**Database Development**

**10.1 Configure Schema**

To manage and store application data effectively, we first need to configure **MongoDB Schemas**. The schemas are based on the ER diagram and represent key entities such as Users, Flights, and Bookings.

Below is an overview of the schema structure created using **Mongoose**:

js

CopyEdit

// User Schema

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' }

});

js

CopyEdit

// Flight Schema

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 }

});

js

CopyEdit

// Booking Schema

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" }

});

**10.2 Connect Database to Backend**

To interact with the database, we need to establish a connection between **Express.js** and **MongoDB** using Mongoose. Below is the sample code used for connection:

js

CopyEdit

const PORT = process.env.PORT || 6001;

mongoose.connect(process.env.MONGO\_URL, {

useNewUrlParser: true,

useUnifiedTopology: true

})

.then(() => {

server.listen(PORT, () => {

console.log(`Running @ ${PORT}`);

});

})

.catch((err) => {

console.log("Error: ", err);

});

* Ensure MONGO\_URL is properly configured in your .env file.
* Use tools like **MongoDB Compass** or **MongoDB Atlas** to verify the connection and view stored collections.