CS5054NP

Module Code & Module Title

CS5054NP Advanced Programming and Technologies

Assessment Type

50% Group Coursework 1

Semester- 4

2024 Autumn

Student Name	London Met ID
Prabal Gurung	22069041
Aman Gurung	22068950
Prabin Thapa Magar	22069043
Rachit Raj Shrestha	22069056
Rahul G.C.	22069057

Assignment Due Date: Friday, May 10, 2024

Assignment Submission Date: Friday, May 10, 2024

Submitted To: Mr. Sandip Adhikari

Declaration

I confirm that I understand my coursework needs to be submitted online via My second teacher under the relevant module page before the deadline for my assignment to be accepted and marked. I am fully aware that late submission will be treated as non-submission and a mark of zero will be awarded

Table of Contents:

Table of Figures:

Table of Table:

1. Introduction:

Nexus Nimbus emerges as a revolutionary force in the field of ecommerce, offering a wide selection of the best laptops. The meaning behind the name “Nexus Nimbus” is a central point for wide selection with speedy delivery of our product. Our website Nexus Nimbus offers a comprehensive platform that is dedicated exclusively for the sale of laptops.

Nexus Nimbus is a web-based system that intends to provides its user with the capability to search, explore and purchase electronics specifically laptops. The website follows the MVC pattern for development. The model package containing the model classes of the system is used for storing and retrieving the back-end data store, controller package contains all the servlets that are used to handle the user’s request and the view package contains all the JSP, CSS and JavaScript pages. The users can create their account using the login/signup feature providing their personal information which is saved in the database. On registration by the user, the user session is started and the user is provided exclusive features such as cart, edit profile etc. The website’s UI is made to be user friendly so the user has no difficulty in using the site. The site also provides an admin panel from which the admin is able to add new product to the site, delete the existing products and update relevant information for the existing products.

In this report we will go through the user interface design of the website along with the class diagram and database schema. This report also provides the names of the method used in the development of the system along with its description. A number of test cases are also conducted to test the functionalities of the site. It also portrays our entire journey of developing the system along with the challenges and problem faced during the journey.

1.1 Aims:

The major aim of this project is to build a fully operational E-commerce website featuring an extensive variety of electronics and gadgets. The site aims to provide a platform that allows for seamless shopping experience to its customer with varieties of product and their detailed information with exceptional customer service.

1.2 Objectives:

The objectives of our project include:

i. Login functionality with user authentication as well as password encryption.

ii. Seamless ordering of product along with the feature to add, remove and view the cart items.

iii. Search feature that allows user to find the products based on their prices and names.

iv. Contact function where users can reach out via phone or email for queries.

v. View and edit profile function when user is logged in.

vi. Admin should be able to add, update and delete the products in the site.

vii. Redirection and clear messages should be displayed in case of failure and issues.

viii. Logout feature for both user and admin along with session expiration.

2. User Interface Design:

The user interface is arguably the most important element of a computer-based system or product. If the interface is poorly designed, the user's ability to tap the computational power of an application may be severely hindered. In fact, a weak interface may cause an otherwise well-designed and solidly implemented application to fail. The user interface is the window into the software. In many cases, the interface molds a user's perception of the quality of the system. If the "window" is smudged, wavy, or broken, the user may reject an otherwise powerful computer-based system. (S.Sridevi, 2014)

Our project Nexus Nimbus has an elegant and user-friendly design. The design of our project has a simple layout with clear navigation, icons, labels and messages to ensure an easily operable site. The pages provide a comprehensive view of the platform's product. The user interface design of our site aims to provide the user an effortless enjoyable user experience.

2.1 Wireframe:

A wireframe is a visual representation or skeletal outline of a web page, mobile app, or software interface that shows the basic layout, structure, and functionality of the final product. It serves as a blueprint or prototype that helps designers, developers, and stakeholders to quickly visualize and test the user interface and user experience (UI/UX) design before investing time and resources into creating a fully functional product. (Velarde, 2023)

The wireframe for our site Nexus Nimbus is given as follows:

i. Wireframe of home page:

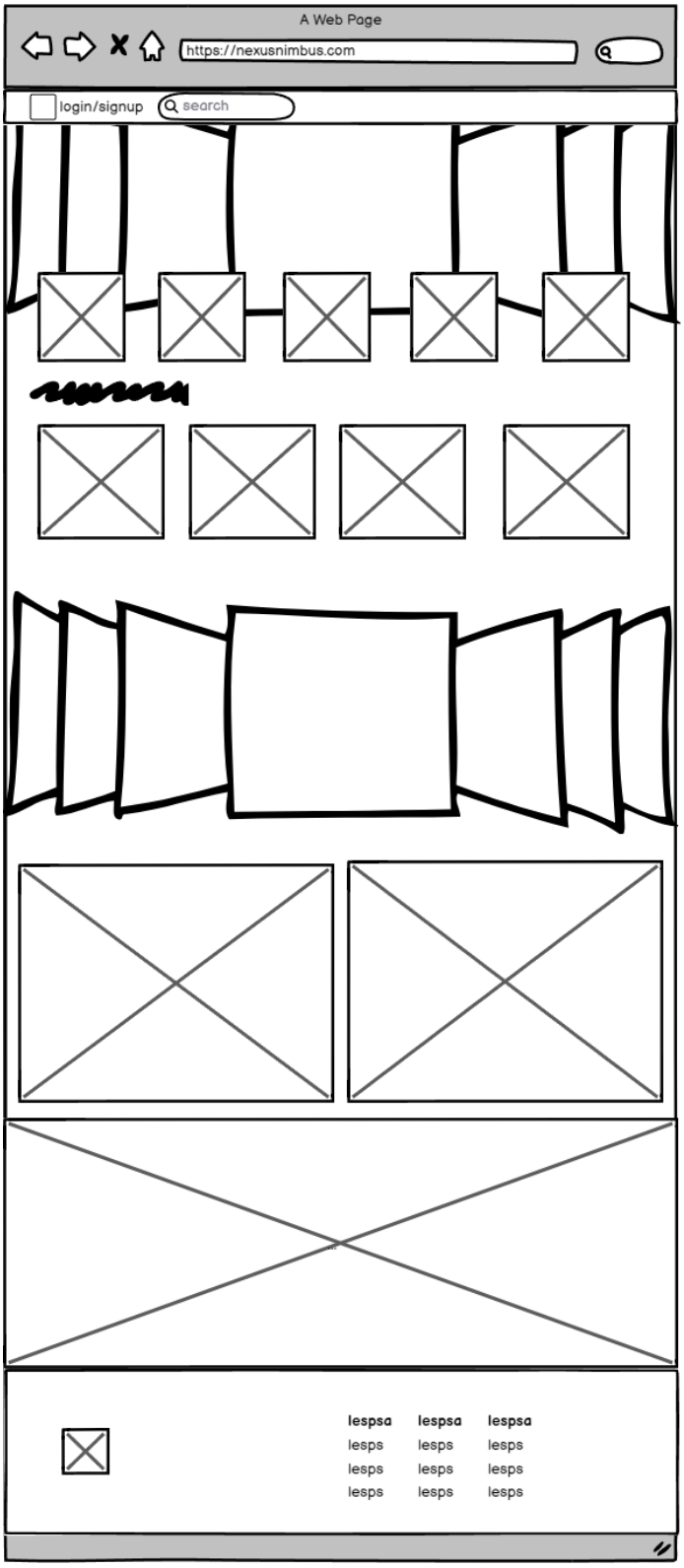

Figure 1: Wireframe of Home page.

ii. Wireframe of product page:

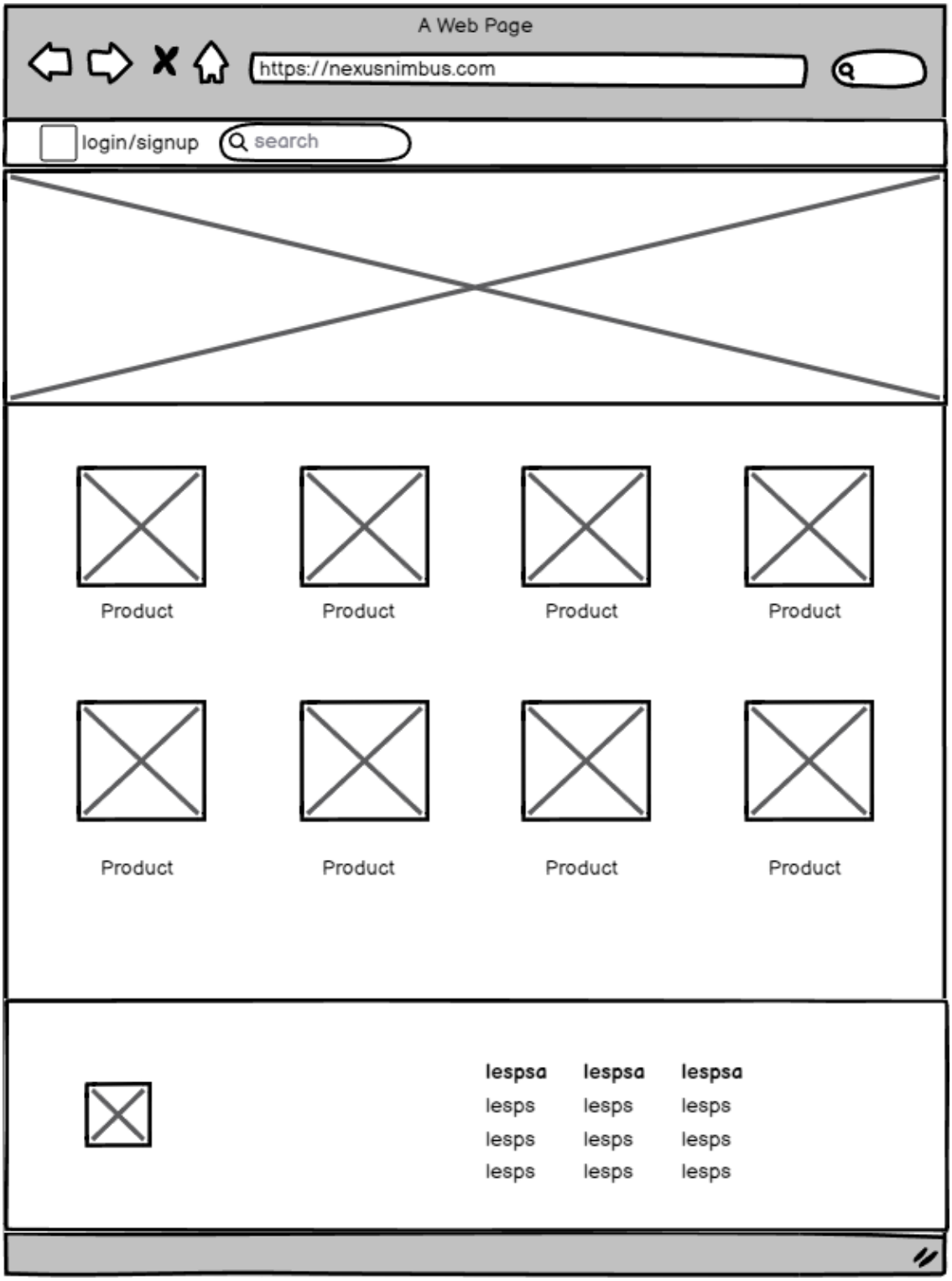

Figure 2: Wireframe of product page.

iii. Wireframe of contact us page:

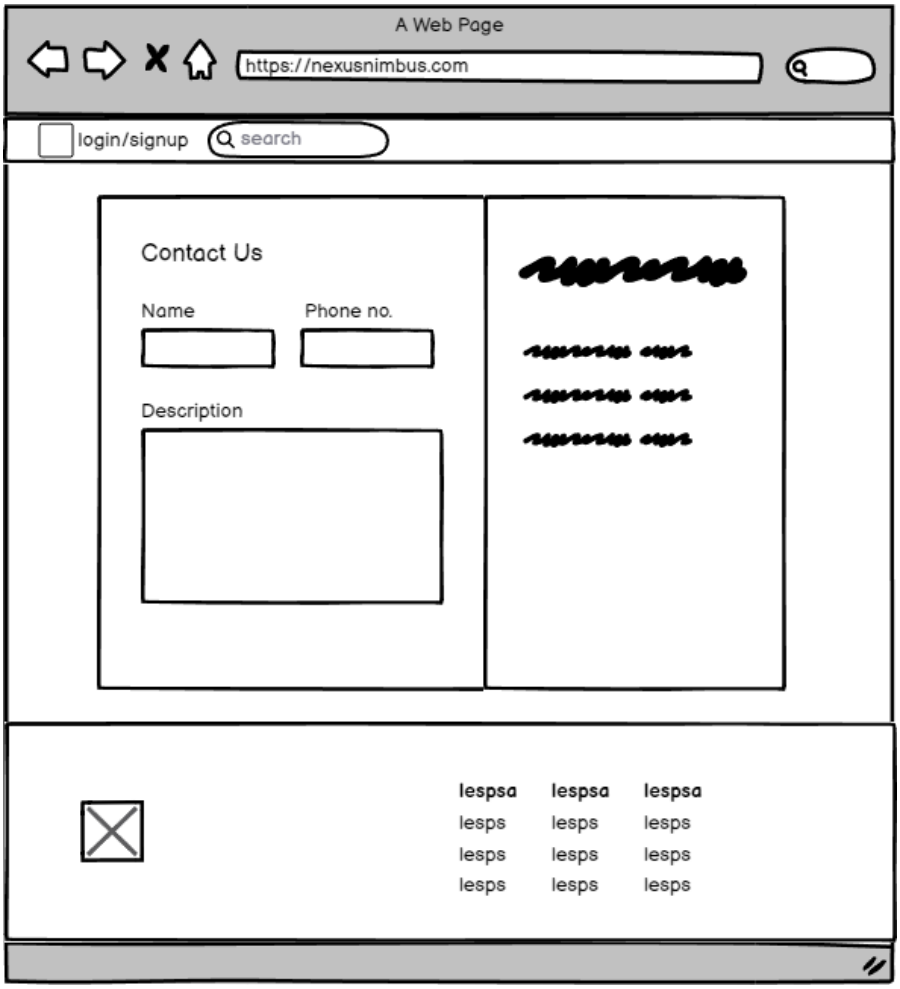

Figure 3: Wireframe of Contact us page.

iv. Wireframe of registration page:

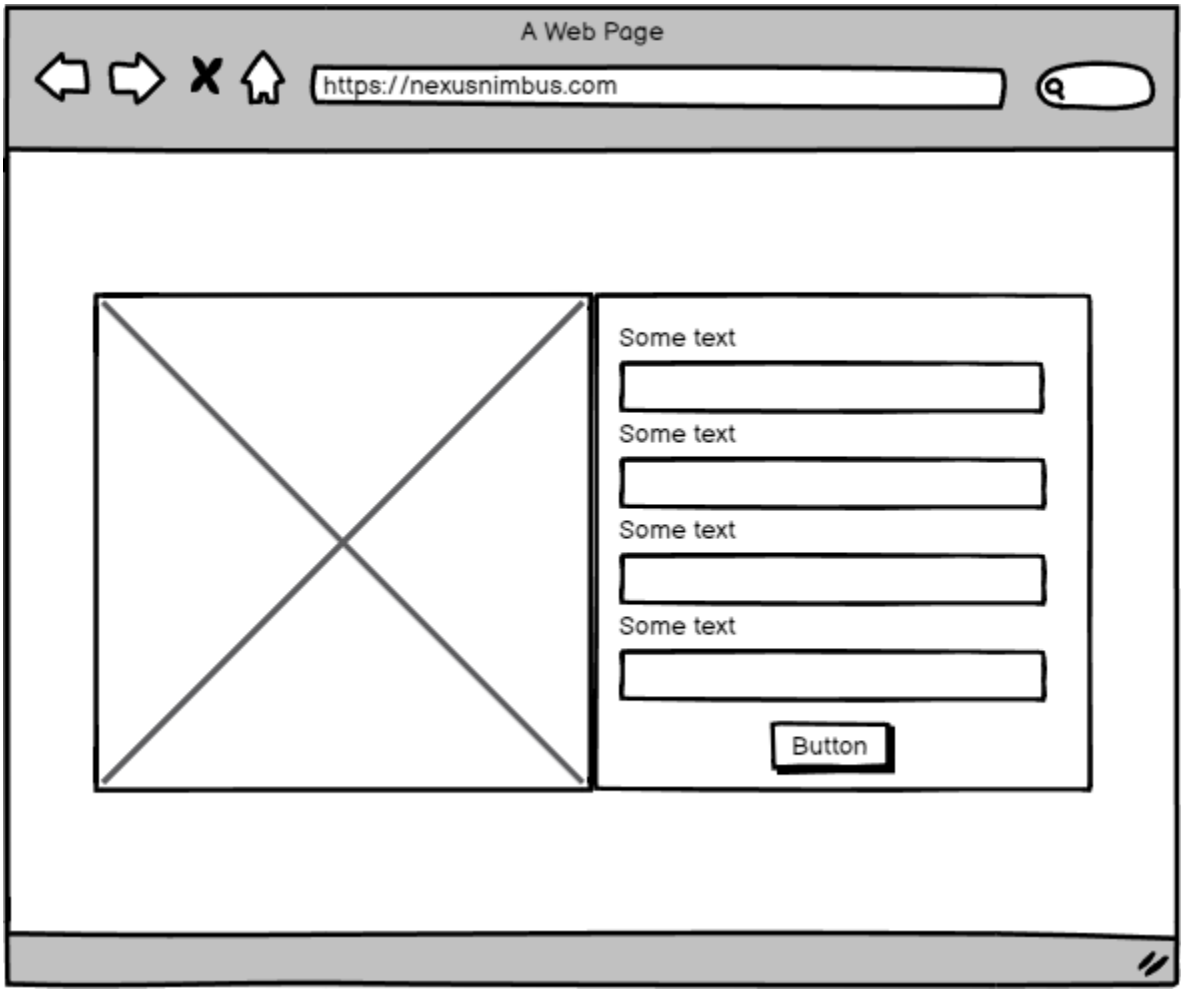

Figure 4: Wireframe of registration page.

v. Wireframe of Login page:

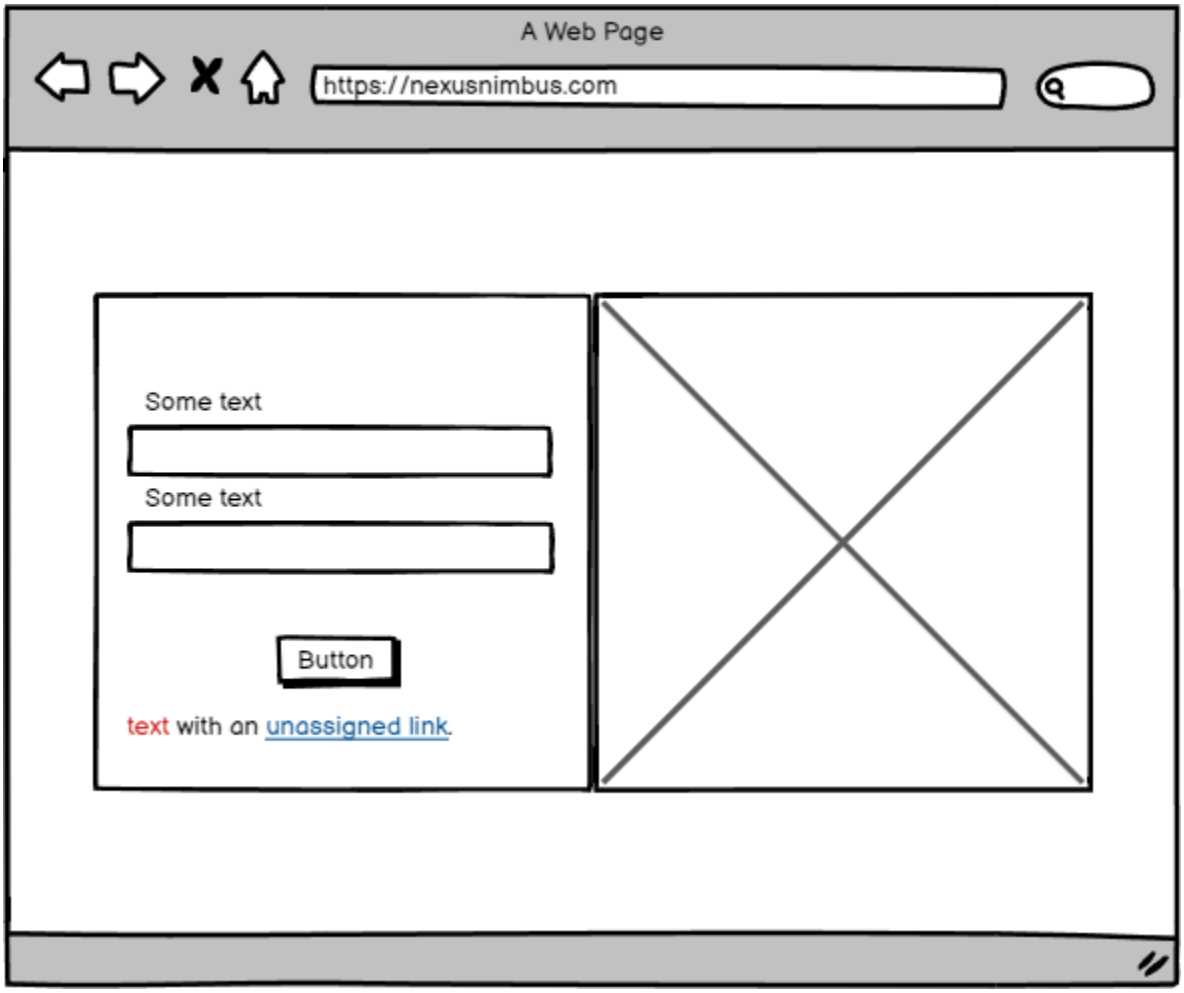

Figure 5: Wireframe of login page.

vi. Wireframe of profile page:

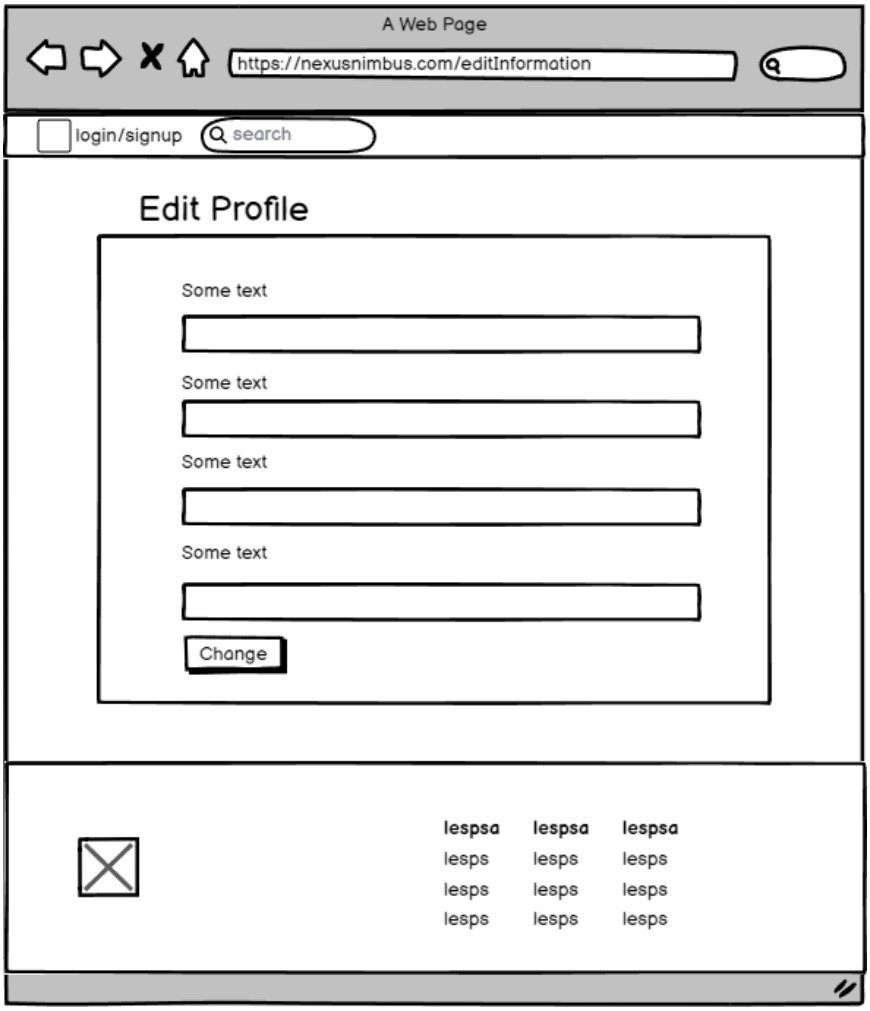

Figure 6: Wireframe of profile page.

vii. Wireframe of product detail page:

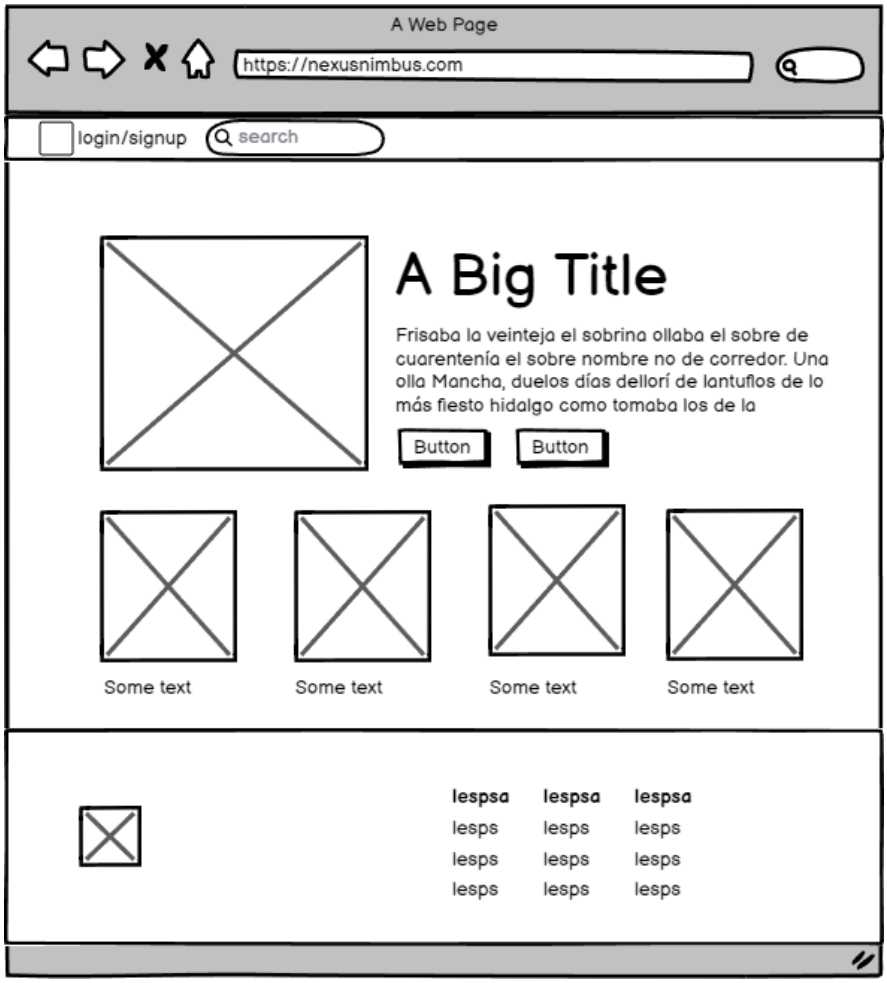

Figure 7: Wireframe of product detail page.

viii. Wireframe of cart page:

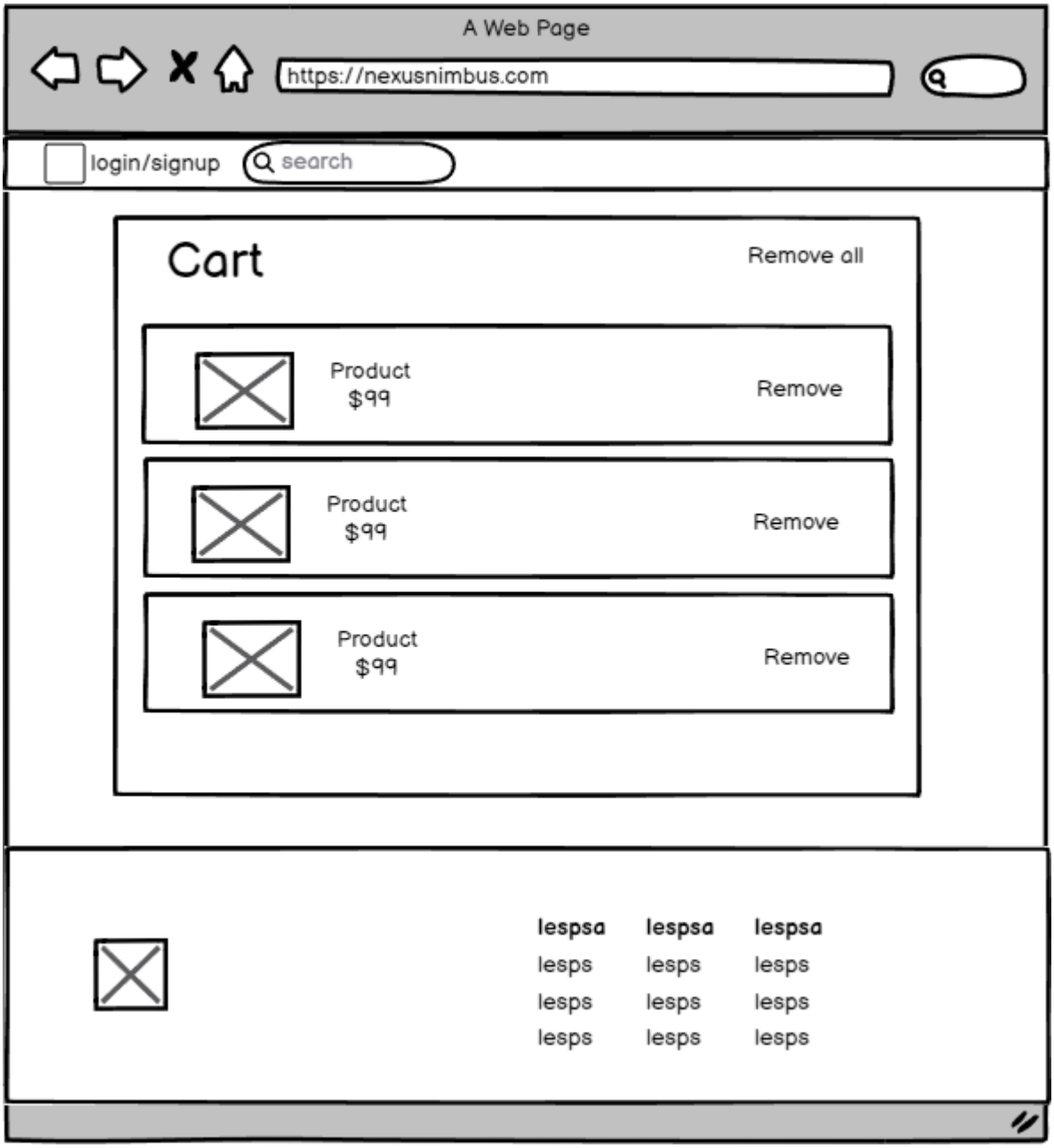

Figure 8: Wireframe of cart page.

ix. Wireframe of admin view product page:

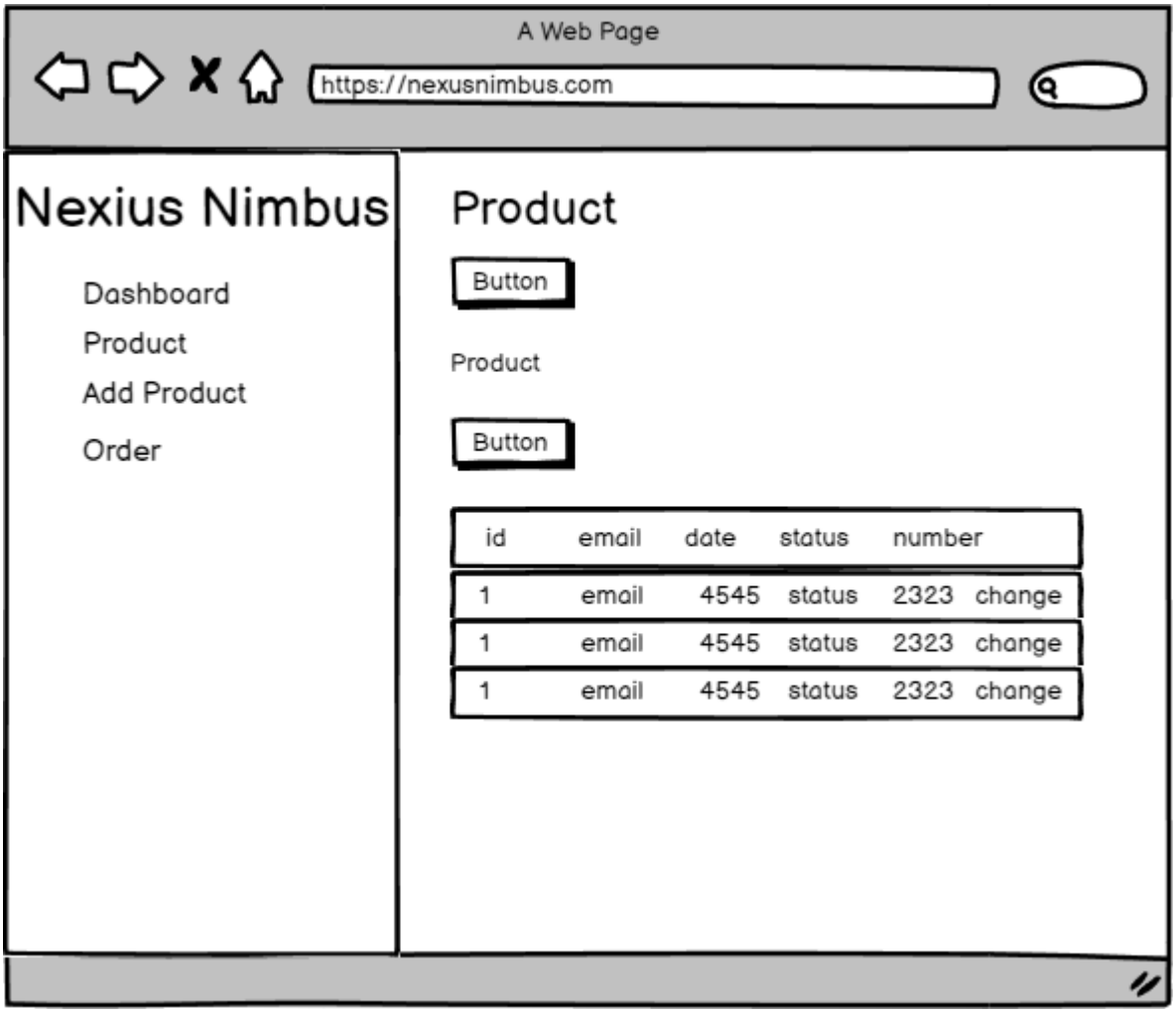

Figure 9: Wireframe of admin view product page

x. Wireframe of admin view order page:

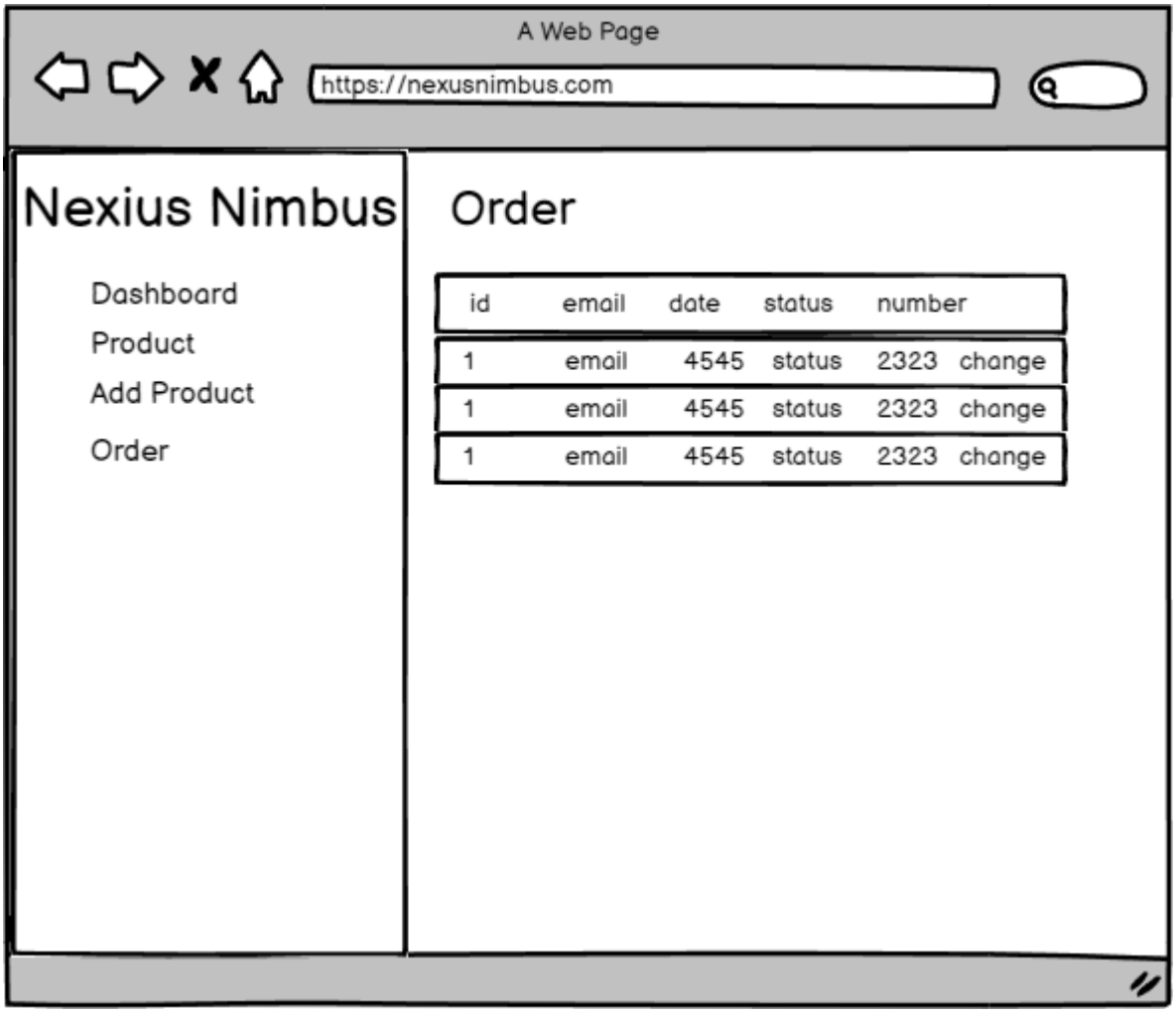

Figure 10: Wireframe of admin view order page.

xi. Wireframe of admin add product page:

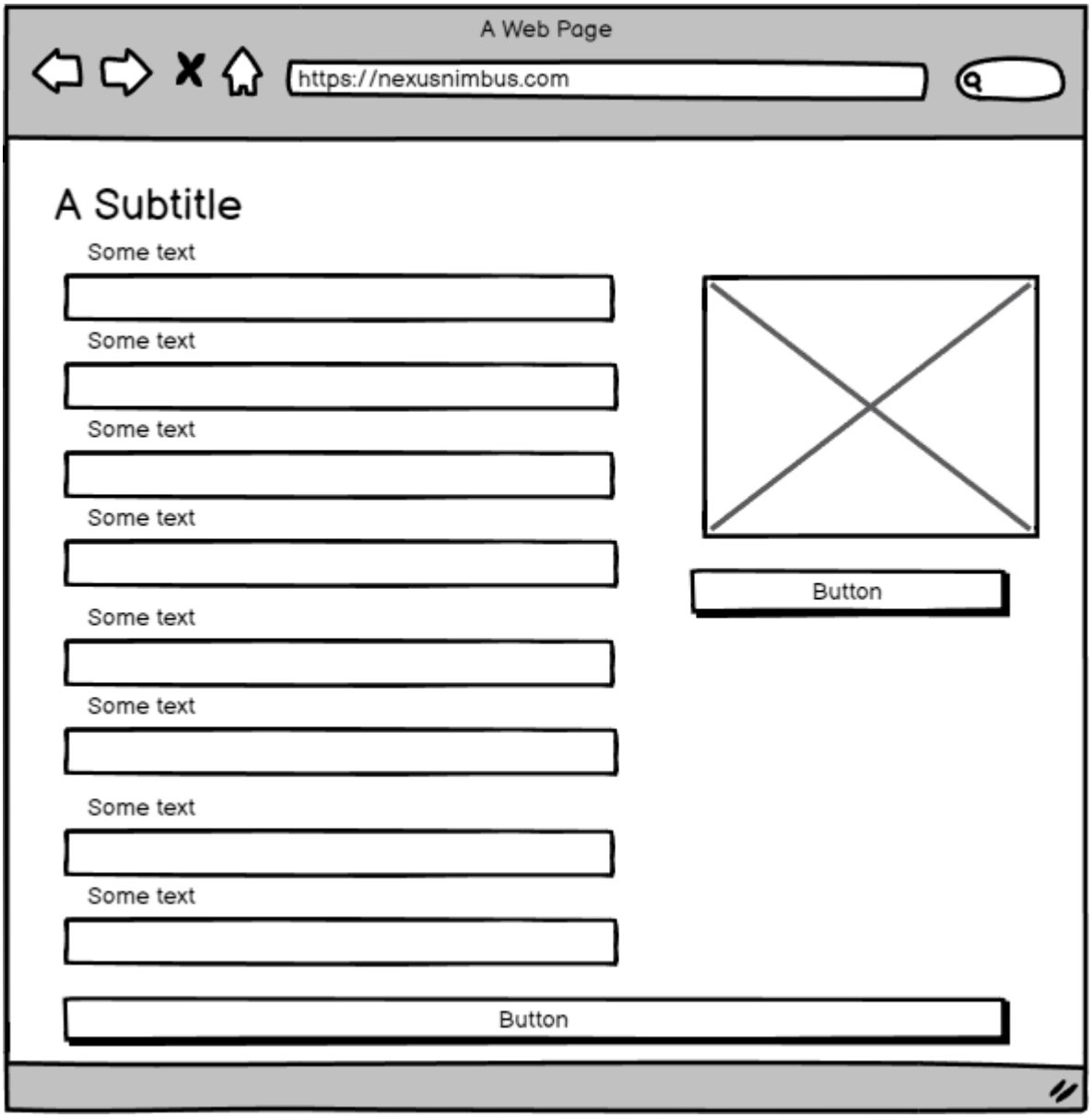

Figure 11: Wireframe of admin add product page.

xii. Wireframe of order history page:

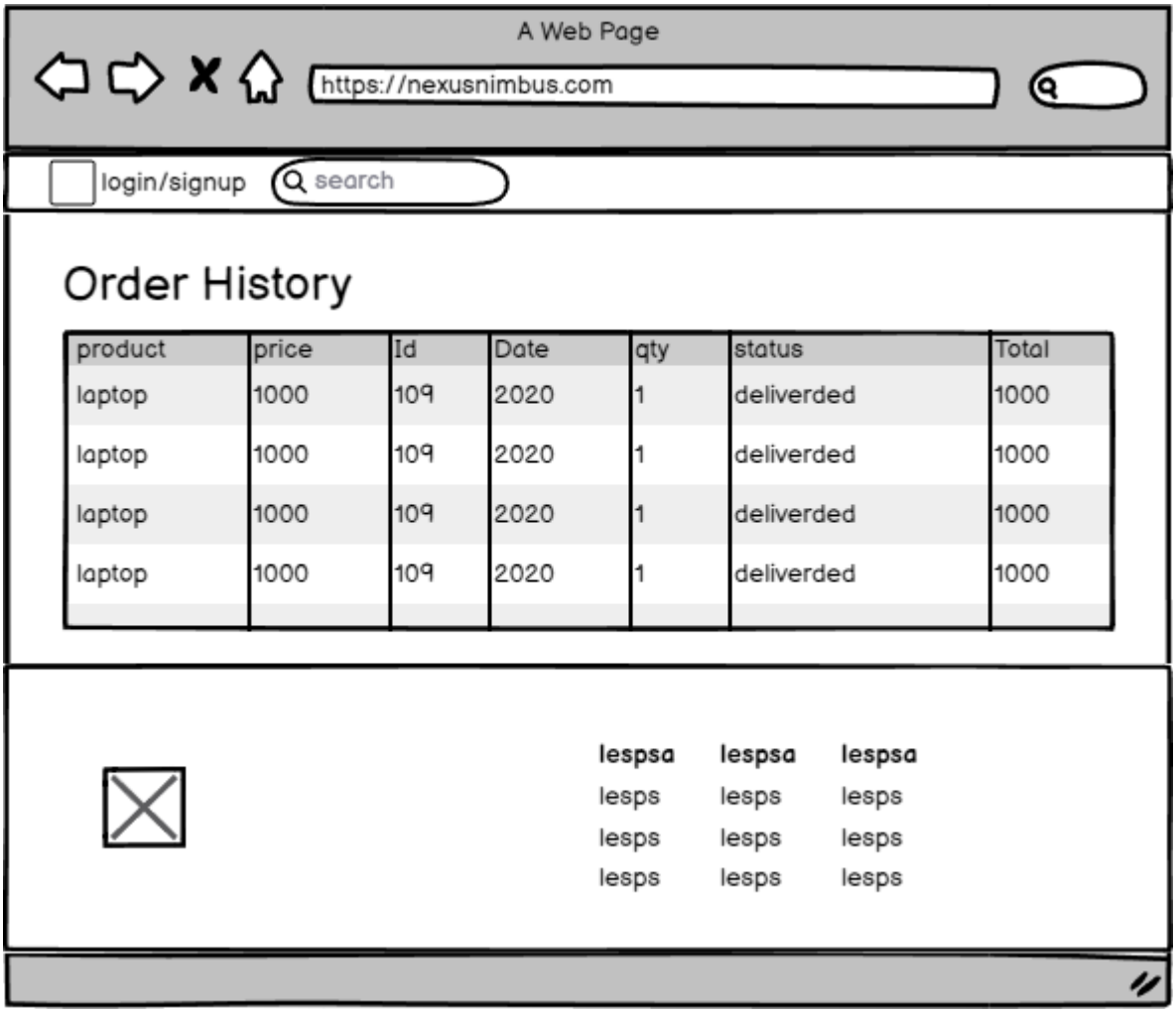

Figure 12: Wireframe of order history page.

2.2 Actual Design:

The actual design for our website Nexus Nimbus is as follows:

i. Actual design of home page:

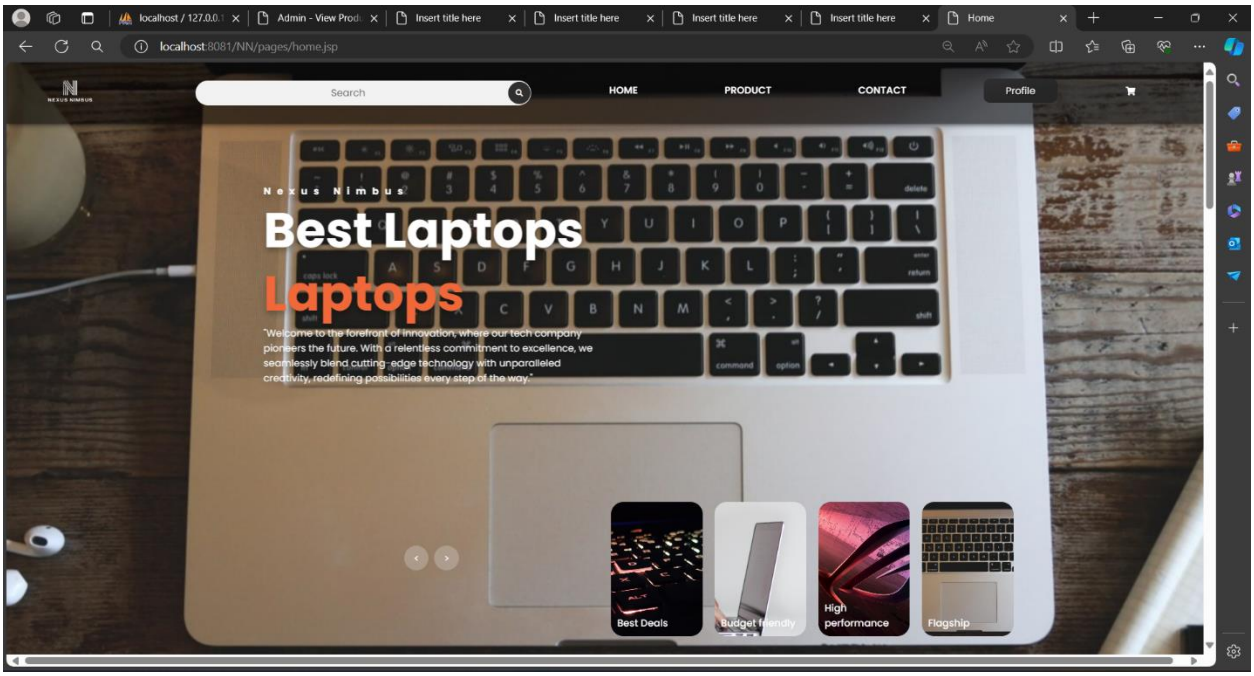

Figure 13: Actual design of home page

ii. Actual design of product page:

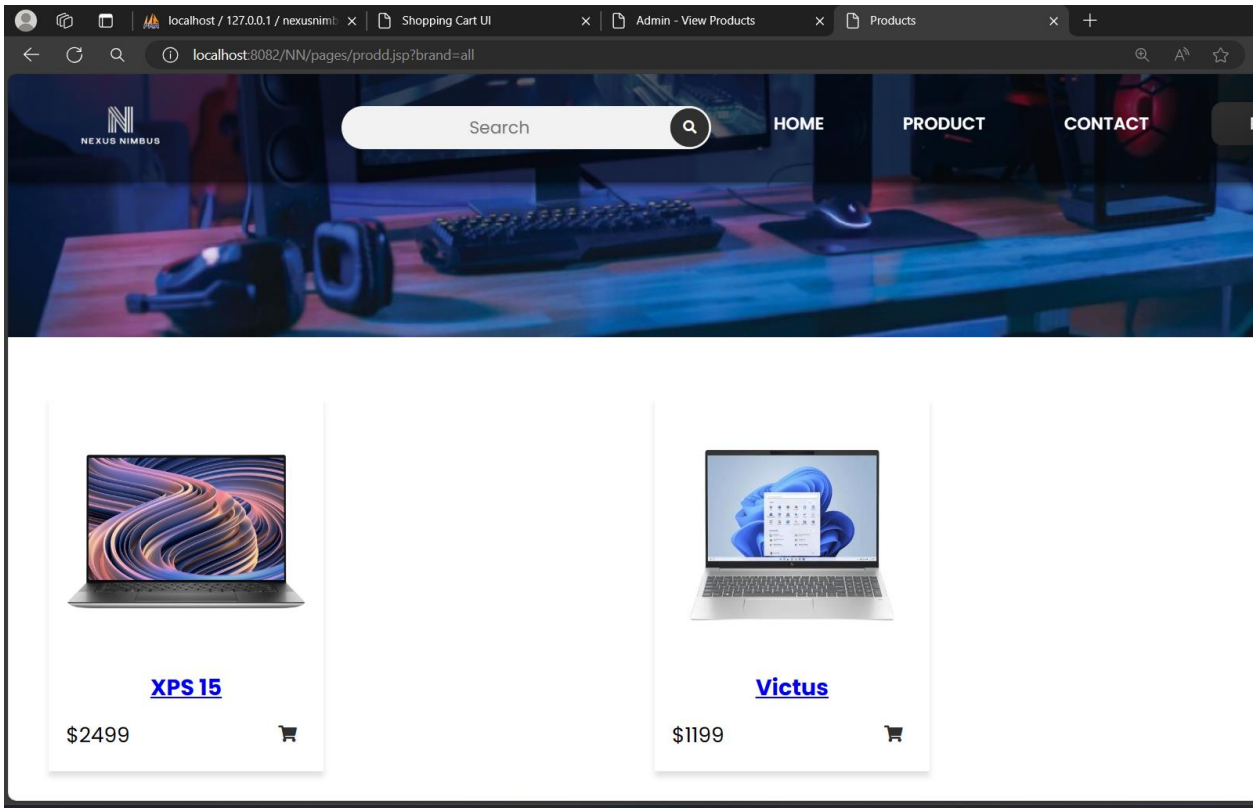

Figure 14: Actual design of product page

iii. Actual design of contact us page:

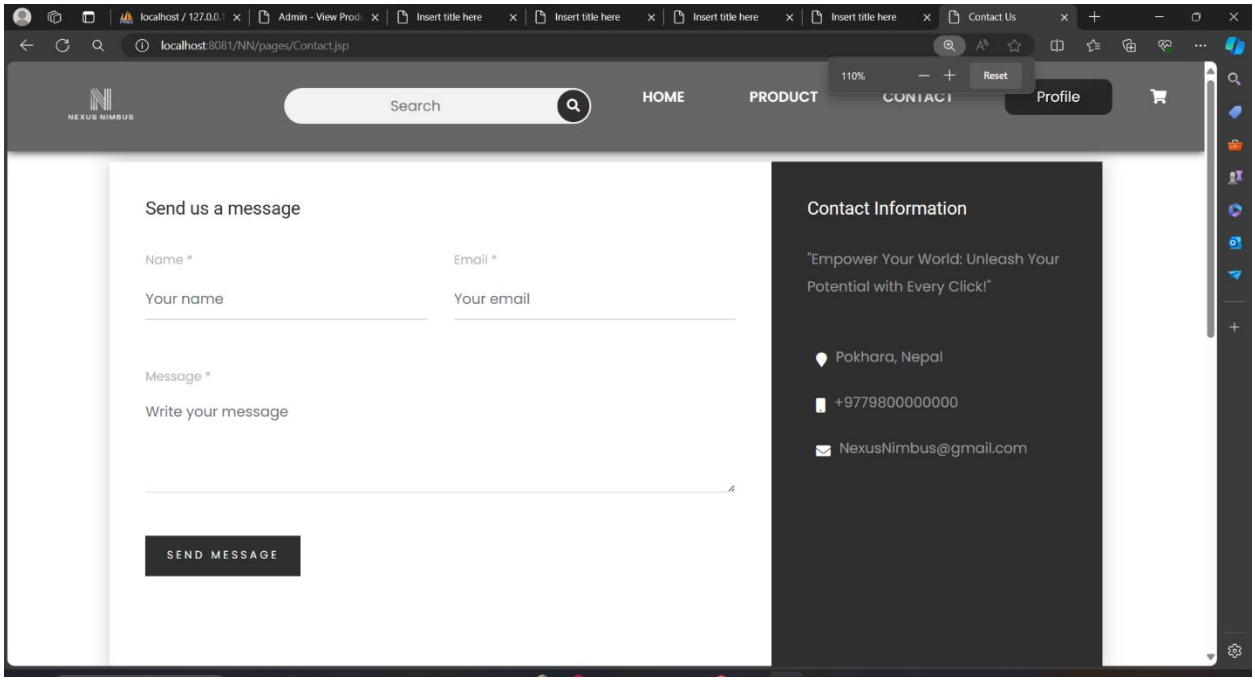

Figure 15: Actual design of contact us page

iv. Actual design of registration page:

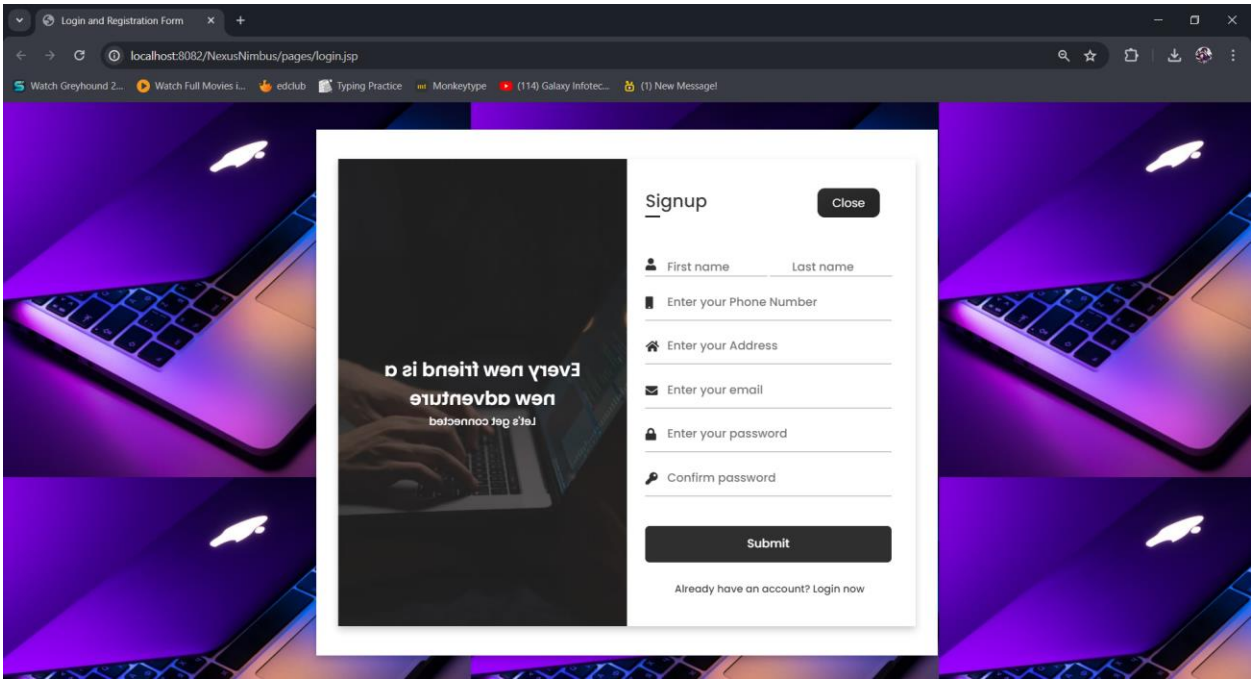

Figure 16: Actual design of registration page

v. Actual design of login page:

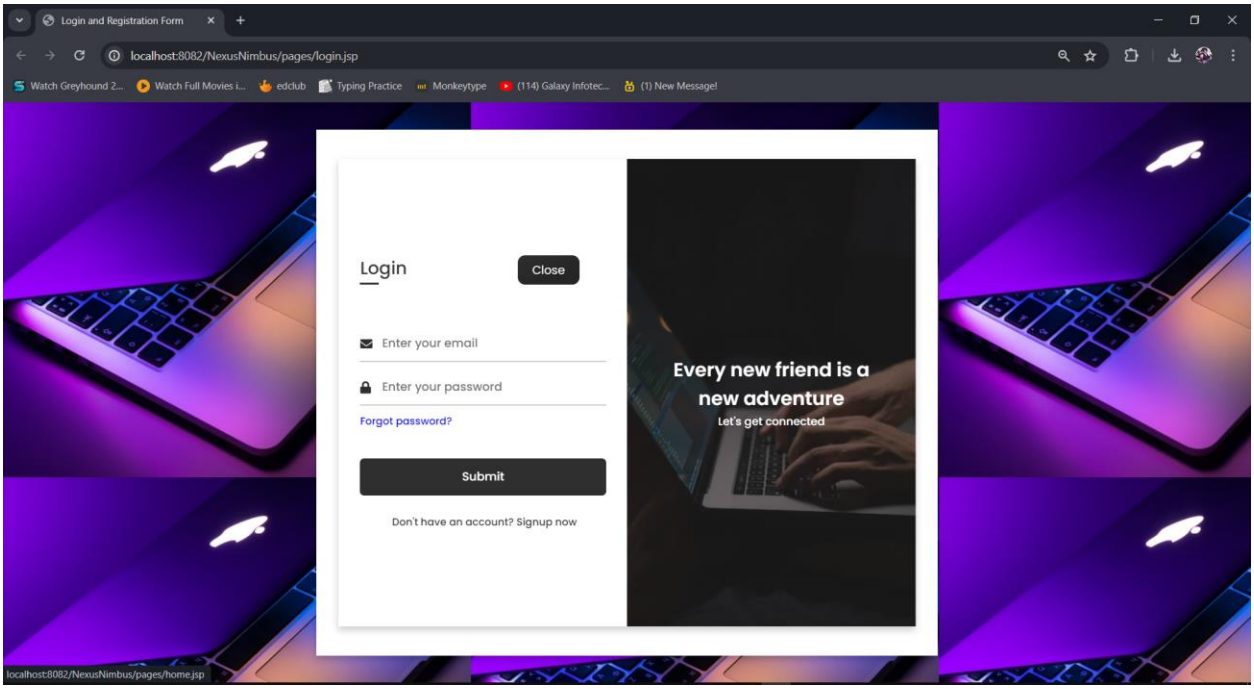

Figure 17: Actual design of login page

vi. Actual design of profile page:

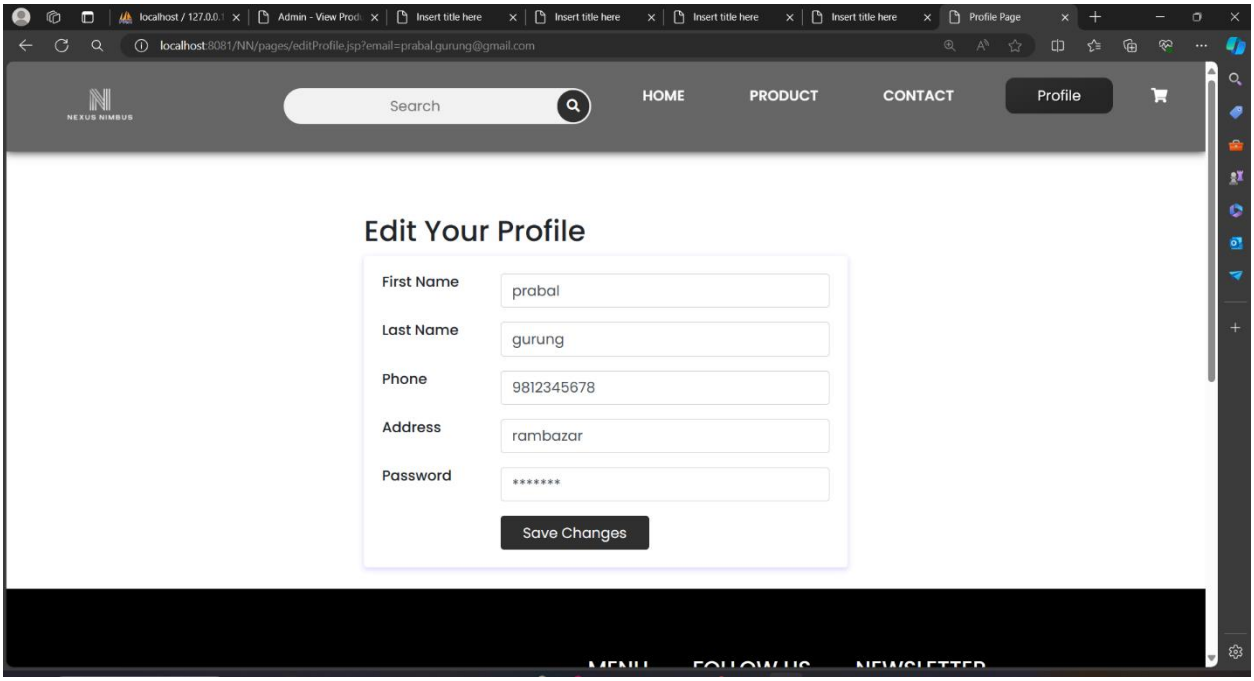

Figure 18: Actual design of profile page

vii. Actual design of product detail page:

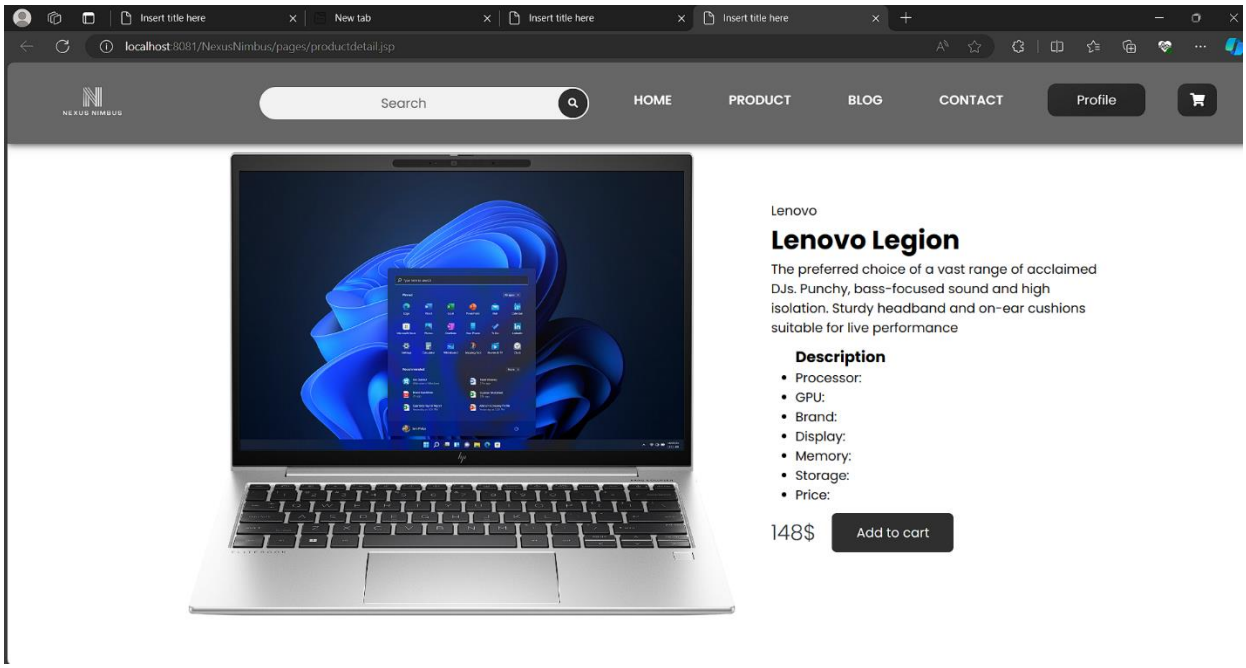

Figure 19: Actual design of product detail page

viii. Actual design of cart page:

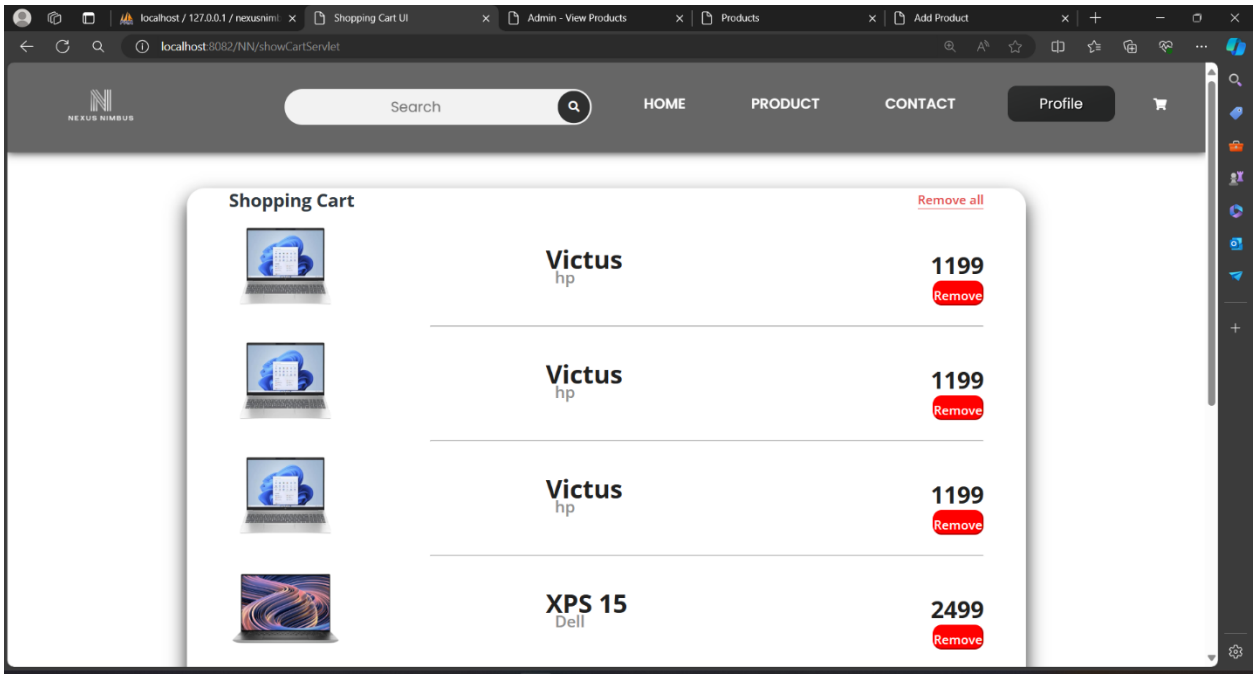

Figure 20: Actual design of cart page

ix. Actual design of admin view product page:

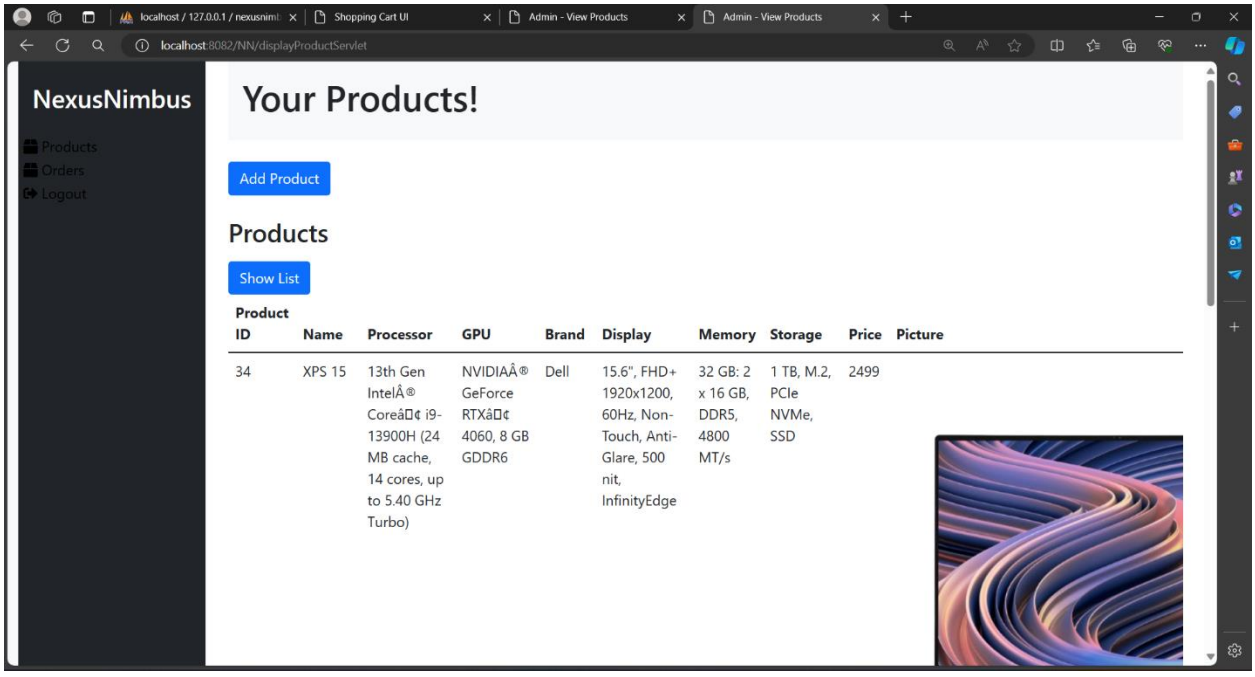

Figure 21: Actual design of admin view product page

x. Actual design of admin view order page:

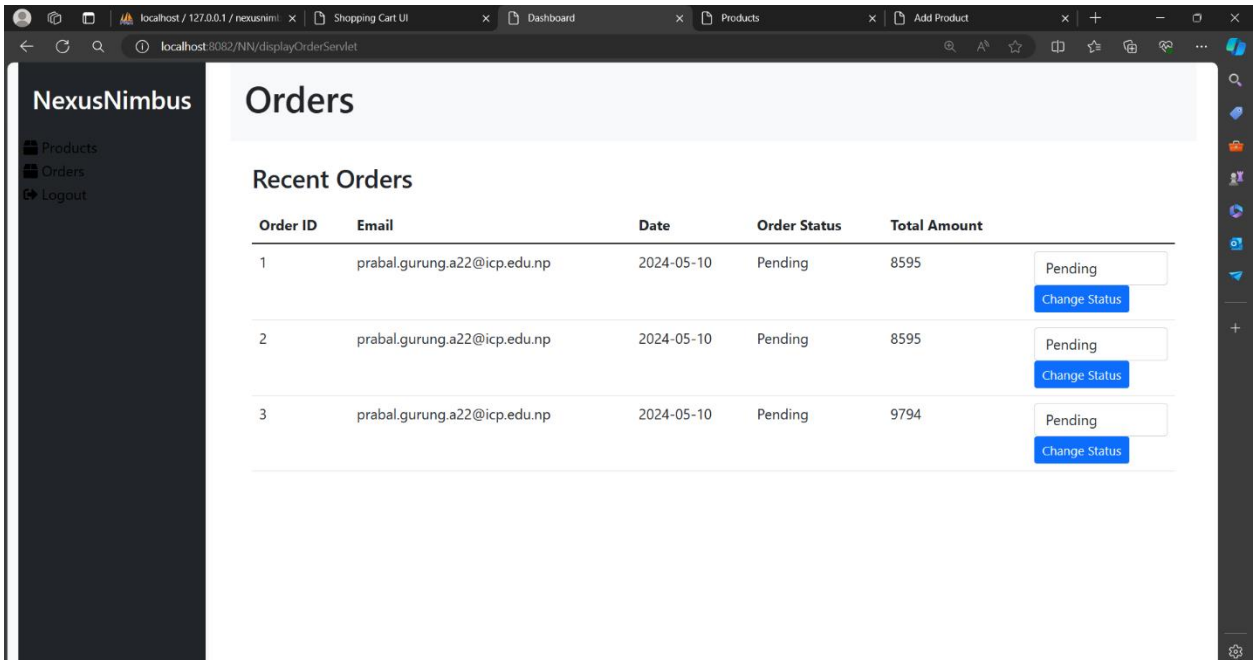

Figure 22: Actual design of admin view order page

xi. Actual design of admin add product page:

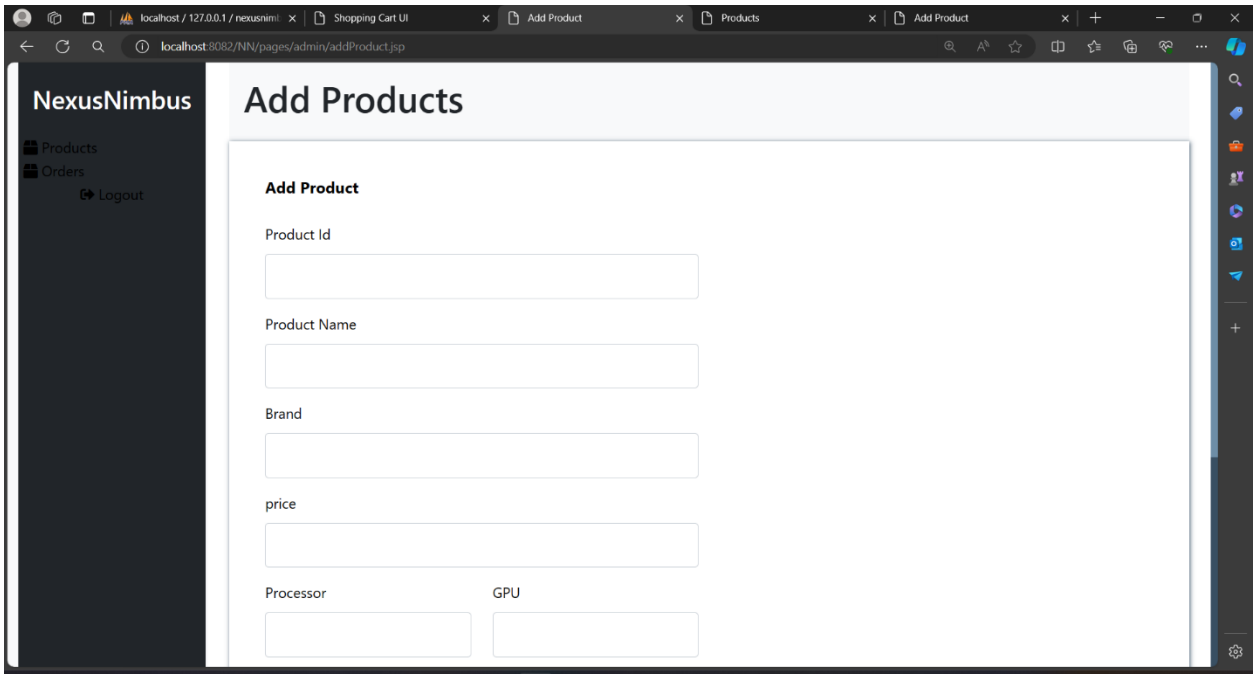

Figure 23: Actual design of admin add product page

xii. Actual design of order history page:

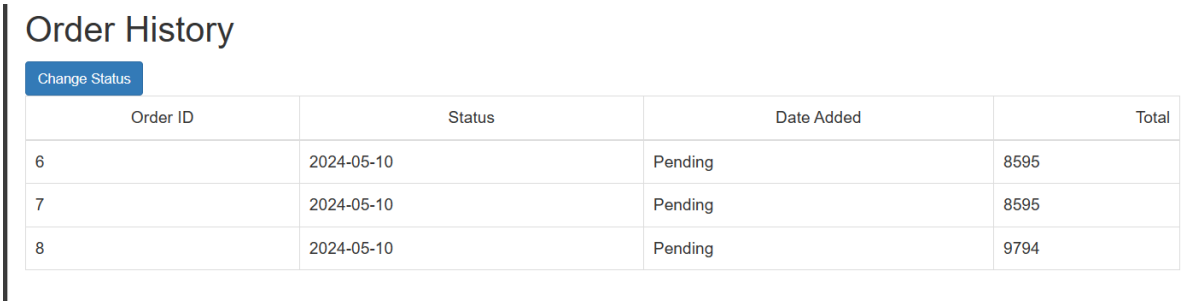

Figure 24: Actual design of order history page

3. Class Diagram:

The class diagram is the most important and most widely used description of an Object-Oriented system. It shows the static structure of the core classes that are used to build a system. The most relevant features (attributes and methods) of each class are provided in the class diagram, together with the optional indication of some of their properties (visibility, type, etc.). (Springer, 2005)

3.1 Overall Class Diagram:

The overall of class diagram of our project is given below:

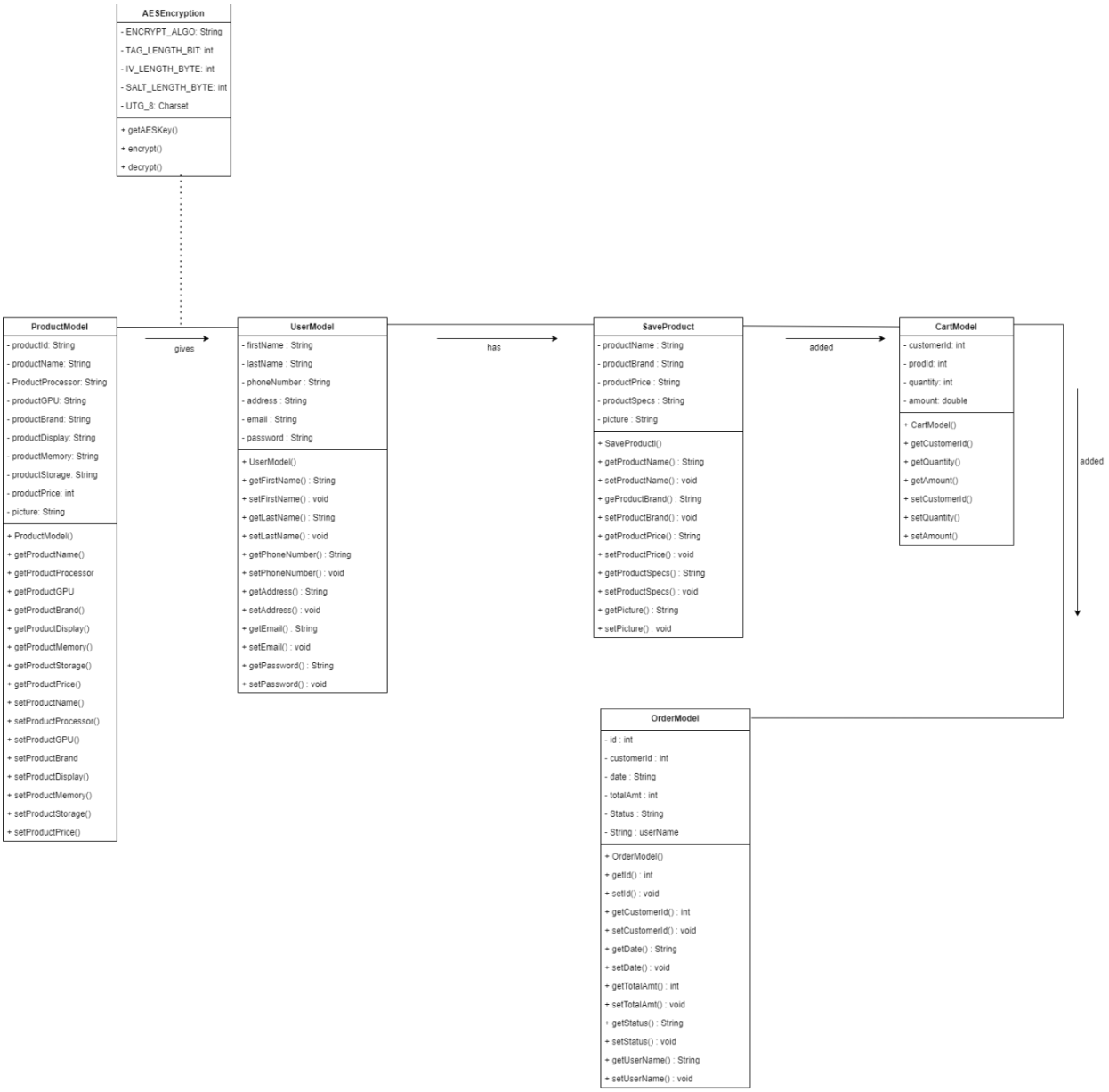

Figure 25: Overall class diagram.

3.2 Individual Class Diagram:

i. Individual class diagram of UserModel:

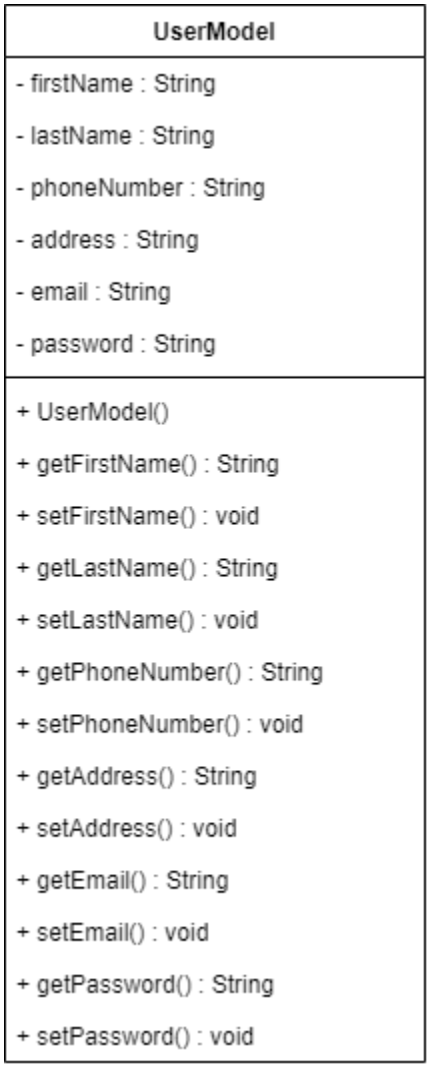

Figure 26: Individual class diagram of UserModel

ii. Individual Class diagram of SaveProduct:

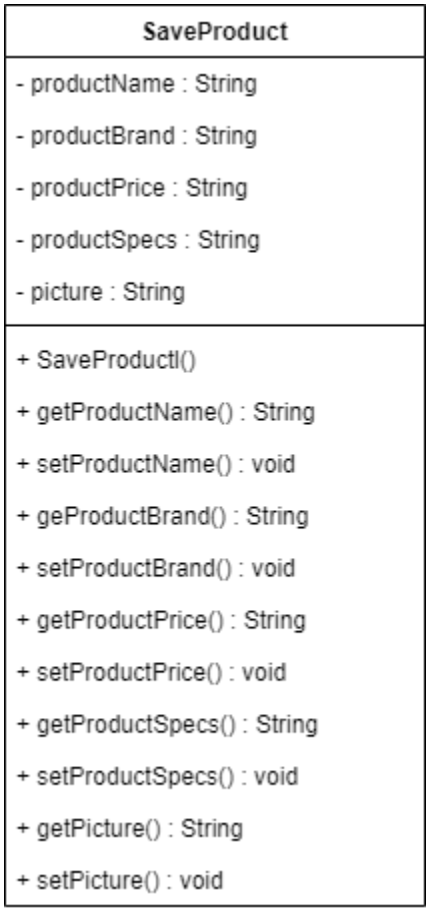

Figure 27: Individual class diagram of SaveProduct.

iii. Individual Class diagram of ProductModel:

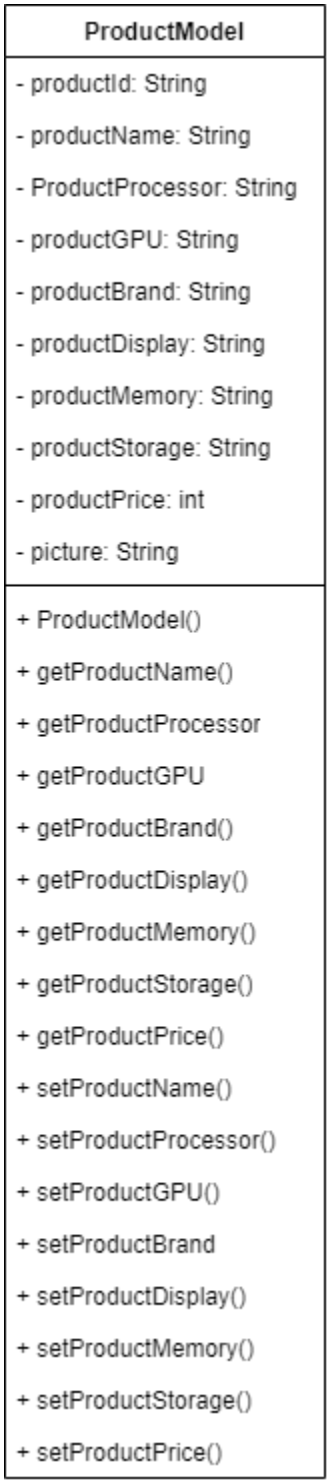

Figure 28: Individual class diagram of ProductModel

iv. Individual class diagram of OrderModel:

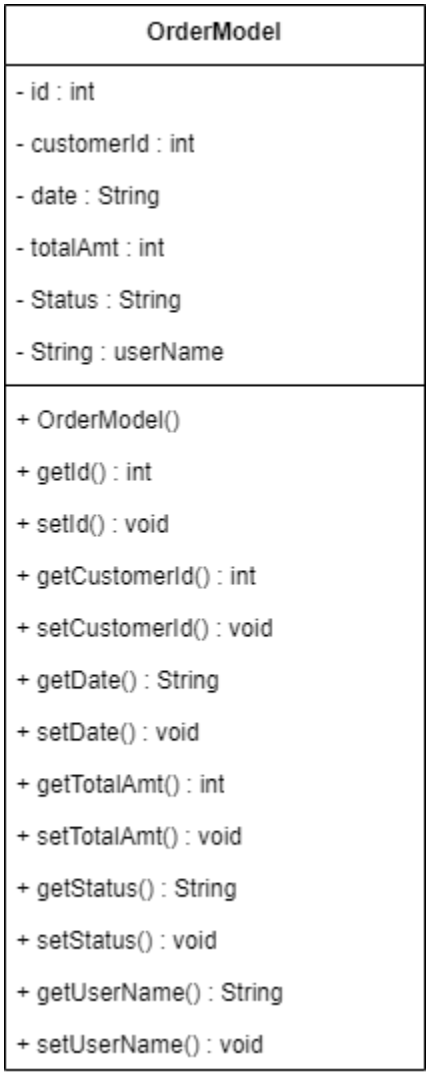

Figure 29: Individual class diagram of OrderModel

v. Individual class diagram of Cart Model:

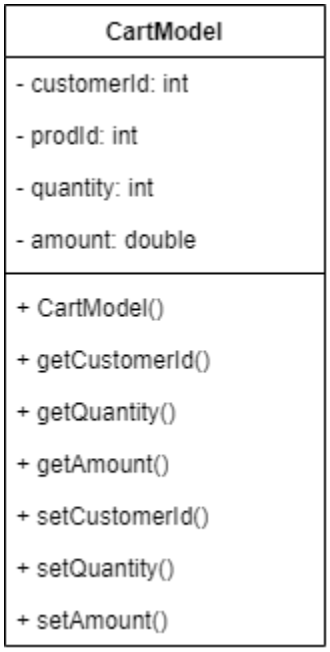

Figure 30: Individual class diagram of Cart Model

vi. Individual class diagram of AES Encryption:

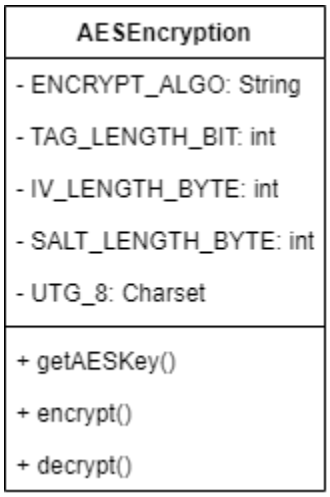

Figure 31: Individual class diagram of AES Encryption

4. Method Description:

Methods are fundamental building blocks in coding that enable programmers to create modular and reusable code. They are designed to perform specific tasks and can be used multiple times throughout your code. This encourages not only a cleaner, more organized codebase but also an efficient way to handle repetitive tasks without rewriting the same code. (Zenva, 2023)

The methods with their description that are used in the development of our site Nexus Nimbus are as follows:

i. Methods of Login:

Table 1: Methods of login.

Method name	Method Description
doPost	Retrieves email and password parameters from the request, validates them against the database from database controller. In success, it redirects user to admin dashboard or homepage according to role. If login fails it will send error message
isAdmin	Checks if user is privileged to use admin or not and according to database returns true or false.
getUserLoginInfo	Queries from the database to validate user credentials and returns value according to validation

ii. Methods of Register page:

Table 2: Methods of register.

Method name	Method Description
doPost	Stores value retrieved from database in UserModel and attempts to add data from DatabaseController and depending on the result returns result.
addUser	Attempts to add new user to database using the provided instance and returns value accordingly

iii. Methods of addToCart:

Table 3: Methods of add to cart.

Method name	Method Description
doPost	Checks if user is logged in. Retrieves value from request parameter then gives to add cart in database. Upon success redirects to admin page.
addToCart	Retrieves the value then adds the record into the cart and returns value according to the message from query set.

iv. Methods of add to order servlet:

Table 4: Methods of add to order.

Method name	Method Description
Add to order servlet	Checks session and if there is session retrieves value from request parameter. It then invokes addOrder and according to result redirects to page.
addOrder	Retrieves value from parameter and runs query to insert order in database then return value accordingly.

v. Methods of change order status:

Table 5: Methods of change order status

Method name	Method Description
doPost	Retrieves value from request parameter and invokes changeOrderStatus and according to result redirects to following pages.
changeOrderStatus	Retrieves value from parameter and runs query to change order status in database then return value accordingly.

vi. Methods of display product page:

Table 6: Methods of display product.

Method name	Method Description
doGet	Invokes getProducts by making list of ProductModel and sends value from the above.
getProducts	Extract values from the database using the queries and adds the value from the database to list ProductModel then returns the list.

vii. Methods of delete product:

Table 7: Methods of delete product

Method name	Method Description
doPost	Retrieves value from request parameter and checks if there is value or not and invokes deleteProduct and according.
deleteProduct	Extract values from the database using the queries and deletes value based on the value received.

viii. Methods of add product:

Table 8: Methods of add product.

Method name	Method Description
dpPost	Retrieves data from request parameter and invokes product model which is then send to addProduct as parameter and a value is returned and according to value redirects to following page.
addProduct	Retrieves the value and inserts the value using queries in database. Returns value according to result.

ix. Methods of show Cart:

Table 9: Methods of show cart.

Method name	Method Description
doGet	Request session value and invokes cartProduct where List ProductModel is requested and returns value of ProductModel.
cartProducts	Retrieves value from parameters and queries cart database accordingly and adds correct value in list and returns.

x. Methods of Update Profile:

Table 10: Methods of Update profile

Method name	Method Description
doPost	Retrieves data from the requested parameter which is used to construct a 'UserModel' object and calls the 'updateUser' method of the DatabaseController class to update the user profile in the database. Redirect the user to the edit profile page with a success message.
updateUser	Retrieves the user's email from the session and prepare SQL statement to update the user details. Return value according to the message from query set.

xi. Methods of display Order:

Table 11:Methods of display order.

Method name	Method Description
doGet	Retrieves a list of orders from the database using the 'getOrders' method of the 'DatabaseController' class. Sets the orders as a request attribute with the key "orders" and forwards the request to the order.jsp page
displayOrder	Retrieves a list of orders from the database according to the SQL query, and if any exceptions occur during the database operation, they are caught and the stack trace is printed.

xii. Methods of edit product:

Table 12: Methods of edit profile.

Method name	Method Description
doPost	Retrieves data from the request parameter. Use the data to construct 'ProductModel' object and call the 'editProduct' method of the 'DatabaseController' class to update the product. Forwards the request to the 'disaplayProductServlet'.
editProduct	Update the fields of product based on the provided product name and attributes are used to set the values using prepared statement and return the value according to the message from query set.

xiii. Methods of remove cart:

Table 13: Methods of remove cart

Method name	Method Description
doPost	Retrieves the product ID from the request parameters and calls the deleteCartProduct method of the DatabaseController to remove the corresponding product from the cart in the database. Forwards the request to the 'showCartServlet' to display the updated cart.
deleteCartProduct	Takes the product ID as input. Execute SQL DELETE query and return value according to the message from query set.

xiv. Method of logout:

Table 14: Methods of logout.

Method name	Method Description
doPost	Accesses the session associated with the request and destroys the session. Then, it redirects the user to the home page using the response send redirect.

5. Test cases:

5.1 Testing of registration of user:

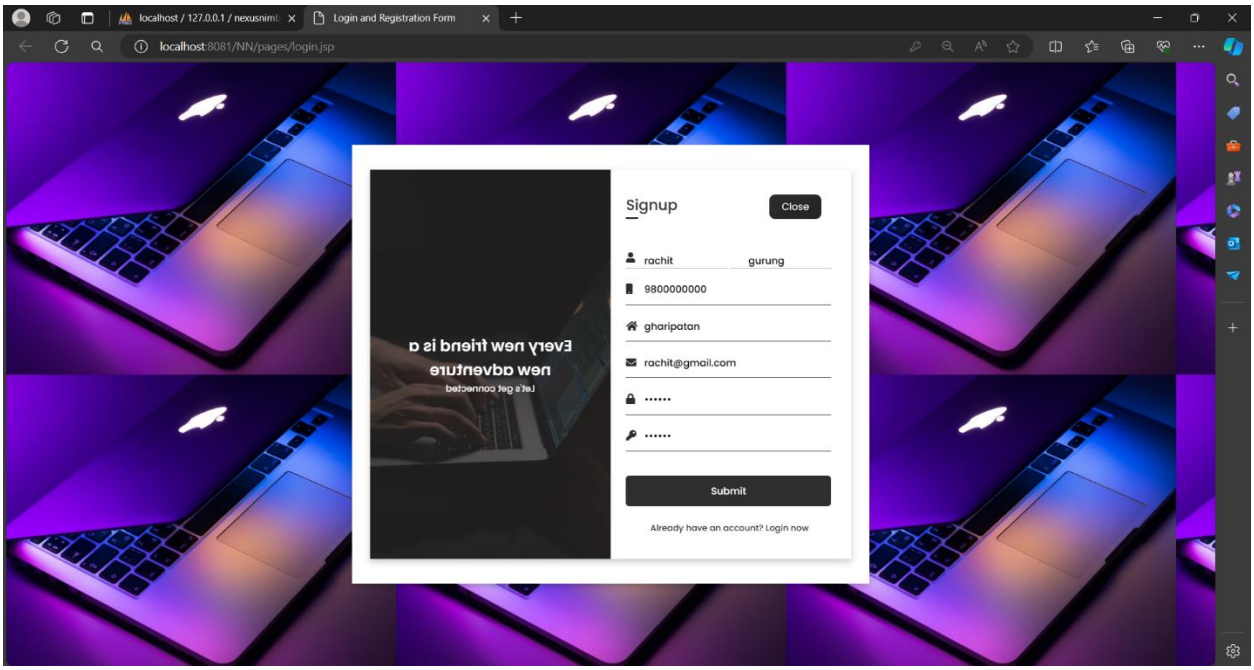

Figure 32: Registration of user in form

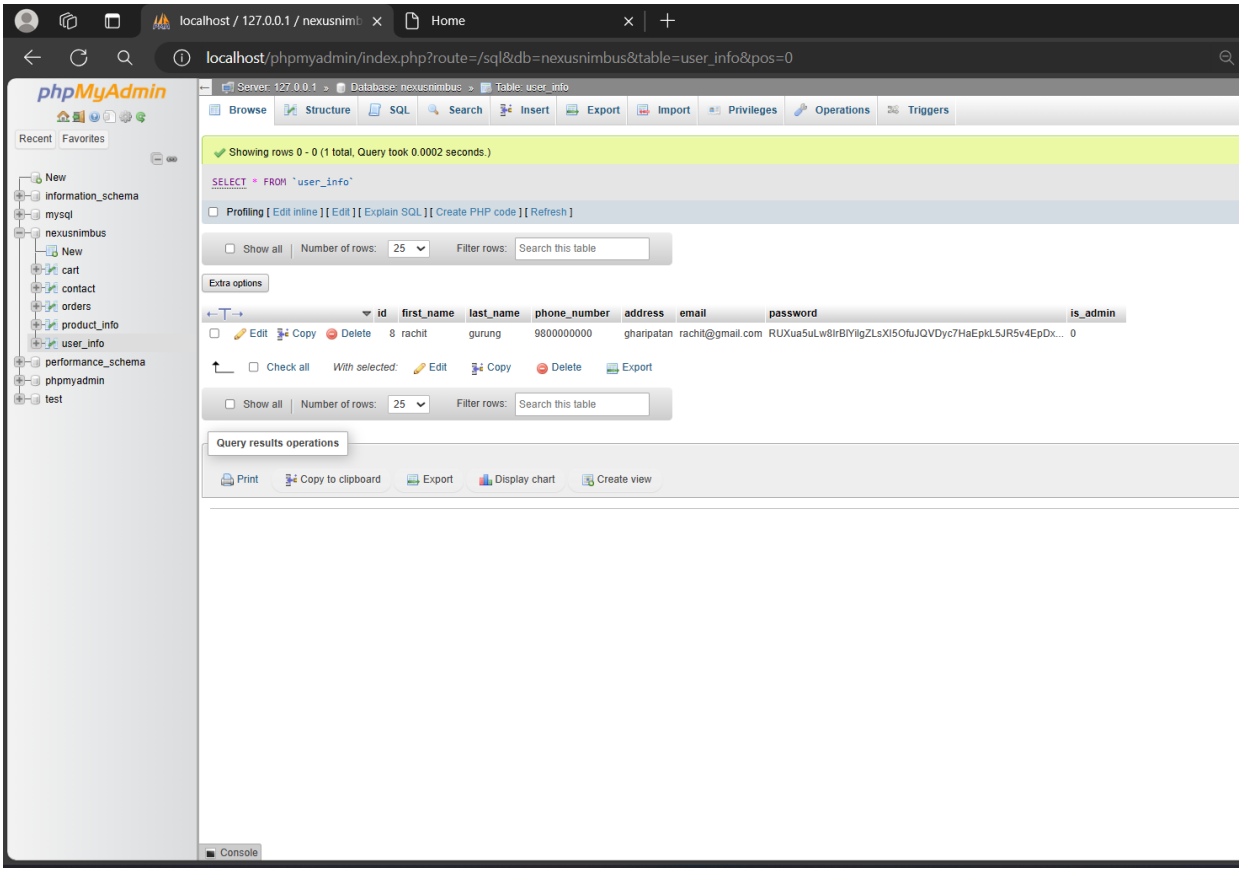

Figure 33: Inserted user data in database.

Table 15: Testing of registration.

Objective	To check whether the user is registered or not.
Action	Fill in the registration form with correct information.
Expected Result	The user data should be inserted in the database.
Actual Outcome	The user data is inserted into the database.
Conclusion	Test successful.

5.2 Testing of invalid user email:

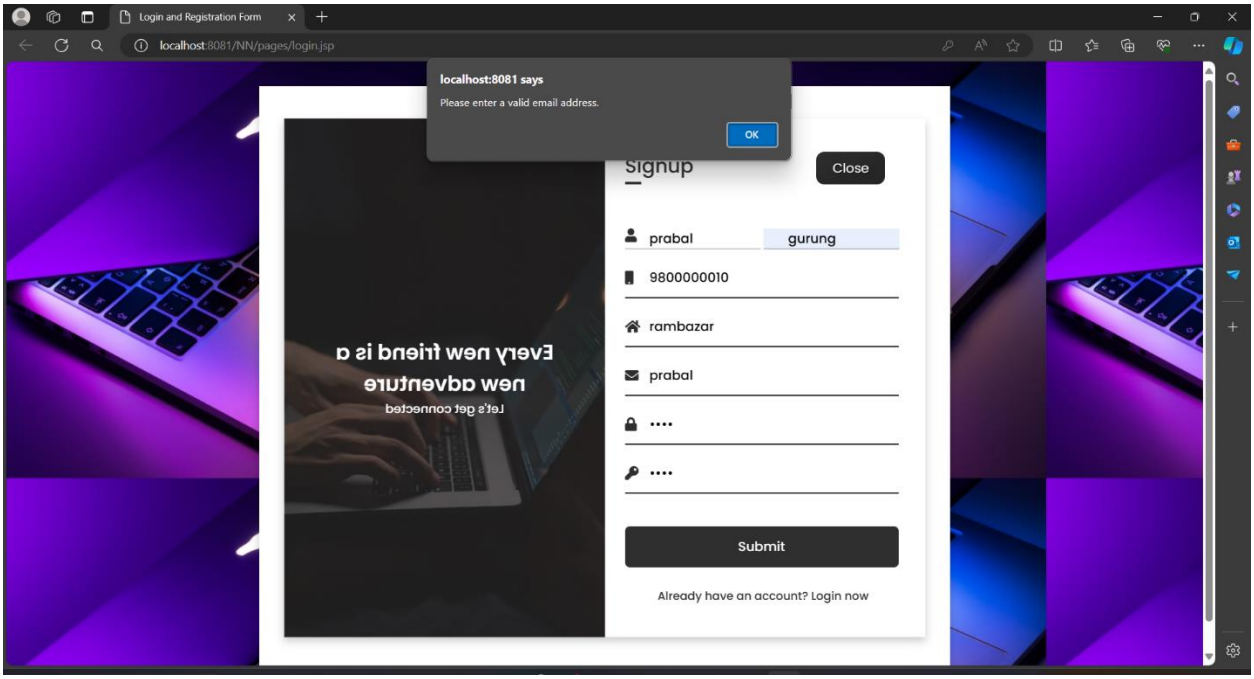

Figure 34: Pop up message for invalid email

Table 16: Testing of invalid user email.

Objective	To check whether appropriate message is shown when invalid email is given.
Action	Fill in the registration form and input incorrect email.
Expected Result	A pop-up message should be shown stating incorrect email.
Actual Outcome	A pop-up message is shown stating incorrect email.
Conclusion	Test successful.

5.3 Testing for already registered email:

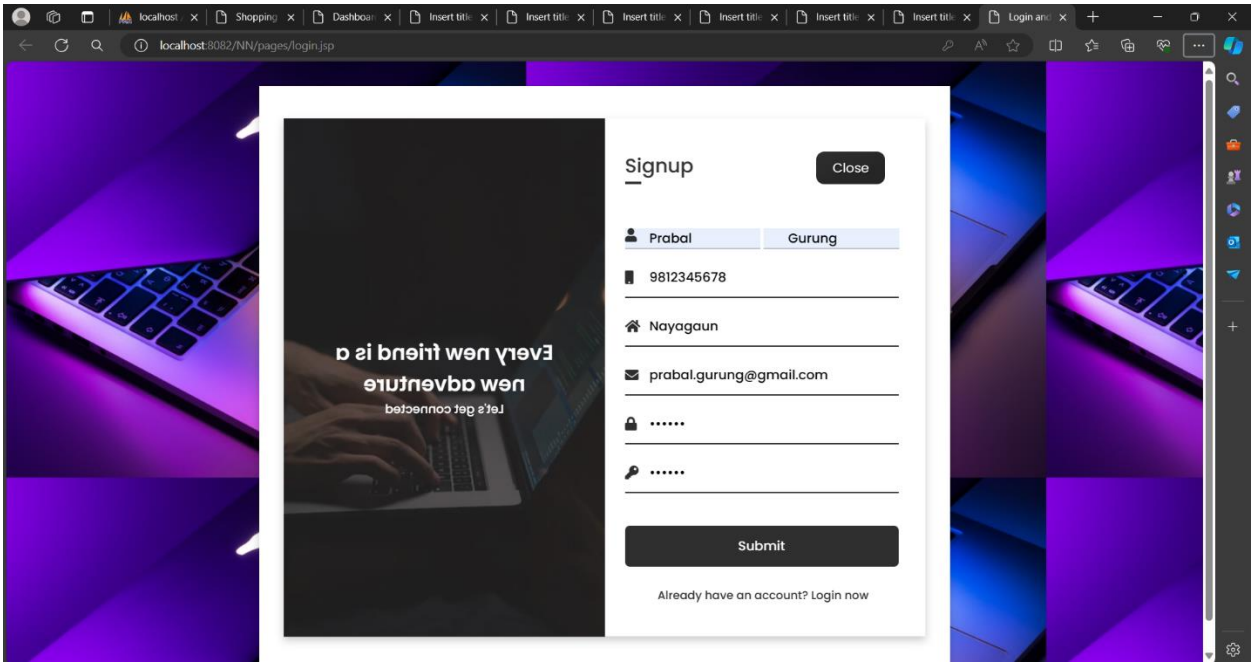

Figure 35: Giving already registered email.

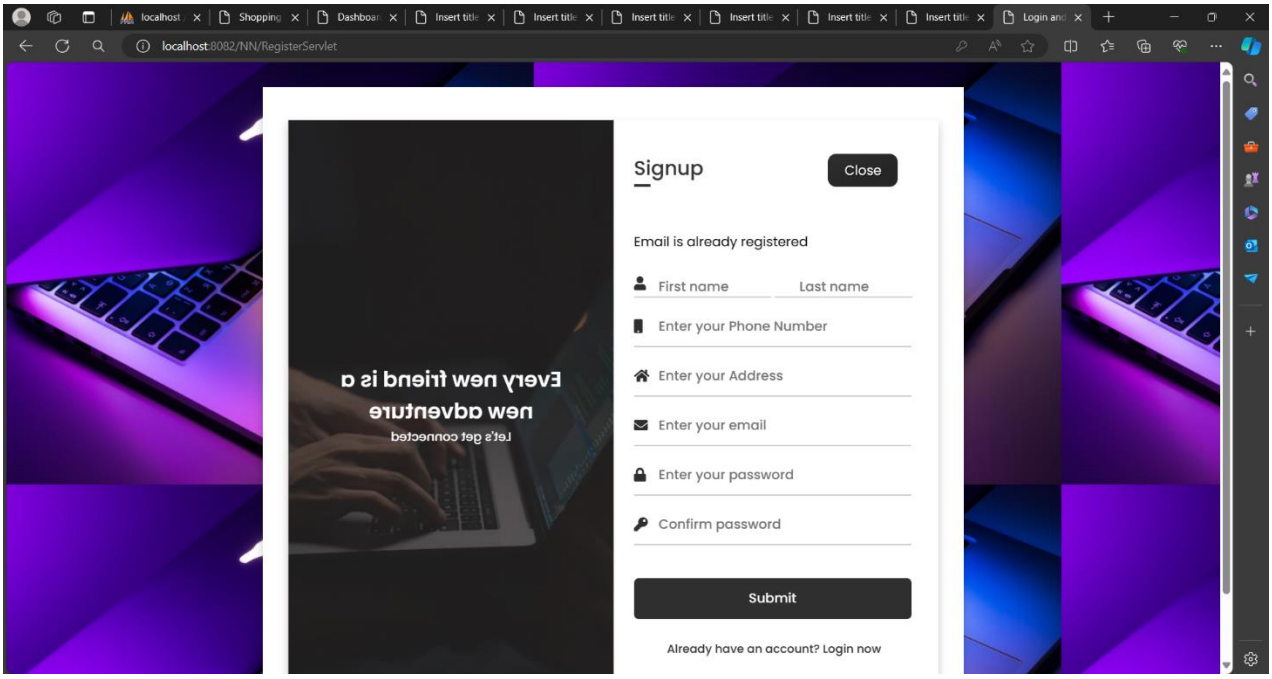

Figure 36: Error message for already registered email

Table 17: Testing for already registered email.

Objective	To check whether appropriate message is shown or not when an already registered email is given.
Action	Fill in the registration form and input an already registered email.
Expected Result	An appropriate message should be shown.
Actual Outcome	An appropriate message is shown stating an already registered email.
Conclusion	Test successful.

5.4 Testing for login of user:

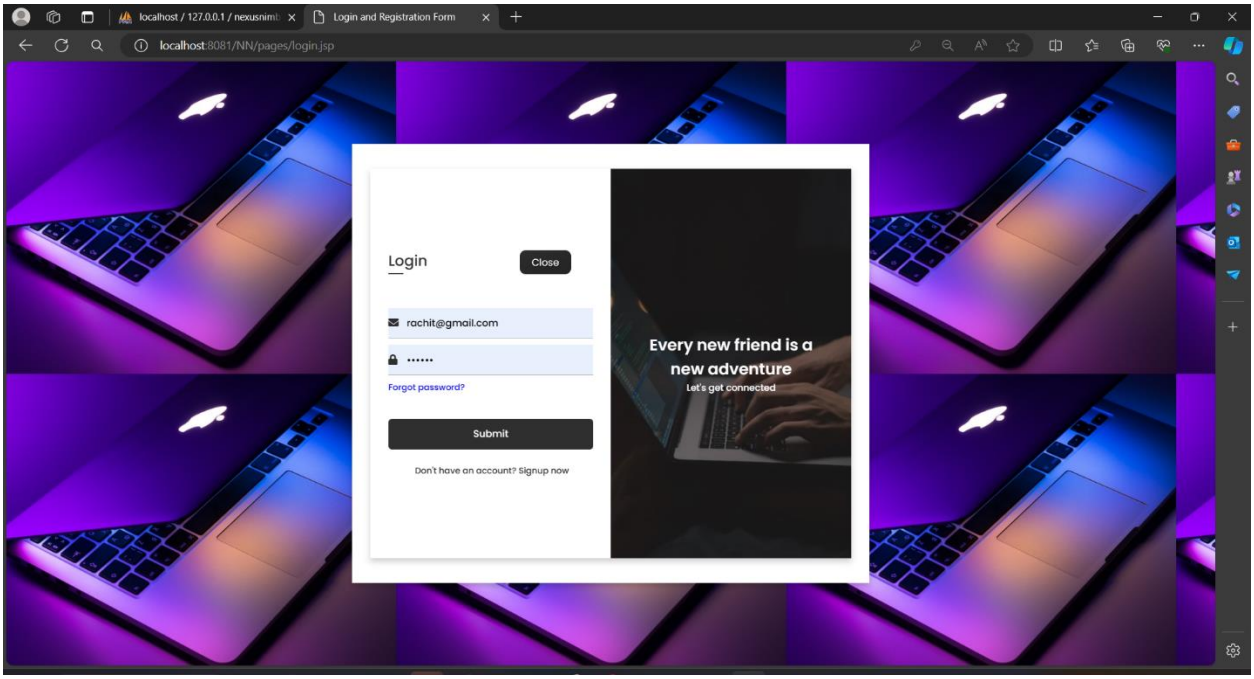

Figure 37: Giving login credentials.

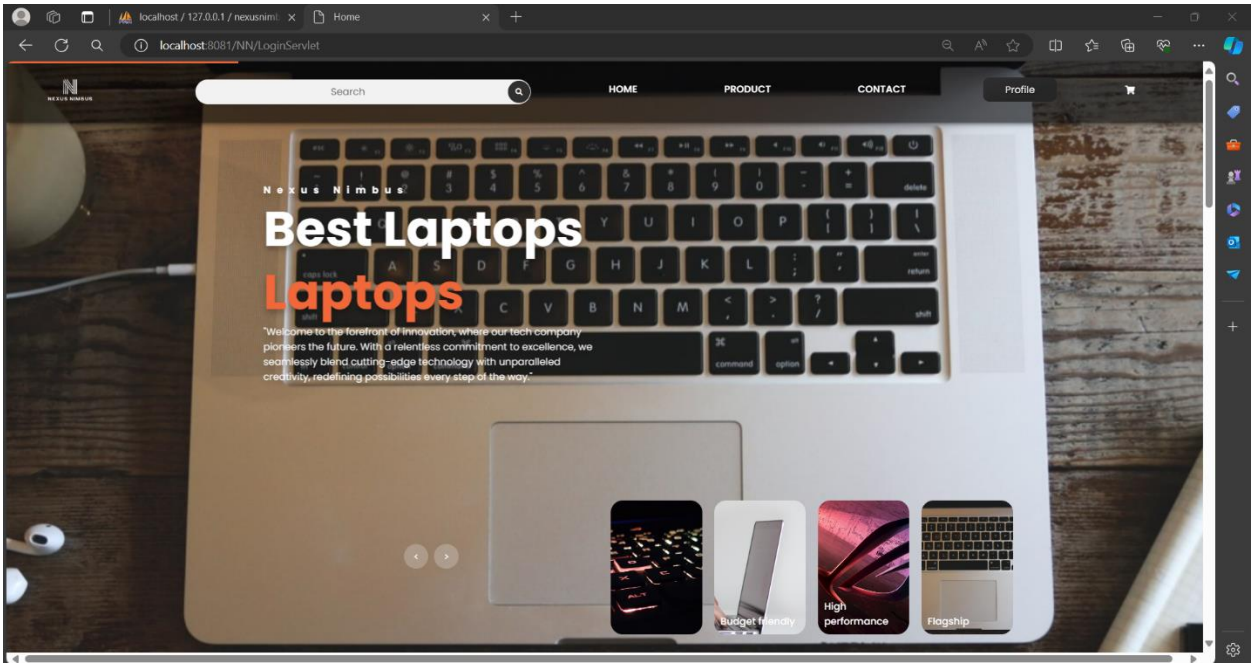

Figure 38: Successful login

Table 18: Testing for login of user

Objective	To check whether the registered user can login or not.
Action	Fill in the email and password of login page.
Expected Result	The user should be logged in and should have profile button in the navbar.
Actual Outcome	The user is logged in and has profile button in the navbar.
Conclusion	Test successful.

5.5 Testing for incorrect password in login page:

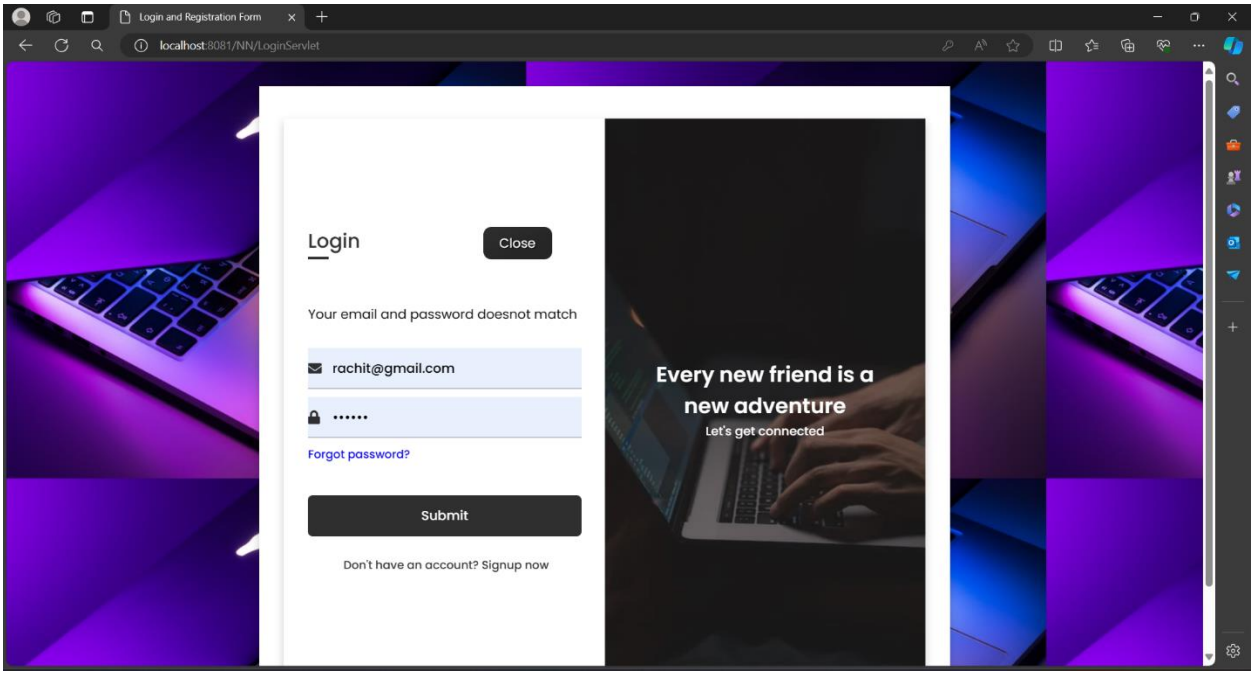

Figure 39: Message for email and password mismatch.

Table 19: Testing for incorrect password in login page

Objective	To check whether an appropriate message is shown or not when the password and email don't match during login.
Action	Fill incorrect input in the login form.
Expected Result	An appropriate message should be shown.
Actual Outcome	An appropriate message is shown stating incorrect email and password.
Conclusion	Test successful.

5.6 Testing for password encryption:

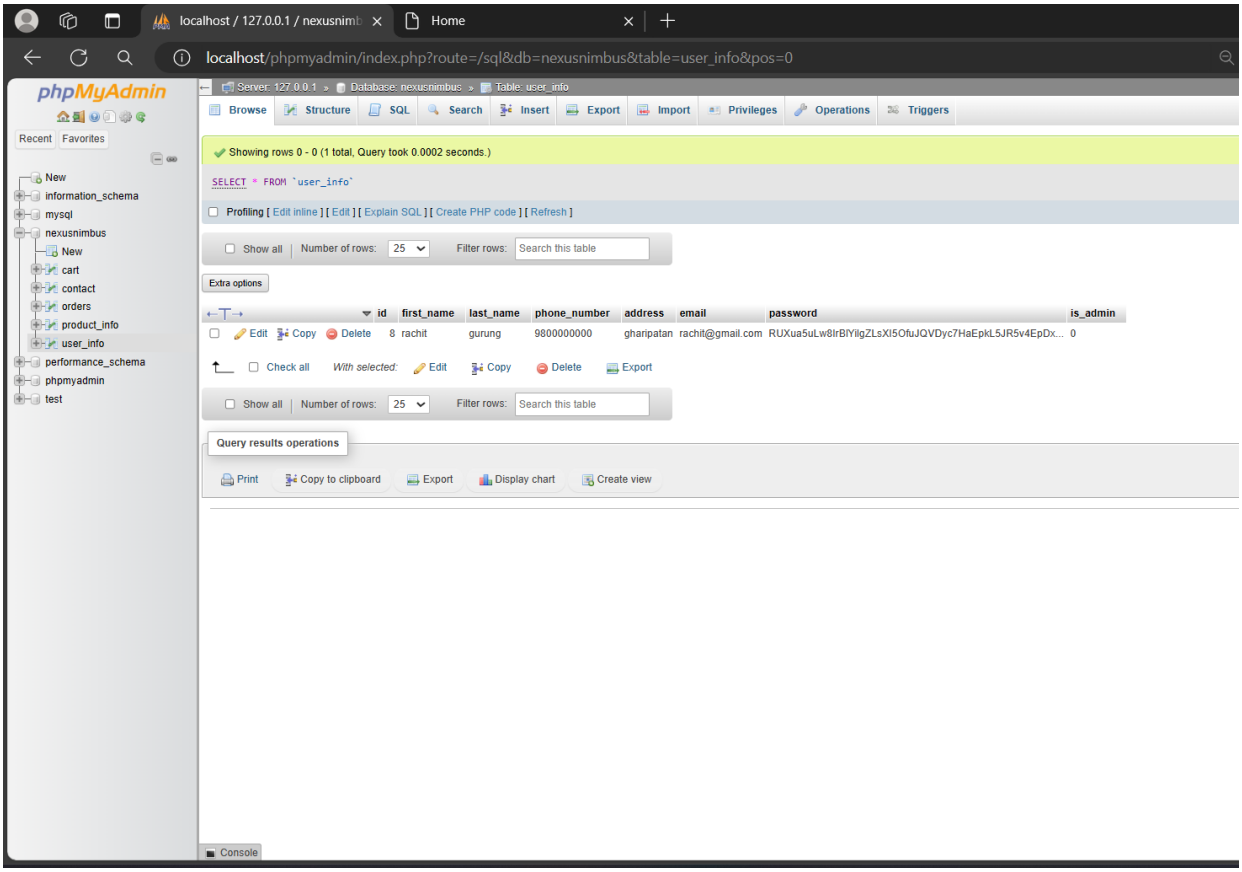

Figure 40: Encryption of password

Table 20: Testing for password encryption

Objective	To check whether the password is encrypted or not during registration.
Action	Fill in the registration form.
Expected Result	The user password should be encrypted.
Actual Outcome	The user password is encrypted.
Conclusion	Test successful.

5.7 Testing for product page:

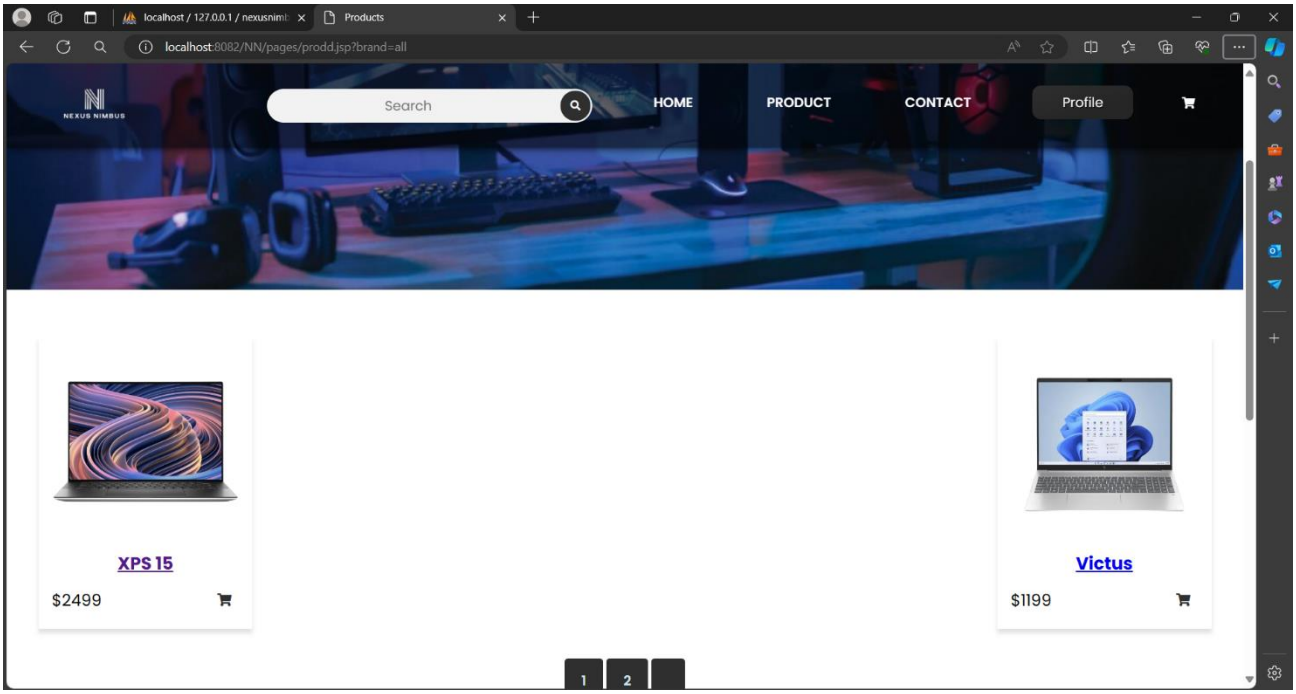

Figure 41: Product page

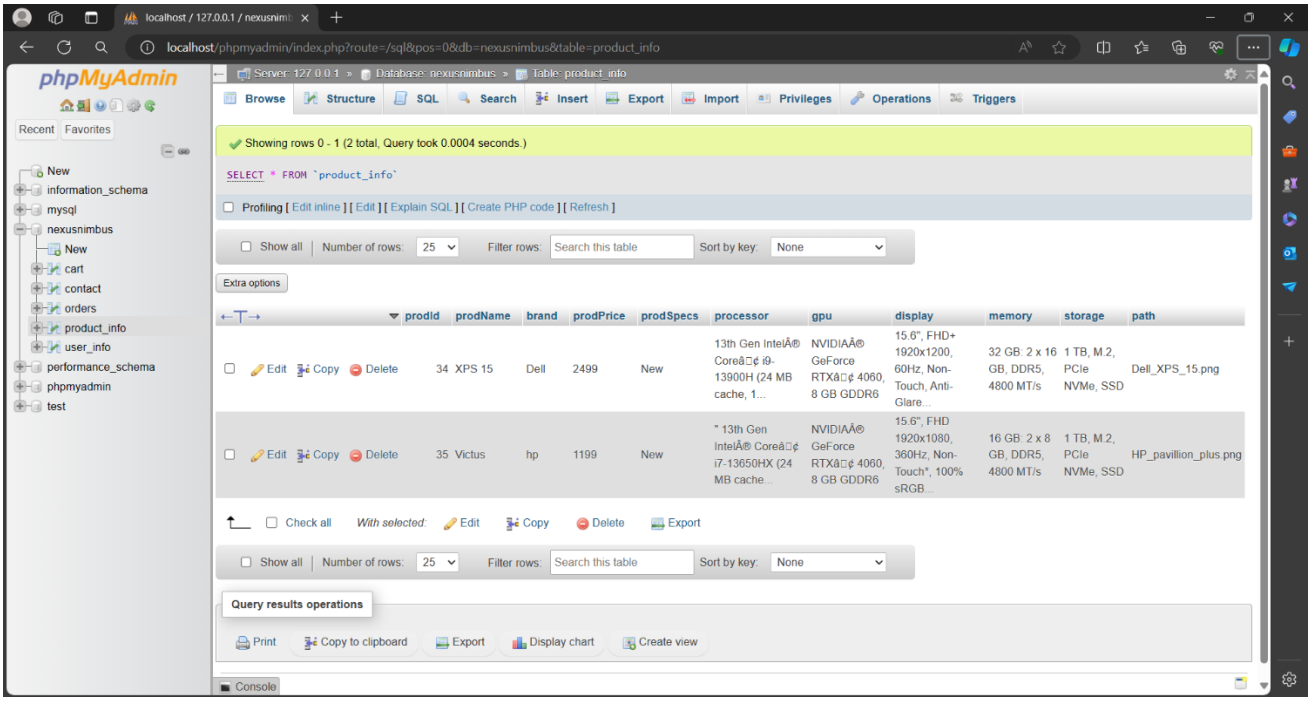

Figure 42: Product Database.

Table 21: Testing for product page

Objective	To check whether the information for product page is extracted from the product database or not.
Action	Click on the product section in the navbar.
Expected Result	The product detail in the product page and database should match.
Actual Outcome	The product detail in the product page and database matches.
Conclusion	Test successful.

5.8 Testing for product detail page:

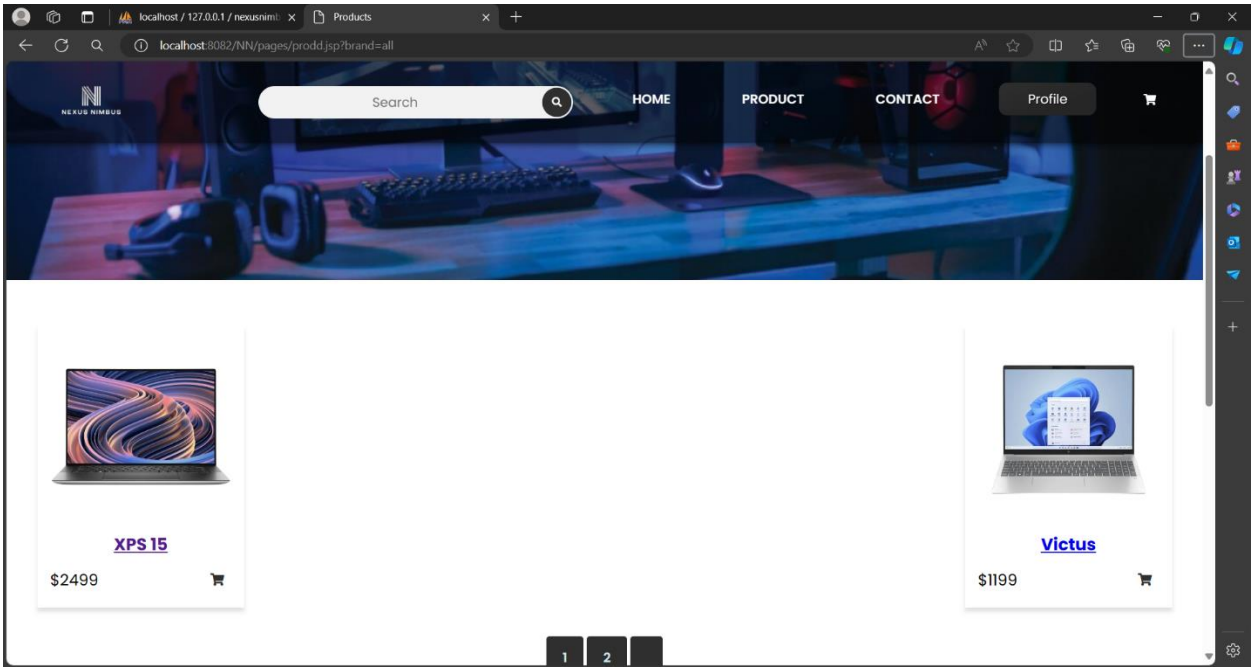

Figure 43: Product page

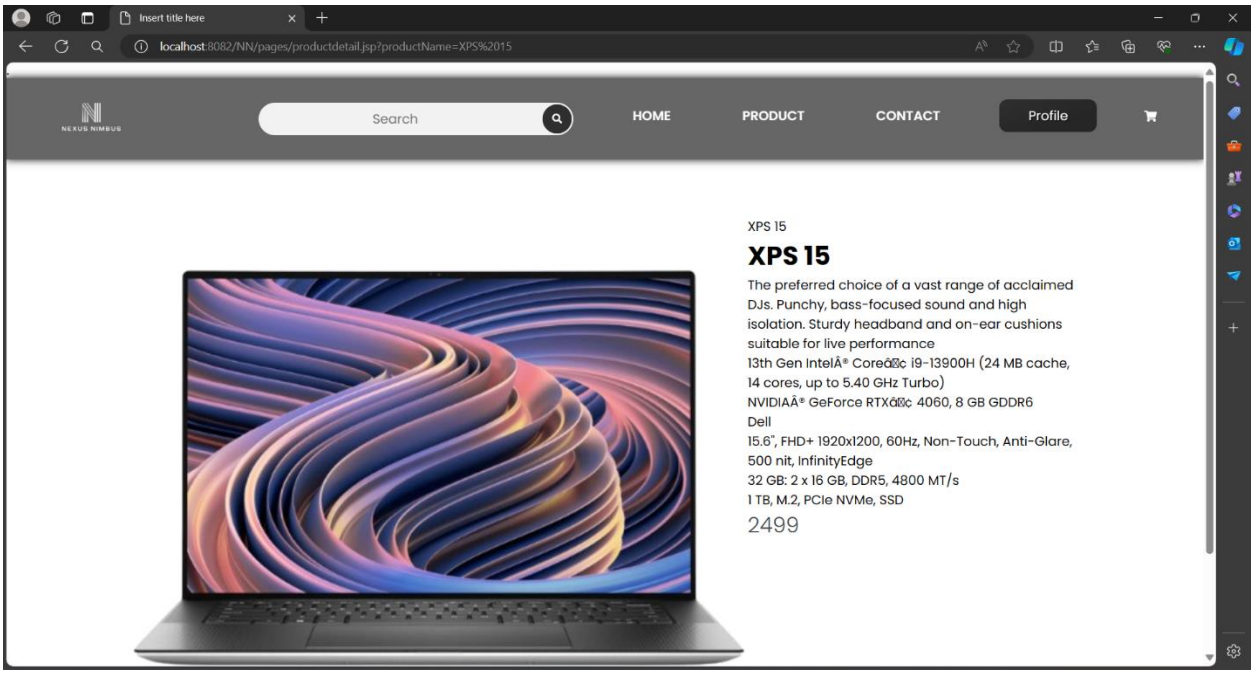

Figure 44: Product detail page

Table 22: Testing for product detail page

Objective	To check whether the product detail page matches with a specific product when clicked on.
Action	Click on a product in the product page.
Expected Result	The product detail page for the specific product should be opened.
Actual Outcome	The product detail page for the specific product is opened.
Conclusion	Test successful.

5.9 Testing for profile page:

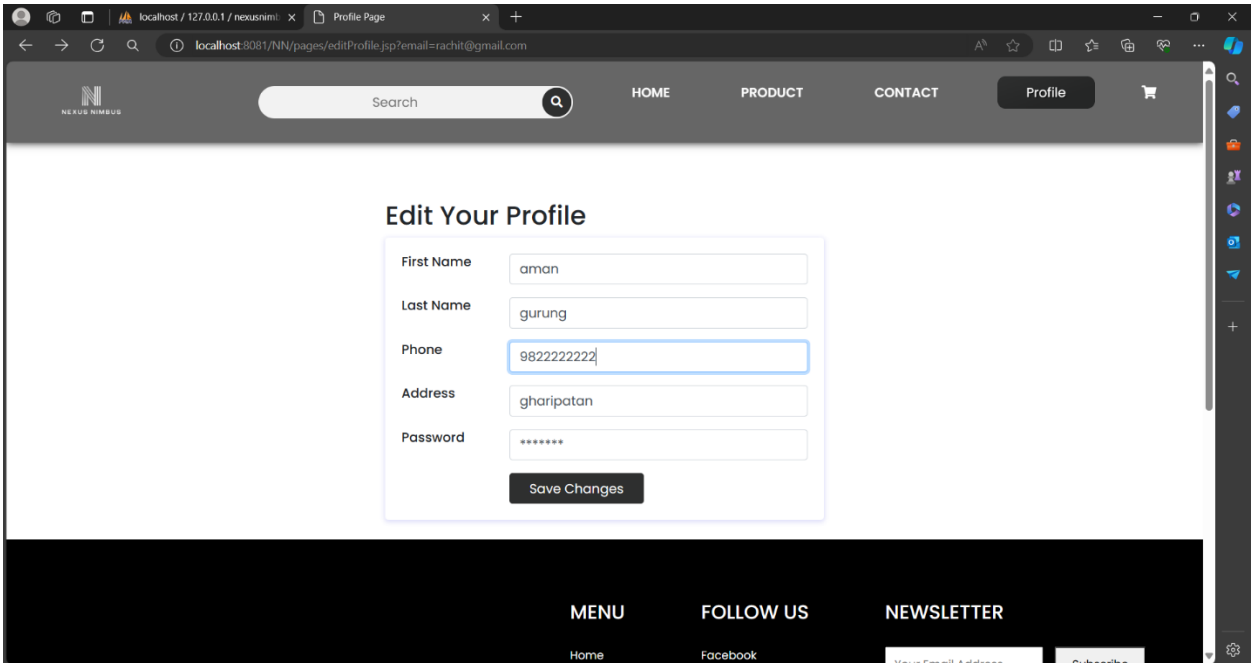

Figure 45: Editing user profile.

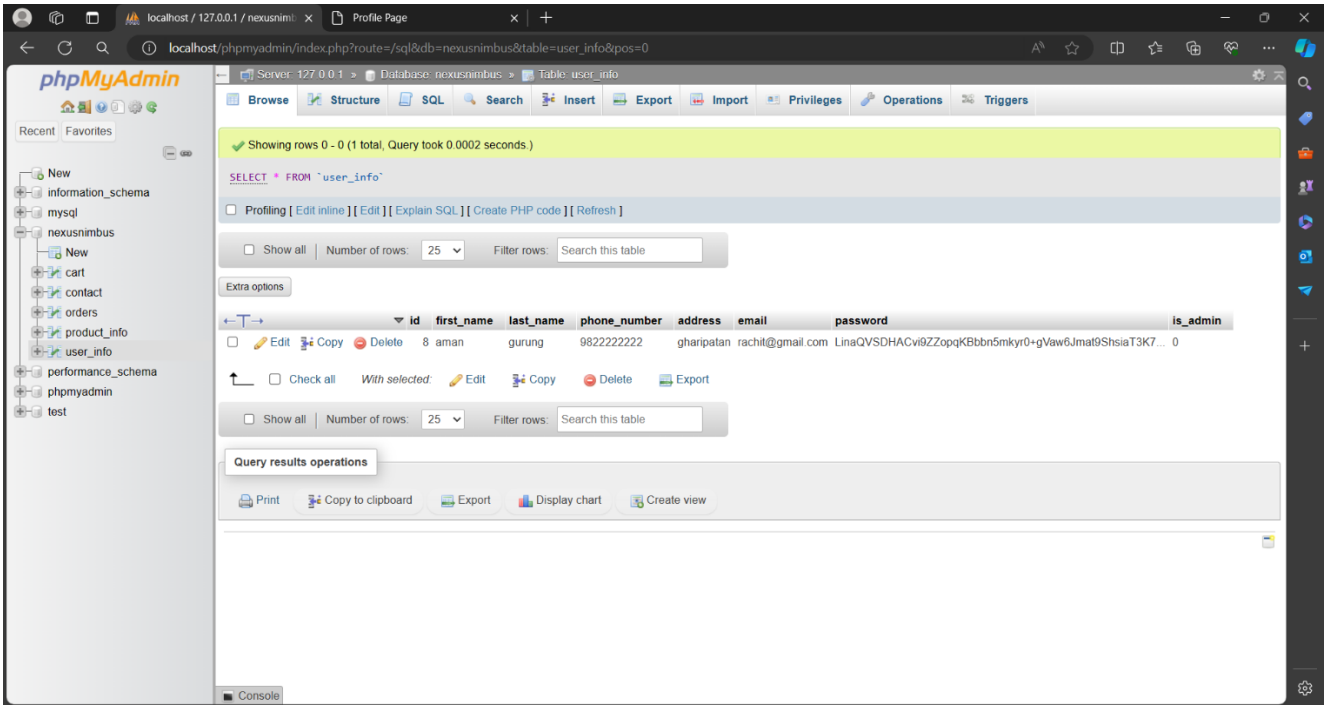

Figure 46: User profile has been updated.

Table 23: Testing for profile page

Objective	To check whether the profile data of the specific user is updated or not.
Action	Open the profile page and insert new data and click on save changes.
Expected Result	The personal data of the user should be updated.
Actual Outcome	The personal data of the user is updated.
Conclusion	Test successful.

5.10 Testing for profile page without log in:

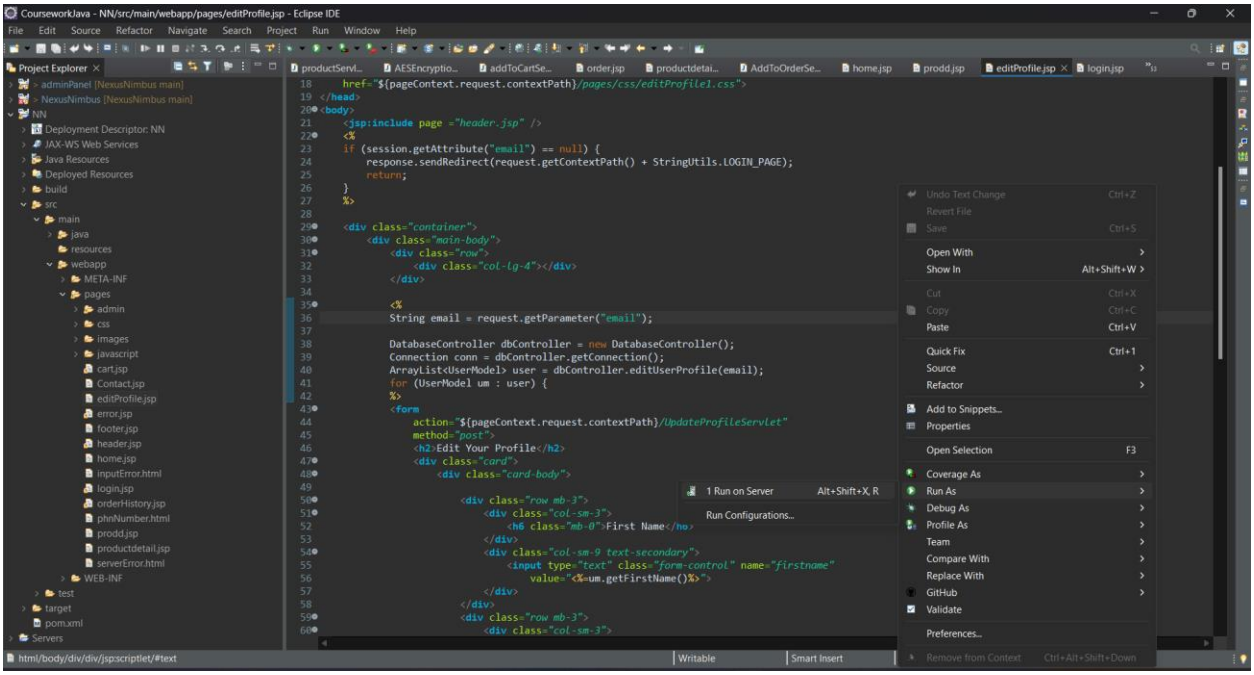

Figure 47: Running profile page directly without login

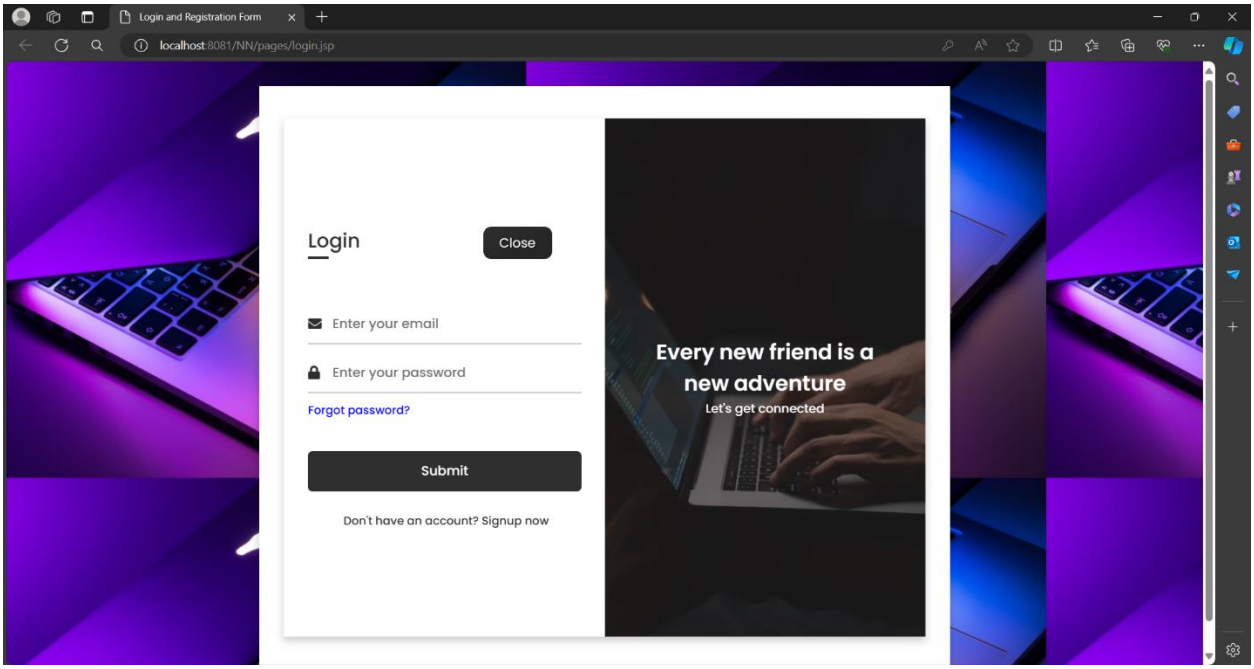

Figure 48: Redirection to login page

Table 24: Testing for profile page without log in

Objective	To check whether the profile page can be accessed without logging in.
Action	Run the profile page directly form the IDE.
Expected Result	The page should be redirected to login page.
Actual Outcome	The page is redirected to login page
Conclusion	Test successful.

5.11 Testing for session destruction when logged out:

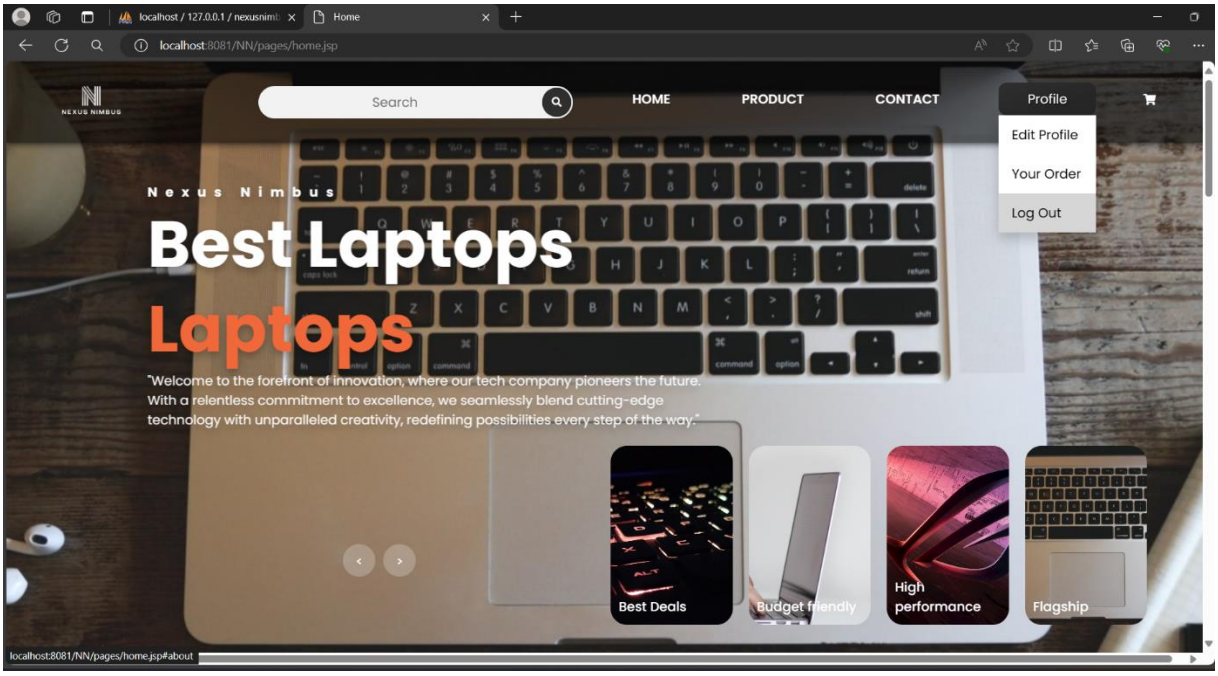

Figure 49: Logging out by user

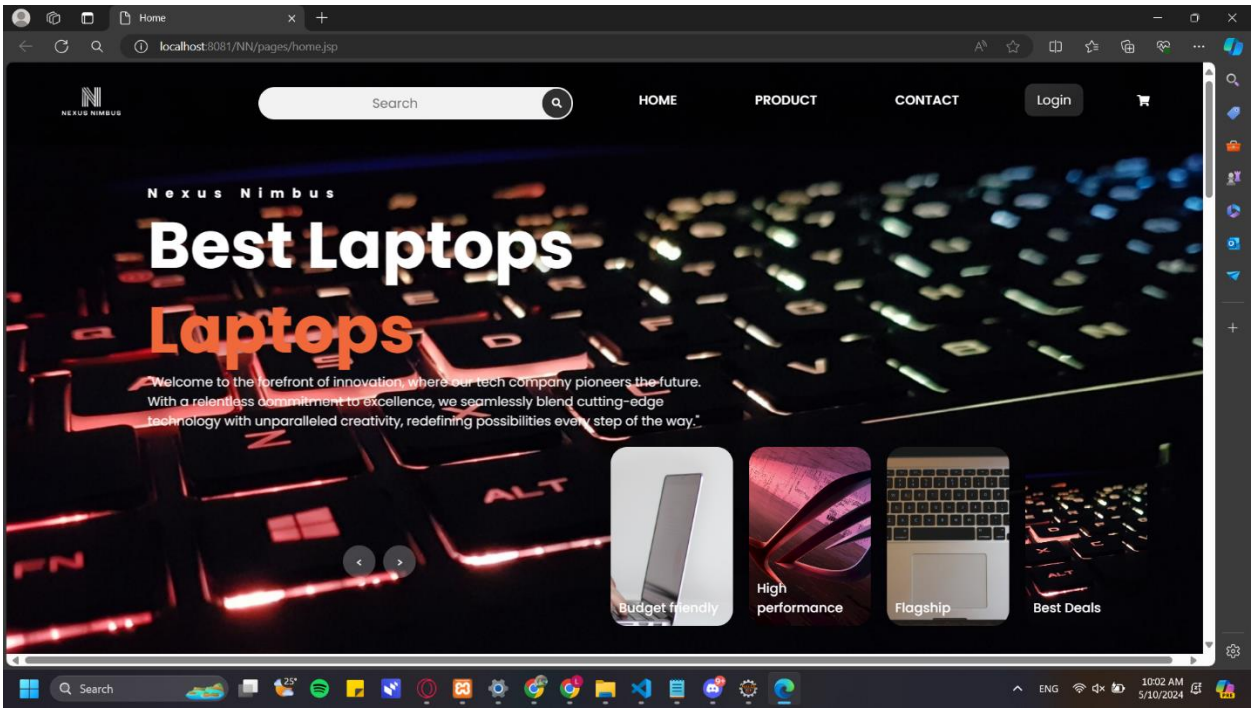

Figure 50: Logout successful.

Table 25: Testing for session destruction when logged out.

Objective	To check whether the session is destroyed or not when logged out.
Action	Click on the logout button in the navbar of user panel.
Expected Result	The session should be destroyed and redirected to home page with login button.
Actual Outcome	The session is destroyed and redirected to home page with login button.
Conclusion	Test successful.

5.12 Testing for add to cart:

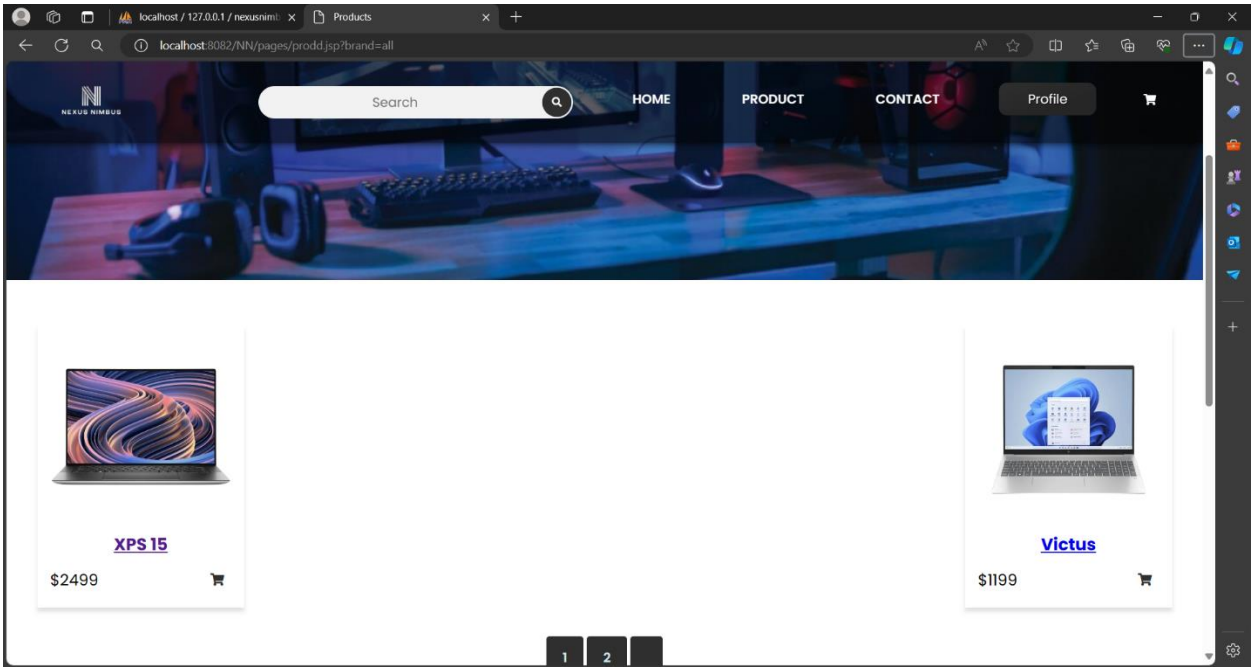

Figure 51: Adding product to cart

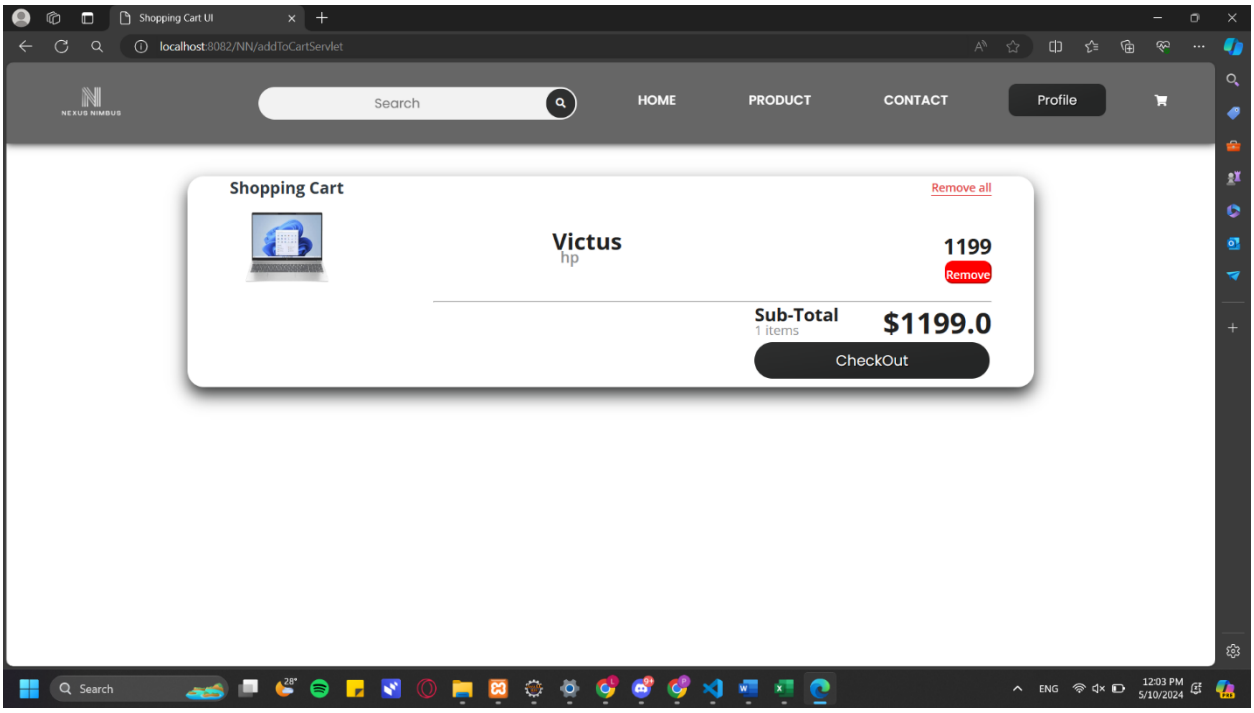

Figure 52: Product added to cart successfully.

Table 26: Testing for add to cart

Objective	To check whether the add to cart feature works or not.
Action	Click on the cart button in the product.
Expected Result	The product should be added to cart.
Actual Outcome	The product is added to cart.
Conclusion	Test successful.

5.13 Testing for check out of cart:

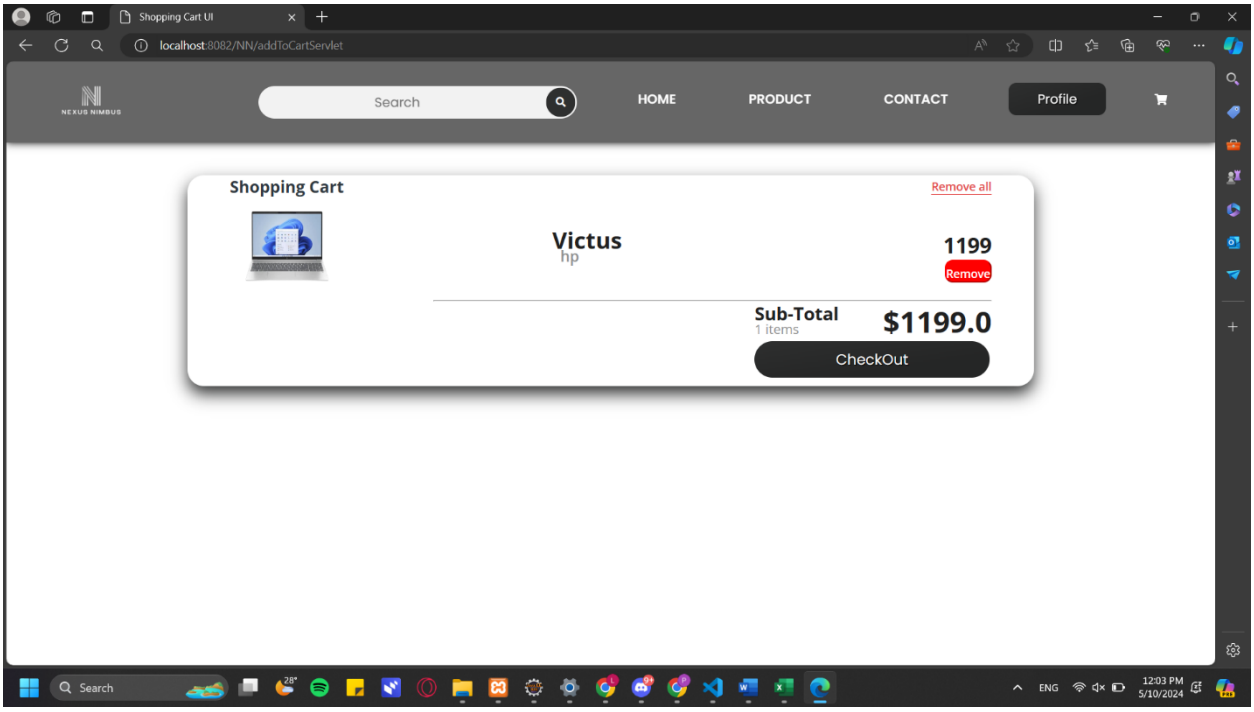

Figure 53: Check out of cart

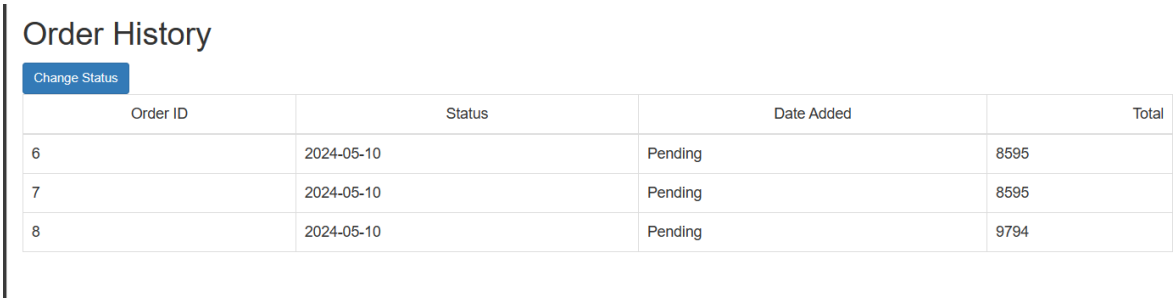

Figure 54: Cart item added to order

Table 27: Testing for check out of cart

Objective	To check whether the checkout feature works or not.
Action	Click on the checkout button in the cart page.
Expected Result	The product in cart should be added to order history.
Actual Outcome	The product in cart is added to order history.
Conclusion	Test successful.

5.14 Testing for add product by admin:

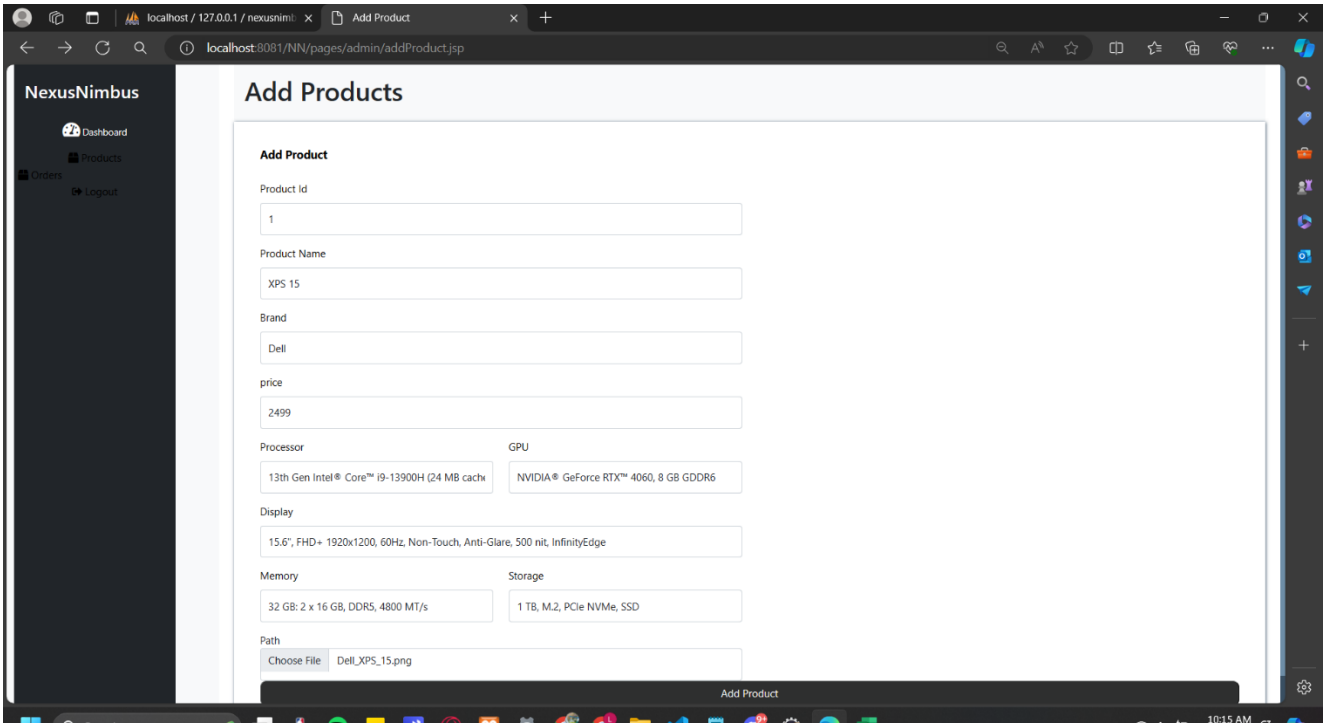

Figure 55: Add product by admin

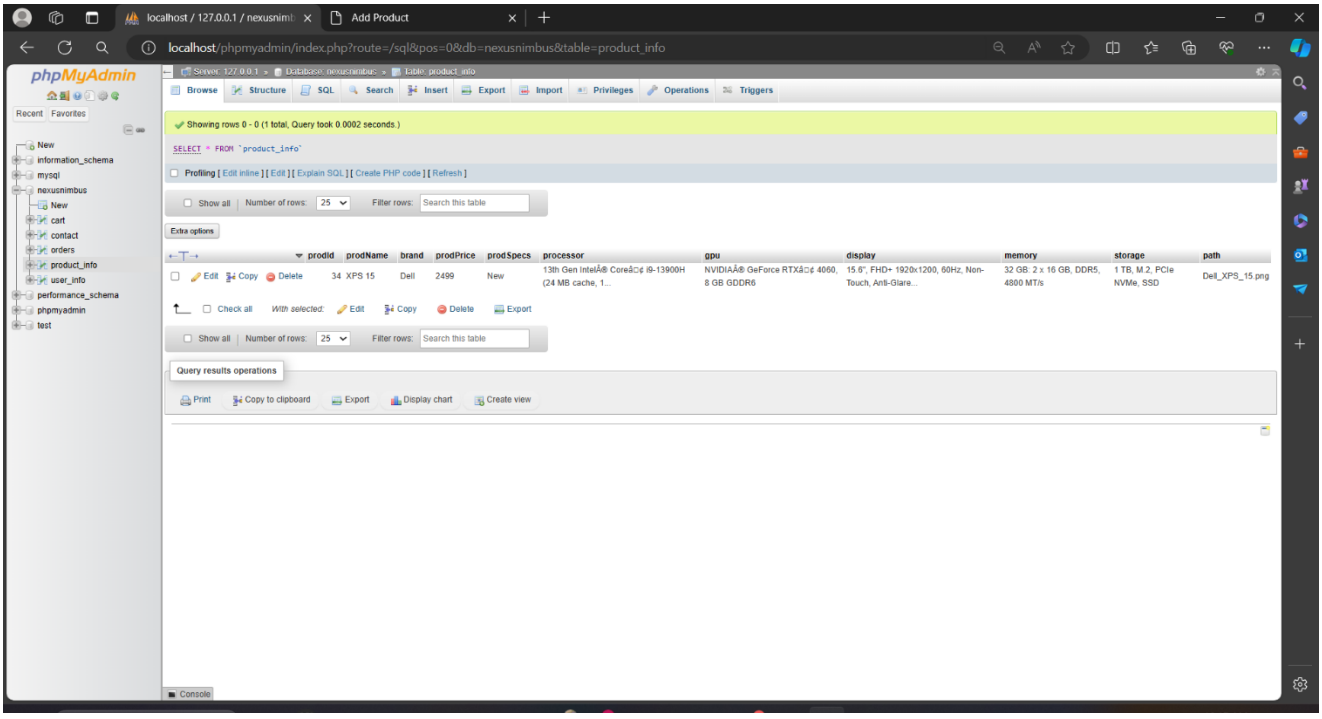

Figure 56: Product added successfully.

Table 28: Testing for add product by admin

Objective	To check whether the admin's add product functionalities work or not.
Action	Login as admin and click on the add product and fill the input.
Expected Result	The product should be added in the database.
Actual Outcome	The product is added in the database.
Conclusion	Test successful.

5.15 Testing for deletion of product:

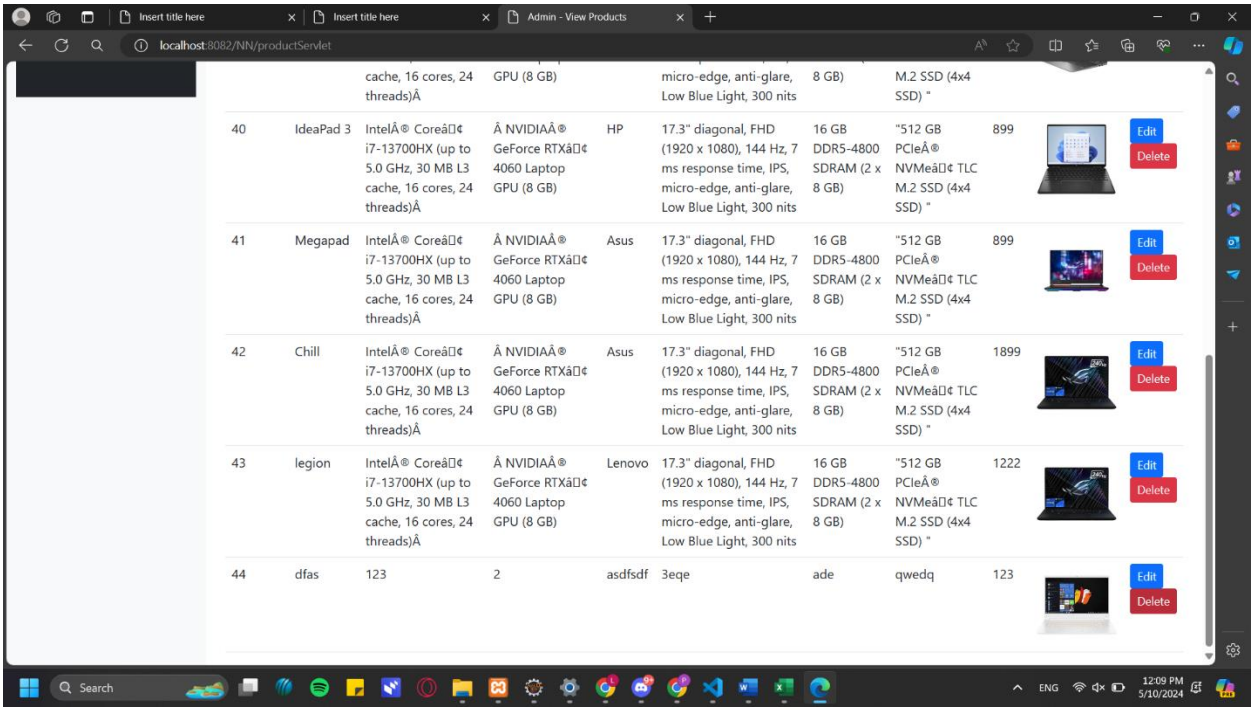

Figure 57: Deletion of 44 id product

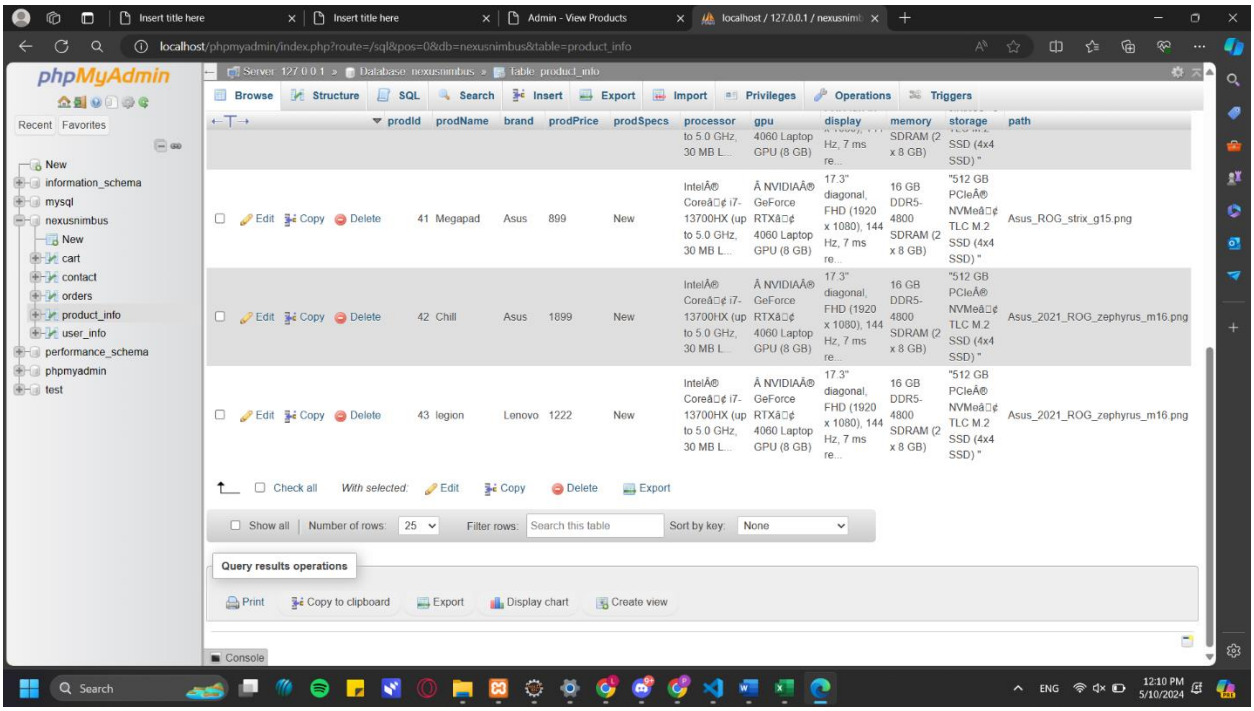

Figure 58: Product database without 44 id product

Table 29: Testing for deletion of product

Objective	To check whether the product delete button works or not.
Action	Login as admin and click on the delete product.
Expected Result	The product should be deleted from the database.
Actual Outcome	The product is deleted from the database.
Conclusion	Test successful.

6. Tools and libraries used:

The tools used in the development of our website Nexus Nimbus are as follows:

i. Eclipse IDE:

Figure 59: Eclipse IDE

Eclipse is the most widely used Integrated Development Environment (IDE) for building Java applications. IBM created the Eclipse Project to implement a Java-based IDE that supports development of embedded Java applications. (Minh, 2020)

Reason for using:

- Versatile and powerful IDE.
- Provides complete functionality to select the name of workspace and manage projects in single workspace.
- Provides many plug-ins and is open source.

ii. XAMPP Control panel:

Figure 60: XAMPP

XAMPP Control Panel is an essential tool. It serves as a graphical user interface (GUI) for managing various components of the XAMPP stack. It is a cross-platform web server solution stack package that is available for free and open source. The XAMPP Control Panel acts as a central hub for managing the services provided by XAMPP. (World, 2024)

Reason for using:

- Open source software.
- Compatible in both windows and Linux environments.
- Allows user to test their code locally on their own computer.

iii. MySQL:

Figure 61: My SQL

MySQL, open-source relational database management software, owned by the computer software company Oracle, that allows users to interact with large amounts of data across multiple databases. MySQL is one of the most popular database management programs used worldwide. (Gisonna, 2024)

Reason for using:

- Open source and scalability.
- High performance.
- Large community and support.

iv. Apache Tomcat:

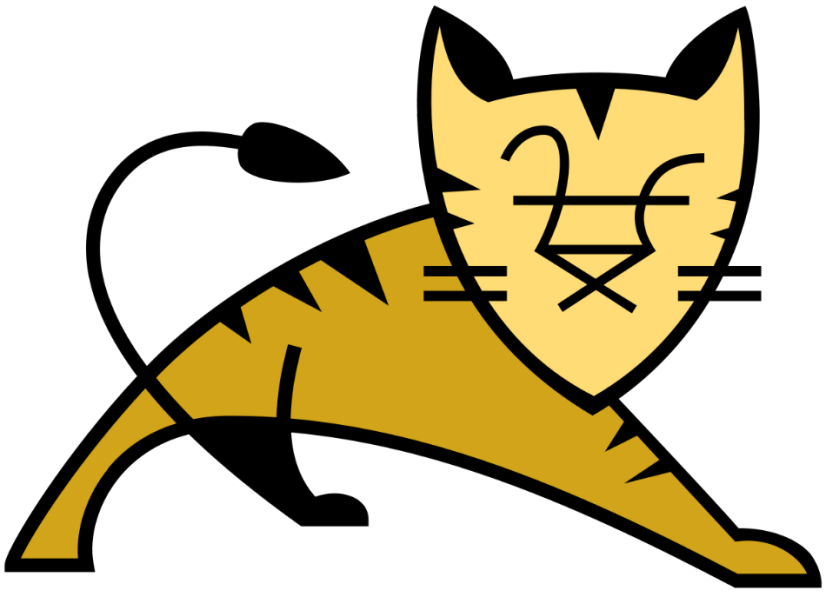

Figure 62: Apache tomcat

Apache Tomcat is a popular open-source Java-based web server that implements several Java EE specifications like Java Servlet, Java Server Pages (JSP), Java EL, and WebSocket enabling Java developers to build dynamic web applications and serve static content over the internet or on private networks. (Singh, 2023)

Reason for using:

- Lightweight and simple to set up.
- Compatible with different operating systems.
- Developers can efficiently build and deploy robust web applications.

v. Balsamiq:

Figure 63: Balsamiq

Balsamiq Wireframes is a user interface design tool for creating wireframes. Balsamiq can use it to generate digital sketches of your idea or concept for an application or website, to facilitate discussion and understanding before any code is written. The completed wireframes can be used for user testing, clarifying your vision, getting feedback from stakeholders, or getting approval to start development. (Balsamiq, n.d.)

Reason for using:

- Able to generate digital sketches of your idea or concept for an application or website.
- Easy to use.
- Simple and intuitive interface that makes it easy for anyone to use.

vi. MS Word:

Figure 64: MS Word

MS Word is a word processing program that allows for the creation of both simple and complex document.

Reason for using:

- User friendly.
- Simple and customizable.

vii. Draw.io

Figure 65: Draw io

Draw.io is free online diagram software for making flowcharts, process diagrams, org charts, UML, ER and network diagrams.

Reason for using:

- Free and easy to use.
- Extremely user friendly.

The libraries used in the development of the project are given below:

a) JRE System Library [JavaSE-21]:

This represents the Java Runtime Environment (JRE) for Java Standard Edition version 21. It provides the core java classes and runtime environment. It includes a set of libraries that provide essential functionality for java applications.

b) Maven Dependencies:

These are the external libraries managed by Maven. Maven simplifies the process of managing dependencies by automatically downloading the required libraries from a central repository and integrating them into your project.

7. Development process:

7.1 Development journey:

Our project's development process includes a number of stages and iterations. The development journey was a long and tedious work that took over a span of one month. The step by step process development of our project Nexus Nimbus is given below:

Step 1: Requirement gathering and selection of name:

On the 6th week of our second semester, we were provided with the coursework to create an E-commerce website. We made an extensive amount of research to gather the requirement needed in our coursework. We visited a number of similar websites that helped us gain a certain number of ideas and visualize our project. We also decided on a name for our project.

Figure 66: Requirement Gathering

Step 2: Requirement Analysis:

After the gathering of all the requirements for our project was done. We started the analysis of the requirements. We determined the key requirements needed for our project then arranged the requirements in the basis of their significance. The data models to be used for our projects were also determined during this phase.

Figure 67: Analyzing requirement by group

Step 3: Designing

After the gathering and analysis of requirements was completed, we went directly to the designing phase. In this phase, we determined the number of pages needed and created the wireframe for each of them. We also designed the class diagram needed for our project in this step.

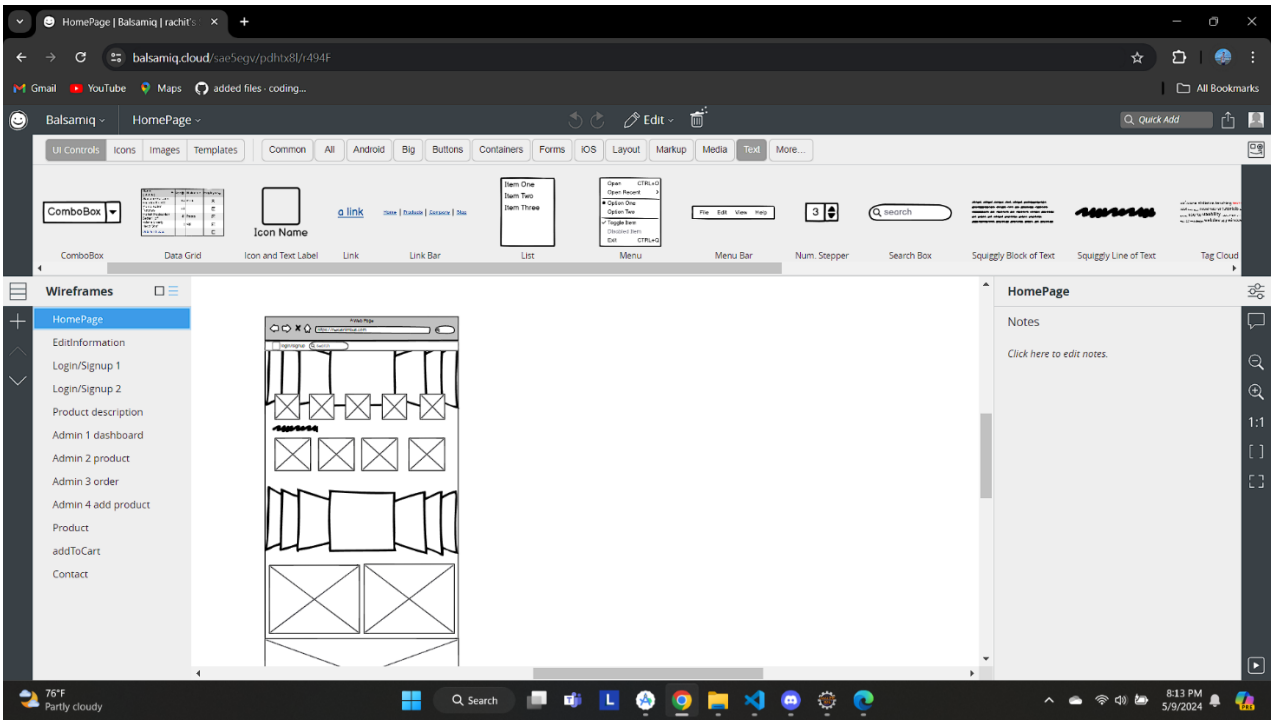

Figure 68: Designing wireframe

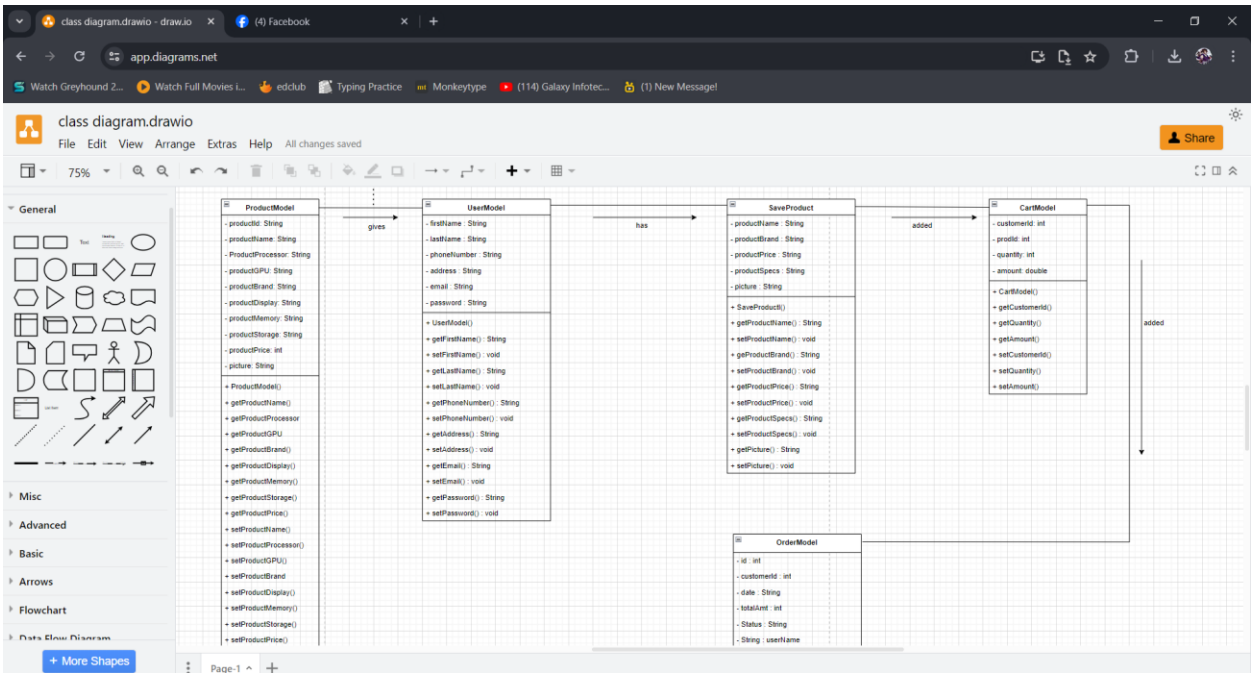

Figure 69: Developing class diagram.

Step 4: Implementation

After the wireframe were designed, we started the implementation immediately. As time was short, we had to start our implementation part quite early. The implementation included setting the prior planning into action. We created JSP pages as per our requirements, made database connectivity, created servlets etc. This process took us a significant amount of time. We used a variety of tools and libraries during this phase, including the Eclipse Development environment, Maven and SQL queries.

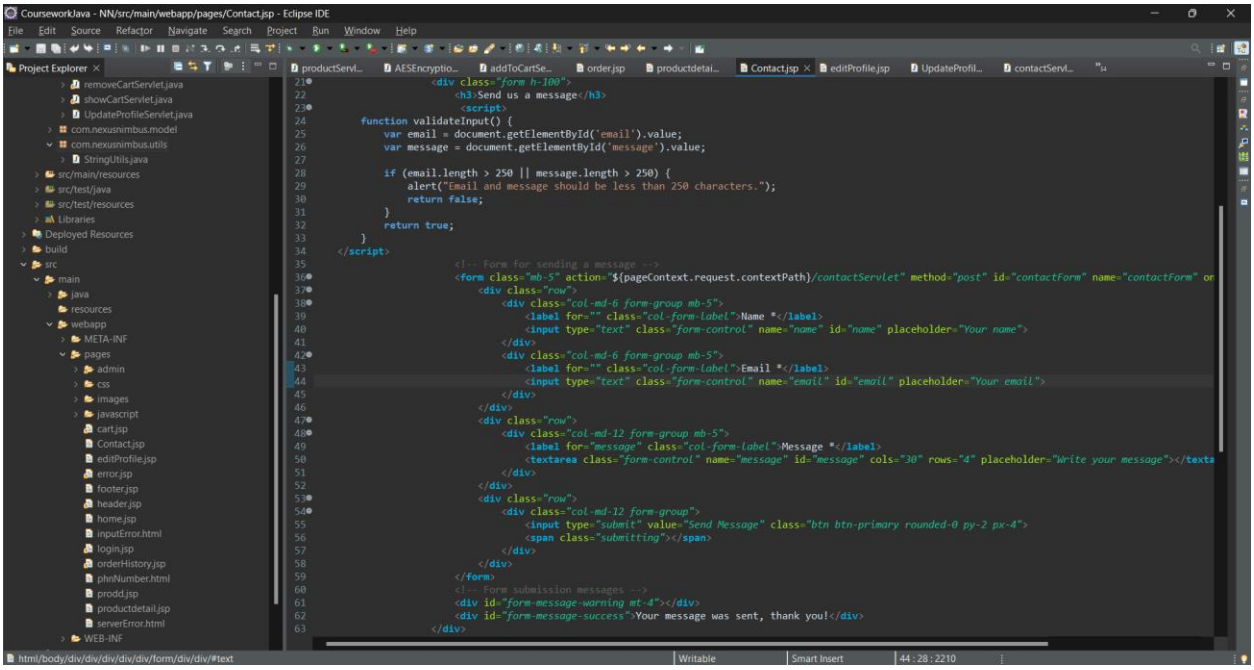

Figure 70: Developing system

Step 5: Testing

After the coding was completed for a single unit, we immediately conducted testing for that unit. We implemented both black box and white box testing for our finished unit. We used the unit method of testing in this process. Most of the testing pertaining to the individually produced units was conducted by our friends Prabal and Prabin.

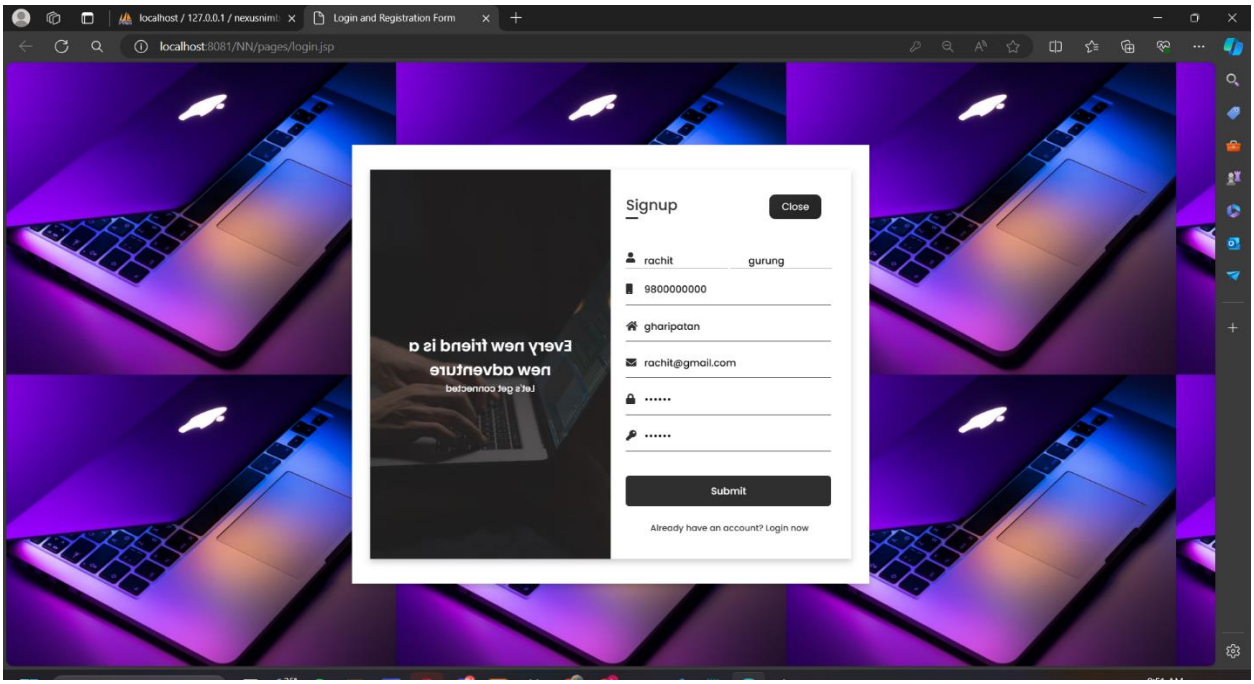

Figure 71: Testing of single unit

Step 6: Review and Update

After the testing was completed, each individual unit were reviewed by the members of the group. The problems encountered during the testing phase was dealt with in this step. Similarly, a number of reviews were provided by each member to develop our code into a more efficient code. The reviewed sections were updated according to the suggestion.

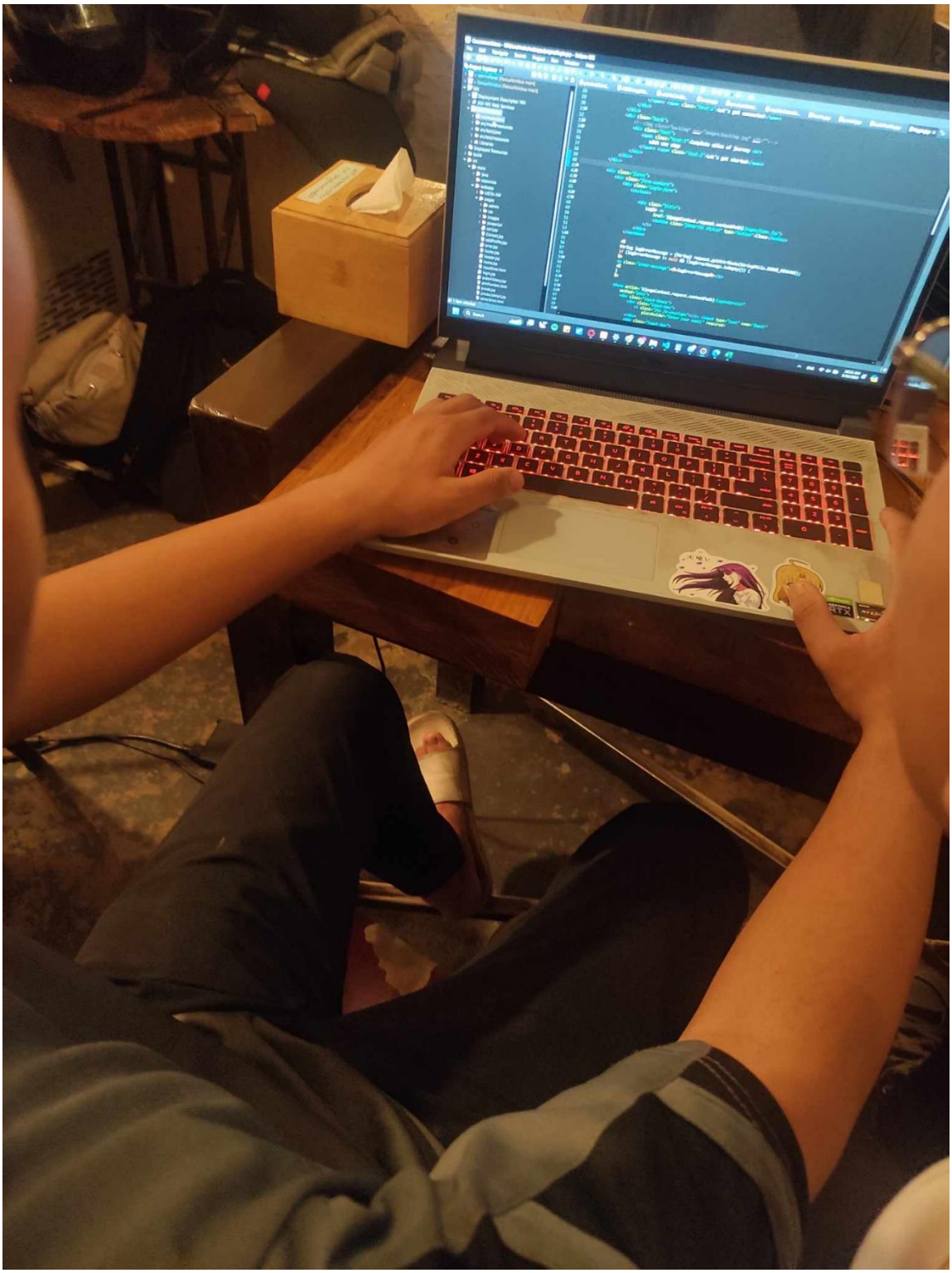

Figure 72: Reviewing code.

Step 7: Integration of the individual code

Once each piece of code had been developed independently, we proceeded to integrate them together. The integration phase was facilitated through GitHub, a collaborative platform for version control and code sharing. Through the use of GitHub, we integrated the code smoothly. This way, our final product worked well and met our goals.

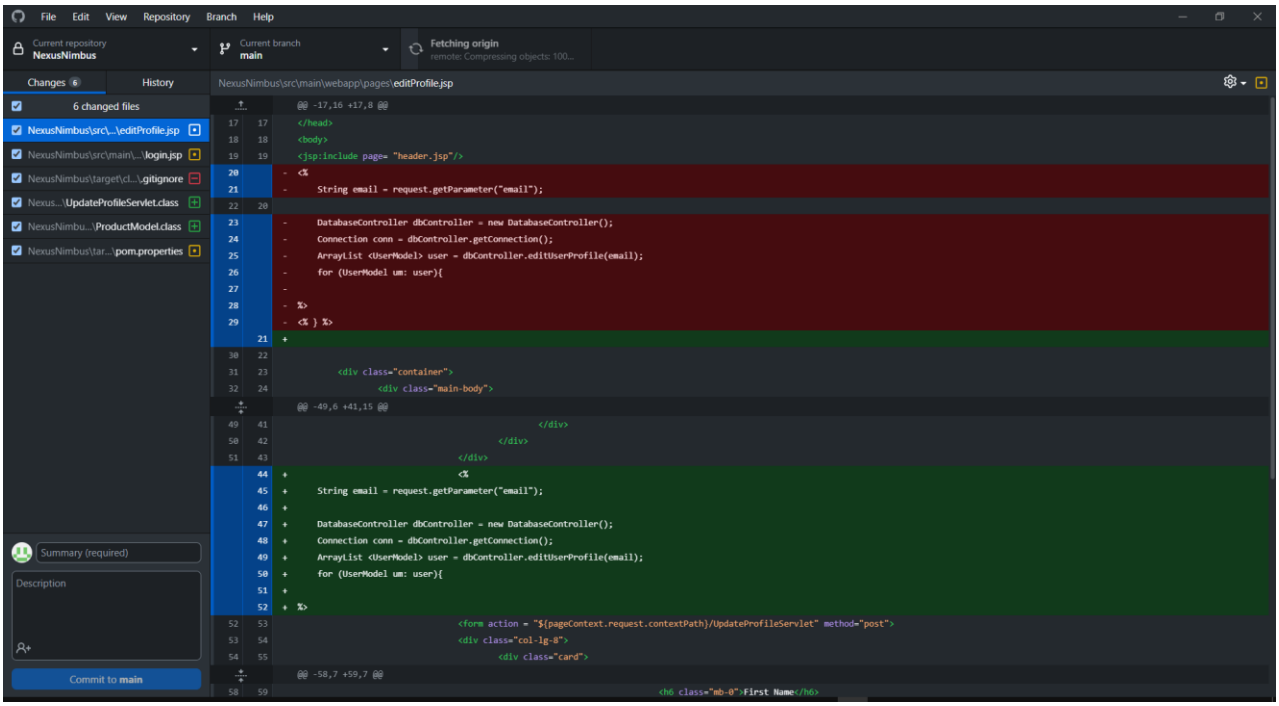

Figure 73: Integration of code

Step 8: Testing of the whole system:

After integrating all the components, the next step was testing the entire system. Our focus during testing was primarily on ensuring that the code merging process had been successful. We examined the functionality and performance of the integrated system, conducting thorough tests to identify any potential issues. This testing phase was crucial in verifying that the combined solution operated as intended and met the project requirements effectively.

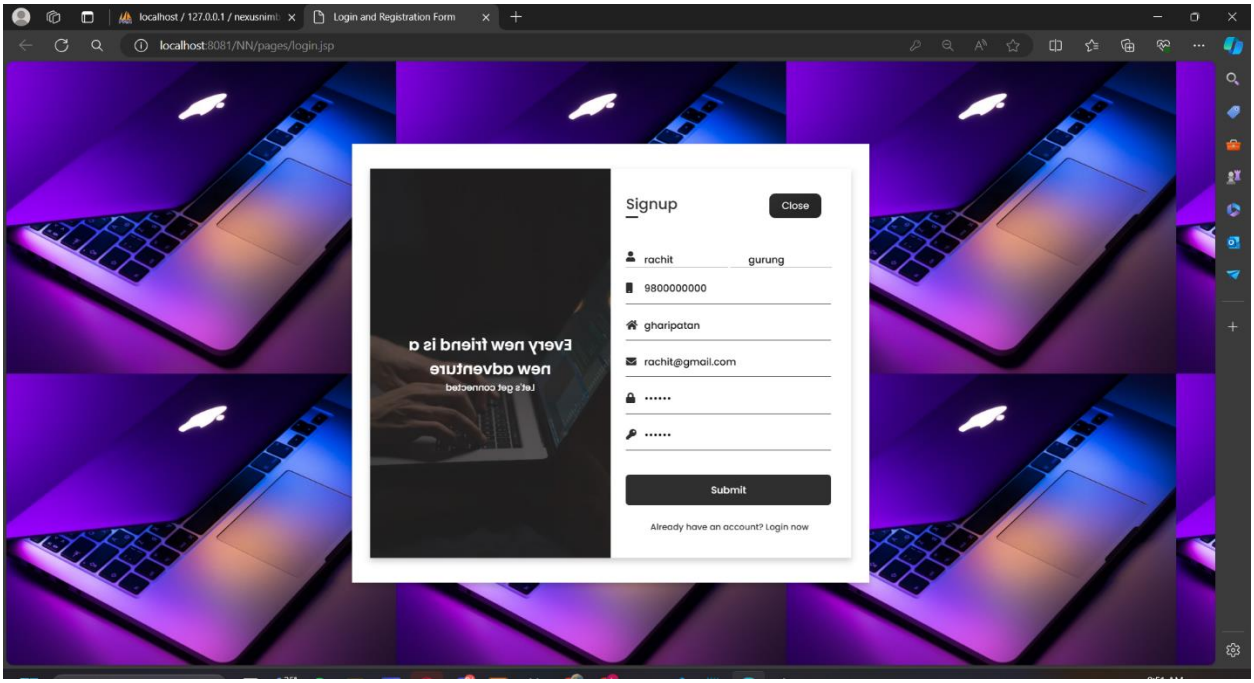

Figure 74: Testing of whole system I

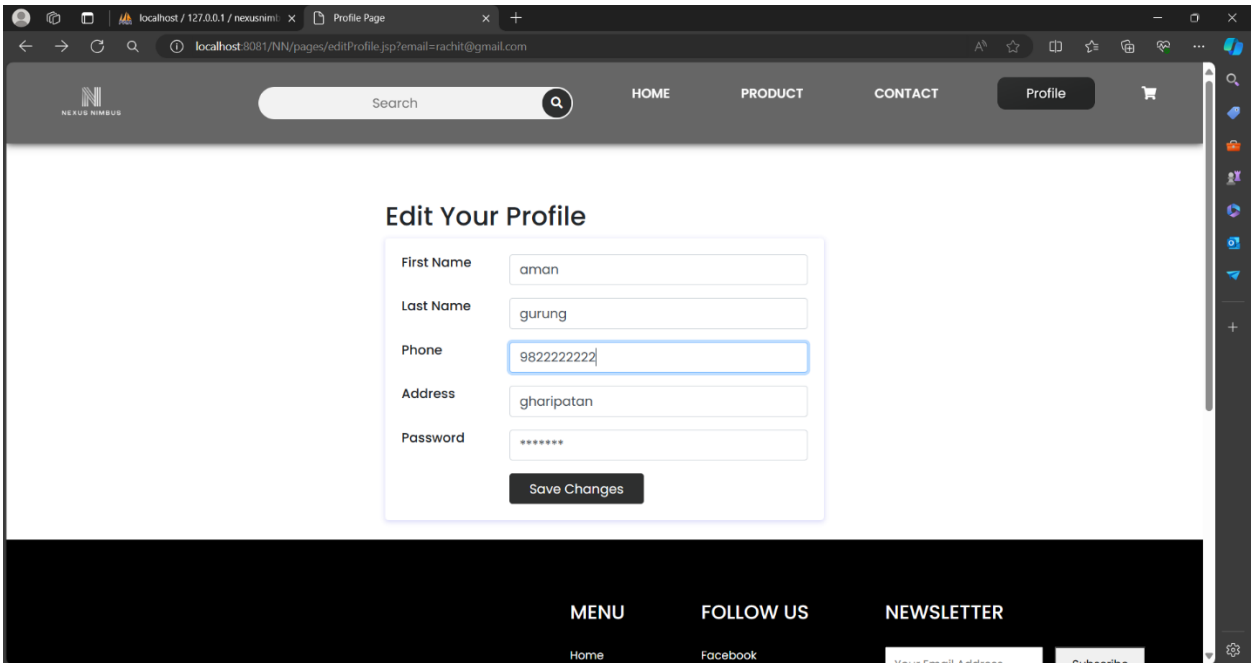

Figure 75: Testing of whole system II

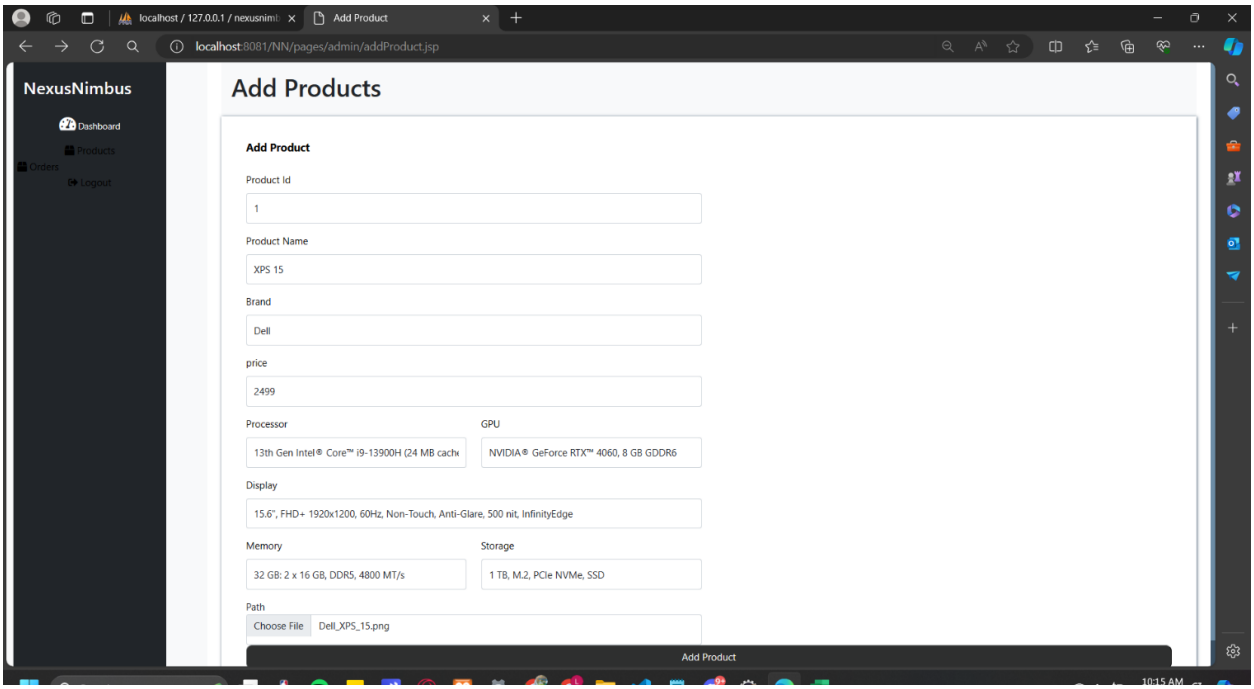

Figure 76: Testing of whole system III

7.2 Individual contribution:

The contribution made by each member of the group in the development and documentation part of our project is given below:

Table 30: Individual contribution

Name of group member	Contribution in development	Contribution in documentation
Prabal Gurung	Cart, order and admin page.	Testing, User interface design, Class diagram, method description.
Aman Gurung	Admin page and session management.	Development process, Class diagram, critical analysis
Prabin Thapa Magar	Edit profile page and session management.	Conclusion, Method description, critical analysis
Rachit Raj Shrestha	Product and product detail page.	User interface design, Testing, tools and libraries used.
Rahul G.C.	Registration and login.	Introduction, aims and objectives, conclusion.

8. Critical Analysis:

The completion of our project Nexus Nimbus stands as a testament to our group's dedication and passion. Our project was successfully developed, providing the users with an aesthetically pleasing and practical user interface for buying electronic products. The data are well maintained and stored in the localhost. Our project utilizes a variety of tools and frameworks like JSP, MySQL, Tomcat Servlet, Maven etc. to build a robust backend and appealing frontend. During the development of our project we faced a number of challenges. Some of the challenges that we faced during the development of our project are as follow:

- Linking of CSS file:

  During the development of the registration page, we encountered a path problem. When a message was to be shown after registration, the registration page was not able to find the CSS. We were able to solve it by using ${pageContext.request.contextPath} in the jsp file.

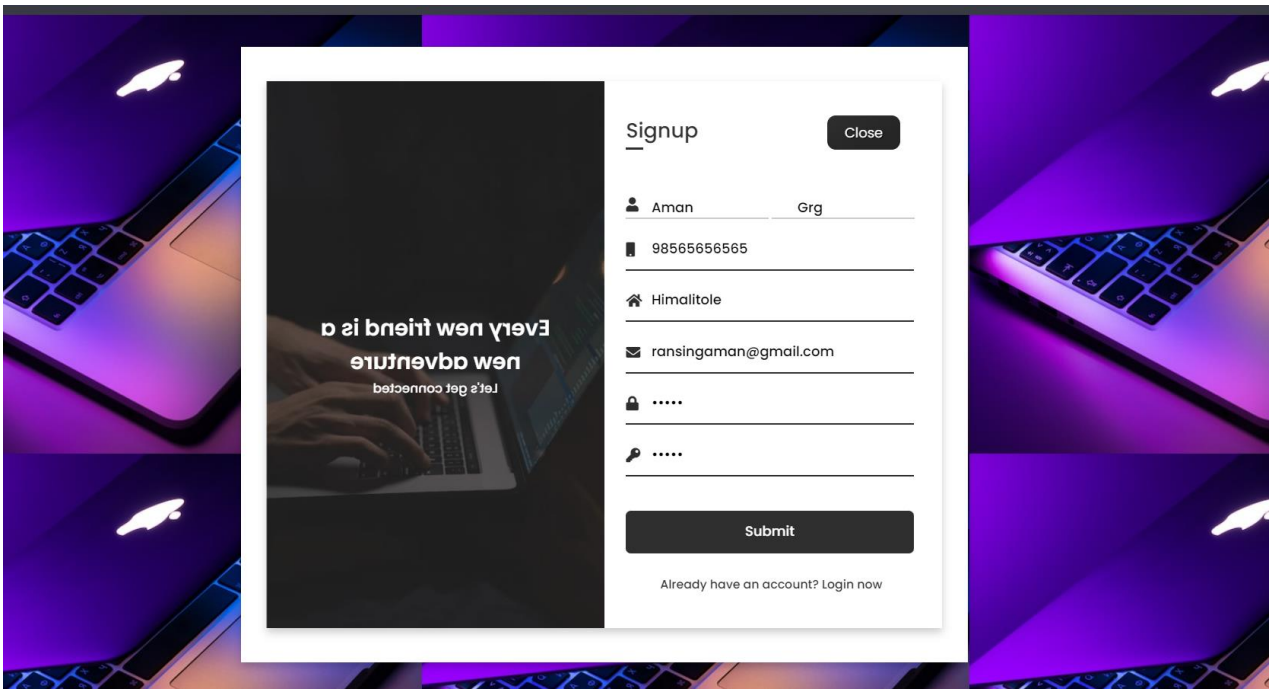

Figure 77: CSS Path issue.

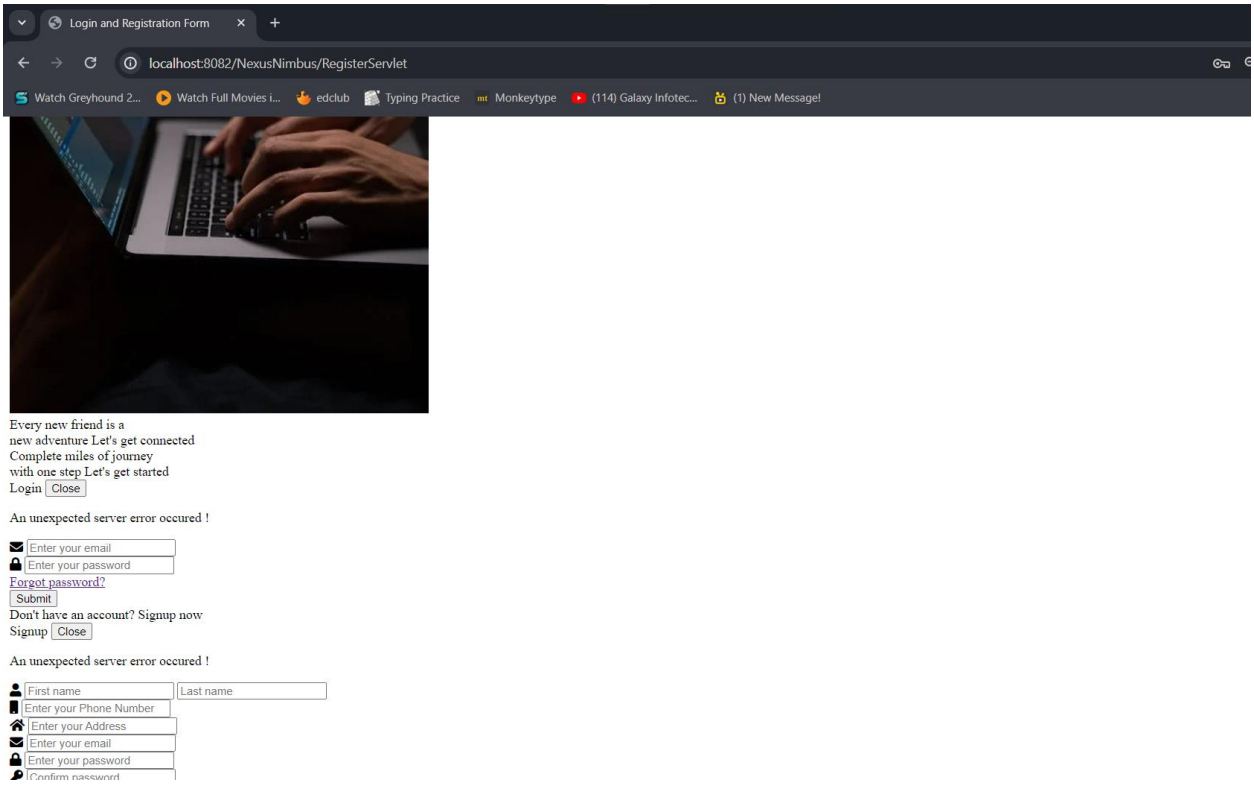

Figure 78: CSS Path issue.

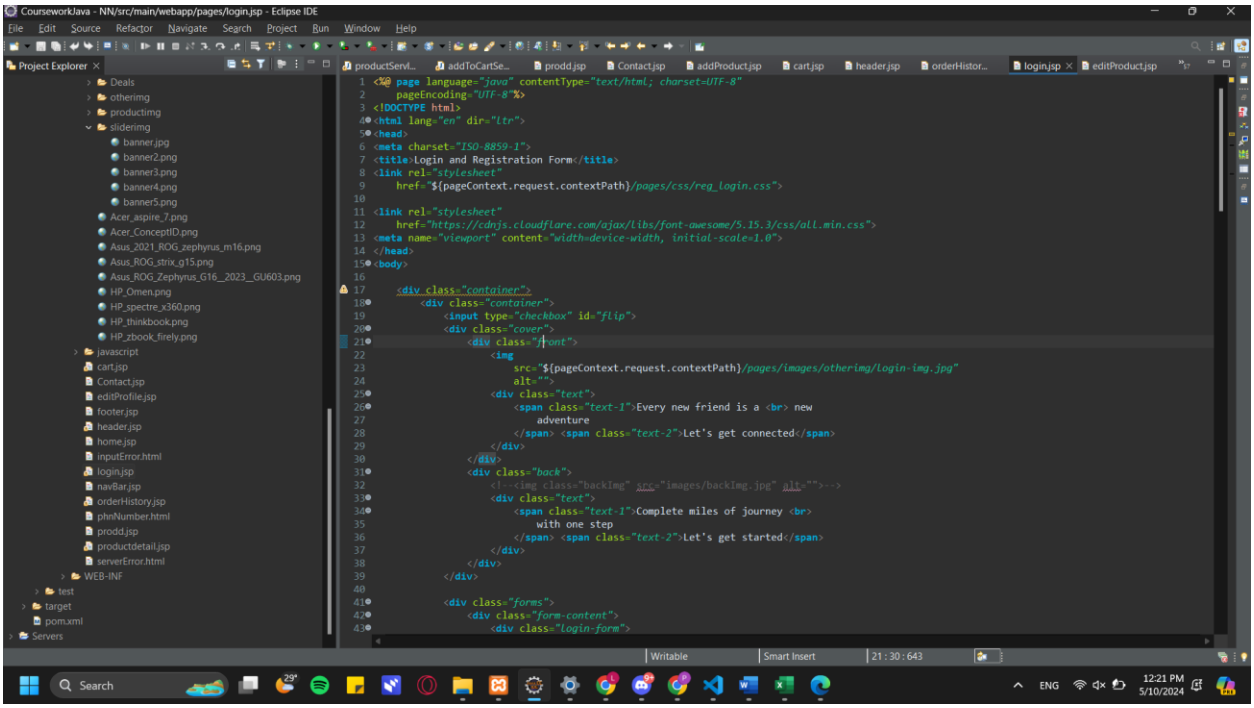

Figure 79: Fixing CSS Path

- Failure of tomcat server:

  During the development of our project, we encountered a problem where the tomcat failed to start. This problem was mitigated when we changed the port number of the server.

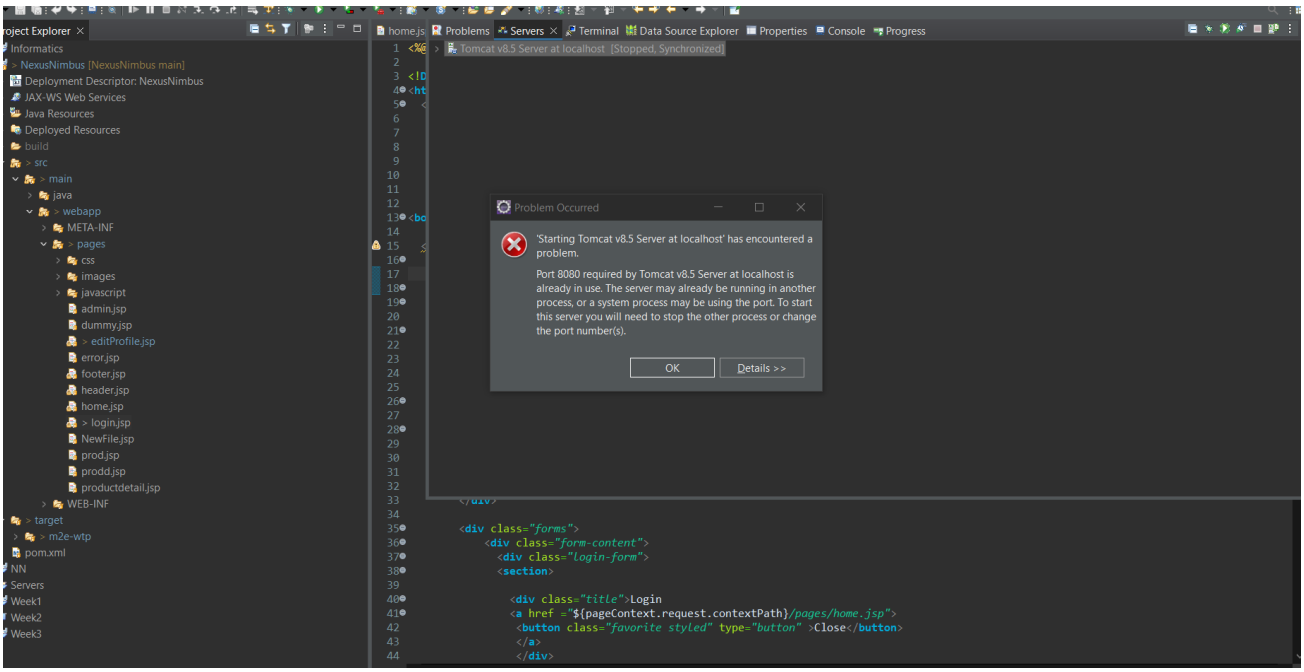

Figure 80: Failure of tomcat

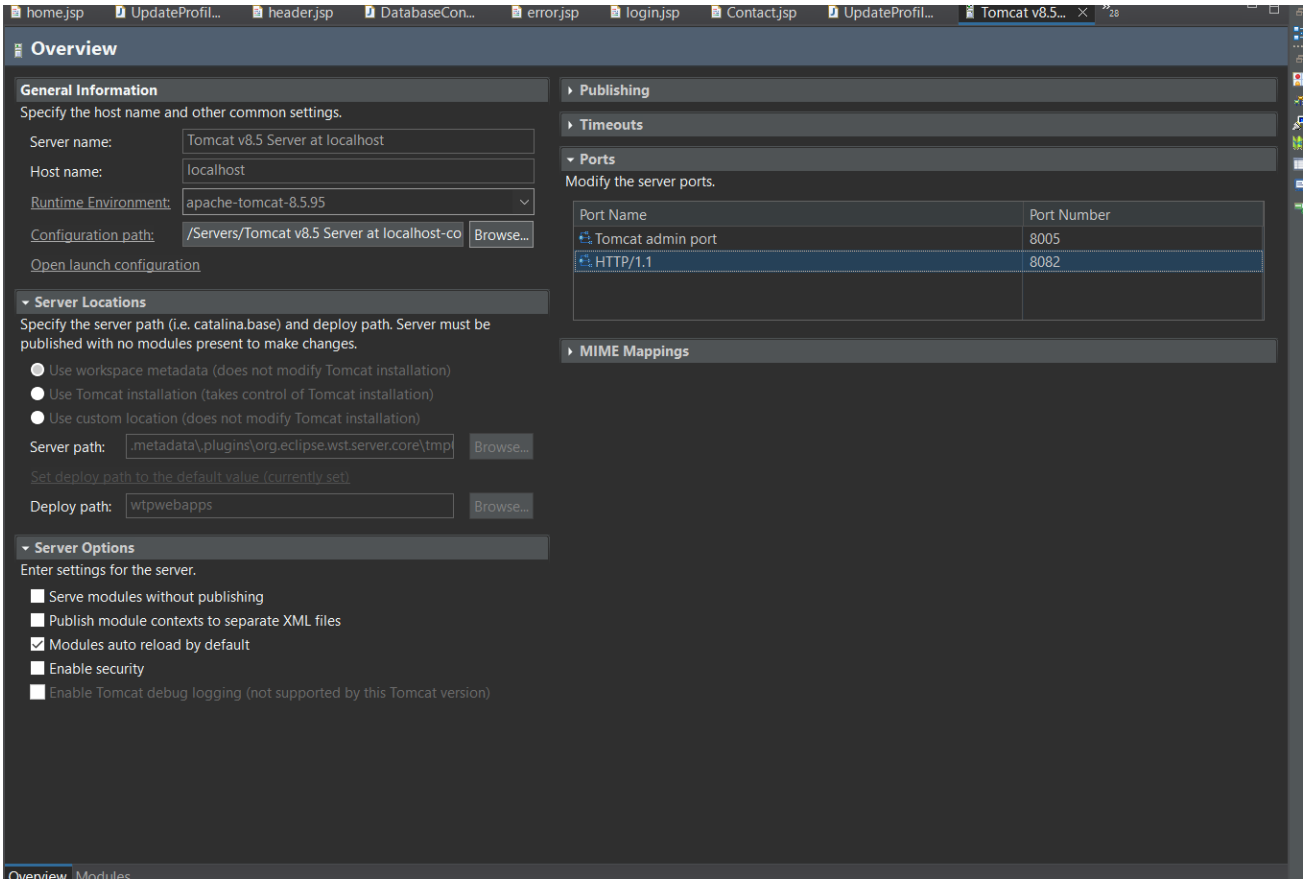

Figure 81: Changing port number

- Problem when uploading image of product:

  We encountered a problem that resulted in error when trying to upload product image while adding product in the database. This problem was resolved when we added @MultiConfig in our addProductServlet.

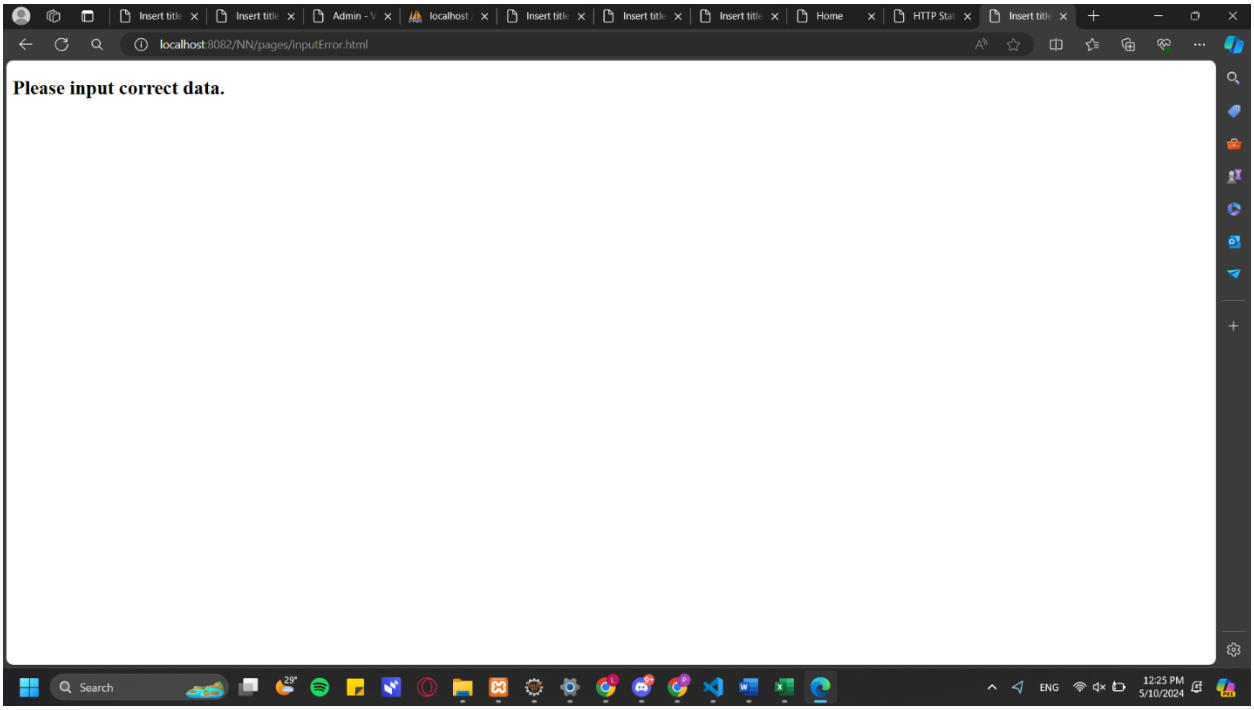

Figure 82: Error while adding picture.

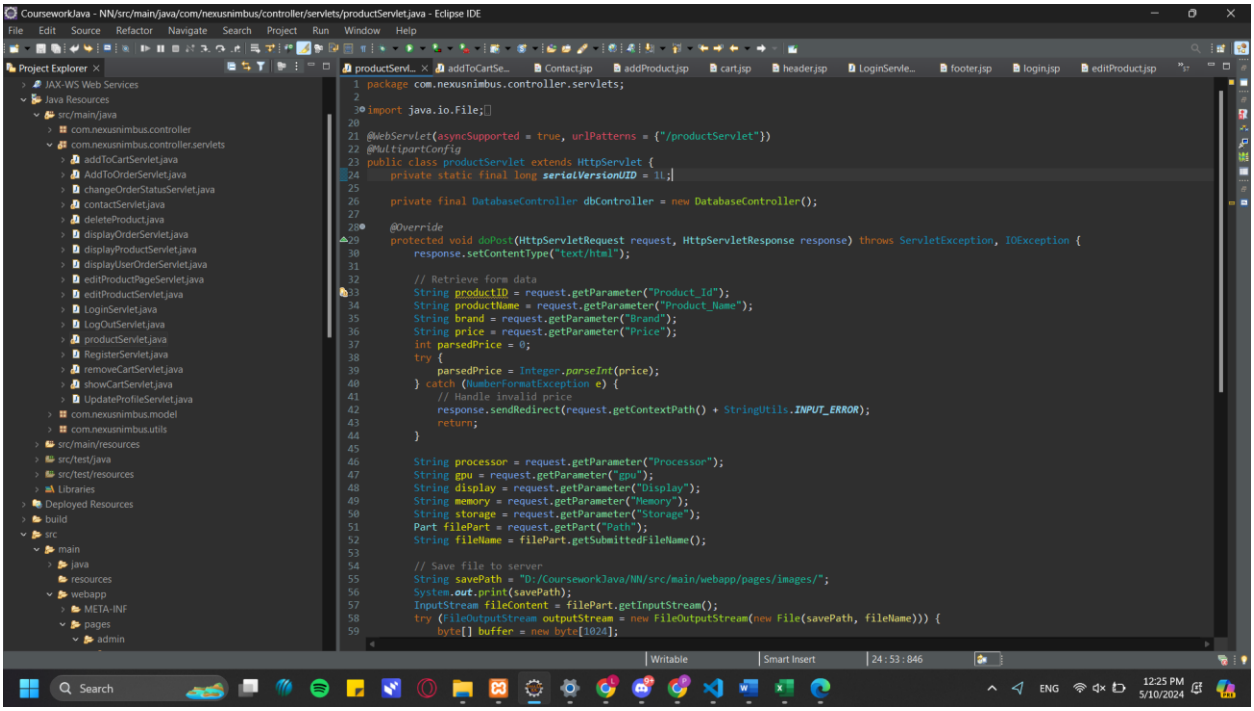

Figure 83: Adding @MultiConfig to resolve issue

9. Conclusion:

In conclusion, learning this module Advanced Programming and Technologies has been a fruitful endeavor. The module has provided us with a solid foundation of dynamic web application development. It has taught us how it works and the concepts needed in order to develop one. We have learnt about the various aspects such as JSP pages, servlets, JSP directives, session management, JSTL, MVC pattern and many more.

The development of our project “Nexus Nimbus” was a challenging but rewarding experience for us. We were able to create a website that caters to the demands of the customer. Our project was able to provide a platform that allows for seamless shopping experience to its customer with varieties of laptops and their detailed information with exceptional customer service. During the development of the project, we learnt a number of things. We learnt about the method required for the interaction between frontend and backend of the project. Database connectivity and manipulation of data from JSP pages was one of the crucial concepts needed in the project. We also learned to store images in the database. We were successful in applying the MVC design pattern to our project. Similarly, the application of session management in our project has been instrumental in ensuring secure and efficient user interactions. We encountered a lot of problems and issues while developing the project. However, we were able to tackle it due to the help of our module leader as well as through extensive research on the internet. We learned about different HTTP response status code as well as the ways to mitigate and tackle on such issues.

Overall, the project has been a valuable experience for us. It has developed our experience in dealing with dynamic web application and has helped us build our knowledge and skill regarding the development of such projects. This project has been a fulfilling journey, providing us with insights into effective collaboration, problem-solving, and the importance of adaptability in the ever-evolving field of technology. Additionally, it has given us a better understanding of the complexities of software development and how it affects both end-users and businesses.

Bibliography

Balsamiq, n.d. *balsamiq.com.* [Online] Available at: https://balsamiq.com/wireframes/desktop/docs/intro/ [Accessed 1 May 2024].

Gisonna, N., 2024. *birtannica.com.* [Online] Available at: https://www.britannica.com/topic/MySQL [Accessed 1 May 2024].

Minh, N. H., 2020. *codejava.net.* [Online] Available at: https://www.codejava.net/ides/eclipse/what-is-eclipse-ide-for-beginner [Accessed 29 April 2024].

S.Sridevi, 2014. USER INTERFACE DESIGN. *International Journal of Computer Science and Information Technology Research,* 2(2), pp. 415-426.

Singh, P., 2023. *medium.com.* [Online] Available at: https://iampravo.medium.com/web-servers-once-for-all-read-7291cb558492 [Accessed 1 May 2024].

Springer, 2005. Class Diagram. *Reverse Engineering of Object Oriented Code. Monographs in Computer Science..*

Velarde, O., 2023. *visme.co.* [Online] Available at: https://visme.co/blog/what-is-a-wireframe/ [Accessed 29 April 2024].

World, P., 2024. *medium.com.* [Online] Available at: https://medium.com/@Rachelshattuck/introduction-to-xampp-control-panel-8b27dfcaa0a3 [Accessed 30 April 2024].

Zenva, 2023. *gamedevacademy.org.* [Online] Available at: https://gamedevacademy.org/what-are-methods-in-programming-complete-guide/ [Accessed 1 May 2024].