Tripinary - Milestone 2 Report

John Camino, Renz Gabrinao, Beverly Yen, Lilian Pham

GitHub Repo

Demo Video

Tripinary Website

Analysis of the project's success in meeting the user needs and solving the problem

Link to Peer Testing Session Survey Questions: Peer Testing Document

Peer Testing Feedback from classmates: Link to the responses are in this Google Sheets

Feedback from Peer Testing Session		
Question we asked testers:	**Feedback received:**	**How our team improved:**
What was your overall impression of the app?	I received no tip about any city searched for. When using the green bar, it was tough to see the difference between the suggested location and the new searched location. overall, I would use it for starting suggestions. Great would love to use this in the future. Very easy to follow and great planning. It looked cohesive and it was clear how to use.	Our team implemented an improvement by migrating our AI model to OpenAI Turbo-GPT, which offers faster speed. Our team also addressed the tip issue, as we realized it was an issue with our Google Places API key. We fixed it by switching it out with a different key.
Did you run into any issues with activity selection or itinerary generation? What were these issues and how did you encounter them?	I didn't run into any errors. I ran into no issues selecting activities Very easy task No just scrolling to find places was tough You could consider making a loading message for when the API is called	We added a static message to tell users to wait a few seconds/minutes as their itinerary is generated. There was no improvement we could make because we were making 1 API call, and if the input size is very large, it will take the AI model longer to generate an itinerary. We added a static message for clarity, so the user will know to wait after they press the button to generate an itinerary because it will depend on their input size.
Were the new suggestions an improvement? How so?	Navigation arrows for the options should be brightened. Oh just error messages. Other than that good	We improved our activity navigation arrow by enlarging it and making it brighter to stand out against activity images. We also added more descriptor error messages, to notify the

Feedback from Peer Testing Session		
Question we asked testers:	**Feedback received:**	**How our team improved:**
	I couldn't tell that by clicking on the itinerary items you could get more information	user the reason why it failed (whether it be an unparsable response given by AI, or other issues). We also fixed the conduction of clicking on itinerary items by adding a static message to let users know that they can hover/press on individual items to learn more about them.

Analysis of the project's SDLC model and how it was used in the project

Our group chose to use the Agile SDLC model for our project. We found that Agile was flexible and allowed us to build our features incrementally, which was useful because our ideas often changed as we worked on the app. We knew that some components of our project would be harder than others (such as getting AI to generate itineraries and making interactive features like the map view and activities carousel) so breaking tasks into smaller pieces just made the workflow easier to manage. With Agile, we were able to add features and get feedback from one another often. To help us use Agile, we also used Scrum and Kanban.

- Scrum helped us work in fixed time periods so we could meet deadlines for Milestone 0, 1, and 2. We did 2-week sprints, and at the end of each, we either implemented a new feature or contributed to documentation.
    - We had stand-ups two times a week, and then daily two weeks before the project was due (July 28th-August 5th). Our stand-ups were 15-minute meetings where we addressed problems right away, like when the Google Maps part for the map panel wasn't connecting correctly, or when the AI's itinerary data was hard to read for our activity list.
    - In sprint planning, we decided exactly what to work on next, picking tasks from our main list. For example, in our first sprint, we focused on getting the basic AI itinerary generation to work.
    - We also had sprint reviews where we discussed with each other what we had completed, like a basic itinerary displayed on the page or activity cards you could click.
- We also used Kanban board (on our GitHub repo) to keep track of where every task was, like building the activities carousel, getting itinerary data from the backend, or fixing the map panel.

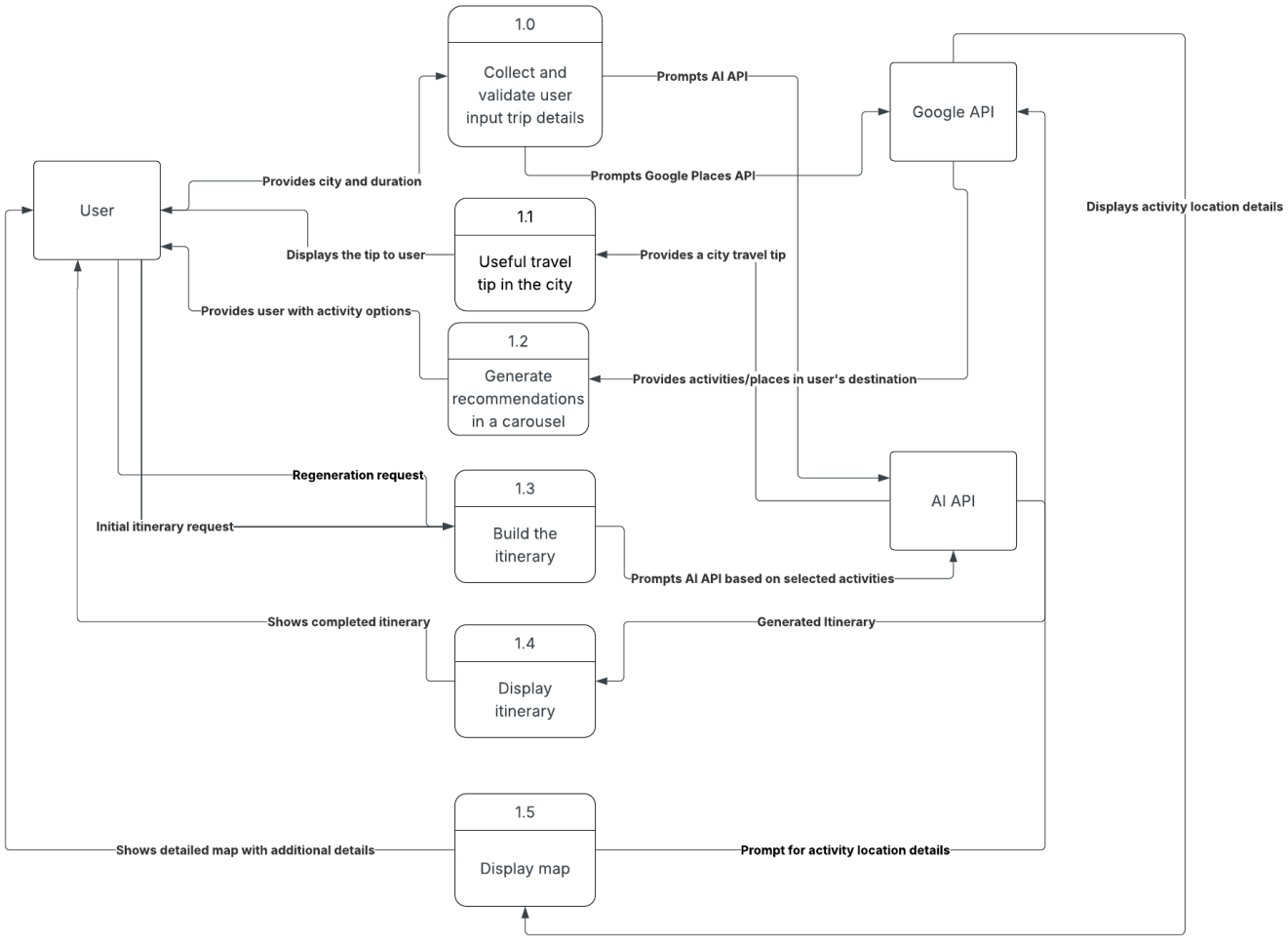
Changes made to the model during the project:

1. Adjusting Sprint Lengths: At first, we tried 1-week sprints, but we realized they were too short. It was tough to finish big tasks, especially with the AI work and figuring out how to use APIs. So, after our first review meeting, we all agreed to switch to 2-week sprints. This gave us enough time to build features without feeling rushed.
2. Allocating more times for testing: As Tripinary got more complex, we realized our regular coding tests were not catching every bug, especially since we all worked on different features and merged them together. So, we started to add special time slots within our sprints for dedicated testing. Team members would specifically try to find bugs, record them down, and make sure all parts of the app, like the main itinerary page and the map pop-up, worked together.

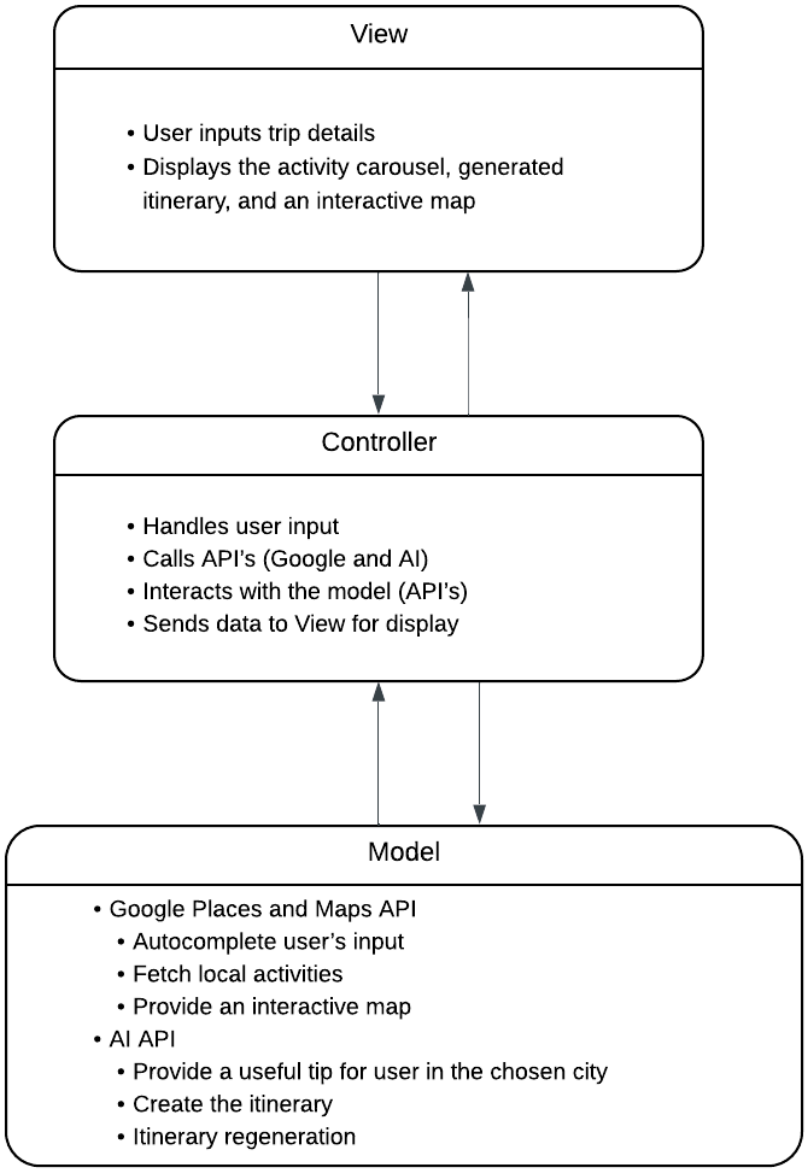
Project Architecture

- An updated (since M1) level 1 DFD of the application's structure and how data flows within the application
- An updated (since M1) MVC model of the application

DFD Level 1:



MVC Diagram:


View
• User inputs trip details
• Displays the activity carousel, generated itinerary, and an interactive map
Controller
• Handles user input
• Calls API's (Google and AI)
• Interacts with the model (API's)
• Sends data to View for display
Model
• Google Places and Maps API
• Autocomplete user's input
• Fetch local activities
• Provide an interactive map
• AI API
• Provide a useful tip for user in the chosen city
• Create the itinerary
• Itinerary regeneration

Project's Future Work and Potential Improvements

1. Improved Place Matching Accuracy

At the moment, Tripinary tries to guess which real-world places match the names in the itinerary by directly putting in the text in the search bar, but this doesn't always work perfectly. In the future, we want to make this process more accurate by using official location IDs and a better text-matching method so we can show the correct reviews, photos and other details for each palace.

2. Third Party Review and Rating Integration

Reviews from platforms such as Yelp are not integrated into the card, and we aim to display live ratings, reviews and guest images via external APIs.

3. Drag and Drop Customization

Currently, itineraries are auto-generated with minimal rearrangement and users can't really manually fine-tune their itineraries. We plan to allow users to rearrange, delete or add their own activities using a drag and drop interface.

4. Share Saved Trips and User Accounts

Future versions of Tripinary could support login functionality, letting users save, share, and revisit past itineraries. We could use resources such as Firebase Authentication to secure user session management. Another cool thing to add would be collaboration between saved itineraries.

5. Multilingual and Accessibility Support

We plan to make Tripinary more accessible around the world by supporting multiple languages and optimizing the interface for screen readers and keyboard navigation.

6. AI-Chatbot Assistant

A travel assistant chatbot to allow users to ask travel-related questions and get real-time help while planning.

Lessons Learned & Project Takeaways

1. Context Sharing Across Multiple Components

One of the challenges we encountered was managing shared states across multiple React components. We had various pieces of data, such as selected places and generated itineraries, that needed to be accessible and consistent across different parts of our application. This process initially led to inconsistencies and bugs, but we resolved this by implementing React Context and using the browser's SessionStorage, allowing us to streamline the flow of data, making state management less prone to errors.

2. API constraints

Another area where we had challenges was dealing with third-party API constraints. Specifically, the free API limits. We almost used up the free credits just from testing. We quickly learned the importance of optimizing our API usage. To mitigate this, we cached data when appropriate and minimized unnecessary requests to stay within the limits

3. AI API malfunctions

The AI API calls often took a long time to respond, which caused delays in loading the itinerary. In some cases, the API failed completely and returned an error, leaving the user without a generated plan. Since the issue stems from the third-party AI service, it's not something we can fully fix on our end. However, in the future, we could add better error handling, retry mechanisms, or even provide cached fallback results or partial itineraries while the full request is being processed.

4. Team Version Control

As a team, something else we learned was the importance of effective version control and collaboration. Since our team was contributing to the code simultaneously, we faced challenges with merge conflicts, overwriting changes, and ensuring everyone was working on the most up-to-date codebase. However, we quickly learned the importance of clearing branches and doing regular pull requests and code reviews.

Features implemented for each API & Changes since Planning

AI API:

1. Timestamped Itinerary List → Creates a personalized travel itinerary based on the user's location preferences and trip duration.
    a. Detailed Description: After the user selects their activity from the homepage and selects 'Generate Itinerary', they are directed to their personalized travel itinerary page. Users

can view their day-by-day itinerary, and click on each individual activity to display an interactive map view. Each activity will be accompanied by a time stamp that they can follow.

2. Itinerary Regeneration → Based on the itinerary that is created from user inputs, this feature would allow users to press a regeneration button that gives them a new itinerary.
    a. Detailed Description: When the user scrolls down to the bottom of the itinerary page, they will see a button labelled 'Regenerate". If they are unsatisfied with their current itinerary, they can press this button and wait a few seconds for a new itinerary to be generated. This new itinerary contains the same activities that they selected previously, but rearranged in a different order and with new filler activities.
3. Destination Tips → Suggests a travel tip based on the destination
    a. Detailed Description: After the user clicks "Plan my Trip!" the Activity Carousel will pop-up which will also include a helpful tip when travelling to their destination.

Google Maps API:

4. Place Search & Review → From Google Place Details API, allows developers to search for a place and get their details like name, address, ratings, summary, etc..
    a. Detailed Description: Based on the map of the itinerary that is generated, this feature allows users to locate and search for specific places (i.e. restaurants, hikes, etc.) and display a map view of its location. Also shows the latest five reviews of specific activities on the user's personalized itinerary. The place review will include star rating (out of 5), description, and name of the reviewer.

5. Activity Suggestions → Recommends specific activities and places (ex: restaurants, hikes, etc.) depending on user inputs and allows users to add this to their itinerary.
    b. Detailed Description: After the user fills out details regarding their trip, including the location and duration, google will recommend a list of activities under different categories, including Food and Drinks, Attractions and Sightseeing, Activities and Recreation and Shopping and display it in an activity carousel.

6. Map Display → This feature displays an interactive map of where each individual activity is located.
    c. Detailed Description: After the itinerary is generated, the user is able to click on specific activities and an interactive map will display.

Changes:

- Initially, our group planned to implement the "Activity Suggestions" feature using an AI API. Instead, we decided later on that it would be a better fit to implement this feature using the Google Maps Places API, as this way we could get reviews of each activity, image, reviews, and place id which would be used later for our itinerary page.
- We replaced the "Activity Suggestions" feature that was (initially) supposed to be implemented with the AI API with the "Helpful Travel Tips" feature. We thought that this would be helpful for the user as it would provide additional context about their travel destination before they make their activity selection.

CI/CD Pipeline

Since we are using Github Actions, we created a continuous integration workflow (ci.yml) that runs our tests automatically whenever code is pulled and pushed to the main branch in our repository. This ensures that the tests are triggered with every change, making catching bugs easier. More details about the unit and integration tests in the next section.

The CD Pipeline is done in a forked repository due to Vercel's limitation of deploying repositories within private organizations. It is as follows, and triggers deployment on push to the main branch. We monitored any pipeline failures after each push to a branch and set notifications to email us when any of the tests failed.

```yaml
name: Deploy Tripinary to Vercel

on:
  push:
    branches:
      - main

jobs:
  deploy:
    runs-on: ubuntu-latest
    steps:
      - name: Checkout Code
        uses: actions/checkout@v4

      - name: Node.JS
        uses: actions/setup-node@v4
        with:
          node-version: '20'

      - name: Install Dependencies
        run: npm ci

      - name: Run Tests
        run: npx vitest run

      - name: Build project
        run: npm run build

      - name: Deploy to Vercel
        run: |
          npm install -g vercel
          vercel --prod --yes --token=$VERCEL_TOKEN \
            --org-id=$VERCEL_ORG_ID \
            --project-id=$VERCEL_PROJECT_ID

          //placed in Github Secrets
```

[CD Pipeline]

Tripinary's Testing Strategy and Implementation

Unit Testing

Slideout

updateMapSource → for the core logic of updateMapSource, verifies that the function correctly forms the Google Maps Embed API URL based on a given search query and destination name

Tip Fetch Function

We created a test that validates the behaviour of the fetchTips function, which retrieves data through a POST request to OpenRouter's chat completion API. This function provides the user with a useful tip based on their destination. This test checks the following:

1. Successful response: This verifies that the function returns a valid tip from the API.
2. Missing response: This verifies if the function returns the correct fallback message when the response from API lacks the expected content.
3. Network failure: This verifies if the function returns the appropriate error message when the API request fails due to network issues.

Handling of Missing Data:

We created a test to simulate a scenario where no itinerary data or selected places are provided to the component.

- It confirms that instead of displaying an empty or broken page, the component shows a user-friendly fallback message (e.g., "No itinerary available. Please select destinations/activities...").

Handling of Error States:

This test simulates a scenario where an error has occurred during the itinerary generation process. It also verifies that the component correctly captures and displays the error message provided by the application state, ensuring that the user is informed of the issue.

Integration Tests

Activity Suggestions:

These tests use the ItineraryContext and PoisContext that tracks which points of interest the user chooses and tests for component rendering.

- Rendering Tests:
    - Checks if input form is in the document screen

    - Checks if an alert is shown on screen if the ‘Plan My Trip’ button is clicked
    - Checks if the ‘Generate Itinerary’ is disabled on page load (means there are no POIs selected)
- Context Tests:
    - Calling updateDestinationName(str) from ItineraryContext should update destinationName attribute from itineraryForm
    - Calling updateDuration(str) from ItineraryContext should update duration attribute from itineraryForm
    - Calling isPoisEmpty() returns true if pois list is empty in PoisContext
- Google Places API NearbySearch test
    - Does not actually test and validate response from Google Places API but rather checks if the findNearbyPlaces() function from poisContext does a proper API call to the Google Place API endpoint. Using vitest, we can inject a mock response to return to the API call and the test checks if the response has an object containing the proper headers, method, and body.

Itinerary Rendering:

- This test uses a mock API response to simulate a successful data fetch.
- It verifies that the Itinerary component correctly displays the structure of the itinerary, confirming that the number of "day labels" matches the mock data (e.g., "Day 1", "Day 2").
- It also checks for the presence and correct formatting of each individual activity, ensuring the time and activity descriptions are displayed as expected.

Slideout

These tests validate the SidePanel component behavior, including its interaction with mocked backend API calls, simulating a user’s experience. They are implemented using Vitest and React Testing Library.

1. Initial Render & Defaults: Tests ensure the SidePanel renders correctly with default map content and information
2. Search Query Prop Change: Verifies that if the searchQuery prop passed to the SidePanel changes, the map updates accordingly, and a new API call is triggered to fetch relevant place details.
3. User Search Interaction: Simulates a user typing into the search bar and clicking the search button. It then asserts that the map updates, the correct backend API call is made, and the fetched place details (name, address, rating, reviews) are displayed.
4. Error Message Display on Failed Fetch: Confirms that if the backend API returns an error (e.g., "No place found"), the SidePanel displays an appropriate user-friendly error message and clears any previously displayed place details.
5. Review Display Logic: Checks that the component correctly renders either the default placeholder reviews or the dynamically fetched Google Place reviews based on the API response.

Backend Integration Tests

Validate the functionality of Express API endpoints, implemented using Supertest to make HTTP requests and Nock to mock external API calls

GET /api/place-details

- → Successful Fetch (verifies that a valid query is returned)
- → Missing Query (API returns 400 status)
- → No Place Found (Checks for 404 status)
- → API error (Checks for 500 status)

Backend Itinerary Generation Test:

This test will validate the logic for generating an itinerary by simulating a request to the /api/generate-itinerary endpoint with valid user input data. It also mocks the response from an AI model (OpenRouter) to ensure the backend can correctly handle and parse the data.

- It verifies that the API endpoint responds with a successful status code (200) and that the returned data is a properly structured itinerary array.

List of Known Bugs/Issues:

Bug Description	Steps to reproduce	Severity
Map Display Placeholder image (e.g., Simon Fraser University) may display for 1-2 seconds before the actual place appears.	After you have entered the trip duration, destination, and selected your activities, you can generate an itinerary. If you press on an individual activity, you may see our placeholder image of SFU for <1 second.	1 - Minor This is just a placeholder image while the place loads, so it has next to no effect on user experience
Sometimes you may get a backend error indicating that JSON data is unparseable. In this case, you may need to regenerate your itinerary again (or multiple times).	After you have entered the trip duration, destination, and selected your activities, you can generate an itinerary. If you have selected activities with unusual characters, then AI may have issues getting a response that is parseable.	2 - Minor/Medium An error message on the Tripinary Main page will appear. Users can regenerate their itinerary.
A load failure after pressing "Generate Itinerary" could indicate that an API key limit has been reached. You can try again after a brief wait.	Our AI model (OpenAI GPT 3.5 Turbo), should have up to 1000 uses per day. Unless this is exceeded, there should be no issue.	2 - Minor/Medium An error message on the Tripinary Main page will appear. Users can wait until the key is valid to use again.
Generating or regenerating an itinerary may take a while, depending on the number of days and activities selected. You may need to wait up to a few minutes.	After you have entered the trip duration, destination, and selected your activities, you can generate an itinerary. If you have many selected activities (we have a limitation of 80), then AI may take a long time to generate an itinerary.	2 - Minor/Medium This is due to the size of input. It will take longer depending on inputs.
After the user clicks the ‘Plan my Trip’ button, sometimes the tips don’t show up because of an API issue/overuse of Google Maps API.	Our Google Places API has a limit of up to $409 free uses. Unless this is exceeded, there should be no issue.	2 - Minor/Medium An error message on the Tripinary Main page will appear. Users can wait until the key is valid to use again.

Appendix

Link to Peer Testing Session Survey Questions: Peer Testing Document

Peer Testing Feedback from classmates: Link to the responses are in this Google Sheets

Provide a list of all group members and their detailed contributions to the project

Group Members:	**Contributions to project:**
Entire Team:	• Reviewed and submitted the group contract • Researched different APIs, tech stacks, and brainstormed different ideas and features for our final project • Participated in many meetings to complete Milestone 0, Milestone 1, Milestone 2 • Created GitHub issues for each relevant task and a Kanban board to keep track of progress • Began to implement the project using Agile Scrum framework • Write unit and integration tests to test individual functions and OpenRouter and Google Places API • Contributed to Peer Testing Session Testing Document, creating and improving from Peer Test Day feedback form • Created and refactored DFD diagram for Milestone 1 and Milestone 2 • Recorded Milestone 1 video, Milestone 2 video, and contributed to writing presentation script for each respective one • Worked on various different app components, and divided pages to individually work on • Wrote unit and integration tests for separate features and components • Contributed to mock up interface design on Figma • Participated in the Peer Review session on July 28, 2025
Beverly Yen	• Completed detailed descriptions for all 12 features (primary and backup options). • Came up with and designed a hand-drawn project logo, and contributed to choosing the final logo. • Researched and decided on programs for web-hosting and front-end website design • Created low-fidelity wireframe for the map display on final project and later transformed it into a mid-fidelity wireframe using Figma • Created Project Schedule table, with deadlines assigned to specific tasks relating to the team's final project • Involved in covering and discussing all key WBS components (project planning, UI/UX design, frontend stack, API setup, testing, deployment, web hosting, documentation) • Created Level 0 & Level 1 Data Flow Diagrams • Involved in editing and creating the script for milestone 1 and 2 video. • In charge of the code for the map slideout display section of the website, including Google Places API and Maps Implementation • Created some tests for our CI/CD pipeline, mainly for the maps slideout and

	backend section • Deployed the website on vercel • Edited Project video for Milestone 1 and Milestone 2
Renz Gabrinao	• Set up and organized the group's shared GitHub repository, creating specific directories and folders corresponding to the project starter template • Collaborated on writing a clear overview for the application's idea, purpose, problems addressed, and user group assistance, integrating it into the proposal document. • Created low-fidelity wireframe for the attraction/activities page on final project and later transformed it into a mid-fidelity wireframe using Figma • Contributed to identifying and discussing potential risks and issues for the final project in Milestone 1, as well as mitigation strategies • Involved in covering and discussing all WBS components (project planning, UI/UX design, frontend stack, API setup, testing, deployment, web hosting, documentation) • Created Level 0 & Level 1 Data Flow Diagrams • Created MVC diagram which is a high-level overview of how the application will be structured • In charge of the code for the activity-suggestions section of the website. • Contributed to Peer Testing Document for Peer Testing Day • Managed our secured API keys for Google Places and OpenRouter.
John Camino	• Project Manager, responsible for keeping track of the team's slack updates, GitHub issues and settling milestones with deadline • Submitted our group's reports through Canvas • Started developing the low-fidelity wireframe and planned user interactions on the project website, detailing pages and feature placement. • Created low-fidelity wireframe for project's homepage and later transformed it into a mid-fidelity wireframe using Figma • Contributed to documenting the reasons for choosing Agile Scrum in the Milestone 1 working document • Brainstormed and documented items for the homepage footer • Involved in covering and discussing all WBS components (project planning, UI/UX design, frontend stack, API setup, testing, deployment, web hosting, documentation) • Created the initial Level 0, and Level 1 Data Flow Diagrams using LucidChart • Worked on the .css and .jsx code of the homepage of the website. • Created tests for the Travel Tips sections of the website • Created the CI workflow (ci.yml file) through GitHub Actions • Updated our presentation slides using Canva and script to match our finalized project • Updated the MVC and level 1 DFD diagram to match our finalized project using LucidChart
Lilian Pham	• Wrote all details for user stories (including demographics, background, characteristics, goals, challenges, behaviors, motivations) • Worked on editing and finalizing the Milestone 0 proposal, reviewing sections like overview, user stories, user personas, feature descriptions, and tech stack to meet

	requirements. ● Created low-fidelity wireframe for project's generated itinerary page and later transformed it into a mid-fidelity wireframe using Figma ● Worked on structuring a Work Breakdown Structure (WBS), breaking the project into manageable tasks and defining deliverables/responsibilities ● Participated in the Risk Assessment and Project Schedule creation. ● Added Phase 1 (core) and Phase 2 (optional) features to the Milestone 0 proposal ● Involved in covering and discussing all WBS components (project planning, UI/UX design, frontend stack, API setup, testing, deployment, web hosting, documentation) ● Wrote integration and unit tests for the CI/CD pipeline, mainly the itinerary generation, itinerary regeneration, and API response from OpenRouter ● Worked on the Itinerary page of the website as well as API integration using OpenRouter. ● Added static messages to app interface after Peer Testing Day feedback

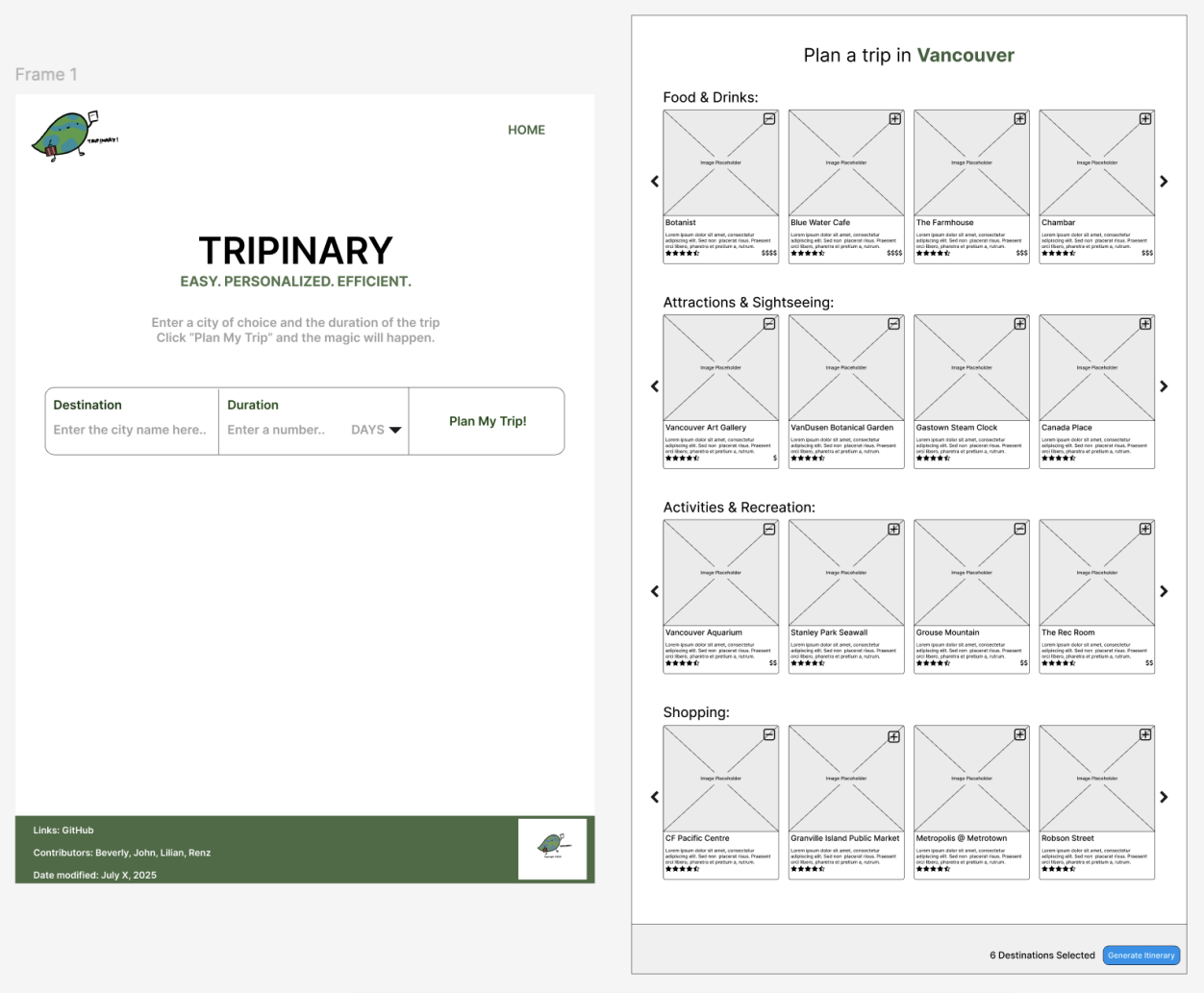
Change Log Table (since previous Milestones)
● Collectively decided to change OpenAI API feature from Activity Suggestions to Destination Trips and we also added a new Destination Tips feature to provide the user with travel tips ● For the Google Maps and Places API features, we combined Place Search and Review as one feature and moved the Activity Suggestions feature here ● Changed sprint model to bi-weekly instead of each week because we felt this was more efficient for the team ● Changed MVC and DFD diagrams via Lucid Chart

Video Presentation Script: Final Presentation Script

Step by Step instructions on how to run application: Step-by-step Instructions

Updated High-Fidelity Prototype Design using Figma:
https://www.figma.com/design/pH1PvnlZYcXPNDE79cyHoX/276-Front-End-Design?node-id=0-1&t=etpi03sqet3otaK

Homepage Wireframe Screenshot (Page 1):



Itinerary Page Wireframe Screenshot (Page 2):