**TripAdvocate – a website about places with user-submitted reviews.**

**Background**

This is a travel-related website with users, places, and reviews, undertaken as an exercise to master the quintessential skills every full stack developer at a minimum is expected to know, which is controlling a back-end database through the front-end web browser. Users log in and use websites every day without giving much thought they are indirectly making database entries. They search for content, create, update, and delete entries effortlessly. And this process should be easy. Making this user experience as seamless as possible and making the front and back ends talk to each other is the goal.

The project is meant to seen by future employers and clients that are not necessarily experienced in IT to know the developer’s capacity to do this type of work, leading to career advancement for the developer.

**Benefits**

The app allows users to make educated decisions to have the best travel experiences about knowing what to expect before they go eat at a restaurant, stay at a hotel, or participate in an activity. It’s an online critique board and costs nothing to the user.

User can read other user’s reviews, and make their own reviews. Administrators can add/delete/edit new places and edit/delete reviews. Each user can have their own account, can view the reviews they’ve made in their account profile.

**Key Features**

What are the key features/functionality that your app will provide to users?

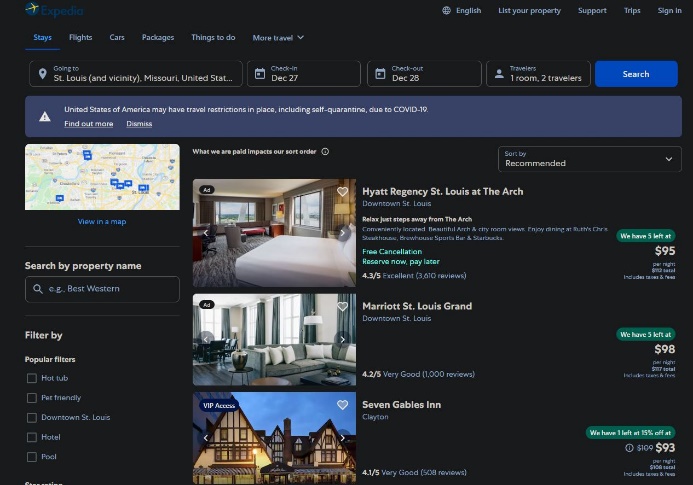
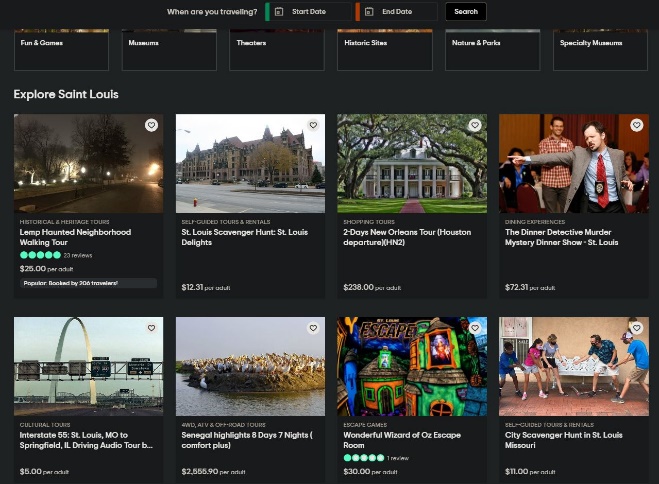
* Users will be able to…
  + View information about places, locations, descriptions, and reviews by others about those places.
  + Create their own reviews on places.
  + Upload photos.
  + Register for an account.
  + Change their account password.
* Admins will be able to do all the actions a user can do, plus
  + Maintain the integrity of the database through removing inappropriate content for any user.
  + See all user’s data (other than passwords), update user information, and delete users.

**Similar Applications**

This app closely mimics TripAdvisor. Another travel-related website is Expedia which has a slightly different focus, mostly hotels and flights instead of activities.

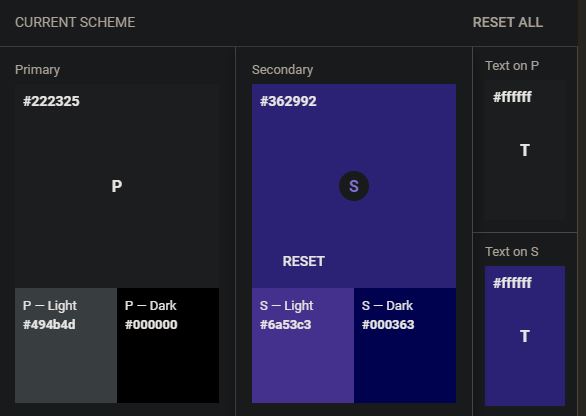
**Expedia**

**TripAdvisor**



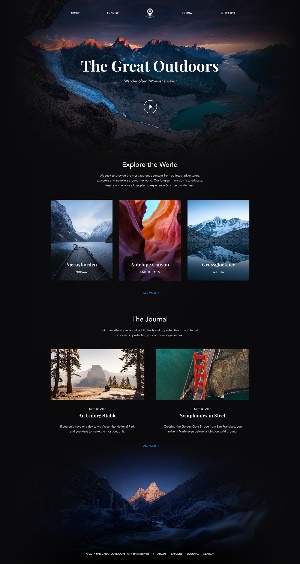
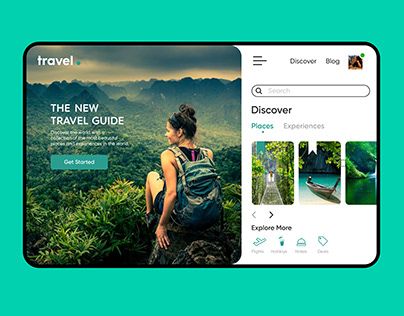
**Color Palette**

Include your color palette below. Example below created with <https://material.io/resources/color/>



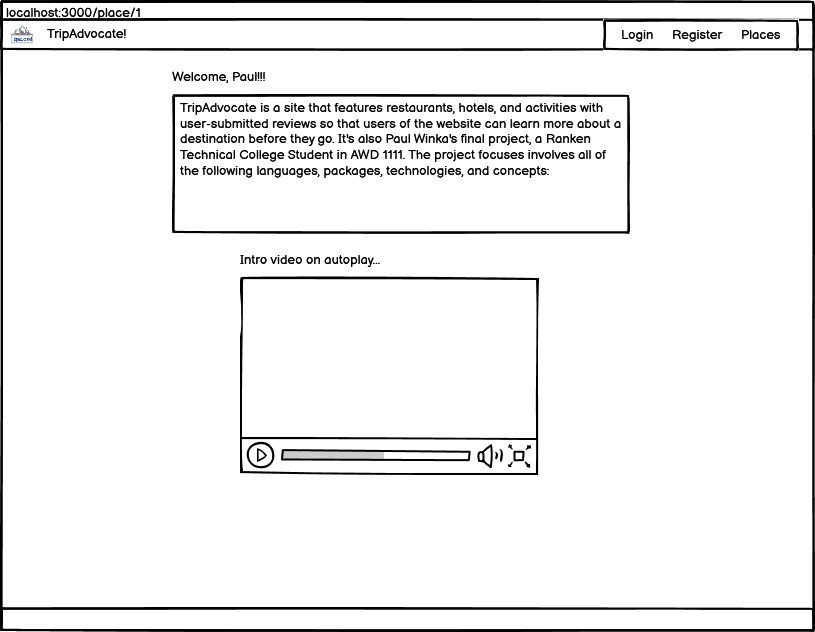
**UI Design Inspiration**

The rich experience here shows the potential for the app once the basic foundation is built.



**Feature - Home Screen**

Home Screen



**Purpose and Background**

The home screen explains what the website is and how to use it and that it’s under development for now. Users may suspect the app works like TripAdvisor, but may not know for certain. Also, for a richer experience, there is an autoplayed video of the head developer. The user can go on from here to login to the website.

**Workflow**

To use the home screen, the user would simply visit the root of the website and view the content.

**Possible Errors & Edge Cases**

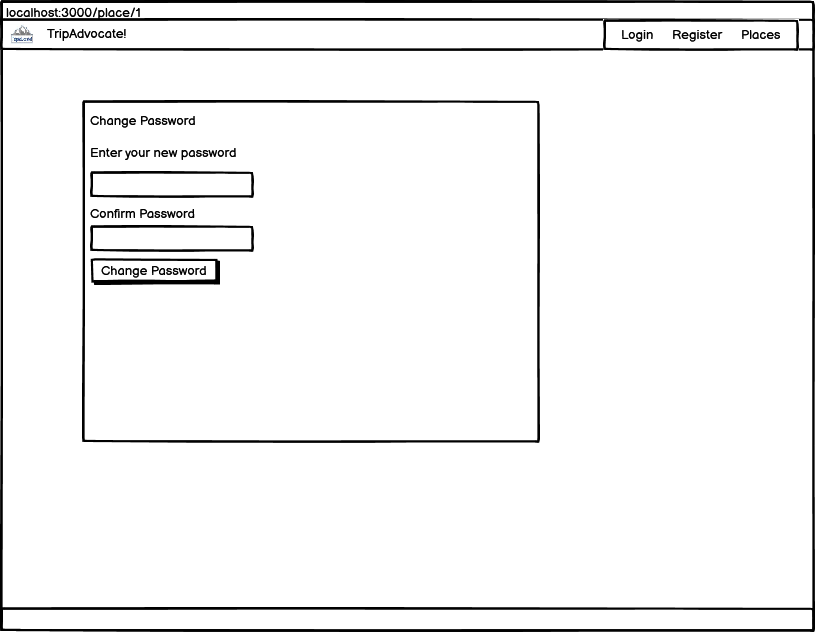
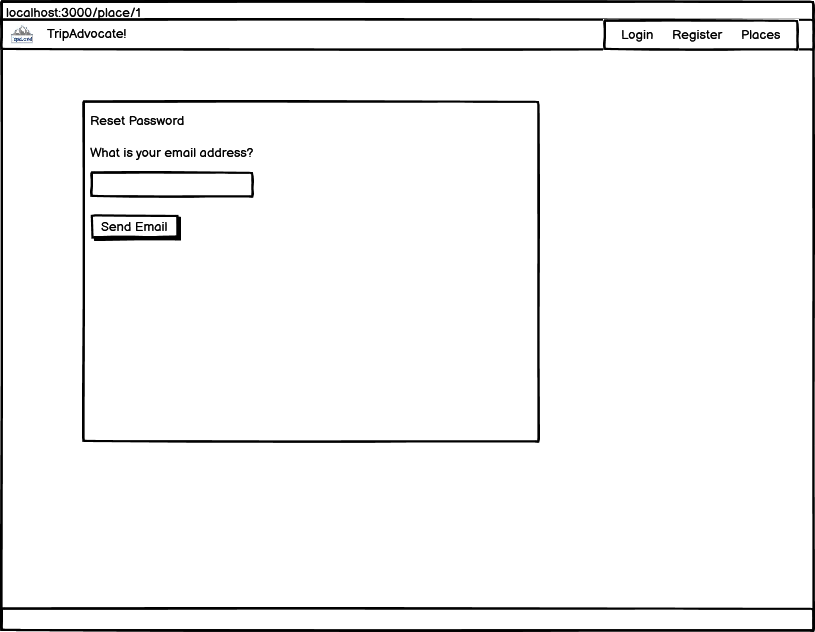
Possible errors would be the video not autoplay or loading properly. The text should load nearly every time. The home screen is visible if the user is logged in or not.

**Error Handling**

Error handling not needed for this portion.

**Feature – registration and logging in**

|  |  |
| --- | --- |
| Screen A | Screen B |

****

**Purpose and Background**

What will this feature allow users to do?

Why do they need this functionality?

Why is this feature important to the project?

Explain the purpose and background of the feature.

The user can create an account to be able to create reviews on places. Also will be able to edit and delete their reviews.

**Workflow**

If the user does not have account, they would first register for an account at ‘/account/register’ by filling out a form.. The registration form will check to make sure there are no duplicate usernames or email addresses. Once the user has a unique username and email and submits the form, the user will receive an email address which will confirm their account through clicking a link they in email using “sendgrid” technology. This verifies the account. From here, the user is fully registered.

The user can then visit ‘/account/login’ to login in to the website using their username and password.

If the password is forgotten, there is a link at the bottom of the login page to allow the user to reset their password by entering their email address. Sendgrid then sends the user another email that takes the user to a link to that allows the change the password.

**Possible Errors & Edge Cases**

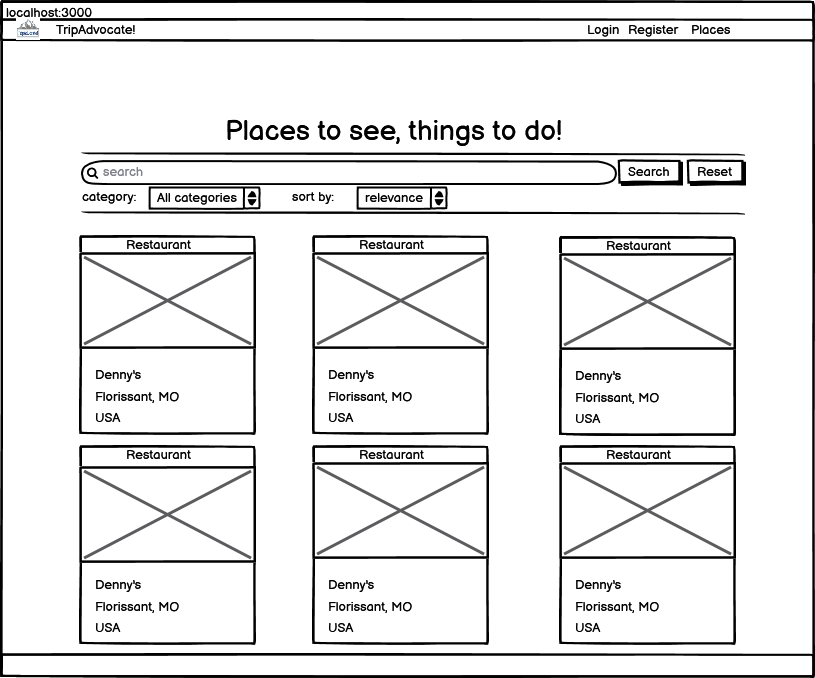
The most common problem with account management is that the user will not receive the sendgrid email that verifies the account. These emails tend to fall into the junk or spam folder of the user’s email account, which is a poor user experience.

Sendgrid also has been known to fail when it encounters requests from new IP addresses. During the development of account management, this continue to the most common point of failure.

**Error Handling**

The user is informed with the Bootstrap “text-danger” class when passwords do not match, usernames or emails are already in the database, or the correct username and password are not used in logging in through queries. Try-catch blocks also ensure the app does not crash.

**Feature – Viewing places and searching them.**



**Purpose and Background**

Searching the database of places is essential for the user to quickly find places that interest them. Search is combine on all text fields of the place’s name, city, state, country, and description. Also, users can filter the places with drop downs. Also, the places are displayed in a gallery that shows the place’s name, image, and location. Bootstrap classes were added to maintain a consistent look to how each item in the image gallery looked. Search appears on other areas of the site as well in a similar fashion.

**Workflow**

Users will visit ‘/place’ and see all the places listed. At the top of the page is the search interface that has three parts. First is a text input, and then a dropdown box for each “all categories” to search among hotels, restaurants, and activities, and one more dropdown that allows for sorting the results based on relevance (based on an mongo db indexing of text fields), name, country, and category (with the latter three being sorted alphabetically). The user manipulates these controls and clicks on send. The dropdowns automatically press the send button on change.

**Possible Errors & Edge Cases**

The search queries not working was the most common error in development. That is, the searches appeared to be valid did not change the search results.

Some parts of the search may be seen as erroneous by a user. A query such as ‘bulg’ will not have “bulgaria” in the results, which may be seen as some users as a broken search feature. Autocomplete also not implemented yet.

**Error Handling**

The ‘.fail’ of the ajax call will pop up an alert to the user if there is a database failure. However, this error did not appear in development and is not expected in the production version. Errors are simply that the database does not submit the query. However, if the user loses their internet connection, this will break the connection to the database, but then the user wouldn’t be able to use other parts of the website either.

**Feature - Purchasing (enter your feature name here)**

|  |  |
| --- | --- |
| Screen A | Screen B |

**Purpose and Background**

What will this feature allow users to do?

Why do they need this functionality?

Why is this feature important to the project?

Explain the purpose and background of the feature.

**Workflow**

How will the user interact with the UI?

What is the user workflow?

Provide step by step instructions.

**Possible Errors & Edge Cases**

What are the possible edge cases?

What could cause an error/exception?

Consider errors that may arise from user error, software glitches, hardware glitches, and networking problems.

**Error Handling**

How will the user be notified of errors/exceptions?

How will the software handle the errors?