**1. Introduction to the Project (1-2 minutes)**

* **Project Overview:**
  + **Our project, TheTechHeads, is an e-commerce platform designed to sell tech-oriented footwear, with a primary focus on Nike shoes. The platform offers users the ability to browse a variety of shoe products, add them to their shopping cart, and securely process payments. The site includes functionality for user registration and authentication, product filtering by categories (men, women, kids), and a streamlined checkout process.**
* **Why is This Project Important?**
  + **Online shopping has become a central aspect of consumer culture, especially with the increasing shift towards digital retail platforms. We chose this project to build a full-stack solution that addresses the essential components of an e-commerce system, including user authentication, product management, shopping cart functionality, and payment processing. This project allowed us to explore critical areas of web development such as front-end design, back-end development, security, and database management. Additionally, it’s an opportunity to simulate the challenges and rewards faced in real-world projects, especially as e-commerce continues to grow globally.**

**2. Project Requirements (2-3 minutes)**

* **Functional Requirements:**
  + **Our system needed to meet several key functional requirements:**
    - **User Registration and Login: Users must be able to create an account and log in securely. After logging in, they should be able to update their profile information.**
    - **Product Display and Browsing: Users should be able to view a catalog of products that are filtered into categories like Men, Women, and Kids.**
    - **Shopping Cart: Users should be able to add products to a cart, modify the quantity of items in the cart, and remove items as needed.**
    - **Checkout and Payment: The system must enable users to complete their purchase by providing valid payment information. The checkout process should calculate the total cost of the items, including shipping and taxes.**
* **Non-Functional Requirements:**
  + **Performance: The platform should load quickly, handle multiple users at the same time, and update shopping cart items in real-time.**
  + **Security: User data must be handled securely, particularly around sensitive areas like password storage and payment processing.**
  + **User-Friendly Design: The interface should be intuitive, responsive across devices, and easy to navigate, ensuring a seamless user experience.**
* **3. Analysis and Design (4-5 minutes)**
* **System Architecture:**
* **Our project uses the Model-View-Controller (MVC) architecture. This separates our code into distinct components: the Model, which handles the data; the View, which handles the user interface; and the Controller, which coordinates between the two.**
* **Model: The model represents the core data structure of the project. It includes entities like User, Product, Order, and Cart.**
* **View: The view is responsible for rendering the UI that the user interacts with. It consists of HTML files styled with CSS, and uses JavaScript for interactive elements.**
* **Controller: The controller handles the logic that connects the user input from the front-end with the data from the back-end. It processes requests, queries the database, and sends data back to the user.**
* **UML Diagram of Classes: The primary classes are:**
* **User: This class represents registered users in the system. It includes fields like userId, name, email, password, and role. The role distinguishes between customers and admins.**
* **Product: Contains product details such as productId, name, description, price, and category.**
* **Order: Stores order information, linking users to the products they purchase. Fields include orderId, userId, productId, and quantity.**
* **Cart: Represents the shopping cart, containing products that users intend to purchase. It stores product IDs and quantities temporarily.**
* **Payment: This class handles payment information, including validation of card details, expiration dates, and CVV.**
* **Data Structures:**
  + **ArrayLists**: Used to store collections of products in the cart.
  + **HashMaps**: Used to map product IDs to quantities in the cart.
* **Algorithms:**
  + For the shopping cart, we implemented logic to dynamically calculate the total price based on the quantity and price of items in the cart. This also updates in real-time as users modify their cart contents.

**4. Database Design (2-3 minutes)**

* **Database Schema: Our database design ensures efficient storage and retrieval of data. We used MySQL as our database to store user, product, cart, and order information.**
  + **Users Table: Stores user information such as userId, email, password, and role.**
  + **Products Table: Stores product details, including productId, name, price, and category.**
  + **Orders Table: Stores orders made by users, linking the user to the product they bought with fields like orderId, userId, productId, and quantity.**
  + **Cart Table: Stores temporary cart data for users as they shop, tracking product IDs and quantities.**

**Sure! Here's an extensive description of your project that should help you present for about 15 minutes. Feel free to adjust the pace and expand on specific sections during the presentation to ensure you cover everything comprehensively.**

**1. Introduction to the Project (1-2 minutes)**

* **Project Overview:**
  + **Our project, TheTechHeads, is an e-commerce platform designed to sell tech-oriented footwear, with a primary focus on Nike shoes. The platform offers users the ability to browse a variety of shoe products, add them to their shopping cart, and securely process payments. The site includes functionality for user registration and authentication, product filtering by categories (men, women, kids), and a streamlined checkout process.**
* **Why is This Project Important?**
  + **Online shopping has become a central aspect of consumer culture, especially with the increasing shift towards digital retail platforms. We chose this project to build a full-stack solution that addresses the essential components of an e-commerce system, including user authentication, product management, shopping cart functionality, and payment processing. This project allowed us to explore critical areas of web development such as front-end design, back-end development, security, and database management. Additionally, it’s an opportunity to simulate the challenges and rewards faced in real-world projects, especially as e-commerce continues to grow globally.**

**2. Project Requirements (2-3 minutes)**

* **Functional Requirements:**
  + **Our system needed to meet several key functional requirements:**
    - **User Registration and Login: Users must be able to create an account and log in securely. After logging in, they should be able to update their profile information.**
    - **Product Display and Browsing: Users should be able to view a catalog of products that are filtered into categories like Men, Women, and Kids.**
    - **Shopping Cart: Users should be able to add products to a cart, modify the quantity of items in the cart, and remove items as needed.**
    - **Checkout and Payment: The system must enable users to complete their purchase by providing valid payment information. The checkout process should calculate the total cost of the items, including shipping and taxes.**
* **Non-Functional Requirements:**
  + **Performance: The platform should load quickly, handle multiple users at the same time, and update shopping cart items in real-time.**
  + **Security: User data must be handled securely, particularly around sensitive areas like password storage and payment processing.**
  + **User-Friendly Design: The interface should be intuitive, responsive across devices, and easy to navigate, ensuring a seamless user experience.**

**3. Analysis and Design (4-5 minutes)**

* **System Architecture:**
  + **Our project uses the Model-View-Controller (MVC) architecture. This separates our code into distinct components: the Model, which handles the data; the View, which handles the user interface; and the Controller, which coordinates between the two.**
  + **Model: The model represents the core data structure of the project. It includes entities like User, Product, Order, and Cart.**
  + **View: The view is responsible for rendering the UI that the user interacts with. It consists of HTML files styled with CSS, and uses JavaScript for interactive elements.**
  + **Controller: The controller handles the logic that connects the user input from the front-end with the data from the back-end. It processes requests, queries the database, and sends data back to the user.**
* **UML Diagram of Classes: The primary classes are:**
  + **User: This class represents registered users in the system. It includes fields like userId, name, email, password, and role. The role distinguishes between customers and admins.**
  + **Product: Contains product details such as productId, name, description, price, and category.**
  + **Order: Stores order information, linking users to the products they purchase. Fields include orderId, userId, productId, and quantity.**
  + **Cart: Represents the shopping cart, containing products that users intend to purchase. It stores product IDs and quantities temporarily.**
  + **Payment: This class handles payment information, including validation of card details, expiration dates, and CVV.**

**sql**

**Copy code**

**+----------------+ +----------------+ +---------------+**

**| User | | Order | | Product |**

**+----------------+ +----------------+ +---------------+**

**| - userId | | - orderId | | - productId |**

**| - name |<----| - userId |------| - name |**

**| - email | | - productId | | - price |**

**+----------------+ | - quantity | | - category |**

**+----------------+ +---------------+**

**^**

**|**

**+-------------+**

**| Cart |**

**+-------------+**

**| - cartId |**

**| - productId |**

**+-------------+**

* **Design Patterns:**
  + **MVC (Model-View-Controller): This pattern separates our application into three interconnected components: the Model, View, and Controller. Each component has a specific responsibility, ensuring a clean and maintainable code structure.**
  + **Singleton: We used the Singleton pattern to manage database connections, ensuring that only one instance of the connection is created and used across the application.**
* **Data Structures:**
  + **HashMaps were used to efficiently store product IDs and quantities in the shopping cart.**
  + **ArrayLists were used to manage collections of products for users.**

**4. Database Design (2-3 minutes)**

* **Database Schema: Our database design ensures efficient storage and retrieval of data. We used MySQL as our database to store user, product, cart, and order information.**
  + **Users Table: Stores user information such as userId, email, password, and role.**
  + **Products Table: Stores product details, including productId, name, price, and category.**
  + **Orders Table: Stores orders made by users, linking the user to the product they bought with fields like orderId, userId, productId, and quantity.**
  + **Cart Table: Stores temporary cart data for users as they shop, tracking product IDs and quantities.**

**lua**

**Copy code**

**+-----------+ +----------+ +----------+**

**| Users | | Orders | | Products |**

**+-----------+ +----------+ +----------+**

**| userId |<-->| userId | | productId|**

**| email | | orderId | | price |**

**+-----------+ +----------+ +----------+**

* **Database Optimization:**
  + **We implemented indexes on frequently queried fields such as userId and productId to improve query performance.**
  + **Normalization was applied to ensure data redundancy was minimized, and relationships were properly established between tables.**

**5. User Interface (2-3 minutes)**

* **Front-End Design: The user interface was built using HTML, CSS, and Bootstrap for responsiveness. The key components include:**
  + **Product Grid: Displays products dynamically from the database, allowing users to browse by category.**
  + **Shopping Cart: Users can add products to their cart, which gets displayed on the checkout page. The cart dynamically updates both the quantity of products and the total cost.**
  + **Checkout and Payment: Users enter their payment details, which are validated before confirming the purchase.**
* **Dynamic Behavior with JavaScript:**
  + **Add to Cart: Uses JavaScript to dynamically update the cart when items are added.**
  + **Cart Totals: Real-time calculation of product totals based on the user’s input, ensuring that the user can see their total as they shop.**

**We also implemented form validation to ensure that inputs for fields like payment details adhere to the expected format.**

**6. Implementation Process (2-3 minutes)**

* **Technologies Used:**
  + **Back-End**: We used **Spring Boot** as the main framework for building the back-end services, handling everything from user authentication to payment processing.
  + **Database**: **MySQL** was used to store user data, product catalogs, and order history.
  + **Front-End**: **HTML**, **CSS**, **Bootstrap**, and **JavaScript** were used to design the interface and manage user interactions.
* **Code Quality and Testing:**
  + **JUnit** was used for testing the back-end services. We created unit tests for critical features like user login, cart operations, and payment processing to ensure that the system works as expected.
  + We also performed **manual testing** on the front-end to ensure that all user interactions, such as adding products to the cart and completing a purchase, function correctly across different devices and browsers.

**7. Accomplishments and Challenges (2-3 minutes)**

* **Accomplishments:**
  + **One of the key accomplishments is the smooth integration between the front-end and back-end, providing users with a seamless experience when browsing, adding items to the cart, and checking out.**
  + **Another significant accomplishment is the payment validation system. It ensures that users input correct card details in a secure manner, preventing erroneous submissions or invalid transactions.**
* **Challenges Faced:**
  + **One challenge was ensuring real-time synchronization between the cart on the front-end and the back-end database. We had to handle edge cases like users refreshing the page or removing items while keeping the data consistent.**
  + **Another challenge was implementing a responsive design that worked across different screen sizes, particularly for the product grid and cart functionality.**
* **Lessons Learned:**
  + **We gained a deeper understanding of full-stack development, particularly the complexities of managing both the front-end and back-end systems. Security, especially around user authentication and payment processing, was an area that required careful attention.**
* **What Would We Change?**
  + **If we could release a future version, we would add more advanced features such as dynamic pricing algorithms based on stock levels or user demand. We would also consider implementing recommendation systems to suggest products to users based on their browsing history.**

**Sure! Here's an extensive description of your project that should help you present for about 15 minutes. Feel free to adjust the pace and expand on specific sections during the presentation to ensure you cover everything comprehensively.**

**1. Introduction to the Project (1-2 minutes)**

* **Project Overview:**
  + **Our project, TheTechHeads, is an e-commerce platform designed to sell tech-oriented footwear, with a primary focus on Nike shoes. The platform offers users the ability to browse a variety of shoe products, add them to their shopping cart, and securely process payments. The site includes functionality for user registration and authentication, product filtering by categories (men, women, kids), and a streamlined checkout process.**
* **Why is This Project Important?**
  + **Online shopping has become a central aspect of consumer culture, especially with the increasing shift towards digital retail platforms. We chose this project to build a full-stack solution that addresses the essential components of an e-commerce system, including user authentication, product management, shopping cart functionality, and payment processing. This project allowed us to explore critical areas of web development such as front-end design, back-end development, security, and database management. Additionally, it’s an opportunity to simulate the challenges and rewards faced in real-world projects, especially as e-commerce continues to grow globally.**

**2. Project Requirements (2-3 minutes)**

* **Functional Requirements:**
  + **Our system needed to meet several key functional requirements:**
    - **User Registration and Login: Users must be able to create an account and log in securely. After logging in, they should be able to update their profile information.**
    - **Product Display and Browsing: Users should be able to view a catalog of products that are filtered into categories like Men, Women, and Kids.**
    - **Shopping Cart: Users should be able to add products to a cart, modify the quantity of items in the cart, and remove items as needed.**
    - **Checkout and Payment: The system must enable users to complete their purchase by providing valid payment information. The checkout process should calculate the total cost of the items, including shipping and taxes.**
* **Non-Functional Requirements:**
  + **Performance: The platform should load quickly, handle multiple users at the same time, and update shopping cart items in real-time.**
  + **Security: User data must be handled securely, particularly around sensitive areas like password storage and payment processing.**
  + **User-Friendly Design: The interface should be intuitive, responsive across devices, and easy to navigate, ensuring a seamless user experience.**

**3. Analysis and Design (4-5 minutes)**

* **System Architecture:**
  + **Our project uses the Model-View-Controller (MVC) architecture. This separates our code into distinct components: the Model, which handles the data; the View, which handles the user interface; and the Controller, which coordinates between the two.**
  + **Model: The model represents the core data structure of the project. It includes entities like User, Product, Order, and Cart.**
  + **View: The view is responsible for rendering the UI that the user interacts with. It consists of HTML files styled with CSS, and uses JavaScript for interactive elements.**
  + **Controller: The controller handles the logic that connects the user input from the front-end with the data from the back-end. It processes requests, queries the database, and sends data back to the user.**
* **UML Diagram of Classes: The primary classes are:**
  + **User: This class represents registered users in the system. It includes fields like userId, name, email, password, and role. The role distinguishes between customers and admins.**
  + **Product: Contains product details such as productId, name, description, price, and category.**
  + **Order: Stores order information, linking users to the products they purchase. Fields include orderId, userId, productId, and quantity.**
  + **Cart: Represents the shopping cart, containing products that users intend to purchase. It stores product IDs and quantities temporarily.**
  + **Payment: This class handles payment information, including validation of card details, expiration dates, and CVV.**

**sql**

**Copy code**

**+----------------+ +----------------+ +---------------+**

**| User | | Order | | Product |**

**+----------------+ +----------------+ +---------------+**

**| - userId | | - orderId | | - productId |**

**| - name |<----| - userId |------| - name |**

**| - email | | - productId | | - price |**

**+----------------+ | - quantity | | - category |**

**+----------------+ +---------------+**

**^**

**|**

**+-------------+**

**| Cart |**

**+-------------+**

**| - cartId |**

**| - productId |**

**+-------------+**

* **Design Patterns:**
  + **MVC (Model-View-Controller): This pattern separates our application into three interconnected components: the Model, View, and Controller. Each component has a specific responsibility, ensuring a clean and maintainable code structure.**
  + **Singleton: We used the Singleton pattern to manage database connections, ensuring that only one instance of the connection is created and used across the application.**
* **Data Structures:**
  + **HashMaps were used to efficiently store product IDs and quantities in the shopping cart.**
  + **ArrayLists were used to manage collections of products for users.**

**4. Database Design (2-3 minutes)**

* **Database Schema: Our database design ensures efficient storage and retrieval of data. We used MySQL as our database to store user, product, cart, and order information.**
  + **Users Table: Stores user information such as userId, email, password, and role.**
  + **Products Table: Stores product details, including productId, name, price, and category.**
  + **Orders Table: Stores orders made by users, linking the user to the product they bought with fields like orderId, userId, productId, and quantity.**
  + **Cart Table: Stores temporary cart data for users as they shop, tracking product IDs and quantities.**

**lua**

**Copy code**

**+-----------+ +----------+ +----------+**

**| Users | | Orders | | Products |**

**+-----------+ +----------+ +----------+**

**| userId |<-->| userId | | productId|**

**| email | | orderId | | price |**

**+-----------+ +----------+ +----------+**

* **Database Optimization:**
  + **We implemented indexes on frequently queried fields such as userId and productId to improve query performance.**
  + **Normalization was applied to ensure data redundancy was minimized, and relationships were properly established between tables.**

**5. User Interface (2-3 minutes)**

* **Front-End Design: The user interface was built using HTML, CSS, and Bootstrap for responsiveness. The key components include:**
  + **Product Grid: Displays products dynamically from the database, allowing users to browse by category.**
  + **Shopping Cart: Users can add products to their cart, which gets displayed on the checkout page. The cart dynamically updates both the quantity of products and the total cost.**
  + **Checkout and Payment: Users enter their payment details, which are validated before confirming the purchase.**
* **Dynamic Behavior with JavaScript:**
  + **Add to Cart: Uses JavaScript to dynamically update the cart when items are added.**
  + **Cart Totals: Real-time calculation of product totals based on the user’s input, ensuring that the user can see their total as they shop.**

**We also implemented form validation to ensure that inputs for fields like payment details adhere to the expected format.**

**6. Implementation Process (2-3 minutes)**

* **Technologies Used:**
  + **Back-End: We used Spring Boot as the main framework for building the back-end services, handling everything from user authentication to payment processing.**
  + **Database: MySQL was used to store user data, product catalogs, and order history.**
  + **Front-End: HTML, CSS, Bootstrap, and JavaScript were used to design the interface and manage user interactions.**
* **Code Quality and Testing:**
  + **JUnit was used for testing the back-end services. We created unit tests for critical features like user login, cart operations, and payment processing to ensure that the system works as expected.**
  + **We also performed manual testing on the front-end to ensure that all user interactions, such as adding products to the cart and completing a purchase, function correctly across different devices and browsers.**

**7. Accomplishments and Challenges (2-3 minutes)**

* **Accomplishments:**
  + **One of the key accomplishments is the smooth integration between the front-end and back-end, providing users with a seamless experience when browsing, adding items to the cart, and checking out.**
  + **Another significant accomplishment is the payment validation system. It ensures that users input correct card details in a secure manner, preventing erroneous submissions or invalid transactions.**
* **Challenges Faced:**
  + **One challenge was ensuring real-time synchronization between the cart on the front-end and the back-end database. We had to handle edge cases like users refreshing the page or removing items while keeping the data consistent.**
  + **Another challenge was implementing a responsive design that worked across different screen sizes, particularly for the product grid and cart functionality.**
* **Lessons Learned:**
  + **We gained a deeper understanding of full-stack development, particularly the complexities of managing both the front-end and back-end systems. Security, especially around user authentication and payment processing, was an area that required careful attention.**
* **What Would We Change?**
  + **If we could release a future version, we would add more advanced features such as dynamic pricing algorithms based on stock levels or user demand. We would also consider implementing recommendation systems to suggest products to users based on their browsing history.**

**8. Conclusion (1 minute)**

* **In conclusion, our project TheTechHeads provided us with valuable hands-on experience in building a full-stack e-commerce application. From the database design to front-end interactivity, we were able to create a functional and user-friendly platform. We are particularly proud of the integration between the shopping cart, checkout process, and payment validation, which delivers a professional-grade user experience. Through this project, we’ve learned about the importance of clean architecture, secure user data handling, and creating a responsive and dynamic interface that adapts to user needs.**