

Wishify

# Progress Report 1

COSC 4P02 Group Project

23rd February 2025

**Group Number: 12**

Geoffrey Jensen – 7148710

Nicholas Parise – 7242530

Ethan Brennan – 6881411

Stephen Stefanidis – 7140030

Justin Thomas Bijoy – 7123550

Anthony Medico – 3383775

# Table of Contents

<b>Table of Contents</b>	<b>1</b>
<b>Introduction</b>	<b>3</b>
<b>Sprints</b>	<b>3</b>
Sprint 1 – Completed	3
Sprint 2 – Completed	3
Sprint 3 – In Progress	3
<b>Frontend Progress</b>	<b>5</b>
Landing Page	5
Wishlist Home Page	6
Wishlist Page	7
Sign-Up and Login Pages	8
Profile Settings Page	10
<b>Backend Progress</b>	<b>11</b>
Authentication & User Management	11
Wishlist Management	11
Event Management	12
User Profile & Interaction	12
Additional Features	12
Current Backend Status	13
Pending Tasks	13
Testing	13
<b>Tools</b>	<b>14</b>
Jira	14
Discord	15
GitHub	15
<b>Meetings</b>	<b>16</b>
Meeting 1: Weekly Meeting for Project	16
Meeting 2: Sprint Retrospect & Kick off Sprint 2	16
Meeting 3: Weekly Meeting for Sprint 2	16
Meeting 4: Progress Check-in with TA	17
Meeting 5: Sprint Retrospect & Kick off Sprint 3	17
<b>Team Contribution</b>	<b>18</b>
GitHub Contributions	18
Work Reporting	18
Jira Work Assignment	19
Sprint 1	19
Sprint 2	20
Sprint 3	21

## Introduction

This progress report describes the creation of the Wishlist management platform - **Wishify**. Throughout the duration of three development sprints, it offers an in-depth review of the project's progress, focusing tasks that have been completed, current work, and future goals. The frontend and backend solution, including capabilities for profile customization, Wishlist and event management, and user authentication, are covered in the report. It also keeps track of team contributions, development tools, and meeting summaries to track teamwork and progress.

## Sprints

### Sprint 1 – Completed

1. Design UI for all pages – Completed
2. Finish database UML – Completed
3. Setup PostgreSQL database – Completed
4. Complete placeholder of Wishlist frontend – Completed
5. Complete placeholder of sign up and sign in frontend – Completed
6. Complete placeholder of list of Wishlist – Completed
7. Create a Landing Page – Completed
8. Complete placeholder profile page – Completed
9. Complete placeholder of event frontend – Pushed to Sprint 2
10. Complete placeholder of list of events – Pushed to Sprint 2

### Sprint 2 – Completed

1. Setup production environment – Completed
2. Implement frontend for sign in pages – Completed
3. Complete placeholder of list of events – Completed
4. Authentication API Endpoints – Completed
5. Research Pop Up Library – Completed
6. Dummy Items for Database – Completed
7. Wishlist API Endpoints – Completed
8. Event API endpoints – Completed
9. Design API specification – Pushed to Sprint 3
10. User / I want to create and modify accounts – Pushed to Sprint 3
11. Implement frontend for home page – Pushed to Sprint 3
12. Complete placeholder of event frontend – Pushed to Sprint 3
13. CSS/Styling of Pages – Pushed to Sprint 3
14. Express API Testing Framework – Pushed to Sprint 3

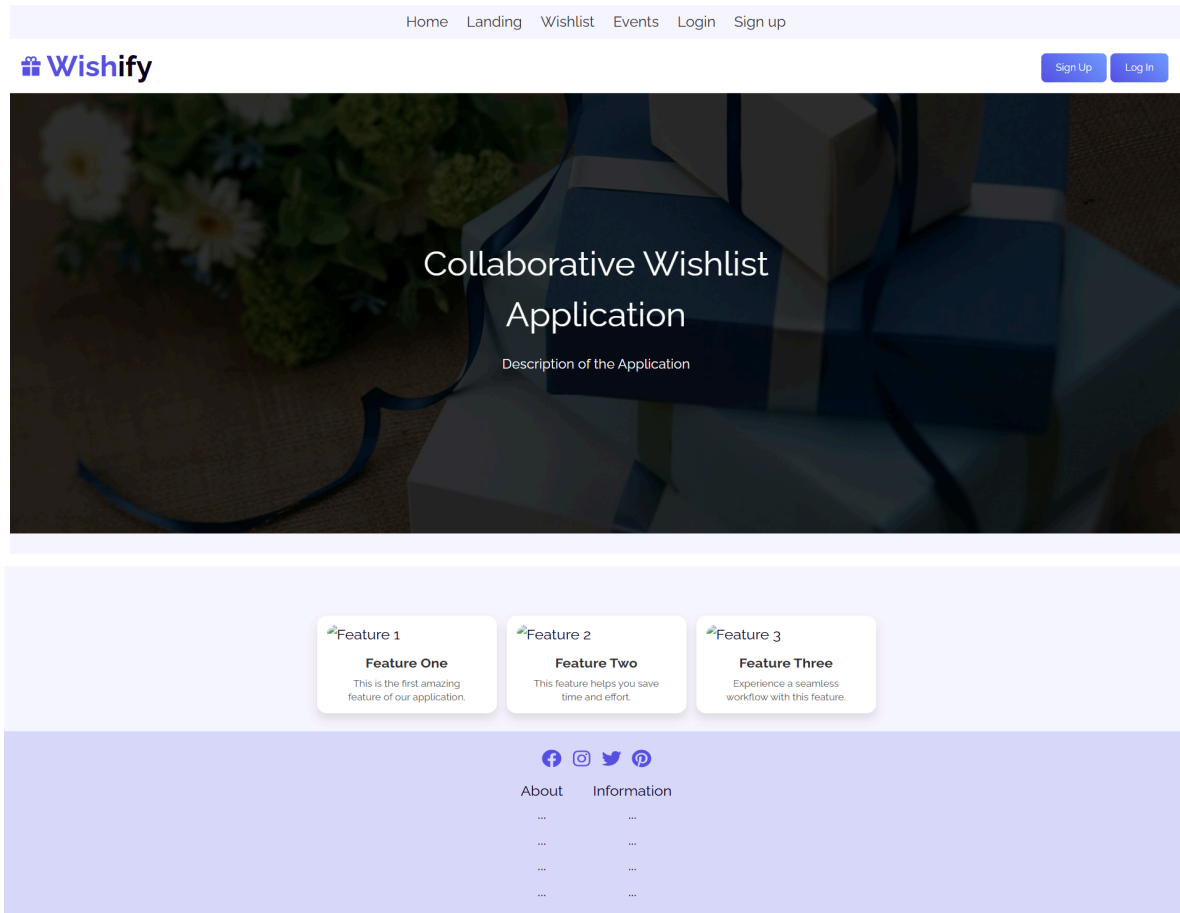
## Sprint 3 – In Progress

1. Complete placeholder of event frontend
2. Implement frontend for home page
3. CSS/Styling of Pages
4. create and modify user accounts
5. Design API specification
6. Express API Testing Framework
7. User / I want to create Wishlist
8. User / I want to delete Wishlist
9. Party Host / I want to set deadlines on Wishlist
10. Event Planner / I want to duplicate Wishlist
11. Research OAuth / Firebase
12. Wishlist Contributor / I want to be able to sort a list by price, priority and contribution status
13. Brainstorm testing framework
14. Wishlist Contributor / I want a blind feature
15. List of Prefix words for profile interests/ and implement to frontend
16. Design Favicon Logo
17. Navigation Bar for all pages

# Frontend Progress

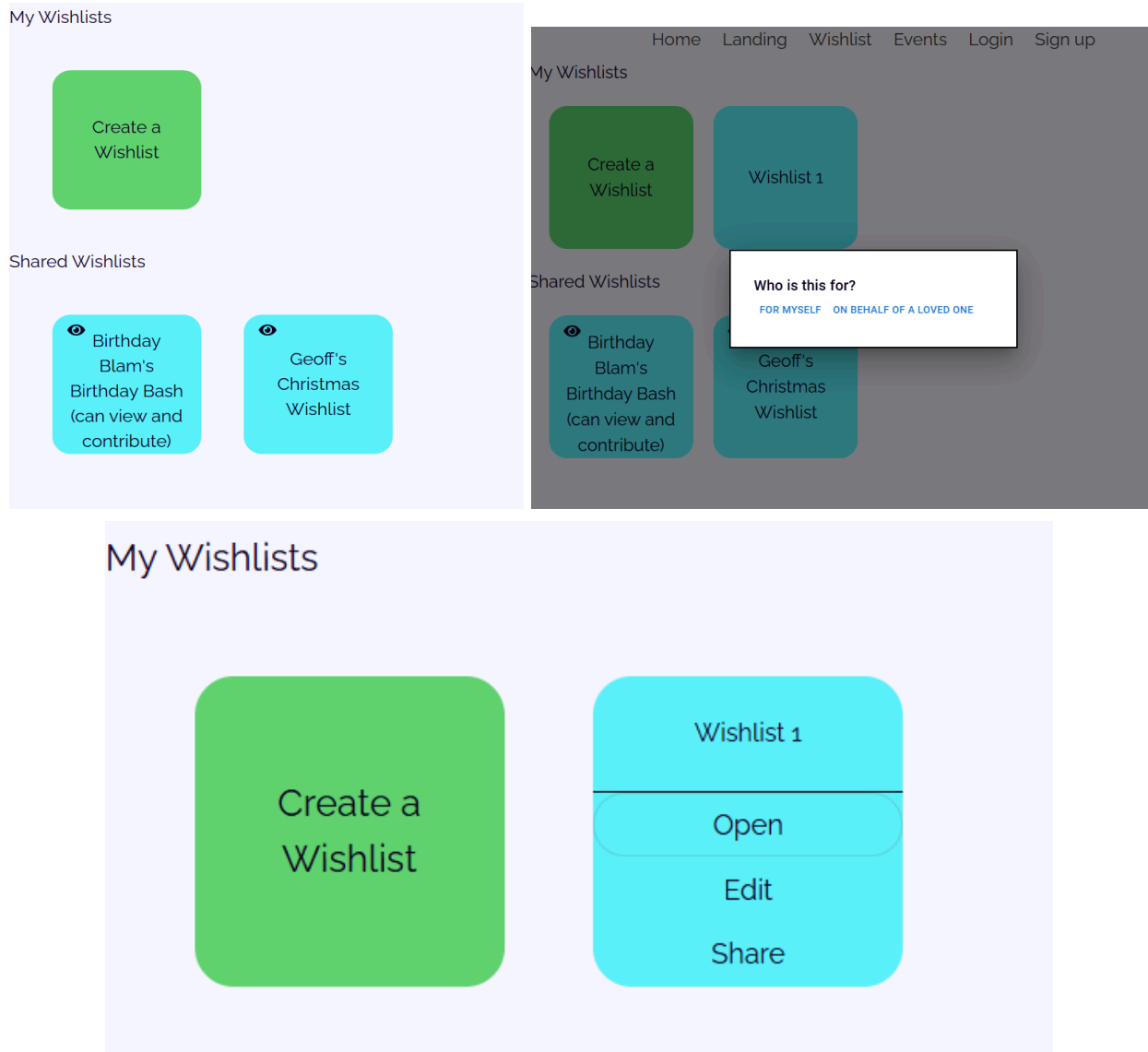
## Landing Page

The landing page serves as the first point of interaction for users who are not logged in. It provides a comprehensive overview of everything Wishify has to offer, giving visitors a clear understanding of its features and benefits.



## Wishlist Home Page

The Wishlist section is where you can view and manage all your saved Wishlist in one place. Here, you'll have full control over your lists with the ability to add, edit, and delete Wishlist as needed.




## Wishlist Page


When you enter a Wishlist, you'll be redirected to the Wishlist Details page, where you can fully manage its contents.

### Geoff's Christmas Wishlist

This is my wishlist for Christmas 2026



Jensen family Christmas

 yesterday

 100 Polar Express Way, North Pole

Sort by

Priority





**Nintendo Switch** Fully Reserved

1

Price: \$450.00

Quantity: 0 available / 1 total





**Nintendo Switch 2**

2

Price: \$500.00

Quantity: 1 available / 1 total





**PS5 Pro**

3

Price: \$960.00

Quantity: 1 available / 1 total



**Hot dogs** Partially Reserved

3

Price: \$5.00

Quantity: 2 available / 5 total

## Sign-Up and Login Pages

Creating an account is the first step to creating and sharing wish lists on Wishify. The sign-up page allows you to create an account with your email address and choose a display name. It provides visual hints and feedback for users to ensure they are entering valid information into the form. When the page is first loaded, the fields offer hints, and the Create Account button is disabled until the form is filled out correctly. As users begin typing into the input fields, a red or green outline becomes visible indicating if the input is valid. Once all fields are valid, the Create Account button becomes active for them to submit the form. Upon submitting, either a success or error message is displayed based on the response from the server. If the account was created successfully, a success message informs them and includes a link for the user to navigate to the login page. If the email address is already in use, an appropriate error message is displayed for that as well. Custom error messages for each type of server response have been implemented. Example sign up page screenshots shown below for different scenarios.

Create your account

Email

bob@wishify.com

Display Name

John Doe

Password

Password Show

Confirm Password

Confirm password Show

Create account

Already have an account? [Sign in here](#)

Create your account

Email

test@test.

Display Name

Mr. Test

Password

..... Show

Confirm Password

Confirm password Show

Create account

Create your account

Email

test@test.com

Display Name

Mr. Test

Password

..... Show

Confirm Password

..... Show

Create account

Confirm password

Hide

Create account

Account successfully created!  
Please proceed to the **login** page.

Already have an account? [Sign in here](#)

Confirm Password

..... Show

Create account

Account with that email address  
already registered.

Already have an account? [Sign in here](#)



The login page is similar, where the button is disabled until a valid email and password (at least 8 characters) is entered. A successful login automatically redirects the user to their home page, while incorrect credentials result in an appropriate error message.

The image displays two versions of a login form side-by-side. The left form is in a 'successful' state, showing a light purple border. It has a title 'Sign in to continue', an 'Email' field with the value 'johndoe@wishify.com', a 'Password' field with a 'Show' toggle, a disabled grey 'Login' button, a blue link 'Forgot password?', and a link 'Don't have an account? Sign up here'. The right form is in an 'error' state. It has a password field with masked characters and a 'Show' toggle, an active blue 'Login' button, a red error message box stating 'Incorrect email or password. Please try again.', and a blue link 'Forgot password?'.

Sign in to continue

Email

johndoe@wishify.com

Password

Password Show

Login

[Forgot password?](#)

Don't have an account? [Sign up here](#)

..... Show

Login

Incorrect email or password. Please try again.

[Forgot password?](#)

Both pages are fully functional with input validation, proper handling of buttons (disabled during authentication), appropriate visual hints and feedback, and redirecting a successful login to the user's homepage.


**Future improvements:** the team is considering the use of OAuth and logging in via social media credentials. Styling tweaks will be made in future sprints, including responsive layouts that work well with mobile.

## Profile Settings Page

The profile settings page is where the user can update various account settings as well as personal information. They may edit their profile picture, display name, biography, likes and dislikes, email and password. Here the user can also delete their account.

Likes and dislikes will be a predetermined list that the user can select from to include in their profile. This lets other users know what categories of gifts would be best for them, as well as categories to avoid.

### Profile settings

[Update Picture](#)

---

Display name:

John Doe

[Edit](#)

---

Biography:

Lorem ipsum dolor sit amet, consectetur adipiscing elit. Nulla quam velit, vulputate eu pharetra nec, mattis ac neque.

[Edit](#)

---

Likes:

Books

Blu-rays

Video games

item 4

item 5

item 6

item 7

item 8adsaadfjk

item 9

item 10

item 11

item 12

[Add/Remove](#)

---

Dislikes:

Cleaning Supplies

[Add/Remove](#)

---

Email address:

johndoe@wishify.com

[Edit](#)

---

Password:

Password strength: weak

[Change Password](#)

---

[Delete account](#)

The UI shown above is implemented, but functionality is being implemented currently during sprint 3. Each edit button will display a modal where the user can update that particular field. For likes and dislikes, a searchable list of predetermined categories will be presented to the user where they can select one or more. Proper validation will be implemented for all fields.

# Backend Progress

## Authentication & User Management

- 1. User Registration (POST /auth/register):**
  - Allows creating a new user account with required information (email, password, display name, picture).
  - Response handling includes 201 (created), 400 (bad request for missing fields), and 409 (email conflict).
- 2. User Login (POST /auth/login):**
  - Facilitates user login with email and password, returning a JWT token for authentication.
  - Handles 200 (successful login), 400 (missing credentials), 401 (invalid credentials), and 500 (server error) responses.
- 3. User Logout (POST /auth/logout):**
  - Validates the provided JWT token and logs out the user, invalidating the token.
  - Handles 200 (successful logout), 401 (no token), and 500 (server error).
- 4. Get User Profile (GET /auth/me):**
  - Returns the user's profile based on a valid JWT token.
  - Handles 200 (successful), 401 (unauthorized), 404 (user not found), and 500 (server error).

## Wishlist Management

- 1. Get All Wishlists (GET /wishlists):**
  - Fetches a list of all wishlists the user is associated with (based on their event membership).
  - Handles 200 (successful), 401 (unauthorized), and 403 (invalid token) responses.
- 2. Create a Wishlist (POST /wishlists):**
  - Allows users to create a new wishlist with details like name, event, description, and image.
  - Handles 201 (created), 400 (missing name), and 500 (server error).
- 3. Get Wishlist Details (GET /wishlists/:id):**
  - Retrieves detailed information of a specific wishlist by its ID.
  - Handles 200 (successful), 401 (unauthorized), 403 (invalid token), and 404 (wishlist not found) responses.
- 4. Delete Wishlist (DELETE /wishlists/:id):**
  - Enables the deletion of a wishlist by its owner.
  - Handles 200 (successful), 401 (unauthorized), 403 (not the owner), and 500 (server error).
- 5. Update Wishlist (PUT /wishlists/:id):**
  - Allows the owner to update the details of their wishlist.
  - Handles 200 (successful update), 401 (unauthorized), 403 (not the owner), 404 (wishlist not found), and 500 (server error).

## Event Management

- 1. Get All Events (GET /events):**
  - Returns a list of events the user is associated with.
  - Handles 200 (successful), 401 (unauthorized), 403 (invalid token), and 500 (server error).
- 2. Create an Event (POST /events):**
  - Allows users to create a new event with details such as name, description, image, address, and city.
  - Handles 201 (created), 400 (missing name), 401 (unauthorized), 403 (invalid token), and 500 (server error).
- 3. Get Event Details (GET /events/:id):**
  - Retrieves detailed information of a specific event by its ID.
  - Handles 200 (successful), 401 (unauthorized), 403 (invalid token), 404 (event not found), and 500 (server error).
- 4. Update Event (PUT /events/:id):**
  - Allows the owner to update event details.
  - Handles 200 (successful update), 401 (unauthorized), 403 (not the owner), 404 (event not found), and 500 (server error).
- 5. Delete Event (DELETE /events/:id):**
  - Allows event owners to delete their events.
  - Handles 200 (successful), 401 (unauthorized), 403 (not the owner), and 500 (server error).

## User Profile & Interaction

- 1. Edit User Profile (PUT /users/:id):**
  - Allows users to update their profile information (display name, biography, picture, etc.).
  - Handles 200 (successful update), 401 (unauthorized), 400 (invalid email), and 500 (server error).
- 2. Delete User Account (DELETE /users/:id):**
  - Provides the functionality for users to delete their account, requiring password verification.
  - Handles 200 (successful deletion), 401 (unauthorized), and 500 (server error).
- 3. Get User Information (GET /users/:id):**
  - Retrieves user details by their ID.
  - Handles 200 (successful), 404 (user not found), and 500 (server error).

## Additional Features

- 1. Likes/Dislikes:**
  - Endpoint for managing user preferences related to likes/dislikes.
  - Returns an array of tags representing these preferences.
- 2. Sharing Wishlists:**
  - Provides functionality for sharing a wishlist link with others. Users can get a shareable link and send it via email.

## Current Backend Status

- **Authentication & User Management:** Fully implemented, with login, logout, and profile functionalities working.
- **Wishlist & Event Management:** Core functionality is complete (create, update, delete, fetch), but further refinement is needed on error handling and permission checks.
- **User Profile Updates:** Account creation, profile updates, and deletions are handled, with proper error messages.
- **Likes/Dislikes Management:** Basic structure implemented.
- **Sharing Features:** Wishlist sharing via links and emails needs implementation for full functionality.

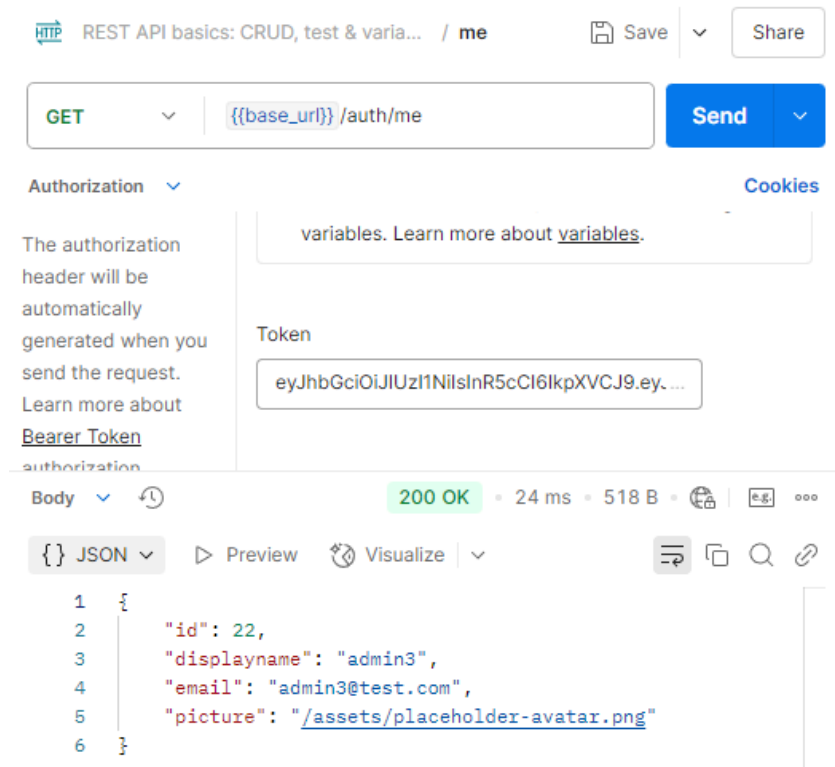
## Pending Tasks

- **User List (GET /users):** This feature is not implemented yet and returns sample data.
- **Event Management Enhancements:** Further validation for user event membership and ownership may need refining.

Overall, the backend is in a functional state but requires final refinements and testing for smoother user experience.

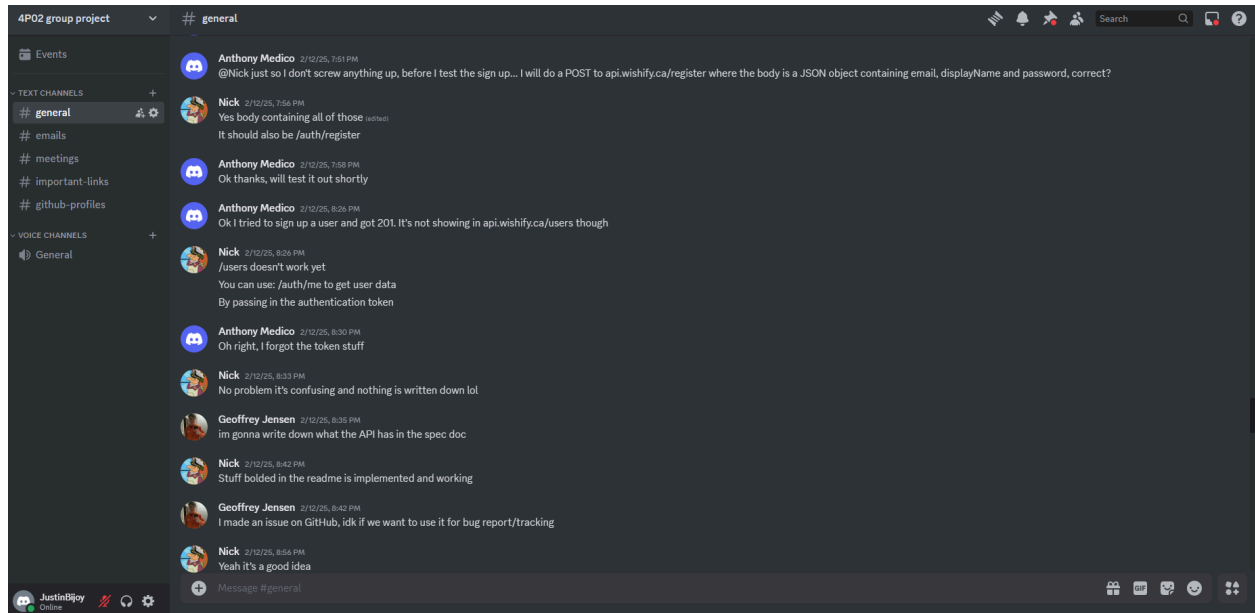
## Testing

Currently testing is being done manually with postman, shown here is the API sending user data. Eventually this testing will be done automatically but in early development it's manual.

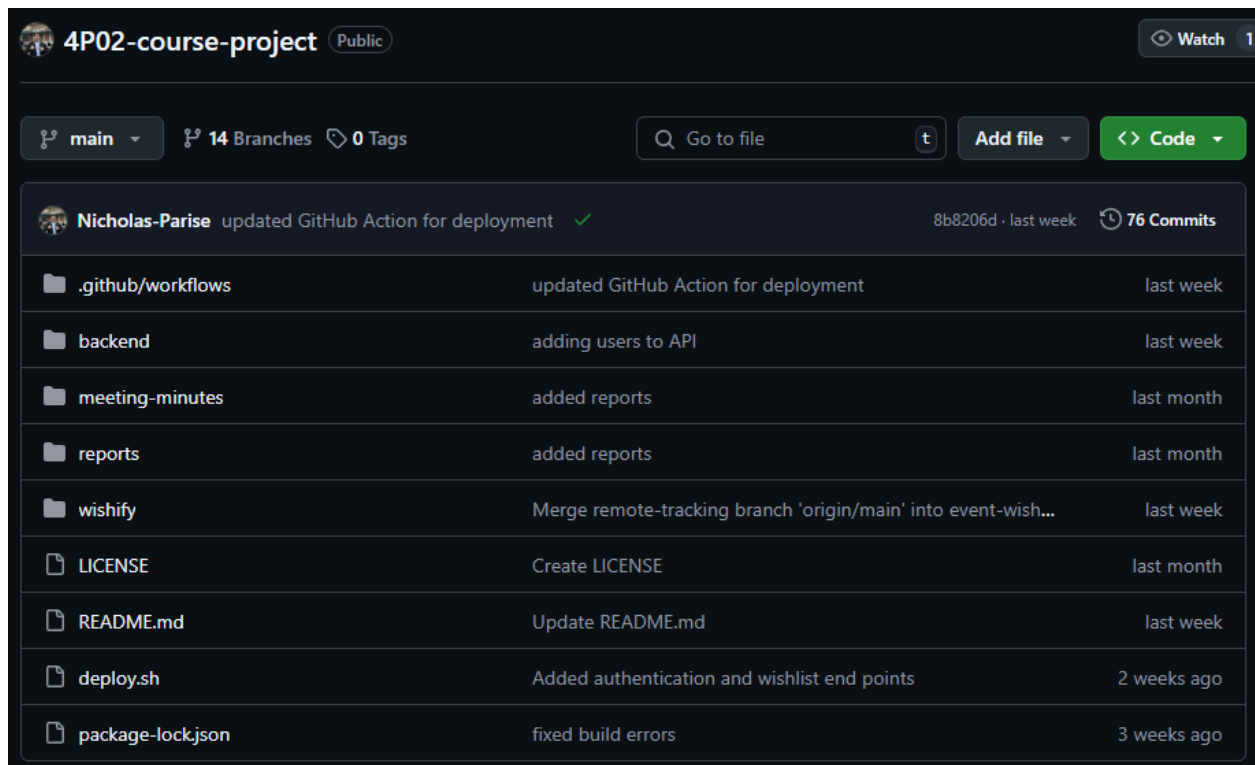




# Discord



# GitHub



# Meetings

## Meeting 1: Weekly Meeting for Project

- Date: 24th January 2025
- Time: 11:00 AM – 2:00 PM
- Venue: Library Room STH503C
- **Key Points:**
  - **Sprint Plan Review:** Sprint 1 scope discussed; all members aligned.
  - **Database UML and Structure Discussion:** Confirmed tables and relationships (Users, Sessions, Events, etc.).
  - **Task Assignments:** Team Members were assigned specific web pages to design (e.g., Sign-in page, Landing page).
  - **Competitor Website Review:** Reviewed features from GoWish.com and discussed improvements for Wishify.
- **Next Steps:** Design assigned pages, set up backend infrastructure, and schedule follow-up meetings.

## Meeting 2: Sprint Retrospect & Kick off Sprint 2

- Date: 31st January 2025
- Time: 11:00 AM – 2:00 PM
- Venue: Library Room STH503C
- **Key Points:**
  - **Sprint Retrospect:** Discussed Sprint 1 outcomes, improvements, and blockers.
  - **Jira Update:** Completed and pending tasks updated in Jira.
  - **Webpages Review:** Members presented designed pages; feedback provided for improvements.
  - **API Specification Discussion:** Defined data requirements and API response formats for each page.
- **Kick off Sprint 2:** New tasks assigned, focusing on design refinement and backend integration.

## Meeting 3: Weekly Meeting for Sprint 2

- Date: 7th February 2025
- Time: 11:00 AM – 2:00 PM
- Venue: Library Room STH503C
- **Key Points:**
  - **Completed Work Review:** Team updates on progress.
  - **Home Page Design Ideas:** Ethan presented design ideas for the homepage.
  - **Profile Pages Adjustments:** Minor design tweaks made for usability.



- CSS Adjustments: Discussed changes for consistency and responsiveness.
- API Specifications Review: Reviewed API elements and clarified documentation.
- Database Requirements: Finalized schema, ensuring proper data relationships.
- **Next Steps:** Start API implementation, continue with CSS changes, and complete webpage designs.

## Meeting 4: Progress Check-in with TA

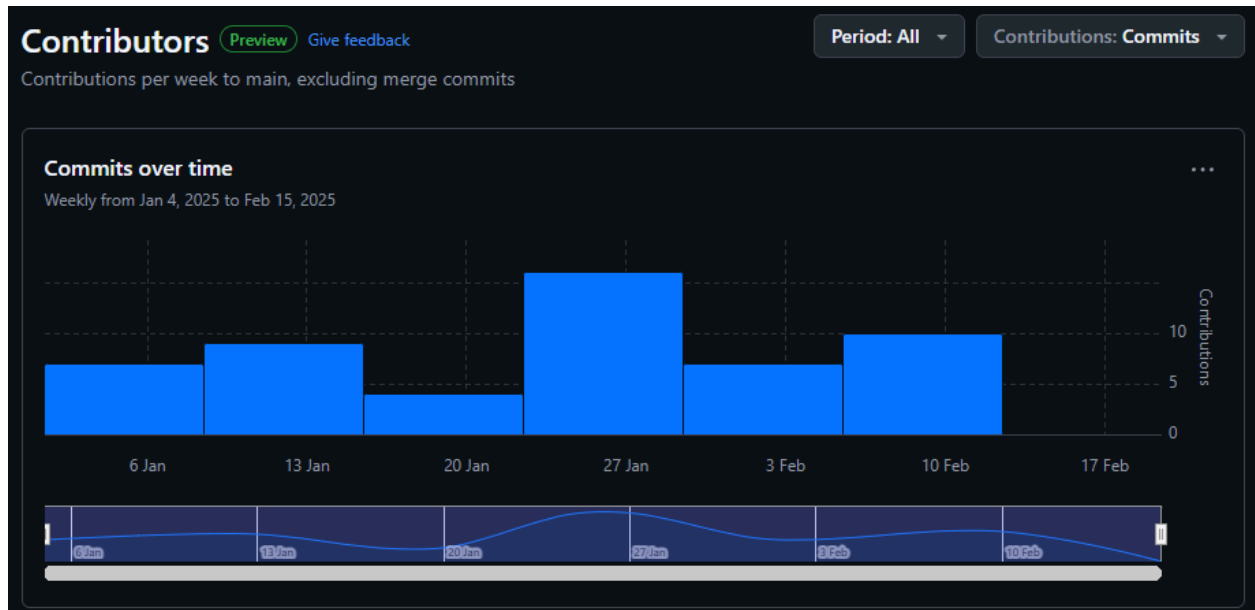
- Date: 13th February 2025
- Time: 12:00 PM – 1:00 PM
- Venue: Online (Microsoft Teams)
- Key Points:
  - **Progress Report:** Presented progress to TA and received feedback.
- **Next Steps:** Continue with webpage and API work, improve design consistency.

## Meeting 5: Sprint Retrospect & Kick off Sprint 3

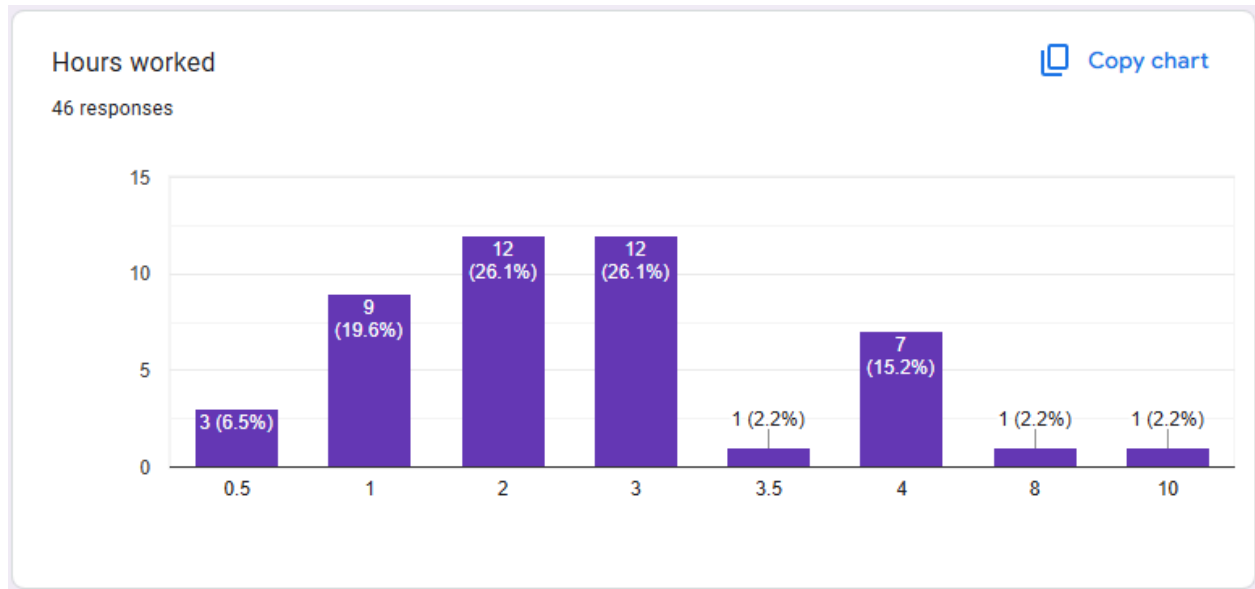
- Date: 14th February 2025
- Time: 11:00 AM – 2:00 PM
- Venue: Library Room STH503C
- Key Points:
  - **Sprint Retrospect:** Reviewed Sprint 2 progress and identified areas for improvement.
  - **Jira Update:** Updated task statuses for Sprint 3.
  - **Review of Outstanding Items:** Listed items to address (e.g., wishlist home page, backend tasks).
  - **Kick off Sprint 3:** Assigned new tasks, discussed upcoming feature additions for Sprint 3.
- **Next Steps:** Continue implementation, finalize API specs, and start the conversion of progress report from PowerPoint to document,\

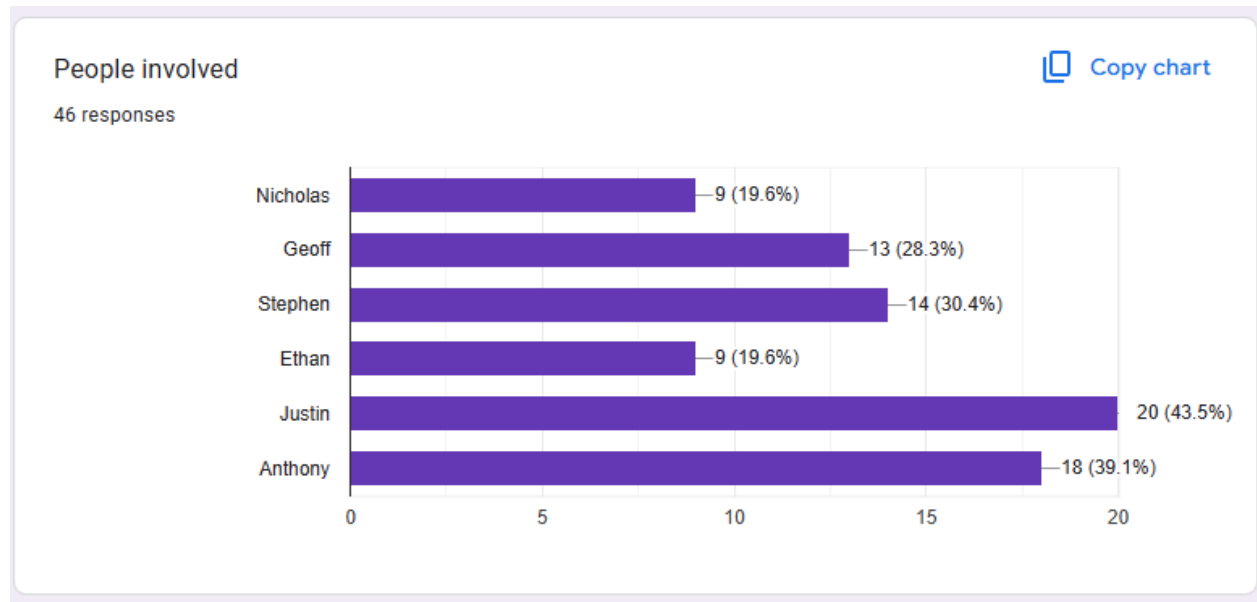
# Team Contribution

## GitHub Contributions



## Work Reporting





This spreadsheet shows the work each individual has reported, and acts as a nice breakdown of contributions – [Contribution Breakdown](#)

## Jira Work Assignment

### Sprint 1

#### Incomplete issues

Key	Summary	Issue type
COSC-13	Complete placeholder of event frontend	Task
COSC-43	Complete placeholder of list of events	Task

Status	Assignee
IN PROGRESS	SS
DONE	SS

#### Completed issues

Key	Summary	Issue type
COSC-6	Design UI for all pages	Task
COSC-7	Finish database UML	Task
COSC-8	Setup PostgreSQL database	Task
COSC-11	Complete placeholder of wishlist frontend	Task
COSC-12	Complete placeholder of sign up and sign in frontend	Task
COSC-41	Complete placeholder of list of wishlist	Task
COSC-42	Create a Landing Page	Task
COSC-44	Complete placeholder profile page	Task

Status	Assignee
DONE	
DONE	NP
DONE	NP
DONE	GJ
DONE	AM
DONE	SS
DONE	JB
DONE	AM

## Sprint 2

### Incomplete issues

Key ▾	Summary ▾	Issue type ▾	Status ▾	Assignee ▾
COSC-14	Design API specification	✓ Task	IN PROGRESS	NP
COSC-15	User / I want to create and modify accounts	📖 Story	IN PROGRESS	AM
COSC-34	Implement frontend for home page	✓ Task	TO DO	EB
COSC-13	Complete placeholder of event frontend	✓ Task	IN PROGRESS	SS
COSC-45	CSS/Styling of Pages	✓ Task	TO DO	JB
COSC-47	Express API Testing Framework	✓ Task	TO DO	JB

### Completed issues

Key ▾	Summary ▾	Issue type ▾	Status ▾	Assignee ▾
COSC-16	Setup production environment	✓ Task	DONE	NP
COSC-33	Implement frontend for sign in pages	✓ Task	DONE	AM
COSC-43	Complete placeholder of list of events	✓ Task	DONE	SS
COSC-46	Authentication API Endpoints	✓ Task	DONE	NP
COSC-48	Research Pop Up Library	✓ Task	DONE	AM
COSC-49	Dummy Items for Database	✓ Task	DONE	NP
COSC-50	Wishlist API Endpoints	✓ Task	DONE	NP
COSC-51	Event API endpoints	✓ Task	DONE	NP

# Sprint 3

## Incomplete issues

Key	Summary	Issue type	Status	Assign
COSC-13	Complete placeholder of event frontend	Task	IN PROGRESS	SS
COSC-34	Implement frontend for home page	Task	TO DO	EB
COSC-45	CSS/Styling of Pages	Task	TO DO	JB
COSC-15	User / I want to create and modify accounts	Story	IN PROGRESS	AM
COSC-14	Design API specification	Task	IN PROGRESS	NP
COSC-47	Express API Testing Framework	Task	TO DO	JB
COSC-17	User / I want to create wishlists	Story	TO DO	GJ
COSC-18	User / I want to delete wishlists	Story	TO DO	GJ
COSC-19	Party Host / I want to set deadlines on wishlists	Story	TO DO	NP
COSC-20	Event Planner / I want to duplicate wishlists	Story	TO DO	NP
COSC-21	Wishlist Contributor / I want a blind feature	Story	TO DO	GJ
COSC-25	Wishlist Contributor / I want to be able to sort a list by price, priority and contribution ...	Story	TO DO	SS
COSC-34	Implement frontend for home page	Task	TO DO	EB
COSC-35	Brainstorm testing framework	Task	TO DO	JB
COSC-45	CSS/Styling of Pages	Task	TO DO	JB
COSC-47	Express API Testing Framework	Task	TO DO	JB
COSC-52	Research OAuth / Firebase	Task	TO DO	AM
COSC-53	List of Prefix words for profile interests/ and implement to frontend	Task	TO DO	AM
COSC-54	Design Favicon Logo	Task	TO DO	AM
COSC-55	Navigation Bar for all pages	Task	TO DO	JB

# Appendices

Meeting Minutes -

[https://drive.google.com/drive/folders/1nsdbSNB-GpEHCatmVDX\\_s\\_xBf3spju8h?usp=drive\\_link](https://drive.google.com/drive/folders/1nsdbSNB-GpEHCatmVDX_s_xBf3spju8h?usp=drive_link)

Jira Link - <https://nicholasparise.atlassian.net/jira/software/projects/COSC/summary>

API Specification Document -

[https://docs.google.com/document/d/1SM71JXlzTIFZVJ1sR-axeLfNjZRz7TaqCHUoisUTTTc/edit?usp=drive\\_link](https://docs.google.com/document/d/1SM71JXlzTIFZVJ1sR-axeLfNjZRz7TaqCHUoisUTTTc/edit?usp=drive_link)

GitHub - <https://github.com/Nicholas-Parise/4P02-course-project/>

Work Contributions -

[https://docs.google.com/spreadsheets/d/1eE5l75YznTklsjcjskV1zBQ9aID18V7N\\_9SvYZ6pKC4/edit?usp=drive\\_link](https://docs.google.com/spreadsheets/d/1eE5l75YznTklsjcjskV1zBQ9aID18V7N_9SvYZ6pKC4/edit?usp=drive_link)