1. Project Description

In the Engineering Academy Block Scheduling project, we created an automated scheduling system for Engineering Academies (EA) statewide, beginning with Austin Engineering Academy. This system will streamline the scheduling of math, science, and engineering courses for both fall and spring semesters, ensuring that classes do not overlap while respecting prerequisite requirements. By replacing the current manual process – which involves spreadsheets, calendars, and Google Forms – the software has significantly improved efficiency. It has provided a smoother experience for both students and administrators, optimizing course planning and reducing administrative workload.

Over the course of four development sprints, our team has built a scheduling system that connects administrators and students. The backend handles critical tasks like bulk user imports via CSV files, managing the course catalog, verifying eligibility, and ensuring a smooth transition from block selection to final registration. On the frontend, we've designed interfaces, including admin dashboards for student uploads and course management, a student portal for login and course registration, and an interactive scheduling tool that detects conflicts in real time while confirming enrollments. This system benefits multiple stakeholders. Students can easily select their course blocks and register for courses automatically. Administrators can efficiently manage user accounts and course offerings without the hassle of manual processes. There are also less chances of error from the admin side and by utilizing this application, the admin can focus on other administrative work. By replacing manual spreadsheets and fragmented tools, this application streamlines scheduling and enhances efficiency for everyone involved.

2. User stories and their description:

1. UI Mockup for adding students as bulk insert (3 points):
   - As an admin of scheduling app
   - So that I can automate and speed up the process
   - I want to add multiple students to the database in a csv format.
   - Status: Completed
2. UI Mockup for students login and homepage (3 points):
   - As a student of the scheduling app
   - So that I can have an account & later register for courses
   - I want to login and view the homepage.
   - Status: Completed
3. UI Mockup for course registration (3 points):
   - As a student of the scheduling app
   - So that I can enroll in courses
   - I want to register for the courses that I select.
   - Status: Completed
4. UI mockup for block selection (3 points):
   - As a student of the scheduling app
   - So that I can enroll in the specific section for the courses I selected
   - I want to select a block to finalize my schedule.
   - Status: Completed
5. Analyzing Existing Code (3 points):
   - The team needs to review the code and try to run it on their devices.
   - Analyze the test cases and check what needs to be improved.
   - Discuss with Professor Shana what changes are needed.
   - Status: Completed
6. Implement controller & view for bulk insertion of users (5 points):
   - As an admin of the scheduling app
   - So that I can add many users at once to the database
   - I want to upload a spreadsheet containing new users in a csv format.
   - Status: Completed
7. Implement controller & view for course registration (3 points):
   - As a student of the scheduling app
   - So that I can enroll in courses

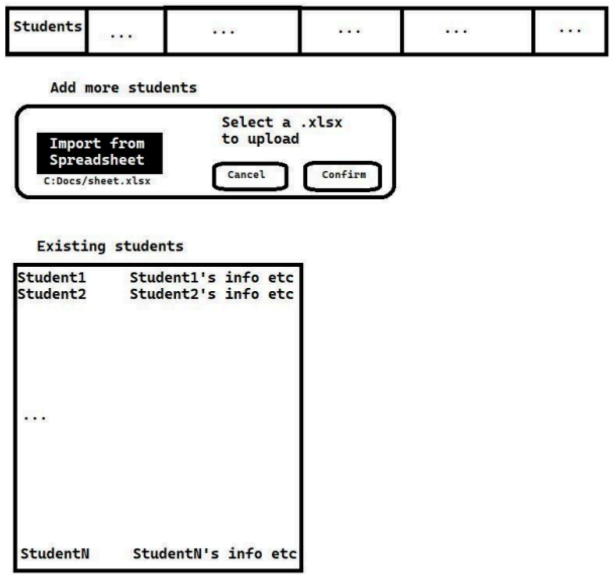
    - I want to select courses that I am eligible for.
    - Status: Completed
8. UI Implementation for Home Page for students (3 points):
    - As a student of the scheduling app
    - So that I can have an account & later register for courses
    - I want to login and view the homepage.
    - Status: Completed
9. UI Implementation for Block Selection (3 points):
    - As a student of the scheduling app
    - So that I can enroll in the specific section for the courses I selected
    - I want to select a block to finalize my schedule.
    - Status: Completed
10. Fetch blocks from the upstream (5 points)
    - As a student of the scheduling app
    - So that I can register for the courses
    - I want to see the courses and the block I selected.
    - Status: Completed
11. Enable only users in DB to be able to login into App (5 points)
    - As an admin of the scheduling app
    - So that I can create schedules of the students
    - I want to allow only the students and admins who are part of the app.
    - Status: Completed
12. Enable non-tamu users to be admins (5 points)
    - As an admin of the scheduling app
    - So that I can have an account & carry out administrative tasks
    - I want to login without a non-tamu email.
    - Status: Completed
13. Add courses to Database (5 points)
    - As an admin of the scheduling app
    - So that I can register students to courses
    - I want to add courses to the app so that students can select the courses.
    - Status: Completed
14. Integrate UI Components for Course Registration (5 points)
    - As a student of the scheduling app
    - So that I can enroll in courses directly through the interface

- I want the registration workflow hooked up to the UI—displaying available courses, allowing block-wise selection
- Status: Completed

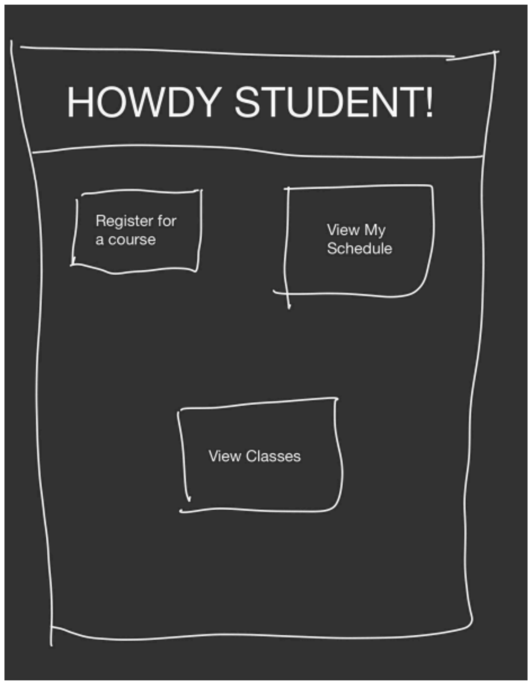
15. Connect Block Selection to Registration Workflow (5 points)
    - As a student of the scheduling app
    - So that my chosen blocks actually translate into enrolled courses
    - I want the block-selection service to hand off my selections directly to the registration logic
    - Status: Completed

Mockups and Screenshots

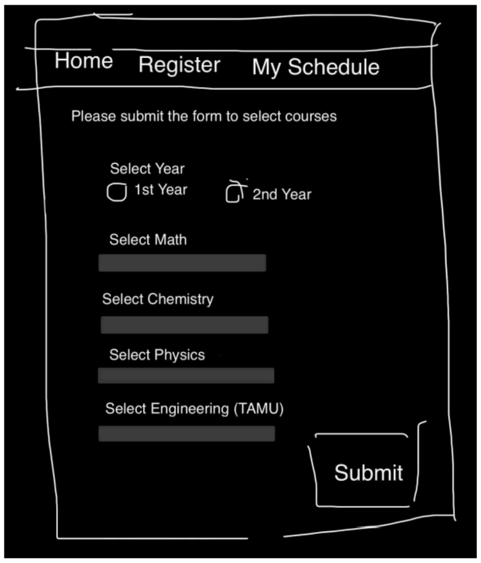
1. UI Mockup for adding students as bulk insert



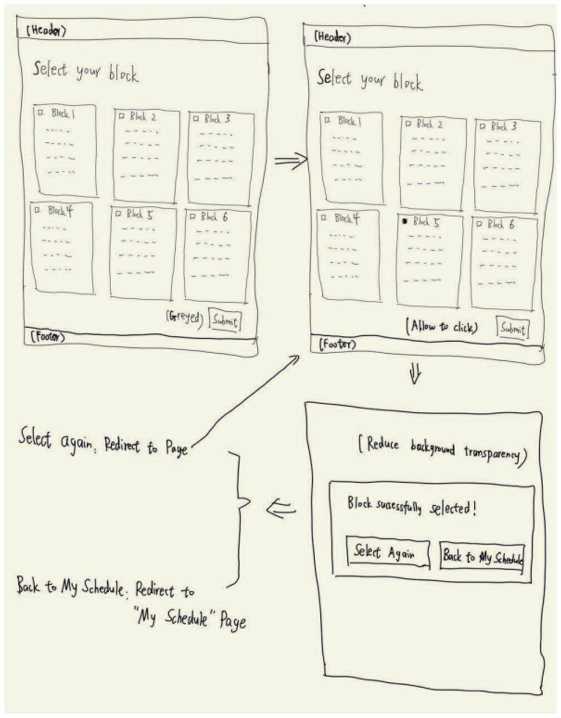
2. UI Mockup for students login and homepage



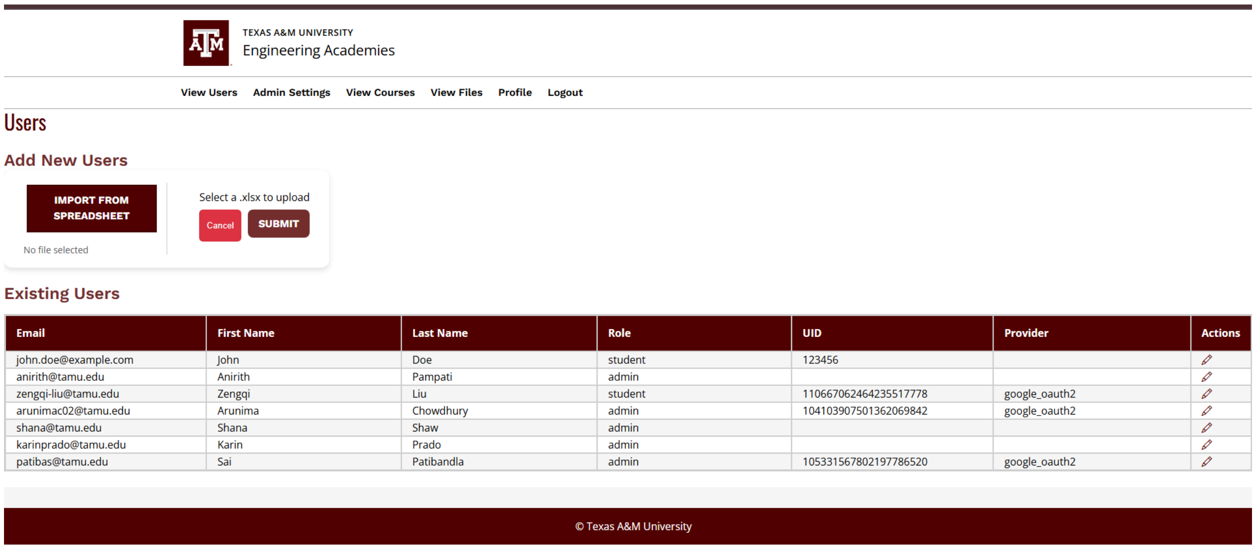
3. UI Mockup for course registration



4. UI mockup for block selection


(Header)
Select your block
Block 1
Block 2
Block 3
Block 4
Block 5
Block 6
(Greyed)
Submit
(Footer)
(Allow to click)
(Reduce background transparency)
Block successfully selected!
Select Again
Back to My Schedule
Select again: Redirect to Page
Back to My Schedule: Redirect to "My Schedule" Page

5. Bulk insertion of users using spreadsheet. Only users in the DB can login. This also allows non-tamu users to login as well if they are in the DB.



Users

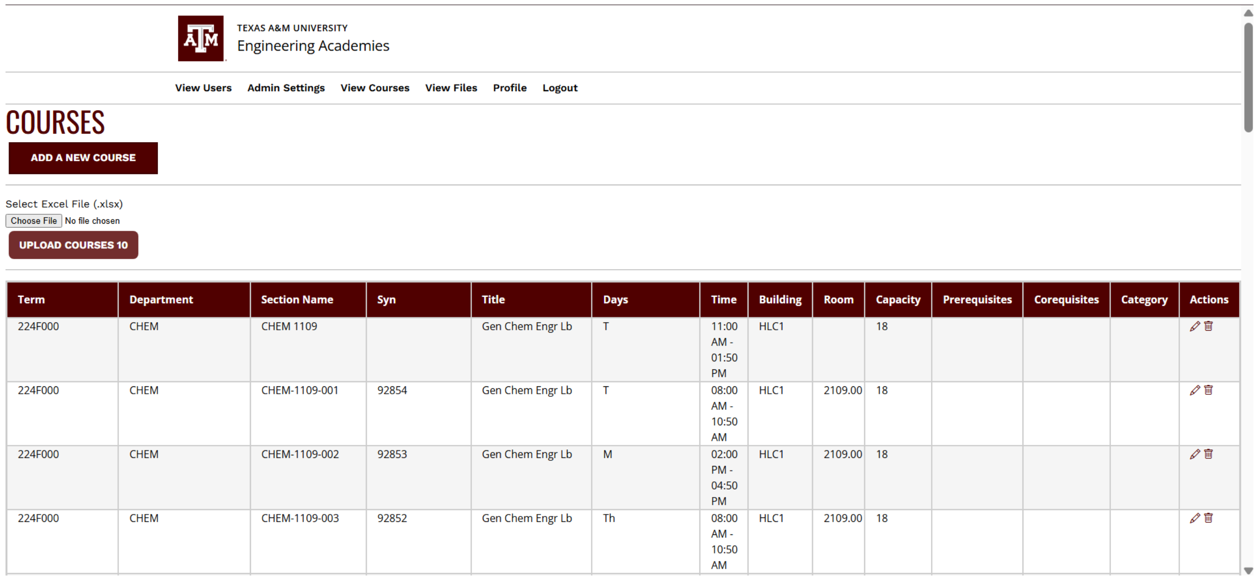
Add New Users

Existing Users

Email	First Name	Last Name	Role	UID	Provider	Actions
john.doe@example.com	John	Doe	student	123456		✎
anirith@tamu.edu	Anirith	Pampati	admin			✎
zengqi-liu@tamu.edu	Zengqi	Liu	student	110667062464235517778	google_oauth2	✎
arunimac02@tamu.edu	Arunima	Chowdhury	admin	104103907501362069842	google_oauth2	✎
shana@tamu.edu	Shana	Shaw	admin			✎
karinprado@tamu.edu	Karin	Prado	admin			✎
patibas@tamu.edu	Sai	Patibandla	admin	105331567802197786520	google_oauth2	✎

© Texas A&M University

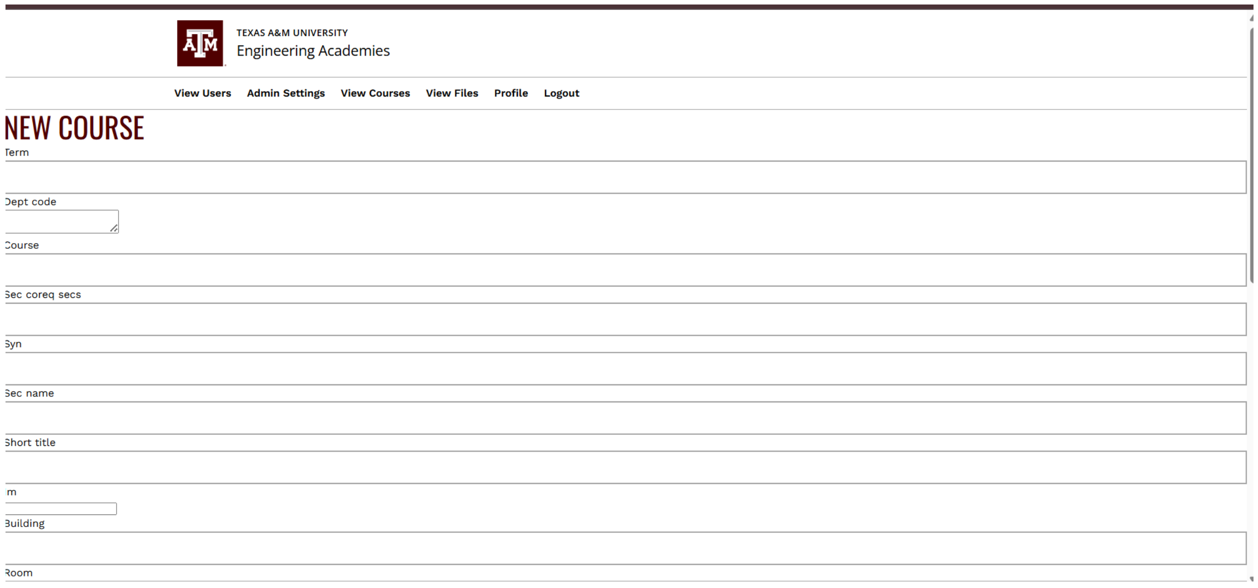
6. Add courses to the database from the admin side. The admin can upload a spreadsheet of courses or add a course one by one.



COURSES

Term	Department	Section Name	Syn	Title	Days	Time	Building	Room	Capacity	Prerequisites	Corequisites	Category	Actions
224F000	CHEM	CHEM 1109		Gen Chem Engr Lb	T	11:00 AM - 01:50 PM	HLC1		18				✎ 🗑
224F000	CHEM	CHEM-1109-001	92854	Gen Chem Engr Lb	T	08:00 AM - 10:50 AM	HLC1	2109.00	18				✎ 🗑
224F000	CHEM	CHEM-1109-002	92853	Gen Chem Engr Lb	M	02:00 PM - 04:50 PM	HLC1	2109.00	18				✎ 🗑
224F000	CHEM	CHEM-1109-003	92852	Gen Chem Engr Lb	Th	08:00 AM - 10:50 AM	HLC1	2109.00	18				✎ 🗑

7. Clicking "Add a new Course", the admin can add one course at a time



NEW COURSE

Term

Dept code

Course

Sec coreq secs

Syn

Sec name

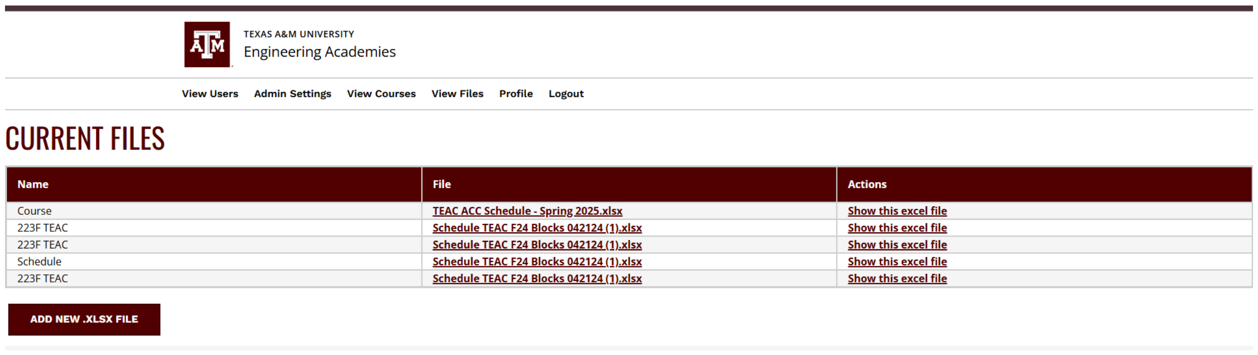
Short title

m

Building

Room

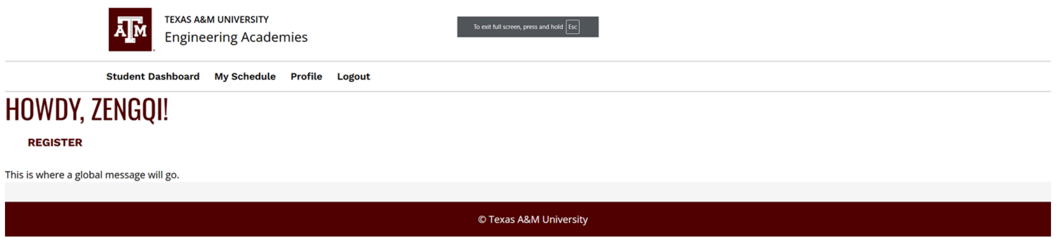
8. List of files added



CURRENT FILES

Name	File	Actions
Course	TEAC ACC Schedule - Spring 2025.xlsx	Show this excel file
223F TEAC	Schedule TEAC F24 Blocks 042124 (1).xlsx	Show this excel file
223F TEAC	Schedule TEAC F24 Blocks 042124 (1).xlsx	Show this excel file
Schedule	Schedule TEAC F24 Blocks 042124 (1).xlsx	Show this excel file
223F TEAC	Schedule TEAC F24 Blocks 042124 (1).xlsx	Show this excel file

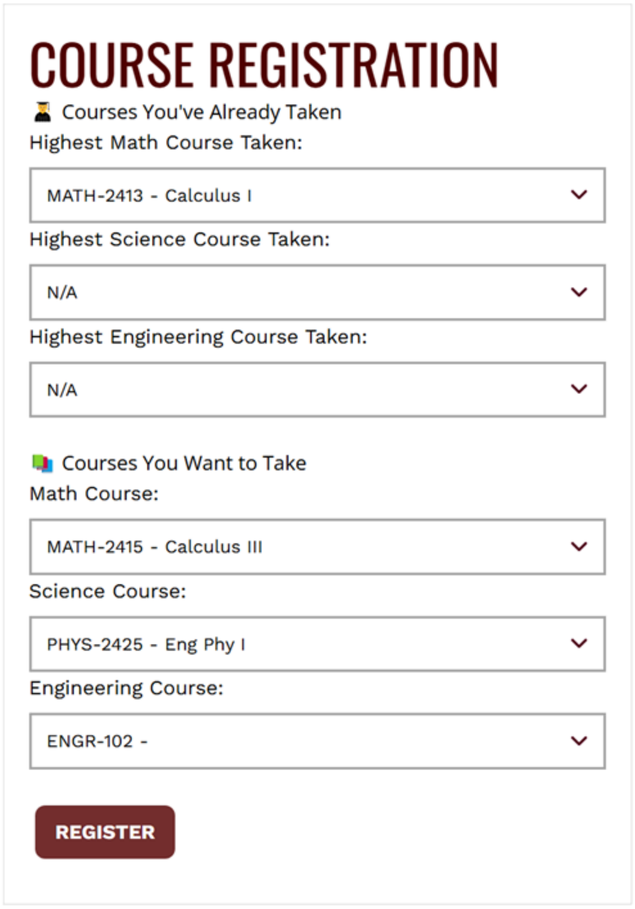
9. Student login page



HOWDY, ZENGQI!

This is where a global message will go.

10. Student course registration form



COURSE REGISTRATION

Courses You've Already Taken

Highest Math Course Taken: MATH-2413 - Calculus I

Highest Science Course Taken: N/A

Highest Engineering Course Taken: N/A

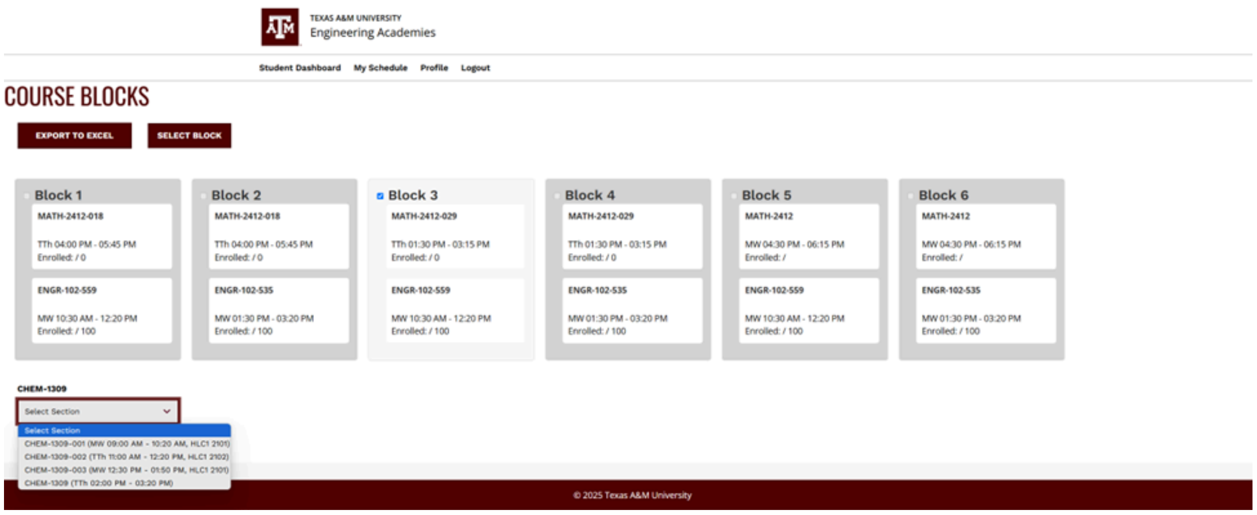
Courses You Want to Take

Math Course: MATH-2415 - Calculus III

Science Course: PHYS-2425 - Eng Phy I

Engineering Course: ENGR-102 -

11. Course block selection on student side



COURSE BLOCKS

Block 1	Block 2	✓ Block 3	Block 4	Block 5	Block 6
MATH-2412-018 TTh 04:00 PM - 05:45 PM Enrolled: / 0	MATH-2412-018 TTh 04:00 PM - 05:45 PM Enrolled: / 0	MATH-2412-029 TTh 01:30 PM - 03:15 PM Enrolled: / 0	MATH-2412-029 TTh 01:30 PM - 03:15 PM Enrolled: / 0	MATH-2412 MW 04:30 PM - 06:15 PM Enrolled: /	MATH-2412 MW 04:30 PM - 06:15 PM Enrolled: /
ENGR-102-559 MW 10:30 AM - 12:20 PM Enrolled: / 100	ENGR-102-535 MW 01:30 PM - 03:20 PM Enrolled: / 100	ENGR-102-559 MW 10:30 AM - 12:20 PM Enrolled: / 100	ENGR-102-535 MW 01:30 PM - 03:20 PM Enrolled: / 100	ENGR-102-559 MW 10:30 AM - 12:20 PM Enrolled: / 100	ENGR-102-535 MW 01:30 PM - 03:20 PM Enrolled: / 100

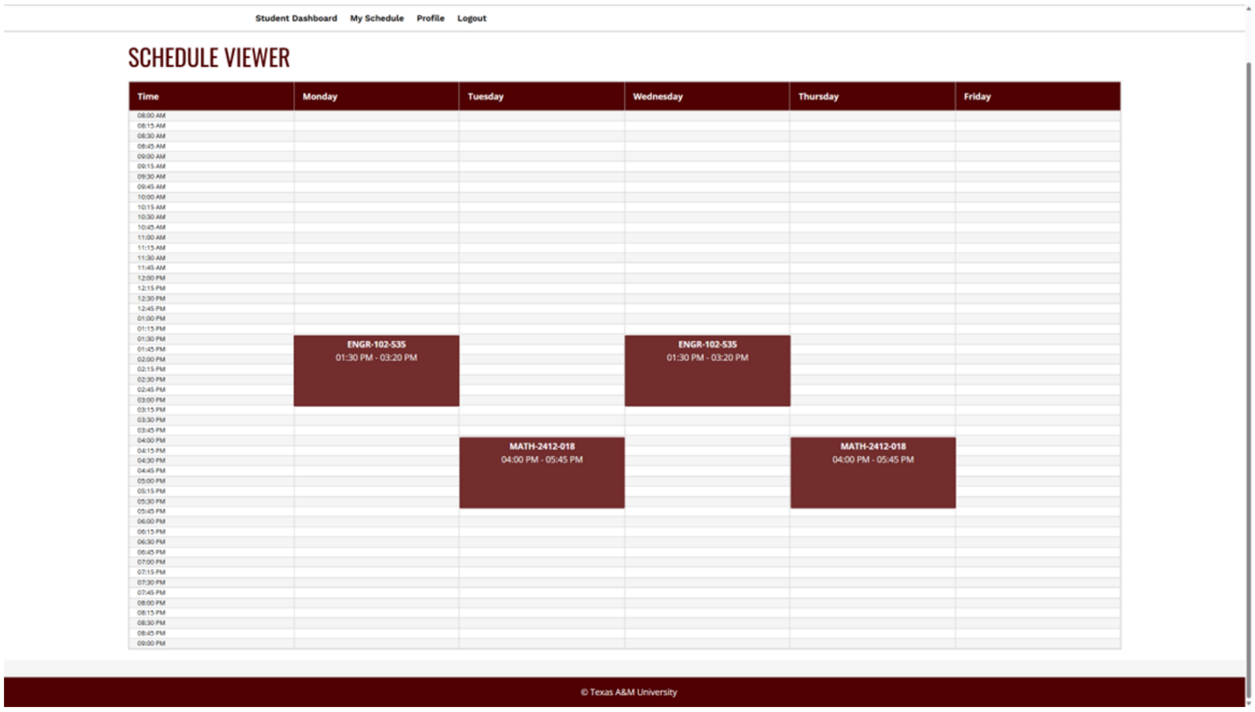
CHEM-1309

Select Section

- Select Section
- CHEM-1309-001 (MW 09:00 AM - 10:20 AM, HLC1 2101)
- CHEM-1309-002 (TTh 11:00 AM - 12:20 PM, HLC1 2102)
- CHEM-1309-003 (MW 12:30 PM - 01:50 PM, HLC1 2101)
- CHEM-1309 (TTh 02:00 PM - 03:20 PM)

© 2025 Texas A&M University

12. Student schedule after selecting a block



Student Dashboard My Schedule Profile Logout

SCHEDULE VIEWER

Time	Monday	Tuesday	Wednesday	Thursday	Friday
08:00 AM					
08:15 AM					
08:30 AM					
08:45 AM					
09:00 AM					
09:15 AM					
09:30 AM					
09:45 AM					
10:00 AM					
10:15 AM					
10:30 AM					
10:45 AM					
11:00 AM					
11:15 AM					
11:30 AM					
11:45 AM					
12:00 PM					
12:15 PM					
12:30 PM					
12:45 PM					
01:00 PM					
01:15 PM					
01:30 PM	ENGR-102-535 01:30 PM - 03:20 PM		ENGR-102-535 01:30 PM - 03:20 PM		
01:45 PM					
02:00 PM					
02:15 PM					
02:30 PM					
02:45 PM					
03:00 PM					
03:15 PM					
03:30 PM					
03:45 PM					
04:00 PM		MATH-2412-018 04:00 PM - 05:45 PM		MATH-2412-018 04:00 PM - 05:45 PM	
04:15 PM					
04:30 PM					
04:45 PM					
05:00 PM					
05:15 PM					
05:30 PM					
05:45 PM					
06:00 PM					
06:15 PM					
06:30 PM					
06:45 PM					
07:00 PM					
07:15 PM					
07:30 PM					
07:45 PM					
08:00 PM					
08:15 PM					
08:30 PM					
08:45 PM					
09:00 PM					

© Texas A&M University

3. Legacy Project

Since it was a legacy project, the first thing the team did in the first iteration is to understand how the existing application works. The team thoroughly understood and analyzed the existing legacy code. In the first iteration, the team set up a development environment, checked out any scratch branches, and ran the application in a production-like setting. Additionally, the team inspected the database schema to understand how the scheduling logic is implemented. We pinpointed areas where changes are needed to assess the code base's structure and how it's maintained.

We discussed weekly with Professor Shana Shaw about what her expectations were from the current team. She noted that while the admin interface includes some basic functionalities, there was significant room for improvement. The team mostly focused on implementing the student side of the application which was not implemented at all. We created a login page for students, allowing them to register for courses, select their course preferences, and implemented logic for block generation based on prerequisites, section availability, and the student's academic year. The major tasks we carried out were integrating the back end logic for the login and authentication page for the students, student registration page and block generation page and then also integrating the UI for these tasks.

Most of the refactoring the team did was on the admin side, since the Professor mentioned that the admin side needed some improvements. Previously, only the tamu email users were able to access the application. However, our professor wanted to add non tamu email users to the admin side as well. So we implemented the logic for adding non tamu email users as well. We also added a feature of adding students and admins in bulk via CSV imports and only the authorized users are allowed to login to the application. Other than that, we made some changes in the course upload feature so that the admin can upload courses in bulk, or add a single course and are allowed to modify or delete them.

4. Team Roles

Sprint 1:
Project owner - Sai Patibandla
Scrum master - Arunima Chowdhury
Developers - Joseph Cisneros, Zengqi Liu

Sprint 2:
Project owner - Sai Patibandla
Scrum master - Joseph Cisneros
Developers - Arunima Chowdhury , Zengqi Liu

Sprint 3:
Project owner - Sai Patibandla
Scrum master - Arunima Chowdhury
Developers - Joseph Cisneros, Zengqi Liu

Sprint 4:
Project owner - Zengqi Liu
Scrum master - Sai Patibandla
Developers - Joseph Cisneros, Arunima Chowdhury

5. Summary of Each Sprint

Sprint 1:

- Team members analyzed the existing codebase, ran the project locally. There were some issues with the login which were fixed in subsequent sprints.
- User stories created after discussing with the customer. UI mockup created for each of the user stories. Mockups were discussed with the customer and got approved by the customer.
- App deployed to Heroku

Points Completed: 15

Sprint 2:

- The UI portion completed for every major feature.
- A few new stories were created after discussing with the customer.
- The team completed most of the controller logic for each major feature.
- App deployed to Heroku with new changes

Points Completed: 14

Sprint 3:

- Completed integration of block selection.
- Login for non tamu users to be admins was work in progress
- Only authenticated users can now login
- Integration of adding courses to database was work in progress

Points Completed: 10

Sprint 4:

- Registration of block selection is completed.
- Login for non tamu users to be admins is completed
- Integration of adding courses to database completed
- Selection of courses form database completed

Points Completed: 20

6. User story points table

Name	**Stories Completed**	**Points Completed**
Sai Patibandla	5	19
Joseph Cisneros	4	16
Zengqi Liu	5	19
Arunima Chowdhury	4	14

7. Customer Meetings

Throughout the course of the project, we met with our customer multiple times to present our progress and gather feedback. Below is a list of customer meetings, their dates, and a summary of discussions and demonstrations during each.

Date	Description of Meeting
02/06/2025	Presented initial UI mockups (Student Login, Course Registration, Block Selection). Customer approved most designs and suggested removing the "View Courses" button from the homepage. Discussed initial project goals and challenges with manual scheduling.
02/12/2025	Demoed the initial Heroku deployment of the app.- Discussed the structure of student data spreadsheets needed for the system. Talked about adjusting block generation logic to avoid too many redundant blocks. Planned feature development priorities for Sprint 2.
02/26/2025	Demoed UI progress: bulk user upload, course registration. Discussed a new requirement: allowing non-TAMU email users to log in as admins. Reviewed next steps including database integration and authenticated login.
03/05/2025	Updates included block selection feature completion and login authentication addition.

03/26/2025	Updates included completed block registration, admin login for non-TAMU users, and course addition progress.
04/24/2025	Demoed integration of course registration with block selection. Received feedback for courses that can be selected as standalone and implemented the Logic for that.

8. Explain BDD/TDD process.

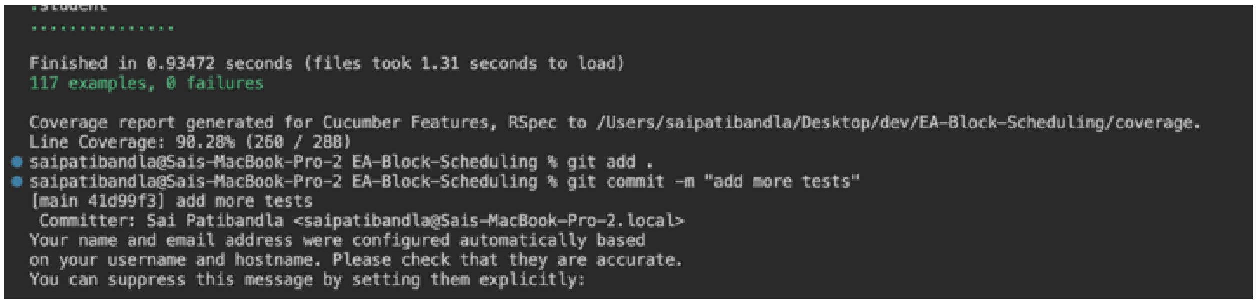
Throughout the project, we practiced a combination of Behavior-Driven Development (BDD) and Test-Driven Development (TDD) methodologies to ensure the quality and reliability of our application. We used **Cucumber** to create feature-level tests that described user interactions and acceptance criteria, aligning development closely with user stories. We wrote basic Cucumber test skeletons before creating the actual features and filled in the details after coding, particularly for mock data and precise UI elements like button text.

We applied **RSpec** for unit and controller-level testing, focusing on building strong backend logic tests. Major application flows had corresponding Cucumber tests, and RSpec tests were written for all critical methods. Some features developed closer to the project deadline still require additional testing. Tests involving multiple redirects and complex login authentication flows proved especially challenging to implement accurately.

If given more time, we would have improved our BDD/TDD process by developing clearer, more detailed feature descriptions upfront. Overall, implementing BDD and TDD practices significantly enhanced our ability to detect bugs early, maintain high code quality, and ensure that the system continued to meet user needs as it evolved.

The final code coverage is around 90%.

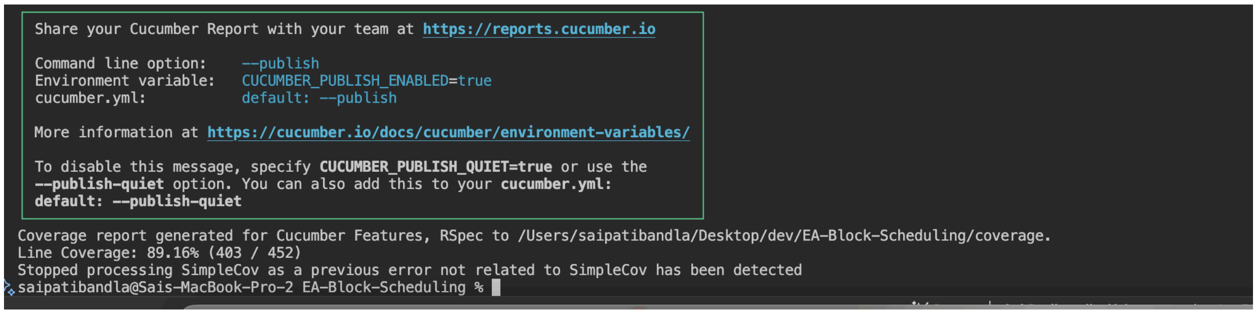
Rspec


```
.student
...............

Finished in 0.93472 seconds (files took 1.31 seconds to load)
117 examples, 0 failures

Coverage report generated for Cucumber Features, RSpec to /Users/saipatibandla/Desktop/dev/EA-Block-Scheduling/coverage.
Line Coverage: 90.28% (260 / 288)
saipatibandla@Sais-MacBook-Pro-2 EA-Block-Scheduling % git add .
saipatibandla@Sais-MacBook-Pro-2 EA-Block-Scheduling % git commit -m "add more tests"
[main 41d99f3] add more tests
 Committer: Sai Patibandla <saipatibandla@Sais-MacBook-Pro-2.local>
Your name and email address were configured automatically based
on your username and hostname. Please check that they are accurate.
You can suppress this message by setting them explicitly:
```

Cucumber


```
Share your Cucumber Report with your team at https://reports.cucumber.io

Command line option:    --publish
Environment variable:   CUCUMBER_PUBLISH_ENABLED=true
cucumber.yml:           default: --publish

More information at https://cucumber.io/docs/cucumber/environment-variables/

To disable this message, specify CUCUMBER_PUBLISH_QUIET=true or use the
--publish-quiet option. You can also add this to your cucumber.yml:
default: --publish-quiet

Coverage report generated for Cucumber Features, RSpec to /Users/saipatibandla/Desktop/dev/EA-Block-Scheduling/coverage.
Line Coverage: 89.16% (403 / 452)
Stopped processing SimpleCov as a previous error not related to SimpleCov has been detected
saipatibandla@Sais-MacBook-Pro-2 EA-Block-Scheduling %
```

9. Configuration Management, Spikes, Branches and Releases

For configuration management, our team maintained a simple branching strategy by primarily working off the **main** branch. The project repository was forked from the original codebase, as we did not have direct access to the original repository. While most changes were pushed directly to the main branch after local testing, we used Pull Requests (PRs) when merging simultaneous feature work to prevent conflicts.

During the early phases, we worked on several spike stories to understand the legacy codebase, configure Heroku deployment, and set up GitHub OAuth integration. Although we experimented with spike tasks, we did not maintain long-term spike branches.

We also established **release tags** for every sprint milestone, resulting in four tags: **Sprint1**, **Sprint2**, **Sprint3**, and **Sprint4**. This helped us mark stable versions that could be referenced later if needed. As a rule, we ensured that all features were tested locally before pushing any changes to GitHub.

Fortunately, we encountered no major issues related to branch management or merges during the project. Having a straightforward branching strategy, combined with disciplined local testing, helped keep our repository clean and minimized integration problems.

10. Production Release Process to Heroku

Deploying the application to Heroku came with a few challenges, particularly because the project was a legacy system without an existing Heroku setup. Although it was a legacy project, there was no functioning production environment, so the team had to set up the Heroku app, configure all services, and deploy the app from scratch. This setup process consumed a significant amount of time that could have otherwise been spent on feature development.

We manually configured environment variables, including Google OAuth credentials, through the Heroku management console. This step was crucial to enable proper authentication workflows. Fortunately, database migrations proceeded smoothly. We utilized Heroku's managed PostgreSQL service, establishing a standalone database instance without major issues.

Our team adopted a regular deployment schedule, pushing updates to Heroku approximately once a week or after the completion of major features. Despite the initial setup challenges, the deployment pipeline worked reliably once established. A key lesson learned was the importance of planning and automating the initial production setup early in the project timeline to minimize disruptions later during active development.

11. Tools/Gems Used

- **GitHub** - Used for version control, collaboration, and project management.
- **SimpleCov** - Used to measure and track test coverage for RSpec and Cucumber tests.
- **RSpec** - Employed for unit and controller-level testing to validate backend logic.
- **Cucumber** - Used for feature-level BDD acceptance testing based on user stories.
- **RuboCop** - Utilized for enforcing Ruby style guide compliance across the codebase.
- **Slack** - Served as the primary communication platform for team coordination.
- **Heroku** - Deployed the production version of the app, managing hosting and PostgreSQL services.
- **Google OAuth** - Integrated for user authentication and authorization.

Overall, the tools and gems we selected were effective in supporting development, testing, security, and deployment activities. There were no major problems or limitations encountered while using these tools. Each contributed to maintaining a high standard of code quality, application security, and streamlined collaboration.

13. Repository Contents and Deployment Process

Repository Structure Overview

- **app/** - Contains main application logic: models, views, controllers, and helpers. Includes `sessions_controller.rb` and `authentication_helper.rb` to manage Google OAuth.
- **config/** - Includes deployment-related configurations:
  - `initializers/omniauth.rb` - Google OAuth credentials configured via environment variables.
- **Gemfile / Gemfile.lock** - Declares and locks gem dependencies including `omniauth`, `omniauth-google-oauth2`, `pg`, and other Rails libraries.
- **Rakefile** - Defines tasks like database migration, seed data loading, etc.
- **app/assets/** - Static assets such as logos used during authentication.
- **README.md** - Provides complete setup instructions including:
  - Prerequisites (Ruby, Rails, PostgreSQL)
  - OAuth configuration
  - Deployment using Heroku
  - Setting admin/student views and permissions
- **config.ru** - For Heroku Rack-based deployment compatibility.
- **.gitignore** - Excludes log, tmp, and node_modules directories from Git tracking.

Local Setup Process

1. Clone the repository.
2. Run bundle install to install all gem dependencies.
3. Run rails db:migrate to set up the database schema.
4. Run rails server to start the application locally.

Note: Developers must manually set up their Google OAuth environment variables for local authentication.

Deployment to Heroku

- Deployments are performed by pushing the main branch to Heroku using git push heroku main.
- After each deployment, database migrations are executed using heroku run rake db:migrate.
- The Heroku environment is configured with necessary environment variables, including the OAuth credentials.

Important Notes for Future Teams

- Ensure that environment variables (especially Google OAuth credentials) are correctly set before running or deploying the application.
- Verify that the Heroku PostgreSQL database is properly linked.
- Follow the documented local setup and deployment steps to avoid issues.

All required scripts, libraries, and configuration files are present in the repository, ensuring that future teams will not face missing dependency issues.

14. Links:

- Github: https://github.com/patibas-tamu/EA-Block-Scheduling
- Heroku App: https://ea-block-scheduling-68f7266eed53.herokuapp.com/
- Project Management: https://github.com/users/patibas-tamu/projects/2

- Demo video: https://www.youtube.com/watch?v=9JMl8d5SoNw

- Demo + Presentation video: https://www.youtube.com/watch?v=aKoVyg2-XZ8