**DECLARATION**

I, Suraj Pandey, hereby declare that the work presented in the project report entitled submitted to Department of Computer Application, Institute of Proff. Excellence & Mgmt., affiliated to Dr. A. P. J Abdul Kalam Technical University, Lucknow, Approved by AICTE for the partial fulfillment of the award of degree of “MASTER IN COMPUTER APPLICATIONS” is an authentic record of my work carried out during the fourth semester, 2023 under the supervision of my Project Guide as “Ms. Supriya Sharma” Department of Computer Application, Institute of Proff. Excellence & Mgmt., Ghaziabad, Uttar Pradesh. The matter embodied in this project report has not been submitted elsewhere by anybody for the award of any other degree.

Suraj Pandey

2101140140029



**INSTITUTE OF PROFESSIONAL EXCELLENCE & MANAGEMENT**

*(Affiliated APJ Abdul Kalam Technical University, Lucknow)*

**CERTIFICATE**

*This is to certify that the Project report submitted by Mr. Suraj Pandey bearing roll no 2101140140029 on the title Expense Tracker web application is a record of project work done by him/ her during the academic year 2022-23 under my guidance and supervision in partial fulfilment of the requirements for the award of the degree of Master of Computer Application from Dr. A. P. J. Abdul Kalam Technical University, Lucknow, is authentic record under the Institute.*

***Project Guide : Head CA Dept:***

*Ms. Supriya Sharma Mr. Naveen Kumar*

**ACKNOWLEDGEMENT**

Completing a task is never a one person effort. It is often the result of invaluable contribution of individuals in a direct or indirect manner that helps in sharing a making of success.

I would like to thank people who were part of this work in numerous ways. In particular, I wish to thank “Ms. Supriya Sharma”, A.P., Department of Computer Application, as my project guide for their suggestions and improvements in this project and providing continuous guidance at each and every stage of the project.

Thanks are extended especially to Mr. Naveen Kumar, Head of Computer Application Department, Institute of Proff. Excellence & Mgmt.,. I had the privilege to receive great inspiration and guidance from him and other faculty members, which helped us to set the objectives of this study and complete the report. I am fortunate to have constant encouragement and practical guidance in all aspect of our study.

I must also be thankful to my classmates and friends for their continuous co-operations and help in completing this project.

Suraj Pandey

2101140140029

|  |  |
| --- | --- |
| **Description** | **Page no** |
| 1. Introduction  * Overview * Background study * Objective * Benefits & Problems |  |
| 1. Tools and Technology used  * Hardware & software requirements * Technology used |  |
| 1. Methodology & requirement analysis  * Methodology * User characteristic * System analysis * Feasibility study * Requirements |  |
| 1. System Design  * Use case diagram * ER diagram * Data Flow Diagram * Activity Diagram |  |
| 1. Snapshots And snippets of code of Project |  |
| 1. Testing  * Testing Strategies |  |
| 1. Future Scope |  |
| 1. Conclusion |  |
| 1. Bibliography |  |

**TABLE OF CONTENT**

1. **INTRODUCTION**
   1. **OVERVIEW**

An Eexpense Tracker web application is a useful tool for managing and monitoring personal or business expenses. It allows users to track their spending, categorize expenses, set budgets, and generate reports to gain insights into their financial habits. Here's an outline of the key features and components which can consider when developing an expense tracker web application.

User Authentication Implement a user registration and login system to ensure secure access to the application. Users should have individual accounts to manage their personal expense data.

Dashboard Create a user-friendly dashboard that provides an overview of expenses, budget status, and other key financial metrics. This will be the main landing page after login, displaying a summary of the user's financial situation.

Expense Entry Allow users to input their expenses by specifying the amount, date, category, and any additional notes. Provide options for recurring expenses and support for multiple currencies if needed.

Expense Categories Provide pre-defined and customization categories (e.g., groceries, transportation, entertainment) to help users organize their expenses. Users should be able to add, edit, and delete categories based on their preferences.

Budget Management Enable users to set budgets for different categories or overall spending limits. The application should provide visual indicators to track progress and alert users when they exceed their set budgets.

Reports and Analytic generate detailed reports and visualizations to help users analyze their spending patterns over time. Common reports include monthly expense breakdowns, category-wise spending, and trends over specific periods.

Notifications and Reminders Implement notifications or email reminders to keep users informed about upcoming bills, approaching budget limits, or irregular spending behavior.

Data Backup and Sync Provide a secure data storage system that allows users to sync their expense data across multiple devices. Regularly backup data to prevent any loss or corruption.

Data Export Enable users to export their expense data in various formats such as CSV or PDF. This feature allows users to analyze their data offline or import it into other personal finance tools.

Security and Privacy Implement proper security measures to protect user data, such as encryption for sensitive information, secure connections (HTTPS), and adherence to privacy regulations.

Mobile Responsiveness Design the application to be mobile-friendly, ensuring it works well on different screen sizes and devices.

User Settings: Provide options for users to customize their preferences, such as currency settings, language preferences, and notification preferences.

* 1. **BACKGROUND STUDY**

Before developing an expense tracker web application, it's essential to conduct a background study to understand the requirements, market, and existing solutions. Here are some key points to consider during the background study.

User Needs Identify the target audience for your expense tracker application. Determine their primary needs and pain points when it comes to managing and tracking expenses. Conduct surveys, interviews, or user research to gather insights and understand their preferences.

We have identified some other point for this like Market Research, Feature Analysis, Technology Stack, Design and User Experience, Monetization Models, Integration Opportunities, Usability Testing, Regulations and Compliance.

* + 1. **What Is Expense Tracker?**

An Eexpense Tracker web application is a software application that allows users to track, manage, and analyze their expenses in a digital format. It provides a convenient and centralized platform for individuals or businesses to monitor their spending habits, categorize expenses, set budgets, and generate reports for financial analysis, Expense tracker web apps typically offer the following key features:

Expense Entry - Users can input their expenses by entering details such as the amount, date, category, description.

Expense Categorization - Expenses can be categorized into different predefined or customizable categories (e.g., food, transportation, utilities) to organize and analyze spending patterns,

Budgeting - Users can set budget limits for specific categories or overall spending and receive notifications or alerts when they approach or exceed their set limits.

Reporting and Analytic application generates reports and visualizations to provide users with insights into their spending habits. This includes expense breakdowns, charts, graphs, and trends over specified time periods.

Receipt and Document Management - Some expense tracker apps allow users to upload and attach receipts or relevant documents to individual expenses for record-keeping purposes, Currency Conversion - If users deal with multiple currencies, the app may offer currency conversion functionality to help track expenses accurately.

Sync and Backup - Users can sync their expense data across multiple devices or platforms, ensuring their information is accessible and up to date. Regular backups of data are performed to prevent loss.

Notifications and Reminders - Users may receive reminders about upcoming bills, payment due dates, or irregular spending patterns to stay on top of their finances.

Security and Privacy Expense tracker web apps implement security measures such as encryption, secure connections (HTTPS), and data privacy policies to protect user information. Integration with Financial Services: Some apps offer integration with banks, credit cards, or financial institutions, allowing users to automatically import transactions and reconcile them with their expenses, Mobile Accessibility Many expense tracker web apps are mobile-responsive or offer dedicated mobile apps, enabling users to track their expenses on the go.

* 1. **OBJECTIVE**

The aim of an expense tracker web app is to simplify the process of expense management, provide insights into spending patterns, and empower users to make informed financial decisions. It helps individuals and businesses stay organized, set financial goals, and gain control over their finances.

To develop an easy-to-use web-based interface where user can search for daily expenses on web apps are mobile-responsive or offer dedicated mobile apps, enabling users to track their expenses on the go.

* 1. **SCOPE**

It's important to note that the scope can be expanded or customized based on the target audience, business requirements, and additional features that may differentiate the app from competitors, but initially we’re keeping in the mind below area.

Expense Management, Expense Entry Users can add new expenses by specifying the amount, date, category, description, and other relevant details.

* Expense Categorization - Users can categorize expenses into predefined or customizable categories for better organization and analysis.
* Expense Editing and Deletion - Users should have the ability to edit or delete previously entered expenses.
* Expense Search and Filtering - Users can search for specific expenses based on criteria such as date, category, or description.
* Receipt and Document Management - Users can upload and attach receipts or documents related to their expenses.
* It's important to note that the scope can be expanded or customized based on the target audience, business requirements.
  1. **LIMITATIONS**

It's important to consider these limitations when developing or using an expense tracker web app and provide solutions or alternatives to mitigate their impact. Regular user feedback and continuous improvement can help address some of these limitations over time, while expense tracker web apps offer numerous benefits, they also have some limitations that should be considered as below: -

* Manual Data Entry - Expense tracker apps rely on users manually entering their expenses, which can be time-consuming and prone to errors. Users may forget to record certain expenses or find the process tedious, leading to incomplete or inaccurate data.
* Dependency on User Discipline - The effectiveness of an expense tracker web app relies on the user's discipline and consistency in entering and categorizing expenses. If users fail to regularly update their expense data, the apps’ insights and reports may be incomplete or unreliable.
* Limited Integration - Expense tracker apps may have limitations in integrating with financial institutions, banks, or credit cards. While some apps offer integration capabilities, not all financial institutions or countries may be supported, limiting the automatic import and synchronization of transactions.
* Security Concerns - Storing financial data in an expense tracker app carries potential security risks. It's crucial to implement robust security measures to protect user information, such as encryption, secure connections, and following best practices for data storage and access control.
* Lack of Real-Time Data - Expense tracker apps typically rely on manual entry and periodic data synchronization. As a result, real-time updates may not be available, and users may not have an immediate view of their current financial situation.
* Limited Financial Analysis - While expense tracker apps provide basic reporting and analysis features, they may not offer advanced financial analysis tools or comprehensive insights into investment strategies, tax planning, or long-term financial goals. Users may need to rely on additional tools or consult financial professionals for in-depth financial planning.
* User Learning Curve - Depending on the complexity and user interface of the app, users may require some time to understand the features and functionality, leading to a learning curve before they can fully utilize the apps’ capabilities.
* Platform Dependency - Expense tracker web apps may be designed for specific web browsers or operating systems, limiting accessibility for users who prefer different platforms or devices.

1. **TOOL & TECHNOLOGY USED**
   1. **HARDWARE AND SOFTWARE REQUIREMENT**

**Hardware requirement:**

Hardware requirements of this project are very common now a days and can be found on any computer system. It is kept in mind while designing that no extra hardware support which can be needed to run the software should be there so that it can be afforded by any firm or industry.

|  |  |  |
| --- | --- | --- |
| **SL** | **Hardware** | **Minimum System Requirement** |
| 01 | Processor | 1 GHz or faster |
| 02 | Memory | 4 GB RAM |
| 03 | Disk store | 128 GB |

**Software requirement:**

|  |  |  |
| --- | --- | --- |
| **SL** | **Software** | **Requirement** |
| 01 | Operating System | Windows 10 |
| 02 | Client-side Scripting | javaScript |
| 03 | Programming Language | C# |
| 04 | IDE/workbench | VisualStudio 2022 |
| 05 | Database | Microsoft SQL Server |
| 06 | Framework | Dot Net Core MVC |
| 07 | Data Visualization | Syncfusion |

* 1. **TECHNOLOGY USED**
     1. **HTML**

Hyper Text Markup Language is used to create the main structure of a web page,  
which outlines the important components in the web page which we see. Hypertext Markup Language (HTML) is the standard markup language for creating web pages and web applications. With Cascading Style Sheets (CSS) and JavaScript it forms a triad of cornerstone technologies for the World Wide Web. Web browsers receive HTML documents from a web server or from local storage and render them into multimedia web pages. HTML describes the structure of a web page semantically and originally included cues for the appearance of the document.

HTML elements are the building blocks of HTML pages. With HTML constructs, images and other objects, such as interactive forms, may be embedded into the rendered page. It provides a means to create structured documents by denoting structural semantics for text such as headings, paragraphs, lists, links, quotes and other items. HTML elements are delineated by tags, written using angle brackets. Tags such as <img /> and <input /> introduce content into the page directly. Others such as <p>...</p> surround and provide information about document text and may include other tags as sub-elements. Browsers do not display the HTML tags but use them to interpret the content of the page.

HTML can embed programs written in a scripting language such as JavaScript which affect the behavior and content of web pages. Inclusion of CSS defines the look and layout of content. The World Wide Web Consortium (W3C), maintainer of both the HTML and the CSS standards, has encouraged the use of CSS over explicit presentational HTML since 1997. HTML markup consists of several key components, including those called tags (and their attributes), character-based data types, character references and entity references. HTML tags most commonly come in pairs like <h1> and </h1>, although some represent empty elements and so are unpaired, for example <img>. The first tag in such a pair is the start tag, and the second is the end tag (they are also called opening tags and closing tags).

* + 1. **CSS**

CSS Cascading Style Sheets (CSS) is a style sheet language used for describing the presentation of a document written in a markup language. Although most often used to set the visual style of web pages and user interfaces written in HTML and XHTML, the language can be applied to any XML document, including plain XML, SVG and XUL, and is applicable to rendering in speech, or on other media. Along with HTML and JavaScript, CSS is a cornerstone technology used by most websites to create visually engaging webpages, user interfaces for web applications, and user interfaces for many mobile applications.

CSS is designed primarily to enable the separation of presentation and content, including aspects such as the layout, colors, and fonts. This separation can improve content accessibility, provide more flexibility and control in the specification of presentation characteristics, enable multiple HTML pages to share formatting by specifying the relevant CSS in a separate “.CSS” file, and reduce complexity and repetition in the structural content.

Separation of formatting and content makes it possible to present the same markup page in different styles for different rendering methods, such as on-screen, in print, by voice (via speech-based browser or screen reader), and on Braille-based tactile devices. It can also display the web page differently depending on the screen size or viewing device. Readers can also specify a different style sheet, such as a CSS file stored on their own computer, to override the one the author specified.

Changes to the graphic design of a document (or hundreds of documents) can be applied quickly and easily, by editing a few lines in the CSS file they use, rather than by changing markup in the documents.

The CSS specification describes a priority scheme to determine which style rules apply if more than one rule matches against a particular element. In this so-called cascade, priorities (or weights) are calculated and assigned to rules, so that the results are predictable. The CSS specifications are maintained by the World Wide Web Consortium (W3C). Internet media type (MIME type) text/CSS is registered for use with CSS by RFC 2318 (March 1998). The W3C operates a free CSS validation service for CSS documents.

* + 1. **SERVER-SIDE SCRIPTING**

Server-side scripting is a technique used in web development which involves employing scripts on a web server which produce a response customized for each user’s (client’s) request to the website. The alternative is for the web server itself to deliver a static web page. Scripts can be written in any of a number of server-side scripting languages that are available. Server-side scripting is distinguished from client-side scripting where embedded scripts, such as JavaScript, are run client-side in a web browser, but both techniques are often used together.

Server-side scripting is often used to provide a customized interface for the user. These scripts may assemble client characteristics for use in customizing the response based on those characteristics, the user’s requirements, access rights, etc. Server-side scripting also enables the website owner to hide the source code that generates the interface, whereas with client-side scripting, the user has access to all the code received by the client. A down-side to the use of server-side scripting is that the client needs to make further requests over the network to the server to show new information to the user via the web browser. These requests can slow down the experience for the user, place more load on the server, and prevent use of the application when the user is disconnected from the server.

When the server serves data in a commonly used manner, for example according to the HTTP or FTP protocols, users may have their choice of a number of client programs (most modern web browsers can request and receive data using both of those protocols). In the case of more specialized applications, programmers may write their own server, client, and communications protocol that can only be used with one another. Programs that run on a user’s local computer without ever sending or receiving data over a network are not considered clients, and so the operations of such programs would not be considered client-side operations. There are several languages that can be used for server-side programming: • PHP • ASP.NET (C# OR Visual Basic) • C++ • Java and JSP • Python • Ruby on Rails and so on.

* + 1. **CLIENT-SIDE SCRIPTING**

Client-side scripting is changing interface behaviors within a specific web page in response to mouse or keyboard actions, or at specified timing events. In this case, the dynamic behavior occurs within the presentation. The client-side content is generated on the user's local computer system.

Such web pages use presentation technology called rich interfaced pages. Client-side scripting languages like JavaScript or ActionScript, used for Dynamic HTML (DHTML) and Flash technologies respectively, are frequently used to orchestrate media types (sound, animations, changing text, etc.) of the presentation. Client-side scripting also allows the use of remote scripting, a technique by which the DHTML page requests additional information from a server, using a hidden frame, XML or HTTP Requests, or a Web service.

The first widespread use of JavaScript was in 1997, when the language was standardized as ECMAScript and implemented in Netscape 3. Example: The client-side content is generated on the client's computer. The web browser retrieves a page from the server, then processes the code embedded in the page (typically written in JavaScript) and displays the retrieved page's content to the user. The most popularly used client-side scripting languages is Java Script.

* + 1. **Dot Net Core MVC**

ASP.NET Core MVC is a web application framework developed by Microsoft that follows the Model-View-Controller (MVC) architectural pattern. It is a part of the .NET Core platform, which is an open-source, cross-platform framework for building modern web applications and services.

MVC is a software design pattern that separates an application into three main components: the model, the view, and the controller. Here's a brief overview of each component:

1. Model - The model represents the application's data and business logic. It defines the structure and behavior of the data and encapsulates the interactions with the underlying database or other data sources.
2. View - The view is responsible for presenting the user interface (UI) to the users. It defines how the data from the model should be displayed and provides the necessary HTML markup, CSS styling, and client-side scripting.
3. Controller - The controller handles the user's requests, processes the input, and orchestrates the flow of data between the model and the view. It contains the logic for handling user interactions, such as capturing form inputs, performing data validation, and determining the appropriate response to send back to the user.

ASP.NET Core MVC provides a framework for building web applications by leveraging these MVC concepts. It offers features like routing, model binding, dependency injection, and Razor views for generating dynamic HTML content. ASP.NET Core MVC is highly modular, flexible, and suitable for creating scalable, high-performance web applications that can run on various platforms, including Windows, macOS, and Linux.

With the introduction of .NET 5 and later versions, Microsoft has unified the frameworks, merging ASP.NET Core and .NET Framework into a single, cross-platform framework called ".NET." Therefore, the term ".NET Core" is being replaced by ".NET" in the context of web development.

* + 1. **Microsoft SQL Server**

Microsoft SQL Server is a relational database management system (RDBMS) developed by Microsoft. It is a robust and feature-rich database platform used for storing, managing, and retrieving data in various applications and systems.

SQL Server supports the Structured Query Language (SQL) for interacting with databases. It provides a scalable and secure platform for managing databases of different sizes, from small applications to large enterprise systems. Here are some key features and components of Microsoft SQL Server:

1. Database Engine The Database Engine is the core component of SQL Server. It handles tasks such as data storage, query processing, transaction management, and security. It supports multiple data models, including relational, XML, spatial, and graph data.
2. Management Tools SQL Server provides various tools for managing and administering databases. SQL Server Management Studio (SSMS) is a graphical user interface (GUI) tool that allows developers and administrators to interact with SQL Server, create databases, write queries, and perform administrative tasks. SQL Server also offers command-line tools and PowerShell modules for automation and scripting.
3. Integration Services (SSIS) SQL Server Integration Services is a platform for building data integration and ETL (Extract, Transform, Load) solutions. It enables the extraction, transformation, and loading of data from various sources into SQL Server or other destinations.
4. Analysis Services (SSAS) SQL Server Analysis Services provides online analytical processing (OLAP) and data mining capabilities. It allows users to create multidimensional models and perform complex analysis on large datasets.
5. Reporting Services (SSRS) SQL Server Reporting Services is a reporting platform that enables the creation, management, and delivery of interactive and paginated reports. It offers a wide range of reporting features and supports various output formats, including PDF, Excel, and web-based formats.
6. Azure SQL Database Microsoft SQL Server also offers a cloud-based version called Azure SQL Database. It provides a fully managed database service in Microsoft Azure, offering scalability, high availability, and automatic backups.

SQL Server is widely used by organizations of all sizes and across different industries. It is commonly used for building business applications, data warehousing, data analytics, and powering websites and e-commerce platforms. SQL Server offers enterprise-level performance, security, and reliability, making it a popular choice for managing critical data and applications.

* + 1. **Syncfusion**

Sync fusion provides a package called "Syncfusion.EJ2.AspNetCore. Mvc" for working with their UI controls and components in ASP.NET Core MVC applications. This package enables developers to easily integrate and use Sync fusion’s UI controls within their MVC projects.

The "Syncfusion.EJ2.AspNetCore. Mvc" package is specifically designed for ASP.NET Core MVC applications and provides server-side wrappers for Sync fusion’s Essential JS 2 (EJ2) controls. These controls include a wide range of UI components such as grids, charts, calendars, dropdowns, tree views, editors, and more.

By using the Sync fusion package, you can leverage pre-built, customizable UI components that are optimized for performance and designed to work well together. The controls offer a consistent and visually appealing user experience across different browsers and devices.

Overall, the "Syncfusion.EJ2.AspNetCore. Mvc" package simplifies the integration and usage of Sync fusion controls within ASP.NET Core MVC applications, allowing developers to build rich and interactive UIs with ease.

1. **SYSTEM DESIGN**
   1. **METHODOLOGY**

This Section describes the methodology applied during the development of the proposed project. A methodology is a model, which project managers employ for the design, planning, implementation, and achievement of their project objectives. Effective project management is essential in absolutely any organization, regardless of the nature of the business and the scale of the organization. From choosing a project to right through to the end, it is important that the project is carefully and closely managed. Based on the nature of my project solution, it was essential to use incremental Software development life cycle (SDLC). The project typically has several Phases and the level of control required over each phase are primarily defined by the nature of the Project, the complexity of the same and the industry to which the Project has to cater to. An Incremental (SDLC) model consists of a number of dependent increments that are completed in a prescribed sequence. Each increment includes a Launching, Monitoring and Controlling, and Closing Process Group for the functions and features in that increment only. Each increment integrates additional parts of the solution until the final increment, where the remaining parts of the solution are integrated.

**3.1.1. 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 JavaScript and Dot Net Core and could iterate from one technology to another to ensure, I implement all the functionalities effectively.
* Dot NET Core MVC is well-suited for iterative development due to its modular and component-based architecture. The framework provides flexibility in building and integrating incremental features, allowing the development team to deliver working software at the end of each iteration. This aligns with the incremental SDLC approach, where each iteration focuses on implementing a subset of features.
* The project had some high-risk features and goals.

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.

* Early Delivery of Value: Incremental SDLC enables the project to deliver functional software increments early, providing immediate value to stakeholders.
* Continuous Feedback and Adaptability: Stakeholder feedback is incorporated into subsequent iterations, allowing for flexibility and adaptability to changing requirements and user needs.
* Risk Mitigation: Breaking the project into smaller increments allows for early issue identification and resolution, reducing overall project risks.
* Enhanced Collaboration: Regular feedback and communication foster collaboration, leading to better teamwork and shared understanding.
* Improved Quality and Testing: Iterative testing ensures higher software quality by identifying and addressing defects early in the development process.
* Efficient Resource Utilization: Teams focus on specific features in each iteration, optimizing resource allocation and utilization.
* Faster Time-to-Market: Delivering usable increments at regular intervals accelerates time-to-market and provides early market advantages.
* Improved Project Transparency: Iterative delivery provides visibility into development progress, promoting transparency and alignment between the team and stakeholders.
  1. **USER CHARACTERISTICS**

This project is mainly divided into two main categories: The Administrators and the  
Visitors/Users.

The store manager and the staff members operate as the administrators or admin. They can add, edit, update products or, delete products thus they able to change the names of products, change prices and add or remove products.

The user can search for products selection, update the cart, remove products from the cart and check out from the shop. The user is also able to update his information such as names, address, and other data.

The visitor is limited to the use of the shop. The visitor can only browse or search for the product.

* 1. **SYSTEM ANALYSIS**

System analysis is the process of planning a new system to either replace or complement an existing system. But before any planning is done the old system must be thoroughly understood and the requirements determined. System analysis is therefore, the process of gathering and interpreting facts, diagnosing problems and using the information to re-comment improvements in the system. If analysis is not done properly then whole project can move in the wrong direction. It also provides a schedule for project work. Analysis task divided into 3 areas:

* Problem Recognition.
* Feasibility Study.
* Requirement Analysis.
  1. **FEASIBILITY STUDY**

Whatever we think need not be feasible. It is wise to think about the feasibility of any problem we undertake. Feasibility is the study of impact, which happens in the organization by the development of a system. The impact can be either positive or negative. When the positives dominate the negatives, then the system is considered feasible. Here the feasibility study can be performed in two ways such as technical feasibility and Economical Feasibility.

**Technical feasibility**

It is technically feasible, since there will not be much difficulty in getting required resources for the development and maintaining the system as well as all the resources needed for the development of the software as well as the maintenance. 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.

**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 must be completed.
* Reporting period

Considering all the above factors it was decided that the allotted time that is 2 months was sufficient to complete the project.

**Economic Feasibility**

Economic Feasibility is about total cost incurred for the system. The software resource requirement of the proposed system is React and MongoDB for functional and backend development and HTML, CSS, JS for the frontend UI.

**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.
* 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.

* 1. **REQUIRENETS SPECIFICATION AND ANALYSIS**

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 analyzed 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 behavior are created. Alternative solutions are analyzed and various project element. Currently who want to buy some shoes or any clothing type they must go to the shop and buy them this is very tedious for customer therefore we upload this site on internet. This website should be developed with an aim to simplify shopping process and keeping transparency and flexibility in performing each operation.

* + 1. **REQUIREMENT GATHERING**

Requirement gathering, also known as requirements elicitation or requirements analysis or Data Collection, is the process of identifying, documenting, and understanding the needs, expectations, and constraints of stakeholders for a specific project or system. It is a critical phase in software development, business analysis, and project management.

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. The ultimate goal is to define and document the requirements accurately and comprehensively.

* + 1. **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 survey-based 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”.

* 1. **REQUIREMENTS**

Requirements are categorized in 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 without login.
* Customer can also add/remove product to cart without login (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.

Admin Module

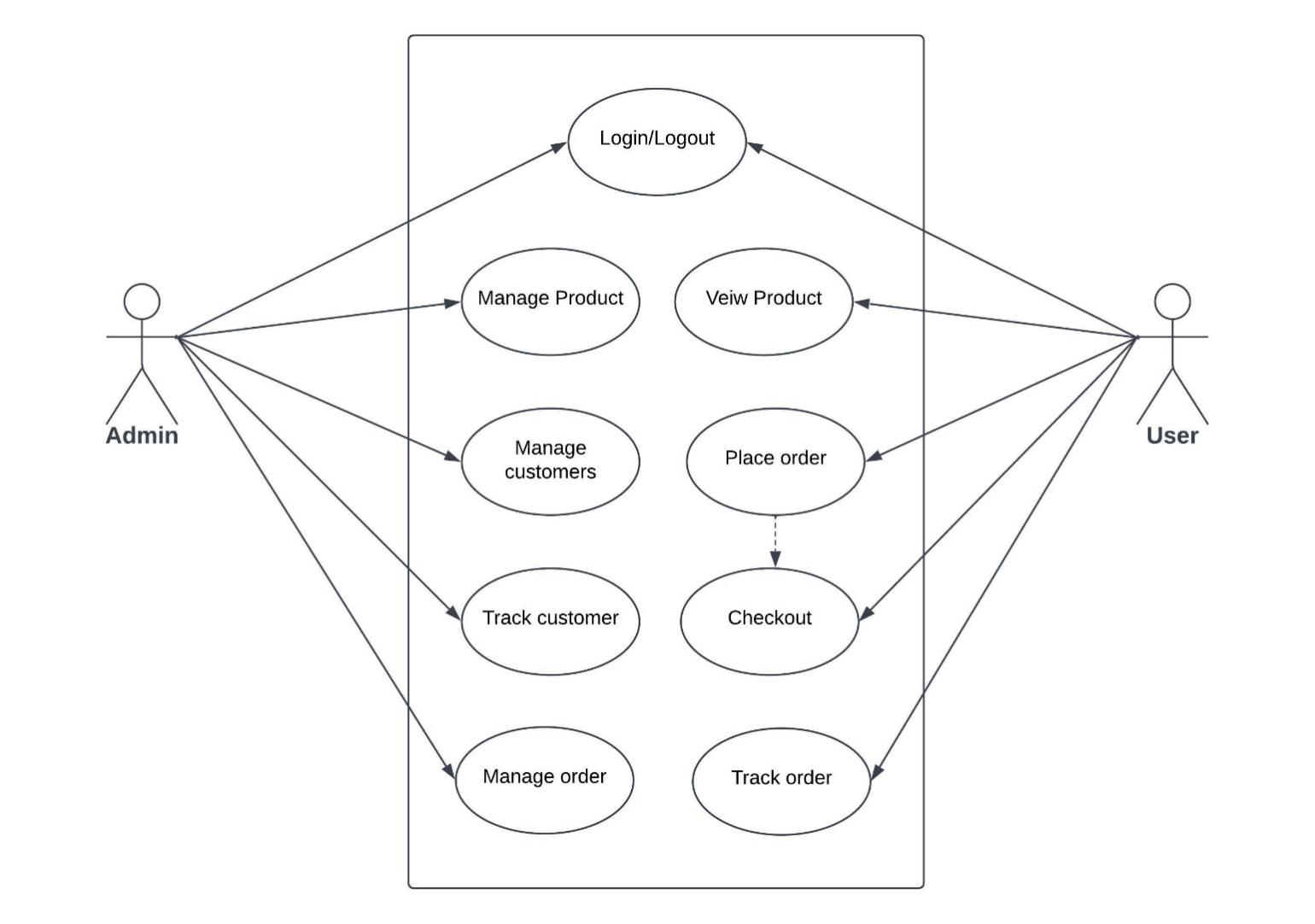
* Admin can provide username, email, password, and your admin account will be created.
* After login, there is a dashboard where admin can see how many customers is registered, how many products are there for sale, how many orders placed.
* Admin can add/delete/view/edit the products.
* Admin can view/edit/delete customer details.
* Admin can view/delete orders.
* Admin can change status of order (order is pending, confirmed, out for delivery, delivered)
* Admin can view the feedbacks sent by customers.

**Non-functional Requirements:**

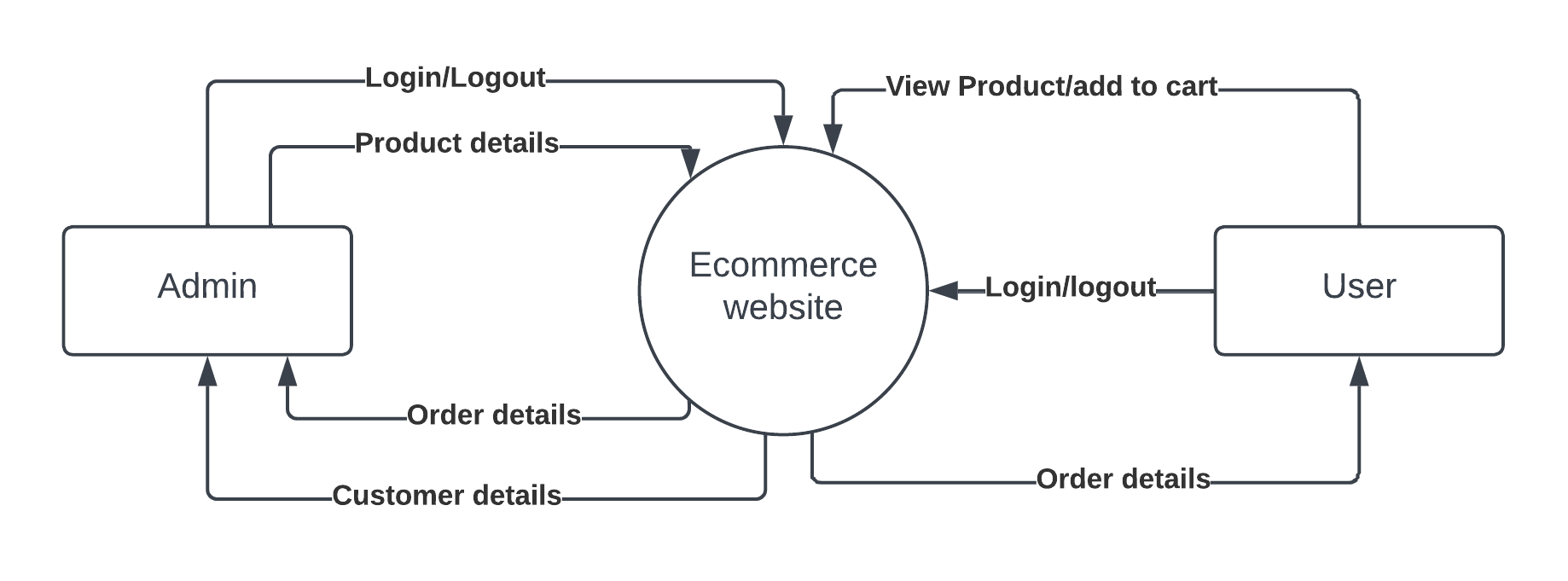
It specifies the quality attribute of a software system. They judge the software system based on Responsiveness, Usability, Security, Portability and other non-functional standards that are critical to the success of the software system.

* 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 multiuser 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.

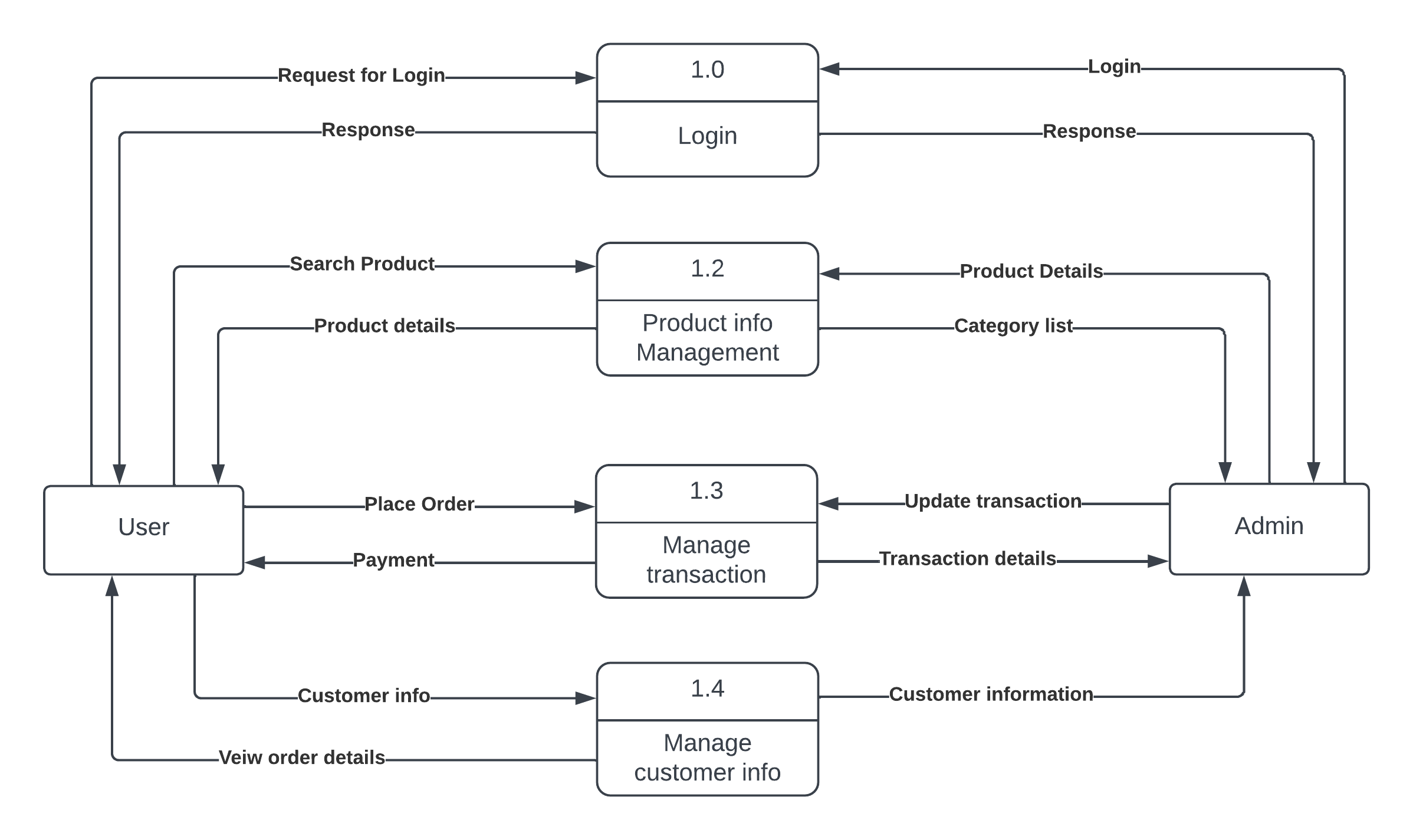
1. **SYSTEM DESIGN**

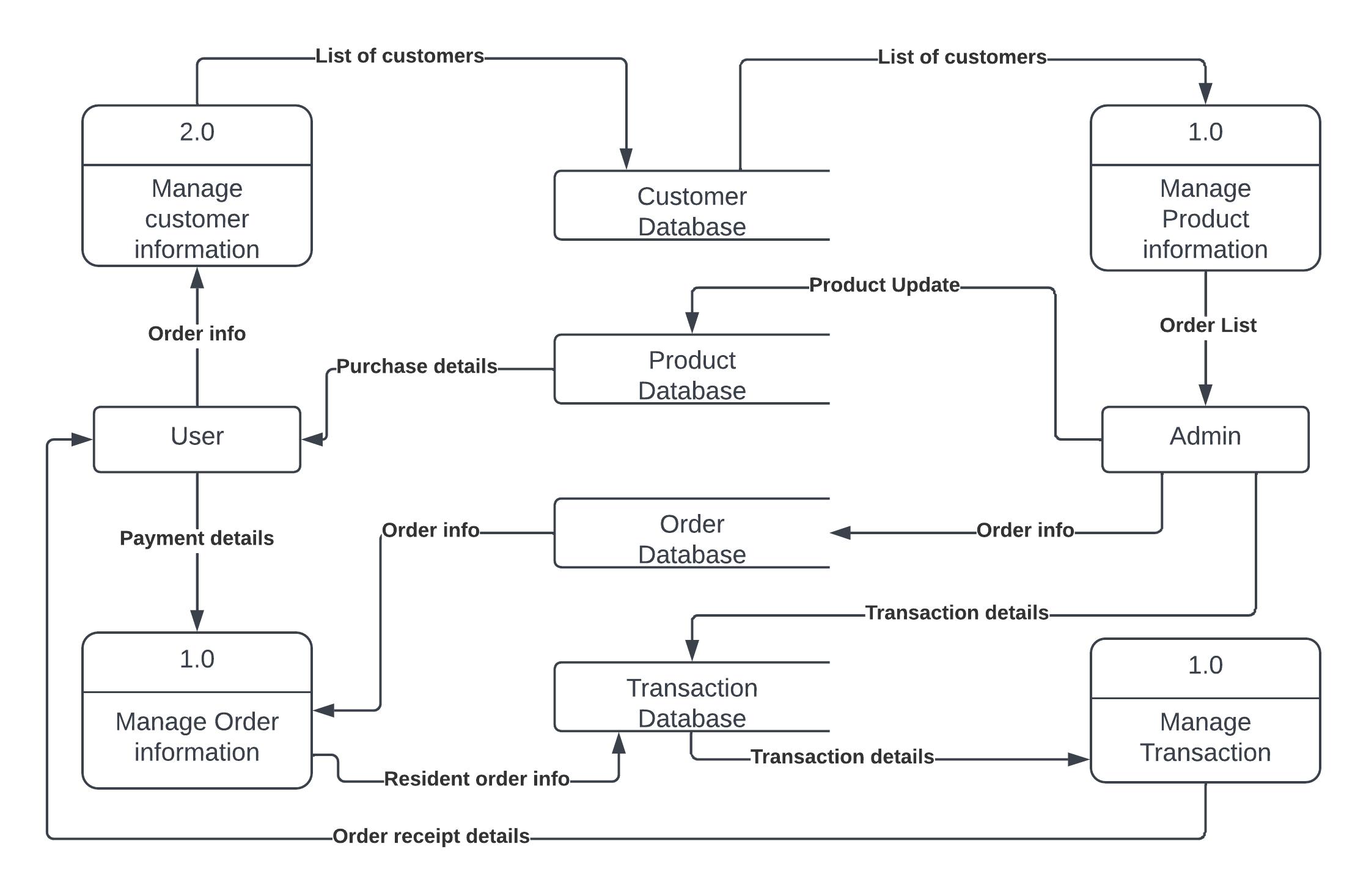
**4.1. USE CASE DIAGRAM** 

**4.3. DATA FLOW DIAGRAM**

**Context Level DFD**

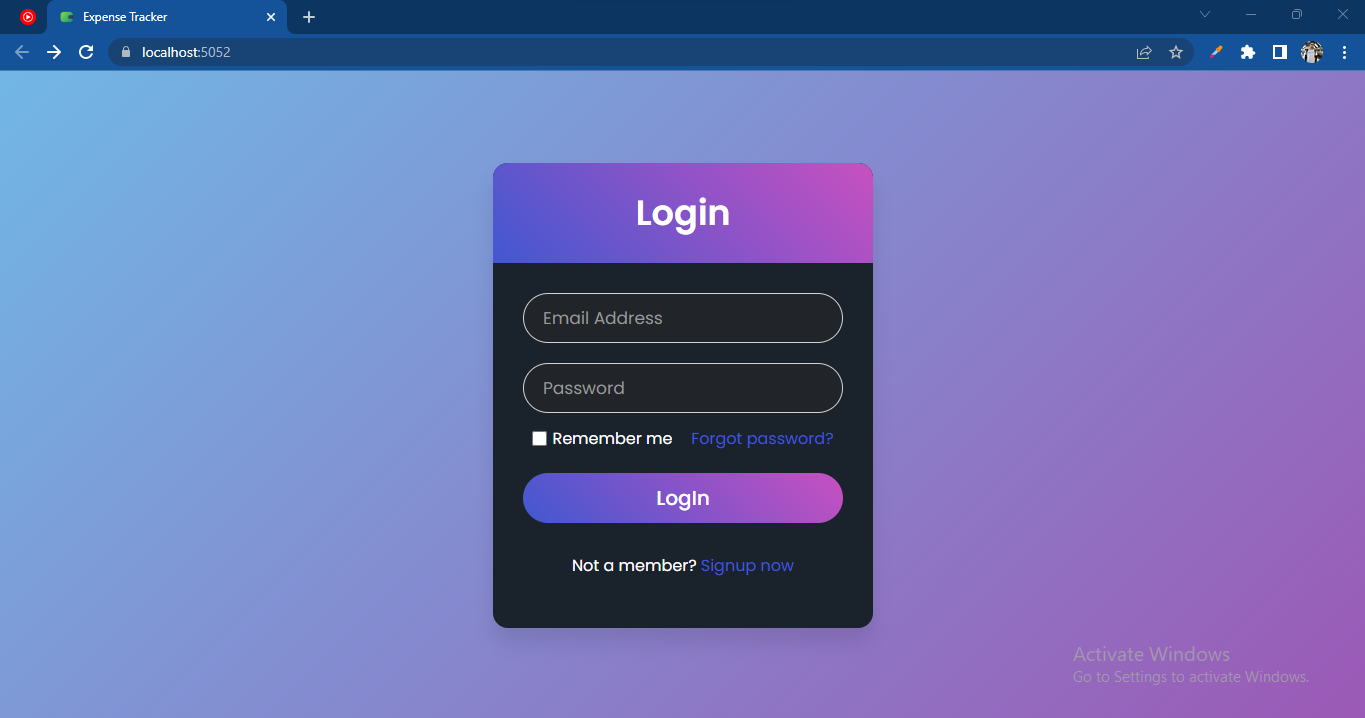
**Level 1 DFD**



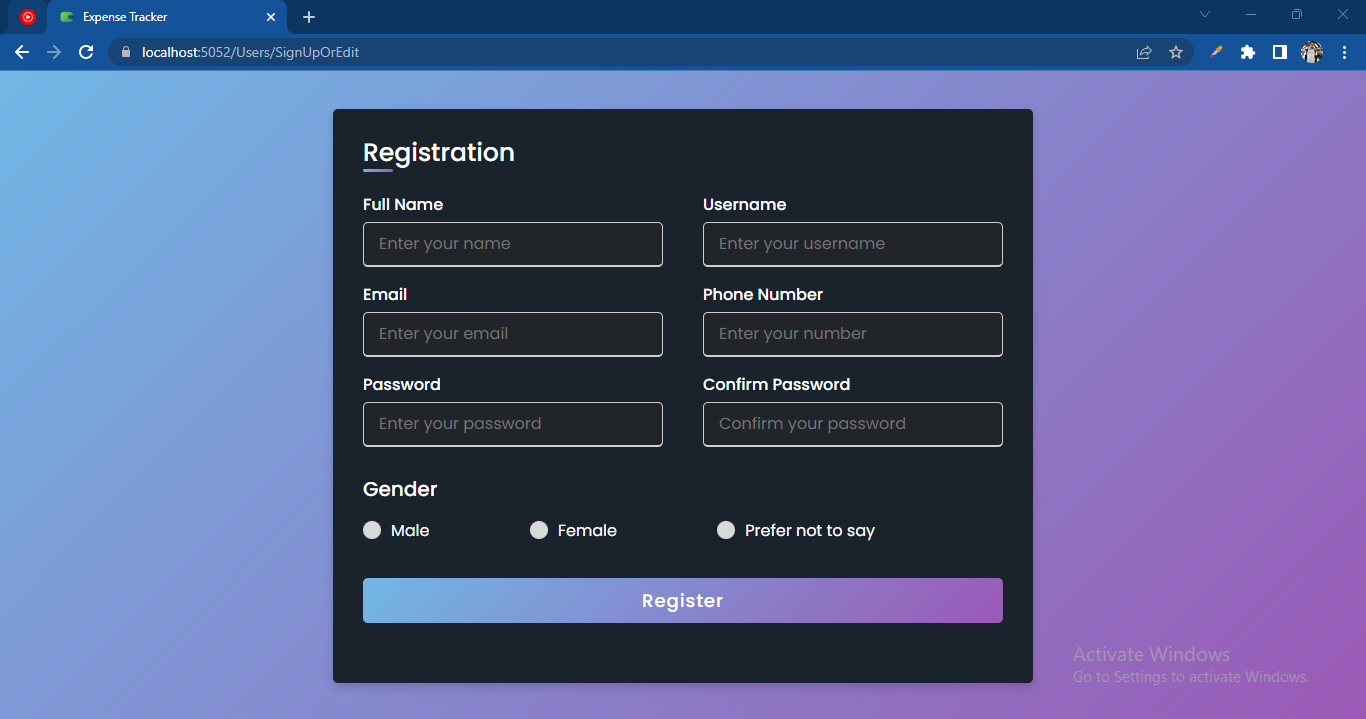
**Level 2 DFD**

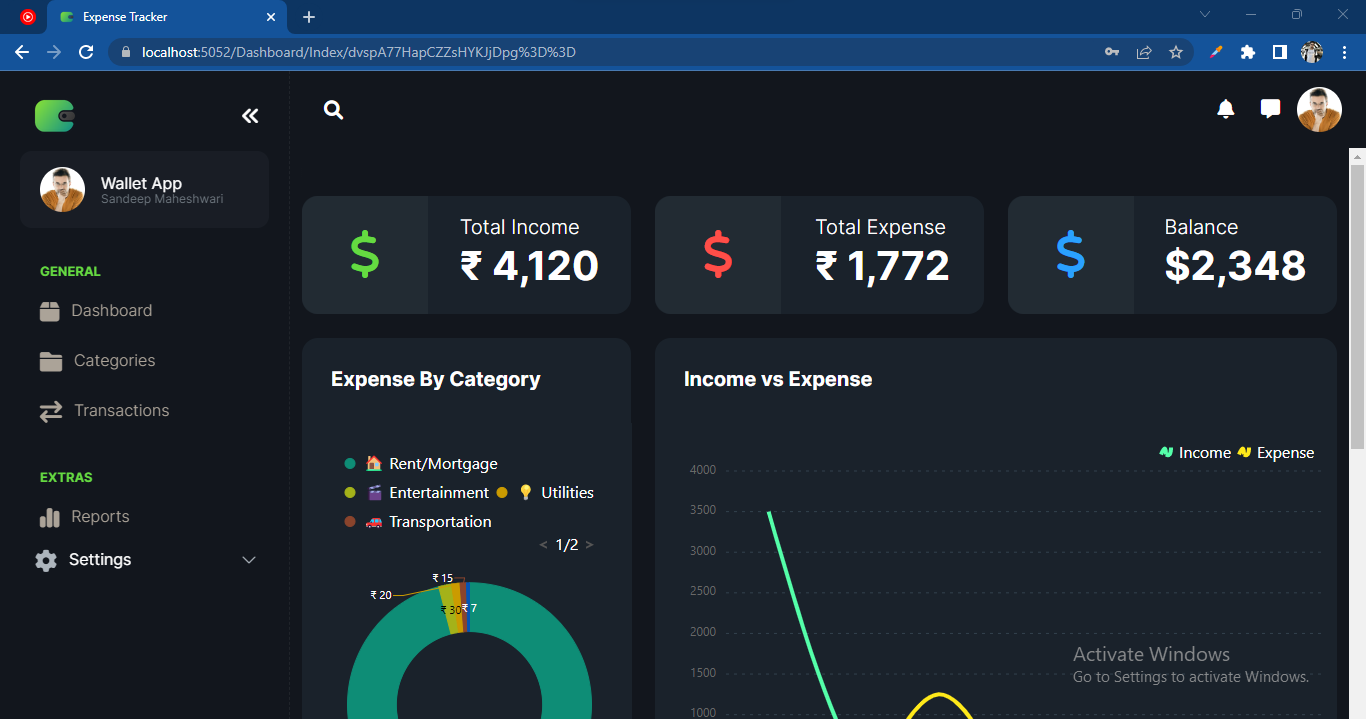
1. **SCREENSHOTS OF PROJECT**

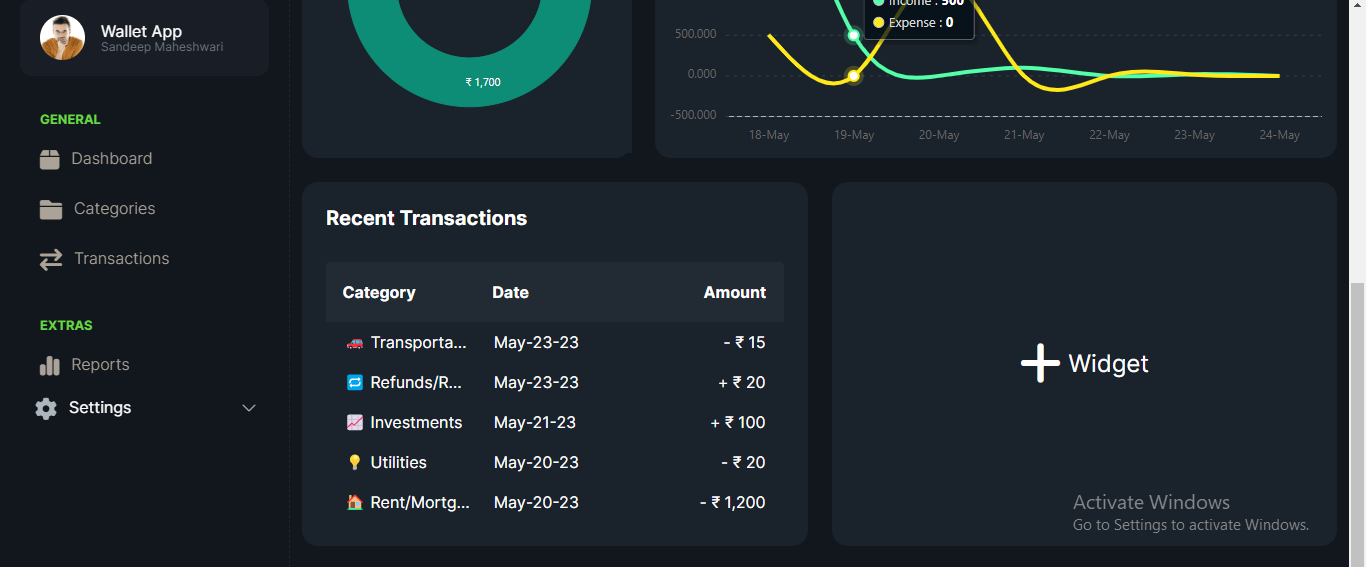
**LOGIN PAGE**



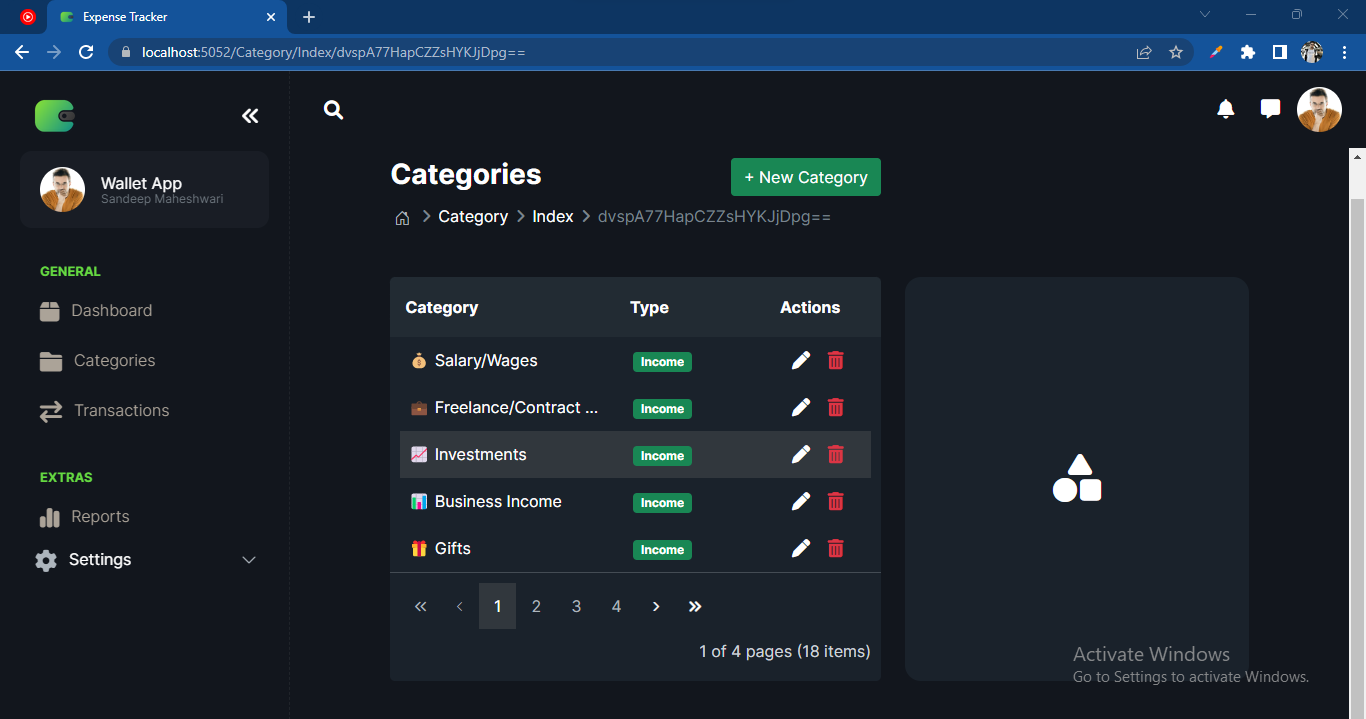
**SIGNUP PAGE**



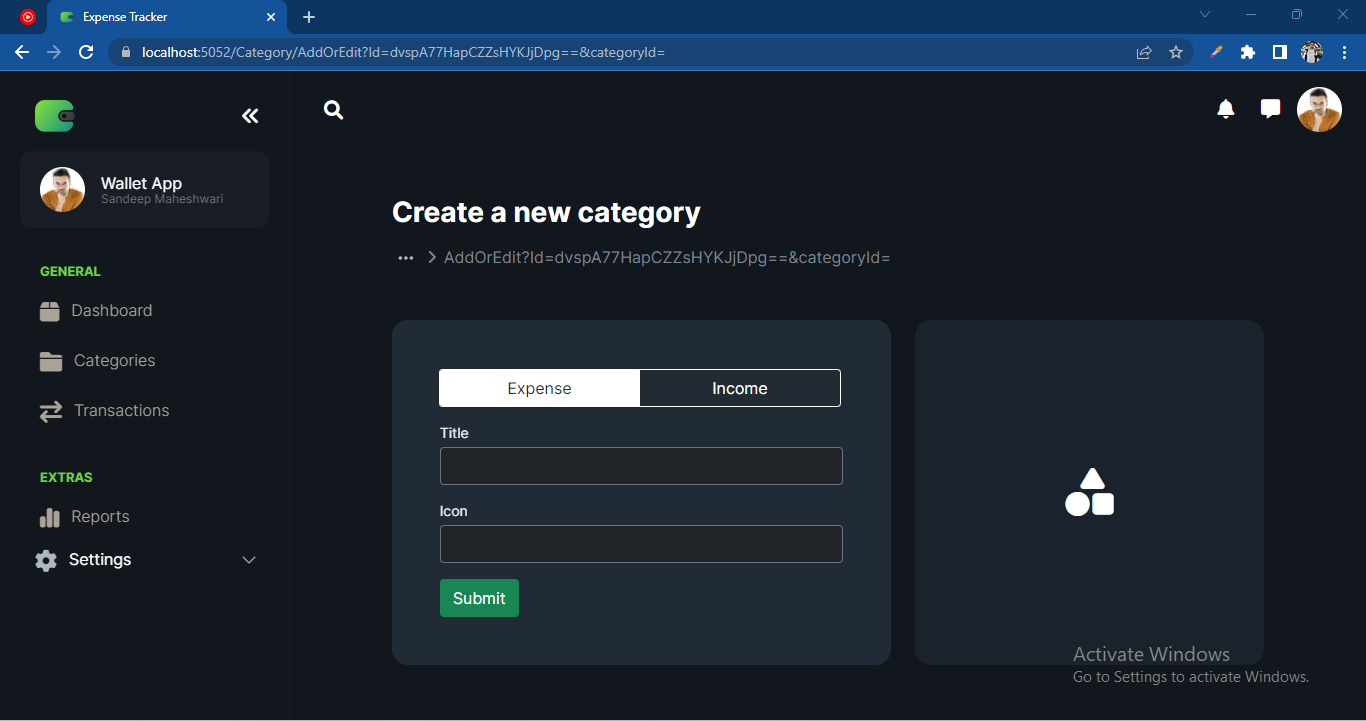
**DASHBOARD**



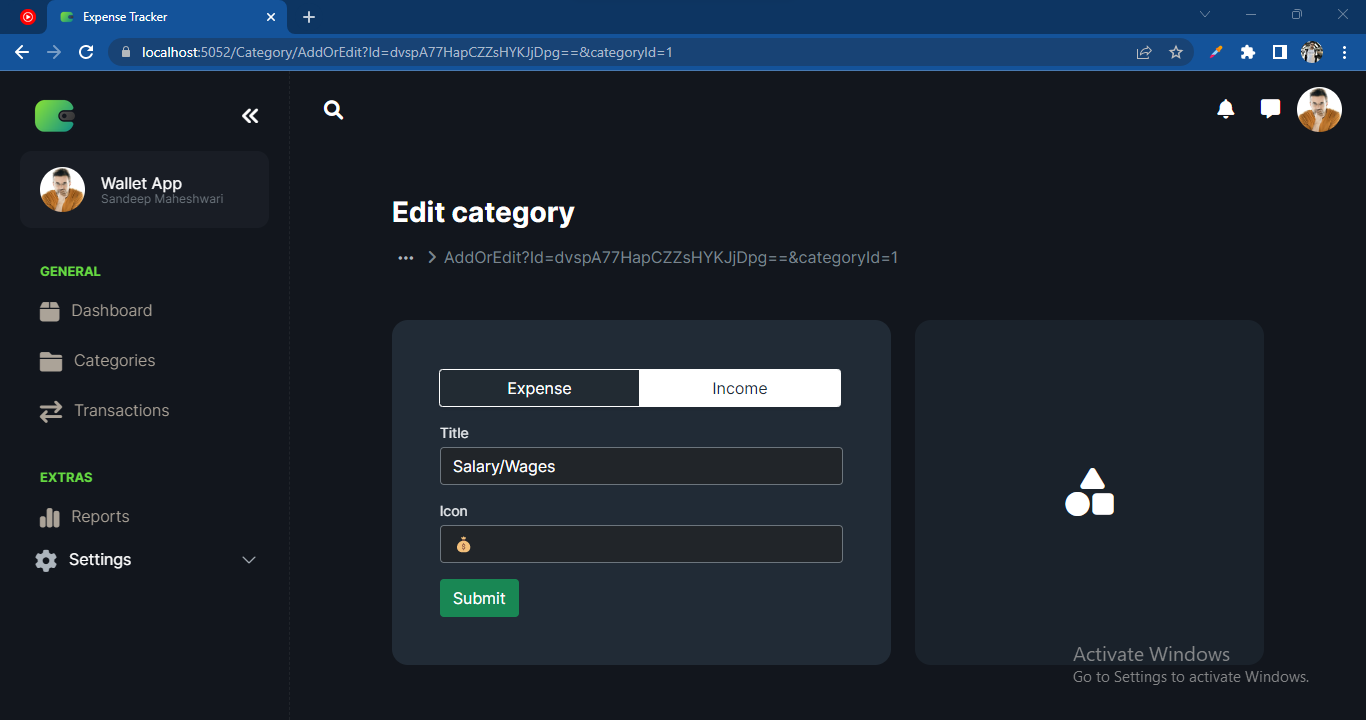
**CATEGORY**



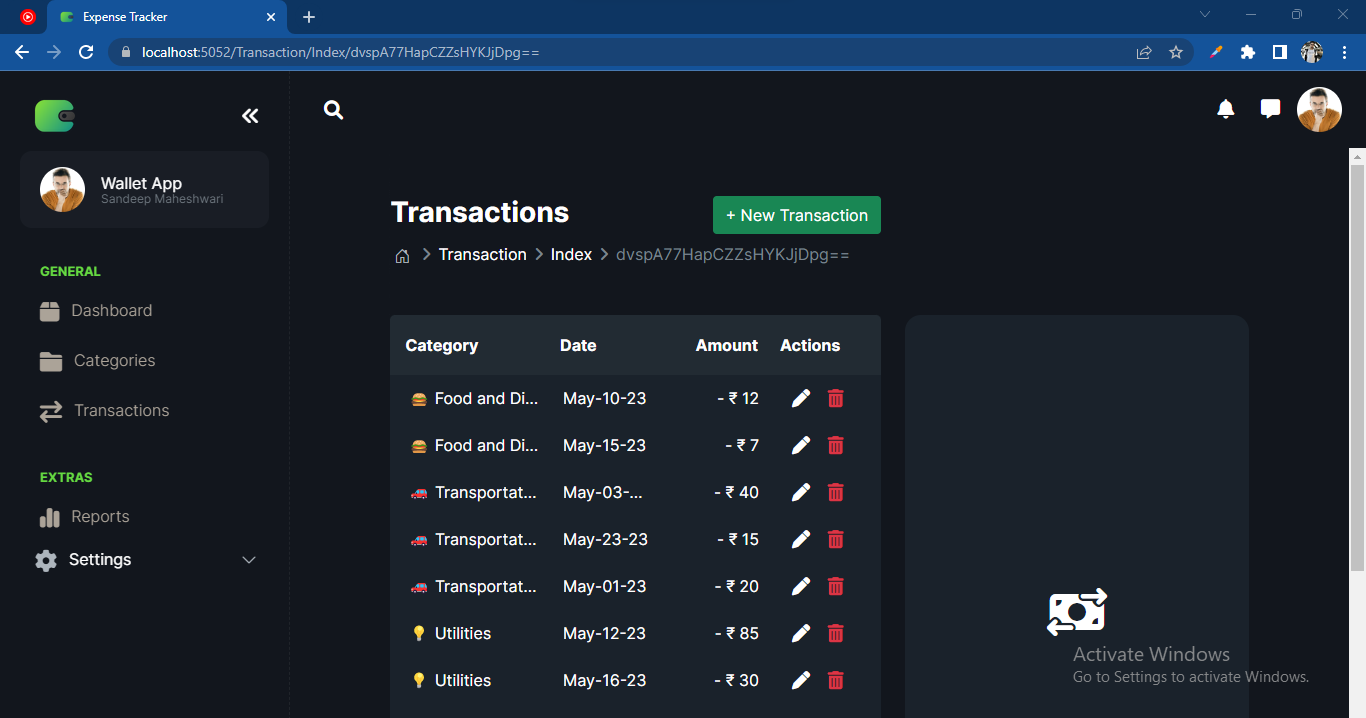
**ADD CATEGORY**

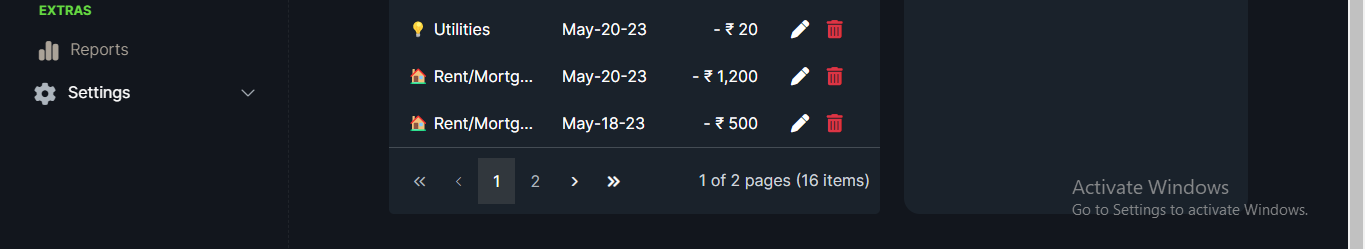
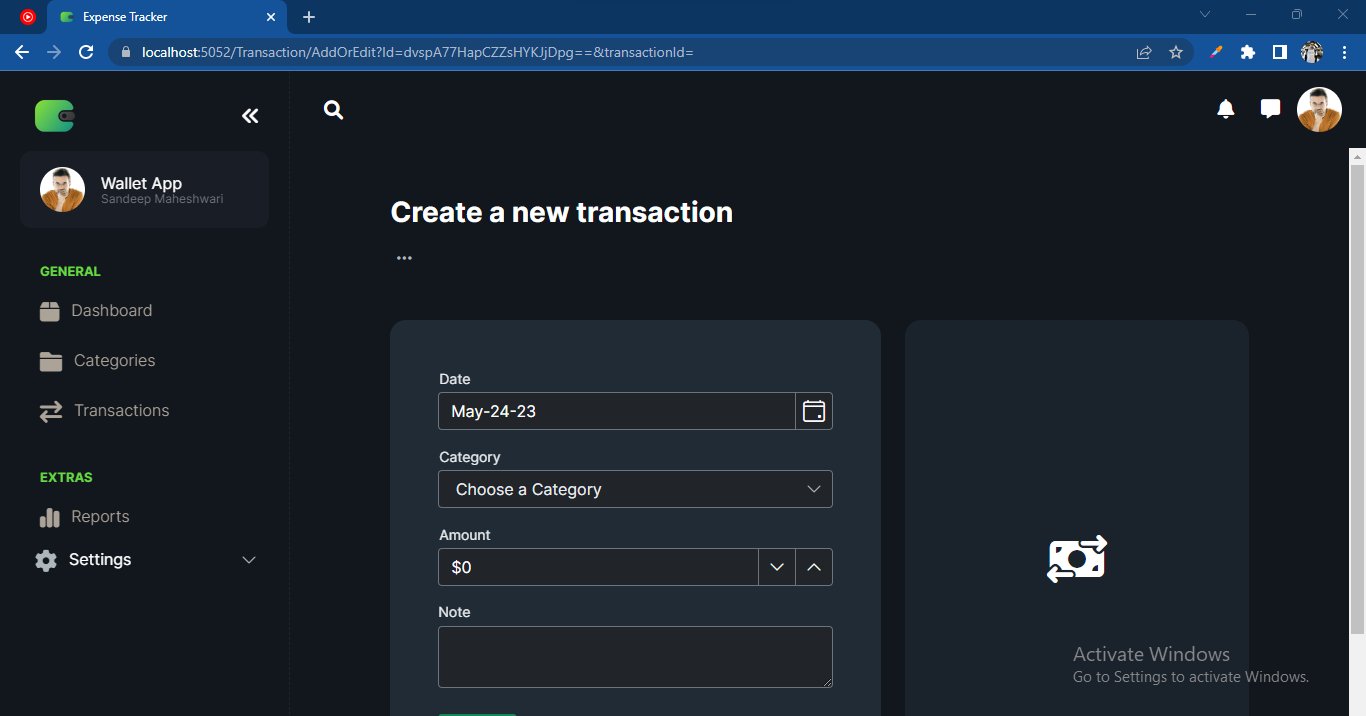


**EDIT CATEGORY**

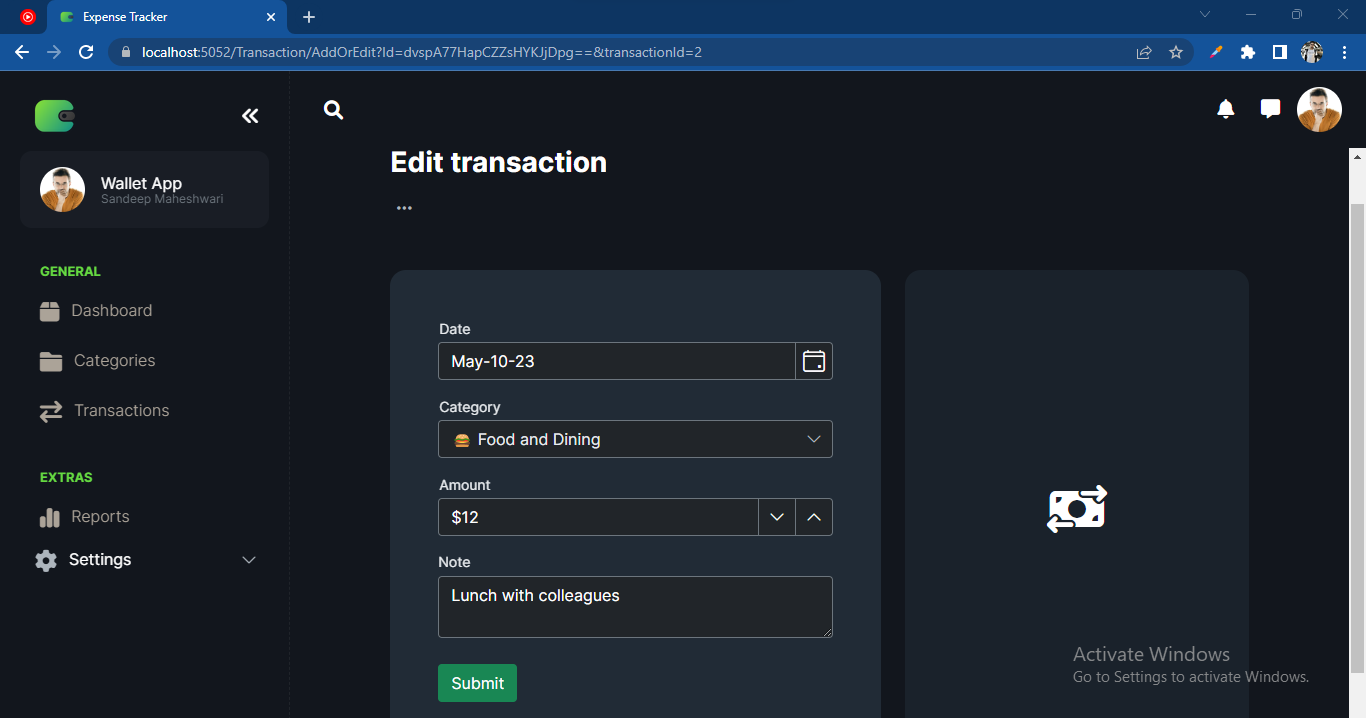


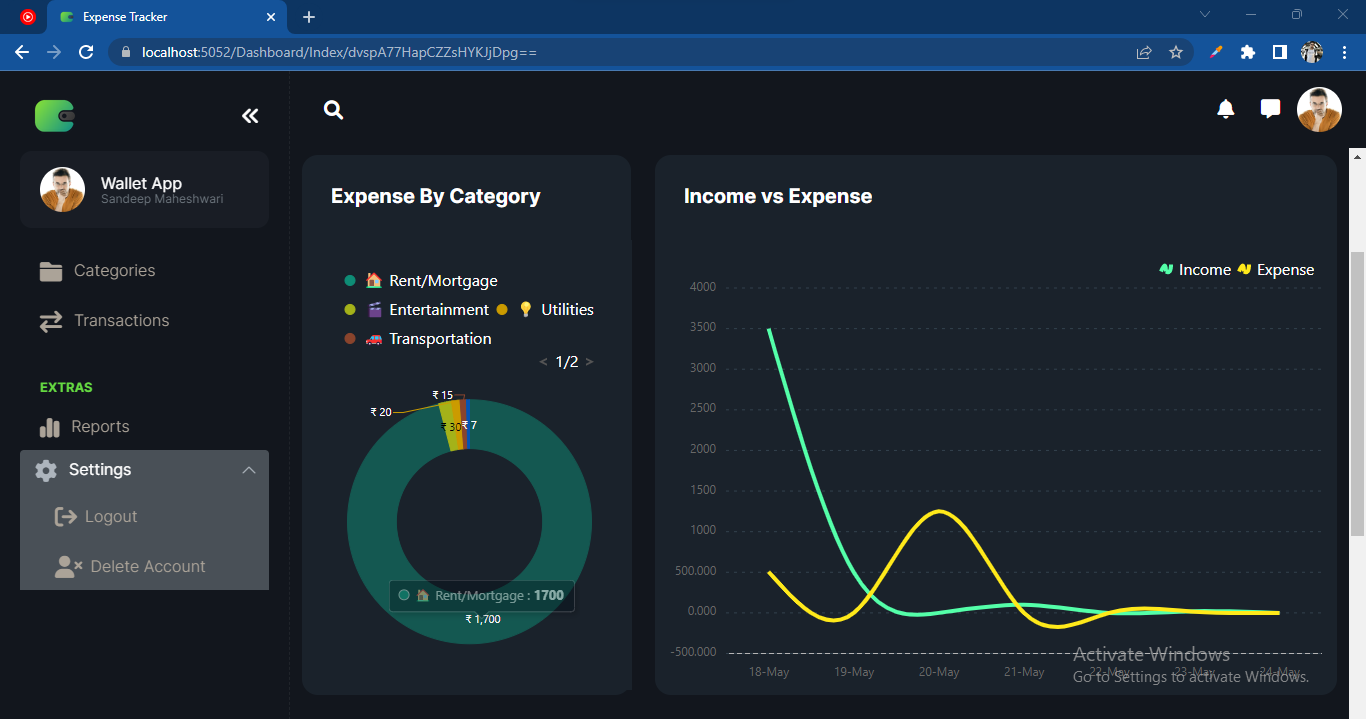
**TRANSACTIONS**



**ADD TRANSACTION**

**EDIT TRANSACTION**



**SETTINGS**

1. **TESTING**

Toolbars work properly. Are all menu function and pull-down sub function properly listed? Is it possible to invoke each menu function using a logical assumption that if all parts of the system are correct, the goal will be successfully achieved? In adequate testing or non-testing will leads to errors that may appear few months later. Testing represents an interesting anomaly for the software engineer. During earlier software engineering activities, the engineer attempts to build software from an abstract concept to a tangible product. Now comes testing. The engineer creates a series of test cases that are intended to “demolish” the software that has been built. In fact, testing is the one step in the software process that could be viewed (psychologically, at least) as destructive rather than constructive. Testing requires that the developer discard preconceived notions of the “correctness” of software just developed and overcome a conflict of interest that occurs when errors are uncovered.

If testing is conducted successfully (according to the objectives stated previously) it will uncover errors in the software. As a secondary benefit, testing demonstrates that software functions appear to be working according to specification, that behavioral and performance requirements appear to have been met. In addition, data collected as testing is conducted provide a good indication of software reliability and some indication of software quality. But testing cannot show the absence of errors and defects, it can show only that software errors and defects are present. It is important to keep this (rather gloomy) statement in mind as testing is being conducted.

**6.1. TESTING STRATEGY**

There are types of testing that we implement. They are as follows:

While deciding on the focus of testing activities, study project priorities. For example, for an on-line system, pay more attention to response time. Spend more time on the features used frequently. Decide on the effort required for testing based on the usage of the system. If the system is to be used by a large number of users, evaluate the impact on users due to a system failure before deciding on the effort.

This create two problem

* Time delay between the cause and appearance of the problem.
* The effect of the system errors on files and records within the system.

The purpose of the system testing is to consider all the likely variations to which it will be suggested and push the systems to limits. The testing process focuses on the logical intervals of the software ensuring that all statements have been tested and on functional interval is conducting tests to uncover errors and ensure that defined input will produce actual results that agree with the required results. Program level testing, modules level testing integrated and carried out.

There are two major types of testing they are:

* White Box Testing.
* Black Box Testing
  1. **WHITE BOX TESTING**

White box sometimes called “Glass box testing” is a test case design uses the control structure of the procedural design to drive test case. Using white box testing methods, the following tests were made on the system.

1. All independent paths within a module have been exercised once. In our system, ensuring that case was selected and executed checked all case structures. The bugs that were prevailing in some part of the code where fixed.
2. All logical decisions were checked for the truth and falsity of the values.
   1. **BLACK BOX TESTING**

Black box testing focuses on the functional requirements of the software. This is black box testing enables the software engineering to derive a set of input conditions that will fully exercise all functional requirements for a program. Black box testing is not an alternative to white box testing rather it is complementary approach that is likely to uncover a different class of errors that white box methods like.

* Interface errors.
* Performance in data structure.
* Performance errors.
* Initializing and termination errors.

1. **FUTURE SCOPE**

The project has a very vast scope in future. The project can be implemented on intranet in future. Project can be updated in near future as and when requirement for the same arises, as it is very flexible in terms of expansion. With the proposed software of database Space Manager ready and fully functional the client is now able to manage and hence run the entire work in a much better, accurate and error free manner. The future scope of an expense tracker web app holds several potential areas for enhancement and innovation. Here are some possible directions for future development:

Automation and Integration - Implement advanced automation features to minimize manual data entry. Explore integrations with financial institutions, banks, credit cards, and receipt scanning services to automatically import and categorize expenses.

Machine Learning and AI - Incorporate machine learning algorithms to analyze spending patterns, detect anomalies, and provide personalized financial recommendations based on user behavior. For example, the app could suggest budget adjustments or identify potential cost-saving opportunities.

Predictive Analytics - Use historical expense data and external factors to predict future spending trends and help users plan their finances proactively. This could include forecasting future expenses, estimating savings targets, or recommending optimal budget allocations.

Smart Budgeting - Enhance budgeting capabilities by allowing users to set dynamic and adaptive budgets that adjust based on income fluctuations or changing financial goals. Implement intelligent notifications and reminders to help users stay within their budgets.

Advanced Reporting and Visualization - Provide advanced reporting and visualization options, including interactive charts, graphs, and customizable dashboards. Allow users to drill down into specific expense categories or time periods for more detailed analysis.

Expense Sharing and Collaboration - Introduce features that enable users to share and split expenses with family members, roommates, or business partners. This could involve shared budgets, expense approvals, or reimbursement tracking.

Mobile App Enhancements - Further optimize the mobile experience by leveraging device capabilities such as camera integration for capturing receipts, voice input for expense entry, or location tracking for automatic expense categorization.

Financial Education and Insights - Offer educational resources, tips, and personalized insights to help users improve their financial literacy and make more informed financial decisions. This could include content on saving strategies, investment opportunities, or debt management.

Gamification and Rewards - Incorporate gamification elements to encourage users to engage with the app and maintain consistent expense tracking. Introduce rewards, badges, or challenges to incentivize positive financial habits and promote user retention.

Integration with Personal Finance Tools Allow users to export data or integrate with popular personal finance management tools such as accounting software, tax filing platforms, or investment management apps.

As technology advances and user expectations evolve, these future enhancements can help expense tracker web apps provide a more seamless, intelligent, and comprehensive financial management experience. Continuously gathering user feedback and staying abreast of emerging technologies will be key to identifying and implementing these future advancements.

The following are the future scope for the project.

* Should be added payment gateway.
* Can be added inventory management system.
* Can be added multiple branches.
* Can be added multilingual to this site.
* And many features can be added this project to make it more robust.

1. **CONCLUSION**

Our project is great improvement over the existing system as it required lots of manual work. This website provides a computerized version of shop manipulate system which will benefit the users as well as the visitor of the shop. It makes entire process online where users can search product and buy various product. It also has a facility for common user by login into the system where user can login and can see status of ordered item as well request for items or give some suggestions. It provides the facility of user login where user can add various categories, review users’ activity and also add info about different products for the customer.

In depth we noticed that this project could help us a lot on the below point as well like In conclusion, an expense tracker web app is a valuable tool for individuals and businesses to manage and track their expenses effectively. It provides a centralized platform to record, categorize, and analyze expenses, enabling users to gain insights into their spending habits and make informed financial decisions. While expense tracker web apps offer numerous benefits, they also have limitations such as manual data entry, dependency on user discipline, limited integration, security concerns, and platform dependencies.

By understanding these limitations and addressing them through user education, intuitive interfaces, robust security measures, and continuous improvement, developers can create expense tracker web apps that provide a reliable and user-friendly experience. By leveraging the app's features such as expense entry, categorization, budgeting, reporting, and notifications, users can take control of their finances, track their progress, and work towards their financial goals. Whether for personal or business use, an expense tracker web app can be a valuable tool for financial management and provide users with insights that promote better financial health.

1. **BIBLIOGRAPHY**

**<https://www.w3schools.com/html/>**

**<https://css-tricks.com/>**

**<https://www.guru99.com/asp-net-tutorial.html>**

**<https://www.tutorialspoint.com/ms_sql_server/index.htm>**

**<https://www.codeproject.com/Learn-ASP-NET-MVC-Step-by-Step-Part>**

**<https://www.javatpoint.com/client-side-scripting>**