Architect For Men Technical Design Document

1. Tech Stack

Frontend: Angular 14
Frontend Libraries: Bootstrap, PrimeNg
Backend: Python, Flask
DB: MySQL
Source Code Control: Git
Task Management: Jira
IDE : Visual Studio Code
Cloud: AWS

2. Accounts and Infrastructure

2.1 Development

The frontend and backend will be running locally on individual machines during development.
The DB would be deployed in a Trial AWS account created using a personal account to have data shared across the team members.

2.2 Production

The idea is to deploy all the projects on AWS, on one single EC2 instance.
The current plan is to manually run the instance with the deployment code for each project, once the code is production ready.

3. Data Sources, Models, Timing

3.1 Data Sources

For the application to be working, some customer data is needed in order to trigger event notifications to the users of the respective zip codes.
This data would be fetched from two sources:

- The existing website shopify account. (Note: Zip Code is not mandatory for the account, so data required might not be available here)
- We are developing a Signup sheet which would be shared with the existing customers of Ninocutz, where we can collect the required details.

This signup sheet is expected to be shared by Micheal to his existing customers, to help collect the data.

3.2 Data Models and Structure

Our choice of database is MYSQL due to its high performance and its flexibility of being open source. The database would have a total of 7 tables wherein all the user, events and images related data would be stored. The following are the tables that would be designed in our database:

- Customer: The customer table is used to store customer related details. The columns in this table are:
    - Customer_ID: This is a unique identifier which will serve as the primary key. The customer identifier is generated by the application.
    - First_name: This column is used to store the first name of the customers and it is of CHAR type. The size of the data is limited to 50 characters.
    - Last_name: This column is used to store the last name of the customers and it is of CHAR type. The size of the data is limited to 50 characters.
    - Email_ID: This column is used to store the mailing id of the customers and it is of CHAR type. The size of the data is limited to 50 characters.
    - Zip_code: This column is used to store the zip codes of the customers and it is of CHAR type. The size of the data is limited to 50 characters.
    - Created_On: This column is used to store the record creation timestamp and it is Datetime type.

- RV User: The RV User table is used to store the Admin/RV user details. The columns in this table are:
    - RV_USER_ID: This is a unique identifier which will serve as the primary key. The RV User identifier is generated by the application.
    - First_name: This column is used to store the first name of the RV User and it is of CHAR type. The size of the data is limited to 50 characters.
    - Last_name: This column is used to store the last name of the RV User and it is of CHAR type. The size of the data is limited to 50 characters.
    - Email_ID: This column is used to store the mailing id of the RV User and it is of CHAR type. The size of the data is limited to 50 characters.
    - Address: This column is used to store the address of the RV User and it is of CHAR type.The size of the data is limited to 500 characters.
    - Password: This column is used to store the password of the RV User and it is of CHAR type.The size of the data is limited to 128 characters. We would be storing the password in the hashed format.
    - Zip_code: This column is used to store the zip codes of the RV User and it is of CHAR type. The size of the data is limited to 50 characters.
    - Created_On: This column is used to store the record creation timestamp and it is Datetime type.

    - Created_By: This column is used to store the RV_USER_ID of the user that generated the RV User and it is of INT type.
    - Modified_By: This column is used to store the RV_USER_ID of the user that modified the RV User and it is of INT type.
    - Modified_On: This column is used to store the record modification timestamp and it is Datetime type.

- Booking: The Booking table is used to store the Customer's RV Event bookings. The columns in this table are:
    - Booking_ID: This is a unique identifier which will serve as the primary key. The Booking ID is generated by the application.
    - Customer_ID: This is the primary key of the customer table which will serve as a foreign key here.
    - Booking_date: This column is used to store the booking date timestamp and it is Datetime type.
    - Number_Of_People_Booked: This column is used to store the number of people participating in the booking and is of INT type.
    - Created_On: This column is used to store the record creation timestamp and it is Datetime type.
    - Modified_By: This column is used to store the RV_USER_ID of the user that modified the RV User and it is of INT type.

- Event: The Event table is used to store the Event details created by the RV/Admin user. The columns in this table are:
    - Event_ID: This is a unique identifier which will serve as the primary key. The Event ID is generated by the application.
    - Image_ID: This is the primary key of the images table which will serve as a foreign key here.
    - Event_Type_ID: This is the primary key of the event type table which will serve as a foreign key here.
    - Event_Name: This column is used to store the first name of the event and it is of CHAR type. The size of the data is limited to 50 characters.
    - Latitude: This column is used to store the latitude details of the RV Event and it is of Decimal type. Here the column supports upto 8-6 digits to the left of the decimal point.
    - Longitude: This column is used to store the longitude details of the RV Event and it is of Decimal type. Here the column supports upto 9-6 digits to the left of the decimal point.
    - Address: This column is used to store the address of the RV User and it is of CHAR type.The size of the data is limited to 500 characters.
    - ZIP_Code: This column is used to store the zip codes of the RV User and it is of CHAR type. The size of the data is limited to 50 characters.
    - Event_Date: This column is used to store the event date timestamp and it is Datetime type.

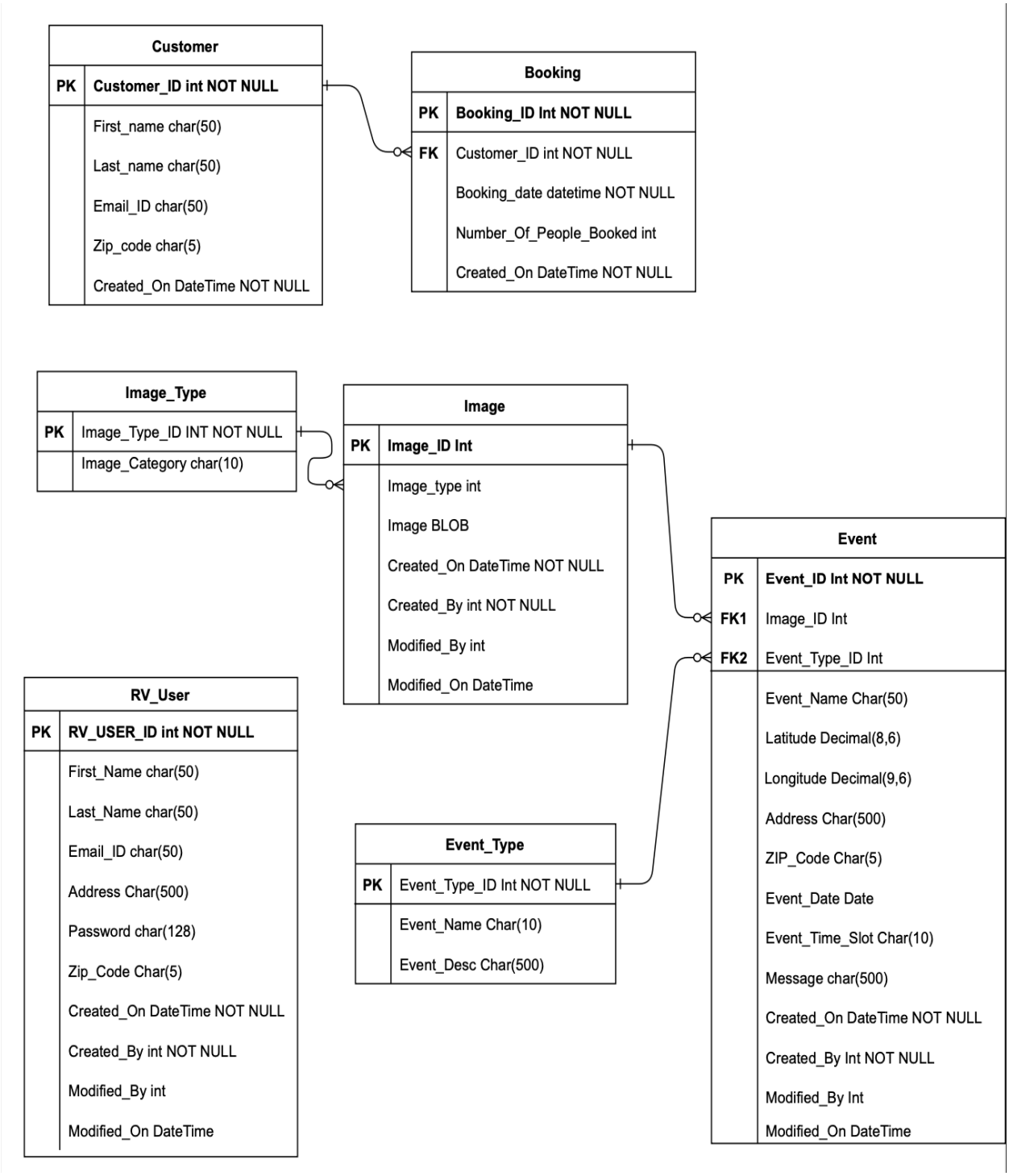
    - Event_Time_Slot: This column is used to store the event timestamp and it is CHAR type.
    - Message: This column is used to store the customized message provided by the RV/Admin User and it is of the type CHAR. The size of the data is limited to 500 characters.
    - Created_On: This column is used to store the record creation timestamp and it is Datetime type.
    - Created_By: This column is used to store the RV_USER_ID of the user that generated the event and it is of INT type.
    - Modified_By: This column is used to store the RV_USER_ID of the user that modified the event and it is of INT type.
    - Modified_On: This column is used to store the record modification timestamp and it is Datetime type.

- Image: This table is used to store the image blobs of events uploaded by the RV/Admin User. The columns in this table are:
    - Image_ID: This is a unique identifier which will serve as the primary key. The Image ID is generated by the application.
    - Image_type: This column is used to store the type of the image i.e casserole or event related image and it is of INT type.
    - Image: This column is used to store the image in the blob format.
    - Created_On: This column is used to store the record creation timestamp and it is Datetime type.
    - Created_By: This column is used to store the RV_USER_ID of the user that uploaded the image and it is of INT type.
    - Modified_By: This column is used to store the RV_USER_ID of the user that modified the image and it is of INT type.
    - Modified_On: This column is used to store the record modification timestamp and it is Datetime type.

- Image_Type: This table is used to store the categories of the images accepted by the application. The columns in this table are:
    - Image_Type_ID: This column is used to store the image type id and is the primary key for the table.
    - Image_Category: This column is used to store the image category and is of the CHAR type.

- Event_Type: This table is used to store the categories of the events accepted by the application. The columns in this table are:
    - Event_Type_ID: This column is used to store the event type id and is the primary key for the table.
    - Event_Desc: This column is used to store the event details and is of the CHAR type.

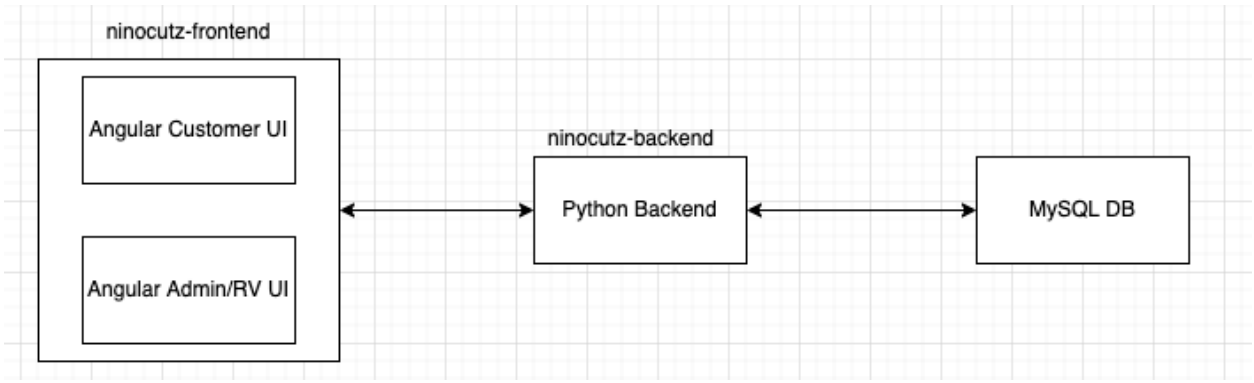
The following is the entity relationship diagram (ERD) depicting the data objects / tables interaction:



3.3 Timing

The data would be persistent in the DB as long as the Admin/RV user deletes them. (Forever if not deleted by Admin)
The customer data would be added as and when the user fills the sign up sheet, it will be used to send the notifications to them based on the zipcode.
If the customer data collected is removed, there wouldn't be any notifications sent to the users, when the RV is available at their location.
If the events data is removed by the Admin/RV user there won't be any information displayed to the customer user in the webpage.

4. System Architecture Diagram



Basic architecture diagram

Ninocutz-frontend: This would include all the components displayed for the customer and the RV/Admin. All components would get the required data from the RESTFUL APIs exposed.

Ninocutz-backend: Contains Python Flask Application which exposes the RESTFUL APIs required for the frontend. Acts as an intermediary for the DB and the Frontend.

MySQL DB: Open Source SQL database management system, used to save the information required for the Application

5. Deployment Methodology

- The existing system is hosted on Bluehost and whereas the new application will be hosted on AWS.
- We would be taking our clients' help to create an AWS account which will be our production environment.
- There would be manual deployment of the project into Production once the code is production ready.
- Seed data if needed would be manually seeded into the production database.