OffbeatEscape

Software Requirement Specification

University of Windsor

Objective

This Software Requirement Specification document is meant to present a detailed description of our product's requirements & features to accelerate user's understanding. The purpose of this document is to help stakeholders, architects and developers visualise the overall product. It shall act as roadmap for development of the project - OffbeatEscape.

Author – Team 4

*Recipient - **Dr. Aznam Yacoub***

*State – **Finalised***

*Updated On – **12-04-2021***

History

Date	Updates	State
28-01-2021		Proposed.
11-02-2021	Specifications updated for User Authentication & Post Configuration – Writer End.	
01-03-2021	Specifications updated for Post Configuration – Reader End & Finding and Adding Friend.	
15-03-2021	Specifications updated for User Authentication, Post Configuration – Writer-Reader end & Adding Friend.	
12-04-2021	Specifications updated for Inspiration history and dashboard features. Non-Functional specifications updated.	Finalised

Index

Introduction

Purpose

This document shall elaborate software requirement specifications for Version 1.0 of the project - OffbeatEscape, a software solution to track the influence of social media on popularity and recognition of destinations off the charted routes. It will be a cloud-based social media web application designed for avid travellers to share their experiences and stories of offbeat locations with their friends and family; and read & explore what all destinations their friends have been discovering.

Document Conventions

Font style and size used for Titles – Calibri Light (22)
Font style and size used for Sub-Titles – Calibri Light (18)
Font style and size used for Information – Calibri (10)

Intended Audience and Reading Suggestions

This document is intended for all stakeholders; clients, product director, developers, testers; to understand the basic idea behind the application, its requirements and the main features required to build the software.

Developers and testers can focus directly on the features (Section-3) while it is suggested for other stakeholders to understand the overall product before getting into detailed functionalities.

Scope

This application will be a social media platform designed for travel enthusiasts to journal and share their experiences, journeys, plans and routes of the offbeat locations with their connections; view, comment-on and bookmark their friend’s posts. User will be able to view the inspiration path that led a post to their reading list. The main goal and vision of this application will be tracking and showcasing the influence of friends in social media on tourism at offbeat locations.

Overall Description

Product Perspective

Over the last decade, social media has had major influence on how people communicate, connect and travel. Users share their personal experiences of journeys to travel destinations; and seek inspiration and validation from their connections or friends on social media for future travel plans. Considering the same, social media applications have turned into marketplace for tourism industry and as a result these applications are more focused on recommending locations.

Our application would provide a platform for users to share their personal experiences and read their friends'. While user will have an option to filter the feed based on his interests, he will be viewing posts tailored to his connections only. The main goal of application will be to measure how a location interest is spread through the network.

Product Functions

Following will be the key features of the product:

- Sign Up and Login interface – Google / FB authentication integrated.
- Sharing Experience while keeping privacy intact
- Finding and Adding Friends
- Inspiration History for a user to see the cycle of events that led a post to his reading list.
- Trending Posts to promote posts that have been saved most no. of times.
- Reading list for all the saved posts.
- General Feed for all posts from friends.

Operating Environment

Our application will be compatible with all modern web browsers and internet explorer versions after 10.

Application will be accessible via desktop, mobile and tablet devices.

Native browser options, like local storage and cookies, will be used.

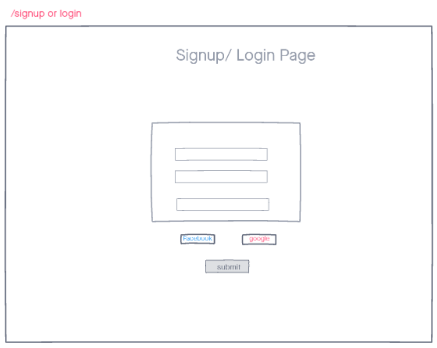
Every call to API will be from the deployed cloud server.

External Interface Requirements

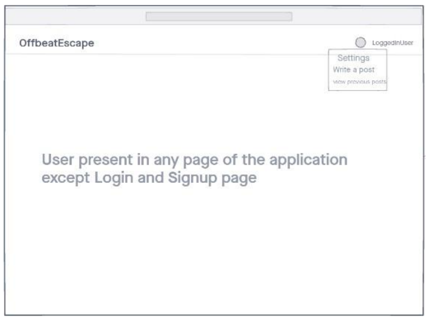
User Interfaces

Following will be the major user interfaces:

- Sign Up / Login Page

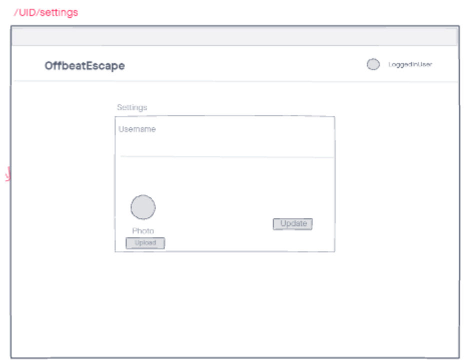


- Basic Display

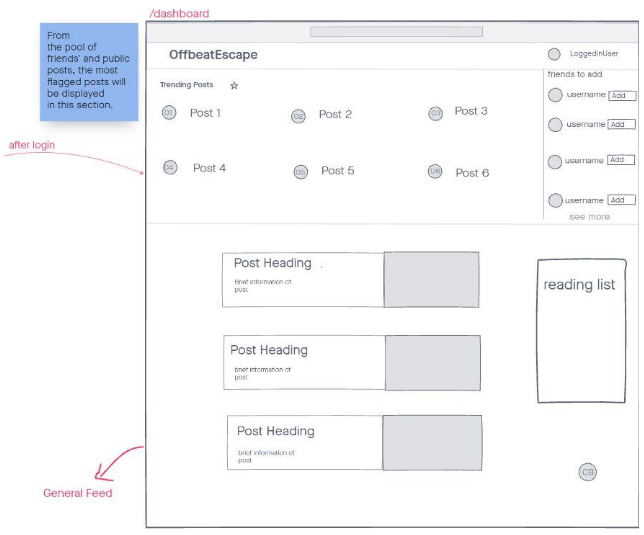


User present in any page of the application except Login and Signup page

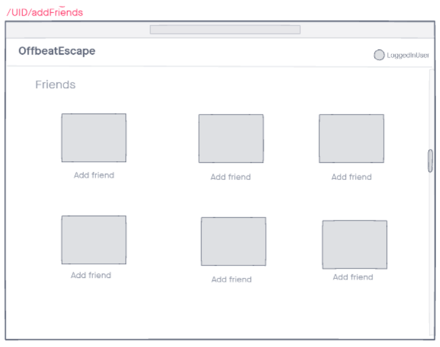
- Profile Settings



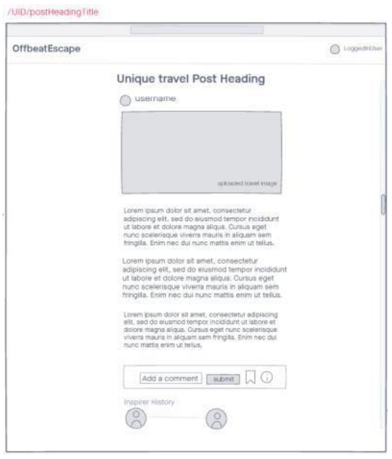
- Dashboard (Accessible after successful login)



- Add-friends Pane.



- General display for a post

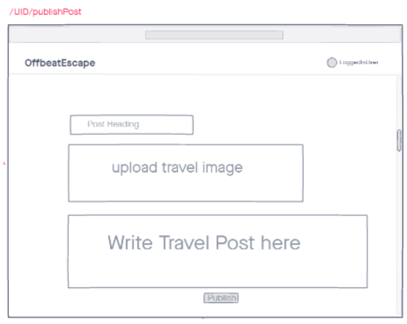


Lorem ipsum dolor sit amet, consectetur adipiscing elit, sed do eiusmod tempor incididunt ut labore et dolore magna aliqua. Cursus eget nunc scelerisque viverra mauris in aliquam sem fringilla. Enim nec dui nunc mattis enim ut tellus.

Lorem ipsum dolor sit amet, consectetur adipiscing elit, sed do eiusmod tempor incididunt ut labore et dolore magna aliqua. Cursus eget nunc scelerisque viverra mauris in aliquam sem fringilla. Enim nec dui nunc mattis enim ut tellus.

Lorem ipsum dolor sit amet, consectetur adipiscing elit, sed do eiusmod tempor incididunt ut labore et dolore magna aliqua. Cursus eget nunc scelerisque viverra mauris in aliquam sem fringilla. Enim nec dui nunc mattis enim ut tellus.

- Writing a post



Hardware Interfaces

It will be a cloud-based web application supported for modern browsers – Google Chrome & Mozilla Firefox. Support for other browsers can be validated at later stages.
Application will be accessible by desktop, mobile and tablet devices.

Software Interfaces

The application will require the following:

- Amazon EC2 using Ubuntu 20.04 will be used for hosting the application.
- AWS S3 bucket will be used for storing images.
- MongoDB will be used for database.
- NGINX

Communication Interfaces

Internet connection will be required to access or upload any posts to application. HTTP protocol will be used for communication over the internet.

System Features

1. User Authentication

User authentication includes successful signup and login for a user. Access to application will only be available to logged in users.

SIGN-UP Interface:

- Allow via email, Google and Facebook.
- If via email,
  Password length shall be restricted to min 8.
- Already registered user shall not be able to register again.
- Password shall be encrypted or hashed before storing in database.

A signup form shall be presented to user for signup/registration.
Form shall provide option for all 3 authentication methods – EMAIL, Facebook & Google.

EMAIL Registration

Parts of form:

- 3 textboxes for – email, password & confirm password will be present.
- A single “Submit” button to submit the form data.
- A link to switch to login page in case user is already registered.
- 2 Buttons to authenticate using Facebook/ Google.

Constraints:

- Check email for pattern validation.
- Password should be of minimum 8 length.
- ConfirmPassword should be same as Password.
- Check for already registered user, if true, prompt user that he is already registered and can directly login to continue.

Flow:

- On submitting the form, the data shall be validated before inserting it into database.
- Once stored successfully in DB, prompt user with successfully registered message and asked for login to continue accessing application.

- If user already registered, reject the submission and prompt user with message indicating existing profile.

Database:

- Store email and password in DB “Users” Collection.
- Password should be encrypted before storing it to database.

LOG-IN Interface:

- User should be able to login after successful signup.
- Login shall be allowed via email, Google and Facebook.
- In case of login via email, for every login attempt, id and password shall be authenticated.
- Forget Password option shall allow user to reset password via email.
- Check for unregistered email-id.

A login form shall be presented to user for logging into the system and accessing dashboard or user’s own profile.
Form shall provide option for all 3 authentication methods – EMAIL, Facebook & Google.

EMAIL Login

Parts of form:

- 2 textboxes for – email & password.
- A single “Submit” button to submit the form data.
- A link to switch to signup page in case user is not already registered.
- 2 Buttons to authenticate using Facebook/ Google.

Constraints:

- Check email for pattern validation.
- Password should be of minimum 8 length.

Flow:

- Entered data shall be validated against database collections.
- If entry matches, prompt login successful and redirect user to dashboard.
- If entry id not found, prompt user with message indicating incorrect email/password.

Facebook Authentication:

“passport-facebook” – Facebook authentication strategy for Passport will be used for user authentication using Facebook.

Data to access and store in database “Users” collection:

- Token – unique Facebook profile identifier.
- Username

Registration should be constrained to user agreeing to private policy that clearly indicates that authenticating via Facebook provides the application access to user’s name.

Google Authentication:

“passport-google-oauth20” – Google OAuth20 authentication strategy for Passport will be used for user authentication using Google.

Data to access and store in database in “Users” collection:

- Token – unique Google profile identifier.
- Username

Registration should be constrained to user agreeing to private policy that clearly indicates that authenticating via Google provides the application access to user’s name.

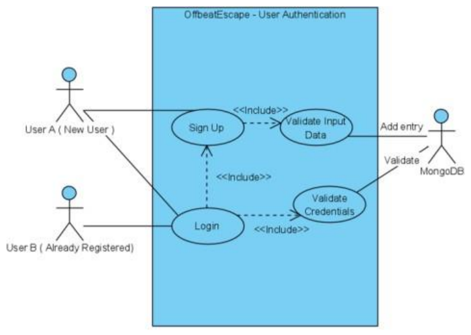


Figure 1 - Use Case Diagram for authentication system

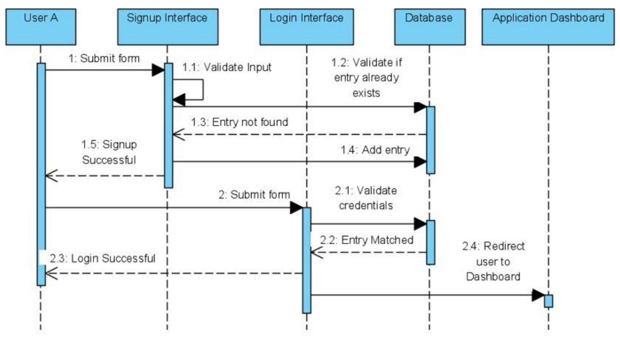


Figure 2 - Sequence Diagram for a New User

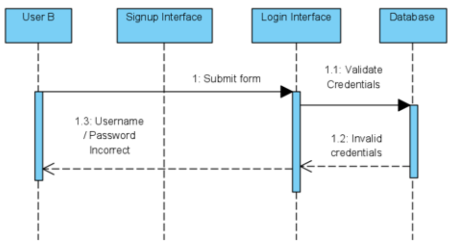


Figure 3 - Sequence Diagram for an unsuccessful Login Attempt

2. Post Configuration – Writer End

Adding / editing / deleting a post & tagging a friend to the post.

ADDING A POST:

- An interface for user to Add or update an existing post.
- Adding write-up should be mandatory, minimum word limit can be added.
- Adding images should be optional for user.
- There should be an option to rate the location of the post.

If "Add Post" is selected from Dashboard >> Post Configuration, display a form for user to add a new post.

Parts of Form:

- Title – The name/headline of the post.
- Description – User story or experience of a location.
- Choose File – Option to attach images to a post.
- Submit Button – To publish the post
- Location – To tag a geo-location
- Rating – for the location

Constraints:

- Headline/Title cannot be left blank.
- Description cannot be left blank.

Flow:

- Entered data in form shall be validated against constraints.
- On clicking Submit, data (title, description, images, location, rating) should be entered to database in posts collection.
- Post shall be visible in "My Posts" section, accessible from Dashboard >> Post Configurations.

UPDATING/EDITING A POST:

- Both write-up and images should be editable.
- Rating for the location must be editable.
- If "My Posts" is selected from Dashboard >> Post Configuration, all the posts published by user. Against each post there will be option to edit & delete.
- Edit form will be exactly like Add form and will have same constraints applied.

- When submitted, data shall be updated in DB.

DELETING A POST:

- Option should be provided for each post in “My Posts”.
- If “My Posts” is selected from Dashboard >> Post Configuration, all the posts published by user. Against each post there will be option to edit & delete.
- If delete is clicked, post will be removed from database and “My Posts”.

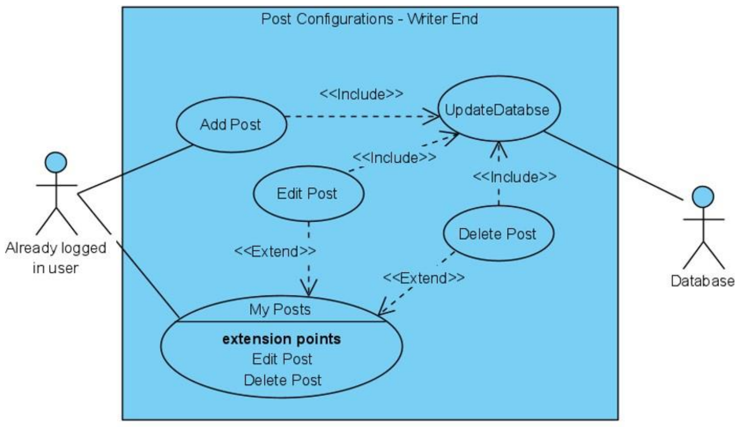


Figure 4 - Use Case Diagram for Post configurations - Writer End

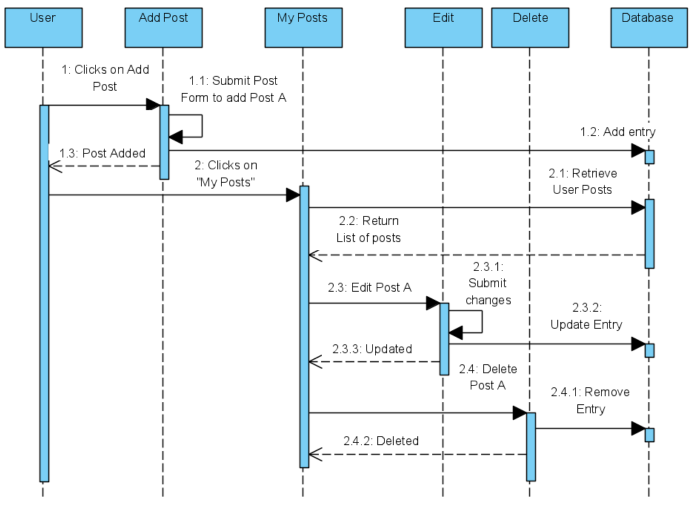


Figure 1 - Sequence Diagram for Post Configuration - Writer End

3. Post Configuration – Reader End

A user can read his friend’s posts through both Dashboard >> Trending Posts & Dashboard >> General Feed. For every post available for reading/viewing, user would have option to save, comment and report a post. When a post is clicked to open it, it will be displayed on a new page with options to save, comment and report it.

SAVE A POST:

- Every viewable post will also be saveable, for future reference or read.
- Every post saved shall be accessible from “Reading List” grid on dashboard.
- Post unique id will be saved to database to keep track of posts.
- In MongoDB, object for user will have set of saved posts linked to it.

COMMENT ON A POST:

- User can add comment to a post.
- Add a comment text box shall be available for user to add a comment on post.
- Recent 3 comments shall only be visible when reading a post.
- Comment will be stored in DB, in the Posts collection.

REPORT A POST:

- User-Only Moderation feature – will rely on users to filter out inappropriate and irrelevant content.
- User shall be able to mark a post as spam/abusive.
- Application shall automatically hide the post if it is reported several times.
- On click, post will be marked as reported in database.
- A counter will be stored to record the no. of times a post has been reported.
- A post reported by 10% of the users will be automatically hidden and considered abusive/spam and author will be notified.

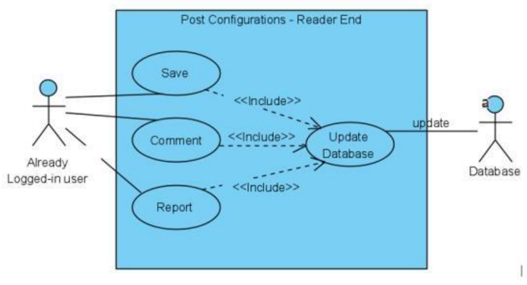


Figure 6 - Use Case Diagram for Post Configurations - Reader End

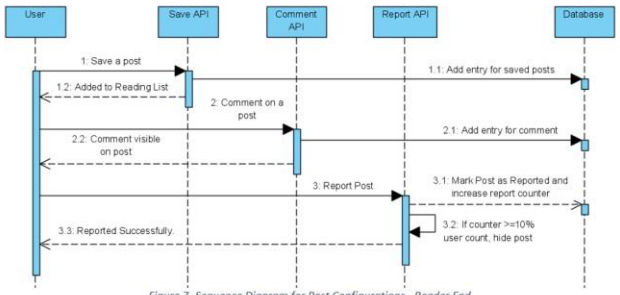

User
Save API
Comment API
Report API
Database
1: Save a post
1.1: Add entry for saved posts
1.2: Added to Reading List
2: Comment on a post
2.1: Add entry for comment
2.2: Comment visible on post
3: Report Post
3.1: Mark Post as Reported and increase report counter
3.2: If counter >=10% user count, hide post
3.3: Reported Successfully.

Figure 7 -Sequence Diagram for Post Configurations - Reader End

4. *Finding and Adding Friends*

- A user can send friend request to a person to add them to their friend list for keeping an update of their stories and experiences while sharing his own with them.
- In case, User A adds no friend, application will act as a private journal for user A. Other users will be able to find posts of user A only if they are marked public.
- A user can only add people already registered on application as friends.
- Option to migrate connection list from Facebook/Google
- When the user goes to the add friends page, he will see the list of existing users in the application.
- When he clicks on add friend, a request is sent to the befriended user for accepting/declining the request.
- When searching for friends, search string will be compared with registered users only.
- Search will work for both full and partially entered data.
- In Database, a Friends set will be stored in Users collection. This friend set shall contain unique user object ids for the friends.
- 3 APIs will be implemented,

- search – to search friends
- friends – to get list of friends
- acceptFriendRequest – to accept a friend request and update the friend list in db.

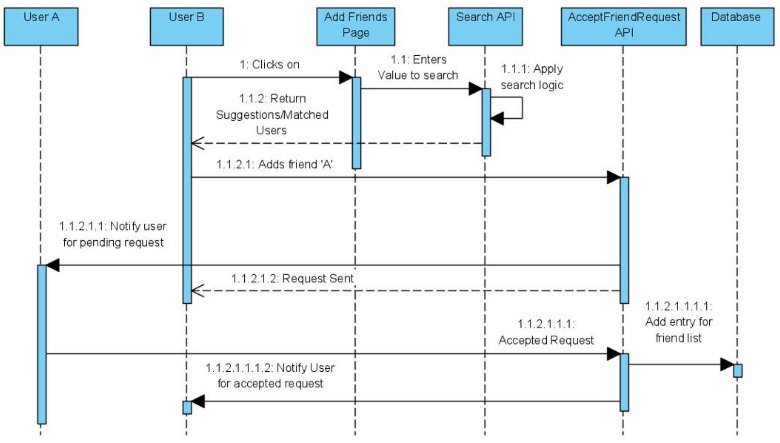

User A
User B
Add Friends Page
Search API
AcceptFriendRequest API
Database
1: Clicks on
1.1: Enters Value to search
1.1.1: Apply search logic
1.1.2: Return Suggestions/Matched Users
1.1.2.1: Adds friend 'A'
1.1.2.1.1: Notify user for pending request
1.1.2.1.2: Request Sent
1.1.2.1.1.1: Add entry for friend list
1.1.2.1.1: Accepted Request
1.1.2.1.1.1.2: Notify User for accepted request

Figure 8 - Sequence Diagram for Add Friend API

5. Inspiration History

Inspiration cycle will be visible on frontend when and if an already saved post is open up by user to view.

API shall return inspiration history for a post saved by user.

User Story :

User A created post - 1

User B saved it.

Now if user B opens the post, inspiration cycle shown will be B->A i.e. B was inspired by A

User C is notified "B saved a Post" and he clicks the notification and saves the post too.

Now if user C opens the post, inspiration cycle shown will be C->B->A i.e. C was inspired by B and so on.

For api to work, following payload is needed :

user : for which cycle is to be fetched.

owner : post owner

title : post title

If user C is logged in, and has saved this post, he shall always see the cycle for himself.

6. General Feed

- All posts from Friends shall be displayed in "General Feed" Grid.
- Only 5 posts shall be visible on 1 page.
- Posts shall be visible in descending fashion as per there publish time, i.e., recent post will be visible on top of list.

7. Trending Posts

- The top 6 posts shall be displayed on dashboard "Trending List" Grid.
- No. of saves will decide the rank of post in trending list.
- Constraint - Posts from friends will only be ranked.

8. Reading List

- Any post saved by user shall be visible in and accessible from "Reading List" Grid.
- Grid will be limited to 5 elements per page.

Non-Functional Requirements

Security

Following features will ensure security at application end:

- User can view, comment, save or report posts of a connection only.
- Social logins work with o-auth tokens and abide by google and facebook's policy.
- All restful services are private and work with a handshake-based cookie system which are decrypted at backend.
- Multifactor Authentication (FE)
- Encoding and sanitizing user input to ensure safety from XSS.
- Private Policy for Google/Facebook authentication.

Market Analysis

How travel software's work till date:

- Majority of the travel-oriented software till date, work on the model of educating users.
- Primarily the focus of these sites is only to post content about a location and let users decide what to do with that content.
- Their market base is only limited to users who are planning to travel recently or users who like to explore other places as a hobby.
- While posting content is a great way to inform users and educate, it still lacks the necessary motivation users need to travel to a certain place.

How OffbeatEscape shall work:

- A social media designed only for travellers filtering everything else. Brings in more dynamism then just sharing content and posts.
- Posts are read by other users and the inspiration travels across the network.
- Creates a trending posts list motivating user to travel and write posts as well.
- Avoid AI based recommendation and get inspired for your travel by other verified users and friends on the network.

References and Market Data supporting above claims:

- https://www.makeuseof.com/tag/6-best-social-media-apps-travelers/#:~:text=Travello&text=Travello%20is%20a%20social%20media,and%20people%20interested%20in%20traveling.&text=If%20you%20want%2C%20you%20can,groups%20of%20like%2Dminded%20travelers.
- https://www.quora.com/Whats-the-best-social-network-for-travel-and-why
- https://www.usatoday.com/story/travel/roadwarriorvoices/2016/02/16/travel-social-networks/80448712/

Technologies & Tools

- AngularJS framework will be used for frontend development.
- ExpressJS framework will be used for API development.
- MongoDB will be integrated with ExpressJS.
- TravisCI will be used for CI/CD.
- GitHub will be used for version control.
- NGINX might be used for web server load balancing.

Cost Analysis

TravisCI (CI/CD): $249/month
GitHub (Version Control System): $21 per user/month
MongoDB: $57/month
Jira: $14 per user/month
AWS EC2 hosting service: $328/month
AWS S3 storage service: $1 to $100/month (depending upon load)
SSL certificate cost: $10 - $1000/monthly
Maintenance: $500-$12,000/year (Regular updates and fixes)
Domain Name: $15/year
Single Resource Cost: $45/hr (considering average software engineer pay in Canada)
Working hours per week: 30
Cost for 5 Resources: $54,000/ 3 months
NGINX