

ACS 56000-02 – Software Engineering

Airtable & Notion 4.0

Sweetwater, Fort Wayne

Final Project Documentation

By

Suchir Santhosh Naik

Manivardhan Sagi

Sumanth Mylar

(Team 02)

Table of contents

Part 1	1
Introduction	1
Purpose	1
Goal	1
Constraints	2
Part 2	3
Project Glossary	3
Part 3	4
Use case diagram	4
Class Diagram	5
Part 4	7
Design and Implementation modules: Use case templates	7
Sequence model	11
State model	12
Part 5	14
Demo Screenshots	14
Part 6	17
Testing report	17
Part 7	20
Tools & Read me	20
Part 8	23
Good practices	23
Part 9	24
Team member contributions	24
Part 10	24
Project Management(Schedule of activities) - Plan Based (Waterfall)	24

Part 1

Introduction

Sweetwater Sound is a retailer that sells music technology and instruments and is one of the most well-known brands in the music industry. This company was founded by Chuck Surack which started off as a mobile recording studio and has since evolved into one of the largest online retailers of music instruments and audio gear in the USA. Sweetwater is currently headquartered in Fort Wayne, Indiana. The company has an array of products, which range from keyboards, guitars, drums to professional recording and audio equipment, which are bought by musicians, recording studios, and sound professionals around the world.

Sweetwater has currently partnered up with Purdue University Fort Wayne by opening the Purdue Fort Wayne Sweetwater Music Center on the Sweetwater corporate campus. This is located on the Sweetwater corporate campus in Fort Wayne. This was funded by Chuck and Lisa Surack from Sweetwater. The campus was set up primarily for students and offers state-of-the-art facilities like recording studios, classrooms, and collaboration spaces. The center offers majors in the Music Industry, Popular music, and also music technology.

The Sweetwater website helps Purdue Fort Wayne students fix appointments, book rooms, create events, and update events for music practice. Our project requires us to fix bugs on the Sweetwater website. Currently, the website has bugs that are preventing users from booking appointments, updating events, and canceling events. It also has a few security-related bugs that need to be fixed. The clients want to prioritize these bug fixes since it is preventing the website from being fully functional.

Purpose

Our purpose for this project is to improve the experience for students when they use the Purdue Fort Wayne Sweetwater Music Center website to book slots for practice, appointments, and fixing the current bugs and restrictions on the website to help the students here at Purdue Fort Wayne book appointments, gears without any difficulties, and have a pleasant experience.

The purpose of our project is to improve the user experience for students utilizing the Purdue Fort Wayne Sweetwater Music Center website. Our objectives include improving the process for booking practice slots and appointments, and fixing existing bugs and restrictions. By doing so, we aim to ensure that students at Purdue Fort Wayne can easily schedule appointments, access necessary gear, and enjoy a seamless user experience.

Goal

The goal of this requirements document is to clearly elucidate the specifications and requirements highlighted by the clients. We aim to fix the 5 bugs discussed by our clients in detail in the next sections, to make the user experience better, improve the security of the website, and also currently remove certain restrictions that prevent students from booking appointments and rooms for practice in certain case scenarios.

The current goals are as follows:

1. Fix the update and cancel event pages to help users update and cancel their previous request
2. Fix the check box in the bookings page which is preventing event bookings
3. Change the timestamp format
4. Fix security issues with the website like data that pops up in the Inspect section when the user logs in.

Constraints

The following technology should be used :

- Airtable for database
- React JS for frontend
- Node Js for backend
- Timeframe – Bugs should be fixed by the end of the Spring semester.

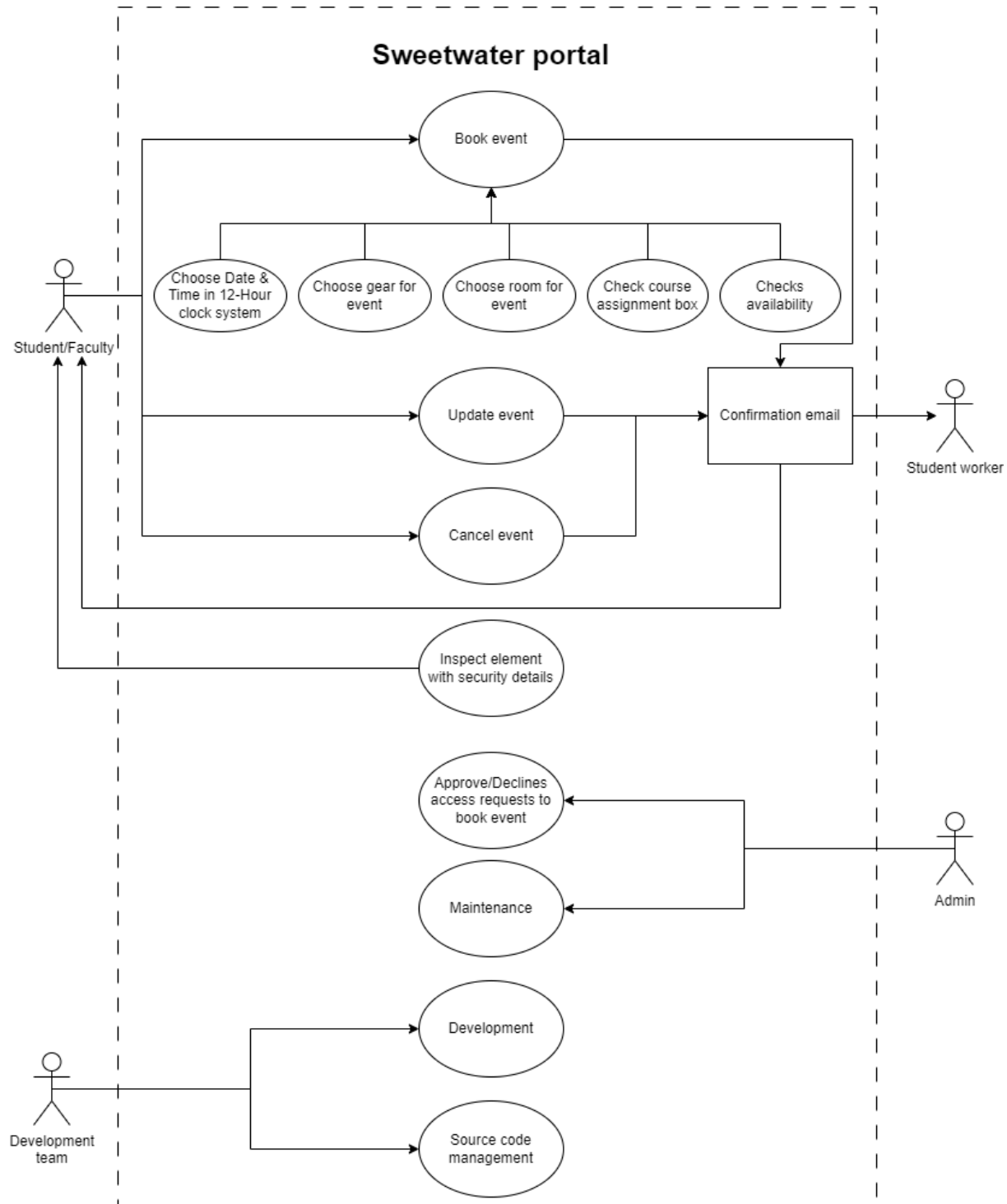
Part 2

Project Glossary

1. **ReactJs** – Frontend framework for developing the website
2. **Air table** – Database for storing the data
3. **Nodejs** – Backend framework for developing the website
4. **Github** – Platform where we will push our code to commit changes or updates and also track changes in the website
5. **Bugs** – Issues in the website
6. **API** (Application Programming Interface): These are protocols that allow different software applications to communicate with each other.
7. **UI (User Interface)**: The visual elements of a software application that users interact with.
8. **UX (User Experience)**: The overall satisfaction that a user has when interacting with a software application
9. **Backend**: This includes databases, servers, and application logic, responsible for processing data and responding to client requests.
10. **Frontend**: This includes the user interface and presentation layer, visible to users and interacts directly with them.
11. **Authentication**: The process of verifying the identity of a user or system

Part 3

Use case diagram



Student/Faculty Interaction:

- They can book an event, during which they choose the date and time (using a 12-hour clock system), select gear for the event, and choose a room.
- They can update or cancel the event.
- The system allows for checking course assignment and availability.

Student Worker Interaction:

- After a booking, a confirmation email is sent to a student worker.

Admin Interaction:

- The admin can approve or decline access requests to book events.
- The admin also oversees maintenance of the portal.

Development Team Interaction:

- Responsible for the maintenance and development of the portal, including source code management.

Class Diagram

SMC Home: Serves as the main landing page with attributes for various URLs that users might need (e.g., booking an event, viewing schedules).

Booking.js: Manages event bookings, holding data such as session title, event type, intended use, and methods to retrieve event lists, room lists, and to send confirmations.

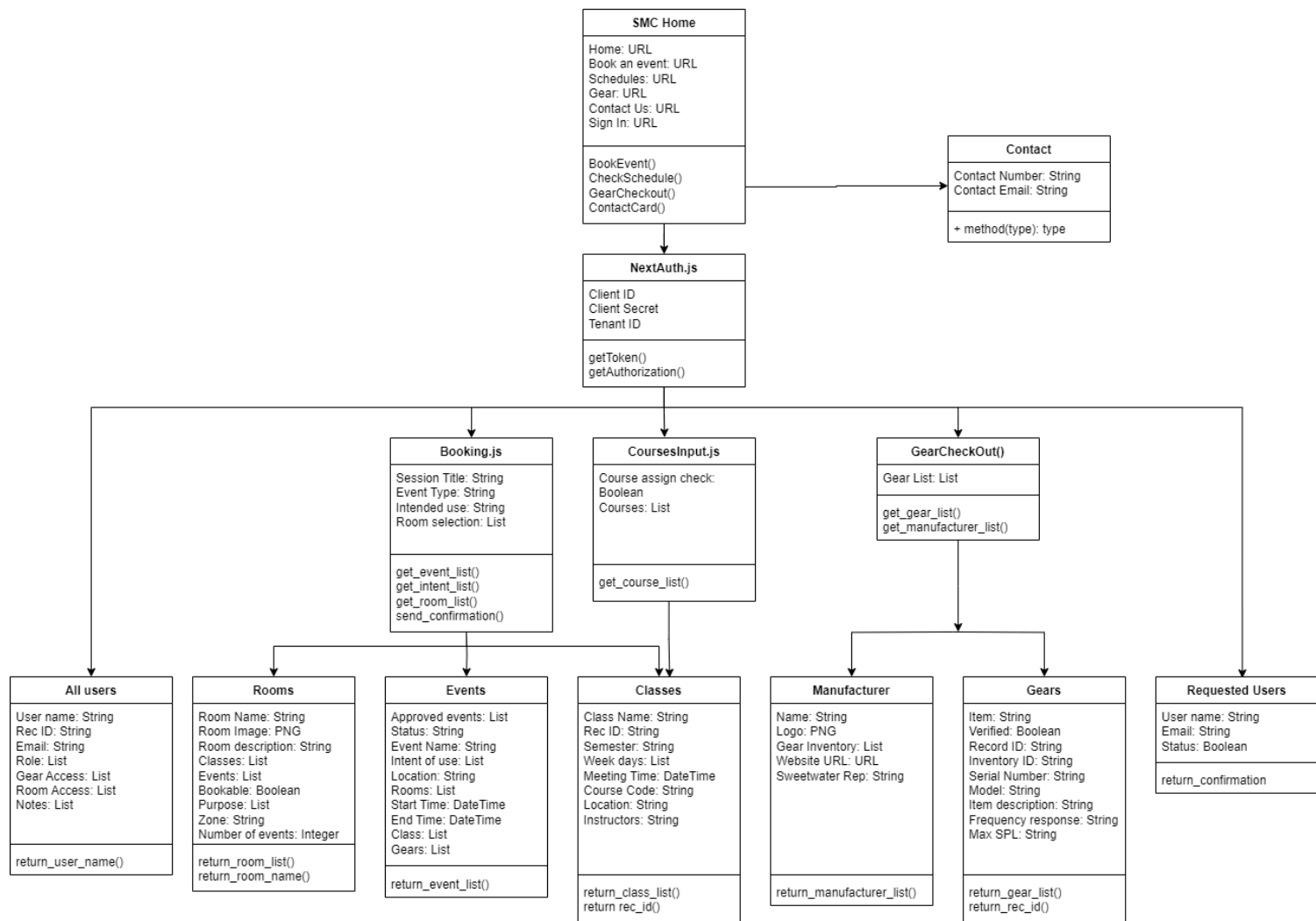
CoursesInput.js: Handles operations related to courses, verifying course assignments, and retrieving course lists.

GearCheckOut: Manages the checkout process for equipment, listing available gear and manufacturers.

NextAuth.js: Deals with authentication services, storing credentials and providing methods for token generation and authorization.

Contact: Stores contact information

Classes, Events, Rooms, Manufacturer, Gears, Requested Users: These are data models storing various attributes relevant to each domain (e.g., room name and description for Rooms, event name and status for Events), and include methods to retrieve lists or specific attributes related to each model.



Part 4

Design and Implementation modules: Use case templates

Use Case #1	Book Event (Unable to book event when “Is this time slot for a course assignment” checkbox is selected)
Brief Description	This use case describes a bug where users are unable to successfully book an event when selecting the "Is this time slot for a course assignment" checkbox.
Actors	Student/Faculty, Student worker, Administrator
Preconditions	1. The student or faculty member must have a PFW account, be registered for a course, and have access to book events. 2. Access to book events will be requested and then approved or declined by the admin.
Main flow	<p>The use case begins when students or faculty members decide to cancel an already booked event at Sweetwater. They initiate this by logging into the portal, selecting the “update event” button, and entering booking ID to cancel the event. Once the event is successfully canceled, a confirmation email will be sent to both the student or faculty member and the student workers involved.</p> <p>Currently, this functionality is not working and needs to be fixed.</p>
Expected result	The system should allow users to book events successfully.
Postconditions	The system should successfully book the event and send confirmation emails to the relevant parties. The booking details should be recorded in the database.

Use Case #2	Update (Unable to update an event)
Brief Description	This use case outlines a bug in the system that prevents users from successfully updating an already booked event.
Actors	Student/Faculty, Student worker
Preconditions	The student or faculty member must access to book an event
Main flow	<p>The use case begins when students or faculty members decide to update an already booked event at Sweetwater. They initiate this by logging into the portal, selecting the “update event” button, and entering booking ID to update the event. Once the event is successfully updated, a confirmation email will be sent to both the student or faculty member and the student workers involved.</p> <p>Currently, this functionality is not working and needs to be fixed.</p>

Expected result	The system should validate the booking ID and allow the user to proceed with updating the event details.
Postconditions	If the use case is successful, a confirmation mail with booking ID and updated event details is sent to both student/faculty and student worker. Also, database entry to be updated.

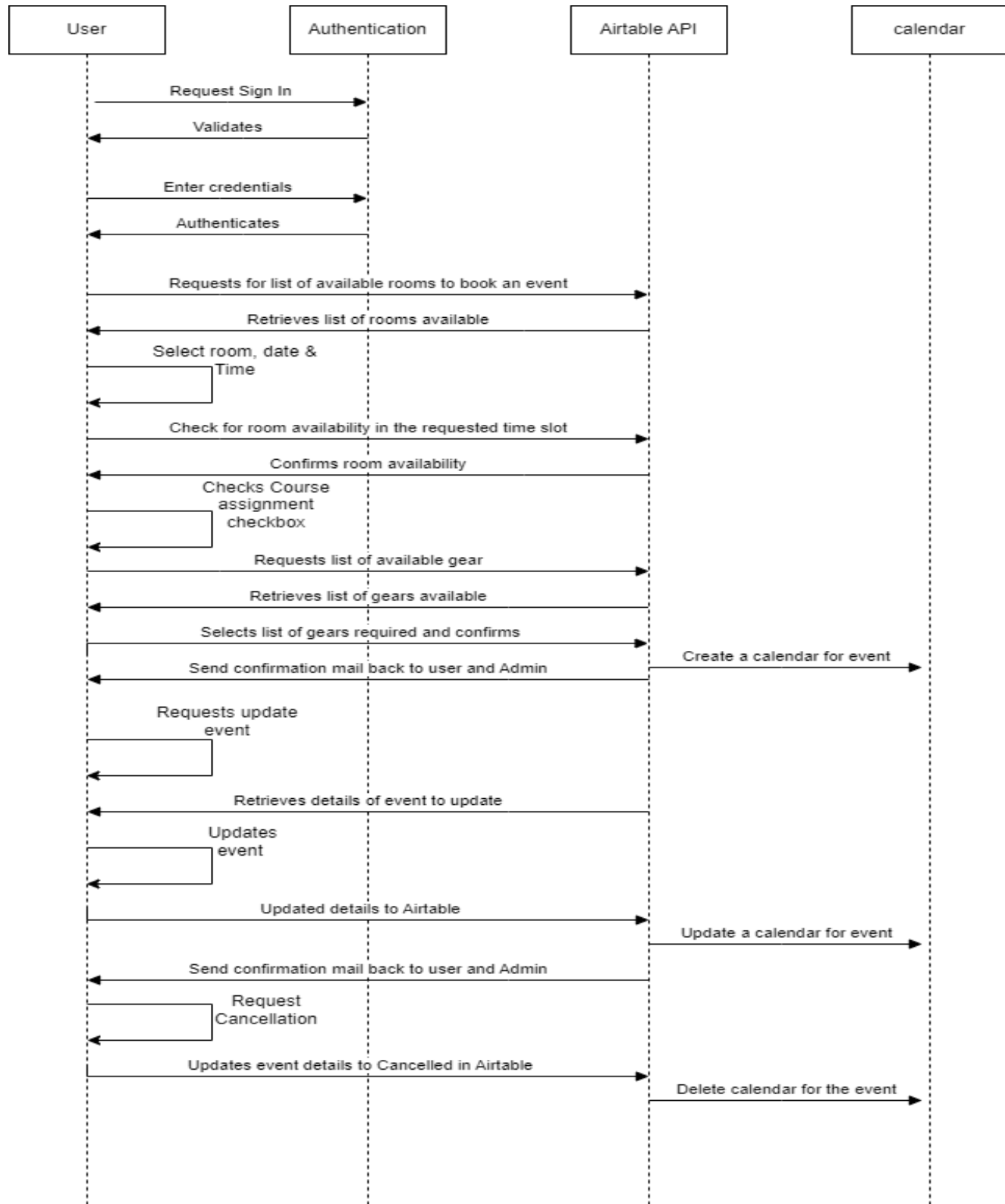
Use Case #3	Cancel Event (Unable to cancel an event)
Brief Description	This use case outlines a bug in the system that prevents users from successfully canceling an already booked event.
Actors	Student/Faculty, Student worker
Preconditions	The student or faculty member must access to book an event
Main flow	<p>The use case begins when students or faculty members decide to cancel an already booked event at Sweetwater. They initiate this by logging into the portal, selecting the “cancel event” button, and entering booking ID to cancel the event. Once the event is successfully canceled, a confirmation email will be sent to both the student or faculty member and the student workers involved.</p> <p>Currently, this functionality is not working and needs to be fixed.</p>
Expected result	The system should validate the booking ID and allow the user to proceed with canceling the event.
Postconditions	If the use case is successful, a confirmation mail will be sent to both student/faculty and student worker. Also, database entry to be updated to “Canceled”.

Use Case #4	Session time upgrade to 12 hour format
Brief Description	This use case outlines the process of upgrading the session time display to a 12-hour format in the system.
Actors	Student/Faculty, Student worker
Preconditions	The student or faculty member must possess the booking ID from the confirmation email received at the time the event was booked.
Main flow	The student/faculty must be able to book an event with session time in 12 hour format rather than military format.
Expected result	The system should allow users to book events successfully with 12 hour format session time.
Postconditions	The system should successfully book the event and send confirmation emails to the relevant parties. The booking details should be recorded in the database.

Use Case #5	Inspect Element (Update - Issue not found in source code)
Brief Description	This use case outlines a bug where backend details are unintentionally displayed when a user inspects elements on the portal website.
Actors	Student/Faculty
Preconditions	The student or faculty member must have access to the Sweetwater booking portal.
Main flow	This use case initiates when the website unintentionally displays data elements of the database when student/faculty inspect the page.
Expected result	Inspecting elements should not reveal any data related details. The user should only see the intended frontend elements.
Postconditions	Inspect elements must be without data elements.

Use Case #6	Source code management (Requirement updated after second call with clients)
Brief Description	<p>This scenario allows the development team to manage the source code of the portal, including the process of merging code from a specific branch (2024.1) into the main branch on GitHub.</p> <p>Update - Create a new branch (2024.4) on SMC's GitHub repository and commit our changes.</p>
Actors	Development team
Preconditions	<p>The Development Team must be granted privileges to access the source code.</p> <p>Update - The Development Team must be granted privileges to create new branch in SMC Github and commit the changes</p>
Main flow	<p>The use case initiates when the Development Team decides to perform development activities, including merging code from a specific branch (2024.1) into the main branch. The team navigates to the GitHub repository. They identify the branch '2024.1' that contains the developed/enhanced features or fixed bugs. The team initiates the merging process to integrate the changes from the '2024.1' branch into the main branch. Conflict resolution, if any, is performed during the merging process.</p> <p>Update - The use case initiates when the Development Team decides to commit requested changes into code repository, by creating a new branch(2024.4) as requested by the clients..The team navigates to the GitHub repository. They create new branch '2024.4'. The team initiates the commit process to integrate the changes from the '2024.1' branch into the new branch.</p>
Expected result	The changes on '2024.1' branch are successfully committed into the new 2024.4 branch on SMC GitHub.
Postconditions	The source code is committed on a new 2024.4 branch, incorporating the developed/enhanced features or fixed bugs from the '2024.1' branch.

Sequence model



User Authentication:

- The user initiates a sign-in request.
- The authentication system validates the user's credentials.
- Upon successful validation, the user is authenticated.

Booking an Event:

- The user requests a list of available rooms to book an event.
- The system retrieves and displays available rooms.
- The user selects a room, date, and time.
- The system checks for room availability in the requested time slot and confirms availability.
- The user requests a list of available gear (equipment) needed for the event.
- The system retrieves and displays the list of gears available.
- The user selects the required gear and confirms.

Event Confirmation and Calendar Update:

- The system sends a confirmation email to the user and the admin.
- It creates an event entry in the calendar.

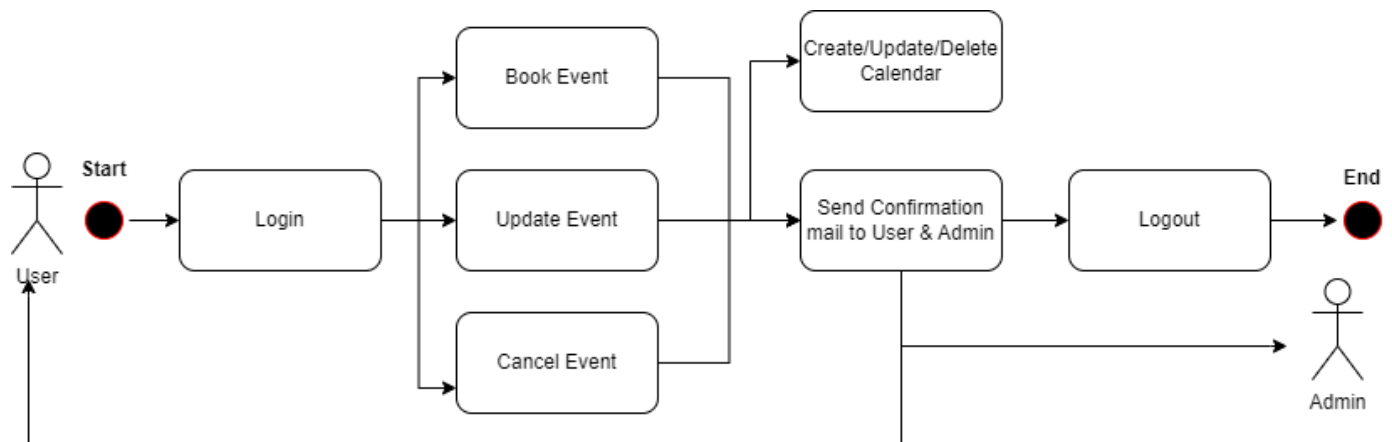
Event Updates:

- If the event details need updating, the user requests an update.
- The user provides updated details, which are sent to Airtable and updated in the calendar.

Cancellation:

- If the event is canceled, the user requests cancellation.
- The system sends a cancellation confirmation email to the user and the admin.
- The event details are updated as canceled in Airtable and the calendar event is deleted.

State model



Start: The process begins when the user starts the event management application.

Login: The user logs into the system, which is the initial step required to access the functionalities related to event management.

Event Operations:

- **Book Event:** After logging in, the user can book a new event. This involves filling out details about the event such as date, time, venue, etc.
- **Update Event:** The user can modify the details of an already booked event. This process is available after booking an event, and the user may change aspects like the event's date, time, or other logistical details.
- **Cancel Event:** The user also has the option to cancel a previously booked event. This includes confirming the cancellation and mentioning the event id.

Calendar Operations:

Create/Update/Delete Calendar: This step involves operations on the calendar itself, likely reflecting any changes made to events (like booking, updating, or cancelling) on a shared or personal calendar.

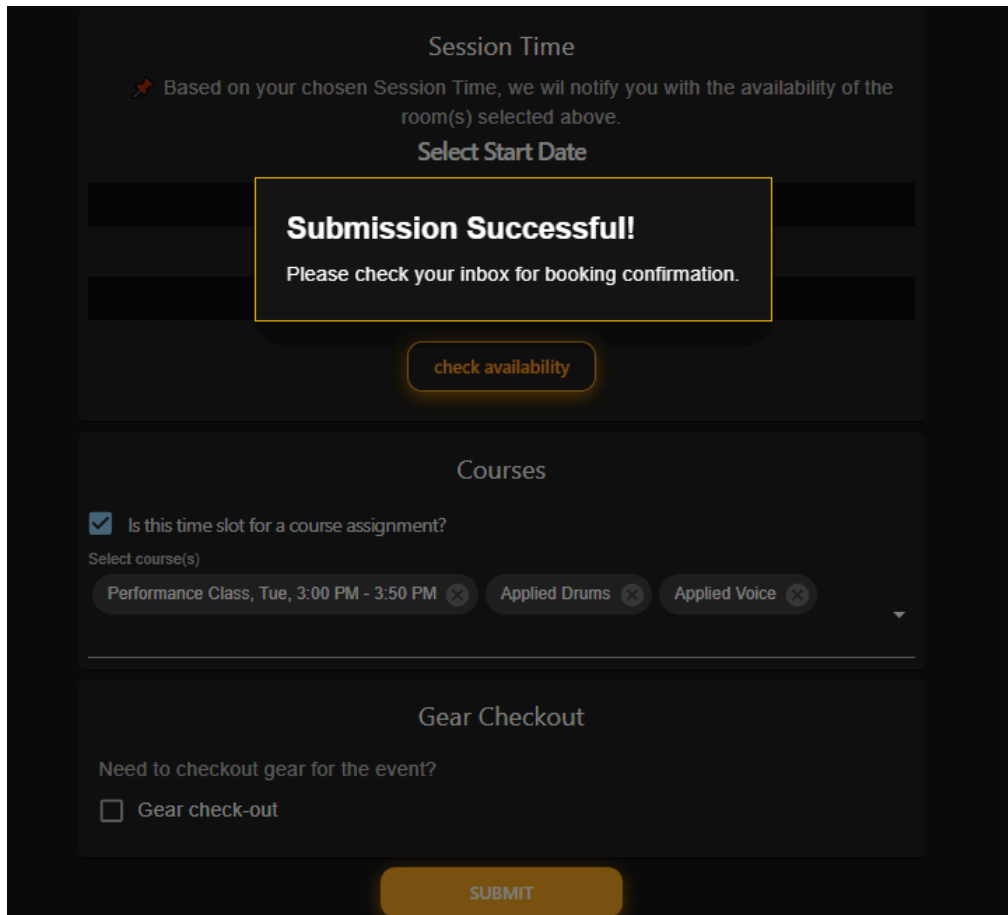
Send Confirmation Mail: After any significant action on an event (booking, updating, or cancelling), a confirmation email is sent to both the user and the admin. This ensures that all parties are informed of the changes or the creation of an event.

Logout: Once all intended actions are completed, the user logs out of the system.

End: The process ends following the logout.

Part 5

Demo Screenshots



Session Time

Based on your chosen Session Time, we will notify you with the availability of the room(s) selected above.

Select Start Date

Submission Successful!
Please check your inbox for booking confirmation.

check availability

Courses

☒ Is this time slot for a course assignment?

Select course(s)

Performance Class, Tue, 3:00 PM - 3:50 PM x Applied Drums x Applied Voice x

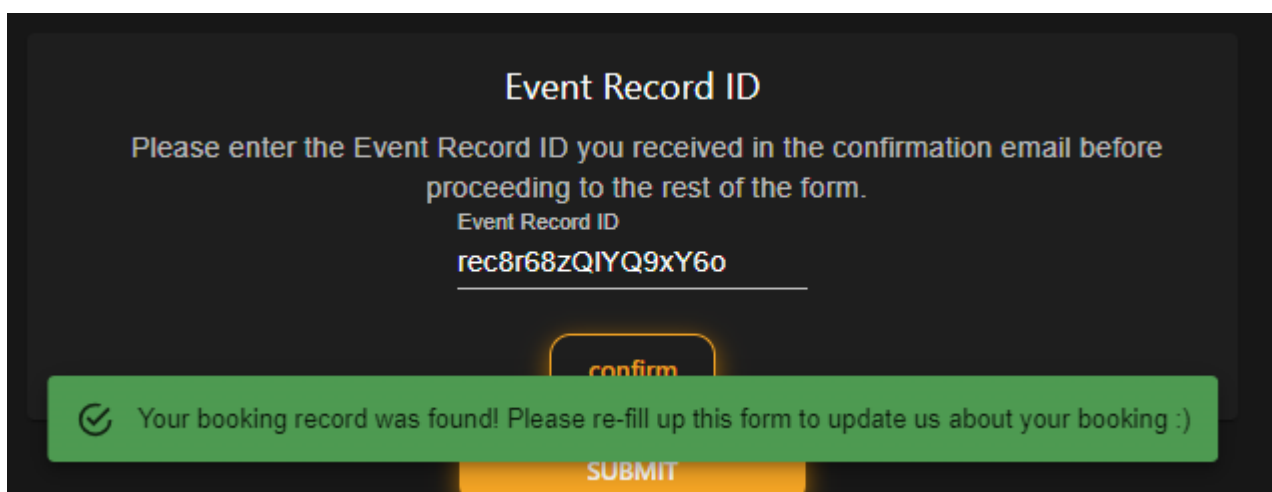
Gear Checkout

Need to checkout gear for the event?

☐ Gear check-out

SUBMIT

Figure 5.1: Screenshot showing the post-changes interface after addressing Use Case #1, where students were unable to create, even when the 'Is this time slot for a course assignment?' checkbox was checked.



Event Record ID

Please enter the Event Record ID you received in the confirmation email before proceeding to the rest of the form.

Event Record ID
rec8r68zQIYQ9xY6o

confirm

✔ Your booking record was found! Please re-fill up this form to update us about your booking :)

SUBMIT

Figure 5.2: Screenshot showing the post-changes interface after addressing Use Case #2, where students were unable to update events.

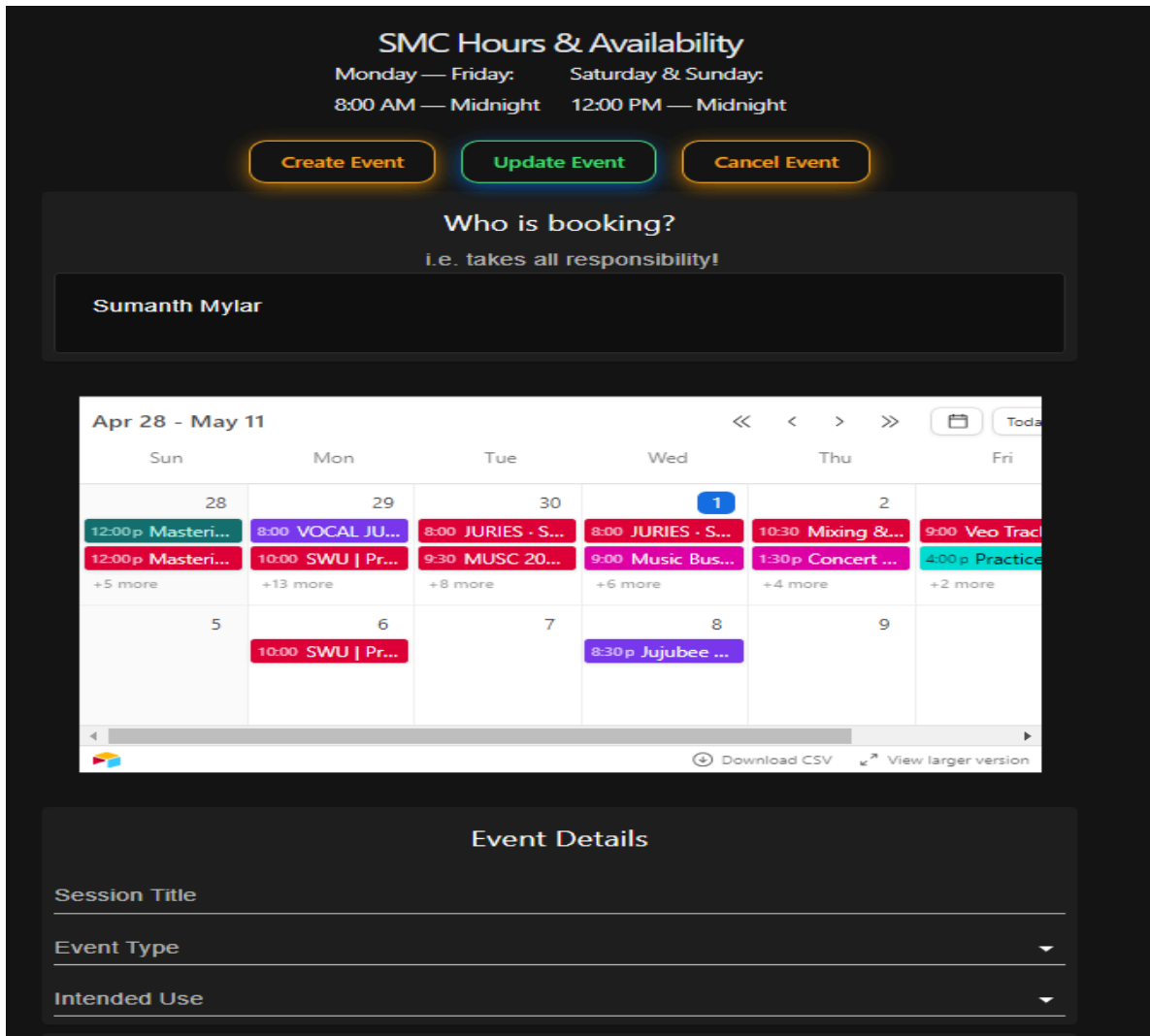


Figure 5.2.1: Screenshot showing the update event form loading after addressing Use Case #2, where students were unable to cancel events.

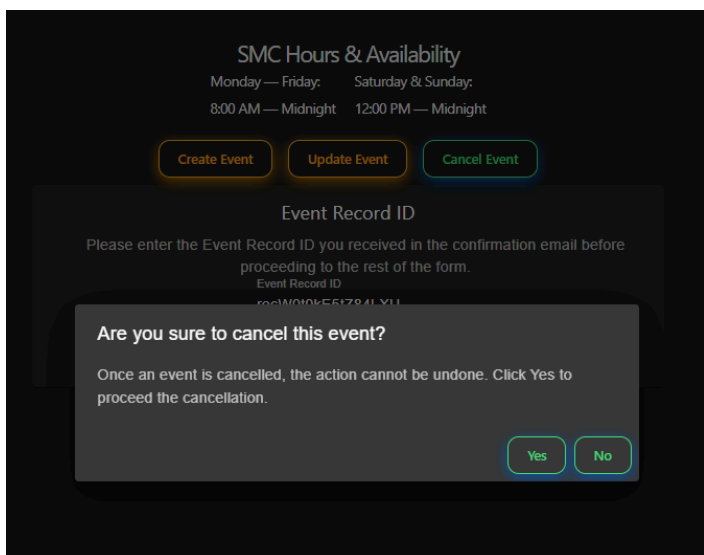


Figure 5.3: Screenshot showing the post-changes interface after addressing Use Case #2, where students were unable to cancel events.

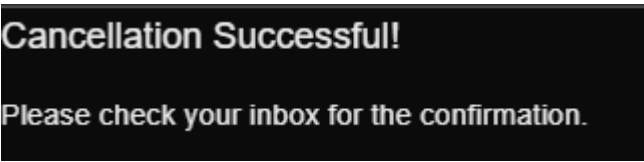


Figure 5.3.1: Screenshot showing the cancellation confirmation message on screen after addressing Use Case #2, where students were unable to cancel events.

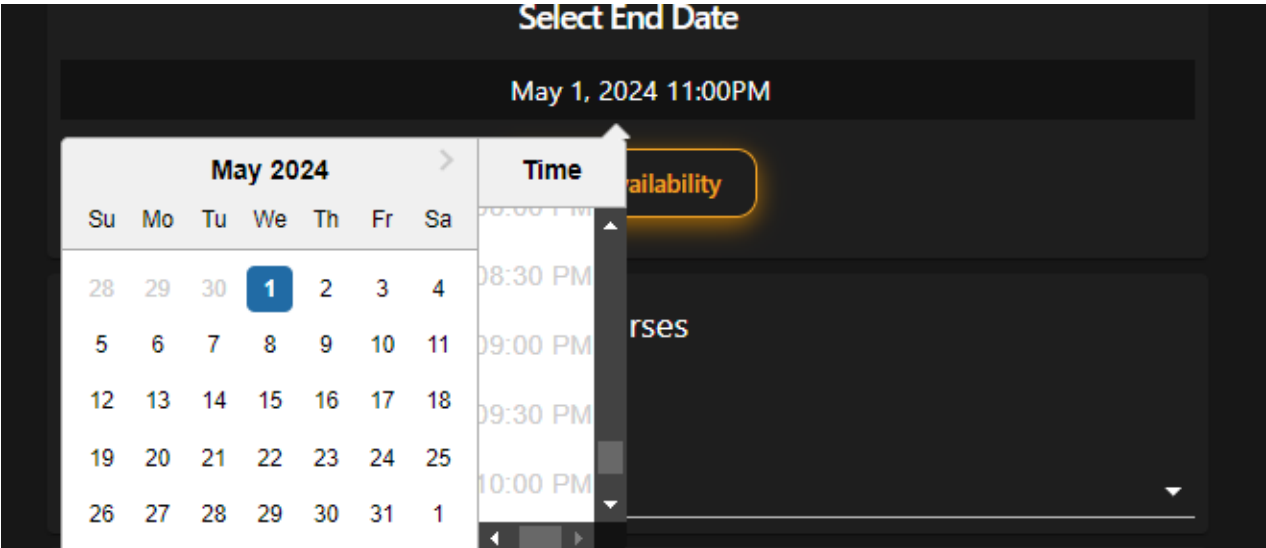


Figure 5.4: Screenshot showing the cancellation confirmation message on screen after addressing Use Case #3, where students were unable to cancel events.

Part 6

Testing report

With the use of Postman, we carried out thorough API testing and UI testing V3.0, which included 3 UI test cases in total. An extensive summary of the testing scenarios used to verify the functionality, performance, and how the system is used by users. For our evaluation method to be consistent and comprehensive, every test case is painstakingly aligned with our testing rubric.

Test Case ID	Test Case	Pre – Condition	Test Steps	Test Data	Expected Results	Actual Result	Status
Book_Event_Test_Case_1	Should be able to book Event	Should be able to login to the website	1.Navigate the Home Page 2.Click on the book event and fill all the required details 3.Select the course and Gear and submit the Event	Event Details	Event should be creates Successfully	Event should be creates Successfully	Pass
Update_Event_Test_Case_1	Should be able to update the old event	Event should be scheduled Before	1. Proceed to Update Event Participant: Mani Course was taken out of the Chosen course Provider and added back to the Available Course The selected course is displayed in databases along with the event pass. 2. Update the	Participant: Mani Event: Practicing Session	Course Choose from Course list is changed .	Database shows the newly selected Course	Pass

			start and finish times Mani, Event: Contributor 3. Current course in the database that's available 4. Confirm the Course and Gear Music Session: "Preso nus: Monitor Station v2 (2 of 2)" Software 5. Check that it is current in Airtable.				
Cancel_Event_Test_Case_1	Should be able to delete event	Event should be scheduled Before	1.Navigate to Delete Event 2. Give the Event Id, then delete the Event	Pre Scheduled Event ID	Event is deleted from Airtable	Event is successfully deleted	Pass
TimePicker_Test_Case_1	Time should be shown in 12 hours format	All other fields are filled out with the correct information	1. Go to the home page. 2. The student/faculty must be able to book an event with session time in 12 hour format rather than military format.	Begin Time: May 1 3:00 pm End 1: May 1 4:00 pm	Time Picker should show the time in 12 hours format	Time Picker should show the time in 12 hours format	Pass
Inspect_Bug_Test_Case_1	The details of other users shouldn't be	Should be able to login to the website	1.Navigate to Home Page 2.Inspect the element and under	Details of the User	Only the user details should be shown	Only the user details should be shown	Pass

	visible to the user		network, response column you should find the details of the User				
--	------------------------	--	---	--	--	--	--

Part 7

Tools & Read me

Development Tools:

- **Airtable:** This tool is used to build databases that store, organize, and allow users to access data.
- **NextAuth.js:** This versatile and fully secure authentication library can be used to synchronize with any Auth service.
- **Authentication via Microsoft Azure Active Directory:** Verifying a user's identity
- **JavaScript:** This programming language supports full stack development.
- **React.js** and **Next.js** are well-known JavaScript frontend frameworks.
- **Github:** Github is a code hosting site. It enables collaboration and is accessible from any location.
- **Vercel:** Vercel is a platform for developers that offers the architecture, procedures, and tools required to create and launch web apps more quickly and without requiring extra configuration. A person selects Next.deployment of a JavaScript application.

Read me:

Register for a Microsoft Identity Platform Account

Register for a Microsoft Identity Platform account by following the instructions at <https://learn.microsoft.com/en-us/entra/identity-platform/quickstart-register-app>.

- Under "Supported Accounts Types," choose "Accounts in this organizational directory only (Purdue University Fort Wayne only - Single tenant)".
- In the Redirect URI section, pick "Web" from the platform dropdown menu, then type "{your vercel website URL}/api/auth/callback/azure-ad" into the bar.

Supported account types

Who can use this application or access this API?

☒ Accounts in this organizational directory (Purdue University Fort Wayne only - Single tenant)

☐ Accounts in any organizational directory (Any Microsoft Entra ID tenant - Multitenant)

☐ Accounts in any organizational directory (Any Microsoft Entra ID tenant - Multitenant) and personal Microsoft accounts (e.g. Sk Xbox)

☐ Personal Microsoft accounts only

[Help me choose...](#)

Redirect URI (optional)

We'll return the authentication response to this URI after successfully authenticating the user. Providing this now is optional and it c changed later, but a value is required for most authentication scenarios.

Web

Register an app you're working on here. Integrate gallery apps and other apps from outside your organization by adding from [Ente](#)

[By proceeding, you agree to the Microsoft Platform Policies](#)

[Register](#)

- After completing the process outlined in the guide, create a "Client Secret." It is important to copy this secret value to a temporary location as it can only be read once.

Using Vercel to Provide Environment Variables

- Create a file named ".env.local" at the project directory's root and define all of the variables below in order to run the code in the local IDE.
- The following variable needs to be set up in Vercel in order to deploy the code in production.

```
1  # This file contains the environment variables
2  NEXTAUTH_SECRET=
3  AZURE_AD_CLIENT_ID=
4  AZURE_AD_CLIENT_SECRET=
5  AZURE_AD_TENANT_ID=
6
7  NEXT_PUBLIC_AIRTABLE_KEY=
8  NEXT_PUBLIC_AIRTABLE_BASE_ID=
```

- After creating the app, you will receive this in the Microsoft Azure platform's app registration screen.

Details

^ Essentials

Display name
[nextauth-test](#)

Application (client) ID
963477db-d80c-4d99-a228-6b98e637ce26

Object ID
c7c2f7ff-b4ef-4c95-9fe0-33c0e9a8d999

Directory (tenant) ID
b7dc318e-8abb-4c84-9a6a-3ae9fff0999f

Supported account types
[My organization only](#)

Client credentials
[0 certificate, 1 secret](#)

Redirect URIs
[1 web, 0 spa, 0 public client](#)

Application ID URI
[api://963477db-d80c-4d99-a228-6b98e637ce26](#)

Managed application in local directory
[nextauth-test](#)

The Application (client) ID and the Directory (tenant) ID in the provided sample file, environment.sample, correspond to environment variables.

- The value you generated in the "Client Secret" step is your AZURE_AD_CLIENT_SECRET.
- Adhering to this manual: Environment Variables: <https://vercel.com/docs/projects/> In order to prepare for deployment, create environment variables.
- NEXT_PUBLIC_AIRTABLE_BASE_ID= NEXT_PUBLIC_AIRTABLE_KEY=
- The Airtable application will be used to obtain the variables.
- Generate the secret Nextauth key: Enter the following command into your command window. As an alternative, you can make one using this website: <https://www.cryptool.org/en/cto/openssl> or go to this website <https://generate-secret.vercel.app/32> and duplicate the amount.
- Assign the generated value to the NEXTAUTH_SECRET value by copying it.
- rand -base64 32 for openssl
- Enter NEXTAUTH_URL=<https://yourwebsite.com>, where the URL is the URL Vercel sent you to access your website.

Starting Website

- Launch the development server first:
- In the terminal, type the following: npm run dev
- To view the outcome, use your browser to open <http://localhost:3000/>.
- Change pages/index.js to begin editing the page. While you alter the file, the page automatically changes.
- You may reach the API routes at <http://localhost:3000/api/hello>. You can modify this endpoint in pages/api/hello.js.
- The path pages/api is mapped to /api/*. Instead of being handled as React pages, files in this directory are handled as API routes.
- This project loads a custom Google font called Inter automatically with the usage of next/font.
- The '.env.local' file is not present on GitHub in order to guard against user data leaks. Please see the part of Read me above for further details on how to create this file. It needs to be adjusted in the project directory's root ".env.local" file. It is also necessary to modify the environment variable on Vercel if this update needs to be deployed.

Part 8

Good practices

Documentation:

- In our project, comments not only elucidate the purpose and logic of code but also offer insights into any potential edge cases, assumptions, or dependencies that might affect its behavior.

Programming Practices:

- Throughout our project, we have consistently adhered to a coding style guide, ensuring uniformity and clarity across the entire codebase.
- We prioritize readability over brevity in our naming conventions, ensuring that future maintainers can easily comprehend the code.

Design:

- We strike a balance between code reuse and maintainability, avoiding overcomplicating code structures solely for the sake of reusability.
- Our modular design approach facilitates the seamless integration of new features while minimizing the impact of changes on existing code.

Design Patterns:

- We actively explore and evaluate new design patterns or architectural paradigms that align with our project's requirements and scalability needs.
- To promote knowledge sharing and facilitate onboarding for new team members, we document and share the design patterns utilized in our codebase.

Cohesion and Coupling:

- Regular code reviews are conducted to identify and address instances of low cohesion or tight coupling between components.
- We foster a culture of refactoring, continuously striving to enhance code quality and reduce dependencies between modules.

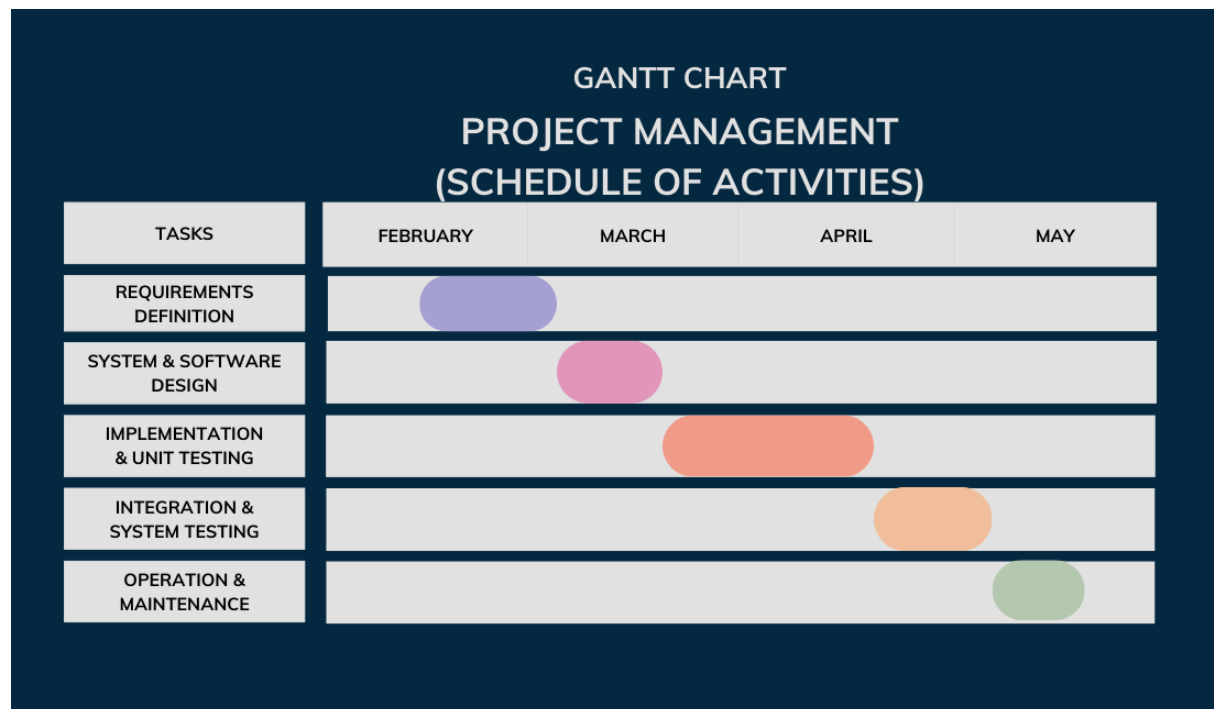
Part 9

Team member contributions

Team Member	Requirements	Design	Documentation	Implementation	Testing	Presentation
Manivardhan	✓	✓	✓	✓	✓	✓
Suchir	✓	✓	✓	✓	✓	✓
Sumanth	✓	✓	✓	✓	✓	✓

Part 10

Project Management(Schedule of activities) - Plan Based (Waterfall)



THANK YOU