# Travel Planner

- ## Contributors:
    - Laura Zang (LauraZang)
    - Alexander Long (allo2301)
    - Jack Slater (JackSlater)
    - Giorgio Shcepis (GiorgioArchie)
    - Arielle Lahtinen (ArielleLah)

Slides for Presentation:

https://docs.google.com/presentation/d/19OGajERQeI0KGLgXa9_NNPW460VEhYLDYhEOXQMaOQM/edit?usp=sharing

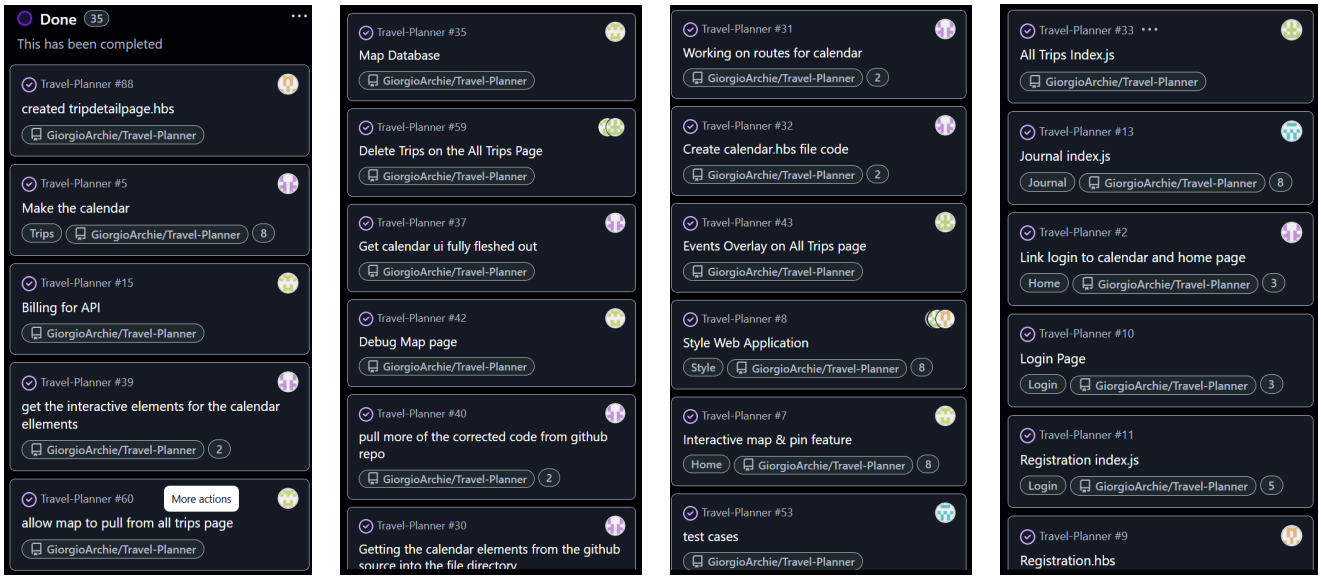
---

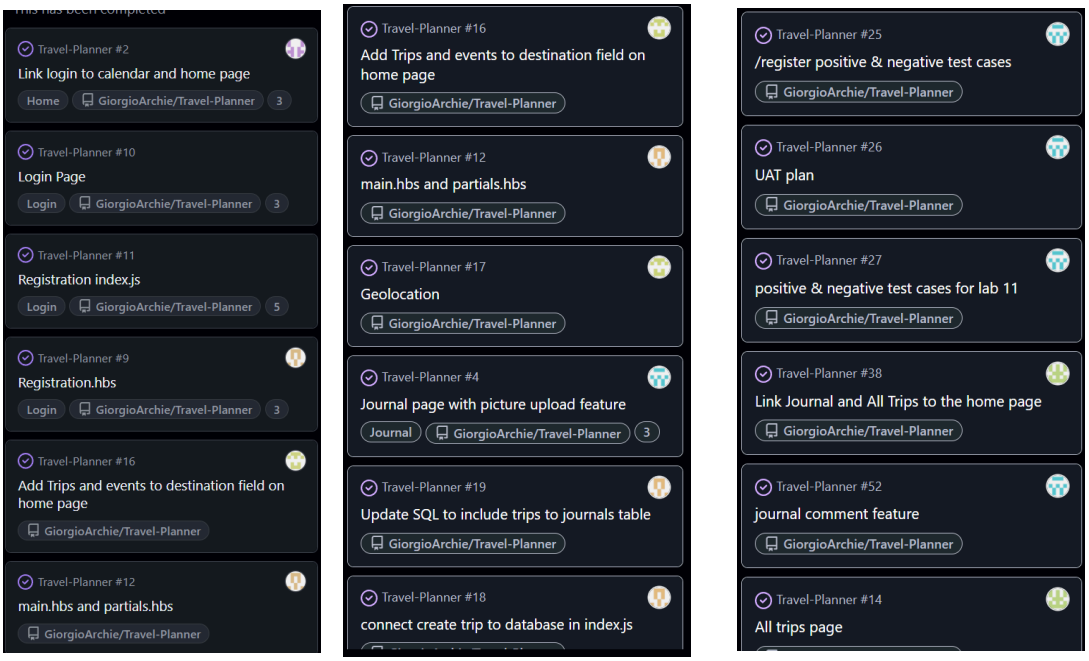
- ## Project Description:

  Travel Planner is a user-friendly web application designed to help individuals organize and document their travel experiences. Users can easily create an account and begin building personalized trip plans by entering information such as trip names, destinations, and travel dates. Once a trip is created, users can add detailed events to their itinerary, including sightseeing activities, dining plans, and other scheduled events. The application also allows users to write journal entries for each trip, giving them a space to record memories, reflections, or highlights of their travels. In addition to journal entries, users have the option to upload and view photos, creating a visual timeline of their journey. The clean and intuitive interface makes it easy for users to navigate between different trips, view event details, and update information as needed. Travel Planner aims to make organizing trips stress-free and memorable by combining planning tools with storytelling features. Whether preparing for an upcoming vacation or reflecting on a past adventure, users can manage their travel experiences in one central, easy-to-access place. The app focuses on making trip planning simple, personal, and enjoyable for travelers of all kinds.

---

- ## Project Tracker - GitHub project board:
    - Link to your Project Tracker (for instructor & TAs) https://github.com/users/GiorgioArchie/projects/1




---

- Video:
  - 5 minute or less video demonstrating your project.

- https://drive.google.com/file/d/1xO_VTo7h0ndB54wKYcc-G-iMuOTPxZz0/view?usp=sharing

---

- VCS:
  - Link to your git Repository. Instructor/TAs will check, weekly, to ensure the following are stored in your VCS repository:
  - https://github.com/GiorgioArchie/Travel-Planner.git
    - Source Code
    - Test Cases
    - Video demo
    - README.md in GitHub
    - Project documentation
    - Project Board

---

- Contributions:
  - A brief (not more than 100 words) from each team member about their contributions.
    - This should include the technologies worked on
    - Features that have contributed to
  - You can also include:
    - A screenshot of the project Board
    - A screenshot of the contributions on GitHub

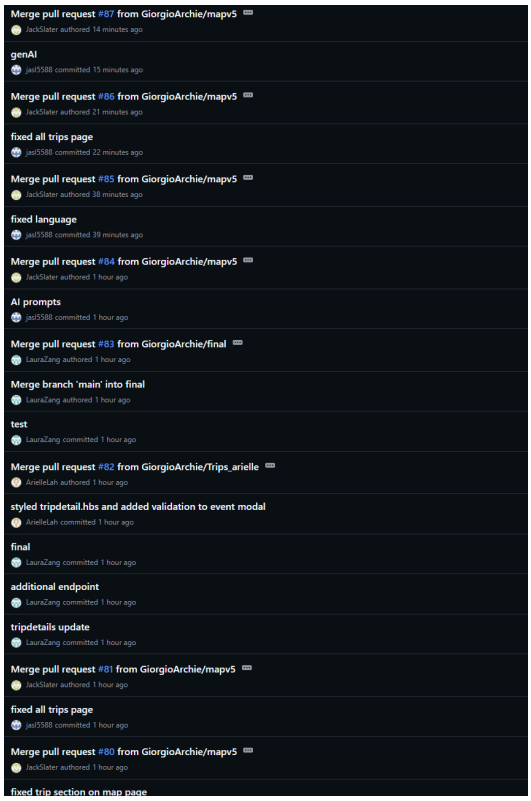
**Jack Slater:** I contributed heavily to backend integration and debugging, especially with Docker, SQL, and Google Maps API setup. I resolved issues with API keys, database connectivity, and table structure, and ensured user-specific data handling for trips and destinations. I developed and refined key frontend features like dynamic trip and journal entry displays, calendar integration, and Google Maps markers. I also handled layout fixes in hbs files, calendar event formatting, and popup UI behavior. Technologies I worked with include Node.js, Express, PostgreSQL, Docker, Handlebars, the Google Maps JavaScript API, and other APIs.

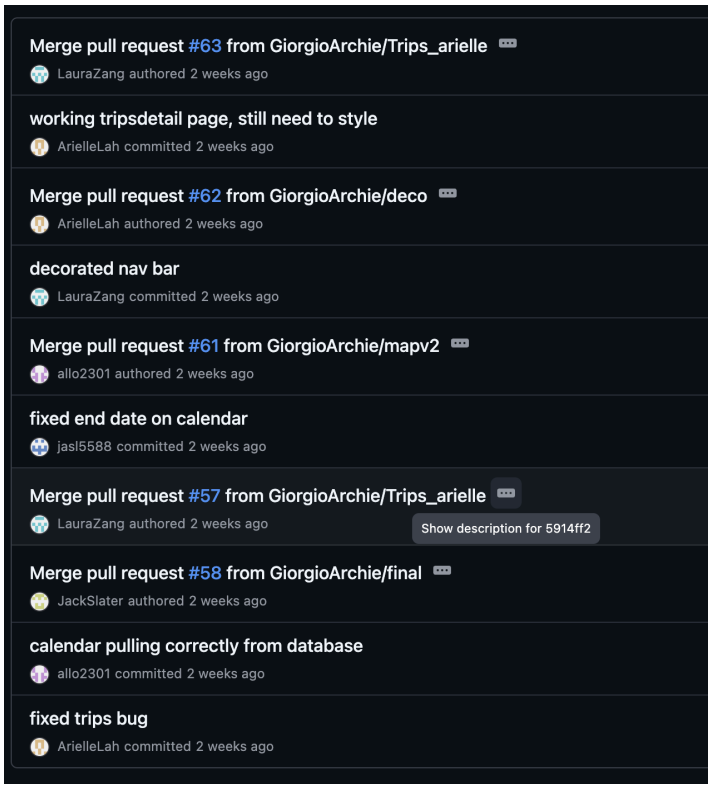
**Alexander Long:** I implemented the /calendar route and server-side logic in index.js to query PostgreSQL and efficiently map trips into Toast UI Calendar events. I styled the calendar with custom CSS along with some general CSS and bootstrap work. Client-side calendar.js initializes and populates the calendar with events. I also implemented robust error handling and static asset serving in index.js. Technologies: Node.js, Express, Handlebars, Toast UI Calendar, CSS. Features include database integration and dynamic event rendering. Screenshots of the project board and GitHub contributions illustrate our workflow and commits.

**Laura Zang:** I worked on building the Journal page functionality, which allows users to add comments and upload photos related to their trips. This involved creating both frontend and backend components to ensure smooth interaction and data storage. I also contributed to testing by writing and refining the server.spec.js file and implementing test cases for key features. In addition, I focused on enhancing the styling across the application to ensure a consistent and visually appealing user interface.

**Arielle Lahtinen:** I created the initial SQL tables and how all the tables connect with a visual diagram. I created the login, registration and “tripdetail” handlebar pages. The “tripdetail” page is rendered from a button of the All trips page and houses all of the events and the journal for that specific trip. I finished the project with the main styling of the “tripdetail” pages adding an event carousel and journal card with a light green background and blue cards. I also created the main.hbs and partials (header, footer, title, and messages) to create a cohesive web application.

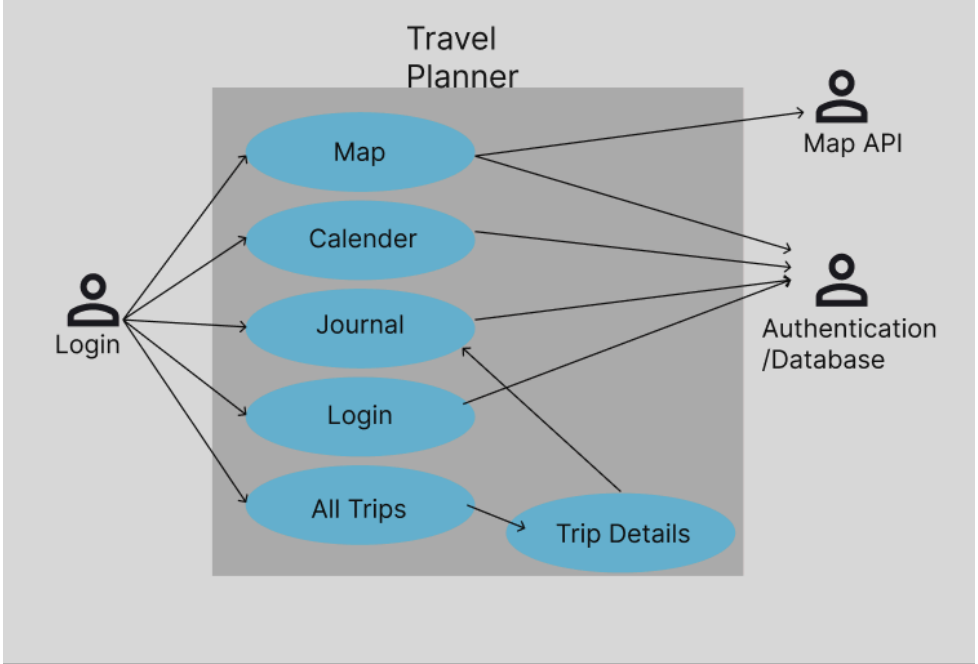
**Giorgio Schepis:** I initially worked on the routing for the pages making sure the nav bar was functioning and that all the buttons route to the correct spots. The next part of my role was creating a functioning All Trips page which would get the created trips from the database and display them on the All Trips screen. Then added the button for adding events onto a trip which allowed me to make individual events onto a trip and add that to the database of events. Then the delete Trips button was added which would mainly remove the trip from the screen and database. Then I loosely worked on the styling and format of the trips on the screen and the overlays for adding information. Lastly I worked on deploying the page through render and troubleshooting.




---

- # Use Case Diagram:
  - You need to include a use case diagram for your project. You can build on the use case diagram you created in the proposal. If you built a complete use case diagram for the proposal, you can include it as is.



---

- # Wireframes:
    - You must include all the wireframes that were created for the project. It is expected that you have at least one for each page. They can be photographs of hand-drawn images.

https://www.figma.com/design/jqb0HuF9wluee6ZAFTxls8/Wireframe?node-id=0-1&p=f&t=4Wx2Kz5lyxj54cvI-0

---

- # Test results:
    - Database connection successful
    - 2025-04-22 10:03:15 web-1 |  ✓ negative: GET /api/destinations without logging in should return 401
    - 2025-04-22 10:03:15 web-1 |  Server!
    - 2025-04-22 10:03:15 web-1 |    ✓ Returns the default welcome message
    - 2025-04-22 10:03:15 web-1 | users_to_destinations table check completed
    - 2025-04-22 10:03:15 web-1 |
    - 2025-04-22 10:03:15 web-1 |  Testing Add User API
    - 2025-04-22 10:03:15 web-1 | trip_name column check completed
    - 2025-04-22 10:03:15 web-1 | Registration attempt with data: {
    - 2025-04-22 10:03:15 web-1 |  username: 'test_user_1745337795682',
    - 2025-04-22 10:03:15 web-1 |  passwordProvided: true,
    - 2025-04-22 10:03:15 web-1 |  test: true
    - 2025-04-22 10:03:15 web-1 | }
    - 2025-04-22 10:03:15 web-1 | Checking if username exists...
    - 2025-04-22 10:03:15 web-1 | Hashing password...
    - 2025-04-22 10:03:15 web-1 | Password hashed successfully
    - 2025-04-22 10:03:15 web-1 | Inserting new user...
    - 2025-04-22 10:03:15 web-1 | User registered successfully: test_user_1745337795682
    - 2025-04-22 10:03:15 web-1 |    ✓ positive : /register successfully (74ms)
    - 2025-04-22 10:03:15 web-1 | Registration attempt with data: { username: undefined, passwordProvided: false, test: true }
    - 2025-04-22 10:03:15 web-1 | Missing username or password
    - 2025-04-22 10:03:15 web-1 |    ✓ negative : /register with missing username or password
    - 2025-04-22 10:03:15 web-1 |
    - 2025-04-22 10:03:15 web-1 |  Testing Redirect
    - 2025-04-22 10:03:15 web-1 |    ✓ GET / should redirect to /login with 302 HTTP status code
    - 2025-04-22 10:03:15 web-1 |
    - 2025-04-22 10:03:15 web-1 |  Testing Render
    - 2025-04-22 10:03:15 web-1 |    ✓ GET /login should render login page with HTML response
    - 2025-04-22 10:03:15 web-1 |
    - 2025-04-22 10:03:15 web-1 |  Testing /api/destinations endpoint
    - 2025-04-22 10:03:15 web-1 | Registration attempt with data: {
    - 2025-04-22 10:03:15 web-1 |  username: 'dest_user_1745337795682',
    - 2025-04-22 10:03:15 web-1 |  passwordProvided: true,
    - 2025-04-22 10:03:15 web-1 |  test: true
    - 2025-04-22 10:03:15 web-1 | }
    - 2025-04-22 10:03:15 web-1 | Checking if username exists...
    - 2025-04-22 10:03:15 web-1 | Hashing password...
    - 2025-04-22 10:03:15 web-1 | Password hashed successfully
    - 2025-04-22 10:03:15 web-1 | Inserting new user...
    - 2025-04-22 10:03:15 web-1 | User registered successfully: dest_user_1745337795682
    - 2025-04-22 10:03:15 web-1 | [GET /api/destinations] Fetching destinations for user: dest_user_1745337795682
    - 2025-04-22 10:03:15 web-1 | Checking if required tables exist...
    - 2025-04-22 10:03:15 web-1 | users_to_destinations table check completed
    - 2025-04-22 10:03:15 web-1 | trip_name column check completed
    - 2025-04-22 10:03:15 web-1 | [GET /api/destinations] Retrieved 0 destinations for user dest_user_1745337795682
    - 2025-04-22 10:03:15 web-1 |    ✓ positive: GET /api/destinations after logging in should return an array (66ms)
    - 2025-04-22 10:03:15 web-1 |
    - 2025-04-22 10:03:15 web-1 |  Testing /logout functionality

```
2025-04-22 10:03:15 web-1  | Registration attempt with data: {
2025-04-22 10:03:15 web-1  |   username: 'logout_user_1745337795683',
2025-04-22 10:03:15 web-1  |   passwordProvided: true,
2025-04-22 10:03:15 web-1  |   test: true
2025-04-22 10:03:15 web-1  | }
2025-04-22 10:03:15 web-1  | Checking if username exists...
2025-04-22 10:03:15 web-1  | Hashing password...
2025-04-22 10:03:16 web-1  | Password hashed successfully
2025-04-22 10:03:16 web-1  | Inserting new user...
2025-04-22 10:03:16 web-1  | User registered successfully: logout_user_1745337795683
2025-04-22 10:03:16 web-1  |     ✓ positive: should logout and redirect to login with a message (67ms)
2025-04-22 10:03:16 web-1  |     ✓ negative: calling /logout without session should still redirect to login
2025-04-22 10:03:16 web-1  |
2025-04-22 10:03:16 web-1  |
2025-04-22 10:03:16 web-1  |   9 passing (393ms)
```

During the testing phase, the application was run in a Dockerized local development environment. The database connection was established successfully, and test cases were executed using the server.spec.js file. Below is a summary of the test outcomes and key observations:

- **Database Connection:** Successfully connected, confirming readiness for test execution.
- **User Registration:**
  - Positive test: A user with valid credentials was registered successfully.
  - Negative test: Registration attempt with missing credentials correctly returned a validation error.
- **API Access (Unauthorized):**
  - GET /api/destinations without logging in returned a 401 Unauthorized response as expected.
- **API Access (Authorized):**
  - After logging in, the same endpoint returned a valid (empty) array of destinations for the test user.
- **Redirect Functionality:**
  - Visiting the root (/) redirected to /login with a 302 status code.
- **Login Page Rendering:**
  - GET /login correctly rendered the login page with a valid HTML response.
- **Logout Functionality:**
  - Logging out with an active session redirected the user to the login page with a success message.
  - Calling /logout without a valid session is still redirected correctly.

All 9 test cases passed. Core functionality for user registration, login, logout, and API endpoint security behaved as expected. The tests confirmed that both positive and negative edge cases are handled correctly. No unexpected errors or crashes occurred during the testing process.

---

- # Deployment:

  We deployed our Travel Planner app on Render by creating an account and connecting it to our GitHub repository. We set up a managed PostgreSQL database and configured environment variables for database connection and session management. Our Node.js web service was set up with npm install as the build command and node index.js as the start command. We initialized the database manually using a local init_db.sh script. The app was deployed successfully and is live at: https://travel-planner-3sgn.onrender.com/