Higher Nationals

Internal verification of assessment decisions – BTEC (RQF)

INTERNAL VERIFICATION – ASSESSMENT DECISIONS					
Programme title	Higher National Diploma in Computing				
Assessor	Mr. Jeykanth		**Internal Verifier**	Mr .Lakindu Premachandra	
Unit(s)	**Unit 29 – Application Program Interfaces**				
Assignment title	online shopping system for OZQ company				
Student's name	Ranudi Gayathmie Kariyapperuma				
List which assessment criteria the Assessor has awarded.	**Pass**		**Merit**		**Distinction**
INTERNAL VERIFIER CHECKLIST					
Do the assessment criteria awarded match those shown in the assignment brief?	Y/N				
Is the Pass/Merit/Distinction grade awarded justified by the assessor's comments on the student work?	Y/N				
Has the work been assessed accurately?	Y/N				
Is the feedback to the student: Give details: • Constructive? • Linked to relevant assessment criteria? • Identifying opportunities for improved performance? • Agreeing actions?	Y/N Y/N Y/N Y/N				
Does the assessment decision need amending?	Y/N				
Assessor signature			**Date**		
Internal Verifier signature			**Date**		
Programme Leader signature (if required)			**Date**		

Confirm action completed

Remedial action taken Give details:			
Assessor signature		**Date**	
Internal Verifier signature		**Date**	
Programme Leader signature (if required)		**Date**	

Pearson Higher Nationals in

Computing

Unit 29 – Application Program Interface

Higher Nationals - Summative

Assignment Feedback Form

Student Name/ID	Ranudi Gayathmie Kariyapperuma KIR/X - 00104243		
Unit Title	**Unit 29 – Application Program Interfaces**		
Assignment Number	**1**	**Assessor**	
Submission Date	31.12.2023	**Date Received 1st submission**	
Re-submission Date		**Date Received 2nd submission**	

Assessor Feedback:

LO1 Examine what an API is, the need for APIs and types of APIs

Pass, Merit & Distinction Descripts **P1** ☐ **M1** ☐ D1 ☐

LO2 Apply the knowledge of API research to design an application that incorporates relevant APIs for a given scenario or a substantial student chosen application

Pass, Merit & Distinction Descripts **P2** ☐ **M2** ☐ D2 ☐

LO3 Implement an application in a suitable development environment

Pass, Merit & Distinction Descripts **P3** ☐ **M3** ☐ **D3** ☐

LO4 Document the testing of the application, review and reflect on the APIs used

Pass, Merit & Distinction Descripts **P4** ☐ **M4** ☐ **D4** ☐

Grade:	**Assessor Signature:**	**Date:**
Resubmission Feedback:		
Grade:	**Assessor Signature:**	**Date:**
Internal Verifier's Comments:		
Signature & Date:		

* Please note that grade decisions are provisional. They are only confirmed once internal and external moderation has taken place and grades decisions have been agreed at the assessment board.

Assignment Feedback

Formative Feedback: Assessor to Student			
Action Plan			
Summative feedback			
Feedback: Student to Assessor			
Assessor signature		**Date**	
Student signature		**Date**	

General Guidelines

1. A Cover page or title page – You should always attach a title page to your assignment. Use previous page as your cover sheet and make sure all the details are accurately filled.
2. Attach this brief as the first section of your assignment.
3. All the assignments should be prepared using a word processing software.
4. All the assignments should be printed on A4 sized papers. Use single side printing.
5. Allow 1" for top, bottom , right margins and 1.25" for the left margin of each page.

Word Processing Rules

1. The font size should be **12** point, and should be in the style of **Times New Roman**.
2. **Use 1.5 line spacing**. Left justify all paragraphs.
3. Ensure that all the headings are consistent in terms of the font size and font style.
4. Use **footer function in the word processor to insert Your Name, Subject, Assignment No, and Page Number on each pag**e. This is useful if individual sheets become detached for any reason.
5. Use word processing application spell check and grammar check function to help editing your assignment.

Important Points:

1. It is strictly prohibited to use textboxes to add texts in the assignments, except for the compulsory information. eg: Figures, tables of comparison etc. Adding text boxes in the body except for the before mentioned compulsory information will result in rejection of your work.
2. Carefully check the hand in date and the instructions given in the assignment. Late submissions will not be accepted.
3. Ensure that you give yourself enough time to complete the assignment by the due date.
4. Excuses of any nature will not be accepted for failure to hand in the work on time.
5. You must take responsibility for managing your own time effectively.
6. If you are unable to hand in your assignment on time and have valid reasons such as illness, you may apply (in writing) for an extension.
7. Failure to achieve at least PASS criteria will result in a REFERRAL grade .
8. Non-submission of work without valid reasons will lead to an automatic RE FERRAL. You will then be asked to complete an alternative assignment.
9. If you use other people's work or ideas in your assignment, reference them properly using HARVARD referencing system to avoid plagiarism. You have to provide both in-text citation and a reference list.
10. If you are proven to be guilty of plagiarism or any academic misconduct, your grade could be reduced to A REFERRAL or at worst you could be expelled from the course

Student Declaration

I hereby, declare that I know what plagiarism entails, namely to use another's work and to present it as my own without attributing the sources in the correct form. I further understand what it means to copy another's work.

1. I know that plagiarism is a punishable offence because it constitutes theft.
2. I understand the plagiarism and copying policy of Edexcel UK.
3. I know what the consequences will be if I plagiarise or copy another's work in any of the assignments for this program.
4. I declare therefore that all work presented by me for every aspect of my program, will be my own, and where I have made use of another's work, I will attribute the source in the correct way.
5. I acknowledge that the attachment of this document signed or not, constitutes a binding agreement between myself and Pearson, UK.
6. I understand that my assignment will not be considered as submitted if this document is not attached to the assignment.

ranudigk@gmail.com
Student's Signature: **Date:** 31.12.2023
(*Provide E-mail ID*) **(*Provide Submission Date*)**

Higher National Diploma in Business

Assignment Brief

Student Name /ID Number	Ranudi Gayathmie Kariyapperuma - KIR/X-00104243
Unit Number and Title	**Unit 29- Application Program Interfaces**
Academic Year	2021/2022
Unit Tutor	Mr. Jeykanth
Assignment Title	
Issue Date	05.11.2023
Submission Date	31.12.2023
IV Name & Date	

Submission format
Part 1 – Report: The submission should be in the form of an individual written report. This should be written in a concise, formal business style using single spacing and font size 12. You are required to make use of headings, paragraphs and subsections as appropriate, and all work must be supported with research. You must provide in-text citations and the reference list using Harvard referencing system. Part 2: Fully functional web solution. **The recommended word count for the report is 4,000–4,500 words excluding annexures. Note that word counts are indicative only and you would not be penalised for exceeding the word.** **Minimum word count – 4,000** **Maximum word count – 5,500**

Scenario

"ELEKS " is a Top 100 Global Outsourcing company. You work as an apprentice web developer for ELEKS . As part of your role, you have been asked to create an online shopping system for OZQ company.

Online shopping has grown its popularity over the years, mainly because people find it convenient and easy to shop from the comfort of their own home or office. This is one of the most enticing factors about online shopping Because of these reasons OZQ company has decided to develope an online shopping system.

OZQ-cart has facilitates business-to-consumer sales through its website. OZQ-cart system helps to buy any type of item online by choosing the listed products from the website. Following are the functional requirements of the system.

- Registration – Customers can view the store but only the members can buy items. To become a member of the website, the customer need to register for the membership.
- Login page - The Login page is peripheral of the secure area of the system and allows the user to log onto the web application. The user can view the store and add their order to the shopping cart.
- Shopping cart – Member can add their searched items to the cart.
- User Profile - The User Profile page is an area that allows the users to maintain their own information. The user can browse and search the items and add to the shopping cart.
- Item Search and Select - Each customer must be able to view the status of the placed order.
- Feedbacks – user can provide opinions/ feedback to the site.

Following are the non-functional requirements of the system.

- Performance
- Usability
- Reliability and availability
- Security

Develop a web based solution for the above scenario.

Activity 1 - Examine what an API is, the need for APIs and types of APIs.

1.1 Examin What is an API (Application Program Interface) ,types and the benefits of APIs and evaluate the potential security issues surrounding APIs with reference .

1.2 Examin the differences between API and SDK and Assess a range of APIs that covers a range of users of the proposed solution.

Activity 2 - Apply the knowledge of API research to design an application that incorporates relevant APIs for a given scenario or a substantial student chosen application

2.1 Apply the knowledge of API research and Analyse the alternative solutions similar to the proposed system that could be enhanced by a suitable API. Design an application that will utilize a range of APIs for the proposed solution and justify the design choices used.

Activity 3 - Implement an application in a suitable development environment

3.1 Develop an application with suitable android and web-site wireframes for the proposed system design in task 2. Provide all the interfaces of the system and the appropriate codes of it.

Activity 4 - Document the testing of the application, review and reflect on the APIs used

4.1 Design and complete a 'black box' and 'white box' testing methods for the developed system and update the system according to the results. Critically evaluate the APIs used within your application and results of your Test Plan. Include a review of the overall success of your multipage website and provide a data security report of the application you developed for the above sceanrio.

Observation Sheet

Activity No	Activity	Learning Outcome	Feedback (Pass/ Redo)
1	Examine what an API is, the need for APIs and types of APIs.	LO1	
2	Apply the knowledge of API research to design an application that incorporates relevant APIs for a given scenario or a substantial student Chosen application.	LO2	
3	Implement an application in a suitable development environment.	LO3	
4	Document the testing of the application, review and reflect on the APIs used.	LO4	

Comments:

Assessor Signature :……

ACKKNOWLEDGEMENT

At last author would like to share the experience while doing the project. Author learns many new things about the networking topics. The best thing which author can share is that author developed more interest in this subject. This Project gave author experience of how to do an event .

A very special thanks to Mr. Jeykanth who teach us this subject and Author thanks for who helped author to do this kind of project. Thank you!

Regards,

The author,

Ranudi Kariyapperuma

Tables of Content

Table of Figures

List of Tables

Analysis of Application Programming Interfaces (API) and Software Development Kits (SDK)

Application Programming Interface(API)

An Application Programming Interface, or API for short, is a collection of guidelines, rules and regulations, and equipment that enables exchange of information between various software programs. It outlines the techniques and data formats that programmers can employ to make it easier for different software systems or components to communicate with one another and function as a unit. Through the use of APIs, which operate as agents, applications can access the features or data of other platforms, operating systems, or services without needing to have a thorough understanding of their behind workings. They facilitate communication between various software components in a standardized manner, hence promoting interchange and improving the development process.

APIs can be used to access databases, hardware components, online services, and operating system features, among other things. They can be divided into several kinds, such as operating system APIs that grant access to operating system functions, web APIs that enable communication via the internet via protocols like HTTP/HTTPS, and library-based APIs that give features within programming languages. Additionally, documentation explaining how to utilize APIs is provided along with them. This documentation covers things like available endpoints, parameters, methods of authentication, and response formats. To properly use the API's features and incorporate it into their apps, developers must have access to this documentation.

To put it simply, APIs are essential to the development of modern software because they facilitate interoperability, encourage creativity by integrating disparate services, and increase productivity by letting programmers use already existing features rather than having to create them from the beginning for each application.

Benefits of Application Programming Interfaces (API)

- Interconnection: APIs serve as a conduit for data sharing and communication across disparate systems. Different apps can collaborate effectively thanks to this interoperability, irrespective of the underlying architectures or technologies.

- Efficiency and Productivity: Developers can save time and effort by utilizing pre-existing APIs instead of creating new functionalities from the ground up. APIs offer pre-made tools and solutions that are simple to include into new applications, accelerating development and raising output levels overall.

- Scalability and Flexibility: By permitting modular design, APIs enable scalable architectures. The ability for developers to add or change features without impacting the system as a whole makes it simpler to adjust to evolving business requirements and support expansion.

- Collaboration and Innovation: APIs promote creativity by enabling developers to imaginatively blend various services and functionality. They promote cooperation between various developers, businesses, or sectors, which results in the development of fresh, integrated solutions.

- Improved User Experience: By facilitating the integration of other services, APIs improve user experience. For instance, API-integrated payment gateways, mapping services, and social media login options improve customer convenience and functionality.

- Possibilities for Monetization: Businesses frequently use APIs to expose specific features or data, which opens up possibilities for revenue. This can entail charging for more consumption over predetermined thresholds, opening up a marketplace for developers, or providing premium API access.

- Standardization and Simplification: The integration process is made simpler by the standardized interfaces and protocols that APIs offer. They make complexity more manageable by abstracting it, enabling more uniform and direct interactions between developers and complicated systems.

- Analytics and Insights: Built-in analytics are a common feature of APIs, offering insightful information about how their services are being utilized. This information is useful for understanding user behavior, optimizing services, and making well-informed decisions.

- Cross-platform compatibility: APIs allow the same features to be accessed by applications created for different platforms. Cross-platform interoperability is crucial in the complex technological environment of today.

- Security and Control: By using authentication, authorization, and rate limitation, APIs provide regulated access to particular features or data. They also make access management and monitoring possible, which improves system security as a whole.

Types of Application Programming Interfaces (API)

In the software industry, APIs, or application programming interfaces, are similar to translators. They set up the guidelines and protocols that various software programs must follow in order to communicate with one another and exchange information, features, or services. In essence, APIs are collections of definitions, tools, and protocols that permit communication between software systems. It take on different forms:

1. Open APIs (Public APIs)

Third-party organizations and outside developers can access Open APIs, also called Public APIs. Companies and organizations release these APIs to the public so that outside developers can access specific features or data. Clear usage guidelines and comprehensive documentation are typically included with open APIs. By encouraging developers from outside the company to

create services or apps that interface with the provider's systems, they want to promote innovation, grow the ecosystem of the platform, and possibly even bring in money through partnerships or higher usage. APIs offered by social networking sites like Facebook and Twitter or payment gateways like PayPal are a few examples.

2. Internal APIs

As the name implies, internal APIs are created exclusively for usage within a firm or organization. These APIs are utilized for internal communication and integration between various software components and services within the company, but they are not accessible to other parties. Internal APIs facilitate communication between different departments, systems, or applications within a company, hence enhancing efficiency, uniformity, and interoperability throughout its infrastructure. They make it possible for various software systems to operate together seamlessly, improving workflow and making data interchange within the ecosystem of the company easier.

3. Composite APIs

A composite API is an interface that combines or aggregates several APIs, frequently from various sources or services. These APIs combine multiple separate API requests into a single endpoint, abstracting the complexity of underlying functionalities. By offering a more user-friendly interface, lowering the quantity of network calls required, and delivering a consolidated view of data or services, composite APIs facilitate the development process. They are especially helpful when an application wants to communicate with several APIs at once or when certain features need to be coordinated across several services.

4. Partner APIs

Specifically created to facilitate cooperation between various organizations, partner APIs grant vetted partners controlled access to particular features or data. These APIs protect privacy, security, and frequently, monetization agreements while enabling safe communication and data exchange between collaborating businesses or entities. Partner APIs facilitate smooth integration between the systems of cooperating companies, which promotes corporate partnerships, joint ventures, and strategic alliances. Examples include APIs for supply chain management between

cooperating businesses and APIs used by financial institutions to safely communicate data with other service providers.

Potential Risks of Application Programming Interfaces (API)

Risks	Description
Security Vulnerabilities	Injection attacks, illegal access, and data breaches are just a few of the security threats that APIs may be subject to. Sensitive information may be exposed by poorly secured APIs if appropriate authorization and authentication procedures are not put in place. For example, improper input validation can result in injection attacks like as cross-site scripting (XSS) and SQL injection. Furthermore, inadequate rate limitation or access controls could allow unwanted access and jeopardize confidential data.
Issues with Stability and dependability	Latency, outages, and inconsistent service can all lead to problems with the stability and dependability of APIs. Dependent apps may not function properly if an API is unstable or unreliable. Errors in the API, network problems, server outages, and other issues could cause downtime or performance reduction. Applications that mostly rely on these APIs could face disruptions that impair productivity and user experience.
Challenges with Versioning and Compatibility	Over time, APIs change and add new features or modify current functions. But versioning issues may arise as a result of these modifications. There may not always be a guarantee of backward compatibility when APIs undergo revisions or version changes. Applications that haven't updated to the new

	API version may break or stop working as a result, creating compatibility problems.
Dependency on Third-Party Services	A lot of third-party platforms or services offer APIs. An organization cannot directly control a dependency that is introduced by relying on these external APIs. Dependent apps may not function as intended if a third-party service goes down, modifies what it offers, or stops supporting an API. This dependence risk emphasizes how crucial it is to have backup plans and diversify your API sources in order to reduce any potential outages.
Insufficient Authorization and Authentication	Insufficient authorization procedures or weak authentication systems can result in security breaches. Hackers may be able to access critical information or functionality without authorization if APIs lack strong authentication procedures, such as OAuth or API keys. Insufficient permission controls may permit users or programs to carry out operations that they shouldn't be able to, which could result in unapproved changes or data leaks.
Privacy and Data Exposure	There are serious hazards associated with APIs that handle data improperly or disclose too much of it. Inadequately constructed APIs may inadvertently reveal private information, breaking privacy laws or disclosing private information to unapproved parties. Inadequate data management practices, such as failing to encrypt confidential data while it's in transit or at rest, may lead to data breaches or leaks, jeopardizing user privacy.

Absence of Monitoring and Logging	Failure to adequately monitor and log API operations may make it more difficult to identify and react quickly to security incidents. It becomes difficult to identify unusual activity, possible attacks, or unauthorized access without thorough records and monitoring systems in place, which delays incident response and mitigation efforts.
Insecure Data Transmission	APIs are vulnerable to eavesdropping and interception when they transfer data over unencrypted connections. Data breaches may result from the transmission of sensitive data in plain text, which leaves it vulnerable to compromise while in route. Data could be intercepted by hackers if Secure Sockets Layer (SSL) or Transport Layer Security (TLS) encryption methods are not used.

Table 1 : Potential Risks of Application Programming Interfaces (API)

Software Development Kit (SDK)

In order to facilitate the development of applications for a particular platform, software framework, or hardware system, software companies and developers often provide Software Development Kits (SDKs), which are collections of tools, libraries, documentation, and sample code. By giving developers access to a collection of tools that simplify and accelerate the application development process, SDKs are intended to make development easier and faster.

The key components of an SDK

- Libraries and application programming interfaces (APIs) are pre-written code modules or interfaces that grant access to particular features or services. They provide developers with ready-to-use building blocks to use into their apps.

- Documentation: Comprehensive documentation instructs developers on how to make efficient use of the SDK's components. It provides recommendations, examples, code snippets, and explanations for using APIs and creating different functionalities.

- Development Tools: Integrated development environments (IDEs), emulators, debuggers, and other tools that help developers create, test, and debug applications are frequently included with SDKs.

- Code and Tutorials: Software Development Kits (SDKs) typically include example programs or code excerpts that show users how to utilize the SDK's features. Developers can use these examples as helpful guides to comprehend and use particular functionality.

SDKs might be tailored for particular programming languages or frameworks, or they can be platform-specific, aimed at operating systems like iOS or Android. For example, there are SDKs for developing games (Unity, Unreal Engine), mobile apps (iOS and Android), cloud services (AWS, Azure), and hardware (IoT devices, hardware APIs).

Tools of Software Development Kits (SDKs)

- Software Development Kits (SDKs) are collections of tools that developers use to help them create applications. These tools are intended to facilitate development and offer essential resources. The following are typical tools included in an SDK:

- Integrated Development Environment (IDE): An IDE is a full-featured software program that offers every tool needed for developing software. To help developers write, modify, debug, and test code, it comes with a code editor, debugger, compiler, and other tools.

- Compiler and Interpreter: SDKs frequently come with interpreters or compilers designed specifically for the platform or programming language they are intended for. These tools convert machine-readable code that can be run from human-readable code.

- Debugger: An SDK's debugging capabilities assist in finding and fixing mistakes or flaws in the code. To troubleshoot problems, they let developers to go through code, set breakpoints, examine variables, and examine program behavior.

- Software libraries and APIs: Libraries are sets of prewritten functions or code that provide particular features. Applications Programming Interfaces, or APIs, specify the tools, protocols, and procedures that allow various software components to connect and communicate with one another. Both offer tools to developers so they may incorporate different features into their apps without having to start from scratch.

- Emulators and Simulators: SDKs frequently come with emulators or simulators for platforms like mobile or IoT. By imitating the target platform's behavior, these tools let developers test apps without the need for real hardware.

- Tutorials and Documentation: Comprehensive documentation found in SDKs offers instructions on how to use a variety of tools and functionalities. Development resources such

as reference documents, sample code, and tutorials help developers comprehend and implement features efficiently.

- Tools for Testing and Profiling: Software Development Kits (SDKs) might come with tools to test the functionality, security, and performance of applications. Profiling tools are useful for analyzing code performance and pinpointing places that might be optimized.

- Deployment Tools: A few SDKs include tools to help with packaging and deploying apps for distribution or delivering apps to the target platform.

Benefits of Software Development Kit (SDK)

- increased pace of development.
- regularity in the creation of applications.
- Gaining entry to specific features.
- simplified incorporation with external services.
- less effort and time spent developing.
- regularity in coding procedures.
- Debugging and testing were made simpler.
- cross-platform development assistance.
- increased output from developers.
- availability of thorough documentation and materials.

Analysis regarding the wide range of APIs that can be used for the proposed solution

Online shopping is a form of modern shopping in a time when technology and convenience coincide. Accepting this change, OZQ Company is ready to introduce OZQ-Cart, an innovative e-commerce platform designed to provide a smooth and user-friendly experience. Given with bringing this idea to life as an apprentice web developer at ELEKS, Author set out to design an involved yet user-friendly web solution.

OZQ-Cart's primary goal is to change traditional purchasing by creating a simple online experience. OZQ-Cart uses a membership-based approach to open the door to an array of products. With an easy-to-use registration process, customers are welcomed into the store's domain and can quickly become valued members with the ability to make purchases.In the world of OZQ-Cart, security is essential, providing a secure environment for exploring and purchasing. The safe section is protected by a robust login page that allows registered customers to browse the virtual aisles and add desired items to their customized shopping carts.Users find comfort in the User Profile when navigating OZQ-Cart—a safe refuge for organizing personal data and preferences. Here, people have the ability to peruse the vast inventory, going with ease from item search to selection, and filling their carts with the goodies they have their eye on.The visibility of order status, an essential element that ensures customers are informed at every stage of their purchasing journey, is part of the ongoing effort to maximize efficiency. The platform also acts as an echo for its users, soliciting comments and viewpoints in order to promote an ecosystem that is driven by the community.

Above and beyond features, OZQ-Cart strives to improve the non-features that support its core. The platform's credibility and usability are reinforced by performance optimization, usability advancements, reliability, and availability.As a motivated and creative young web developer, my vision was to create a digital space where convenience and technology come together and users are transformed from consumers into informed customers. OZQ-Cart is more than simply an online store; it's an example of how cutting-edge technology and user-centered design can work together to change modern form.

As a web developer, authors vision was to create a digital space where convenience and technology come together and users are transformed from consumers into informed customers. OZQ-Cart is more than simply an online shopping platform; it's a monument to the marriage of state-of-the-art technology and customer-focused design, transforming online buying for the selective modern consumer.

A range of APIs for a particular platform that covers a range of uses.

API Category	**Identification & Use Case**	**Researched APIs**	**Evaluation Summary**
Login API	Application requiring secure user authentication. Objective: Enable access to the app's secure area without compromising credentials.	• OAuth • Auth0 • Firebase Authentication • Okta	OAuth: Widespread adoption, robust security. Auth0: Auth as a service, multiple methods. Firebase Auth: Secure, multi-platform. Okta: Security features, complex auth.
Site Search API	E-commerce site aiming for a powerful, accurate search feature for products.	• Algolia • Elasticsearch • Google Custom Search API • Azure Cognitive Search	Algolia: Speed, customization, scalability. Elasticsearch: Rich querying. Google Custom Search: Simplicity. Azure Cognitive Search: AI-driven.

Product Information API	E-commerce platform needing comprehensive product data for display.	• Amazon Product Advertising • Walmart Marketplace • eBay • Etsy	Amazon Product Advertising: Extensive database, accuracy. Walmart, eBay, Etsy: Coverage.
Payment API	E-commerce site requiring a secure payment processing system.	• Stripe • PayPal • REST API • Square • Braintree • Payment Gateway	Stripe: Robust security, multiple methods. PayPal: Trusted, feature-rich. Square, Braintree: Functionality, fees.
Shopping Cart API	E-commerce platform needing seamless cart management for user experience.	• Shopify • Magento WooCommerce BigCommerce	Shopify: Robust cart features, user-friendly. Magento: Complex but powerful. WooCommerce, BigCommerce: Functionality.
Messaging/Notifications	Application needing a robust engagement system via messaging.	• Twilio • Firebase • Cloud Messaging Amazon • SNS • SendGrid	Twilio: Reliability, scalability, various message types. Firebase Cloud: Integration, Firebase support. Amazon SNS, SendGrid: Features, integration.

Table 2 : A range of APIs for a particular platform that covers a range of uses.

Justification of API used to develop OZQ Shopping Site

Not only does OZQ require technology to create a secure environment for online shoppers, but it also requires a reliable gateway for user access. The one author prefer is for the Google Authentication API because it offers a secure and easy-to-use authentication mechanism. Its charm is found in a conglomeration of qualities that speak to our desire for greatness. It transforms user access by embracing the power of Single Sign-On (SSO) and harmonizing it with the well-known domain of Google credentials. This encourages higher levels of user engagement by improving the user journey and streamlining the onboarding process. Beneath the surface of seeming simplicity is the reinforcement of Google's security leadership.

It guards our customers' data with a barrier built on trust and dependability, acting as a sentinel against unwanted access. This trustmark, which is closely associated with Google's brand, encourages consumer confidence and may increase our user base. Furthermore, its smooth integration and developer-friendly APIs provide a bridge to quick implementation, which is helpful in our pursuit of speed without sacrificing quality. Its platform versatility predicts a story of future-proofing, perfectly lining up with our growth trajectory. And supporting it all is Google's strong infrastructure, which guarantees our presence even at the busiest times with performance that echoes dependability. The selection of Google Authentication API creates a safe, user-focused, and scalable tune for this digital symphony that resonates with the functional and non-functional notes of OZQ-cart's goals.

Functional and Non-functional Requirements of OZQ web based Online Shopping Store System

Functional requirements

- For users to be able to purchase things on the platform, they must be able to register as members of the system.
- Users should be able to access the online store and handle orders with ease if there is safe and intuitive login functionality in place.

- It is necessary to offer a thorough shopping cart function that allows registered users to choose the things they want to buy.
- For a more seamless shopping experience, user profiles should have strong administration capabilities that let users change personal information and quickly add things to their carts.
- Order status visibility is essential for enabling clients to monitor and stay informed about the status of their orders.
- A feedback mechanism that enables users to voice their ideas and offer insightful commentary on the purchasing experience should be incorporated into the system.
- It is imperative to incorporate user-friendly search and selection functions for products that allow speedy and effective discovery of required things.
- Incorporating a secure payment gateway is vital for ensuring users' seamless and secure financial transactions while making purchases.
- Giving users access to their order history makes it easier for them to place new orders and allows them to review their past purchases.
- By adding a wishlist function, users would be able create lists of things they would like to buy or contemplate in the future.

Non-Functional requirements

- The system should function very well, ensuring quick reaction times and smooth user interactions.
- In order to provide accessibility and user-friendliness across several devices, a mobile-responsive design is essential.
- Upholding a 99.9% system uptime pledge is essential for preserving dependability and constant availability.
- ensuring data security by encrypting user data to keep users private and prevent illegal access.
- For consumers to have a comfortable and productive browsing experience, pages should load quickly.
- Navigability and usability are improved by intuitive and user-centric user interface designs.

- It is imperative that the system has cross-browser compatibility in order to guarantee optimal performance across various web browsers.
- By putting in place a scalable infrastructure, the system may expand in the future and easily handle growing user demands.
- System stability and protection against data loss depend on regular backups of the system.
- Ensuring secure payment transactions is crucial to safeguarding users' financial information when conducting transactions.

Explanation of the user interfaces and features of the OZQ web based Online Shopping store system

- Welcome Interface: The welcome interface acts as the digital storefront's entry point, offering users their initial impression of the platform. This page typically combines visually appealing elements, such as graphics, slogans, or promotional banners, welcoming visitors and providing an overview of the platform's offerings. It's designed to engage users, direct them to key sections or promotions, and set the tone for their browsing or shopping experience.

- Sign-In Interface: The sign-in interface provides a secure gateway for registered users to access their accounts. It features input fields for users to enter their credentials, ensuring authentication before granting access to personalized features like saved preferences, order history, or account settings. This interface prioritizes security while offering convenience to returning users.

- Sign-Up Interface: The sign-up interface facilitates the onboarding process for new users. It presents a form or series of input fields, prompting users to provide necessary details like name, email, password, and other relevant information for account creation. The aim is to simplify registration while gathering essential data for personalized user experiences.

- Customer Feedback Interface: This interface serves as a channel for users to express their opinions, suggestions, or experiences. It often includes forms or sections where users can submit feedback, reviews, or comments, enabling the platform to collect valuable insights to improve services, products, or overall user experience.

- User Profile Interface: The user profile interface is a personalized space allowing users to manage their account information. It typically includes sections for users to update personal details, change passwords, adjust preferences, and view past orders or interactions. This interface emphasizes user autonomy and customization.

- Shopping Store Interface: This section functions as the digital catalog or storefront, showcasing available products. It displays a range of items, organized into categories or sections, along with detailed descriptions, images, prices, and possibly reviews. The interface may incorporate filters or search functionalities for easy navigation, enabling users to explore and select desired products efficiently.

- Shopping Cart Interface: The shopping cart interface is where users review and manage selected items before making a purchase. It displays a summary of chosen products, quantities, prices, and allows users to modify or remove items as needed. This interface streamlines the checkout process, providing users with a clear overview of their selected items before finalizing their purchase.

OZQ web based Online Shopping Store System Design Wireframes and User Interfaces

OZQ web based Online Shopping Store System Design Wireframes

Welcome Interface

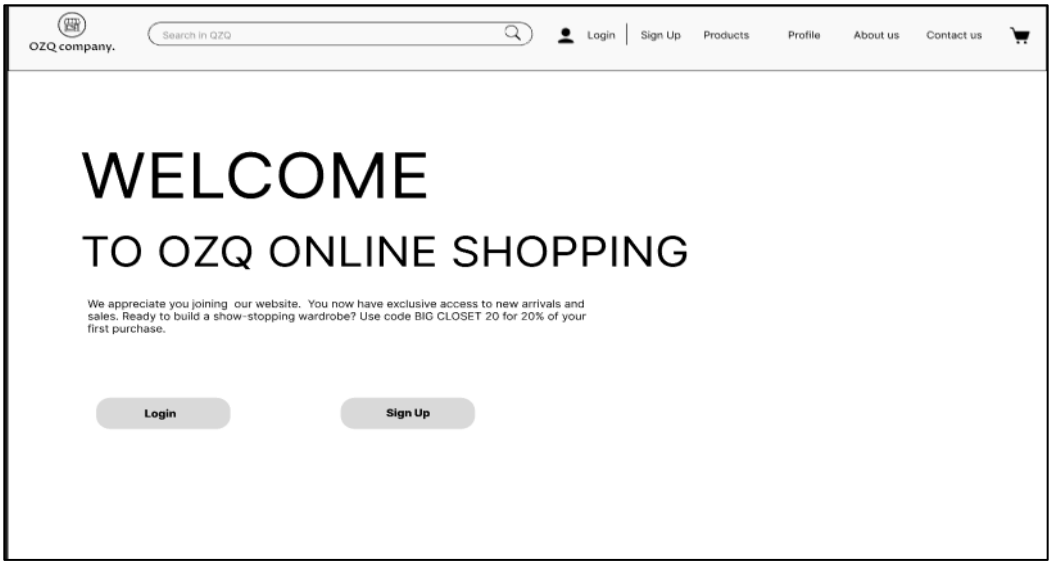


Figure 1 : Welcome Interface Wireframe (Developed by Author)

Sign In Interface

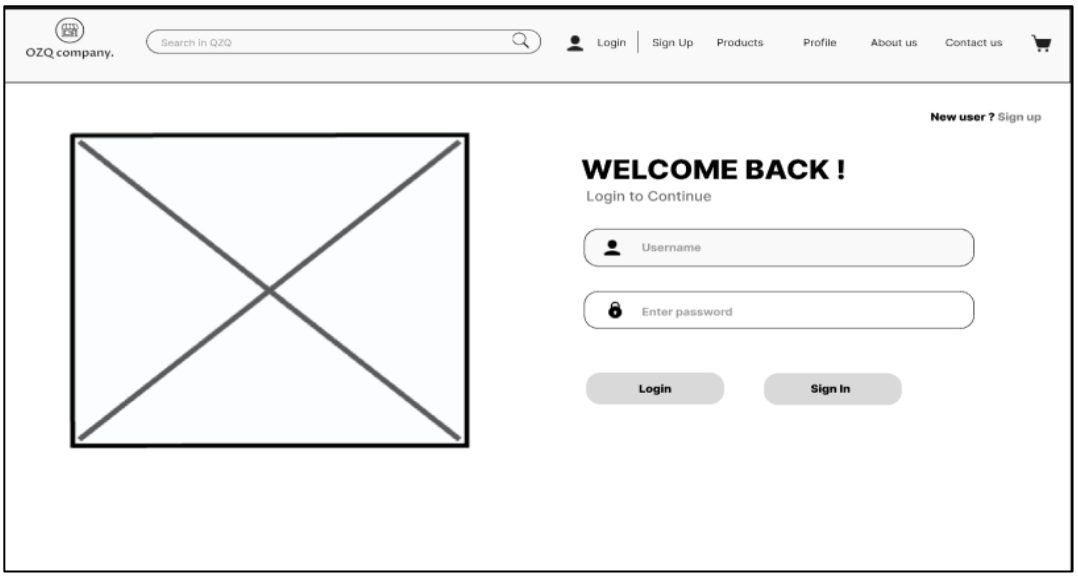


Figure 2 : Sign In Wireframe (Developed by Author)

Sign Up Interface

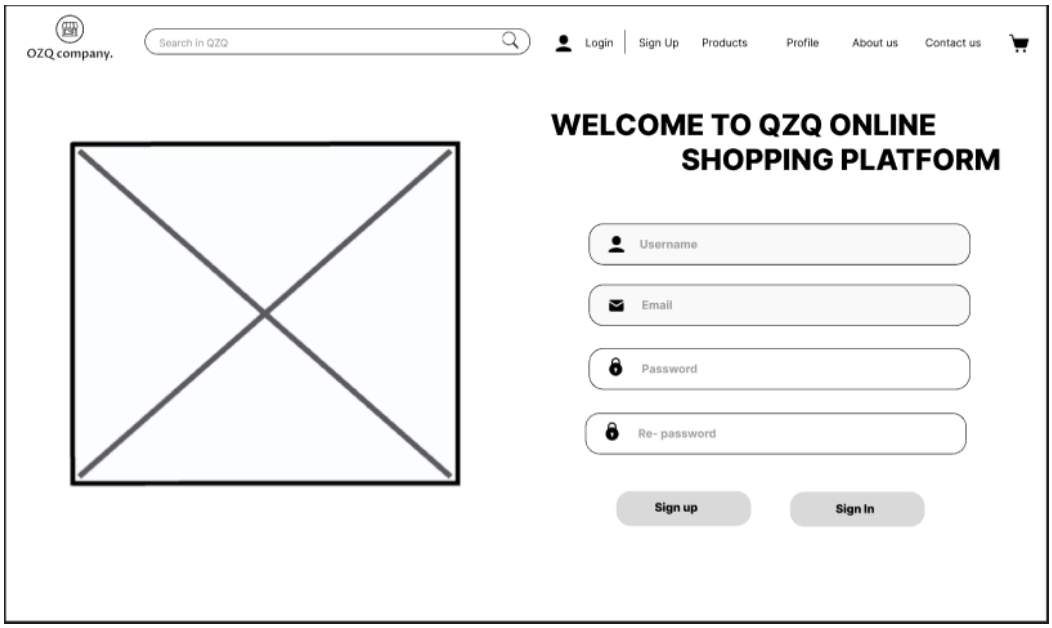


Figure 3: Sign Up Wireframe (Developed by Author)

User Profile Interface

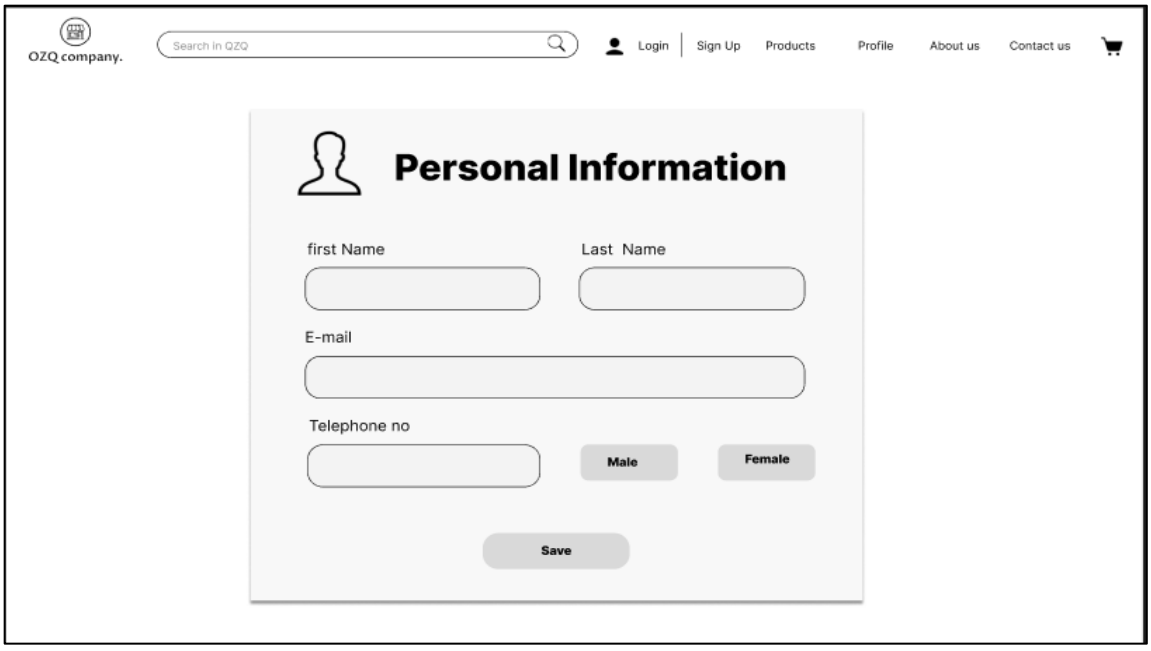


Figure 4: User Profile Wireframe (Developed by Author)

Customer Feedback Interface

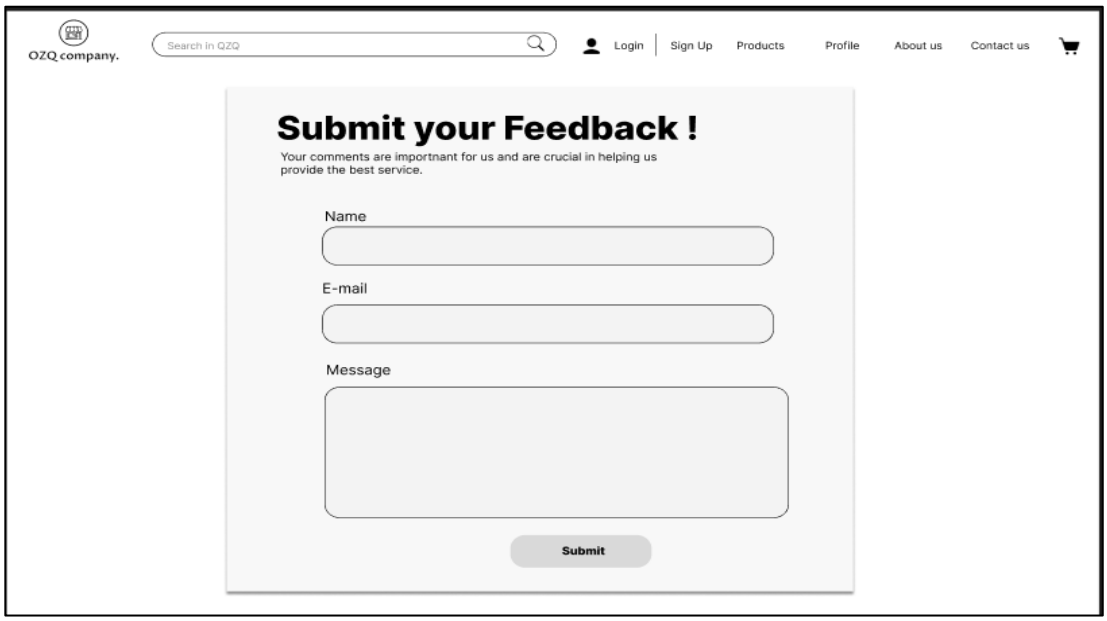


Figure 5: Customer Feedback Wireframe (Developed by Author)

Shopping store Interface

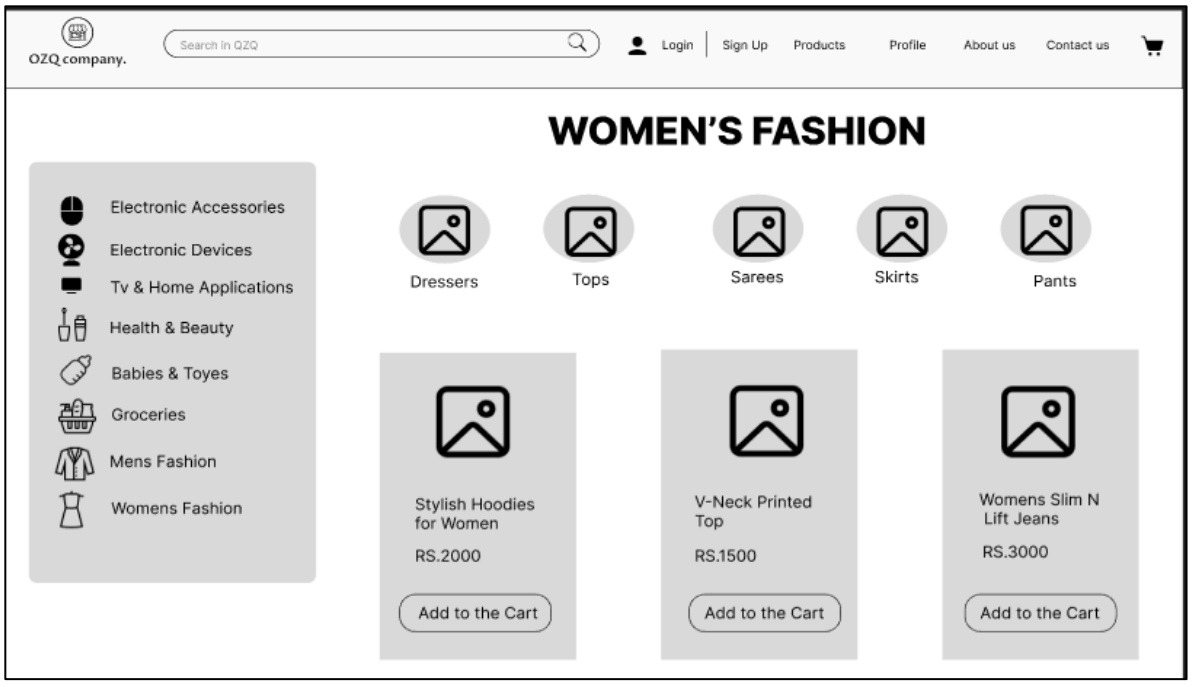


Figure 6 : Shopping store Wireframe (Developed by Author)

Shopping Cart Interface

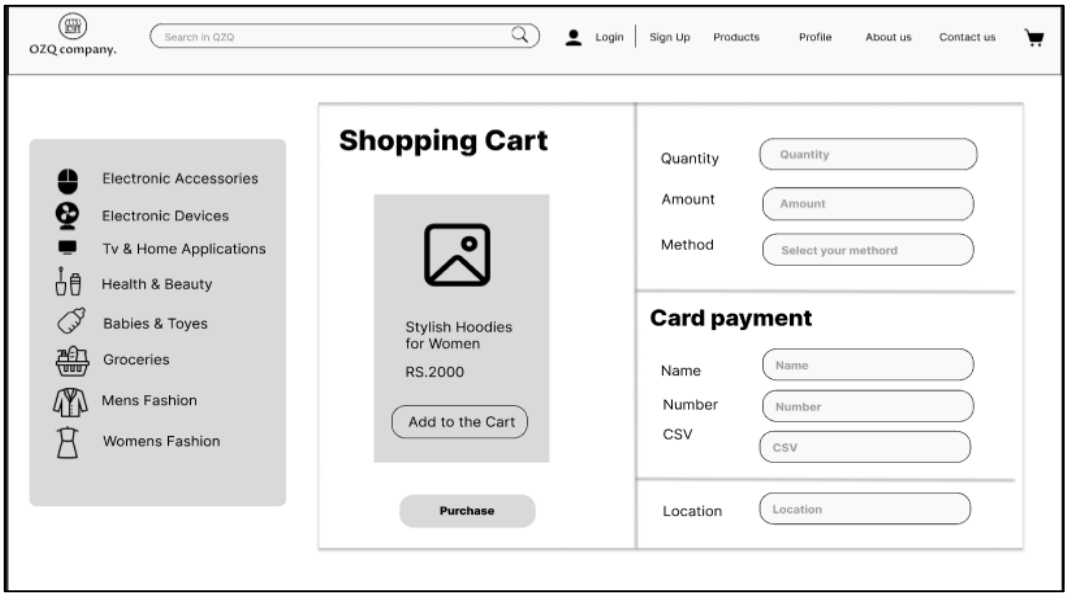


Figure 7 : Shopping cart Wireframe (Developed by Author)

OZQ web based Online Shopping Store System Design Interfaces

Welcome Interface

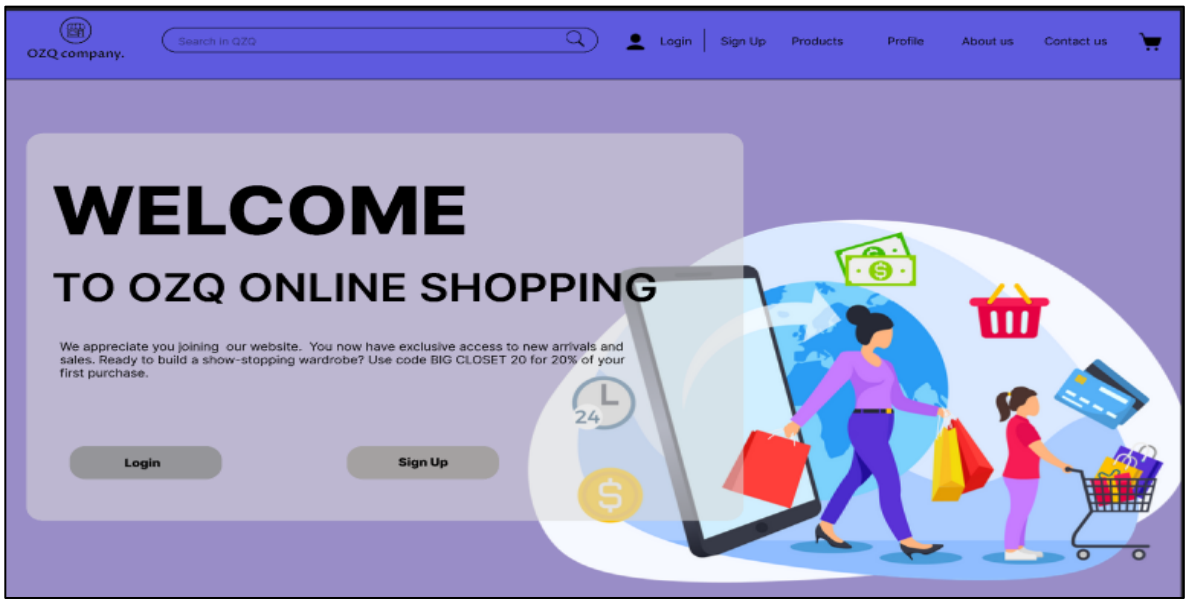


Figure 8 : Welcome Interface (Developed by Author)

Sign In Interface

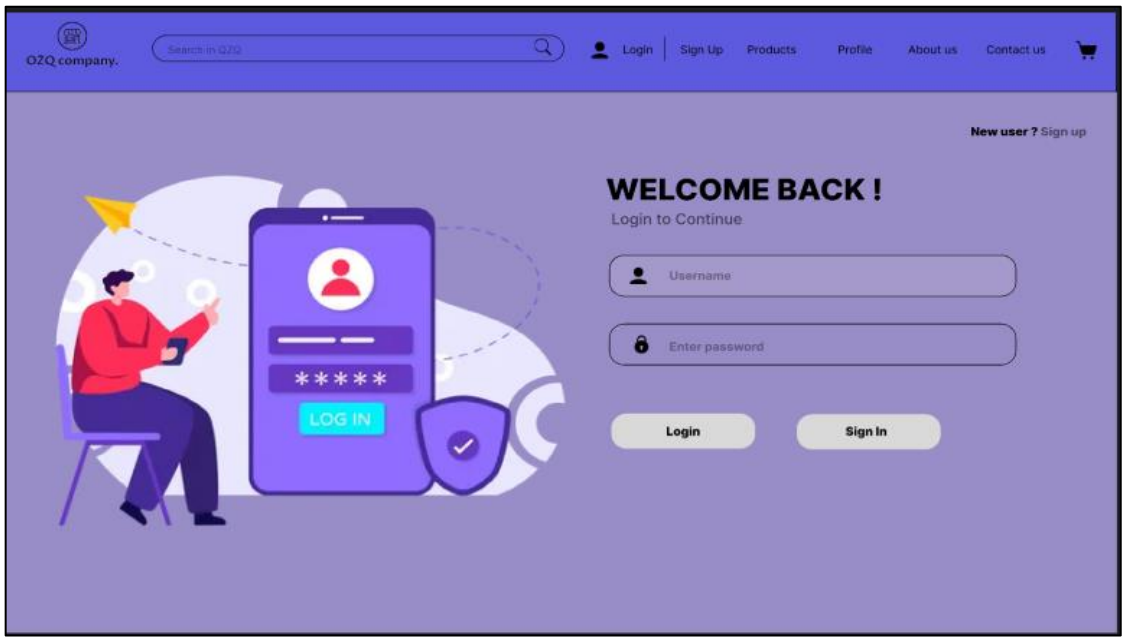


Figure 9 : Sign In Interface (Developed by Author)

Sign Up Interface

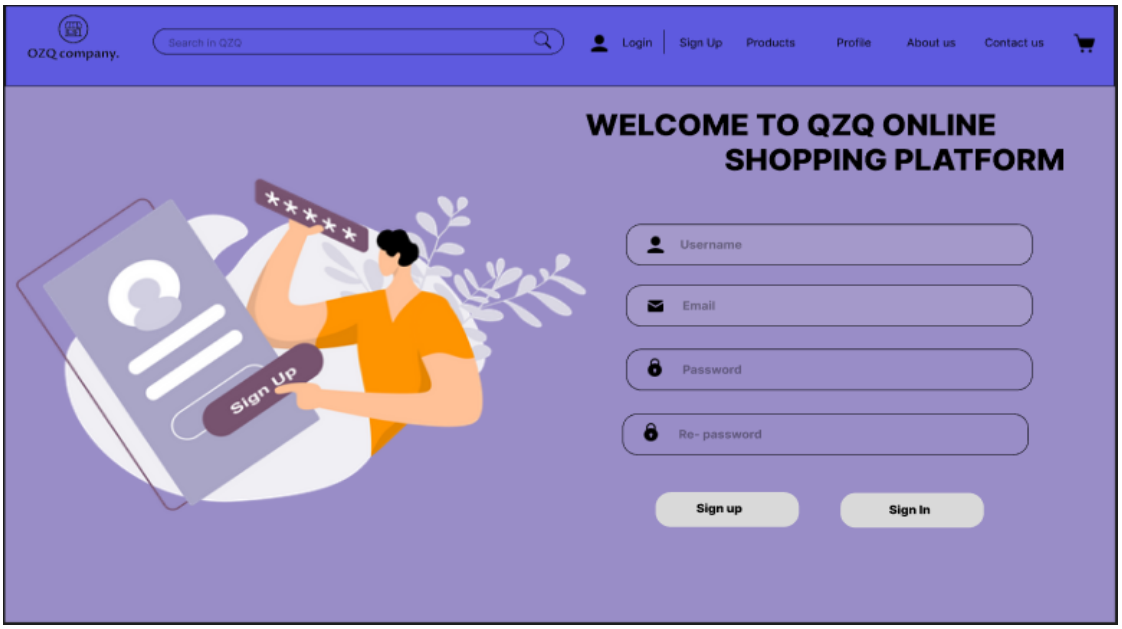


Figure 10: Sign Up Interface (Developed by Author)

User Profile Interface

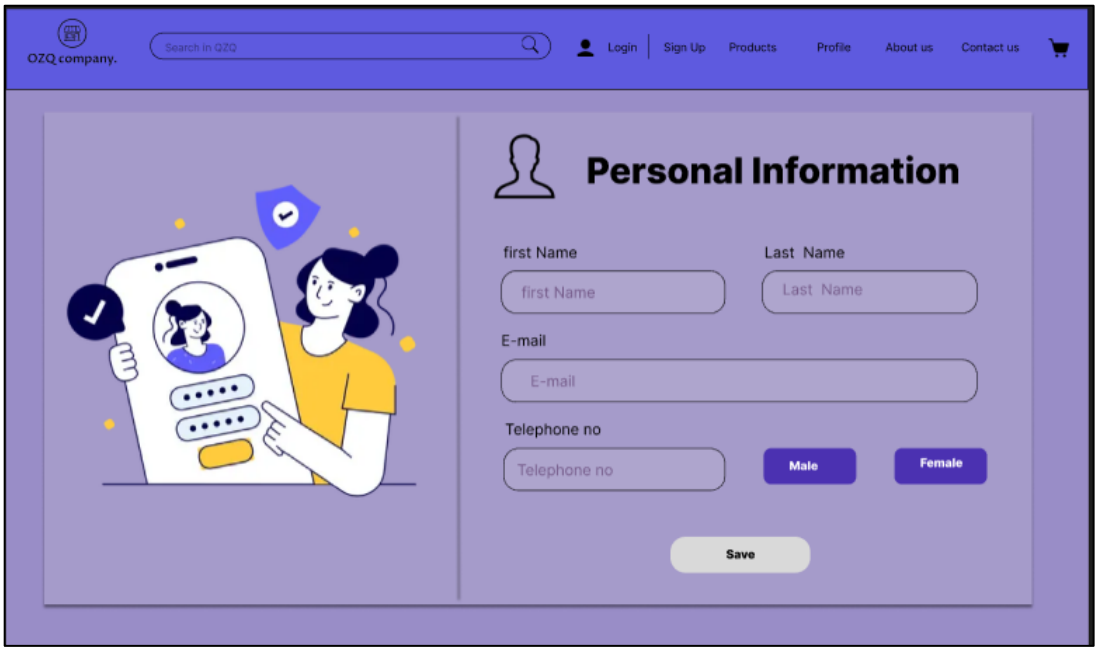


Figure 11: User Profile Interface (Developed by Author)

Customer Feedback Interface

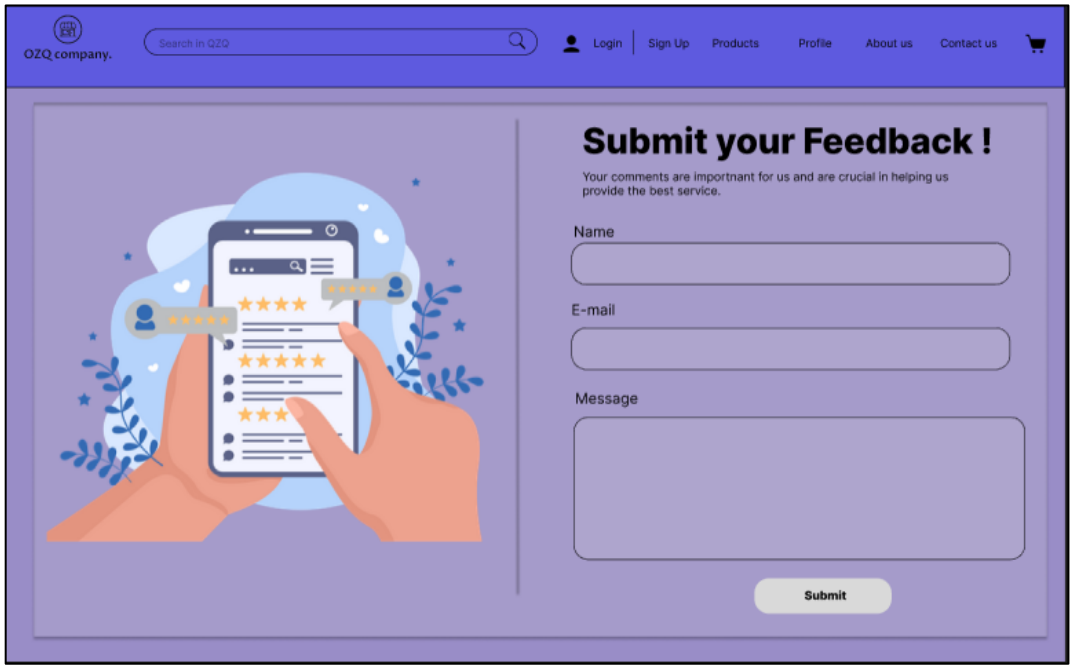


Figure 12: Customer Feedback Interface (Developed by Author)

Shopping store Interface

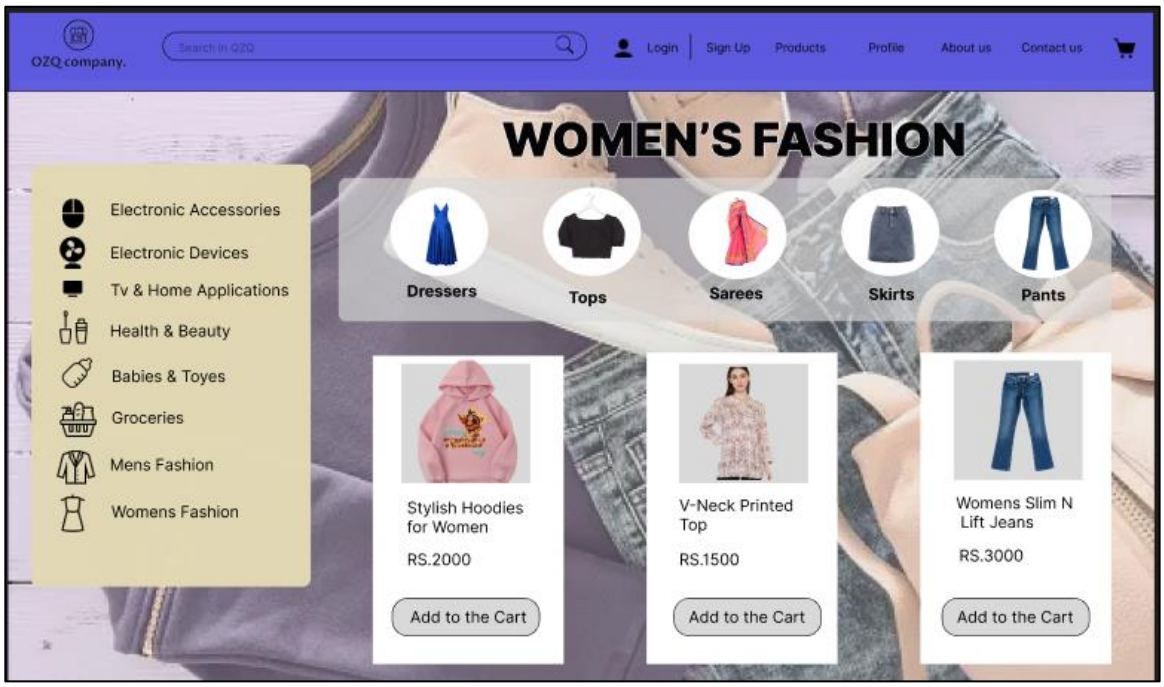


Figure 13 : Shopping store Interface (Developed by Author)

Shopping Cart Interface

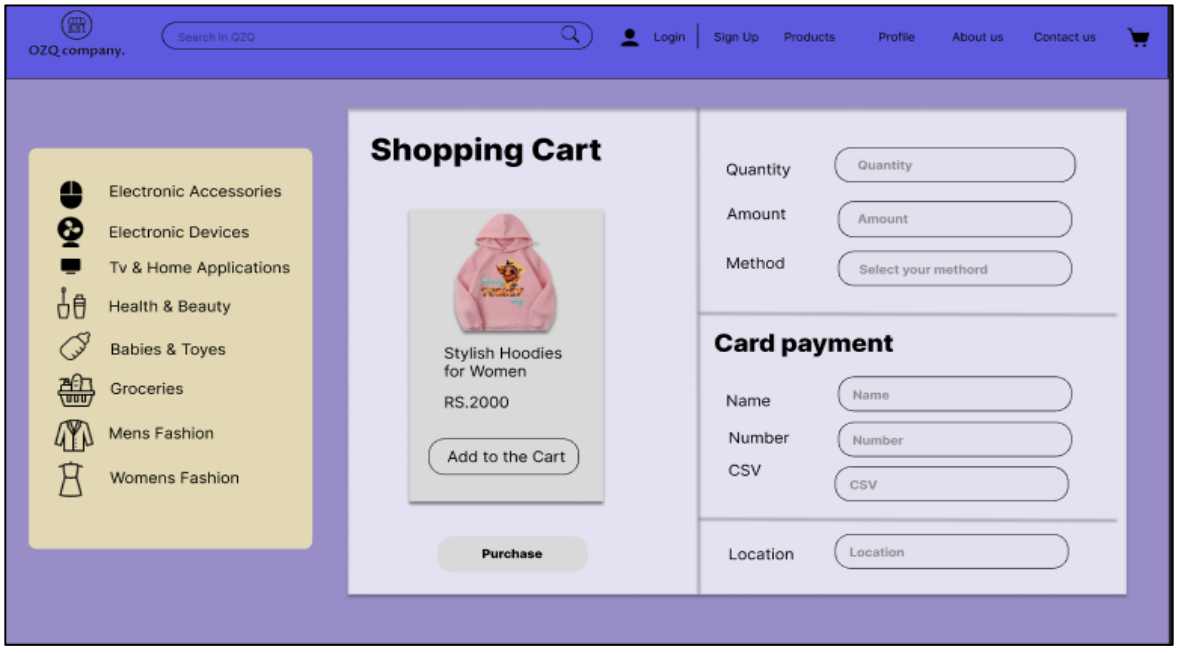


Figure 14 : Shopping cart Interface (Developed by Author)

Basic Steps in Creating an ASP.NET Web API. Application

- Set Up Environment:

Ensure you have the necessary tools installed. Install Visual Studio, and if you're working with .NET Core, ensure you have the .NET Core SDK.

- Create a New Project:

Open Visual Studio.

Click on "Create a new project."

Choose "ASP.NET Web Application" as the project type.

Select "Web API" as the project template.

Specify project name, location, and solution name.

Click "Create."

- Design the API Structure:

Identify the endpoints (routes) for your API.

Plan the HTTP methods (GET, POST, PUT, DELETE) for each endpoint.

Decide on the data models (DTOs) and controllers needed.

- Create Controllers:

Right-click on the "Controllers" folder in your project.

Select "Add" -> "Controller."

Choose "API Controller - Empty."

Name the controller and click "Add."

- Define Routes:

Use attributes like [RoutePrefix] or [Route] to define routes for the API endpoints.

Decorate controller actions with [HttpGet], [HttpPost], [HttpPut], or [HttpDelete] attributes.

Implement Actions:

Write methods within the controllers to handle API requests.

Implement logic to perform CRUD operations or any other business logic required.

- Handle Requests and Responses:

Use model binding to handle incoming requests.

Serialize objects into the desired response format (JSON/XML).

- Test Your API:

Use tools like Postman or Swagger to test your API endpoints.

Check that each endpoint functions as intended, returns the correct data, and handles errors properly.

- Implement Security (if needed)

Implement authentication and authorization mechanisms if your API requires security.

Use authentication middleware, JWT, OAuth, or other methods.

- Documentation:

Consider using Swagger/OpenAPI to document your API. This makes it easier for other developers to understand and consume your API.

- Debug and Refine:

Test thoroughly and debug any issues.

Refactor and optimize your code as necessary.

- Deployment:

Choose a hosting platform (Azure, AWS, IIS, etc.) and deploy your ASP.NET Web API application.

User can identify how to do the basic steps with below pictures

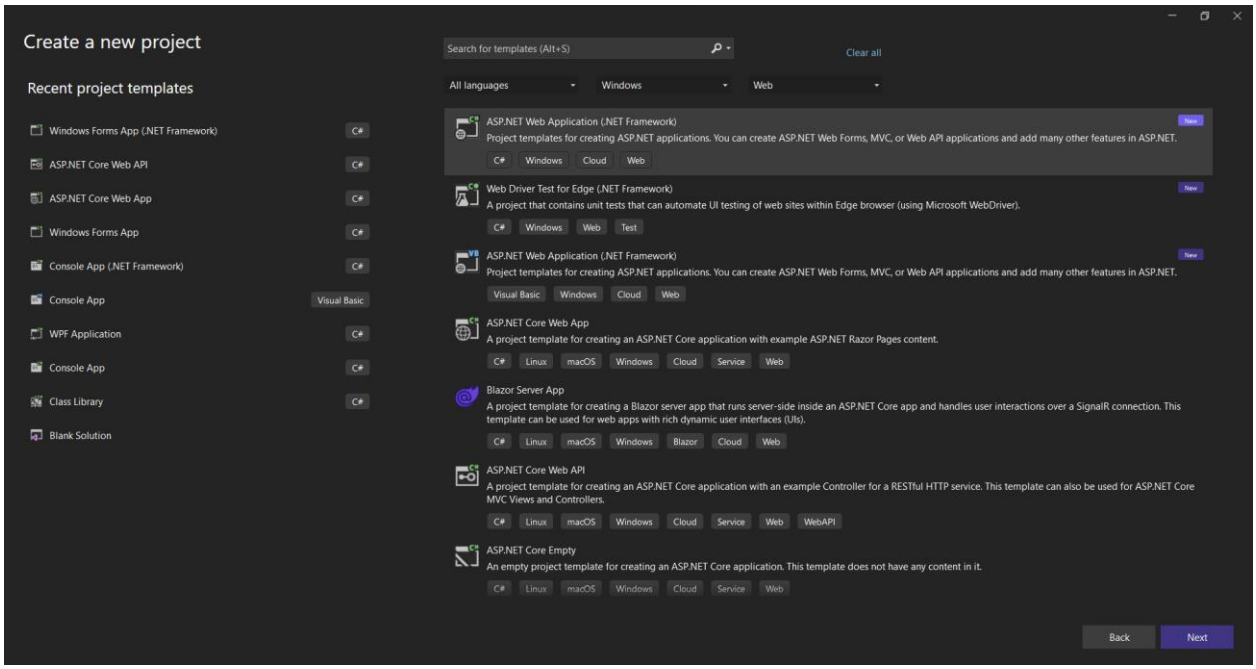


Figure 15: Basic steps to create an ASP.NET Web APLI Application

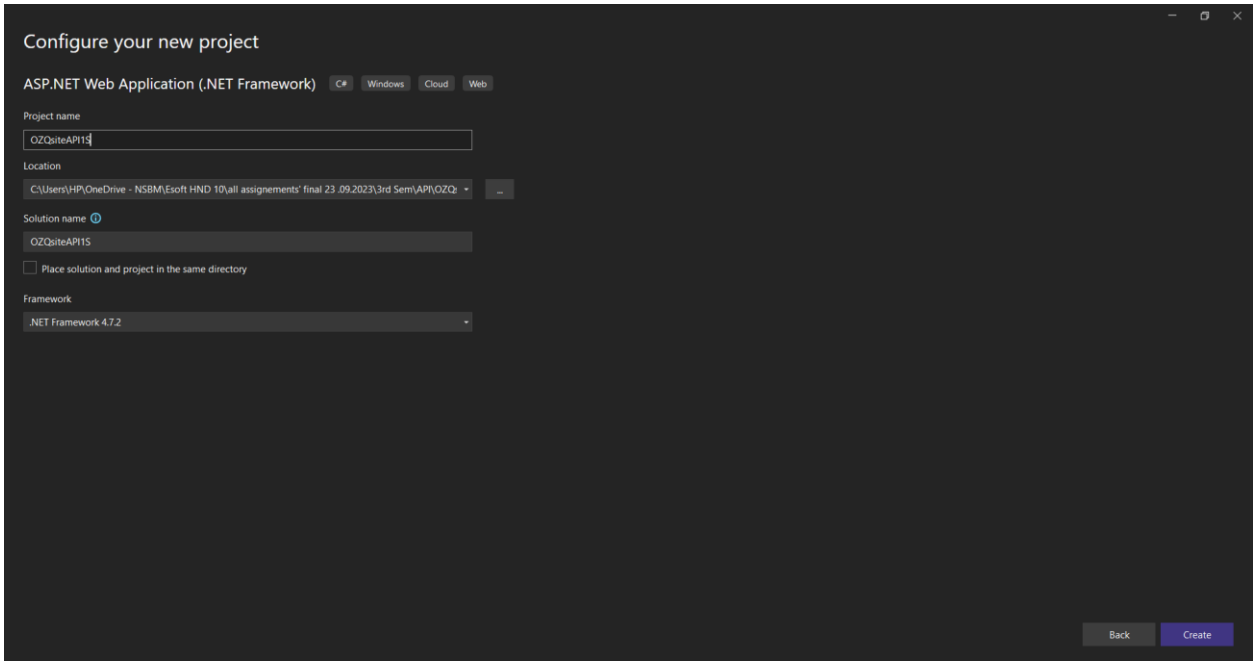


Figure 16 : Basic steps to create an ASP.NET Web APLI Application

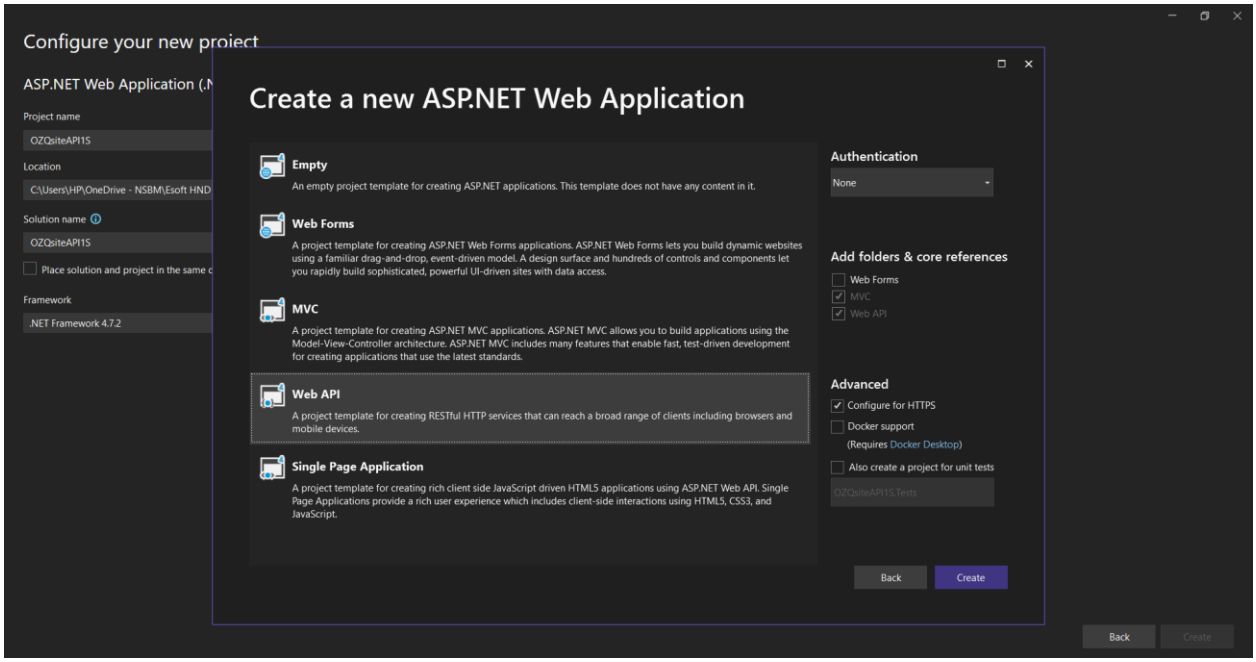


Figure 17 : Basic steps to create an ASP.NET Web APLI Application

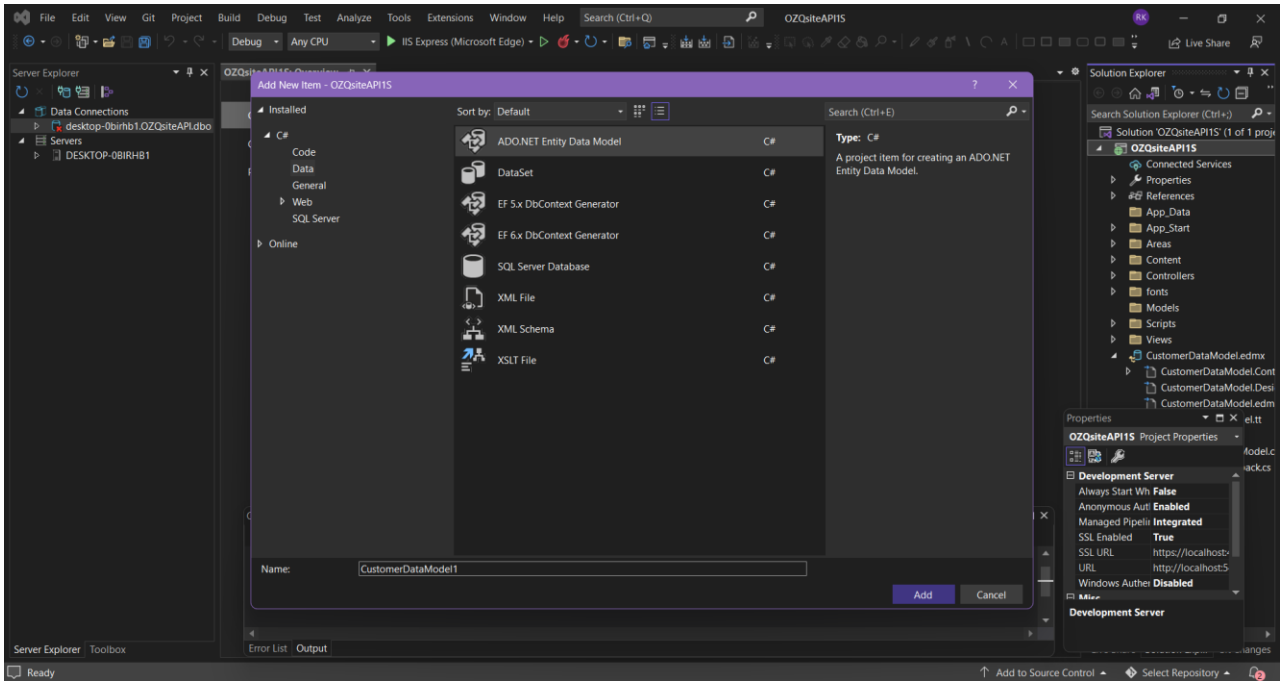


Figure 18 : Basic steps to create an ASP.NET Web APLI Application

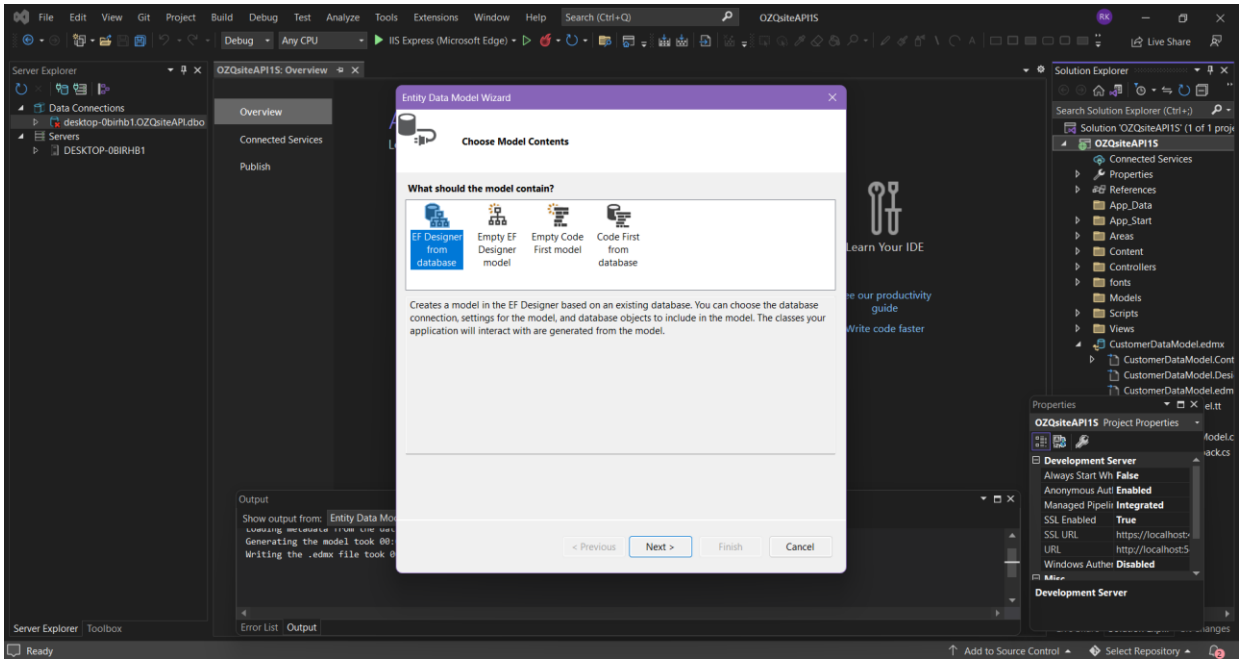


Figure 19: Basic steps to create an ASP.NET Web APLI Application

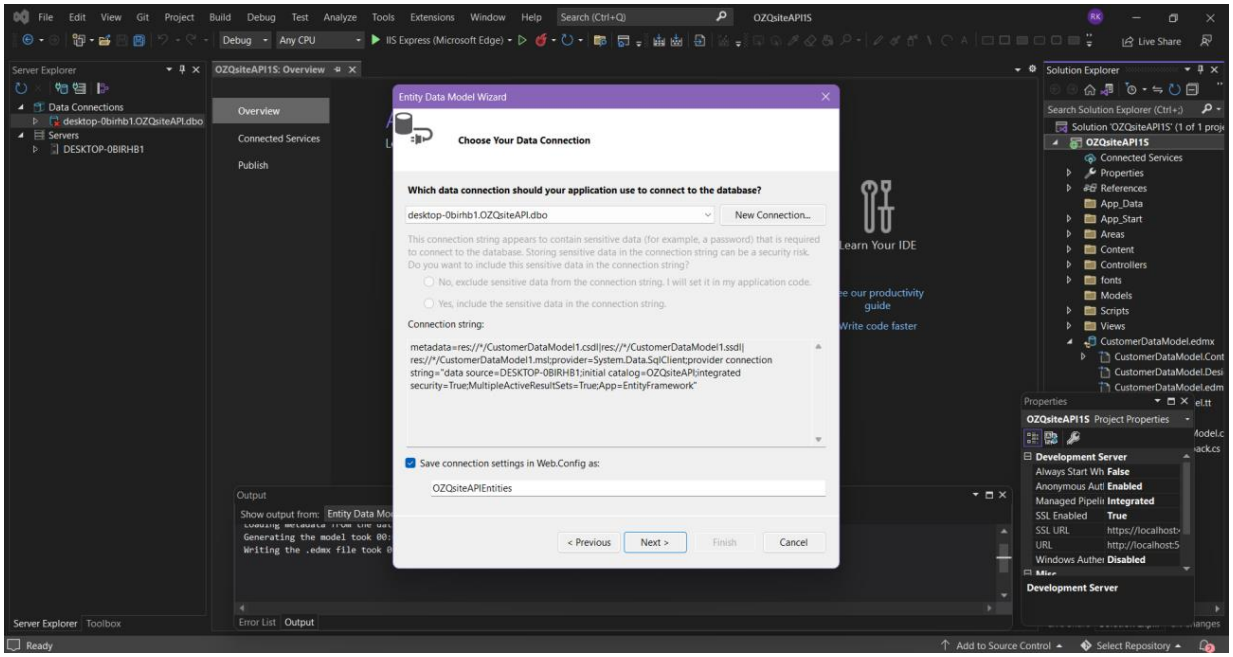


Figure 20: Basic steps to create an ASP.NET Web APLI Application

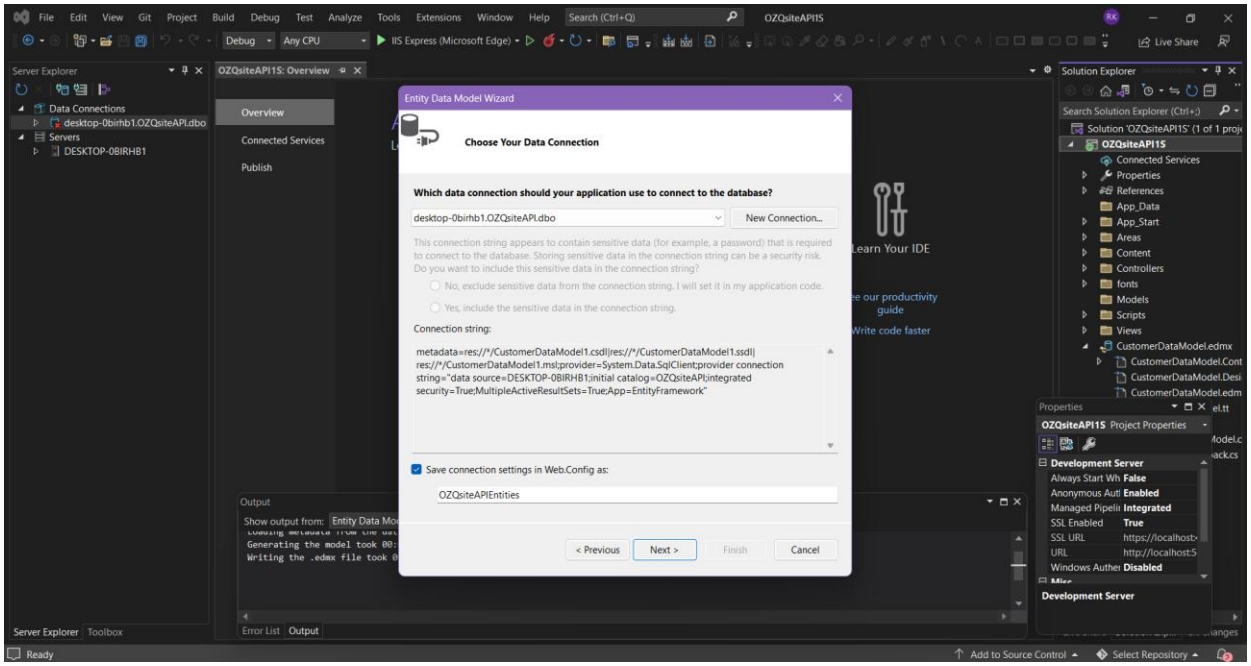


Figure 21: Basic steps to create an ASP.NET Web APLI Application

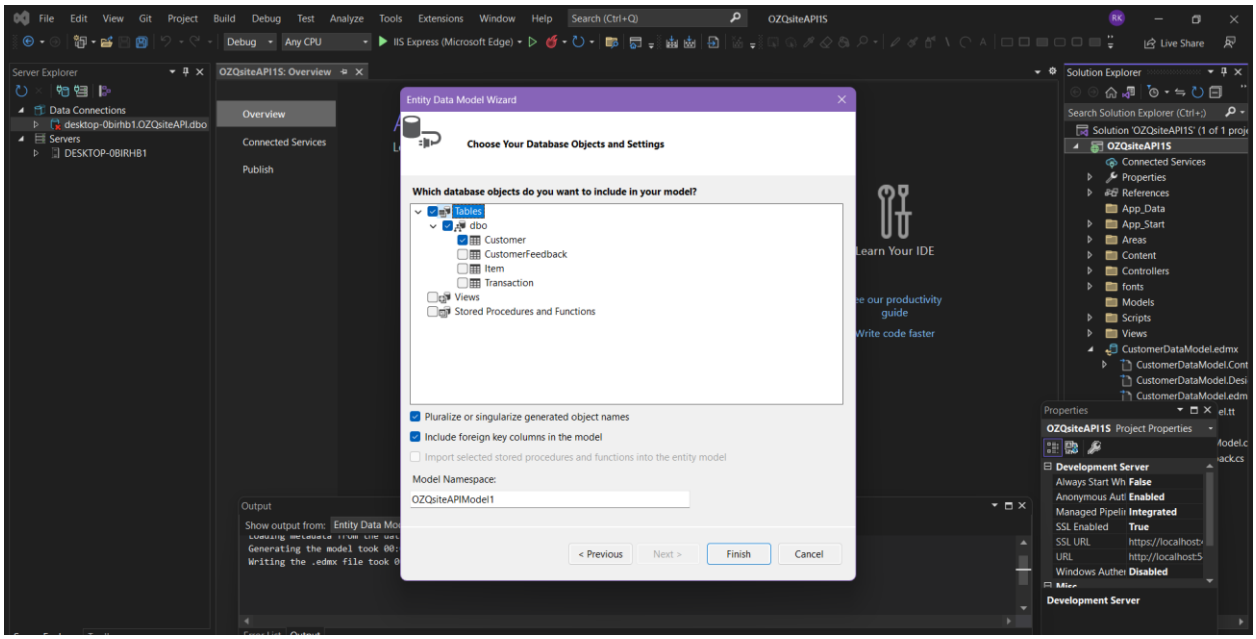


Figure 22: Basic steps to create an ASP.NET Web APLI Application

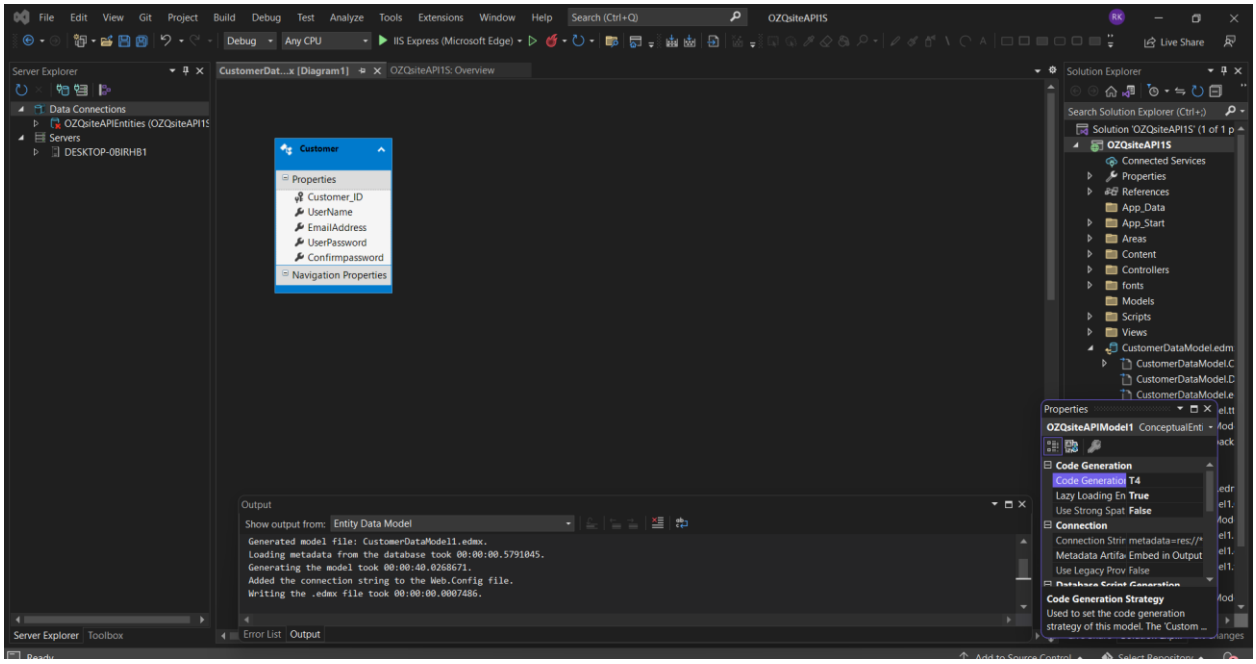


Figure 23: Basic steps to create an ASP.NET Web APLI Application

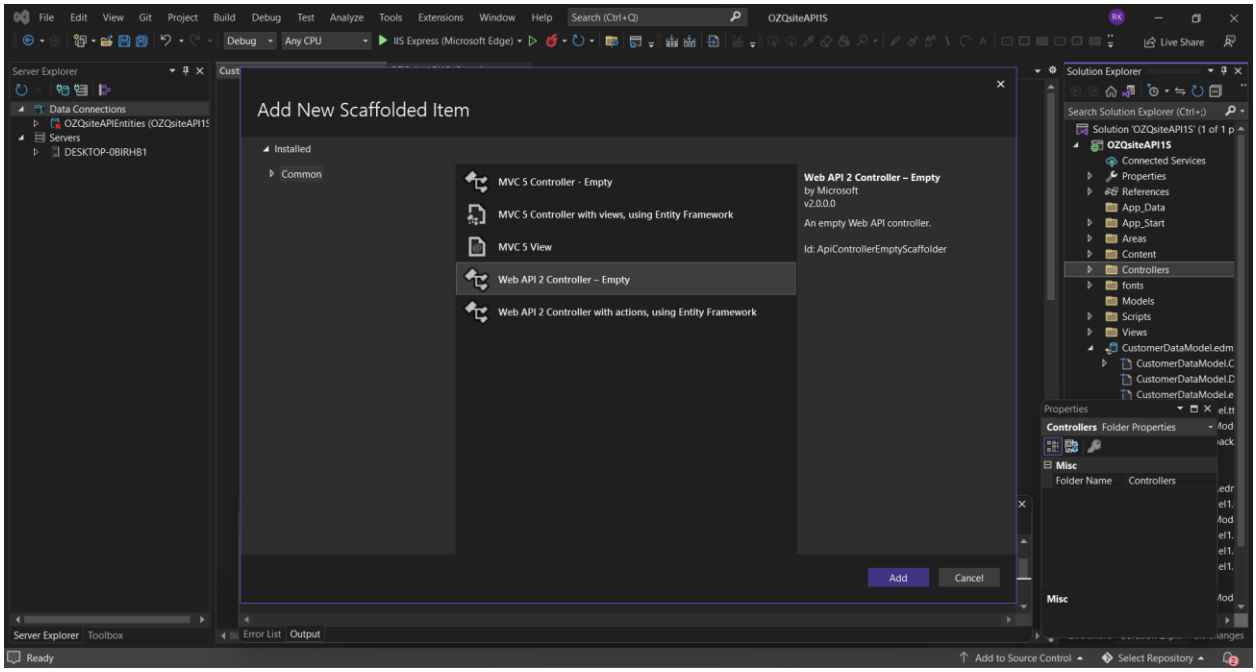


Figure 24: Basic steps to create an ASP.NET Web APLI Application

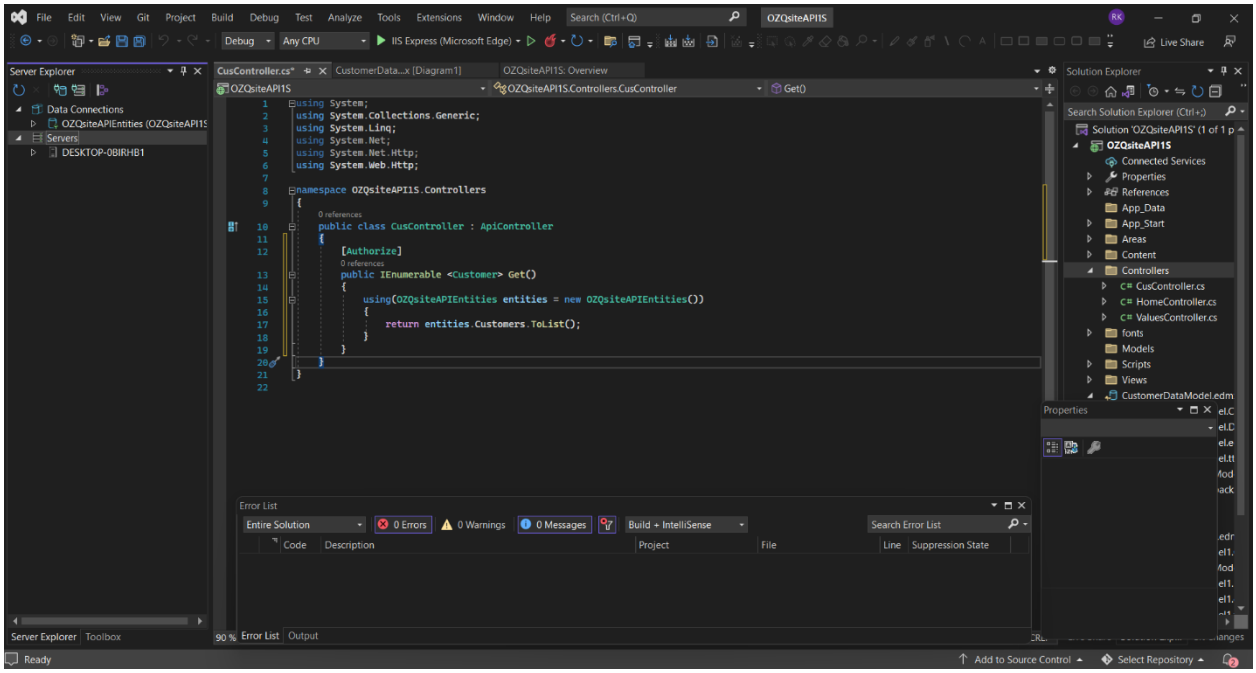


Figure 25 : Basic steps to create an ASP.NET Web APLI Application

Evidences for developing OZQ online shopping system using tools and technologies.

Evidences of HTML5 codes use to develop interfaces of OZQ online shopping system

Welcome Page Interface

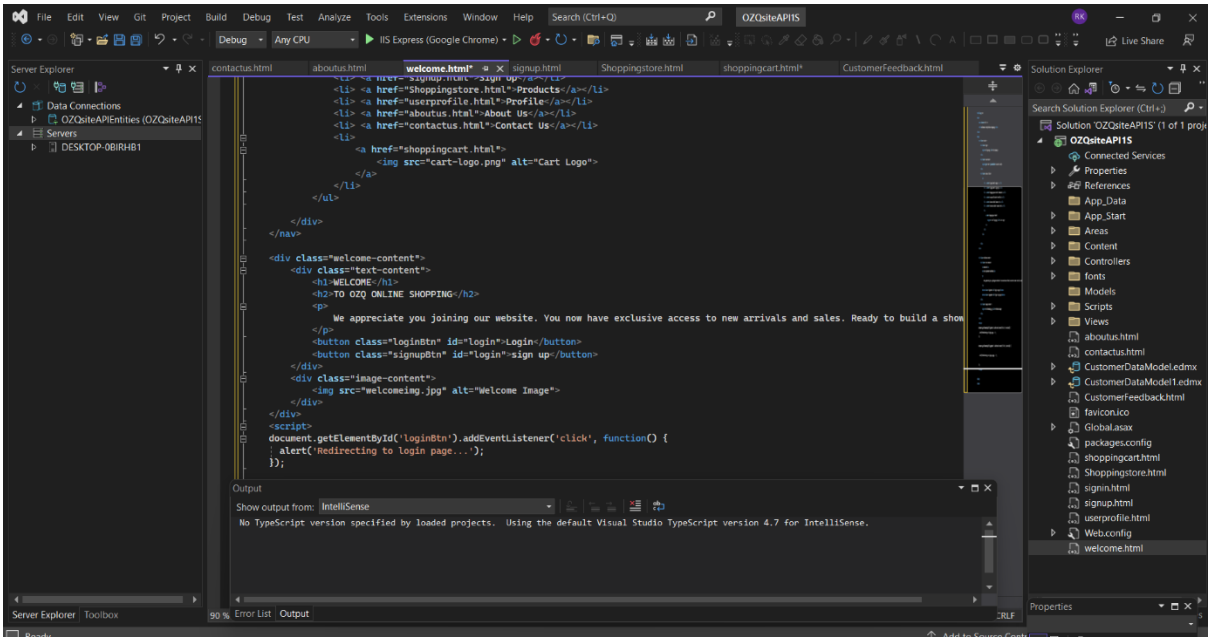


Figure 26 : Evidences for developing OZQ online shopping system

Shopping Cart Interface

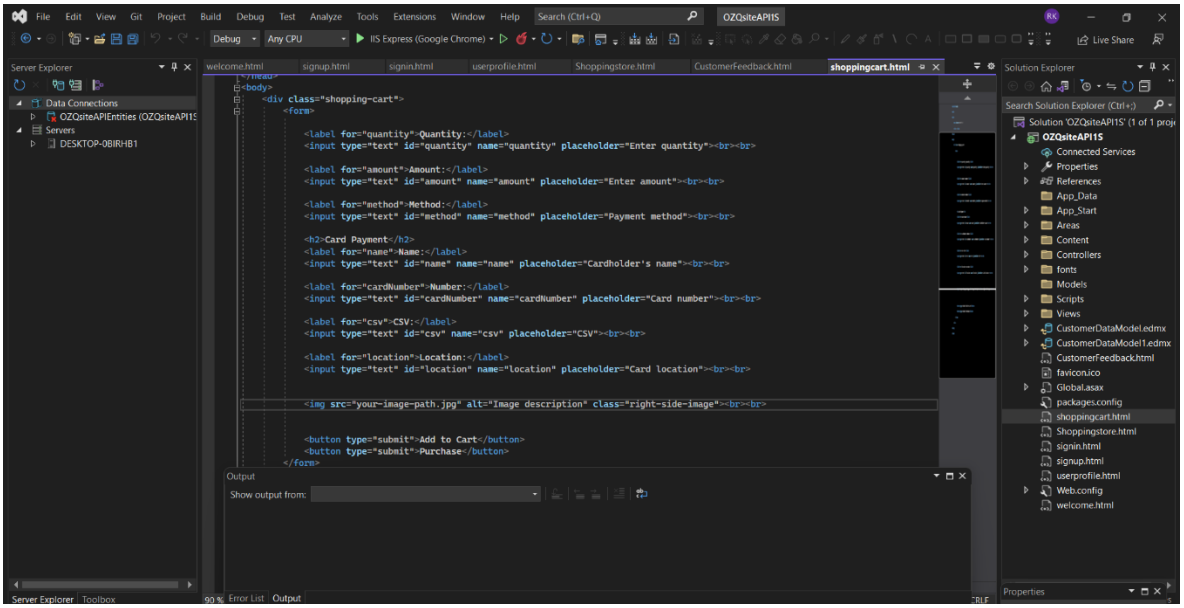


Figure 27: Evidences for developing OZQ online shopping system

Customer Feedback Interface

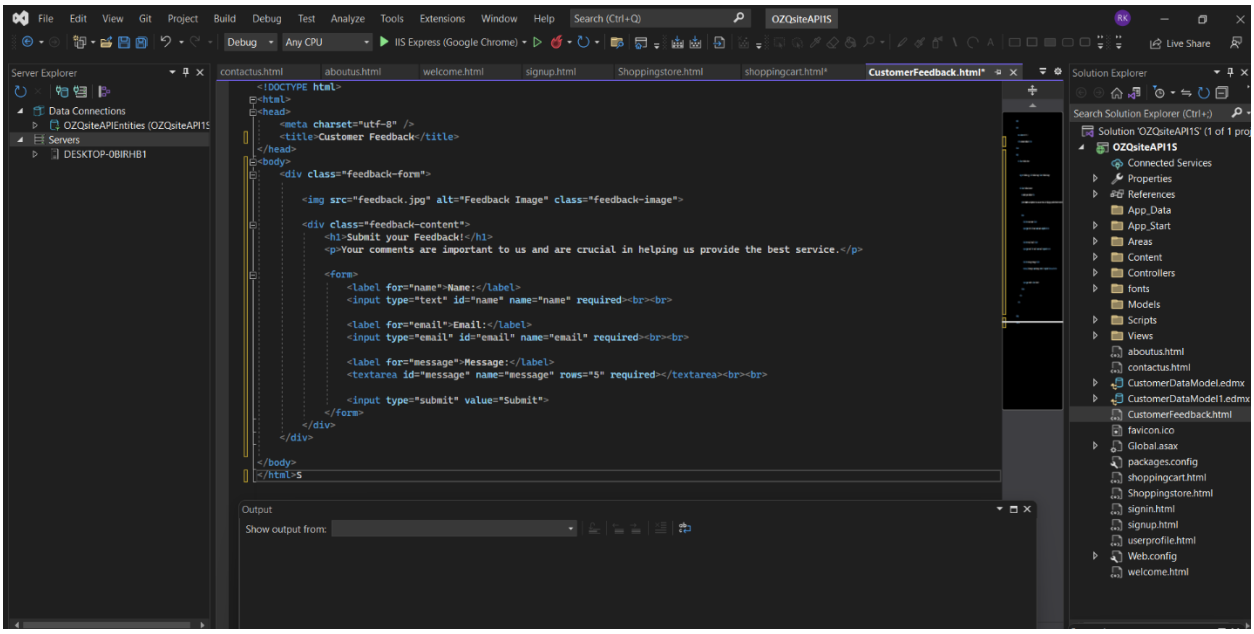


Figure 28: Evidences for developing OZQ online shopping system

Navigation Bar Section

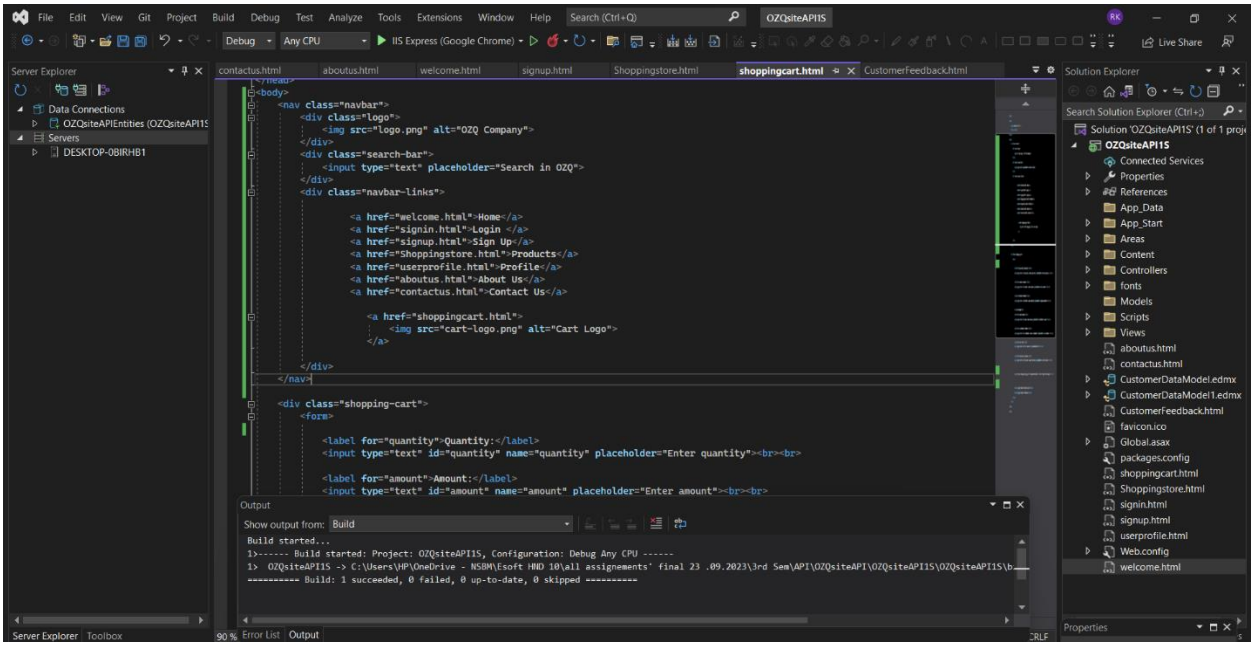


Figure 29: Evidences for developing OZQ online shopping system

Sign in Interface

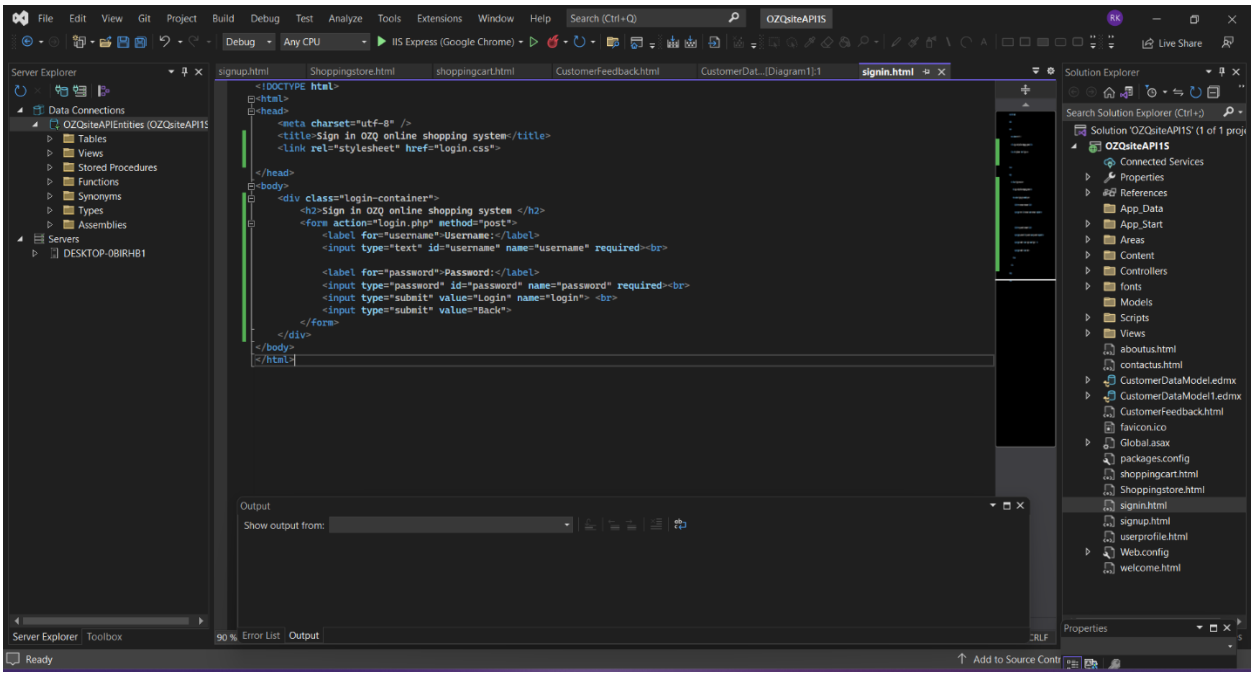


Figure 30: Evidences for developing OZQ online shopping system

Sign up Interface

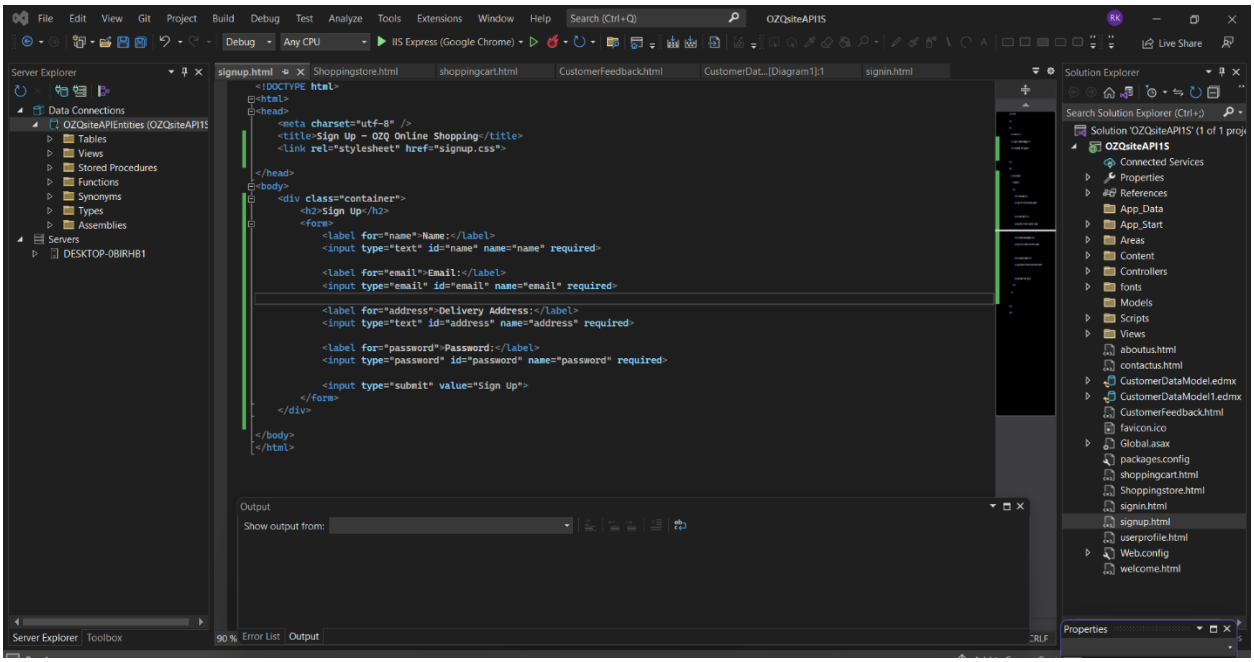


Figure 31: Evidences for developing OZQ online shopping syst

User profile Interface

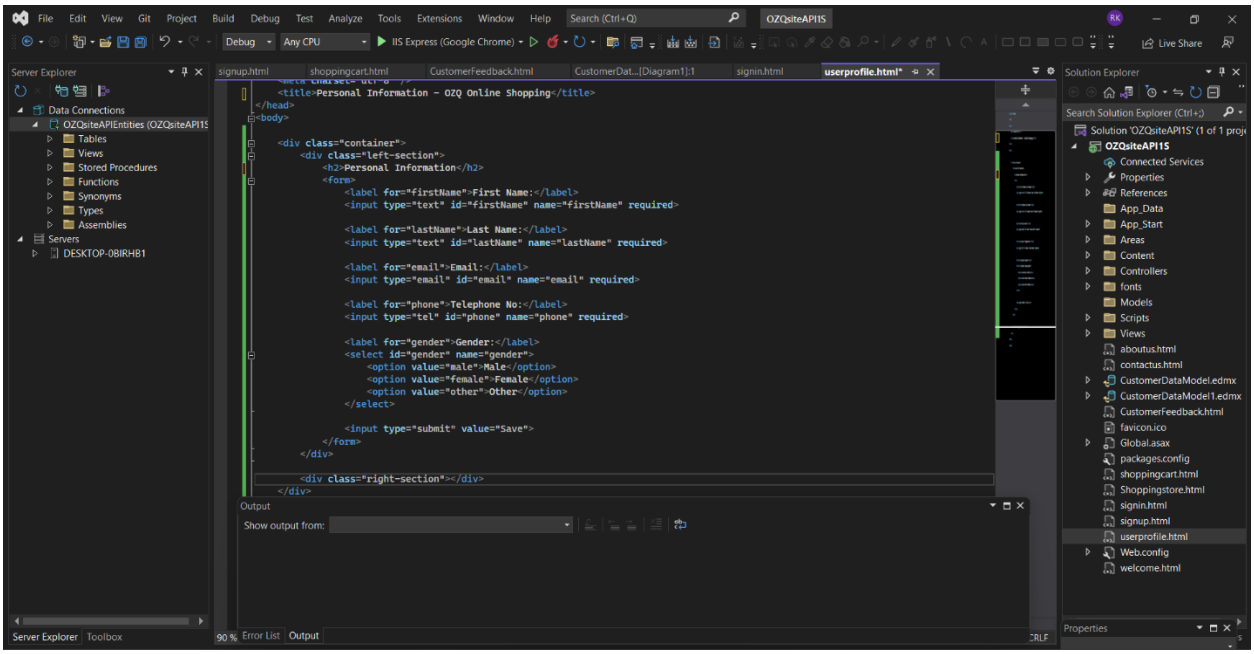


Figure 32: Evidences for developing OZQ online shopping system

Shopping store Interface

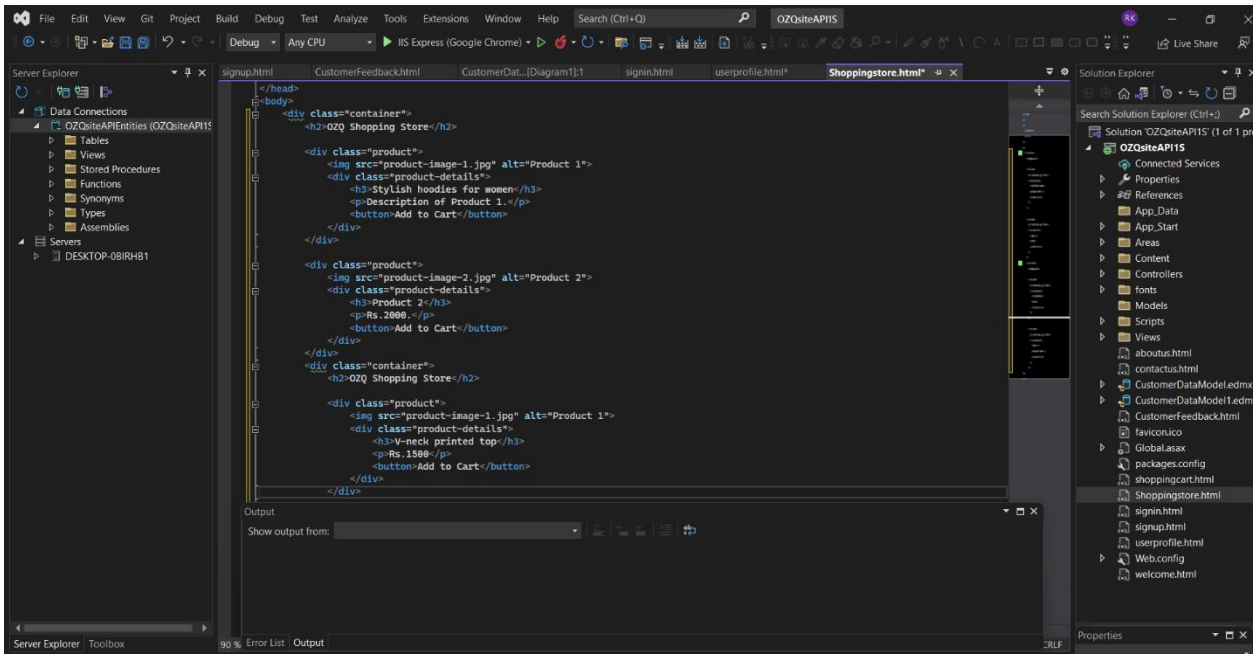


Figure 33: Evidences for developing OZQ online shopping system

Evidences of CSS3 codes use to develop interfaces of OZQ online shopping system

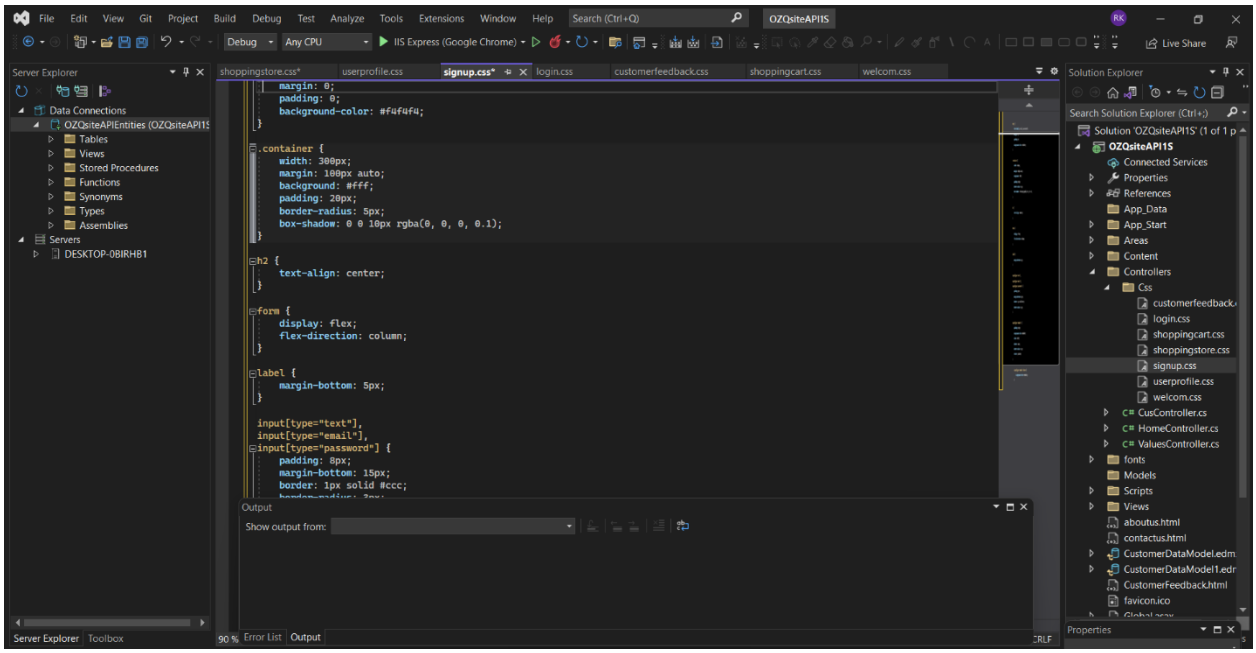


Figure 34 : Evidences for developing OZQ online shopping system

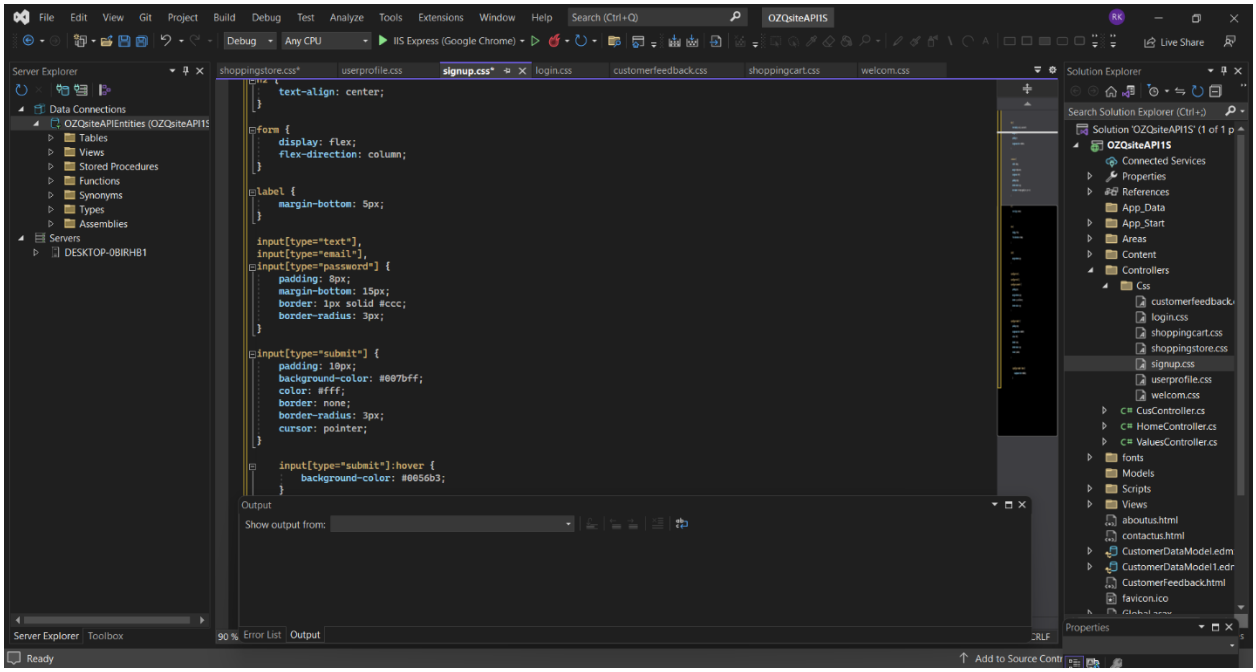


Figure 35 : Evidences for developing OZQ online shopping system

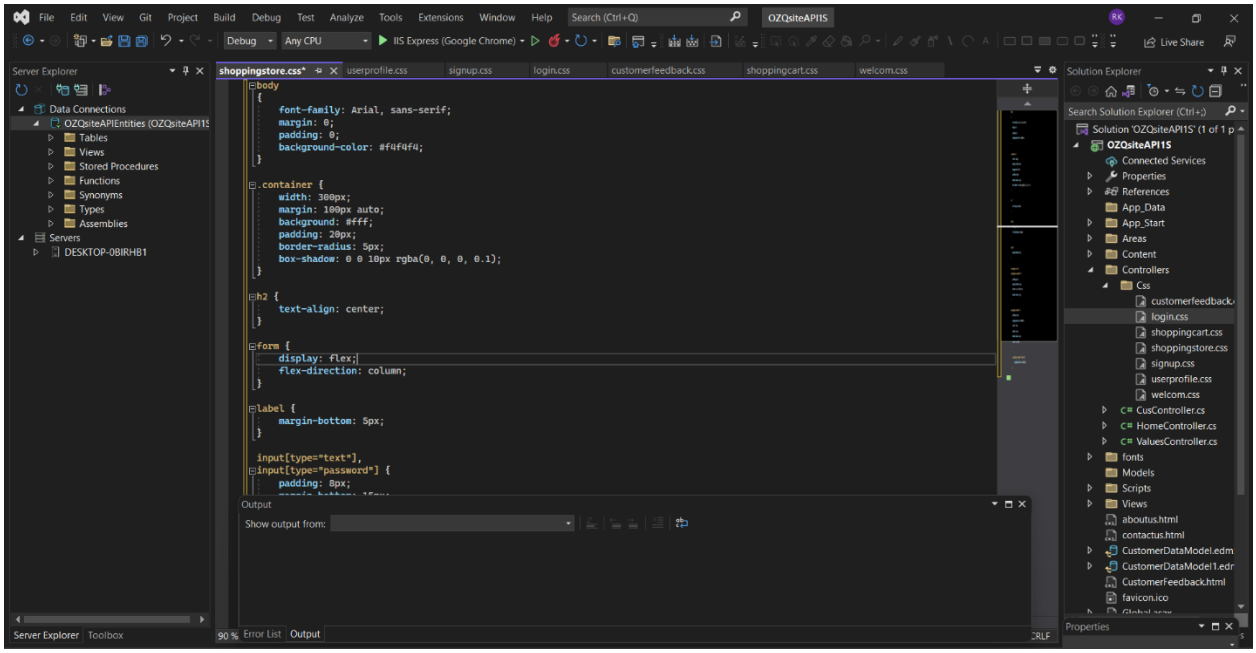


Figure 36 : Evidences for developing OZQ online shopping system

Evidence of jQuery codes used to develop user interfaces

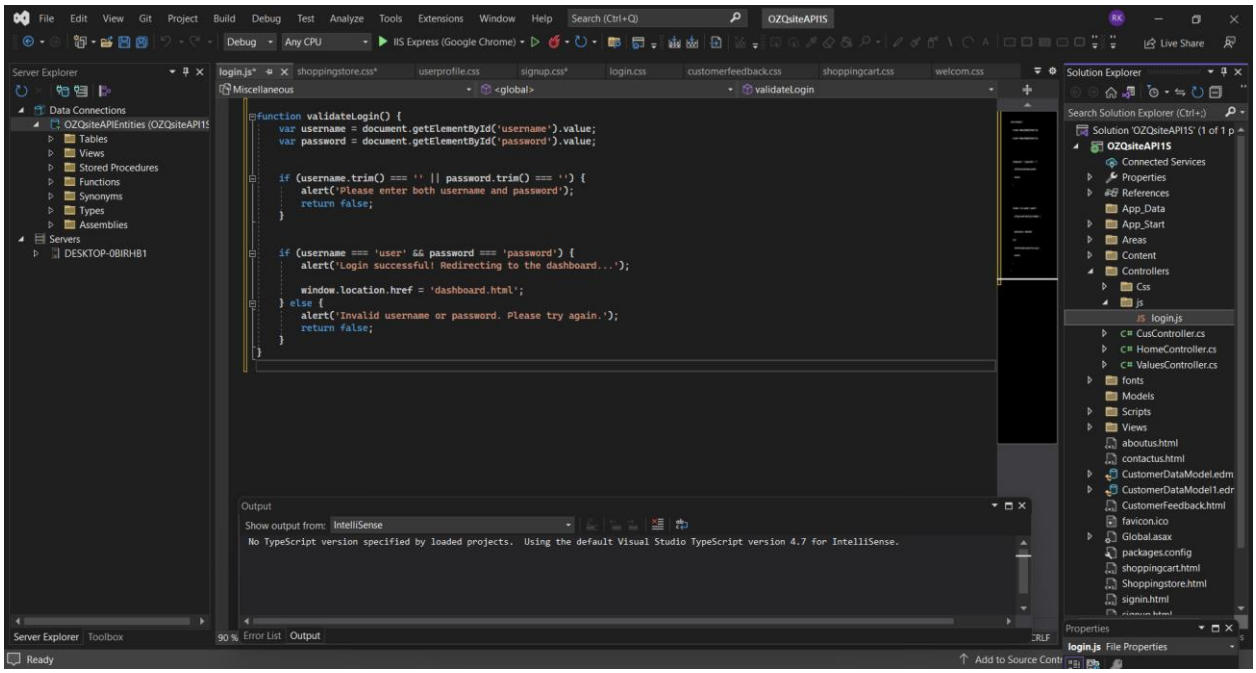


Figure 37 : Evidences for developing OZQ online shopping system

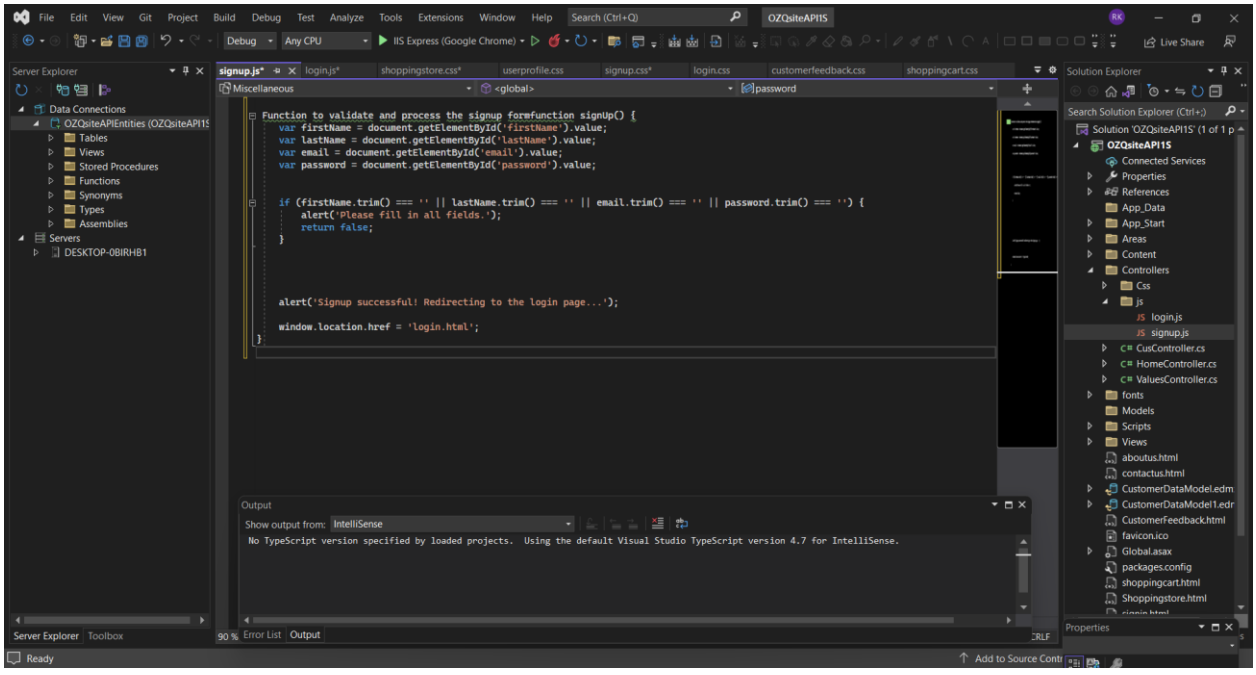


Figure 38 : Evidences for developing OZQ online shopping system

OZO Web Application API Database Data Model Diagram

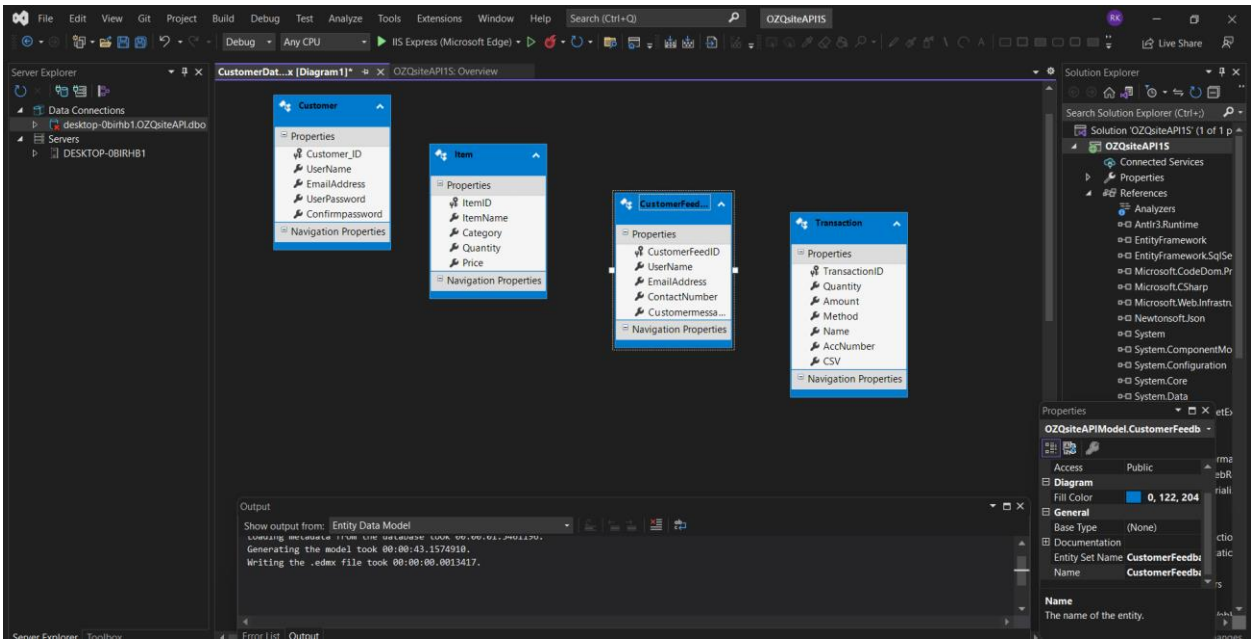


Figure 39 : Evidences for developing OZQ online shopping system

Test Plan of OQZ web based Online Shopping Store System

Test Case ID	Test Scenario	Test Steps	Expected Result	Actual Result	Status
TC001	Valid User Registration	1. Click on 'Register'. 2. Fill valid user details. 3. Submit the registration form.	User should be registered successfully.	User is registered successfully.	Pass
TC002	Invalid User Registration	1. Click on 'Register'. 2. Fill invalid/empty	Error messages for each invalid	Proper error messages are displayed.	Pass

		user details. 3. Submit the form.	field displayed.		
TC003	Successful User Login	1. Enter valid login credentials. 2. Click on 'Login'.	User should be logged in and directed to the dashboard.	User is logged in and directed to the dashboard.	Pass
TC004	Failed User Login	1. Enter invalid login credentials. 2. Click on 'Login'.	Error message for failed login displayed.	Proper error message for failed login is displayed.	Pass
TC005	Adding Items to Cart	1. Browse products and click 'Add to Cart' for selected items. 2. View the shopping cart.	Selected items should be added to the shopping cart.	Selected items are added to the shopping cart.	Pass
TC006	Removing Items from Cart	1. Open the shopping cart. 2. Remove an item. 3. Verify the updated cart.	The item should be removed from the shopping cart.	The item is successfully removed from the cart.	Pass
TC007	Checkout Process	1. Proceed to checkout from the shopping cart. 2. Input billing information. 3. Place the order.	Order details and confirmation message displayed.	Order is successfully placed with confirmation.	Pass

TC008	Update User Profile	1. Navigate to 'User Profile'. 2. Update user details. 3. Save the changes.	User details should be updated and saved.	User details are updated and saved successfully.	Pass
TC009	Providing Feedback	1. Navigate to 'Feedback'. 2. Submit feedback.	Feedback should be successfully submitted.	Feedback is successfully submitted.	Pass
TC010	System Performance Test	Simulate heavy load on the system during peak hours.	System should respond within acceptable time frames.	System responses are within acceptable limits.	Pass
TC011	Usability Testing	Evaluate the user interface intuitiveness by user feedback and tests.	Users should find the interface intuitive and easy to navigate.	Users find the interface intuitive and easy.	Pass
TC012	Security Testing	Test security measures like encryption, secure connections, etc.	System should pass security tests without vulnerabilities.	System passes security tests without issues.	Pass

Table 3 : Test Plan of OQZ web based Online Shopping Store System

Whitebox Testing

White-box testing, sometimes referred to as structural or glass-box testing, is a software testing technique that evaluates the internal organization of an application's source code. White-box testing looks at the internal logic, structure, and flow of the code, in contrast to black-box testing, which concentrates on functionality without knowledge of internal code.

To verify that every component operates as intended, this testing method entails dissecting the application's source code, architectural layout, and internal operations. In order to verify the accuracy of each code segment, path, branch, and data flow within the program, testers build test cases based on their understanding of the software's architecture and implementation.

In software development, white-box testing is frequently carried out at the unit, integration, and system levels. It assists in locating logical mistakes, finding errors in the code, making sure all code paths are checked, and confirming that the program operates as intended.

White-box testing guarantees sufficient code coverage and aids in code quality optimization, software reliability improvement, and overall system performance through methods like code coverage analysis, path testing, control flow testing, and data flow testing.

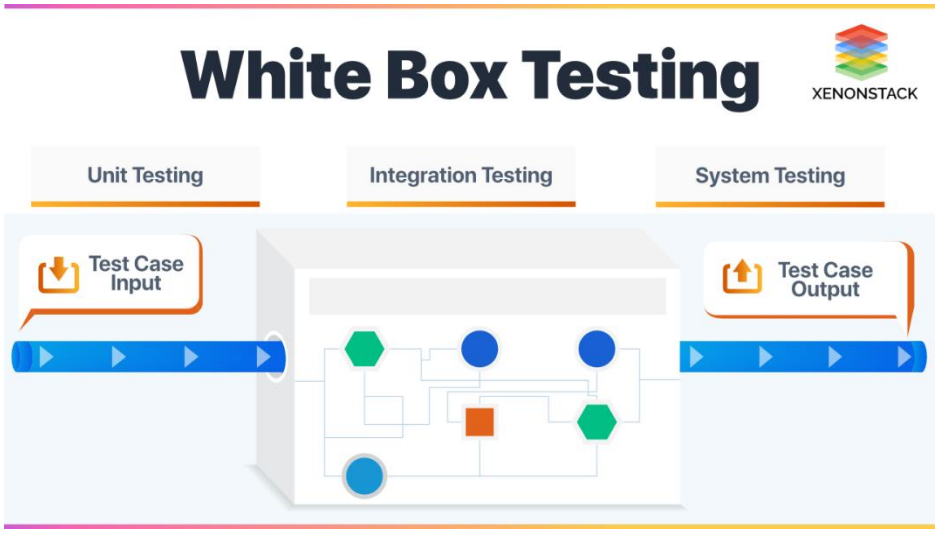


Figure 40:white Box testin

Errors found when continuing the white box testing and the way they were fixed

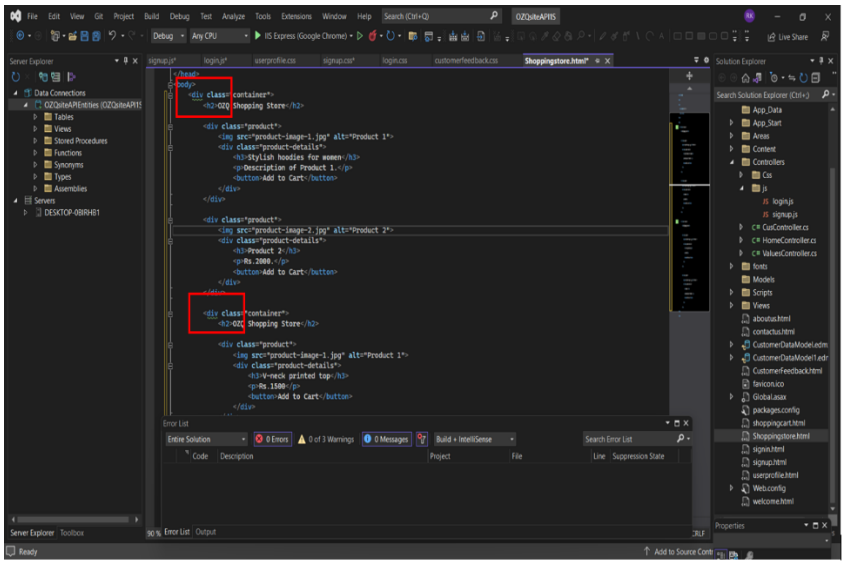
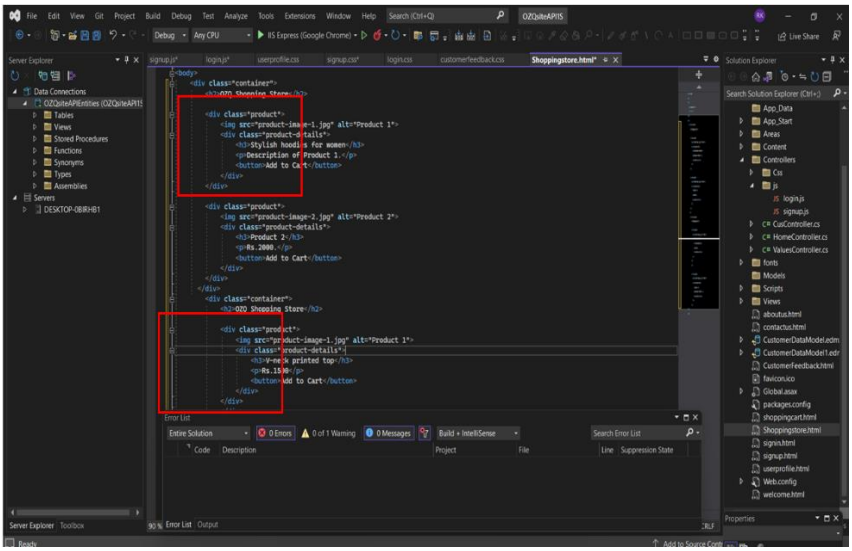
Test Case	01
Test Type	White Box Testing
Language	Html5
Errors count	2
Error Code	
Solution code	

Table 4 : Whitebox Testing

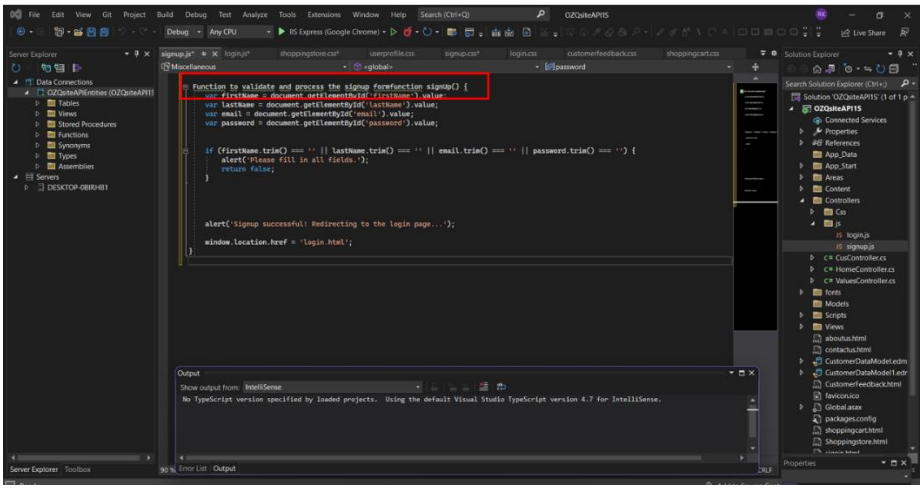
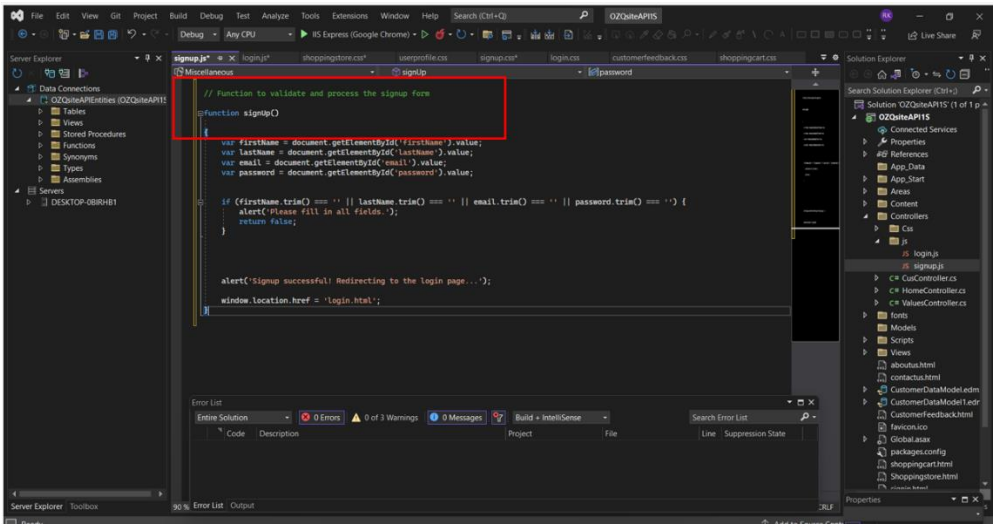
Test Case	02
Test Type	White Box Testing
Language	JavaScript
Errors count	1
Error Code	
Solution code	

Table 5 : Whitebox Testing

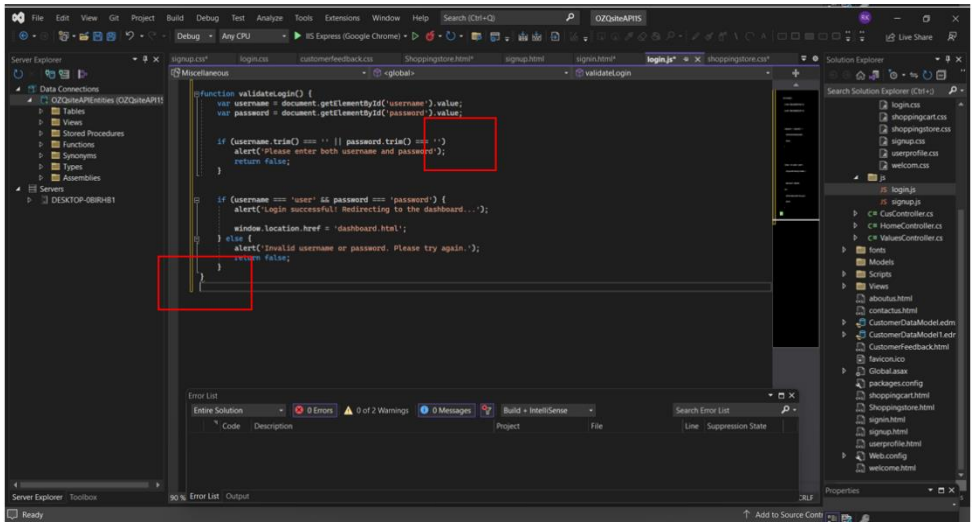
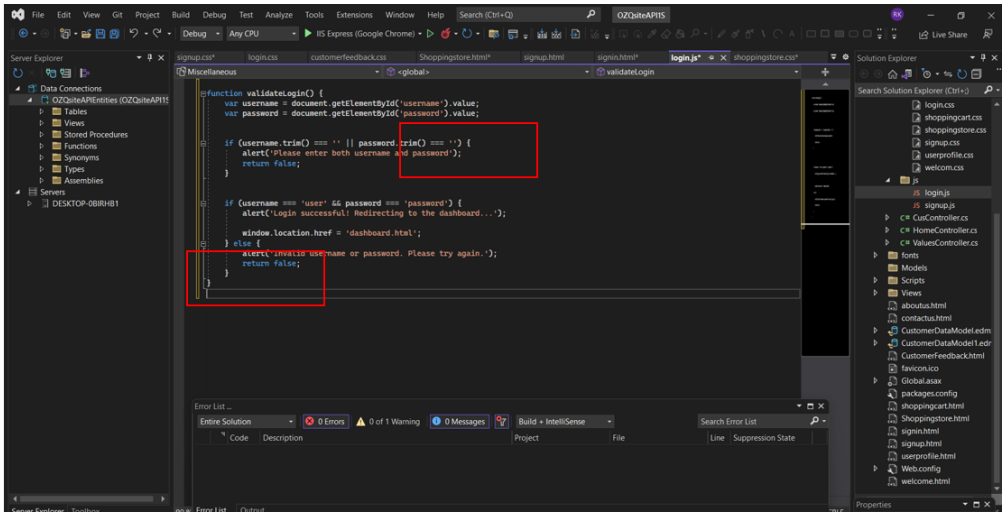
Test Case	03
Test Type	White Box Testing
Language	JavaScript
Errors count	2
Error Code	
Solution code	

Table 6 : Whitebox Testing

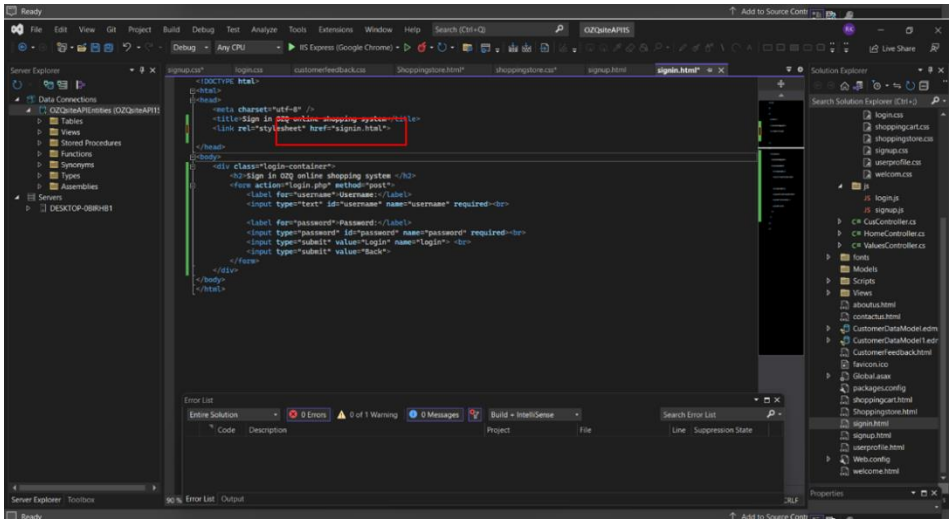
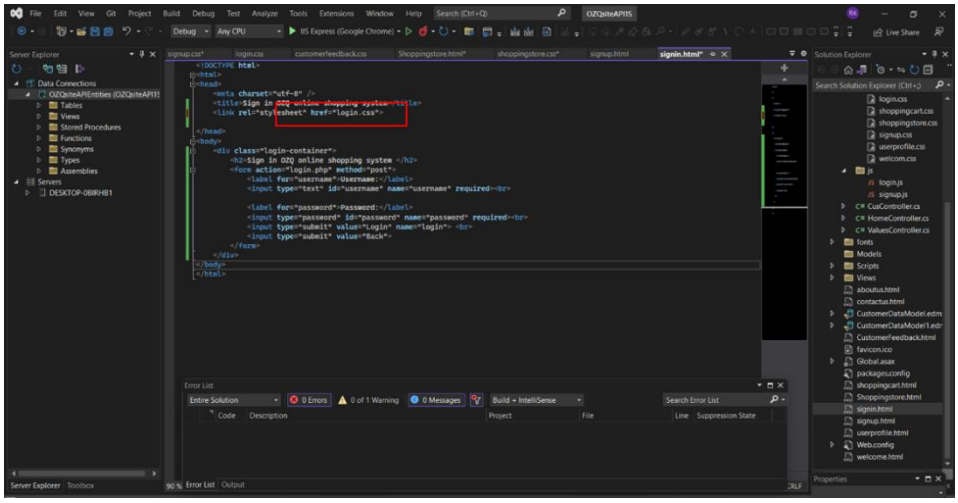
Test Case	04
Test Type	White Box Testing
Language	Html5
Errors count	1
Error Code	
Solution code	

Table 7 : Whitebox Testing

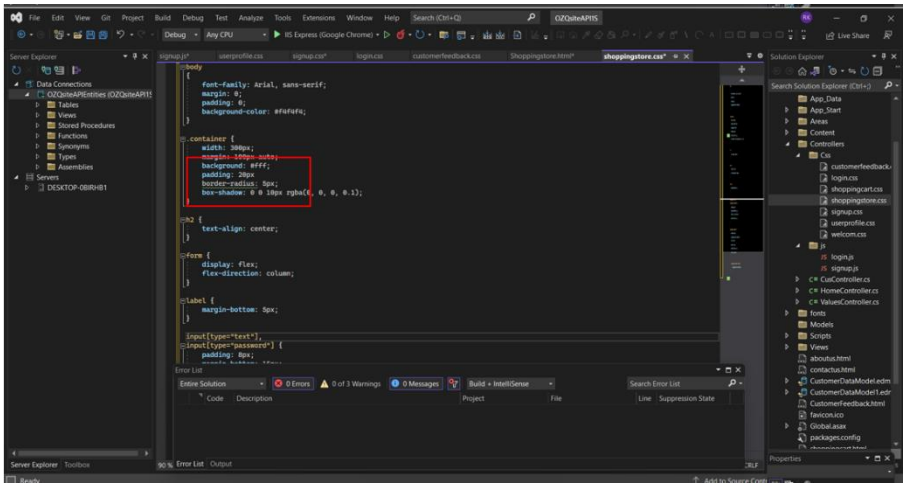
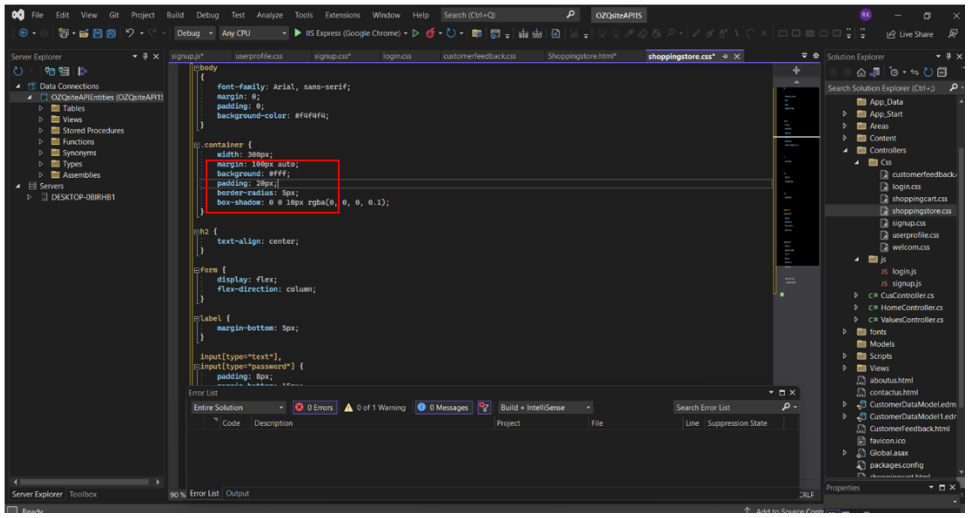
Test Case	05
Test Type	White Box Testing
Language	CSS3
Errors count	1
Error Code	
Solution code	

Table 8 : Whitebox Testing

Black Box Testing

Black box testing stands as a fundamental method in software testing, examining an application without delving into its internal code. This approach revolves around understanding the software's functionality based solely on its specifications and requirements. Testers approach the system as an external user, focusing on inputs and expected outputs, without access to the internal codebase or system architecture. By applying various test scenarios aligned with user expectations and system requirements, this method aims to validate whether the software meets specified criteria and functions as intended. It's a user-centric approach that ensures the application aligns with user needs and performs as expected without delving into the intricacies of its internal structure.

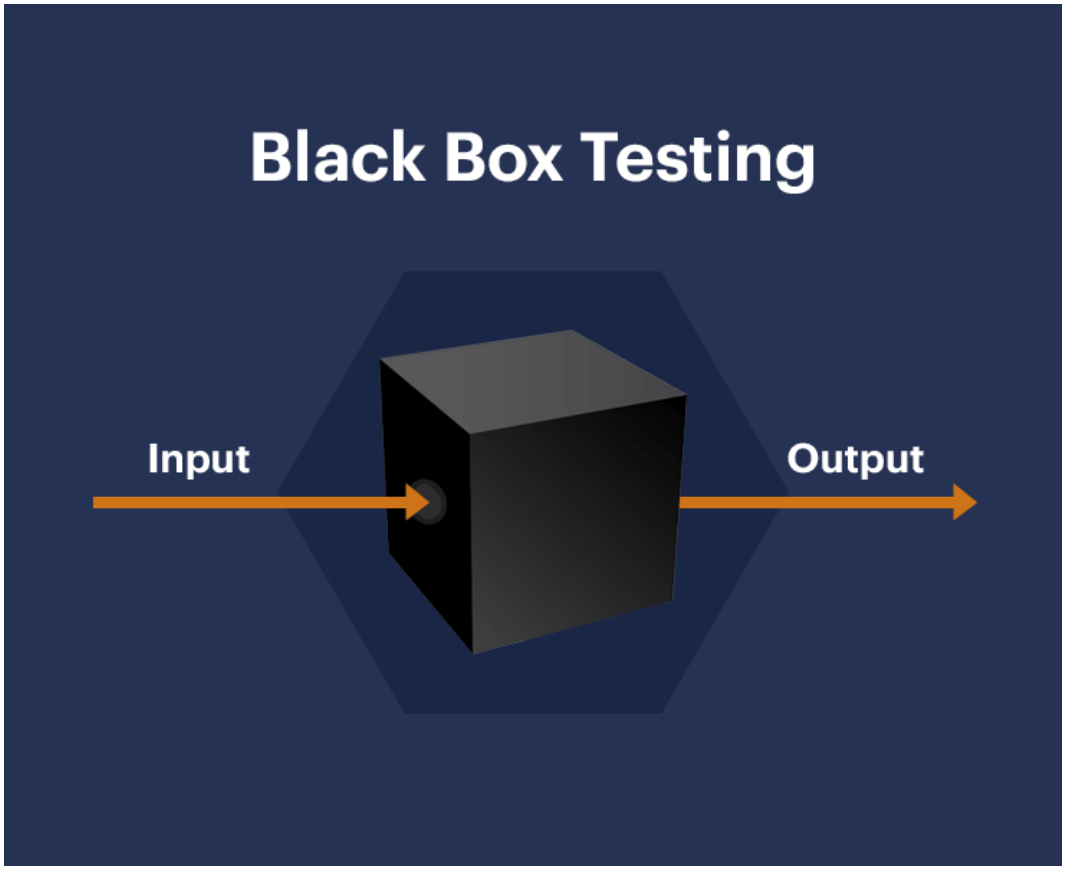


Figure 41 : Black box testing

Black Box Testing Result Evidence

Error 01

User Interface	Navigation Link	The interface that should take the user through	Status
Shopping store interface	Shopping Cart	Sign Up interface	Fail

Table 9 : Black box testing

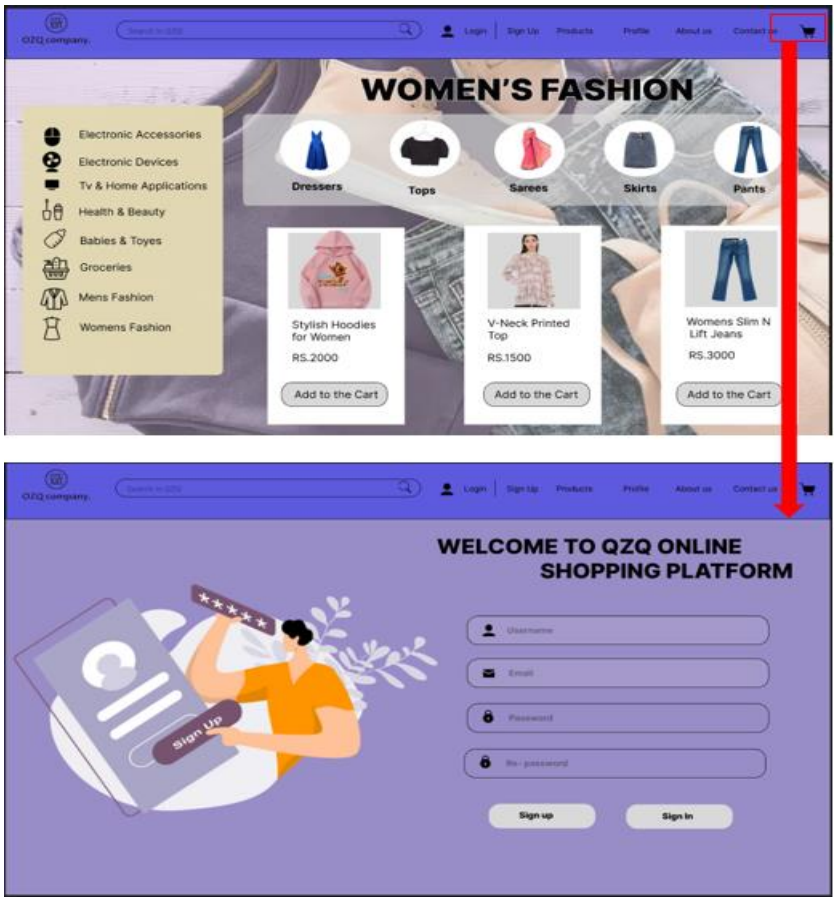


Figure 42 : Blackbox testing

Correction of Error 01

User Interface	Navigation Link	The interface that should take the user through	Status
Shopping store interface	Shopping Cart	User profile	sucess

Table 10 : Black box testing

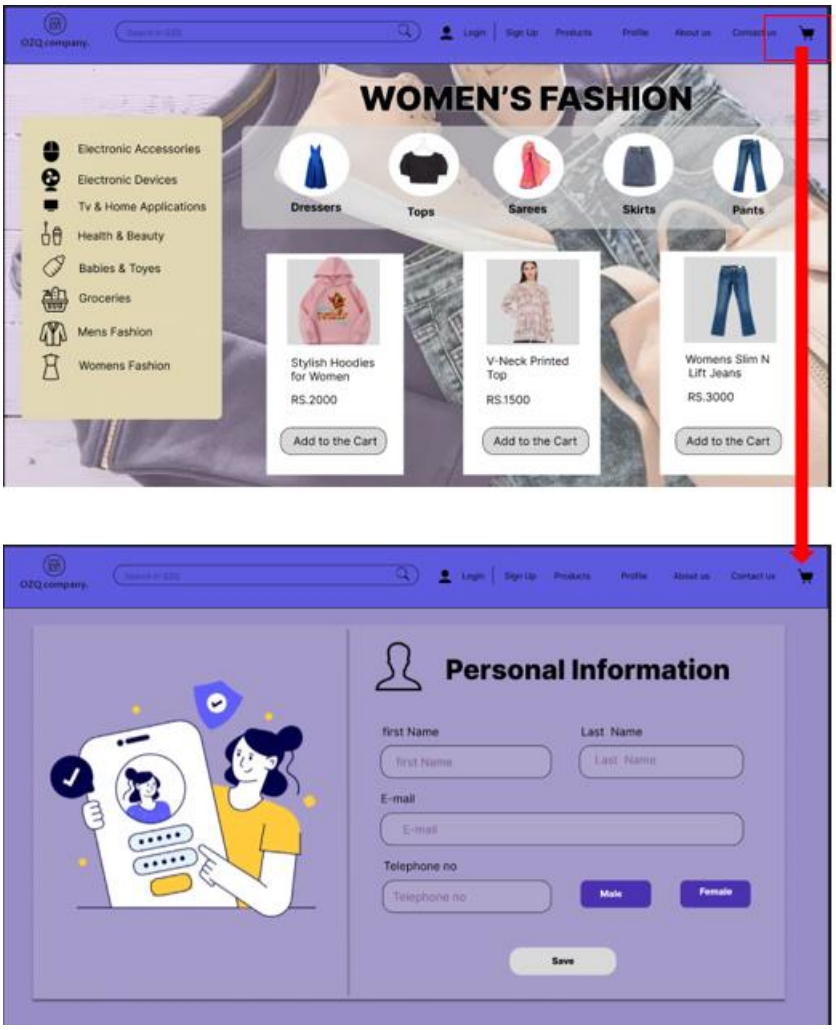


Figure 43 :Blackbox testing

Error 02

User Interface	Navigation Link	The interface that should take the user through	Status
Welcome Page	Login	Shopping store	Fail

Table 11: Black box Testing

Figure 44 : blackbox testing

Correction of Error 02

User Interface	Navigation Link	The interface that should take the user through	Status
Welcome Page	Login	Login interface	sucess

Table 12 : Black box Testing

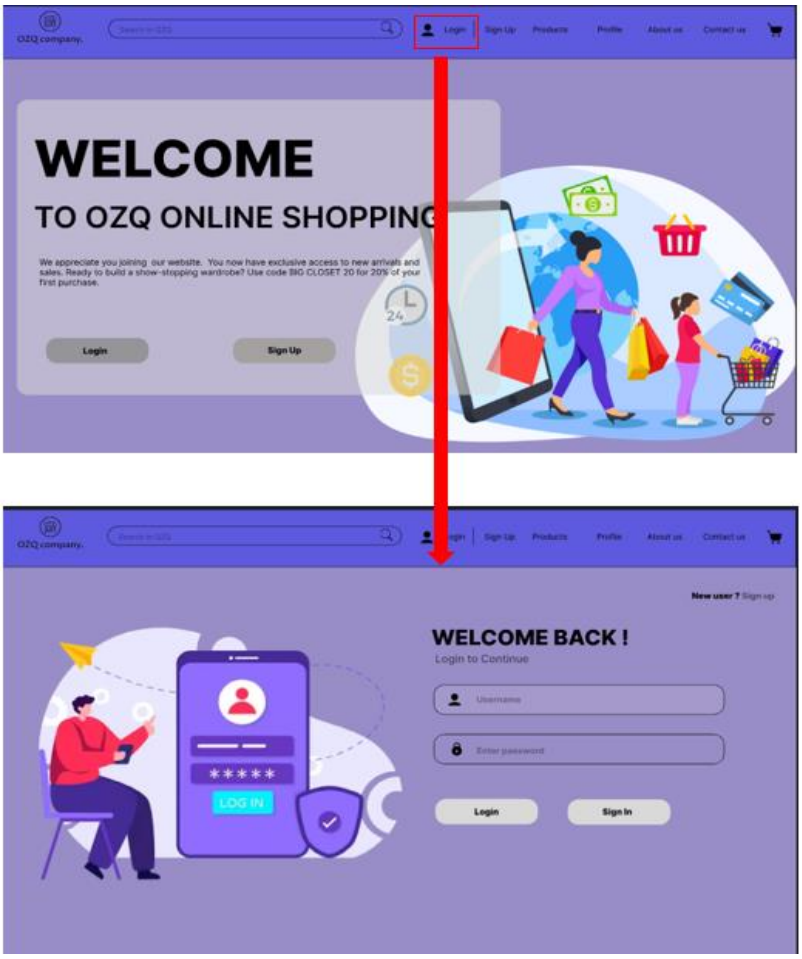


Figure 45 : blackbox testing

Error 03

User Interface	Navigation Link	The interface that should take the user through	Status
User profile	Shopping cart	Sign up	Fail

Table 13: Black box Testing

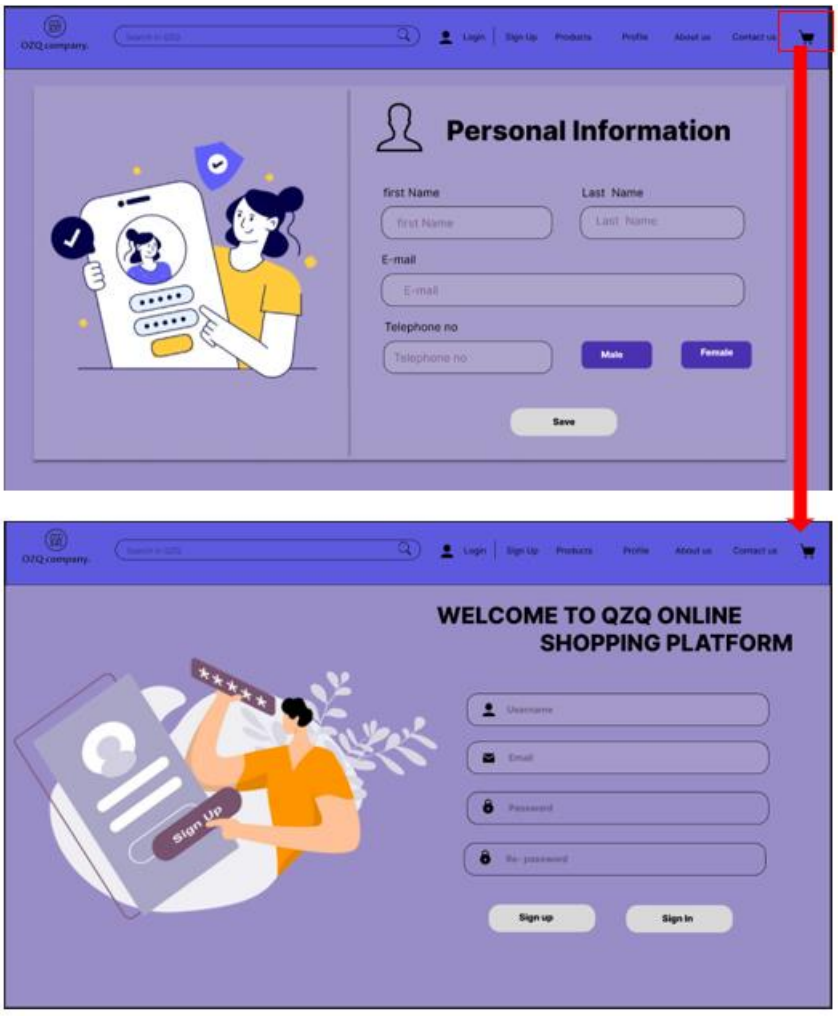


Figure 46 : Black box testing

Correction of Error 03

User Interface	Navigation Link	The interface that should take the user through	Status
User profile	Shopping cart	Shopping cart	sucess

Table 14: Black box Testing

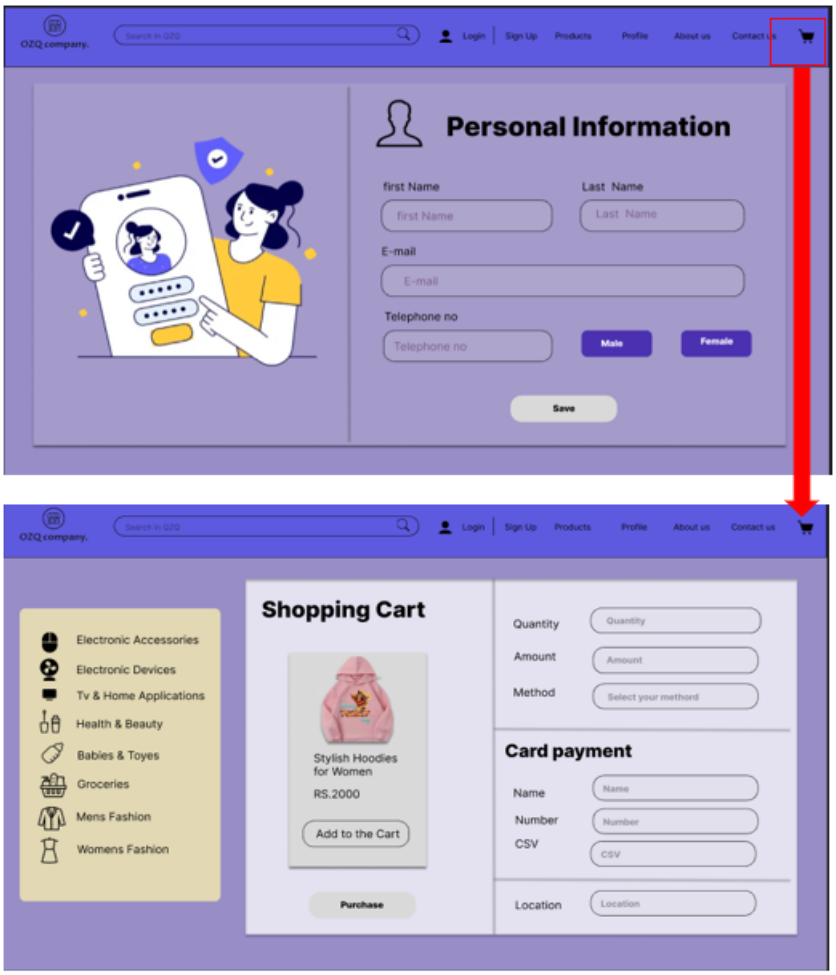


Figure 47 : Black box testing

Grading Rubric

Grading Criteria	Achieved	Feedback
LO1 Examine what an API is, the need for APIs and types of APIs		
P1 Examine the relationship between an API and a software development kit (SDK).		
M1 Asses a range of APIs for a particular platform that covers a range of uses.		
D1 Evaluate potential security issues surrounding APIs		
LO2 Apply the knowledge of API research to design an application that incorporates relevant APIs for a given scenario or a substantial student chosen application		
P2 Analyse an existing application that could be extended with a suitable API.		
M2 Design an application that will utilise an API for a given purpose.		

D2 Create a design for a chosen substantial application that will utilise a range of APIs, justifying choices..		
LO3 Implement an application in a suitable development environment		
P3 Build on an existing application framework to implement an API.		
M3 Develop an application that utilises an API.		
D3 Construct an application utilising multiple APIs, following the designs in LO2		
LO4 Investigate scenarios with respect to design PatternsLO4 Document the testing of the application, review and reflect on the APIs used		
P4 Design and complete a 'white box' test of the application, recording the results.		
M4 Conduct 'black box' tests of your application, recording the results.		
M5 Update the application accordingly with the results.		

D4 Critically evaluate the APIs used within your application. Provide a data security report of your application.		