**Full Stack Development with MERN**

**Database Design and Development Report**

| Date | 20-07-2024 |
| --- | --- |
| Team ID | SWTID1720166168 |
| Project Name | Journify |
| Maximum Marks |  |

**Project Title**: Journify   
**Date**: 20 - 07 - 2024  
**Prepared by**: SWTID1720166168

**Objective**

The objective of this report is to outline the database design and implementation details for the Journify app, including schema design and database management system (DBMS) integration.

**Technologies Used**

* **Database Management System (DBMS):** MongoDB
* **Object-Document Mapper (ODM):** Mongoose

**Design the Database Schema**

The database schema is designed to accommodate the following entities and relationships:

**1. Users**

* Username
* Email
* User Type
* Password
* Approval

**2. Flights**

* Flight Name
* Flight ID
* Departure City
* Destination City
* Departure Time
* Arrival Time
* Base Price of Flight
* Total Seats

**3. Booking**

* Flight Name
* Flight ID
* Departure City
* Destination City
* Email
* Mobile Number
* Seats
* Passengers Information
* Total Price
* Booking Date
* Journey Date
* Journey Time
* Seats class
* Booking Status

**Implement the Database using MongoDB**

The MongoDB database is implemented with the following collections and structures:

Database Name: Newdb

1. Collection: users

const userSchema = new mongoose.Schema({   
 username: { type: String, required: true },   
 email: { type: String, required: true, unique: true },   
 usertype: { type: String, required: true },   
 password: { type: String, required: true },   
 approval: {type: String, default: 'approved'}   
});

2. Collection: Flight

const flightSchema = new mongoose.Schema({   
 flightName: { type: String, required: true },   
 flightId: { type: String, required: true },   
 origin: { type: String, required: true },   
 destination: { type: String, required: true },   
 departureTime: { type: String, required: true },   
 arrivalTime: { type: String, required: true },   
 basePrice: { type: Number, required: true },   
 totalSeats: { type: Number, required: true }  
});

3. Collection: comments

const bookingSchema = new mongoose.Schema({   
 user: { type: mongoose.Schema.Types.ObjectId, ref: 'User', required: true },   
 flight: { type: mongoose.Schema.Types.ObjectId, ref: 'Flight', required: true },   
 flightName: {type: String, required: true},   
 flightId: {type: String},   
 departure: {type: String},   
 destination: {type: String},   
 email: {type: String},   
 mobile: {type: String},   
 seats: {type: String},   
 passengers: [{ name: { type: String }, age: { type: Number } }],   
 totalPrice: { type: Number },   
 bookingDate: { type: Date, default: Date.now },   
 journeyDate: { type: Date },   
 journeyTime: { type: String },   
 seatClass: { type: String},   
 bookingStatus: {type: String, default: "confirmed"}   
});

**Integration with Backend**

* Database connection: Give Screenshot of Database connection done using Mongoose

* The backend APIs interact with MongoDB using Mongoose ODM Key interactions include:
  + User Management: CRUD operations for users.
  + Flight Management: CRUD operations for flights with all the information regarding flights available.
  + Booking Management: CRUD operations for the booking made by user on Journify, including all the required information.