CS 411 Final Project Report

Changes Since Initial Proposal

The main goals and ideas behind the project have remained unchanged since the original poster. The goal was to create a user friendly travel planning app and that is the final deliverable for this project. Some slight changes have been made to the exact scope of the project based on data availability, technical feasibility, and time constraints. Exact details of these changes will be discussed in more detail in subsequent sections of this report.

Application Achievements, Success, and Failures

We believe that our application succeeded in creating a simple and user-friendly way for users to plan vacations and discover attractions that they find interesting. The preferences of the user are recognized over time by the app based on previous itineraries that were put together by the user; this helps the user easily sort by attractions that are most highly recommended for them. Our app also allows users to easily plan all their trips and save them for later all within the app for convenience. Users can look back at past itineraries and plan their next trips based on what they did and didn't like from their last trip.

One place that our app fell short of our original ambitions was in planning out the exact order and logistics of traveling between attractions during a trip itinerary. We recognized a need for a travel planner to also plan routes and help users determine the most optimized route to see all of their chosen attractions and how transportation may work out between these attractions. Because the application lacks these advanced planning features in its current state it ended up being more similar to existing travel planning apps and websites. Through this project we found it difficult to implement these features because of limited data on the times that should be spent at a particular location as well as a lack of available and free data about transportation and lodging. Additionally, the algorithms needed to determine the optimal routes with all the different variables could quickly become complicated and implementation could exceed the time allocated to this project. Ultimately, it became clear that these reasons among other may contribute to why there are not travel planners with these kind of feature widely available and free to the public.

Modifications of Database Schema and Data Sources

The overall schema of the database remained the same throughout the project, there were five tables. An Attractions Table, an Itinerary table, an ItineraryItem table, a Route table, and a User table. We searched many different sites after our initial proposal to find data that would suit our use case. After compiling a list of potential data sources we thought about how each one would fit best into our planned data schema. We eventually settled on a Kaggle dataset linked here: https://www.kaggle.com/datasets/maedemaftouni/global-tourist-hotspots-us-india-iran

The above dataset was chosen because it already fit the planned schema pretty well. The only changes that would need to be made were to the exact attributes in each table such as users and attractions. Details about the changes to the table implementations are discussed in the next section.

Changes to UML Diagram and Table Implementations

- Discuss what you change to your ER diagram and/or your table implementations. What are some differences between the original design and the final design? Why? What do you think is a more suitable design?
    - We changed the attributes of the user and attraction tables
    - This was necessary to interface with the dataset that we had already available and to avoid unnecessary reprocessing of data to fit the somewhat arbitrary attributes that were originally set up
    - Simply included all the attribute ratings that were available in the data set rather than creating some kind of formula to weight them based on a different set of user preferences

The only major change that was made to the original iteration of the database design was some modifications to the table implementation. The table attributes were tweaked after finding the data set so that the dataset could be loaded and used without a lot of modifications. All of the tables are laid out in detail below.

The Attractions table to hold a listing of all attractions and their attributes which can be seen below. The only changes were to include city and state instead of destination and creating a main category attribute.

```
+---------------+--------------+------+-----+---------+----------------+
| Field         | Type         | Null | Key | Default | Extra          |
+---------------+--------------+------+-----+---------+----------------+
| attraction_id | int          | NO   | PRI | NULL    | auto_increment |
| name          | varchar(255) | NO   |     | NULL    |                |
| main_category | varchar(100) | YES  |     | NULL    |                |
| rating        | float        | YES  |     | NULL    |                |
| popularity    | int          | YES  |     | NULL    |                |
| description   | text         | YES  |     | NULL    |                |
| address       | varchar(255) | YES  |     | NULL    |                |
| city          | varchar(100) | YES  | MUL | NULL    |                |
| state         | varchar(100) | YES  | MUL | NULL    |                |
+---------------+--------------+------+-----+---------+----------------+
```

Next, was an Itinerary table which holds all of the itineraries created by users. The attributes of this table can be seen below as well. Destination was replaced with destination_city and destination_state and the start and end times were removed from the original model.

```
+-------------------+--------------+------+-----+---------+----------------
| Field             | Type         | Null | Key | Default | Extra
+-------------------+--------------+------+-----+---------+----------------
| itinerary_id      | int          | NO   | PRI | NULL    | auto_increment
| user_id           | int          | YES  | MUL | NULL    |
| start_date        | date         | YES  |     | NULL    |
| end_date          | date         | YES  |     | NULL    |
| destination_city  | varchar(100) | YES  |     | NULL    |
| destination_state | varchar(100) | YES  |     | NULL    |
+-------------------+--------------+------+-----+---------+----------------
```

After that is the ItineraryItem table which holds each item that is part of a given itinerary the attributes are listed below. This table did not change at all since the first iteration.

```
+---------------+----------+------+-----+---------+----------------+
| Field         | Type     | Null | Key | Default | Extra          |
+---------------+----------+------+-----+---------+----------------+
| item_id       | int      | NO   | PRI | NULL    | auto_increment |
| itinerary_id  | int      | YES  | MUL | NULL    |                |
| attraction_id | int      | YES  | MUL | NULL    |                |
| day_number    | int      | YES  |     | NULL    |                |
| start_time    | datetime | YES  |     | NULL    |                |
| end_time      | datetime | YES  |     | NULL    |                |
| notes         | text     | YES  |     | NULL    |                |
+---------------+----------+------+-----+---------+----------------+
```

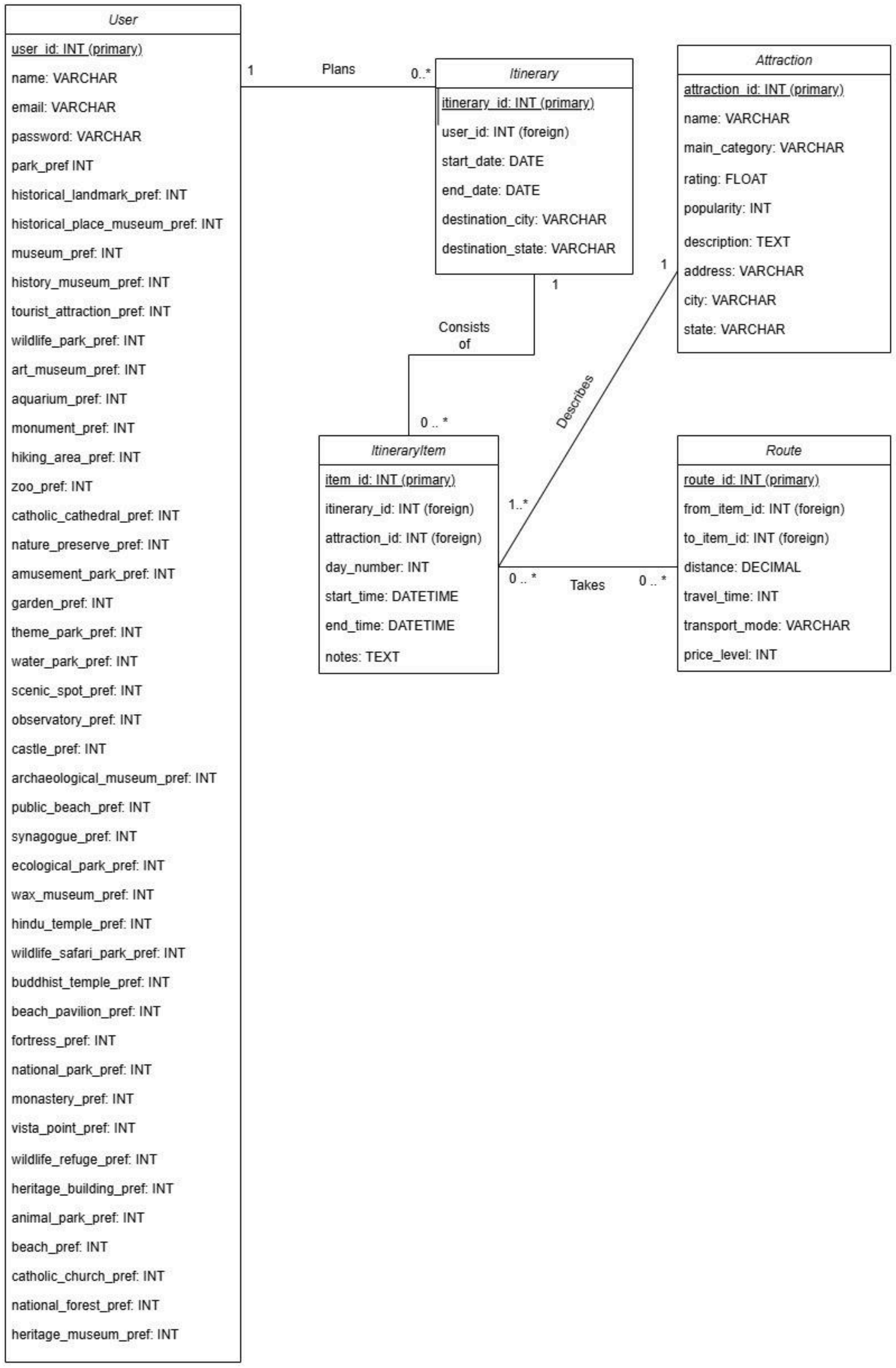
The next table is Route which was not yet utilized in the frontend design. Route is intended to hold transportation means, cost, and duration for each leg of an itinerary. This proved challenging to implement and is part of potential future work on the application. The Route table attributes are below. This table also remains unchanged from the initial implementation.

```
+----------------+---------------+------+-----+---------+----------------+
| Field          | Type          | Null | Key | Default | Extra          |
+----------------+---------------+------+-----+---------+----------------+
| route_id       | int           | NO   | PRI | NULL    | auto_increment |
| from_item_id   | int           | YES  | MUL | NULL    |                |
| to_item_id     | int           | YES  | MUL | NULL    |                |
| distance       | decimal(10,2) | YES  |     | NULL    |                |
| travel_time    | int           | YES  |     | NULL    |                |
| transport_mode | varchar(100)  | YES  |     | NULL    |                |
| price_level    | int           | YES  |     | NULL    |                |
+----------------+---------------+------+-----+---------+----------------+
```

Finally is the Users table which contains basic user information like the user_id, name, email, and password which are created by the user when they create an account. Then, the user preferences are stored and updated based on a trigger in the app that updates learned preferences from the users own selections. The preferences are all stored as an INT from 1-10. All of the different preferences stored can be seen below. Substantially more preference categories were added to the user table both to increase the accuracy of the recommended attractions but also to match the categories in the attractions data source.

```
+-----------------------------+--------------+------+-----+---------+----+
| Field                       | Type         | Null | Key | Default |Extra
+-----------------------------+--------------+------+-----+---------+----+
| user_id            | int           | NO   | PRI | NULL | auto_increment |
| name                        | varchar(255) | YES  |     | NULL    |
| email                       | varchar(255) | YES  | UNI | NULL    |
| password                    | varchar(255) | YES  |     | NULL    |
| park_pref                   | int          | YES  |     | NULL    |
| historical_landmark_pref    | int          | YES  |     | NULL    |
| historical_place_museum_pref | int         | YES  |     | NULL    |
| museum_pref                 | int          | YES  |     | NULL    |
| history_museum_pref         | int          | YES  |     | NULL    |
| tourist_attraction_pref     | int          | YES  |     | NULL    |
| wildlife_park_pref          | int          | YES  |     | NULL    |
| art_museum_pref             | int          | YES  |     | NULL    |
| aquarium_pref               | int          | YES  |     | NULL    |
| monument_pref               | int          | YES  |     | NULL    |
| hiking_area_pref            | int          | YES  |     | NULL    |
| zoo_pref                    | int          | YES  |     | NULL    |
| catholic_cathedral_pref     | int          | YES  |     | NULL    |
| nature_preserve_pref        | int          | YES  |     | NULL    |
| amusement_park_pref         | int          | YES  |     | NULL    |
| garden_pref                 | int          | YES  |     | NULL    |
| theme_park_pref             | int          | YES  |     | NULL    |
| water_park_pref             | int          | YES  |     | NULL    |
| scenic_spot_pref            | int          | YES  |     | NULL    |
| observatory_pref            | int          | YES  |     | NULL    |
| castle_pref                 | int          | YES  |     | NULL    |
| archaeological_museum_pref  | int          | YES  |     | NULL    |
| public_beach_pref           | int          | YES  |     | NULL    |
| national_forest_pref         | int          | YES  |     | NULL    |
| catholic_church_pref         | int          | YES  |     | NULL    |
| heritage_museum_pref         | int          | YES  |     | NULL    |
| beach_pref                   | int          | YES  |     | NULL    |
| synagogue_pref               | int          | YES  |     | NULL    |
| ecological_park_pref         | int          | YES  |     | NULL    |
| wax_museum_pref              | int          | YES  |     | NULL    |
| hindu_temple_pref            | int          | YES  |     | NULL    |
| wildlife_safari_park_pref    | int          | YES  |     | NULL    |
| buddhist_temple_pref         | int          | YES  |     | NULL    |
| animal_park_pref             | int          | YES  |     | NULL    |
| wildlife_refuge_pref         | int          | YES  |     | NULL    |
| heritage_building_pref       | int          | YES  |     | NULL    |
| vista_point_pref             | int          | YES  |     | NULL    |
| national_park_pref           | int          | YES  |     | NULL    |
| monastery_pref               | int          | YES  |     | NULL    |
| fortress_pref                | int          | YES  |     | NULL    |
| beach_pavilion_pref          | int          | YES  |     | NULL    |
+------------------------------+--------------+------+-----+---------+-----
```

Next is the final UML updated to the final model.



Functionalities Added or Removed

The main functionality that was removed from our program was the route planning function. This function would help users organize and optimize the routes that they chose to see all of their selected attractions in their itinerary. The route would choose the order of attractions and also select a mode of transportation and associated cost with each route. The reason that we ended up removing this functionality is because it was difficult to get data that was compatible and free and could be used to determine all of the route attributes. Beyond that the algorithms to optimize the route for time or cost would have taken quite a while to implement and were decided to ultimately be outside the scope of this initial project.

Advanced Database Programs

1. Trigger to update user preferences:

A trigger was used to update user preferences and learn from the user's choices to inform their preferences. Each time the user selects an attraction, the attractions that are in the same category have their recommendation match score go up by 1%. This trigger enhances the user experience by allowing the app to automatically incorporate the users preferences without the user needing to manually input anything.

2. Stored procedure for sharing itineraries:

A stored procedure was used to share itineraries to other users. This feature specifically enhances the app and makes it easier for users who are planning a trip together to collaborate on planning. Many prospective users likely look to travel with others, this feature makes the app much more appealing to such users.

3. Foreign and primary key constraints:

The foreign and primary key constraints ensure a smooth user experience, avoiding any data error or confusing query results. The constraints enforce our relationships and auto-update any changes, ensuring data integrity.

Technical Challenges

Throughout the project the team faced several technical challenges:

1. Our database was hacked and the data was locked. The hacker requested bitcoin in exchange for unlocking the data. We were able to easily get around it by having backed the data up with google cloud and simply restoring from that backup
2. We had some brief issues during the first demo getting the backend and front end working together so that we could demonstrate a user logging into the system. The issues simply ended up being a combination of all having the same .env file and adding each user's IP address to the authorized networks on the Google Cloud dashboard for our database.

3. Another technical challenge was definitely working with the networking of the SQL and VM instance. This was my first time working with a VM instance on a cloud, rather than my own computer. Although the setup itself was relatively simple, it was difficult for us to learn to connect our frontend as well as local backend to the actual SQL server on GCP. After lots of googling and searching for the answer, we discovered that we had to allow for specific TCP connections between our own networks and the server, as well as adding multiple stages of authentication to fully enable usage of the SQL instance.
4. This was my (Aansh) first time working with Node.js, which we used for both our frontend and backend, so I struggled a bit in getting it setup. It also made it initially difficult to implement the code because I was not familiar with the syntax, but I got faster through the project.

Changes

The only changes that have been made since the original proposal are the slight changes in the table attributes and the removal of route planning functionality in the app. These changes have been discussed in previous sections already.

Future Work

In the future we would like to make the travel planning app into a one stop shop for all travel planning. The first improvement would be to implement the front end side of the route planning functionality and determining route optimization algorithms for cost or time. This would require more data integration with different possible modes of transportation as well as times and costs for each. The next step would be even further integration with travel partners so that users can choose to book travel through the app such as hotels, plane tickets, train tickets, etc. Eventually we would like to integrate our data with the data of other companies so that our app can serve as a single interface for users to plan, schedule, and book all of their travel arrangements.

Team Division of Labor

Describe the final division of labor and how well you managed teamwork.

Ryan - Frontend/Backend initial setup, Store procedure, CRUD operations, Sharing Itinerary feature, Presentation

Aansh - Attraction data, search functionality, add attraction to itinerary, itinerary date change, new user creation

Darren - Recommendation functionality, Get recommendation stored procedure, preference learning trigger, Presentation

Chason - Written report, User data, Presentation