Section 1 - Project Description

1.1 Project

Meet In The Middle

1.2 Description

The Meet in the Middle App is a web application which will help a group of users find mutually accessible locations.

Meet In The Middle (MITM) is an app designed for families and friend groups, and is intended to simplify event planning and social coordination. Through the app, users will create profiles, add friends to their profile, and set their preferences. To make planning easier, the app will suggest meeting places based on distance, traffic, and travel preferences. The app will help users and groups plan outings with event itineraries. Additional features of the MITM app include real-time traffic updates, event notifications, and privacy controls to ensure a seamless and enjoyable social planning experience.

1.3 Revision History

Date	Comment	Author
10/8/2023	Version 0.1 – Initial Draft	Craig Walkup

Contents

Section 2 - Overview

2.1 Purpose

This document is intended to be used by the MITM development team and by project maintainers.

2.2 Scope

This document describes every component and integration of the Meet in the Middle app.

The primary benefits of the Meet in the Middle (MITM) app is to facilitate easy social planning and event coordination among family, friends, and groups. We aim to streamline the process of selecting meeting destinations based on user preferences, distance, and traffic, while the goals include enhancing social connectivity and making event planning more efficient and enjoyable.

2.3 Requirements

Your mileage may vary -- we typically break down the requirements to provide a ballpark estimate.

2.3.1 Estimates

#	Description	Weeks Est.
1	UX/UI	3
2	Frontend Development	4
3	Backend Development	4
	TOTAL:	11

2.3.2 Traceability Matrix

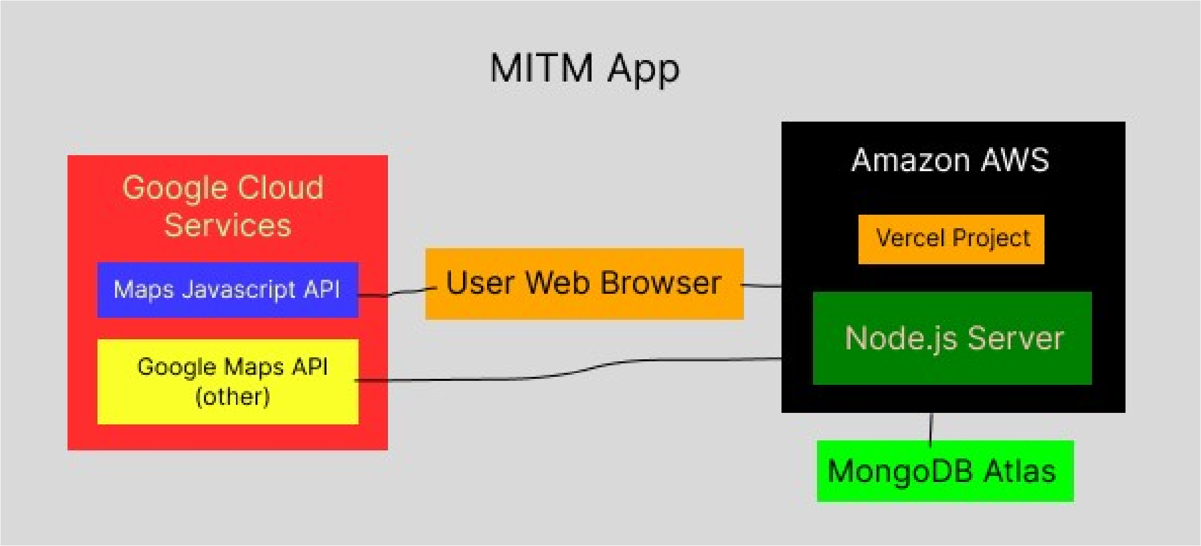
Cross reference this document with your requirements document and link where you satisfy each requirement

SRS Requirement	SDD Module
Req 2	2
Req 3	3.1, 3.2, 3.3

Section 3 - System Architecture

The MITM will use the Express web framework with Amazon AWS hosting and MongoDB Atlas integrations.

Describe/include a figure of the overall system architecture (and where this module fits in)

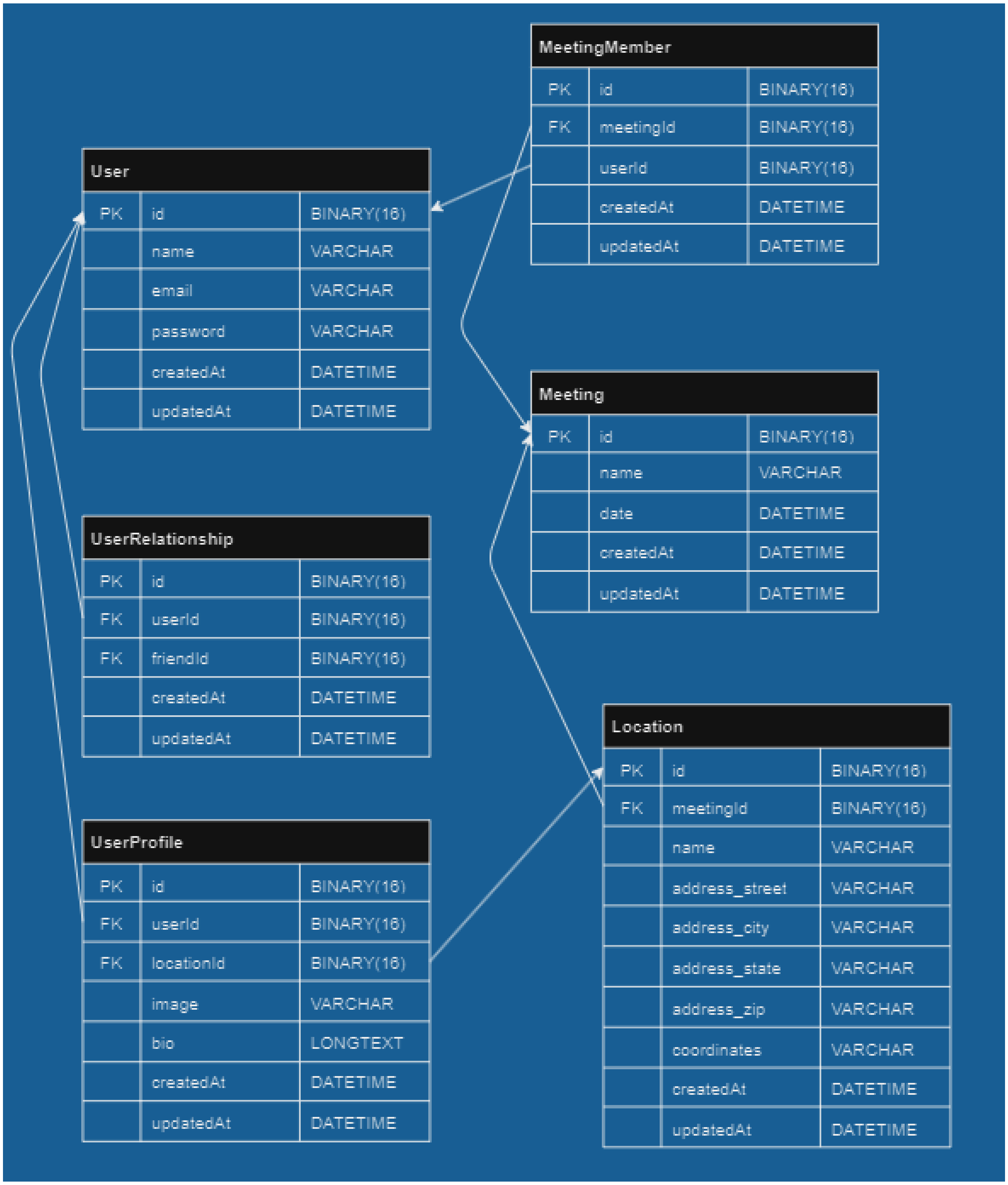


The site will be deployed through Vercel. Express will handle the requests and MongoDB atlas will provide storage services to the server application. The web browser will use Maps Javascript API and the Node.js server (which is deployed via Amazon AWS)

Section 4 - Data Dictionary

Endpoints and Data Fields:

All tables and data are further explained in Section 6. Each of the following data fields will contain an api endpoint in the backend of the application.



MeetingMember		
PK	id	BINARY(16)
FK	meetingId	BINARY(16)
	userId	BINARY(16)
	createdAt	DATETIME
	updatedAt	DATETIME

User		
PK	id	BINARY(16)
	name	VARCHAR
	email	VARCHAR
	password	VARCHAR
	createdAt	DATETIME
	updatedAt	DATETIME

Meeting		
PK	id	BINARY(16)
	name	VARCHAR
	date	DATETIME
	createdAt	DATETIME
	updatedAt	DATETIME

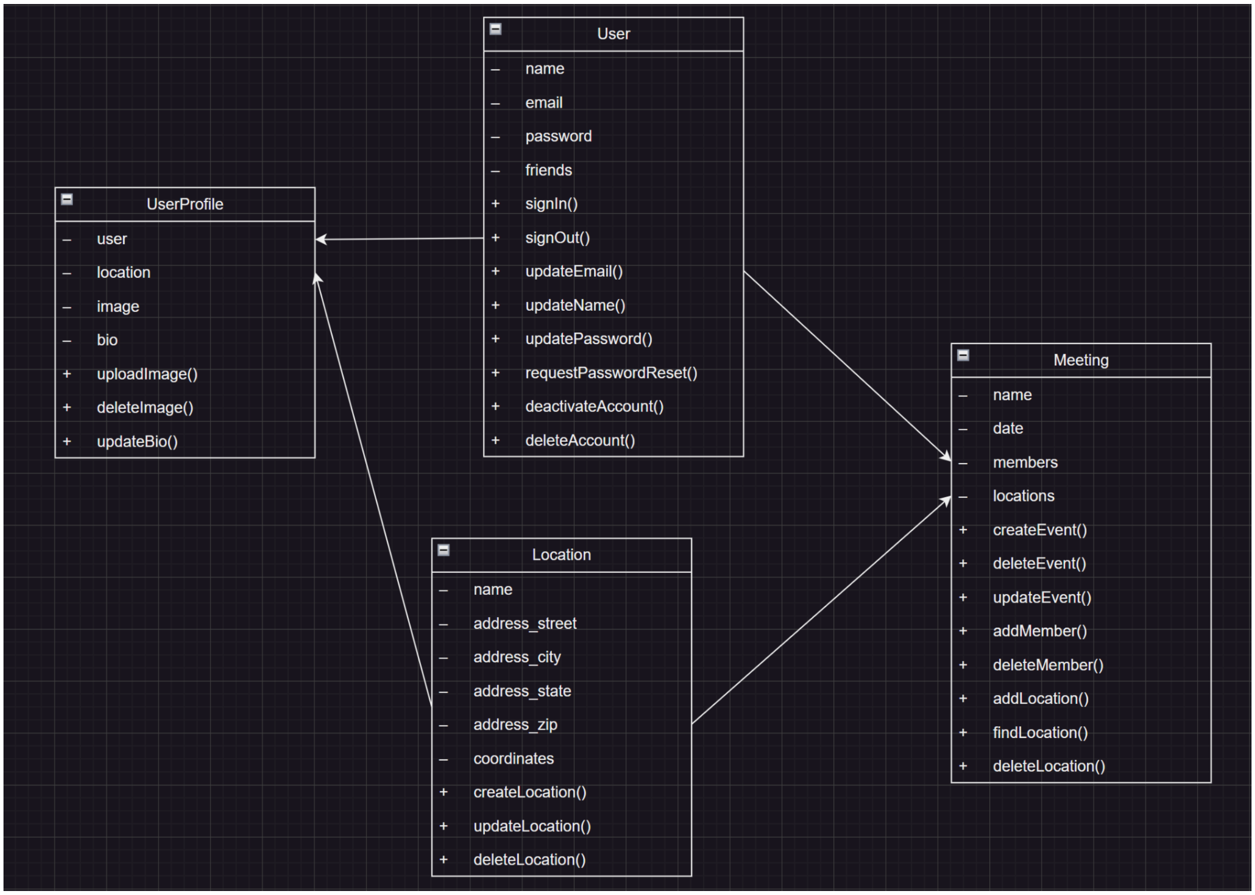
UserRelationship		
PK	id	BINARY(16)
FK	userId	BINARY(16)
FK	friendId	BINARY(16)
	createdAt	DATETIME
	updatedAt	DATETIME

Location		
PK	id	BINARY(16)
FK	meetingId	BINARY(16)
	name	VARCHAR
	address_street	VARCHAR
	address_city	VARCHAR
	address_state	VARCHAR
	address_zip	VARCHAR
	coordinates	VARCHAR
	createdAt	DATETIME
	updatedAt	DATETIME

UserProfile		
PK	id	BINARY(16)
FK	userId	BINARY(16)
FK	locationId	BINARY(16)
	image	VARCHAR
	bio	LONGTEXT
	createdAt	DATETIME
	updatedAt	DATETIME

Section 5 - Software Domain Design

5.1 Software Application Domain Chart



5.2 Software Application Domain

A Comprehensive high level description of each domain (package/object wherever it is better to start) within the scope of this module (or within the greater scope of the project if applicable)

5.2.1 Meet in the Middle Public Website

The web page is retrieved from the Express server. It features a navigation bar. Various pages in the domain use Maps Javascript API to display key information.

5.2.1.3.1 Event List

Users will retrieve a list of events.

5.2.1.3.1 Itinerary Editor

N/A

5.2.1.3.1.1 Itinerary Map

N/A

5.2.1.4 Plans

N/A

5.2.1.5 Account

N/A

5.2.1.6 Logout

The backend is built using a REST API and therefore logout will occur when the cached JWT expires. No logout button is needed on the frontend

5.2.1.7 Location Search Results

N/A

5.2.2 Express Server

The server will prepare location search results, profile pages, and event pages to be sent back to the user's web browser. It will be powered by Amazon AWS services, Next.js, various Google Maps APIs, and MongoDB Atlas.

5.2.2.1 Login

The first time a User opens the page, a login page is displayed by the NextJs application, with a login field, password field, submit button, and "create account" button.User email and password is sent to the backend which returns a JWT for authentication. On subsequent calls to the backend the JWT is used as authentication.

If instead the user signs up using their Google Email, the backend will instead connect to Google API and acquire user email and provide authentication using OAuth2.

5.2.2.1.1 Registration

The user presses the “create account” button from the sign in page. Upon submission of the registration form, account credentials are stored in the database for the user.

The registration form will include the user’s:

Location

Username

Password

5.2.2.1.2 Sign In Submission

The “Sign In” form is submitted. If the credentials are valid, a JWT is returned for authentication. On subsequent calls to the backend the JWT is used as authentication. Upon a successful login, the user is redirected to the “Profile” page.

5.2.2.2 Dashboard (Profile)

The dashboard will exist on the user’s homepage and provides information about the user’s friends, events, and possible meeting locations. The profile page contains a link to the “Create Event” form.

5.2.2.3 Events

Events drive the meetup process. They connect users and compile user information. The user will see a list of events. The user will be able to retrieve an event page.

5.2.2.3.1 Create

Upon submission of the event form, a new event is added to the database. From the returned page, the user will be able to navigate to the locations map and add locations to the event.

5.2.2.3.2 Interactive Map

NextJs creates an interactive map for the user. The map will provide both the location of a user and the friend that the user needs to meet with.

5.2.2.3.3 Add Location

Processing a profile form. Profiles are added to events?

1. If the form is successfully processed the information in the form is posted to the backend at which point a User is created.
2. If there was an error in processing the user we are presented with an error and asked to fill in the form again with accurate info.

5.2.2.3.5 Event Get: List

The server will return a list of the user's events.

5.2.2.3.6 Event Get: Detail

The server will return a detailed view of an event.

5.2.2.4 Friends

The user will be able to find and collaborate with their friends. Friends are automatically added to/are selected upon the creation of the event(s)? Attendees can decline the add?

5.2.3 Search for Locations

The server will return the search results for the page.

1. The results have been cached. The search result template is filled out and sent back to the user.
2. The results have not yet been found. Google Maps API is queried for locations and suitable locations are selected, (cached?), and returned.

5.2.3 MongoDB Atlas

MongoDB Atlas is an online database storage system. The MITM web app will use MongoDB to store all database information. Collections are analogous to a relation, and a document to an entry. The Mongoose package from NodeJs creates the database schema and uses CRUD operations to manage the database. See Section 4 for database structure and Section 6 for database information.

Section 6 – Data Design

Describe the data contained in databases and other shared structures between domains or within the scope of the overall project architecture.

6.1 Persistent/Static Data

6.1.1 Dataset

The dataset of this application is based around a User. After the user establishes their credentials via a JWT they will have access to the following data.

6.1.2 Static Data

This web app uses Mongoose as a NodeJs package. The models created via Mongoose will not change throughout the life cycle of the application. Models will include User Profile Model, Meeting Model, Location Model, and User Relationship Model.

6.1.3 Persisted data

User Profile, location, and recommendations will all change as the app is updated.

🧑User Profile:

Each User will contain a unique identifier that is generated by Mongoose. A unique email will also be associated with the user. A password and some metadata will also be stored within the User field.

🧑User Profile Information:

User profile stores more detailed information about the User including a short bio. Currently “User Profile” and “Profile Information” are two separate fields but this may change according to the needs of the app.

📍Location:

Location of a recommended meeting is stored in the database. Location contains name, address, coordinate info and the unique Google address that is provided by Google API. Location will also store a meeting id that will link to a specific meeting.

🤝Meeting:

The meeting field is linked to a specific location via a meeting id.

6.2 Transient/Dynamic Data

Not all data is persistent. The following data does not need to be stored in a database but will be acquired dynamically

🪙Javascript Web Token

JWT will be cached on the device and used as a credential every time a request is posted to the backend.

📲Messaging

There is the future possibility of building messaging into the application. If this feature is added the data will be cached on the device and not stored in the database.

6.3 External Interface Data

Data is gathered from external API’s to provide additional features to the User.

🗺️Google Maps API

This API displays an interactive map for the user to view their location and the location of friends.

🛂Google Auth via Passport

Passport is a NodeJs package that will be used to exchange authentication information between the front end and back end of the application. When a user first signs up for the application their Google email address is acquired from Passport.

6.4 Transformation of Data

Describe any data transformation that happens between design elements.

🧑User Data

User data will be obtained from the authentication service and modified to fit the standard format of the database.

📍Location Data

Data received from Google API is modified into a standard format across the platform.

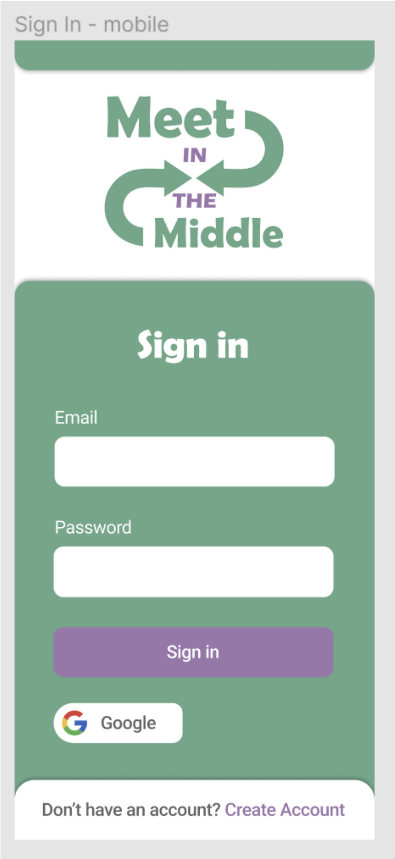
🤔Recommendations

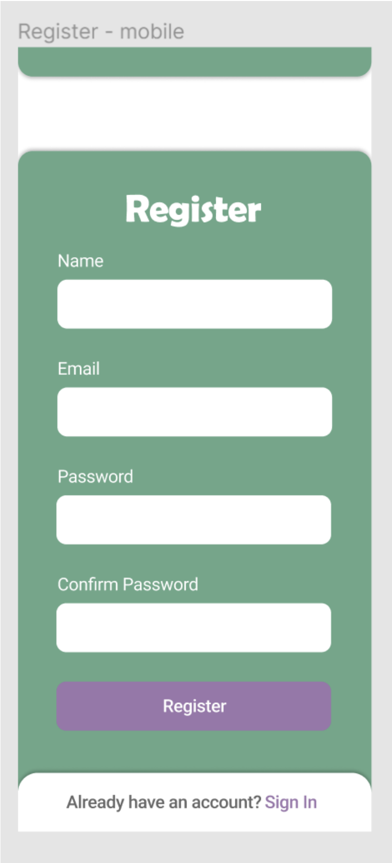
The location algorithm uses both user data and preferences to create recommendations. Recommendations are then stored in a meeting and sent to the front end.

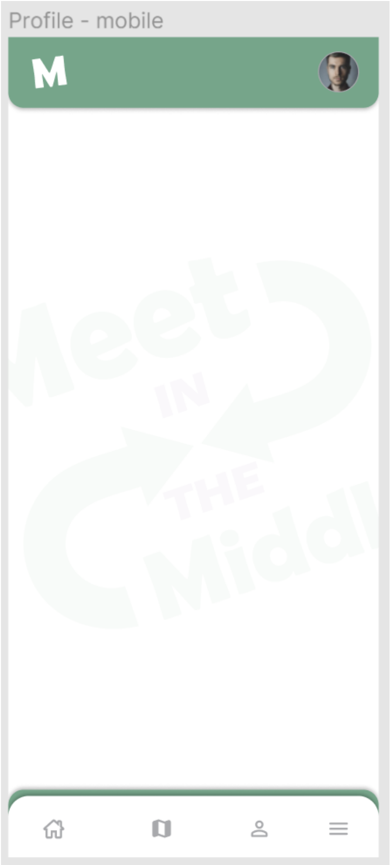
Section 7 - User Interface Design

7.1 User Interface Design Overview

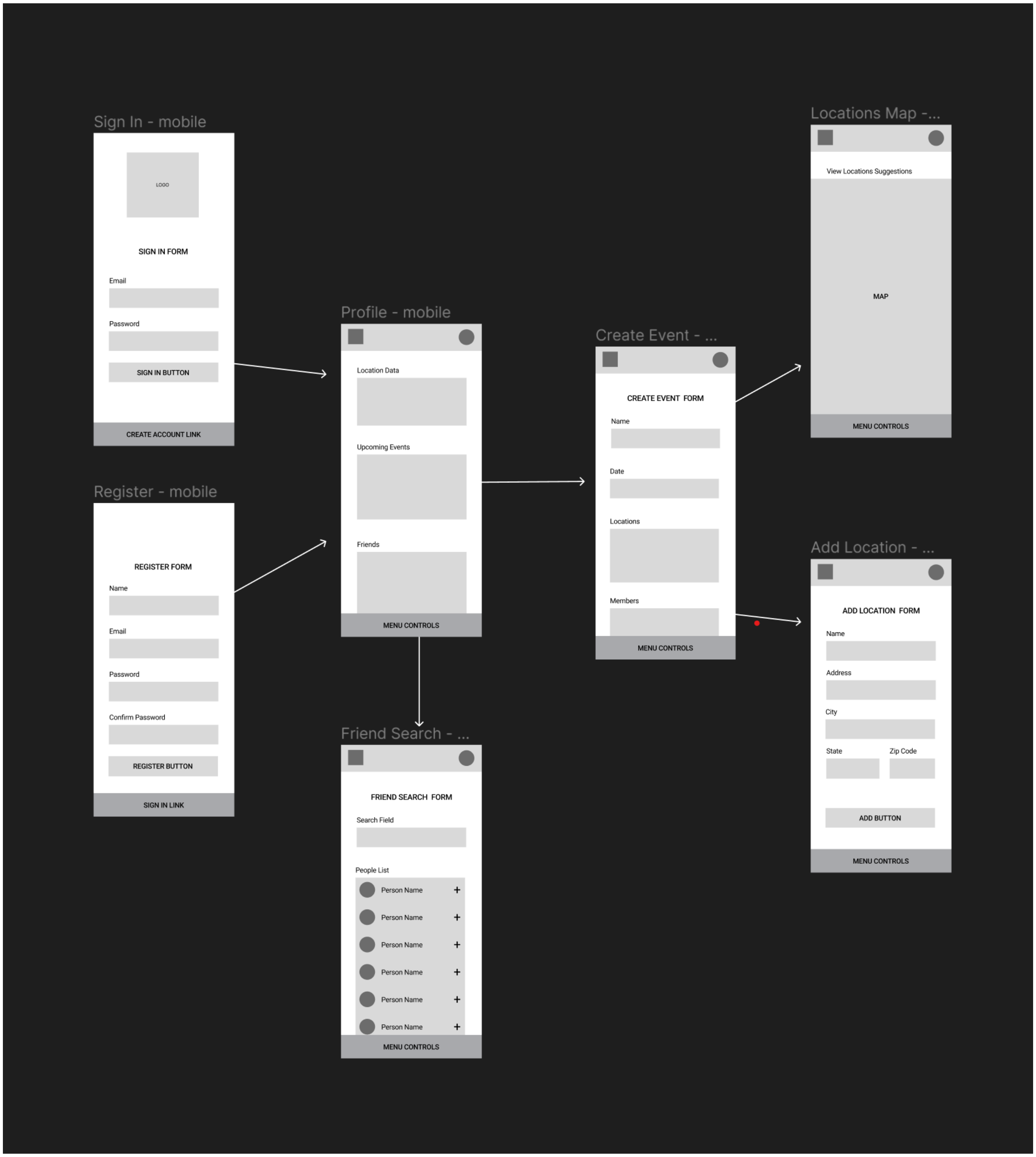
The user interface design consists of a simple color scheme with three primary colors following the 60-30-10 rule to provide a stylistic and clean experience. The main design is centered around the logo with soft accents and a rounded appearance. The main pages will utilize a menu bar at the bottom of the application for ease of access and control. The links at the bottom navigation will consist of a home link, a maps link, a person search link, and a more link that will provide a menu for main navigation. This will include the ability to view the map, handle events, handle friends, and handle their location. There will be a header at the top with the profile image and logo. The profile image will also serve as a secondary navigation menu with the profile controls and logout link.



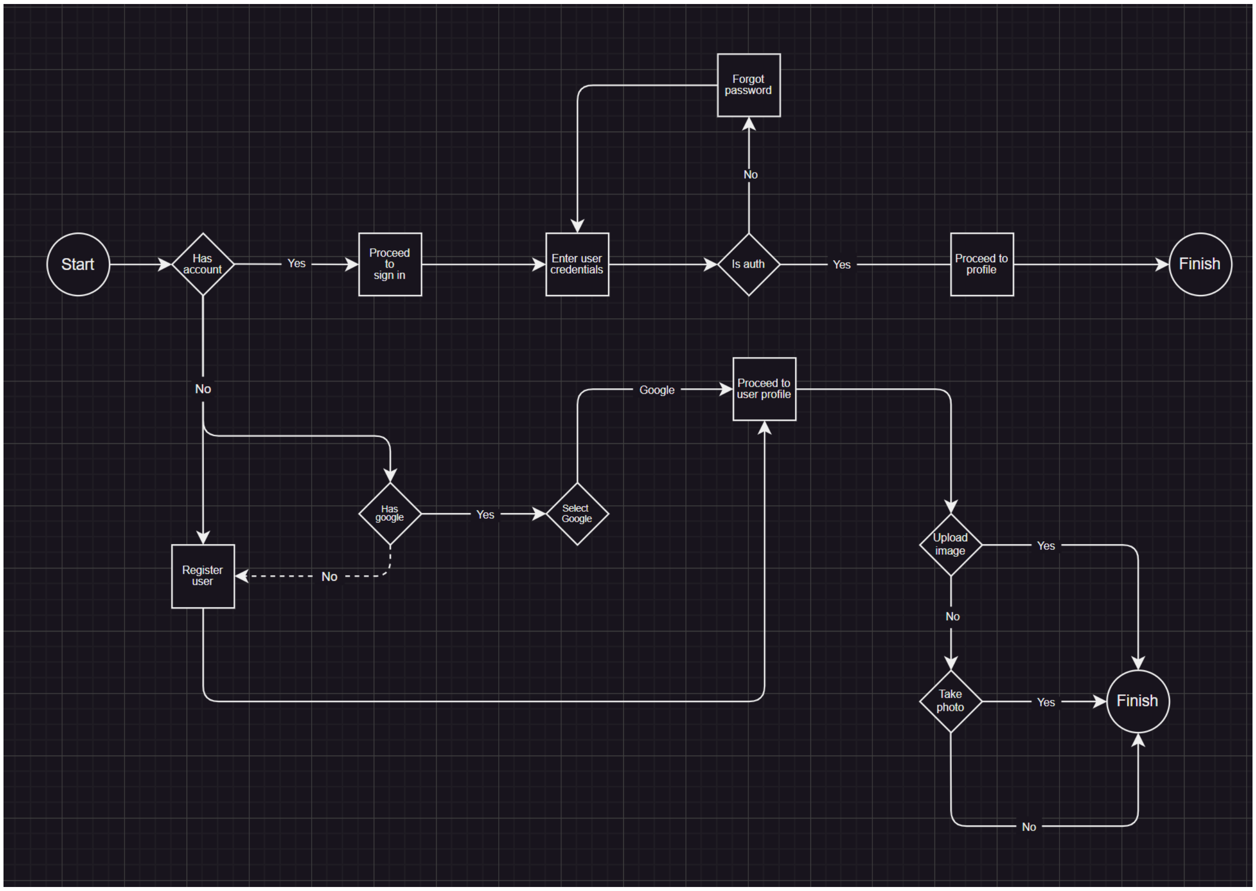




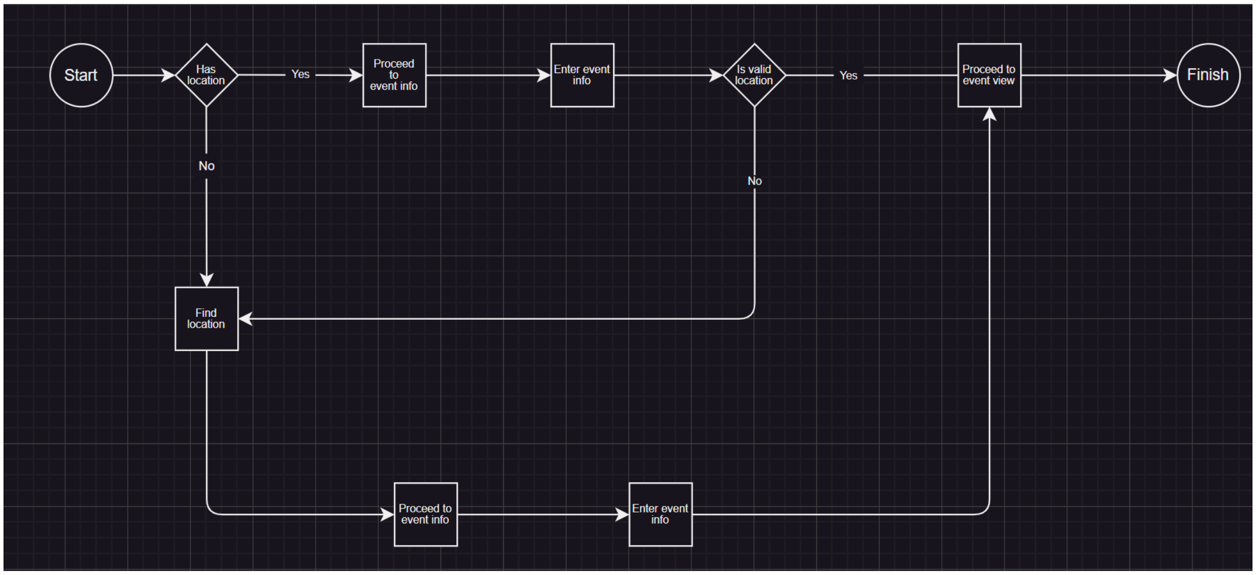
7.2 User Interface Navigation Flow



User authentication and authorization flow chart:

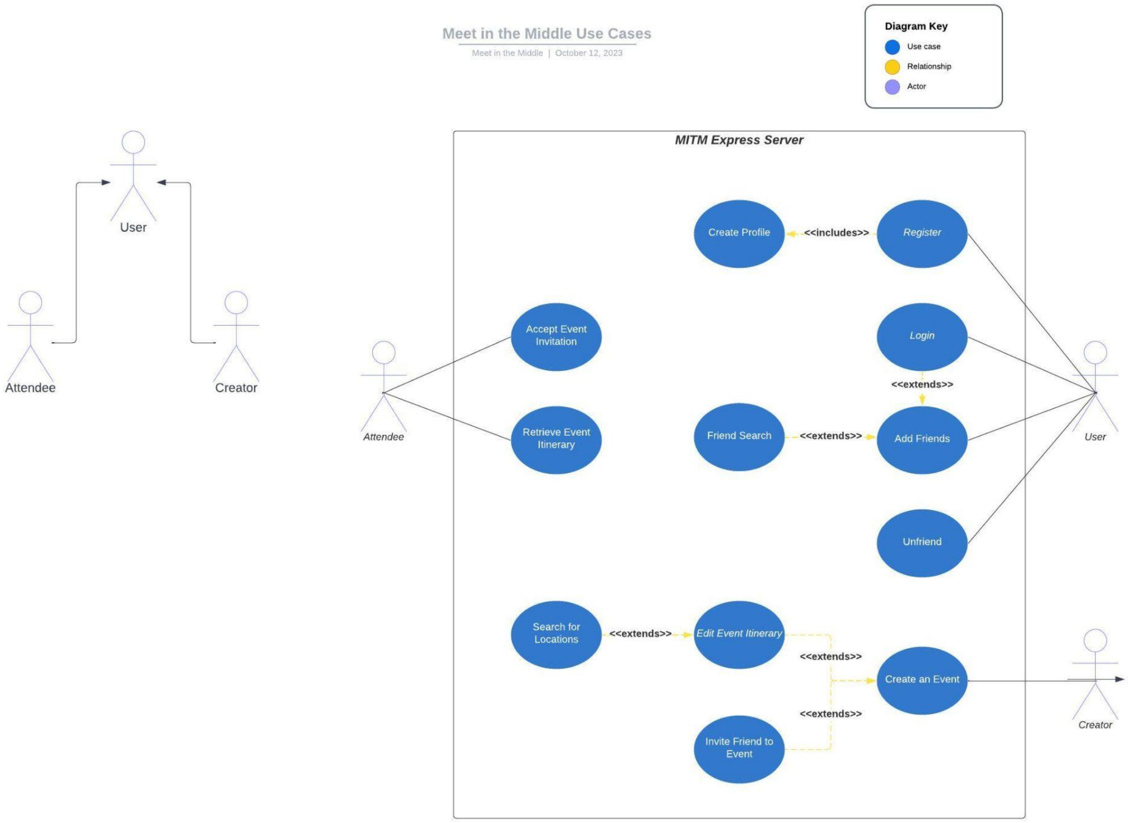


Event creation flow diagram:



7.3 Use Cases / User Function Description

After the initial authorization the user will be taken to their main profile screen to get a quick overview of what is immediately available. The bottom navigation allows the user to go back to the profile overview via the home link, view a map of locations via the maps link, view friends and add friends via the people link, and get access to other navigation opens via the menu link. The main menu will allow users to navigate to the events management where they will be able to add events, modify events, and delete events. User management will allow for users to be searched by name or email and added. Users must be friends to add individuals to event meetings. Forms will be designed and implemented to control the flow of data.



Section 8 - Other Interfaces

Identify any External Interfaces used in the execution of this module, including technology and other pertinent data.

8.1 Interface X

The following is the list of External Interfaces that are known to be necessary for the MITM web app. Additional Interfaces will be added as needed.

🚀Google Cloud Services

> Multiple Google cloud Services will be used with the project. This will require both a Google account and a Google API key.
>
> 🛂Google Auth via Passport
>
> > Authentication is available via Google Account. Passport is used in the NodeJs backend server to communicate and obtain authentication credentials.
>
> 🗺️Google Map API
>
> > The Google Map API is contacted by both the font and backend. The front end uses the service to acquire and send locations to the backend. The backend also contacts Google Map API to determine meetings.

💿AWS

> AWS Amplify allows for full stack deployment of web applications. This is where the application is deployed for use.

Section 9 - Extra Design Features / Outstanding Issues

This is a list of known issues and corrections needed for the development of the MITM web app.

Outstanding Issues:
A comprehensive description
Proof of concept
Completion

Section 10 – References

💻Github MITM Project:
https://github.com/CSCI150-LAB01/Meet_in_the_Middle

Section 11 – Glossary

Glossary of Terms / Acronyms

MITM - Meet In The Middle
The user - An authenticated web browser.