## Project Name

Atlas Itinerary

## Team Member Names

Gabriel Taylor, Dustin Zook, Sorelle Djuissi, Gabriyel Sorensen

## Abstract

This project proposes a personal travel itinerary planner that helps people design detailed, time-slotted trips without the pressure of booking. After a user creates an account, they can build multiple trip plans, each focusing on a chosen destination. Within a trip, the user can select their preferred lodging, restaurants, activities, and attractions, then schedule them into specific days and times to create a daily flow on their trip.

## Description

The purpose of the Atlas Itinerary is to give people a simple but effective tool to organize their trips without the pressure of booking. Most people begin trip planning with scattered notes, random websites, and fluttering ideas, but have the problem of not knowing how to fit all the pieces of their trip together. A user might know which hotel they want, which restaurants they want to try, and which attractions are a must-see, but without a daily structure, these plans can feel overwhelming or incomplete. Many travelers also face the frustration of forgetting key details, double booking activities, or realizing too late that the timing of their plans doesn’t line up. Atlas Itinerary addresses this by offering a central place where users can design, customize, and save detailed trip schedules that show what they can do and when they can do it. Instead of juggling sticky notes, phone reminders, and separate calendars, users will finally have an all-in-one system to map out their travels.

The usefulness of this project reaches a wide range of users. Casual vacationers can use this system to draft itineraries so that they can balance leisure time with sightseeing, ensuring that trips are enjoyable rather than rushed. Families can plan days that work around meal times, nap schedules, and attractions that appeal to different ages of the family. For example, a family trip to Orlando could include mornings at theme parks, afternoons for rest, and evenings for family dining, all while being outlined in a trip plan. Business professionals, who have very limited free time, can arrange their lodging, meetings, and activities that maximize their productivity while still leaving time for leisure. Even students getting ready for study abroad trips or other types of group travel can use Atlas Itinerary to create their day-by-day plans that can be adjusted as they go. The ability to save multiple trips in one account means the users can store past itineraries for reference, allowing them to refine future plans. This system offers clarity and flexibility, helping the user reduce stress that can come when planning and managing multiple trip details.

There are existing systems for trip planning, but those systems are lacking in important ways. Systems like TipIt and Wanderlog rely on importing reservations and booking confirmations, which makes it good for organizing purchased travel, but it lacks the possibility for brainstorming and having flexibility while planning. While other options like physical notes or note apps have the space for full lists that allow for creativity, but lack time scheduling and visual choices during the planning process. This often leads to lists that look good on paper but fail when trying to match open hours, travel time, and meal breaks. Google Trips used to offer a similar solution to this problem, but was discontinued in 2019, leaving users without a solution or tool for their trip planning. Because of the lack of tools, users use a combination of documents, notebooks, calendars, and others, which leads to confusion during the trip planning. The lack of a streamlined tool shows there is a need for a system that is specifically designed for travel planning.

Atlas Itinerary sets itself apart from competitors by focusing strictly on the trip planning rather than purchasing and booking. Without the need to purchase, users are allowed to create and experiment with multiple trip plans before committing to reservations and bookings. This allows creativity and flexibility, letting users explore what-if scenarios. For example, a user planning a trip to Paris could create one itinerary focusing on museums and landmarks, and another itinerary focusing on food and entertainment. They could compare both itineraries before deciding which one will fit best. One of the main features of the Atlas Itinerary is the use of time slots. Instead of just listing an assortment of attractions, restaurants, hotels, etc., the user decides when an activity or event fits into the day. Lodging is assigned by nights, meals are in breakfast, lunch, or dinner windows, and the attractions and events will have specific time frames set by the users, as this creates a daily timeline that the user can follow. An example plan could be breakfast at a local cafe at 8:00 AM, a museum tour at 9:30 AM, lunch at a quiet restaurant at noon, bus tour at 2:00 PM, free time to shop from 3:00 PM to 6:00 PM, and dinner at 7:00 PM. by giving users a structured layout that matches real time, the system makes it easier to double book, and helps have a more enjoyable trip.

Another strength of Atlas Itinerary is its ability to store and manage multiple trips for each user. Users can revisit past itineraries for inspiration for future trips, duplicate itineraries, or adjust previous itineraries to include new destinations or attractions. This feature is helpful for people who take similar trips, such as family vacations or business trips. If there is a group traveling together, each person can make a trip and compare their itineraries to make a final choice. The comparison feature helps users see how different choices affect the itinerary, making the planning more intentional. For example, if two friends are traveling to Chicago, one might have an itinerary planned around shopping and sightseeing, while the other might have an itinerary focused on museums. Being able to see both plans helps users make decisions that work for everyone in the group.

Because of its added level of flexibility, organization, and time-based structure, Atlas Itinerary offers a clear advantage over other systems. It gives the user a creative way to imagine their trips without the pressure of booking and the freedom to adjust and refine the plans. Atlas Itinerary is a travel planning tool that is not just about where to go, but how to make the most out of each day of the trip.

## Feature List

1. User Accounts: users can sign up, log in, and save their trip itineraries
2. Trip Management: users can create multiple trips. Can also edit, delete, save, and view their trips.
3. Lodging Selection: users can add hotels to a trip and assign the number of nights.
4. Dining options: users can add restaurants and pick the specific meal of the day (breakfast, lunch, dinner).
5. Attractions and Activities: users can add events, attractions, and activities, and choose the time frame for each.
6. Daily Timeline View: trips are displayed in a day-by-day schedule showing the hotels, meals, and activities the user has chosen.
7. Mobile Access: The system will be accessible on mobile, so users can use all the features on the go.

A list of features that will be completed if there is time

1. Trip Duplication: users can duplicate a previous trip to modify it as a new trip.
2. Comparison View: users can compare two itineraries side by side.
3. Drag and Drop: users can rearrange items in their itinerary by dragging them into new slots.

A list of features you would like to implement, but cannot complete this semester.

1. Interactive Map: users can view locations of hotels, restaurants, and attractions on an interactive map.
2. Plan Sharing: users can share their itinerary with other users and contribute to the plans.
3. Suggested Itineraries: recommend preset itineraries based on the user's trip destination.

## Initial Set of Technologies

## Platform: Web

## Browser: Firefox

## Operating System: Windows

## IDE: IntelliJ

## Preferred IDE for developing in Java.

## Languages: Java

## For developing backend services for our Web application.

## HTML, CSS, JavaScript

## Necessary for designing web pages and understanding how users will interact with the project.

## 3rd Party Framework: Spring Boot

## Helps with building backend applications in Java.

## Server Software: TBD

## Communication Software: Discord

## Allows for communication outside of class.

## AI: not to use for coding, but to ask basic questions

## Server Information

## Firebase.

## Data Sources

## TripAdvisor Content API <https://www.tripadvisor.com/developers>

## Google Maps platform API <https://developers.google.com/maps/apis-by-platform>

## OpenStreetMap API <https://www.openstreetmap.org/about>

These API’s will allow for access to data such as places, locations, and attractions with descriptions and photos.

## Team Members’ Backgrounds.

* Gabriel Taylor: Background in Computer Systems. Languages like C#, C++, JavaScript, Python, and SQL, as well as a base knowledge of other programs like Racket and Haskell. Used to generating base ideas for endeavors and helping to push projects along.
* Dustin Zook: Experience with Java, JavaFX, HTML, CSS, bootstrap, Javascript, API’s, JSON, and Spring Boot. Databases I have done work in are MySQL, Microsoft SQL Server, and Azure.
* Gabriyel Sorensen: Experience with Java, JavaFX, C#, HTML, CSS, JavaScript, Vue, Python. Used Microsoft SQL.
* Sorelle Djuissi: Experience with Java, JavaFX, HTML, CSS, JavaScript, Python, MySQL, Microsoft SQL Server, JSON, with additional exposure to specialized paradigms such as Prolog (logic programming), Haskell (functional programming), and Racket (Lisp family).

## Dependencies, Limitations, and Risks

* A potential limitation is the need for data on various locations. Through the use of API’s like TripAdvisor Content API or Google Maps Platform API, location data, such as reviews, photos, descriptions, etc., can be acquired. Many of these API’s have a limit on how many calls can be made before they start costing money. For example, according to TripAdvisor Content API, the first 5000 calls made every month are free, but any calls exceeding that limit will be charged to the billing account provided when signing up. This may be a limitation if this project is to remain free
  + The team can handle this issue either by limiting traffic on the web application, paying money in case these limits are exceeded, or by searching for alternatives that are free to use.
* A potential risk is the use of Spring Boot. The majority of team members don’t have much experience with this technology, so using Spring Boot might be risky, as members who aren’t experienced with this technology will likely need to spend additional time becoming familiar with it.
  + The team should be able to handle this issue by reading online documentation, searching for tutorials, etc. Along with this, all members are familiar with Java, so learning Spring Boot in order to work on this project should be feasible.
* By allowing users to create accounts, be able to log in, save trip itineraries, make edits to them, and delete itineraries, the project will depend on a server and database that can handle user authentication, storing account information, and keeping track of itineraries that are saved, edited, deleted, etc. In order for the project to have all of this functionality, the server-side code will need to be implemented before any of these functions can be developed and tested.
  + The team should be able to handle this by splitting the work evenly between frontend and backend functionality. While some members work to get server code implemented so users can create accounts and save itineraries, other members can work on designing web pages and creating the interface users interact with in order to use the features the project has.
* The project depends on the data that can be acquired from API’s like Google Maps Platform API, TripAdvisor Content API, etc. before primary features of the project can be implemented, such as users being able to search for locations and add them to a travel itinerary, access to the content these API’s provide along with integrating them with the project will need to be done.
  + The team can handle this issue by working quickly to acquire API keys and integrating the API’s into the project so the content and features from the API’s can be utilized.

## Timeline

1. Week 2 (Sep 1-5):
   1. Draft and submit the project proposal (all members)
2. Week 3 (Sep 8-12):
   1. define detailed requirements such as core features and “if” time features (all members)
   2. Create use cases and user stories(all members)
3. Week 4 (Sep 15-19):
   1. System design/ how features fit together(all members)
   2. Create initial wireframes or mockups for UI to show trip management and itinerary view( Sorelle)
   3. Design structure for database(Dustin & Taylor)
   4. Document entities and relationships(Dustin & Taylor)
4. Week 5 (Sep 22-26):
   1. Set up project structure(all members)
   2. Set up database on Firebase(Dustin)
   3. Educating team members on how Firebase works and aiding in database setup by pitching ideas for database information(Taylor)
   4. Begin basic account creation and login system (front-end: design and style login UI) (Sorensen)
   5. Build simple API calls in a test to fetch hotels or restaurants by location(Dustin)
5. Week 6 (Sep 29-Oct 3):
   1. Implement trip management-CRUD operations (Sorensen)
   2. Develop trip management UI components (create, update, delete trip pages)
   3. Implement user storage for the database(Dustin)
6. Week 7 (Oct 6-10):
   1. Implement the lodging selection of adding hotels and assigning nights
   2. Implement dining options of selecting restaurants and the meal assignment (Sorensen)
   3. Integrate API’s into lodging and dining features(Dustin)
   4. Build lodging/dining UI screens and integrate with API results (Sorelle)
7. Week 8 (Oct 13-17):
   1. Implement attractions and activities features with time slots
   2. Add trip data handling, such as trip creation, lodging nights, restaurants, and activities (Sorensen)
   3. Integrate API for attractions and activities lookup(Dustin)
   4. Develop attractions/activities UI and calendar input fields (Sorelle)
8. Week 9 (Oct 20-24):
   1. Implement a timeline view day by day schedule
   2. Connect the timeline view to the database records (Sorensen)
   3. Connect API data to the timeline view(Dustin)
   4. Build interactive timeline/day-by-day itinerary view UI (Sorelle)
9. Week 10 (Oct 27-31):
   1. Review the core features and make adjustments if necessary(all members)
   2. Test database with multiple users and trips(Dustin)
   3. Sorelle: Refine UI/UX consistency (colors, layout, navigation)
10. Week 11 (Nov 3-7):
    1. Implement the extra features if we have time(all members)
    2. Optimize API calls to reduce duplicate calls(Dustin)
    3. Assist with UI integration for extra features (Sorelle)
11. Week 12 (Nov 10-14):
    1. Conduct full system tests(all members)
    2. Write a technical report(all members)
    3. Support usability testing and document UI design choices (Sorelle)
12. Week 13 (Nov 17-21):
    1. Finalize implementation/make sure all features are implemented and working(all members)
    2. Polish the frontend and user experience (Sorelle)
13. Week 14 (Nov 24-28):
    1. Final round of testing(all members)
14. Week 15 (Dec 1-5):
    1. Final presentation(all members)