BUS RESERVATION SYSTEM

1. Understand the Requirements

- User Stories: Identify the primary users (e.g., travelers, admins) and what they need from the system (e.g., booking tickets, viewing available routes, etc.).
- Key Features: List out essential features like:
    - Route selection
    - Date picker
    - Passenger details form (Name, Contact, Seats)
    - Seat availability check
    - Booking confirmation
    - Payment options (if applicable)

2. Create Wireframes

- Low-Fidelity Wireframes: These are simple, basic sketches that layout the structure of your bus reservation system without focusing on design. You can create these directly in Figma or on paper.
    - Main Pages to Design:
        - Home page (Landing page with routes and a "Book Now" button)
        - Reservation form (For selecting route, date, passenger info)
        - Booking confirmation page (Confirmation of the reservation with details)
- Wireframe Components:

    - Navigation bar (with links to Home, Book Now, Routes, etc.)
    - Route dropdown menu
    - Date picker
    - Form fields for name, contact, number of seats, etc.
    - Submit button

3. Design High-Fidelity Mockups

- Layout and Structure:
    - Start by setting up artboards for each screen (Home, Reservation, Confirmation, etc.).
    - Use Figma's grid system to align elements and maintain consistency in spacing and alignment.
- Design Components:
    - Typography: Choose readable fonts (e.g., Roboto, Open Sans) and use clear headings for section titles.
    - Colors: Choose a color scheme that is user-friendly and represents your brand (for example, green for booking confirmation, blue for the background).
    - Buttons: Design clear call-to-action buttons (e.g., "Book Now", "Confirm Booking").
    - Inputs: Use input fields with clear labels for the name, contact, seats, and route.
    - Forms: Ensure that the form fields are aligned, and the text boxes are large enough for mobile devices.
- Add Imagery: Include relevant images, such as bus icons, background images for the homepage, or placeholder images for routes.

- Interactive Components: Use Figma’s interactive components to simulate the form submission, date selection, and route dropdown interactions.

4. Add Interactivity (Prototyping)

- Figma allows you to create interactive prototypes that simulate the flow of the application:
  - Link the Screens: Set up navigation between screens (e.g., clicking "Book Now" takes the user to the reservation form, and submitting the form goes to the confirmation page).
  - Buttons and Actions: Use Figma’s “Prototype” tab to create clickable buttons and form interactions.
  - Transitions: Add smooth transitions between screens (e.g., sliding, fading, or instant transition).
- Simulate Form Input: You can simulate the user typing in the form fields, selecting a date, or choosing a number of seats.
- Confirmation Page: After filling out the reservation form, clicking the "Submit" button should show a confirmation screen with booking details.

5. Make the Design Responsive

- Frames for Different Screen Sizes: Create multiple frames in Figma for different screen sizes (Desktop, Tablet, Mobile). Ensure the layout works well across all devices by adjusting the grid and spacing.
- Auto Layout: Use Figma’s auto-layout feature to ensure that the design adapts to various screen sizes.
- Responsive Design: Consider how elements will stack or adjust on smaller devices. For instance:
  - Form inputs should expand to fill the width of mobile screens.

    - Buttons should be large enough for easy tapping on mobile devices.

6. Get Feedback and Refine

- Feedback from Stakeholders: Share your design with stakeholders (e.g., project managers, developers, and users) and gather feedback. Iterate and improve based on the input.
- User Testing: Conduct usability testing using Figma's sharing features to get real user feedback. Ensure the form is easy to navigate, especially on mobile devices.

7. Handoff to Development

- Prepare Design Files for Handoff: Once the design is finalized, you can hand off the files to the development team.
    - Use Figma's Inspect Panel to provide CSS code, font styles, and color codes to the developers.
    - Share the design file with a link so developers can access the assets (images, fonts, icons, etc.).
    - Include annotations for complex components, like buttons or specific layouts.

Example Figma Design Structure:

- Frame 1: Home Page
    - Hero section with a "Book Now" button.
    - Available routes listed as cards or buttons.
- Frame 2: Reservation Form
    - Route dropdown.
    - Date picker.
    - Name, Contact, Seats input fields.

    - Submit button.
- Frame 3: Booking Confirmation
    - Confirmation message with route, date, and passenger details.

Key Components to Include in Your Figma Design:

1. Navigation Bar: For easy navigation between the homepage, booking form, and confirmation page.
2. Dropdowns: For route selection (with predefined options).
3. Date Picker: To select travel date.
4. Input Fields: For passenger name, contact details, and number of seats.
5. Buttons: "Book Now", "Submit", "Confirm", etc.
6. Error Messages: Display validation errors when form fields are not filled.
7. Confirmation Message: Displaying a successful booking message after form submission.

8.Responsive Design Elements: Make sure to adjust the design for mobile, tablet, and desktop sizes.

PYTHON CODE

```
from flask import Flask, render_template, request, redirect, url_for


app = Flask(__name__)


# Sample bus routes and availability
```

```
routes = ["Delhi - Agra", "Mumbai - Pune", "Chennai - Bangalore", "Kolkata -
Patna"]

seats_available = {

    "Delhi - Agra": 20,

    "Mumbai - Pune": 15,

    "Chennai - Bangalore": 10,

    "Kolkata - Patna": 25

}


@app.route('/')

def home():

    return render_template('index.html', routes=routes)


@app.route('/book', methods=['POST', 'GET'])

def book():

    if request.method == 'POST':

        # Get form data

        route = request.form['route']

        travel_date = request.form['date']

        name = request.form['name']

        contact = request.form['contact']

        seats = int(request.form['seats'])


        # Check seat availability
        if seats <= seats_available.get(route, 0):
            seats_available[route] -= seats  # Deduct booked seats
            return render_template('confirmation.html', name=name,
contact=contact,
                                   route=route, travel_date=travel_date, seats=seats)
        else:
            return render_template('error.html', error="Not enough seats
available.")

    return render_template('book.html', routes=routes)


@app.route('/confirmation')
def confirmation():
    return render_template('confirmation.html')


@app.route('/error')
def error():
    return render_template('error.html')


if __name__ == '__main__':
    app.run(debug=True)
```

MONGO DB

```
from flask import Flask, render_template, request, redirect, url_for

from pymongo import MongoClient

from bson.objectid import ObjectId


app = Flask(__name__)


client = MongoClient("mongodb://localhost:27017/")  connection string for MongoDB Atlas

db = client['bus_reservation']  # Database name

reservations_collection = db['reservations']


routes = ["Delhi - Agra", "Mumbai - Pune", "Chennai - Bangalore", "Kolkata - Patna"]

seats_available = {

    "Delhi - Agra": 20,

    "Mumbai - Pune": 15,

    "Chennai - Bangalore": 10,

    "Kolkata - Patna": 25

}


@app.route('/')

def home():
    return render_template('index.html', routes=routes)


@app.route('/book', methods=['POST', 'GET'])
def book():
    if request.method == 'POST':
        # Get form data
        route = request.form['route']
        travel_date = request.form['date']
        name = request.form['name']
        contact = request.form['contact']
        seats = int(request.form['seats'])

        if seats <= seats_available.get(route, 0):

            reservation = {
                'route': route,
                'travel_date': travel_date,
                'name': name,
                'contact': contact,
                'seats': seats
            }
            reservation_id =
reservations_collection.insert_one(reservation).inserted_id
```

```
            seats_available[route] -= seats
            return render_template('confirmation.html',
reservation_id=reservation_id, name=name,
                                   contact=contact, route=route, travel_date=travel_date,
seats=seats)

        else:
            return render_template('error.html', error="Not enough seats
available.")

    return render_template('book.html', routes=routes)

@app.route('/confirmation')
def confirmation():
    return render_template('confirmation.html')

@app.route('/error')
def error():
    return render_template('error.html')

@app.route('/reservations')
def reservations():

    all_reservations = reservations_collection.find()
    return render_template('reservations.html', reservations=all_reservations)


if __name__ == '__main__':
    app.run(debug=True)
```