# ****Expanse Tracker using ASP.Net Core MVC****

## ****Table of Contents****

1. **Introduction**
   1. 1.1 Purpose
   2. 1.2 Scope
   3. 1.3 Definitions, Acronyms, and Abbreviations
   4. 1.4 References
2. **Overall Description**
   1. 2.1 Product Perspective
   2. 2.2 Product Features
   3. 2.3 User Classes and Characteristics
   4. 2.4 Operating Environment
   5. 2.5 Constraints
   6. 2.6 Assumptions and Dependencies
3. **System Features**
   1. 3.1 Feature 1: Category Management
   2. 3.2 Feature 2: Transaction Management
   3. 3.3 Feature 3: Dashboard and Reports
4. **External Interface Requirements**
   1. 4.1 User Interfaces
   2. 4.2 Hardware Interfaces
   3. 4.3 Software Interfaces
5. **System Attributes**
   1. 5.1 Performance Requirements
   2. 5.2 Security Requirements
   3. 5.3 Availability Requirements
   4. 5.4 Maintainability Requirements
   5. 5.5 Portability Requirements
6. **Other Non-Functional Requirements**
   1. 6.1 Legal and Regulatory Requirements
   2. 6.2 Backup and Recovery
7. **Appendices**

## ****1. Introduction****

### ****1.1 Purpose****

The purpose of the **Expense Tracker** application is to help users track their daily expenses and organize them into categories. The application allows users to add transactions, manage categories, and view their expense history in an easy-to-use interface.

### ****1.2 Scope****

This software is intended for use by individuals or small businesses who want to manage their expenses. Users will be able to:

* Create categories for different types of expenses (e.g., Food, Entertainment).
* Record transactions, including the amount, category, and date.
* View the transactions by category and generate simple reports.

### ****1.3 Definitions, Acronyms, and Abbreviations****

* **Transaction:** A record of an expense including the amount, date, and associated category.
* **Category:** A classification for different types of expenses.
* **Syncfusion:** A third-party UI component library used for the application’s interface.
* **EF Core (Entity Framework Core):** A tool for database management in .NET.

### ****1.4 References****

* [Syncfusion Documentation](https://www.syncfusion.com)
* [ASP.NET Core MVC Documentation](https://docs.microsoft.com/en-us/aspnet/core/mvc/)

## ****2. Overall Description****

### ****2.1 Product Perspective****

The **Expense Tracker** is a standalone web application built using the **ASP.NET Core MVC** framework. It uses **Syncfusion** for its UI components and **SQL Server** as the database.

### ****2.2 Product Features****

* **Category Management:** Users can create and manage categories for their expenses (e.g., Food, Rent).
* **Transaction Management:** Users can record transactions and link them to specific categories.
* **Dashboard:** A summary of recent transactions and the total amount spent, categorized by type.
* **Reports (optional feature):** Users can view detailed reports of their spending patterns over time.

### ****2.3 User Classes and Characteristics****

* **Admin:** Can add/edit categories and view all transactions.
* **User:** Can add/edit their own transactions and view their personal expense history.

### ****2.4 Operating Environment****

* **Server-side:** ASP.NET Core MVC application hosted on a web server (Windows/Linux).
* **Client-side:** Web browser (e.g., Chrome, Firefox).
* **Database:** SQL Server used for data storage.

### ****2.5 Constraints****

* The application must run smoothly on most modern web browsers.
* The database is hosted on **SQL Server**, so SQL Server must be available.

### ****2.6 Assumptions and Dependencies****

* The application depends on **Syncfusion UI components** for the sidebar and other elements.
* The database depends on **Entity Framework Core** for communication with SQL Server.

## ****3. System Features****

### ****3.1 Feature 1: Category Management****

* **Description:** Users can create, view, update, and delete expense categories.
* **Inputs:** Category name, icon, type.
* **Outputs:** Success/failure message upon saving a category.
* **Pre-condition:** User must be logged in.
* **Post-condition:** New category is saved and appears in the category list.

### ****3.2 Feature 2: Transaction Management****

* **Description:** Users can record transactions, including the amount, date, and the category.
* **Inputs:** Amount, date, category selection.
* **Outputs:** Success/failure message upon saving a transaction.
* **Pre-condition:** User must have at least one category.
* **Post-condition:** Transaction is saved to the database and linked to a category.

### ****3.3 Feature 3: Dashboard and Reports****

* **Description:** Users can view their transactions on a dashboard, with reports summarizing total spending by category.
* **Inputs:** None (automatic display based on saved data).
* **Outputs:** A graphical representation of total expenses by category.

## ****4. External Interface Requirements****

### ****4.1 User Interfaces****

* **Login Page:** Allows users to log in.
* **Dashboard:** Displays the total spending and recent transactions.
* **Category Management Page:** Allows users to create and edit categories.
* **Transaction Page:** Allows users to record and view their transactions.

### ****4.2 Hardware Interfaces****

There are no specific hardware interfaces required for this application. It is a web-based solution that works on standard computers and servers.

### ****4.3 Software Interfaces****

* **Database:** The application uses **SQL Server** to store categories and transactions.
* **UI Components:** The application uses **Syncfusion** for the sidebar and other UI elements.

## ****5. System Attributes****

### ****5.1 Performance Requirements****

The application should be able to handle up to **10,000** transactions without noticeable performance degradation. It should load pages in under **2 seconds**.

### ****5.2 Security Requirements****

* **Authentication:** Users must log in to access the application.
* **Authorization:** Only users with appropriate roles can manage categories and view all transactions.
* **Data Protection:** Sensitive information like transaction details should be encrypted where necessary.

### ****5.3 Availability Requirements****

The application should be available **24/7** with a minimum downtime for maintenance. An uptime of **99%** is required.

### ****5.4 Maintainability Requirements****

The application should be easy to update, with clear separation between the **UI**, **backend**, and **database** components.

### ****5.5 Portability Requirements****

The application should be deployable on Windows and Linux servers. It should be compatible with modern web browsers.

## ****6. Other Non-Functional Requirements****

### ****6.1 Legal and Regulatory Requirements****

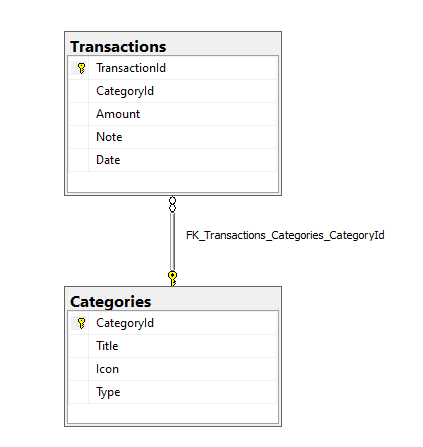
The application must comply with relevant data protection regulations (e.g., GDPR) if users' personal information is stored.

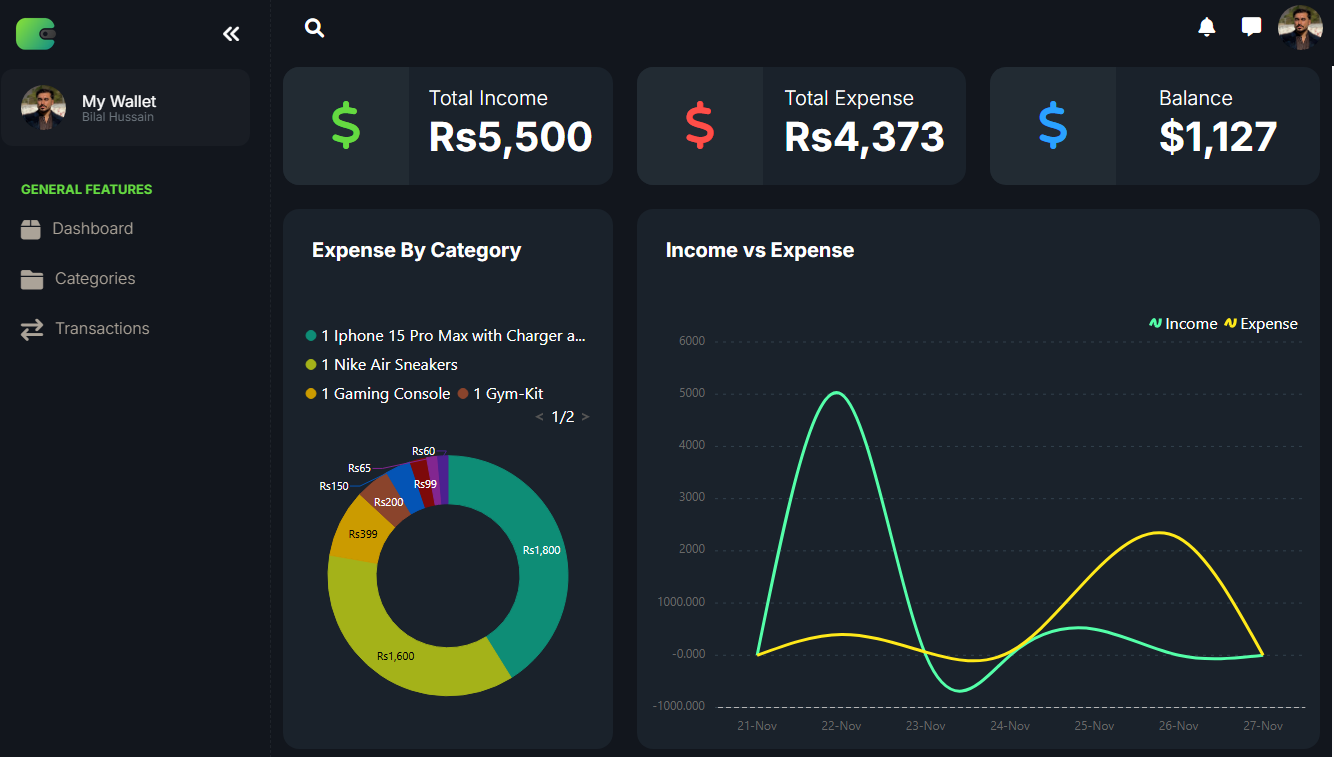
### ****6.2 Backup and Recovery****

The application should back up user data regularly (e.g., daily). In case of a failure, it should be possible to recover the data to the last backup point.

## ****7. Appendices****

* **Database Schema Diagram:**



* **UI Mockups:**