TAFE’s Alumni App

Action Plan  
**Cluster - Innovation Project Part I - ICT50220   
(All pathways)**

TEAM PROJECT ALUMNI 2024

Table of Contents

[Version control 3](#_Toc183679122)

[GitHub repository 3](#_Toc183679123)

[File versions 3](#_Toc183679124)

[Executive summary 4](#_Toc183679125)

[Stakeholders (including contact/communication method) 5](#_Toc183679126)

[Communication tools 6](#_Toc183679127)

[Scope 7](#_Toc183679128)

[Key Objectives 7](#_Toc183679129)

[Project deliverable 7](#_Toc183679130)

[Project Boundaries 7](#_Toc183679131)

[Limitation 7](#_Toc183679132)

[Task outline 8](#_Toc183679133)

[Project Setup and Planning 8](#_Toc183679134)

[Setup version control repository 8](#_Toc183679135)

[Define project scope and objectives 8](#_Toc183679136)

[Create Project timeline and milestones 8](#_Toc183679137)

[Identify and assign team roles and responsibilities 9](#_Toc183679138)

[Frontend Development 10](#_Toc183679139)

[Design user interface (UI) and user experience (UX) 10](#_Toc183679140)

[Design Example 11](#_Toc183679141)

[Develop responsive website layout 12](#_Toc183679142)

[Create components and modules for the website 12](#_Toc183679143)

[Implement navigation and search functionality 12](#_Toc183679144)

[Backend Development 13](#_Toc183679145)

[Set up backend server 13](#_Toc183679146)

[Develop APIs for data management and interaction 13](#_Toc183679147)

[Implement authentication and authorization mechanisms 13](#_Toc183679148)

[Integrate with existing systems (if applicable) 14](#_Toc183679149)

[Content management System (CMS) Development 15](#_Toc183679150)

[Build a CMS for managing website content 15](#_Toc183679151)

[Implement user roles and permissions 15](#_Toc183679152)

[Integrate CMS with frontend and backend components 15](#_Toc183679153)

[Testing and Quality Assurance 16](#_Toc183679154)

[Conduct unit, integration, and system testing 16](#_Toc183679155)

[Perform usability testing with target users 16](#_Toc183679156)

[Address identified bugs and issues 16](#_Toc183679157)

[Deployment and Launch 17](#_Toc183679158)

[Deploy website to production environment 17](#_Toc183679159)

[Conduct final testing and quality assurance 17](#_Toc183679160)

[Launch website and announce to alumni community 17](#_Toc183679161)

[Deliverables 18](#_Toc183679162)

[Hardware 18](#_Toc183679163)

[Software 20](#_Toc183679164)

[Documentation 21](#_Toc183679165)

[Training Materials 21](#_Toc183679166)

[Website (Mobile first principle) 22](#_Toc183679167)

[Mobile app (Cross-platform) 22](#_Toc183679168)

[Software System 22](#_Toc183679169)

[Resources 23](#_Toc183679170)

[Human Resources 23](#_Toc183679171)

[Project manager 23](#_Toc183679172)

[Frontend developers & UI/UX designers 23](#_Toc183679173)

[Backend developers 23](#_Toc183679174)

[Content creator and testers including performance 23](#_Toc183679175)

[Budget 24](#_Toc183679176)

[Development costs 24](#_Toc183679177)

[Hosting costs 24](#_Toc183679178)

[Marketing 24](#_Toc183679179)

[Project Gantt Chart via Trello with TeamGantt 25](#_Toc183679180)

[Timelines with Critical Path 27](#_Toc183679181)

[High Priority Tasks – 3rd Feb to 30th April 27](#_Toc183679182)

[Medium Priority Tasks (ongoing → 19th May) 30](#_Toc183679183)

[Low Priority Tasks – Anytime 30](#_Toc183679184)

[Critical Path Summary 30](#_Toc183679185)

[Communication Milestones 31](#_Toc183679186)

[Assignment of tasks, resources and timelines into a chart or diagram showing critical path. 32](#_Toc183679187)

[Considerations of contingency/risk management. 33](#_Toc183679188)

[Identifying Potential Risks: 33](#_Toc183679189)

[Technical Risks 33](#_Toc183679190)

[Resource Risks 33](#_Toc183679191)

[External Risks 33](#_Toc183679192)

[Developing Mitigation strategies: 33](#_Toc183679193)

[Technical Risks: 33](#_Toc183679194)

[Resource Risks: 35](#_Toc183679195)

[External Risks: 35](#_Toc183679196)

[Risk management tools and techniques: 36](#_Toc183679197)

[Risk assessment 36](#_Toc183679198)

[Risk mitigation 36](#_Toc183679199)

[Risk monitoring and control 36](#_Toc183679200)

[Contingency planning 37](#_Toc183679201)

[Contingency/Risk management examples. 38](#_Toc183679202)

[Technical Risk 38](#_Toc183679203)

[Resource Risks 39](#_Toc183679204)

[External Risk 40](#_Toc183679205)

[Communication agreements for client, project team and organisation. 41](#_Toc183679206)

[Communication agreements 41](#_Toc183679207)

[Client Communication 41](#_Toc183679208)

[Team Communication 41](#_Toc183679209)

[Organisation Communication 41](#_Toc183679210)

[Sample Meeting Agender 42](#_Toc183679211)

[Sign-off for project 43](#_Toc183679212)

# Version control

## GitHub repository

https://github.com/Lucas-lufa/Innovation

## File versions

|  |  |  |
| --- | --- | --- |
| **Version** | **Date** | **Changes** |
|  |  |  |
| **v 1.5** | 27th November 2024 | Removed alternatives and stick to one direction  Updated Critical Path, timeline and task and ore specific  Adjusted Task on Trello and generate Gantt Chart via TeamGantt |
| **v 1.4** | 24th October 2024 | Added Scope section |
| **v 1. 3** | 12th October 2024 | Create risk management and deadline information |
| **v 1. 2** | 11th October 2024 | Create resources timeline and considerations information |
| **v 1.1** | 10th October 2024 | Create stakeholders, task outline and delivery’s information |
| **v 1.0** | 09th October 2024 | Layout setting |
| **v 0.0** | 08th October 2024 | Created files and discussed each team member’s solutions. |
|  |  |  |

# Executive summary

This project aims to create a dedicated platform to enhance alumni engagement, networking, and resource access for North Metropolitan TAFE WA. The website and mobile app will provide features such as career opportunity by Alumni, online communities between Alumni, current Students and staffs (include lecturer), event listings, group formation, and mentorship opportunities.

Current challenges faced by alumni include the lack of an active alumni website and limited engagement on existing platforms like LinkedIn. These challengers highlight the need for a tailored platform to address the specific needs of north Metropolitan TAFE WA’s Alumni community.

The proposed solution is to develop a dedicated alumni website and mobile app that provides a central hub for alumni to connect, network, and access resources. The platform will offer the following features:

* Career: Voluntary Job listing and career advice by Alumni
* Online communities: Discussion forums, interest groups, alumni directories if agree
* Event listings: Upcoming events, registration and networking opportunities
* Group formation: Opportunities for alumni to create and join groups based on interest or shared experiences. Also, students can make team for collaboration or group studies with other classmates or other classes
* Mentorship opportunities: Connect alumni with mentors for career guidance and support.

The development and launch of the new alumni website and app are anticipated to take approximately 4 months.

A diagram of a group of people

Description automatically generated

# Stakeholders (including contact/communication method)

|  |  |  |
| --- | --- | --- |
| **Name** | **Role** | **Contact** |
| **Arron Clifford** | Clients &  Project Manager | [aaron.clifford@nmtafe.wa.edu.au](mailto:aaron.clifford@nmtafe.wa.edu.au) |
| **Lucas Fadersen** | Student & Full stack developer | [v092953@tafe.wa.edu.au](mailto:v092953@tafe.wa.edu.au) |
| **Liam Hubbard** | Team Leader,  Student & Full stack developer | [20136106@tafe.wa.edu.au](mailto:20136106@tafe.wa.edu.au) |
| **Sae Inoue** | Student & Full stack developer | [20103543@tafe.wa.edu.au](mailto:20103543@tafe.wa.edu.au) |
| **Yunjeong Gu** | Student & Full stack developer | [20114937@tafe.wa.edu.au](mailto:20114937@tafe.wa.edu.au) |
| **Sukhwan Ko** | Student & Full stack developer | [20111176@tafe.wa.edu.au](mailto:20111176@tafe.wa.edu.au) |
| **additional Stackholders** | | |
| Consider using evolve consultants for specific tasks. | | |

## Communication tools

* Email
* Teams: Our alumni team is based at TAFE, and our team accounts are integrated with Microsoft accounts (school account). This allows for easy communication with other client (lecturer) through Teams. Teams also integrates with calendars, meeting scheduling, and video calling, also it a convenient and centralised communication channel.
* Trello: Our alumni team consists of 5 people, so simpler features will be required. Therefore, we choose Trello as user-friendly project management platform. While other platforms offer more advanced features, we won’t be using them in this project.

|  |  |
| --- | --- |
|  | **Trello** |
| **Pros** | Simple and intuitive interface, highly customisable, great for small teams and agile methodologies. |
| **Cons** | Can be complex to set up and learn.  May not be suitable for large, complex projects. |

# Scope

This project aims to develop a dedicated platform to enhance alumni engagement, networking, and resource access for North Metropolitan TAFE WA. The platform will serve as a central hub for alumni to connect, share experiences, and access relevant resources.

## Key Objectives

* Create a user-friendly website and mobile app that is accessible to alumni from diverse backgrounds.
* Provide a range of features to support alumni engagement.
* Foster a sense of community and belonging among alumni.
* Enhance alumni’s connection to North Metropolitan TAFE WA and its current students.

## Project deliverable

* A fully functional alumni website with responsive design and intuitive navigation.
* A mobile app compatible with iOS and Android devices, offering similar features and functionality as the website.
* A comprehensive content management system for managing website and app content.
* Integration with existing systems such as the TAFE’s student database and event calendar (If applicable)
* User training materials to guide alumni on how to effectively use the platform.

## Project Boundaries

* The project will focus on developing a platform specifically for North Metropolitan TAFE WA alumni.
* While the platform will provide opportunities for networking and collaboration, it will not serve as a job board or employment agency.
* The platform will not offer financial or legal advice to alumni.

## Limitation

* The project team has the necessary skills and expertise to complete the project.
* The project will be completed within the allocated budget and timeline.
* The project will have access to the required resources, including data and technical infrastructure.

# Task outline

## Project Setup and Planning

### Setup version control repository

1. Create a new Git repository on a platform like GitHub
2. Add Project files to the repository
3. Configure initial commit message
4. Setup remote repository for collaboration.

### Define project scope and objectives

1. Conduct stakeholder interviews and workshops to gather requirements.
2. Define the overall goals and objectives of the alumni website.
3. Identify key features and functionalities to be included.
4. Establish project boundaries and limitations.

### Create Project timeline and milestones

1. Conduct a framework evaluation to compare the strengths and weaknesses of different frontend and backend frameworks (e.g. React/React Native vs Flutter)
2. Consider team expertise and project requirements when making the final decision.
3. Document the chosen frameworks and their rationale.
4. Develop a design strategy using Figma
   1. Create user personas and empathy maps to understand the target audience.
   2. Conduct user research to identify pain points and needs
   3. Develop wireframes and prototypes to visualize the website’s design.
   4. Iterate on the design based on feedback and testing.
5. Develop a detailed project timeline, including key milestones and deadlines.
6. Assign tasks to team members and allocate resources.
7. Identify potential dependencies between tasks.
8. Create a project management plan to track progress and manage risks.
9. Conduct a risk assessment to identify potential challenges and develop mitigation strategies.

### Identify and assign team roles and responsibilities

1. Conduct a team assessment to identify the skills and expertise of each team member.
2. Define roles and responsibilities based on team members’ strengths and project requirements.
3. Selected a suitable team management tool which is Trello to facilitate collaboration, task assignment, and progress tracking.
4. Configure the chosen team management tool to match the project’s specific needs and workflows.

## Frontend Development

### Design user interface (UI) and user experience (UX)

1. Create user personas: Develop detailed profiles of the target users to understand their needs, goals, and behaviours.
2. Conduct user research: Gather insights into user preferences, pain points, and expectations through interviews, surveys, or usability testing.
3. Develop wireframes: Create low-fidelity sketches of the website’s layout and structure.
4. Design prototypes: Create interactive prototypes to test the usability and functionality of the design.
5. Iterate on the design: Continuously refine the design based on feedback and testing.

### Design Example

|  |  |  |
| --- | --- | --- |
| Screens screenshot of a cell phone  Description automatically generated | Screens screenshot of a phone  Description automatically generated | Screenshot of a screenshot of a profile case  Description automatically generated |

|  |  |  |
| --- | --- | --- |
|  |  |  |

### Develop responsive website layout

1. Create a responsive grid / flex system: Design a grid / flex system that adapts to different screen sizes and devices.
2. Develop responsive components: Ensure that all website components (e.g. hears, footers, navigation) are responsive and adjust to different screen sizes.

### Create components and modules for the website

1. Identify reusable components: Break down the website into reusable components (e.g. buttons, forms, cards).
2. Develop component libraries: Create a library of reusable components that can be used throughout the website.
3. Implement modular architecture: Organise the codebase into modules to improve maintainability and scalability.

### Implement navigation and search functionality

1. Design a clear and intuitive navigation structure.
2. Implement a search feature
3. Optimise search results for relevance and accuracy.
4. Provide search suggestions to help users. Find what they’re looking for.

## Backend Development

### Set up backend server

1. Set up development environment: Install necessary tools and dependencies such as Laravel.
2. Configure server settings: Configure the server for optimal performance and security.
3. Database setup: Create and configure the database (e.g. MySQL, MariaDB and so on) for storing website data.

### Develop APIs for data management and interaction

1. Define API endpoints: Determine the endpoints needed for data retrieval, creation, medication, and deletion.
   1. Required Sensitive data from users.
      1. Tafe email: This will be used to verify the user’s identity as a current TAFE student.
      2. Alternative email: A personal email address that can be used for access after graduation.
      3. Nickname: As preferred display name.
   2. Additional data from users
      1. Course information
2. Implement RESTful API design principle: Adhere to RESTful standards for API structure and communication.
3. Develop API routes and controllers: Create routes and controllers to handle API requests and responses.
4. Implement data validation and error handling: Validate input data and provide appropriate error messages.

### Implement authentication and authorization mechanisms

1. Choose an authentication method: Select a suitable method like OAuth2, JWT, or basic authentication like email and password.
2. Implement user registration and login: Allow users to create accounts and log in to the website.
3. Manage user roles and permissions: Assign different roles and permissions to users based on their access levels.
4. Implement password hashing and salting: Store user passwords securely using hashing algorithms.

### Integrate with existing systems (if applicable)

1. Identify integration points: Determine which systems need to be integrated with the alumni website.
2. Develop integration APIs: Create APIs to exchange data between the website and other systems.

## Content management System (CMS) Development

### Build a CMS for managing website content

1. Utilise Laravel’s CMS features: Leverage Laravel’s built-in CMS capabilities, such as routing, controllers and Eloquent ORM, to streamline development.
2. Create content types: Define the different types of content that will be managed on the website
3. Design content templates: Create templates for different content types using preferred templating engine e.g., Flutter
4. Implement content editing features: Allow users to create, edit and delete content using Laravel’s forms and validation.
5. Integrate with frontend and backend: Connect the CMS with the frontend Flutter and backend components using Laravel’s API capabilities.

### Implement user roles and permissions

1. Define user roles: Determine the different roles and permissions required for managing website content.
2. Create user profiles: Allow users to create account and assign roles using Laravel’s authentication features.
3. Implement role-based access control (RBAC): Restrict user access to specific content and functionalities based on their roles using Laravel’s authorisation mechanisms.
4. Grant and revoke permissions: Manage user permissions as needed.

### Integrate CMS with frontend and backend components

1. Develop API endpoints: Create APIs for the CMS to interact with the frontend Flutter and backend components using Laravel’s routing and controllers.
2. Implement data synchronisation: Ensure that data is consistently updated between the CMS and other components using Laravel’s Eloquent ORM.
3. Handle content delivery: Configure the CMS to deliver content to the frontend when requested.
4. Test and debug integrations: Verity that the CMS integrates seamlessly with the rest of the website.

## Testing and Quality Assurance

### Conduct unit, integration, and system testing

1. Develop test cases: Create detailed test cases to cover various scenarios and use cases.
2. Conduct unit testing: Test individual components or modules in isolation to ensure they function as expected.
3. Conduct integration testing: Test how different components interact with each other to ensure seamless integration.
4. Conduct system testing: Test the entire system to verify that it meets the specified requirements and functions as intended.
5. Conduct white-box testing: Examine the internal structure and logic of the code to identify potential defects.
6. Conduct black-box testing: Test the application’s external behaviour without knowledge of the internal code.

### Perform usability testing with target users

1. Recruit test participants: Select a representative sample of target users to participate in usability testing.
2. Design usability test tasks: Create tasks that simulate real-world scenarios and user goals.
3. Observe user behaviour: observe users as they interact with the website and identify any difficulties or pain points.
4. Gather feedback: Collect feedback from users through interviews or surveys.

### Address identified bugs and issues

1. Track and prioritise bugs: use a bug tracking tool to record and prioritise identified issues.
2. Assign bugs to developers”: Assign bugs to the appropriate team members for resolution.
3. Fix Bugs and retest: Address identified bugs and retest the affected areas.
4. Document bug fixes: Record information about bug fixes for future reference.

## Deployment and Launch

### Deploy website to production environment

1. Prepare production environment: Set up a production server with the necessary hardware and software.
2. Configure server settings: configure the server for optimal performance and security.
3. Deploy website files: Transfer the website files to the production server.
4. Update DNS settings: Point the domain name to the production server.
5. Test deployment: Verify that the website is accessible and functioning correctly in the production environment.

### Conduct final testing and quality assurance

1. Perform smoke testing: Verify that the website’s critical functionalities are working as expected.
2. Conduct regression testing: Ensure that recent changes have not introduced new bugs.
3. Gather feedback from stakeholders: Obtain final feedback from stakeholders before launch.

### Launch website and announce to alumni community

1. Create a launch plan: Develop a plan for the website launch, including timing, communication channels, and promotional activities.
2. Update website content: Ensure that all website content is up-to-date and accurate.
3. Promote the launch: Utilise various communication channels (e.g. email, social media, website announcements) to inform the alumni community about the launch.
4. Monitor website traffic and user feedback: Track website usage and gather feedback from alumni to identify areas for improvement.

# Deliverables

### Hardware

Cloud-based server: Azure App Service with Docker Containers.

###### Scalability:

Cloud-based servers can be easily scaled up or down to meet changing demands. This means that Alumni App can quickly add or remove resources as needed, without having to invest in new hardware.

###### Containerisation:

The website and its components will be packaged into Docker containers for efficient deployment and management.

###### Cost-effectiveness:

Cloud-based servers can be more cost-effective than traditional on-premises servers. This Alumni app only pay for the resources it uses, and there are no upfront costs for hardware or infrastructure.

###### Flexibility:

Cloud-based servers offer a high degree of flexibility. Also customise its server configuration to meet the app’s specific needs.

###### Reliability:

Cloud-based servers are typically more reliable than traditional on-premises server. Cloud providers have invested heavily in redundant infrastructure and disaster recovery plans to ensure that your data is safe and accessible.

###### Security:

Cloud-based servers are often more secure than traditional on-premises servers. Cloud providers have dedicated security teams that are constantly monitoring for threats and vulnerabilities.

###### Accessibility:

Cloud-based servers can be accessed from anywhere in the world with an internet connection. This makes it easy for collaborate with remote teams and customers.

###### Reduced maintenance:

Cloud providers manage the underlying infrastructure. Which mean cloud providers are maintaining hardware, software updates or security patches.

###### Disaster recovery:

Cloud-based servers can be used to implement disaster recovery plans. In the event of a disaster, this can quickly recover its data and service from the cloud.

###### Sustainability:

Cloud-based servers can be more sustainable than traditional on-premises servers. Cloud providers often use renewable energy sources to power their data centres.

### Software

1. Frontend framework: **“Flutter (Dart)”**
2. CSS framework: **Tailwind CSS**
3. Design tool: **Figma**
4. Backend framework: **Laravel** (PHP)
5. CMS: Custom CMS created by Laravel
6. Database: **SQLite** for development purpose then migrate to “MySQL”
7. Version control system: **Git** with **GitHub**
8. Team management tool: **Trello**
9. Development and testing tools
   1. IDE
   2. Debugging tools
   3. Testing frameworks
   4. Functional test (Black & white box test)
10. Containerisation: **Docker**
11. Webserver: **Nginx**
12. Deployment tools: Azure App service
13. Deployment option 2: Screencraft

### Documentation

###### Project plan:

Use this action plan to use

###### Requirement documents

1. Design documents: UI/UX designs, wireframes, and prototypes
2. Technical documentation: Detailed explanations of system architecture, code structure and implementation.
3. User manual: Instructions for website administrators and users.
4. Training materials:
   1. Tutorials
   2. Guides
   3. Presentations for training purposes.
5. Policy documents:   
    **This will be used from TAFE’s documentation** 
   1. Code of ethics: A set of principles that guide the behaviour
   2. Code of conduct: A set of rules that govern the behaviour
   3. Privacy policy: A statement outlining how the organisation collects, uses, and protects user data.
   4. Terms of service: A legal agreement between the website and its users.
6. Final report: Summary of project outcomes, lessons learned, and recommendations.

### Training Materials

###### Website administration training

Tutorials and guides for managing website content and settings. e.g.:

1. Document
2. Video
3. Tooltip

###### User training

Instructions for alumni on how to use the website’s features and functionalities.

### Website (Mobile first principle)

1. Fully functional alumni website with all planned features and functionalities.
2. Responsive design compatible with various devices and screen sizes.
3. Intuitive navigation and user-friendly interface.
4. Search functionality for easy content discovery.
5. Integration with existing system (if applicable)

### Mobile app (Cross-platform)

1. Support native mobiles apps by cross-platform framework for iOS and Android devices.
2. Similar or same features and functionalities as the website.
3. Optimised for mobile devices with responsive design
4. Offline capabilities (if applicable)

### Software System

1. Custom-built software system for managing alumni data and interactions
2. Scalable and maintainable architecture.
3. Deployed as Docker containers on Azure App Service.
4. Uses Nginx as the web server

###### Why Nginx not Apache server?

1. Performance: Nginx is known for high performance and scalability especially when handling a large number of concurrent connections
2. Lightweight: Nginx is a lightweight webserver with a smaller memory footprint compared to Apache. Its saving resource cost.

# Resources

## Human Resources

### Project manager

* Arron Clifford

Oversees the project’s progress, ensures tasks are completed on time, and manages communication between team members and stakeholders.

### Frontend developers & UI/UX designers

* ALL TEAM MEMBERS APPLIED

Collaborate to develop the user interface and user experience of the App, including layout, design, functionality, and interaction. Create a user-friendly interface that is easy to navigate and sue. Implement responsive design to ensure the app is accessible and usable across different devices and screen sizes. Adhere to accessibility standards (WCAG) to make the website inclusive for user with disabilities.

### Backend developers

* ALL TEAM MEMBERS APPLIED

Backend developer will be responsible for developing the server-side logic and functionality of the website, including database interactions, APIs, and integrations with other systems. They will also need to configure server settings for Docker containers and Azure App Service, such as network settings, resource allocation, and security configurations. Azure App Service provides a managed environment for running web applications, allowing for each deployment and management of Docker containers.

### Content creator and testers including performance

* ALL TEAM MEMBERS APPLIED

Develop and manage website content, such as posts, events, and alumni profiles. Ensure the website’s functionality and performance through rigorous testing including accessibility checks for WCAG compliance, keyboards navigation and screen reader compatibility.

## Budget

### Development costs

###### Human Resources

At this is a TAFE school project, we will not incur costs for human resources. The team will be composed of students working on the project as part of their studies.

###### Software

We will primarily use free versions, educational plans, and open-source programs, minimising software costs.

###### Hardware

Team members will utilise their own devices for development, eliminating hardware costs.

### Hosting costs

###### Azure App Service

This Alumni App can leverage the Azure for Students plan to utilise Azure App Service, which provides 10 web, mobile, or API apps with 1GB of storage and 1 hour per day for free. This allocation should be sufficient for our presentation purposes.

###### Domain registration

As this is a school project, we will avoid incurring additional costs for domain registration unless it is deemed necessary for future use. The application will initially use the public IP address provided by Azure App Service for presentation purposes.

###### Deploy Option 2 using TAFE Webhosting – ScreenCraft

Compatibility: Verify if TAFE’s webhosting platform supports the required programming languages such as PHP and database system

Resource Allocation: Determine the available resources (CPU, Memory, Storage) and if they meet the application’s requirements.

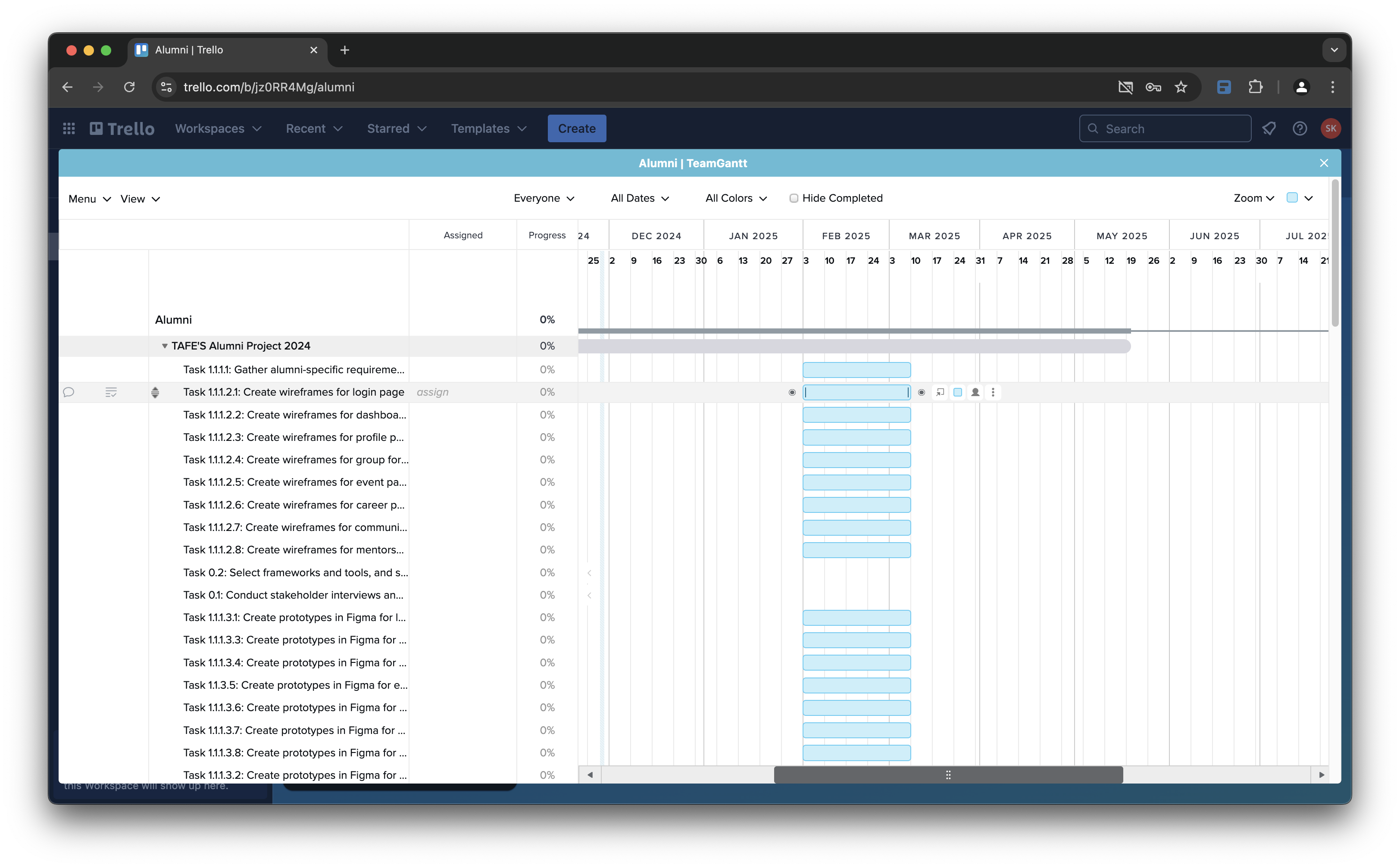
Deployment Method: Consider using Docker container for easier deployment and management if supported by TAFE’s platform.

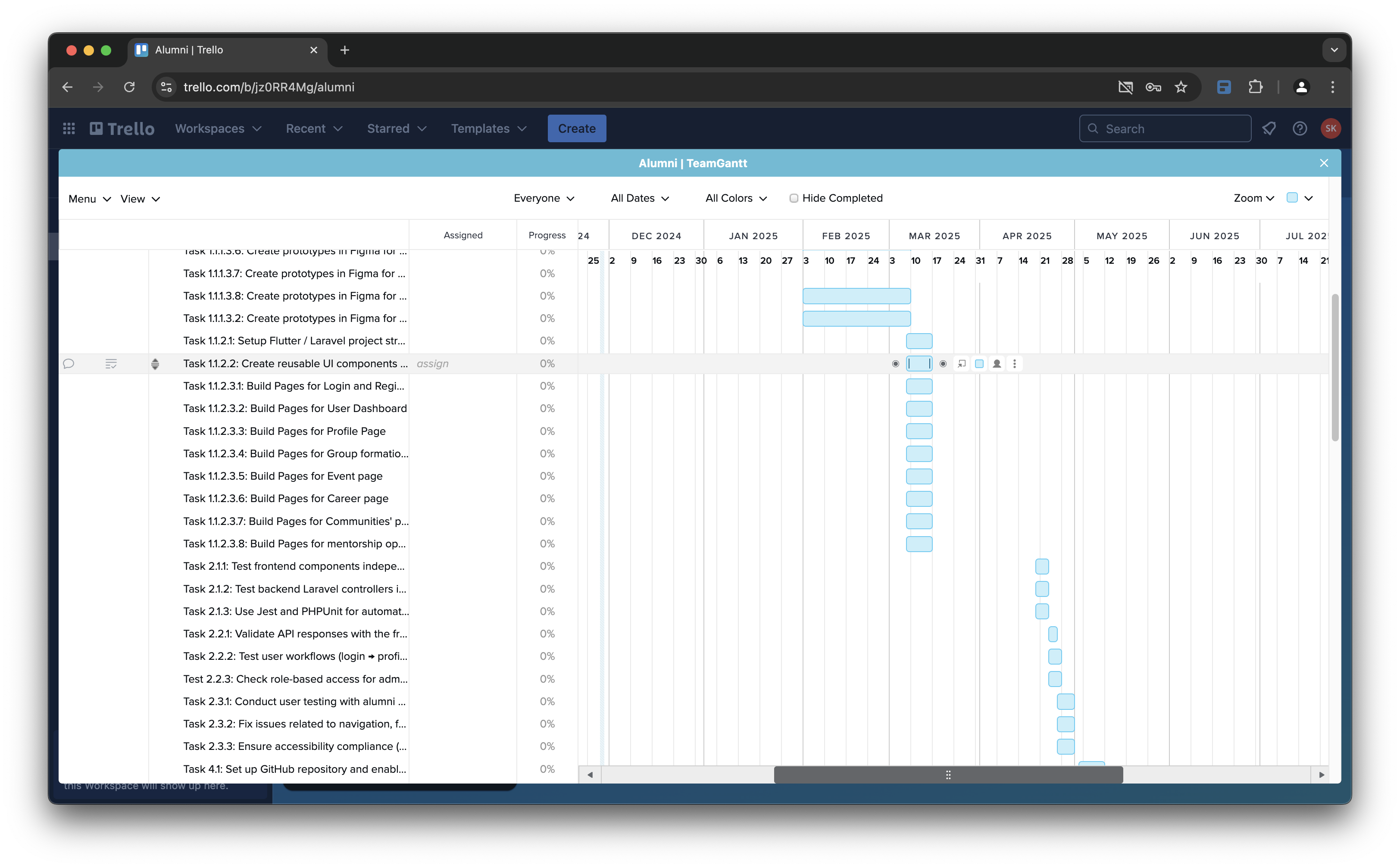
### Marketing

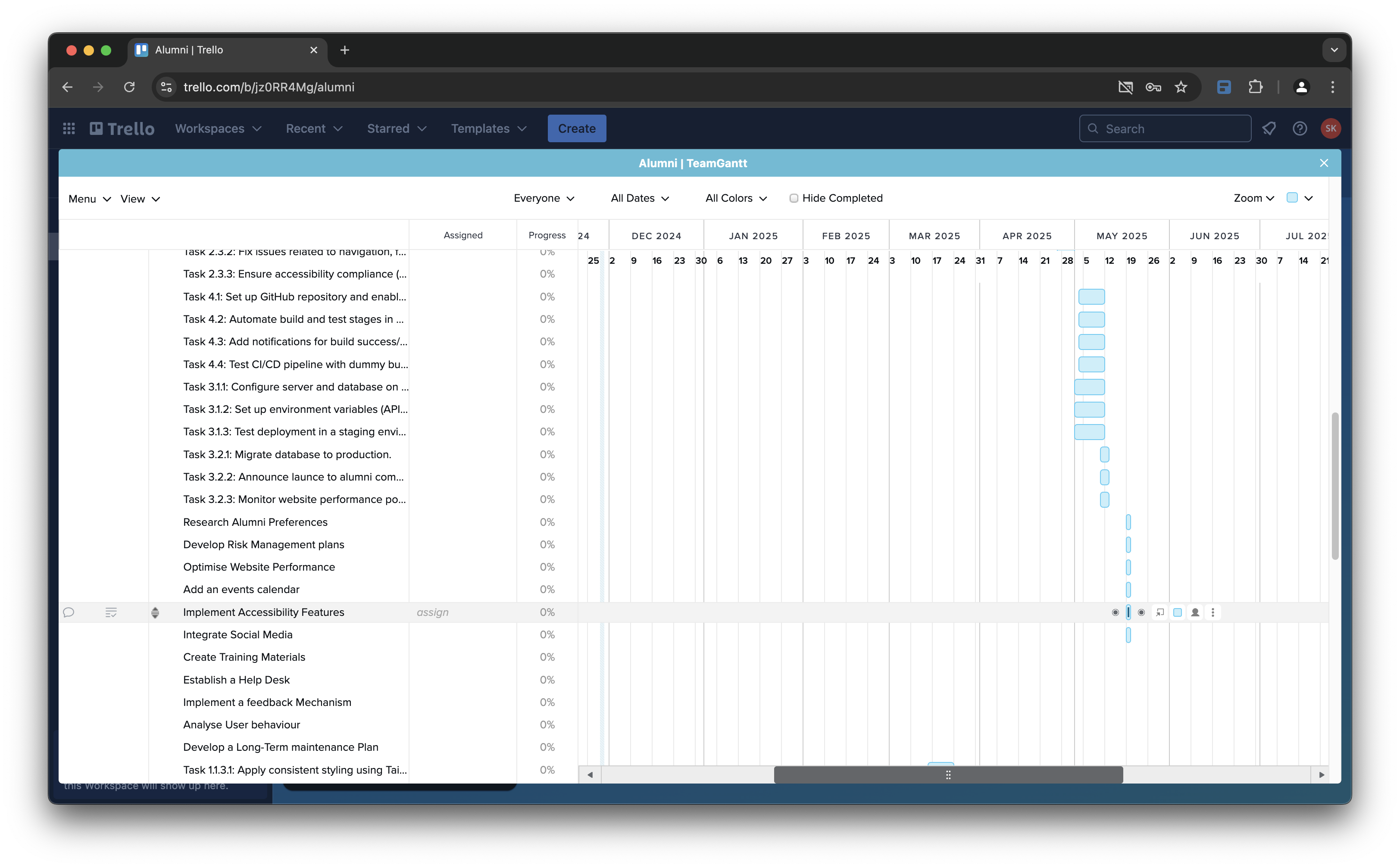
This is a TAFE school project with the potential to launch the app if it is deemed serviceable. If successful, this project could be beneficial for several reasons. One reason is the opportunity to collaborate with other class students, fostering teamwork and knowledge sharing. We could also consider collaborating with Cert IV in Marketing and Communication class students in Perth on-campus students who are interested in contributing to the project.

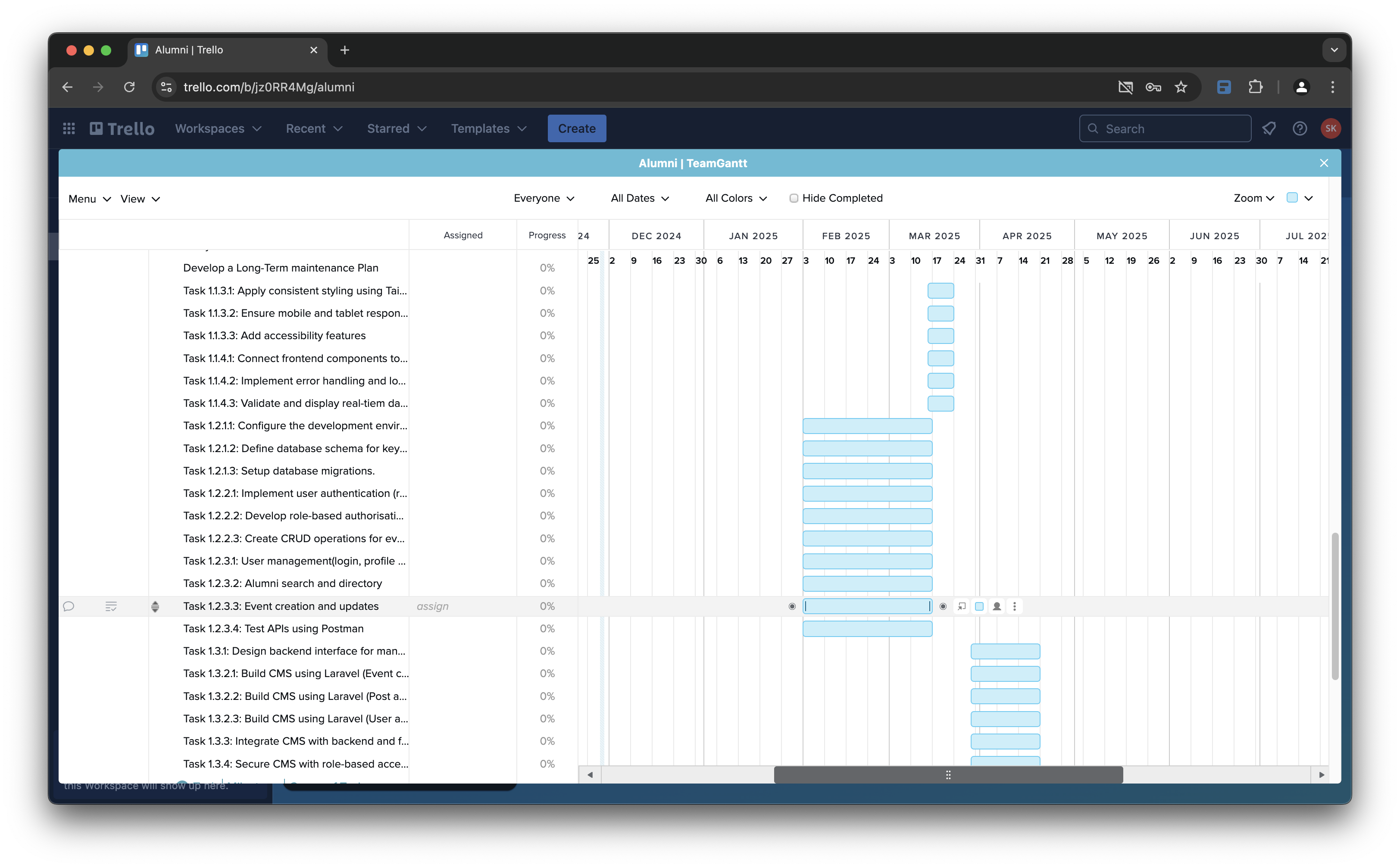
Alternatively, **TAFE’s Marketing team will handle this section, if applicable.**

# Project Gantt Chart via Trello with TeamGantt









# Timelines with Critical Path

## High Priority Tasks – 3rd Feb to 30th April

#### Phase 0: Project Setup and Planning (Completed)

* Conduct stakeholder interviews and define project scope and objectives.
* Select frameworks and tools, and set up the development environment

#### Phase 1: Frontend and Backend Development (3rd Feb → 18th April)

**Critical Path Milestones**Figma designs → Frontend Development → API integration → CMS development

→ Feedback from stakeholder and client

1. Frontend Development (3rd Feb → 21st March)

* Design and finalise Figma prototypes – Due: 7th March
  + Gather alumni-specific requirements for UI/UX
  + Create wireframes:
    - for login page
    - for dashboard page   
      (e.g. recently updated information, deleted information)
    - for profile page
    - for group formation page
    - for event page
    - for career page
    - for communities’ page
    - for Mentorship opportunities page
  + Create prototypes in Figma:
    - for login page
    - for dashboard page   
      (e.g. recently updated information, deleted information)
    - profile page
    - for group formation page
    - for event page
    - for career page
    - for communities’ page
    - for Mentorship opportunities page
* Develop core frontend components – Due: 14th March
  + Setup Flutter / Laravel project structure
  + Create reusable UI components (e.g., buttons, forms, navigation etc.)
  + Build Pages:
    - Login and Registration
    - User Dashboard
    - Profile Page
    - Group Formation Page
    - Event Page
    - Career Page
    - Communities’ Page
    - Mentorship Opportunities Page
* Styling and Responsiveness Due: 21st March
  + Apply consistent styling using Tailwind CSS
  + Ensure mobile and tablet responsiveness
  + Add accessibility features (e.g. Keyboard navigation)
* Integrate frontend with APIs – Due: 21st March
  + Connect frontend components to API endpoints.
  + Implement error handling and loading states.
  + Validate and display real0tiem data.

1. Backend Development (3rd Feb → 18th April)

* Set up Backend server and databases – Due: 14th March
  + Configure the development environment (Laravel, database setup).
  + Define database schema for key entitles (users, events, pots etc)
  + Set up database migrations.
* Implement authentication and authorisation – Due: 14th march
  + Implement user authentication (registration, login, logout).
  + Develop role-based authorisation (Superuser, Admin, Staff, Alumni and Guests)
  + Create CRUD operations for events and announcements
* Develop and test APIs – Due: 21st March
  + User management (login, profile update)
  + Alumni search and directory
  + Event creation and updates.
  + Test APIs using Postman

1. CMS Development (28th March → 18th April)

* Design backend interface for managing content (e.g. events, users)
* Build CMS using Laravel
  + Event creation/editing
  + Post and announcement management
  + User and role management
* Integrate CMS with backend and frontend
* Secure CMS with role-based access (e.g. admin only access)

#### Phase 2: Testing and Quality Assurance (19th April → 30th April)

**Critical Path Milestones**Unit testing → Integration testing→ Usability testing.

1. Unit Testing (19th → 22nd April)

* Test frontend components independently.
* Test backend Laravel controllers independently.
* Use Jest and PHPUnit for automated testing.

1. Integration Testing (23rd → 25th April)

* Validate API responses with the frontend.
* Test user workflows (login → profile → search directory).
* Check role-based access for admins and alumni

1. Usability Testing (26th → 30th April)

* Conduct user testing with alumni volunteers
* Fix issues related to navigation, from validation, and error feedback.
* Ensure accessibility compliance (WCAG standards).

#### Phase 3: Deployment and Launch (1st May → 12th May)

**Critical Path Milestones**Deployment prep → Website launch

1. Deployment Prep (1st → 10th May)

* Configure server and database on Azure App service.
* Set up environment variables (API keys, database credentials)
* Test deployment in a staging environment.

1. Launch Announcement (11th → 12th May)

* Migrate database to production.
* Announce launce to alumni community
* Monitor website performance post-launch

#### Phase 4: CI/CD Pipeline Setup (3rd May → 10th May)

**Critical Path Milestones**CI/CD pipeline Step → Test Pipeline

* Set up GitHub repository and enable branch protections
* Automate build and test stages in Azure Pipelines.
* Add notifications for build success/failure
* Test CI/CD pipeline with dummy builds before full integration.

## Medium Priority Tasks (ongoing → 19th May)

* **Research Alumni Preferences:** Conduct surveys or interviews to understand the needs and expectations of Alumni
* **Develop Risk Management plans**
* **Optimise Website Performance**
* **Add an events calendar**
* **Implement Accessibility Features**
* **Integrate Social Media**

## Low Priority Tasks – Anytime

* **Develop a Comprehensive user Manual**
* **Create Training Materials**
* **Establish a Help Desk**
* **Implement a feedback Mechanism**
* **Analyse User behaviour**
* **Develop a Long-Term maintenance Plan**

# Critical Path Summary

**Dependencies:**

* Finalising Figma designs → Frontend components
* Backend APIs → API integration
* Usability testing →Final deployment

# Communication Milestones

###### Regular meetings

Every Tuesday (Allocated Innovation class time)  
Agender: Checkup teams progress, identify hurdles (bug issues) and reallocate progress.

###### Fortnight project status updates

Every even week of Team meetings

Agender: Update progress tracks, problem-solving issues, provide feedback, and address client requests.

###### Client feedback sessions

Each end of phase

Agender: Confirmation before moving onto the next phase

###### Final presentation

End of project

Agender: Present the final app

# Considerations of contingency/risk management.

## Identifying Potential Risks:

### Technical Risks

* Framework compatibility issues
* Database performance bottlenecks
* Security vulnerabilities
* Integration challenges with existing systems.

### Resource Risks

* Team member availability and skill gaps
* Budget constraints
* Time constrains

### External Risks

* Changes in requirements
* Technological advancements
* Economic factors

## Developing Mitigation strategies:

### Technical Risks:

###### Framework compatibility issues

* Conduct thorough research and testing before selecting frameworks.
* Use well-established frameworks with a proven track record of compatibility.
  + Frontend Frameworks: **Flutter**
  + CSS Frameworks: **Tailwind CSS**
  + Backend Frameworks: **Laravel-PHP**
* Consider using a technology stack with known compatibility, such as:
  + **LEMP Stack (Linux, Nginx, MySQL, PHP)**

###### Database performance bottlenecks

* Optimise database queries and indexes
* Scale the database horizontally or vertically as needed
* Use caching mechanisms to reduce database load.

###### Security vulnerabilities

* Implement security best practices, such as input validation, output encoding, and regular security audits
* Use a web application firewall (WAF) to protect against common web attacks.
  + Azure Web Application Firewall (Students Account has 100$ Credit can be applied)
* Keep software and libraries up to date with the latest security patches.

###### Integration challenges with existing systems (if applicable)

* Develop clear integration plans and test thoroughly before deployment
* Use well-defined APIs and protocols for communication between systems.
* Consider using middleware or integration platforms to simplify the integration process.

### Resource Risks:

###### Team member availability and skill gaps

* Crosstrain team members to increase their skillset and flexibility.
* Develop contingency plans for team member absences, such as assigning backup tasks
* Provide opportunities for team members to learn new skills and stay updated on industry trends.

###### Budget constraints

* Prioritise tasks and allocate resources effectively to ensure that the most critical features are developed within the budget
* Explore cost-saving options, such as use open-source software or cloud-based services.
* Monitor the budget closely and adjust plans as needed.

###### Time constraints

* Develop a realistic project timeline and monitor progress closely.
* Be prepared to adjust the timeline if necessary, such as by prioritising tasks or reducing the scope of the project.

### External Risks:

###### Changes in requirements

* Maintain open communication with stakeholders and be flexible to adapt to changes in requirements.
* Incorporate a change management process to manage and control changes effectively.

###### Technological advancements

* Stay updated on industry trends and consider incorporating new technologies if beneficial.
* Conduct regular technology assessments to identify potential improvements.

## Risk management tools and techniques:

### Risk assessment

* Risk Identification: identify potential risks that could impact the project’s success.
* Risk analysis: Access the likelihood and impact of each identified risk.
* Risk prioritisation: Rank risks based on their potential impact and likelihood of occurrence.

###### Tools and techniques

* Risk matrix: A visual tool to access the likelihood and impact of risks.
* SWOT analysis: A tool to identify strengths, weaknesses, opportunities, and threats.
* Scenario planning: A technique to explore different future scenarios and their potential impacts.

### Risk mitigation

* Avoidance: Eliminate the risk entirely if possible
* Reduction: Reduce the likelihood or impact of the risk
* Acceptance: Accept the risk and develop contingency plans to deal with it if it occurs.

###### Tools and techniques

* Risk mitigation plan: A document outlining strategies to address identified risks.
* Contingency planning: Developing backup plans for potential challenges or setbacks.
* Sensitivity analysis: Assessing the impact of changes in key variables on the project.

### Risk monitoring and control

* Regular monitoring: Track the progress of risk mitigation strategies and identify any new risks that may emerge.
* Performance measurement: Measure the effectiveness of risk management efforts.
* Corrective action: Take corrective action if risks materialise or if mitigation strategies are not effective.

###### Tools and techniques

* Risk register: A document for tracking identified risks, their likelihood, impact and mitigation strategies.
* Key performance indicators (KPIs): Metrics to measure the effectiveness of risk management efforts.
* Regular reviews: Conduct periodic reviews to assess the project’s risk profile and adjust mitigation strategies as needed.

### Contingency planning

* Develop contingency plans: Create backup plans for potential challenges or setbacks.
* Identify alternative solutions: Explore alternative approaches or resources if the original plan fails.
* Allocate resources: Ensure that sufficient resources are available to implement contingency plans.

###### Tools and techniques

* Scenario planning: Explore different future scenarios and develop contingency plans for each.
* Emergency response plan: A detailed plan for responding to unexpected event.
* Backup and recovery procedures: Implement procedures for backing up data and recovering from system failures.

## Contingency/Risk management examples.

### Technical Risk

###### Example 1: Framework Compatibility Issues

A team of developers was working on a project using React and Laravel. They encountered compatibility issues when trying to integrate a third-party library for user authentication. The library was designed for a different frontend framework and did not work as expected with React.

**Mitigation strategy**

The team researched alternative libraries compatible with React and Laravel. They eventually found a suitable library and successfully integrated it into the project.

###### Example 2: Database Performance Bottlenecks

A popular e-commerce website experienced slow load times during peak traffic periods. After investigation, the team discovered that the database was becoming overwhelmed with large amount of data.

**Mitigation strategy**

The team optimised database queries, added missing indexes, and scaled the database horizontally to improve performance.

###### Example 3: Security Vulnerabilities

A healthcare website was discovered to be vulnerable to SQL injection attacks, allowing malicious users to access sensitive patient data.

**Mitigation strategy**

The team patched the vulnerable code, implemented input validation and output encoding, and conducted regular security audits to prevent future attacks.

### Resource Risks

###### Example 1: Team member availability and skill gaps

A key frontend developer goes on maternity leave during the critical stage of designing and developing the user interface.

**Mitigation strategy**

The team members will cross-train to take on the absent member’s responsibilities, ensuring a smooth transition and minimising disruption. Also, the team will re-prioritise tasks, focusing on the most critical areas to ensure that essential work continues. Additionally, the team will provide additional training to member who need to develop new skills, enabling them to effectively support the team’s goals and maintain a high level of performance.

###### Example 2: Budget constraints

The team needs to cut costs to stay within budget, potentially impacting the project’s quality or features.

**Mitigation strategy**

The team will explore cost-saving options such as using open-source software or cloud-based services, which can significantly reduce expenses without compromising quality. Additionally, the team will leverage students' educational plans to reduce costs, while also considering the possibility of using additional features for educational purposes, such as Azure's student plan, which offers free deployment with limitations but is sufficient for presentation purposes.

###### Example 3: Time constraints

The project is delayed due to unforeseen challenges with integrating a third-party API.

**Mitigation strategy**

The team will reassess priorities and focus on the most critical tasks, ensuring that our efforts are aligned with the project's objectives. Second, the team will allocate additional resources to the project if necessary, providing the necessary support to accelerate progress and meet deadlines. Finally, the team will consider extending the project timeline if feasible, allowing for a more realistic and achievable timeline that ensures the delivery of a high-quality product.

### External Risk

###### Example 1: Changes in requirements

A client requested new features to be added to the project after the initial requirements were defined.

**Mitigation strategy**

The team evaluated the impact of the changes on the project timeline and budget and developed a plan to incorporate the new features.

###### Example 2: Technological advancements

A new framework was released that offered significant performance improvements over the one currently being used.

**Mitigation strategy**

The team assessed the feasibility of migrating to the new framework and weighed the benefits against the costs.

# Communication agreements for client, project team and organisation.

## Communication agreements

### Client Communication

* Regular meetings: Fortnightly during allocated innovation class time.
* Communication channels: Email and Teams.
* Feedback sessions: End of each phase (Week 5, Week 11, Week 13 and Week 16)
* Client access: Provide a dedicate client access for project updates, documents and communication history. (Team’s channel, Trello and GitHub repository)

### Team Communication

* Weekly meeting: Brief weekly meetings to discuss progress, blockers, and next steps on innovation class time.
* Communication channels: Email, Trello, GitHub and Teams.
* Project management software: Use Trello to track tasks, assign responsibilities, and communicate progress.

### Organisation Communication

* Regular updates: Provide fortnightly during allocated innovation class time.
* Communication channels: Email, Teams, **Trello**(main) and **Discord**(main).
* Dedicated contact person: Available team member to handle communication with stakeholders. (Currently **Liam** will contact product manager & client.

### Sample Meeting Agender

Meeting Title: Regular project meeting

Date: 19-11-2024

Time: 10:00 am

Location: Classroom 306

Attendants:

* John Doe (Backend)
* Jane Doe (Frontend)
* Lee Doe (Researcher)
* …

Agender:

* Risk of Project
* Debugging issues
* General problems
* Discussion of Q&A

# Sign-off for project

Client Sign-off

Signature \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_             Date \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Supervisor Sign-off

Signature \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_             Date \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_