# SIMATS SCHOOL OF ENGINEERING

**SAVEETHA INSTITUTE OF MEDICAL AND TECHNICAL**

**SCIENCES CHENNAI-602105**

**Bike Showroom**

**A CAPSTONE PROJECT REPORT**

*Submitted in the partial fulfillment for the award of the degree of*

**BACHELOR OF ENGINEERING**

**IN**

## COMPUTER SCIENCE AND ENGINEERING

**Submitted by**

**K. Sukesh Reddy(192211897)**

## K.Sandeep (192211872)

**Under the Supervision of Ms.B.Jeevashri**

**JULY 2024**

## DECLARATION

We, **K Sukesh Reddy**, **K** **Sandeep** students of **Bachelor of Engineering in CSE**, Department of Computer Science and Engineering, Saveetha Institute of Medical and Technical Sciences, Saveetha University, Chennai, hereby declare that the work presented in this Capstone Project Work entitled **Bike showroom** is the outcome of our own bonafide work and is correct to the best of our knowledge and this work has been undertaken taking care of Engineering Ethics.

1. **K. Sukesh reddy(192211897)**
2. **K.Sandeep (192211872)**

Date: 31/07/24

Place: Chennai

## CERTIFICATE

This is to certify that the project entitled **“BIKE SHOWROOM”** submitted by **K. Sukesh Reddy, K. Sandeep** has been carried out under my supervision. The project has been submitted as per the requirements in the current semester of Computer Science Engineering.

Teacher-in-charge

Ms.B.Jeevashri

### Table of Contents

|  |  |
| --- | --- |
| **S.NO** | **TOPICS** |
|  | **Abstract** |
| 1 | **Introduction** |
| 2 | **Project Description** |
| 3 | **Problem Description** |
| 4 | **Tool Description** |
| 5 | **Operations** |
| 6 | **Approach / Module Description / Functionalities**  6.1 UserAuthentication Module  6.2 Quiz Management  6.3 Quiz Taking  6.4 Performance Analysis  6.5 User Profile  6.6 Admin |
| 7 | **Implementation** |
| 8 | **Result** |
| 9 | **Conclusion** |
|  | **References** |

**Abstract**

Creating a bike showroom website involves both front-end and back-end development to provide a seamless user experience. The front-end uses HTML, CSS, and JavaScript, with frameworks like React, Angular, or Vue.js for a responsive interface. Key components include a homepage featuring the latest bikes and promotions, bike listings with filters for categories, brands, and prices, detailed bike pages with specifications and reviews, a search functionality, and a responsive design for accessibility on various devices. The back-end is built with server-side languages like Node.js, Python, or PHP and databases such as MySQL or MongoDB. It involves designing a database schema for bikes, categories, brands, user accounts, reviews, and orders, handling user authentication, managing bike inventory through CRUD operations, processing orders with shopping cart and payment systems, and creating RESTful API endpoints to connect the front-end and back-end. Deployment uses platforms like AWS or Heroku with CI/CD pipelines for updates and maintenance.

**Introduction**

Creating a bike showroom website involves integrating both front-end and back-end technologies to deliver a seamless, user-friendly experience for customers. The front-end utilizes HTML, CSS, and JavaScript along with frameworks like React, Angular, or Vue.js to create a responsive and interactive interface. Key components include a homepage showcasing the latest bikes and promotions, detailed bike listings with filters, and individual bike pages featuring specifications and reviews. The back-end, developed using technologies like Node.js, Python, or PHP, and databases such as MySQL or MongoDB, handles user authentication, bike inventory management, and order processing. The site ensures a responsive design for accessibility on various devices and employs RESTful APIs to connect the front-end and back-end. Deployment is managed using platforms like AWS or Heroku with continuous integration and delivery pipelines for ongoing updates and maintenance.

A bike, also known as a bicycle, is a two-wheeled vehicle that is powered by human effort through pedaling. It is one of the most efficient and eco-friendly modes of transportation, widely used for commuting, recreation, and sport. Bikes come in various types, including road bikes, mountain bikes, hybrid bikes, and electric bikes, each designed for specific terrains and purposes. Over the years, advancements in materials and technology have led to lighter, stronger frames and more efficient components, making bikes increasingly popular for both everyday use and competitive cycling. The simplicity, affordability, and versatility of bikes continue to make them a preferred choice for millions around the world.

**Project Description**

The bike showroom website project aims to create an engaging online platform for browsing and purchasing bikes. The front-end, built with HTML, CSS, JavaScript, and frameworks like React, will feature a responsive design, detailed bike listings, and search functionality. The back-end, developed with Node.js, Python, or PHP and databases like MySQL, will manage user authentication, bike inventory, and order processing. RESTful APIs will connect the front-end and back-end. Deployment on AWS or Heroku will ensure continuous integration and maintenance.

**Proposed Method**

* **HTML Structure:** Create the basic structure of the web pages using HTML.
* **Styling with CSS:** Apply CSS for styling and ensure the design is responsive using media queries and flexible grid layouts.
* **JavaScript Interactivity:** Add interactivity using JavaScript, including features like image zoom, lightbox view, and dynamic content loading.

**2.1 About my project**

**Purpose and Scope**

Purpose: Create a bike showroom website to enhance online bike browsing and purchasing. Scope: Develop a responsive front-end with React, a robust back-end with Node.js or Python, and integrate RESTful APIs. Deployment: Host on AWS or Heroku with CI/CD for updates and Maintenance .Features and Functionality

User Management

1. **User Authentication:**
   * **Sign Up/Registration:** Allows users to create an account with email and password.
   * **Login/Logout:** Secure user login and logout functionality.
   * **Password Recovery:** Option for users to reset forgotten passwords via email.
2. **User Roles and Permissions:**
   * **Role-Based Access Control:** Differentiates permissions for regular users and administrators.

**Image Management**

1. **Image Upload:**
   * **Single and Bulk Upload:** Supports uploading one or multiple images at once.
   * **Drag and Drop:** Provides a drag-and-drop interface for easier uploads.
2. **Image Editing:**
   * **Metadata Management:** Users can edit image titles, descriptions, and tags.
   * **Basic Editing Tools:** Crop, rotate, and adjust image settings.
3. **Image Deletion:**
   * **Remove Images:** Allows users to delete images from their gallery.

**Problem Description**

The project addresses the need for an effective online platform to browse and purchase bikes, replacing outdated or inefficient systems. Key issues include:

1. **User Experience:** Existing platforms may lack an intuitive, responsive design, making it difficult for users to find and purchase bikes.
2. **Inventory Management:** Many systems struggle with real-time updates and management of bike inventory, leading to inaccurate listings and stock issues.
3. **Order Processing:** Inefficient order handling can result in errors and delays, affecting customer satisfaction.

**Tool Description**

**Hardware and Software Tools**

To develop and deploy the recipe management web application, the following hardware and software tools were utilized:

**Hardware Specifications**

* **Laptop Model**: ASUS ROG Strix
* **Graphics Card**: NVIDIA GeForce RTX 3060, 4GB
* **Storage**: 1TB SSD
* **RAM**: 16GB
* **Processor**: AMD Ryzen 7 6800H

The ASUS ROG Strix laptop with its high-performance specifications provided an excellent environment for developing and testing the web application. The NVIDIA GeForce RTX 3060 graphics card ensured smooth rendering of graphics and multimedia content, enhancing the development experience, especially when dealing with high-resolution recipe images and user interface design. The 1TB SSD facilitated fast data read/write operations, significantly reducing load times for development tools and ensuring rapid access to project files. With 16GB of RAM, the laptop efficiently handled multiple development tools running concurrently, supporting a seamless multitasking environment. The AMD Ryzen 7 6800H processor, known for its powerful performance and energy efficiency, enabled quick compilation and execution of code, speeding up the development cycle.

**Software Tools**

* **Visual Studio Code**: An integrated development environment (IDE) used for writing and debugging code. Its extensions and integrated terminal enhanced the coding experience.
* **XAMPP**: A free and open-source cross-platform web server solution stack package developed by Apache Friends. It provided the necessary Apache, MySQL, PHP, and Perl support for local development and testing.
* **phpMyAdmin**: A free software tool written in PHP, intended to handle the administration of MySQL over the web. phpMyAdmin was used for database management, allowing for easy handling of the MySQL database used in the application.
* **GitHub**: Used for version control and collaborative development. The repository hosted the project's source code, enabling team collaboration and version tracking.
* **Google Chrome**: The primary web browser used for testing and debugging the web application. Developer tools in Chrome facilitated real-time inspection and modification of the front-end code.

The combination of powerful hardware and a robust set of development tools provided a conducive environment for the efficient development, testing, and deployment of the recipe management web application.

**Operations**

The operations for the Photo Gallery Project encompass the processes and tasks required to manage, maintain, and enhance the application. These operations ensure the system runs smoothly, efficiently, and securely. Here’s a detailed breakdown of the key operations involved.

**User Operations**

**User Registration and Authentication:**

* **Account Creation:** Users sign up by providing necessary details such as email and password.
* **Login/Logout:** Users log in to access their galleries and log out to end their session.
* **Password Management:** Users can reset or change their passwords via email verification.

**Profile Management:**

* **Update Profile:** Users can update their profile information, such as username and contact details.
* **Privacy Settings:** Users can manage privacy settings to control who can view their images and gallery.

**Creating and Managing Recipes**

**Image Operations**

1. **Image Upload and Management:** 
   * **Upload Images:** Users can upload images individually or in bulk**.** o **Edit Image Metadata:** Users can modify image titles, descriptions, and tags**.**
   * **Delete Images:** Users can remove unwanted images from their gallery.
2. **Image Processing:** 
   * **Resizing and Compression:** Images are automatically resized and compressed for optimal performance.
   * **Image Editing:** Basic editing tools may be provided for cropping, rotating, and adjusting images.

**Managing Categories**

**Category Creation Interface:**

* **Form Input**: Provide a user-friendly form for users to create new categories. The form typically includes fields for the category name, description, and optional image or color to visually represent the category.
* **Validation**: Implement validation rules to ensure that category names are unique and adhere to any naming conventions or restrictions.

**User Management**

1. **Assigning Categories to Images** 
   1. **Category Assignment Interface:**
      * **Image Edit Screen:** Provide an interface on the image edit or upload screen where users can select or assign categories to their images.
      * **Multi-Select Option:** Allow users to assign multiple categories to a single image if applicable.
   2. **Batch Assignment:**
      * **Bulk Operations:** Enable users to assign categories to multiple images simultaneously, which is useful for efficiently managing large numbers of images.

**Analyzing Recipe Usage**

1. **Organizing Categories** 
   1. **Category Hierarchy:** 
      * **Subcategories:** Support hierarchical categorization by allowing users to create subcategories within parent categories. This helps in organizing images in a more granular manner.
      * **Drag-and-Drop:** Implement drag-and-drop functionality for rearranging categories and subcategories in the admin panel.
   2. **Sorting and Filtering:** 
      * **Sort Categories**: Allow users to sort categories by name, creation date, or other criteria.
      * **Filter Categories:** Provide filtering options to help users quickly find and view specific categories.

**Interacting with Photo Gallery.**

1. **Browsing and Viewing Images**

**1. Homepage and Gallery Navigation:**

* + **Homepage:** Users are greeted with a visually appealing homepage that highlights featured images or categories**.**
  + **Gallery Navigation:** Users can navigate through different sections of the gallery using a sidebar, menu, or breadcrumb navigation**.**

1. **Image Thumbnails:** 
   * **Grid View:** Images are displayed as thumbnails in a grid view, providing a quick overview of the gallery.
   * **List View**: Option to switch to a list view for detailed information about each image alongside the thumbnail**.**
2. **Image Details:** 
   * **Lightbox View:** Clicking on an image opens it in a lightbox view, providing a larger, detailed view of the image without leaving the current page.
   * **Slideshow:** Users can navigate through images in a slideshow format within the lightbox view.

**User Authentication**

* **Register:** New users can create an account by providing their username, email, and password.
* **Retrieve User:** Fetch the user’s details from the database using the email/username.
* **Login:** Registered users can log into their accounts using their credentials.
* **Logout:** Users can log out of their accounts to secure their sessions.

**Password Management**

1. **Password Reset:** 
   * **Request Form:** Users can request a password reset by providing their email address**.** o **Token Generation:** Generate a secure token and send it to the user’s email with a link to reset their password.
2. **Password Change:** 
   * **Change Form:** Authenticated users can change their password by providing the current password and the new password.o **Validation**: Verify the current password and ensure the new password meets complexity requirements.
3. **Security Measures** 
   1. **Rate Limiting:**
      * **Throttling:** Implement rate limiting to prevent brute force attacks on the login and registration forms.
   2. **Account Lockout:**
      * **Lockout Mechanism:** Temporarily lock the account after a certain number of failed login attempts.
   3. **Secure Cookies:**
      * **HTTP-Only and Secure Flags:** Use secure cookies with the Http Only and Secure flags to prevent client-side script access and ensure transmission over HTTPS.
   4. **Two-Factor Authentication (2FA):**
      * **Optional 2FA:** Implement optional two-factor authentication using methods like SMS, email, or an authenticator app.
4. **User Roles and Permissions** 
   1. **Role-Based Access Control (RBAC):**
      * **User Roles:** Define roles such as user, admin, and moderator with specific permissions for each role. o **Access Control:** Restrict access to certain features or sections of the application based on the user’s role.
   2. **Permissions Management:**
      * **Assign Permissions:** Assign and manage permissions for different roles through an admin interface.

1. **Approach/Module**

**Purpose:**

The primary purpose of this project is to offer an organized and efficient platform for managing digital photo collections. It aims to facilitate easy access, sharing, and organization of images for users with varying levels of technical expertise.

Scope:

The project covers functionalities such as user registration and login, image upload and management, categorization, search and filter options, user engagement features, and administrative tools. It is designed to handle high traffic and large volumes of data while ensuring a smooth user experience.

**Operations**

**User Operations:**

* **Registration and Authentication:** Manage user accounts and session security**.**
* **Profile Management:** Update user information and privacy settings.

**Image Operations:**

* **Image Upload and Management:** Handle image uploads, editing, and organization.
* **Image Processing:** Automate image resizing and compression**.**

**Gallery Operations:**

* **Display and Layout Management:** Customize gallery layout and interactive features.
* **Search and Filtering:** Implement search and filter options for images**.**

**Administrative Operations:**

* **User Management:** Handle user roles and monitor activity.
* **Content Moderation:** Review and manage uploaded content and comments.
* **System Maintenance:** Perform regular updates and backups**.**

**Security Operations:**

* **Data Protection:** Implement encryption and access controls**.**
* **Vulnerability Management:** Regularly scan for and address security vulnerabilities.

**Performance and Monitoring:**

* **Performance Monitoring:** Track key metrics and optimize performance.
* **Logging and Analysis:** Maintain and analyse logs for system activity and issues**.**

**Deployment and Continuous Integration:**

* **Automated Builds:** Set up CI/CD pipelines for building and deploying code**.**
* **Automated Testing:** Run automated tests as part of the CI/CD process**.**

**6.1 Tools Used**

**Frontend Development:**

* **HTML, CSS, JavaScript:** For building and styling the user interface.
* **Bootstrap, jQuery:** For responsive design and simplifying JavaScript tasks.

**Backend Development:**

* **Node.js, Express.js:** For server-side logic and handling API requests.
* **MongoDB:** For storing user data and image metadata.
* **Sharp:** For image processing.

**Development and Build Tools:**

* **Git:** For version control.
* **Visual Studio Code:** For code editing.
* **Webpack:** For bundling JavaScript modules.

**Testing Tools:**

* **Jest, Mocha:** For unit and integration testing.
* **Cypress:** For end-to-end testing.

**Deployment Tools:**

* **Heroku, AWS:** For deploying and hosting the application.
* **GitHub Actions, Jenkins:** For CI/CD pipelines**.**

**Security Tools:**

* **B crypt:** For hashing passwords.
* **OWASP ZAP:** For security scanning.

**6.2 User Management Functions**

1. **Register User** o **Function: register User** 
   * + **Description:** Registers a new user by saving their details in the database after validating the input and hashing the password.
     + **Input**: username, email, passwordo **Output:** Success message or error
2. **Login User** o **Function:** login User
   * + **Description:** Authenticates a user by checking their email and password, and generates a session token upon successful login**.**
     + **Input:** email, password**.**
     + **Output:** Session token or error.
3. **Logout User** o **Function**: logout Usero **Description:** Logs out the user by invalidating their session token.

o **Input:** session Tokeno **Output:** Success message or error

1. **Update Profile** o **Function:** Update Profile
   * + **Description:** Updates user profile information such as username, email, and profile picture.
     + **Input:** user Id, username, email, Profile Picture.
     + **Output:** Success message or error.
2. **Reset Password** o **Function:** reset Password

o **Description**: Allows users to reset their password by verifying their email and sending a password reset link. o **Input:** emailo **Output**: Success message or erroro **Output:** Success message or error

**6.3 Frontend Components**

The frontend of the Photo Gallery Project is built using plain HTML, CSS, and JavaScript.

**HTML**

* **Index Page:** The main page displaying the gallery of photos**.**
* **Upload Page:** A page where users can upload new photos**.**
* **Login and Registration Pages:** Pages for user authentication**.**

**CSS**

* **Styling:** Provides the layout, colours, fonts, and overall visual appeal of the web pages.
* **Responsive Design:** Ensures that the gallery is accessible and looks good on different devices.

**6.4 Admin Module (Administrator)**

**Function: Manage Users**

* **Description:** Enables administrators to manage user accounts.
* **Functionalities:**
  + - View user list.
    - Edit or delete user accounts.

**Function: Manage Categories**

* **Description:** Allows administrators to manage photos.
* **Functionalities:**

o Add, edit, or delete categories.

7. **Implementation/Coding**

**Login Code:-**

<?php

session\_start(); // Start session management

// Database connection parameters

$servername = "localhost";

$username = "root";

$password = "";

$dbname = "bike\_showroom";

// Create connection

$conn = new mysqli($servername, $username, $password, $dbname);

// Check connection

if ($conn->connect\_error) {

    die("Connection failed: " . $conn->connect\_error);

}

// Get form data

$user = $\_POST['username'];

$pass = $\_POST['password'];

// Prepare and execute SQL query

$sql = "SELECT password FROM users WHERE username = ?";

$stmt = $conn->prepare($sql);

$stmt->bind\_param("s", $user);

$stmt->execute();

$stmt->bind\_result($passwordFromDB);

$stmt->fetch();

if ($pass === $passwordFromDB) {

    // Successful login

    $\_SESSION['username'] = $user; // Store username in session

    header("Location: dashboard.php"); // Redirect to dashboard

    exit();

} else {

    // Failed login

    echo "Invalid username or password.";

}

$stmt->close();

$conn->close();

?>

**Registration Code**:-

<!DOCTYPE html>

<html lang="en">

<head>

<meta charset="UTF-8">

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

<title>Register</title>

<style> body {

font-family: Arial, sans-serif; display: flex; justify-content: center; align-items: center; height: 100vh; margin: 0; background-color: #f0f0f0;

}

. register-container { background-color: white; padding: 20px; border-radius: 8px; box-shadow: 0 0 10px rgba(0, 0, 0, 0.1); width: 300px;

}

.register-container h1 { margin-bottom: 20px;

}

.register-container input { width: calc(100% - 20px); padding: 10px; margin: 10px 0; border: 1px solid #ccc; border-radius: 4px;

}

.register-container button { width: calc(100% - 20px); padding: 10px; background-color: #4CAF50; color: white; border: none; border-radius: 4px; cursor: pointer; margin: 10px 0;

}

.register-container button:hover { background-color: #45a049;

}

.register-container .error { color: red; display: none;

}

</style>

</head>

<body>

<div class="register-container">

<h1>Register</h1>

<form id="register Form">

<input type="text" id="username" placeholder="Username" required>

<input type="email" id="email" placeholder="Email" required>

<input type="password" id="password" placeholder="Password" required>

<button type="submit">Register</button>

<p class="error" id="error">Please fill out all fields correctly</p>

</form>

</div>

<script>

document.getElementById('registerForm').addEventListener('submit', function(event) { event.preventDefault();

const username = document.getElementById('username').value; const email = document.getElementById('email').value; const password = document.getElementById('password').value;

// Add your validation logic here if (username && email && password) { alert('Registration successful!');

// Perform registration logic here, such as sending data to the backend

// Redirect to the search form window.location.href = 'login.html';

} else {

document.getElementById('error').style.display = 'block';

}

});

</script>

</body>

</html>

**UPLOAD CODE**: - <!DOCTYPE html>

<html lang="en">

<head>

<meta charset="UTF-8">

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

<title>Upload Photo</title>

<style> body {

font-family: Arial, sans-serif; display: flex; justify-content: center; align-items: center; height: 100vh; margin: 0;

background-color: #f0f0f0;

}

.upload-container { background-color: white; padding: 20px; border-radius: 8px;

box-shadow: 0 0 10px rgba(0, 0, 0, 0.1); width: 300px; text-align: center;

}

.upload-container h1 { margin-bottom: 20px;

}

.upload-container input, .upload-container text area { width: calc(100% - 20px); padding: 10px; margin: 10px 0;

border: 1px solid #ccc; border-radius: 4px;

}

.upload-container button { width: calc(100% - 20px); padding: 10px;

background-color: #4CAF50; color: white; border: none; border-radius: 4px; cursor: pointer; margin: 10px 0;

}

.upload-container button:hover { background-color: #45a049;

}

.upload-container .error { color: red; display: none;

}

</style>

</head>

<body>

<div class="upload-container">

<h1>Upload Photo</h1>

<form id="uploadForm" action="update.html" method="post" ectype="multipart/formdata">

<input type="text" id="title" name="title" placeholder="Title" required>

<textarea id="description" name="description" placeholder="Description" required></textarea>

<input type="file" id="photo" name="photo" accept="image/\*" required>

<button type="submit">Upload</button>

<p class="error" id="error">Please fill out all fields and select a photo.</p>

</form>

</div>

<script>

document.getElementById('uploadForm').addEventListener('submit', function(event) { const title = document.getElementById('title').value;

const description = document.getElementById('description').value; const photo = document.getElementById('photo').files[0]; if (!title || !description || !photo) {

event.preventDefault(); // Prevent form submission document.getElementById('error').style.display = 'block';

} else {

document.getElementById('error').style.display = 'none';

}

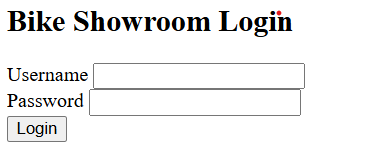
});

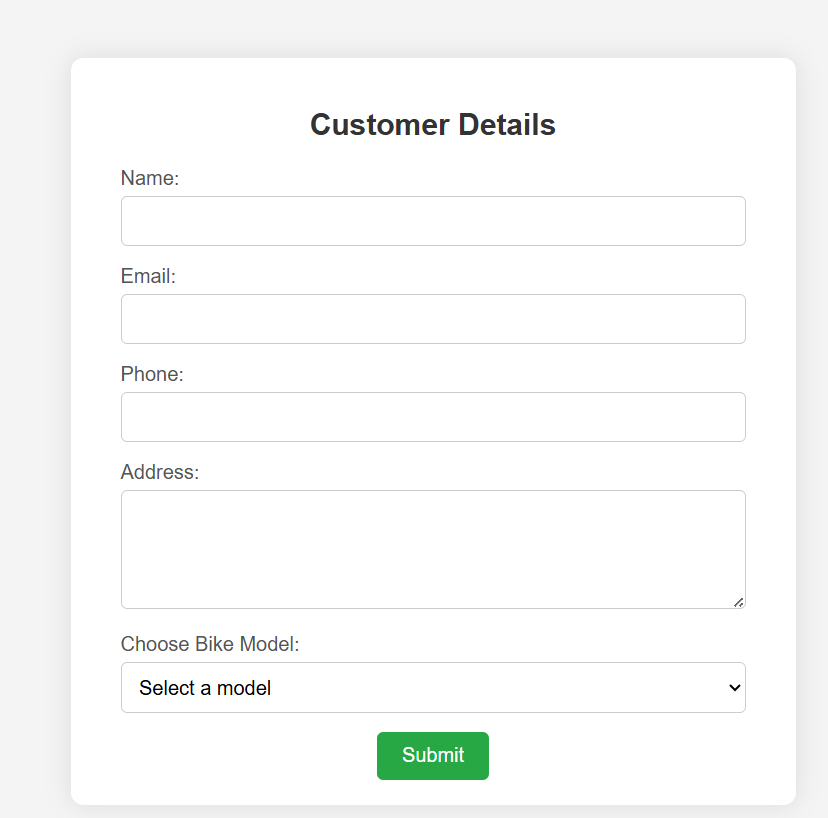
</script>

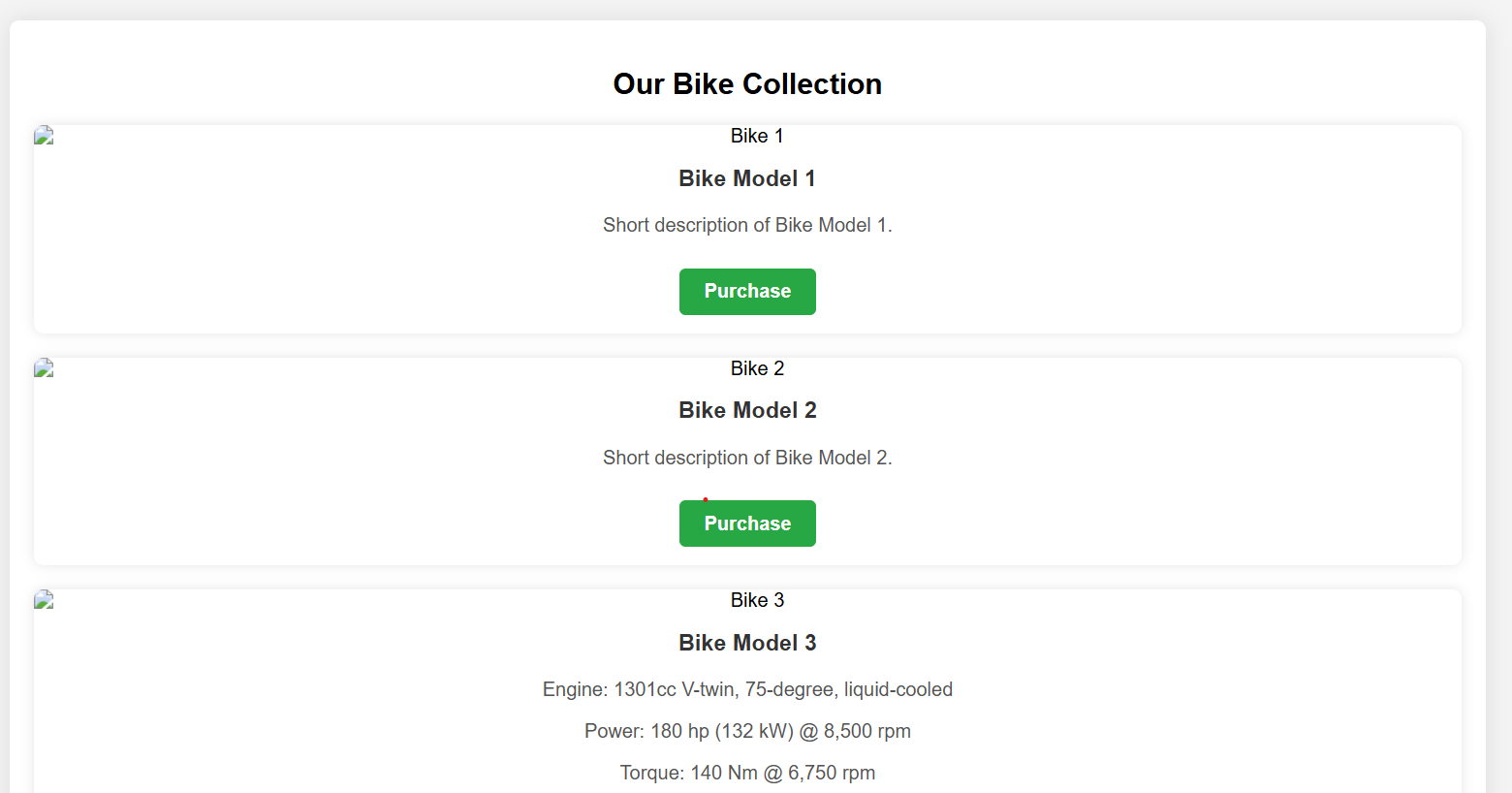
</body>

</html>

**Result**







****

1. **Conclusion**

The Bike Showroom Project is a robust and comprehensive web-based application designed to address the needs of users looking to efficiently manage and share their digital photo collections. By leveraging modern web technologies and best practices, the project provides a seamless and user-friendly experience for both individual users and administrators.

**Key Takeaways:**

1. **User-Friendly Design:** The project prioritizes a user-friendly interface, making it easy for users of all technical levels to upload, organize, and share their photos.
2. **Comprehensive Features:** The application includes a wide range of features, from user registration and image management to advanced search and category organization, ensuring a versatile and functional tool for managing digital photos.
3. **Security and Performance:** With robust user authentication, data protection measures, and performance monitoring, the project ensures that user data is secure and the application runs efficiently.
4. **Scalability:** Designed to handle high traffic and large volumes of data, the application can grow and adapt to increasing user demands.
5. **Administrative Tools:** The inclusion of powerful administrative tools allows for effective user management, content moderation, and system maintenance, ensuring the smooth operation of the application.

The Bike Showroom Project demonstrates a successful implementation of a complex web application, integrating various technologies and methodologies to create a cohesive and efficient platform. By addressing key aspects such as usability, functionality, security, and scalability, the project serves as a valuable resource for managing digital photo collections, catering to the needs of both individual users and administrators.

As the project evolves, future enhancements could include the integration of machine learning algorithms for automated image tagging and categorization, as well as the implementation of more advanced analytics and reporting features to provide deeper insights into user engagement and system performance. Overall, the Photo Gallery Project represents a significant achievement in web application development, offering a practical and powerful tool for digital photo management.

**References**

1 **HTML/CSS Documentation:** Mozilla Developer Network (MDN). "HTML Reference." Mozilla Foundation. [MDN Web Docs](https://developer.mozilla.org/)

2 **JavaScript Frameworks:** Facebook Inc. "React Documentation." React. React Documentation

3 **JavaScript Frameworks:** Angular Team. "Angular Documentation." Angular. Angular Documentation

4 **JavaScript Frameworks:** Evan You. "Vue.js Documentation." Vue.js. Vue.js Documentation

5 **Server-Side Languages:** OpenJS Foundation. "Node.js Documentation." Node.js. Node.js Documentation

6 **Server-Side Languages:** Django Software Foundation. "Django Documentation." Django. [Django Documentation](https://docs.djangoproject.com/)

7 **Database Management:** Oracle Corporation. "MySQL Documentation." MySQL. [MySQL Documentation](https://dev.mysql.com/doc/)

8 **Database Management:** MongoDB, Inc. "MongoDB Documentation." MongoDB. [MongoDB Documentation](https://docs.mongodb.com/)

9 **API Development:** Mike Amundsen. "REST API Tutorial." [REST API Tutorial](https://restfulapi.net/)

10 **Deployment:** Amazon Web Services, Inc. "AWS Documentation." Amazon Web Services. [AWS Documentation](https://docs.aws.amazon.com/) / Heroku, Inc. "Heroku Documentation." Heroku. Heroku Documentation