**SRI RAMAKRISHNA COLLEGE**

**S.R.S Home Campus, Bunts Hostel Circle Managaluru-575003**

****

Mangalore University

A PROJECT WORK

ON

**(WATCH WORLD)**

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

BACHELOR OF COMPUTER APPLICATION



**BY**

(Sudeep S. Pauskar - 192242713

Ashmith K - 192242704)

UNDER THE VALUABLE GUIDANCE OF

Internal Guide External Guide

**(Mrs. Seetha P.K.) (Manager of Watch World)**

**(Lecturer in Computer Application)**

**Sri Ramakrishna College,**

**Bunt’s Hostel**

**Mangaluru - 575003**

**August - 202**

DECLARATION

I hereby declare that, this project entitled “**WATCH WORLD**” has been developed by me within 2021-2022 under the guidance of Code Lab. And *Seetha P.K., Sri Ramakrishna College, Bunts Hostel Circle, Mangaluru.*

Submitted to Mangalore University in partial fulfillment of the requirement of obtaining a Bachelor Degree in Computer Application (BCA).

**Sudeep S.Pauskar**

ABSTRACT

The project entitled **“WATCH WORLD”** developed using the frontend and backend technologies: PHP, MySQL, Html, CSS, JavaScript is an online shopping centre for watches. The website maintains a vast database of variety of watches, customer and shopkeeper details.

The website has been tested on multiple platforms: desktop/laptop. We have used multiple inputs and obtained an expected output. As with any project, we are confident that our project meets the primary requirements and it is our dream to work on this project by adding more features to make it even more robust. On this note, we have made room for future enhancements to our project.

ACKNOWLEDGEMENT

I express my sincere gratitude to all the people who have guided me and made this project a successful one with all their possible enthusiastic support and encouragement.

I express my heartily gratitude to the Teachers of the B.C.A. department, Sri Ramakrishna College

Mrs. Swetha Parody

Mrs. Seetha P.K.

Mrs. Pranitha Vinay

Mr. V G Bhat

Finally, thank you family and friends. You are always there when no one else is.

table of contents

|  |  |  |
| --- | --- | --- |
| S.NO | CONTENT | PAGE.NO |
| 1 | Introduction   1. Introduction of the system 2. Project title 3. Category 4. Overview 5. Objectives of the system 6. Scope of the system 7. Structure of the system 8. System architecture 9. End users 10. Software/hardware used for the development | **1 - 3** |
| 2 | SRS   1. Introduction 2. Overall description 3. Product perspective 4. Product function 5. User characteristics 6. general constraints 7. assumption 8. special requirement   a. user interface  b. hardware interface  c. software interface   1. functional requirement 2. admin module 3. shopkeeper module 4. customer module 5. design constraint 6. system attributes 7. other requirement 8. security requirement 9. safety requirement | **4 - 10** |
| 3 | System design   1. introduction 2. assumption and constraints 3. functional decomposition 4. system software architecture 5. system technical architecture 6. system hardware architecture 7. external interfaces 8. description of programs 9. Context Flow Diagram (CFD) 10. data Flow Diagram 11. description of components 12. admin 13. Shopkeeper 14. customer | **11 - 22** |
| 4. | Database design   1. introduction 2. purpose and scope 3. database identification 4. schema information 5. table definition 6. physical design 7. data dictionary 8. er Diagram 9. database administration 10. system information 11. DBMS configuration 12. Support software required 13. Storage requirement 14. Backup and recovery | **23 - 33** |
| 5 | Detailed design   1. Introduction 2. Structure of the software package 3. Modular decomposition 4. admin 5. Login 6. Inputs 7. Procedural details 8. File i/o interfaces 9. Outputs   b. Shopowner management   1. Inputs 2. Procedural details 3. File i/o interfaces 4. Outputs   c. Customer management   1. Inputs 2. Procedural details 3. File i/o interfaces 4. Outputs   f. Feedback   1. Inputs 2. Procedural details 3. File i/o interfaces 4. Outputs 5. Shopowner 6. Registration 7. Inputs 8. Procedural details 9. File i/o interfaces 10. Outputs 11. Login   a. inputs  b. Procedural details  c. File i/o interfaces  d. Outputs  c. View feedback  a. inputs  b. Procedural details  c. File i/o interfaces  d. Outputs  d. Add products  a. inputs  b. Procedural details  c. File i/o interfaces  d. Outputs     1. Customer 2. Registration 3. Inputs 4. Procedural details 5. File i/o interfaces 6. Outputs 7. Login 8. Inputs 9. Procedural details 10. File i/o interfaces 11. Outputs 12. Feedback   a. inputs  b. Procedural details  c. File i/o interfaces  d. Outputs   1. Add to cart   a. inputs  b. Procedural details  c. File i/o interfaces  d. Outputs  e. View products  a. inputs  b. Procedural details  c. File i/o interfaces  d. Outputs  f. Payment module  a. inputs  b. Procedural details  c. File i/o interfaces  d. Outputs | **34 - 47** |
| 6 | Program coding listing   1. database connection 2. Admin Login 3. Customer Login 4. Shopowner Login 5. Add Products 6. Upload Blog 7. Add to Cart 8. Send feedback 9. Order Payment 10. Update Cart 11. Update Profile | **48 - 60** |
| 7 | User interface   1. Home Page 2. Admin Login 3. Customer Login 4. Shopowner Login 5. Admin home page 6. Blog page 7. Products page 8. Customer Profile Page 9. Feedbacks 10. Shopping Cart 11. Checkout Form 12. Order Status | **61 - 66** |
| 8 | Testing   1. Introduction 2. Testing Objectives 3. Testing Methods 4. Testing Steps 5. Test Cases:  * Admin Login Form * Admin Home Page * Adding Customer * Adding Shopowner  1. System Testing 2. System Testing Tables | **67 - 76** |
| 9 | Conclusion | **77** |
| 10 | Limitations | **78** |
| 11 | Scope for enhancement (future scope) | **79** |
| 12 | Abbreviations and Acronyms (list) | **80** |
| 13 | Bibliography / References (list in specified format) | **81** |

List of figures

|  |  |  |  |
| --- | --- | --- | --- |
| S.no | Figure no. | Figure name | Page no. |
| 1 | 1.1 | structure of theSystem | **2** |
| 2 | 1.2 | System architecture | **2** |
| 3 | 3.3.1 | System software architecture | **12** |
| 4 | 3.3.2 | System technical architecture | **13** |
| 5 | 3.3.3 | System hardware architecture | **13** |
| 6 | 3.4.1 | Context Flow Diagram | **14** |
| 7 | 3.4.2 | DFD level 1 for admin | **15** |
| 8 | 3.4.3 | DFD level 1 for shopkeeper | **16** |
| 9 | 3.4.4 | DFD level 1 for customer | **17** |
| 10 | 3.4.5 | DFD level 2 for feedback | **18** |
| 11 | 3.4.6 | DFD level 2 for add products | **18** |
| 12 | 5.1 | functional component 1 for admin | **35** |
| 13 | 5.2 | functional component 1 forshopowner | **35** |
| 14 | 5.3 | functional component 1 forcustomer | **36** |
| 15 | 5.4 | Procedural details of admin login | **37** |
| 16 | 5.5 | Procedural details of shopowner management | **38** |
| 17 | 5.6 | Procedural details of customer management | **39** |
| 18 | 5.9 | Procedural details of feedback | **40** |
| 19 | 5.10 | Procedural details of shopowner login | **41** |
| 20 | 5.11 | Procedural details of add products | **42** |
| 21 | 5.13 | Procedural details of feedback view | **43** |
| 22 | 5.14 | Procedural details of customer login | **44** |
| 23 | 5.15 | Procedural details of add to cart | **45** |
| 24 | 5.16 | Procedural details of view products | **46** |
| 25 | 5.17 | Procedural details of payment | **47** |

List of tables

|  |  |  |  |
| --- | --- | --- | --- |
| s.no | Table number | Table name | Page.no |
| 1 | 4.1 | admin | **27** |
| 2 | 4.2 | blog | **27** |
| 3 | 4.3 | booking | **28** |
| 4 | 4.4 | cart | **28** |
| 5 | 4.5 | customer | **28 - 29** |
| 6 | 4.6 | feedback | **29** |
| 7 | 4.7 | products | **29** |
| 8 | 4.8 | Requested\_services | **30** |
| 9 | 4.9 | service | **30** |
| 10 | 4.10 | shopowner | **30 - 31** |
| 11 | 4.11 | orders | **31** |
| 12 | 4.12 | Orders\_details | **31 - 32** |
| 13 | 4.13 | payment | **32** |

**INTRODUCTION**

i**. Introduction of the system**

1. **project title**

**WATCH WORLD**

1. **category**

PHP, MYSQL, Apache

1. **overview**

WATCH WORLD is an online web application system which provides online shopping centre for buying different watches by sitting at the comfort of their homes. Customers can register and give feedback for the products. Various shopkeepers can register themselves on the application and add their products and services , which will be available for the customers to buy.

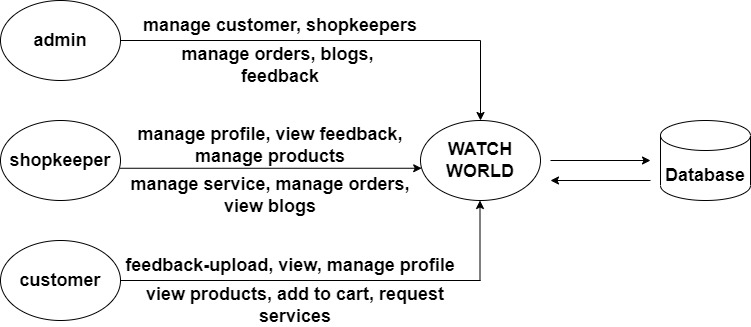
ii**. OBJECTIVE**

* The primary objective of the online shopping centre is to make the experience of buying watches easy.
* One can go through the various available options of watches and select the one which they like and add it to the cart.
* Customers can also provide feedback about the products.
* Customers can also request for the available services provided by the shops.

iii**. SCOPE OF THE SYSTEM**

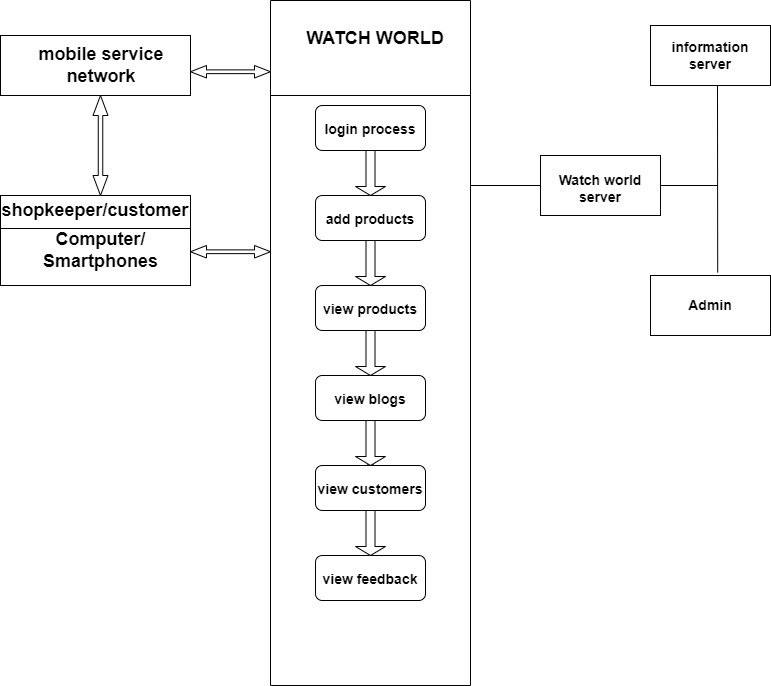
1. Information is online, efficient and systematic.
2. Customers can provide their valuable feedback about the products which may help other customers.
3. Since the system is online, it can be accessed from anywhere with the help of the internet connection.

iv.**STRUCTURE OF THE SYSTEM**

****

**Figure 1.1 structure of the System**

**V.** **SYSTEM ARCCHITECTURE**

****

**Figure 1.2 System architecture**

**VI. END USERS**

a. SHOPKEEPERS

b. CUSTOMERS

**VII. HARDWARE AND SOFTWARE SPECIFICATION**

**SOFTWARE REQUIREMENTS**

* User Interface Design: HTML, CSS, JavaScript
* Server-side Scripting: PHP
* Backend: MySQL.
* Operating system: Windows
* Text editor: Sublime Text 3
* Web server: Apache
* Database server: MySQL
* Browsers**:** Chrome, Firefox or any other browsing application

**HARDWARE REQUIREMENTS**

* Processor: Intel dual core or above
* Processor speed: 2GHz
* RAM – 1 GB
* Hard Disk – Minimum 40 GB

**S.R.S.**

i.) **Introduction**

A software requirements specification (SRS) is a document that captures complete description about how the system is expected to perform. Software requirements specification establishes the basis for an agreement between customers and contractors or suppliers on how the software product should function. Software requirements specification is a rigorous assessment of requirements before the more specific system design stages, and its goal is to reduce later redesign.

ii.) **OVERALL DESCRIPTION**

The following details describe the overall functionality of the web application “**WATCH WORLD**”.

a.) **PRODUCT PERSPECTIVE**

The **WATCH WORLD** is a fully independent product and not a part of any other system. The users of the system are categorized as Admin, Customers and Shopkeepers. The system gives online access to each separate user. This system provides an easy way for customers to buy watches with the best quality and affordable price. This product is self contained. It provides a simple database rather than complex ones for high requirements and it provides a good and easy graphical user interface(GUI) to both new as well as experienced users of the system. The main purpose of this system is to allow customers to check different watches from different stores by sitting at home.

b.) **PRODUCT FUNCTIONS**

* Provides an online platform for both the shopkeeper and the customers to sell and view different watches respectively.
* All the details related to watches from different shops can be fund at the same place.
* The shopkeeper can add and remove watches from their stores.
* Admin can upload blogs.
* Shopkeepers and Admin can view the feedback.
* Customers can give feedback about the products.
* Shopkeepers can provide various services to the customers.
* Admin can keep a track of the sales of the watches of all the shops.
* Customers can rent the watches.

c.) **USER CHARACTERISTICS**

* There are essentially three classes of users for **WATCH WORLD** namely admin, customer and shopkeeper.
* Admin approves shopkeeper and customer registration.
* Admin can keep a track of the sales of the watches of all the shops.
* Admin can upload blogs.
* Shopkeeper can manage its profile.
* The shopkeeper can add and remove watches from their stores.
* Shopkeeper can view the feedback.
* Shopkeeper can manage the services.
* Customers can manage his profile.
* Customers can add products to their cart.
* Customers can give feedback about the products.
* Customers can request for available services.
* Customers can view feedback of other customers.

d.) **GENERAL CONSTRAINTS**

* Must be coded efficiently enough to run well on provided server hardware.
* Client side code or web pages must be able to run efficiently on low end hardware.
* The database will be created and maintained in a way that makes it of reasonable and manageable size.

e.) **ASSUMPTIONS**

* Authentication is provided for this application. Only registered Users can access.
* Every user must enter valid login id and password to use this web application.
* It can be accessed over the Internet.
* This website is a centralized system.

iii.) **Special Requirements**

* All the interactions of the software with people, hardware and other software are specified here.
* The system should be simple and easy to understand as well as to use.
* The system must be able to run without any complications.

a.) **USER INTERFACE:**

* To make this application useful for maximum reach of user we have created very user-friendly interfaces. User with basic skill of computer can use this system.
* The system provides a user-friendly GUI to the users.
* The shopkeeper can add and remove watches from their stores.
* An error messages is raised when invalid performance occurred.

b.) **HARDWARE INTERFACE:**

* Processor: Pentium® dual core or above
* RAM: 2GB and above
* Hard disk Utilization: 320GB and above
* Input Devices: Mouse, Keyboard
* Output Devices: Monitor, Printer

c.) **Software INTERFACE:**

* Browser: Microsoft Edge, Google chrome, Mozilla Firefox.
* Server: Apache Server.
* IDE: Sublime Editor.

iv.) **Functional Requirements:**

a.) **Admin Module:**

The module can be accessed by the admin only. Shopkeeper or Customers cannot access this module. Sub modules are:

* **Login**: Admin can login with username and password.
* **Shopkeepers Management:** Admin can view the details of all shopkeepers registered through the website, verify them and has authority to remove the shopkeepers from the website.
* **Customer Management:** Admin can view the details of all the Customers registered through the website.
* **Feedback module:** Admin can view the feedback of customers.
* **Manage Blogs:** Admin can add blogs.
* **Manage Orders:** Admin can view the monthly sales of the different shops.

b.) **Shopkeeper Module:**

The shopkeeper will have to register to the application in order access the application. Shopkeeper has following functions:

* **Registration:** Shopkeeper needs to give primary information to the system for authorization and then he can use the web application.
* **Login**: Shopkeeper can login with username and password.
* **Feedback:** Shopkeeper can view feedback of the customers about a product.
* **Manage Profile:** Shopkeeper can manage the shop details.
* **Manage Products:** Shopkeeper can add and remove watches from their stores.
* **Manage Orders:** Shopkeeper can view the order details.
* **Manage Service:** Shopkeeper can add and remove the services.
* **Manage Rental:** Shopkeeper can see the requests of the customers for renting the products.
* **View Blogs:** Shopkeeper can view the blogs.

c.) **Customer Module:**

Customer has to register to the application in order access the application. Customer has following functions:

* **Registration:** Customer needs to give primary information to the system for authorization and then he can use the web application.
* **Login**: Customer can login with username and password.
* **Feedback:** Customers can upload a feedback about a product.
* **Manage Profile:** Customers can manage their profile details.
* **View products:** Customers can view the different products.
* **Add to cart:** Customers can add products to their cart.
* **Request services:** Customers can request for available services.
* **View feedback:** Customers can view feedback of other customers.

v.) **Design Constraints:**

* This section lists the design requirements for the “**ADVATECH”** application.
* The user interface (UI) must have specific fonts and font sizes.
* The system shall match the fonts and font sizes used for all the pages of the application.
* The information of all users and subjects must be stored in a database that is accessible by the website.
* This system will be running 24 hours a day.
* Users may access from any computer that has internet browsing capabilities, internet connection by entering accurate login ID.

vi.) **System Attributes:**

The Quality of the website is maintained in such a way so that it can be very user friendly to all the users of the website.

* **Reliability**: Good validation of user inputs will be done to avoid entering incorrect username and password.
* **Maintainability:** During maintenance stage, the SRS can be referred for the validation.
* **Portability:** This system can be run in any operating system and browser.
* **Compatibility:** This System will be Compatible with almost all the web servers.
* **Flexibility:** The system keeps on updating the data according to the changes that takes place.
* **Timeliness:** The system carries out all the operations with consumptions of very less time.
* **Security**: Each time there is a security violation; System restricts the user from accessing that function.

vii.) **Other Requirements**

**Safety Requirements:**

* Consistency: Checking the fact that all clients must be attached to one server, so there is an appropriate control of the information.
* Authorization: Checking for the entity and provide features for them.

**Security Requirements:**

* The proposed website is secured.
* There are different categories of users namely administrator, Shopkeeper and Customer.
* Depending upon the user, access rights are decided.
* If the user is an administrator then he can modify and delete the data.
* Customer cannot upload any products nor he can modify the products of the shops.

**SYSTEM DESIGN**

**i. INTRODUCTION**

Systems design is the process of defining the architecture, modules, interfaces, and data for a system to satisfy specified requirements. Systems design could be seen as the application of systems theory to product development. Systems design implies a systematic approach to the design of a system. It may take a bottom-up or top-down approach, but either way the process is systematic wherein it takes into account all related variables of the system that needs to be created—from the architecture, to the required hardware and software, right down to the data and how it travels and transforms throughout its travel through the system.

**ii. ASSUMPTIONS AND CONSTRAINTS:**

**ASSUMPTIONS**

One Assumption about the product is that it will always be used on systems that have enough performance, the system does not have enough hardware resource available for the application, there may be scenarios where the application does not work as intended or even at all.

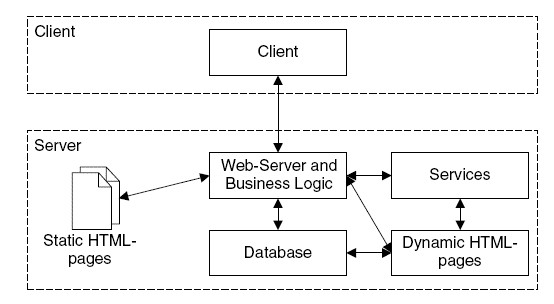
**CONSTRAINTS**

* The developed system should run under any platform i.e. Unix, Linux, Mac, Windows etc.
* All mandatory fields should be filled by an individual.
* There can be security risks involved.
* Details provided by the individual during his sign up stored on database.
* More space is required to keep all the records. Database should not be overloaded.

1. **FUNCTIONAL DECOMPOSITION**

**a. SYSTEM SOFTWARE ARCHITECTURE**

The XAMPP software has following software architecture:



**Figure 3.3.1 System Software Architecture**

XAMPP stands for Cross-platform, Apache MySQL and Perl. It is an open-source package of web solutions that includes Apache distribution for many servers and command line executables along with modules such as Apache server, PHP and Perl.

XAMPP helps a local host or server to tests its website and clients via computers and laptops before releasing it to the main server. it is a platform that furnishes a suitable environment to test and verify the working of projects based on Apache, Perl and MySQL database and PHP through the system of the host itself.

**b. SYSTEM TECHNICAL ARCHITECTURE**

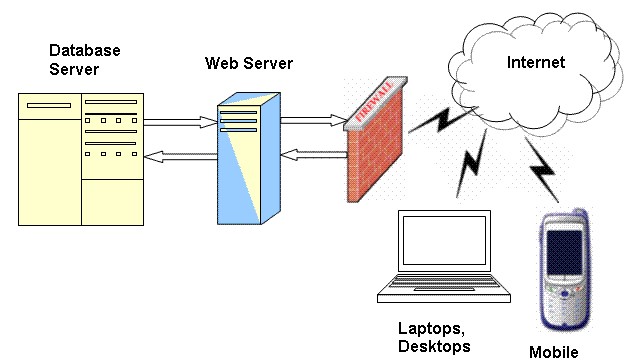
The Apache server used to execute PHP programming and MYSQL server used to store database records. The project “ADVATECH” works under following technical



**3.3.2 System Technical Architecture**

**c. SYSTEM HARDWARE ARCHITECTURE**

**Hardware** architecture is primarily concerned with internal electrical interfaces among the systems components or subsystems, and the interface between the system and its external environment, especially the devices operated by or the electronic displays viewed by the user.



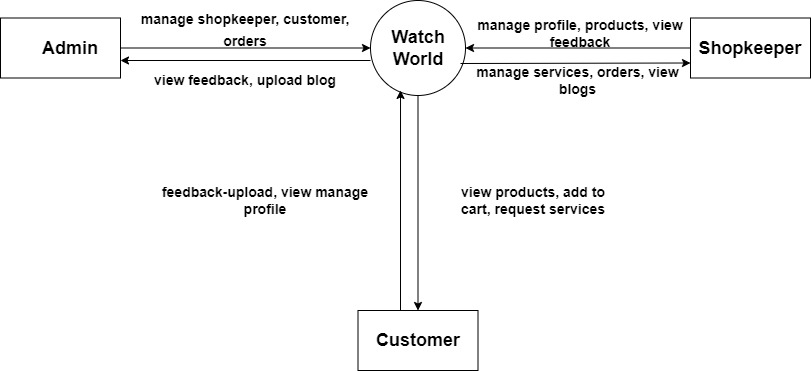
**Figure 3.3.3 System Hardware Architecture**

**d. EXTERNAL INTERFACES**

* **Name of the application**: **WATCH WORLD**
* **Details of interface**: Admin interface, Shopkeeper interface, Customer interfac**e.**
* **Type of Interface**: menu driven interface, form-based interface.

**iv. Description of Programs**

**a. CONTEXT FLOW DIAGRAM**



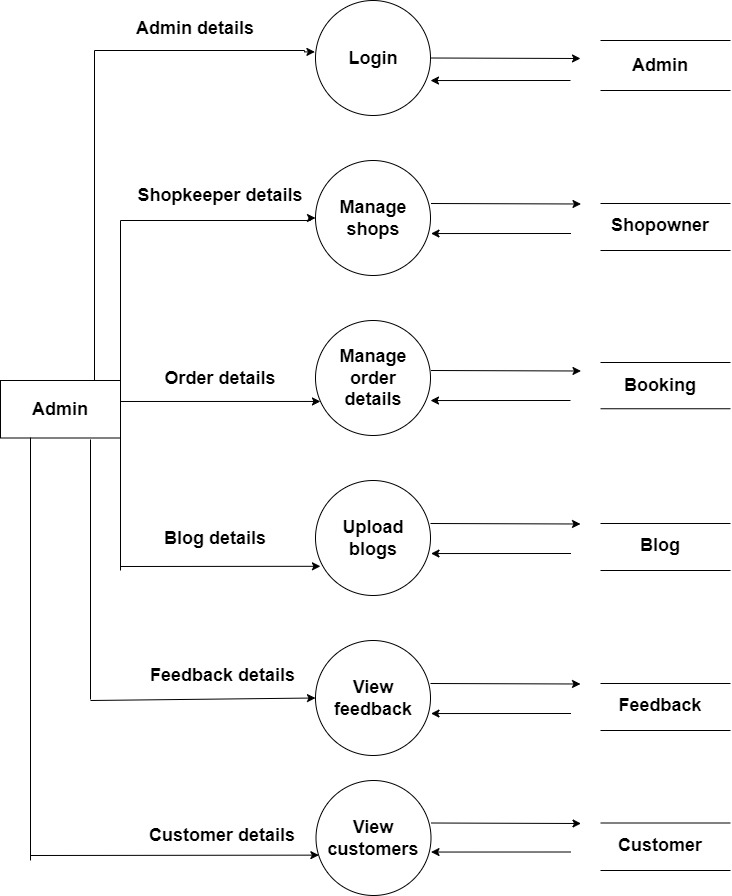
**Figure 3.4.1 context flow diagram**

**b. DFD**

**DFD LEVEL 1 FOR SYSTEM**

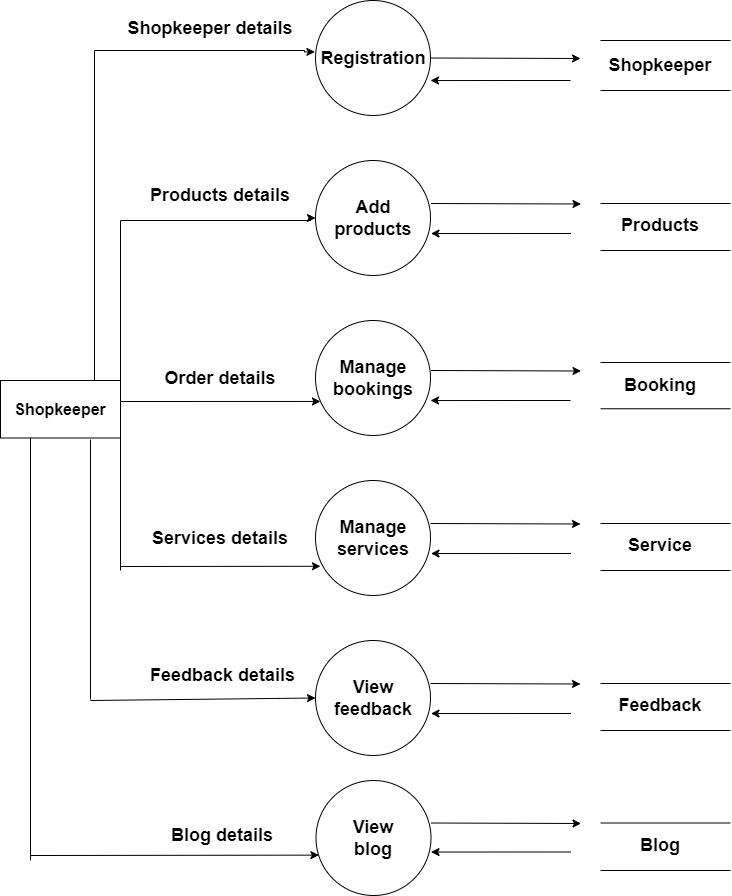
****

**DFD LEVEL 1 FOR ADMIN**



**Figure 3.4.2 DFD level 1 for admin**

**DFD LEVEL 1 FOR SHOPKEEPER**

****

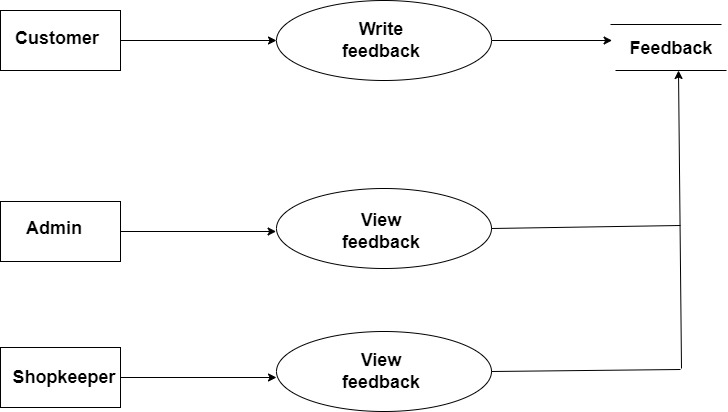
**Figure 3.4.3 DFD level 1 for shopkeeper**

**DFD LEVEL 1 FOR CUSTOMER**

****

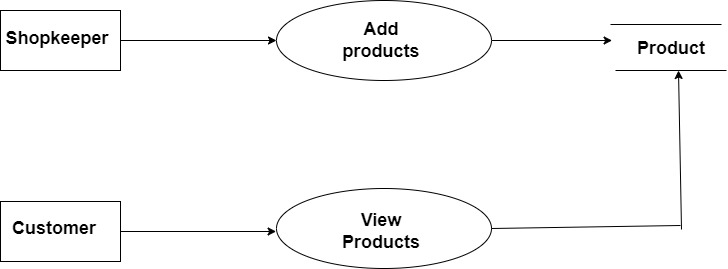
**Figure 3.4.4 DFD level 1 for customer**

**DFD LEVEL 2 FOR FEEDBACK**

****

**Figure 3.4.6 DFD level 2 for feedback**

**DFD LEVEL 2 FOR ADD PRODUCTS**

****

**Figure 3.4.7 DFD level 2 for add products**

**v. DESCRIPTION OF COMPONENTS**

**. MODULAR DECOMPOSITION COMPONENTS**

1. **Admin:**

**Login:**

**Input:** username, password

**File I/O interface:** Admin Table

**Output:** Entered Username and password will be checked for validity. If it is valid admin will be directed to Homepage.

**Shopowner management:**

**Input**: shopowner id, shopowner name

**File I/O interface:** ShopownerTable

**Output:** Admin can view the details of all shopowners registered through the website, verify them and has authority to remove the shopowners from the website.

**Customer management:**

**Input:** customer id, customer name

**File I/O interface:** customer Table

**Output:** Admin can view the details of all the customers registered through the website, verify them and has authority to remove the customers from the website.

**Feedback module:**

**Input: f** \_id, c\_id.

**File I/O interface:** feedback Table

**Output:** this will let the admin know about all the details.

**Manage orders module:**

**Input:** p \_id, b\_id, c\_id.

**File I/O interface:** booking Table

**Output:** this will let the admin know about all the details.

**Upload blogs module:**

**Input:**  bl\_id.

**File I/O interface:** blog Table

**Output:** this will let the admin know about all the details.

1. **Shopowner:**

**Registration**

**Input:** shopowner name,email,address,contact,password.

**File I/O interface:** shopowner Table

**Output:** if approved by admin,then shopowner will be directed to login page

**Login**

**Input:** username, password.

**File I/O interface:** shopowner Table

**Output:** Entered Username and password will be checked for validity.

If it is valid shopowner will be directed to Homepage.

**Feedback module:**

**Input:** f\_id, c\_id.

**File I/O interface:** feedback Table.

**Output:** the shopowner can view the feedback.

**Add products module:**

**Input:** p\_id, p\_name.

**File I/O interface:** product Table.

**Output:** the shopowner can add the products.

**Manage booking module:**

**Input:** b\_id, p\_id, c\_id.

**File I/O interface:** booking Table.

**Output:** the shopowner can view the orders.

**Manage services module:**

**Input:** s\_id, dis.

**File I/O interface:** service Table.

**Output:** the shopowner can view the requested services and

add more services.

1. **customer:**

**Registration**

**Input:** customer name, contact no, email

**File I/O interface:** customer Table

**Output:** open home page

**Login:**

**Input:** username, password

**File I/O interface:** customer Table

**OUTPUT:** Entered Username and password will be checked for validity. If it is valid customer will be directed to Homepage.

**Feedback**

**Input:** f\_id.

**File I/O interface:** feedback Table

**Output:**. He can write the feedback.

**Add to cart**

**Input:** p\_id, c\_id, qty, price.

**File I/O interface:** Cart Table

**Output:**. He can add products to the cart.

**View Products**

**Input:** p\_id.

**File I/O interface:** Products Table

**Output:**. He can view the products available for buying.

**Request services**

**Input: s**\_id, c\_id.

**File I/O interface:** service Table

**Output:**. He can request for the available services.

**View blogs**

**Input:** bl\_id.

**File I/O interface:** blog Table

**Output:**. He can view the blogs that are uploaded by the admin.

**Payment:**

**Input:** pay \_id, od\_id, p\_id, c\_id

**File I/O interface:** payment Table

**Output: it** will direct to a page where he can view his payment details and make a payment.

**DATABASE DESIGN**

**a) INTRODUCTION**

Database is an organized collection of data, generally stored and accessed electronically through computer systems. Database design is a collection of tasks or processes that enhance the designing, development, implementation, and maintenance data management system. The main objective of database design is to produce physical and logical design models of the proposed data-based system. The logical model is primarily concentrated on the requirement of the data. Physical data base design models include a translation of the logical design model of the database by keep control of physical media using hardware resources and software systems such as DBMS.

**b) PURPOSE AND SCOPE**

The ultimate purpose of a database management system is to store and transform data into information to support making decisions. The physical database: the collection of files that contain the data. The database engine: the software that makes it possible to access and modify the contents of the database

**4.2.1 PURPOSE**

This database requirement specification describes the function and performance requirements by the WATCHWORLD.

This database stores

* Admin login records.
* The shopowner and customer details.
* Blog details.
* Booking details.
* Cart details.
* Feedback details.
* Product details.
* Requested service details.
* Service details.
* Order details.
* Payment details.
* Ordered products details.

**iii. DATABASE IDENTIFICATION**

* A Primary key is a special relational database table column designed to uniquely identify each table record.

A FORIEGN KEY is a column that is used to establish and enforce a link between the two tables to control the data that can be stored in a FORIEGN KEY table.

* The database consists of tables, each of which has columns and rows. Each row is a dataset that applies to a single item, and each column contains characteristics that describe the rows. In the database, these columns are called as attribute.
* Primary key and FORIEGN KEY are defined with same name.
* Make a separate table for each set of related attributes, and give each table a primary key.
* If an attribute depends on only part of a multi-valued key, remove it to a separate table.
* If attributes do not contribute to a description of the key, remove them to a separate table

## iv. SCHEMA INFORMATION

A database schema represents the logical configuration of all or part of a relational database. it can exist both as a visual representation and as asset of formulas known as integrity constraints that govern a database. These formulas are expressed in a data definition language such as SQL.

A database schema is the skeleton structure that represents the logical view of the entire database. It defines how the data is organized and how the relations among them are associated. It formulates all the constraints that are to be applied on the data.

A database schema defines its entities and the relationship among them. It contains a descriptive detail of the database, which can be depicted by means of schema Diagrams. It’s the database designers who design the schema to help programmers understand the database and make it useful.

**v. TABLE DEFINITION**

**TABLE NAME: admin**

|  |  |  |
| --- | --- | --- |
| username | password | a\_id |

**TABLE NAME: blog**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| bl\_id | title | des | image | date |

**TABLE NAME: ordered products**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| b\_id | c\_id | p\_id | quantity | amount |

**TABLE NAME: cart**

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| cart\_id | c\_id | p\_id | quantity | price | date |

**TABLE NAME: customer**

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| c\_id | username | password | c\_name | email | address | contact | image | status |

**TABLE NAME: feedback**

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| f\_id | c\_id | message | name | email | subject | date |

**TABLE NAME:products**

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| p\_id | p\_name | p\_price | p\_description | p\_image | stock |

**TABLE NAME:requested\_services**

|  |  |  |  |
| --- | --- | --- | --- |
| c\_id | s\_id | sr\_name | email |

**TABLE NAME:service**

|  |  |  |  |
| --- | --- | --- | --- |
| s\_id | s\_name | s\_price | s\_description |

**TABLE NAME:shopowner**

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| s\_id | username | password | so\_name | email | address | contact | image | status |

**TABLE NAME: orders**

|  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| o\_id | total\_amt | c\_id | f\_name | l\_name | addrs1 | addrs2 | state | pincode | contact | status |

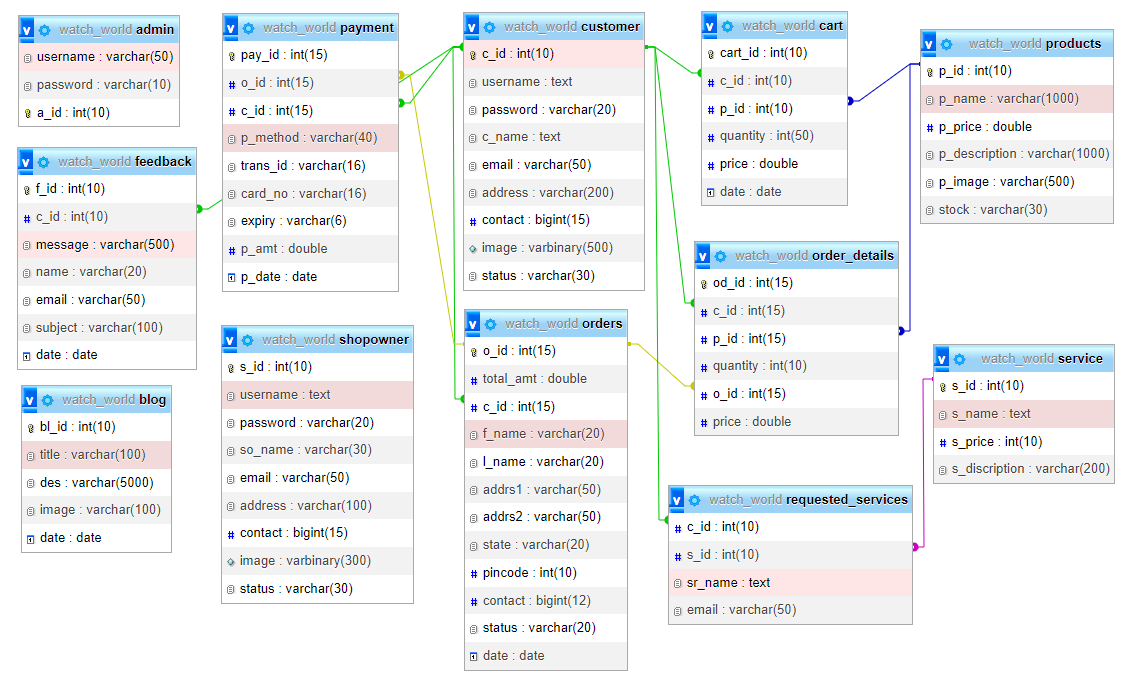
**TABLE NAME: orders\_details**

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| od\_id | c\_id | p\_id | quantity | o\_id | price |

**TABLE NAME: payment**

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| pay\_id | o\_id | c\_id | p\_method | trans\_id | card\_no | expiry | p\_amt | p\_date |

**vi. PHYSICAL DESIGN**

****

**vii. DATA DICTIONARY**

**TABLE STRUCTURE**

The database watch\_world is organized into following tables:

* admin
* blog
* booking
* cart
* customer
* feedback
* products
* rent
* requested\_services
* shopowner

**TABLE NAME: admin**

This table is used to store the login details of admin

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **ADMIN TABLE** | | | | | |
| SL NO | FIELD NAME | DATA TYPE | SIZE | CONSTRAINTS | DESCRIPTION |
| 1 | username | varchar | 50 | PRIMARY KEY | Admin’s user id |
| 2 | password | varchar | 10 | NOT NULL | Admin’s password |
| **3** | a\_id | int | 10 | NOT NULL | Admin id |

**TABLE NAME: blog**

This table is used to store blog details.

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **BLOG TABLE** | | | | | |
| SL NO | FIELD NAME | DATA TYPE | SIZE | CONSTRAINTS | DESCRIPTION |
| 1 | bl\_id | int | 10 | PRIMARY KEY | Blog id |
| 2 | title | varchar | 5000 | NOT NULL | Title of blog |
| 3 | des | varchar | 100 | NOT NULL | Description |
| 4 | image | varchar | 100 | NOT NULL | Image |
| 5 | date | date | date | NOT NULL | Date |

**TABLE NAME: booking**

This table is used to store booking details

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **BOOKING TABLE** | | | | | |
| SL NO | FIELD NAME | DATA TYPE | SIZE | CONSTRAINTS | DESCRIPTION |
| 1 | b\_id | int | 10 | PRIMARY KEY | Booking id |
| 2 | c\_Id | int | 10 | FOREIGN KEY | Customer id |
| 3 | p\_id | int | 10 | FOREIGN KEY | Product id |
| 4 | quantity | int | 20 | NOT NULL | Quantity |
| 5 | amount | int | 20 | NOT NULL | Amount |

**TABLE NAME: cart**

This table is used to store booking details

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **CART TABLE** | | | | | |
| SL NO | FIELD NAME | DATA TYPE | SIZE | CONSTRAINTS | DESCRIPTION |
| 1 | cart\_id | int | 10 | PRIMARY KEY | Cart id |
| 2 | c\_id | int | 10 | FOREIGN KEY | Customer id |
| 3 | p\_id | int | 10 | FOREIGN KEY | Product id |
| 4 | quantity | int | 50 | NOT NULL | Quantity |
| 5 | price | double |  | NOT NULL | Price |
| 6 | date | date |  | NOT NULL | Date |

**TABLE NAME: customer**

This table is used to store customer details

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **CUSTOMER TABLE** | | | | | |
| SL NO | FIELD NAME | DATA TYPE | SIZE | CONSTRAINTS | DESCRIPTION |
| 1 | c\_id | int | 10 | PRIMARY KEY | Customer id |
| 2 | username | text |  | NOT NULL | Username |
| 3 | password | varchar | 20 | NOT NULL | Password |
| 4 | c\_name | text |  | NOT NULL | Customer name |
| 5 | email | varchar | 50 | NOT NULL | Email |
| 6 | address | varchar | 200 | NOT NULL | Address |
| 7 | contact | int | 15 | NOT NULL | Contact |
| 8 | image | varbinary | 500 | NOT NULL | Image |
| 9 | status | varchar | 30 | NOT NULL | Status |

**TABLE NAME: feedback**

This table is used to store feedback details

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **FEEDBACK TABLE** | | | | | |
| SL NO | FIELD NAME | DATA TYPE | SIZE | CONSTRAINTS | DESCRIPTION |
| 1 | f\_id | int | 10 | PRIMARY KEY | Feedback id |
| 2 | c\_id | int | 10 | FOREIGN KEY | Customer id |
| 3 | message | varchar | 500 | NOT NULL | Message |
| 4 | name | varchar | 20 | NOT NULL | Name |
| 5 | email | varchar | 50 | NOT NULL | Email |
| 6 | subject | varchar | 100 | NOT NULL | Subject |
| 7 | date | date |  | NOT NULL | Date |

**TABLE NAME: products**

This table is used to store products details

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **PRODUCTS TABLE** | | | | | |
| SL NO | FIELD NAME | DATA TYPE | SIZE | CONSTRAINTS | DESCRIPTION |
| 1 | p\_id | int | 10 | PRIMARY KEY | Product id |
| 2 | p\_name | varchar | 1000 | NOT NULL | Product name |
| 3 | p\_price | double |  | NOT NULL | Product price |
| 4 | p\_description | varchar | 1000 | NOT NULL | Description |
| 5 | p\_image | varchar | 500 | NOT NULL | Image |
| 6 | stock | varchar | 30 | NOT NULL | Stock |

**TABLE NAME: requested\_services**

This table is used to store requested services details

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **REQUESTED\_SERVICES TABLE** | | | | | |
| SL NO | FIELD NAME | DATA TYPE | SIZE | CONSTRAINTS | DESCRIPTION |
| 1 | c\_id | int | 10 | FOREIGN KEY | Customer id |
| 2 | s\_id | int | 10 | FOREIGN KEY | Service id |
| 3 | sr\_name | text |  | NOT NULL | Service name |
| 4 | email | varchar | 50 | NOT NULL | Email |

**TABLE NAME: service**

This table is used to store service details

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **SERVICE TABLE** | | | | | |
| SL NO | FIELD NAME | DATA TYPE | SIZE | CONSTRAINTS | DESCRIPTION |
| 1 | s\_id | int | 10 | PRIMARY KEY | Service id |
| 2 | s\_name | text |  | NOT NULL | Service name |
| 3 | s\_price | int | 10 | NOT NULL | Service price |
| 4 | s\_description | varchar | 200 | NOT NULL | Description |

**TABLE NAME: shopowner**

This table is used to store shopowner details

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **SHOPOWNER TABLE** | | | | | |
| SL NO | FIELD NAME | DATA TYPE | SIZE | CONSTRAINTS | DESCRIPTION |
| 1 | s\_id | int | 10 | PRIMARY KEY | Shopowner id |
| 2 | username | text |  | NOT NULL | Username |
| 3 | password | varchar | 20 | NOT NULL | Password |
| 4 | so\_name | varchar | 30 | NOT NULL | Full name |
| 5 | email | varchar | 50 | NOT NULL | Email |
| 6 | address | varchar | 100 | NOT NULL | Address |
| 7 | contact | int | 10 | NOT NULL | Contact |
| 8 | image | varbinary | 300 | NOT NULL | Image |
| 9 | status | varchar | 30 | NOT NULL | status |

**TABLE NAME: orders**

This table is used to store order details.

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **ORDERS TABLE** | | | | | |
| SL NO | FIELD NAME | DATA TYPE | SIZE | CONSTRAINTS | DESCRIPTION |
| 1 | **o\_id** | int | 15 | PRIMARY KEY | Order id |
| 2 | total\_amt | double |  | NOT NULL | Total amount |
| 3 | c\_id | int | 15 | FOREIGN KEY | Customer id |
| 4 | f\_name | varchar | 20 | NOT NULL | Full name |
| 5 | l\_name | varchar | 20 | NOT NULL | Last name |
| 6 | addrs1 | varchar | 50 | NOT NULL | Address 1 |
| 7 | addrs2 | varchar | 50 | NOT NULL | Address 2 |
| 8 | state | varchar | 20 | NOT NULL | State |
| 9 | pincode | int | 10 | NOT NULL | Pincode |
| 10 | contact | bigint | 12 | NOT NULL | Contact |
| 11 | status | varchar | 20 | NOT NULL | Status |
| 12 | date | date |  | NOT NULL | Date |

**TABLE NAME: order\_details**

This table is used to store orders details.

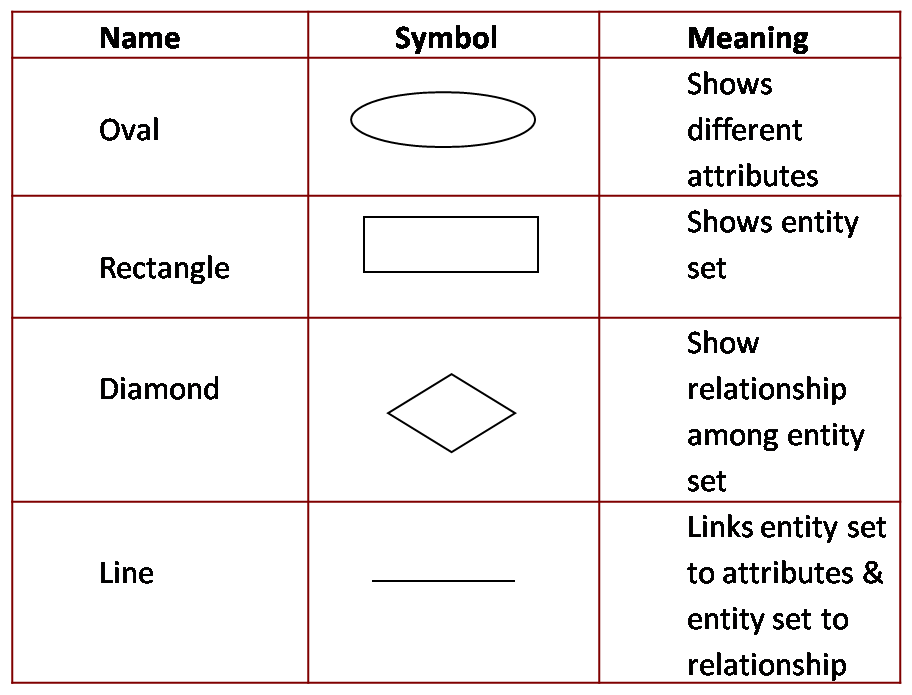
|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **ORDERS TABLE** | | | | | |
| SL NO | FIELD NAME | DATA TYPE | SIZE | CONSTRAINTS | DESCRIPTION |
| 1 | **od\_id** | int | 15 | PRIMARY KEY | Order details id |
| 2 | c\_id | int | 15 | FOREIGN KEY | Customer id |
| 3 | p\_id | int | 15 | FOREIGN KEY | Product id |
| 4 | quantity | int | 10 | NOT NULL | Quantity |
| 5 | o\_id | int | 15 | FOREIGN KEY | Order id |
| 6 | price | double |  | NOT NULL | Price |

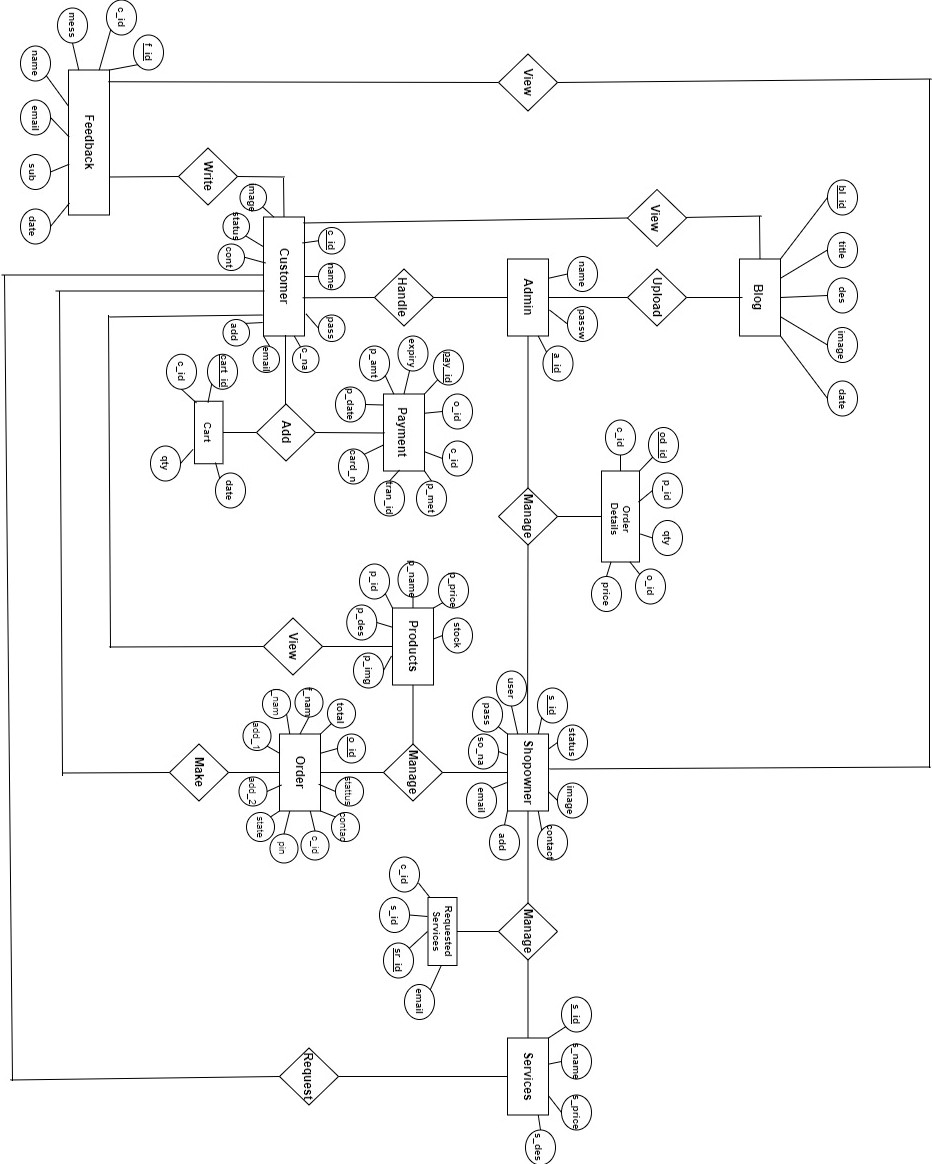
**TABLE NAME: payment**

This table is used to store payment details.

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **PAYMENT TABLE** | | | | | |
| SL NO | FIELD NAME | DATA TYPE | SIZE | CONSTRAINTS | DESCRIPTION |
| 1 | **pay\_id** | int | 15 | PRIMARY KEY | Payment id |
| 2 | o\_id | int | 15 | FOREIGN KEY | Order id |
| 3 | c\_id | int | 15 | FOREIGN KEY | Customer id |
| 4 | p\_method | varchar | 40 | NOT NULL | Payment method |
| 5 | trans\_id | varchar | 16 | FOREIGN KEY | Transaction id |
| 6 | card\_no | varchar | 16 | NOT NULL | Card no |
| 7 | expiry | varchar | 6 | NOT NULL | Expiry date |
| 8 | p\_amt | double |  | NOT NULL | Payment amount |
| 9 | p\_date | date |  | NOT NULL | Payment date |

1. **ER DIAGRAM**

****

****

**DETAIL DESIGN**

**i.) INTRODUCTION**

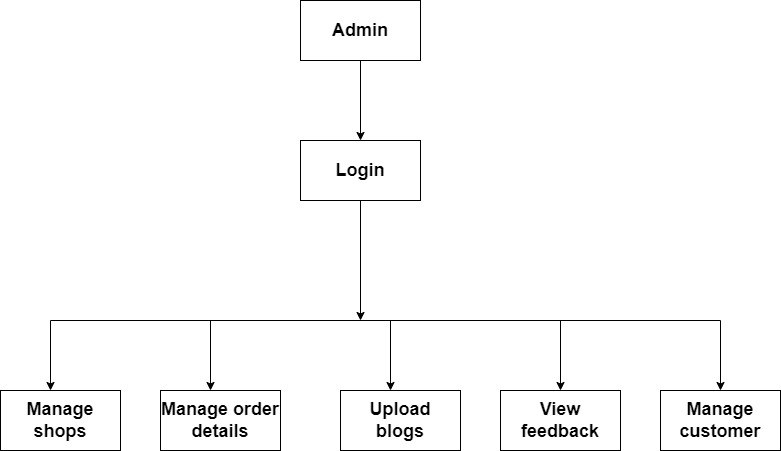
Detailed design is the second level of the design process. During detailed design, we specify how the module in the system interacts with each other and the internal logic of each of the modules specified during system design is decided, hence it is also called as logic design.

Detailed design essentially expands the system design and database design to contain a more detailed description of the processing logic and data structures so that the design is sufficiently complete for coding. Detailed design is the phase where the design is refined and plans, specifications and estimates are created. Detailed design will include outputs such as 2D and 3D models, P & ID's, cost build up estimates, procurement plans etc. This phase is where the full cost of the project is identified.

**ii. STRUCTURE OF THE SOFTWARE PACKAGE**

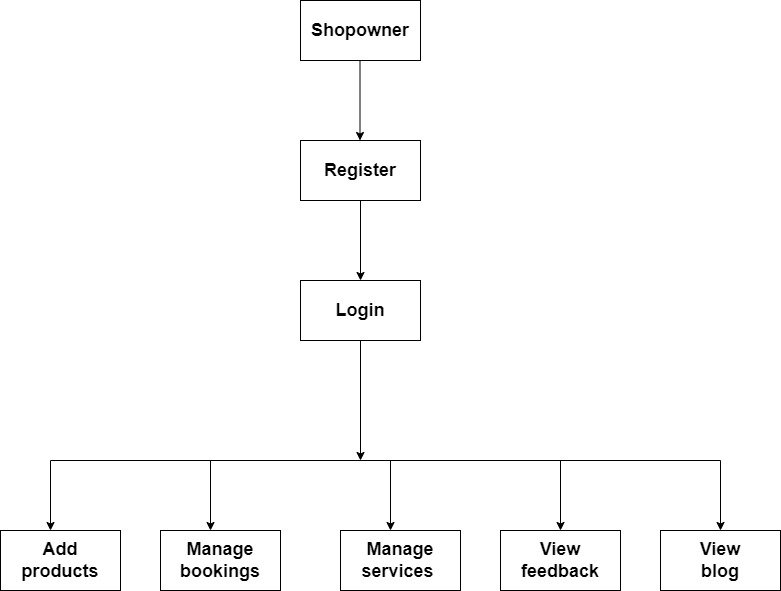
The functional components identified during the system design are listed here:

FUNCTIONAL COMPONENT 1: **ADMIN**

****

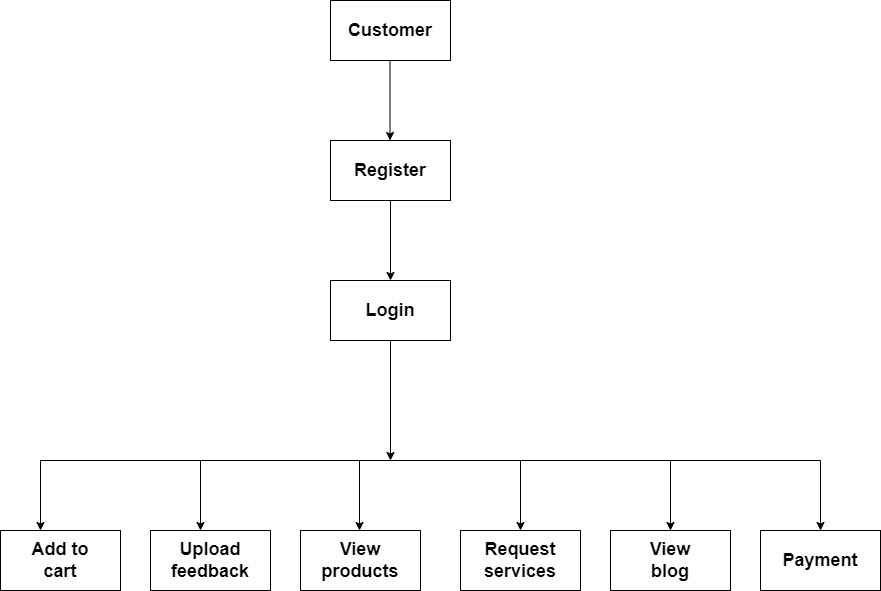
5.1 FUNCTIONAL COMPONENT 1 forADMIN

FUNCTIONAL COMPONENT 2: **SHOPOWNER**

****

5.2 FUNCTIONAL COMPONENT 1 forshopowner.

FUNCTIONAL COMPONENT 3: **CUSTOMER**

****

5.3 FUNCTIONAL COMPONENT 1 forcustomer

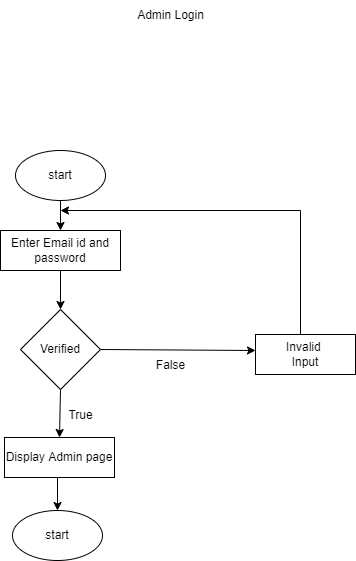
**iii. MODULAR DECOMPOSITION OF THE SYSTEM**

**a. ADMIN:**

**LOGIN:**

**Input:** username, password

**Procedural details:**



**Figure 5.4 procedural details of admin login**

**File I/O interface:** Admin Table

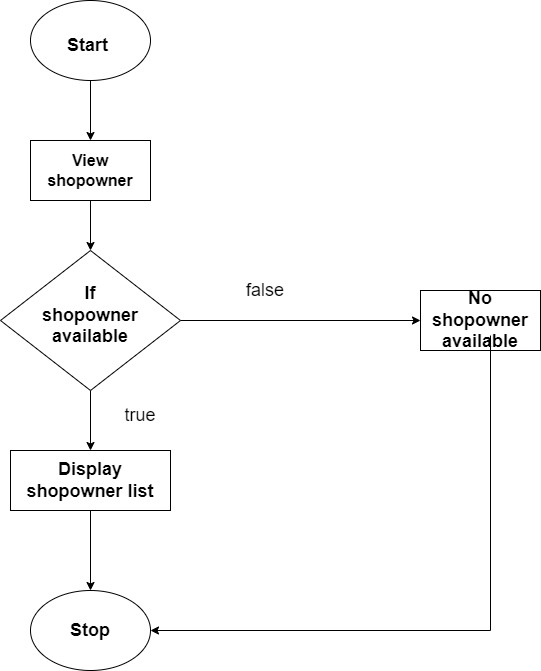
**Output:** Entered Username and password will be checked for validity.

If it is valid admin will be directed to Homepage.

**SHOPOWNER MANAGEMENT:**

**Input**: teacher id, teacher name

**Procedural details:**

****

**Figure 5.5 procedural details of Shopowner management**

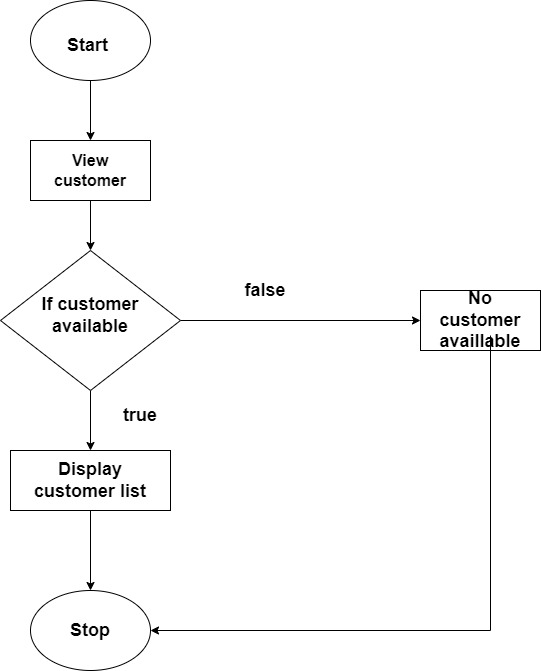
**File I/O interface:** teacher Table

**Output:** Admin can view the details of all teachers registered through the website, verify them and has authority to remove the teachers from the website.

**CUSTOMER MANAGEMENT:**

**Input:** student id, student name

**Procedural details:**

****

**Figure 5.6 procedural details of customer management**

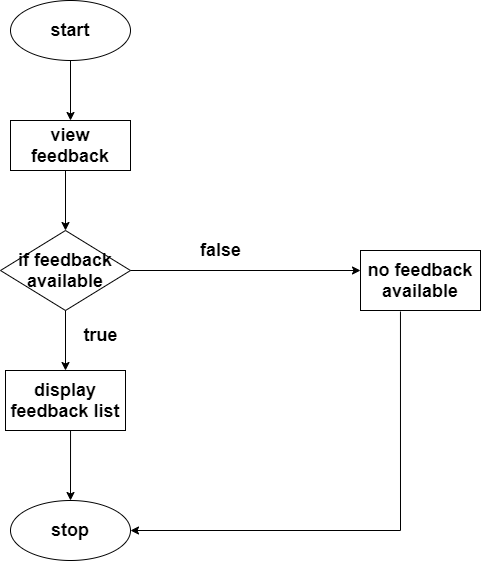
**File I/O interface:** student Table

**Output:** Admin can view the details of all the student registered through the website, verify them and has authority to remove the students from the website.

**FEEDBACK:**

**Input:** feedback\_id, feedback\_date, feedback\_name

**Procedural details:**



**Figure 5.9 procedural details of feedback**

**File I/O interface:** feedback Table

**Output:** Admin can view and respond to feedback and reviews of students

**5.3.2 SHOPOWNER:**

**REGISTRATION:**

**Input:** shopowner name,email,address,contact

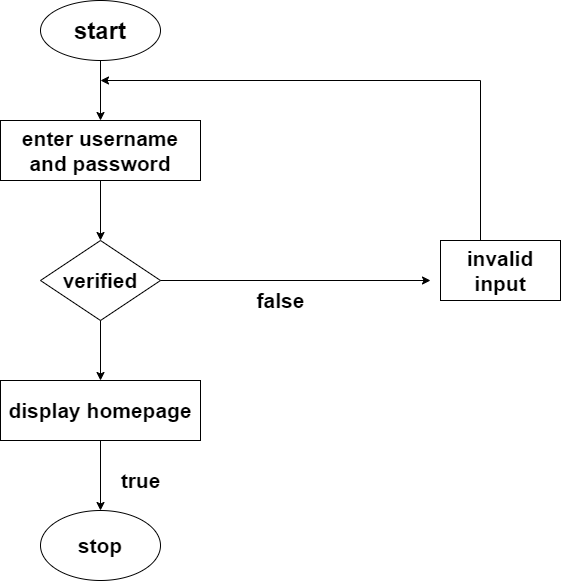
**File I/O interface:** Shopowner Table

**Output:** open registration page for registering

**LOGIN:**

**Input:** username, password

**Procedural details:**

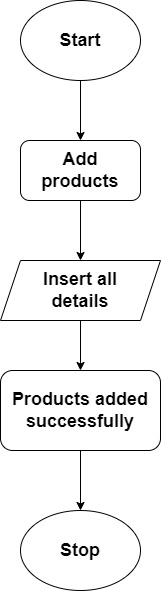


**Figure 5.10 procedural details of shopowner login**

**File I/O interface:** shopowner Table

**Output:** Entered Username and password will be checked for validity.

If it is valid teacher will be directed to Homepage.



**Figure 5.11 procedural details of add products.**

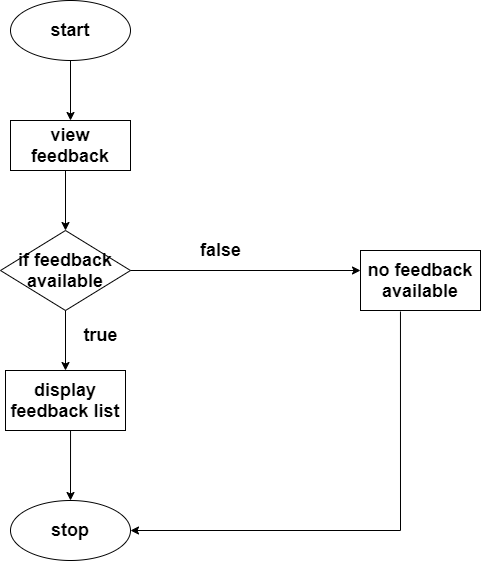
**File I/O interface:** study material Table

**Output:** The teacher can add, edit or delete study materials such as videos, ppt, documents etc

**FEEDBACK**

**Input:** f\_id.

**Procedural details:**



**Figure 5.13 procedural details of feedback view**

**File I/O interface:** feedback Table.

**Output:**, the teacher can view the feedback.

**5.3.3 CUSTOMER:**

**REGISTRATION**

**Input:**, customer name,contact\_no,email.

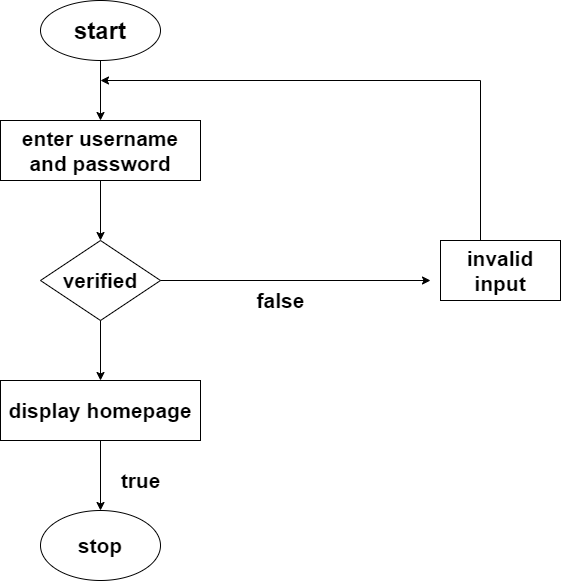
**File I/O interface:** customer Table

**Output:** open registration page

**LOGIN:**

**Input:** username, password

**Procedural details:**



**Figure 5.14 procedural details of customer login**

**File I/O interface:** customer Table

**OUTPUT:** Entered Username and password will be checked for validity.

If it is valid customer will be directed to Homepage.

**FEEDBACK**

**Input:** f\_id.

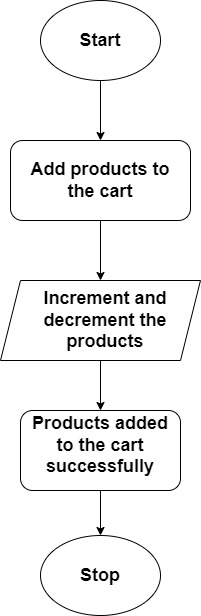
**File I/O interface:** feedback Table

**Output:** He can write the feedback.

**ADD TO CART**

**Input:** p \_id, c\_id.

**Procedural details:**

****

**Figure 5.15 procedural details of add to cart.**

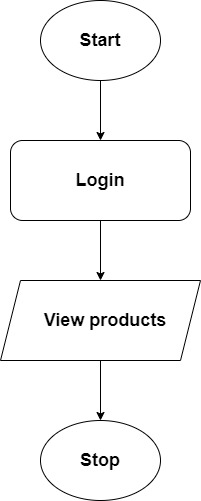
**File I/O interface:** cart Table

**Output:** it will direct to a page where he can add the products in the cart .

**VIEW PRODUCTS:**

**Input:** st\_id,teacher\_id,class\_id

**Procedural details:**

****

**Figure 5.16 procedural details of view products**

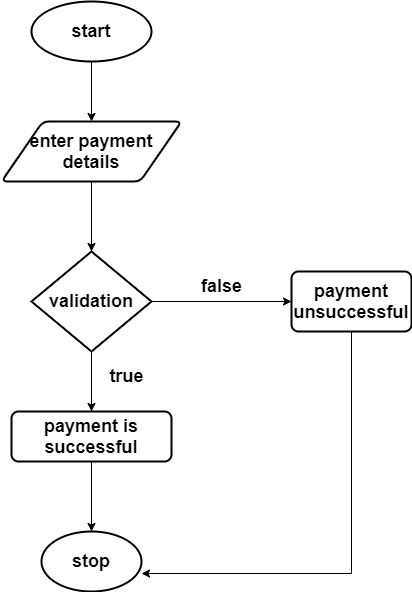
**File I/O interface:** products Table

**Output:** all the products uploaded by the shopowner will be available to him.

**PAYMENT:**

**Input:** pay \_id, c\_id, p\_id.

**Procedural details:**



**Figure 5.17 procedural details of payment**

**File I/O interface:** payment Table

**Output: it** will direct to a page where he can view his payment details and make a payment.

**Program code listing**

**a) Database connection**

<?php

/\*\*

\* Created by PhpStorm.

\* User: your name

\* Date: todays date

\* Time: todays time

\*/

ob\_start();

session\_start();

//=========== database connection variables ====================

define('DB\_SERVER', "localhost"); // database host name eg. localhost or 127.0.0.1

define('DB\_USER', "root"); // database user name eg. root

define('DB\_DATABASE', "watch\_world"); //database name

define('DB\_PASSWORD', ""); //database user password

define('DB\_TYPE', 'mysql'); //database drive eg. mysql, pgsql, mongodb etc

//========== site details described here ========================

define('SITE\_TITLE', 'demo.com');

define('SITE\_TAG\_LINE', 'give your tag line of your project here');

//contact information

define('SITE\_CONTACT', 'your number');

//email information

define('SITE\_EMAIL\_INFO', 'your mail id');

//url information

define('BASE\_URL', 'http://localhost/cud operation/');

// included main class

require\_once 'app/Main.php';

require\_once 'app/Controller.php';

require\_once 'app/Admin.php';

// /\*\*

// \* @param $class

// \*/

// function \_\_autoload($class) {

// require\_once 'app/' . $class.'.php';

// }

**b) Admin login**

<?php

include '../../config.php';

$admin = new Admin();

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

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

$stmt=$admin->ret("select \* from `admin` where `username`='$username' AND `password`='$password'");

$num =$stmt->rowCount();

if ($num>0) {

$row=$stmt->fetch(PDO::FETCH\_ASSOC);

$a\_id=$row['a\_id'];

$\_SESSION['a\_id']=$a\_id;

echo "<script>alert('Login successful..');window.location='../index.php';</script>";

// code...

}

else{

echo "<script>alert('Login Failed..');window.location='../login/login.php';</script>";

}

?>

**c) Customer login**

<?php

include '../config.php';

$admin = new Admin();

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

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

$stmt=$admin->ret("select \* from `customer` where `username`='$username' AND `password`='$password'");

$num =$stmt->rowCount();

if ($num>0) {

$row=$stmt->fetch(PDO::FETCH\_ASSOC);

$c\_id=$row['c\_id'];

$\_SESSION['c\_id']=$c\_id;

echo "<script>alert('Login successful..');window.location='../index.php';</script>";

// code...

}

else{

echo "<script>alert('Login Failed..');window.location='../login/login.php';</script>";

}

?>

**d) Shopowner login**

<?php

include '../../config.php';

$admin = new Admin();

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

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

$stmt=$admin->ret("select \* from `shopowner` where `username`='$username' AND `password`='$password' AND`status`='approved'");

$num =$stmt->rowCount();

if ($num>0) {

$row=$stmt->fetch(PDO::FETCH\_ASSOC);

$s\_id=$row['s\_id'];

$\_SESSION['s\_id']=$s\_id;

echo "<script>alert('Login successful..');window.location='../index.php';</script>";

// code...

}

else{

echo "<script>alert('Your account is pending!');window.location='../login/login.php';</script>";

}

?>

**e) Add Products**

<?php

include '../config.php';

$Admin = new Admin();

$name =$\_POST['product'];

$price =$\_POST['price'];

$description =$\_POST['des'];

$quantity =$\_POST['qty'];

$targetdir='upload/';

$image=$targetdir.basename($\_FILES['image']['name']);

move\_uploaded\_file($\_FILES['image']['tmp\_name'], $image);

$stmt=$Admin->cud("INSERT INTO `products` (`p\_name`,`p\_price`,`p\_description`,`p\_image`,`stock`)

VALUES('$name','$price','$description','$image','$quantity')","Saved");

echo "<script>alert('Uploaded successfully');window.location='../viewproducts.php';</script>/";

?>

**f) Upload Blog**

<?php

include '../../config.php';

$Admin = new Admin();

$des =$\_POST['des'];

$title =$\_POST['title'];

$targetdir='upload/';

$image=$targetdir.basename($\_FILES['image']['name']);

move\_uploaded\_file($\_FILES['image']['tmp\_name'], $image);

$stmt=$Admin->cud("INSERT INTO `blog` (`des`,`image`,`date`,`title`)

VALUES('$des','$image',now(),'$title')","Saved");

echo "<script>alert('Uploaded successfully');window.location='../viewblog.php';</script>";

?>

**g) Add to cart**

<?php

include '../config.php';

$Admin = new Admin();

$price =$\_GET['price'];

$pid =$\_GET['pid'];

$cid =$\_GET['cid'];

$quantity='1';

$stmt=$Admin->cud("INSERT INTO `cart` (`c\_id`,`p\_id`,`quantity`,`price`,`date`)

VALUES('$cid','$pid','$quantity','$price',now())","Saved");

echo "<script>alert(' Product Added To Cart');window.location='../viewcart.php';</script>";

?>

**h) Send Feedback**

<?php

include '../config.php';

$Admin = new Admin();

$message =$\_POST['message'];

$name =$\_POST['name'];

$email =$\_POST['email'];

$subject =$\_POST['subject'];

$cid =$\_POST['cid'];

$stmt=$Admin->cud("INSERT INTO `feedback` (`message`,`c\_id`,`name`,`email`,`subject`,`date`)

VALUES('$message','$cid','$name','$email','$subject',now())","Saved");

echo "<script>alert('Feedback sent successfully');window.location='../feedback.php';</script>";

?>

**i) Order payment**

<?php

include '../config.php';

$admin=new Admin();

$pay=$\_POST['payment\_method'];

$trans1=$\_POST['transaction'];

$cno3=$\_POST['card\_no'];

$expiry3=$\_POST['expiry'];

if($pay==""){

echo"<script>alert('Incomplete Transaction...');window.location='../checkout.php';</script>";

}elseif($pay=='upi' && $trans1==""){

echo"<script>alert('Incomplete Transaction...Give Transaction Id...');window.location='../checkout.php';</script>";

}elseif($pay=='card' && $cno3=="" && $expiry3==""){

echo"<script>alert('Incomplete Transaction...Give Card details...');window.location='../checkout.php';</script>";

}else{

$total\_amt=$\_POST['total\_amt'];

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

// $user\_type=$\_POST['user\_type'];

$fname=$\_POST['f\_name'];

$lname=$\_POST['l\_name'];

$addrs1=$\_POST['addrs1'];

$addrs2=$\_POST['addrs2'];

$contact=$\_POST['contact'];

// $city=$\_POST['city'];

$pincode=$\_POST['pincode'];

$state=$\_POST['state'];

$status='order received';

$stmt=$admin->Rcud("INSERT INTO `orders` (`total\_amt`, `c\_id`, `f\_name`, `l\_name`, `addrs1`, `addrs2`, `contact`, `pincode`, `state`, `date`, `status`) VALUES ('$total\_amt','$user\_id','$fname','$lname','$addrs1','$addrs2','$contact','$pincode','$state',now(),'$status')","saved");

$pay=$\_POST['payment\_method'];

// if($pay=='cod'){

// $trans=$\_POST['transaction'];

// $dots2='-----';

// $st=$admin->cud("INSERT INTO `payment` (`o\_id`,`user\_type`,`user\_id`,`p\_method`,`trans\_id`,`card\_no`,`expiry`,`p\_amt`,`p\_date`) VALUES('$stmt','$user\_type','$user\_id','$pay','$dots2','$dots2','$dots2','$total\_amt',now())","saved");

// }

if($pay=='upi'){

$trans=$\_POST['transaction'];

$dots='-----';

$st=$admin->cud("INSERT INTO `payment` (`o\_id`,`c\_id`,`p\_method`,`trans\_id`,`card\_no`,`expiry`,`p\_amt`,`p\_date`) VALUES('$stmt','$user\_id','$pay','$trans','$dots','$dots','$total\_amt',now())","saved");

}

if($pay=='card'){

$cno1=$\_POST['card\_no'];

$expiry1=$\_POST['expiry'];

$dots1='-----';

$stm=$admin->cud("INSERT INTO `payment` (`o\_id`,`c\_id`,`p\_method`,`trans\_id`,`card\_no`,`expiry`,`p\_amt`,`p\_date`) VALUES('$stmt','$user\_id','$pay','$dots1','$cno1','$expiry1','$total\_amt',now())","saved");

}

$cart=$admin->ret("SELECT \* FROM `cart` WHERE `c\_id`='$user\_id' ");

while($row=$cart->fetch(PDO::FETCH\_ASSOC)){

$price=$row['price'];

$qty=$row['quantity'];

$prod=$row['p\_id'];

$od=$admin->Rcud("INSERT INTO `order\_details`( `o\_id`, `c\_id`,`p\_id`,`quantity`,`price`) VALUES ('$stmt','$user\_id','$prod','$qty','$price')","saved");

$q=$admin->ret("SELECT \* FROM `order\_details` WHERE `p\_id`='$prod' AND `c\_id`='$user\_id' AND `od\_id`='$od' ");

while($r=$q->fetch(PDO::FETCH\_ASSOC)){

$pd=$r['p\_id'];

$qt=$r['quantity'];

$p=$admin->ret("SELECT \* FROM `products` WHERE `p\_id`='$pd'");

$sp=$p->fetch(PDO::FETCH\_ASSOC);

$newqty=$sp['stock']-$qt;

$spid=$sp['p\_id'];

$update=$admin->cud("UPDATE `products` SET `stock`='$newqty' WHERE `p\_id`='$spid'","updated");

}

}

$s=$admin->cud("DELETE FROM `cart` WHERE `c\_id`='$user\_id' ","deleted");

echo "<script>alert('Your order has been placed....');window.location='../orderstatus.php';</script>";

}

?>

**j) Update cart**

<?php

include '../config.php';

$admin=new admin();

$cid=$\_SESSION['c\_id'];

$cartid=$\_GET['cartid'];

$qty=$\_GET['qty'];

$stmt=$admin->cud("UPDATE `cart` SET `quantity`='$qty' WHERE `cart\_id`= '$cartid'","saved");

// echo "";

$st=$admin->ret("SELECT \* FROM `cart` INNER JOIN `products` ON cart.p\_id=products.p\_id WHERE `c\_id`='$cid'");

$num=$st->rowCount();

?>

<div id="tablecart">

<?php

$gtotal=0;

$st=$admin->ret("SELECT \* FROM `cart` INNER JOIN `products` ON cart.p\_id=products.p\_id WHERE `c\_id`='$cid'");

while($row=$st->fetch(PDO::FETCH\_ASSOC)){

?>

<!-- cross mark = &#10005;-->

<div class="row">

<div class="row main align-items-center">

<div class="col-2"><img class="img-fluid" src="shopowners/controller/<?php echo $row['p\_image'];?>" width="250" <?php echo $row['p\_image'] ?>></div>

<div class="col">

<div class="row text-muted"><?php echo $row['p\_name'] ?></div>

<!-- <div class="row"><?php echo $row['p\_description'] ?></div> -->

</div>

<div class="col"><style type="text/css"></style>

<button onclick="decrement(<?php echo $row['cart\_id']?>)" >-</button>

<input type="text" id="<?php echo $row['cart\_id']?>" value="<?php echo $row['quantity']?>" name="quantity" readonly>

<button onclick="increment(<?php echo $row['stock']?>,<?php echo $row['cart\_id']?>)">+</button>

</div>

<div class="col">₹ <?php echo $row['price'] ?> </div>

<div class="col">Total ₹<?php echo $total=$row['quantity']\*$row['price']?></div>

<div class="col"> <span class="close"><a href="controller/deletecart.php?id=<?php echo $row['cart\_id']?>"onclick="return confirm('Do you want to delete?')">Remove</a></span></div>

</div>

</div>

<?php

$gtotal=$gtotal+$total;

} ?>

<div class="back-to-shop"><a href="products.php">&leftarrow;<span class="text-muted">Back to shop</span></a></div>

<div class="col-md-12 summary">

<div><h5><b>Summary</b></h5></div>

<hr>

<div class="row">

<div class="col" style="padding-left:0;">ITEMS </div>

<div class="col text-right"> <?php echo $num ?></div>

</div>

<div class="row" style="border-top: 1px solid rgba(0,0,0,.1); padding: 2vh 0;">

<div class="col">TOTAL</div>

<div class="col text-right">₹<?php echo $gtotal?></div>

</div>

<a class="btn" href="checkout.php" style="color:white">CHECKOUT</a>

</div>

</div>

**k) Update profile**

<?php

include '../config.php';

$admin=new Admin();

$id=$\_SESSION['c\_id'];

$name=$\_POST['fullname'];

$email=$\_POST['email'];

$address=$\_POST['address'];

$contact=$\_POST['contact'];

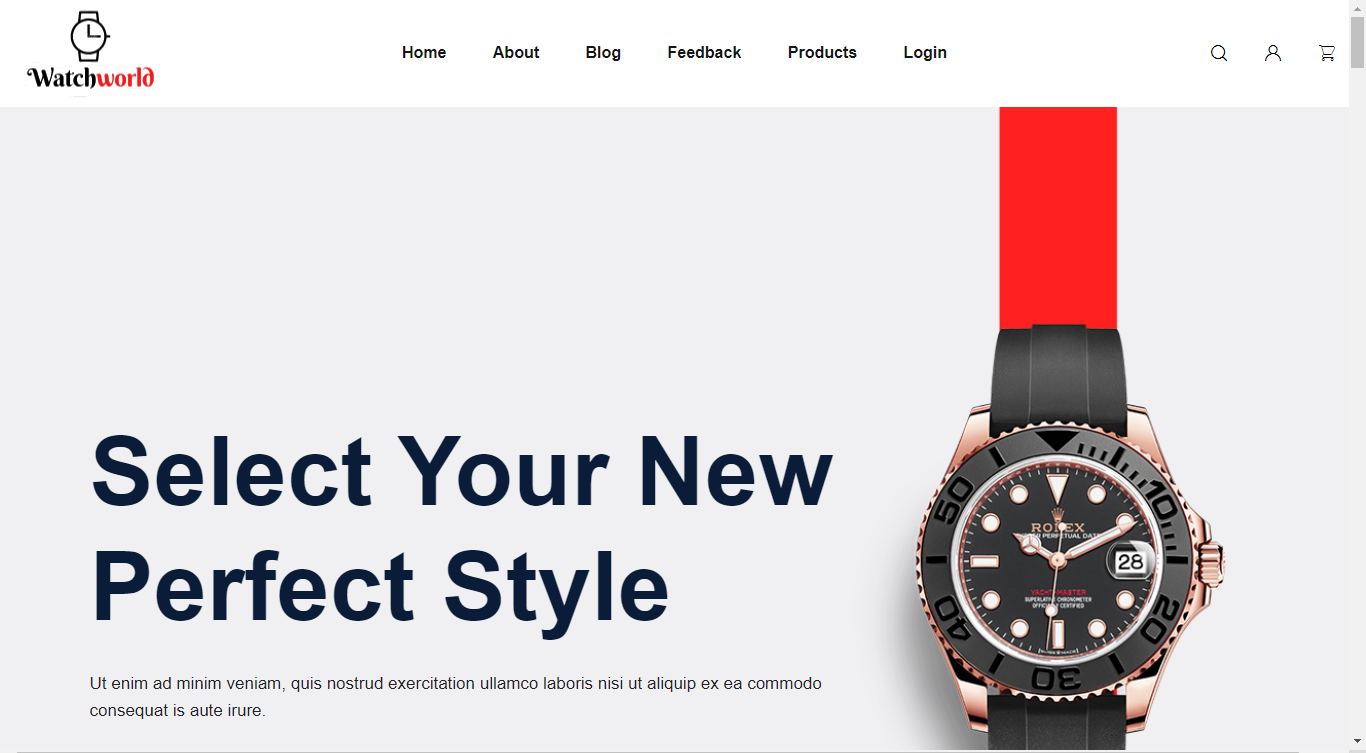
$stmt=$admin->cud("UPDATE `customer` SET `c\_name`='$name',`email`='$email',`address`='$address',`contact`='$contact' WHERE `c\_id`='$id'","updated");

echo "<script>alert(' Profile updated');window.location='../profile.php';</script>";

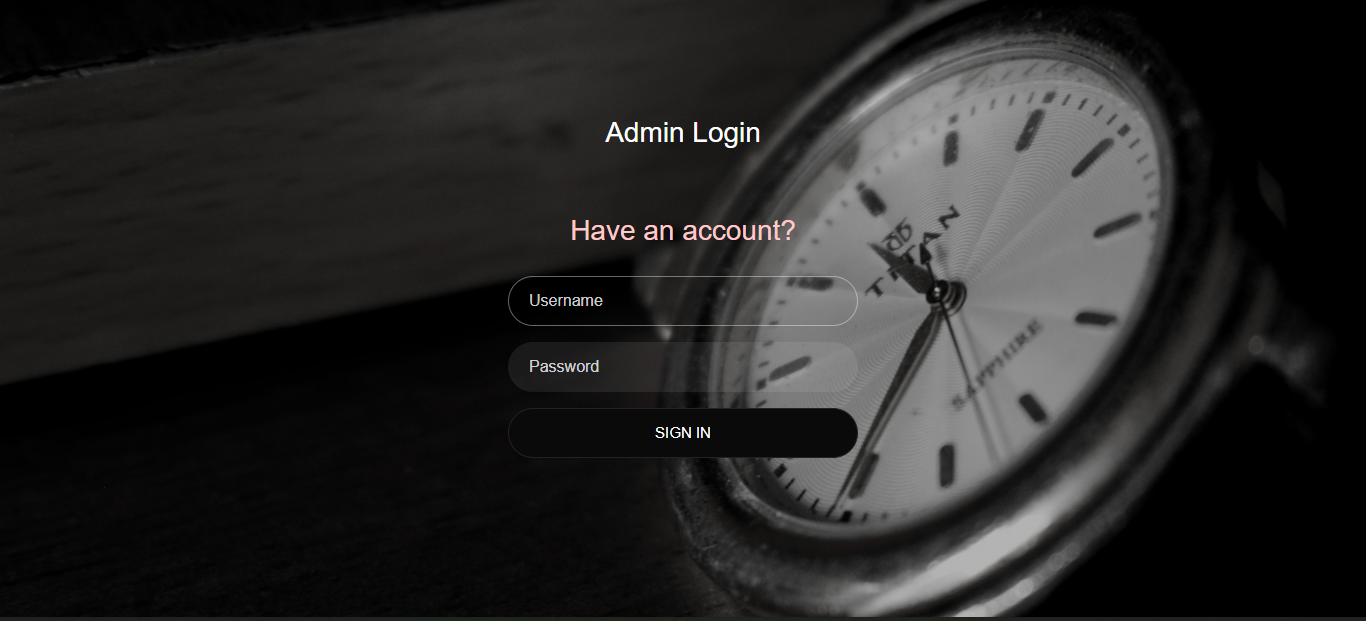
?>

**User interface**

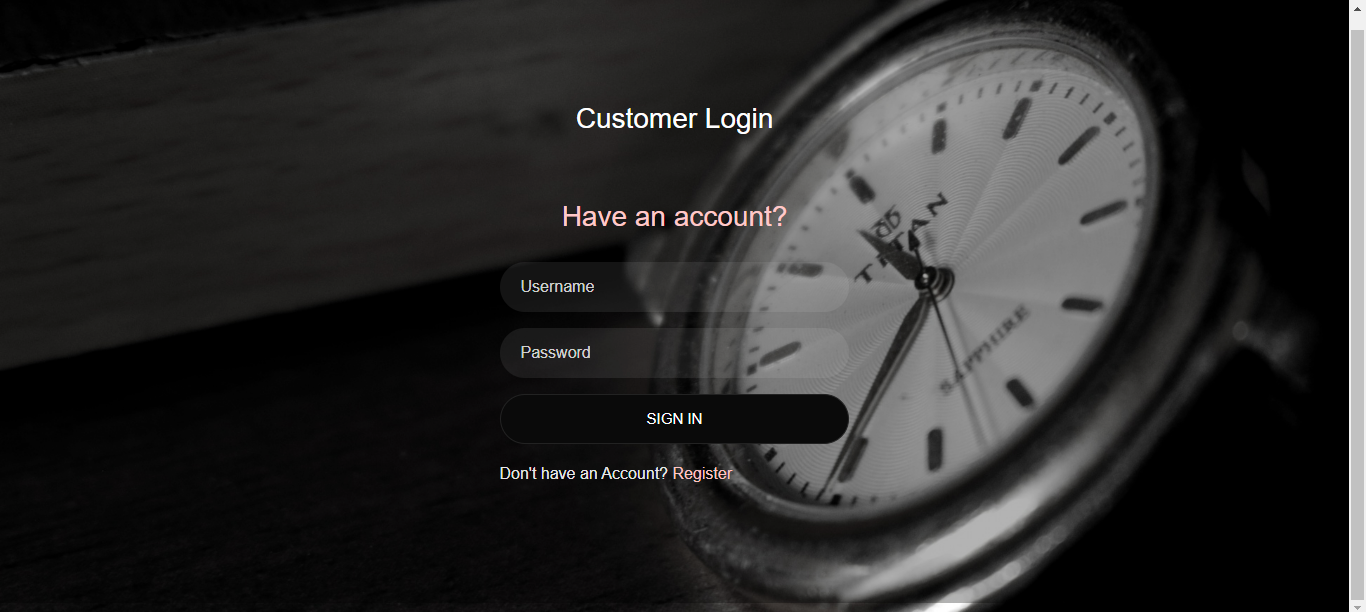
**Home page**

****

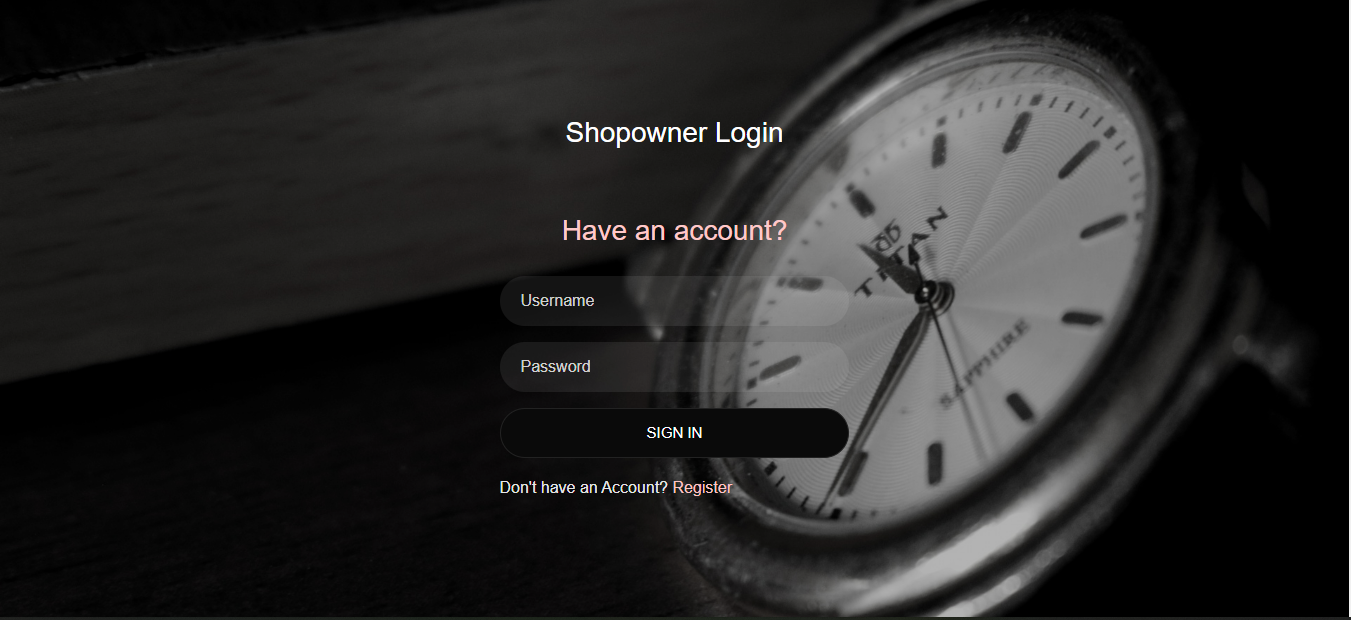
**ADMIN LOGIN**



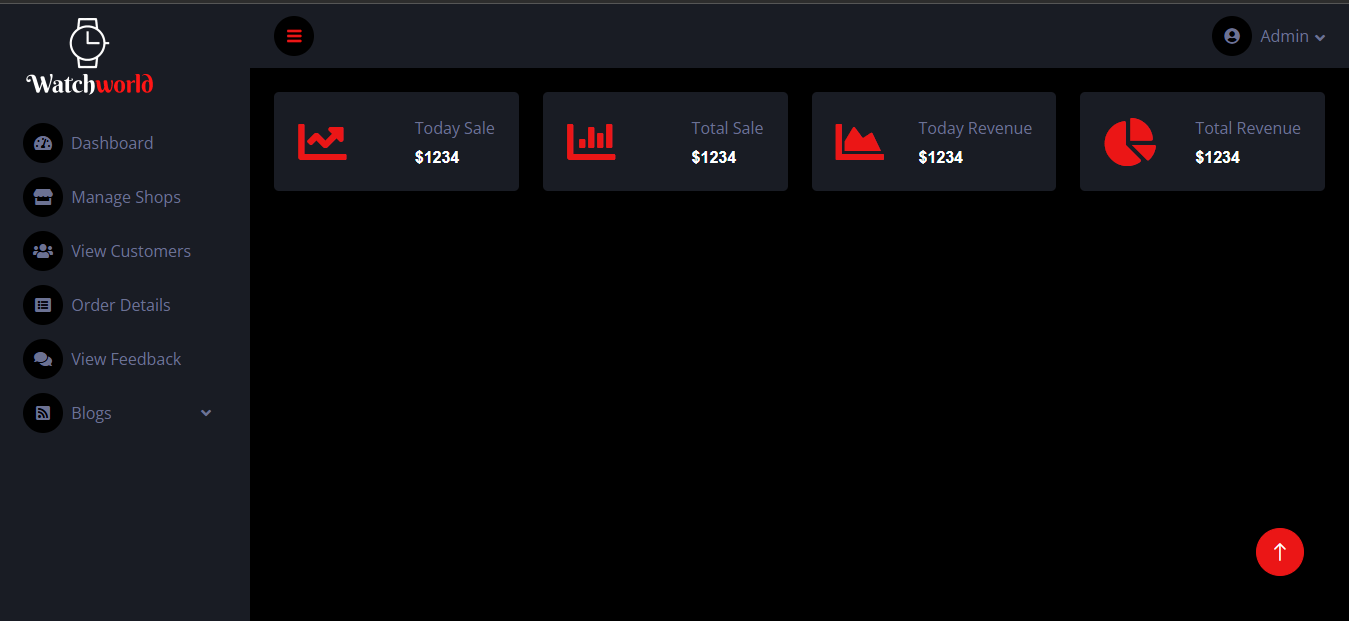
**CUSTOMER LOGIN**

****

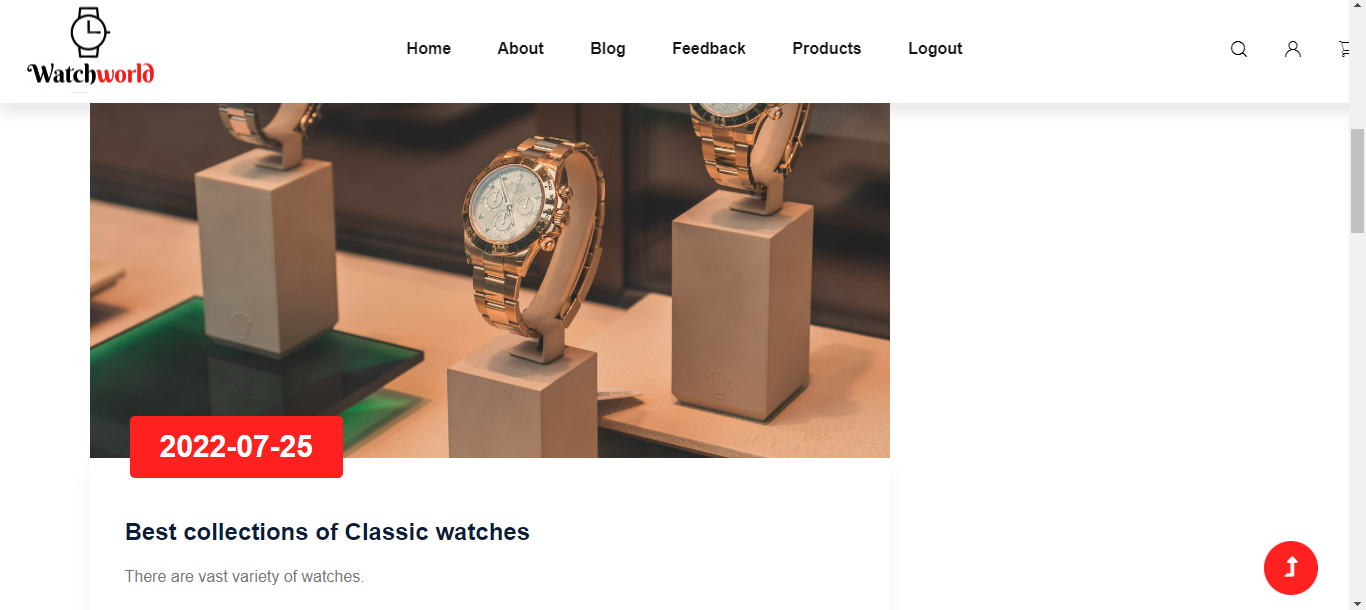
**SHOPOWNER LOGIN**

****

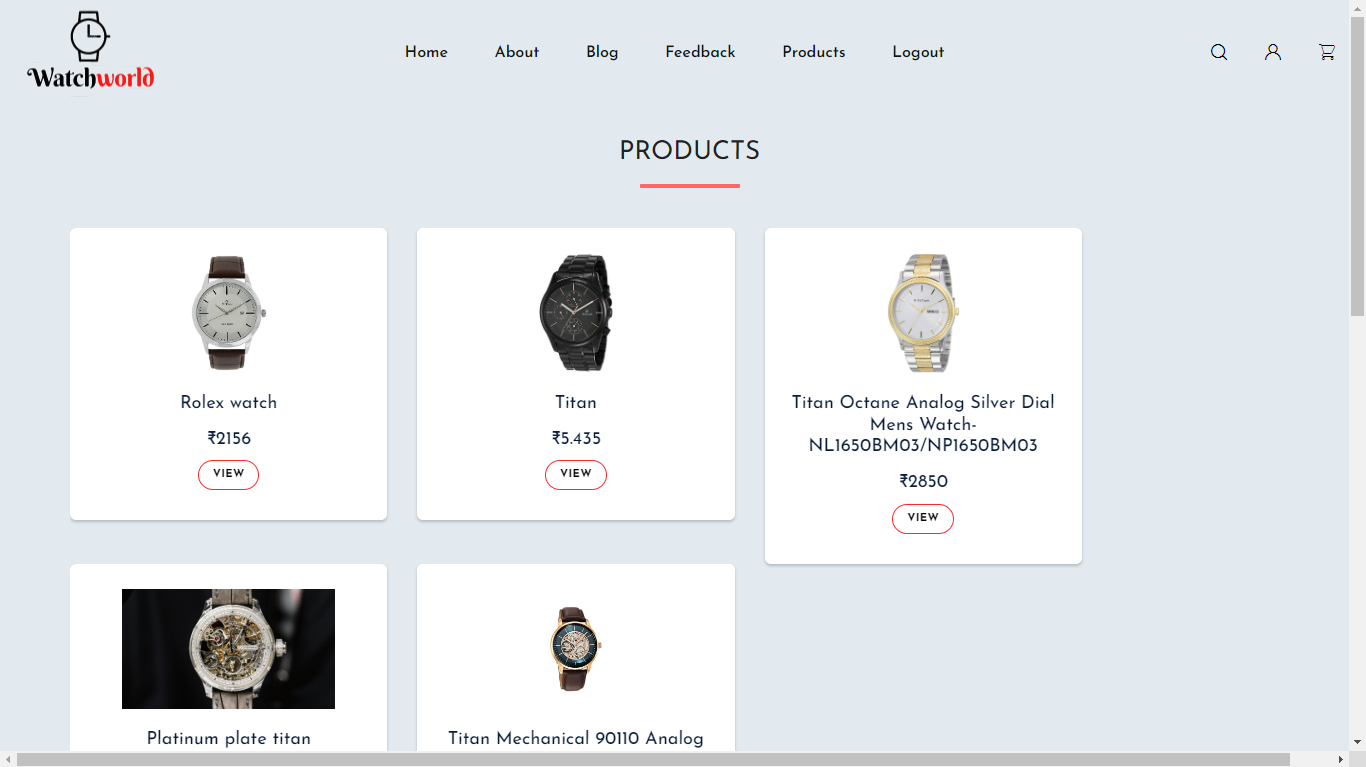
**ADMIN HOMEPAGE**

****

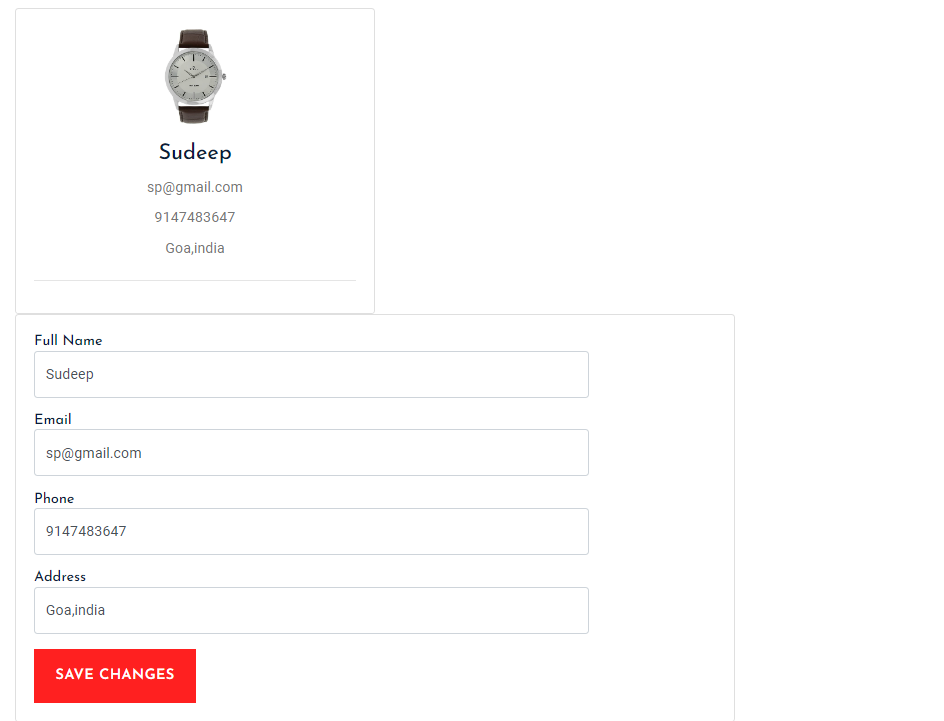
**BlOG PAGE**

****

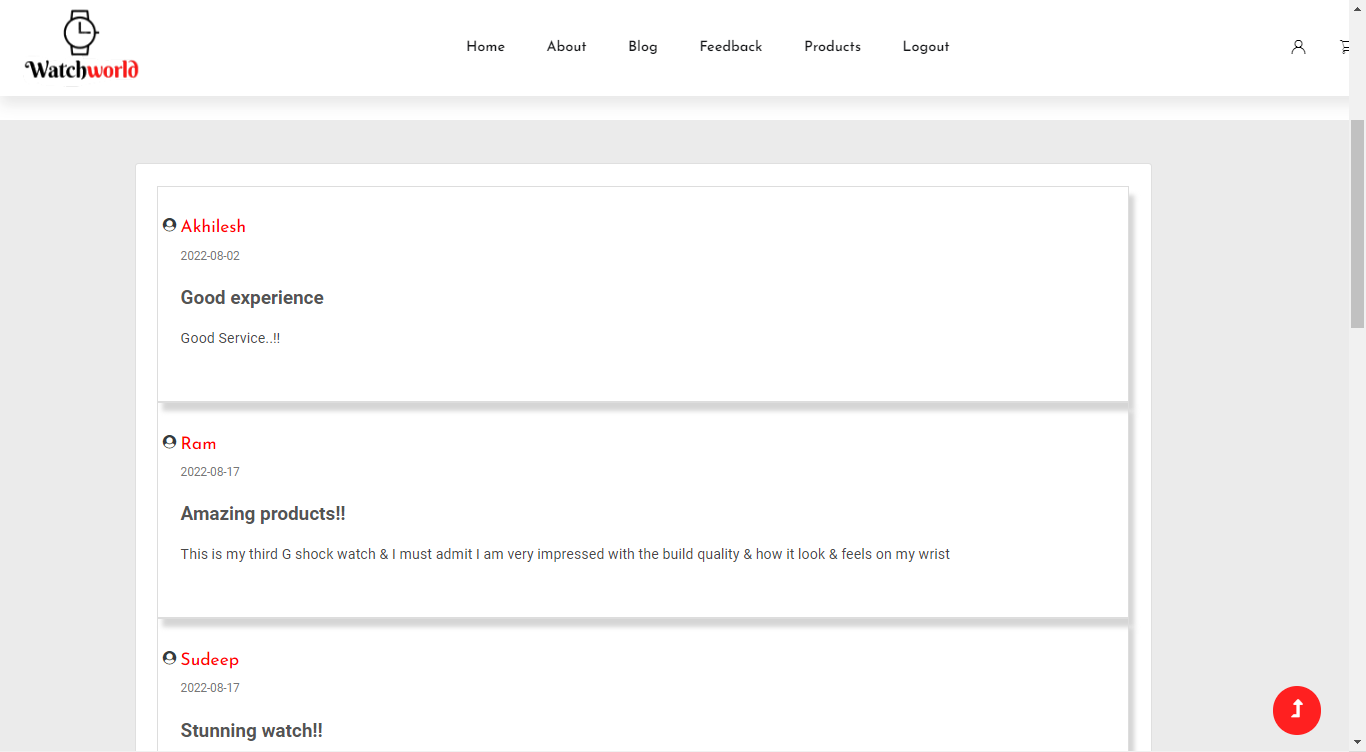
**PRODUCTS PAGE**

****

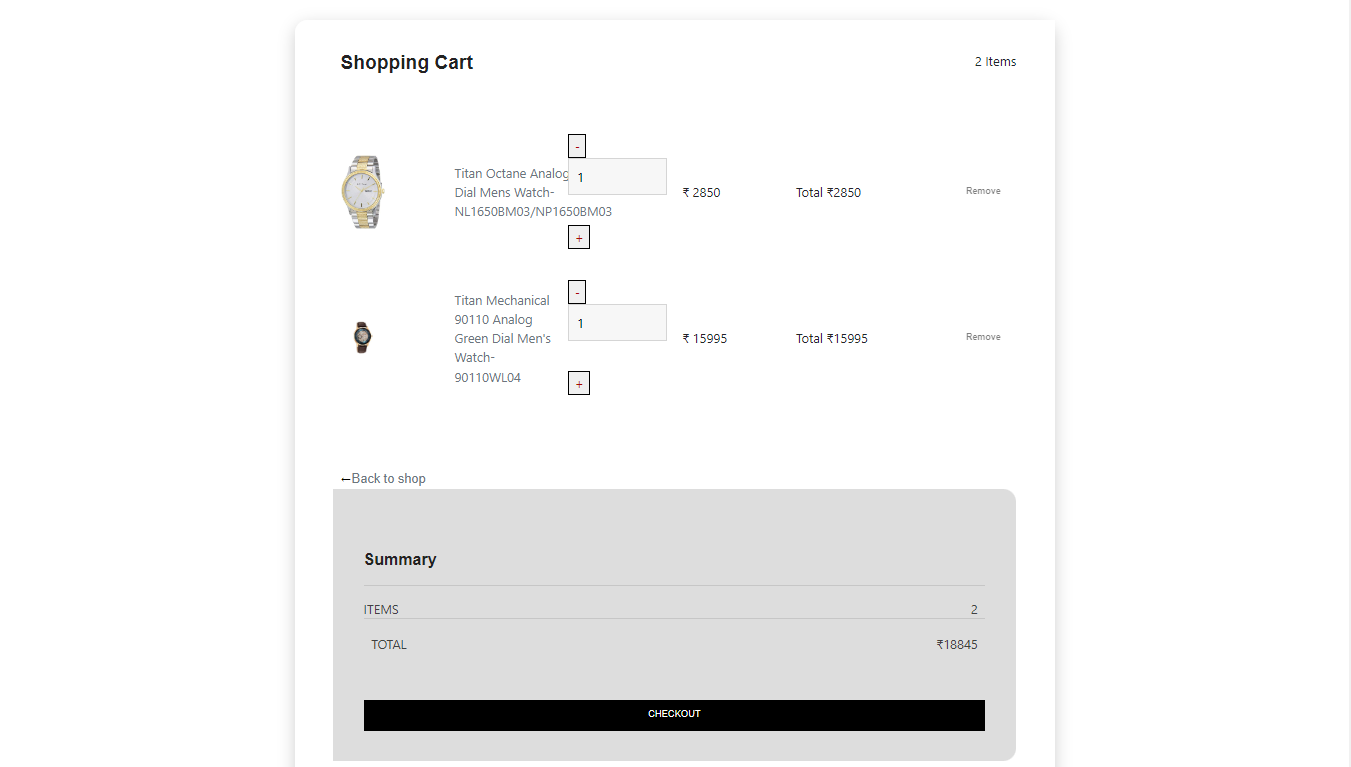
**CUSTOMER PROFILE PAGE**

****

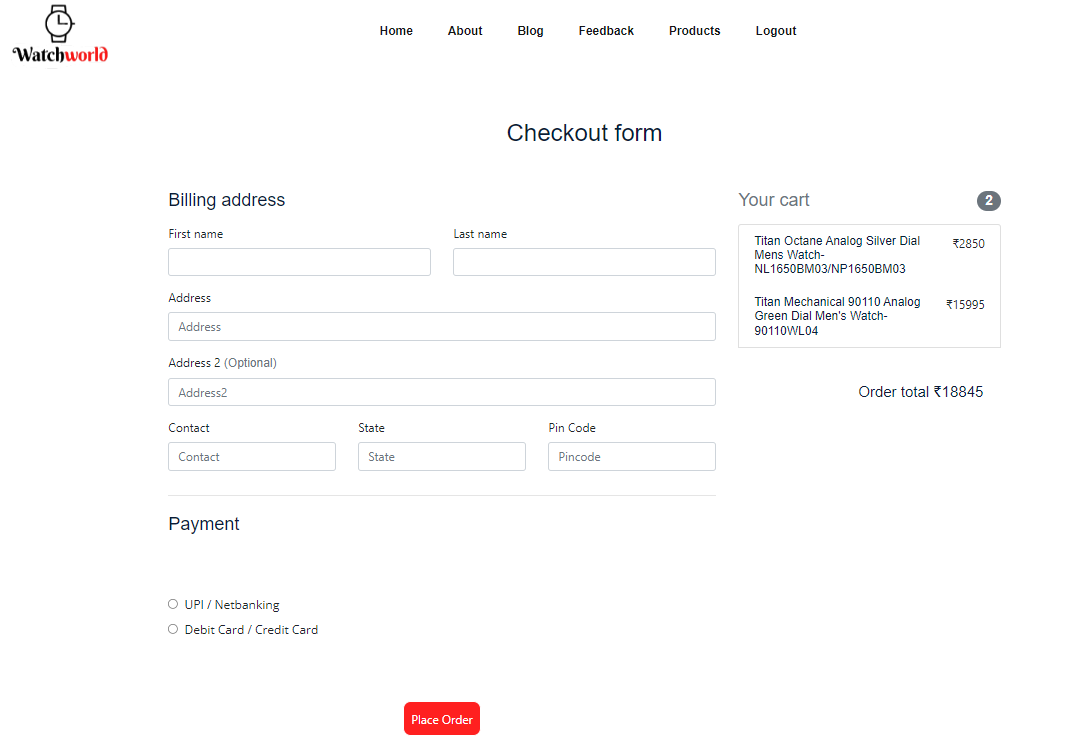
**FEEDBACKS**

****

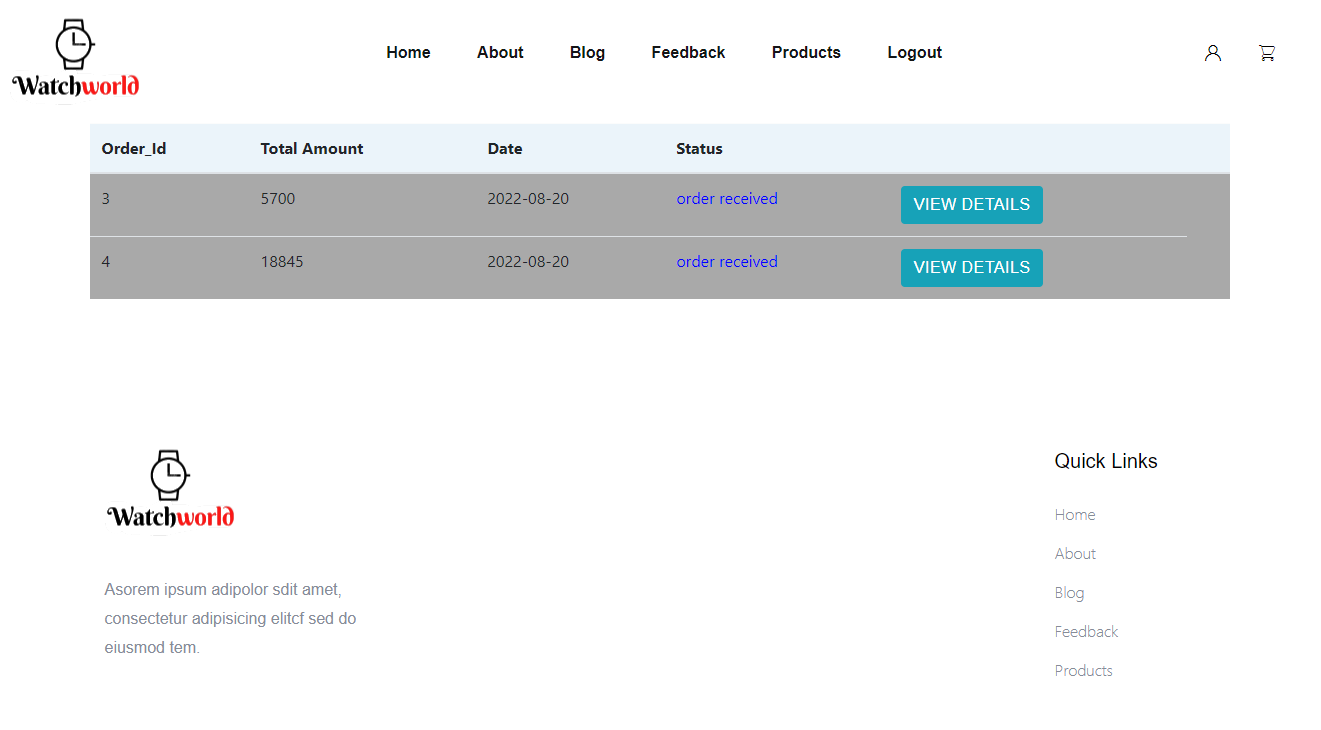
**SHOPPING CART**

****

**CHECKOUT FORM**

****

**ORDER STATUS**

****

## TESTING

## 8.1 Introduction

Software testing is an investigation conducted to provide stack holders with information about the quality of the product or service under test. Testing has been defined as the process of analysing a software item to detect the differences between existing and required conditions and to evaluate the features of the software item. Software testing is the process used to assess the quality of computer software.

It involves operation of a system or application under controlled conditions and evaluating the results. The controlled conditions should include both normal and abnormal conditions. Testing should intentionally attempt to make things go wrong to determine if things happen when they should. It is oriented to ‘detection’.

Software testing has three main purposes:

The verification process confirms that the software meet its technical specifications. A “specification” is a description of a function in terms of a measurable output value given a specific input value under specific preconditions.

The validation process confirms that the software meets the business requirements.

A defect is a variance between the expected and actual result. The defect’s ultimate source may be traced to a fault introduced in the specification, design, or development phases. Not all the defects will necessarily result in failures.

There are two types of software testing:

Black box testing-internal system design is not considered in this type of testing. test are based on requirements and functionality.

White box testing-this testing is based on knowledge of thaw internal logic of an application’s code. Also known as glass box testing. Internal software and code working should be known for this type of testing. Tests are based on coverage of code statements, branches, paths and conditions.

A test case is a software testing document, which consists of event, action, input, output, expected result and actual result. Clinically defined a test case is an input and an expected result. This can be pragmatic as ‘for condition x your derived result is y’; where as other test cases described in more detail the input scenario and what results might be expected. It can occasionally be a series of steps but one with expected results or expected outcome. A test case should also contain a place for the actual result.

White box testing is applicable at the unit, integration and system levels of the software testing process.

## 8.2 Testing objectives

Finding defects which may get created by the programmer while developing the software

Gaining confidence in and providing information about the level of quality

To prevent defects

To make sure that the end results meets the business and user requirements.

To ensure that it satisfies the BRS that is Business Requirement Specification and SRS that is System Requirement Specification.

## 8.3 Testing methods

System testing is the stage of implementation. This is to check whether system works accurately and efficiently before live operation commences. Testing is vital to the success of the system. The candidate system is subject to a variety of tests: online response, volume, stress, recovery, security and usability tests. A series of tests are performed for the proposed system is ready for user acceptance testing.

## 8.4 Testing steps

### 8.4.1 Unit testing

Unit testing focuses efforts on the smallest unit of software design. This is known as module testing. The modules are tested separately. The test is carried out during programming stage itself. In this step, each module is found to be working satisfactory as regards to the expected output from the module.

### 8.4.2 Integration testing

Data can be lost across an interface. One module can have an adverse effect on another, sub functions, when combined, may not be linked in desired manner in major functions. Integration testing is a systematic approach for constructing the program structure, while at the same time conducting test to uncover errors associated within the interface.

The objective is to take unit tested modules and builds program structure. All the modules are combined and tested as a whole.

### 8.4.3 Validation

At the culmination of the integration testing, Software is completely assembled as a package. Interfacing errors have been uncovered and corrected and a final series of software test begin in validation testing. Validation testing can be defined in many ways, but a simple definition is that the validation succeeds when the software functions in a manner that is expected by the customer. After validation test has been conducted, one of the three possible conditions exists.

The function or performance characteristics confirm to specification and are accepted.

A deviation from specification is uncovered and a deficiency lists is created.

Proposed system under consideration has been tested by using validation test and found to be working satisfactory.

### 8.4.4 Output testing

After performing the validation testing, the next step is output testing of the proposed system, since no system could be useful if it does not produce the required output in a specific format. The output format on the screen is found to be correct. The format was designed in the system design time according to the user needs. For the hard copy also; the output comes as per the specified requirements by the user. Hence output testing did not result in any correction for the system.

### 8.4.5 User acceptance testing

User acceptance of a system is the key factor for the success of any system. The system under consideration is tested for the user acceptance by constantly keeping in touch with the prospective system users at the time of developing and making changes whenever required.

This is done in regard to the following point:

Input screen design.

Output screen design.

Online message should be guide to the user.

Format of reports and other outputs.

## 8.5 Test cases

### 8.5.1 Admin Login form

|  |  |  |  |
| --- | --- | --- | --- |
| **Sl. No** | **Test Condition** | **Expected Result** | **Result** |
| 1 | If admin clicks on login button without entering username and password. | Please fill out this field | Successful |
| 2 | If username is blank but password is entered. | Please fill out this field | Successful |
| 3 | If password is blank but username is entered. | Please fill out this field | Successful |
| 4 | If the username or password is incorrect. | Login failed | Successful |
| 5 | If the valid username and valid password is entered | System displays admin’s main home page. | Successful |

### 8.5.2 Admin Home page

|  |  |  |  |
| --- | --- | --- | --- |
| **Sl No** | **Test Condition** | **Expected Result** | **Result** |
| 1 | When the user clicks the “manage shops” in the menu item. | List of the registered shops appear. | Successful |
| 2 | When the user clicks the “view customers” in the menu item. | It displays a list of customers. | Successful |
| 3 | When the user clicks the “view feedback” in the menu item. | It displays all the feedbacks uploaded by the customers. | Successful |
| 4 | When the user clicks the “Blogs” in the menu item. | It displays a drop down button for view and upload blogs. | Successful |
| 5 | When the user clicks the “view blog” in the menu item. | It displays all the uploaded blogs. | Successful |
| 6. | When the user clicks the “upload” the menu | It displays a form to upload a new blog. | Successful |
| 7. | When the user clicks the “order details” the menu | It displays all the orders. | Successful |
| 8. | When the user clicks the “view feedback” the menu | It displays all of the feedbacks. | Successful |

### 8.5.3 Adding customer

|  |  |  |  |
| --- | --- | --- | --- |
| SL NO | TEST CONDITION | EXPECTED OUTPUT | TEST RESULT |
| 1. | If customer name is empty | Please fill out this filed | Successful |
| 2. | If customer name is not in alphabets | Please match the requested format | Successful |
| 3. | If customer’s contact number is empty | Alert the user student contact number should not be empty | Successful |
| 4. | If contact number is not in number | Please match the requested format | Successful |
| 5. | If email id is empty | Please fill out this field | Successful |
| 6. | If email id is only alphabet | Please match the requested format | Successful |
| 7. | If password is empty | Please fill out this field | Successful |

### 8.5.4 Adding shopowner

|  |  |  |  |
| --- | --- | --- | --- |
| SL NO | TEST CONDITION | EXPECTED OUTPUT | TEST RESULT |
| 1. | If shopowner name is empty | Please fill out this field | Successful |
| 2. | If shopowner name is not in alphabets | Please match the requested format | Successful |
| 3. | If shopowner contact number is empty | Please fill out this field | Successful |
| 4. | If contact number is not number | Please match the requested format | Successful |
| 5. | If address is empty | Please fill out this field | Successful |
| 6. | If password is empty | Please fill out this field | Successful |
| 8. | If file is not selected | Please select the file | Successful |

**System Testing:**

System Testing is the testing of a complete and fully integrated software product. Usually, software is only one element of a larger computer-based system. Ultimately, software is interfaced with other software/hardware systems. System Testing is actually a series of different tests whose sole purpose is to exercise the full computer-based system.

**System testing tables:**

|  |  |  |
| --- | --- | --- |
| Sl No | Test condition | Test report |
| 1. | System loading | Successful |
| 2. | System run procedure | Successful |
| 3. | File I/O operation | Successful |
| 4. | Database communication | Successful |
| 5. | Server/client interaction | Successful |
| 6. | Memory usage | Normal |
| 7. | System processor usage | Normal |
| 8. | Authentication/Authorization | Successful |

**Conclusion**

This web application makes shopping experience easy and convenient, both for the customers as well as for the shopowners.

**Limitation**

* Human error: error can be easily done by humans.
* Data management issue:since there is lot of data,it becomes difficult to manage as it increases.
* Scurity: there is no extra security like otp etc.
* Shopowner need to wait until admin accepts her as a shopowner.

**Scope for enhancement**

* We will be trying to make this web application available for the android users as well over the coming years.
* We will be adding the renting service soon
* We will be trying to add as many as possible shops to this application so that there will be a vast variety of watches available for the users to choose from.
* More services will be added over the passing time.

**Abbreviations and Acronyms**

**Definition, Acronyms and Abbreviations:**

1. PHP Hypertext Pre-processor.
2. CSS Cascading Style Sheet.
3. HTML Hyper Text Mark-up Language.
4. OS Operating System.
5. SRS Software Requirement Specifications.
6. DFD Data Flow Diagrams.
7. CFD Context Flow Diagrams.
8. ERD Entity Relationship Diagram.
9. MySQL MY Structural Query Language.
10. GUI Graphical User Interface.
11. Web-based application An application that runs on the internet.

**Bibiliography**

**Websites:**

* [www.w3schools.com](https://www.w3schools.com/)
* [www.geeksforgeeks](https://www.geeksforgeeks.org/)
* [www.tutorialspoint.com](https://www.tutorialspoint.com/index.htm)