MIT Art, Design and Technology University
MIT School of Computing, Pune

Department of Information Technology

Lab File

Practical - Full Stack Development

Class - L.Y. (SEM-VII) SMAD

Name of the Practical Coordinator

Prof. Rohini Bhosale

A.Y. 2024 – 2025 (SEM-VII)

Full Stack Development

SEMESTER – VII

Course Code:		**Course Credits:**	**04**
Teaching Hours / Week (L:T:P):	**2:0:4**	**CA Marks:**	**40**
Total Number of Teaching Hours:	**--**	**END-SEM Marks:**	**60**

Course Pre-requisites: Web Technology

Course Description:

a. This course offers a comprehensive foundation in full stack web development using Node.js, Express.js, EJS, and MySQL.
b. Students will learn to build dynamic and responsive web applications by integrating front-end and back-end technologies.
c. The course emphasizes hands-on project-based learning and real-world application development.
d. Learners will design RESTful APIs and manage databases for seamless client-server interaction.
e. Through practical exercises, students will gain industry-ready skills for scalable web application development.

Course Learning Objectives: This course will enable the students to:

1. Introduce students to server-side development using Node.js and its ecosystem.
2. Equip learners with the ability to develop scalable and efficient web servers using Express.js.
3. Enable the use of EJS for dynamic web page rendering and data binding.
4. Provide hands-on experience in integrating relational databases like MySQL with Node.js.
5. Facilitate the development of full stack applications through project-based learning using REST APIs and client-server integration.

Course Outcome:

1. CO1: Build responsive front-end user interfaces using Bootstrap layout and component system.
2. CO2: Analyze the core principles, architecture, and modular structure of Node.js for robust server-side application development.
3. CO3: Construct and deploy scalable web servers using Node.js and Express.js to efficiently manage HTTP protocols and client-server communication.
4. CO4: Design and synthesize dynamic, data-driven user interfaces using EJS templating integrated with backend services.
5. CO5: Implement and manage CRUD operations by interfacing Node.js with MySQL and engineer RESTful APIs for structured data exchange.

UNIT – I	**Responsive Interfaces with Bootstrap Components**	**14 Hours**

Main Topic-1: Introduction to Media Queries, Responsive Web Design Using Media Queries, Syntax, Uses of Media Queries, Benefits of Responsive Design, Elements of a Responsive Design, Working of Responsive Design **Main Topic-2: Introduction to Bootstrap: Bootstrap Concept, Uses of Bootstrap, Bootstrap library source, Bootstrap Basics- container, jumbotron, page header, buttons, tables;** **Main Topic-3: Bootstrap Grid System, Bootstrap Images, Bootstrap Forms, Bootstrap Flexbox- horizontal direction, vertical direction.**		
Pedagogy	**ICT Teaching / Power Point Presentation and Videos**	
	Self-study / Do it yourself	
	Experiential Learning Topics:	
	Case Study / PBL - Project Based Learning	
UNIT – II	**Backend Development Fundamentals with Node.js**	**10 Hours**
Main Topic-1: Introduction to Node.js: What is Node.js? Why use it? Capabilities and file structure **Main Topic-2: Setting up Node.js Development Environment – Installing Node.js and npm, NPM and project creation, Node.js modules, Creating a basic web server, Node.js console**		
Pedagogy	**ICT Teaching / Power Point Presentation and Videos**	
	Self-study / Do it yourself	
	Experiential Learning Topics:	
	Case Study / PBL - Project Based Learning	
UNIT – III	**Building Web Servers using Express.js**	**12 Hours**
Main Topic-1: Introduction to Express.js – Features, benefits, use cases **Main Topic-2: Express Server Setup – Installing Express.js, project setup, configuration, building Express - based web server**		
Pedagogy	**ICT Teaching / Power Point Presentation and Videos**	
	Self-study / Do it yourself	
	Experiential Learning Topics:	
	Case Study / PBL - Project Based Learning	
UNIT – IV	**Templating and Dynamic Views with EJS**	**12 Hours**
Main Topic-1: Integrating EJS Templates, Creating EJS partials and views, Passing data to views, rendering variables, Looping through data in templates **Main Topic-2: Building dynamic front-ends using EJS with Express**		
Pedagogy	**ICT Teaching / Power Point Presentation and Videos**	
	Self-study / Do it yourself	
	Experiential Learning Topics:	
	Case Study / PBL - Project Based Learning	

UNIT – V	Database Integration and REST API Development	12 Hours
Main Topic-1: Connecting Node.js with MySQL, Installing MySQL, setting up the connection, Performing CRUD operations using Node.js and MySQL **Main Topic-2: Building RESTful APIs using Express.js, HTTP methods (GET, POST, PUT, DELETE), Handling requests and responses using Express, Using body-parser for form data**		
Pedagogy	**ICT Teaching / Power Point Presentation and Videos**	
	Self-study / Do it yourself	
	Experiential Learning Topics:	
	Case Study / PBL - Project Based Learning	

COs	PO-1	PO-2	PO-3	PO-4	PO-5	PO-6	PO-7	PO-8	PO-9	PO-10	PO-11	PO-12	PSO-1	PSO-2
CO-1	3	3	2	3	3					3		3	2	
CO-2	3	2			2					3		3	2	
CO-3	3	3	2	3	3					2		3	2	
CO-4					3				2	3		2		3
CO-5					3				2	3		2		3
3 – HIGH, **2** – MEDIUM, **1** - LOW														

List of Assignments:

Sr. No. **Assignment Title**

1. **Create a Static Homepage Layout Using Bootstrap**

   1.1 Design a responsive homepage using Bootstrap components like container, jumbotron, headers, and buttons.

   1.2 Use the grid system to create a structured and adaptive layout suitable for any web app.

2. **Design a Grid-Based Information Page**

   2.1 Develop a multi-column content layout using Bootstrap Grid.

   2.2 Display items such as cards, product previews, or blog entries with consistent alignment and spacing.

3. **Set Up a Node.js Project with Express Server**

   3.1 Initialize a new Node.js project using npm init.

   3.2 Install Express.js and set up a basic web server with routes like Home (/), About (/about), and Contact (/contact).

4. **Implement Modular Routing and Middleware Logging**

   4.1 Organize server logic using separate route modules (e.g., userRoutes, productRoutes).

4.2 Add middleware to log request URLs, methods, and timestamps for all incoming traffic.

5. **Create a Data Submission Form and Handle POST Requests**

5.1 Build an HTML form (e.g., for adding items or submitting feedback) and handle its POST request using Express.

5.2 Display the submitted data dynamically on a confirmation view.

6. **Serve Dynamic Content Using Server-Side Rendering**

6.1 Render a list of items (e.g., products, posts, services) dynamically using EJS with hardcoded or temporary in-memory data.

6.2 Route should accept data from the server and render it in the view using loops.

7. **Create Dynamic Detail Pages Using URL Parameters**

7.1 Use route parameters to render individual item details dynamically via EJS.

7.2 Display properties like title, description, and date using passed route data.

8. **Build a Reusable Layout Using EJS Partials**

8.1 Create common EJS partials (header, footer, navbar) and include them across views.

8.2 Maintain consistency across all pages by using a shared layout template.

9. **Connect Node.js to MySQL and Perform Basic CRUD**

9.1 Create MySQL tables and connect to them using a MySQL Node.js package.

9.2 Implement basic CRUD operations from Express routes to the database.

10. **Develop RESTful APIs for Data Access and Manipulation**

10.1 Build a set of RESTful APIs using Express.js for performing operations.

10.2 Test the endpoints using Postman and ensure they return structured JSON responses.

Experiment No.1

Problem Statement:

To design and develop a responsive homepage for a web application using Bootstrap, ensuring that the layout adjusts seamlessly across different screen sizes (desktop, tablet, and mobile). The homepage should use key Bootstrap components such as containers, jumbotron (hero section), headers, buttons, and the grid system to create an organized and visually appealing interface.

Objective:

To understand the structure and implementation of Bootstrap's responsive grid system.

- To utilize Bootstrap components (like containers, rows, and columns) for layout creation.
- To design a static homepage that adapts to various screen resolutions.
- To enhance design efficiency by leveraging Bootstrap's pre-built CSS and JS utilities.

Theory:

Bootstrap is a popular front-end framework used to build responsive and mobile-first websites quickly. It provides a structured approach to layout design through its grid system and reusable UI components.

Key Concepts:

1. Container:

The outermost element that aligns and centers content with padding for responsiveness.

Example: ...

2. Grid System:

Bootstrap uses a 12-column grid to create flexible layouts.

Example:

```
<div class="row">
  <div class="col-md-6">Column 1</div>
  <div class="col-md-6">Column 2</div>
</div>
```

3. Jumbotron (Hero Section):

A large, attention-grabbing section for showcasing main content or messages.

Example:

```
<div class="jumbotron text-center bg-light">
  <h1>Welcome to TurboTalks</h1>
  <p>Your hub for automotive insights.</p>
  <a href="#" class="btn btn-primary">Learn More</a>
```

```
</div>
```

4. Responsive Design:

Bootstrap's predefined classes automatically adapt the layout to various screen sizes (e.g., .col-md-4, .col-sm-6, .col-lg-3).

5. Utility Classes:

Bootstrap provides classes for spacing, colors, and typography (e.g., .text-center, .mt-3, .btn-outline-dark).

Code: home.ejs:

```
<!DOCTYPE html>

<html lang="en">

 <head>

  <%- include("../partials/head") %>

 </head>

 <body>

  <%- include("../partials/header")%>

  <section

   class="hero-section text-center d-flex flex-column justify-content-center align-items-center bg-dark text-white p-5"

   style="min-height: 80vh">

   <h1 class="display-3 fw-bold">Welcome to TurboTalks</h1>

   <p class="lead">Your daily dose of Car News, Reviews & Insights</p>

   <a href="/news" class="btn btn-warning btn-lg mt-3"

    >Explore Latest News</a >

  </section>

  <section class="container my-5" id="featured">

   <div class="text-center mb-5 py-3 featured-title">

    <h2>🔥 Featured Cars 2025</h2>

   </div>

   <div class="row g-4">

    <div class="col-md-4">

     <div class="card h-100 shadow-sm">
```

```
    <img
      src="/assets/pagani.jpg"
      class="card-img-top"
      alt="Pagani Huyara BC"
    />
    <div class="card-body">
      <h5 class="card-title text-center">Pagani Huyara BC</h5>
      <p class="card-text text-center">
        A tribute to Pagani's first customer, Benny Caiola, the Huayra
        BC is an even more extreme iteration of the already formidable
        Huayra. Its AMG-sourced twin-turbocharged V12 engine is
        meticulously crafted to deliver breathtaking power. The car's
        construction is a masterclass in lightweight engineering,
        featuring advanced carbon-titanium materials. Active
        aerodynamics, with striking movable flaps, ensures phenomenal
        downforce and stability. The interior is a bespoke work of art,
        blending exposed mechanical elements with the finest leathers
        and carbon fiber.
      </p>
      <a href="/reviews" class="btn btn-outline-dark">Read Review</a>
    </div>
  </div>
</div>
<div class="col-md-4">
  <div class="card h-100 shadow-sm">
    <img
      src="/assets/bugatti.jpg"
      class="card-img-top"
      alt="Bugatti Divo"
```

```
    />
    <div class="card-body">
     <h5 class="card-title text-center">Bugatti Divo</h5>
     <p class="card-text text-center">
      Crafted for corners, the Bugatti Divo is a coach-built hypercar
      based on the Chiron, with a focus on aerodynamic efficiency and
      handling. Its aggressive design, featuring a larger fixed rear
      wing and numerous air inlets, generates significantly more
      downforce. While it shares the Chiron's monstrous 8.0-liter
      quad-turbo W16 engine, the Divo's chassis and suspension are
      tuned for agility on winding roads. Production was strictly
      limited to just 40 units, making it an exceptionally rare and
      coveted machine. The Divo is a testament to Bugatti's ability to
      create a car that is as thrilling in the bends as it is in a
      straight line.
     </p>
     <a href="/reviews" class="btn btn-outline-dark">Read Review</a>
    </div>
   </div>
  </div>
  <div class="col-md-4">
   <div class="card h-100 shadow-sm">
    <img
     src="/assets/jesko.jpg"
     class="card-img-top"
     alt="Koenigsegg Jesko"
    />
    <div class="card-body">
     <h5 class="card-title text-center">Koenigsegg Jesko</h5>
```

```
    <p class="card-text text-center">
      The Koenigsegg Jesko, a Swedish marvel of engineering, is a
      hypercar designed for ultimate track performance and high speed.
      It boasts a re-engineered 5.0-liter twin-turbo V8 engine that
      can produce up to 1600 horsepower on E85 biofuel. A
      revolutionary nine-speed multi-clutch Light Speed Transmission
      allows for instantaneous gear changes. Its advanced
      aerodynamics, including a massive, active rear wing, provide
      incredible downforce for superior cornering. The Jesko is a
      showcase of Koenigsegg's relentless pursuit of innovation and
      record-breaking speed.
    </p>
    <a href="/reviews" class="btn btn-outline-dark">Read Review</a>
   </div>
  </div>
</div>
<div class="col-md-4">
  <div class="card h-100 shadow-sm">
   <img
    src="/assets/bmw.jpg"
    class="card-img-top"
    alt="BMW M5 Competition Sport"
   />
   <div class="card-body">
    <h5 class="card-title text-center">BMW M5 Competition Sport</h5>
    <p class="card-text text-center">
      The BMW M5 CS stands as the most powerful and track-focused M5
      ever produced by the German automaker. Its 4.4-liter M TwinPower
      Turbo V8 engine has been tuned to deliver an exhilarating 627
```

```
      horsepower. Extensive use of carbon fiber in the hood, roof, and
      other components significantly reduces weight, enhancing
      agility. The M xDrive all-wheel-drive system can be configured
      for pure rear-wheel-drive hoonery. With its exclusive
      gold-bronze accents and four individual bucket seats, the M5 CS
      is a luxurious yet ferociously capable super sedan.
    </p>
    <a href="/reviews" class="btn btn-outline-dark">Read Review</a>
   </div>
  </div>
</div>
<div class="col-md-4">
 <div class="card h-100 shadow-sm">
  <img
   src="/assets/amg.jpg"
   class="card-img-top"
   alt="Mercedes AMG GTR"
  />
  <div class="card-body">
   <h5 class="card-title text-center">Mercedes AMG GTR</h5>
   <p class="card-text text-center">
    Nicknamed "The Beast of the Green Hell," the Mercedes-AMG GT R
    was honed on the demanding Nürburgring Nordschleife. Its
    handcrafted 4.0-liter biturbo V8 engine produces a thunderous
    soundtrack and immense power. An advanced nine-mode traction
    control system, derived from motorsport, allows drivers to
    fine-tune the car's handling characteristics. The GT R's
    aggressive aerodynamics, including a large rear wing and active
    front grille shutters, provide exceptional grip. This car
```

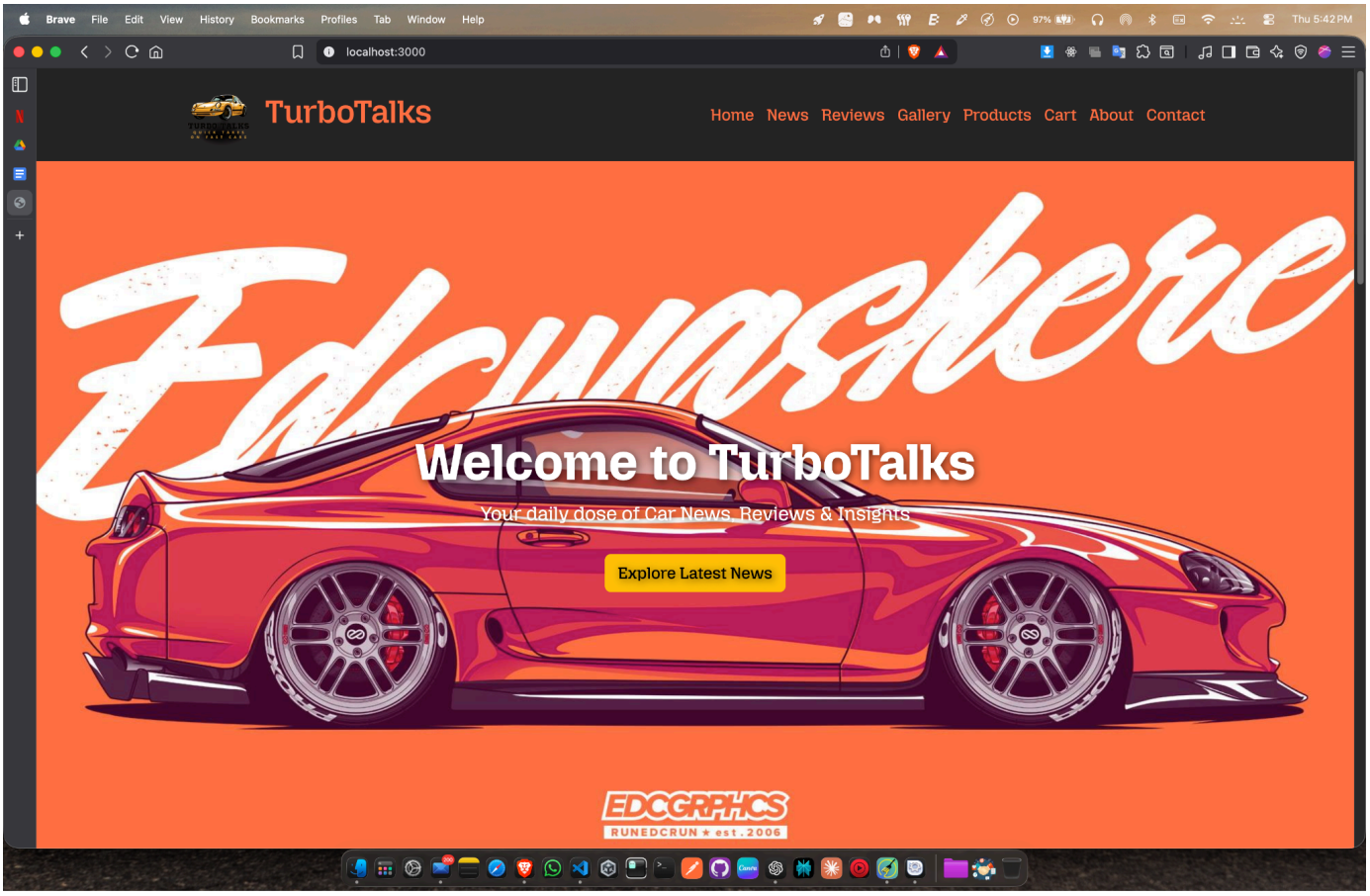
```
        represents the pinnacle of AMG's performance-oriented
        engineering for the road and track.
       </p>
       <a href="/reviews" class="btn btn-outline-dark">Read Review</a>
     </div>
    </div>
   </div>
   <div class="col-md-4">
    <div class="card h-100 shadow-sm">
     <img src="/assets/rs7.jpg" class="card-img-top" alt="Audi RS7" />
     <div class="card-body">
      <h5 class="card-title text-center">Audi RS7</h5>
      <p class="card-text text-center">
        The Audi RS7 is a high-performance sportback that seamlessly
        blends blistering speed with everyday usability. Its potent
        4.0-liter twin-turbo V8 engine delivers startling acceleration,
        complemented by Audi's legendary Quattro all-wheel-drive system.
        The RS adaptive air suspension provides a dynamic driving
        experience, balancing comfort and sharp handling. Its sleek,
        coupe-like design conceals a spacious and luxurious interior
        equipped with cutting-edge technology. The RS7 is the
        quintessential grand tourer, capable of crossing continents at
        pace and in style.
       </p>
       <a href="/reviews" class="btn btn-outline-dark">Read Review</a>
     </div>
    </div>
   </div>
  </div>
```

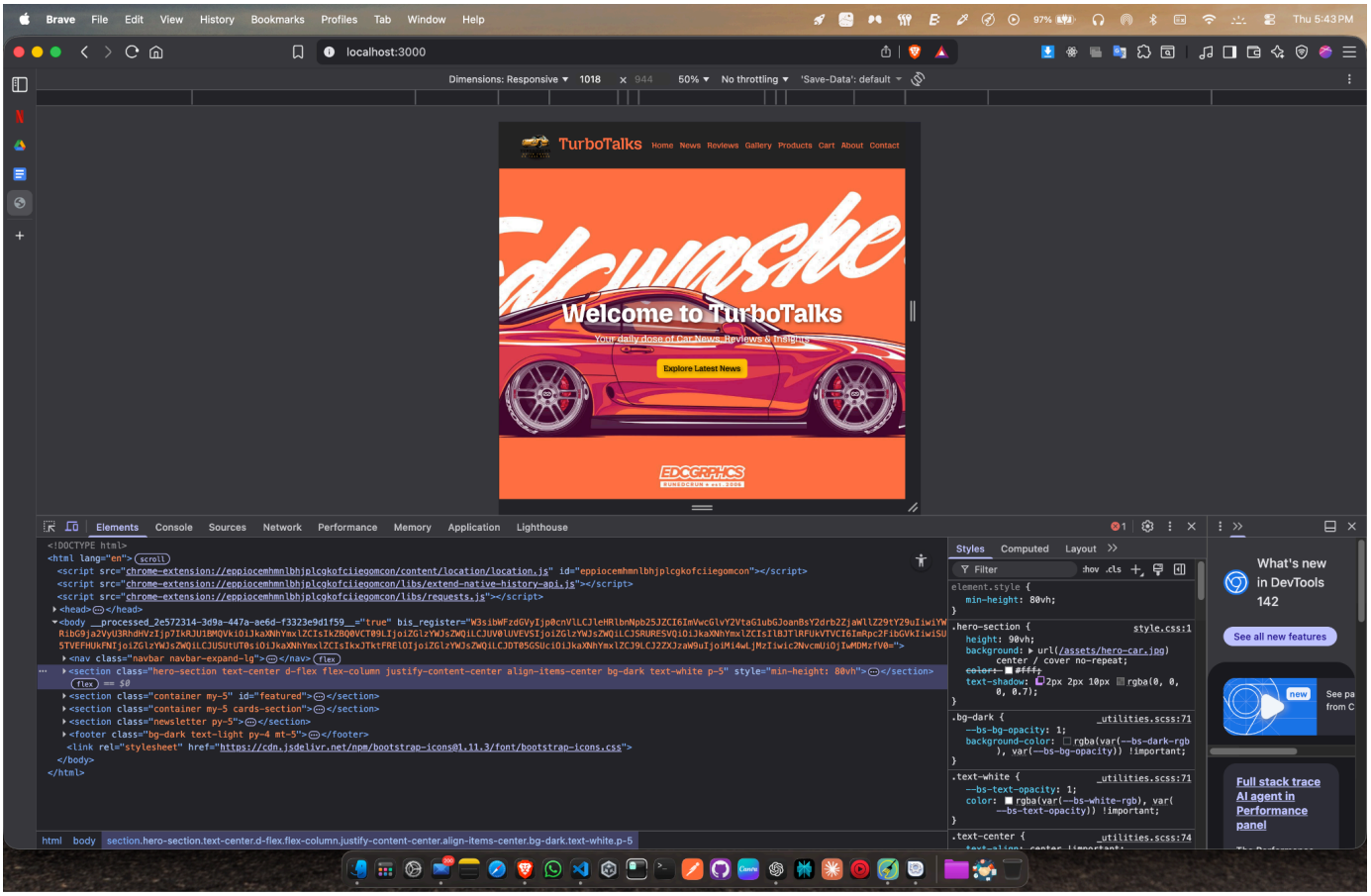
```
</section>
<section class="container my-5 cards-section">
 <div class="row text-center g-4">
  <div class="col-md-4">
   <div class="card-custom h-100">
    <h3>  Latest News</h3>
    <p>Catch up on the hottest updates in the auto world.</p>
    <a href="/news" class="btn btn-dark">View News</a>
   </div>
  </div>
  <div class="col-md-4">
   <div class="card-custom h-100">
    <h3>  Expert Reviews</h3>
    <p>In-depth reviews and first-hand experiences with top cars.</p>
    <a href="/reviews" class="btn btn-dark">View Reviews</a>
   </div>
  </div>
  <div class="col-md-4">
   <div class="card-custom h-100">
    <h3>  Car Gallery</h3>
    <p>Stunning images of supercars, classics, and concept cars.</p>
    <a href="/gallery" class="btn btn-dark">View Gallery</a>
   </div>
  </div>
 </div>
</section>

<section class="newsletter py-5">
 <div class="container">
```

```
<div
 class="card shadow-lg border-0 rounded-4 mx-auto"
 style="max-width: 700px"
>
 <div class="card-body text-center p-5">
  <h3 class="fw-bold mb-3">Stay Updated with TurboTalks 🚗</h3>
  <p class="text-muted mb-4">
   Join our mailing list to get the latest car news, reviews, and
   insights delivered to your inbox.
  </p>
  <form class="row g-2 justify-content-center">
   <div class="col-12 col-md-8">
    <input
     type="email"
     class="form-control form-control-lg rounded-pill"
     placeholder="Enter your email"
     required
    />
   </div>
   <div class="col-12 col-md-3">
    <button class="btn btn-warning btn-lg w-100 rounded-pill">
     Subscribe
    </button>
   </div>
  </form>
 </div>
</div>
</div>
</section>
```

```
  <%- include("../partials/footer") %>

 </body>

</html>
```

Output:





Conclusion:

By using Bootstrap's grid system and responsive components, a static homepage layout can be developed efficiently without writing extensive custom CSS. The page automatically adjusts to different devices, maintaining usability and aesthetics. This demonstrates how Bootstrap simplifies responsive web design, ensuring faster development and a consistent layout across all platforms.

Experiment No.2

Problem Statement:

To create a multi-column, grid-based information page using Bootstrap's grid system that organizes and displays multiple elements — such as product previews, cards, or blog entries — with consistent alignment, spacing, and responsiveness.

The aim is to ensure that the layout adjusts dynamically across devices (desktop, tablet, and mobile) while maintaining a clean and professional look.

Objective:

- To understand and implement Bootstrap's grid layout system for structuring content.
- To design a uniform multi-column display using cards or item containers.
- To achieve responsive alignment and spacing for different screen sizes using Bootstrap classes.
- To enhance the readability and visual hierarchy of information on a webpage.

Theory:

Bootstrap's Grid System is based on a 12-column layout that enables developers to build responsive, organized, and flexible web designs. By dividing the page into rows and columns, the grid ensures that elements resize and realign automatically based on the screen width.

Key Concepts:

1. Container:
- The parent element that holds all rows and columns.
- Example:

```
<div class="container">
  <!-- Grid content here -->
</div>
```

2. Row and Columns:
- Every grid begins with a .row inside which .col-* elements are defined.
- Example:

```
<div class="row">
  <div class="col-md-4">Column 1</div>
  <div class="col-md-4">Column 2</div>
  <div class="col-md-4">Column 3</div>
</div>
```

3. Responsive Breakpoints:
- Bootstrap columns adjust based on screen width using classes like:

- .col-1g-* → large devices
- co1-md-* → medium devices
- . col-sm-* → small devices
- . col-* → extra small devices
- This ensures smooth adaptability across all devices.

4. Cards:

- Bootstrap cards are flexible content containers used for displaying images, text, and links.
- Example:

```
<div class="card">
 <img src="image.jpg" class="card-img-top" alt="...">
 <div class="card-body">
  <h5 class="card-title">Product Title</h5>
  <p class="card-text">Short description about the product.</p>
 </div>
</div>
```

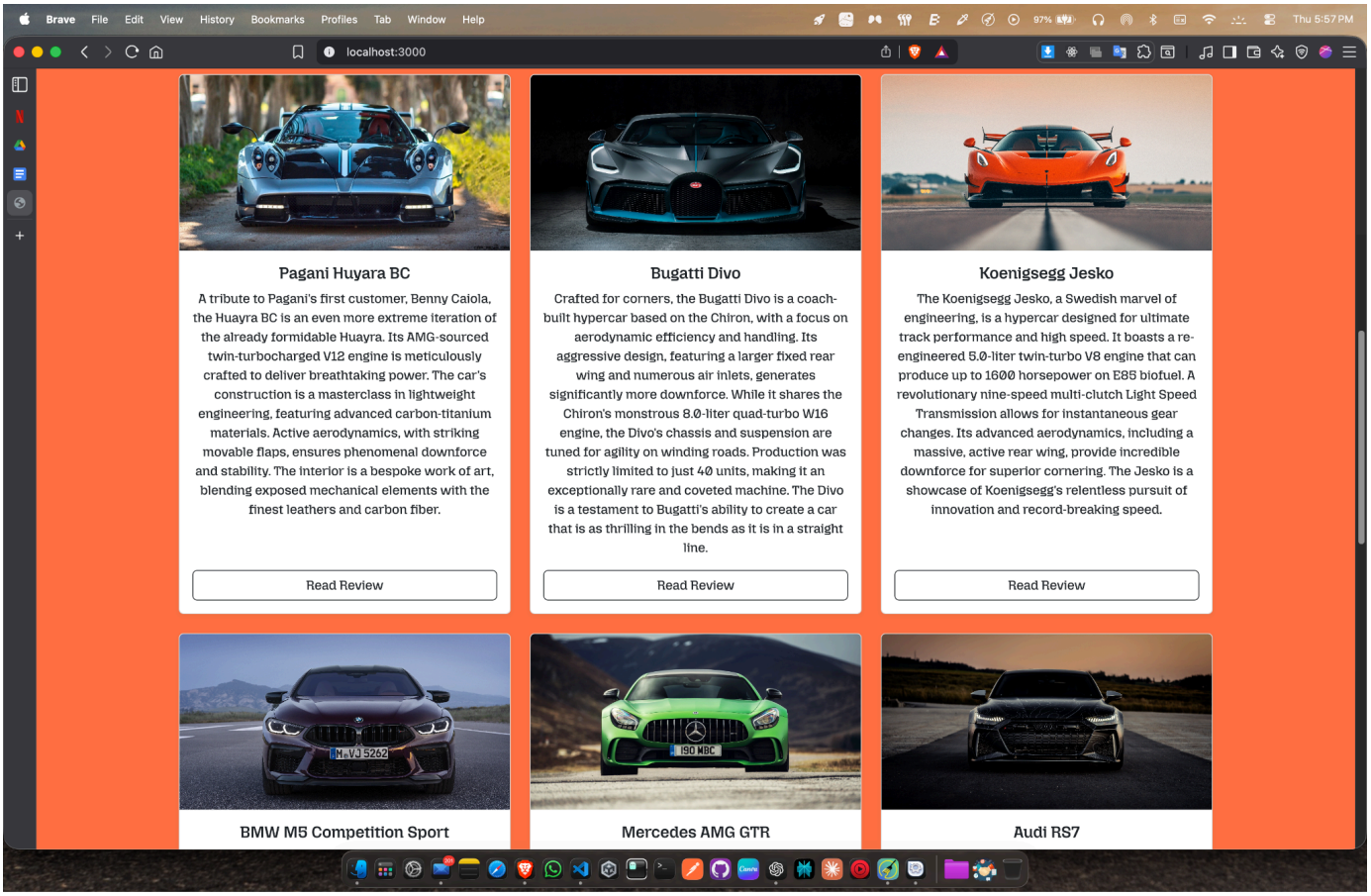
Code: reviews.ejs:

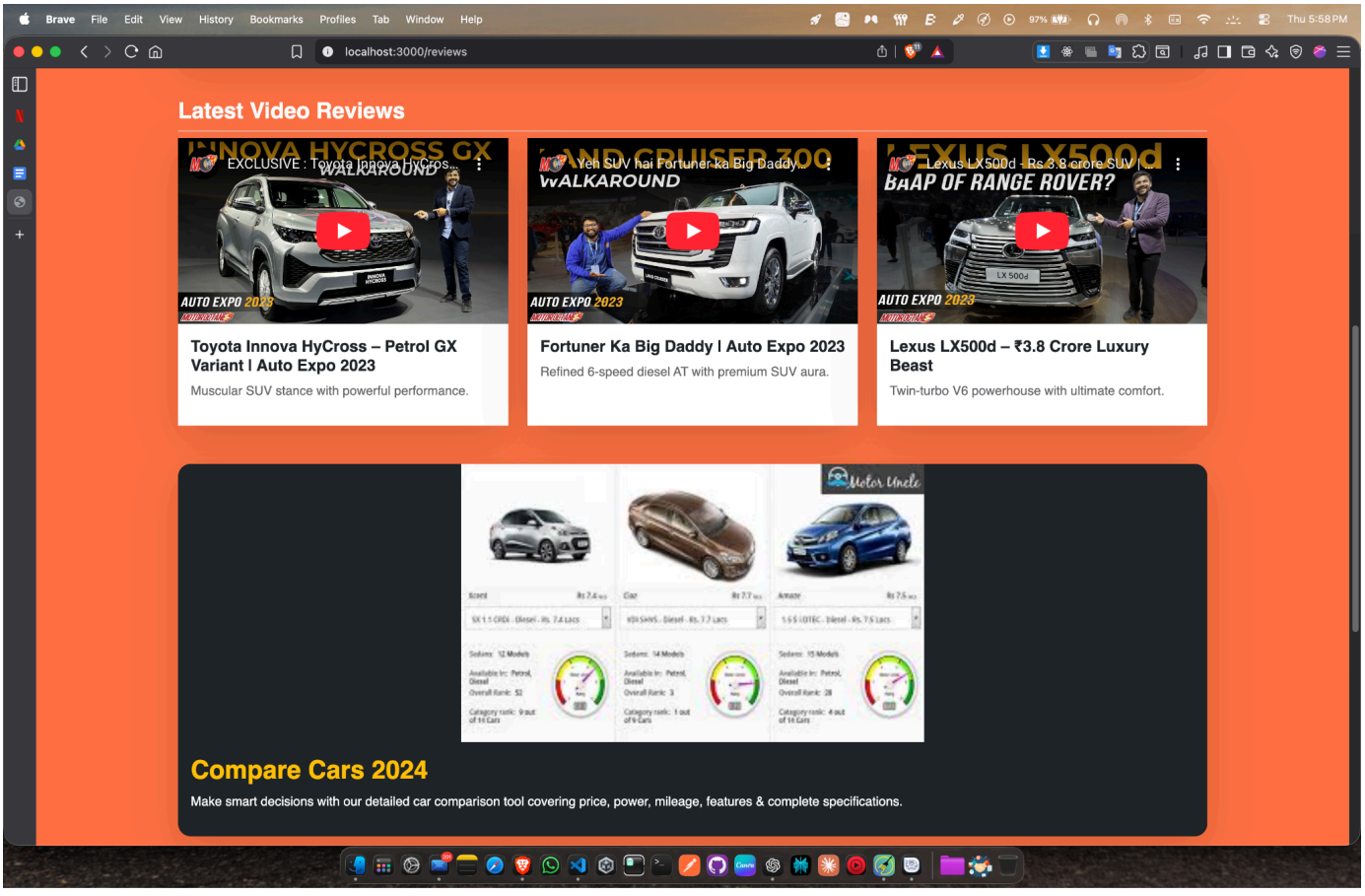
```
<%- include("../partials/head") %>
<body style="background-color: #FF7244; color: #fff; font-family: 'Poppins', sans-serif;">
 <%- include("../partials/header") %>
 <section class="hero text-center text-white d-flex align-items-center justify-content-center position-relative">
  <div class="overlay position-absolute top-0 start-0 w-100 h-100" style="background: rgba(0, 0, 0, 0.4);"></div>
  <div class="hero-content position-relative z-2">
   <h1 class="display-4 fw-bold">Car Reviews & Comparisons</h1>
   <p class="lead">Honest opinions, real performance – straight from the auto world</p>
  </div>
 </section>
 <div class="container my-5">
  <div class="card bg-dark text-white border-0 rounded-4 overflow-hidden shadow-lg">
   <img src="/assets/autoexpo.jpg" class="card-img-top" style="height:380px; object-fit:contain;">
   <div class="card-body">
```

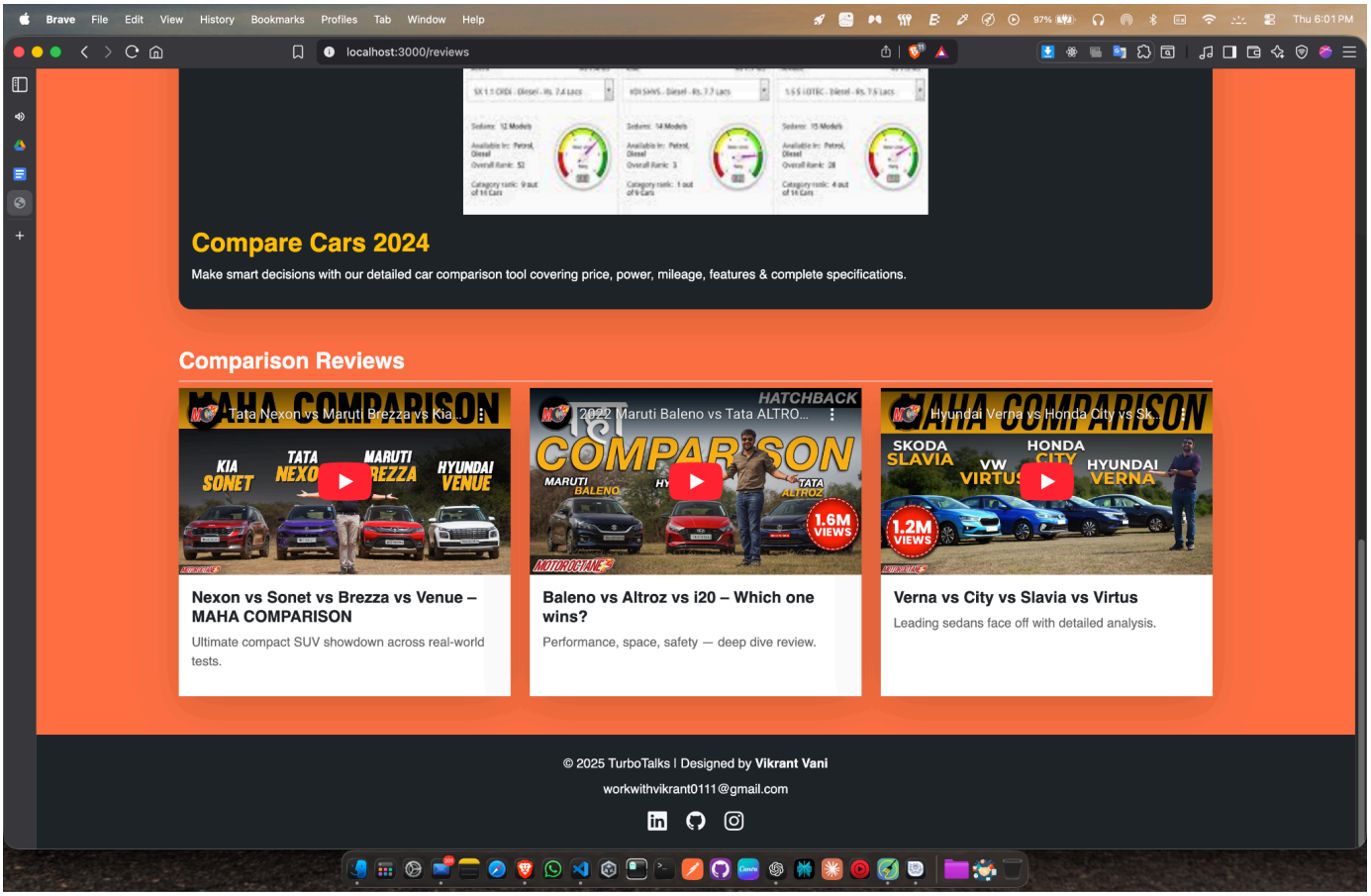
```
    <h2 class="fw-bold text-warning">AUTO EXPO 2023 – The Biggest Auto Extravaganza</h2>
    <p>
     A signature event bringing together the latest automotive products, technologies,
     concepts and futuristic trends from global manufacturers.
    </p>
   </div>
  </div>
 </div>
<div class="container my-5">
 <h3 class="fw-bold border-bottom pb-2">Latest Video Reviews</h3>
 <div class="row g-4">
  <% reviewsList.forEach(review => { %>
  <div class="col-md-6 col-lg-4">
   <div class="card shadow-lg rounded-4 h-100 border-0">
    <div class="ratio ratio-16x9">
     <iframe src="<%= review.video %>" allowfullscreen></iframe>
    </div>
    <div class="card-body bg-white">
     <h5 class="fw-bold text-dark"><%= review.title %></h5>
     <p class="text-muted"><%= review.desc %></p>
    </div>
   </div>
  </div>
  <% }) %>
 </div>
</div>
<div class="container my-5">
 <div class="card bg-dark text-white border-0 rounded-4 overflow-hidden shadow-lg">
  <img src="/assets/comparison.jpeg" class="card-img-top" style="height:350px; object-fit:contain;">
```

```
    <div class="card-body">
      <h2 class="fw-bold text-warning">Compare Cars 2024</h2>
      <p>
        Make smart decisions with our detailed car comparison tool covering price, power,
        mileage, features & complete specifications.
      </p>
    </div>
  </div>
</div>
<div class="container my-5">
  <h3 class="fw-bold border-bottom pb-2">Comparison Reviews</h3>
  <div class="row g-4">
    <% compareList.forEach(comp => { %>
    <div class="col-md-6 col-lg-4">
      <div class="card shadow-lg rounded-4 h-100 border-0">
        <div class="ratio ratio-16x9">
          <iframe src="<%= comp.video %>" allowfullscreen></iframe>
        </div>
        <div class="card-body bg-white">
          <h5 class="fw-bold text-dark"><%= comp.title %></h5>
          <p class="text-muted"><%= comp.desc %></p>
        </div>
      </div>
    </div>   <% }) %>
  </div>
</div>
<%- include("../partials/footer") %>
</body>
</html>
```

Output:


localhost:3000
Pagani Huyara BC
A tribute to Pagani's first customer, Benny Caiola, the Huayra BC is an even more extreme iteration of the already formidable Huayra. Its AMG-sourced twin-turbocharged V12 engine is meticulously crafted to deliver breathtaking power. The car's construction is a masterclass in lightweight engineering, featuring advanced carbon-titanium materials. Active aerodynamics, with striking movable flaps, ensures phenomenal downforce and stability. The interior is a bespoke work of art, blending exposed mechanical elements with the finest leathers and carbon fiber.
Read Review
Bugatti Divo
Crafted for corners, the Bugatti Divo is a coach-built hypercar based on the Chiron, with a focus on aerodynamic efficiency and handling. Its aggressive design, featuring a larger fixed rear wing and numerous air inlets, generates significantly more downforce. While it shares the Chiron's monstrous 8.0-liter quad-turbo W16 engine, the Divo's chassis and suspension are tuned for agility on winding roads. Production was strictly limited to just 40 units, making it an exceptionally rare and coveted machine. The Divo is a testament to Bugatti's ability to create a car that is as thrilling in the bends as it is in a straight line.
Read Review
Koenigsegg Jesko
The Koenigsegg Jesko, a Swedish marvel of engineering, is a hypercar designed for ultimate track performance and high speed. It boasts a re-engineered 5.0-liter twin-turbo V8 engine that can produce up to 1600 horsepower on E85 biofuel. A revolutionary nine-speed multi-clutch Light Speed Transmission allows for instantaneous gear changes. Its advanced aerodynamics, including a massive, active rear wing, provide incredible downforce for superior cornering. The Jesko is a showcase of Koenigsegg's relentless pursuit of innovation and record-breaking speed.
Read Review
BMW M5 Competition Sport
Mercedes AMG GTR
Audi RS7


localhost:3000/reviews
Latest Video Reviews
EXCLUSIVE : Toyota Innova HyCros...
Yen SUV hai Fortuner ka Big Daddy...
Lexus LX500d - Rs 3.8 crore SUV |
AUTO EXPO 2023
Toyota Innova HyCross – Petrol GX Variant I Auto Expo 2023
Muscular SUV stance with powerful performance.
Fortuner Ka Big Daddy I Auto Expo 2023
Refined 6-speed diesel AT with premium SUV aura.
Lexus LX500d – ₹3.8 Crore Luxury Beast
Twin-turbo V6 powerhouse with ultimate comfort.
Compare Cars 2024
Make smart decisions with our detailed car comparison tool covering price, power, mileage, features & complete specifications.



Conclusion:

By utilizing Bootstrap's grid and card components, we can efficiently build structured, responsive, and visually balanced pages for showcasing products, articles, or information.

This approach minimizes the need for custom CSS and ensures the layout automatically adapts to different screen sizes, improving the user experience and maintaining a professional, uniform design.

Experiment No.3

Problem Statement:

To create and configure a Node.js project that uses the Express.js framework to serve dynamic web pages. The goal is to initialize a Node.js environment, install required dependencies, and set up an Express-based web server capable of handling basic routes like Home (/), About (/about), and Contact (/contact).

Objective:

- To learn how to initialize a new Node.js project using npm init.
- To understand how to install and configure Express.js, a minimal and flexible Node.js framework.
- To build a simple server application that listens on a specific port and serves static/dynamic content.
- To create and manage routes that respond to different URLs such as /, /about, and /contact.

Theory:

1. Node.js Overview:

Node.js is a JavaScript runtime environment that allows developers to run JavaScript on the server side.

It uses an event-driven, non-blocking I/O model, making it efficient for developing scalable network applications.

2. Express.js Overview:

Express.js is a web application framework built on top of Node.js. It simplifies tasks like:

- Creating a web server
- Managing HTTP requests and responses
- Defining routes
- Handling middleware for authentication, logging, etc.

3. Project Setup Steps:

1. Initialize Node.js Project: npm init -y

This creates a package.json file containing metadata and dependencies for the project.

2. Install Express.js: npm install express

3. Create Server File (index.js or app.js): Example:

```
const express = require('express');

const app = express();

const PORT = 3000;

// Home Route

app.get('/', (req, res) => {

  res.send('<h1>Welcome to Home Page</h1>');

});
```

```
// About Route
app.get('/about', (req, res) => {
  res.send('<h1>About Us</h1><p>This is an Express.js application.</p>');
});
// Contact Route
app.get('/contact', (req, res) => {
  res.send('<h1>Contact Page</h1><p>Email: support@example.com</p>');
});
// Start Server
app.listen(PORT, () => {
  console.log(`Server running at http://localhost:${PORT}`);
});
```

4. Run the Server

```
node index.js
```

Code: Index.js:

```
const express = require("express");
const path = require("path");
const session = require("express-session");
const app = express();
var mysql = require("mysql2");
const connection = mysql.createConnection({
  host: "localhost",
  user: "vikrant01",
  password: "vicky@123",
  database: "test_db",
});
```

```
app.set("view engine", "ejs");
app.set("views", path.join(__dirname, "views"));
app.use(express.static(path.join(__dirname, "public")));
app.use(express.urlencoded({ extended: true }));
app.use(
 session({
  secret: "turbotalks-secret",
  resave: false,
  saveUninitialized: false, }));
function isAuthenticated(req, res, next) {
 if (req.session.user) return next();
 return res.redirect("/login");}
app.get("/login", (req, res) => {
 if (req.session.user) return res.redirect("/");
 res.render("pages/login", { error: null });});
app.post("/login", (req, res) => {
 const { email, password } = req.body;
 if (!email || !password) {
  return res.render("pages/login", {
   error: "Please enter both email and password.",  });}
 const sql = `SELECT * FROM logindb WHERE email = ?`;
 connection.query(sql, [email], (err, results) => {
  if (err) {
   console.error("Database Fetch Error:", err);
   return res.render("pages/login", {
    error: "Database error. Please try again later.",   });  }
  if (results.length === 0) {
   return res.render("pages/login", {
```

```
        error: "No user found with this email.", }); }
    const user = results[0];
    if (user.password === password) {
      console.log("Login successful:", user.email);
      req.session.user = user.email;
      return res.redirect("/");
    } else {
      console.log("Invalid password for:", user.email);
      return res.render("pages/login", {
        error: "Invalid password. Please try again.",  });  } });});});
app.get("/register", (req, res) => {
  if (req.session.user) return res.redirect("/");
  res.render("pages/register", { error: null });});
app.post("/register", (req, res) => {
  const { fullname, email, password, confirmpassword, phone, state } = req.body;
  if (
    !fullname ||
    !email ||
    !password ||
    !confirmpassword ||
    !phone ||
    !state
  ) { return res.render("pages/register", { error: "All fields are required" });}
  if (password !== confirmpassword) {
    return res.render("pages/register", { error: "Passwords do not match" }); }
  const sql = `INSERT INTO registerdb (fullname, email, password, confirmpassword, phone, state)
               VALUES (?, ?, ?, ?, ?, ?)`;
  const values = [fullname, email, password, confirmpassword, phone, state];
```

```
 connection.query(sql, values, (err, results) => {
  if (err) {
    console.error("Database Insert Error:", err);
    return res.render("pages/register", {
      error: "Database error, please try again.",  });  }
  console.log("Registration successful!", results);
  req.session.user = email;
  res.redirect("/");  });});});
app.get("/logout", (req, res) => {
 req.session.destroy(() => res.redirect("/login"));});
app.get("/", isAuthenticated, (req, res) => {
 res.render("pages/index");});
app.get("/news", isAuthenticated, (req, res) => {
 const newsList = [  {
    date: "Mar 25, 2025",
    image: "/assets/news1.jpg",
    title: "Automotive advertising - fact or fiction?",
    description:
      "It’s high time the automobile industry tightens up its advertising claim",  }, {
    date: "Mar 18, 2025",
    image: "/assets/news2.jpg",
    title: "India's new EV policy offers a level playing field",
    description:
      "New and existing players will be able to benefit from this policy and schemes",  },{
    date: "Oct 28, 2025",
    image: "/assets/news3.webp",
    title: "Honda 0 Series SUV to launch in India in 2026",
    description:
```

"The all-electric SUV will be imported to India as a completely built unit (CBU).", }, {

date: "Oct 24, 2025",

image: "/assets/news4.avif",

title: "Yamaha to showcase radical EV",

description: "Hybrid two-wheeler concepts at Tokyo Motor Show", }, {

date: "Oct 15, 2025",

image: "/assets/news5.avif",

title: "Nitin Kohli on demand for premium Volkswagens",

description:

"TIn conversation with Autocar India, Volkswagen India's new Brand Director Nitin Kohli talks of his focus areas, developing VW's image, GST impact on sales and more.", }, {

date: "Sept 05, 2025",

image: "/assets/news6.webp",

title: "2026 Porsche 911 Turbo S video review",

description:

"The Porsche 911 Turbo S has got an overhaul and is now a whole different beast. What's changed, and how does it all come together", }, {

date: "Aug 20, 2025",

image: "/assets/news7.avif",

title:

"Maruti Baleno turns 10: evolution of India's most successful premium hatchback",

description:

"Ten years on, Baleno still defines success in India’s hatchback market.", }, {

date: "Sept 07, 2025",

image: "/assets/news8.avif",

title: "Italian GT Sprint: Mahaveer Raghunathan wins at Monza",

description:

"TRaghunathan and co-driver Ferrari also finish runners up in the GT3 Pro-Am class of the championship.", }, {

date: "Sept 10, 2025",

```
    image: "/assets/news9.avif",
    title: "2025 Tata Sierra to launch on November 25",
    description:
     "The new Sierra will slot above the Curvv and compete with the Hyundai Creta, Maruti Grand Vitara and other midsize SUVs.",    },  ];
 res.render("pages/news", { newsList });});
app.get("/reviews", isAuthenticated, (req, res) => {
 const reviewsList = [{
    title: "Toyota Innova HyCross – Petrol GX Variant | Auto Expo 2023",
    video: "https://www.youtube.com/embed/QQ-zS7RprDw",
    desc: "Muscular SUV stance with powerful performance.",
  },  {
    title: "Fortuner Ka Big Daddy | Auto Expo 2023",
    video: "https://www.youtube.com/embed/Afim_iA3038",
    desc: "Refined 6-speed diesel AT with premium SUV aura.",
  },   {
    title: "Lexus LX500d – ₹3.8 Crore Luxury Beast",
    video: "https://www.youtube.com/embed/cJlNpZUAe9M",
    desc: "Twin-turbo V6 powerhouse with ultimate comfort.",
  },  ];
 const compareList = [{
    title: "Nexon vs Sonet vs Brezza vs Venue – MAHA COMPARISON",
    video: "https://www.youtube.com/embed/XUEMRPppTF0",
    desc: "Ultimate compact SUV showdown across real-world tests.",
  },  {
    title: "Baleno vs Altroz vs i20 – Which one wins?",
    video: "https://www.youtube.com/embed/w5qDwXK472I",
    desc: "Performance, space, safety — deep dive review.",
```

```
  }, {
    title: "Verna vs City vs Slavia vs Virtus",
    video: "https://www.youtube.com/embed/CT0WS2868CI",
    desc: "Leading sedans face off with detailed analysis.",
  }, ];
 res.render("pages/reviews", { reviewsList, compareList })});
app.get("/gallery", isAuthenticated, (req, res) => res.render("pages/gallery"));
app.get("/products", isAuthenticated, (req, res) =>
 res.render("pages/products"));
app.get("/cart", isAuthenticated, (req, res) => res.render("pages/cart"));
app.get("/about", isAuthenticated, (req, res) => res.render("pages/about"));
app.get("/contact", isAuthenticated, (req, res) => res.render("pages/contact"));
const PORT = 3000;
app.listen(PORT, () =>
 console.log(`TurboTalks running at http://localhost:${PORT}`));
```

Code: about.ejs:

```
<!DOCTYPE html>
<html lang="en">
 <head>
  <%- include("../partials/head") %>
  <style>
   body {
    background-color: #ff7244;}
   .hero-about {
    min-height: 50vh;
    display: flex;
    align-items: center;
    justify-content: center;
```

```
    color: white;
    text-align: center;   }
  .hero-about h1 {
    font-size: 3rem;
    font-weight: 700; }
  .about-section {
    background-color: #fff;
    border-radius: 15px;
    padding: 50px 30px;
    margin: -50px auto 50px auto;
    max-width: 1200px; }
  .about-section p {
    color: #555;
    font-size: 1.1rem; }
  .vision-card {
    background-color: #fff8f5;
    border-radius: 15px;
    padding: 30px;
    text-align: center;
    box-shadow: 0 5px 15px rgba(0, 0, 0, 0.1);
    transition: transform 0.3s, box-shadow 0.3s; }
  .vision-card:hover {
    transform: translateY(-8px);
    box-shadow: 0 12px 25px rgba(0, 0, 0, 0.15); }
  .vision-card h4 {
    font-weight: 600;
    margin-bottom: 15px;}
</style>
```

```
</head>
<body>
 <%- include("../partials/header", {name: 'Vikrant'}) %>
 <section class="hero-about">
  <div class="container">
   <h1>About TurboTalks</h1>
   <p class="lead">
    Your ultimate destination for car news, reviews & insights
   </p>
  </div>
 </section>
 <section class="about-section">
  <div class="container">
   <div class="row align-items-center g-5">
    <div class="col-md-6">
     <img
      src="/assets/me.jpg"
      alt="Car"
      class="img-fluid rounded shadow-lg"/>
    </div>
    <div class="col-md-6">
     <h2>About Me</h2>
     <p>
      Hi! I am Vikrant Vani, the creator of TurboTalks. I am passionate
      about cars and automotive innovation. Through this blog, I share
      the latest car news, expert reviews, and stunning visuals of
      supercars, classics, and concept vehicles.
     </p>
```

```
        <p>
          TurboTalks is designed to keep car enthusiasts informed, inspired,
          and engaged with high-quality content and insights into the
          ever-evolving automotive world.
        </p>
      </div>
    </div>
  </div>
</section>
<section class="container my-5">
  <div class="row g-4 text-center">
    <div class="col-md-6">
      <div class="vision-card h-100">
        <h4>My Vision</h4>
        <p>
          To create the most engaging and informative car blog, where
          automotive enthusiasts can discover news, reviews, and insights in
          one place.
        </p>
      </div>
    </div>
    <div class="col-md-6">
      <div class="vision-card h-100">
        <h4>My Mission</h4>
        <p>
          Deliver accurate news, expert reviews, and high-quality content
          that keeps car lovers informed and inspired to explore the world
          of automobiles.
```

```
          </p>
         </div>
       </div>
     </div>
   </section>
   <%- include("../partials/footer") %>
  </body>
</html>
```

Code: contact.ejs:

```
<!DOCTYPE html>
<html lang="en">
 <head>
   <%- include("../partials/head") %>
   <style>
     body {
       background-color: #ff7244;}
     .hero-contact {
       min-height: 40vh;
       display: flex;
       align-items: center;
       justify-content: center;
       text-align: center;
       color: white; }
     .contact-section {
       max-width: 700px;
       margin: -50px auto 50px auto;}
     .contact-card {
       border-radius: 15px;
```

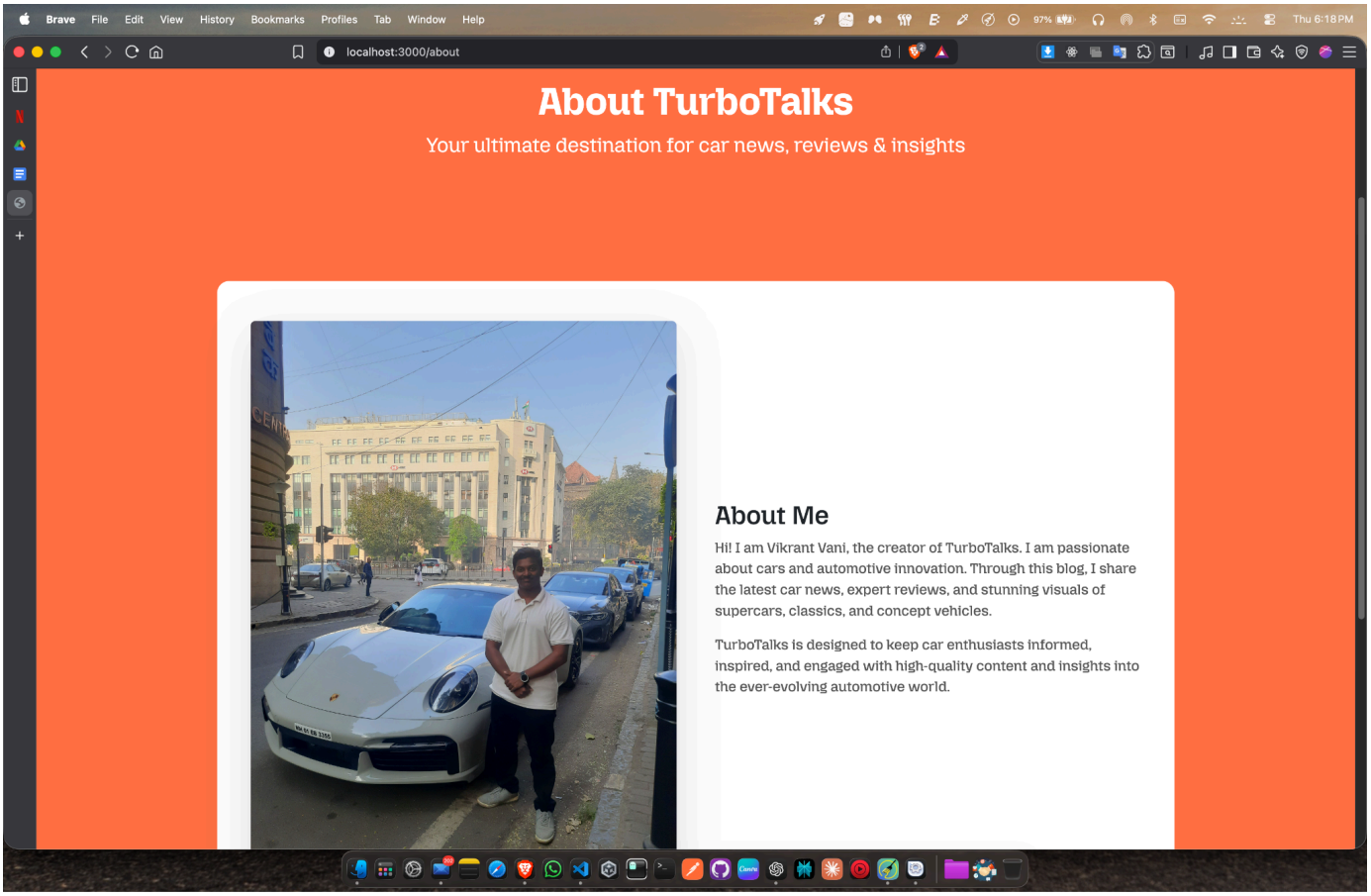
```
    padding: 40px 30px;
    background-color: #fff;
    box-shadow: 0 5px 20px rgba(0, 0, 0, 0.1); }
  .contact-card h2 {
    font-weight: 700;
    margin-bottom: 25px;
    color: #ff7244; }
  .contact-card .form-control:focus {
    box-shadow: 0 0 5px #ff7244;
    border-color: #ff7244;}
  .contact-card button {
    background-color: #ff7244;
    border: none;}
  .contact-card button:hover {
    background-color: #e65c30; }
  .social-links {
    margin-top: 25px;
    text-align: center; }
  .social-links a {
    margin: 0 10px;
    font-size: 1.5rem;
    color: #ff7244;
    transition: color 0.3s; }
  .social-links a:hover {
    color: #e65c30; }
 </style>
</head>
<body>
```

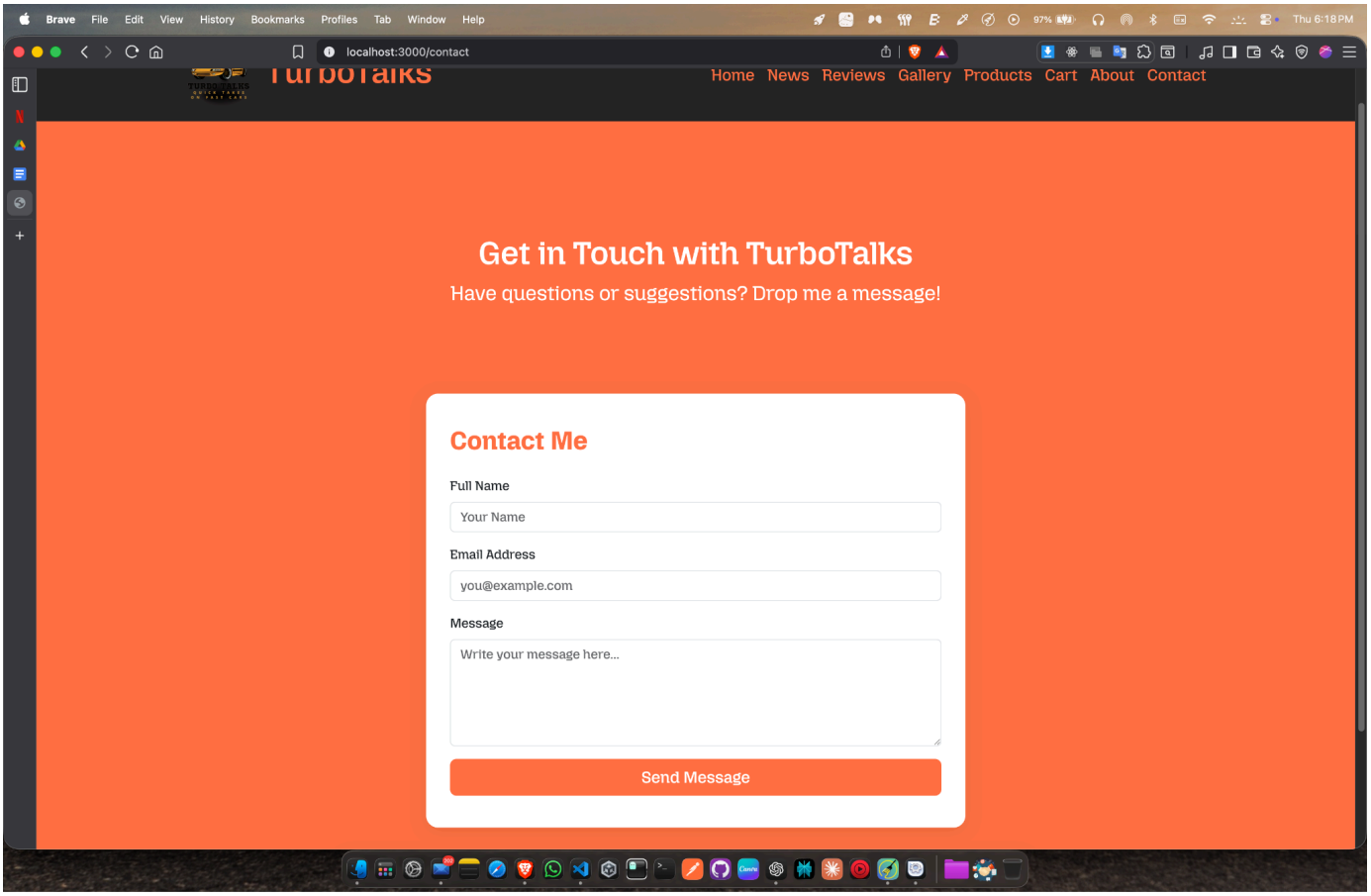
```
<%- include("../partials/header") %>
<section class="hero-contact">
  <div class="container">
    <h1>Get in Touch with TurboTalks</h1>
    <p class="lead">Have questions or suggestions? Drop me a message!</p>
  </div>
</section>
<section class="contact-section">
  <div class="container">
    <div class="contact-card">
      <h2>Contact Me</h2>
      <form action="/send-message" method="POST">
        <div class="mb-3">
          <label for="name" class="form-label">Full Name</label>
          <input
            type="text"
            class="form-control"
            id="name"
            name="name"
            placeholder="Your Name"
            required />
        </div>
        <div class="mb-3">
          <label for="email" class="form-label">Email Address</label>
          <input
            type="email"
            class="form-control"
            id="email"
```

```
            name="email"
            placeholder="you@example.com"
            required/>
          </div>
          <div class="mb-3">
           <label for="message" class="form-label">Message</label>
           <textarea
            class="form-control"
            id="message"
            name="message"
            rows="5"
            placeholder="Write your message here..."
            required
           ></textarea>
          </div>
          <button type="submit" class="btn btn-lg w-100 text-white">
           Send Message
          </button>
         </form>
        </div>
       </div>
     </section>
     <%- include("../partials/footer") %>
   </body>
</html>
```

Output:

```
PROBLEMS  OUTPUT  TERMINAL  DEBUG CONSOLE  PORTS

vikrantvani@Vikrants-MacBook-Air turbotalks % node index.js
TurboTalks running at http://localhost:3000
Login successful: vickyvani2003@gmail.com
```





Conclusion:

This experiment demonstrates how to set up a Node.js + Express.js environment and create a basic web server capable of handling multiple routes. Using Express simplifies server creation, reduces boilerplate code, and enhances flexibility for future additions like middleware, database integration, or templating engines (e.g., EJS). It forms the foundation for modern full-stack web development.

Experiment No.4

Problem Statement:

Monolithic route definitions make server code hard to maintain and extend. There is also no centralized logging of incoming HTTP requests, which makes debugging and auditing difficult. The task is to refactor routing into modular route files and add middleware that logs request method, URL and timestamp for every request.

Objective:

- Split server routes into modular files (e.g., routes/userRoutes.js, routes/productRoutes.js) for clarity and maintainability.
- Create middleware that logs method, URL and timestamp for each incoming request.
- Apply the middleware globally so logs capture all traffic and aid debugging.

Theory:

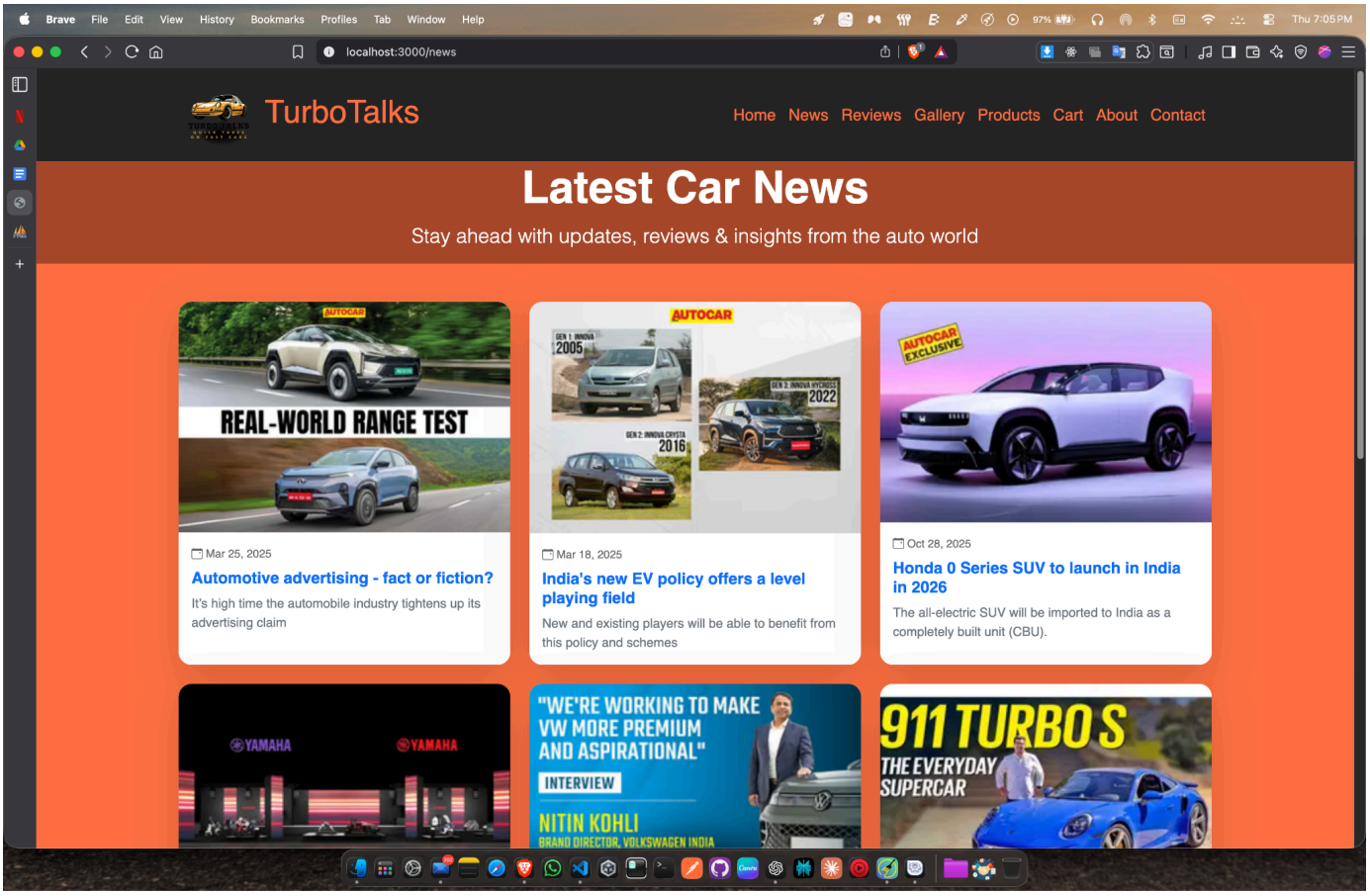
Modular routing separates concerns: each route module encapsulates related endpoints and middleware, making the codebase easier to navigate, test and scale. Express middleware is a function that receives (req, res, next) and can inspect/modify the request/response or perform side effects (like logging). Global middleware runs for every request, so it's ideal for cross-cutting concerns such as logging, authentication, or error handling.

Code: news.ejs:

```
<%- include("../partials/head") %>

<body style="background-color: #FF7244; color: #fff; font-family: 'Poppins', sans-serif;">

 <%- include("../partials/header") %>

 <section class="hero text-center text-white d-flex align-items-center justify-content-center position-relative">

  <div class="overlay position-absolute top-0 start-0 w-100 h-100" style="background: rgba(0, 0, 0, 0.4);"></div>

  <div class="hero-content position-relative z-2">

   <h1 class="display-4 fw-bold">Latest Car News</h1>

   <p class="lead">Stay ahead with updates, reviews & insights from the auto world</p>

  </div>  </section>

 <div class="container my-5">

  <div class="row g-4">

   <% newsList.forEach(function(news) { %>

   <div class="col-md-6 col-lg-4">

    <div class="card h-100 shadow-lg border-0 news-card rounded-4 overflow-hidden">
```

```
        <img
          src="<%= news.image %>"
          class="card-img-top"
          alt="Car News Image"/>
        <div class="card-body bg-white text-dark">
          <p class="card-text text-muted small mb-2"><i class="bi bi-calendar-event"></i> <%= news.date %></p>
          <h5 class="card-title fw-bold text-primary"><%= news.title %></h5>
          <p class="card-text text-secondary"><%= news.description %></p>
        </div>  </div>  </div>  <% }) %>
    </div>  </div>
  <%- include("../partials/footer") %>
</body>
</html>
```

Output:



Conclusion:

Using modular route files and a logging middleware improves code organization, makes the server easier to grow, and provides a reliable audit trail of HTTP activity. This approach reduces technical debt and simplifies debugging and future feature addition.

Experiment No.5

Problem Statement:

Static pages cannot accept or persist user input. The problem is to build an HTML form (e.g., add item or feedback) and write server-side logic to handle the POST submission, then show a confirmation view containing the submitted data.

Objective:

- Create a form with appropriate name fields and method POST.
- Implement an Express POST route to receive form data (use express.urlencoded).
- Validate the input and render a confirmation EJS view that displays submitted data dynamically.

Theory:

HTML forms send user input to the server via HTTP methods (GET/POST). On the server, express.urlencoded() parses application/x-www-form-urlencoded payloads. After validation, the server can respond by rendering a view (server-side render) that includes the submitted values — providing immediate feedback and creating a dynamic user flow without a client-side single-page app.

Code: login.ejs:

```
<!DOCTYPE html>

<html lang="en">

 <head>

  <meta charset="UTF-8" />

  <title>TurboTalks | Login</title>

  <link rel="icon" type="image/png" href="/assets/logo.png" />

  <link

   href="https://cdn.jsdelivr.net/npm/bootstrap@5.3.3/dist/css/bootstrap.min.css"

   rel="stylesheet"/>

  <style>

   body {

    background: #0c0c0c;

    color: white;

    font-family: "Poppins", sans-serif;}

   .login-container {
```

```
  max-width: 450px;
  background: #1a1a1a;
  padding: 35px;
  border-radius: 12px;
  margin: 80px auto;
  box-shadow: 0 0 25px rgba(255, 114, 68, 0.4);}
h3 {
  text-align: center;
  color: #ff7244;
  font-weight: bold;}
.btn-google {
  background: white;
  color: #000;
  font-weight: 600;}
input {
  background: #111;
  color: white;}
input:focus {
  box-shadow: 0 0 10px #ff7244;
  border-color: #ff7244 !important;}
.login-btn {
  background: #ff7244;
  border: none;
  font-weight: 600;}
.login-btn:hover {
  background: #ff5722; }
.show-pass-btn {
  font-size: 14px;
```

```
    border: none;
    background: transparent;
    color: #ff7244;
    cursor: pointer;}
  .register-link {
    color: #ff7244;
    font-weight: 500;
    text-decoration: none; }
  .register-link:hover {
    text-decoration: underline;}
  .error-msg {
    color: #ff4444;
    text-align: center;
    margin-bottom: 10px;
    font-weight: 500;}
 </style>
</head>
<body>
 <div class="login-container">
  <h3>Welcome to TurboTalks 🚗</h3>
  <p class="text-center mt-2">Join the TurboTalks India Community</p>
  <% if (typeof error !== "undefined") { %>
  <div class="error-msg"><%= error %></div>
  <% } %> <a
   href="https://accounts.google.com/signin"
   class="btn btn-google w-100 mt-3">
   Sign In with Google
  </a>
```

```
<p class="text-center mt-3">OR</p>
<form action="/login" method="POST" id="loginForm">
 <div class="mb-3">
  <input
   type="email"
   class="form-control"
   placeholder="Enter your Email"
   id="email"
   name="email"
   required/>
 </div>
 <div class="mb-2">
  <input
   type="password"
   class="form-control"
   placeholder="Enter your Password"
   id="password"
   name="password"
   required
  />
 </div>
 <button class="show-pass-btn" type="button" id="togglePass">
  Show Password
 </button>
 <button
  class="btn login-btn w-100 mt-4"
  type="submit"
  id="submit-btn"
```

```
    disabled >
    LOGIN
   </button>
   <p class="text-center mt-3">
    New to TurboTalks?
    <a href="/register" class="register-link">Register Here</a>
   </p>
  </form>
</div>
<script>
 const email = document.getElementById("email");
 const password = document.getElementById("password");
 const submitBtn = document.getElementById("submit-btn");
 email.addEventListener("input", validate);
 password.addEventListener("input", validate);
 function validate() {
  const validEmail = email.value.includes("@") && email.value.length >= 6;
  const validPassword = password.value.length >= 6;
  submitBtn.disabled = !(validEmail && validPassword);}
 document
  .getElementById("togglePass")
  .addEventListener("click", function () {
   if (password.type === "password") {
    password.type = "text";
    this.textContent = "Hide Password";
   } else {
    password.type = "password";
    this.textContent = "Show Password"; }});
```

```
  </script>
 </body>
</html>
```

Code: register.ejs:

```
<!DOCTYPE html>
<html lang="en">
 <head>
  <meta charset="UTF-8" />
  <title>TurboTalks | Register</title>
  <link rel="icon" type="image/png" href="/assets/logo.png" />
  <link
   href="https://cdn.jsdelivr.net/npm/bootstrap@5.3.3/dist/css/bootstrap.min.css"
   rel="stylesheet" />
  <style>
   body {
    background: #0c0c0c;
    color: white;
    font-family: "Poppins", sans-serif;}
   .register-container {
    max-width: 550px;
    background: #1a1a1a;
    padding: 35px;
    border-radius: 12px;
    margin: 70px auto;
    box-shadow: 0 0 25px rgba(255, 114, 68, 0.4); }
   h3 {
    text-align: center;
    color: #ff7244;
```

```
      font-weight: bold; }
    .form-control,
    .form-select {
      background: #111;
      color: white;
      border: 1px solid #444; }
    .form-control:focus,
    .form-select:focus {
      border-color: #ff7244 !important;
      box-shadow: 0 0 10px #ff7244 !important; }
    .register-btn {
      background: #ff7244;
      border: none;
      font-weight: 600;
      width: 100%; }
    .register-btn:hover {
      background: #ff5722; }
    .login-link {
      color: #ff7244;
      text-decoration: none;
      font-weight: 500; }
    .login-link:hover {
      text-decoration: underline;}
  </style>
</head>
<body>
  <div class="register-container">
    <h3>Create Your TurboTalks Account 🚗</h3>
```

```
<p class="text-center mt-2">
  Join the community of car enthusiasts today
</p>
<% if (error) { %>
<div class="alert alert-danger text-center py-2"><%= error %></div>
<% } %>
<form action="/register" method="POST" id="registerForm">
  <div class="mb-3">
    <label class="form-label">Full Name</label>
    <input
      type="text"
      name="fullname"
      class="form-control"
      placeholder="Enter your full name"
      required />
  </div>
  <div class="mb-3">
    <label class="form-label">Email</label>
    <input
      type="email"
      name="email"
      class="form-control"
      placeholder="Enter your email"
      required />
  </div>
  <div class="mb-3">
    <label class="form-label">Password</label>
    <input
```

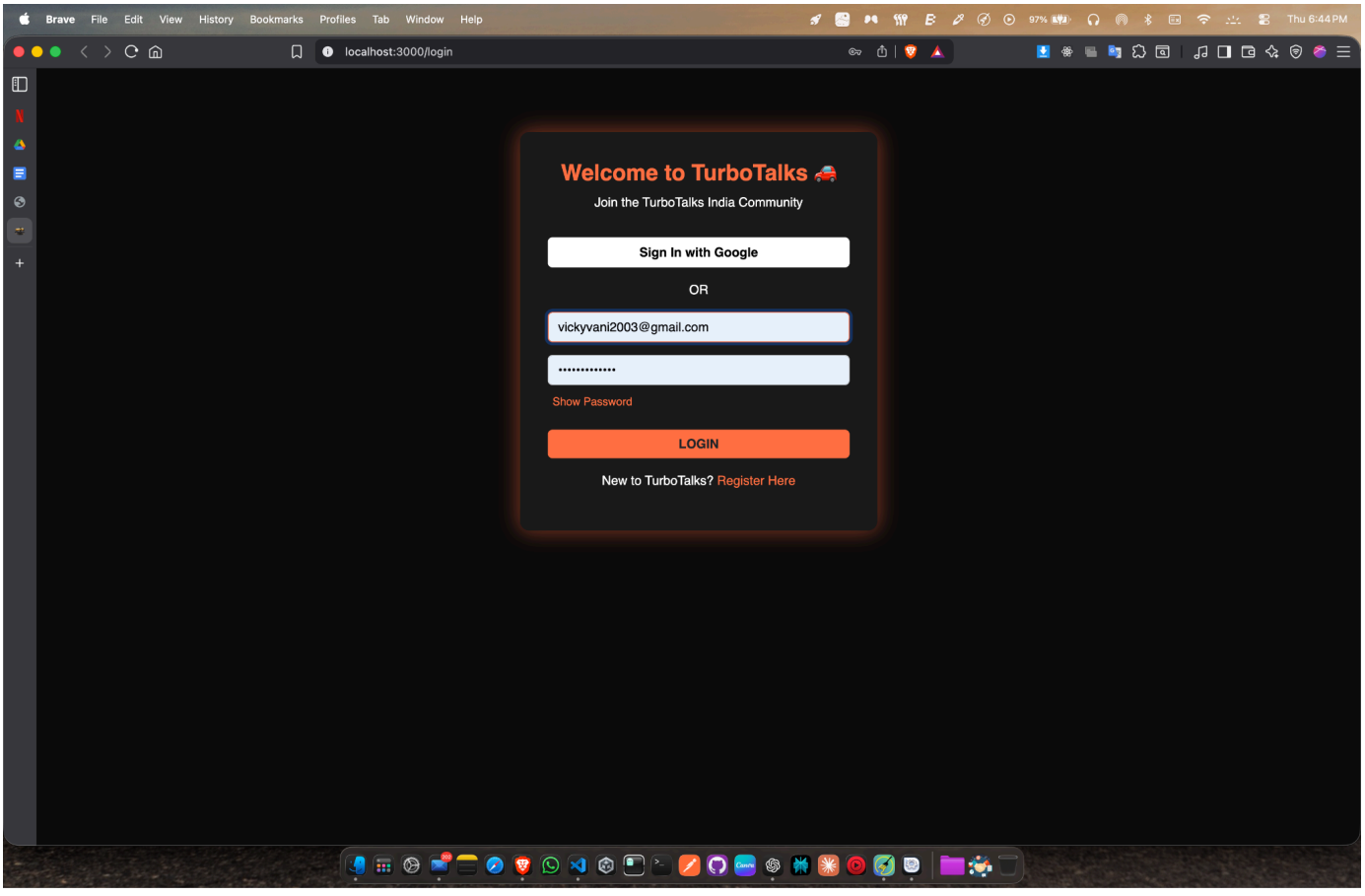
```
    type="password"
    name="password"
    id="password"
    class="form-control"
    placeholder="Create a password"
    required />
</div>
<div class="mb-3">
  <label class="form-label">Confirm Password</label>
  <input
    type="password"
    name="confirmpassword"
    id="confirmPassword"
    class="form-control"
    placeholder="Confirm your password"
    required />
</div>
<div class="mb-3">
  <label class="form-label">Contact Number</label>
  <input
    type="text"
    name="phone"
    class="form-control"
    placeholder="Enter your phone number"
    maxlength="10"
    required />
</div>
```

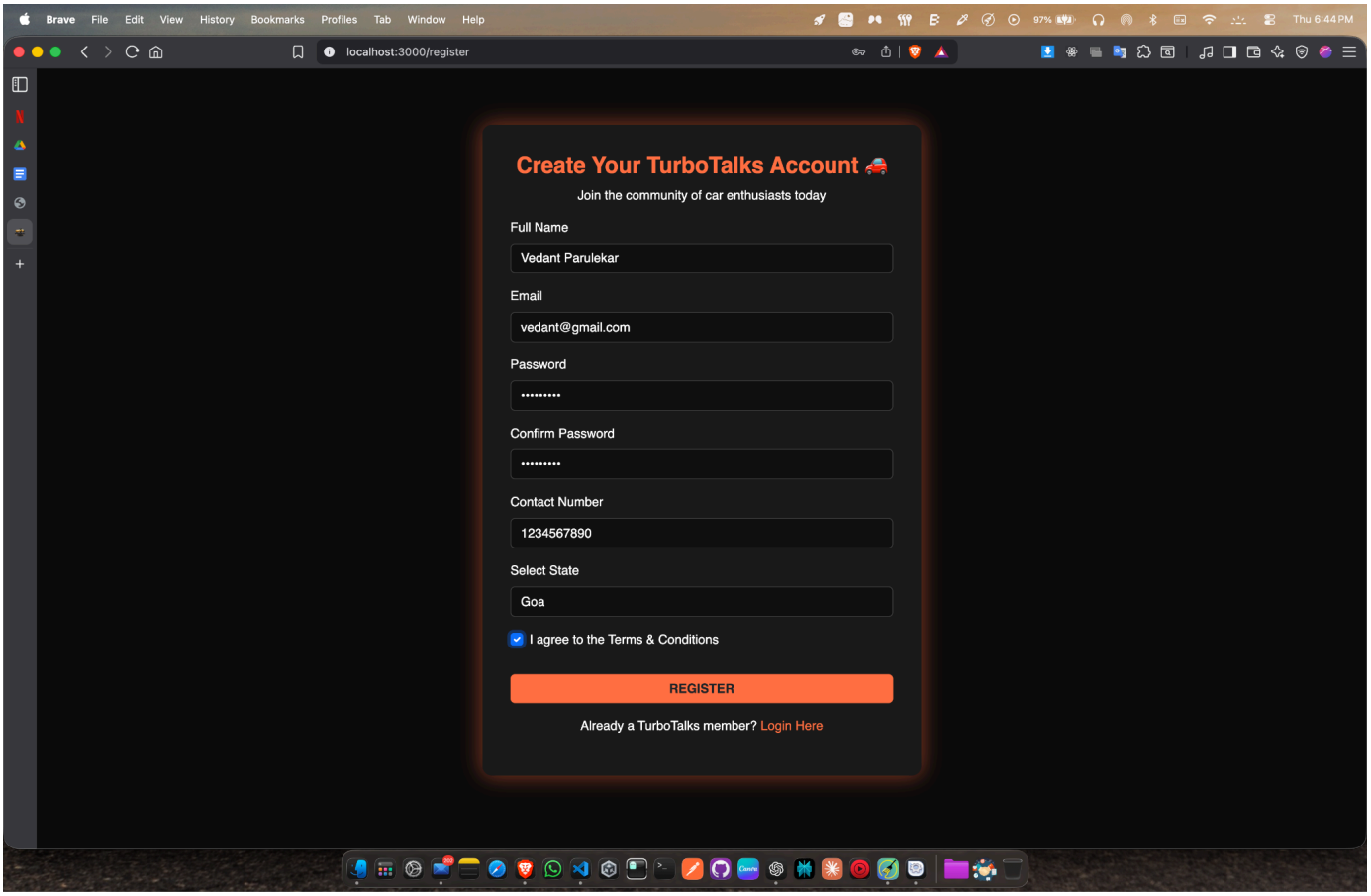
```
<div class="mb-3">
  <label class="form-label">Select State</label>
  <select name="state" class="form-select" required>
    <option value="">-- Select State --</option>
    <option>Maharashtra</option>
    <option>Karnataka</option>
    <option>Gujarat</option>
    <option>Tamil Nadu</option>
    <option>Uttar Pradesh</option>
    <option>Delhi</option>
    <option>West Bengal</option>
    <option>Goa</option>
  </select>
</div>
<div class="form-check mb-3">
  <input
    class="form-check-input"
    type="checkbox"
    id="termsCheck"
    required />
  <label class="form-check-label" for="termsCheck">
    I agree to the Terms & Conditions
  </label>
</div>
<button
  type="submit"
  class="btn register-btn mt-3"
  id="registerBtn"
```

```
      disabled>
      REGISTER
    </button>
    <p class="text-center mt-3">
      Already a TurboTalks member?
      <a href="/login" class="login-link">Login Here</a>
    </p>
  </form>
</div>
<script>
  const nameInput = document.querySelector('input[name="fullname"]');
  const emailInput = document.querySelector('input[name="email"]');
  const passInput = document.querySelector('input[name="password"]');
  const confirmInput = document.querySelector("#confirmPassword");
  const termsCheck = document.querySelector("#termsCheck");
  const registerBtn = document.querySelector("#registerBtn");
  const validate = () => {
    const validName = nameInput.value.length >= 3;
    const validEmail = emailInput.value.includes("@");
    const validPass = passInput.value.length >= 6;
    const passMatch = passInput.value === confirmInput.value;
    registerBtn.disabled = !(
      validName &&
      validEmail &&
      validPass &&
      passMatch &&
      termsCheck.checked ); };
```

```
    [nameInput, emailInput, passInput, confirmInput, termsCheck].forEach(
      (el) => {
        el.addEventListener("input", validate); } );
  </script>
 </body>
</html>
```

Output:





Conclusion:

Handling POST submissions on the server enables interactive features like adding products or feedback. When paired with basic validation and a confirmation view, it creates a reliable user experience and lays the groundwork for persisting data to a database.

Experiment No.6

Problem Statement:

Static HTML requires manual updates for content changes. The goal is to render lists of items (products, posts, services) dynamically on pages using EJS and server-provided data.

Objective:

- Use EJS to render a list of items passed from the server.
- Demonstrate rendering using either hardcoded in-memory arrays or temporary server-side datasets.
- Use EJS loops (<% items.forEach(...) %>) to output repeated components like cards.

Theory:

Server-side rendering (SSR) generates HTML on the server based on data, then sends finished markup to the client. SSR reduces client complexity, improves first-load SEO, and integrates naturally with templating engines like EJS. Passing arrays/objects from Express into a view allows templates to iterate and render components consistently.

Code: products.ejs:

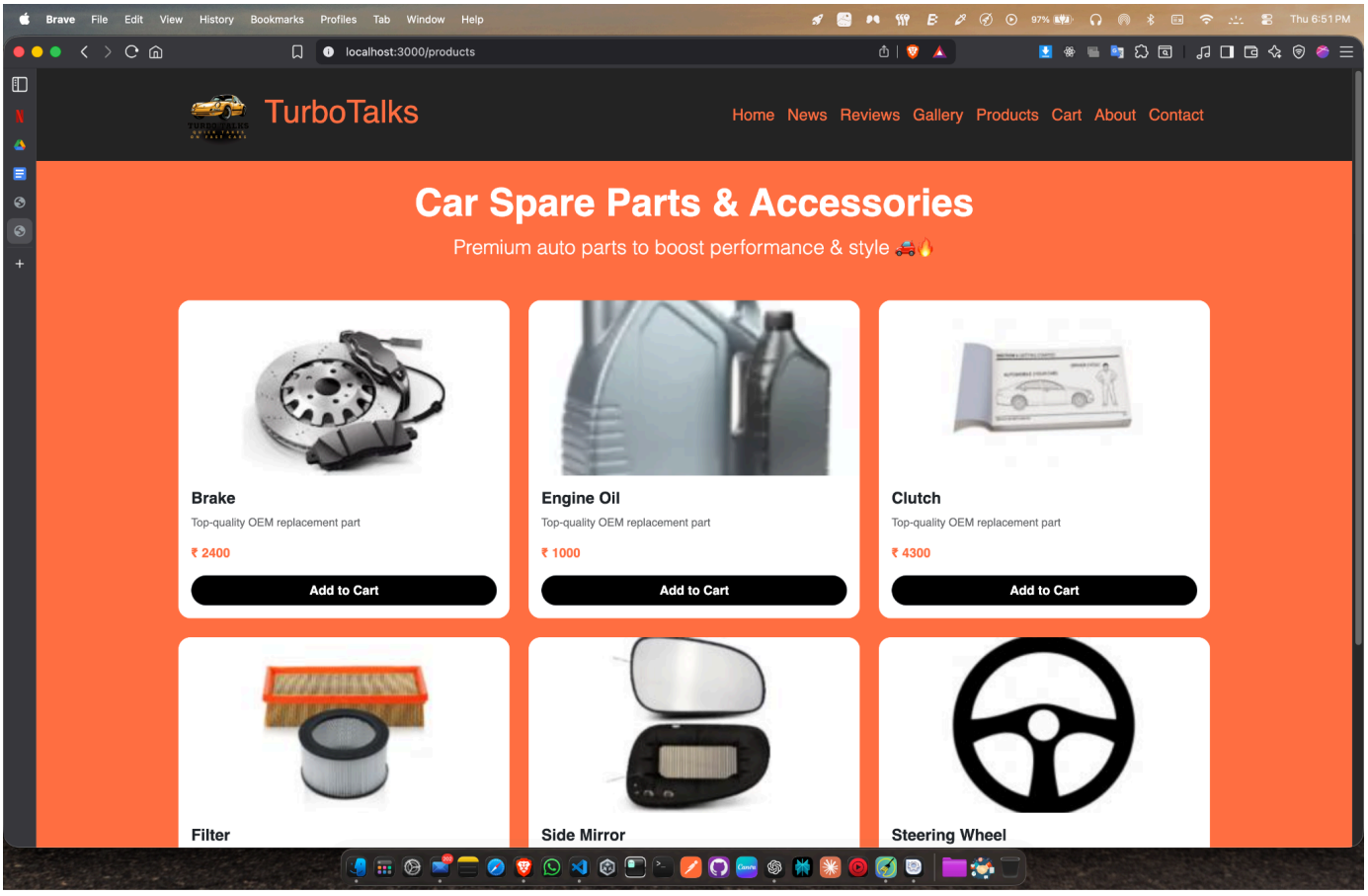
```
<!DOCTYPE html>

<html lang="en">

 <head>

  <%- include("../partials/head") %>

  <style>

   body {

    background-color: #ff7244;

    color: #fff;

    font-family: "Poppins", sans-serif; }

   .product-card {

    border: none;

    background: #ffffff;

    border-radius: 15px;

    transition: transform 0.3s ease, box-shadow 0.3s ease;

    overflow: hidden;  }
```

```
    .product-card:hover {
      transform: translateY(-5px);
      box-shadow: 0 12px 25px rgba(0, 0, 0, 0.3); }
    .product-card img {
      height: 220px;
      object-fit: cover; }
    .price {
      font-weight: bold;
      color: #ff7244; }
    .btn-dark {
      background-color: #000;
      border-radius: 30px;
      font-weight: 600;}
    h2 {
      font-weight: 700;
      text-shadow: 0 3px 10px rgba(0, 0, 0, 0.2); }
  </style>
</head>
<body>
  <%- include("../partials/header") %>
  <section class="hero text-center">
    <h1 class="display-5 fw-bold mt-4">Car Spare Parts & Accessories</h1>
    <p class="lead">Premium auto parts to boost performance & style 🚗🔥 </p>
  </section>
  <div class="container my-5">
    <div class="row g-4">
      <% const products = [ { name: 'Brake', price: 2400, image:
      '/assets/brake.jpg' }, { name: 'Engine Oil', price: 1000, image:
```

```
  '/assets/oil.webp' }, { name: 'Clutch', price: 4300, image:
  '/assets/clutch.jpg' }, { name: 'Filter', price: 2100, image:
  '/assets/filter.jpg' }, { name: 'Side Mirror', price: 1870, image:
  '/assets/mirror.jpg' }, { name: 'Steering Wheel', price: 5600, image:
  '/assets/steering.jpg' } ]; %> <% products.forEach(p => { %>
  <div class="col-md-6 col-lg-4">
   <div class="card product-card">
    <img src="<%= p.image %>" alt="<%= p.name %>" />
    <div class="card-body text-dark">
     <h5 class="card-title fw-bold"><%= p.name %></h5>
     <p class="card-text small text-muted">
      Top-quality OEM replacement part
     </p>
     <p class="price">₹ <%= p.price %></p>
     <button
      class="btn btn-dark w-100"
      onclick="addToCart('<%= p.name %>', <%= p.price %>, '<%= p.image %>')">
      Add to Cart
     </button>
    </div>
   </div>
  </div>
  <% }) %>
 </div>
</div>
<%- include("../partials/footer") %>
<script>
 let cart2 = JSON.parse(localStorage.getItem("cart2")) || [];
```

```
    function updateCart() {
      const cartBadge = document.querySelector(".cart-count");
      if (cartBadge) cartBadge.textContent = `(${cart2.length})`;}
    function addToCart(name, price, image) {
      cart2.push({ name, price, image });
      localStorage.setItem("cart2", JSON.stringify(cart2));
      updateCart(); }
    window.onload = updateCart;
  </script>
 </body>
</html>
```

Output:



Conclusion:

SSR with EJS is a simple, powerful way to serve dynamic lists. It keeps rendering logic centralized, improves SEO, and is ideal for content-driven pages such as product listings or blog indexes.

Experiment No.7

Problem Statement:

A list page alone is insufficient. Users need individual detail pages (e.g., product details) driven by the item selected. The problem is to implement route parameters that render item-specific pages dynamically.

Objective:

- Use Express route parameters (e.g., /products/:id) to identify items.
- Fetch the item data (from memory, DB, or dataset) by id and render a detail EJS page.
- Display properties like title, description, date and images on the detail page.

Theory:

Route parameters allow expressive, RESTful URLs. In Express, a parameter like :id becomes req.params.id. The server uses that id to fetch the correct resource and passes it to the template for server-side rendering. This pattern is the basis for resource-specific pages in web apps.

Code: cart.ejs:

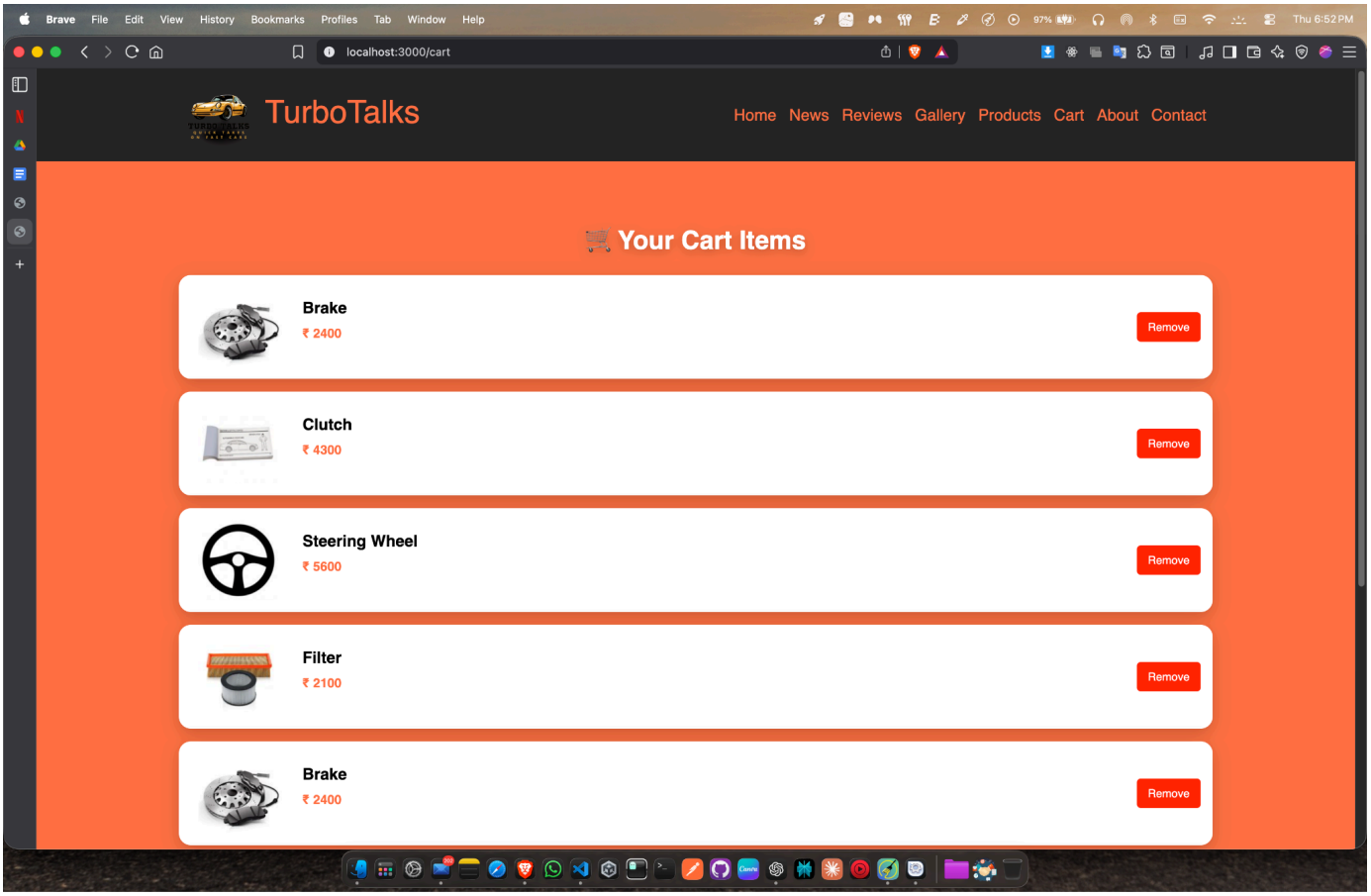
```
<!DOCTYPE html>

<html lang="en">

 <head>

  <%- include("../partials/head") %>

  <style>

   body {

    background-color: #ff7244;

    color: #ffffff;

    font-family: "Poppins", sans-serif; }

   .cart-container {

    margin-top: 80px;}

   .cart-card {

    background: #ffffff;

    color: #000;

    border-radius: 15px;

    overflow: hidden;
```

```
  padding: 15px;
  box-shadow: 0 6px 20px rgba(0, 0, 0, 0.25);
  display: flex;
  align-items: center; }
.cart-card img {
  width: 120px;
  height: 100px;
  object-fit: cover;
  border-radius: 10px;
  margin-right: 20px;}
.price {
  font-weight: bold;
  color: #ff7244; }
.remove-btn {
  background-color: #ff0000;
  color: white;
  border: none;
  padding: 8px 14px;
  border-radius: 5px;
  cursor: pointer;
  font-size: 14px;
  transition: 0.2s ease;}
.remove-btn:hover {
  background-color: #a70000; }
h2 {
  font-weight: 700;
  text-shadow: 0 4px 12px rgba(0, 0, 0, 0.3); }
```

```
    .empty-msg {
      text-align: center;
      padding: 50px;
      font-size: 22px;
      font-weight: 600; }
  </style>
</head>
<body>
  <%- include("../partials/header") %>
  <div class="container cart-container">
    <h2 class="text-center mb-4">🛒 Your Cart Items</h2>
    <div id="cartItemsContainer"></div>
  </div>
  <%- include("../partials/footer") %>
  <script>
    function removeItem(productName) {
      let cartItems = JSON.parse(localStorage.getItem("cart2"));
      cartItems = cartItems.filter((item) => item.name !== productName);
      localStorage.setItem("cart2", JSON.stringify(cartItems));
      displayCart();}
    function displayCart() {
      const cartItems = JSON.parse(localStorage.getItem("cart2")) || [];
      const container = document.getElementById("cartItemsContainer");
      container.innerHTML = "";
      if (cartItems.length === 0) {
        container.innerHTML = `<p class="empty-msg">Your cart is empty 🚗💨 <br><a href="/products" class="btn
btn-dark mt-3">Shop Now</a></p>`; }
      cartItems.forEach((item) => {
```

```
        const element = document.createElement("div");
        element.className = "cart-card mb-3";
        element.innerHTML = `
                <img src="${item.image}" alt="${item.name}">
                <div class="flex-grow-1">
                  <h5 class="fw-bold">${item.name}</h5>
                  <p class="price">₹ ${item.price}</p>
                </div>
                <button class="remove-btn" onclick="removeItem('${item.name}')">Remove</button> `;
        container.appendChild(element);
      });
      updateCartCount(cartItems.length);
    }
    function updateCartCount(count) {
      const badge = document.querySelector(".cart-count");
      if (badge) badge.textContent = `Cart (${count})`;
    }
    window.onload = displayCart;
  </script>
 </body>
</html>
```

Output:



Conclusion:

Dynamic detail routes provide a natural navigation model and improve user experience. They support deep linking, better SEO, and maintainable route definitions for item-specific pages.

Experiment No.8

Problem Statement:

Repeated markup (header, footer, navbar) in every view causes duplication and inconsistencies. The solution is to create reusable EJS partials and include them in all pages.

Objective:

- Create partial templates for header.ejs, footer.ejs, and navbar.ejs.
- Include these partials in pages using EJS <%- include('partials/header') %>.
- Provide a consistent layout and reduce duplication across views.

Theory:

Partials are fragments of templated markup that can be included in multiple views. They encourage DRY (Don't Repeat Yourself) principles and make it trivial to update global UI elements in one place. Combined with layout templates, partials help maintain consistent site structure and styling.

Code: head.ejs:

```
<meta charset="UTF-8" />

<meta name="viewport" content="width=device-width, initial-scale=1.0" />

<title>TurboTalks - Blog</title>

<style>

 @import
url("https://fonts.googleapis.com/css2?family=Mozilla+Headline:wght@200..700&family=Yatra+One&display=swap
");

</style>

<link

 href="https://cdn.jsdelivr.net/npm/bootstrap@5.3.7/dist/css/bootstrap.min.css"

 rel="stylesheet"/>

<link rel="stylesheet" href="/css/style.css" />
```

Code: header.ejs:

```
<nav class="navbar navbar-expand-lg">

 <div class="container">

   <img

    src="/assets/logo.png"

    alt="TurboTalks Logo"
```

```
    class="img-fluid logo me-2"  />
   <h1 class="navbar-brand">TurboTalks</h1>
   <div class="collapse navbar-collapse" id="navbarMenu">
    <ul class="navbar-nav ms-auto">
     <li class="nav-item"><a class="nav-link" href="/">Home</a></li>
     <li class="nav-item"><a class="nav-link" href="/news">News</a></li>
     <li class="nav-item">
      <a class="nav-link" href="/reviews">Reviews</a>
     </li>
     <li class="nav-item">
      <a class="nav-link" href="/gallery">Gallery</a>
     </li>
     <li class="nav-item">
      <a class="nav-link" href="/products">Products</a>
     </li>
     <li class="nav-item"><a class="nav-link" href="/cart">Cart</a></li>
     <li class="nav-item"><a class="nav-link" href="/about">About</a></li>
     <li class="nav-item">
      <a class="nav-link" href="/contact">Contact</a>
     </li>
    </ul>
   </div>
  </div>
</nav>
```

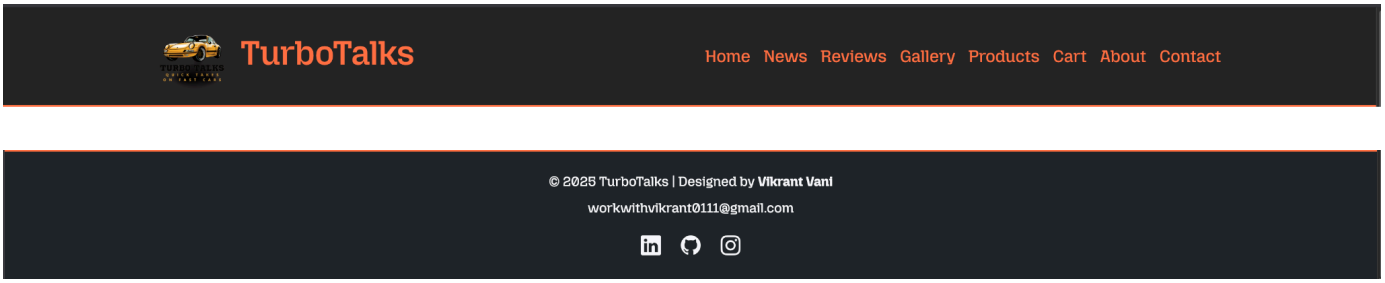
Code: footer.ejs:

```
<footer class="bg-dark text-light py-4 mt-5">
 <div class="container text-center">
  <p class="mb-2">
```

```
    © 2025 TurboTalks | Designed by <strong>Vikrant Vani</strong>
  </p>
  <p class="mb-3"> <a
    href="mailto:workwithvikrant0111@gmail.com"
    class="text-light text-decoration-none" >
    workwithvikrant0111@gmail.com
   </a>
  </p>
  <div class="d-flex justify-content-center gap-4"> <a
    href="https://www.linkedin.com/in/vikrant-vani-96b329209/"
    class="text-light fs-4"
    target="_blank">
    <i class="bi bi-linkedin"></i>
   </a> <a
    href="https://github.com/codeby-vikrant"
    class="text-light fs-4"
    target="_blank">
    <i class="bi bi-github"></i>
   </a> <a
    href="https://www.instagram.com/vikrantsvani"
    class="text-light fs-4"
    target="_blank">
    <i class="bi bi-instagram"></i>
   </a>
  </div>
 </div>
</footer>
```

```
<link
  rel="stylesheet"
  href="https://cdn.jsdelivr.net/npm/bootstrap-icons@1.11.3/font/bootstrap-icons.css"
/>
```

Output:



Conclusion:

Using EJS partials yields a modular, maintainable template system. It speeds development, enforces UI consistency, and simplifies global updates such as changing navigation or site branding.

Experiment No.9

Problem Statement:

Web applications often need to persist and manipulate data. The task is to connect the Express app to MySQL and implement create, read, update and delete (CRUD) operations through routes.

Objective:

- Configure a MySQL connection (e.g., using mysql2).
- Build Express routes to perform CRUD:
- Create: POST /items → insert row into a table.
- Read: GET /items and GET /items/:id → fetch rows.
- Update: PUT /items/:id or POST /items/:id/edit → modify row.
- Delete: DELETE /items/:id or POST /items/:id/delete → remove row.
- Use parameterized queries to prevent SQL injection.

Theory:

A relational database like MySQL stores persistent data in tables. Node.js connects to MySQL via drivers (mysql2) that support prepared statements and connection pooling. CRUD operations are mapped to HTTP verbs and routes; parameterized queries (? placeholders) protect against injection and ensure safe value substitution.

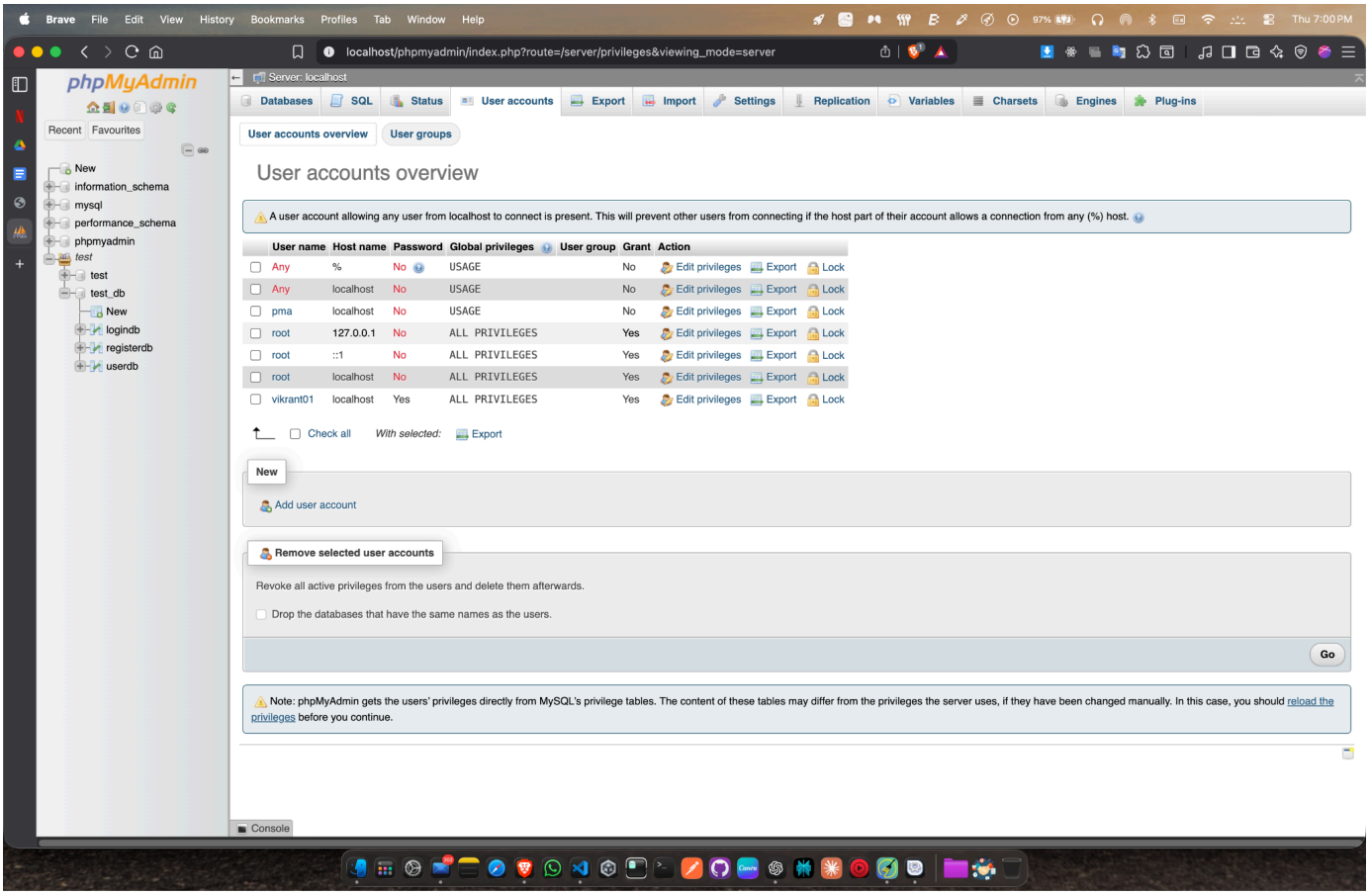
Code:

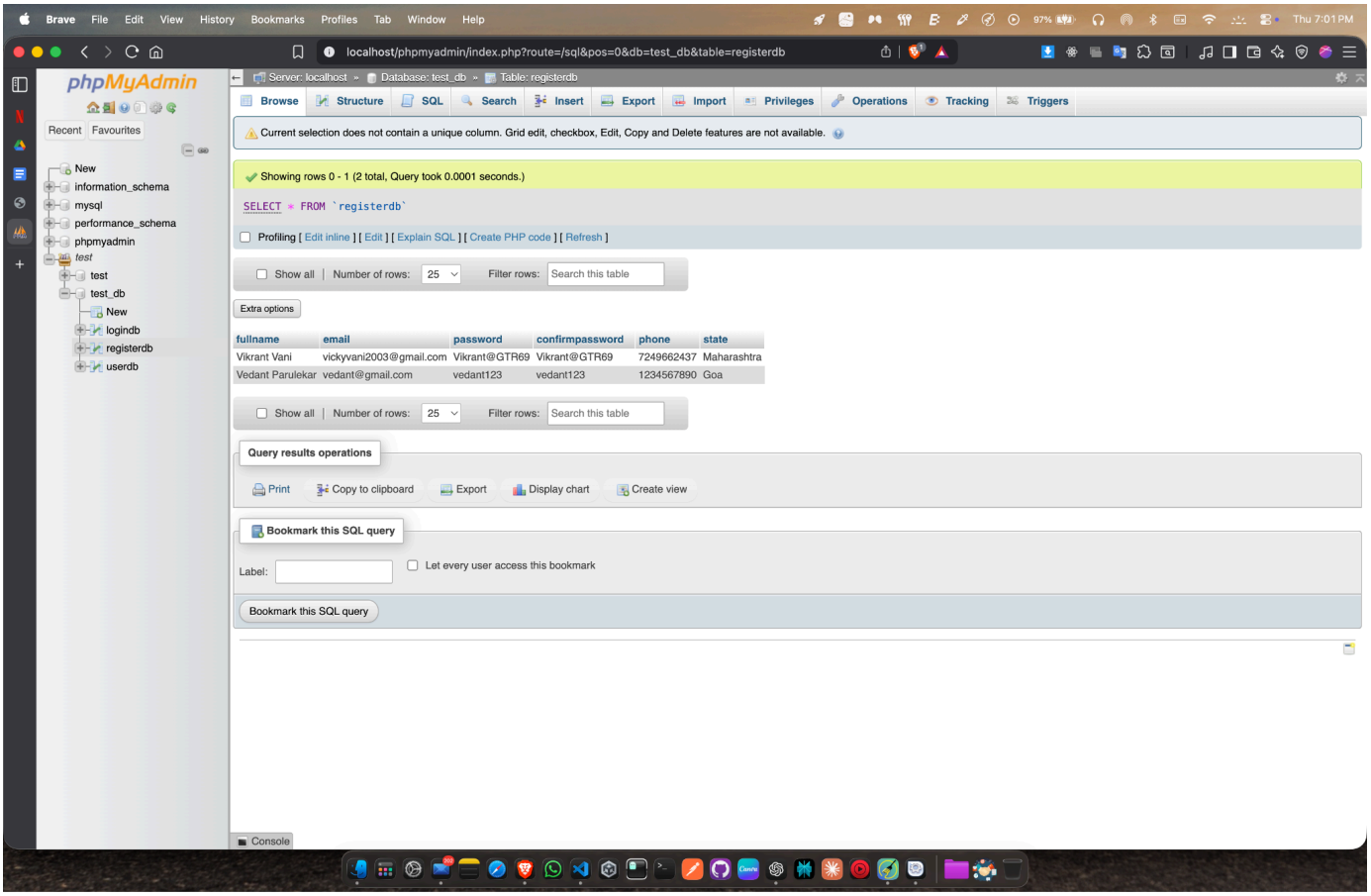
```
app.post("/login", (req, res) => {

 const { email, password } = req.body;

 if (!email || !password) {

  return res.render("pages/login", {

   error: "Please enter both email and password.",  });}

 const sql = `SELECT * FROM logindb WHERE email = ?`;

 connection.query(sql, [email], (err, results) => {

  if (err) {

   console.error("Database Fetch Error:", err);

   return res.render("pages/login", {

    error: "Database error. Please try again later.", });  }

  if (results.length === 0) {

   return res.render("pages/login", {

    error: "No user found with this email.",});}
```

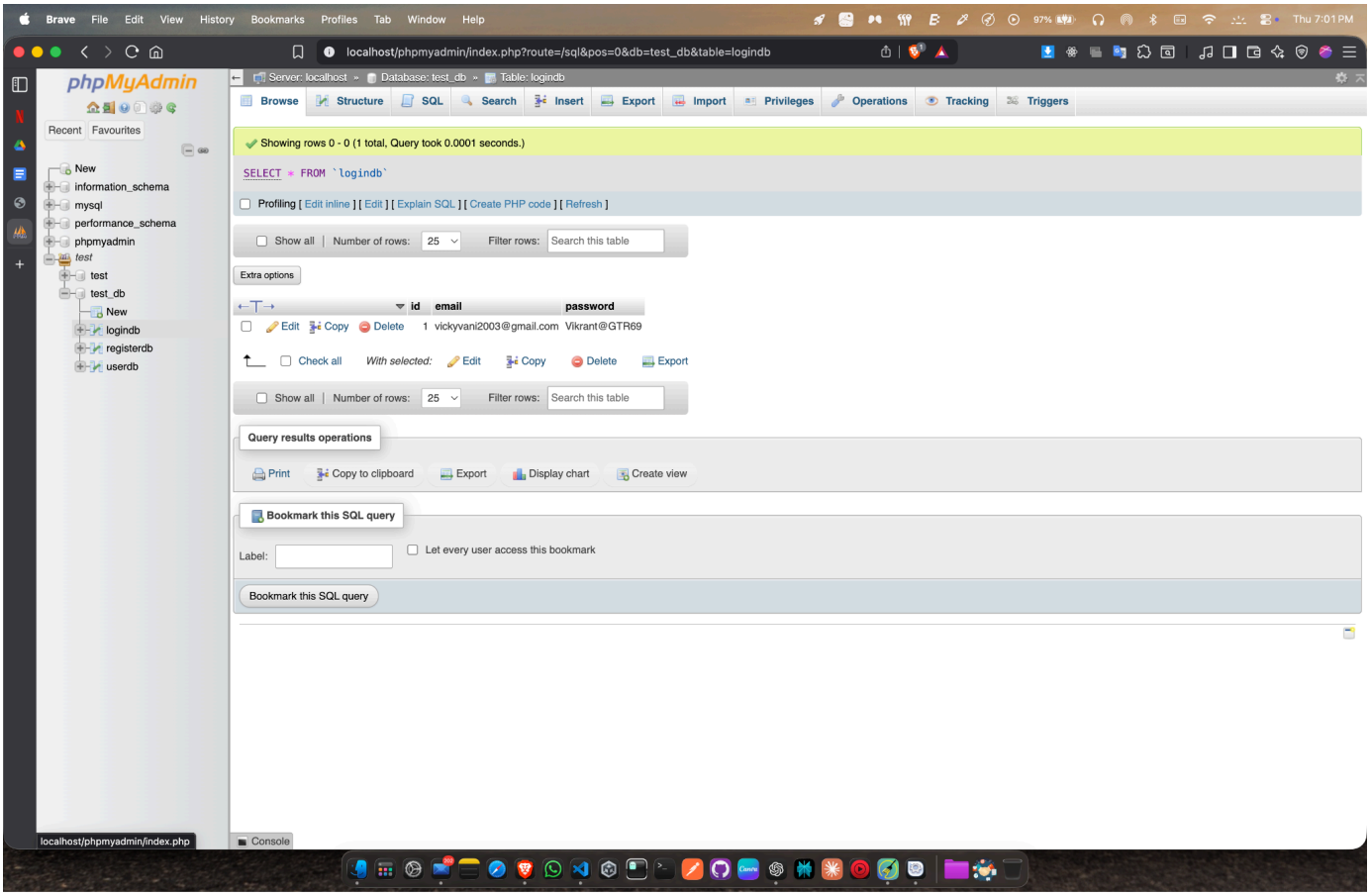
```
    const user = results[0];
    if (user.password === password) {
      console.log("Login successful:", user.email);
      req.session.user = user.email;
      return res.redirect("/");
    } else {
      console.log("Invalid password for:", user.email);
      return res.render("pages/login", {
        error: "Invalid password. Please try again.",});   }  });});
app.post("/register", (req, res) => {
  const { fullname, email, password, confirmpassword, phone, state } = req.body;
  if (
    !fullname ||
    !email ||
    !password ||
    !confirmpassword ||
    !phone ||
    !state
  ) {return res.render("pages/register", { error: "All fields are required" }); }
  if (password !== confirmpassword) {
    return res.render("pages/register", { error: "Passwords do not match" });}
  const sql = `INSERT INTO registerdb (fullname, email, password, confirmpassword, phone, state)
               VALUES (?, ?, ?, ?, ?, ?)`;
  const values = [fullname, email, password, confirmpassword, phone, state];
  connection.query(sql, values, (err, results) => {
    if (err) {
      console.error("Database Insert Error:", err);
      return res.render("pages/register", {
```

```
    error: "Database error, please try again.", }); }
console.log("Registration successful!", results);
req.session.user = email;
res.redirect("/"); });});});
```

Output:




phpMyAdmin
Server: localhost » Database: test_db » Table: registerdb
Browse Structure SQL Search Insert Export Import Privileges Operations Tracking Triggers
Current selection does not contain a unique column. Grid edit, checkbox, Edit, Copy and Delete features are not available.
Showing rows 0 - 1 (2 total, Query took 0.0001 seconds.)
SELECT * FROM `registerdb`
Profiling [Edit inline] [Edit] [Explain SQL] [Create PHP code] [Refresh]
Show all | Number of rows: 25 Filter rows: Search this table
Extra options
fullname email password confirmpassword phone state
Vikrant Vani vickyvani2003@gmail.com Vikrant@GTR69 Vikrant@GTR69 7249662437 Maharashtra
Vedant Parulekar vedant@gmail.com vedant123 vedant123 1234567890 Goa
Query results operations
Print Copy to clipboard Export Display chart Create view
Bookmark this SQL query
Label:
Let every user access this bookmark
Bookmark this SQL query
Console


phpMyAdmin
Server: localhost » Database: test_db » Table: logindb
Browse Structure SQL Search Insert Export Import Privileges Operations Tracking Triggers
Showing rows 0 - 0 (1 total, Query took 0.0001 seconds.)
SELECT * FROM `logindb`
Profiling [Edit inline] [Edit] [Explain SQL] [Create PHP code] [Refresh]
Show all | Number of rows: 25 Filter rows: Search this table
Extra options
id email password
Edit Copy Delete 1 vickyvani2003@gmail.com Vikrant@GTR69
Check all With selected: Edit Copy Delete Export
Query results operations
Print Copy to clipboard Export Display chart Create view
Bookmark this SQL query
Label:
Let every user access this bookmark
Bookmark this SQL query
Console

Conclusion:

Connecting Node.js to MySQL and implementing CRUD makes the app data-driven and persistent. Safe query practices and organized route handlers enable robust data manipulation that can scale to more advanced features.

Experiment No.10

Problem Statement:

Server-rendered views are great for user-facing pages, but separating data access via JSON APIs is essential for interoperability (mobile apps, SPA frontends). The task is to build RESTful endpoints that expose resources and support Create/Read/Update/Delete operations.

Objective:

- Design RESTful endpoints for a resource (e.g., /api/products):
- GET /api/products → list items
- GET /api/products/:id → get single item
- POST /api/products → create item
- PUT /api/products/:id → update item
- DELETE /api/products/:id → delete item
- Return JSON responses with proper status codes and error messages.
- Test endpoints using Postman or similar tools.

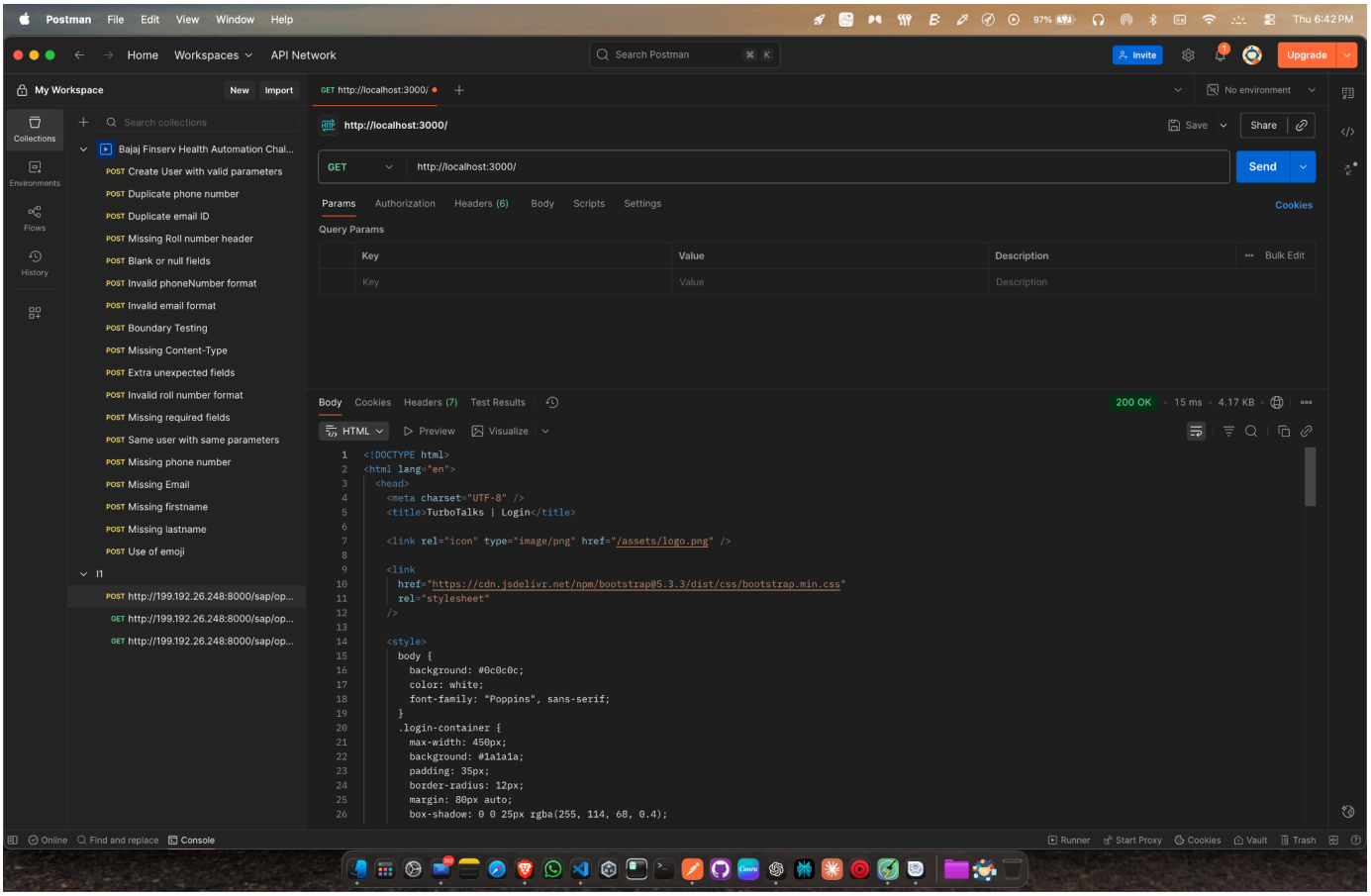
Theory:

REST principles map resources to URLs and operations to HTTP verbs. APIs should be stateless and return structured JSON. Proper status codes (200, 201, 400, 404, 500) indicate success/errors. APIs decouple the client and server, enabling multiple clients (web, mobile) to use the same backend.

Code:

```
app.get("/gallery", isAuthenticated, (req, res) => res.render("pages/gallery"));

app.get("/products", isAuthenticated, (req, res) =>

  res.render("pages/products"));

app.get("/cart", isAuthenticated, (req, res) => res.render("pages/cart"));

app.get("/about", isAuthenticated, (req, res) => res.render("pages/about"));

app.get("/contact", isAuthenticated, (req, res) => res.render("pages/contact"));
```

Output:



Conclusion:

A RESTful API layer provides programmatic access to application data and forms the backbone for integrations. Building and testing these endpoints ensures the application can serve multiple frontends and external consumers reliably.

Prepared By – TTT Coordination Committee		
Prof. Rohini Bhosale	**Prof. Rohini Bhosale**	**Prof. Dr. Prashant Dhotre**
Subject Teacher	**Course Coordinator**	**Head - Department of CSE**