

## **Electronics Store Inventory Management System**

A project submitted

in partial fulfillment of the requirements for the degree of

Bachelor of Science in Computer Science

**By**

**Muhammad Faizan Ali 2k17-BSCS-134**

**Umer Khan 2k17-BSCS-129**

**Noman Haidar 2k17-BSCS-103**

**Supervised By**

**Dr. Omer Aziz**

**Department of Computer Science**

**NFC Institute of Engineering & Technology, Multan, Pakistan**

# **Acknowledgment**

All praise for Almighty Allah whose uniqueness, oneness, and wholeness are unchallengeable, and Who showed us the way in all the darkness and situations. All the respects are for Holy Prophet Hazrat Muhammad (SAWW), who enabled us to recognize the oneness of our creator.

After that, we acknowledge our parents and family for their love, support, encouragement, and for everything they have done for us. They are the biggest support for us. We thank them for always being there with us.

We express our deepest sense of gratitude to our supervisor Dr. Omer Aziz for providing us an opportunity to design and develop a project entitled “Electronic store Inventory management system” and guiding us till the completion of this project.

We pay our heartiest gratitude to all our teachers for their encouragement, moral support, and guidance.

We acknowledge with enormous pleasure the tenacious interest, cheering attitude, and constant encouragement rendered by Dr. Naeem Aslam (Head, Department of Computer Science, NFC IET). His constant determination for excellence in everything that happens at NFC IET and self-sacrificing inspiration has always helped us to move ahead.

Regards,

Muhammad Faizan Ali

Noman Haider

Umer khan

# **Undertaking**

This is to affirm that the project entitled “Electronic store Inventory management system” is an original work done by the undersigned, in partial fulfillment of the requirements for the degree “Bachelor of Science in Computer Science” at Computer Science Department, NFC Institute of Engineering& Information Technology.

All the analysis, design, and system development have been accomplished by the undersigned. Moreover, this project has not been submitted to any other college or university.

# **Abstract**

Electronic store Inventory management system is feasible, unique & supportive for the inventory management, where This project eradicates the paperwork, human error, manual interruption & speeds up the process, and gives the admin the ease of managing products. The web-based system is designed for laptops/pc which help admin to understand it better.

The system is working appropriately & meeting all the user requirements

Table of Contents

[Chapter 1: Introduction 6](#_Toc64223303)

[1.1. Introduction 6](#_Toc64223304)

[1.2. Project Aims and Objectives 7](#_Toc64223305)

[1.3. Vision Statement 7](#_Toc64223306)

[1.4. Significance of the Study 7](#_Toc64223307)

[1.5. Scope and Limitations 7](#_Toc64223308)

[1.5.1. Scope 7](#_Toc64223309)

[1.5.2. Limitations 7](#_Toc64223310)

[1.6. Definition of Terms 7](#_Toc64223311)

[Chapter 2: Existing System 8](#_Toc64223312)

[2.1. Drawbacks in the existing systems 8](#_Toc64223313)

[2.2. Solution of These Problems 8](#_Toc64223314)

[2.3. System Requirements Specification 8](#_Toc64223315)

[2.3.1. User Interfaces 8](#_Toc64223316)

[2.3.2. Hardware Interfaces 8](#_Toc64223317)

[2.3.3. Software Interfaces 9](#_Toc64223318)

[2.4. Functional Requirements 9](#_Toc64223319)

[Following are the Functional Requirements: 9](#_Toc64223320)

[Login (Admin): 9](#_Toc64223321)

[Signup (admin): **Error! Bookmark not defined.**](#_Toc64223322)

[Search products/Upload a product: 9](#_Toc64223323)

[2.4.1. Output Design 10](#_Toc64223324)

[The objectives of the input design are: 10](#_Toc64223325)

[2.4.2. Output Definition **Error! Bookmark not defined.**](#_Toc64223326)

[2.4.3. Output Media **Error! Bookmark not defined.**](#_Toc64223327)

[2.4.4. Input Design **Error! Bookmark not defined.**](#_Toc64223328)

[2.4.5. Input Stages 10](#_Toc64223329)

[2.5. Use Case Diagram 11](#_Toc64223330)

[2.5.1. Admin Use Case 12](#_Toc64223331)

[Figure 1: Admin Use Case Diagram 12](#_Toc64223332)

[2.6. Non-Functional Requirements 13](#_Toc64223333)

[2.8. Requirement Shell for Functional Requirement 13](#_Toc64223334)

[2.8.1. Requirement #:1 13](#_Toc64223335)

[2.9. Requirement Shell for Non-Functional Requirement 13](#_Toc64223336)

[2.9.1. Requirement #: 1 13](#_Toc64223337)

[Chapter 3: Software Process Model 14](#_Toc64223338)

[3.1. Software Process Model 14](#_Toc64223339)

[3.2. Which Model I used 15](#_Toc64223340)

[3.2.1. Phases of Iterative Waterfall Method 15](#_Toc64223341)

[Chapter 4: Proposed System 16](#_Toc64223342)

[4.1. Need for the proposed system: 16](#_Toc64223343)

[4.2. Feature of Proposed System: 16](#_Toc64223344)

[Chapter 5: System Design 17](#_Toc64223345)

[5.2. Context Diagram 17](#_Toc64223346)

[5.3. Data Flow Diagram 17](#_Toc64223347)

[2nd Level Admin DFD 18](#_Toc64223348)

[5.4. Activity Diagram 19](#_Toc64223349)

[5.4.1. Admin Side 19](#_Toc64223350)

[5.4.2. Admin Side 19](#_Toc64223351)

[5.5. Class Diagram 20](#_Toc64223352)

[5.6. ER Diagram 22](#_Toc64223353)

[5.6.1. Relationship in ERD 22](#_Toc64223354)

[5.6.1.1 One-to-One Relationship 22](#_Toc64223355)

[5.6.1.2 One-to-Many Relationship 22](#_Toc64223356)

[5.7. Stakeholders Interface 23](#_Toc64223357)

[5.7.1. Customer Interface 23](#_Toc64223358)

[5.7.2. Admin Interface 24](#_Toc64223359)

[Chapter 6: Development 24](#_Toc64223360)

[6.1. Selection of tools and technology 24](#_Toc64223361)

[6.1.1. Hardware: 24](#_Toc64223362)

[6.1.2. Software: 24](#_Toc64223363)

[6.2. Graphical User Interface: 25](#_Toc64223364)

[6.2.1. Login Page 25](#_Toc64223365)

[6.2.2. Sign up Page **Error! Bookmark not defined.**](#_Toc64223366)

[6.2.3. Dashboard page 26](#_Toc64223367)

[6.2.3. Cart. 27](#_Toc64223368)

[Chapter 7: Software Testing 30](#_Toc64223369)

[7.1. Testing Process 30](#_Toc64223370)

[7.2. Test Case Design 30](#_Toc64223371)

[7.3. Black Box Testing (Behavioral Testing) 31](#_Toc64223372)

[7.3.1. Test Data 32](#_Toc64223373)

[7.4. White Box Testing (structural testing) 33](#_Toc64223374)

[Chapter 8: Implementation and Training 34](#_Toc64223375)

[8.1. Implementation 34](#_Toc64223376)

[8.2. Training 34](#_Toc64223377)

[Conclusion 34](#_Toc64223378)

[References 35](#_Toc64223379)

# Chapter 1: Introduction

## 1.1. Introduction

Electronic store in Pakistan is growing rapidly and playing a vital role in our society in many ways. Pakistan has observed an immense rise in e-commerce business in the previous few years. We are converting manual business into digital business Our main objective is to ease the Customer to maintain electronics products. One of the best features of our application provides to reduce manual labor and also allows the admin to search for their products or upload.

## 1.2. Project Aims and Objectives

As technology is advancing day by day, we aim to provide a competitive system that meets the challenges of today’s technology.

* Our web Application will mechanize the manual system.
* We want to provide the facility to our admin to manage products in a few clicks by sitting in shop.
* There will be an admin panel that allow him to manage its inventory.

## 1.3. Vision Statement

Our vision is to enhance our relationship with owner by providing him a user-friendly application Moreover, our vision is to modernize the manual system into digital system.

## 1.4. Significance of the Study

It is the era of the internet, so everyone prefers less effort rather than to go to the physical method. Our Application will provide electronic products services for a well-known person for many years that has been providing original products more than 30 years.

## 1.5. Scope and Limitations

## 1.5.1. Scope

* Introducing digital system reduce the manual effort.
* Managing Record that help to access it easily.
* Keep an eye on Sales profit & loss.
* Notify when the products are going to be out of stock.
* Providing the feature of returning or changing the products.

## 1.5.2. Limitations

* Only admin can access the application.
* Only offline system is available.
* In future admin can access its system from anywhere by providing online system.

## 1.6. Definition of Terms

Terms used in our project are:

* **Admin:** A person who works for the store & controls the database & applications & services provided for the admin panel. And admin can also manage orders from the admin panel.
* **Purchase:** To purchase from different vendors and update its database.
* **Source listing:** This module is supposed to identify the products by their description.
* **Add products:** This module is suppose to take the inputs from an input device
* **Update products:** This module will update products.

# Chapter 2: Existing System

## 2.1. Drawbacks in the existing systems

The Customer is using this type of application **first time**.

Previously the owner was using manual system to keep track of its entire shop system. Which cause him to forgot things to maintain.

## 2.2. Solution of These Problems

Nowadays there are facilities to access all these things by digitally which help the customer to depend on it and help him to improve his business which tells him all things available in his shop

## 2.3. System Requirements Specification

The SRS document is the documents that explain the proposed objective, requirements, and functionality of the software.

## 2.3.1. User Interfaces

* Front-End (web Application): Tail wind, CSS, HTML, JavaScript.
* Back-End (Web Application): PHP (phpMyAdmin SQL server)

## 2.3.2. Hardware Interfaces

* PC/Laptop (For Admin Panel)

## 2.3.3. Software Interfaces

Following software, Interfaces & tools are used for our project.

**Table 1: Software Interfaces**

|  |  |
| --- | --- |
| **Software** | **Description** |
| Operating System | Window &Mac is used for their amazing support, user-friendliness, and reliability |
| Database Engine | To keep the record of the products, admin data & Applications Firewall is used. |

## 2.4. Functional Requirements

Functional requirements define the fundamental actions that are mandatory to be performed by the system. A functional requirement will describe the behavior of the functions of the system when certain conditions are met.

## Following are the Functional Requirements:

* Login (Admin)
* Search Products/Add product
* Printed Billing

## Login (Admin):

* Admin can log in to the web Application using his/her username, email & password, but to log in, he/she must signup first.
* Admin needs to log in as well to manage customer’s orders.
* All the information regarding the customer & admin is stored in the database so when they log in the credentials are verified every time.

## Search products/Add product:

* After the admin is logged in he/she can maintain its inventory.

## 2.4.1. Output Design

The most important task of any system is the design of output. During output design, developers identify the type of outputs needed and consider the necessary output controls and prototype report layouts.

## The objectives of the input design are:

* To design data entry and input procedures
* To reduce input volume
* To design source documents for data capture or devise other data capture methods

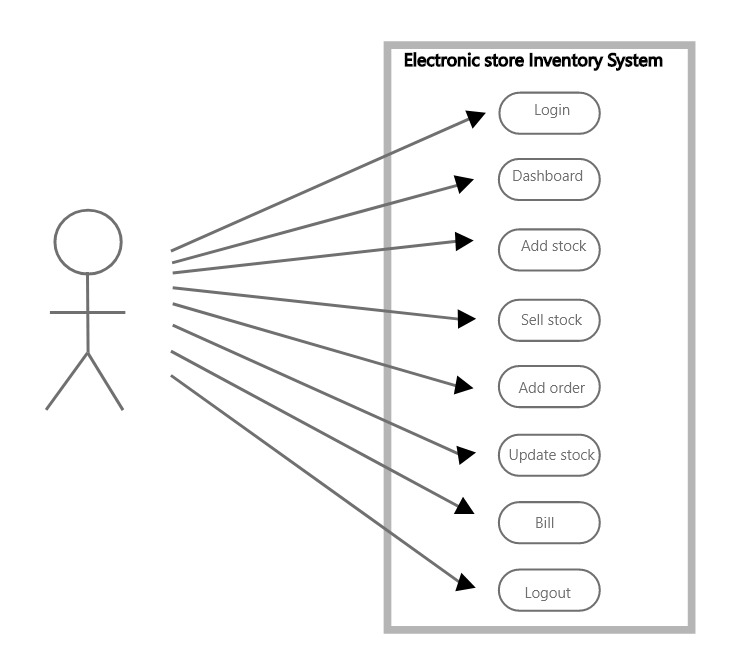
* To design input data records, data entry screens, user interface screens, etc.
* To use validation checks and develop effective input controls.

## 2.4.5. Input Stages

**Admin Side:**

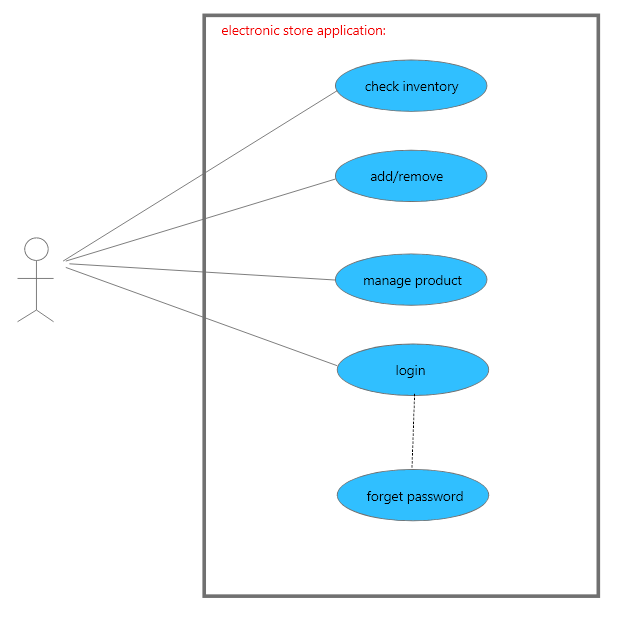
* Admin logs into the admin panel by entering its credentials.
* After Admin login an admin panel is opened where all the active orders are listed, which can be managed by the admin (Accepted/Rejected).
* Admin can control and see all the user’s info.

## 2.5. Use Case Diagram



## 2.5.1. Admin Use Case

* To access the admin panel the admin must log in for authentication purposes.
* The Admin login is secured with double md5 encryption of the password.
* Admin can manage the customer’s orders.



## Figure 1: Admin Use Case Diagram

## 2.6. Non-Functional Requirements

A non-functional requirement will describe how a system should act and what restrictions are there on its functionality.

**Following are Non-Functional Requirements:**

* Performance Requirement
* Safety Requirement

**Performance Requirement:**

The Application should be easily accessible and have good performance even on older hardware to avoid any lags or system hang-ups. And should be memory efficient to avoid any performance issues.

**Safety Requirement:**

The Application is almost 100% safe and all the user data is protected by the double md5 encryption method so that if any hacker finds a way in all the data including passwords and sensitive personal information will be encrypted and there is no way one can decode double md5 protection easily.

# 2.8. Requirement Shell for Functional Requirement

# 2.8.1. Requirement #:1

Requirement type: Functional

Description: The application will allow the admin to register.

Originator: ABC

Priority:10 Dependency: Conflicts:

Supporting Material: None.

# 2.9. Requirement Shell for Non-Functional Requirement

# 2.9.1. Requirement #: 1

Requirement type: Functional

Description: The application will allow the admin to register.

Originator: ABC

Priority:10 Dependency: Conflicts:

Supporting Material: None.

# Chapter 3: Software Process Model

## 3.1. Software Process Model

A software process model depicts the order in which the software development activities will occur. It shows the order of the phases of the software life cycle. It is a basic illustration of the software process. It is a standard format for planning, organizing, and maintaining software.

**There are different software process models such as:**

* Waterfall model
* Iterative Waterfall model
* V model
* Spiral model
* Prototyping model
* Agile methods

Each model is selected based on some constraints and under certain parameters (such as size, duration, budget, etc.).

Normally, the software process model covers the lifetime of the software.

**There are 6 main phases of the software development life cycle:**

* Planning

* Requirement Analysis
* Designing
* Development
* Testing
* Deployment

## 3.2. Which Model I used

We used the **Iterative Waterfall model** as the user requirements were not changed and we had a

final requirement document. The iterative Waterfall model is very simple to understand and use. And it is one of the most widely used software development models.

**The iterative waterfall model provides feedback paths from every phase to its preceding phases, which is the main difference from the classical waterfall model.**

**Figure 2: Iterative Waterfall Method**

## 3.2.1. Phases of Iterative Waterfall Method

**Requirement Definition:**

All the requirements for the system are gathered in this phase according to the need of the client by validating them and checking the possibility to implement them. A Requirement Specification Document is generated in this phase which will be used as input for the succeeding phase.

**System and Software Design:**

Before implementation, the system architecture should be designed that depicts the major blocks of the system and interfaces. Software and hardware components are specified in this phase. The system

architecture is specified in the code modules. In this phase, the Software Design Document is generated as output and will be used as input in the next phase.

**Implementation and Unit Testing:**

Based on the Software Design Document, actual coding starts. The system is established in the form of smaller units. Each unit is verified for its functionality (unit testing).

Integration and System Testing: After unit testing, all the individual components are unified into a complete system and tested again to check the integration. After the successful integration of the system, the complete system is validated against the initial requirements.

**Operation and Maintenance:**

The system is delivered to the client and testing is done by the client. After the successful delivery, the system is maintained at each stage.

# Chapter 4: Proposed System

## 4.1. Need for the proposed system:

When the system was manual, we found it difficult to manage the products and their sales & was very difficult to manage user’s products orders, and was very time consuming for both the Store and the customers. Now it is easy to manage the products and their sales and the user’s product orders. And everything is done digitally through web. And this upgrade will also give the store boost in their business as well.

## 4.2. Feature of Proposed System:

* User-Friendly.
* Secure.
* Provided Universal design for the user's easiness.
* Cost-saving by reducing money cost.

# 

# Chapter 5: System Design

## 5.2. Context Diagram



**Figure 3: Context Diagram**

Figure 11 is the context diagram of our system. This type of diagram usually depicts the system in the middle, without any description of its internal structure, encircled by all its interrelating systems, environment, and activities.

This diagram explains the relationship of the system with other entities. Admin can manage the User’s order.

The database contains all the data of the users, admin, products, contact, and user’s orders.

## 5.3. Data Flow Diagram

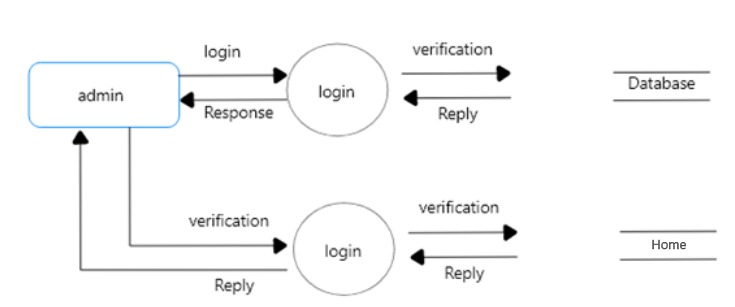
A DFD is a diagram of the system as a link of functional processes associated with each other by "pipelines" of data (data flows) and "holding tanks" of data. The processes alter the data. The diagram divides the system into its modules.

The DFD makes use of symbols for processes, data stores, and data flows. Note that external entities do not show on a DFD, but data stores do. The information flowing into and out of the system (the data flow) will be retained, and the one major process will be decomposed into several more refined process.

A bubble shows a process that transmutes data inputs into data outputs. A curved line shows the stream of data into or out of a data store. A set of parallel lines shows a collection of data items at rest, a place where data resides. A data store indicates that the data is stored somewhere to be used later or by the other processes in a different order.

**1st level Admin DFD**

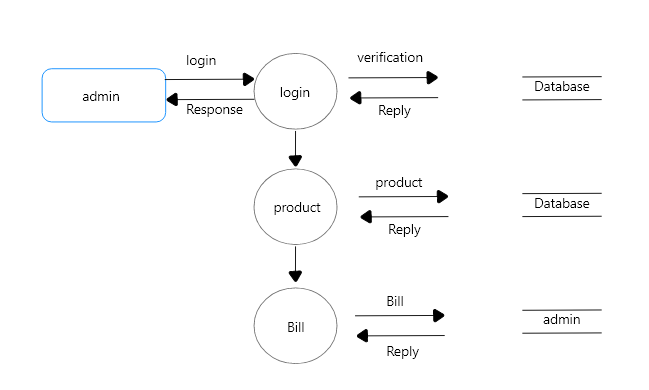
* Admin can log in and his login requests are Checked against his record in the database.
* Admin can manage the user’s order by viewing the data present in the database. Against all these operations the database will respond to admin accordingly.



**Figure 4: 1st level Admin Side DFD**

## 2nd Level Admin DFD

Admin can view the user’s orders



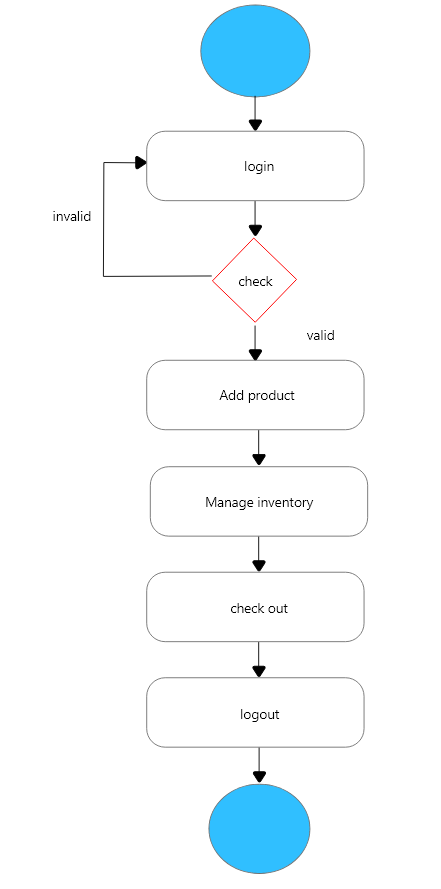
**Figure 5: 2nd level Admin Side DFD**

## 5.4. Activity Diagram

## 5.4.1. Admin Side

* Admin will log in for authentication & verification purposes. If the email, password & key code is valid, access to the admin control panel will be granted to the admin otherwise he will have to login again with the correct login info.
* After a successful login, the admin can perform various tasks.
* Admin can manage the User’s products orders.
* After that, the admin can log out of the system.

## 5.4.2. Admin Side



**Figure 6: Activity Diagram**

## 5.5. Class Diagram

**Overview of Class Diagram**

The static model of OO is the class diagram (and the object diagram to represent a specific instance of a class), which includes components of attributes, services, and relationships, and concepts of hierarchy, abstraction, encapsulation, inheritance, and polymorphism.

Identifying class responsibilities and then proceeding "upward" to use case scenarios for validation is called the "bottom-up" approach. It is a fine way to proceed, but so is "top-down," which begins with use cases and works down through scenarios to identify classes.

When beginning with classes, you may want to first identify them through abstraction or looking for noun phrases in existing documentation, and then list the responsibilities of each class, identify attributes, identify operations (behavior—again through abstraction or looking for verb phrases), and, lastly, employ use case scenarios to validate the class diagram.

**Classes**

A class is an abstraction of something in the application domain, generally stated as a noun. It can be conceptual or physical; it reflects the capabilities of a system to keep information about it, interact with it, or both.

A class consists of attributes (data that describes the object and that the system needs to store about the object), relationships with other classes, and operations (behavior of the object, which describes its responsibilities).

A class diagram portrays the contents of classes (a collection of declarative, static, model elements) to each other.

**Objects**

Objects are instances of a class and share the properties (attributes and operations) of the class. Each object has its own identity and its own set of values for the attribute.

An object is an entity that encapsulates both state and behavior. The state is represented by attributes and links; operations and state machines represent behavior.

A state stores the effects of the operations of the entity. An object diagram displays objects and their associations/relationships (links) at a point in time.

Object diagrams may be portrayed as collaboration diagrams because the relationships that "link" objects are shown.

**Attributes**

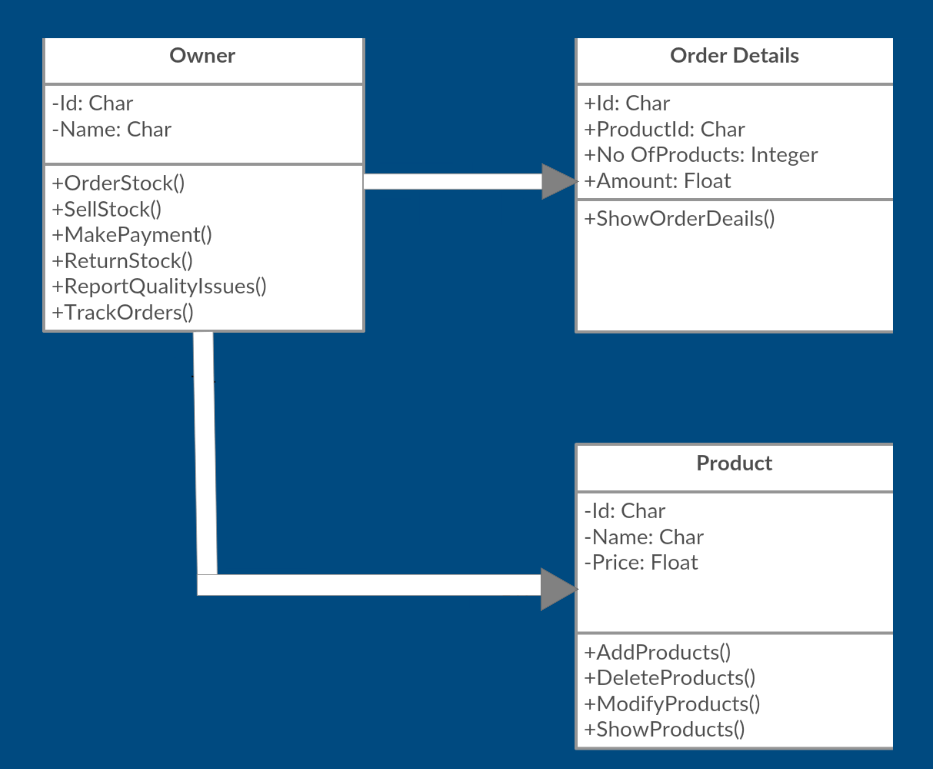
An attribute is a description of a range of values that may be held by the instances of the class. This information is internal to the object. It has also been described as a characteristic, and as:

* a property of an object, conveying information

* usually, a noun
* describing the object in real-world terms
* perhaps a state indicator
* having a data type
* the same for objects of the same class (may differ in value, but not in-kind)
* Able to take on a value defined by an enumerated domain (set of specific values).

**Class Diagram of Electronic Store:**

The following figure depicts the classes used in Electronic Store with appropriate attributes, methods, and their inference.

**Figure 7: Class Diagram**

## 5.6. ER Diagram

An Entity Relationship Diagram is a model that classifies the entities that reside in a system and the relationships amongst those entities. An ERD is frequently used to imagine a relational database: each entity denotes a database table, and the relationship lines signify the keys in one table that point to certain records in related tables. ERDs may also be more conceptual, not essentially capturing every table required within a database, but helping to diagram the key concepts and relationships.

**Figure 8: ER Diagram**

## 5.6.1. Relationship in ERD

## 5.6.1.1 One-to-One Relationship

In a one-to-one relationship, one record of the table can only be associated with only one record of the other table.

## 5.6.1.2 One-to-Many Relationship

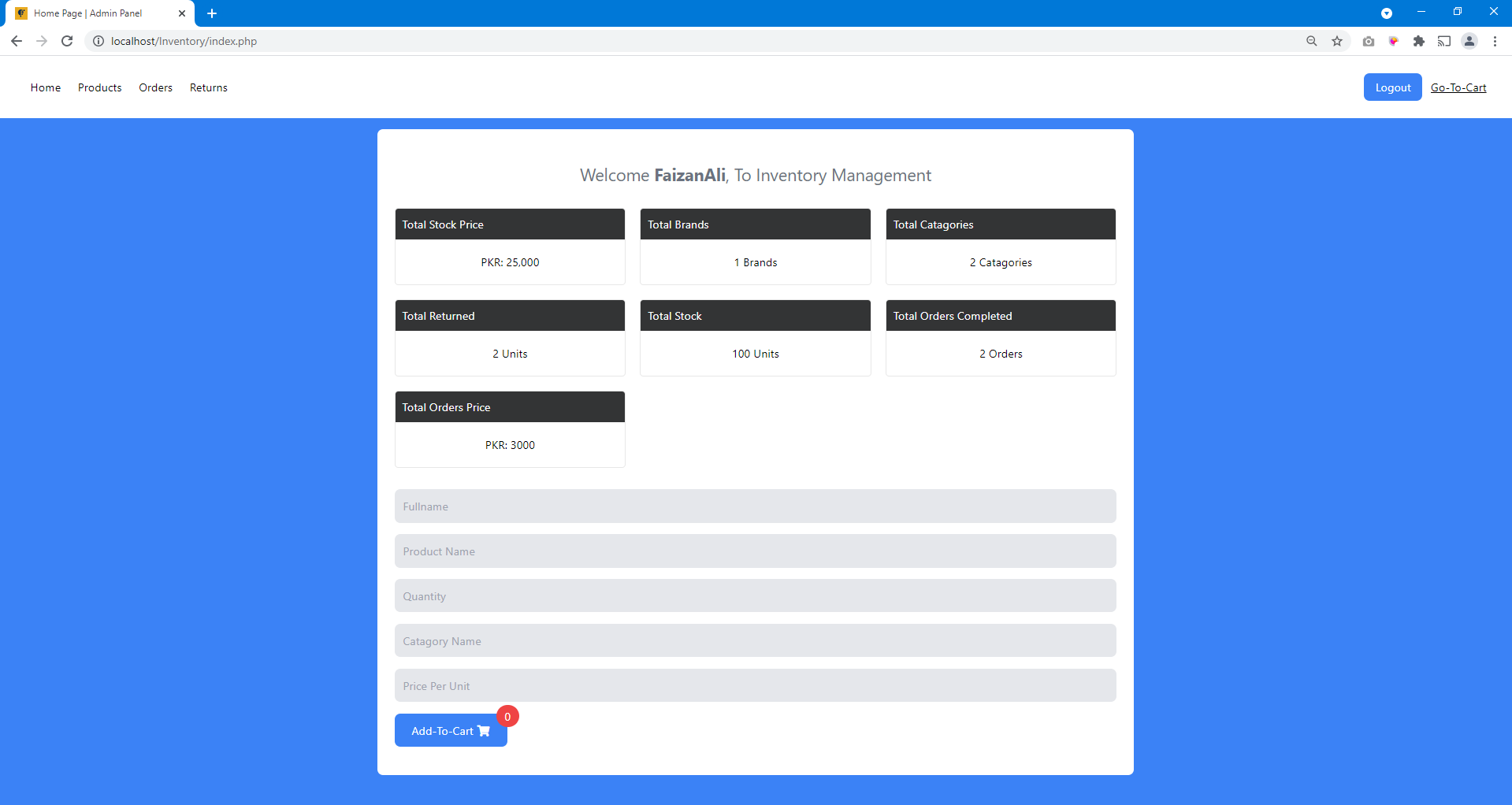
In a one-to-many relationship, one record of the table can be associated with many records of the other table.

## 5.7. Stakeholders Interface

## 5.7.1. Customer Interface

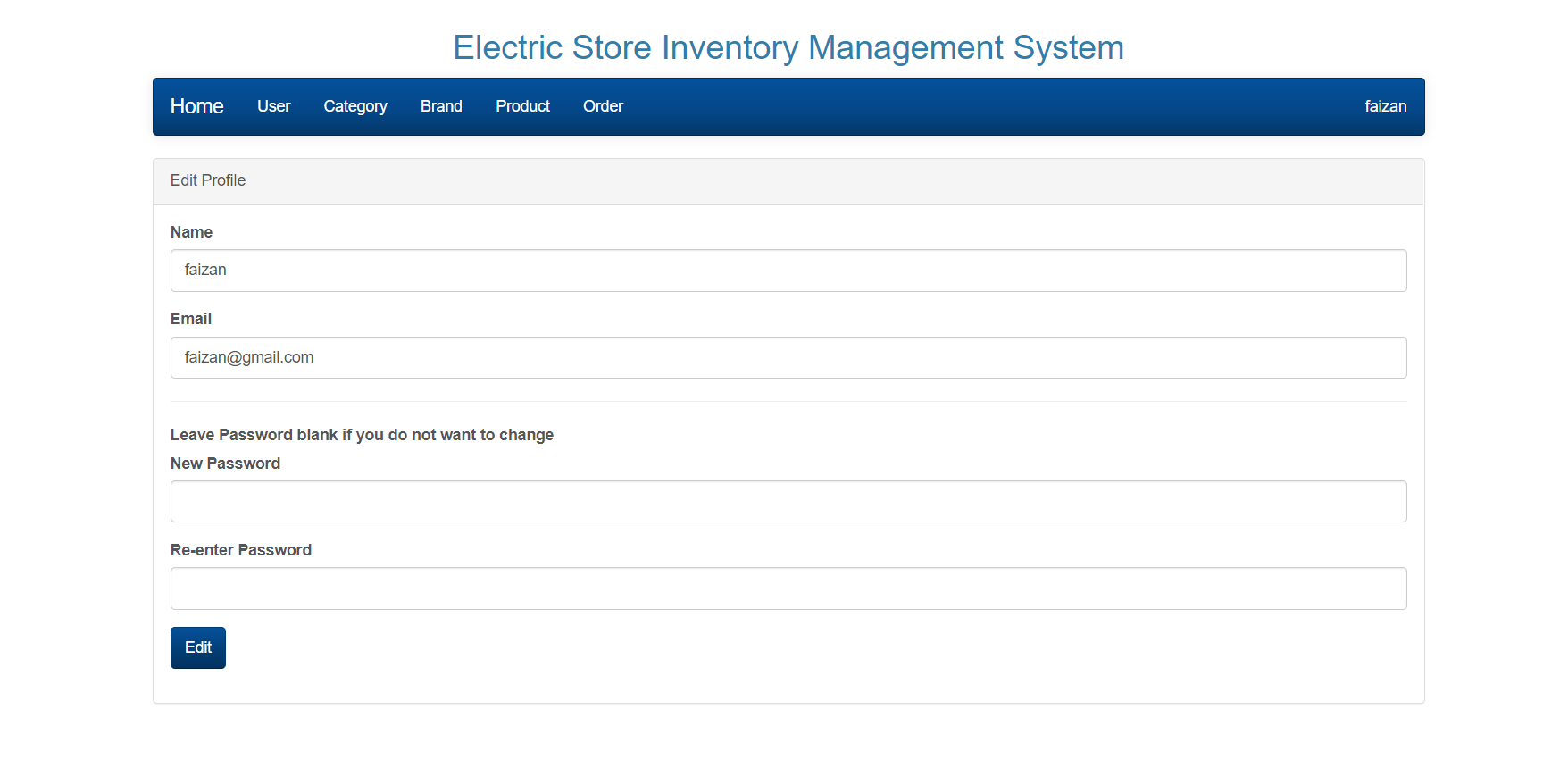
The following figures show the customer’s interface.

**Web Application:**



**Figure 9: Customer’s Interface (Web Application)**

## 5.7.2. Admin Interface



**Figure 10: Admin’s Interface (Admin Panel)**

# Chapter 6: Development

## 6.1. Selection of tools and technology

## 6.1.1. Hardware:

* Recommended Window 7,8,10 or latest version
* macOS Mojave10.14.6, macOS High Sierra10.13.6 or newer.

## 6.1.2. Software:

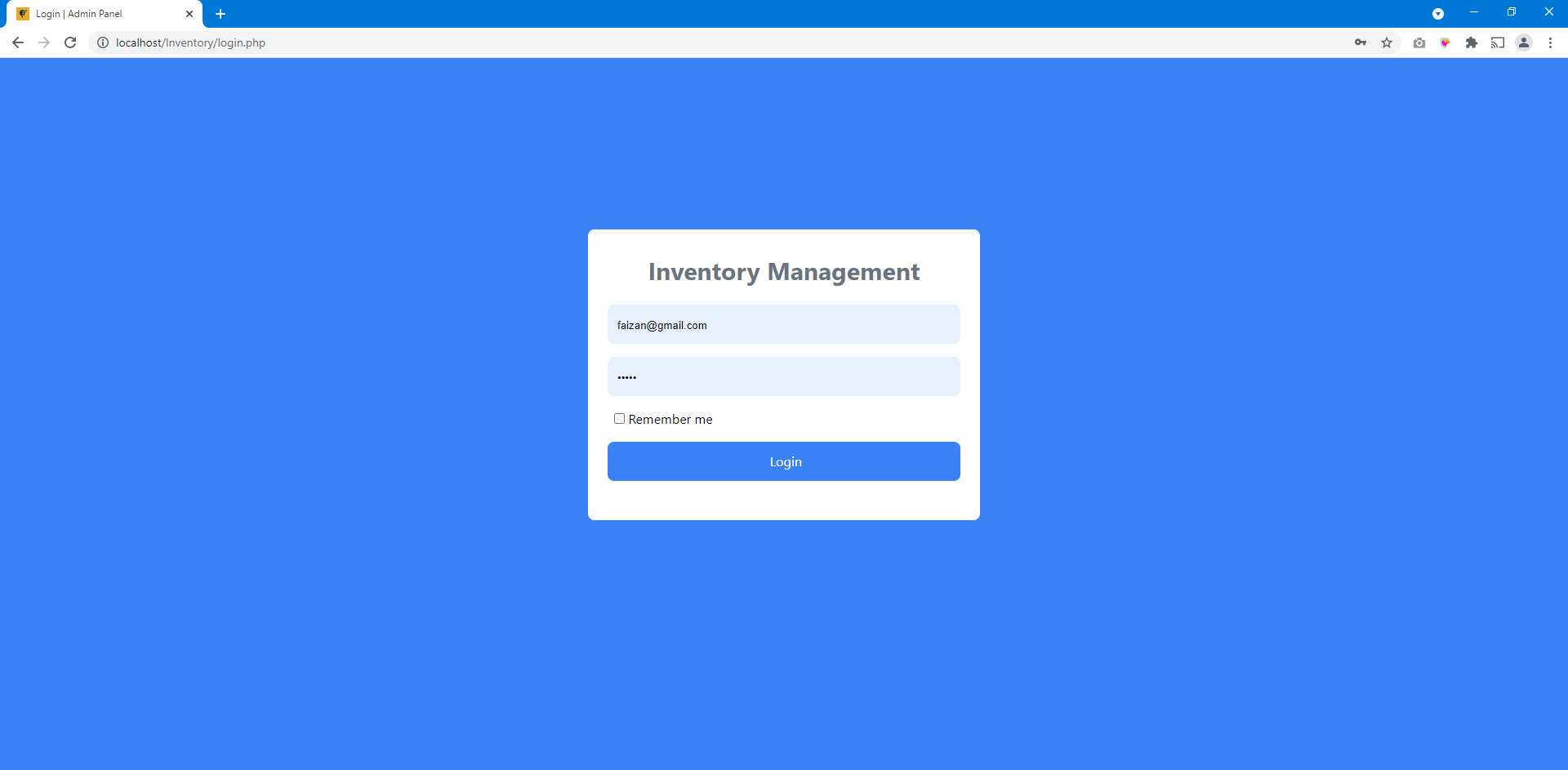
**Software & Tools used to develop the web Application are as follows:**

* **Adobe XD**
* Adobe XD is a vector-based user experience design tool for web apps and mobile apps, developed and published by Adobe Inc.
* **Visual Studio Code**
* Visual Studio Code is a free source-code editor made by Microsoft for Windows, Linux, and macOS. Features include support for debugging, syntax highlighting, intelligent code completion, snippets, code refactoring, and embedded Git.
* **HTML**
* Hypertext Markup Language is the standard markup language for documents designed to be displayed in a web browser.
* **PHP 7.4**
* PHP is a general-purpose scripting language that is especially suited to web development.
* **WAMP Local Host SQL Server**
* WampServer refers to a solution stack for the Microsoft Windows operating system, created by Romain Bourdon and consisting of the Apache web server, OpenSSL for SSL support, MySQL database, and PHP programming language.

## 6.2. Graphical User Interface:

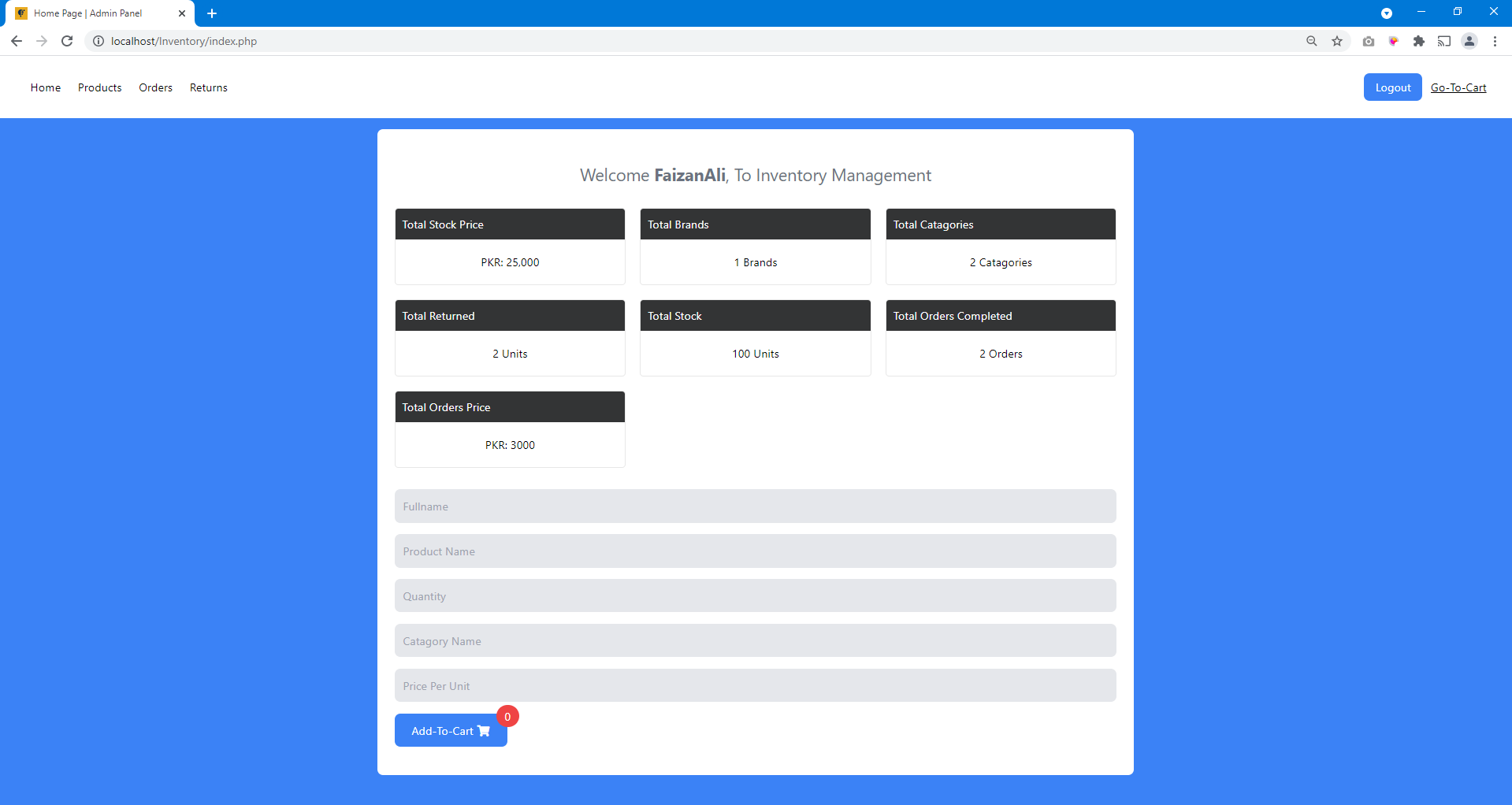
## 6.2.1. Login Page

The login page requires email and password, if the email & password combo is correct then the user will be logged in and can access the Application after clicking the login button.



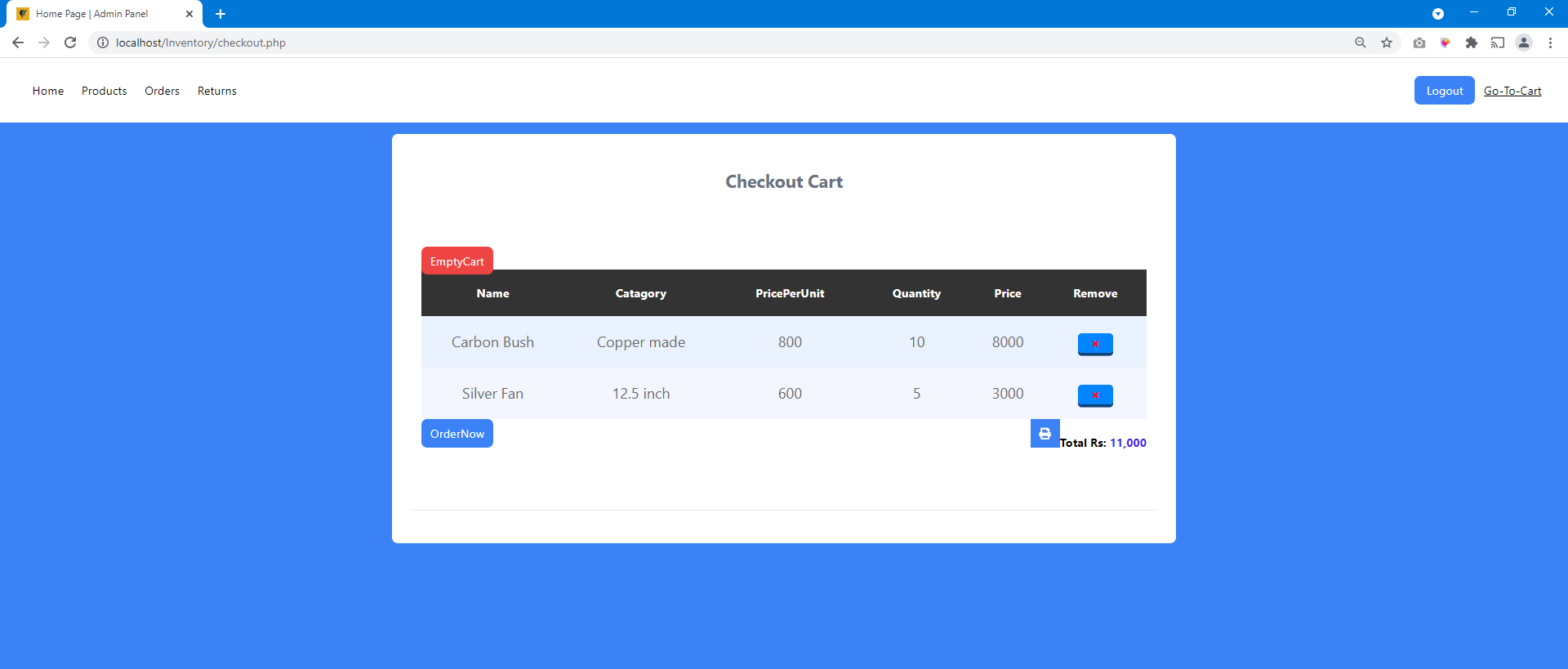
**Figure 11: Login Page (WebApplication)**

## 6.2.3. Dashboard page



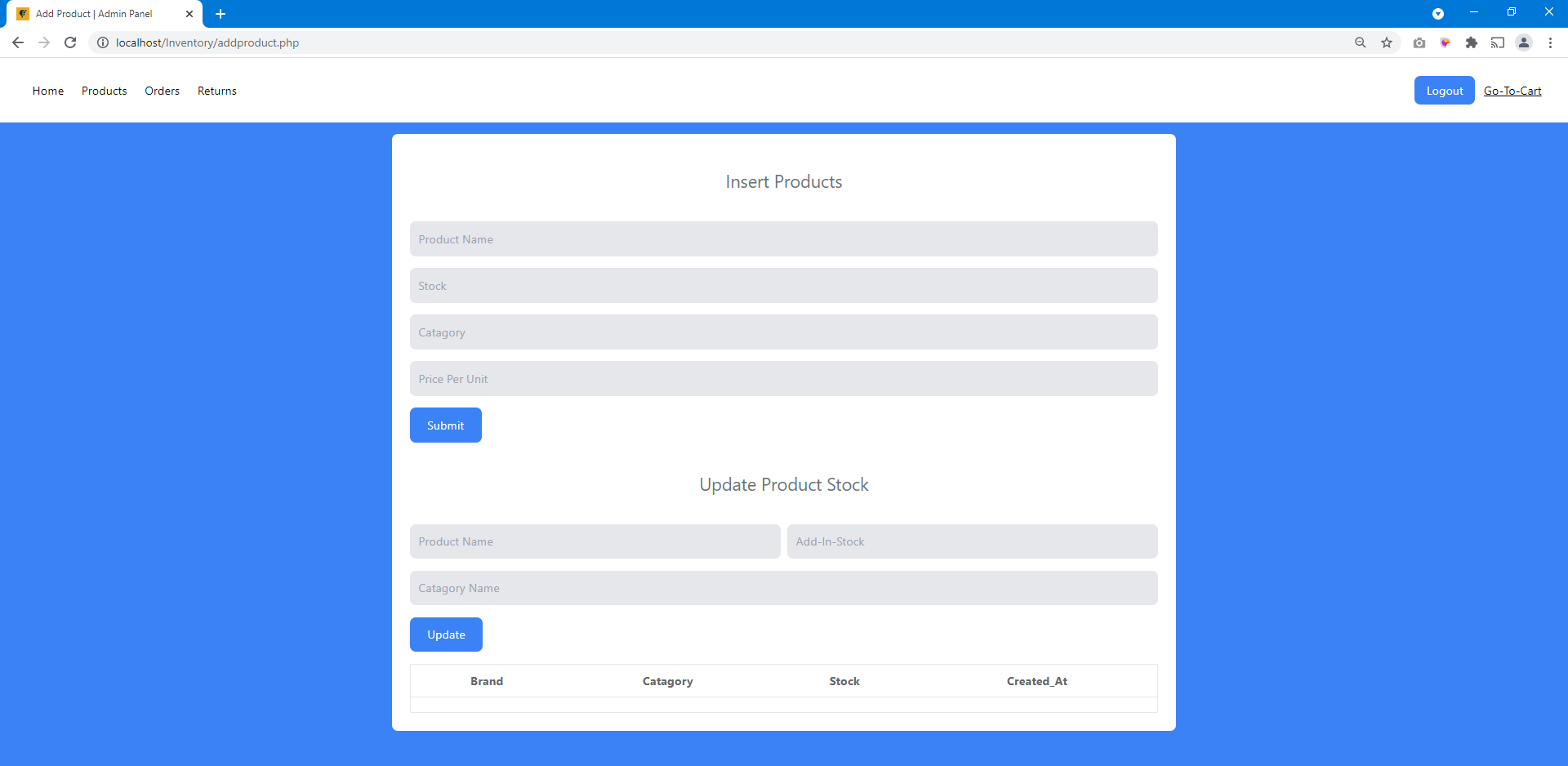
**Figure 12: Home Page (Web application)**

## 6.2.3. Cart.



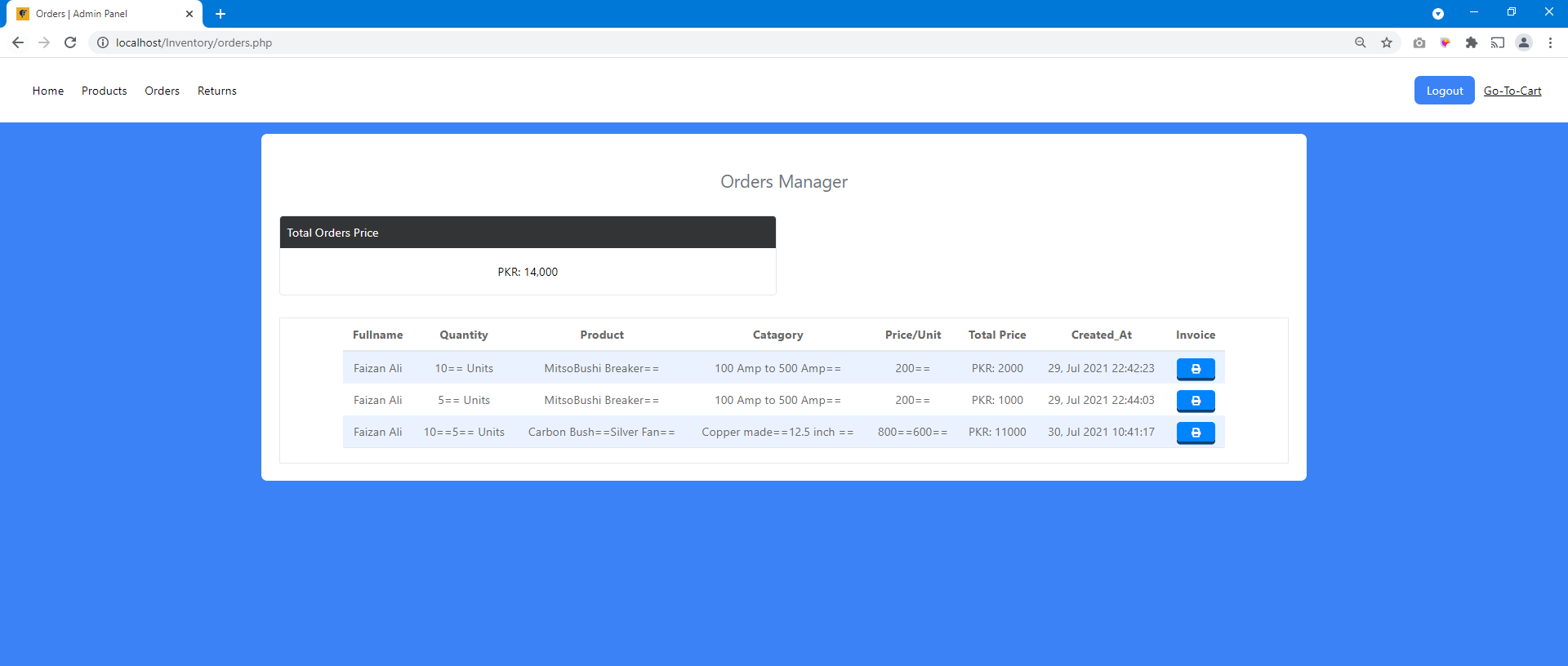
**Figure 13: Add to Cart**

Add product

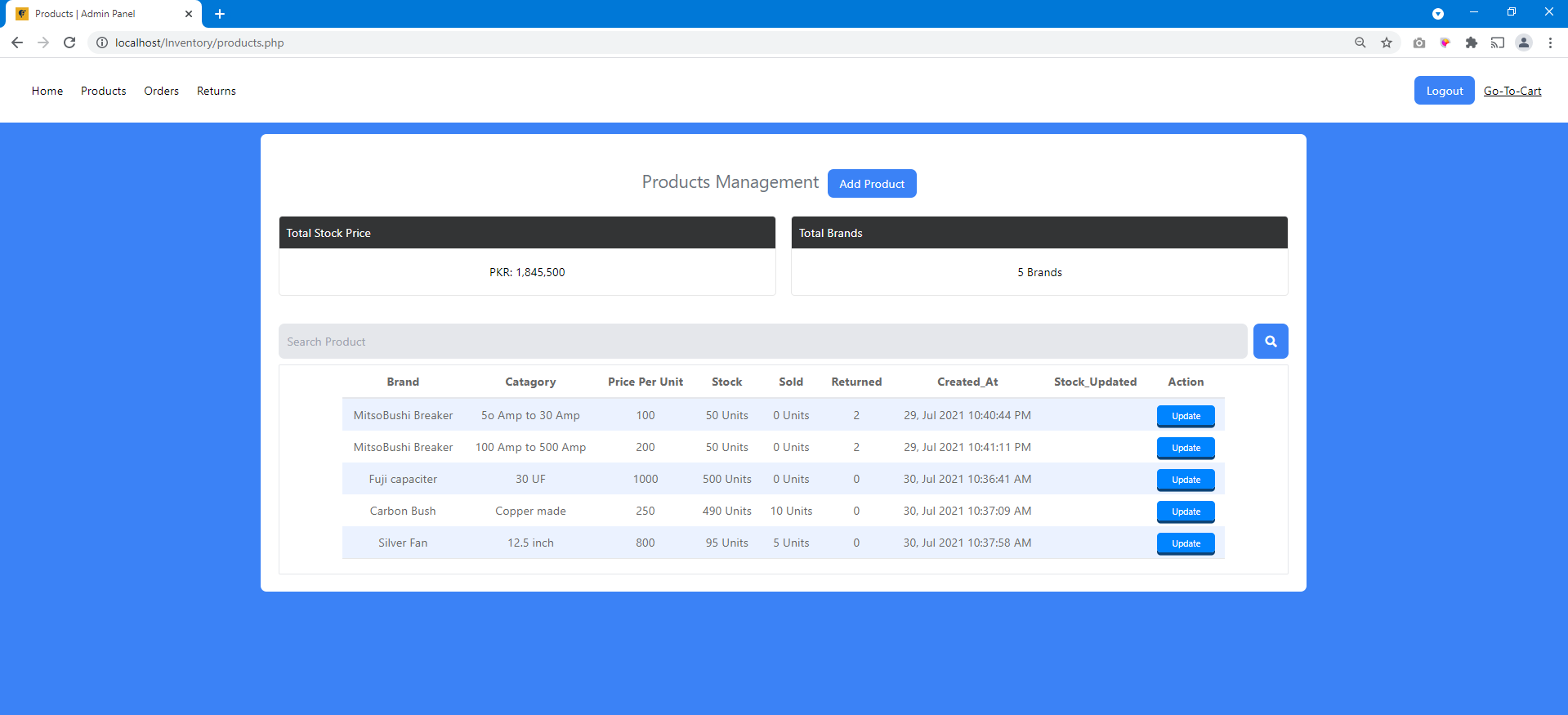


**Figure Add Product**

Order

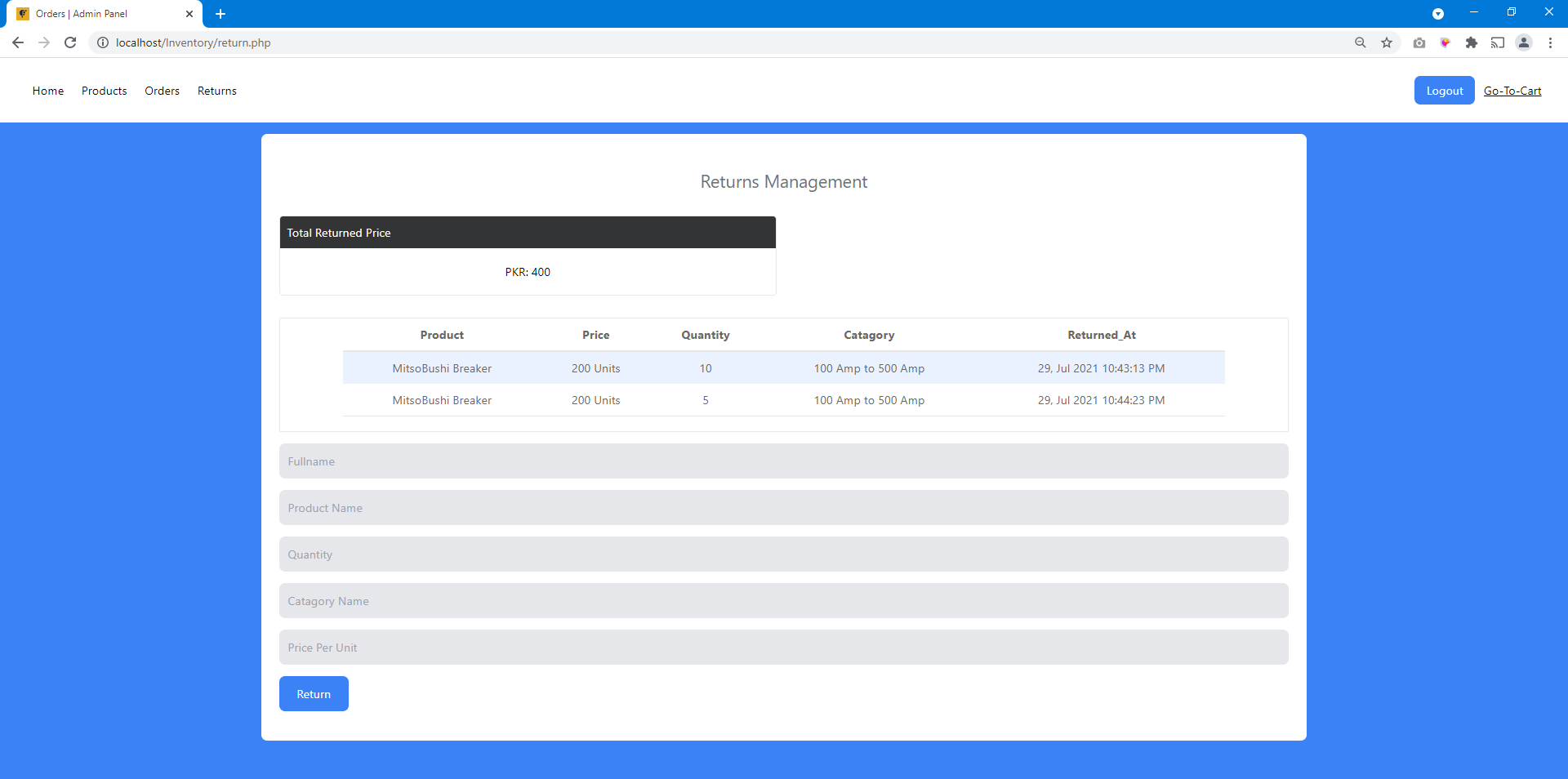


Product



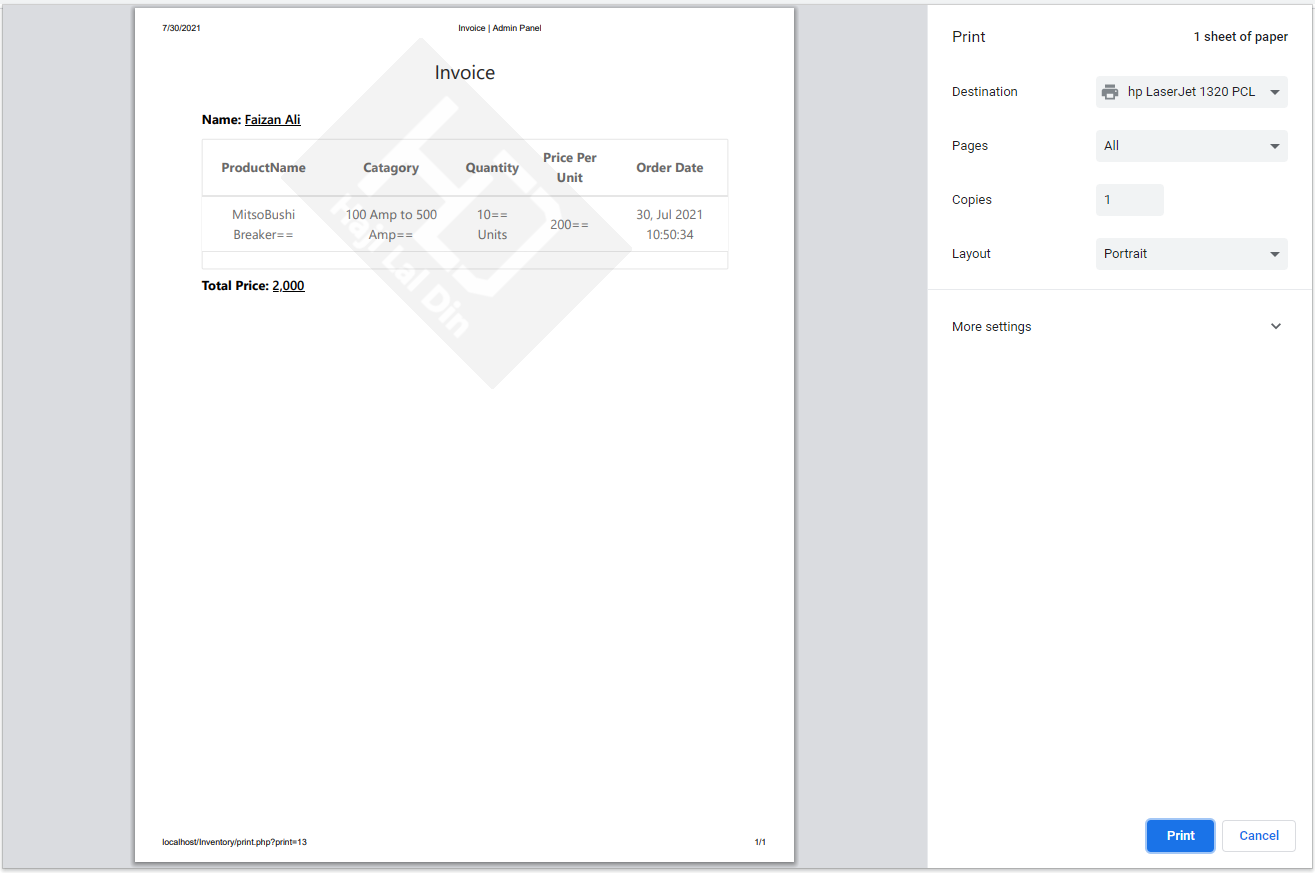
**Figure Product**

Return Product



**Figure Return Product**

Print Bill



**Figure Print Bill**

# Chapter 7: Software Testing

## 7.1. Testing Process

The testing process is the process that ensures that the system fulfills the requirement and is working properly. It is basically for verification and validation. Our application is based on the “Iterative Waterfall model” and is verified and validated by the stakeholder at every stage of the development process. Both validation and verification are diverse events. The major diversity among the two can be concise as follows:

**Verification** checks that the system is according to its requirement. Validation checks that the program meets the expectations of the stakeholder.

**Validation** is the procedure of examining whether the system captures the customer’s requirements, while verification is the procedure of inspecting that the software encounters the specification.

## 7.2. Test Case Design

**Test Cases:**

Test cases are formulated to guarantee that all the functionalities of the system have been checked at least one-time during testing and that all logical circumstances have been tested. Using White-Box testing, a software engineer can formulate test cases that

* Ensure logical verdicts on their correct and false verges.
* Exercise all logical verdicts on their correct and false verges.
* Implement all loops at margins and inside their operational limits.
* Exercise core data erection to guarantee their rationality.

The test case requirement for system assessment must have consented for assessment before system testing instigates.

**Test Case Design**

**Step 1:** Write the test case explanation

**Table 2: Test Case Design**

|  |  |
| --- | --- |
| **Test Case no.** | **Test Case Description** |
| 1 | Check the response when a valid email and password is entered |

**Step 2:** To Implement the test case. Input some test data

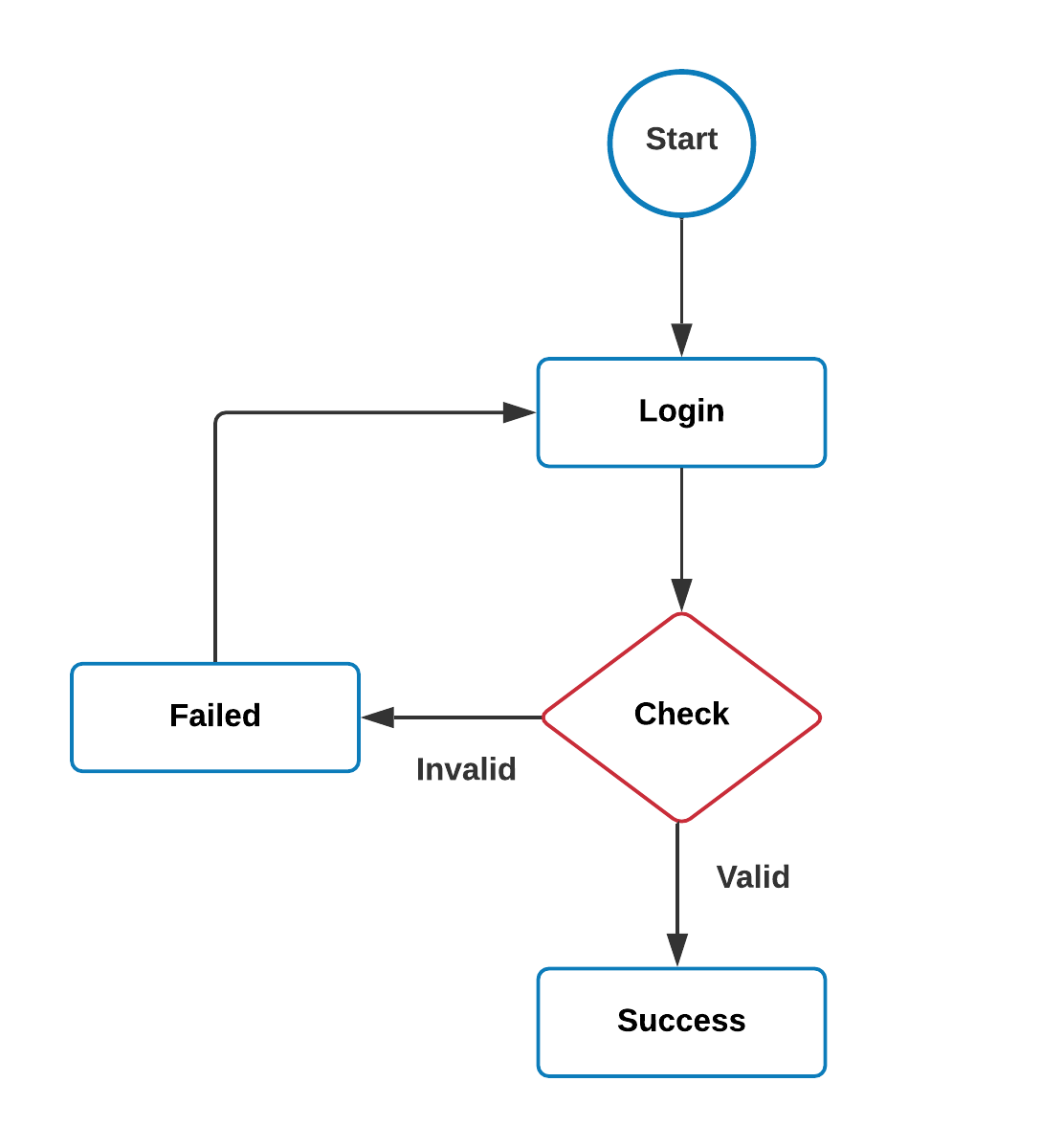
**Step 3:** Check the behavior of the system against test data

## 7.3. Black Box Testing (Behavioral Testing)

Black box testing is a software testing technique in which the functionality of the software is tested without looking at the internal code structure, implementation details, and knowledge of internal paths of the software. We just focus on the outputs and inputs of the system rather than knowing about internal knowledge.

**Flow of Login**

The flow starts with login. The user/admin enters the valid username and password. The credentials are checked & verified from the database. If the email and password are correct, the login is valid otherwise, an invalid login attempt occurs. The user/admin has to login again.



**Figure 14: Flow of Login**

## 7.3.1. Test Data

**Login Object Table**

The following test case table shows login entries.

**Table 3: Login Object Table**

|  |  |  |
| --- | --- | --- |
| **Sr. no** | **Object name** | **Object type** |
| 1 | Email | Text Field |
| 2 | Password | Password Field |
| 3 | Login | Button |

**Valid Input Data Table**

The following test case table describes valid input for login.

**Table 4: Valid Input Data Table**

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Sr. no** | **Email** | **Password** | **Expected**  **Result** | **Actual Page** | **Result** |
| 1 | Faizanali134@gmail.com | 2k17-134 | Home Page | Login | Pass |
| 2 | Nomanhaider103@gmail.com | 2k17-103 | Home Page | Login | Pass |
| 3 | umerkhan129@gmail.com | 2k17-129 | Home Page | Login | Pass |

**Invalid Input Data Table for Login**

The following test case table describes invalid inputs for login

**Table 5: Invalid Input Data Table**

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Sr. no** | **Email** | **Password** | **Expected**  **Result** | **Actual Page** | **Result** |
| 1 | Faizanali134@gmail.com | Pa214 | Invalid email or  password | Login | Fail |
| 2 | Nomanhaider103@gmail.com | Awerwer5 | Invalid email or  password | Login | Fail |
| 3 | umerkhan129@gmail.com | 12we98 | Invalid email or  password | Login | Fail |

## 7.4. White Box Testing (structural testing)

White box testing focuses on internal structured design. In this, the tester gives some random inputs and expect some output from the system. Programming skills and implementation knowledge is required. It is based on the internal structure of the system. Testing is very complex, so we need a high knowledge of programming before performing this.

**There are white-box testing points that are:**

* Internally organized paths in the coding progressions.
* The drift of definite inputs over the code with anticipated output.

# Chapter 8: Implementation and Training

## 8.1. Implementation

The implementation plan includes detail of all activities that must take place to implement the new system and put it into operation. It identifies the personnel responsible for the tasks and prepares a time chart for implementing the system.

**The implementation plan has the following steps:**

* List all files necessary for implementation.
* Identify all data required to make new files during implementation.
* List all new files and procedures that go into the new system.

The implantation plan should predict possible problems and must know how to solve them. The usual problem may be a missing document, mixed data formats between current data files, errors in data translation, missing data, etc.

## 8.2. Training

Training is an important step because in this step we must ensure the successful implementation and use of the system in the client environment.

Training will be done on the admin side. We will train the admin on how to use the system so that they can use the system without any problem. We will train the admin by entering some data in front of him and then allowing him to use the system after he understands well how to use the system.

# Conclusion

After having a complete study & research on Inventory management, we can see a great change in the response of people in many ways.

As we start to examine, Inventory management has to do with keeping accurate records of goods. The main purpose of inventory management is to help businesses easily and efficiently manage the ordering, stocking, storing and using of inventory. By effectively managing your inventory, you'll always know what items are in stock, how much of them there are, and where they are located. The coding is done in a simplified and easily understandable manner so that the team trying to enhance the project can do so without facing any problem.

We have learned so much & collected a lot of practical knowledge from this project, which we think, shall make us stand in a good position in the future

Please give us 4Gpa.

# References

1. <https://www.w3schools.com/html/html_css.asp>
2. <https://stackoverflow.com>
3. <https://github.com>
4. <https://www.000.webhost.com/>
5. <https://www.tutorialspoint.com/index.htm>