4/8/2019

Niklesh Shakya

Expense Tracker

Daily expense tracking system

Table of Contents

[**1** **Introduction** 2](#_Toc5573583)

[**2** **Software Requirement Specification** 3](#_Toc5573584)

[**2.1** **Functional Requirement** 3](#_Toc5573585)

[**2.2** **Non- Functional Requirement** 4](#_Toc5573586)

[**2.2.1** **Performance Requirements:** 4](#_Toc5573591)

[**2.2.2** **Security Requirement:** 4](#_Toc5573592)

[**2.2.3** **User Requirement** 4](#_Toc5573593)

[**2.2.4** **Environment Requirement** 4](#_Toc5573594)

[**2.3** **Resource Requirement** 5](#_Toc5573595)

[**2.3.1** **Hardware** 5](#_Toc5573596)

[**2.3.2** **Software** 5](#_Toc5573597)

[**3** **System** **Design** 6](#_Toc5573598)

[**3.1** **Architectural Design** 6](#_Toc5573599)

[**3.2** **Data Dictionary** 7](#_Toc5573600)

[**3.3** **User Interface Design** 10](#_Toc5573601)

[**3.3.1** **User Register** 10](#_Toc5573602)

[**3.3.2** **User Login** 10](#_Toc5573603)

[**3.3.3** **Dashboard** 11](#_Toc5573604)

[**3.3.4** **Expense Category** 12](#_Toc5573605)

[**3.3.5** **Expense Category Create** 12](#_Toc5573606)

[**3.3.6** **Daily Expense** 13](#_Toc5573607)

[**3.3.7** **Daily Expense Create** 13](#_Toc5573608)

[**3.3.8** **Expense Report** 14](#_Toc5573609)

[**3.3.9** **Expense Limit Update** 14](#_Toc5573610)

[**3.3.10** **Change Password** 15](#_Toc5573611)

[**4** **Deployment Guide** 16](#_Toc5573612)

[**4.1** **System Requirements** 16](#_Toc5573613)

[**4.2** **Provided Materials** 16](#_Toc5573614)

[**4.3** **Steps to setup** 16](#_Toc5573615)

# **Introduction**

Expense Tracker is an N-tier web based application developed by Niklesh Shakya. This application has been developed to fulfill the requirement of Dynamic software Architecture Course. This application has many limitation which can be overcome in future.

The main purpose of the system is to track all the daily expenses made by the user. The user can generate report based on his expenses on the transaction date of the expenses or category of the item.

This application is developed in .Net platform, using C# MVC. And for database MSSQL has been used.

## **Software Requirement Specification**

## **Functional Requirement**

Functional Requirements is termed as the requirement that the system must do. These give the services of the system. It also states how the system should react to particular inputs and how the system should have behave in particular situation.

Functional requirements of Expense Tracker are:

* Register a new User for a new Account.

Users need to create a new account before they can keep record of their expenses.

* Login User

System has to make sure that the user is the user that have the authorization to access the system, so the user must be authenticated by login process.

* Create Item Category

After authorization, User needs to create Category before they can start adding expenses. Users can have their own item categories for items that they entered.

* Create Daily Expenses

User can add item name, date of transaction, amount, category etc. in the daily expense.

* Create Expense Limit

User can add a limit to the amount of money they want to expend. The expense limit is used notify the user if they have expend more amount than they desired.

* View Reports

Reports can be generated based on the transaction date, category or both.

## **Non- Functional Requirement**

Non-functional requirement are the requirements that are not directly concerned to the specified function delivered by the system. They must relate to the emergent system properties such as reliability, response time and store occupancy. Non-functional requirements are sometimes known as quality attribute which are important in addition to obvious features and functions.



### **Performance Requirements:**

Our system will be used by the users to track their daily expenses.

* Separating the Database layer with the Business layer and application layer.

The architecture of the system needs to be design with clear separation between application layer, business layer and database layer.

* The system is expected to handle various errors in different ways that prevent the loss of the information.
* The system is expected to handle the large amount of data.

### **Security Requirement:**

* User Authentication

To ensure the security of the user, the system needs to authenticate on every user before the user can access the system. The way to do this is to make sure that the user needs to login first. The user information will be saved in the session. The session will be cleared every time the user’s logouts of the system.

* Encrypting password in the database

For protecting the user’s account from outside or from the system’s administrator itself, the system encrypt the password when storing into the database. So even the administrator of the system cannot open and retrieve the data on the database, password information of the user is in encrypted format and cannot be stolen.

### **User Requirement**

The user of our system should have the basic knowledge of the computers and internet browsing.

### **Environment Requirement**

Our application will run in any standard web browser with internet connection.

## **Resource Requirement**

### **Hardware**

**System:** Laptop/desktop with internet connection.

* **Server Side Requirement**

Processor – 2.0GHz

Ram – 4 GB or above

Hard Disk – 4 GB (free space)

* **Client Side Requirement**

Processor – 1.0GHZ

RAM – 1GB or above

### **Software**

The application is developed using C# MVC (Entity framework).

Front End: Bootstrap 3.3.7/html/CSS/JavaScript

Back End: C# MVC (Entity Framework), MSSQL

# **System** **Design**

System design is the process of defining the architecture, components, modules, interfaces, and data for a system to satisfy specified requirements.

## **Architectural Design**

Our system follow N tier architecture which is mostly composed of presentation tier, logic tier, and a data tier developed and maintained as independent modules. The N-tier model is a software architecture pattern

Business Logic Layer

Data Access Layer

Application Layer

## **Data Dictionary**

* Table name: Users

Description: To keep the detail of user registration

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| S.N | Fieldname | Datatype | Constraint | Description |
| 1 | Id | Int | Primary key  Not Null  Auto Increment | Store auto incremented id |
| 2 | Name | Nvarchar(max) | Not null | Store full name |
| 3 | Email | Nvarchar(max) | Not null | Store Email |
| 4 | Address | Nvarchar(max) | Null | Store Address |
| 5 | Occupation | Nvarchar(max) | Null | Store occupation |
| 6 | Password | Nvarchar(max) | Not null | Store password |
| 7 | Status | Bit | Not null | Store status |
| 8 | CreatedAt | DateTime | Not null | Store Created At date |
| 9 | RoleId | Int | Foreign Key  Int  Not null | To store the foreign key of role |

* Table Name: Roles

Description: To keep the detail of the roles

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| S.N | Fieldname | Datatype | Constraint | Description |
| 1 | Id | Int | Not null  Primary Key  Autoincrement | Store auto incremented Id |
| 2 | RoleName | Nvarchar(max) | Not null | Store role name |
| 3 | Remarks | Nvarchar(max) | Null | Store remarks |
| Id | CreatedAt | Datetime | Not null | Store created at date |

* Table Name: Categories

Description: To keep the detail of category

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| S.N | Fieldname | Datatype | Constraint | Description |
| 1 | Id | Int | Primary  Not null  Auto Increment | Store auto incremented Id |
| 2 | Name | Nvarchar(50) | Not null | Store name of the category |
| 3 | Remarks | Nvarchar(150) | Null | Store remarks |
| 4 | Status | Int | Not null | Store status |
| 5 | CreatedBy | Int | Foreign Key  Int  Not null | Store user id |
| 6 | CreatedAt | Datetime | Not null | Store created at date |
| 7 | Modified At | Datetime | Null | Stored modified at dat |

* Table Name: Expenses

Description: To keep the detail of the expenses made by the user

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| S.N | Fieldname | Datatype | Constraint | Description |
| 1 | Id | Int | Primary  Not null  Auto Increment | Store auto incremented Id |
| 2 | Date | Datetime | Not null | Store expense date |
| 3 | Amount | Decimal(18, 2) | Not null | Store expense amount |
| 4 | Item | Nvarchar(100) | Not null | Store expense item |
| 5 | Description | Nvarchar(150) | Null | Store description |
| 6 | CreatedBy | Int | Not null | Stored user id |
| 7 | CategoryId | Int | Foreign Key  Int  Not null | Store category id |
| 8 | ModifiedAt | Datetime | Null | Store modified at date |

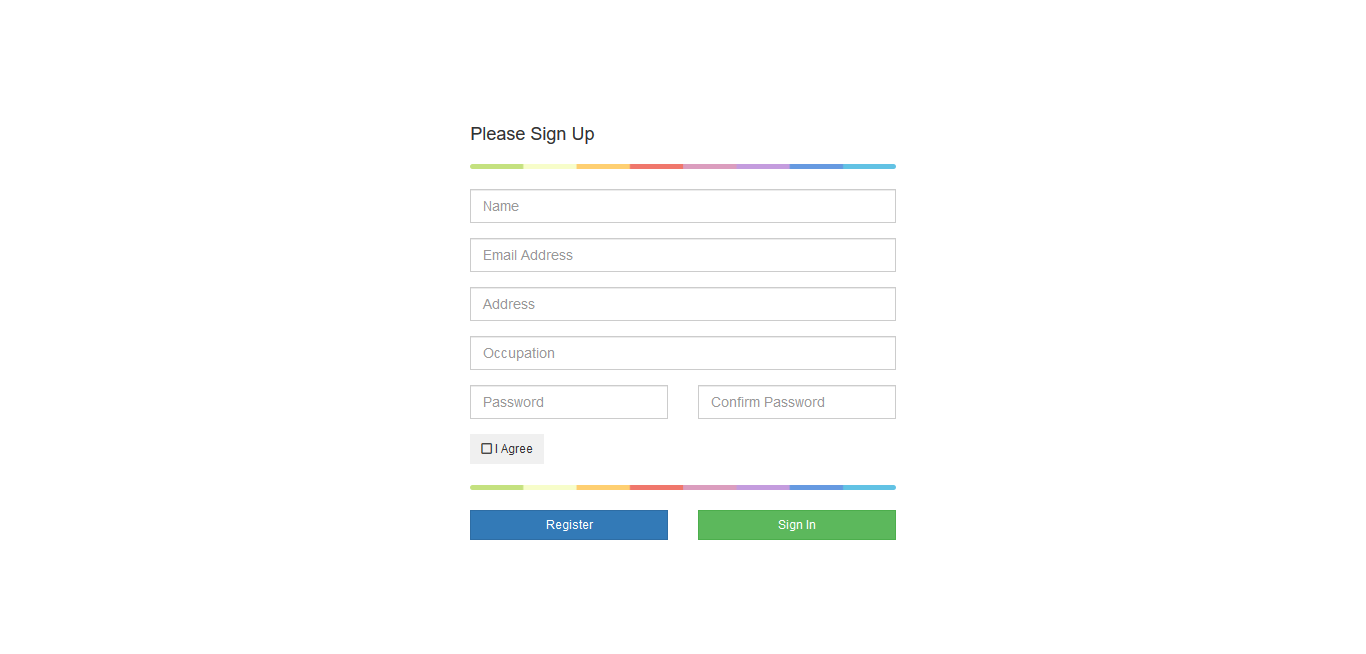
* Table Name: Expense Limit

Description: To keep the detail of the Expense Limit made by the user

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| S.N | Fieldname | Datatype | Constraint | Description |
| 1 | Id | Int | Primary  Not null  Auto Increment | Store auto incremented Id |
| 2 | LimitAmount | Decimal(18, 2) | Not null | Store limit amount |
| 3 | CreatedBy | Int | Not null | Stored user id |
| 4 | Status | Bit | Null | Store expense limit status |

## **User Interface Design**

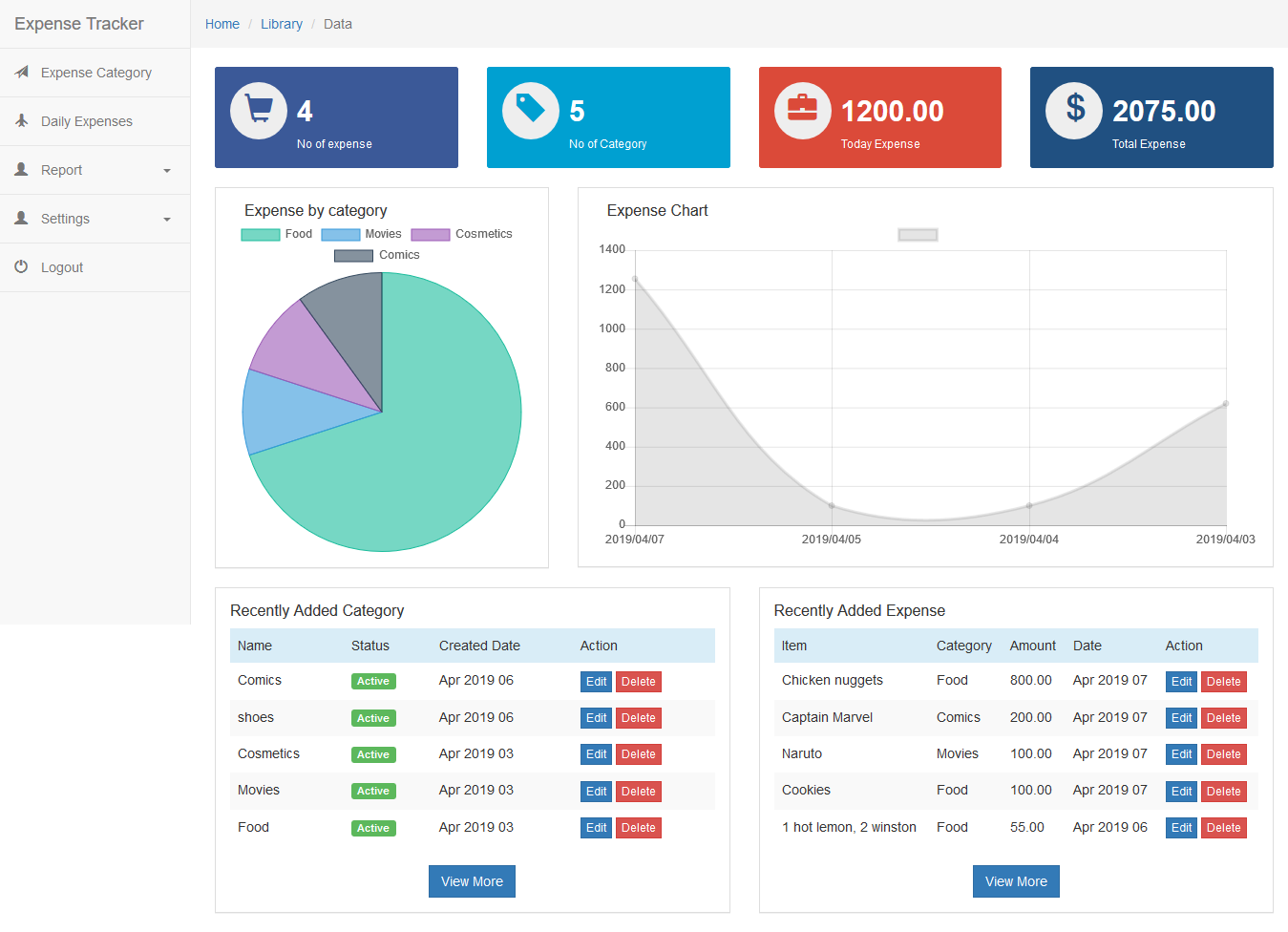
### **User Register**



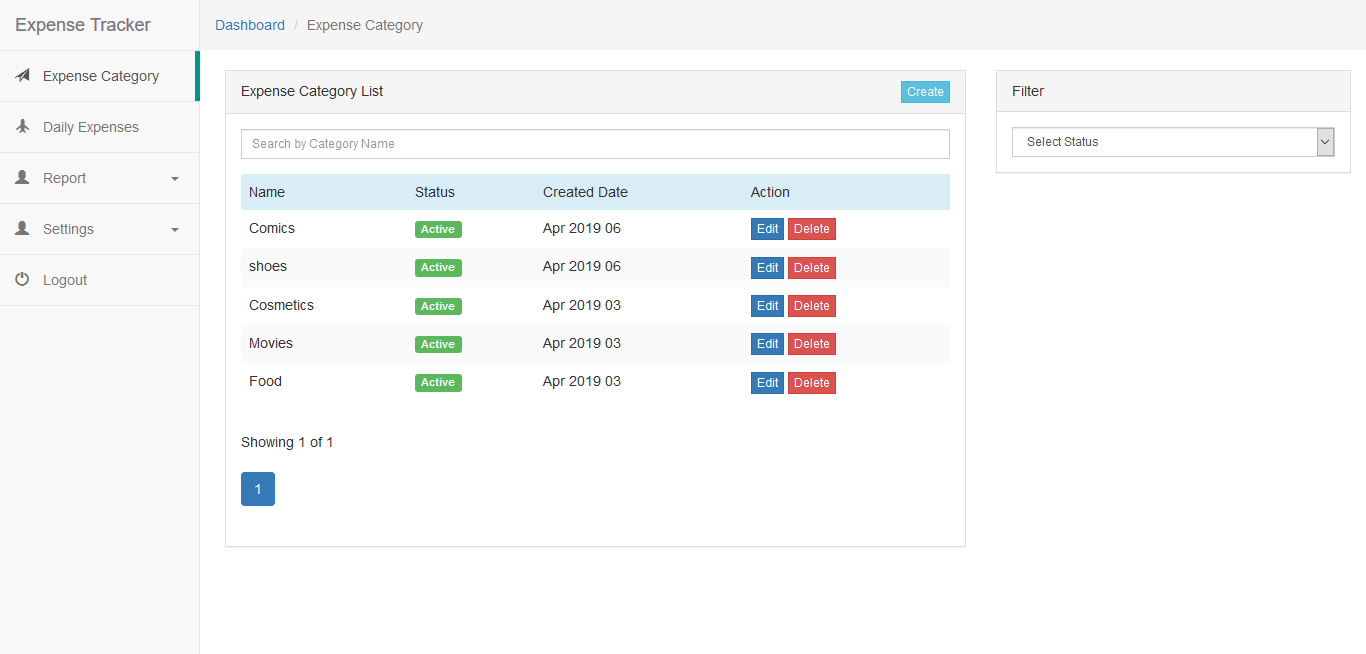
### **User Login**



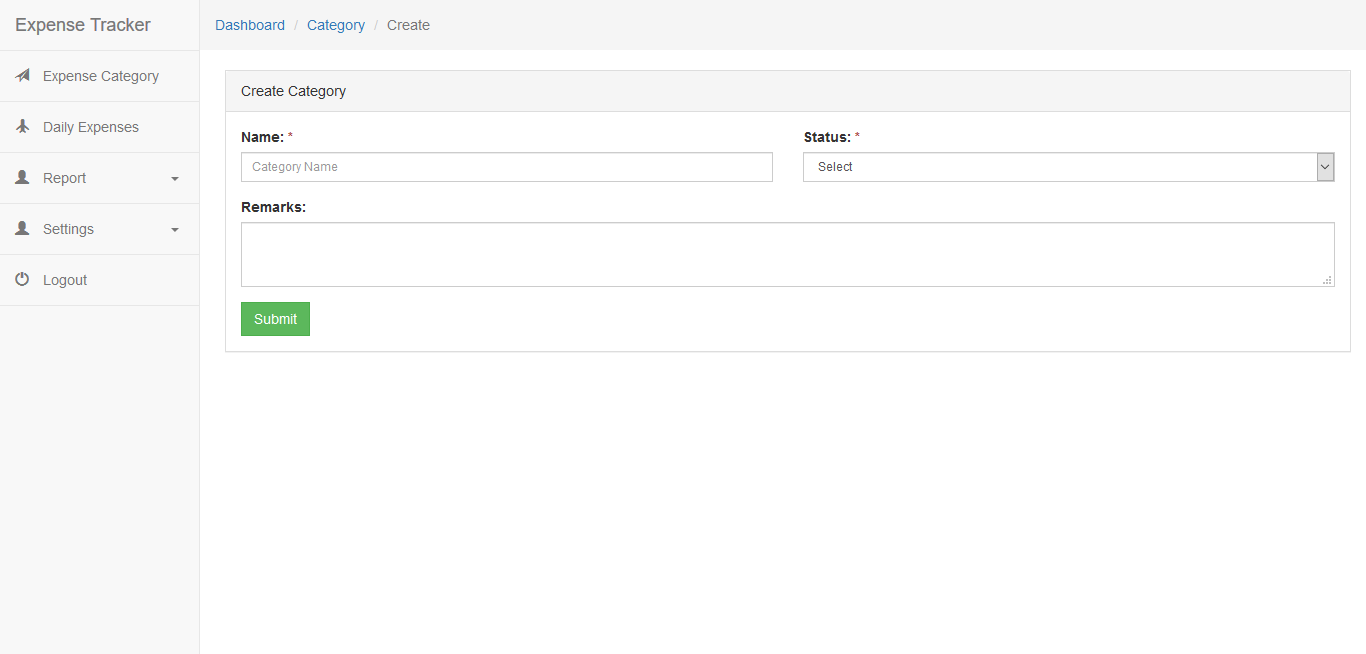
### **Dashboard**



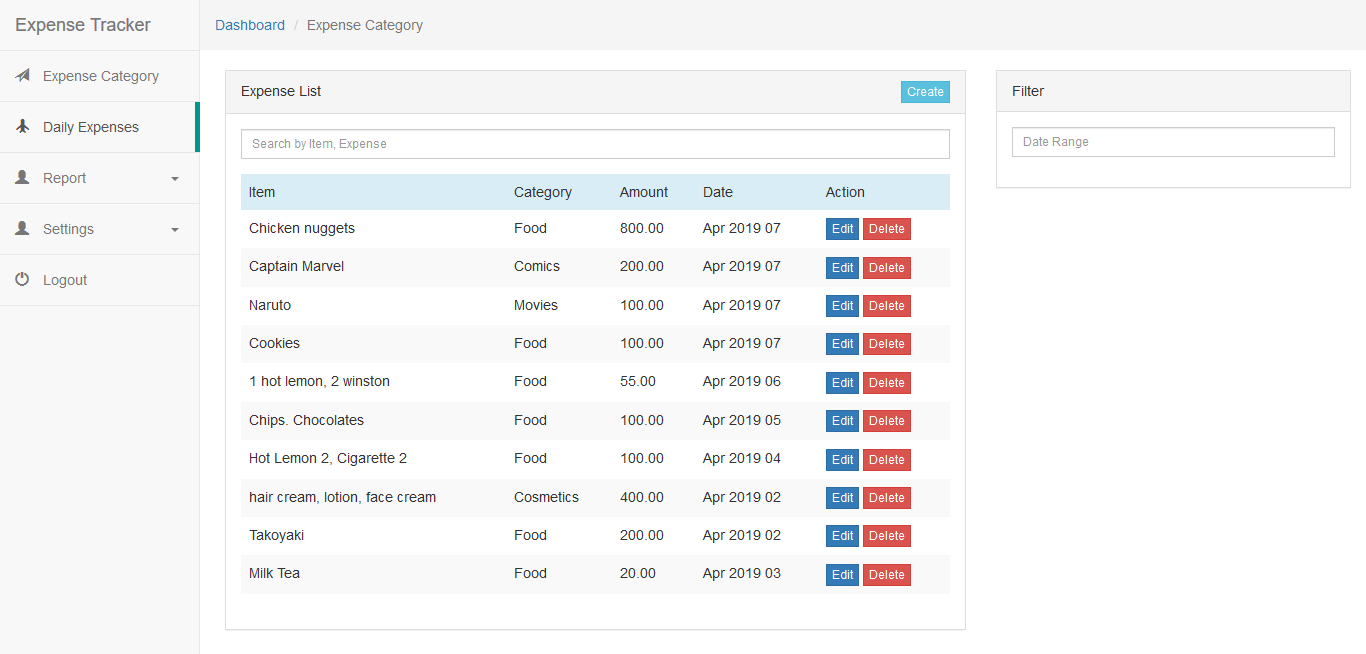
### **Expense Category**



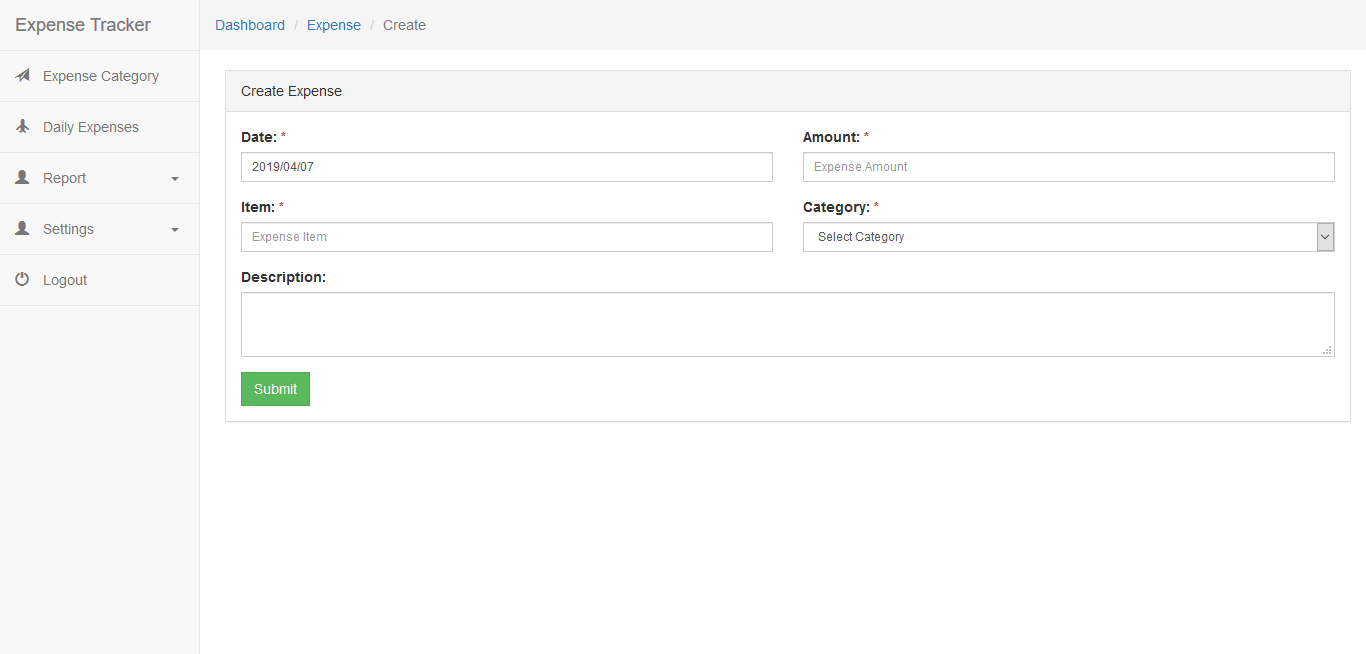
### **Expense Category Create**



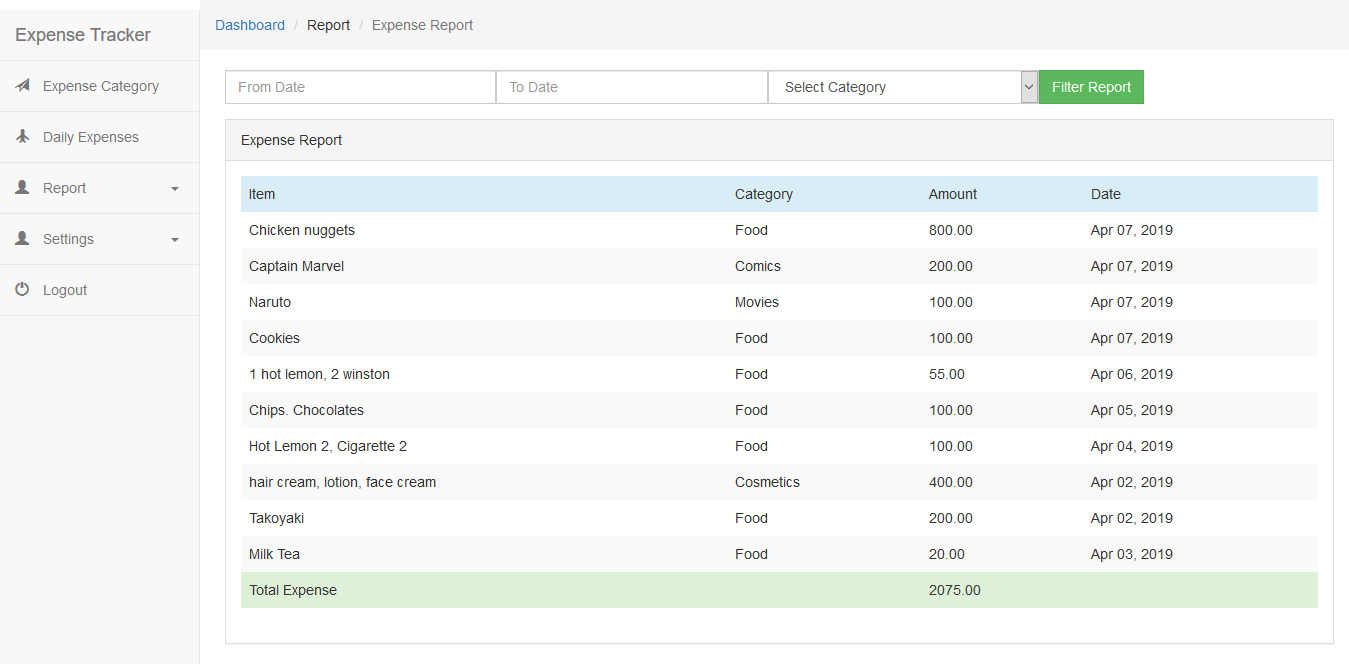
### **Daily Expense**



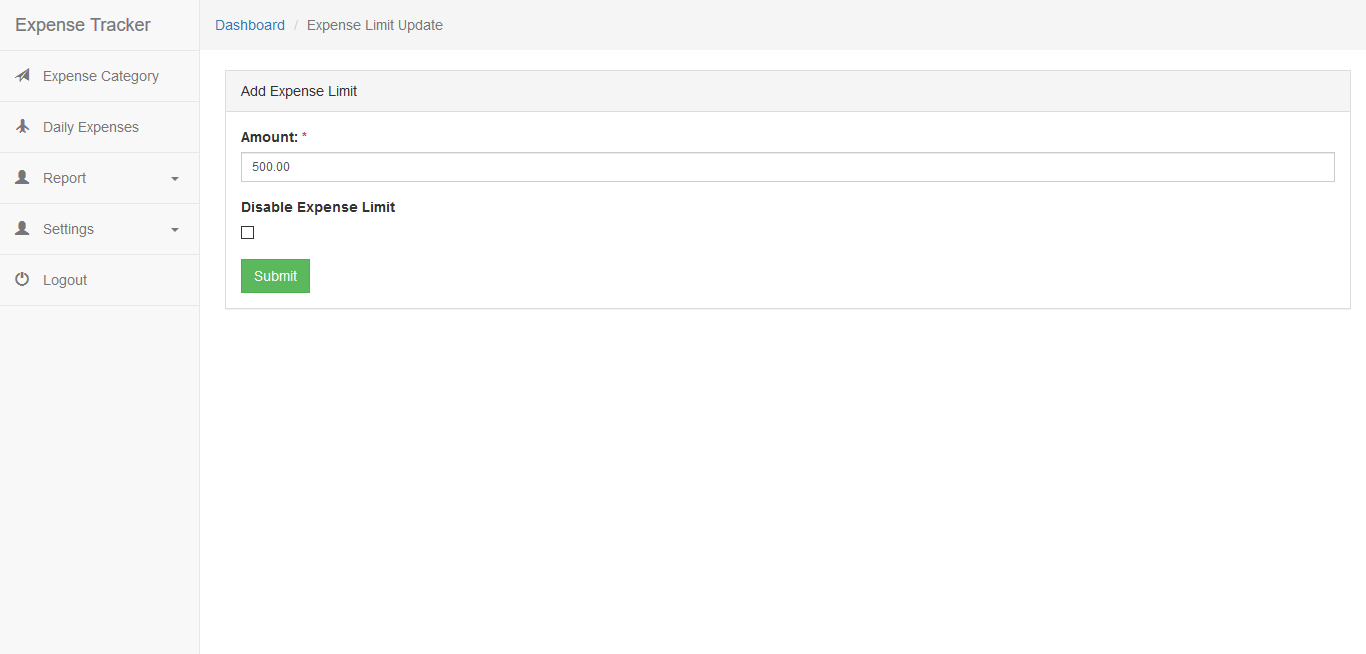
### **Daily Expense Create**



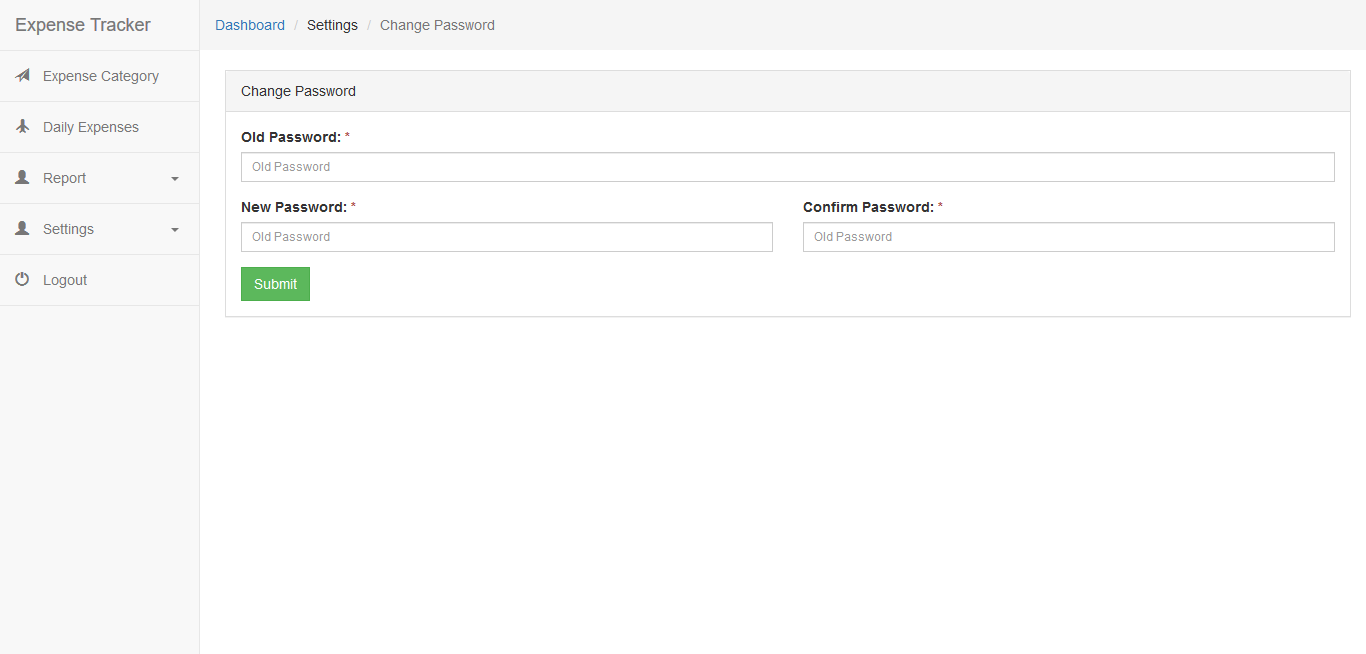
### **Expense Report**



### **Expense Limit Update**



### **Change Password**



# **Deployment Guide**

## **System Requirements**

* MS SQL Server must be installed
* Dot net framework 4.5.2 or higher must be installed

## **Provided Materials**

* Expense Tracker zip file
* Database Backup and Script

## **Steps to setup**

* Restore the backup of the database or run the script file by creating database with name Expense Tracker.
* Unzip the zip file.
* Change the database settings in web config.
* Either publish and Host the application in IIS server or open the application using visual studio and press ctrl + f5 to run the application.