ACKNOWLEDGEMENT

It gives me great pleasure in submitting this project entitled "**MyCart**" developed at Universal Infotech as a part of the curriculum of Bsc Computer Science ( Semester 6 ) .

I avail this opportunity to express our heartfelt gratitude to a number of people who extend their full support and co - operation in developing this project and also imparting Knowledge to us in various other domain of software technology.

We would like to take opportunity to thank our collage , **Government Science College,Valod**. For giving us this tremendous opportunity to work in the real - time project.

Ireally thank my external project guide \_\_\_\_\_\_\_ who was always there to guide me through the development of the project . He is one of the major sources behind the success of the project . We immensely appreciate the tip he has constantly given us during the project . It was an enormous pleasure to work with him.

I am thankful to the faculty of the institute for their constant guidance not only during training period but also throughout college career.

Finallyi would like to thank My Parents for their support throughout the project. I owe a special debt to my family & friends for their support blessing and encouragement for me.

ABSTRACT

**An online shopping system is a process in which people (specifical customers) are being provided with the option of purchasing goods and services directly from the seller, all in a real-time environment**

**The Online Shopping is a Mobile application based application intended for online retailers. The mainobjective of this application is to make it interactive and its ease of use. It would makesearching, viewing and selection of a product easier.**

INDEX

INDEX

|  |  |  |
| --- | --- | --- |
| SR. NO | CONTENTS | PAGE NO |
| 1 | INTRODUCTION   * 1. COMPNY PROFILE   1.2 PROJECT INTRODUCTION  1.3 OBJECTIVE AND SCOPE |  |
| 2 | ENVIRONMENT DESIGN  2.1 DEVELOPMENT DESCRIPTION |  |
| 3 | METHODOLOGY |  |
| 4 | DESIGN AND ARCHITECTURES  4.1 Process Flow Diagram  4.2 UML  4.3 Data Flow Diagrams  4.4Flow Chart  4.5UML Diagram  4.5 Activity Diagram  4.6 Data Design  4.7 ER Diagram |  |
| 5 | Design Layouts |  |
| 6 | Testing |  |

1.1COMPANY PROFILE

Company Name : **Universal InfoTech**

Project Guide : Anirudh Bhatt

Address : 1stFloor ,Triveni Apartment

Near HDFC Bank,Station Road

Bardoli,DI-Surat,Guj(India)

Contact No : +91 997 871 0101

Email : [info@universalinfotech.in](mailto:info@universalinfotech.in)

Website : [www.universalinfotech.in](http://www.universalinfotech.in)

1.2 PROJECT PROFILE

Project Title :**Mycart**

Duration : 3 Months

Tools :Android Studio

Database Server :Firebase

Operating system :Windows 10

1.3 OBJECTIVE & SCOPE

**Objective :**

* To shop wile in the comfort of your own home ,without having to step out

of the door.

* Sell at lower rate due to less over head.
* Provide home delivery free of cost.
* Secured Transaction.

**Scope:**

* The current system can be extended to allow the users to create accounts and save products in to wish list
* This product has great future scope. Online shopping Internet software developed on and Mobile Application.
* This project alsoprovides security with the use of Login-id and Password, so that any unauthorizedusers can not use your account.
* The only Authorized that will have proper accessauthority can access the software

**Admin Module (Site-Admin Interface):**

Administrator module does contain following modules.

1. Login and Logout
2. Event Management
3. Service Management
4. Booking Management
5. User Management
6. Generate Report

* Admin login into the admin panel.
* Admin refer the client detail and can also block the client.
* Admin add/delete/update the Event .
* Admin add/delete /update the Services .
* Admin add/update users .

**User(Staff)-Administrator (User-Admin Interface):**

* User can login, see their profile, and also manage it.
* User can manage add events ,Service, Book events.
* User can get reports.

2 DEVELOPMENT DESCRIPTION

**FLUTTER**

Flutter is an open source framework to create high quality, high performance mobile applications across mobile operating systems - Android and iOS. It provides a simple, powerful, efficient and easy to understand SDK to write mobile application in Google’s own language, Dart

In this scenario, Flutter – a simple and high performance framework based on Dart language, provides high performance by rendering the UI directly in the operating system’s canvas rather than through native framework.

Flutter also offers many ready to use widgets (UI) to create a modern application. These widgets are optimized for mobile environment and designing the application using widgets is as simple as designing HTML.

**FIREBASE**

Firebase is a backend platform for building Web, Android and IOS applications. It offers real time database, different APIs, multiple authentication types and hosting platform.

Firebase can power your app's backend, including data storage, user authentication, static hosting, and more Focus on creating extraordinary user experiences. We will take care of the rest. Build cross-platform native mobile and web apps with our Android, iOS, and JavaScript SDKs. You can also connect Firebase to your existing backend using our server-side libraries or our REST API.

3 METHODOLOGY

**3.1.Justification for the Methodology:**

This model can be used when the requirements of the complete system are clearly defined and understood, like the case of this project where,

* Major requirements were evidently defined; however, some details evolved with time
* There was a need to complete the project within a short time schedule
* A new technology is being used or the resources with needed skill set are not available. I was learning Flutter and Firebase and could iterate from one technology to another to ensure I effective implement all the functionalities.
* The project had some high-risk features and goals.

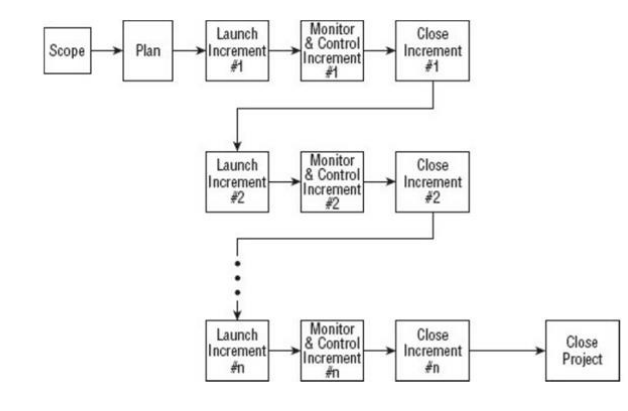


Figure : Incremental Project Management Life Cycle

The Incremental model is much better equipped to handle change. Each incremental functionality is verified by the customer and hence the relative risk in managing large and complex projects is substantially reduced. On the downside, there is a possibility of gold plating, wherein the functionalities not really required end up being built into the Product or Deliverable. In a nutshell, Incremental SDLC provide plethora of advantages inducing,

* Generates working software quickly and early during the software life cycle
* This model is more flexible and less costly to change scope and requirements
* It is easier to test and debug during a smaller iteration
* In this model customer can respond to each built.
* Lowers initial delivery cost.
* Easier to manage risk because risky pieces are identified and handled during it’diteration

**3.2. System Analysis:**

Analysis is an important part of any project; is analysis is not done properly then whole project move in the wrong direction. It also provides a schedule for proper project work. Analysis task divided into 3 areas:

* Problem Recognition.
* Feasibility Study.
* Requirement Analysis.

**3.3. Feasibility Study:**

Feasibility study of the system is a very important stage during system design. Feasibility study is a test of a system proposal according to its workability impact on the organization, ability to meet user needs, and effective use of resources. Feasibility study decides whether the system is properly developed or not. There are five types of feasibility as mentioned below:

1. Technical Feasibility
2. Time Schedule feasibility
3. Operational feasibility
4. Implementation feasibility
5. Economic Feasibility

**3.3.1Technical Feasibility**

Technical feasibility corresponds to determination of whether it is technically feasible to develop the software. Here those tools are considered, which will be required for developing the project. The tools, which are available, and tools, which will be required, are taken into account. Considering all above points and aspects it is observed that the cost incurred in developing this project from a technical perspective would not be too high. Thus, it is feasible for company as well as for me to develop this system.

**3.3.2 Time Feasibility**

Time feasibility corresponds to whether sufficient time is available to complete the project.

Parameters considered:

* Schedule of the project.
* Time by which the project has to be completed.
* Reporting period

Considering all the above factors it was decided that the allotted time that is 3 months was sufficient to complete the project.

**3.3.3 Operational Feasibility**

Operational feasibility corresponds to whether users are aware of interface environment and sufficient resources are available or not.

Parameters considered:

* ▪People with a basic knowledge of computers would be able to use our system very effectively and easily, as the system would have an intuitive GUI. The director andn employees of La BELLE Fashions have a basic operating knowledge of computers, so understanding the working of the system and using it would be easy from the decisionmaker’s point of view.
* ▪All the relevant necessary resources for implementing and operating this system are already present in office

Bearing in mind the above factor, it was observed that the cost would be incurred in developing this project from an operational standpoint would be low. Thus, it would be operational feasible for the company.

**3.3.4 Implementation Feasibility**

Implementation Feasibility is about basic infrastructure required to develop the system. Considering all below points, it is feasible to develop system

Factors considered:

* All the minimum infrastructure facility required like PC, books, technical manuals are provided.
* ▪ Proper guidance is provided.
* All necessary data and files are provided.

**3.3.4 Implementation Feasibility**

Economic Feasibility is about total cost incurred for the system. The software resource requirement of the proposed system is Firebase for functional and backend development Flutter for the frontend UI.

**3.4.** **Requirements Analysis and Specification:**

A complete understanding of software requirement is essential to the success of a web- development effort. No matter how well designed or well coded, a poorly analysed and specific program will disappoint user and bring grief to the developers.

The requirement analysis task is process of discovery, refinement, modified and specification. The software scope, initially established by the system engineer and refined during project planning, is refined in detail. Models of the required data, information and control flow, and operational behaviour are created. Alternative solutions are analysed and various project element.

Currently who want to buy some shoes or any clothing type they have to go to the shop and buy them this is very tedious for customer therefore we upload this site on internet. This web-site should be developed with an aim to simplify shopping process and keeping transparency and flexibility in performing each operation.

**3.4.1Requirements Gathering**

Also known as data collection. Data Collection is an important aspect of any type of research study. Inaccurate data collection can impact the results of a study and ultimately lead to invalid results. The methods used to gather the projects requirements involves Quantitative research to review the existing systems in the market.

**3.4.2 Data Collection Methods**

This study used quantitative techniques like online survey and questionnaire. Qualitative data collection methods play an important role in impact evaluation by providing information useful to understand the processes behind observed results and assess changes in people’s perceptions of their well-being. Furthermore, qualitative methods can be used to improve the quality of surveybased quantitative evaluations by helping generate evaluation hypothesis; strengthening the design of survey questionnaires and expanding or clarifying quantitative evaluation findings. These methods are characterized by the following attributes:

* They tend to be open-ended and have less structured protocols
* They rely more heavily on interactive interviews; respondents may be interviewed several times to follow up on a particular issue, clarify concepts or check the reliability of data
* They use triangulation to increase the credibility of their findings
* Generally, their findings are not generalizable to any specific population, rather each case study produces a single piece of evidence that can be used to seek general patterns among different studies of the same issue

Existing written and visual materials were assessed to find important data and information towards the development of the system. Information about appointment managements, patient’s management were collected. During data collection, the investigation found out how the current system operates, not only that but also tried out which problems are faced and how best they can be settled.

Requirement analysis and specification may appear to be relatively simple task, but appearances are deceiving. Communication content is very high, chances for misinterpretations or misinformation abound. Ambiguity is probable. The dilemma that confronts a software engineer may best be understood by repeating the statement of an anonymous customer: “I know you believe you’re understood what you think I said, but I am not sure you realize that what you heard is not what I meant”.

**3.4.3 Requirements**

The requirements form the proposed system was categorized into functional and non- functional requirements.

**Functional Requirements:**

The following is the desired functionality of the new system. The proposed project would cover:

**Customer Module**

• Customer can view/search products.

• Customer can also add/remove product to cart (if customer try to add same product in cart. It will add only one)

• When customer try to purchase product, then he/she must login to system.

• After creating account and login to system, he/she can place order.

• If customer click on pay button, then their payment will be successful and their order will be placed.

• Customer can check their ordered details by clicking on orders button.

• Customer can see the order status (Pending, Confirmed, Delivered) for each order

• Customer can Download their order invoice for each order

**Admin Module**

• Admin can provide username, email, password and your admin account will be created.

• After login, there is a dashboard where admin can see how many customers is registered, how many products are there for sale, how many orders placed.

• Admin can add/delete/view/edit the products.

• Admin can view/edit/delete customer details.

• Admin can view orders.

• Admin can change status of order (order is pending, confirmed, out for delivery, delivered)

**Non-Functional Requirements:**

It specifies the quality attribute of a software system. They judge the software system based on Responsiveness, Usability, Security, Portability and other nonfunctional standards that are critical to the success of the software system

• Availability: The system should remain operational in any day and any place.

• Accuracy: There is a need to optimize the system to ensure more accurate results and calculations.

• Usability: The system should provide a User-friendly user interface and tooltips to enhance itself and be effectively responsive.

• Secure: The system must be able to provide security against any external injections by using a layered security system. Implementation of user login functionalities also ensures the system is secure from unauthorized persons.

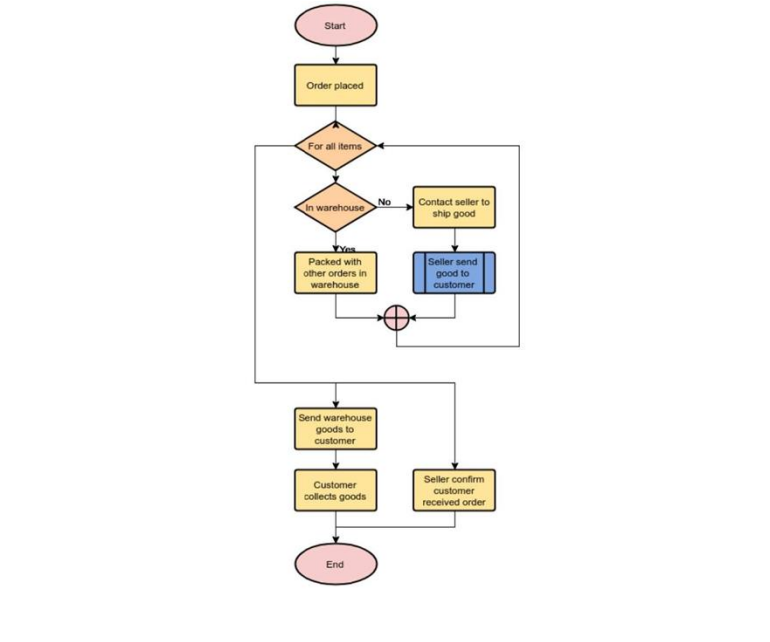
• Performance of the system: Response time is very good for given piece of work. The system will support multi user environment.

• Reliability of the system: The system will be highly reliable and it generates all the updates information in correct order. Data validation and verification is done at every stage of activity. System recovery will also be speed.

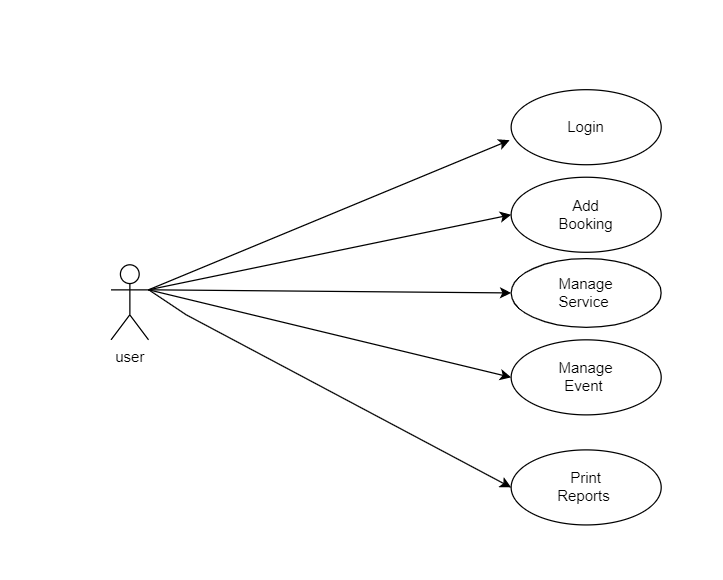
4 DESIGN AND ARCHITECTURES

Design is the first step in development phase for any techniques and principles for the purpose of defining a device , a process or system in sufficient detail to permit its physical realization. Once the software requirement have been analyzed and specified the software design involves three technical activities-Design, Coding, Implementation, Testing that are required to build and verify the software.The design activities are of main importance in this phase, because in this activities decisions ultimately affecting the success of the software implementation and its ease of maintenance are made. These decision has the final bearing upon reliability and maintainability of the a system. Design is only way to accurately transfer the customers requirements into finished software or system .Design is the place where quality is fostered in development. Software design is the process through which requirements are translated into a representation of software. Software requirement is conducted in two steps. Preliminary design is concerned with the transformation of requirements into data.

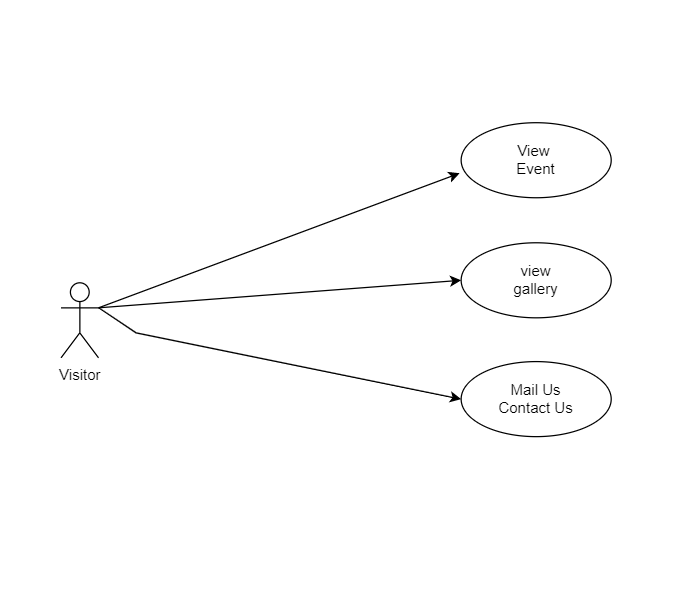
**Process Flow Diagram:**



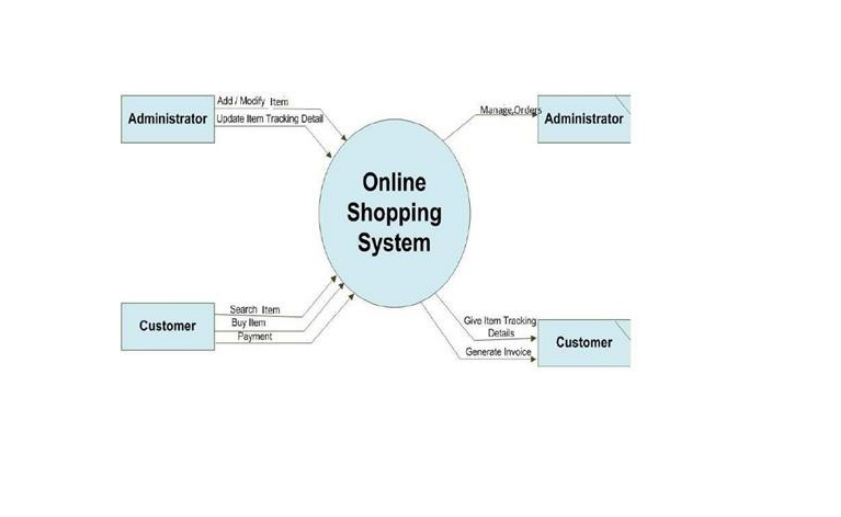
**Use Case For User:**

****

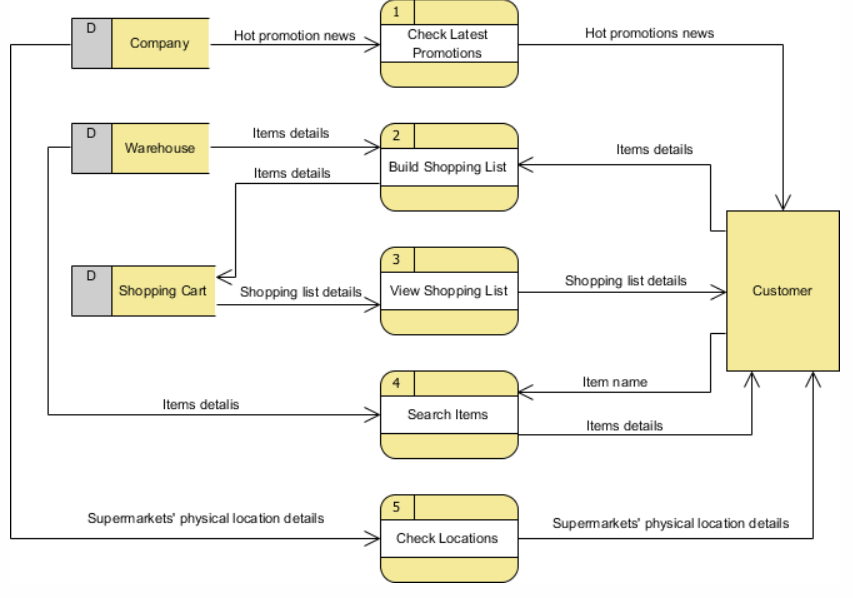
**Use Case For Visitor:**

****

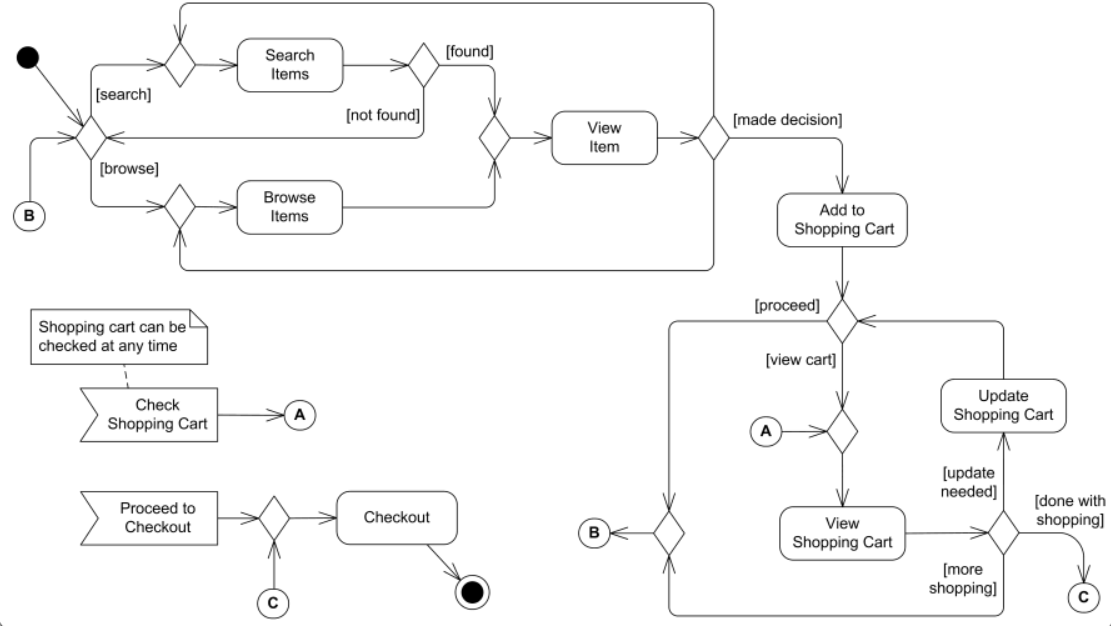
**Data Flow Diagram:**

****

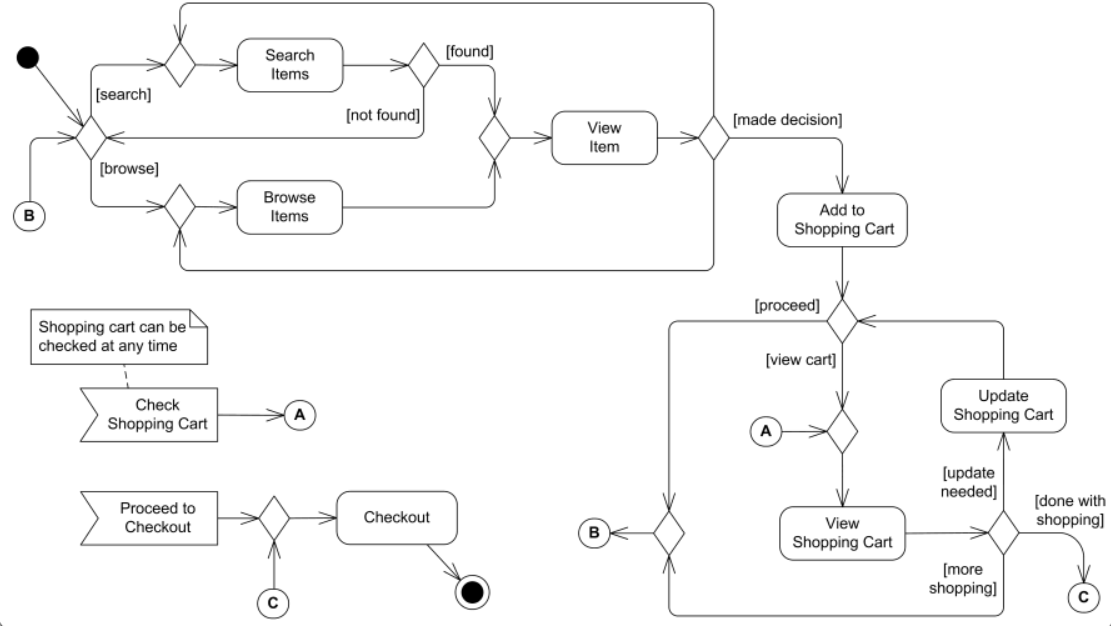
Flow Chart:

****

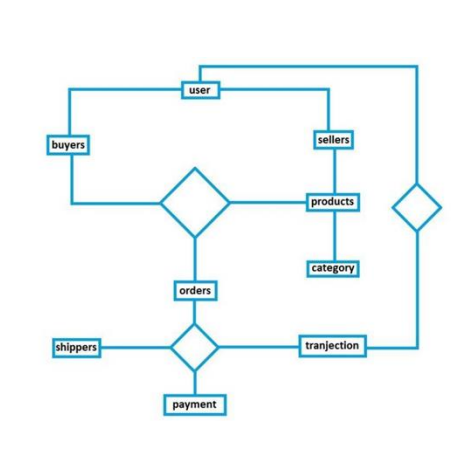
UML Diagram

******

Activity Diagram

******

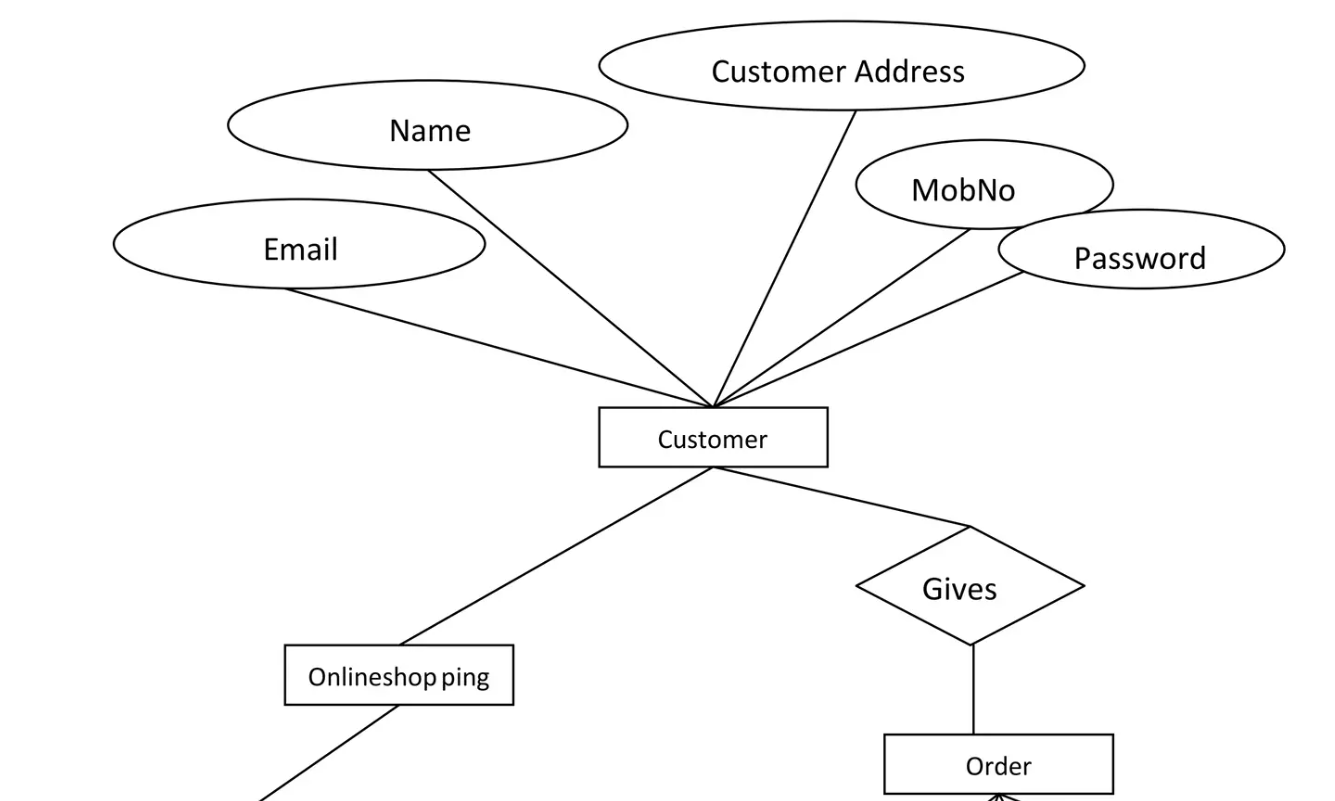
Data Design:

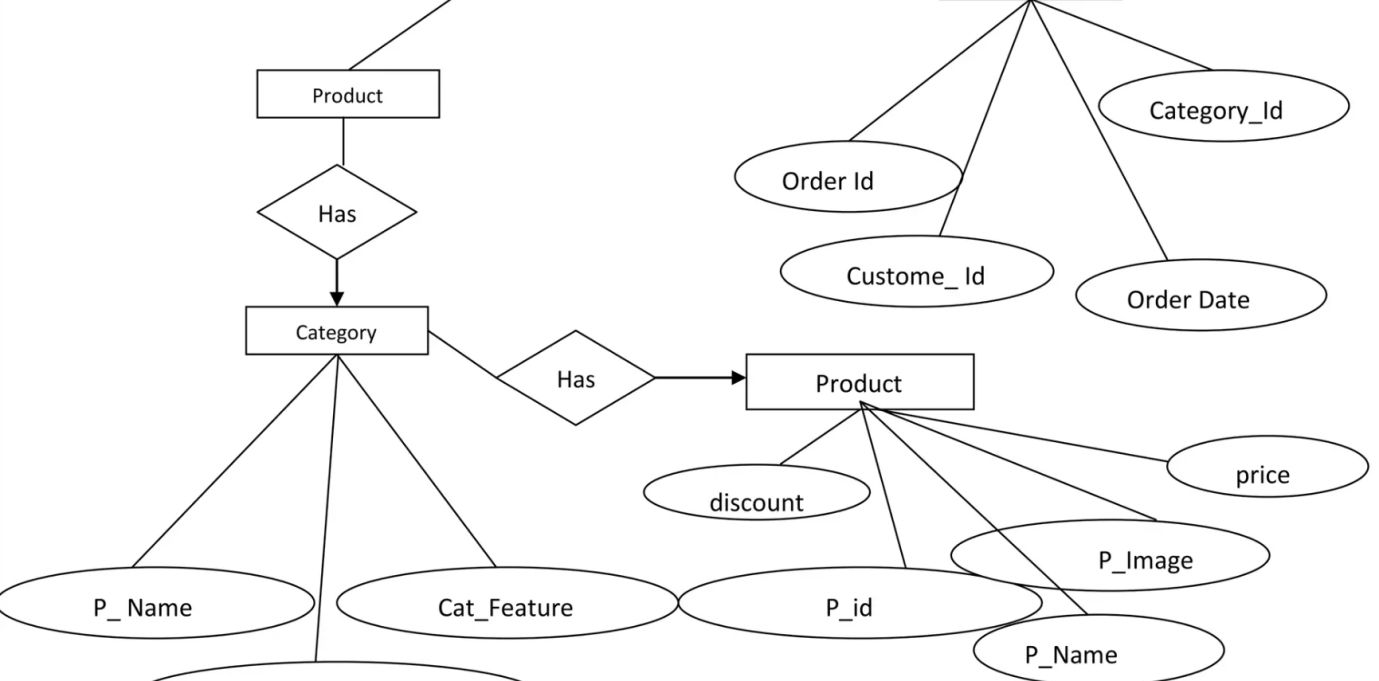


**ER Diagram:**

An Entity Relationship Diagram (ERD) is a graphical tool to express the overall structure of a database. It is based on a perception of a real world which consists of a set of basic objects. An entity is a person, place, thong or event of interest to the organization and about which data are captured, stored or processed. The attributes are various kinds of data that describes an entity. An association of several entities in an Entity-Relationship model is called relationship

ER-DIAGRAM FOR EVENT MANAGEMENT SYSTEM

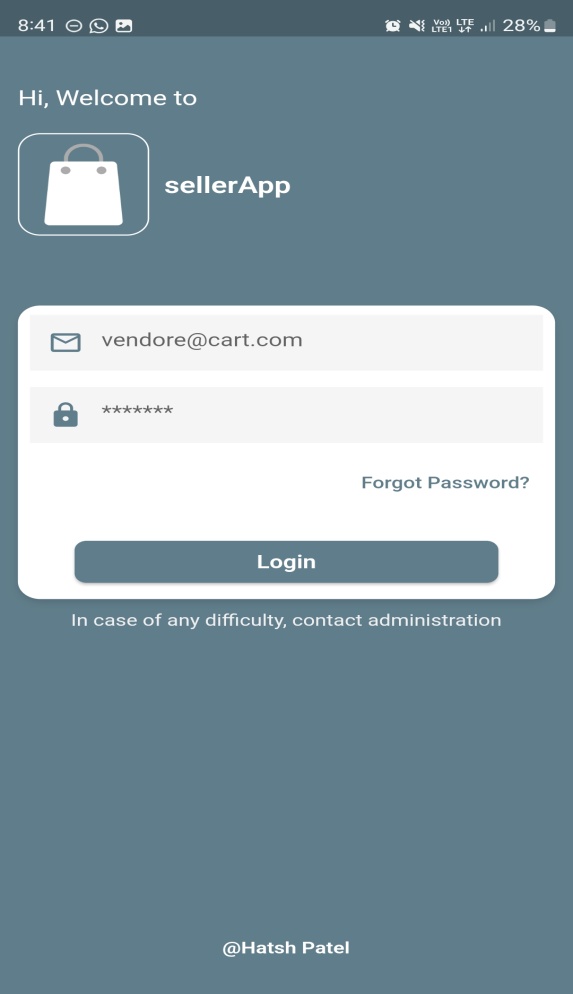
******

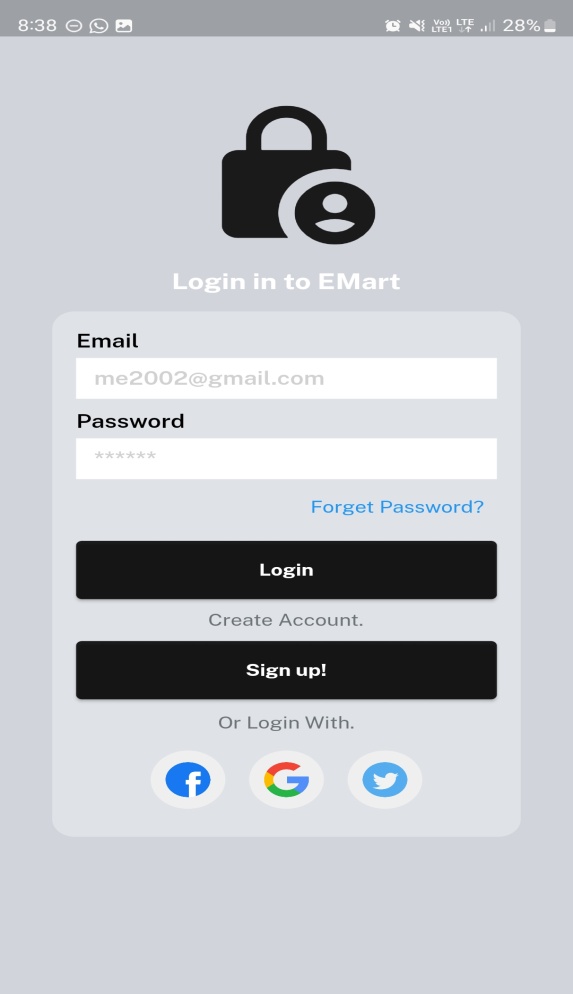
******

5.5.1 TABLE description

SCREEN LAYOUT

# Login Page:





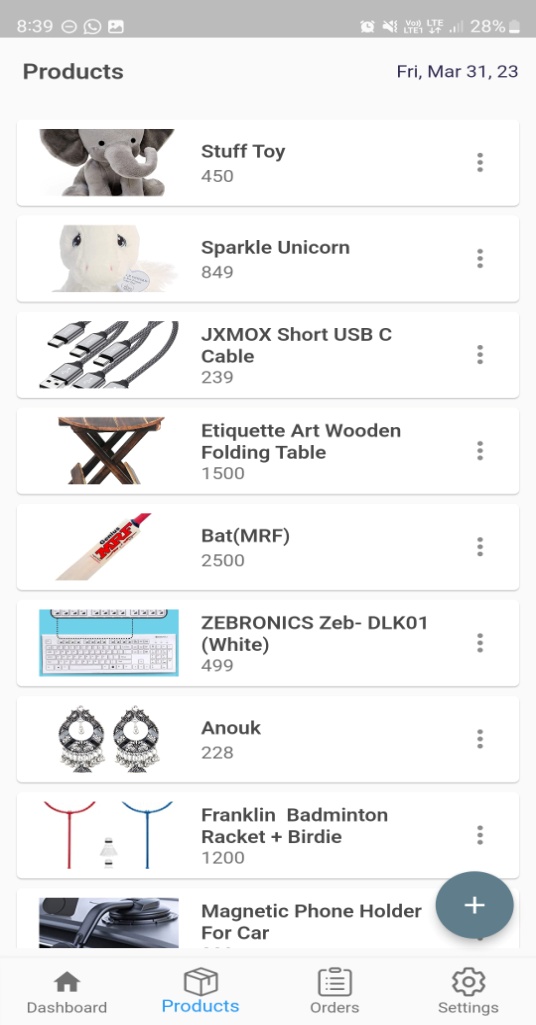
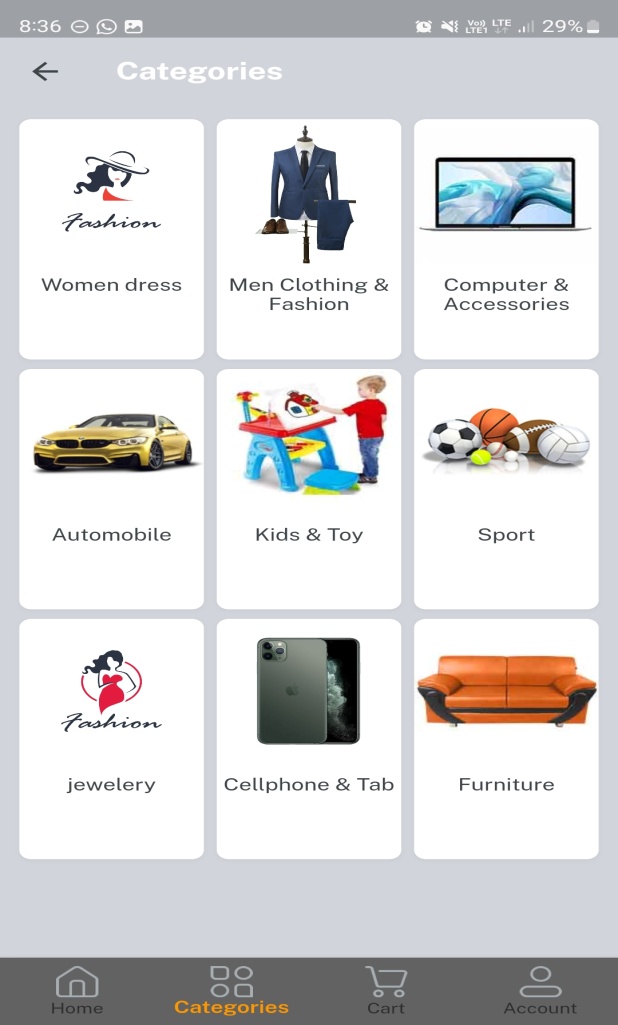
Admin Login User Login

# Home Screen:

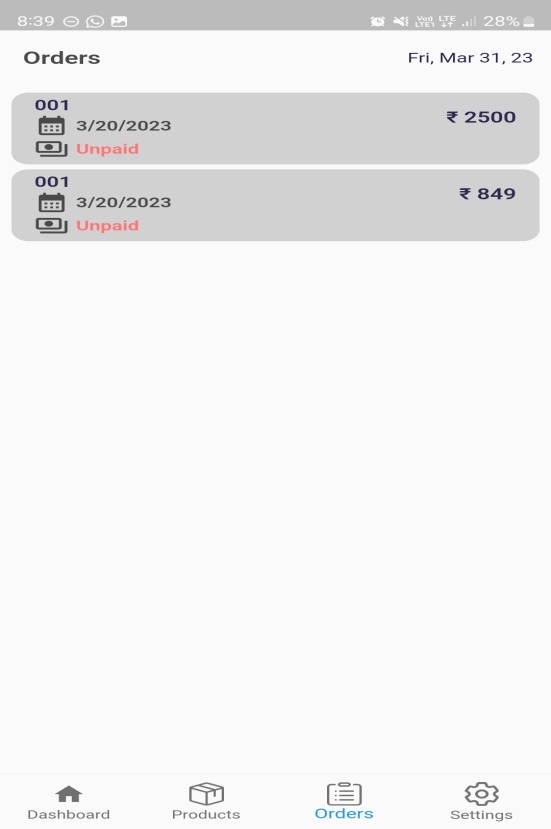
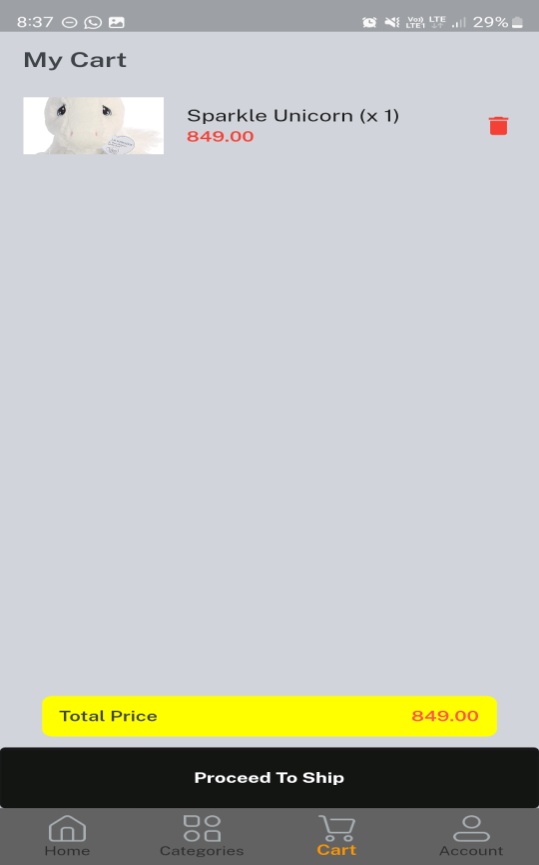
# Screenshot_20230331_083930.jpg Screenshot_20230331_083643.jpg

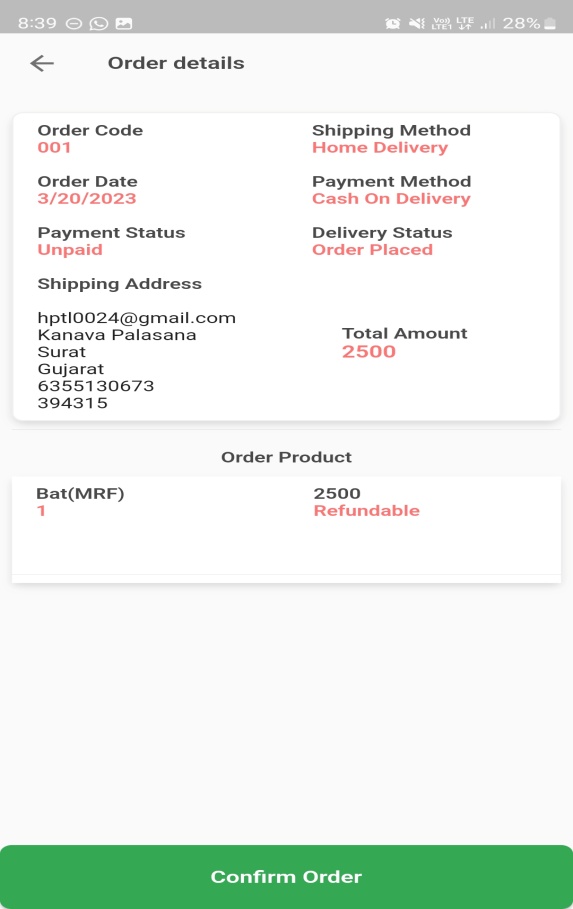
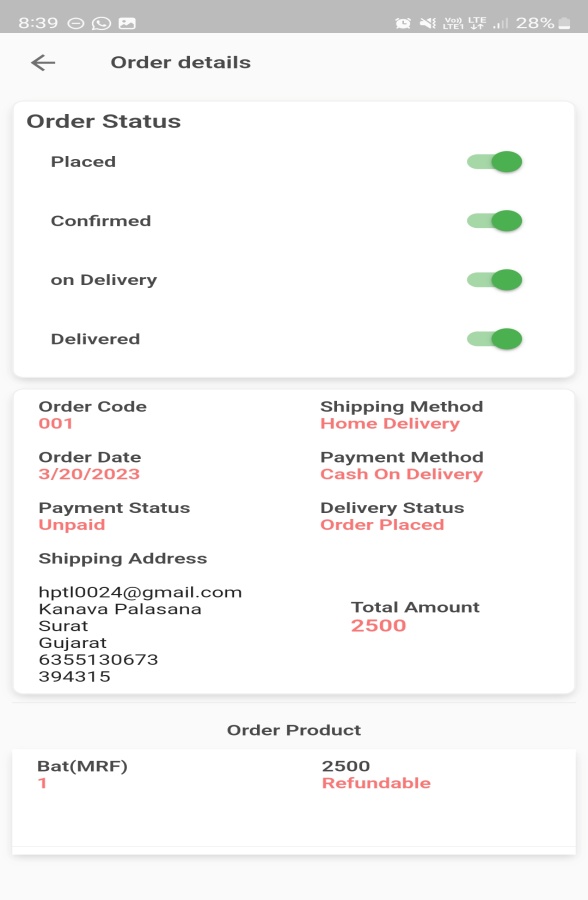
Admin User

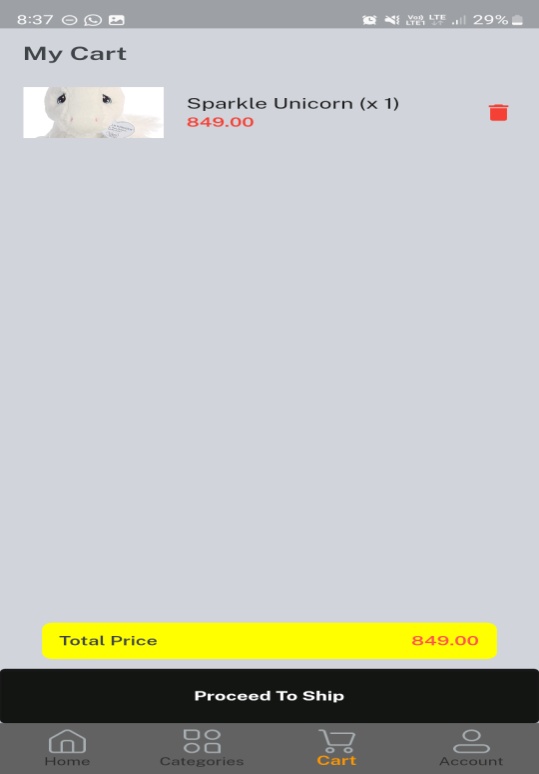
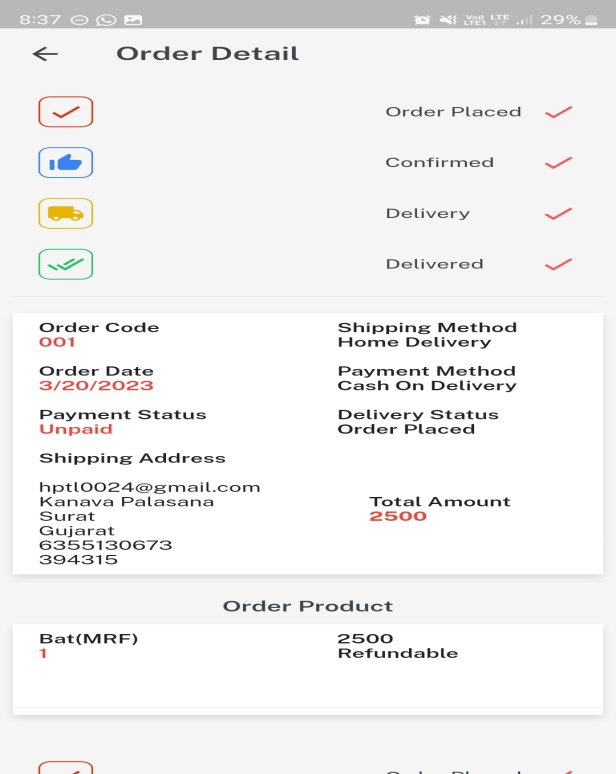
# Other Screen:

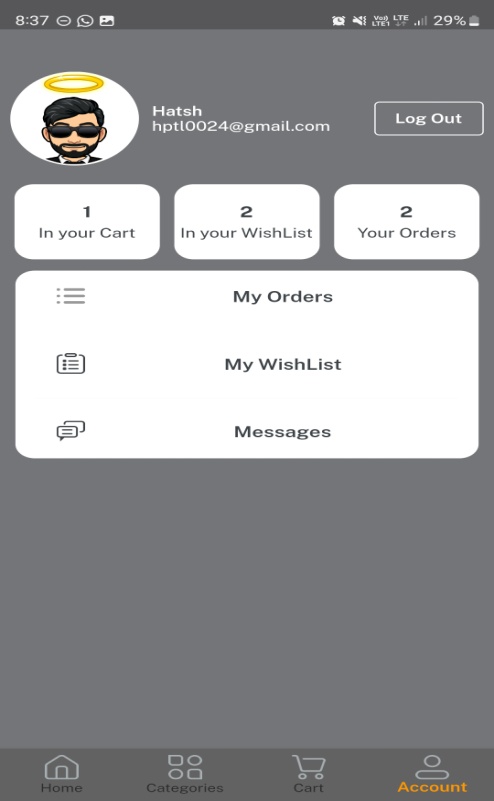
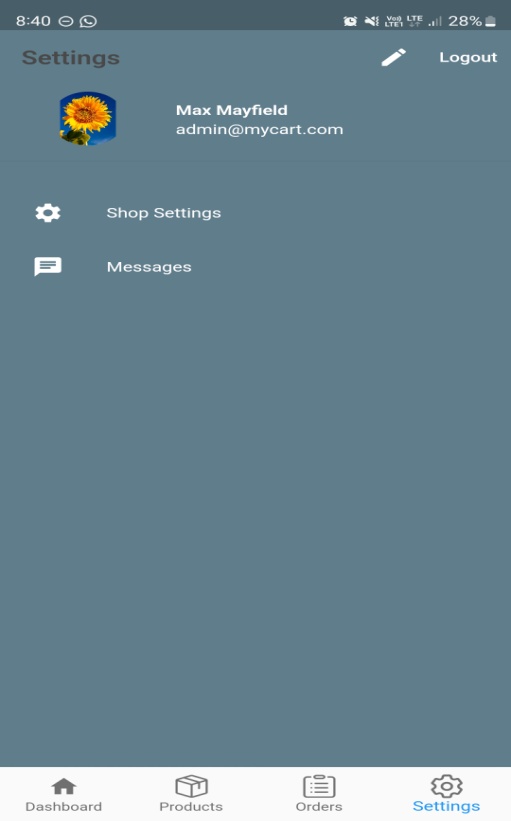
 

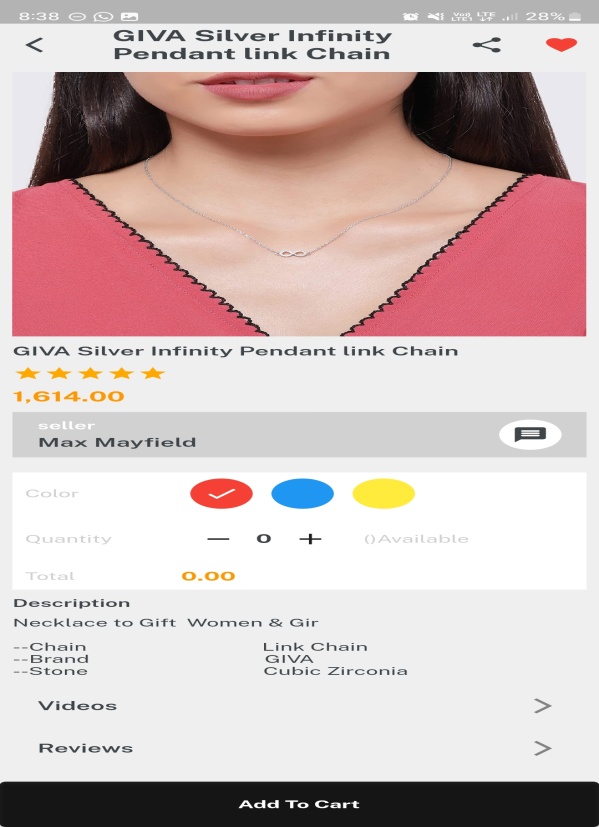
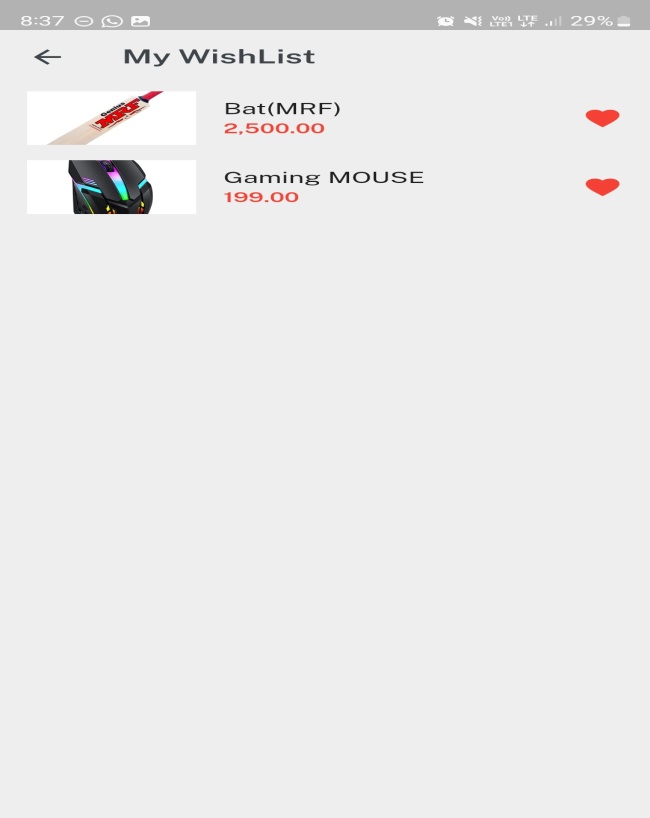
Admin User

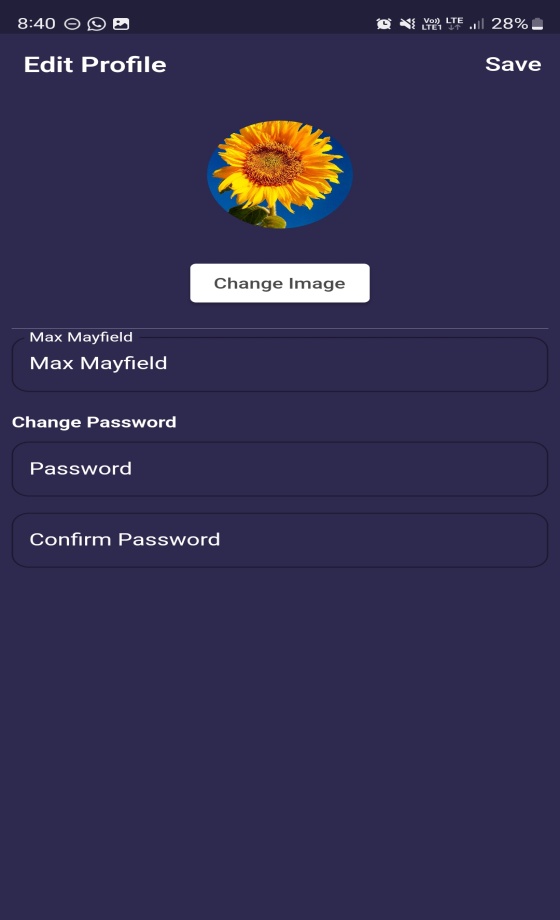
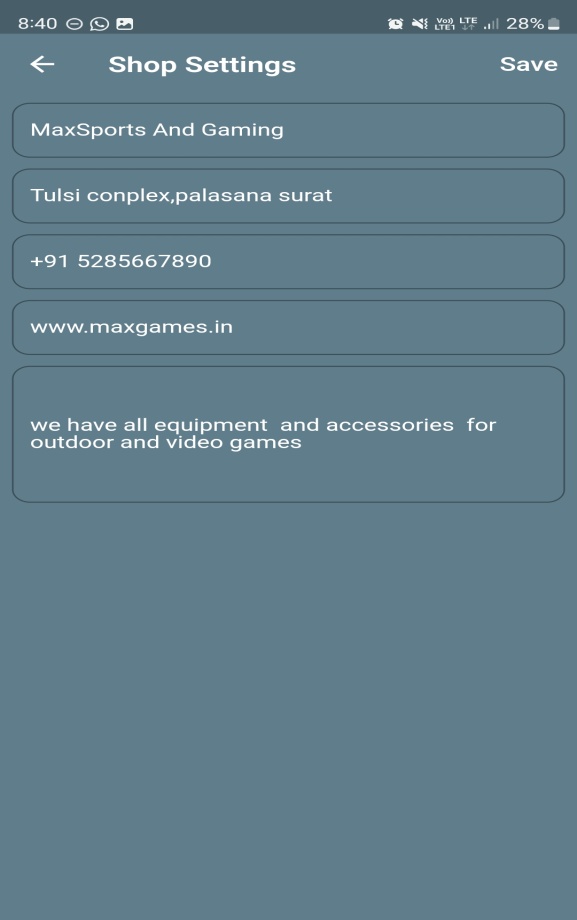
 

6.TESTING

**6.1 THE TESTING SPECTRUM**

The term implementation has different meanings ranging from the conversation of a basic application to a complete replacement of a computer system. The procedures however, are virtually the same. Implementation includes all those activities that take place to convert from old system to new. The new system may be totally new replacing an existing manual or automated system or it may be major modification to an existing system. The method of implementation and time scale to be adopted is found out initially. Proper implementation is essential to provide a reliable system to meet organization requirement.

**6.1.1 Unit Testing**

In [computer programming](https://en.wikipedia.org/wiki/Computer_programming), unit testing is a [software testing](https://en.wikipedia.org/wiki/Software_testing) method by which individual units of [source code](https://en.wikipedia.org/wiki/Source_code), sets of one or more computer program modules together with associated control data, usage procedures, and operating procedures, are tested to determine whether they are fit for use. Intuitively, one can view a unit as the smallest testable part of an application. In [procedural programming](https://en.wikipedia.org/wiki/Procedural_programming), a unit could be an entire module, but it is more commonly an individual function or procedure. In [object-oriented programming](https://en.wikipedia.org/wiki/Object-oriented_programming), a unit is often an entire interface, such as a class, but could be an individual method. Unit tests are short code fragmentscreated by programmers or occasionally by [white box testers](https://en.wikipedia.org/wiki/White-box_testing) during the development process. It forms the basis for component testing. Ideally, each [test case](https://en.wikipedia.org/wiki/Test_case) is independent from the others. Substitutes such as [method stubs](https://en.wikipedia.org/wiki/Method_stub), [mock objects](https://en.wikipedia.org/wiki/Mock_object), [fakes](https://en.wikipedia.org/wiki/Mock_object#Mocks.2C_fakes.2C_and_stubs), and [test harnesses](https://en.wikipedia.org/wiki/Test_harness) can be used to assist testing a module in isolation. Unit tests are typically written and run by [software developers](https://en.wikipedia.org/wiki/Software_developer) to ensure that code meets its design and behaves as intended.

**6.1.1.1 Benefits**

The goal of unit testing is to isolate each part of the program and show that the individual parts are correct. A unit test provides a strict, written [contract](https://en.wikipedia.org/wiki/Design_by_Contract) that the piece of code must satisfy. As a result, it affords several benefits.

* **Find problems early**

Unit testing finds problems early in the [development cycle](https://en.wikipedia.org/wiki/Development_cycle). In [test-driven development](https://en.wikipedia.org/wiki/Test-driven_development) (TDD), which is frequently used in both [extreme programming](https://en.wikipedia.org/wiki/Extreme_programming) and [scrum](https://en.wikipedia.org/wiki/Scrum_(software_development)), unit tests are created before the code itself is written. When the tests pass, that code is considered complete. The same unit tests are run against that function frequently as the larger code base is developed either as the code is changed or via an automated process with the build. If the unit tests fail, it is considered to be a bug either in the changed code or the tests themselves. The unit tests then allow the location of the fault or failure to be easily traced. Since the unit tests alert the development team of the problem before handing the code off to testers or clients, it is still early in the development process.

* **Facilitates Change**

Unit testing allows the programmer to [refactor](https://en.wikipedia.org/wiki/Refactoring) code or upgrade system libraries at a later date, and make sure the module still works correctly (e.g., in [regression testing](https://en.wikipedia.org/wiki/Regression_testing)). The procedure is to write test cases for all [functions](https://en.wikipedia.org/wiki/Subroutine) and [methods](https://en.wikipedia.org/wiki/Method_(computer_science)) so that whenever a change causes a fault, it can be quickly identified. Unit tests detect changes which may break a [design contract](https://en.wikipedia.org/wiki/Design_by_contract).

* **Simplifies Integration**

Unit testing may reduce uncertainty in the units themselves and can be used in a [bottom-up](https://en.wikipedia.org/wiki/Top-down_and_bottom-up_design) testing style approach. By testing the parts of a program first and then testing the sum of its parts, [integration testing](https://en.wikipedia.org/wiki/Integration_testing) becomes much easier.

* **Documentation**

Unit testing provides a sort of living documentation of the system. Developers looking to learn what functionality is provided by a unit, and how to use it, can look at the unit tests to gain a basic understanding of the unit's interface ([API](https://en.wikipedia.org/wiki/Application_programming_interface)).Unit [test cases](https://en.wikipedia.org/wiki/Test_case) embody characteristics that are critical to the success of the unit. These characteristics can indicate appropriate/inappropriate use of a unit as well as negative behaviors that are to be trapped by the unit. A unit test case, in and of itself, documents these critical characteristics, although many software development environments do not rely solely upon code to document the product in development.

**6.1.2 Integration Testing**

Integration testing (sometimes called integration and testing, abbreviated I&T) is the phase in [software testing](https://en.wikipedia.org/wiki/Software_testing) in which individual software modules are combined and tested as a group. It occurs after [unit testing](https://en.wikipedia.org/wiki/Unit_testing) and before [validation testing](https://en.wikipedia.org/wiki/Verification_and_validation_(software)). Integration testing takes as its input [modules](https://en.wikipedia.org/wiki/Module_(programming)) that have been unit tested, groups them in larger aggregates, applies tests defined in an integration [test plan](https://en.wikipedia.org/wiki/Test_plan) to those aggregates, and delivers as its output the integrated system ready for [system testing](https://en.wikipedia.org/wiki/System_testing).

* **Purpose**

The purpose of integration testing is to verify functional, performance, and reliability [requirements](https://en.wikipedia.org/wiki/Requirement) placed on major design items. These "design items", i.e., assemblages (or groups of units), are exercised through their interfaces using [black-box testing](https://en.wikipedia.org/wiki/Black-box_testing), success and error cases being simulated via appropriate parameter and data inputs. Simulated usage of shared data areas and [inter-process communication](https://en.wikipedia.org/wiki/Inter-process_communication) is tested and individual [subsystems](https://en.wikipedia.org/wiki/Subsystem) are exercised through their input interface. [Test cases](https://en.wikipedia.org/wiki/Test_case) are constructed to test whether all the components within assemblages interact correctly, for example across procedure calls or process activations, and this is done after testing individual modules, i.e., unit testing. The overall idea is a "building block" approach, in which verified assemblages are added to a verified base which is then used to support the integration testing of further assemblages.Software integration testing is performed according to the software development life cycle (SDLC) after module and functional tests. The cross-dependencies for software integration testing are: schedule for integration testing, strategy and selection of the tools used for integration, define the cyclomatical complexity of the software and software architecture, reusability of modules and life-cycle and versioning management.Some different types of integration testing are big-bang, [top-down, and bottom-up](https://en.wikipedia.org/wiki/Top-down_and_bottom-up_design), mixed (sandwich) and risky-hardest. Other Integration Patterns[[2]](https://en.wikipedia.org/wiki/Integration_testing#cite_note-2) are: collaboration integration, backbone integration, layer integration, client-server integration, distributed services integration and high-frequency integration.

* **Big Bang**

In the big-bang approach, most of the developed modules are coupled together to form a complete software system or major part of the system and then used for integration testing

This method is very effective for saving time in the integration testing process. However, if the test cases and their results are not recorded properly, the entire integration process will be more complicated and may prevent the testing team from achieving the goal of integration testing.A type of big-bang integration testing is called "usage model testing" which can be used in both software and hardware integration testing. The basis behind this type of integration testing is to run user-like workloads in integrated user-like environments. In doing the testing in this manner, the environment is proofed, while the individual components are proofed indirectly through their use. Usage Model testing takes an optimistic approach to testing, because it expects to have few problems with the individual components. The strategy relies heavily on the component developers to do the isolated unit testing for their product. The goal of the strategy is to avoid redoing the testing done by the developers, and instead flesh-out problems caused by the interaction of the components in the environment. For integration testing, Usage Model testing can be more efficient and provides better test coverage than traditional focused functional integration testing. To be more efficient and accurate, care must be used in defining the user-like workloads for creating realistic scenarios in exercising the environment. This gives confidence that the integrated environment will work as expected for the target customers.

* **Top-down And Bottom-up**

Bottom-up testing is an approach to integrated testing where the lowest level components are tested first, then used to facilitate the testing of higher level components. The process is repeated until the component at the top of the hierarchy is tested.All the bottom or low-level modules, procedures or functions are integrated and then tested. After the integration testing of lower level integrated modules, the next level of modules will be formed and can be used for integration testing. This approach is helpful only when all or most of the modules of the same development level are ready. This method also helps to determine the levels of software developed and makes it easier to report testing progress in the form of a percentage.Top-down testing is an approach to integrated testing where the top integrated modules are tested and the branch of the module is tested step by step until the end of the related module.Sandwich testing is an approach to combine top down testing with bottom up testing.

**6.1.4 Black-Box Testing**

Black-box testing is a method of [software testing](https://en.wikipedia.org/wiki/Software_testing) that examines the functionality of an application without peering into its internal structures or workings. This method of test can be applied virtually to every level of software testing: [unit](https://en.wikipedia.org/wiki/Unit_test), [integration](https://en.wikipedia.org/wiki/Integration_testing), [system](https://en.wikipedia.org/wiki/System_testing) and [acceptance](https://en.wikipedia.org/wiki/Acceptance_test). It typically comprises most if not all higher level testing, but can also dominate unit testing as well.

* **Test Procedures**

Specific knowledge of the application's code/internal structure and programming knowledge in general is not required. The tester is aware of what the software is supposed to do but is not aware of how it does it. For instance, the tester is aware that a particular input returns a certain, invariable output but is not aware of how the software produces the output in the first place.

* **Test Cases**

Test cases are built around specifications and requirements, i.e., what the application is supposed to do. Test cases are generally derived from external descriptions of the software, including specifications, requirements and design parameters. Although the tests used are primarily functional in nature, non-functional tests may also be used. The test designer selects both valid and invalid inputs and determines the correct output, often with the help of an [oracle](https://en.wikipedia.org/wiki/Oracle_(software_testing)) or a previous result that is known to be good, without any knowledge of the test object's internal structure.

* **Test Design Techniques**

Typical black-box test design techniques include:

* [Decision table](https://en.wikipedia.org/wiki/Decision_table) testing
* [All-pairs testing](https://en.wikipedia.org/wiki/All-pairs_testing)
* [Equivalence partitioning](https://en.wikipedia.org/wiki/Equivalence_partitioning)
* [Boundary value analysis](https://en.wikipedia.org/wiki/Boundary_value_analysis)
* [Cause–effect graph](https://en.wikipedia.org/wiki/Cause%E2%80%93effect_graph)
* [State transition](https://en.wikipedia.org/wiki/State_transition) testing
* [Use case](https://en.wikipedia.org/wiki/Use_case) testing
* [Domain analysis](https://en.wikipedia.org/wiki/Domain_analysis)
* Combining technique

**6.1.5 White-Box Testing**

White-box testing (also known as clear box testing, glass box testing, transparent box testing, and structural testing) is a method of testing [software](https://en.wikipedia.org/wiki/Software) that tests internal structures or workings of an application, as opposed to its functionality (i.e. [black-box testing](https://en.wikipedia.org/wiki/Black-box_testing)). In white-box testing an internal perspective of the system, as well as programming skills, are used to design test cases. The tester chooses inputs to exercise paths through the code and determine the appropriate outputs. This is analogous to testing nodes in a circuit, e.g. [in-circuit testing](https://en.wikipedia.org/wiki/In-circuit_test) (ICT). White-box testing can be applied at the [unit](https://en.wikipedia.org/wiki/Unit_testing), [integration](https://en.wikipedia.org/wiki/Integration_testing) and [system](https://en.wikipedia.org/wiki/System_testing) levels of the [software testing](https://en.wikipedia.org/wiki/Software_testing) process. Although traditional testers tended to think of white-box testing as being done at the unit level, it is used for integration and system testing more frequently today. It can test paths within a unit, paths between units during integration, and between subsystems during a system–level test. Though this method of test design can uncover many errors or problems, it has the potential to miss unimplemented parts of the specification or missing requirements.

White-box test design techniques include the following [code coverage](https://en.wikipedia.org/wiki/Code_coverage) criteria:

* [Control flow](https://en.wikipedia.org/wiki/Control_flow) testing
* Data flow testing
* Branch testing
* Statement coverage
* Decision coverage
* [Modified condition/decision coverage](https://en.wikipedia.org/wiki/Modified_condition/decision_coverage)
* Prime path testing
* Path testing

White-box testing is a method of testing the application at the level of the source code. These test cases are derived through the use of the design techniques mentioned above: [control flow](https://en.wikipedia.org/wiki/Control_flow) testing, data flow testing, branch testing, path testing, statement coverage and decision coverage as well as modified condition/decision coverage. White-box testing is the use of these techniques as guidelines to create an error free environment by examining any fragile code. These White-box testing techniques are the building blocksof white-box testing, whose essence is the careful testing of the application at the source code level to prevent any hidden errors later on.[[1]](https://en.wikipedia.org/wiki/White-box_testing#cite_note-level-1) These different techniques exercise every visible path of the source code to minimize errors and create an error-free environment. The whole point of white-box testing is the ability to know which line of the code is being executed and being able to identify what the correct output should be.

**6.1.6 System Testing**

System testing of software or hardware is testing conducted on a complete, integrated system to evaluate the system's compliance with its specified [requirements](https://en.wikipedia.org/wiki/Requirements). System testing falls within the scope of [black-box testing](https://en.wikipedia.org/wiki/Black-box_testing), and as such, should require no knowledge of the inner design of the code or logic. As a rule, system testing takes, as its input, all of the "integrated" software components that have passed [integration testing](https://en.wikipedia.org/wiki/Integration_testing) and also the software system itself integrated with any applicable hardware system(s). The purpose of integration testing is to detect any inconsistencies between the software units that are integrated together (called assemblages) or between any of the assemblages and the hardware. System testing is a more limited type of testing; it seeks to detect defects both within the "inter-assemblages" and also within the system as a whole.

* **Testing The Whole System**

System testing is performed on the entire system in the context of a [Functional Requirement](https://en.wikipedia.org/wiki/Functional_requirements) Specification(s) (FRS) and/or a [System Requirement](https://en.wikipedia.org/wiki/Requirements_analysis) Specification (SRS). System testing tests not only the design, but also the behavior and even the believed expectations of the customer. It is also intended to test up to and beyond the bounds defined in the software/hardware requirements specification(s).