Travelwise Technical Report
10/20/2020

Motivation

Travelwise is a web app that enables consumers to make ideal travel plans. Through the use of various RESTful APIs, Travelwise is an ideal platform for users to compare destinations by safety, affordability, and more. A feature of Travelwise is providing users with up-to-date statistics on COVID in a given region. Travelwise streamlines the process of making travel plans, where users can search for destinations, corresponding flights, and hotels all in the same place.

User Stories

10/20/2020

User Story 1:

- *"As a user of Travelwise, I would like to see a footer on the website. It would make the website look a little more visually appealing. It would also make it slightly easier to navigate."*
- Approach:
    - We have not made a footer because our front-end is still working on the general design of our website. Once we have a general consensus on what our website should look like, we will implement a footer if it fits our UI/UX.
- Estimated Time:
    - 30 mins
- Implementation:
    - Implemented
- Actual implementation time:
    - 30 mins

User Story 2:

- *"As a customer of Travelwise, I would like to see a more visually appealing landing/home page. Since this is the first page users will see when they go to Travelwise, I think it's important to make it professional to establish a level of credibility. I think it should include some sort of media whether it's an image or slideshow."*

- Approach:
    - Added pictures to the landing page, still working on a theme for the website itself so for now that will be our visual until we redo the entire front-end UI/UX. We were more worried about getting functionality in this phase.
- Estimated Time:
    - 30 minutes- a few days for the entire website redesign
- Implementation:
    - Implemented (kind of)
- Actual implementation time:
    - 30 minutes

User Story 3:

- *“As a user of TravelWise, when I am looking at flights, I would like to know the city that corresponds with the airport that is displayed. At the moment, the flights display the abbreviations of the airports. When flights arrive or depart from more obscure locations, it will be difficult for me to tell what city corresponds to what airport.”*

- Approach:
    - Our current models display the city correlated with the airport instance.
- Estimated Time:
    - 2 days
- Implementation:
    - Implemented
- Actual implementation time:
    - 1 day

User Story 4:

- *“As a customer of TravelWise I would like to know more information about the different country scores. Currently, the website lists different numbers for different categories for each country. However, I don't really know what those numbers mean in the big picture. I can't tell what is a relatively good score and what is a relatively bad score. If you could add some clarification then the information would be more digestible.*
- Approach:
    - Scores are threat levels out of 100 against certain populations such as women or rating of dangers (high medical score means bad medical care, high theft score means a high amount of robbing). Ratings are done by several grading organizations (is what the API creator says). We'll improve clarification on the website once dynamic pages are set up.
- Estimated Time:
    - 1 hr, once figure out how we want to lay everything out, we can add a description area/pop-up
- Implementation:
    - Not implemented yet
- Actual implementation time:
    - NA

User Story 5:

- *“I would like to see a bit more on the instance pages for cities. Add some more media and information other than what is presented on the model page."*

- Approach:

    - We will be adding media soon once we find a good flag API that works with our current database, will probably use unsplash for our image assets, and add a flag for a covid visual
- Estimated Time:
    - 2 weeks
- Implementation:
    - Not implemented yet
- Actual implementation time:
    - NA

User Stories

9/29/2020

User Story 1:

- *“As a customer of Travel Wise I would want to know the number of COVID cases in the country of my destination. This is because my travel plans are now affected by the number of people infected in my destination. The effects of a greater infection rate could mean more restrictions on places I’m allowed to go as well as a higher risk of catching the disease and then having to spend 2 more unplanned weeks at my destination.“*

- Approach:
    - We will have destination city instance pages link to associated COVID stats instance pages of the country/region.
- Estimated Time:
    - 3 days
- Implementation:
    - Each city now has a link to it’s COVID stats showing things such as number of new cases.
- Actual implementation time:
    - NA

User Story 2:

- *“As a user of Travelwise who may have a flight with layovers, I would like to know how much time there is in between connections. Information about the layovers for a flight is helpful because I may want to avoid a flight if it has layovers that are too short or too long. This information will help me weigh the pros and cons of the different flights listed on TravelWise and will improve my browsing experience.”*

- Approach:
    - Currently, travelwise displays arrival and departure times, enabling users to manually calculate layover times. However, for superb user experience, we will calculate layover times and clearly display them.
- Estimated Time:
    - 2 weeks. We first must get an operational backend to make this feature.
- Implementation:

    - The models have been redesigned to only airports, covid, and cities for now.
- Actual implementation time:
    - NA

User Story 3:

- *"As someone who has had to travel on a tight budget before, I think in addition to having hotel and flight prices sortable, it would be nice to filter them by range. Other services like Google Flights have this feature and it would make sense to include this functionality on your website. Overall, this would make browsing easier."*

- Approach:
    - We can implement this once we have implemented our database. We will add price range as a searchable attribute, allowing users to specify a lower and upper price.
- Estimated Time:
    - 2 weeks. We first must implement an operational database to make this feature.
- Implementation:
    - The models have been redesigned to only airports, covid, and cities for now.
- Actual implementation time:
    - NA

User Story 4:

- *"Since safety seems to be important to the application, on top of covid stats at the destination, if possible, I would like to see the covid procedures implemented by the flight company that I end up choosing. Companies that have very loose procedures may not be seeing high customer traffic right now. By providing the user this information, it covers one more area of concern regarding COVID."*

- Approach:
    - This seems like a difficult thing to implement. It is a very reasonable concern, so we will attempt to connect users to resources where they can find out this information. The obstacle is that it would be hard to scrape or store this type of information into our database. So our goal will be to be able provide external links to a flight's airline company website, where users can find such information.
- Estimated Time:
    - 3 weeks. We first must implement an operational database to make this feature and then we must find a way to associate company websites to given flights.
- Implementation:
    - There is not really a feasible way to find the COVID procedures of every flight corporation. Also the models have been redesigned to only airports, covid, and cities for now, but the safety scores for each city will still be there to provide secure travels.
- Actual implementation time:
    - NA

User Story 5:

- *“As a customer of Travelwise, I am unsure of the functionality of the dropdown under the search bar on each model page. I believe the intention is for the selected field to be the column each table is sorted by, although this is a guess. It would be beneficial to users of this site if the purpose of the dropdown was labeled, or if the default text in the dropdown indicated its purpose, such as "Order columns by:"*

- Approach:
    - We will likely change the default text to be more descriptive, such as “order by safety.”
- Estimated Time:
    - 1 day
- Implementation:
    - We created buttons to sort by most categories.
- Actual implementation time:
    - NA

RESTful API

For destination information (cities), we use GeoDB cities API:

- https://rapidapi.com/wirefreethought/api/geodb-cities?endpoint=5990a0b4e4b075a0d1d6da26

For destination safety scores, we use Amadeus's Safe Place API:

- https://developers.amadeus.com/self-service/category/destination-content/api-doc/safe-place-api/api-reference

For COVID statistics on given regions, we use COVID19 API:

- https://documenter.getpostman.com/view/10808728/SzS8rjbc

For airports, we use Amadeus's Flight Searcher API:

- https://developers.amadeus.com/self-service/category/air/api-doc/flight-offers-search/api-reference

For hotels, we (will) use Amadeus's Hotel Search API:

- https://developers.amadeus.com/self-service/category/hotel/api-doc/hotel-search/api-reference

Models

For this second phase, our three models are Cities, Airports, and COVID regions.
Lower safety scores correspond to a safer environment. Scores are 1-100, 1 being unlikely to suffer from the metric, to 100 being very likely to suffer from the metric.

- **Destination Cities**
    - City ID
    - Name
    - Country
    - Country Code
    - Region
    - Longitude
    - Latitude
    - LGBTQ Safety Score

    - Medical Safety Score
    - Overall Safety Score
    - Physical Harm Safety Score
    - Political Freedom Safety Score
    - Theft Safety Score
    - Women Safety Score
- **Airports**
    - Iata code
    - Name
    - City Name
    - Country Code
    - Country Name
    - Time Offset
    - Longitude
    - Latitude
- **COVID Statistics**
    - Country Code
    - Country
    - Total Cases
    - Total Deaths
    - New Cases (daily)
    - New Deaths (daily)

Tools

We are using a variety of tools:

- **Docker:** We will use docker to create a docker image that packages our tool-chain and dependencies into one convenient container.
- **Postman:** We have used Postman to scrape our initial instances of our models. We have used Postman to create a shared workspace, making it easy to collaborate with team members when working with our RESTful APIs. We also have used Postman to design Travelwise's API. Down the road, we will use Postman to further design and implement our API, once we have a more functional backend.
- **React.js:** We use React.js for our front end.
- **Bootstrap:** We use Bootstrap as our primary CSS framework.
- **Yarn:** We use Yarn, a package manager, for our React App. It makes documenting and installing our dependencies easy and streamlined.
- **Visual Studio Code (LiveShare)**: We utilize Visual Studio Code in developing our React App. We also utilize the LiveShare extension, allowing real-time code collaboration and pair programming.
- **GitLab:** We utilize Gitlab for our version control.
- **Slack:** We've integrated slack to our GitLab repo, for communication purposes and issue tracking.

Hosting

- **Namecheap:** We used namecheap to buy our domains because they support whois protection and have cheap domain names.
- **AWS Amplify:** AWS Amplify allows us to host websites by simply committing and pushing code to our Git Repository. The service allows us to focus on simply writing code and not having to directly interact with our build server/terminal.

- **Create-React-App:** The Create-React-App allows us to have a modern build setup with pre-set configurations that make our react app compatible with most technologies. It is officially supported by Facebook, the creators of React so it is extremely reliable.

Database

- **Postman:** documentation of our api: https://travelwiselive.postman.co/collections/12799472-e9e5eb99-d6fe-49df-9435-6f8fc6c7e0d6?version=latest&workspace=4e45581b-ddf8-4fd5-82b8-31c31b43a9b5
- **Flask-restless:** Used to connect database to server
- **Flask-SQLAlchemy:** Used to operate on our database in flask
- **PostgreSQL:** Our relational database management system
- **pgAdmin4:** The GUI we used to construct the database.
- **PlantUML:** Used for UML diagram
- **Tables:** Cities, airports, countries. All tables have columns that are also a column of other tables. Example of city, airport, and country in order.
- **NOTE**: We weren't very familiar with SQL datatypes going into this, and we discovered (before we had enough time to fix it) that the way we stored strings into the database were actually arrays of strings. What this meant is that when we processed the data as a json, we had to peel off the array characters to properly display the content. This is shown on every line with a string surrounded by the following expression: [“”]. We have to rescrape all the API's to fix our data, so we plan to save that for phase III of the assignment.
- city_id: 3100212
- country: ["France"]
- country_code: ["FR"]
- latitude: 48.8412
- lgbtq: 0
- longitude: 2.3003
- medical: 0
- name: ["15th arrondissement of Paris"]
- overall: 41
- physical: 0
- political: 28
- region: ["Île-de-France"]
- theft: 0
- women: 0

Back-end

- **Flask:** Backend framework
- **AWS Elastic Beanstalk:** Used to deploy flask server
- **AWS RDS:** Used to host our PostgresSQL database
- **Database:** Info above.

Testing

- **Newman:** Used for Postman API Tests, automates tests for each request to API
- **Jest:** Used for Typescript testing, tests if components rendered correctly
- **Selenium:** Used for GUI testing, tests user interaction
- **Unittest:** Python extension used to test backend in yml pipeline
- **Docker:** Used for CI

Pagination

React-Pagination: Handles the logic of rendering a pagination component, since our databases are currently small, we make one fetch to the back-end and paginate the data when someone loads up one of our model pages.
Bootstrap: Handles the design of our pagination component, might customize later with our own CSS