

# **Project Report**

## **Faculty Teaching Assignment (AggieAssign)**

Created as a part of **CSCE 606** coursework at Texas A&M University,  
College Station

by

**Abel Gizaw**

**Colby Endres**

**Navya Nandimandalam**

**Navya Unnikrishnan**

**Pavithra Gopalakrishnan**

**Wahib Kapdi**

**Yuqi Fan**

# Table of Contents

<b>Project Implementation Summary</b>	<b>2</b>
<b>User Story Descriptions</b>	<b>2</b>
Create a New Schedule	3
Schedules List View	5
Data View	6
Schedule Editing View	8
Schedule Solver Algorithm	9
<b>Team Roles</b>	<b>9</b>
<b>Sprint summary</b>	<b>10</b>
<b>Story Point Distribution</b>	<b>15</b>
<b>Customer meeting notes</b>	<b>18</b>
Customer Meeting #1	18
Customer Meeting #2	18
Customer Meeting #3	19
Customer Meeting #4	19
Customer Meeting #5	20
Customer Meeting #6	20
<b>BDD/TDD Process</b>	<b>20</b>
<b>Configuration Management</b>	<b>21</b>
<b>Production Release Notes</b>	<b>21</b>
<b>Tools/Gems</b>	<b>21</b>
<b>Project &amp; Repo Description</b>	<b>22</b>
Data	22
Application Code Base	22
Schedule Solver	23
Installation and Project Setup	24
<b>Relevant Links</b>	<b>24</b>

## Project Implementation Summary

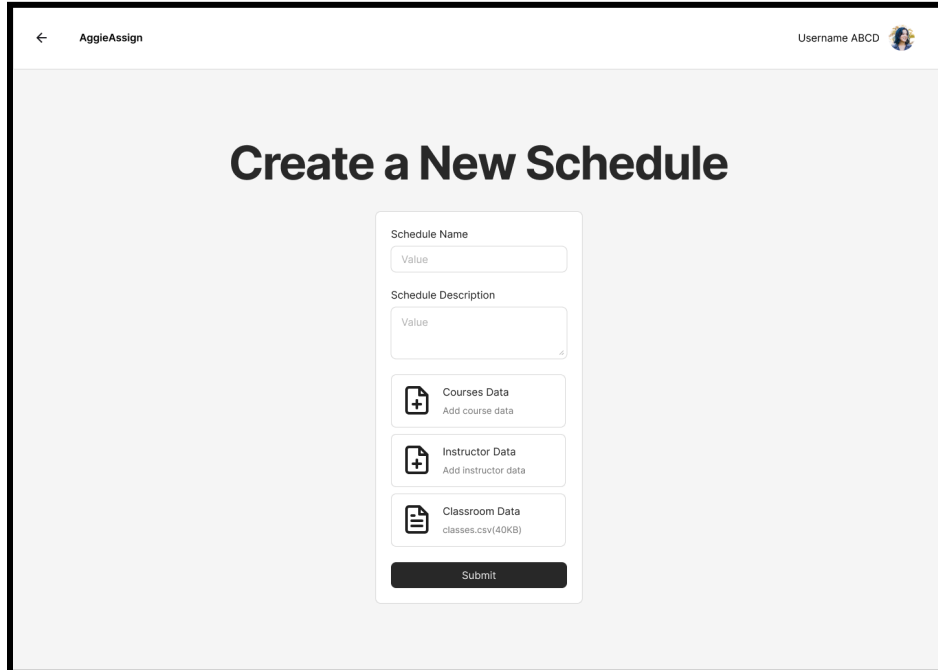
The Teaching Assignment Matching App was built on the obstacles faced by Professor Bettati and the Assistant Department Head with their manual system of matching faculty with assignments. The overarching customer need is to simplify an otherwise burdensome process of assigning faculty to courses in such a way that optimizes faculty satisfaction and satisfies Assistant Department Head needs, such as availing required courses to available professors. The hard constraints for time, rooms, and professors' availability, besides course preferences as soft constraints, are managed through an algorithm used in the app. The iterative nature of the application allows the Assistant Department Head to come up with a schedule, lock certain courses, and continue making changes until a result that best suits their needs is developed.

This would involve, in particular, direct participation by Professor Bettati in managing and iterating the schedule, and the Assistant Department Head in defining constraints and monitoring overall faculty assignments. The app allows these stakeholders to upload necessary data, select preferred course schedules, and exploit the algorithm to produce optimal faculty assignments that balance both faculty preferences and institutional needs. It has been a much more effective, flexible, and user-friendly alternative to the previously pursued manual scheduling process and thus drastically reduced the administrative burden while improving stakeholder satisfaction.

## User Story Descriptions

The overall sprint wise and developer wise story split is given below. This section just pertains to the UI Mockups and how we went about implementing them. The points not accounted for in this section are points related to code refactoring, bug fixing and documentation.

## Create a New Schedule

A mockup of a web page titled "Create a New Schedule". The page has a light gray background. At the top, there is a header bar with a back arrow, the text "AggieAssign", and a user profile icon with the text "Username ABCD". The main content area features the title "Create a New Schedule" in a large, bold, black font. Below the title is a form with the following elements: a "Schedule Name" label above a text input field with placeholder text "Value"; a "Schedule Description" label above a text input field with placeholder text "Value"; three buttons with plus icons and labels: "Courses Data" (Add course data), "Instructor Data" (Add Instructor data), and "Classroom Data" (classes.csv(40KB)); and a "Submit" button at the bottom.

**Mockup for Create Schedule's Page**

<b>Status</b>	Completed with some changes to the initial plan
<b>Points Allotted Initially</b>	2
<b>Points Taken So far</b>	7

The goal of the application requires us to create multiple schedules and also store the required history of the schedules. Hence a create schedules page.

The stories relating to this page were:

1. User sees a landing page with an overview of his generated schedules
2. Test CSV Upload
3. And the individual stories regarding data upload and parsing, only the upload part was handled in these stories:
  - a. User should be able to see the Instructors View
  - b. User should be able to see the Rooms View
  - c. User should be able to see the Instructors View
  - d. Instructor Preference - Parse data and populate table

In the end due to limitations with creation of a schedule and uploading csv files, it was decided to split this page into two pages. One for the creation of the schedule and one for data upload.

AggieAssign

Hi, Wahib! [Logout](#)

Create a New Schedule

Schedule name

Enter schedule name

Semester name

Enter semester name

Save Schedule

Back

## Create schedule page in the implementation

AggieAssign

Hi, Wahib! [Logout](#)

Schedule Details

Schedule Name: Test  
Semester: Test

BackView Data

Upload Room Data

197 rooms uploaded on November 23, 2024 at 03:39 PM

Select Room Data (CSV)

Choose FileNo file chosen

Upload Room Data

Upload Course Data

138 courses uploaded on November 23, 2024 at 03:39 PM

Select Course Data (CSV)

Choose FileNo file chosen

Upload Course Data

Upload Instructor Data

72 instructors uploaded on November 23, 2024 at 03:40 PM

Select Instructor Data (CSV)

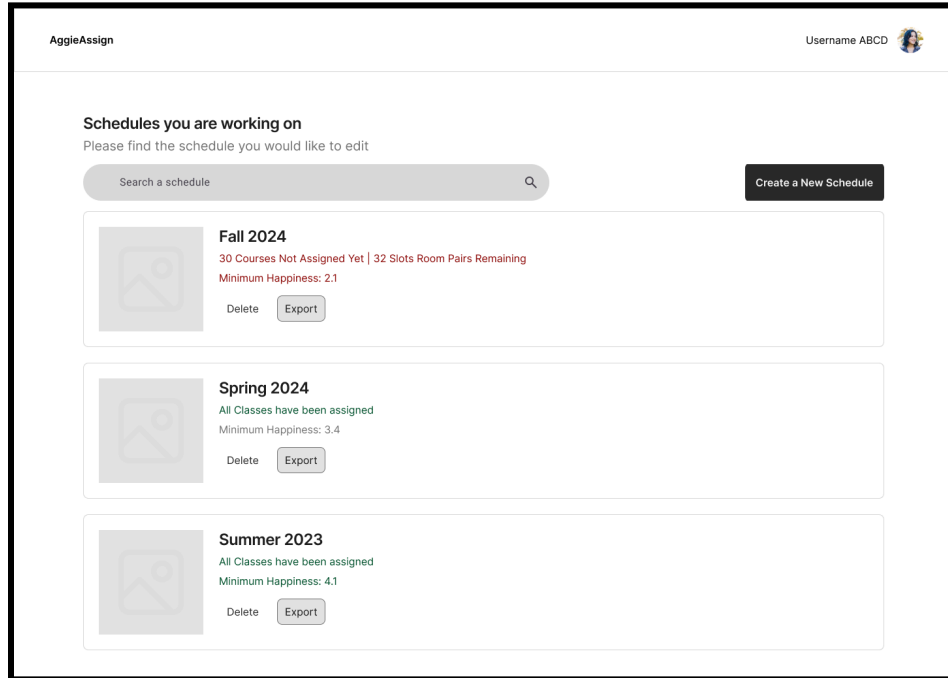
Choose FileNo file chosen

Upload Instructor Data

## Upload Data Page in the implementation

4

## Schedules List View



Mockup for the landing page

Status	Need to Have parts completed, Good to have parts remaining
Points Allotted Initially	3
Points Taken So far	7

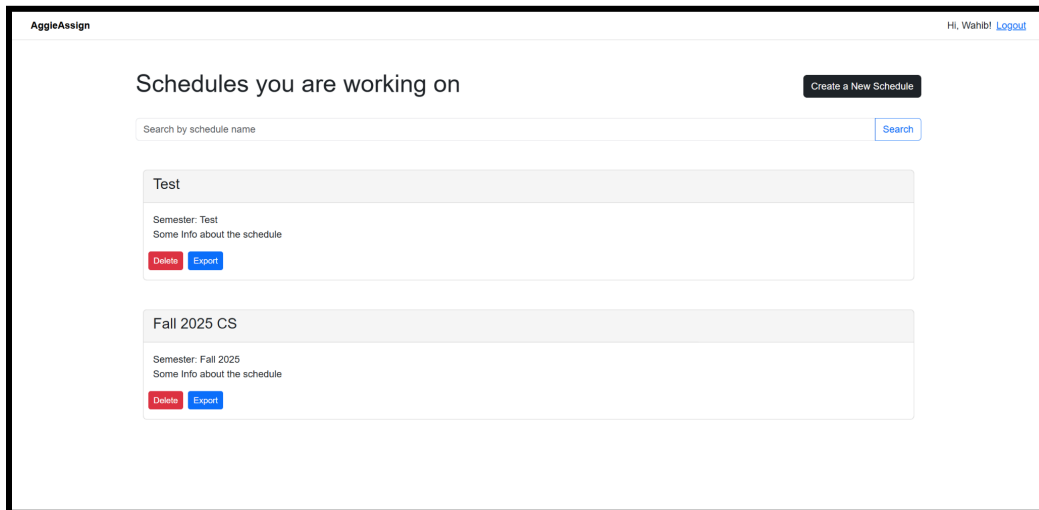
This page is the landing page of the entire application. The related stories to page are the:

1. User sees a landing page with an overview of his generated schedules.
2. Users should be able to download their generated schedules as csv.
3. User should be able to see only his generated schedules.

Not implemented features:

1. Showing the schedule metrics.

This page ended up looking quite similar to our mockup without the final statistics being shown.



The landing page in our implementation

## Data View

The screenshot shows the 'Fall 2024' data view in AggieAssign. The top bar includes a back arrow, 'AggieAssign', 'Username ABCD' with a profile icon, and buttons for 'Add Predefined Courses' and 'Export'. The main heading is 'Fall 2024'. Below this is a tabbed interface with 'Courses', 'Room', and 'Insturctors' (note the typo). The 'Room' tab is selected, showing a table with the following columns: Room, Capacity, Facilities, and Available for. The table lists 20 rows of room data.

Room	Capacity	Facilities	Available for
ZACH 108	105	AV LAB	MWF MW TTR
HRBB 112	105	LAB	MWF TTR
HRBB 110	105	AV	TTR
EABA 101	105	NONE	MWF MW TTR
EABB 101	105	AV LAB	MWF TTR
PETR 118	105	LAB	TTR
ZACH 108	105	AV	MWF MW TTR
HRBB 112	105	NONE	MWF TTR
HRBB 110	105	AV LAB	TTR
EABA 101	105	LAB	MWF MW TTR
PETR 118	105	AV	MWF TTR
ZACH 108	105	NONE	TTR
HRBB 112	105	AV LAB	MWF MW TTR
HRBB 110	105	LAB	MWF TTR
EABA 101	105	AV	TTR
PETR 118	105	NONE	MWF MW TTR
ZACH 108	105	AV LAB	MWF TTR
HRBB 112	105	LAB	TTR
HRBB 110	105	AV	MWF MW TTR

Mockup to View the uploaded Data

Status	Completed as planned with additional features
Points Allotted Initially	8
Points Taken So far	14

This view was important as a user may want to view the data they have uploaded and keep editing it. This was an important intermediary step before the user was finally able to edit their schedule.

The relevant stories include:

1. User should be able to see the Instructors View
2. User should be able to see the Rooms View
3. User should be able to see the Instructors View
4. User should be able to see Time Slot View
5. Instructor Preference - Parse data and populate table
6. User should be able to hide courses from the generator
7. Users should be able to download their generated schedules as csv.

The view is implemented completely.

Campus ▲	Building Code ▼	Room Number ▲	Capacity ▲	Lecture Hall	Learning Studio	Lab	Comments ▲
CS	EABA	118	75	Yes	Yes	Yes	
CS	EABA	121	100	Yes	Yes	Yes	
CS	EABB	106	118	Yes	No	No	
CS	HRBB	113	65	Yes	Yes	No	
CS	HRBB	124	135	Yes	No	No	
CS	HRBB	126	30	Yes	Yes	No	
CS	ONLINE	ONLINE	9999	No	No	No	
CS	ZACH	244	100	Yes	Yes	No	
CS	ZACH	310	100	Yes	Yes	No	
CS	ZACH	350	100	Yes	Yes	No	
CS	ZACH	590	34	No	No	Yes	
CS	ZACH	582	24	No	No	Yes	

**View Data page in the implementation**



## Schedule Editing View

The screenshot shows the 'Schedule Editing View' for 'Fall 2024'. The interface includes a header with 'AggieAssign' and 'Username ABCD'. Below the header, there are tabs for 'MWF', 'MW', and 'TTR'. The main area is a grid showing time slots (7:00 AM - 8:00 AM to 6:00 PM - 7:00 PM) and rooms (EABB118, PETR118, HRBB118, ZACH140, HRBB110, PETR121). Courses are assigned to specific time slots and rooms. For example, CSCE 606 is assigned to 3:00 PM - 4:00 PM in room 105, and CSCE 612 is assigned to 10:00 AM - 11:00 AM in room 45. The interface also includes a search bar for courses and a 'Generate Remaining' button.

**Schedule Editing View**

<b>Status</b>	Completed with Need to Have features, Incomplete Good to have features.
<b>Points Allotted Initially</b>	8
<b>Points Taken So far</b>	16

This view was of utmost importance. One of the heavier parts of the project. In this view the user is allowed to assign courses to the time slots, block time slots, and assign instructors or edit a generated schedule.

The relevant stories include:

1. User should be able to view the Schedule Room/ Time Slot Matrix
2. User should be able to Block out Time Slots
3. Adding a room booking changes the tab
4. User should be able to hide courses from the generator
5. User should be able to lock courses with time slots - After generating schedule
6. User should only be able to add allowed instructors to a room booking.

Features not implemented:

1. The side pane should be drawer to allow more real estate.
2. ONLINE course implementation.

### 3. Allowing drag and drop and searching among courses

Currently, this view is also completed but it can definitely be improved. To be more responsive to the user needs and create a better user experience.

**Schedule Editing View in the current implementation**

## Schedule Solver Algorithm

This does not have a UI Mockup related to it. The basic premise of this part of the project is to convert the problem of assigning courses to the correct time slots and rooms to a linear programming problem. Details about implementation and story points are mentioned in the upcoming sections.

The relevant stories include:

1. Research constraint-optimization algorithm design
2. Modify the algorithm to handle overlapping time slots
3. User should be able to generate schedule by uploading data
4. Remove kludge that forces profs and courses to be equal
5. Incorporate instructor preferences into algorithm

Current status of the implementation has the following known issues:

1. Unlocks locked courses.
2. Does not lock instructors.
3. Does block overlapping slots.
4. Fails at larger sizes.
5. Not storing the generated happiness metric.

## Team Roles

<b>Sprint 1</b>	<b>PO - Pavithra Gopalakrishnan</b>
-----------------	-------------------------------------

	<b>Scrum Master</b> - Navya Unnikrishnan
<b>Sprint 2</b>	<b>PO</b> - Abel Gizaw <b>Scrum Master</b> - Wahib Kapdi
<b>Spring 3</b>	<b>PO</b> - Colby Endres <b>Scrum Master</b> - Navya Priya Nandimandalam
<b>Sprint 4</b>	<b>PO</b> - Yuqi Fan <b>Scrum Master</b> - Pavithra Gopalakrishnan

## Sprint summary

### Sprint 1 Accomplishments

- **Data Collection and Analysis Started:**  
Began analyzing client data to identify key patterns and understand the requirements for the algorithm.
- **Pivotal Tracker Set Up:**  
Set up Pivotal Tracker for task management, enabling clear sprint planning and progress tracking.
- **CI/CD Pipeline Set Up:**  
Configured Continuous Integration/Continuous Deployment to automate the testing, building, and deployment process.
- **Code Climate Set Up:**  
Integrated Code Climate to monitor code quality and track potential issues early in the development cycle.
- **Project Hosted on Heroku:**  
Deployed the application on Heroku, providing a live environment for testing and feedback.
- **UI Mock-ups Created and Corroborated by the Client:**  
Designed and reviewed UI mock-ups with the client, incorporating their feedback to ensure the interface met expectations.
- **Database Design Created and Corroborated by the Client:**  
Created a database schema and validated it with the client to ensure it supports the project's data needs.
- **User OAuth Set Up (with .tamu.edu IDs Only):**  
Implemented OAuth authentication to restrict access to users with .tamu.edu email addresses, ensuring authorized access.
- **Uploading CSV Started:**  
Initiated functionality for uploading user CSV files to input data into the system.

- **Integration and Unit Test Suite Set Up:**

Established a test suite to ensure the system's components function correctly through integration and unit tests.

Sprint 1 Points Completed:

Story/Bug	Story Points
Figma UI Mockups	3
Create User stories and set up pivotal tracker	1
Sprint Documentation	1
Working Agreement Document	2
Code Climate, CI, Readme	2
Acceptance Tests	1
Set up recurring client meetings	1
On login, user should see a simple landing page with a welcome text	1
Deploy App to Heroku and setup test suite	1
User should be able to login using oAuth	2
Analyse Client Data	1
Tests for csv upload	1
Databse Design diagram	2

Sprint 2 Accomplishments:

- **The navigation header bar was developed:**

Created a user-friendly navigation header bar to provide easy access to different sections of the application.

- **Room view was implemented with filters and sorting capabilities, displaying the availability of rooms for teaching courses:**

Developed a room view that allows users to filter and sort available rooms based

on criteria like room type and capacity, helping users find suitable rooms for teaching.

- **Instructor View was implemented, displaying some of the teaching preferences:**  
Implemented an instructor view that showcases teaching preferences, such as preferred teaching times and courses, enabling better scheduling.
- **A time slot view was implemented, displaying the time slots available to teach each of the courses along with the days:**  
Created a time slot view showing available teaching times for each course, making it easier for instructors and administrators to plan classes.
- **The feature was designed to enable the creation and deletion of schedules. Schedules serve as the framework around which rooms, time slots, and instructors are assigned:**  
Developed a feature that allows the creation and deletion of schedules, organizing rooms, time slots, and instructors efficiently for each course.
- **PoCs on constraint optimization algorithms were conducted on test data. We finalized one of the algorithms that is optimal for our data:**  
Conducted proof-of-concept tests on various constraint optimization algorithms and selected the most effective one for scheduling based on test data.

Sprint 2 Points Completed:

Story/Bug	Story Points
User should be able to see the Header Bar and Logout from there	1
User should be able to see the Room View	2
User sees a landing page with an overview of his generated schedules	3
Link Github and Pivotal Tracker	1
User should be able to see Time Slot View	1
User should be able to see the Instructors View	2
Research constraint-optimization algorithm design	4
User should be able to upload data as csv files	1

Sprint 3 Accomplishments

- **Course View was created:**  
Developed a user interface to display course details in an organized and accessible format.
- **A schedule room/time slot matrix for visualizing the allotment was developed:**  
Implemented a matrix to help visualize the allocation of rooms and time slots for classes.
- **The ability to add predefined courses to ensure no other class can get scheduled at the same time was implemented:**  
Added functionality to define courses in advance and prevent scheduling conflicts by blocking overlapping time slots.
- **Instructor preferences data was parsed to try the algorithm on soft constraints:**  
Processed the instructor preference data and incorporated it into the algorithm to accommodate soft constraints in scheduling.
- **Filtering of time slots bug was fixed in the time slot view:**  
Fixed the issue with time slot filtering not functioning correctly in the time slot view.

Sprint 3 Points Completed:

Story/Bug	Story Points
User should be able to edit the added rooms in a schedule by uploading a new csv	1
Document zero code to a deployed application	1
User should be able to view the Schedule Room/ Time Slot Matrix	2
Create blocked time slots and predefined courses	2
Instructor Preference - Parse data and populate table	2
User should be able to navigate to the previous pages using Back Buttons	1
Fix Filtering on the Times Slot Page	2
User should be able to Block out Time Slots	1
User should be able to see the Courses Constraints View	4
User should directly land onto the schedules view	1

Modify the algorithm to handle overlapping time slots	1
Research constraint-optimization algorithm design	1
User should be able to generate schedule by uploading data	2

#### Sprint 4 Accomplishments:

- **Successfully resolved hosting issues related to the algorithm on Heroku:**  
Fixed hosting issues, ensuring the algorithm runs smoothly on the Heroku platform.
- **Enabled the ability to generate schedules by uploading data:**  
Added functionality to generate schedules by allowing users to upload relevant data.
- **Developed the locking course functionality:**  
Implemented a feature to lock certain courses, preventing them from being rescheduled.
- **Incorporated instructor preferences into the algorithm:**  
Integrated instructor preferences into the scheduling algorithm to better align with soft constraints.
- **Added functionality to hide courses from the generator:**  
Developed the ability to hide specific courses from the schedule generator, providing more flexibility.
- **Integrated various soft constraints into the system:**  
Incorporated multiple soft constraints, such as instructor preferences and time slot limitations, into the scheduling system.
- **Enabled a user view where a user sees only their schedules:**  
Created a personalized user view that displays only the schedules relevant to the logged-in user.
- **Added feature to export schedules:**  
Allows users to download schedules to their local system.

#### Sprint 4 Points Completed:

Story/Bug	Story Points
User should be able to lock courses with time slots - After generating schedule	2
User should be able to hide courses from the generator	2
Section ID and Number Fixes in JS	1

Handle preferences which have similar names between courses table and instructors table	1
User should be able to see only their generated schedules	2
User should be able to Block out Time Slots	2
Adding a room booking changes the tab	1
User should be able to download their generated schedules as csv	2
User should be able to generate schedule by uploading data	2
Remove kludge that forces profs and courses to be equal	1
Incorporate instructor preferences into algorithm	2
Hosting the algorithm on heroku	2
PoC - Soft constraints with the algorithm	1
User should only be able to add allowed instructors to a room booking	3

## Story Point Distribution

<i>Sprint</i>	<i>Story</i>	<i>Name</i>	<i>Points</i>	<i>Total Points</i>
Sprint 1	Figma UI Mockups (3)	Wahib Sabir Kapdi	3	15
Sprint 2	User should be able to see the Header Bar and Logout from there (1)		3	
	User should be able to see the Room View (2)			
Sprint 3	User should be able to edit the added rooms in a schedule by uploading a new csv (1)		5	
	Document zero code to a deployed application (1)			
	User should be able to view the Schedule Room/ Time Slot Matrix (2)			



	Create blocked time slots and predefined courses (1)			
Sprint 4	User should be able to lock courses with time slots - After generating schedule (1)		4	
	User should be able to hide courses from the generator (2)			
	Section ID and Number Fixes in JS (1)			
Sprint 1	Create User stories and set up pivotal tracker (1)	Navya Unnikrishnan	4	13
	Sprint Documentation (1)			
	Working Agreement Document(1)			
	Code Climate, CI, Readme (1)			
Sprint 2	User sees a landing page with an overview of his generated schedules (3)		3	
Sprint 3	Instructor Preference - Parse data and populate table (2)		3	
	User should be able to navigate to the previous pages using Back Buttons (1)			
Sprint 4	Handle preferences which have similar names between courses table and instructors table (1)		3	
	User should be able to see only his generated schedules. (2)			
Sprint 1	Working Agreement Document (1)	Pavithra Gopalakr...	3	11
	Acceptance Tests (1)			
	Set up recurring client meetings (1)			
Sprint 2	Link Github and Pivotal Tracker (1)		2	
	User should be able to see Time Slot View (1)			
Sprint 3	Fix Filtering on the Times Slot Page (2)		3	
	User should be able to Block out Time Slots (1) *			
Sprint 4	User should be able to Block out Time Slots (2)*		3	
	Adding a room booking changes the tab (1)			
Sprint 1	On login, user should see a simple landing page with a welcome text (1)	Navya Priya Nand...	3	11
	Code Climate and CI (1)			
	Deploy App to Heroku and setup test suite (1)			

Sprint 2	User should be able to see the Instructors View (2)		2	
Sprint 3	Create blocked time slots and predefined courses (1)		3	
	User should be able to see the Courses Constraints View (2)			
Sprint 4	User should be able to lock courses with time slots - After generating schedule (1)		3	
	User should be able to download their generated schedules as csv (2)			
Sprint 1	User should be able to login using oAuth (2)	Colby Endres	2	11
Sprint 2	Research constraint-optimization algorithm design (2)		2	
Sprint 3	User should directly land onto the schedules view (1)		3	
	Modify the algorithm to handle overlapping time slots (1)			
	Research constraint-optimization algorithm design (1)			
Sprint 4	User should be able to generate schedule by uploading data (1)		4	
	Remove kludge that forces profs and courses to be equal (1)			
	Incorporate instructor preferences into algorithm (2)			
Sprint 1	Analyse Client Data(1)	Yuqi Fan	2	9
	Tests for csv upload (1)			
Sprint 2	Research constraint-optimization algorithm design (1)		2	
	User should be able to upload data as csv files (1)			
Sprint 3	User should be able to generate schedule by uploading data (2) *		2	
Sprint 4	Hosting the algorithm on heroku (2)		3	
	User should be able to generate schedule by uploading data (1)			
Sprint 1	Databse Design diagram (2)	Abel Gizaw	2	9
Sprint 2	Research constraint-optimization algorithm design (1)		1	
Sprint 3	User should be able to see the Courses Constraints View (2)		2	
Sprint 4	PoC - Soft constraints with the algorithm (1)		4	

User should only be able to add allowed instructors to a room booking. (3)			
--	--	--	--

**Total = 79 points**

## Customer meeting notes

### Customer Meeting #1

1:45 PM 16th September 2024

Place: PETR 102B

#### **Discussing the DB and Application Design with the Client**

- Showed the client Figma designs
- Try to add course schedule-locking mechanisms
  - Generate somewhat different schedules for every play
  - Allow removal and addition of courses from the same place
  - Look into converting labs into sections
  - Try moving the scheduling page together with the viewing page.
- Showed the client our DB Design
  - Suggested we allow multiple schedules for the same semester
  - Try timeslot storage
  - Try generalizing relations further
- Reminded him to send the data for analysis and algorithm generation purposes.

### Customer Meeting #2

1:45 PM October 16, 2024

Place: PETR 102B

**Presented and showcased the constraint optimization algorithm using test data and clarified doubts related to the input files. Also captured additional requirements for the views.**

- Constraint Optimization Algorithm
  - Check if multiple schedules can be generated using the same data, out of which the client can pick any one.
- Additional Requirements
  - Rooms csv can be uploaded for every schedule
  - Instructor preferences and details will be available in the same csv

- Add Block Time Slots feature in the Predefined Courses Screen, ensuring no classes can be scheduled during this time.

### Customer Meeting #3

1:45 PM October 23, 2024

Place: PETR 102B

**Demonstrated Landing page, views - schedule, instructors, time slots, and CSV file upload. Discussed regarding the constraint optimization algorithm.**

- The client expressed interest in a feature to manually add/remove individual time slots from the view
- The client put forward a potential FERPA issue for publicly-hosted identifying data
- No changes are needed on our side, the professor will run the application
- locally
- We need to add build instructions
- Constraint Optimization Algorithm
  - Potential Issue where the room/class/time assignment prevents all professors from teaching was discussed.
  - We need to test this scenario with our algorithm
  - Introduce nondeterminism in the scheduling algorithm so that many schedules can be generated.
  - Unify ILP/bipartite matching steps into one cost function
  - Discussed deriving cost function that considers wasted rooms and professor happiness.

### Customer Meeting #4

1:45 PM October 30, 2024

Place: PETR 102B

**Presented and showcased the constraint optimization algorithm using test data as well as the current UI skeleton and updated features. In addition, we clarified doubts related to time schedules and additional requirements pertaining to exotic courses and predetermination of certain aspects of the schedule.**

## Customer Meeting #5

1:45 PM November 13, 2024

Place: PETR 102B

**Presented and showcased the current MVP using actual data provided by the client.**

- Allowed the client to mentally walk through his requirements alongside our implementation to clarify any confusion.
- Gathered small changes to implement in the coming sprint and clarified doubts pertaining to conflicting courses and instructors' CSV files.

## Customer Meeting #6

1:45 PM November 20, 2024

Place: PETR 102B

**Presented and showcased our final project and allowed the professor to see a start-to-end demo of the product. Answered all questions pertaining to actual usage and discussed areas for future improvement. Gathered general feedback on our team's work the entire semester.**

## BDD/TDD Process

For our behavior-driven design, we implemented cucumber scenarios that would test the behavior of our application, to ensure that its behavior would be in accordance with what Dr. Riccardo Bettati would like. Furthermore, our user stories were SMART and intuitive, and they contained both imperative and declarative scenarios. By using cucumber tests we were able to tell if the application was working properly without checking every single thing on the application by hand. Also, it ensured that during development we wouldn't forget any requirements.

For our Test-driven development, we used Rspec for unit testing to ensure that our individual components worked properly and intended so it wouldn't cause other errors in the application. Through TDD we were able to get a better understanding of our code and how it works even before any coding. Also, by testing edge cases we can know that those individual components work as intended no matter the situation.

## Configuration Management

We used GitHub as the repository for our configuration and implemented development branches that would eventually merge into the main after being reviewed by someone else to ensure that it would not negatively affect the main branch. In total 19 development branches were created, with a total of 70 pull requests. We did have a few spikes about the design and implementation of the algorithm. Furthermore, we have 4 git releases, 1 after every sprint, with tags named to reflect their associated sprint.

## Production Release Notes

During our process to production release, we encountered several issues that are not present to our local test environment. For our algorithm, a major part depends on the glpk library which solves linear problems but is not installed in the heroku. And for the heroku environment, the memory is more limited compared to our local environment, we did have to upgrade our environment to allow more memory to be allocated. We also tweaked our gem file to include a matrix which needs to be specifically listed for it to work on the heroku environment.

Our final release overcomes these challenges by offering a complete deployment compatible with the Heroku environment for stability in user service. Data upload, constraint handling, iterative rescheduling, and algorithmic processing are some of the features included in this release while maintaining user friendliness. Having addressed these technical challenges, the app is now ready to serve the needs of stakeholders efficiently and with a seamless user experience.

## Tools/Gems

The following tools and libraries were used for the project:

Name	Type	Purpose
GitHub	Website	Version control
Slack	App	Communication between developers
Pivotal Tracker	Website	Track story points for Agile development
CodeClimate	Website	Detect code smells and readability issues
Rulp	Gem	Integer Linear Program

		solver for schedule generation
CSV	Gem	Parse CSV data
algorithms	Gem	Priority Queue implementation for reducing instructor hours
faker	Gem	Generate fake user data for test suite
memory_profiler	Gem	Profile algorithm to reduce memory usage
SimpleCov	Gem	Calculate coverage for test suite

## Project & Repo Description

The repo can be found at <https://github.com/tamu-edu-students/Faculty-Teaching-Assignment/tree/main>. The project is mainly a Ruby on Rails project. The contents of the repo can be widely divided into 3 parts:

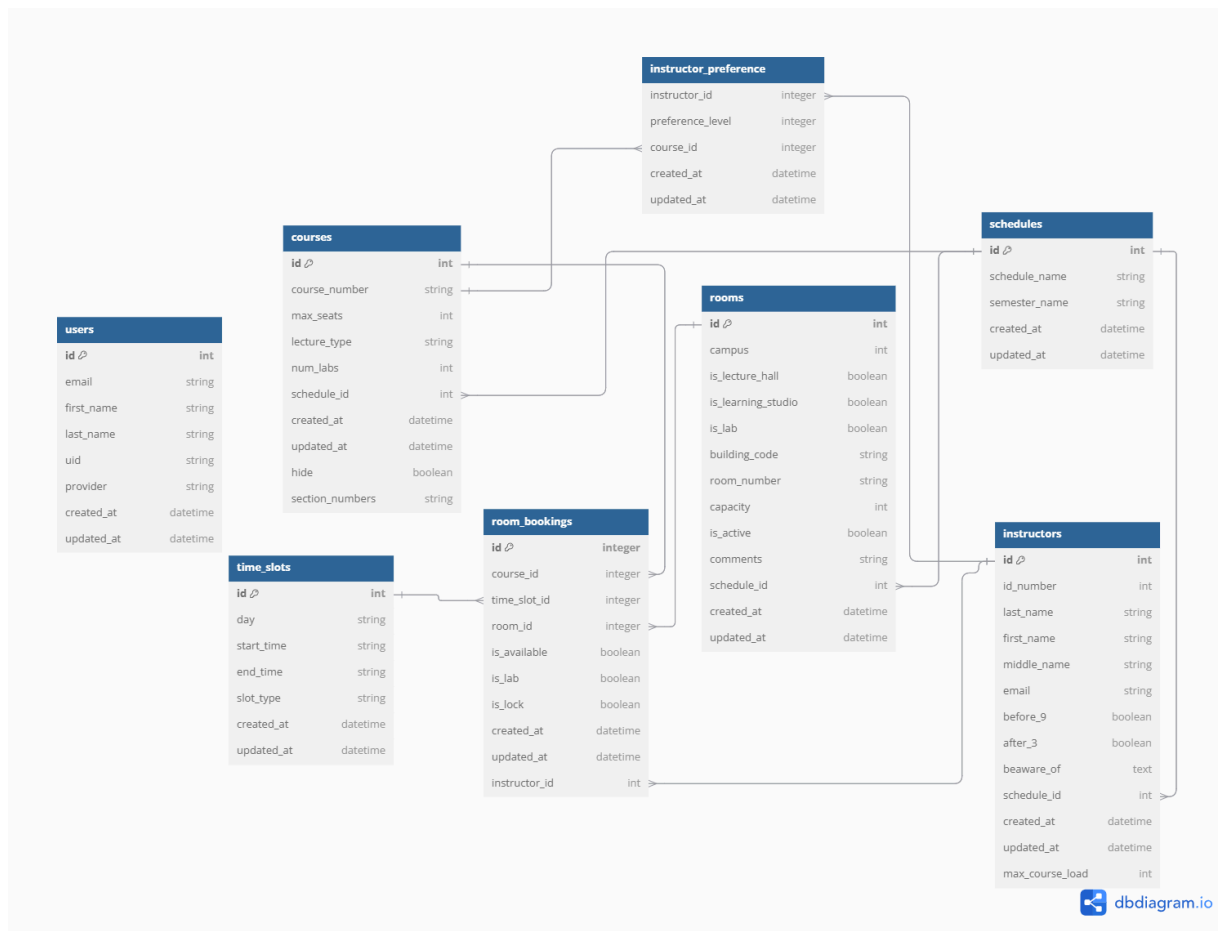
1. Data
2. Application Code Base
3. Schedule Solver

### Data

This is sample data. That can be found in the `/db/sample_data` folder. These files are used to upload the data for a given schedule. This includes a sample file .csv format for courses, instructors with their teaching preferences and rooms data.

### Application Code Base

The code base is pretty self explanatory for anyone who knows Rails. I'll just mention the purpose of specific scripts that are not a part of the DB Schema hence don't fall under the standard rails structure of Models, Controllers and Views in the table below.



Script	Purpose
app/controllers/sessions_controller.rb	To manage Login Session and Omni Auth
app/javascript/room_bookings.js	To handle the click event on table grid cells.
app/services/csv_handler.rb	To handle the upload for CSV files.
app/services/schedule_solver.rb	To solve the schedule using RULP.
app/views/shared/_courses_list.html.erb	Partial View to display left hand pane on the Predefined Courses View.
app/views/shared/_insched_nav.html.erb	Partial view for the header with the tab navigation in the data view.

### Schedule Solver

One extra command needs to be run for the solver to be functional. As noted in the README for the project.

```
rake glpk:install
```



This command downloads the necessary binaries for running glpk solver on the system. In case you already have glpsol setup on your system make sure that while running the project uses the one inside the repository.

Our current implementation of the Solver user Linear Programming. It tries to satisfy several constraints:

Constraint
Professors are not scheduled at the times they don't want
Each professor is scheduled for their contracted hours
A professor cannot be scheduled for two concurrent classes
For each pair of overlapping times, ensure that at most one is used
Courses that require special designations get rooms that meet them
Respect locked courses
Every class has exactly one assigned room and time
No two classes can share one room at the same time
Room capacity $\geq$ class enrollment

## Installation and Project Setup

For this just follow the steps given in the README for the project repository.

<https://github.com/tamu-edu-students/Faculty-Teaching-Assignment/tree/main?tab=readme-ov-file#build-instructions>

## Relevant Links

Deployed App: <https://faculty-teaching-assignment-31f5f9c405bc.herokuapp.com/>

GitHub Repository: <https://github.com/tamu-edu-students/Faculty-Teaching-Assignment>

Pivotal Tracker: <https://www.pivotaltracker.com/n/projects/2721604>

Team Working agreement:

<https://github.com/tamu-edu-students/Faculty-Teaching-Assignment/blob/main/documentation/Fall2024/Team%20Working%20Agreement.md>

Code Climate Report:

<https://codeclimate.com/github/tamu-edu-students/Faculty-Teaching-Assignment>

DB Schema: <https://dbdiagram.io/d/Teaching-Assignment-schema-66fca0d5fb079c7ebd06a77c>

User Documentation:

<https://github.com/tamu-edu-students/Faculty-Teaching-Assignment/blob/main/documentation/Fall2024/AggieAssign%20-%20User%20Documentation.pdf>