CFG Full Stack Development Group 1

MOODIE PROJECT

Rima Pipadwala | Alissa Monova | Lisa Osinowo | Elizabeth Simpson | Danielle Shuttleworth | Gabriella Snow | Osakpolor Orumwense

Moodie

Let your mood pick the movie

Introduction

Aims and Objectives

Moodie is a web-based application that helps users find the perfect movie by translating their mood, emotions, and vibe into personalisation film recommendations. It uses playful quizzes and a roulette-style generator to cut through choice fatigue and offer users movies that “just feel right” based on how their day is going. Moodie aims to help indecisive film fans stop endless scrolling so they can have more time to watch films.

Background

The inspiration for Moodie came shortly after our group formed and met to discuss which industries we all had a collective interest in. Through a preferential voting system, we decided to build a film app. With endless options available on streaming services nowadays, we quickly realised a common issue for film fans is simply picking something to watch. Scrolling through the options can often take longer than watching the film itself, so we decided to build a tool that is fun to use and helps users decide what film to watch, saving them time and energy.

Users initially land on a home page which has some information about Moodie and its creators, and the option to Start Moodie. Upon starting Moodie, the user is asked to choose one of two paths - the Moodie Quiz or the Random Movie Generator.

The Moodie quiz asks users to complete a quiz for a personalised recommendation. Each quiz question and combination of answers corresponds to parameters used to filter API requests from The Movie Database API (TMDB API). Upon completion of the quiz, the user is presented with a randomised recommendation based on their answers.

The Random Movie Generator is intended to provide a quicker, random recommendation. Users need only to select a genre, and spin the wheel for a random film recommendation. The genre selection returns the top 20 most popular films in each genre through an API request to TMDB API, which are rendered on the wheel. Spinning the wheel randomly selects one of these films to recommend to the user.

Once the quiz is complete, or the roulette has been spun, the user is presented with a results page showing their film recommendation, including title, movie poster, genre and synopsis. Users can also start again, choose a different path, and get as many recommendations as they like.

Requirements

The *technical requirements* for the project are listed in the tech stack below. Instructions for how to set up and run the project can be found in the README.md file on the GitHub repository.

Tech Stack

Frontend:

- **React with Vite** (a JavaScript library using JSX (allowing use of HTML-like code in JavaScript) for UI build.
- **Figma** (wireframes for initial UI design).
- **CSS** for styling, including **Bootstrap** (used only for responsiveness).

Backend:

- **Node.js** (functionality includes proxy server interacting with API, server-side logic such as filtering and rate limiting, running JS test files).
- **dotenv** for securely managing environment variables like the API key.

API Integration:

- The Movie Database (TMDB)

Version Control:

- Git and Github (code review, deployment)

Project Management:

- Jira
- Confluence
- Google Docs
- Slack

Documentation:

- Google Docs

Team Collaboration:

- Google Meet
- Zoom for daily/weekly stands,

- Slack for more frequent communication

Minimum Viable Product (MVP) and Stretch Goals

After ideation and design phase we divided our application into those required for a minimum viable produced and those which could be seen as stretch goals, as listed below:

MVP

- ★ **Homepage**
  - User can view Moodie ‘hero’ section
  - User can view About section
  - User can ‘Start Moodie’ by clicking Button in Start Moodie section
  - User can view information about creators in Creators section
- ★ **Header**
  - User to follow links to different pages: Home, About, Creators
  - User can selected ‘Start Moodie’ button to be taken to Choose Path section
  - User can click on Moodie logo and be taken back to Homepage
- ★ **Footer**
- ★ **Choose Path Page**
  - Use is presented with two buttons which can be clicked to go to Quiz questions or Randomise
- ★ **Quiz Questions**
  - User is presented user with a number of quiz questions which render:
    - Quiz Question
    - Image
    - Answer Buttons
- ★ **Results Page**
  - User is presented with result of Quiz or result of random roulette spin
- ★ **Random Path**
  - User is provided a list of genre and can selected their desired genre
- ★ **Roulette Wheel**
  - User is able to click ‘spin’ on the roulette wheel which animates a ‘spin’ of different films connected to the chosen genre.

Stretch Goals

- ★ **Login Page**
  - User is able to login to the app
    - App remember user
    - App allows user to sign up

- ★ **History Page**
  - User can see their history of previous results/suggestions from interactions with the app
- ★ **Dark/Light mode**
  - User can toggle between light and dark mode

User Experience

We have designed our application to be intuitive and easy to navigate, using different elements such as main buttons as 'call-to-actions' so that a user can progress through the application.
We have also taken into consideration accessibility in our choice of styling and our colour scheme, using online tools to test this where applicable.

Responsive Design

Although the application has been built for the web, we also created wireframes for mobile and tablet versions.

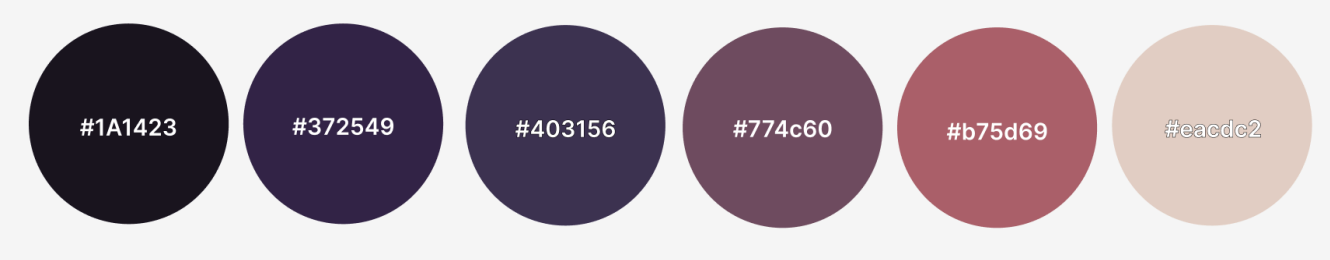
Design and Architecture

We decided to use specific fonts to ensure accessibility needs are met (E.g. Dyslexia finds certain font families easier to read).

We created both low-fidelity and high-fidelity wireframes together as a group, with designs for MVP, as well as stretch goals.

We created avatars for each creator based on a genre.

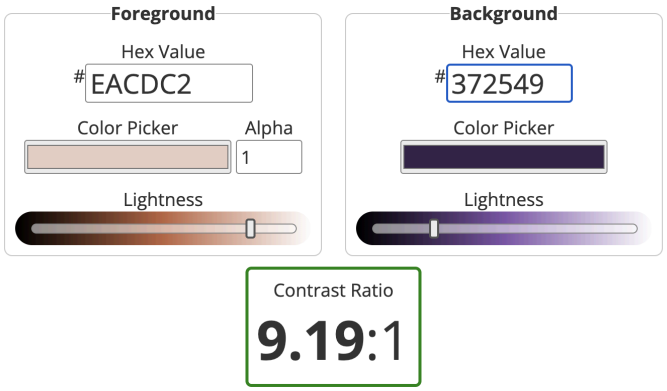
Colour Scheme



Colours used give off a ‘retro movies’ feel.

We initially designed Allows for dark/light mode that complement one another.

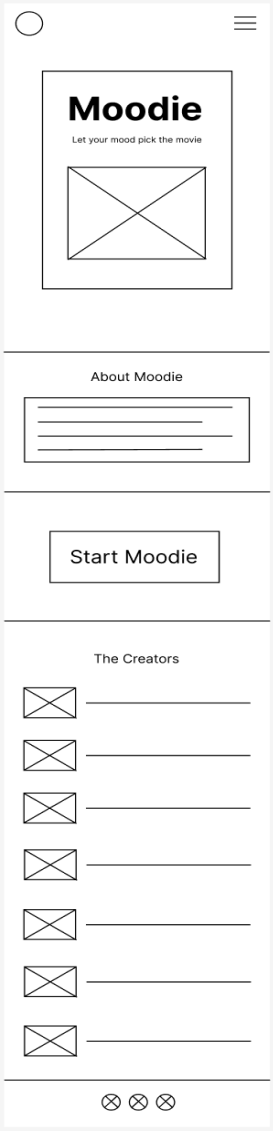
We also used accessibility tools such as WebAIM: Contrast Checker to ensure our colour scheme was accessible to both humans and screen readers.



Low Fidelity Wireframe

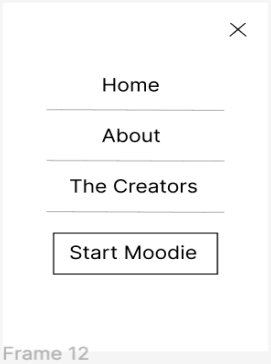
Home page

Frame 10

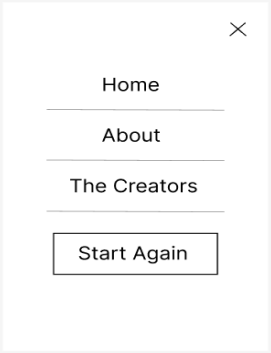


Burger menu options

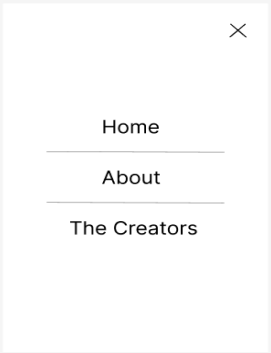
Frame 11

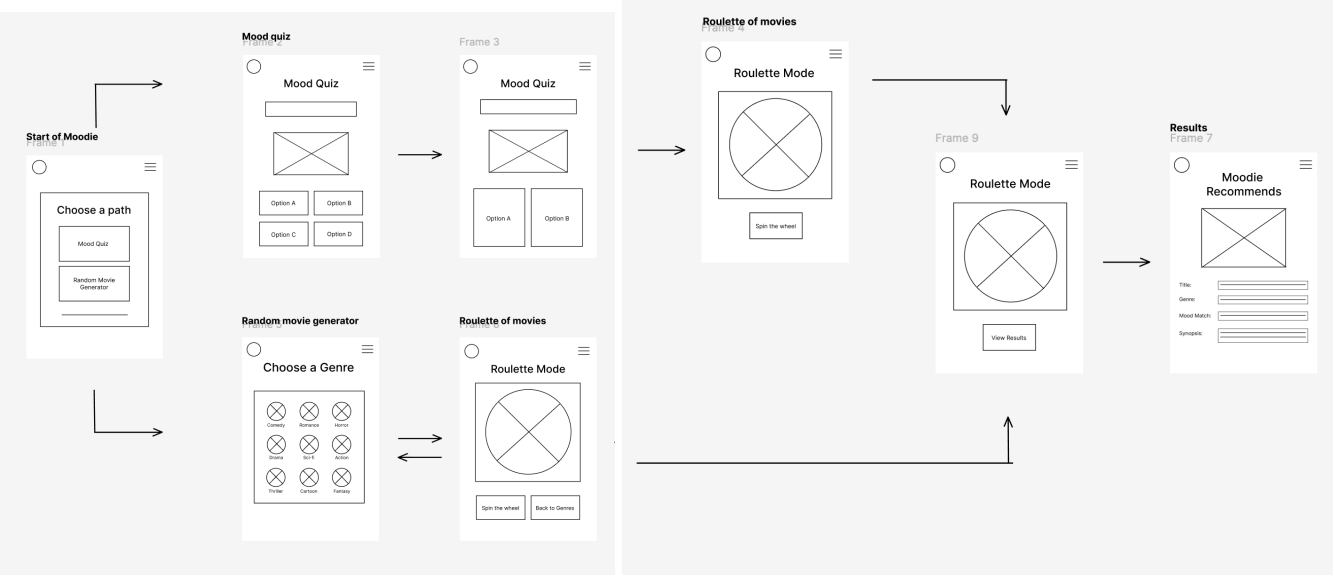


Frame 12



Frame 13

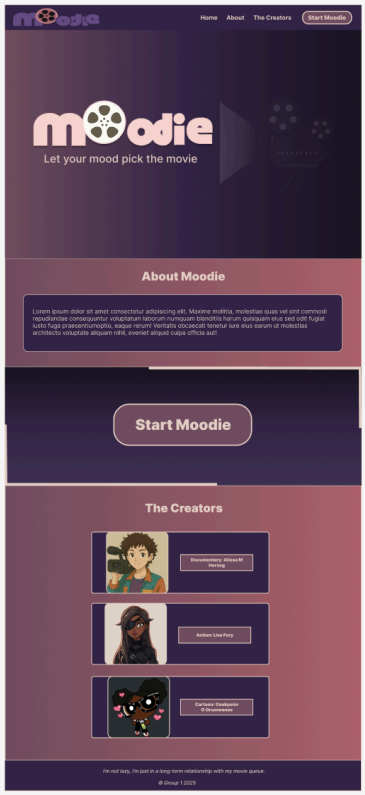




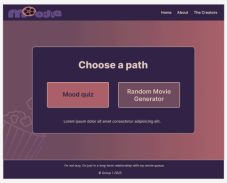
High Fidelity Wireframes

Desktop

Desktop - 4



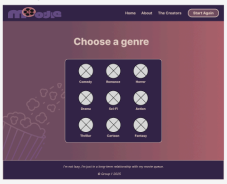
Desktop - 16



Desktop - 18

Desktop - 24

Desktop - 19

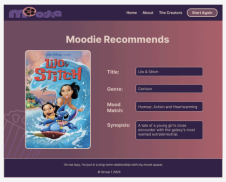


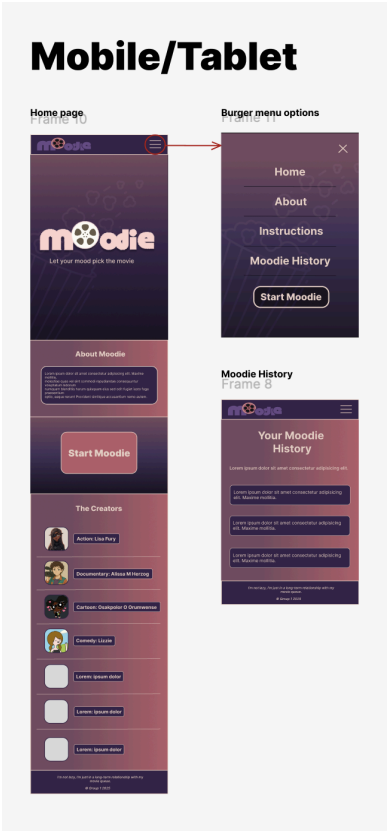
Desktop - 20

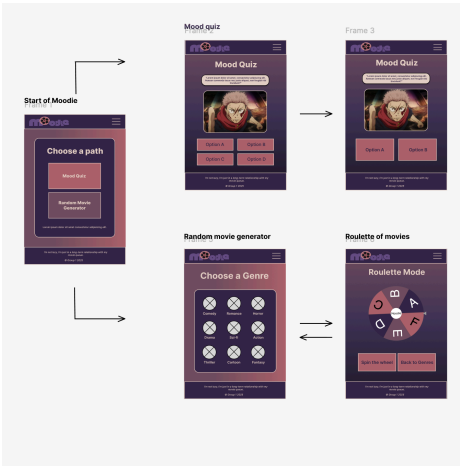
Desktop - 21

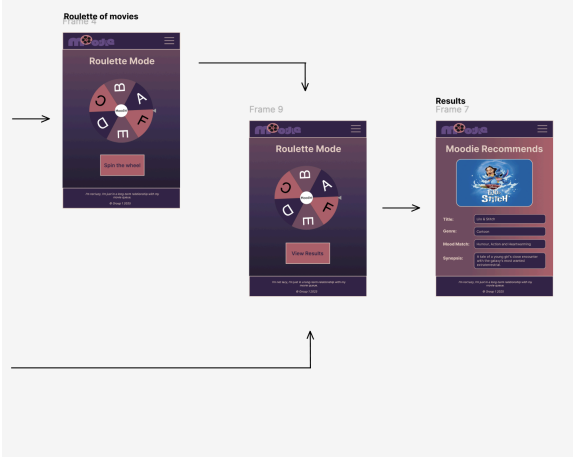
Desktop - 22

Desktop - 23









[Wireframes Link]

Implementation & Execution

Development Approach

Our team followed an Agile development methodology to ensure flexibility and continuous improvement throughout the project. Before outlining our roadmap, we began with a series of collaborative ideation sessions to generate and refine initial concepts, the problem statement we wanted to address and the high-level requirements for our application. These early meetings allowed us to explore various directions, align on a shared vision with a shared interest in the final product, and lay the foundations for the project's overall direction.

Following this we moved into a design-focussed phase that involved researching colour schemes, fonts, general visual styles, and user interface preferences, which are detailed in the previous ***Specifications & Design*** section. Once a general design direction was established, we created wireframes to map out the structure and flow of the product which helped to identify the different components and pages that were required to be built by the team. For this, we used Figma as our main tool.

With the wireframes now in place, we utilised a number of meetings to make our final decisions on the functionality we wanted to include in our application, separating into a Minimum Viable Product (MVP) and stretch goals. We developed a high-level roadmap to guide the project's progression. Using Jira, we discussed and organised the tasks required to build our product, using user stories to ensure we were aligned with project goals. Since Jira was new to several team members, we supplemented it with Google Sheets when necessary to track dependencies and ensure smooth coordination across tasks.

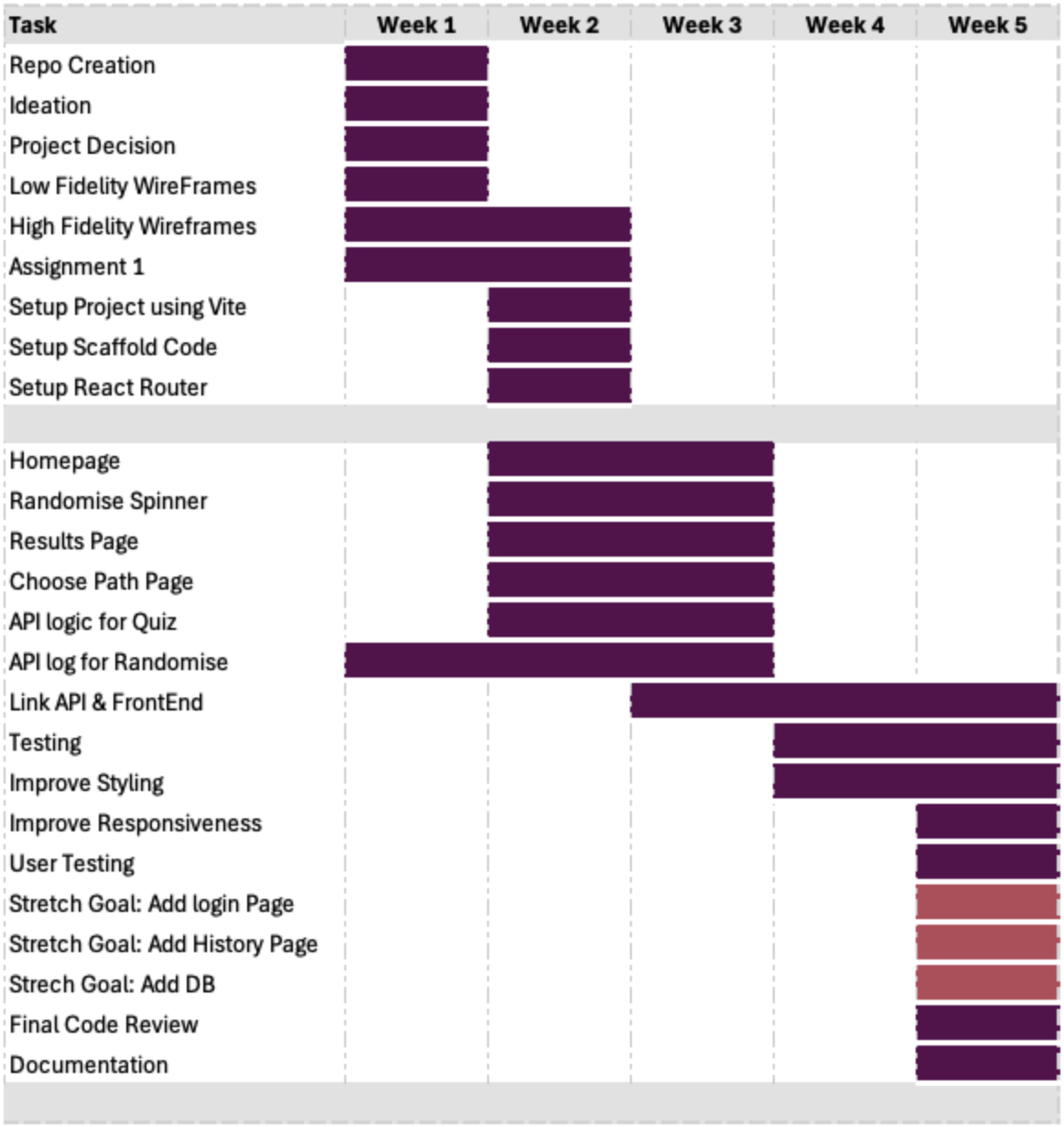
The team maintained regular communication, checking in with each other almost daily in a stand-up format where group members could identify what they were currently working on, any blockers they had or any issues they potentially were facing. Zoom meetings were

utilised to facilitate when the group wanted to collaborate in real-time, allowing the group to quickly come together for immediate tasks, decision-making, bug/issue fixing, or discussions that required multiple team member input. Again, Slack was used to communicate these meetings and a followup message was shared outlining what took place on the call, what was discussed, any outcomes and follow-up actions following the call so that the whole group was kept up to date.

During the initial project discussions and ideation phases, the group decided to approach the Final Project as an opportunity to apply the new skills we had acquired over the previous 12 weeks, whilst also learning together and experimenting with the development process. We aimed to be as agile and collaborative as possible, ensuring that each team member had a chance to contribute to the areas they were most interested in or wanted to practice, whether it was in front end, back end, design or project management. The focus on collective growth and experience became our core Group 1 principle.

Roadmap

Below outlines the Roadmap for the design, execution and review of our application. As we worked in an agile environment, with differing commitments and availability we used this as a guide to ensure the project progressed as required whilst allowing us to be flexible to any changes.



During the project, we broke the roadmap down into smaller tasks using different tools such as Jira and Google Sheets to ensure that everyone was comfortable with what was currently under development, if anyone required any further assistance as well as ensuring an even spread of the work. An example of this can be found below:

Task/To Do	Owner(s)	In Progress/Complete
• Connect API to User Answers and filter through the database to produce X number of results	Lizzie	Pretty much complete - needs tweaks on styling and results display.

Roles

Our group embraced a flexible approach to roles and responsibilities, rotating them as needed throughout the project. This allowed each team member to explore different aspects of the application and development process.

Below, is an overview of different roles and responsibilities team members took on during the course of the project.

	Lisa	Lizzie	Rima	Gaby	Alissa	Osayi	Danni
Initial Repo Setup						X	
Project Definition	X	X	X	X	X	X	X
Design Decisions	X	X	X	X	X	X	X
Wireframes	X	X			X		
API research		X			X		X
Project/Product Management			X			X	X
Setup - Vite	X		X				
Setup - Scaffold		X			X	X	X
Setup - React Router	X	X			X		X
HomePage	X						
Front End Quiz Logic		X			X		X
Back End Quiz Logic		X			X		
Styling	X	X	X	X	X	X	X

Responsiveness	X					X	
Roulette Wheel			X	X			
Results Page		X	X			X	
Errors/Bugs						X	
API integration		X		X	X		
Testing					X		X
Results Page			X			X	
Final Review of Repo		X			X		X
README		X					
Documentation			X		X		X
Presentation	X					X	

All group members were involved at critical checkpoints and decisions during the project.

Tools and Libraries

The below table outlines tools and libraries utilised in the creation of ***Moodie***.

JavaScript Library	React
JavaScript Runtime Environment	Node.js
Development Tool/Build Tool	Vite
Version Control	Git and Github to manage our project. We utilised branches for building features and merged into main via PRs. We
Project Management	Jira, Confluence, Google Sheets, Slack
Testing	Vitest, JavaScript
Styling Library	Bootstrap (Responsiveness only)
Design and Styling	Figma

Implementation Process

Achievements
We have successfully and collaboratively worked as a team and have a styled MVP with working functionality. We have iterated designs and decisions on multiple occasions to arrive at our final product.

Challenges
Application Challenges
We experienced a number of challenges with styling and responsiveness, merge conflicts from working on overlapping functionality across different branches, and a few major bugs that temporarily broke the app. In some instances we had to use Git to revert changes and restore stable versions of the project.
There were also moments where we needed to reconsider and slightly revise earlier design or functionality decisions. Any major changes were always discussed openly and collaboratively, with input from the whole group where possible.

All of the above has been an invaluable learning curve for ourselves as new developers.

External Challenges
In addition to the technical aspects of the project, our team also navigated a range of external circumstances that impacted our workflow at various points. These included personal commitments, illnesses and other responsibilities outside of the project. Despites these challenges, we maintained a strong sense of teamwork and adaptability, supporting one another with compassion and stepping in where needed to keep the project moving forward. Our open communication and mutual respect allowed us to stay aligned and ultimately enabled us to deliver a functioning MVP.

Change Process
Although big decisions that impacted the application as a whole were made as a group, we also encouraged a level of individual autonomy for smaller changes and decisions, as long as these were clearly communicated - team members were trusted to take initiative within their current areas of focus.

We also utilised the Pull Request process to review changes and used this opportunity to open discussion with the group if decisions had been made that required other's input.

Agile Development

During the development process, our group exhibited and used the following agile elements:

- Iterative Development
    - We worked in short development cycles, ensuring we lined up dependencies and pivoted where required.
    - We used group meetings and discussions to improve the app, and how the team was working together, step by step.
- Refactoring
    - We made improvements to the code structure as we progressed and developed the application.
    - This ranged from ensuring our nomenclature was consistent where possible, to abstracting components and ensuring routing was used appropriately.
    - We also used refactoring on multiple occasions to ensure our styling was consistent across the application.
- Code Reviews
    - When building different aspects of the application on individual branches, we used Pull Requests to merge into the main branch.
    - We used information names for PRs and listed high-level bullet points of changes made so that other team members could review, query, or raise any concerns before merging into main.
- Task/Backlog Management
    - Initially we utilised Jira for task creation and task management but as the project progressed, due to some team members feeling less confident with this tool, we moved task ownership and delegation to google sheets.
- Pair/Group Programming
    - On a number of occasions we used pair/group programming, using the driver-navigator technique to work together on setting up the scaffold code, certain functionality with ownership overlap, and working on major bugs together.

Testing Strategy

Quiz Logic

The main testing priority was to ensure that all possible quiz answer combinations returned results (film recommendations) when the quiz answer parameters were used to query the API. After creating an initial set of quiz questions corresponding to filtering API parameters, we created a script to generate all possible quiz answer combinations as JSON.

Questions were adapted to limit options for answers, to create a manageable number of combinations. Then a quiz logic test script was written to test that all possible combinations return a film result from the API by creating API queries from the JSON data. The script logs any failures in a separate JSON file. Questions, answers, and filtered parameters were adapted until no failures in testing were recorded, using a Test Driven Development Approach.

User testing

User testing was carried out by Alissa's flatmate Ed. His experience included positive feedback and room for improvement as follows:

- Enjoyed the UI and colour scheme, found it pleasing, especially the colour changing buttons on the quiz.
- Enjoyed the Moodie quiz, felt "cared for" as Moodie did all the work for him of picking a film.
- Liked the creators section but felt like the creator card should reveal more info about the creators upon clicking. The hover functionality gave Ed the impression that the cards would do something when clicked.
- Skipped past the about section, wasn't really interested in reading it.
- Felt the app was really straightforward and obvious to use.
- Would have like there to be back buttons so that he could change some of his answers to the quiz
- Found the Choose a genre page a little overwhelming initially with so many options, but enjoyed the icons / buttons and their UI.

- Enjoyed the randomise wheel and how simple and fun it was to spin it.
- Thought the app was a good idea overall.

Unit testing

Unit testing was carried out on a few key buttons to ensure that components render correctly and that events are handled. After research and issues implementing jest solely in isolation, we used ***Vitetest*** as our testing framework for this, due to it being designed specifically for projects that use Vite. If we had more time we would continue to build up our testing suite.

This area is probably our weakest area, as we could have done more robust testing to achieve greater code coverage, but we prioritised getting a functioning MVP and testing the application manually when building functionality.

Moodie

Let your mood pick the movie

Conclusion

Overall we managed to successfully deliver a minimum viable product that carries out the requirement initially set out, implementing user interactivity to provide a film recommendation. We successfully adapted the product throughout its lifecycle using Agile development, collaboration and communication. The user experience can be improved through gathering more data and incorporating user feedback. Our development process could be improved through better implementation of testing.