Team 5 (Tiles - Social Map Tracker)

1. Daniel Aguilar
   - Student ID: 033021922
   - Email: daniel.aguilar02@student.csulb.edu
2. Jayden Woodall
   - Student ID: 032379215
   - Email: Jayden.woodall01@student.csulb.edu
3. Mischa Waydo
   - Student ID: 031959224
   - Email: mischa.waydo01@student.csulb.edu
4. Edwin Sedano
   - Student ID: 025668238
   - Email: edwin.sedano@student.csulb.edu
5. Nicholas Parra
   - Student ID: 031747116
   - Email: nicholas.parra02@student.csulb.edu
6. Nick Vincent Quinto
   - Student ID: 029876117
   - Email: nickvincent.quinto01@student.csulb.edu

Preface

Version	Date	Changes
1.0	09/14/2024	Initial Version
1.1	10/7/2024	Added Authentication
1.2	11/20/2024	Implemented Database

Purpose

This document serves as an all-encompassing guide for the development, features, and use cases of the web-app “Tiles - Social Map Tracker.”

Team 5 (Tiles - Social Map Tracker)

Audience

This document is intended for the following stakeholders: project overseer, developers, testers, users, as well as all other individuals involved in the project lifecycle.

Introduction

Project Overview

“Tiles” is a social map tracker with a focus on keeping friends and family connected through a visualized travel map. Its primary functions involve recording new areas that a member of a group has visited and sharing them visually with the rest of the group, providing activity logs to keep group members updated with new events within the group, and providing insightful map illustrations to document a user’s travel.

Project Goals

- Keep groups of family and friends connected during travel.
- Provide real-time “location unlocked” event updates.
- Generate insightful map visuals based on a group’s travel history.

Glossary

- **Tile:** A predetermined portion of the Earth’s map in the shape of a hexagon, typically the size of a small city.
- **Location unlocked event:** A real-time update to the list of unlocked tiles each user has.
- **Web-App:** A JavaScript enabled website which utilizes programming to dynamically perform certain functions.

User Requirements and Use Cases

User Stories

1. As a registered user, I want to log in securely so that my location history is private.
2. As a group member, I want to receive real-time updates on the tiles that people in my group are unlocking so I can stay updated on where my friends are going.

3. As a group member, I want to join a group with my friends and see the location history of each of my friends and myself as we travel so that we can stay connected despite being away from each other.
4. As a group member, I want to track my location history separately from that of my group so that I can compare my progress to my group's progress.
5. As a hiker, I want to manually add locations to my location history so that I can track locations I visited before downloading the app.
6. As a group owner, I want to have control over who can/cannot join my group so that I can limit membership to certain people.
7. As a group owner, I want to have control over whether someone can stay in the group.
8. As a group member, I want to be able to leave a group whenever I want to so that I can join a different group more suited to my interests.
9. As a nature enthusiast, I want to view the specific location history of one of the other members of my group so that I can discuss our similar travel patterns.
10. As a travel blogger, I want to have a clear layout of locations I have visited before so that I can make a list of places I want to visit next

Use Case: Joining a Group

Identifier	UC-2 Joining a Group with Friends
Purpose	Update user's and friend's groups with the newly created group
Requirements	User Story #2
Development Risks	None
Pre-conditions	Each member has created an account and has not maxed out their total joinable groups. User is on Groups page.
Post-conditions	Each member is added to a group and has their groups list and map updated. User is on Groups page.

Table 1: Typical Course of Action

Seq#	Actor's Action	System's Response
1	User choose an option to join a group	System displays groups to join in and an option to create a new group
2	User chose an existing group	System adds the user to the group

3	User invites friend to the group	System sends invite notification to invited friends to join the group

Table 2: Alternative Course of Action

Seq#	Actor's Action	System's Response
1	User tries to join a group, but already has reached the maximum groups joined.	System sends an error message saying that the user already joined the maximum amount of groups
2	User tried to leave in an existing group to have a space for a new group	System removes the user from the existing group and updates the list of groups the user joined in.
3	User successfully joined a new group	System updates the user's group and location history

Table 3: Exceptional Course of Action

Seq#	Actor's Action	System's Response
1	User tries to join a group, but the group is already full	System sends an error message saying "The group that you are joining is full"
2	User tries to invite an unregistered friend	System sends an error message saying "The person being invited is not registered"
3	User tries to log in, and encounters a technical issue	System sends an error message saying "There was an issue logging in. Please check your internet connection and try again"

System Architecture

Components

- **Frontend:** Web-based user interface built with React.

- **Backend:** C++ REST API (built with CrowCpp) and a Node.js Server (data processing, authentication, and C++ REST API integration).
- **Database:** Firestore NoSQL cloud storage.
- **Authentication:** Firebase Authentication (using email and password provider).

Architectural Pattern

We utilized the standard Client-Server architecture for our Web-App. Below is a general diagram of how our architecture works (using eraser.io to generate the diagram).