TechTravel Itinerary

Project Team: Daouda Mohamed, Senih Tosun, Max Honeycutt, Landon Alison

Demo

GitHub

Table of Contents

5. System – Analysis Perspective

- Identify subsystems – analysis point of view
- System (Tables and Description)
    - Data analysis
        - Data dictionary (Table - Name, Data Type, Description)
    - Process models

6. Project Scrum Report

- Product Backlog (Table / Diagram)
- Sprint Backlog (Table / Diagram)
- Burndown Chart

7. Subsystems

7.1 Subsystem 1

- Initial design and model
- Data dictionary
- If refined (changed over the course of project)
    - Reason for refinement (Pro versus Con)
    - Changes from initial model
    - Refined model analysis
    - Refined design (Diagram and Description)
- Scrum Backlog (Product and Sprint - Link to Section 6)
- Coding
    - Approach (Functional, OOP)
    - Language
- User training
    - Training / User manual (needed for final report)
- Testing

7.2 Subsystem 2

- Initial design and model
    - Illustrate with class, use-case, UML, sequence ..... diagrams
    - Design choices
- Data dictionary
- If refined (changed over the course of project)
    - Reason for refinement (Pro versus Con)
    - Changes from initial model

    - Refined model analysis
    - Refined design (Diagram and Description)
- Scrum Backlog (Product and Sprint - Link to Section 6)
- Coding
    - Approach (Functional, OOP)
    - Language
- User training
    - Training / User manual (needed for final report)
- Testing

7.3 Subsystem 3

- Initial design and model
    - Illustrate with class, use-case, UML, sequence ..... diagrams
    - Design choices
- Data dictionary
- If refined (changed over the course of project)
    - Reason for refinement (Pro versus Con)
    - Changes from initial model
    - Refined model analysis
    - Refined design (Diagram and Description)
- Scrum Backlog (Product and Sprint - Link to Section 6)
- Coding
    - Approach (Functional, OOP)
    - Language
- User training
    - Training / User manual (needed for final report)
- Testing

7.4 Subsystem 4

- Initial design and model
    - Illustrate with class, use-case, UML, sequence ..... diagrams
    - Design choices
- Data dictionary
- If refined (changed over the course of project)
    - Reason for refinement (Pro versus Con)
    - Changes from initial model
    - Refined model analysis
    - Refined design (Diagram and Description)
- Scrum Backlog (Product and Sprint - Link to Section 6)

- Coding
    - Approach (Functional, OOP)
    - Language
- User training
    - Training / User manual (needed for final report)
- Testing

7.5 Subsystem 5

- Initial design and model
    - Illustrate with class, use-case, UML, sequence ..... diagrams
    - Design choices
- Data dictionary
- If refined (changed over the course of project)
    - Reason for refinement (Pro versus Con)
    - Changes from initial model
    - Refined model analysis
    - Refined design (Diagram and Description)
- Scrum Backlog (Product and Sprint - Link to Section 6)
- Coding
    - Approach (Functional, OOP)
    - Language
- User training
    - Training / User manual (needed for final report)
- Testing

8. Complete System

- Final software/hardware product
- Source code and demonstration video
- Evaluation by client and instructor
- Team Member Descriptions

1. Project Definition

Why:

- The main purpose of this web application is to allow users to create itineraries that would allow them to ease the process of travel planning. Also, this web application serves as a hub for all accessible travel information which, in theory, should help users find information about their future travel location and facilitate planning their itineraries.

What:

- The goal of this project, as previously stated, is
  a. To ensure that users have an appropriate application that allows them to build an itinerary for their trip/s.
  b. To allow them to find all the information they need to build their desired itinerary which will include the major transportation, the activities, the hotel, and the local transportation options.
  c. To be able to keep track of the travel plans even if there are changes to the initial schedule.
  d. To help the traveler keep track of the things they haven't done yet on the trip by having access to all of the activities they set out to see.
  e. To allow the traveler to share their travel plan with other travelers.
  f. Allow the traveler to update the status of activities.

How:

These goals will be achieved by creating a website/web application that will allow users to do the aforementioned tasks spelled out in our What section. This application will be built in Javascript and use MongoDB as the database. We will do most of our actions using HTML forms and backend functionality.

2. Project Requirements Analysis

Functionality:

1. Major Feature: Itinerary
    a. Component 1: Destination Guide
        i. Detail 1: Provide the available forms of transportation to and from the city and inside the city.
        ii. Detail 2: Ability to record transportation to this destination.
        iii. Detail 3: Ability to add activities.
        iv. Detail 5: Ability to remove activities.
        v. Detail 6: Ability to change transportation if more options are available.
    b. Component 2: Activity Guide
        i. Detail 1: Ability to add an activity to the guide and set its priority by the date you want to do it.
        ii. Detail 2: Ability to organize added activities by priority on the guide.
        iii. Detail 3: Ability to remove activity from the guide.
    c. Component 3: Complete Schedule
        i. Detail 1: Ability to organize the transportation guide and activity guide based on the commute time estimates and how long you plan to stay at a certain place.
        ii. Detail 2: Present this organized data in a calendar format, The calendar format will have a whole trip view, a day view, and a specific activity view.

        iii. Detail 3: To update this Complete Schedule it will path you to the respective change form of the attribute.

    d. Component 4: Standard Functionality

        i. Detail 1: Ability to create an Itinerary.
        ii. Detail 2: Ability to delete an Itinerary.
        iii. Detail 3:Ability to edit an Itinerary.
        iv. Detail 4: Ability to view an Itinerary.
        v. Detail 5: Ability to share an Itinerary via a link.

2. Major Feature: City Search

    a. Component 1: Search Bar

        i. Detail 1: Users can search for a city by name (autofill in the search bar).
        ii. Detail 2: Redirect users to the searched city's detail page where they have all the necessary information about that city.
        iii. Detail 3: Users have the option to add the searched city to their itinerary.

    b. Component 2: City Detail Page

        i. Detail 1: Top attractions and things to do are displayed using an outside service embedded into the detail page.
        ii. Detail 2: Top-rated restaurants, bars, and cafes are displayed using an outside service embedded into the detail page.
        iii. Detail 3: Display a map of the city where users can view and zoom in/out of the map to get a sense of the city.
        iv. Detail 4: Another tab where the user clicks to get redirected to the hotel deals.

    c. Component 3: Hotel Options/Recommendations
        i. Detail 1: Show hotels for the city searched.
        ii. Detail 2: Show hotel pricing from different companies.
        iii. Detail 3: Search hotels/stays on a certain date using an outside service like Google Hotels.
3. Major Feature: Travel Transportation Tracking
    a. Component 1: Flight Tracker
        i. Detail 1: Allow users to find their flight via flight number and track it in real-time.
    b. Component 2: Train Tracker
        i. Detail 1: Allow users to find their train via train number and track it in real-time.
4. Major Feature: User Profile
    a. Component 1: Trips
        i. Detail 1: Allows users to view trip history.
        ii. Detail 2: Allow users to see future history.
        iii. Detail 3: Allow users to plan for future trips.
    b. Component 2: Itinerary
        i. Detail 1: Allow access to their personal itinerary page which allows them to create, delete, update, view, and share itineraries.
    c. Component 3: Personal Information/Credentials
        i. Detail 1: Allows users to input their full name and change their full name if needed.

ii. Detail 2: Allows users to access their username (which will primarily be their email address) and change it if needed.
iii. Detail 3: Allow users to change their password if they have forgotten it or would like a new password.
iv. Detail 4: Save email addresses and passwords in a database with a unique profile ID.
v. Detail 5: Allow users to store their home city as a starting point within their itineraries (optional).

Usability:

User interface:

- Materialize CSS
- EJS Templating

Performance:

- Express
- Node.js

System:

Hardware:

- Windows Devices
- MacOS Devices
- Linux Devices
- Android Devices
- iOS Devices
- Any device with compatible up-to-date web browsing functionality

Software:

- VSCode
- MongoDB Cloud
- GitHub

Database:

- MongoDB

Security:

- The server has a SSL certificate and runs on HTTPS to ensure encryption
- Password creation has a limitation that password creations must be 8 characters long
- Users must be logged in to access itinerary functionality
- Secondary users can only view the itinerary of other users if they have the given link from the Primary user

3. Project Specification

Focus / Domain

- ➔ Focus
    - ◆ Travel Planning and Organizing
- ➔ Domain
    - ◆ Travel

Libraries / Frameworks / Development Environment

- ➔ Frontend Tools
    - ◆ EJS Templating Engine: Templating engine for generating HTML that is dynamic, this will allow transfer of data between the backend and frontend.
    - ◆ Materialize CSS: Frontend framework for designing responsive and modern UI components.
- ➔ Backend Tools
    - ◆ Express: Node.js framework for building the backend server and routes etc.
    - ◆ Node.js: JavaScript runtime for server-side development.
    - ◆ MongoDB: NoSQL database for storing and managing data.
- ➔ APIs
    - ◆ Google Maps
    - ◆ Google Calendar
    - ◆ Flight Tracker
    - ◆ Amtracker v3

Platform

- ➔ Platforms
    - ◆ PC and Mobile

Genre

- ➔ Genre
    - ◆ Travel Planning Website/Web Application

Teamwork division

Workload Distribution:

- ➔ Daouda: Responsible for User Profile Design and Implementation, with involvement in shaping the overall design.
- ➔ Landon: Flight & Train Tracking System.
- ➔ Senih: Responsible for implementing Search Functionality, setting up user authentication, and contributing to key database CRUD operations.
- ➔ Max: Responsible for designing the Itinerary Page and overseeing CRUD operations within the create and edit pages of the itinerary.

4. System - Design Perspective

Identifiable Subsystems

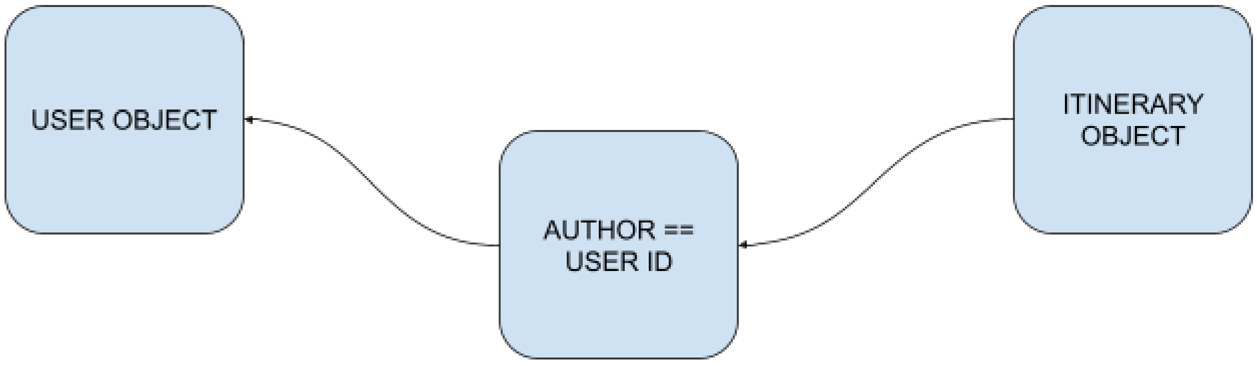
- ➔ Itinerary Hub
- ➔ City Search
- ➔ User Profile
- ➔ Transit Tracker

Sub-System Communication

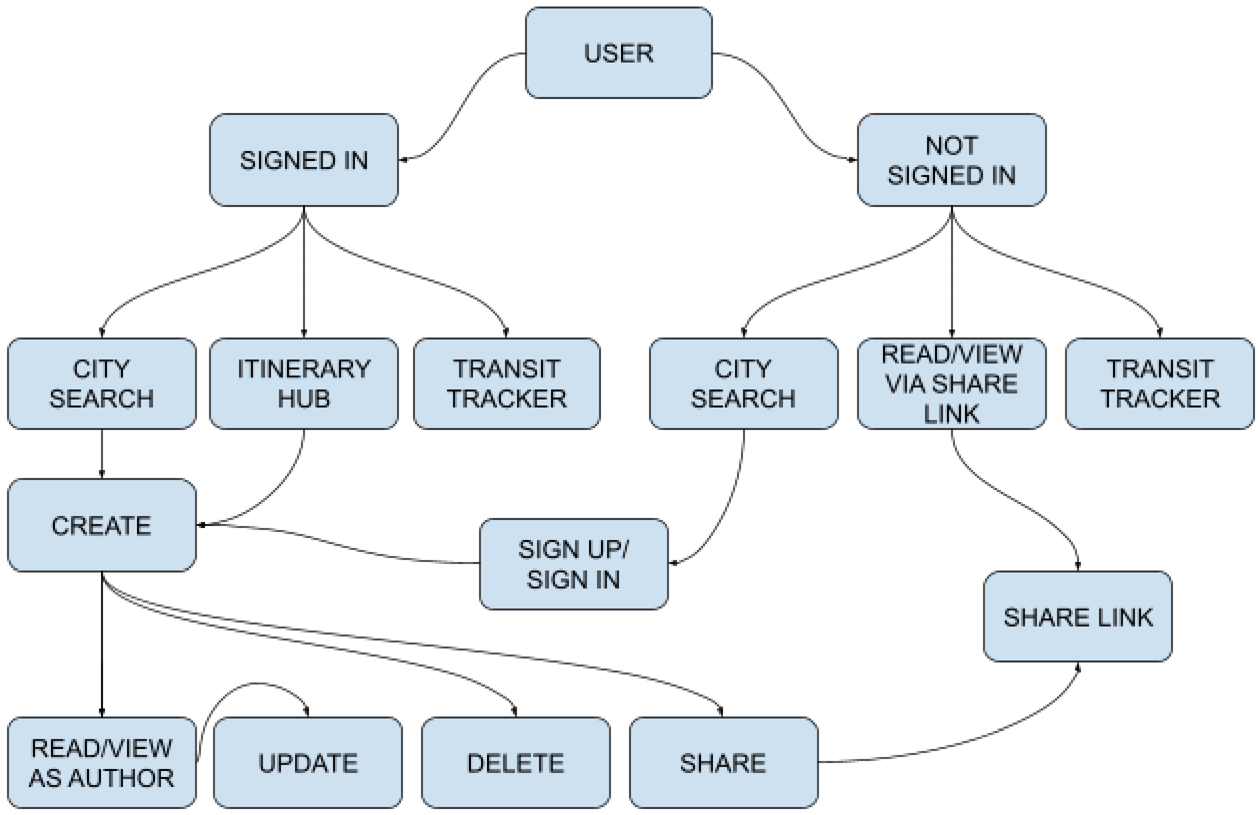
- ➔ All subsystems are accessible from every other page via the taskbar.
- ➔ The User Profile contains a Trip history that allows the user to view their status whether they be past, ongoing, or future trips.

Entity Relationship Model

In the diagram below depicting the relationship between User objects and Itinerary objects specifically the fact that the itinerary objects have a foreign key relationship via their author data type which is the user ID from the User objects. This relationship is many itineraries to one user.



Overall Operation - System Model



5. System - Analysis Perspective

Identifiable Subsystems:

- Itinerary Hub
- City Search
- User Profile
- Transit Tracker

System:

- Data Dictionary
  - Collection Name: `Users`
    - ‘_id’: ObjectId
    - `username`: String (Unique)
    - `password`: String (Encrypted)

        - `displayName`: String
        - `homeCity`: String
    - Collection Name `itinerary`
        - ‘_id’: ObjectId
        - ‘author’: ObjectId(User._id)
        - `itineraryName`: String
        - `startingCity`: String
        - `startDate`: Date
        - `endDate`: Date
        - Destination: Array of Destination objects
    - Collection Name `destination`
        - `name`: String
        - `transportation`: String
        - `hotel`: String
        - Activities: Array of Activity objects
    - Collection name `activity`
        - `activityName`: String
        - `activityDate`: Date
        - `transportation`: String

6. Project Scrum Report

- Week 1: Setting up the codebase, team members familiarizing themselves with the languages/technologies.
- Week 2: Creating the necessary database tables, and the respective folders in the repo for our respective tasks.
- Week 3: Barebones template of the website should be ready and team members should be working on their assigned tasks.
- Week 4: Team members should be making progress on their tasks towards completion.
- Week 5: Development & Design should be finished towards the end of the week, and small bug fixes should be undertaken.
- Week 6: Testing edge cases and finding out if any bugs are present.

7. Subsystems

7.1 Subsystem: User Authentication and Registration (Senih)

Initial Design and Model:

Objective: To establish a robust and secure user registration and authentication system for the application.

Components:
1. User Registration Form
2. Database Schema for User Data Storage
3. Passport.js Configuration for Authentication

Data Dictionary

User Registration Data:
- Username: Unique identifier for each user.
- Password: Securely encrypted password for user authentication.
- Display Name: Name displayed publicly within the application.
- Default Location: Home city or default location for user activities

Database Schema:
- Collection Name: `Users`
- `username`: String (Unique)
- `password`: String (Encrypted)
- `display_name`: String
- `default_location`: String

Refinements:

User Registration Form:
- Validate input fields to ensure data integrity.
- Implement frontend feedback for users during registration.

Database Schema:
- Consider additional fields for future scalability or user preferences.

Passport.js Configuration:
- Explore additional authentication strategies for enhanced security (e.g., JWT).

Scrum Backlog:

User Stories:
- As a user, I want to register with a unique username and password.
- As a user, I want to set my display name and default location during registration.
- As a user, I want my password securely encrypted and stored.

Tasks:
- Implement frontend registration form.
- Configure backend routes for user registration.
- Integrate passport.js for authentication.
- Test registration and authentication flows.

Coding:

Frontend:
- Develop user registration form using HTML/CSS/JavaScript with EJS template.
- Implement client-side validation for input fields.

Backend:
- Set up Express.js routes for user registration.
- Integrate passport.js for authentication middleware.
- Implement MongoDB queries for user data storage.

Testing:
- Input validation testing and repeatedly trying invalid inputs to make sure the system is preventing them.
- Conduct integration testing to ensure proper data flow.
- Perform security testing to identify vulnerabilities.

User Training:

Documentation:
- Provide user documentation on how to register and log in.
- Explain password security best practices to users.

Onboarding:
- Offer interactive tutorials or guides within a demo YouTube video.
- Provide support channels for users facing registration/authentication issues.

7.2 Subsystem: Search Functionality (Senih)

Initial Design and Model:

Objective: To provide users with a seamless search experience to find cities and access detailed information about them.

Components:

1. Search Bar Implementation
2. Auto-fill Functionality
3. City Detail Page
4. Itinerary Creation Page

Data Dictionary:

Search Data:
- Input: Text input from users for city search.
- Auto-fill Suggestions: List of cities to assist users in selecting desired cities.

City Detail Data:
- Guide: Information about attractions, landmarks, and activities in the city through Wikivoyage.
- Foods & Drinks: Recommendations for restaurants, cafes, and bars.
- Map: Interactive map displaying the city's layout and points of interest.
- Hotel Deals: Offers and information on accommodations in the city.

Itinerary Data:
- Selected Cities: Selected cities added to the itinerary by the user.
- Additional Details: Other fields in the itinerary creation will be filled by the user, which is discussed in the itinerary subsystem below.

Refinements:

Search Bar:
- Enhance UI/UX for a user-friendly search experience.
- Optimize search functionality for fast and accurate results.

Auto-fill Functionality:
- Implement efficient algorithms for auto-fill suggestions based on user input.
- Ensure compatibility with various browsers and devices.

City Detail Page:
- Design an intuitive layout for easy navigation and information access.
- Incorporate responsive design for mobile and desktop users.

Scrum Backlog:

User Stories:
- As a user, I want to search for cities using a search bar.
- As a user, I want auto-fill suggestions to help me select cities quickly.
- As a user, I want detailed information about a selected city, including guides, restaurants, maps, and hotels.
- As a user, I want to add a selected city to my itinerary and begin planning my trip.

Tasks:
- Implement frontend search bar with auto-fill functionality.
- Create backend routes for handling city search requests.
- Design and develop city detail page layout and components.

Coding:

Frontend:
- Develop search bar component using HTML/CSS/JavaScript with EJS template for dynamic page display together with the backend route.
- Implement auto-fill functionality using JavaScript and Materialize JS.
- Design city detail page UI HTML/CSS.

Backend:
- Set up Express.js routes for handling city search requests.
- Develop logic to retrieve city data from a JSON file.
- Itinerary endpoints are discussed by Max below, including adding cities to the itinerary.

Testing:
- Repeatedly test the city detail page for successful display of different locations.
- Test city detail page components for proper rendering and functionality.
- Conduct integration testing to ensure seamless flow between search, city detail, and itinerary pages.

User Training:

Documentation:
- YouTube video as a demonstration of usage and the software document.

Onboarding:
- Include tooltips or guided tours within the application to introduce users to search features and itinerary creation in the form of a YouTube video. (demo)

Testing:

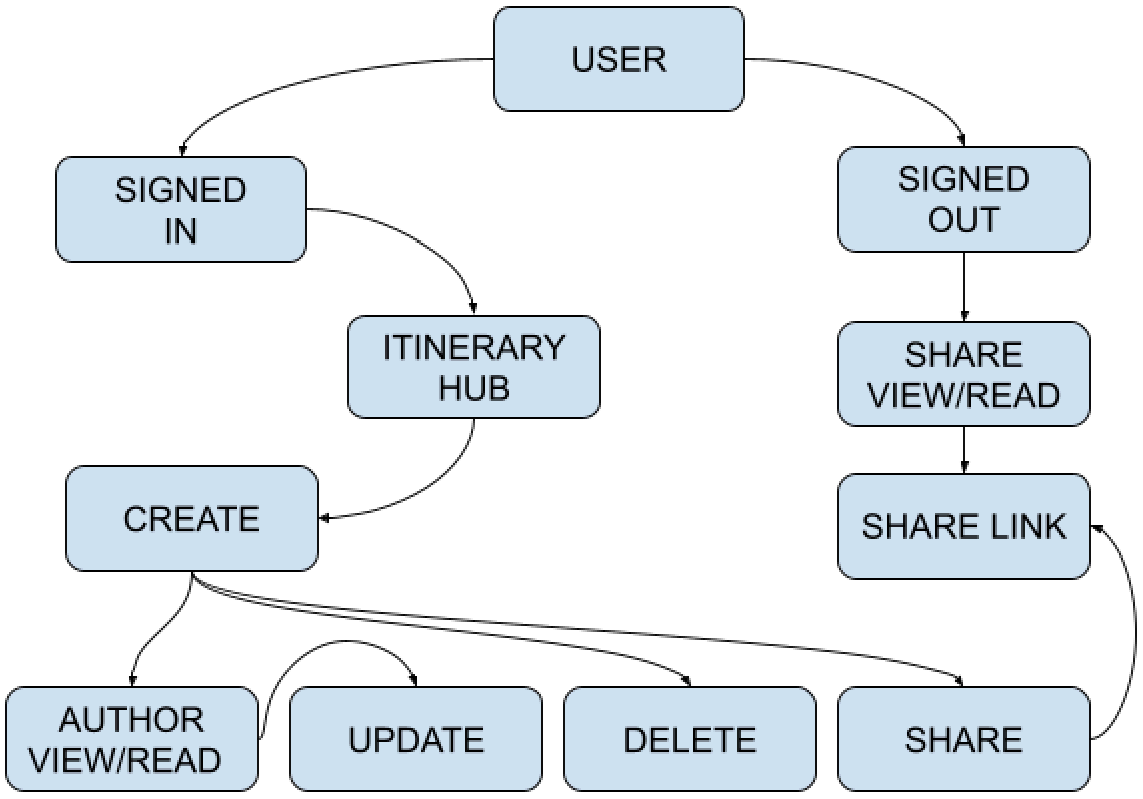
Test Cases:
- Verify search functionality with various input scenarios.
- Test auto-fill suggestions for accuracy and responsiveness.
- Validate city detail page components for correct display of information.
- Ensure itinerary creation page functions properly, allowing users to add cities and input additional details.

Tools:
- Manual testing was done, however, as a refinement item these should be included in unit testing and QA testing.

7.3 Subsystem: Itinerary Hub (Max)

Initial design and model:

The CRUD structure uses EJS form pages to create and update data and use EJS view pages to read itinerary data into two representations of the itinerary a user one and a shared one and produce a share page to test and distribute the share link. The model of this design is described in the following image:



To further explain the model, if the user is signed out/not authenticated cannot access the itinerary hub and they can only view an itinerary via a share link but cannot update or delete that itinerary especially if they are not the creator of the itinerary. If the user is signed in they can access the itinerary hub which will allow them to create an itinerary and view, update, share, and delete afterwards. Once the User has created an itinerary it is stored in the database and is only updateable, shareable, and deletable by that same user.

Data Dictionary:

Itinerary objects contain(_id: ObjectId, author: contains (id which is the id of the user), itineraryName: String, startingCity: String, startDate: Date, endDate: Date, Destinations: Array of Destination Objects.) Destination Objects contain(name: String, transportation: String, hotel: String, Activities: Array of Activity objects, startDate: Date, endDate: Date). Activity objects contain(activityName: String, activityDate: Date, transportation: String).

Refinements:

This subsystem has gone through multiple refinements whether that be changing the data types held within the itinerary or design fixes. To name a few the initial idea about the itinerary was going to be a relational table for the destinations and activities but given the cumbersome nature of an itinerary such as this, we felt it would be better to make one collection to represent all itineraries and make the instances contain all of the data accessible via the id associated with the user id. The UI design has undergone a multitude of color fixes, layering fixes, and scaling fixes.

Scrum Backlog:

The last things I hoped to be able to implement but ran out of time would be the ability to record the accessibility of the transportation and to add the priority functionality to the activities and the biggest I must say would be separating the multi-level view for both the user/shared itineraries into the calendar format, Day and Activity instead of being one whole thing using a list format as the view currently uses. I also wanted to add a confirmation dialog to the delete button instead of the instant delete which was noted at the Jackson Library Presentations.

Coding:

I did all of my coding in VS Code. I created my routes for viewing, updating, sharing, and deleting itineraries. I created all of the .ejs pages for the itinerary and made most of the refinements. I made the CSS for itinerary.css. The itinerary-create.css and itinerary-create.ejs were created by Senih and I used the itinerary-create.ejs as a basis for itinerary-update.ejs using its CSS for consistency

User Training:

This subsystem is pretty simple for the users as once you create an account and access the Itinerary Hub it prompts you to create an itinerary and then once you create one you can update the itinerary by using the edit button and share it using the share button and delete it by using the delete button.

Testing:

This subsystem was tested by actively trying to break the system. I attempted to access the view and edit pages from accounts that don't match the author which led to a backend check to verify the user matches the author. Daouda also noted that you can't simply add some the destinations without also setting up some of the activities, we resolved this by changing the requirements of the itinerary objects.

7.4 Subsystem Summary: Real-Time Multi-Modal Transportation Tracking(Landon)

Subsystem Description:

This subsystem is a comprehensive tracking solution for real-time monitoring of trains and airplanes. Utilizing the APIs from Amtrak (Amtracker) and aviationstack, it provides accurate, live data crucial for travel and logistics coordination.

Functional Capabilities:

1. Data Acquisition: It interfaces with Amtracker and aviationstack APIs to pull live data feeds, ensuring that users have access to the most current information about train and airplane statuses.

2. Data Points for Trains:
   - Geolocation: Current latitude and longitude to track the train's exact position.
   - Route Information: Identifies the train's service name, providing context for the tracking data.
   - Timing: Reports any delays with a timestamp, vital for real-time adjustments.
   - Journey Details: Lists the origin and destination stations, offering a complete view of the train's scheduled path.
   - Operational Status: Indicates if the train is en route, delayed, or at a standstill.

3. Data Points for Airplanes:
   - Flight Identification: Provides the flight number, connecting the data to a specific flight.
   - Airline Details: Names the airline, adding another layer of detail for logistics tracking.
   - Travel Path: Gives the departure and arrival airports, outlining the flight's trajectory.
   - Spatial Data: Shares real-time latitude, longitude, and altitude, pinpointing the airplane's location in the sky.
   - Direction: Notes the plane's compass heading, showing the direction of travel.
   - Flight Status: Updates with the current flight condition, such as in-flight, landed, or delayed.

4. User Interface and Experience:
   - Interactive Dashboard: Users can enter a train or airplane ID to track, and the system dynamically generates a data-rich profile for the selected vehicle.
   - Data Presentation: Information is displayed in a clean, easy-to-read format, with clear indicators for data availability.

5. Error Handling and Data Integrity:

- Fallback Procedures: When certain data points are unavailable from the APIs, the system smartly displays 'Unavailable', ensuring that the user interface remains informative and free of erroneous data.
- Data Precision: All geolocation data is precise to the decimal degree, and altitude is reported in feet, ensuring high fidelity in tracking.

6. Integration and Adaptability:
   - API Integration: Seamlessly incorporates data from disparate API sources, presenting it in a unified interface.
   - Adaptive Design: Capable of scaling with additional data sources or changing API specifications.

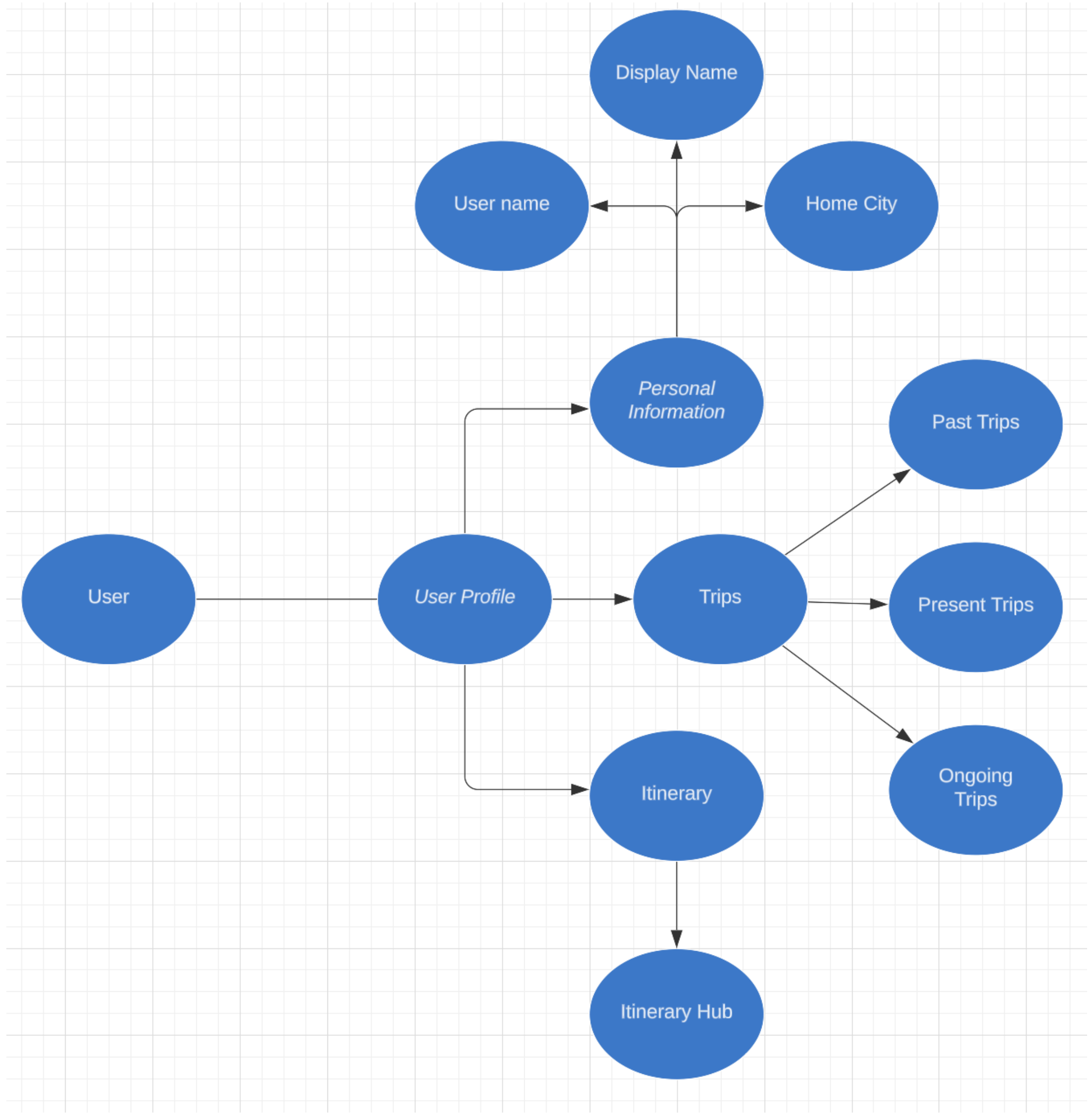
Subsystem Use Cases:

- For passengers and cargo services to monitor and plan for travel with up-to-the-minute information.
- By logistics coordinators to optimize route planning and resource allocation.
- For customer service representatives to provide accurate travel updates and manage expectations during service disruptions.

7.5 Subsystem: User Profile

Initial design and model:

- User profile design consisted in implementing a profile for users that was unique to their own accounts. Users are introduced to their profile page which displays their name and various pages they can access. Personal information allows users to view their full name, user names, and home city and change any of this information if desired. Users can view their future, past, and ongoing trips. They can click on the trip to access their itinerary hub and view their itinerary. Buttons have been added to allow users to access their itinerary with ease also.



Data dictionary:

- Collection Name: `Users`
    - `username`: String (Unique)
    - `password`: String (Encrypted)
    - `displayName`: String
    - `homeCity`: String
- Collection Name `itinerary`
    - `itineraryName`: String (Unique)
    - `startingCity`: String
    - `startDate`: Date
    - `endDate`: Date
- Collection Name `destination`
    - `name`: String

Refinements:

- None

Scrum Backlog:

- There was only one backlog regarding the scrum plan and that was the implementation of the change password. Aside from that, the profile was a fairly easy function to execute.

Coding:

- The coding for the profile page was done within VSCode using Express JS (EJS) and CSS. The EJS was used to write the barebone of the website, javascript was used to implement the backend connection to the database, and CSS was used to design the page. All the files related to this function are personal.ejs, profile.ejs, trips.ejs, trips.css, personal.css, and profile.css.

User Training:

- As for user training, this is a very understandable and straightforward function. You click on a button and it will lead you to where you intend to go within the website. The personal information page has a text box that will hold the value of the current information and be easily edited by clicking into the text box and editing its content. After the content has been edited the user can hit the change button at the bottom of the page and all changes will be changed. All this can be demonstrated in a tutorial video or implementing a help button.

Testing:

- The testing for this function was just test-casing situations that could break the profile page. As of currently, there are no bugs that break the profile aside from screen sizing/ratio, this issue will be fixed through CSS eventually.

8. Complete System

Final Software/Hardware product

- Final Software Product: https://www.techtravelitinerary.com/

Source code and demonstration video

- Source code: https://github.com/D-Moham/490-Project
- Video: https://www.youtube.com/watch?v=hz0YFildu-4&t=4s

Evaluation by Client and Instructor

Team Member Descriptions

- Daouda Mohamed Moumouni: User Profile Design and Implementation, with involvement in shaping the overall design.
- Senih Tosun: Responsible for implementing Search Functionality, setting up user authentication, and contributing to key database CRUD operations.
- Max Honeycutt: Responsible for designing the Itinerary Page and overseeing CRUD operations within the create and edit pages of the itinerary.
- Landon Alison: Flight & Train Tracking System.