frontend in Angular). Our backend was built

using Flask, a lightweight but powerful Python web framework, and incorporates a clear separation of concerns that improves maintainability, scalability, and collaboration among team members while also providing our group the most usability since it is what our group considered our best language.

3.2.1.1 Flask

Flask uses a route file, or set of route files, to handle requests from the frontend by structuring the functions with decorators that specify the applicable URL. We opted to use a unified route file due to the smaller size of our project, serving requests for authentication, data updates on the backend, file handling, component data population and more. When information is passed to the frontend, we typically use jsonify to format the data so it can be parsed in a standard way by our typescript functions. While there are many HTTP request types you can use with these route functions, we only needed GET and POST for our project. GET simply allows the frontend to fetch information from the backend, while POST is used to send files or change information.

Some examples of functionalities that use GET/POST include:
@application.route("/training/upload", methods=["POST"])

Adds a file to our list of training CSVs, which are stored in a JSON called training_files.json at the highest level of our project directory tree.

@application.route('/edit/save/<id>', methods=['POST'])

Modifies object information on a CourseSection object using the details pane's edit tab on the frontend.

@application.route('/edit/<id>', methods=['GET'])

Gets information from a CourseSection object to populate into the details pane without any modification of information.

@application.route('/conflicts/active', methods=['GET'])

Returns all active conflicts, which are Conflict object string attributes added to a list of dictionaries, with each dictionary giving the conflict details to jsonify and pass to the frontend.

Flask uses Gunicorn, a WSGI (Web Server Gateway Interface) -compliant server commonly used in Python web deployments. Gunicorn is an industry-standard deployment tool and integrates easily with web services and containers. During development, we extended the Gunicorn timeout limit to 120 seconds to accommodate slow CSV parsing and large assignment file loads. This issue was later resolved by optimizing the parsing functions, but the configuration remains in place to handle potential future scaling scenarios.

-Controller modules, Utility modules, Files(PKI Building Info, Input CSVs, Conflict testing CSVs), Data modules
-Route file structure and function, notable imports(pickle, CORS, datetime, zoneinfo, flask_jwt_extended, urlib.parse, json), development DB contingency logic, training CSV file handling.

3.2.1.2 Python Libraries

Our main route file manages the API's external interface and contains notable imports:

- **pickle –** for serializing and deserializing Python objects across sessions and route functions.
- **CORS –** to allow secure cross-origin requests from the Angular frontend.
- **datetime, zoneinfo –** for accurate and timezone-aware scheduling.
- **flask_jwt_extended –** for secure token-based user authentication.
- **urllib.parse, json –** for parsing incoming request data and building clean responses.

We implemented development contingencies to support a local database fallback. Special logic also handles the presence or absence of a training CSV file used for room usage frequency mapping.

3.2.1.3 CourseSection Model

CourseSection objects primarily handle the parsing and sanitization of raw data attributes extracted from the input spreadsheet. Key data includes meeting patterns, room assignments, enrollment numbers, and various other course identifiers. To make these usable, functions like parse_meetings() convert meeting pattern strings into structured tuples, similar to how Classroom objects handle meeting times. For example, a string like "MWF 9am-10am" is converted into tuples containing day and minute-based start and end

times. This structured data is more practical for route functions and other models to work with, allowing straightforward access to precise meeting details.

Data remediation is another crucial process handled by CourseSection objects. The method .update_meetings() regenerates the state of a CourseSection by combining parsed meeting and room data into a tupled schedule format. This gives us consistency with the Classroom's minute-based schedules and makes adding/subtracting courses from a room schedule much easier. When .update_meetings() is called, it rebuilds both the meeting pattern string and associated room data, aligning them precisely with Classroom object schedules. This step resolves ambiguities that might have arisen from parsing older or inconsistent spreadsheets.

Older spreadsheet formats present another challenge since they occasionally lack specific attributes, notably the "Notes#2" column. To address this, CourseSection objects default any missing attributes safely to empty strings during instantiation. This prevents potential runtime errors and makes the application more stable when handling future changes to input spreadsheet format.

2.3.1.4 Classroom Model

Classroom objects are responsible for parsing, sanitizing, and managing scheduling data within our system. Initially, raw data inputs must be converted into usable formats, which we handle through several helper functions. The parse_time() function converts time strings such as "9am" or "10:30pm" into integer minute values relative to midnight, providing a uniform numeric basis for comparisons. The parse_meeting_line() function builds upon this by taking more complex meeting patterns like "MW 3pm-4:15pm; F 8:30am-10:20am" and returning structured lists of tuples, each containing the day and corresponding start and end times in minutes. Additionally, the strip_decimal() or similar helper functions ensure numeric data doesn't contain trailing decimals (e.g., converting "30.0" to "30"), maintaining consistency in data storage and processing.

The core scheduling mechanism in Classroom objects uses a minute-based schedule. This is represented as a list containing an element for every minute of a 7-day week. When courses are added or removed, the Classroom object updates this schedule by marking or unmarking specific minute ranges with corresponding course identifiers. Functions like add_course_section() and remove_course_section() change this structure, ensuring accurate, real-time updates as schedules are modified. Once the minute-level data is established, the gather_intervals() function transforms these detailed schedules into

concise intervals, making it easier to identify continuous blocks of time each class occupies, instead of being a huge unified array of schedule data.

Conflict detection leverages these intervals. The method find_conflicts() iterates through all the intervals, checking for overlaps between different course sections within the same classroom. When overlaps are found, we make tuples that give us the necessary details from both courses to construct a conflict. The resulting conflict information uses course IDs, times, and room assignments. The conflicts are actually instantiated by our assigner module. This module relies heavily on find_conflicts() to automatically identify and manage scheduling issues to make sure many conflicts aren't generated with the auto-assignment logic.

2.3.1.5 Conflict Model

Conflict objects track scheduling problems between courses or rooms, helping identify issues clearly for users. Each Conflict gets a unique identifier constructed by combining course IDs, room numbers, times, and an optional descriptive message. This prevents duplicate conflict reporting by using a consistent, repeatable format for each unique scenario.

Conflict messages provide context messages when conflicts aren't just straightforward time overlaps. This includes when enrollment capacities exceed room capacities or ambiguous meeting data can't be reliably parsed. The conflict_factory creates these messages when needed, helping users quickly understand and fix unusual scheduling scenarios beyond simple time conflicts.

2.3.1.6 AssignmentFile Model

The AssignmentFile model is central to how we organize and manage the scheduling data in our application. It holds dictionaries containing objects like CourseSections, Classrooms, and Conflicts, which are accessed frequently. Dictionaries use hashing, allowing us to instantly retrieve specific objects using unique identifiers (keys), as opposed to iterating through arrays or linked lists, which can become slow with large datasets. This ensures our backend remains efficient, especially when handling a large number of objects and frequent queries from the frontend.

The AssignmentFile also maintains lists to keep track of active versus ignored conflicts and prevent us from having to recalculate these lists when conflicts are updated to reflect changes requested from the frontend.

2.3.1.7 Room Scoring Module

The room_scorer module supports the assignment logic by generating room frequency maps based on historical data. We give training CSV file names as input, it analyzes these files and produces a dictionary mapping each room to the number of times it has been successfully assigned to a particular class. These frequency maps feed into the assigner module and provide a data-driven method for reliable and historically consistent room assignments.

2.3.1.8 Assigner Module

The assigner module manages room assignments for course sections using a prioritized approach from most reliable (least assignments) to least reliable (most assignments) methods. It uses frequency maps built from historical CSV data to determine which rooms courses have successfully used in the past. This gives us a reliable baseline for room assignments. Additionally, our analysis of extensive JSON data provided by our project sponsor was used to conduct department-conscious assignments, ensuring consistency so specific departments repeatedly use their preferred rooms.

If specific department information isn't sufficient, the assigner falls back on less-specific frequency data to make informed guesses. Predictive logic assigns by scanning course attributes to determine special requirements. For example, the assigner searches section.attributes for keywords like "projector," checks whether section.section_type indicates a "lab" or "laboratory" (signifying computer needs), and looks into section.comments for "hdmi" requirements. Assignments only proceed if rooms meet these keyword-based requirements and match the class capacity, preferring rooms with the minimum suitable capacity to avoid wasteful assignments (avoiding placing a 5-person class in a 60-person room if a 10-person room is available).

The assigner also respects crosslisted courses. Before initiating assignment logic, it verifies whether crosslisted sections already have room assignments. If they do, the assigner automatically adopts that room, bypassing standard assignment strategies.

2.3.1.9 Exporter Module

The exporter module updates spreadsheet data to reflect the most recent classroom assignments. It operates by recreating the class key used for navigating the CourseSection object dictionary for each data row found in the original export spreadsheet. It then compares these keys against existing CourseSection objects. When a match occurs, the module updates the spreadsheet row's room field with the assignment stored in the CourseSection object. This straightforward check-and-update approach ensures spreadsheet data always aligns with the latest internal room assignment decisions.

Unimplemented features include persistent user workspaces, which would store information about how a user has their application set up such as column customizations, automatic assignment strategy selections, training CSV files uploaded, whether or not the right pane is collapsed, and what view the user is currently on. Additionally, we weren't able to incorporate a visual representation of the AssignmentFile model as a list of files to load or save on the home screen. As such, the database transactions needed to make this work were also not implemented. The AssignmentFile objects still have utility by neatly packing the CourseSections, Classrooms, and Conflicts into one serializable structure with our Python pickle library. It also maintains lists of ignored versus active conflicts between courses and rooms to avoid constant regeneration of this information.

2.3.1.10 Factory Modules

Factory modules handle data extraction, sanitization, and the instantiation of our model objects. The course_section_factory reads CSV files and builds CourseSection objects, performing initial data cleanup and using exclusion logic to prevent unhandled classes from being populated, such as ignoring online-only or special-term sections. The classroom_factory processes room details from CSV data, creating Classroom objects and linking them to the relevant course sections, ensuring default values are set when information is missing. Lastly, the conflict_factory uses data from both classrooms and course sections to identify scheduling conflicts based on timing overlaps, ambiguous parsing results, or capacity constraints, generating conflict messages to flag issues for the user.

2.3.1.11 The Problem of Parsing Meeting Times with Assigned Rooms

Parsing conflicts can occur when the provided data about room assignments and meeting times is ambiguous or incomplete in the assignment spreadsheet, making it challenging to clearly determine how to match meetings with rooms. To manage this, we've defined several conditional strategies.

Example 1:

Meeting Pattern: MW 8:00AM - 9:55AM; F 10:00AM - 11:25AM

Room(s): PKI 116; PKI 276

In the scenario provided above, it seems obvious the first room (116) is assigned to the meeting time on Monday/Wednesday whereas the second room (276) is assigned to the meeting time on Friday. Maybe PKI 276 is a lab to compliment the lecture in PKI 116, for instance.

Example 2:

Meeting Pattern: M 8:00AM - 9:55AM; M 8:00AM - 9:55AM; T 9:30AM - 10:45AM

Room(s): PKI 116; PKI 276

Here we have two duplicate meeting times, followed by a unique meetings time. We are forced to make assumptions about which rooms are reserved. Luckily, our methodology was something we could cross-check with the previous assignments that were made in past semesters using the Mavlink portal, which students use to register for classes and more. While we haven't been able to guarantee validity for all outputs of our parsing functions, this risk is mitigated by the analysis we have conducted for edge cases, by the fact that these assignments are likely conducted more manually by the registrar and aren't something our automatic assignment will be often exposed to, and the fact that our automatic assignments aren't blindly accepted via an output CSV, but rather each change is provided to the registrar by the Classroom Coordinator by using our Change Log export feature.

Condition 1 handles cases where exactly two rooms are listed, but there are more than two meeting times. Here, we first check for duplicate meeting times:

- **Condition 1a**: If duplicate meeting times are found, all meetings from the first duplicate onward are assigned to the second room, and prior meetings to the first.
- **Condition 1b**: If no duplicates are found, we simply assign the last listed meeting time to the second room and all earlier meeting times to the first room.

Condition 2 addresses unusual scenarios:

- **Condition 2a**: If exactly two rooms are listed but only one meeting time exists, it's a double-booking scenario. This means both rooms are explicitly reserved for the same single class session.
- **Condition 2b**: If more than two rooms are listed but fewer meeting times are provided than rooms, the assignment is unclear. This triggers a parsing conflict, prompting the user to clarify the intended assignments, since the logic cannot safely guess room-meeting relationships in this case.

Condition 3 considers cases where there are more than two rooms and more than three meeting times. To address potential complexity here, we use a backward assignment strategy: the last meeting time gets assigned to the last room, the second-to-last meeting time to the second-to-last room, and so on. Any leftover meeting times (if they exist) are assigned to the first room listed.

In simpler, common scenarios:

- If the number of rooms exactly matches the number of meeting times, each room is simply assigned to each meeting time in sequential order.
- If only one room is listed, it's assumed all provided meeting times apply to this single room. Overlapping or duplicate meeting times within this single-room scenario indicate a data-entry error, as logically, a single room cannot host two classes simultaneously.

By following these structured conditions, we do our best to ensure that ambiguous scheduling data is clearly flagged, and accurate assignment is maintained wherever possible.

3.2.2 Angular Frontend

Our Angular frontend follows a standard approach, utilizing components and sub-components to structure both visual and functional aspects of the web interface. Each component is made up of three main files: a TypeScript file for functional logic, an HTML file for structural layout and dynamic data population, and an SCSS file for visual styling.

3.2.2.1 Request and Data Flow

To provide clarity on request and data flow in our application, we handle backend interactions by having leaf components subscribe to data returned by the Flask backend, while actual GET and POST requests are routed exclusively through the home component to the data.service component. This approach stems naturally from our frontend structure, where a centralized "services" directory, positioned at the same level as the home component, manages backend requests. Although it's likely alternative methods exist to directly provide leaf components access to the services directory, our approach simplifies the data handling logic: leaf components receive data indirectly by subscribing to observables provided through the home component.

3.2.2.2 Angular Libraries

We used Angular Material libraries to enhance our user interface. These include MatCardModule, MatButtonModule, MatTabsModule, MatTableModule, MatSortModule, MatFormFieldModule, MatInputModule, MatCheckboxModule, and MatDialogModule. Each of these offer structured layouts, user inputs, dialogs, and interactive elements. For much-needed feedback to the user during CSV uploads, we used MatProgressSpinnerModule as a loading icon while automatic assignments are made and the table is populated.

3.2.2.3 Enforcing Authentication

Additionally, our frontend features an AuthInterceptor component that monitors and intercepts outgoing HTTP requests. If a JSON web token (JWT) is stored in localStorage, the interceptor appends an "Authorization: Bearer <token>" header, allowing our Flask backend to differentiate between authenticated and unauthenticated users. Consequently, backend routes can specify which endpoints require authentication using decorators above each privileged route function allowing us to implement simple authentication into the program.

3.2.2.4 Environment Configuration

Our environment configurations enable seamless switching between local development and production deployments without altering hard-coded URLs. Each request URL dynamically references "${environment.apiBaseUrl}" according to the current environment settings. Running the frontend locally is as simple as executing the command npx ng serve, which connects requests to the development backend at "http://localhost:5000". When deploying to our AWS production environment, we generate optimized static files via ng build --configuration=production, which automatically reference the appropriate production URLs.

3.3 Data

Our relational database is incredibly simple for a couple reasons. The first is our authentication is the only table that is functioning in production. The other table was for assignment files, which was never implemented. Even if it were implemented, our project mainly relies on an input CSV and an output CSV or an output Change Report, and for this reason, our project doesn't have a large need for data persistence. The next reason is due to the fact that we have a reliably small number of courses and classrooms compared to programs that deal with huge datasets.

```
+------------------+
| Tables_in_maindb |
+------------------+
| assignment_files |
| users            |
+------------------+

+-------+--------------+------+-----+---------+-------+
| Field | Type         | Null | Key | Default | Extra |
+-------+--------------+------+-----+---------+-------+
| id    | varchar(255) | NO   | PRI | NULL    |       |
| obj   | json         | NO   |     | NULL    |       |
+-------+--------------+------+-----+---------+-------+

+-----------------------------+-----------------------------
```

3.4 Execution Flow View

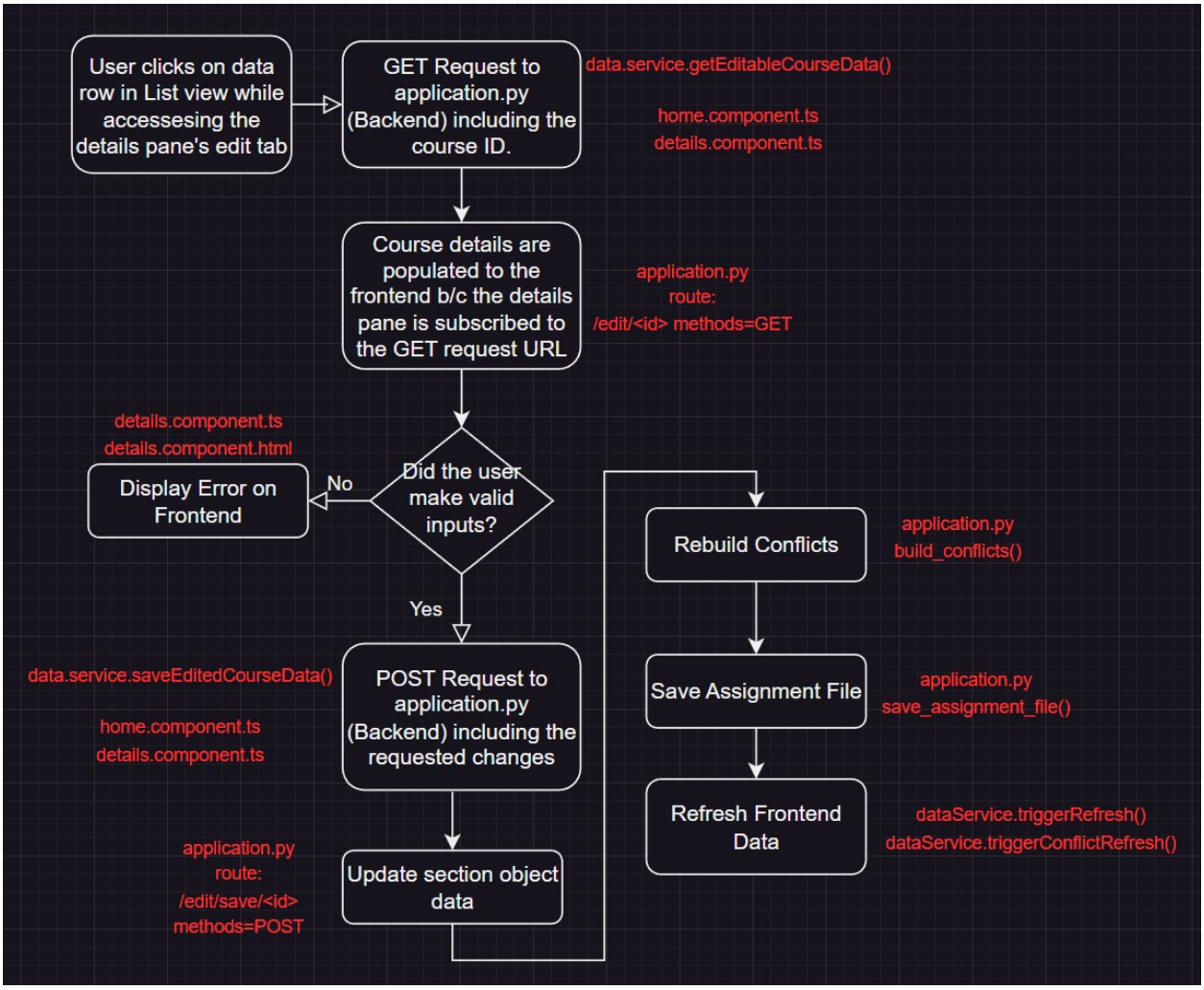


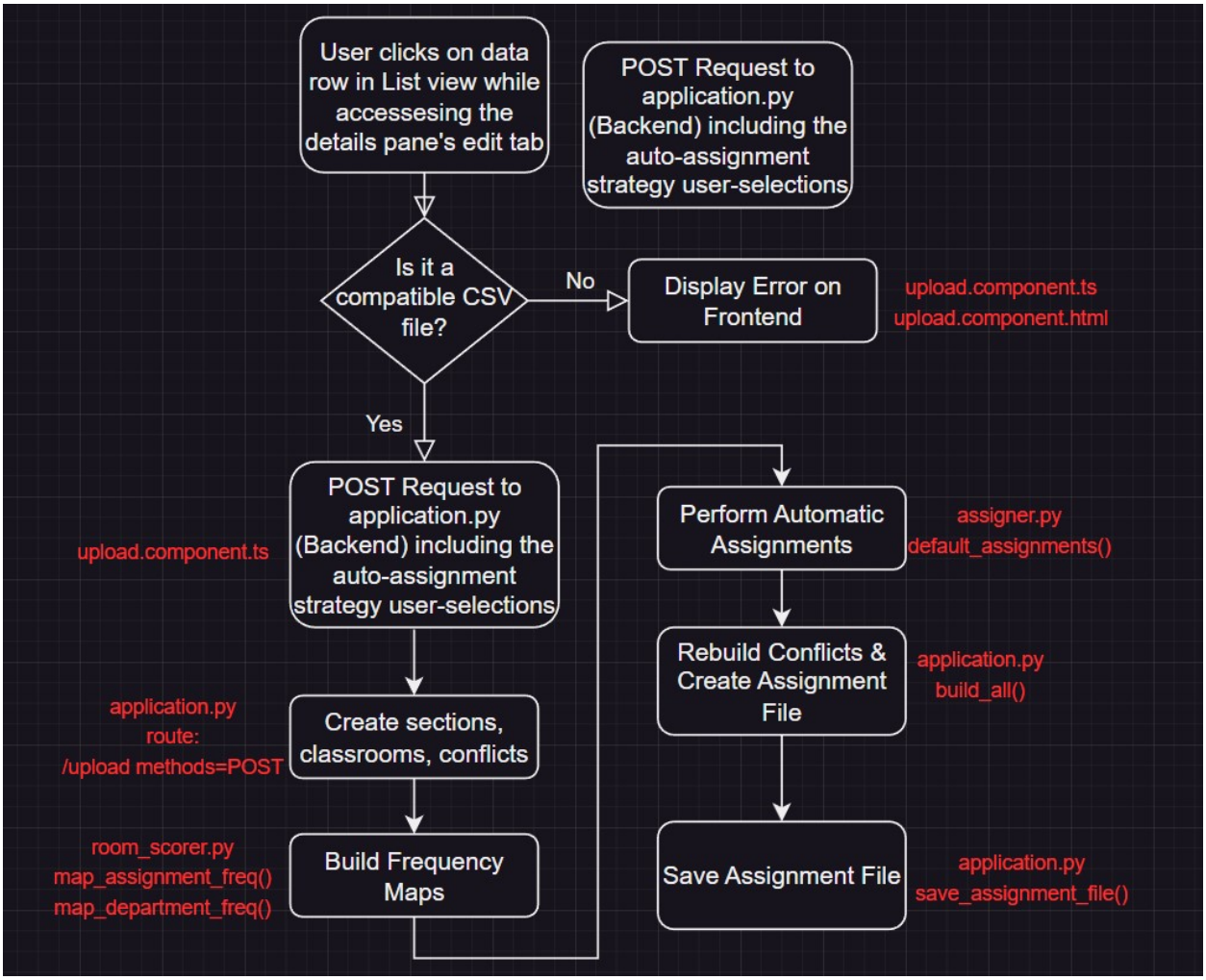
Figure 5-1: Flow Chart for editing course data using the home page's edit tab in the details pane.

In this diagram we can see how the user is viewing the edit tab in the details pane and when they select a row of data from the list, the frontend's data.service.ts called a function getEditableCourseData() to request information from the backend's Flask route file.

If the user alters information in the edit tab and clicks "save changes" then the data.service again makes a function call, but this time we make a POST request to URL /edit/save/<id>. This frontend call passes the section ID to the backend so we can use it as a key to the CourseSection object dictionary and retrieve the corresponding CourseSection object and make edits to it's data according to the user input. However, if the user has invalid input in

the edit tab fields, the frontend will not make this request and will display an input validation error message to the user.

Assuming valid input is received, the Flask route function then rebuilds Conflicts because any changes to the course room or schedule can result in new conflicts. We then update the assignment file and save it. The details pane is subscribed to these changes and makes an update to its data, and we see these changes take effect thanks to data.service's refresh functions.



In this diagram we can see the process followed when a user uploads an input CSV file in the upload component. If the file is a valid CSV file, we continue, otherwise we display a file error on the frontend.

Our upload component uses data.service to make a POST request to the Flask route function /upload. The function does some preparation by establishing section objects from the CSV with build_sections() and makes sure there's no AssignmentFile already being used to pull active and ignored conflicts. In most every case there won't be an

AssignmentFile pre-existing for a new input CSV and we move on to building frequency maps with our module room_scorer.py in our Controller directory, which is included in the build_all() function in application.py We use those frequency maps to perform default_assignments() with our assigner.py module. We rebuild conflicts that might have arisen during assignment and save to an assignment file. We save out assignment file.

3.5 Screenshots

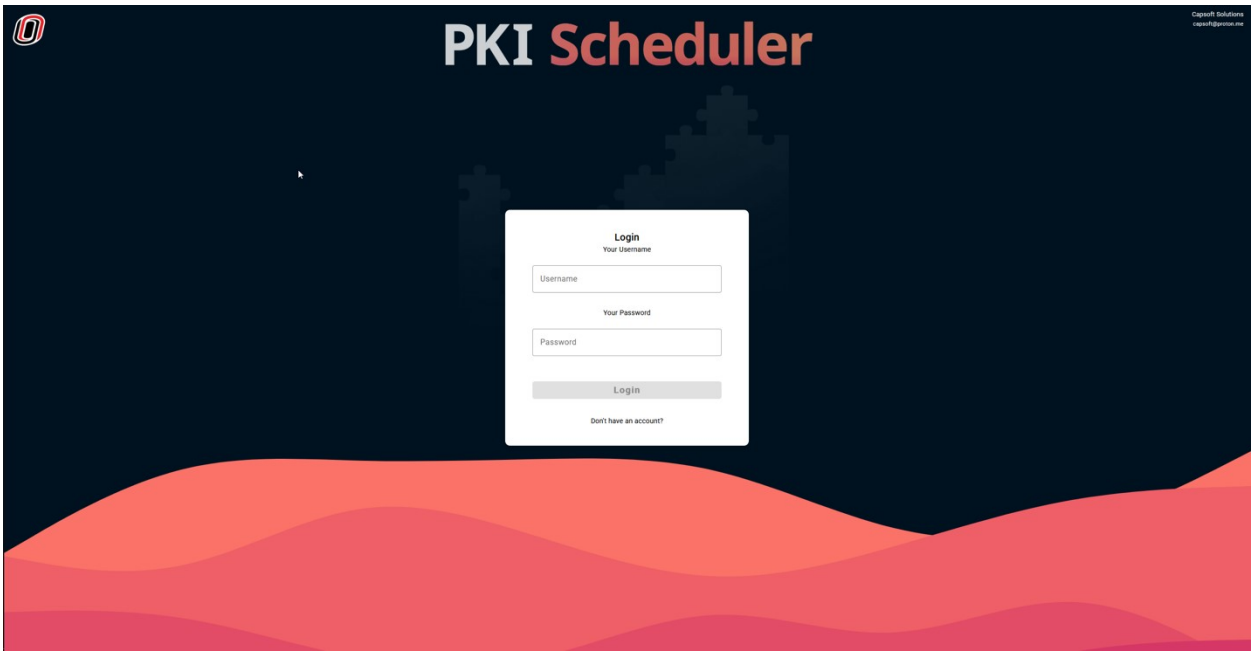


Figure 6-1: Login screen

Figure 6-2: Registration

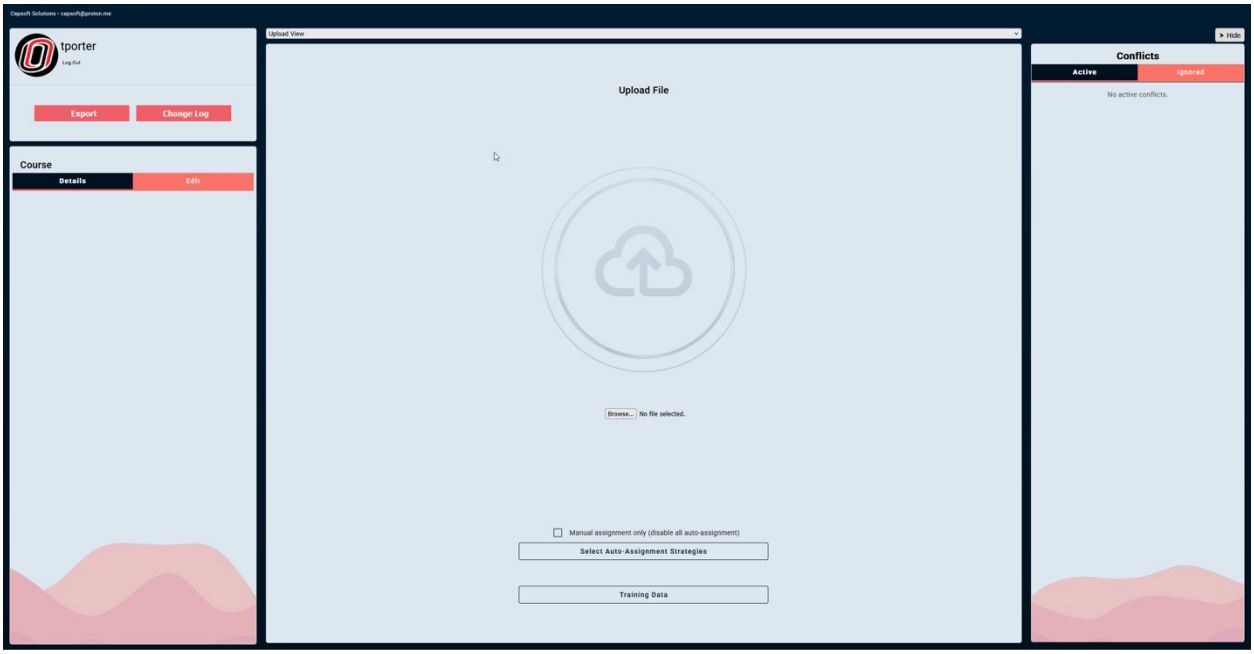


Figure 6-3: Upload

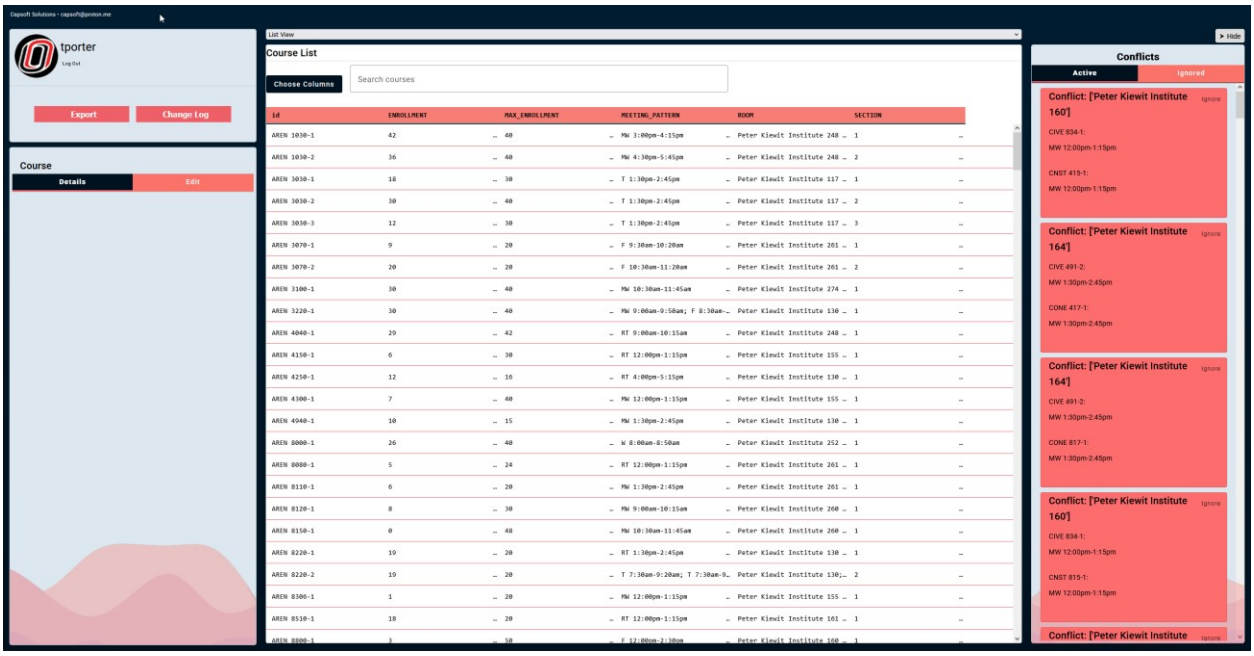


Figure 6-4: List View (Section View)

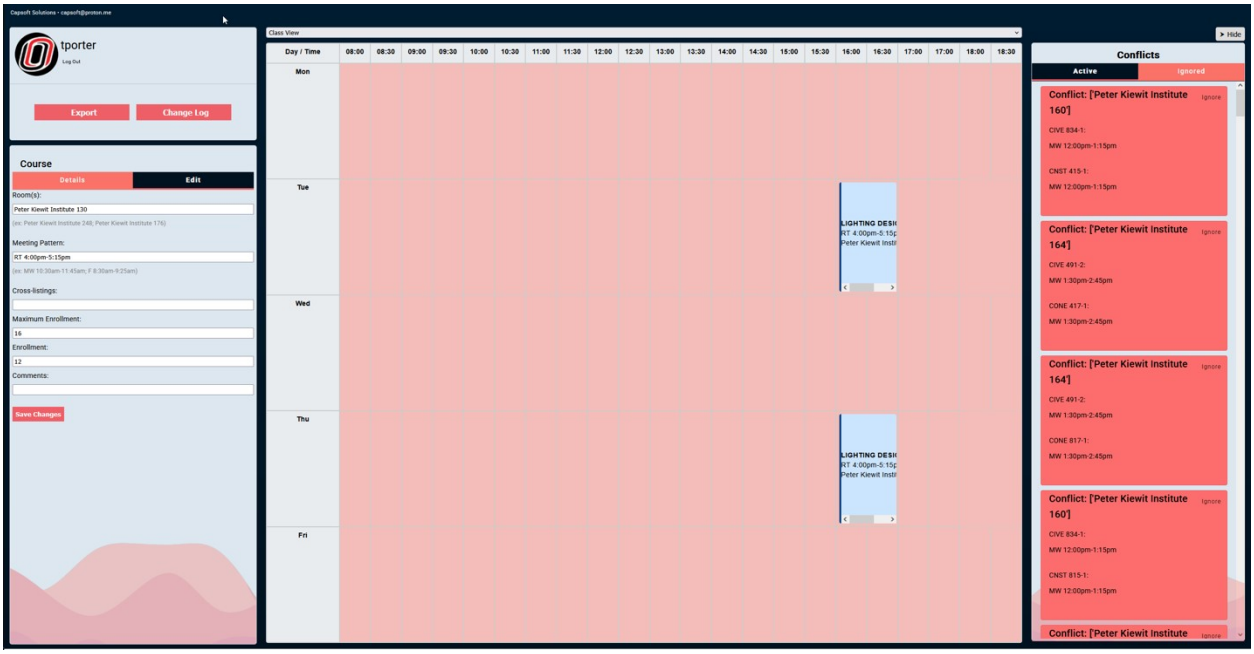


Figure 6-5: Class View (Calendar View)

4. Implementation

4.1 Design Changes

Changes in our project were surprisingly low due to much of the time being spent on object models and conceptual work at the beginning rather than excessive prototyping. This worked well for us as requirements became clearer as we received historical assignment data and further direction on use cases.

Initially we had planned to have program states to allow users to undo/redo changes they made. This was over-shadowed by more important features and we didn't have as much practical application as we anticipated for the extensive work required to design model states for the program.

We also didn't think we would need a database at all during points of the project but ultimately decided to prioritize after a sponsor meeting where we talked about how the data we were working with, while not currently considered sensitive information, could be in the future.

Our focus on the interactive calendar view was greatly compromised when prioritizing predictive assignment logic and the ability to add new CSVs to influence historical assignment logic. The calendar view was a focal point for our project, but missing the requirements to demonstrate proper assignment logic and the idea of our most reliable assignment logic becoming quickly outdated was not something we could accept.

We also intended for both panes to be collapsable on the left and right. This proved to be difficult with the structure of our html and we might ahve considered it with more time, but ultimately the details, edit, and change log export were all important aspects of everyday use for this program and collapsing it seemed unnecessary.

The change log itself was a last minute addition by request of our sponsor due to the University's process for making changes to the assignments. They required all changes to go through the registrar and interpreting the differences between our output spreadsheet and the original would not have been ideal. The change log allows users to easily show what changes were being requested in a more direct fashion.

4.2 Deployment

Deployment of our web platform is using Amazon Web Services (AWS) resources—VPC, Elastic Beanstalk, EC2, RDS, and S3—along with our users, their computers, and their browsers. The Virtual Private Cloud (VPC) provides a private network for AWS resources and defines private and public subnets to control access and provide a layer of security from external networks. Private subnets don't have direct internet access, so private information like our relational database remains hidden from public view.

The backend is hosted on individual instances on EC2 servers. Elastic Beanstalk (EB) scales the number of EC2 server instances based on user demand. This isn't just switching traffic to different active servers; it automatically deploys and tears down EC2 instances to preserve resources when they're not needed. EC2 allows quick deployment of virtual server instances within the VPC.

Relational Database Service (RDS) manages database configuration, maintenance, backups, and scaling. The RDS instance operates within a dedicated EC2 server inside the VPC, separate from Elastic Beanstalk-managed resources. This ensures persistent storage of data, protection against unintended resource termination, and secure database access by backend EC2 instances for reading and writing operations.

4.2.1 Directory Structure

Our project is divided into several high-level directories: BackEnd, FrontEnd, Documentation, and Prototyping.

The BackEnd directory houses all Python code and CSV files that either directly contribute to the program functionality, or were used to test the program's automatic assignment logic.

The subdirectories in the backend include Model, Controller, Util, and Data. The Model directory contains the data models used to describe each object of the program. The Util folder contains all factory scripts that populate these models. The Controller directory contains all logic to manipulate and make use of these objects for meaningful output. The Data directory deals with database transactions on behalf of the User data model.

At the top level of our application's frontend, the primary components are home, login, and registration. The login and registration components are simpler, each featuring a single component to handle both interface elements and interactions. The home component, however, is more complex and includes multiple nested sub-components to manage its richer functionality.

The nested sub-components within home include the conflict manager pane, conflict cards, details pane, file manager pane (deprecated), profile pane, and all three functional views (upload view, list/class view, calendar/room view).

4.3 Technical Issue Encountered

Challenges with migration of Angular UI to AWS when setting up the connections between the Elastic Beanstalk backend to work with the S3 static frontend. This is ultimately what prompted the use of an environment directory to configure the production URLs in the frontend code using base URLs that can be quickly tested by changing the one environment file instead of swapping out many URLs in each typescript file.

Issue with finding a framework to get the calendar view working, and ultimately hardcoding the calendar onto to the UI, which created some time crunch and delayed our ideal development timeline which led to buggy behavior and left us with no time to implement interactivity.

Implementing predictive logic for auto-assignments took a lot of prototyping of CSV fields to find the most reliable ways to use the data.

Some confliction in our team effort to create CSS for each component, and remediating those development styles took a considerable amount of time.

Consistent database behavior took several days and contributed to delay that prevented us from incorporting assignment files to be displayed on the interface and to utilize the database as well.

5. Testing

5.1 Overall Testing Strategy

Our testing strategy focused on a combination of automated and manual testing to ensure the classroom scheduling platform functions correctly and meets the desired requirements. The aim was to validate the application's business logic, usability, and performance in real-world scheduling scenarios. Given the nature of the project, we divided the testing into the following key areas:

- Unit Testing – To ensure individual components in the application work as expected.

Integration Testing – To verify that different parts of the system work together seamlessly.

- End-to-End Testing – To simulate real-world user interactions, ensuring that the entire workflow functions as intended.
- Manual Testing – To test user interaction, usability, and edge cases that automated tests might not cover.
- Performance Testing – To assess how the application handles heavy data loads or many concurrent users.

This approach helped us catch issues early, maintain a high quality of code, and ensure that the platform could be used efficiently by end users.

5.1.1 What We Planned to Use

Initially, we had a comprehensive plan for using a variety of testing tools that covered both frontend and backend testing. The following tools were identified at the start of the project:

- Jest or Jasmine (Angular Unit Testing): For testing individual Angular components to ensure proper rendering, input handling, and service interactions.
- Selenium (End-to-End Testing): To simulate complete user workflows such as file uploads, room assignment changes, and conflict resolution processes.
- Pytest (Backend Unit Testing): To validate backend logic, data models, and API endpoints built with Python/Flask.
- Cypress (UI Regression Testing): To ensure that the user interface remains consistent and functional across updates.

These tools were selected based on their ability to provide comprehensive test coverage and ensure the application's stability during development.

5.1.2 What We Ended Up Using

As development progressed, the tools used were adapted based on their suitability for the project and the team's familiarity. Here's what we ended up using:

- Jasmine (Frontend Testing): Initially, we planned to use Jest, but due to better integration with Angular, we chose Jasmine for frontend unit testing. Jasmine was employed to test component creation, service logic, and mock API responses.
- Pytest (Backend Testing): Pytest was heavily used for backend testing, particularly to validate Flask routes, API interactions, and logic related to CourseSection and Classroom objects. It allowed us to quickly identify and fix bugs in the backend logic, and its simplicity helped streamline the testing process.
- Cypress (End-to-End Testing): We replaced Selenium with Cypress due to its better support for Angular, ease of integration, and built-in debugging tools. Cypress simulated realistic user behavior, testing full workflows including login, file uploads, conflict resolution, and data export. Its snapshot features made identifying visual regressions easier, improving the user experience.

5.2 Realistic Test Data

To ensure that our testing covered real-world scenarios, we used realistic test data obtained from multiple sources:

- Historical Data: We collaborated with the Peter Kiewit Institute (PKI) to obtain authentic course scheduling data from past semesters. This data included time slots, room capacities, and instructor assignments. This gave us a practical understanding of scheduling issues and helped simulate typical scheduling conflicts.
- Edge Cases: We manually generated datasets containing complex scenarios, such as overlapping classes, incorrect room assignments, and unusual scheduling patterns. This helped us validate the system's robustness and its ability to handle exceptional cases.

5.3 Tests Conducted

We conducted the following tests throughout the development phase:

- Backend API Tests: Pytest was used to validate the functionality of backend API endpoints, ensuring that data was correctly processed and responses were returned as expected. The tests included:
  - Verifying data integrity when loading, saving, or updating CourseSection and Classroom objects.
  - Checking that conflicts were correctly detected and flagged.
  - Ensuring that API endpoints returned correct status codes and responses under various conditions.
- End-to-End Workflow Tests: Cypress scripts were used to test complete user workflows. Key scenarios included:
  - Login: Ensuring users could authenticate and access the platform.
  - File Uploads: Verifying that CSV files could be uploaded and the data was correctly processed.
  - Room Assignments: Testing the automatic room assignments and manual overrides.
  - Conflict Resolution: Ensuring that conflicts were detected and resolved as expected.
  - Data Export: Validating that the generated CSVs reflected the correct assignments and conflicts.
- Manual User Experience Tests: We performed manual tests to assess the platform's usability. This included navigating the interface, interacting with the scheduling tools, and ensuring that all features were intuitive and efficient for the classroom coordinators.
- Edge Case Tests: Manual tests specifically targeted rare or unexpected input scenarios, such as:
  - Empty or malformed CSV files.
  - Classes that required special room configurations (e.g., computer labs).
  - Overlapping or conflicting class schedules.

5.4 Tests Results

Testing yielded generally positive results across all areas:

- Backend Testing: Pytest helped us identify and resolve logical errors early. The backend logic was robust and capable of handling real-world scheduling conditions. We also found that the system correctly identified and flagged scheduling conflicts.
- End-to-End Testing: Cypress tests were invaluable in ensuring that the entire user workflow worked smoothly. It helped us detect UI regressions and functionality issues that arose from code changes. Key workflows, like class assignment and conflict resolution, performed well under testing.
- Manual Testing: Usability testing revealed a few areas for improvement in the UI/UX. These were addressed to ensure that the platform was user-friendly and intuitive. Performance tests identified some bottlenecks in handling large datasets, which were subsequently optimized.

5.5 Tests Not Conducted

Although we had a comprehensive testing plan, the following tests were not conducted due to time constraints or resource limitations:

- Cross-browser Compatibility Testing: While we tested on the most common browsers (Chrome and Firefox), we did not extensively test the application across a wide range of browsers or older versions.
- Load Testing: Although we performed performance testing, we did not conduct extensive load testing with a high number of concurrent users. This would have been useful to identify potential issues under heavy user load.
- User Acceptance Testing (UAT): While manual user experience testing was conducted, we did not carry out formal UAT with a larger group of actual classroom coordinators or other end users. This would have provided more comprehensive feedback on the platform's usability.

6. Summary

Trevor was our Project Manager and took team meeting notes. He also did a substantial amount of development on the Angular frontend views of the web application. Tyler was our implementation lead and did the vast majority of the backend development, frontend integration, and AWS migration. Fara and Haresh both took active roles in the testing of the frontend and backend. Fara and Honora both worked on data analysis, requirements analysis, and project research. Fara, Haresh, and Honora also contributed to some implementation tasks. Code milestone presentations were presented by Tyler, the testing presentation was performed by Haresh, and Honora did presentations over wireframes, goal modeling, and the demo poster. Honora designed the demo poster that will be submitted with the final project.

6.1 Outcome of Risks

Several risks were identified at the start of our project, and our experiences confirmed some of them while others proved less problematic than expected:

- Hosting Concerns: This risk did not become a major issue. While there was initial uncertainty about using UNO servers or Google Cloud, we successfully deployed the project using Amazon Web Services (AWS). This platform aligned well with our team's experience and provided reliable infrastructure for deployment.
- Inconsistent Input Data: This risk did not materialize. Contrary to our expectations, the input CSV files were mostly consistent in structure and content. This allowed for more streamlined parsing logic and fewer exceptions than originally anticipated.
- Lack of Long-Term Support: This remains a valid long-term concern. To mitigate this, we produced detailed technical documentation, included an exportable change log feature, and ensured the codebase is modular and readable to support future updates by new coordinators or developers.
- Changing Requirements and Future Scalability: This risk partially materialized. While there were no new room constructions or major curriculum overhauls during development, we did identify the need for future flexibility. As a result, we designed our data models and scheduling logic to support potential future constraints such as room features or department-based preferences.
- Insufficient Data: This risk somewhat affected our original approach. Although we received several historical spreadsheets, the volume and diversity of the data were limited. As a result, we shifted focus from predictive modeling to reliable rule-based

scheduling and included mechanisms for historical room usage to guide assignments.

- Restricted Access to Room Maps and Details: This risk did materialize. University policies prevented us from accessing detailed floor plans and room capability information. To work around this, we implemented simplified room tagging and created assignment logic based on inferred requirements from class attributes.
- Unanticipated Problem – AWS Setup Complexity: Although AWS was ultimately a successful deployment platform, the initial configuration—including EC2 access, database permissions, and networking setup—took significantly more time than expected.
- Late Addition of Critical Features: The request for a downloadable change log file came late in the project, requiring some backend logic changes and extra time to ensure it worked reliably across test cases.
- App Stability Across Browsers: During development, the application experienced login issues on certain computers when using Edge and Chrome—users were sometimes unable to authenticate despite valid credentials. This inconsistency made debugging and testing more difficult. However, after refining the application now logs in and functions correctly across all major browsers.

6.2 Milestone Summary

Milestone 1: Project Tasks and Timeline Scheduled. Create basic project structure including data models and prototype logic to display them. Create Angular frontend project structure and a basic homepage and login page.

Milestone 2: Working Prototype for uploading a CSV on the web platform, assigning classes through rudimentary logic, and outputting a CSV with the class allocations. Allocation conflicts and data input errors should be identified. Create prototypes for testing and have a plan of what tests are reasonable to implement.

Milestone 3: Implement sorting logic that seeks best fit scenarios of classes that aren't assigned to rooms. User authentication in place. Allocation conflicts and data input errors should be displayed to the user. Angular frontend components should include: top-left navigation/account box, middle-left file details panel (based on current selection), central component to navigate to different views; list view should be close to finished, other calendar views should be in-progress. State management should have a model, and session management should be functional. Relational database is designed.

Milestone 4: Angular front-end central component views are functional. State management is functional for undo and redo. Design should be approaching final state. Input validation should be functional. Bottom-left component file manager should be functional. Conflict manager in right pane should be functional. AWS implementation should be explored and in basic functioning stages. Relational database is functional.

Milestone 5: Program is tested and deployed in AWS production environment. Design is finalized. Default user accounts are defined and ready for hand-off to client.

6.3 Future Extensions

Based on our experience, the following future improvements are recommended:

Expand undo/redo state history and improve state rollback accuracy.

Implement real-time collaborative editing between users.

Add analytics for room utilization and schedule efficiency.

Integrate with UNO's student system for automated data syncing.

Extend calendar view with drag-and-drop editing.

Implement more advanced machine learning models for predictive scheduling.

6.4 What we Learned

Tyler

I learned a lot about object interaction, what to expect from object roles when designing models, and how greatly they can benefit the organization of entry-point code. It was great to practice creating modules with well-defined responsibilities governed by the namespace. This is the largest project I've worked on and I'm very happy to have developed the project to a very close representation of our original plan and scope.

I've learned that playing to my strengths works to a degree but to push forward to the end-goal and sometimes all you can do is try to minimize the time spent on new topics or

deadends. Thorough research for libraries and development strategies for your specific use-case has helped immensely to avoid rework and give us time to fit in important features. It's important to sprint when an idea is clear so you have time to brainstorm on more difficult aspects of the project because time is going to be the most valuable resource.

I really enjoyed having a project sponsor and key stakeholders to guide the vision of the project rather than focusing on using certain educational concepts like design patterns or restrictions that prevent you from solving a problem in any way you can to fit the requirements.

I found experience gained in any project or any field is a multiplier for how effectively you can communicate your ideas to stakeholders and teammates, including photoshop, video software, diagram software, and any failed project that gave me some small experience to use when it counts.

Trevor

I learned a lot about what it takes to code in Angular. Both in the framework itself but also writing in typescript which was a new experience. Angular itself wasn't a hard concept to understand as it follows other frameworks such as Django which i have used in the past but provided much greater usability when it came to implementation.

Additionally, i learned a lot more about working in Flask, something that i have had limited usage with in the past but was still enjoyable to get to dive deeper into with a higher complexity project.

I learned that when tasks are broken up into manageable chunks and split into a cohesive group to work on a project it sometimes becomes difficult to understand the whole scope of the project you are working on which requires you to need to take a step back sometimes and just study your peers work to better understand how it all ties into the bigger picture of what you are working on.

It also becomes much easier to work on projects of this size as a group when everyone has a similar idea of what the project is supposed to look like. weather that be the sponsor giving direction, or the group just coming together to spit ball ideas it definitely saves time in the long run to take the time and get a clear picture right away.

Fara

I developed a deeper understanding of backend testing strategies, focusing on using Pytest to validate API endpoints and data models. Writing tests to ensure that our backend logic was accurate and resilient to various data conditions was a significant learning experience.

I learned how to use Cypress for end-to-end testing of the user workflows. I was able to simulate real-world scenarios ensure that the application behaved as expected.

Writing clear and concise documentation was part of my responsibilities. This included documenting the functionality of the backend, testing processes, and the reasoning behind certain design choices. I gained a better understanding of how to create documentation that serves both the development team and end users.

Although I did not play a central role in the actual coding and implementation, following the development process closely helped me learn about the challenges and trade-offs that arise during implementation. I gained an understanding of how decisions made during the planning and design stages affect the final product and how important it is to maintain flexibility in addressing unanticipated problems. Observing the coordination between the frontend and backend teams allowed me to appreciate the value of clear communication, version control, and the iterative development process.

Honora

I learned the basic workings of the Angular frontend development protocol and got a good refresher on developing in HTML and CSS. I did not play a critical role in the development process but the small implementation tasks I took on gave me a stronger desire to work on some projects of my own this summer and learn more about the integration between Python and Angular.

I was relatively new to video recordings before this semester and was able to quickly find and learn to not only record but also edit my presentations. I was able to use these skills in my final presentation for another class this semester as well. I was also able to flex my

design skills with the final presentation poster which was another new task for me. I'm very excited to see it printed out at our demo day event.

Finally, I learned how to better collaborate with a group in a remote setting. I experienced this last semester in Human Computer Interaction, but this project was much larger in scale, and we were a lot more dependent on our Trello board and communication stream to keep each other in the loop. I think we did a great job collaborating this semester and meeting all our project requirements.

Haresh

I was able to leverage my existing Angular experience in the project by designing the Home page. I also utilized Angular Material libraries in the design and learned more about design strategies and CSS techniques.

Additionally, I had the opportunity to dive deeper into creating effective designs, which was a valuable experience for me since it was an area I previously lacked confidence in. I also wrote some Angular tests for the services we used. This experience helped me better understand

how to mock service calls in Angular and how to effectively use the Jasmine testing framework.

7. Local and Global Impacts

Computing applications present a wide variety of benefits to society, both on a local and global level. Scheduling applications, in general, can be used to manage your appointments and personal time, plan events, coordinate your team's time and activities in the workplace, and more.

On the local level, a classroom scheduling application can provide the following benefits:

- Reduced workload on administrative staff

  One of the key goals in developing our application was to reduce the workload on the classroom coordinator. This position is currently tasked with allocating PKI classrooms manually so developing an automated solution would substantially free up their time to allocate to other key tasks in their workload. The conflict management feature of our application helps mitigate scheduling errors, so less time is spent identifying them before they are resolved.

- Improved resource utilization and increased flexibility for teachers and students

  Classrooms and equipment (labs, large classrooms, projectors) are allocated more efficiently. This reduces overlap and downtime and ensures no class is without the required equipment to the maximum degree.
  It is also easier to accommodate any scheduling preferences made by staff, as real-time updates allow the classroom coordinator to make faster changes when necessary. Having a centralized platform for performing all these tasks will help with communication with the registrar and any other stakeholders and will help to reduce confusion in the scheduling process. Additionally, communication to the registrar is standardized by using the exportable Change Log feature instead of taking an additional step to summarize all requested changes to present to the registrar.

- Environmental benefits

  Optimizing the classroom schedule and appropriately allocating the classrooms can reduce any unnecessary lighting of unused rooms. While most of UNO's lights are sensor-activated, this could potentially benefit other universities or businesses that do not have this upgrade.

The global impacts and benefits of a classroom scheduling application are similar, but on a larger scale:

- Increases educational efficiency and improves access to education

  Optimizing the scheduling of classes allows the university to minimize idle classroom time, therefore allowing them to potentially accommodate more students. Minimizing this idle

time also allows for a better rotation of resource use in specialized classrooms so there is little need for building new infrastructure.

Administrative overhead needs are reduced, which allows universities across the board to free up resources for teacher and student support.

- Enables standardization and data-driven educational planning

  Data aggregation can inform policy decisions at the local, national, and global levels. This is especially useful for universities with online and international/exchange platforms. Global institutions can maintain a level of consistency and identify trends in resource allocation, teaching methods, and attendance.

8. Appendices

8.1 Relevant URLs

Project Tracking System: Capstone Project Trello Board | Trello
Version controlled code repository: CSCI4970-Capsoft-Solutions-Project
Shared folder: Tyler Porter - OneDrive
Group Communication: Discord
API Documentation:
Live Demo: capsoftsolutions.com

8.2 Setup/build/installation/deployment instructions

Our program has also been adapted to run locally and in production seamlessly. Any failed database connections results in fallback logic that allows the user to login with admin/password. This was achieved by using a base URL in the frontend, and using configuration files for this base URL that differ depending on how the FrontEnd is built.

To build the project for production and populate the static files to be uploaded to AWS S3, run: ng build --configuration=production -Navigate to the FrontEnd directory and find the "dist" folder which contains all necessary files to upload.

To build the frontend locally, navigate to the frontend directory and run: npx ng serve

Complete Instructions for running locally:

1. You may need to install dependencies listed in the previous release notes such as flask and pandas.
2. Open a terminal in your ide, or command line interface, and run the application.py in the BackEnd directory of the project.
3. Install NodeJS as necessary on your computer.
4. Open a new terminal and navigate to our projects FrontEnd folder. Run commands: 'npm install' and 'ng add @angular/material'.
5. Build and run the frontend with 'npx ng serve'.
6. Click the link in the terminal to load the webpage on your local.
7. Upload an input CSV file from the interface.
8. Navigate to http://localhost:4200 to navigate to the landing page.

8.3 Team Weekly Meeting Notes

CSCI 4970 Capstone Project

01.23.25 Meeting Notes
Meeting Location: Class Meeting
Notes:

- General Discussion, Groups are still being assigned.
- Project Documentation
- Final Expectations
- Grading Rubric
- Version Control
- Trello Board

Tasks Assigned:

- Shared Project Folder

CSCI 4970 Capstone Project

01.26.25 Meeting Notes
Meeting Location: Discord meeting
All in Attendance
Notes:

- Group Roles
- Project Specification (Angular with Python Backend)
- Discussed Team Roster + Shared Project Folder & Context Document

Tasks Assigned/Worked on During Meeting:

- Project Plan & Project Proposal
- Individual Journals
- Capsoft Solutions

Tasks Completed:

- Team Roster + Shared Project Folder
- Created Repository for project

CSCI 4970 Capstone Project

01.26.25 Meeting Notes
Meeting Location: Discord Meeting

Topics:
1. Roles
-Project Manager: Trevor
-Client Liaison: Honora
-Requirements Analyst: Fara
-Architect/Technical Lead: Tyler
-Lead Researcher: Honora
–Scribe: Trevor
-Quality Assurance: Fara

CSCI 4970 Capstone Project

01.28.25 Meeting Notes
Meeting Location: Client Meeting
Subject: First Meeting with Client/Sponsor
Attendance: Everyone
Haresh Logged off Early

Project Discussion:
Regular Meeting Time: 4:00 pm on Tuesday as of Today
Link to the class size information:
IS&T Classroom, Labs, & Conference Rooms | College of Information Science & Technology | University of Nebraska Omaha.

No Classroom Coordinator to work with.
Google Cloud Platform to host website
Email distribution has been tabled for the time being.
Will not be using student interface system.
Need more spreadsheets without filters
getting more information about specialized equipment for certain classes

Spreadsheet from the registrar explaining all courses
Time Slot/Date
Room Number
Special Notes
No priority currently in place (Potential idea after application is running)
Need to be able to allow correct allocation of students into classrooms depending on variables such as computer access and lab availability.

Some classes are specialized and will never move, those should be the easiest to fill in first.
Cross Listed courses or courses that are able to occupy the same room
Figure out a way to deal with TBA assignments
No conflictions between time (REQUIREMENT)
Distance Education classes
Partially Online should be considered Online for the time being

CSCI 4970 Capstone Project

01.28.25 Meeting Notes
Meeting Location: Class Meeting
Attendance: All in attendance

Tasks Discussed:
Group Exercise
Context Documentation

Tasks Completed:
Group Exercise

CSCI 4970 Capstone Project

01.30.25 Meeting Notes
Meeting Location: Class Meeting
Attendance: All in attendance

Tasks Discussed:
Touched base on Group Exercise and other submittable due next week
Context Documentation

Tasks Completed:
Group Exercise

CSCI 4970 Capstone Project

02.02.25 Meeting Notes
Meeting Location: Discord Meeting

Attendance: All in attendance

Tasks Discussed:
Got Hareesh added as a collaborator in the GitHub
Presenting the Client Context (Honora as client liaison)
Starting Wireframing Group Exercise
Starting Context Document

Discussed GitHub repository for project (Everything in one)
Discussing Development Ideas (Cleaning Data), Angular
Class Breakdown and grouping?

Tasks Assigned:
Presenting the Client Context (Honora as client liaison)
Wireframing Group Exercise
Starting Context Document
Class Diagram(Tyler)

Tasks Completed:
Project Proposal
Project Plan
Context Document

Questions for Sponsor:
Google Cloud for deployment or AWS (Group is leaning toward AWS)?
Update on Spreadsheet Data?
Should our project be implementing a database (future developments)?
If the school has a scheduling method?
How has there not been a prior solution to this problem?
Are j term classes guaranteed to not conflict with regular session classes, and can therefore be assigned to an otherwise claimed room?
The Room (O) column is filled out on our spreadsheet. I'm guessing this won't be filled out when the class coordinator inputs the spreadsheet into our program? Otherwise, what prevents us from assigning rooms according to the column?

CSCI 4970 Capstone Project

02.04.25 Client Meeting

Meeting Location: Client Meeting
Attendance: All in attendance

Still need more data

Topics Discussed:
Answers to Questions:
Google Cloud coupons
No new data
Revisit implementing database
Timeline

Will not have column (O) to work with
Do not need to worry about J-Term Classes (they will be going away)

More Complete list of advanced requirements classes:
Durham School Scott Campus Facilities | College of Engineering | University of Nebraska–Lincoln

CSCI 4970 Capstone Project

02.04.25 Meeting Notes
Meeting Location: Class Meeting
Attendance: All in attendance

Topics Discussed: Google Calander style of front end.
Updating the Trello board to include developmental tasks.

Tasks Assigned:
Learn more about Angular (Group)
Develop back end of Program (Tyler and Hareesh)

Tasked Completed:
Wireframe

CSCI 4970 Capstone Project

02.06.25 Meeting Notes

Meeting Location: Class Meeting
Attendance: All in attendance

Topics Discussed:
Goal model(Trevor)
Wireframe(Tyler & Hareesh)
Turning in our context document
Discussed next week's object modeling lecture and group exercise.
This week starts more group work in classes

Tasks Completed:
Context Doc (Again)

CSCI 4970 Capstone Project

02.09.25 Meeting Notes
Meeting Location: Discord Meeting Sunday
Attendance: All in attendance

Topics Discussed:
Goal model(Trevor)
Wireframe(Tyler & Hareesh) with document text
Requirements document (Everyone)
State management
Back end management

Tasks Completed:
Requirements document
Goal model (Will need updated for final submission)

Questions: Respecify the usage and need for user authentication

CSCI 4970 Capstone Project

02.11.25 Client Meeting
Meeting Location: Client Meeting
Attendance: All in attendance

No update on spreadsheet
R stands for Thursday in the spreadsheet
Leaning towards Preference for authentication on the program

Topics Discussed:
Update Project Plan:
Roles
No data to analysis or learn from
    Risk analysis missing data from spreadsheet
    Spreadsheet itself changing (additional columns)
    Mapping of Data

Trello Board:
Git hub tasks
break up into workable tasks(should only take a day to a week max)
parse data
import csv
logic for pulling data

Goal Model from requirements outline:
Easily Organize becomes main goal

CSCI 4970 Capstone Project

02.11.25 Client Meeting
Meeting Location: Class Meeting
Attendance: All in attendance

Discussion:
Updating Documents based on Client Meeting
Completing group exercises along with creating our own object model and logical decomposition

CSCI 4970 Capstone Project

02.16.25 Meeting Notes
Meeting Location: Discord Meeting
Attendance: All in attendance

Discussion:
Working towards milestone 1
GQC Group exercise done

CSCI 4970 Capstone Project

02.18.25 Meeting Notes
Meeting Location: Client Meeting
Attendance: All in attendance

Discussion:
No new update for the new Data
Updating the Trello Board

CSCI 4970 Capstone Project

03.04.25 Meeting Notes
Meeting Location: Client Meeting
Attendance: All in attendance

Discussion:
Deployment view: needs to include the software somewhere in the diagram
Development View Diagram: Good (for now)

CSCI 4970 Capstone Project

03.04.25 Meeting Notes
Meeting Location: Class Meeting
Attendance: All in attendance

Discussion:
Work on Mid Semester Project Report as well as Development to the project itself
UI and Backend

CSCI 4970 Capstone Project

03.06.25 Meeting Notes

Meeting Location: Class Meeting
Attendance: All in attendance

Discussion:
Work on Mid Semester Project Report

CSCI 4970 Capstone Project

03.11.25 Meeting Notes
Meeting Location: Class Meeting
Attendance: All in attendance

Discussion:
Testing (we are okay to use our actual test cases for our project)
Fall 2025 data
Moving Milestone 2 to be due past spring break.
Manual View

Haresh working of video
Trevor upload of csv and Api backend
Tyler finishing up backend
Honora working on json file (separate)
Fara fixed excel sheets and fixed references (added)

CSCI 4970 Capstone Project

03.13.25 Meeting Notes
Meeting Location: Class Meeting
Attendance: All in attendance

Discussion:
Meeting with Sponsor cancelled unless we create a meeting.

Everyone continues project work based on the last meeting.
Honora: Moved Json script into Prototype folder (Still working)
Still working on test cases and creating the login UI

CSCI 4970 Capstone Project

03.25.25 Meeting Notes
Meeting Location: Client Meeting
Attendance: All in attendance

Discussion:
Working Towards:
Transferring to AWS
User authenticating
Interactive Room by Room View

CSCI 4970 Capstone Project

03.25.25 Class Meeting
Attendance: All in attendance

Tasks Completed or in Progress and Discussion on:
(Trevor) Upload and displaying current data
(Fara) fixed excel sheets completed still working on references
(Honora) Json script into Prototype folder complete pushing into branch
(Tyler) More Assignment Logic, Video for Milestone 2
(Haresh) re-design of login page

New Tasks:
Completed list view and other views
AWS change

CSCI 4970 Capstone Project

03.27.25 Class Meeting
Attendance: All in attendance

Tasks Completed or in Progress and Discussion on:
(Trevor) research on interactive list views
(Tyler) Conflict management update working on authentication
(Honora) push Json files to github
(Fara) working on references working on backend testing and autodocs

(Haresh) working on login

CSCI 4970 Capstone Project

04.01.25 Client Meeting
Fara and haresh delayed
Attendance: All in attendance
Brief Project Update

CSCI 4970 Capstone Project

04.01.25 Class Meeting

Tasks Completed or in Progress and Discussion on:
Fara (Documentation) Sphinx
Trevor(initialization of data on homepage)
Honora(Working on components)
Tyler(getting front end put on aws)

CSCI 4970 Capstone Project

04.01.25 Client Meeting
Attendance: All in attendance

Tasks Completed or in Progress and Discussion on:
Fara (Backend Testing
Trevor(initialization of data on homepage)
Tyler(getting front end put on aws)
Heresh(Login page complete)

CSCI 4970 Capstone Project

04.10.25 Class Meeting
Attendance: All in attendance

Tasks Completed or in Progress and Discussion on:
Fara (Backend Testing)
Trevor(Section view and started Calander view helped with file details)

Tyler(Finished File Details)
Heresh(Login page complete)

CSCI 4970 Capstone Project

04.15.25 Client Meeting
Attendance: Tyler and Haresh Absent

Pytest in trello
Meeting with classroom coordinator (Schedule a time to meet)

CSCI 4970 Capstone Project

04.15.25 Group Meeting
Attendance: All in attendance

Tasks Completed or in Progress and Discussion on:
Honora(Poster, maybe working on components)
Tyler(File Details and Searching for classes in Section View)
Haresh(Input validation)
Trevor(Calander View)
Fara(Pytest)

CSCI 4970 Capstone Project

04.22.25 Client Meeting
Attendance: All in attendance

Tasks Completed or in Progress and Discussion on:
Honora(front end for Edit tab)
Tyler(Back end logic for Edit tab in file details)
Haresh(Input validation)
Trevor(Calander View)
Fara(Finished up testing and began work on documentation)

CSCI 4970 Capstone Project

04.24.25 Client Meeting

Attendance: All in attendance

Tasks Completed or in Progress and Discussion on:
Honora(front end for Edit tab)
Tyler(AWS is not working)
Haresh(Finished Input validation for registration)
Trevor(nearly completed Calander View)
Fara(finish backend documentation)

CSCI 4970 Capstone Project

04.29.25 Client Meeting ` Haresh Missing
Attendance: All in attendance

Tasks Completed or in Progress and Discussion on:
Honora(Poster)
Tyler(AWS works Finishing Details)
Haresh(was gone)
Trevor(Small tweaks to logic and design)
Fara(end to end testing using cyprus)

CSCI 4970 Capstone Project

05.1.25 Client Meeting ` Haresh Missing
Attendance: All in attendance

Tasks Completed or in Progress and Discussion on:
The entire group discussed and began discussion on the final report.

8.4 Individual Journals

8.4.1 Tyler Porter

<u>05.08.25</u>

Completed Last Meeting:

1. Implement frontend in production env.

2. Frontend column sorting function
3. AssignmentFile needs a filename field, which has a default value (term, year, timestamp, num)
4. Make sure remote synchronous classes (which have room assignments and meeting times even though students are present) are being adjusted for any edits... just like cross-listed classes are.
5. Adjust the div color for login and registration from white, to the off-white color used in home.
6. Add checkboxes on the upload component so the user can include or omit certain scheduling logic in case one is unreliable.
7. Add an upload methodology for new "finalized" CSVs so the user can add subsequent schedules/assignments to the CSV training to data to constantly improve the historical scheduling, which is the most reliable strategy. Include ability to revert to default, and omit or include certain CSVs from training data in case major changes happen to the facility that makes old training data a liability.

Notes for Current Meeting:

- Milestone 5 Tasks

Tasks to Complete:

1. download csv export
2. authentication
3. Crosslistings should be assigned to a room when one of them is assigned. Changes to this should be changed on all crosslistings

---

<u>05.06.25</u>

Completed Last Meeting:

1. Implement frontend in production env.
2. Frontend column sorting functions. (Do we need to sort with frontend and redo how data is populated into columns or keep formatted strings?)
3. AssignmentFile needs a filename field, which has a default value (term, year, timestamp, num)
4. Incorporate Haresh's frontend input validation logic.
5. Fix conflict manager horizontal scaling, which incorrectly remains the same size with reduced resolution, which narrows the other two columns.
6. Make the button font smaller for export and change log, so change log never gets put on two lines which makes it taller than the export button.

7. Incorporate "Capsoft" somewhere on the login/registration/home page somehow... maybe small text?
8. Apply the capsoft domain address to our AWS implementation.

Notes for Current Meeting:

- Milestone 5 Tasks

Tasks to Complete:

1. download csv export
2. authentication
3. Crosslistings should be assigned to a room when one of them is assigned. Changes to this should be changed on all crosslistings

---

05.01.25

Completed Last Meeting:

1. Implement frontend in production env.
2. Frontend column sorting functions. (Do we need to sort with frontend and redo how data is populated into columns or keep formatted strings?)
3. AssignmentFile needs a filename field, which has a default value (term, year, timestamp, num)?

Notes for Current Meeting:

- Milestone 5 Tasks

Tasks to Complete:

1. download csv export
2. authentication
3. Crosslistings should be assigned to a room when one of them is assigned. Changes to this should be changed on all crosslistings

---

04.28.25

Completed Last Meeting:

1. Section view search bar and necessary route functions.
2. "Edit" tab on the details pane. Route functions for setting data.

Notes for Current Meeting:

- Milestone 4 Tasks

Tasks to Complete:

1. AssignmentFile needs a filename field, which has a default value (term, year, timestamp, num)?
2. Loading icons when things hang.
3. Finish prototype DB logic. Ensure its working in production env.

04.25.25

Completed Last Meeting:

1. Section view search bar and necessary route functions.
2. "Edit" tab on the details pane. Route functions for setting data.

Notes for Current Meeting:

- Milestone 4 Tasks

Tasks to Complete:

1. AssignmentFile needs a filename field, which has a default value (term, year, timestamp, num)?
2. Loading icons when things hang.
3. Finish prototype DB logic. Ensure its working in production env.

04.22.25

Completed Last Meeting:

1. Update the AssignmentFile to have lists for active and ignored conflicts.
2. Fix route functions to activate or ignore conflicts.

Notes for Current Meeting:

- Milestone 4 Tasks

Tasks to Complete:

1. Section view search bar and necessary route functions.
2. "Edit" tab on the details pane. Route functions for setting data.
3. AssignmentFile needs a filename field, which has a default value (term, year, timestamp, num)?

04.15.25

Completed Last Meeting:

1. Frontend details component and conflict manager component

Notes for Current Meeting:

- Milestone 3 wrap-up

Tasks to Complete:

1. Authentication
2. AWS BackEnd
3. DB

04.10.25

Completed Last Meeting:

1. Frontend details component and conflict manager component

Notes for Current Meeting:

- Milestone 3 wrap-up

Tasks to Complete:

1. Authentication
2. AWS BackEnd
3. DB

04.03.25

Completed Last Meeting:

1. Studied for MFS

Notes for Current Meeting:

- Delay development to study

Tasks to Complete:

1. Pass MFS
2. Authentication
3. AWS BackEnd
4. DB

04.01.25

Completed Last Meeting:

1. AWS Backend Migration complete
2. Studying for MFS

Notes for Current Meeting:

- Discussed Angular component structures
- Honora will work on angular components

Tasks to Complete:

1. Studying for MFS
2. AWS Frontend migration exploration

03.27.25

Completed Last Meeting:

1. Conflict mgmt logic improvement
2. Started authentication
3. Started AWS migration to production env

Notes for Current Meeting:

- Fara will working on Doxygen
- Honora will work on angular components

Tasks to Complete:

1. AWS migration
2. Authentication

03.25.25

Completed Last Meeting:

1. Assignment logic
2. Conflict detection improvements
3. Export functionality

Notes for Current Meeting:

- Mid Semester-Project Report
- UI and Backend

Tasks to Complete:

1. User interface
2. Backend (finish classroom data model)
3. State model ideation

03.06.25

Completed Last Meeting:

1. Initial state model
2. Deployment view changes
3. Scheduling logic to maintain schedules for each room

Notes for Current Meeting:

- Mid Semester-Project Report
- UI and Backend

Tasks to Complete:

1. User interface
2. Backend (finish classroom data model)
3. State model ideation

03.04.25

Completed Last Meeting:

1. Information displays on frontend from backend
2. Login page is created
3. Created classroom data model

Notes for Current Meeting:

- Development view discussion
- Deployment view discussion

Tasks to Complete:

1. User interface
2. Backend (finish classroom data model)
3. State model ideation

02.16.25

Completed Last Meeting:

1. Refactored data models (to instantiate with dictionaries)
2. Goal Modeling progress

3. Object model progress
4. Basic CSV data extraction

Notes for Current Meeting:

- Logical decomposition
- Updated requirements document after sponsor meeting

Tasks to Complete:

1. Research classroom attributes
2. Backend development (Produce output for frontend)

02.11.25

Completed Last Meeting:

1. Requirements document
2. Goal Modeling progress

Notes for Current Meeting:

- CSV parsing continuity risks
- Team role reassignment
- Trello board organization
- Authentication is needed

Tasks to Complete:

1. Goal Modeling
2. Backend development (CSV parsing)

02.09.25

Completed Last Meeting:

1. Context Document
2. Created instructional documentation for project setup
3. Selected AWS over Google Cloud
4. UX Design documentation

Notes for Current Meeting:

- Goal Model discussion
- Worked on Requirements document
- Completed Requirements document

Tasks to Complete:

1. Clarify scope of project with sponsor (authentication?)
2. Backend development

02.06.25

Completed Last Meeting:

1. Context Document
2. Created instructional documentation for project setup
3. Selected AWS over Google Cloud

Notes for Current Meeting:

- Discussed State Management feature
- Requirements Document
- State of backend

Tasks to Complete:

1. Develop backend
2. Requirements Document

02.04.25

Completed Last Meeting:

1. Expanded on instructional documentation in the project
2. Created main route file for Flask and state management model
3. CSV analysis complete
4. Wireframe

Notes for Current Meeting:

- Google Calendar style of frontend
- Angular frontend
- Special classes such as J-term are excluded from scheduling

Tasks to Complete:

1. Develop backend
2. Study Angular

02.02.25

Completed Last Meeting:

1. Finished project structure and environment
2. Created instructional documentation for project setup
3. Selected AWS over Google Cloud

Notes for Current Meeting:

- Context Document
- Wireframe for group project
- Discussed branch structure for repository

Tasks to Complete:

1. Context Document
2. Wireframes
3. Class Diagram

01.30.25

Completed Last Meeting:

1. Project structure and environment basically done
2. Analyzed CSV for parsing strategy

Notes for Current Meeting:

- Context Document
- Research AWS versus Google Cloud
- Special equip and other attributes of classrooms

Tasks to Complete:

1. Finish setting up project structure/environment
2. Context Document
3. Continue analyzing CSV

01.28.25

Completed Last Meeting:

1. Working on project structure
2. Worked on project plan

Notes for Current Meeting:

- PKI Classroom Coordinator responsibilities
- Need to update team roster
- Special equip and other attributes of classrooms

Tasks to Complete:

1. Finish setting up project structure/environment
2. CSV Prototyping

01.26.25

Completed Last Meeting:

1. Started outline for project proposal and plan
2. Created project folder
3. Submitted shared folder and Team Roster

Notes for Current Meeting:

- Discussed Team name: Capsoft Solutions
- Confirmed team roster (Haresh joins)
- GitHub Repo created

Tasks to Complete:

1. Create project structure and make initial commit to repository
2. CSV Prototyping

8.4.2 Trevor Ahlers

Jan 23rd

- Created Trello Board and filled out with Tasks and Assignments
- Updated Team Journal & Trello

Jan 26th

- Created Docs for Project Plan and Proposal. Began Filling them out
- Created Repo for Project
- Updated Team Journal & Trello

Jan 28th

- Began Research on Angular
- Work on Project Plan, Project Proposal, and context Document
- Updated Team Journal & Trello

Feb 2nd

- Work with wireframing
- Updated Team Journal & Trello

Feb 4th

- More Angular research
- Updated Team Journal & Trello

Feb 6th

- Working on Goal Model (briefly)
- Updated Team Journal & Trello

Feb 9th

- Finished Goal Model
- Requirements Document
- Updated Team Journal & Trello

Feb 11th

- Updated documents based on client meeting
- Updated Team Journal & Trello

Feb 19th

- Loging Page and Home Screen Initialized
- Updated Team Journal & Trello

March 4th

- API List view integrate initially created
- Project Report work
- Updated Team Journal & Trello

March 6th

- Work on Mid Semester Report
- Updated Team Journal & Trello

March 10th

- Working on getting upload to work for the homepage
- Updated Team Journal & Trello

March 17th

- Completed upload and display of most recent data
- Updated Team Journal & Trello

March 25th

- Researching Doxygen
- Updated Team Journal & Trello

March 27th

- Updated Team Journal & Trello
- Began work on creating API call to backend to initialize data on Frontend

April 1st

- Updated Team Journal & Trello
- Finished work on creating API call to backend to initialize data on Frontend
- Began Work on Section/List View Design

April 8th

- Finished work on Section/List View Design
- Began Work on Calander view
- Made Modifications to File details

April 10th

- Kept working on Calander view
- Made Slight Design and CSS tweaks
- Updated Trello and Notes

April 22nd

- Kept working on Calander view
- Made Slight Design and CSS tweaks
- Updated Trello and Notes

April 23rd

- Nearly Complete with Calander View CSS does not co-operate

April 24th

- Finish Calander View Didn't think it was getting done for a second there
- Small Tweaks made to logic and design
- Updated Trello and Notes
- Began work on Upload enhancement

April 25th

- Updated Trello and Notes
- Scrapped Upload enhancement

May 1st

- Began work on final turn-ins

8.4.3 Fara Abdulloev

May 6th

• Continued work on Final Report

• Added the Requirements section

May 5th

• Resolved merge conflicts and pushed completed Pytest unit tests to the dev-fara branch.

• Focused on compiling content for the Final Report, including testing methodology and architecture overview.

May 4th

• Worked on initial version of the Final Report, outlining project objectives, system features, and development progress.

•Reviewed previous documentation (proposal, requirements, wireframes) to integrate and reflect project evolution.

May 2nd

• Created and pushed a new branch for Cypress end-to-end tests.

• Wrote initial test cases simulating real-world user flows through the frontend.

• Debugged issues related to Git branch access and configured environment for E2E testing.

May 1st

• Participated in a structured team review of outstanding tasks and deliverables.

• Assigned documentation and testing responsibilities among team members.

• Used in-class work time to clean up the frontend interface and improve user experience consistency.

April 29th

• Weekly sponsor sync-up to report progress and receive final-stage feedback.

• Dedicated class time to polishing UI behavior, conflict reporting visuals, and assignment logic.

April 24th

• In-class development session to address small bugs in section editing and dynamic updates to the conflict list.

• Reviewed assignment data integrity after multiple conflict overrides.

April 23rd

• Relocated static HTML documentation files to the /docs folder to enable GitHub Pages auto-deployment.

• Resolved configuration errors in .yml workflow setup for automatic documentation publishing.

• Validated that API documentation renders correctly on GitHub.

April 20th

• Wrote automated frontend Cypress tests to validate UI behavior (e.g., login, upload, conflict resolution).

• Explored test case scenarios to simulate real user interaction and verify edge cases.

April 15th

• Finalized structure and content for the project report.

• Continued development of test cases for UI logic in a separate Git branch.

• Faced Git merge issues while working in parallel across multiple branches.

April 8th

• Held a productive in-class development sprint to complete Milestone 3 deliverables.

• Ensured integration of automated room assignment logic with CSV export.

• Weekly sponsor meeting to present latest implementation of conflict detection.

April 1st

• Successfully generated project documentation using Sphinx and published to a separate Git branch.

• Ensured all modules, classes, and functions contained standardized docstrings.

• Performed functional tests on the latest application build.

March 25th

• Researched Doxygen and Sphinx for auto-generating code documentation.

• Enhanced backend modules with structured docstrings.

• Weekly sponsor meeting focused on backend stability and data integrity

March 17th

• Minimal project contributions due to MFT preparation.

• Drafted backend testing cases and identified test scenarios for core logic components.

March 10th

• Researched pytest framework to implement backend testing and edge case validation.

• Weekly sponsor meeting reviewing recent logic updates and data quality assurance.

March 4th

• Weekly sponsor meeting

• Discussed progress and planned next steps

February 27th

• Wrote a custom Python script to parse JSON files from previous semesters for room assignment history.

• Began standardizing PKI classroom feature data (e.g., projector, HDMI, capacity) for learning input.

• Used class time for collaborative development and logic review.

February 18th

• Team meeting with sponsor

• Worked on project during class

February 16th

• Internal team meeting via Discord to clarify next deliverables and distribute development roles.

February 11th

• Finalized Project Proposal and Context Documentation, ensuring clarity for stakeholders.

• Weekly sponsor meeting to align vision and confirm upcoming milestones.

February 9th

• Team meeting on Discord

• Completed the Goal Model

• Finalized Requirements Document

February 4th

• Continued editing and refining the Requirements Document.

• Weekly sponsor meeting to review wireframe structure and interface planning.

• Studied examples of wireframes and UI kits for frontend planning.

February 2nd

• Began drafting Requirements Document, defining functional specifications and user expectations.

• Held team sync via Discord.

• Continued refining project plan and early logic models.

January 28th

• Expanded Project Proposal and added system context overview.

• Conducted research on the Angular framework for frontend development.

• Met with project sponsor to discuss initial direction and receive approval on early documents.

January 26th

• Initiated capstone project with team.

• Started drafting the initial Project Plan and Proposal.

• Outlined tentative roadmap for deliverables and set up team communication channels.

8.4.4 Honora Shanno

Week 16 (5/4-5/10)

Friday, May 9

9:30-9:45am

Final journal update for this week

Updated my journal section in shared folder and in final report

2:30-4:30pm

Demo day at PKI

Thursday, May 8

6-7:15pm

Class and Capstone work time

10:30-11:30pm

Recorded demo poster video

Edited video and uploaded to Capsoft shared folder

11:30-11:45pm

Updated my "what I learned section in shared document"

Tuesday, May 6

7:30-9am

Emailed Dr. Siy cancelling sponsor meeting for today

Started researching local and global impacts of classroom scheduling app

Added relevant URLs to Project Report appendix

Added journal headings for each team member to Project Report

Updated shared journal and added to Project Report

Updated communication transcript section with recent communications

12:15-12:55pm

Worked on local/global impacts

Updated communication transcript with Dr. Siy's reply

Added local/global impacts to Final Report

6-7:55pm

Class and Capstone work time

Tyler, Trevor and myself worked on Project report in Discord from 7-7:55

Week 15 (4/27-5/3)

Friday, May 2

7:45-8:45pm

- Finished transcribing email communications
- Added/formatted headings in final report, added communication transcription to appendices
- Working on Local and Global Impacts next

Thursday, May 1

6-7:15pm

- Class and Capstone work time
- Outlined final documentation requirements and our divide and conquer strategies
- Touched base and made sure we all know what we're responsible for as we approach Demo day

7:15-8pm

- Updated local journal, got behind and missed updating the last week and a half
- Still need to update shared journal
- Added new tasks in Trello (poster demo video, sponsor communication transcription for final report)

8:30-9:30pm

- Updated deployment model view on poster
- Obtained feedback on final sections
- Proofread entire poster
- Submitted poster for printing

10-10:45pm

- Began transcribing email communications with Dr. Siy to word doc for addition to final report

Wednesday, April 30

10pm-12am

- Continued work on the demo poster
- Uploaded screenshots Tyler sent me and reorganized design
- Filled in most sections and sent to group for feedback

Tuesday, April 29

4-4:30pm

- Sponsor meeting with Dr. Siy for updates and final requirements
- Converted and uploaded meeting video to group folder

6-7:15pm

- Class and Capstone work time

9:30-11:00pm

- Went over demo poster content requirements and design quidelines
- Chose a poster template from Canva and started work on poster. Filled out main headings and did first draft of design/layout

Week 14 (4/20-4/26)

Thursday, April 24

6-7:15pm

Class and capstone work time

Tuesday, April 22

3:50-4:35pm

Discussed meeting possibilities with group and returned Dr. Siy's email

Met with Dr. Siy (4-4:25)

Converted mp4 recording of meeting and uploaded to shared folder

6-7:15pm

Class and capstone work time

Sunday, April 20

7:45-8pm

Made sure changes were pushed to my branch for Tyler to work on backend Edit tab

Fiddled with .css again to try and resolve double scroll bar

Week 13 (4/13-4/19)

Saturday, April 19

9-10:30pm

Cleaned up extraneous code from css file

Worked on getting scrollable object to stay within the bounds of the course details component

Committed and pushed changes to my branch

Friday, April 18

10-11:30pm

Worked on Edit tab

Added labels and text boxes for all fields to edit to a form on component

Thursday, April 17

2-3:30pm

Picked implementation task from our TODO list, updated Trello tasks

Implemented Edit tab on Course Details pane

6-7:15pm

Class and project work time

9-10:45pm

Checked Tyler's updates to conflict cards

Attempted Edit tab with Tyler's new changes, could not get things working with switchTab

Reverted to mat-lib tabs

Updated personal journal, local copies and shared copies

Tuesday, April 15

3:30-4pm

Updated personal notes and personal Trello tasks

Taking point on public demo poster and will look forward to requirements for final report

4-4:25pm

Trevor, Fara and myself met with Dr. Siy for sponsor meeting
Uploaded meeting recording to project folder
6-7:15pm
Class and project work time
Reviewed TO DO list for milestone 4
Merged and updated local project to milestone 3 dev branch

Week 12 (4/7-4/12)

Friday, April 11
8-8:30pm
Reviewed milestone submission videos
Thursday, April 10
6-7:15pm
Class and milestone 3 work time
Tuesday, April 8
4-4:25pm
Met with Zach and Dr. Siy (sponsor meeting) from 4-4:15
Project Summary in final report—talk about what changed vs what had been planned (ie. Accelerated AWS migration moving to milestone 3, etc)
Waited for meeting to render, uploaded Zoom recording to shared folder
6-7:15pm
Class and milestone 3 work time
Monday, April 7
9:05-9:20pm
Updated Trello task
Updated personal journal and migrated updates to shared journal
Pulled any available updates to my dev branch

Week 11 (3/30-4/6)

Thursday, April 3
6-7:15pm
Class and Capstone work time
Redefined next personal code task
Tuesday, April 1
2:30-3:00pm
Worked on learning how to run FrontEnd on my machine
Ran npm install in project directory and npm install –g @angular/cli
4-4:15pm
Group met with sponsor, weekly check-in
Uploaded video recording to shared folder
6-7:15pm
Class and Capstone work time

Monday, March 31

7:50-8:10am

Synced branch with dev branch

Merged and pushed json parsing file to main dev branch

Updated shared journal with input from my local journal

Week 10 (3/23-3/29)

Thursday, March 27

6-7:15pm

Class and Capstone work time

Wednesday, March 26

8:45-9:45pm

Uploaded Tues sponsor meeting recording to group folder

After discussion with group yesterday we will print historical output and analyze it manually

Updated the script to parse a file with department codes, one with course sections and locations, and one with course sections, locations, and all course info (to potentially look at data for class capacities)

Cleaned out test lines, updated documentation in script, checked output files were what I was expecting

Committed and pushed 5 files to my dev branch (script, json, and output files)

Updated Trello

Tuesday, March 25

2:00-2:15pm

Finished transferring and formatting journal

Uploaded journal to group folder

4:00-4:15pm

Team meeting with sponsor

4:15-4:45pm

Worked on pulling out classroom capacity, need to figure out how to JUST get the PKI classes

Pondering returning the data to .json format or what format would be best?

Looked at returning data as excel instead of text, need pandas info

6:00-7:15pm

Class and capstone work time

Haresh was out sick for class; the rest of us discussed upcoming deliverable (Doxygen), Milestone 3, and what all we will be working on for the next couple days

Monday, March 24

9:30-10:30pm

Started transferring journal to Word doc to upload to group folder

Week 9 (3/16-3/22)

Spring Break

Kept up with team chat and implantation progress but unexpectedly went out of town for family reasons and did not complete any major data analysis tasks

Week 8 (3/9-3/15)

Thursday, March 13

6-7:15

Class, capstone work time

Tuesday, March 11

4-4:40pm

Sponsor meeting with Dr. Siy

Received new spreadsheet with info for Fall 2025

4:40-5:20pm

Met with Trevor, Tyler, and Fara on Discord voice channel to discuss milestone 2, Angular progress, and backend progress

5:20-5:50

Brought my dev branch up to date

Added documentation to parsing file. Working on parsing specific data to separate files

Parsed department codes out but they printed for every semester, working on printing just unique department codes to one file next

Moved parsing file and .json file into Prototyping folder on dev-honora

6-7:15

Class, capstone work time

Week 7 (3/2-3/8)

Thursday, March 6

12:30-1:30pm

3-4:15pm

Finally able to access all levels of .json file. Have a script that pulls the semester code, course code, course section, and location for classes scheduled to Peter Kiewit Institute. Will need to script comparisons or compare the data manually to see which ones are always in the same classroom

4:30-5pm

Reviewed progress on the mid-semester project report and added some content to the 'similar applications' section

6-7:15pm

Class and project work time

Wednesday, March 5

2-4:30pm

5-8:30pm

Spent several hours intermittently working on parsing the nested .json file we received. I have not worked with this many levels of nesting or .json data much before so I am learning from scratch and probably doing it pretty exhaustively. The goal is to extract classes that are historically always assigned to the same room in PKI to hard code in our scheduler for future semesters.

Tuesday, March 4

4-4:30 pm

Met with sponsor and updated on progress, reviewed Trello tasks

4:30-5:15pm

Met with group in lieu of Sunday meeting. Caught Trevor up on what was missed while he was gone

Reviewed deliverables for this week

Discussed progress on individual tasks

6-7:15

Class and project work time

Week 6 (2/23-3/1)

Thursday, February 27

6-7:15

Class and project work time

Wednesday, February 26

8:15-9pm

Loaded fa21-fa24.json to local computer

Created a python script, started working on learning how to parse data from the .json file

Added items from yesterday's class to Trello board, added upcoming sponsor meetings

Tuesday, February 25

4-4:20pm

Met with team sponsor to update and check progress ahead of code milestone 1

6-7:15

Class and project work time

Sunday, February 23

11-11:15 am

Team meeting, touched base on milestone 1 and confirmed we were on pace to deliver this week

Week 5 (2/16-2/22)

Thursday, February 20

6-7:15pm

Class, project work time. Deliverables submitted for NFRs and Design

Tuesday, February 18

1:45-2:45pm

Reviewed GQM video

Went back and reviewed other lecture videos I haven't been watching

4-4:20pm

Group met with sponsor

Remembered to record, uploaded meeting recording to group folder

6-7:20pm

Class

Sunday, February 16

7-7:25pm

Group meeting; discussed upcoming milestones for the next two weeks

Looked over GQM group exercise for Tuesday

Week 4 (2/9-2/15)

Thursday, February 13

3-3:45pm

Added Tyler's suggestion to Wireframes presentation plan and created slides

Recorded, edited, and uploaded Wireframes presentation to group folder for review by group members

5:45-6pm

Reviewed goal modeling documentation/presentation

6-7:25pm

Class and breakout work

Used class time to recreate/update goal model, added updated goal model to requirements document and updated accompanying explanation

7:45-8:45pm

Updated and submitted Initial Requirements Outline

Updated Goal Model presentation plan, created slides for presentation

Recorded, edited, and submitted goal model presentation

Wednesday, February 12

8-9pm

Created and scripted Goal Model Presentation Plan and submitted to group for review and edits

Still plan to record/edit both presentations tomorrow afternoon before class meets

Tuesday, February 11

3:30-3:40pm

Uploaded Tyler's wireframe images to local computer

4-4:50pm (forgot to record meeting, set up a weekly reminder on my phone for future meetings)

Team meeting with sponsor

Dr. Siy would like authentication implemented

Add risks to project plan: data changing & not receiving new data
Work on timebox tasks in Trello board (break project up)

6-7:15

Class and team breakouts

9-10pm

Created and scripted Wireframes Presentation Plan and submitted it to group for review and edits
Plan to do the same tomorrow night for Goal Modeling Presentation, and record/edit both presentations on Thursday before class

Sunday, February 9

11am-12:15pm

Weekly team meeting; discussed wireframes and goal modeling deliverables
Started and worked on requirements document

Week 3 (2/2-2/8)

Thursday, February 6

5:40-5:55pm

Found a deliverable page in Canvas to submit Context presentation, submitted with link
Minor format changes to Context Document to prepare for submission

6-7:15pm

Class

Wednesday, February 5

1:30-4pm

Created skeleton slides from context document
Created script and plan for presentation
Recorded presentation
Downloaded/installed CapCut PC for editing. Edited out mistakes or errors, cut starting and stopping of recording
Reviewed and uploaded video presentation to group folder for group to review and publish. Final runtime 4:15

Tuesday, February 4

4-4:55

Met with sponsor

5-5:30

Reviewed lo-fi prototypes/wireframes and prepared for class
Converted and uploaded meeting recording to recordings file on group page
Drafted and sent meeting recap email

6-7:15

Class

Monday, February 3

10-10:20 pm

Research and review how to clone repository
Learned how to clone Tyler's dev branch into my own dev branch and push it back to our repository

10:20-10:40 pm

Downloaded and installed OBS Studio for recording and editing video presentations for this project
Fiddled with settings for about 10 mins

Sunday, February 2

11am-1:30pm

Group met at 11am
Worked on project proposal and plan, finalized and submitted both deliverables
Worked on context document for our project

To do:

Look into desktop version of Github, review pushing and pulling
Look forward to group activities and presentation requirements since I am both the client liaison and research analyst (possible script, requirements, etc)
Create skeleton slides for context presentation and record presentation, submit to group members for review by/before client meeting on Tuesday

Week 2 (1/26-2/1)

Thursday, January 30

6:00-7:15

Class

Wednesday, January 29

7:55-8:00

Spent a few minutes reviewing Context Documentation exercise from Tuesday's class and adding a few lines to our submission

8:00-8:05

Looked over goal modeling exercise for Thursday class
Reviewed some general project information

Tuesday, January 28

4-5:15 pm

Group meeting with sponsor (Dr. Siy) discussing project specifications and expectations

5:35-5:50

Recap discussion with group over Discord, drafted e-mail of recap and action items for sponsor, Worked on setting up recurring Zoom meeting for Sponsor meetings each week

6:00-7:15

Class

Monday, January 27

Spent about 10 mins collectively during the work day responding to communications with Dr. Siy and setting up meeting time that will work with group members

Sunday, January 26

Group met at 1pm, see meeting notes section for details

Spent 1 hour and 15 minutes meeting with group and working on project plan and project proposal outlines

9:15-9:25 pm

Spent 10 minutes drafting email to Dr. Siy inquiring if he is our project sponsor and if/when we can meet with him

9:25-9:30 pm

Spent about 5 minutes briefly looking up Angular, found a tutorial to follow for creating a basic Angular app

Looked over context group activity and context documentation for group activity due Tuesday

Week 1 (1/19-1/25)

Saturday, January 25

9-9:30 pm: spent approx. 30 mins going over project requirements and preparing for team meeting on Sunday, reviewing software engineering process

Notes:

-divvy up special roles as discussed in first lecture

Project Manager, Client Liaison, Requirements Analyst,

Architect/Technical Lead, Quality Assurance/Testing Lead, Lead Researcher, Scribe

-deliverables for Tuesday 1/28 are team roster and shared folder link

Tyler set up shared folder

Trevor set up Trello board

We just need to create a document with roster and link and submit

**Context documentation group activity

-deliverables for Thursday 1/30 are project proposal and project plan

Start creation of documents after Sunday's meeting

-reviewed software engineering basics, it has been a number of semesters since I have taken this course

8.4.5 Haresh Chandrasekar

May 6, 2025

- Attended Class Meeting
- Worked on Project Report
- Pushed Angular test into main branch, again, due to being lost in some merge conflicts.

April 27, 2025

- Meeting with group
- Finished issue with validations, but had issue with merging
  Successfully fixed merge conflicts despite new push from a different branch
- Started work on the Export Button

April 22, 2025

- Worked on validation in Angular
- Met with sponsor
- Meeting Notes: Discussed issue with calendar view, added new task to Trello
- Continued validation work

April 17, 2025

- Discussion about coding test in the backend and front-end
- General updates from teams

April 9, 2025

- Worked on the Login Page and Registration

April 1, 2025

- Found design for login template
  Angular Login Page Material Design Example

March 30, 2025

- Remodeled the login page to material design
- Still researching design
- Referencing sample example designs

March 27, 2025

- Registration under app

- Added an HTTP request on login
- Researching login template

March 15, 2025

- Added outline page and wrote a GET Test case for angular testing purposes.
- Merge to dev

March 4, 2025

- Merged pull request changes into dev or main

March 3, 2025

- Pushed changes to add home page and modify login page
- Some cards created using mat-cards; others hardcoded in CSS

March 1, 2025

- Still figuring out CSS patterns for certain text boxes
  Potentially using mat-cards as an alternative

- Worked on logout design picture
- Stil having issue aligning certain boxes around.

February 20, 2025

- Viciously research MDC related style for alignment.
- Researching materials labs potentially

February 18, 2025

- Team Meeting

February 16, 2025

- Extracted column names from patterns and pushed to GitHub

February 10, 2025

- Worked on understanding backend code and Python model logic
- Started work on a Trello task

February 9, 2025

- Team Meeting on Discord

January 28, 2025

- Started work on project proposal

January 27, 2025

- Group Meeting

January 23, 2025

- Team Meeting

8.5 Email Transcripts of Discussions with Clients

Subject: Capsoft Solutions Meetings

From: Harvey Siy Tues 5/6/25 8:48am

To: Honora Shanno

CC: _Tyler Porter, Trevor Ahlers, Haresh Chandrasekar, Fara Abdulloev

CapSoft team,

Thanks for letting me know.

I left some comments in the Mid-Semester Report submission that you can use when working on the Final Report. Let me know if you have questions.

Harvey

From: Honora Shanno Tues 5/6/25 7:30am

To: Harvey Siy

CC: Tyler Porter, Trevor Ahlers, Haresh Chandrasekar, Fara Abdulloev

Good morning Dr. Siy,

Group 3 has completed the Teaser Video and Demo Poster, and is working toward the completion of Code Milestone 5 and the Final Project Documentation. Since there are no major updates outside of that, I have cancelled our Zoom meeting for this afternoon. If any major questions arise we will touch base with you in class tonight or on Thursday. Please also reach out if there is anything else you need to ask us.

Thanks,

Honora

Subject: Capstone Videos Access

From: Fara Abdulloev Mon 5/5/25 4:17pm

To: Harvey Siy

Milestone 3: https://unomail-my.sharepoint.com/:v:/g/personal/tporter_unomaha_edu/EcY9Da8-kt1Fk6QSUalR1k8B_eh9wKBmiPHn_1lHmHo0wA?nav=eyJyZWZlcnJhbEluZm8iOnsicmVmZXJyYWxBcHAiOiJPbmVEcml2ZUZvckJ1c2luZXNzIiwicmVmZXJyYWxBcHBQbGF0Zm9ybSI6IldlYiIsInJlZmVycmFsTW9kZSI6InZpZXciLCJyZWZlcnJhbFZpZXciOiJNeUZpbGVzTGlua0NvcHkifX0&e=Xc2lke

Milestone 4: https://unomail-my.sharepoint.com/:v:/g/personal/tporter_unomaha_edu/EanL5rTN8BdEnR5bSlbc4ogBzLSChvTojTXvlktbiPYaTw?nav=eyJyZWZlcnJhbEluZm8iOnsicmVmZXJyYWxBcHAiOiJPbmVEcml2ZUZvckJ1c2luZXNzIiwicmVmZXJyYWxBcHBQbGF0Zm9ybSI6IldlYiIsInJlZmVycmFsTW9kZSI6InZpZXciLCJyZWZlcnJhbFZpZXciOiJNeUZpbGVzTGlua0NvcHkifX0&e=Gur0be

Teaser: https://unomail-my.sharepoint.com/:v:/g/personal/tporter_unomaha_edu/Ecysimc50UpAuEUQAcv2rxgB6ywVT8wNj27v2z8Gq-bGGg?nav=eyJyZWZlcnJhbEluZm8iOnsicmVmZXJyYWxBcHAiOiJPbmVEcml2ZUZvckJ1c2luZXNzIiwicmVmZXJyYWxBcHBQbGF0Zm9ybSI6IldlYiIsInJlZmVycmFsTW9kZSI6InZpZXciLCJyZWZlcnJhbFZpZXciOiJNeUZpbGVzTGlua0NvcHkifX0&e=4UUA4k

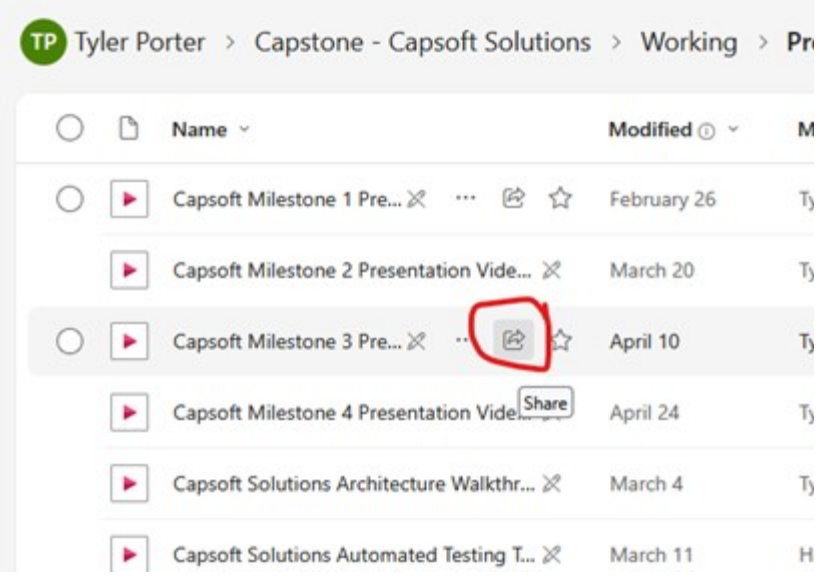
From: Harvey Siy Mon 5/5/25 3:04pm

To: Honora Shanno, Tyler Porter, Trevor Ahlers, Haresh Chandrasekar, Fara Abdulloev

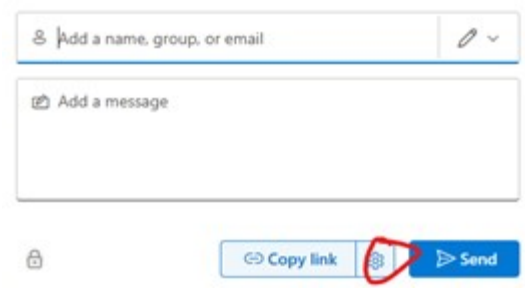
CapSoft team,

We just learned that the project videos your team submitted are not accessible to anyone aside from Zach and I.

To create a publicly shareable link on OneDrive, click on the share icon next to the video entry on OneDrive:



On the dialog box that pops up, click the gear next to "Copy Link":



Select "Anyone" and change access to "Can View".

Copy the resulting link and share with interested parties.

Could you resend the links, at least for Milestone 3, Milestone 4 and the Teaser? Just reply to this email with the links.

Thanks,

Harvey

Subject: Group 3 Availability for Classroom Coordinator Demo

From: Honora Shanno Tues 4/15/25 6:35pm

To: Harvey Siy

CC: Tyler Porter, Trevor Ahlers, Haresh Chandrasekar, Fara Abdulloev

Good evening Dr. Siy,

Following up on our meeting this afternoon, our group has the following times available on a weekly basis:

-Tuesdays at 4pm or anytime after (in lieu of sponsor meeting or in addition to, depending on the time)

-Tuesdays/Thursdays during class/breakout time.

If neither of these days will work with the classroom coordinator's schedule, it is possible she could meet with one group member at a different time. These are just the times that will work for everyone from our group to be there.

Thanks,

Honora

From: Harvey Siy Mon 4/21/25 3:49pm

To: Honora Shanno

CC: Tyler Porter, Trevor Ahlers, Haresh Chandrasekar, Fara Abdulloev

Honora and team,

Gloria is usually gone by 4pm. Her hours are 8am-4pm, Monday-Friday. What's the biggest subset of the team that can meet her within that timeframe?

Thanks,

Harvey

From: Honora Shanno Tues 4/22/25 3:57pm

To: Harvey Siy

CC: Tyler Porter, Trevor Ahlers, Haresh Chandrasekar, Fara Abdulloev

Dr. Siy,

Unfortunately all group members also work full time during those hours on weekdays. Perhaps Gloria would be available to attend demo day? We could also make a video addressing any questions she may have.

Thanks,

Honora

Subject: Milestone 2 Submission Extension

From: Honora Shanno Tues 3/13/25 5:29pm

To: Harvey Siy

CC: Tyler Porter, Trevor Ahlers, Haresh Chandrasekar, Fara Abdulloev

Good afternoon Dr. Siy,

I am emailing on behalf of group 3 - Capsoft Solutions - to let you know that we will be taking the extension on the Milestone 2 due date. We will be submitting our video walkthrough after we return from Spring Break.

Thanks,

Honora

Subject: Fall 2025 Schedule

From: Harvey Siy Tues 3/11/25 4:42pm

To: Honora Shanno, Tyler Porter, Trevor Ahlers, Haresh Chandrasekar, Fara Abdulloev

Here's the Fall2025 schedule I shared in the Zoom chat: Fall2025.csv

Subject: Historical classroom schedules data

From: Harvey Siy Tues 2/25/25 4:28pm

To: Honora Shanno, Tyler Porter, Trevor Ahlers, Haresh Chandrasekar, Fara Abdulloev

Team 3,

Here are UNO's class schedules from Fall 2021 to Fall 2024: classes-fa21-fa24.zip

It might be useful if you plan to rely on past classroom assignments to base your new assignments.

I'll let you know when I hear back from the Dean's office regarding our request for newer schedule spreadsheets.

Harvey

Subject: Capsoft Solutions Meeting Recap 2/4

From: Honora Shanno Tues 2/4/25 5:52pm

To: Harvey Siy

CC: Tyler Porter, Trevor Ahlers, Haresh Chandrasekar, Fara Abdulloev

Good evening,

Here is a brief recap of what we discussed today with action items for next week's meeting:

1. The biggest item for us currently is getting our hands on more spreadsheets and data from more recent semesters, particularly spreadsheets without the classroom column filled out (though we can clear that out of current spreadsheets if need be). Additionally, we are hopeful more recent spreadsheets would have less filtering and would show all courses to schedule (totally online, partially online, in-person).
2. List of specialized classrooms in PKI that certain classes must be scheduled in. Fara obtained some links that list some of the specialized rooms and we also have the link for IS&T Classrooms, Labs, and Conference Rooms, but we want to make sure we have all details possible on specialized rooms.

Thank you for your time this afternoon, and we look forward to seeing you next week!

Thanks,

Honora

Subject: Capsoft Solutions Recurring Meeting Invitation

From: Honora Shanno Tues 1/28/25 7:21pm

To: Harvey Siy

CC: Tyler Porter, Trevor Ahlers, Haresh Chandrasekar, Fara Abdulloev

Honora Shanno is inviting you to a scheduled Zoom meeting.

Topic: Capsoft Solutions

Time: Feb 4, 2025 04:00 PM Central Time (US and Canada)

Every week on Tue, until May 13, 2025, 15 occurrence(s)

Please download and import the following iCalendar (.ics) files to your calendar system.

Weekly:
https://unomaha.zoom.us/meeting/tJwrfuypqT0vEtSyqgIeqKGEqm_krzGFvYwl/ics?icsToken=DEnIkAM94EDTtetjwgAALAAAAHjuvdeM3TgVW3krluryWQT9AY3jYDQCKiBnt-xIYNEHxEg3eZS5tAfBQQE48pHgwpYSy8CUlGmoLjv7RjAwMDAwMQ&meetingMasterEventId=kagKj1fDQ_O8wGsKLNCmrg

Join Zoom Meeting

https://unomaha.zoom.us/j/98695176580

Meeting ID: 986 9517 6580

One tap mobile

+13462487799,,98695176580# US (Houston)

+16694449171,,98695176580# US

Dial by your location

• +1 346 248 7799 US (Houston)

• +1 669 444 9171 US

• +1 669 900 6833 US (San Jose)

- +1 719 359 4580 US
- +1 253 205 0468 US
- +1 253 215 8782 US (Tacoma)
- +1 301 715 8592 US (Washington DC)
- +1 305 224 1968 US
- +1 309 205 3325 US
- +1 312 626 6799 US (Chicago)
- +1 360 209 5623 US
- +1 386 347 5053 US
- +1 507 473 4847 US
- +1 564 217 2000 US
- +1 646 876 9923 US (New York)
- +1 646 931 3860 US
- +1 689 278 1000 US

Meeting ID: 986 9517 6580

Find your local number: https://unomaha.zoom.us/u/abqnzTkMch

Join by SIP

- 98695176580@zoomcrc.com

Join by H.323

- 144.195.19.161 (US West)
- 206.247.11.121 (US East)
- 221.122.88.195 (Mainland China)
- 115.114.131.7 (India Mumbai)
- 115.114.115.7 (India Hyderabad)
- 159.124.15.191 (Amsterdam Netherlands)
- 159.124.47.249 (Germany)
- 159.124.104.213 (Australia Sydney)
- 159.124.74.212 (Australia Melbourne)
- 170.114.134.121 (Hong Kong SAR)
- 170.114.180.219 (Singapore)
- 64.211.144.160 (Brazil)
- 159.124.168.213 (Canada Toronto)
- 159.124.196.25 (Canada Vancouver)

• 170.114.194.163 (Japan Tokyo)

• 147.124.100.25 (Japan Osaka)

Meeting ID: 986 9517 6580

Subject: Capsoft Solutions Meeting Recap 1/28

From: Honora Shanno Tues 1/28/25 5:43pm

To: Harvey Siy

CC: Tyler Porter, Trevor Ahlers, Haresh Chandrasekar, Fara Abdulloev

Good afternoon,

We wanted to provide a brief summary of our meeting notes with actions items for the upcoming week.

1. When you get a chance we would like any spreadsheet you currently have available, and the link to available classrooms you showed us in the meeting. If possible, we would also like to have access to more spreadsheets without information/fields filtered (include remote classes, Mammel Hall reassignments, TBA instructors that aren't picked yet, etc)
2. Google Cloud Access/Virtual Machine Access for hosting our project (we will still have a back-up option picked)
3. More information about specialized equipment for certain classes, which rooms have which specialized equipment not listed on Canvas (concrete lab for engineering, STEEL lab for cybersecurity)

Tuesdays at 4pm will work for all teammates so I will get my Zoom account linked with UNO and set up a recurring meeting for that time block.

Thanks,

Honora

From: Harvey Siy Tues 1/28/25 5:57pm

To: Honora Shanno

CC: Tyler Porter, Trevor Ahlers, Haresh Chandrasekar, Fara Abdulloev

CapSoft team,

Here are the 3 files I shared today. I'll send you more spreadsheets if I'm able to find the newer ones.

Also, the room information can be found here: https://www.unomaha.edu/college-of-information-science-and-technology/about/classrooms.php

Here's a list of specialized classrooms that Engineering uses that are not on the list: PKI 117, 127, 130, 248

Let me dig up the instructions for sharing access to Google Cloud Platform and get back to the class on that.

Harvey

Attachments: Fall 2022.csv, Spring2023.csv, Fall 2023 Scheduling Timeline.docx

Subject: Zoom Meeting Invitation – Capsoft Solutions Spec Meeting

From: Honora Shanno Mon 1/27/25 2:55pm

To: Harvey Siy

CC: Tyler Porter, Trevor Ahlers, Haresh Chandrasekar, Fara Abdulloev

Please see below for a Zoom link for tomorrow, Jan 28 at 4pm.

Click https://us05web.zoom.us/j/83701207399?pwd=Q8V5TPHzpDkY9apmz5nfzTDzL25k1F.1 to start or join a scheduled Zoom meeting.

Subject: Group 3 Meeting Times

From: Honora Shanno Sun 1/26/25 9:27pm

To: Harvey Siy

Good evening Dr. Siy,

I am emailing on behalf of Group 3, Capsoft Solutions. Are we correct in assuming that you are our liaison/sponsor for the PKI Scheduler Project? If so, we wanted to reach out and discuss some possible meeting times to go over project specifications and requirements as we draft our Project Proposal and Project Plan documents this week.

Our set group meeting time is every Sunday at 11am. We will also meet during class times as that becomes available, and immediately after class on Tuesdays/Thursdays if we need to wrap up any assignments or group activities due that day. Please let us know if either of those time blocks work well with your schedule, or if you have another suggested time we could meet with you. We are looking forward to working with you!

Sincerely,

Honora Shanno

From: Harvey Siy Mon 1/27/25 1:24pm

To: Honora Shanno

Honora,

Yes, I'll be the sponsor.

I'm available any time this afternoon between now and 5pm or tomorrow between 1:30-6pm. If those time don't work, I'll find some more options for Wednesday and Thursday.

Thanks,

Harvey

From: Honora Shanno Mon 1/27/25 2:54pm

To: Harvey Siy

Dr. Siy,

Tomorrow at 4pm will work for the majority of my teammates. I have created a Zoom meeting for that time and will send out the link shortly.

Thanks,

Honora

From: Honora Shanno Tue 1/28/25 4:37pm

To: Harvey Siy

The same link above should work to rejoin.

9. References

Amazon RDS and Aurora Documentation. (2023). Amazon.com. https://docs.aws.amazon.com/rds/

Amazon Web Services. (2019). *What Is AWS Elastic Beanstalk? - AWS Elastic Beanstalk*. Amazon.com. https://docs.aws.amazon.com/elasticbeanstalk/latest/dg/Welcome.html

Angular. (2025). *Angular Material*. Angular Material. https://material.angular.dev/

AWS. (2022). *Amazon Simple Storage Service Documentation*. Amazon.com. https://docs.aws.amazon.com/s3/

Cypress testing solutions | Cypress Documentation | Cypress Documentation. (2024, November 8). Cypress.io. https://docs.cypress.io/app/get-started/why-cypress

Durham School Scott Campus Facilities | Durham School of Architectural Engineering & Construction | College of Engineering | Nebraska. (2025). Unl.edu. https://engineering.unl.edu/durhamschool/durham-school-scott-campus-facilities/

Flask-CORS — Flask-Cors 3.0.7 documentation. (2013). Readthedocs.io. https://flask-cors.readthedocs.io/en/latest/

Flask-JWT-Extended's Documentation — flask-jwt-extended 4.4.4 documentation. (n.d.). Flask-Jwt-Extended.readthedocs.io. https://flask-jwt-extended.readthedocs.io/en/stable/

Full pytest documentation — pytest documentation. (n.d.). Docs.pytest.org. https://docs.pytest.org/en/stable/contents.html

Google Calendar. (n.d.). Accounts.google.com. https://www.google.com/calendar

IS&T Classroom, Labs, & Conference Rooms. (2023, March 17). University of Nebraska Omaha. https://www.unomaha.edu/college-of-information-science-and-technology/about/classrooms.php

Overview — Sphinx 4.0.0+ documentation. (n.d.). Www.sphinx-Doc.org. https://www.sphinx-doc.org/en/master/

Photopea. (n.d.). *Photopea | Online Photo Editor*. Photopea.com. https://www.photopea.com/

Python Software Foundation. (2019). *pickle — Python object serialization — Python 3.7.3 documentation*. Python.org. https://docs.python.org/3/library/pickle.html

Salesforce Einstein 1 Platform for Application Development. (n.d.). Salesforce. https://www.salesforce.com/products/platform/

Skedda. (n.d.). *Venue Booking System - Venue Software | Skedda*. Www.skedda.com. https://www.skedda.com/

(2023). Amazon.com. https://docs.aws.amazon.com/route53/