CS 490

Design Document

MapItUp

Abhijay Gupta

Ankush Jain

Sharoon Srivastava

Table of contents

|  |  |
| --- | --- |
| Title | Page |
| Purpose | 3 |
| Design Outline | 6 |
| Design Issues | 8 |
| Design Details | 10 |

Purpose

Today, travel has become increasingly sought after. However, there is no single platform to help these people organize their trips and travels in an organised manner. Although existing web apps enable users to rate and review hotels and tourist spots, they do not facilitatekeeping track of the places visited. They also don’t provide users the ability to share the photos with other users.With MapItUp, we plan to give the users a platform where they can organize and share information about their travels.

**Requirements :**

**Functional**:

* As a user, I would like to create an account
* As a user, I would like to update my account information (including password, email, etc)
* As a user, I would like to update my profile (including profile picture, location, etc)
* As a user, I would like to store the places I’ve visited
* As a user, I would like to view the places I’ve visited on an interactive world map
* As a user, I would like to add reviews and ratings to the places I’ve visited
* As a user, when I review a destination, I would like to add the travel cost for a particular destination
* As a user, I would like to mark a destination as my favorite
* As a user, I would like to view my favorite destinations
* As a user, I would like to see the percentage of the world I’ve travelled
* As a user, I would like to add photos to the places I’ve visited
* As a user, I would like to mark a particular photo of mine as public or private
* As a user, I would like to view public photos of the places visited by all users
* As a user, I would like to view the ratings and reviews of places visited by all users
* As a user, I would like to follow other users
* As a user, I would like to view my followers
* As a user, I would like to exchange messages with other users
* As a user, I would like to view the popular travel destinations for a certain month or duration
* As a user, I would like to view featured travellers
* As a user, I would like become a featured traveller if I meet the requirements
* As a user, I would like to search for travel destinations and filter them based on ratings, distance, weather, travel cost etc
* As a user, I would like to view the distance, weather and travel cost of a destination
* As a user, I would like to deactivate my account
* As a user, I would like to mark a review as helpful or unhelpful (if time permits)
* As a user, I would like to sort the reviews based on several factors (including helpfulness, date etc) (if time permits)
* As a user, I would like to login / sign-up using my facebook credentials (if time permits)

**Non Functional**:

* As a developer, I would like my service to be scalable by using mySQL to manage my database
* As a developer, I would like the least possible amount of data storage on the client
* As a developer, I would like to validate all user accounts to prevent redundancy
* As a developer, I would like the API methods to have publically available documentation
* As a developer, I would like the interactive map to be smooth and non laggy
* As a developer, I would like to have a secure mySQL database to store the personal travel information for all users
* As a developer, I would like to be able to test on a local development server easily
* As a developer, I would like users to be able to navigate through the web app easily
* As a developer, I would like my service to have fast response times

Design Outline

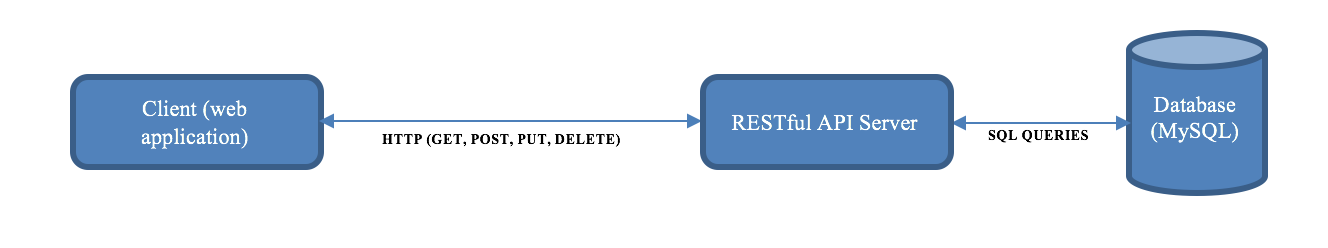
**Overview and Components**

The software product will consist of:

1. A web app as the client.
2. A back-end server exposing a RESTful API.
3. A database being accessed by the server.

**Architecture**

Abstractly, the product will make use of the client-server paradigm. Users will use the web application which will interact with the server. The client and server will be implemented separately and interact using HTTP requests. All the information will be stored in the database which would be accessed by the server whenever there is a HTTP request from the client.



**RESTful API Server**

The RESTful API server will interact with the client and the database, maintain abstractions, and send data to the client. The server is independent from the client. JSON data format will be used to send and receive data on the client and the server.

**Client**

The client would be a web application that will communicate with the server via HTTP requests. The app will be written using JavaScript. The app needs to be well designed and easy to use. As such the UI and UX elements for the app will need to be taken into consideration.

**Database**

The database being used is MySQL that would be accessed and queried by the back-end server. The database would store information about the users and destinations.

Design Issues

**Issue 1: Software Framework to use**

Option 1: Flask

Option 2: Django

Option 3: Express/node.js

Decision: We’re planning to use the Express framework because:

* It’s easier to define dependencies in Express as compared to other frameworks.
* Django is structured in a way which could take up a significant amount of time to learn.
* Express is a framework for node.js and is one of the best frameworks for a multi page web app.
* All the team members are familiar with Javascript, and Express uses Javascript

**Issue 2: Database Architecture**

Option 1: MongoDB

Option 2: MySQL

Option 3: SQLite

Decision: We’re planning to use MySQL since:

* MongoDB is a non relational database and hence would not provide efficient searching for places to visit
* All the team members have experience with relational databases and MySQL is a relational database
* SQLite is not as scalable as MySQL

**Issue 3: Reducing redundancy to avoid multiple profiles from a single user**

Option 1: Sign in using a social media platform

Option 2: Enter a valid email address and confirm it

Option 3: Register via a phone number and confirm with a verification code sent as a text message

Decision: We’re planning to use social media sign in since

* Phone number validation requires a monetary overhead
* Verification through emails requires email validation and sending of temporary verification links to a user’s inbox. This entails a development and maintenance overhead that could be avoided
* Facebook or social media validation process is easier to implement and is free of charge. We could use the existing Graph API to save time and implement reusability

**ISSUE 5: Enabling chat between every user or specific user types.**

Option 1: Enable chat between all types of users. In this scenario a user can search for any other user and exchange messages with them.

Option 2: Limit chat between followers. Messages can only be sent if at least one follows the other.

Option 3: Users should request exchanging messages with users they are not following or users that do not follow them. The users would only be allowed to exchange messages if the other user accepts the request.

Decision: Option 1 was chosen. This is because we don’t want to get into the complications of ‘read’ receipts which we would have to do for option 3. Furthermore, we don’t want to limit exchanging messages with only followers.

Design Details

**System Components**

1. **Client:** The client will be a web application built natively using Javascript using angularjs. It will make use of material design to drive a tabbed user interface that allows the user to quickly switch between views to accomplish different tasks. Every view will be associated with an activity that will drive its behavior as well as a layout that will arrange components. The client will make HTTP requests to the API server. The client will be built to parse JSON data format as it will make calls to the API server as well as receive login information from the Facebook API.
2. **API Server:** The API Server will be built using the Javascript programming language with express framework for node.js. It will respond to HTTP Get requests from client by responding with JSON. It will make use of SQL Queries to communicate with the database.

1. **Database:** MySQL will be used for the database. It will hold a variousrelated tables encompassing user information and ratings.

**Search functionality**

Users will be able to search for places to visit based on various filters. These include - ratings & reviews, most visited, trending, weather, average cost, distance, etc. We would also provide the users to sort the search results.

**Personal database functionality**

Users will have the option of storing their personal information. These include - the places they visited, the date of the visit and the duration they stayed. They will also have the option of including a review of the places they visited and about their trip as a whole.

**Follower/Following functionality**

Users would be able to follow other users. This can further extend into following famous/featured travellers. We would also provide the users the ability to see who their followers are.

**Ratings and review functionality**

Users would have the ability to rate and review the places they visit. Other users can read these ratings and reviews for a better judgement of the destination.

**Photos functionality**

Users will be allowed to upload photos of their travels to their profile. These could be private or public depending on the user's’ preference.

**Facebook login**

The app will make use of the Facebook graph API for registration and login. The API will generate a token for every user that will be stored in the database and referred to for login. The API will also provide general information about the user thus limiting the need for a registration or login form on the client and ensuring faster, user friendly logins.

**Class Descriptions**

**User:** The default user class associated with every user once an account iscreated. The class store basic information such as name, email and password. More information can be stored later such as location, percentage of world covered, etc. It has a many to one relationship with the back end server as a single server can handle multiple users.

**Ratings:** The Ratings class stores the ratings and reviews for a particular destination. Itstores a value between one and five for a rating and a textual review. It has a many to one relationship with the Place class because each rating class is associated with one destination and the destinations are associated with multiple ratings.

**Place:** This class would store the name, location (longitude and latitude), population etc. for a particular place or destination in the world. It has a one to many relationship with the Ratings class as a destination is associated with multiple ratings.

**Sequence Diagrams**

1. Sequence of events when user first opens the app



2. Sequence of events when user registers for the app



1. Sequence of events when the user opens the app when he is already registered (signed in)



1. Sequence of events when the user adds country to list of places visited

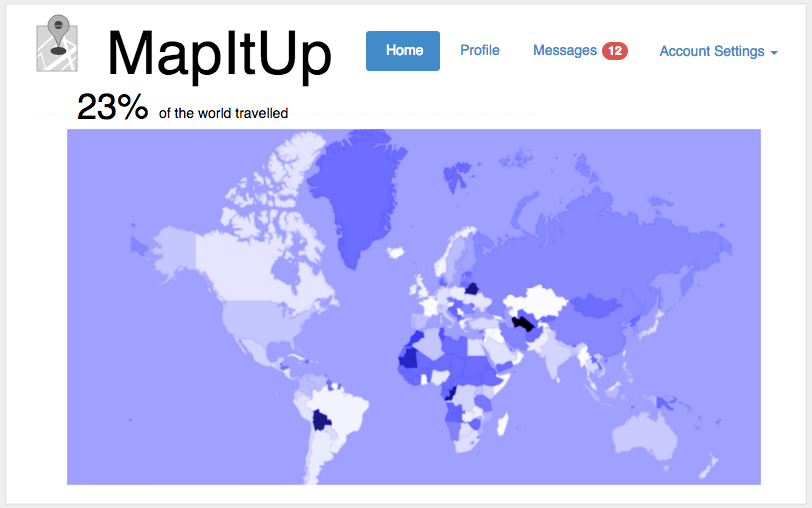


**UI mockup**

* Sign in page



* Homepage



* Ratings and reviews for a country

