Group Trip Planner and Expense Tracker

CSC 680 Spring 2025 - Final Project

Yash Pachori, Majd Alnajjar, Thomas Bercasio, and Michelle Nguyen

Project Overview:

Our project is an iOS app for planning group trips, managing itineraries, and tracking shared expenses so that people can spend more time enjoying their trips and less time worrying about expenses and timing. We had three must-have features: multi-user sign in with a single “trip” group code/invite, an itinerary screen showing date-by-date plans, and expense tracking to keep tabs on who paid for what. If we are able to successfully implement all of those, we were thinking of implementing these nice-to-have features: automatic expense splitting and settlement calculations (taking inspiration from Splitwise), MapKit integration to list all planned spots or loding locations, and a real-time chat or announcement board for group members.

Technical Overview:

This app is an itinerary and expense manager app built using the concepts we learned in class. It uses SwiftUI for the front end and Firebase for the database and back end. Making sure that the app is modular, reusable, and maintainable was one of our main priorities during development. The primary goal of this app is the creation and management of events and itineraries, as well as the management of expenses that were incurred during an event. TripManager.swift is the central controller for trip data. It manages state across the app using the @Published variable. It stores things such as a list of events, expense data, itinerary items, locations, and members. Events can be created or edited using AddEventView.swift and EditEventView.swift, which contain the UI for adding and editing events. From there the events get added to TripManager.swift. ItineraryView.swift displays a day-by-day schedule of a specific

trip or event using a calendar-style and list UI to track activities. The management of expenses is handled in ExpensesView.swift and SettlePaymentView.swift. These provide an expense breakdown per-user and splits event costs across attendees. The settling logic allows tracking who owes what to whom. Outside of the core features of this app, user authentication is handled in LoginPage.swift and RegisterView.swift. DashboardView.swift aggregates trip info to users and displays a summary of events, expenses, itinerary, and locations. A map is integrated into LocationView.swift where users can view the location of events.

Summary of Work Completed:

We first started with getting the database set up as using Firebase was something new to us and we figured it would cause the most trouble for us. Event and Expense tables were first added and we made sure that we can pull stuff from the database before we got started on working on the front end. From there, we began work on our P1 features. We had to modify them a bit and drop a few P1s due to time constraints, and we narrowed them down to a sign-in functionality, an itinerary screen, and expense tracking.

Development Environment:

- This project code was written using the Xcode IDE
- Firebase was used to handle user authentication and database management.
- Developed on a Macbooks using MacOS Sequoia 15.4
- Tested on simulated Iphone 16.

Assumptions Made:

One of the biggest assumptions we made was thinking that developing this project with only 2 real Macs and 2 virtual machines would be easy. 2 of our group members did not have Macbooks and were not able to acquire one so they decided to set up a virtual machine running macOS Sequoia. We were aware of the difficulties that would come from that such as how slow the VM would be but we figured that it's something we could work around. However it turned out that one of the group members was unable to get the VM to work despite numerous attempts, and while the other one did get it to work, it was incredibly slow and was very difficult to develop on. But despite that we found workarounds. The group member that couldn't get the VM to work would meet up in person with the members that had a Mac and would help with writing the code by writing the logic and algorithm. They also worked on the database since Firebase could be accessed through a browser and also helped with debugging. The others got to work on completing the necessary tasks.

Another assumption that we made was that we would have enough time to get the project done properly. We started working on this in March and got some work done early but we had to pause to focus on work from other classes. We would periodically do some work for this project but didn't really focus on it until a little later into the semester. When we did finally dedicate more attention to this project, we were running out of time and getting close to the due date, but we did manage to get a lot of work done in that time.

Class Descriptions:

- **TripManager:** This class acts as the central controller for the entire application's state. It manages all the core data modules such as events, expenses, and itinerary items. It provides shared functions for modifying data sets and it also handles logic like cost splitting and debt settling. Using the @EnvironmentObject pattern, it serves as a global state using SwiftUI.
- **AddEventView**: This class provides the user interface for creating new events. It includes input fields for event details and interacts with TripManager to save new events. It uses SwiftUI form elements like text fields, date pickers, and toggles to gather user input.
- **EditEventView**: This class allows users to edit previously created events. It pre-fills the input fields with existing data and updates the event in TripManager after edits are saved. It shares most of its structure with AddEventView but operates in update mode.
- **ExpensesView**: This class displays a detailed breakdown of trip expenses. It lists who paid, how much they paid, and how costs are distributed among members. It pulls data from TripManager to calculate individual balances and debts.
- **SettlePaymentView**: This class handles the user interface and logic for settling shared expenses. It shows a summary of who owes whom and allows users to clear debts. It updates the shared expense data in TripManager upon completion.
- **ItineraryView**: This class provides a visual schedule of trip activities grouped by date. It organizes events from TripManager in a structured view so users can easily track what's planned for each day of the trip.

- **LocationView**: This class allows users to input and manage trip locations. It may include a list or a simple form for entering location names. These locations are used when creating or editing events.
- **LoginPage**: This class provides the user interface for user login. It collects email and password input and interacts with the authentication backend (likely Firebase) to validate credentials. It navigates to the main app view on success.
- **RegisterView**: This class allows new users to register an account. It gathers user credentials and communicates with the backend to create a new user entry. Upon success, it redirects users to the main dashboard or login screen.
- **DashboardView**: This class serves as the main summary screen after login. It shows key trip metrics, such as upcoming events and expense summaries, and acts as a hub linking to the itinerary, expenses, and event creation screens.
- **MainView**: This is the main page of the app that you see once you start it. You can login/register through there.

Database Overview:

- **Event:** This collection represents an individual trip activity or planned event. It stores information such as event name, description, date, cost, location, and list of attendees. It's used to populate the itinerary and expense calculations.
- **Expense**: This collection models a financial transaction within a trip. It stores the payer's name, the amount paid, a description, and the list of users the cost is split with. It is used in the app's cost-sharing logic to determine debts and balances.

- **Attendees:** This collection represents a user that will be attending the trip. It stores a user's name. It's used to show the attendees of a trip/event as well as see who owes what to whom.

Reflection:

Yash: This was certainly a challenging project because this is the first time I have ever coded using Swift. There was a big learning curve and I had to refer back to the recorded lectures often in order to clear up confusion. But I feel like my team was very supportive and helpful and we were able to get a lot of work done. One thing I wish I did was try and find other ways I could have procured a Macbook because I was stuck with a VM that barely even worked and I ended up having to help my group mates with their code and use online Swift compilers to test code. But I am glad that I was still able to be of some help to my team and I'm glad we were able to have something to present in class. I did a little bit of work on the database and I wrote some of the files for back end integration, so I'm happy that I did something. Thomas and Michelle were definitely the ones that carried this project and I think they deserve the most props.

Thomas: This project presented a lot of challenges logistically as we had to work around the fact that only Michelle and I had Macbooks to work on to do all the work for this project. Additionally, we all had very sporadic times we could meet up and actually work on the project over the course of the semester which presented its own challenges when it came to scheduling and overall productivity. The stars aligned in the last couple weeks of school for us to work on this project, and we gave it our best effort. Given the circumstances I think everyone on my team

did as good a job as I could have hoped for, and I think even though the end product wasn't complete we all learned a lot from this class about software development and about the kind of work that goes into creating a mobile app especially. I certainly learned a ton, and I'll most likely continue to work on some more projects in Swift or at the very least in my spare time work on finishing the work left undone for this project.

Michelle: I thought the project was a good learning experience taking everything we learned this semester to create something on our own. I learned a lot with my team and I was grateful that we all worked together on this. Everyone worked well together and met deadlines. I mainly focused on the frontend of our app and designed the UX/UI of our app. It was a lot of work and there were times where I wanted to give up, but with my team's support and helping each other out, we were able to come together and create our app.

Majd: This project was especially challenging for me due to not having access to a Mac, which made working in Xcode nearly impossible. Although I attempted to use a virtual machine, it was too slow to be practical, so I contributed in other ways—primarily in debugging, reviewing code, and helping shape backend logic. I focused on `TripManager.swift`, where I helped with event creation and expense tracking, and I took the lead on fixing a major UI bug involving multiple back buttons by inspecting and refactoring navigation across several views. While my contributions were more targeted, I stayed involved throughout and supported the team wherever possible. This experience taught me how to stay adaptable and find ways to contribute despite technical limitations.

Conclusion:

This Group Trip Planner and Expense Tracker app effectively incorporates many crucial components such as user authentication, itinerary planning, and expense tracking. The application offers a clear and modular structure that enables scalability and ease of maintenance through the careful use of Firebase and SwiftUI. Through cooperation, flexibility, and perseverance, our team was able to overcome scheduling issues and hardware constraints. Whether it was through feature implementation, database setup, frontend design, or backend logic, each member made a significant contribution to the finished product. We were able to demonstrate our grasp of client-side state management and mobile app development by successfully delivering the essential functionality, even though not all of the planned features were fully implemented. Above all, the project was an excellent opportunity to learn about full-stack integration, mobile frameworks, and teamwork.