SMART EXPENSE TRACKER

A Project Report for Industrial Training and Internship

submitted by

SAYAN MONI
SOUMYA GHOSH
ANKAN DAS
TRISHA SETH
SONALI MALIK

In the partial fulfillment of the award of the degree of

BCA

in the

BCA(CSE)

Of

JIS UNIVERSITY

Ardent Computech Pvt. Ltd.

ARDENT COMPUTECH PVT. LTD.
High-End Technology Training and Project

ARDENT® GROUP OF COMPANIES
INTERNSHIP | DEVELOPMENT | PROJECT
An ISO 9001 : 2015 Certified Company

Ardent Computech Pvt. Ltd.
Drives you to the Industry
Module 132, SDF Building, Sector V, Salt Lake, Pin - 700091
www.ardentcollaborations.com

CERTIFICATE FROM SUPERVISOR

is to certify that **Sayan Moni, Soumya Ghosh, Ankan Das, Trisha Seth, Sonali Malik ,23CS2061129,23CS2061159,23CS2061020,23CS2061185,23CS2061152**have completed the project titled **Smart Expense Tracker** under my supervision during the period from 16/06/2025 to 05/07/2025 which is in partial fulfillment of requirements for the award of the **BCA** degree and submitted to the Department of his **BCA(CSE)**of **JIS University**.

_____

Signature of the Supervisor

Date: 05/07/2025

Name of the Project Supervisor: Subhojit Santra

Ardent Computech Pvt. Ltd.
Drives you to the Industry
Module 132, SDF Building, Sector V, Salt Lake, Pin - 700091
www.ardentcollaborations.com

BONAFIDE CERTIFICATE

Certified that this project work was carried out under my supervision

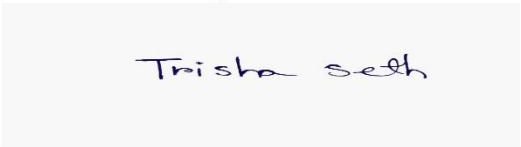
Smart Expense Tracker *i*s the bonafide work of

Name of the student: **Sayan Moni** ***Signature:***

Name of the student: **Soumya Ghosh** ***Signature:***

Name of the student: **Ankan Das** ***Signature:***

Name of the student: **Trisha Seth** ***Signature:***



Name of the student: **Sonali Malik** ***Signature:***

SIGNATURE

Name :

PROJECT MENTOR

SIGNATURE

Name:

EXAMINERS

Ardent Original Seal

ACKNOWLEDGEMENT

The achievement that is associated with the successful completion of any task would be incomplete without mentioning the names of those people whose endless cooperation made it possible. Their constant guidance and encouragement made all our efforts successful.

We take this opportunity to express our deep gratitude towards our project mentor, ***Subhojit Santra*** for giving such valuable suggestions, guidance and encouragement during the development of this project work.

Last but not the least we are grateful to all the faculty members of **Ardent Computech Pvt. Ltd.** for their support.

CONTENT PAGE

1. COMPANY PROFILE

ARDENT (Ardent Computech Pvt. Ltd.), formerly known as Ardent Computech Private Limited, is an ISO 9001:2015 certified Software Development and Training Company based in India. Operating independently since 2003, the organization has recently undergone a strategic merger with ARDENT Technologies, enhancing its global outreach and service offerings.

ARDENT Technologies

ARDENT Technologies delivers high-end IT services across the UK, USA, Canada, and India. Its core competencies lie in the development of customized application software, encompassing end-to-end solutions including system analysis, design, development, implementation, and training. The company also provides expert consultancy and electronic security solutions. Its clientele spans educational institutions, entertainment companies, resorts, theme parks, the service industry, telecom operators, media, and diverse business sectors.

ARDENT Collaborations

ARDENT Collaborations, the Research, Training, and Development division of ARDENT (Ardent Computech Pvt. Ltd.), offers professional IT-enabled services and industrial training programs. These are tailored for freshers and professionals from B.Tech, M.Tech, MBA, MCA, BCA, and MSc backgrounds. ARDENT (Ardent Computech Pvt. Ltd.) provides Summer Training, Winter Training, and Industrial Training to eligible candidates. High-performing students may qualify for stipends, scholarships, and additional benefits based on performance and mentor recommendations.

Associations and Accreditations

ARDENT (Ardent Computech Pvt. Ltd.) is affiliated with the National Council of Vocational Training (NCVT) under the Directorate General of Employment & Training (DGET), Ministry of Labour & Employment, Government of India. The institution upholds strict quality standards under ISO 9001:2015 certification and is dedicated to bridging the gap between academic knowledge and industry skills through innovative training programs.

2. INTRODUCTION

A Smart Expense Tracker is a powerful tool designed to help individuals and businesses efficiently manage their finances by tracking, categorizing, and analyzing expenses. It offers features like automated expense recording, budget setting, real-time notifications, and detailed financial insights. By using a Smart Expense Tracker, users can gain better control over their spending, make informed financial decisions, and achieve their financial goals more effectively.It allows user to : - Record and categorize expenses

- Set budgets and track spending

- Receive notifications and alerts for overspending

- Generate reports and analytics to understand spending habits - Make informed financial decisions

2A.OBJECTIVE

1.Automate Expense Tracking: Develop a system that automates the process of tracking expenses, eliminating the need for manual entry and reducing errors.

2. Categorize and Analyze Expenses: Provide a feature to categorize expenses and generate reports, enabling users to analyze their spending patterns and identify areas for improvement.

3. Improve Financial Decision Making: Provide users with a comprehensive view of their expenses, enabling them to make informed financial decisions and achieve their financial goals.

2B.SCOPE

1. **User Registration and Login:**

   Develop a secure registration and login system for users.

2. **Expense Tracking:**

   Create a feature to track expenses, including date, amount, category, and description.

3. **Categorization and Reporting:**

   Develop a feature to categorize expenses and generate reports.

4. **Alerts and Notifications:**

   Implement a system to generate alerts and notifications when expenses exceed a certain limit.

3. SYSTEM ANALYSIS

3A.IDENTIFICATION OF NEED

System analysis is a crucial phase in the development of our project Smart Expense Tracker, involving creating an efficient and user-friendly platform.

Where users can simply create account and add their daily or monthly expenses by just Login using their created password and will got a auto generated pdf report by Admin

Key Needs Identified:

1. **Efficient Expense Management**

   Users need to easily record and track their daily expenses in various categories.

2. **Budget Planning**

   Helps users set monthly budgets to avoid overspending.

3. **Real-Time Monitoring**

   Instant updates on transactions to keep users aware of their spending.

4. **Data Visualization**

   Graphs and charts to give users clear insights into where their money goes.

3B.FEASIBILITY STUDY

The feasibility study of our Smart Expense Tracker project indicates that the concept is viable and promising across several key areas.

The Smart Expense Tracker project is feasible and practical across all key areas. Technically, it can be developed using widely available tools like React Native, with scalable backend solutions like Node.js. Economically, it has strong revenue potential through category, monthly report. Operationally, it is easy to use, scalable, and meets user needs. Legally, it is viable with compliance to data protection and content licensing laws. The estimated development time is1month, making it achievable within a reasonable timeframe and budget.

Overall, the Smart Expense Tracker project is both realistic and promising, offering a sustainable and impactful solution in the rapidly growing industry.

3C.WORKFLOW

This Document plays a vital role in the development life cycle (SDLC) as it describes the complete requirements of the system. It is meant for use by the developers and will be the basic during the testing phase. Any changes made to the requirements in the future will have to go through a formal change approval process.

The Waterfall Model was the first Process Model to be introduced. It is also referred to as a linear-sequential life cycle model. It is very simple to understand and use. In a waterfall model, each phase must be completed before the next phase can begin and there is no overlapping in the phases.

The waterfall model is the earliest SDLC approach that was used for software development. The waterfall Model illustrates the software development process in a linear sequential flow; hence it is also referred to as a linear-sequential life cycle model. This means that any phase in the development process begins only if the previous phase is complete. In the waterfall model phases do not overlap.

Waterfall Model Design:

The waterfall approach was the first SDLC Model to be used widely in Software Engineering to ensure the success of the project. In "The Waterfall" approach, the whole process of software development is divided into separate phases. In the Waterfall model, typically, the Outcome of one phase acts as the input for the next phase sequentially.

Iterative Waterfall Design:

Definition: The Iterative Waterfall Model is a variation of the traditional Waterfall model, which is a linear and sequential software development methodology. In the Iterative Waterfall Model, the development process is divided into small, manageable cycles, allowing for the revisiting and refinement of phases before progressing to the next stage. It combines the systematic structure of the Waterfall model with the flexibility of iterative development.

The sequential phases in Iterative Waterfall model are:

➢ **Requirement Gathering and Analysis:** All possible requirements of the system to be developed are captured in this phase and documented in a requirement specification doc.

➢ **System Design:** The requirement specifications from the first phase are studied in this phase and system design is prepared. System Design helps in specifying hardware and system requirements and also helps in defining overall system architecture.

➢ **Implementation:** With inputs from system design, the system is first developed in small programs called units, which are integrated in the next phase. Each unit is developed and tested for its functionality which is referred to as Unit Testing.

➢ **Integration and Testing:** All the units developed in the implementation phase are integrated into a system after testing each unit. Post integration the entire system is tested for any faults and failures.

➢ **Deployment of the system:** Once the functional and non-functional testing is done, the product is deployed in the customer environment or released into the market.

➢ **Maintenance:** Some issues come up in the client environment. To fix those issues patches are released. Also to enhance the product some better versions are released. Maintenance is done to deliver these changes in the customer environment.

All these phases are cascaded to each other in progress and are seen as flowing steadily downwards (like a waterfall) through the phases. The next phase is started only after the defined set of goals are achieved for the previous phase and it is signed off, so the name “Iterative Waterfall Model”. In this model, phases do not overlap.

- **Advantages:**

1 . Flexibility: Iterations permit adjustments based on feedback.

2 . Early Delivery: Partial systems can be delivered incrementally.

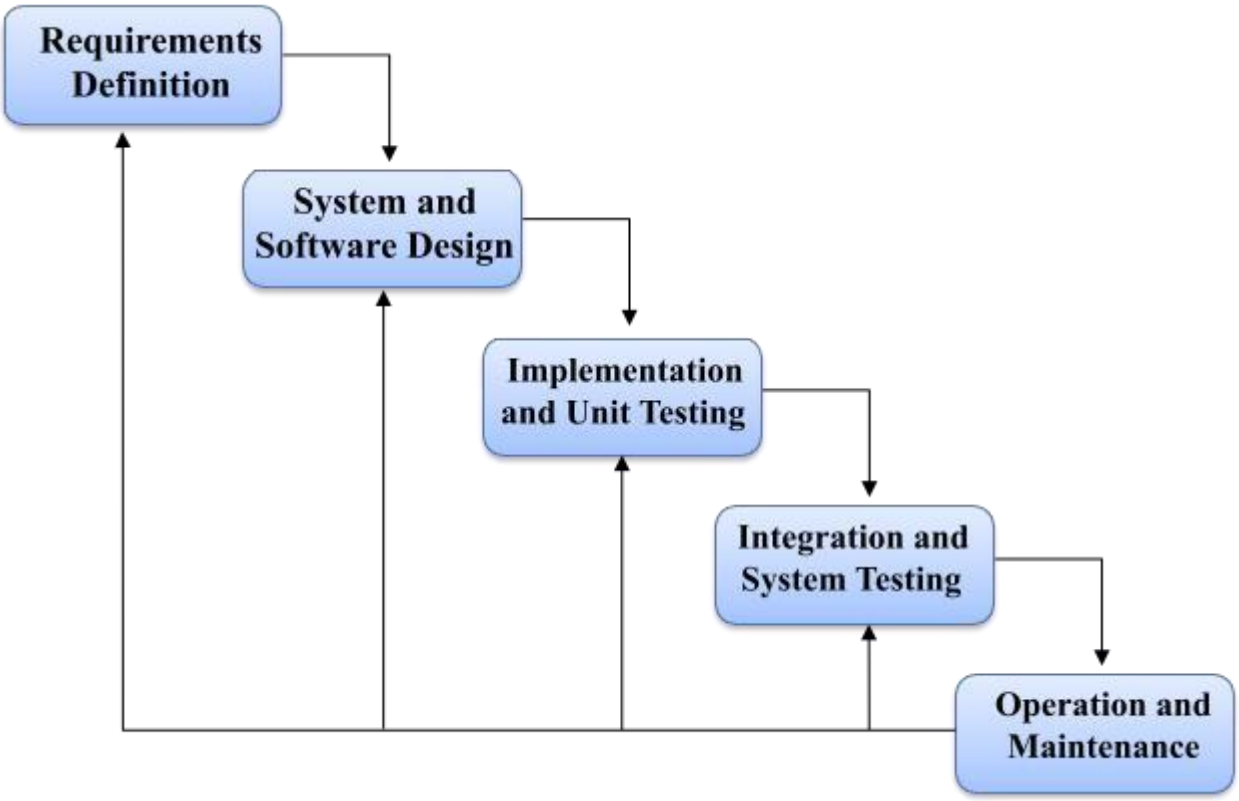
3 . Risk Management: Identifying and addressing issues early in the process.

- **Disadvantages:**

1. Increased Complexity: The iterative nature can make the process more complex.

2. Potential for Scope Creep: Frequent iterations may lead to scope changes.

3. Resource Intensive: Continuous revisiting of phases may demand more resources.



- **<u>Applications</u>:**

The Iterative Waterfall Model is suitable for projects with evolving or unclear requirements. It is commonly used in software development projects where regular feedback and refinement are essential.

Additionally, it is applicable in scenarios where partial system delivery is beneficial, allowing stakeholders to assess progress and make adjustments.

3D .STUDY OF THE SYSTEM

Modules: The modules used in this software are as follows:

- **Register:**

  1 . User Register: Here, the user will register to add their expenses .
  2 . Admin Register: Here, the admin will register to handle all the databases.

- **Login:**
  Here users will login with their Mail ID & created password.

- **Dashboard:**
  Where user can see their previous expenses record& add new expenses using their category, amount .

- **Expense list:**
  Previous expenses list are presented their with amount & category.
- **Expenses PDF Report:**
  A special features that provide auto-generated PDF to users.
- **About:** This page will show the details about the website developers.

3E.INPUT AND OUTPUT

The main inputs, outputs and the major function the details are:

INPUT:

1. Users can log in by entering their **credentials** on the login page.
2. Users can add their expenses
3. Users can remember their category.

OUTPUT:

1. Users can view the **dashboard & expense entry category & add their expenses record .**

2. The **admin** can access a **centralized database** that includes details of **users, add new update** ensuring efficient system management.

3F.SOFTWARE REQUIREMENT SPECIFICATIONS

Software Requirements Specification provides an overview of the entire project. It is a description of a software system to be developed, laying out functional and non-functional requirements. The software requirements specification document enlists enough necessary requirements that are required for the project development. To derive the requirements we need to have a clear and thorough understanding of the project to be developed. This is prepared after detailed communication with the project team and the customer.

The developer is responsible for:

- Developing the system, which meets the SRS and solves all the requirements of the system.
- Demonstrating the system and installing the system at the client's location after acceptance testing is successful.
- Submitting the required user manual describing the system interfaces to work on it and also the documents of the system.

Functional Requirements:

A. User Registration and Authentication:

1. Users should be able to create accounts securely.
2. The system should authenticate users and manage login sessions.

B. Dashboard & Add:

1. Users should be able to add their expenses.

.

C. Entered data Display:

1. Entered expenses can see .
2. Users should be able to download their expenses record.

Hardware Requirements:

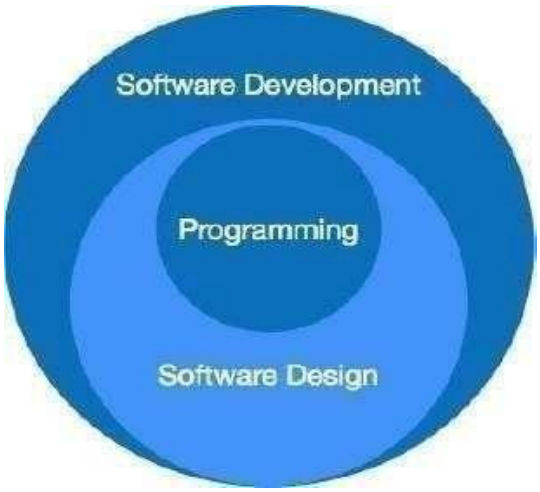
1 . Computer has Intel I3 Processors
2. 8 GB RAM
3. SSD-ROM Drive

Software Requirements:

1. Windows 11 OS
2. Visual Studio Code
3. Mongo DB Atlas

3G.SOFTWARE ENGINEERING PARADIGM APPLIED

Software paradigms refer to the methods and steps, which are taken while designing the software. There are many methods proposed and are in work today, but we need to see where in software engineering these paradigms stand. These can be combined into various categories, though each of them is containedin one another.



The programming paradigm is a subset of Software design paradigm which is further a subset of the Software development paradigm.

There are two levels of reliability. The first is meeting the right requirements. A careful and thorough systems study is needed to satisfy this aspect of reliability. The second level of systems reliability involves the actual work delivered to the user. At this level, the system's reliability is interwoven with software engineering and development.

There are three approaches to reliability.

1. **Error avoidance:** Prevents errors from occurring in software.
2. **Error detection and correction:** In this approach, errors are recognized whenever they are encountered, and correcting the error by the effect of the error of the system does not fail.
3. **Error to lerance:** In this approach, errors are recognized whenever they occur, but enables the system to keep running through degraded performance or Applying values that instruct the system to continue process.

4. SYSTEM DESIGN

4A. DATA FLOW DIAGRAM

A data flow diagram (DFD) is a graphical representation of the "flow" of data through an information system, modeling its process aspects. A DFD is often used as a preliminary step to create an overview of the system, which can later be elaborated.

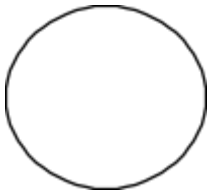
DFD can also be used for the visualization of data processing (structured design). A DFD shows what kind of information will be input to and output from the system, where the data will come from and go to, and where the data will be stored. It does not show information about the timing of the process or information about whether processes will operate in sequence or in parallel (which is shown on a flowchart).

This context-level DFD is next "exploded", to produce a Level 1 DFD that shows some of the detail of the system being modeled. The Level 1 DFD shows how the system is divided into sub-systems (processes), each of which deals with one or more of the data flows to or from an external agent, and which together provide all of the functionality of the system as a whole. It also identifies internal data stores that must be present for the system to do its job and shows the flow of data between the various parts of the system.

Data flow diagrams are one of the three essential perspectives of the structured-systems analysis and design method SSADM. The sponsor of a project and the end users will need to be briefed and consulted throughout all stages of a system's evolution. With a data flow diagram, users can visualize how the system will operate, what the system will accomplish, and how the system will be implemented. The old system's data-flow diagrams can be drawn up and compared with.

How any system is developed can be determined through a data flow diagram model. In the course of developing a set of leveled data flow diagrams, the analyst/designer is forced to address how the system may be decomposed into component sub-systems and to identify the transaction data in the data model. Data flow diagrams can be used in both the Analysis and Design phase of the SDLC. There are different notations to draw data flow diagrams. Defining different visual representations for processes, data stores, data flow, and external entities.

DFD Notation:



Function

File / Database

Input / Output

Flow

DFD Example:

Steps to Construct Data Flow Diagram:

Four Steps are generally used to construct a DFD.

Process should be named and referred for easy reference. Each name should be representative of the reference.

The destination of flow is from top to bottom and from left to right.

When a process is distributed into lower-level details they are numbered.

The names of data stores, sources, and destinations are written in capital letters.

Rules for constructing a Data Flow Diagram:

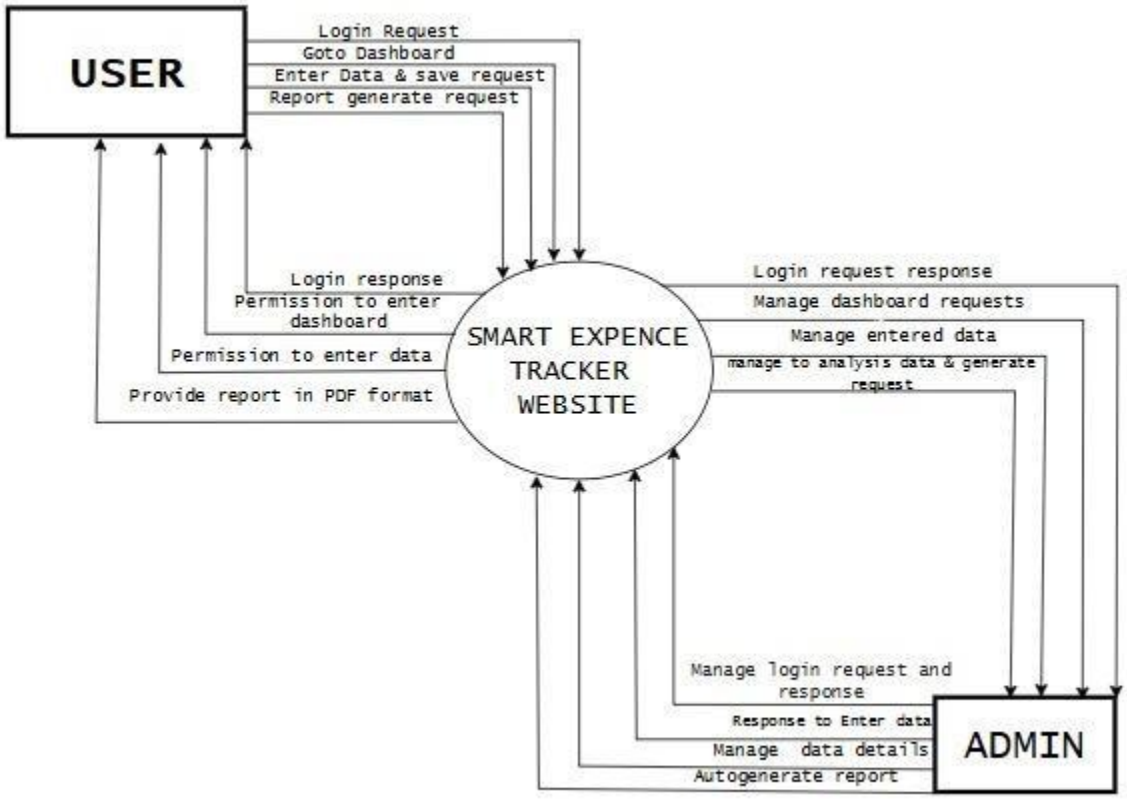
Arrows should not cross each other.

Squares, Circles, and Files must bear a name.

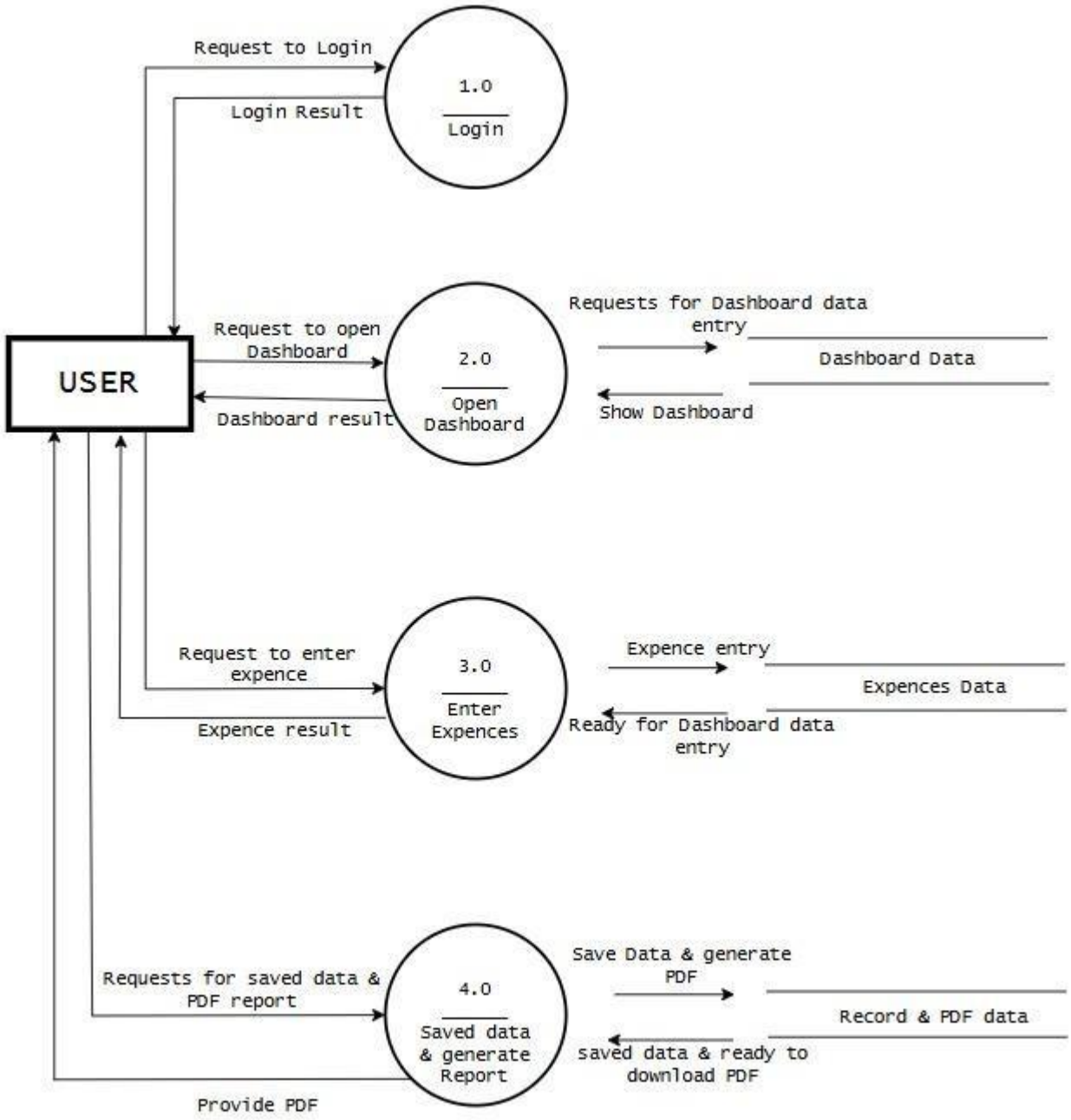
Decomposed data flow squares and circles can have the same names.

Draw all data flow around the outside of the diagram.

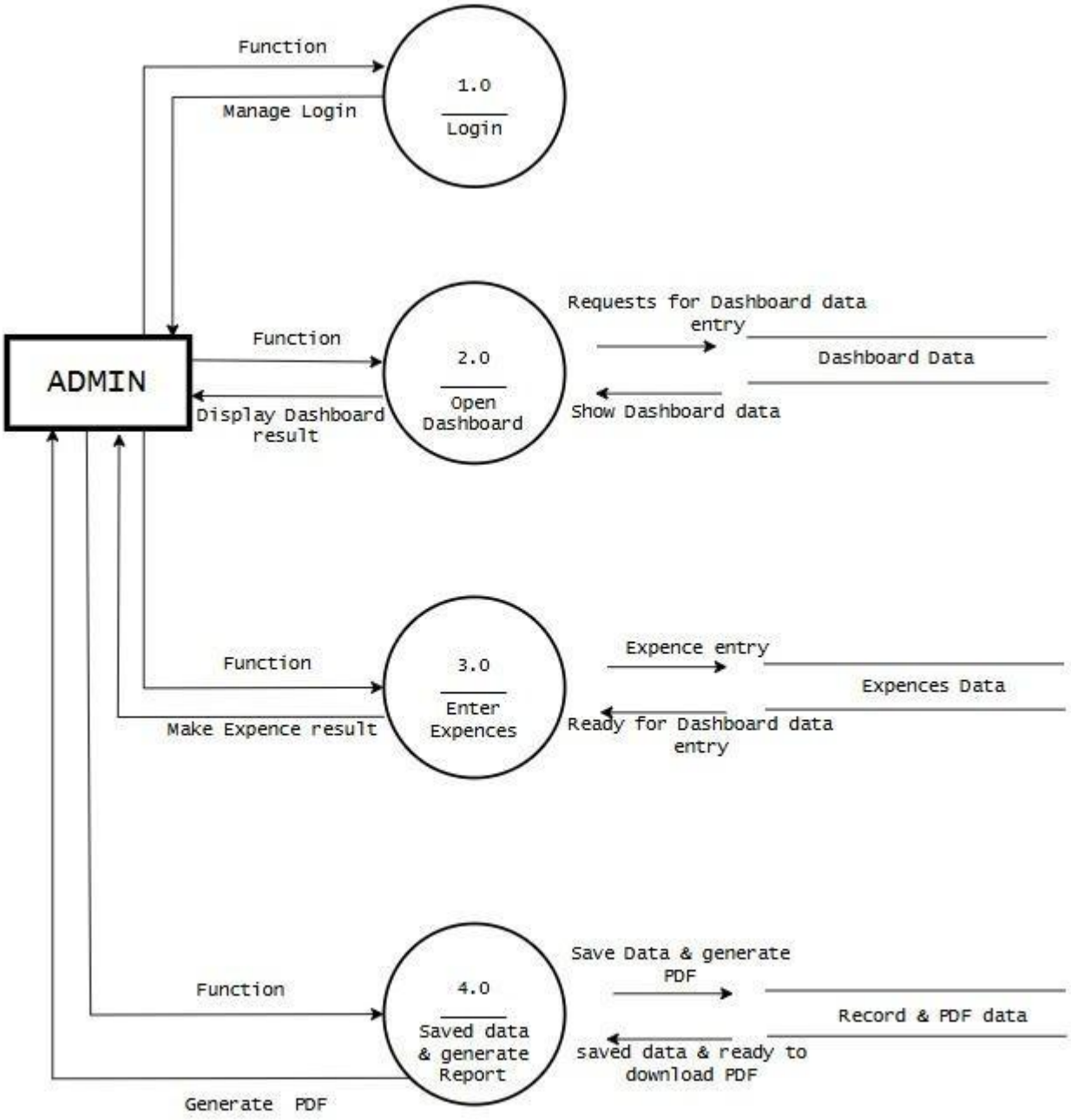
- **LEVEL 0 DFD OR CONTEXT DIAGRAM:**



● **LEVEL 1 DFD:**


Request to Login
1.0
Login
Login Result
USER
Request to open Dashboard
2.0
Open Dashboard
Dashboard result
Requests for Dashboard data entry
Dashboard Data
Show Dashboard
Request to enter expence
3.0
Enter Expences
Expence result
Expence entry
Expences Data
Ready for Dashboard data entry
Requests for saved data & PDF report
4.0
Saved data & generate Report
Provide PDF
Save Data & generate PDF
Record & PDF data
saved data & ready to download PDF

● **LEVEL 1 DFD:**


Function
1.0
Login
Manage Login
ADMIN
Function
2.0
Open Dashboard
Display Dashboard result
Requests for Dashboard data entry
Dashboard Data
Show Dashboard data
Function
3.0
Enter Expences
Make Expence result
Expence entry
Expences Data
Ready for Dashboard data entry
Function
4.0
Saved data & generate Report
Generate PDF
Save Data & generate PDF
Record & PDF data
saved data & ready to download PDF

4B.SEQUENCE DIAGRAM

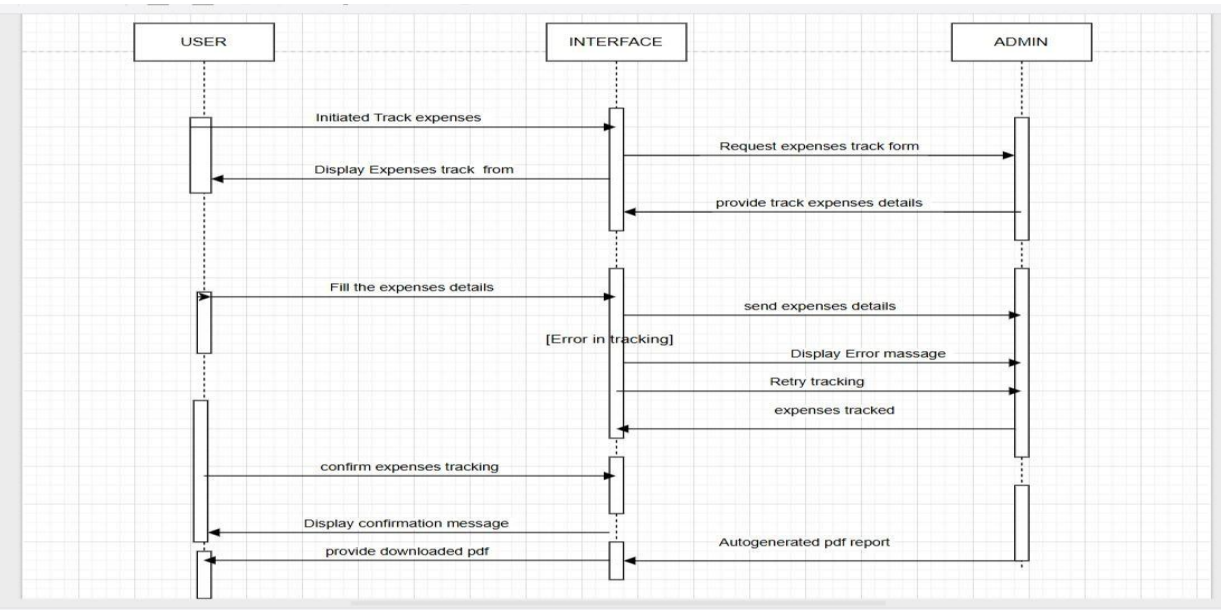
A Sequence diagram is an interaction diagram that shows how processes operate with one another and what is their order. It is a construct of a Message Sequence Chart. A sequence diagram shows object interactions arranged in a time sequence. It depicts the objects and classes involved in the scenario and the sequence of messages exchanged between the objects needed to carry out the functionality of the scenario. Sequence diagrams are typically associated with use case realizations in the Logical View of the system under development.

Sequence diagrams are sometimes called event diagrams or event scenarios.

A sequence diagram shows, as parallel vertical lines (lifelines), different processes or objects that live simultaneously, and, as horizontal arrows, the messages exchanged between them, in the order in which they occur. This allows the specification of simple runtime scenarios in a graphical manner.

A sequence diagram is the most common kind of interaction diagram, which focuses on the message interchange between several life lines .A sequence diagram describes an interaction by focusing on the sequence of messages that are exchanged, along with their corresponding occurrence specifications on the lifelines.

The following nodes and edges are typically drawn in a UML sequence diagram: lifeline, execution specification, message, fragment, interaction, state invariant, continuation, and destruction occurrence.



4C.USE CASE DIAGRAM

A Use case diagram at its simplest is a representation of a user's interaction with the system that shows the relationship between the use rand the different use cases in which the user is involved. A use case diagram can identify the different types of users of a system and the different use cases and will often be accompanied by other types of diagrams as well.

So only static behavior is not sufficient to model a system rather dynamic behavior is more important than static behavior. In UML there are five diagrams available to model dynamic nature and a use case diagram is one of them. Now as we have to discuss that the use case diagram is dynamic in nature there should be some internal or external factors for making the interaction.

These internal and external agents are known as actors. So, use case diagrams consist of actors, use cases, and their relationships. The diagram is used to model the system/subsystem of an application. A single-use case diagram captures a particular functionality of a system.

So, to model the entire system numbers of use case diagrams are used. The purpose of a use case diagram is to capture the dynamic aspect of a system. But this definition is too generic to describe the purpose.

Because the other four diagrams (activity, sequence, collaboration, and State chart) are also having the same purpose. So, we will look into some specific purpose that will distinguish it from the other four diagrams.

Use case diagrams are used to gather the requirements of a system including internal and external influences. These requirements are mostly design requirements. So, when a system is analyzed to gather its functionalities use cases are prepared and actors are identified.

Now when the initial task is complete use case diagrams are modeled to present the outside view. So, in brief, the purposes of use case diagrams can be as follows:

Used to gather requirements of a system.

Used to get an outside view of a system.

Identify external and internal factors influencing the system.

How to draw Use Case Diagram?

Use case diagrams are considered for high level requirement analysis of a system. So, when the requirements of a system are analyzed, the functionalities are captured in use cases.

So, we can say that uses cases are nothing but the system functionalities written in an organized manner. Now the second things which are relevant to the use cases are the actors. Actors can be defined as something that interacts with the system.

The actors can be human user, some internal applications or may be some external applications. So, in a brief when we are planning to draw use case diagram, we should have the following items identified.

Functionalities to be represented as a use case

Actors

Relationships among the use cases and actors.

Use case diagrams are drawn to capture the functional requirements of a system. So, after identifying the above items we have to follow the following guidelines to draw an efficient use case diagram.

The name of a use case is very important. So, the name should be chosen in such a way so that it can identify the functionalities performed.

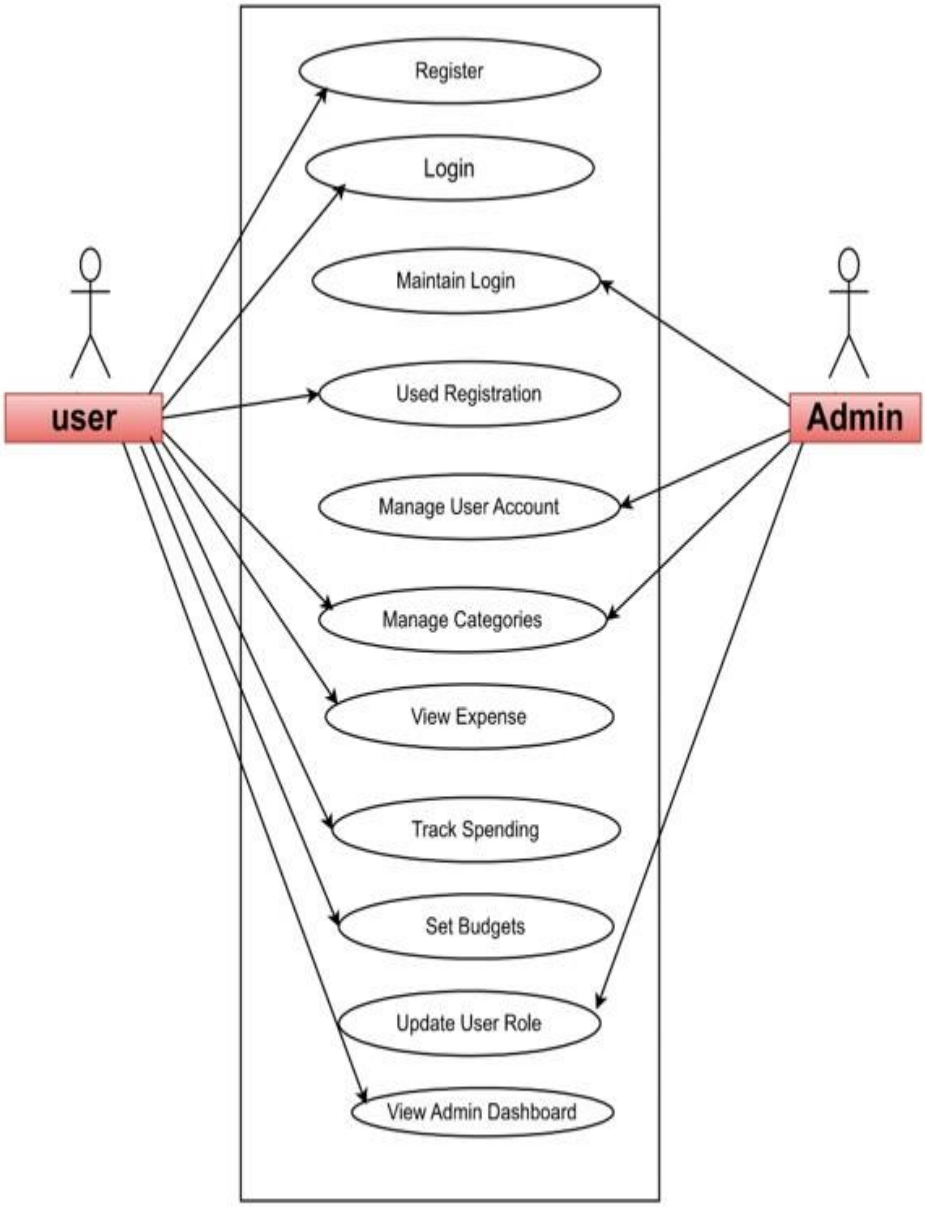
Give a suitable name for actors.

Show relationships and dependencies clearly in the diagram.

Do not try to include all types of relationships. Because the main purpose of the diagram is to identify requirements.

Use note whenever required to clarify some important point

● **USE CASE DIAGRAM**

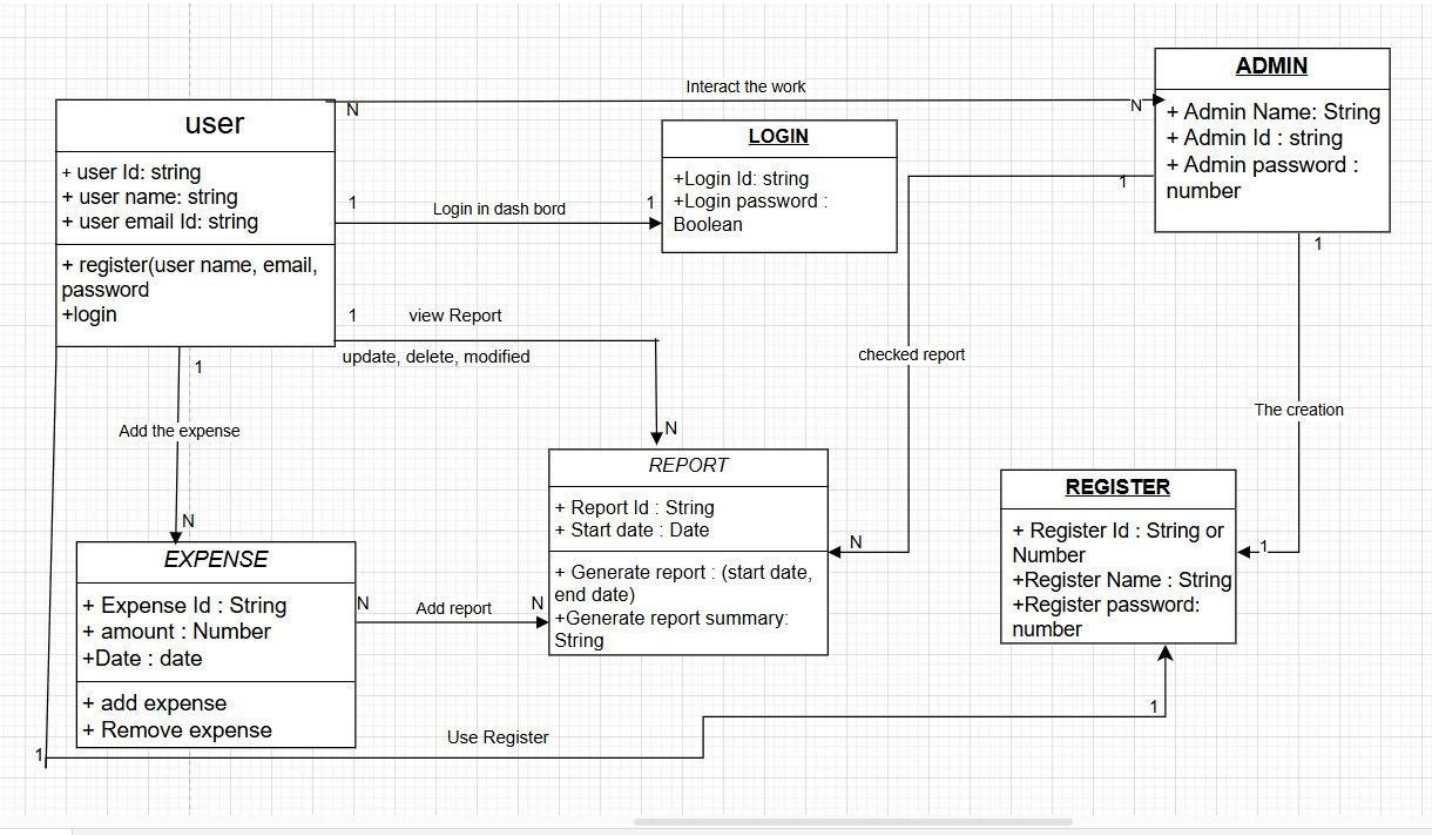


4D.SCHEMA DIAGRAM

The schema is an abstract structure or outline representing the logical view of the database as a whole. Defining categories of data and relationships between those categories, database schema design makes data much easier to retrieve, consume, manipulate, and interpret.

DB schema design organizes data into separate entities, determines how to create relationships between organized entities, and influences the applications of constraints on data. Designers create database schema to give other database users, such as programmers and analysts, a logical understanding of data.

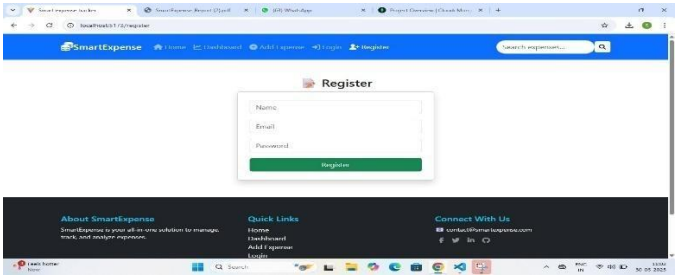
- **SCHEMA DESIGN:**



5. UI SNAPSHOT

❖ **FRONTEND : -**

1) Register page

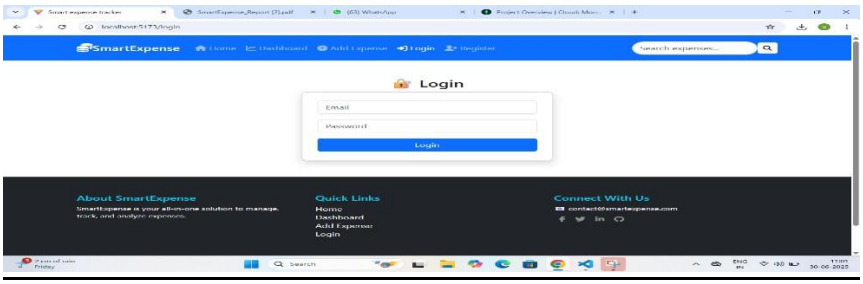


✓ **CODE**

```
// src/pages/Register.js import React,
{ useState } from "react"; import {
useNavigate } from "react-router-dom";

const Register = () => {   const
[form, setForm] = useState({
name: "",     email: "",
password: "",
  });
  const navigate = useNavigate();

  const handleChange = (e) => {     setForm({
...form, [e.target.name]: e.target.value });
  };

  const handleRegister = (e) => {
e.preventDefault();

    // Dummy registration logic
console.log("Registered User:", form);
alert("Registration successful!");
navigate("/login");
```

```
  };

  return (
    <div className="container mt-5">
      <h3 className="text-center">📝 Register</h3>
      <div className="card shadow p-4 mx-auto" style={{ maxWidth: "400px" }}>        <form
onSubmit={handleRegister}>
          <input            type="text"
name="name"
className="form-control mb-3"
placeholder="Name"
value={form.name}
onChange={handleChange}
            required          />
<input            type="email"
name="email"
className="form-control mb-3"
placeholder="Email"
value={form.email}
onChange={handleChange}

required
/>
          <input
type="password"
name="password"
className="form-control mb-3"
placeholder="Password"
value={form.password}
onChange={handleChange}
            required
          />
          <button type="submit" className="btn btn-success w-100">
            Register
          </button>
        </form>
      </div>
    </div>
  );
};

export default Register;
```

2)Login page

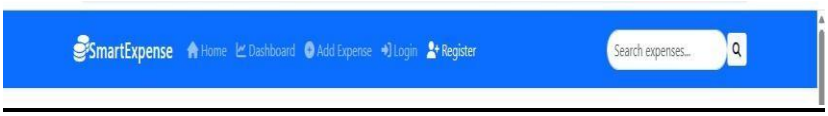


✓ **CODE**

```
import React, { useState } from "react";
import axios from "axios"; import {
useNavigate } from "react-router-dom";

const Login = () => {  const [form, setForm] = useState({
email: "", password: "" });  const [error, setError] =
useState("");  const navigate = useNavigate();

 const handleChange = (e) => {    setForm({
...form, [e.target.name]: e.target.value });
 };

 const handleLogin = async (e)
=> {    e.preventDefault();    try
{
    const res = await axios.post(${import.meta.env.VITE_API_BASE_URL}/login,
form);      localStorage.setItem("token", res.data.token);      alert("Login
successful!");      navigate("/dashboard");
  } catch (err) {      setError(err.response?.data?.message || "Invalid
email or password");
```

```
  }
 };

 return (
  <div className="container mt-5">
   <h3 className="text-center">🔐 Login</h3>
   <div className="card shadow p-4 mx-auto" style={{ maxWidth: "400px" }}>
    {error && <div className="alert alert-danger">{error}</div>}
    <form onSubmit={handleLogin}>
     <input type="email"
name="email"
className="form-control mb-3"
placeholder="Email"
value={form.email}
onChange={handleChange}

required
/>
     <input
type="password"
name="password"
className="form-control mb-3"
placeholder="Password"
value={form.password}
onChange={handleChange}
      required
     />
     <button type="submit" className="btn btn-primary w-100">
      Login
     </button>
    </form>
   </div>
  </div>
 );
};

export default Login;
```

3)Header page



- ✓ **CODE**

```
import React from "react"; import {
NavLink } from "react-router-dom";

const Header = ({ onSearch }) => {
const handleSearch = (e) => {
onSearch(e.target.value);
};

return (
<nav className="navbar navbar-expand-lg navbar-dark bg-primary shadow-sm py-3">
<div className="container">
{/* Logo / Brand - Left */}
<NavLink className="navbar-brand d-flex align-items-center" to="/">
<i className="fas fa-coins fa-lg mr-2"></i>
<span style={{ fontWeight: "600", fontSize: "1.3rem", letterSpacing: "0.5px" }}>
SmartExpense
</span>
```

```
</NavLink>

{/* Mobile Toggler */}
<button className="navbar-
toggler" type="button" data-
toggle="collapse" data-
target="#navbarNav" aria-
controls="navbarNav" aria-
expanded="false" aria-
label="Toggle navigation"
>
<span className="navbar-toggler-icon"></span>
</button>

{/* Navigation Links + Search */}
<div className="collapse navbar-collapse justify-content-between" id="navbarNav">
{/* Nav Links */}
<ul className="navbar-nav mr-auto ml-3">
<li className="nav-item">
<NavLink to="/home" className={({ isActive }) => nav-link ${isActive ? "active
fontweight-bold" : ""}}>
<i className="fas fa-home mr-1"></i> Home
</NavLink>
</li>
<li className="nav-item">
<NavLink to="/dashboard" className={({ isActive }) => nav-link ${isActive ? "active
font-weight-bold" : ""}}>
<i className="fas fa-chart-line mr-1"></i> Dashboard
</NavLink>
</li>
<li className="nav-item">
<NavLink to="/addexpense" className={({ isActive }) => nav-link ${isActive ? "active
font-weight-bold" : ""}}>
<i className="fas fa-plus-circle mr-1"></i> Add Expense
</NavLink>
</li>
<li className="nav-item">
<NavLink to="/login" className={({ isActive }) => nav-link ${isActive ? "active
fontweight-bold" : ""}}>
<i className="fas fa-sign-in-alt mr-1"></i> Login
</NavLink>
</li>
```

```
<li className="nav-item">
<NavLink to="/register" className={({ isActive }) => nav-link ${isActive ? "active
fontweight-bold" : ""}}>
<i className="fas fa-user-plus mr-1"></i> Register
</NavLink>
</li>
</ul>

{/* Search Bar - Input and Button inline */}
<form className="form-inline d-flex align-items-center my-2 my-lg-0">
<input className="form-control mr-2"
type="search" placeholder="Search
expenses..." aria-label="Search" style={{
width: "200px", borderRadius: "20px" }}
onChange={handleSearch}
/>
<button className="btn btn-light btn-sm" type="button">
<i className="fas fa-search"></i>
</button>
</form>
</div>
</div>
</nav>
);
};

export default Header;
```

4)Home Page

- ✓ **CODE**

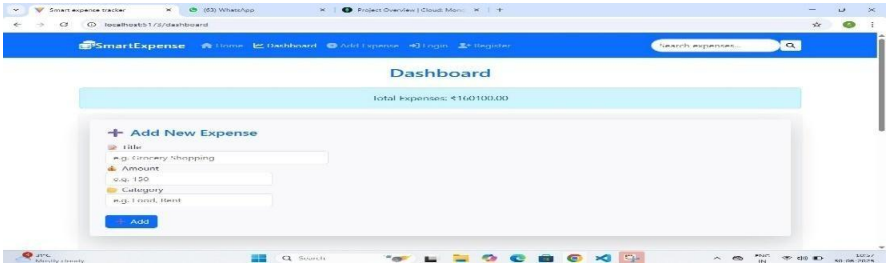
```
import React from 'react'; import {
Link } from 'react-router-dom';
```

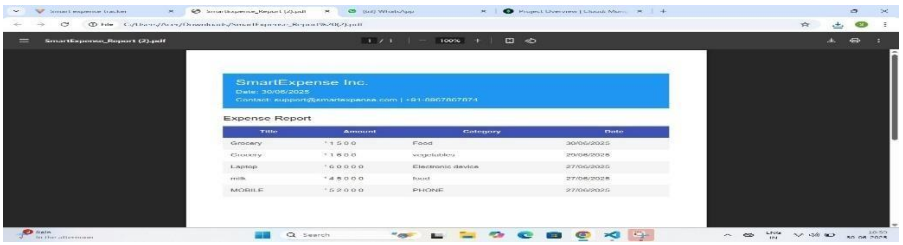
```
import img1 from './img1.jpg';
import img2 from './img2.jpg';
import img3 from './img3.jpg';

const Home = () => (
  <div>
    {/* Hero Section */}
    <div          className="jumbotron          jumbotron-fluid          bg-info          text-white
        mb-5 animate_animated animate_fadeIn">
      <div className="container text-center py-5">
        <h1          className="display-4          font-weight-bold          animate_animated
animate_zoomInDown">
          Welcome to <span style={{  textShadow: '2px 2px #000' }}>SmartExpense</span>
        </h1>
        <p className="lead my-4 animate_animated animate_fadeInUp">
          Your ultimate platform for expense tracking, analysis, and smarter money
management.
        </p>
        <p className="animate_animated animatefadeIn animate_delay-1s">
          Get insights, set financial goals, and visualize spending patterns in just a few clicks.
        </p>
        <div className="mt-4 animate_animated animatefadeInUp animate_delay-2s">
          <Link to="/dashboard" className="btn btn-light btn-lg mx-2 shadow">
            <i className="fas fa-chart-line mr-2"></i> Dashboard
          </Link>
          <Link to="/addexpense" className="btn btn-outline-light btn-lg mx-2 shadow">
            <i className="fas fa-plus-circle mr-2"></i> Add Expense
          </Link>
        </div>
      </div>
    </div>

    {/* Features */}
    <div className="container mb-5">
      <h2          className="text-center          mb-4    text-primary    animate_animated
animate_fadeInDown">🔍 Why Choose SmartExpense?</h2>
      <p className="text-center text-muted mb-5 animate_animated animate_fadeIn">
Experience seamless budgeting, real-time tracking, and financial empowerment all in
one place.
      </p>

      <div className="row text-center">
```

```
      {/* Add Expense */}
      <div className="col-md-4 mb-4 animate_animated animate_fadeInLeft">
       <div className="card h-100 shadow border-0 hover-shadow transition">
        <img src={img1} alt="Add
Expense" className="card-img-top p-4"
style={{ height: "200px", objectFit: "contain" }}
        />
        <div className="card-body">
         <h5 className="card-title text-info">📝 Easy Logging</h5>
         <p className="card-text">
          Log your expenses quickly with category selection, tags, and optional notes.
         </p>
         <p><strong>💡 Tip:</strong> Use voice input on mobile to log faster!</p>
        </div>
       </div>
      </div>

      {/* View Reports */}
      <div className="col-md-4 mb-4 animate_animated animate_fadeInUp">
       <div className="card h-100 shadow border-0 hover-shadow transition">
        <img src={img3}
alt="Reports" className="card-img-top p-
4" style={{ height: "200px", objectFit:
"contain" }}
        />
        <div className="card-body">
         <h5 className="card-title text-success">📊 Smart Reports</h5>
         <p className="card-text">
          Instantly view monthly breakdowns, graphs, and charts to understand your habits.
         </p>
         <p><strong>🗀 Export:</strong> Download your reports as PDF anytime!</p>
        </div>
       </div>
      </div>

      {/* Insights */}
      <div className="col-md-4 mb-4 animate_animated animate_fadeInRight">
       <div className="card h-100 shadow border-0 hover-shadow transition">
        <img src={img2}
alt="Insights" className="card-img-top p-
```

```
4"              style={{ height: "200px", objectFit:
"contain" }}
          />
          <div className="card-body">
            <h5 className="card-title text-warning">📈 Smart Insights</h5>
            <p className="card-text">
              Get personalized recommendations and trend forecasts based on your past data.
            </p>
            <p><strong>🔔 Alerts:</strong> Set up monthly budget notifications!</p>
          </div>
        </div>
      </div>
    </div>
  </div>

  {/* Call To Action Section */}
  <div className="bg-light py-5 mt-5 animate_animated animate_fadeInUp">
    <div className="container text-center">
      <h2 className="mb-3">Ready to take control of your finances?</h2>
      <p className="mb-4 text-muted">
        Sign up now and start your journey to smarter spending. It's quick and free!
      </p>
      <Link to="/register" className="btn btn-primary btn-lg px-4 shadow">
        Register Now
      </Link>
    </div>
  </div>
 </div>
);

export default Home;
```

5)Dashboard page





✓ **CODE**

```
import React, { useState, useEffect } from
"react"; import axios from "axios"; import
AddExpense from "./AddExpense"; import
ExpenseList from "./ExpenseList"; import
jsPDF from "jspdf"; import autoTable from
"jspdf-autotable";

const Dashboard = () => {   const
[expenses, setExpenses] = useState([]);

 useEffect(() => {
fetchExpenses();
 }, []);

 const fetchExpenses = async () => {
   try {       const res = await
axios.get("http://localhost:5000/api/expenses");
setExpenses(res.data);     } catch (err) {
     console.error("❌ Fetch error:", err.message);
   }
 };

 const addExpense = async (form) => {     try {       const res =
await axios.post("http://localhost:5000/api/expenses", {
       ...form,
       amount: parseFloat(form.amount),
     });
     setExpenses([res.data, ...expenses]);
```

```
  } catch (err) {       console.error("❌ Add
error:", err.message);
  }
 };

 const deleteExpense = async (id) => {     try {       await
axios.delete(http://localhost:5000/api/expenses/${id});
setExpenses(expenses.filter((e) => e._id !== id));
  } catch (err) {
    console.error("❌ Delete error:", err.message);
  }
 };

 const generatePDF = ()
=> {     const doc = new
jsPDF({       orientation:
"portrait",       unit:
"mm",       format: "a4"
  });

  const marginLeft = 15;
const marginTop = 20;

  const companyName = "SmartExpense Inc.";     const reportDate = new
Date().toLocaleDateString();     const contactInfo = "Contact:
support@smartexpense.com | +91-8967867874";

  // Header background
doc.setFillColor(33, 150, 243); // blue
doc.rect(marginLeft, marginTop, 180, 30,
"F");

  // Header text
  doc.setTextColor(255, 255, 255);
doc.setFontSize(18);     doc.text(companyName,
marginLeft + 5, marginTop + 10);

  doc.setFontSize(11);     doc.text(Date: ${reportDate},
marginLeft + 5, marginTop + 17);     doc.text(contactInfo,
marginLeft + 5, marginTop + 24);
```

```
   // Section Title     doc.setTextColor(40, 40, 40);
doc.setFontSize(14);     doc.text("Expense Report",
marginLeft, marginTop + 40);

   // Table     autoTable(doc, {       startY:
marginTop + 45,       margin: { left: marginLeft,
right: marginLeft },       head: [["Title",
"Amount", "Category", "Date"]],       body:
expenses.map((exp) => [
       exp.title,
       ₹${exp.amount},         exp.category,
new Date(exp.date).toLocaleDateString(),
     ]),
styles: {
fontSize: 10,
cellPadding: 3,
     },
     headStyles: {
fillColor: [63, 81, 181],
textColor: 255,
halign: "center",
     },
     didDrawPage: (data) => {         const pageCount =
doc.internal.getNumberOfPages();
doc.setFontSize(9);         doc.setTextColor(150);
       doc.text(Page ${pageCount}, doc.internal.pageSize.width - marginLeft - 10,
doc.internal.pageSize.height - 10);
     },
   });

   doc.save("SmartExpense_Report.pdf");
 };

 const total = expenses.reduce((acc, curr) => acc + curr.amount, 0);

 return (
   <div className="container py-4 animate_animated animate_fadeIn">
     <h2 className="text-center mb-4 text-primary animate_animated animate_zoomIn">
       Dashboard
```

```
    </h2>

    <div        className="alert alert-info text-center font-weight-bold
shadow-sm"        style={{ fontSize: "1.1rem" }}
    >
      Total Expenses: ₹{total.toFixed(2)}
    </div>

    {/* Add Expense Form */}
    <div className="mb-4 p-3 bg-light rounded shadow-sm animate_animated
animate_fadeInUp">
      <AddExpense onAdd={addExpense} />
    </div>

    {/* Expense List */}
    <div className="mb-4 animate_animated animate_fadeInUp">
      <ExpenseList expenses={expenses} onDelete={deleteExpense} />
</div>

    {/* PDF Button */}
    <div className="text-center">
      <button          className="btn btn-primary btn-lg
shadow-sm px-4 py-2"          onClick={generatePDF}
style={{            transition: "all 0.3s ease",
borderRadius: "25px",
        }}
        onMouseEnter={(e) =>
          (e.target.style.backgroundColor = "#0056b3")
        }
        onMouseLeave={(e) =>
          (e.target.style.backgroundColor = "#007bff")
        }
      >
        <i className="fas fa-download mr-2"></i>Download PDF Report
      </button>
    </div>
   </div>
  );
};
```

export default Dashboard;

6)AddExpense page

✓ **CODE :**

```
import React, { useState } from "react";

const AddExpense = ({ onAdd }) => {
 const [form, setForm] = useState({ title: "", amount: "", category: "" });

 const handleSubmit = (e) => {
e.preventDefault();
onAdd(form);
  setForm({ title: "", amount: "", category: "" });
 };

 return (
  <div className="card shadow-lg border-0 p-4 mb-4 bg-light">
   <h4 className="mb-3 text-primary font-weight-bold">
    ➕ Add New Expense
   </h4>

   <form onSubmit={handleSubmit}>
    <div className="form-row align-items-end">
     <div className="form-group col-md-4">
      <label className="font-weight-semibold">📝 Title</label>
      <input
       type="text"
className="form-control"
placeholder="e.g. Grocery Shopping"
       value={form.title}
```

```
          onChange={(e) => setForm({ ...form, title: e.target.value })}
required
         />
        </div>

        <div className="form-group col-md-3">
         <label className="font-weight-semibold">💰 Amount</label>
         <input
type="number"
className="form-control"
placeholder="e.g. 150"
value={form.amount}
          onChange={(e) => setForm({ ...form, amount: e.target.value })}
required
         />
        </div>

        <div className="form-group col-md-3">
         <label className="font-weight-semibold">🗁 Category</label>
         <input
          type="text"
className="form-control"
placeholder="e.g. Food, Rent"
value={form.category}
          onChange={(e) => setForm({ ...form, category: e.target.value })}
required
         />
        </div>
         <br></br>
        <div className="form-group col-md-2 text-right">
         <button
type="submit"
          className="btn btn-primary btn-block font-weight-bold"
          style={{ height: "100%" }}
         >
          ➕ Add
         </button>
        </div>
       </div>
     </form>
```

```
  </div>
 );
};

export default AddExpense;
```

7)ExpenseList page

CODE

```
import React from "react";

const ExpenseList = ({ expenses, onDelete }) => (
  <div className="table-responsive">
    <table className="table table-striped table-bordered shadow-sm">
      <thead className="thead-dark">
        <tr>
          <th>Title</th>
          <th>Amount (₹)</th>
          <th>Category</th>
          <th>Date</th>
          <th>Action</th>
        </tr>
      </thead>
      <tbody>
        {expenses.length > 0 ? (
expenses.map((item) => (
```

```
<tr key={item._id}>
<td>{item.title}</td>
              <td>₹{item.amount}</td>
              <td>{item.category}</td>
              <td>{new Date(item.date).toLocaleDateString()}</td>
              <td>
                <button className="btn btn-sm btn-outline-danger" onClick={()
=> onDelete(item._id)}>Delete</button>
              </td>
            </tr>
          ))
        ) : (
          <tr>
            <td colSpan="5" className="text-center text-muted">No expenses
recorded.</td>
          </tr>
        )}
      </tbody>
    </table>
  </div>
);

export default ExpenseList;
```

8)Footer page

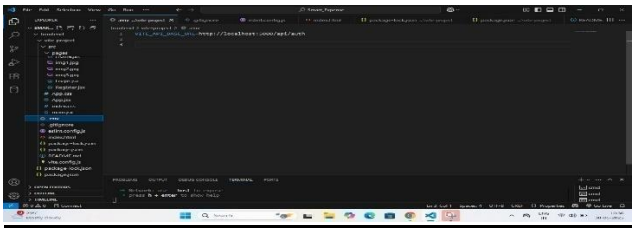
- ✓ ***CODE***

```
import React from "react"; import {
Link } from "react-router-dom";

const Footer = () => (
  <footer className="bg-dark text-white pt-5 pb-3 mt-5">
    <div className="container">
      <div className="row">
```

```
      {/* About Section */}
      <div className="col-md-4 mb-4">
        <h5 className="text-info">About SmartExpense</h5>
        <p className="small">
          SmartExpense is your all-in-one solution to manage, track, and analyze expenses.
        </p>
      </div>

      {/* Quick Links */}
      <div className="col-md-4 mb-4">
        <h5 className="text-info">Quick Links</h5>
        <ul className="list-unstyled">
          <li><Link     className="text-white          text-decoration-none"
to="/home">Home</Link></li>
          <li><Link     className="text-white          text-decoration-none" to="/dashboard">
Dashboard</Link></li>
          <li><Link className="text-white text-decoration-none" to="/addexpense"> Add
Expense</Link></li>
          <li><Link     className="text-white          text-decoration-none"
to="/login">Login</Link></li>
        </ul>
</div>

      {/* Contact & Social */}
      <div className="col-md-4 mb-4">
        <h5 className="text-info">Connect With Us</h5>
        <p className="small mb-2">📧 contact@smartexpense.com</p>
        <div>
          <a     href="https://facebook.com"  target="_blank"          rel="noopener noreferrer"
className="text-white-50 mx-2">
            <i className="fab fa-facebook-f"></i>
          </a>
          <a     href="https://twitter.com"       target="_blank"          rel="noopener noreferrer"
className="text-white-50 mx-2">
            <i className="fab fa-twitter"></i>
          </a>
          <a     href="https://linkedin.com"    target="_blank"          rel="noopener noreferrer"
className="text-white-50 mx-2">
            <i className="fab fa-linkedin-in"></i>
          </a>
          <a     href="https://github.com"       target="_blank"          rel="noopener noreferrer"
className="text-white-50 mx-2">
```

```
            <i className="fab fa-github"></i>
          </a>
        </div>
      </div>
    </div>

    <hr className="border-secondary" />

    <div className="text-center small text-muted">
      © {new Date().getFullYear()} <strong>SmartExpense</strong>. All
      rights reserved.<br />
      Designed & Developed by <strong>Soumya Ghosh</strong>
    </div>
  </div>
 </footer>
);

export default Footer;
```

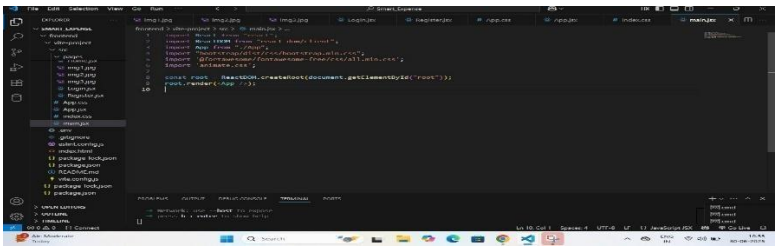
9).env page



✓ **CODE**

```
VITE_API_BASE_URL=http://localhost:5000/api/auth
```

10)main.jsx

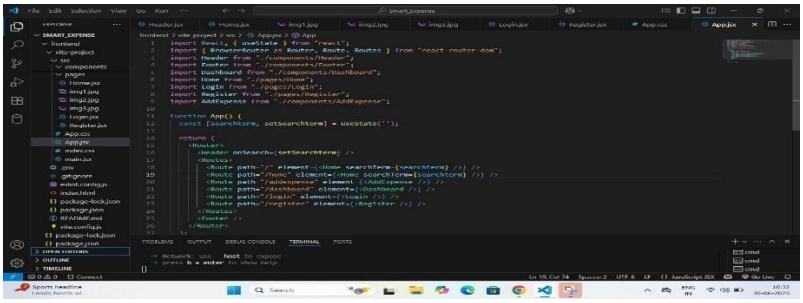


✓ **CODE**

import React from "react"; import ReactDOM from "react-dom/client"; import App from "./App"; import "bootstrap/dist/css/bootstrap.min.css"; import '@fortawesome/fontawesome-free/css/all.min.css'; import 'animate.css';

const root = ReactDOM.createRoot(document.getElementById("root")); root.render(<App />);

11)App.jsx



C
O
D
E

```
import React, { useState } from "react"; import { BrowserRouter as Router, Route, Routes } from "react-router-dom"; import Header from "./components/Header"; import Footer from "./components/Footer"; import Dashboard from "./components/Dashboard"; import Home from "./pages/Home"; import Login from "./pages/Login"; import Register from "./pages/Register";
import AddExpense from "./components/AddExpense";

function App() {   const [searchterm, setSearchterm] = useState('');

 return (
  <Router>
   <Header onSearch={setSearchterm} />
   <Routes>
    <Route path="/" element={<Home searchTerm={searchterm} />} />
    <Route path="/home" element={<Home searchTerm={searchterm} />} />
    <Route path="/addexpense" element={<AddExpense />} />
```

```
        <Route path="/dashboard" element={<Dashboard />} />
        <Route path="/login" element={<Login />} />
        <Route path="/register" element={<Register />} />
      </Routes>
      <Footer />
    </Router>
  );
}

export default App;
```

❖ **BACKEND:-**

1. Controllers-authController.js:-

```
// controllers/authController.js
const User =
require("../models/User"); const
bcrypt = require("bcryptjs"); const
jwt = require("jsonwebtoken");

// Register controller exports.register =
async (req, res) => {   const { name,
email, password } = req.body;

  try {     const existingUser = await User.findOne({ email });
if (existingUser)       return res.status(400).json({ message:
"User already exists" });

   const hashedPassword = await bcrypt.hash(password, 10);

   const newUser = new User({ name, email, password: hashedPassword });
await newUser.save();

   res.status(201).json({ message: "User registered successfully ☑" });
```

```
  } catch (error) {     res.status(500).json({ message:
"Server error ❌" });
  }
};

// Login controller exports.login =
async (req, res) => {   const {
email, password } = req.body;

  try {     const user = await User.findOne({ email });     if (!user)
return res.status(400).json({ message: "Invalid email or password"
});

   const isMatch = await bcrypt.compare(password, user.password);
if (!isMatch)       return res.status(400).json({ message: "Invalid
email or password" });

   const token = jwt.sign({ userId: user._id }, process.env.JWT_SECRET, {
expiresIn: "1d",
   });

   res.json({ token, user: { id: user._id, name: user.name, email: user.email } });
  } catch (error) {     res.status(500).json({ message:
"Server error ❌" });
  }
};
```

- **Controllers-expenseController.js:-**

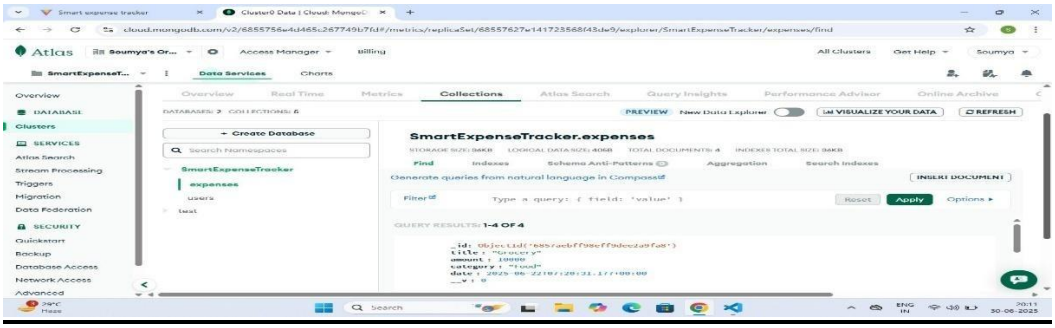
```
const Expense = require("../models/Expense");

exports.getExpenses = async (req, res) => {   try {     const expenses =
await Expense.find({ user: req.userId }).sort({ date: -1 });
res.json(expenses);   } catch (err) {     res.status(500).json({ message:
err.message });
  }
};
```

```
exports.addExpense = async (req, res) => {   const { title, amount, category }
= req.body;   try {     const newExpense = new Expense({ title, amount,
category, user: req.userId });     await newExpense.save();
res.status(201).json(newExpense);
  } catch (err) {     res.status(500).json({
message: err.message });
  }
};

exports.deleteExpense = async (req, res) => {   try {     await
Expense.findOneAndDelete({ _id: req.params.id, user: req.userId });
res.json({ message: "Deleted successfully" });
  } catch (err) {     res.status(500).json({
message: err.message });
  }
};
```

2. Middleware-authMiddleware.js:-

```
const jwt = require("jsonwebtoken");

module.exports = (req, res, next) => {   const token = req.header("x-auth-token");
if (!token) return res.status(401).json({ message: "No token, authorization denied"
});   try {
    const decoded = jwt.verify(token,
process.env.JWT_SECRET);     req.userId = decoded.id;
next();   } catch (err) {     res.status(401).json({ message:
"Token is not valid" });
  }
};
```

3. Models-Expense.js



CODE

```
const mongoose = require('mongoose'); const
expenseSchema = new mongoose.Schema({
  title: {
type:
String,
required:
true
  },
  amount: {
type:
Number,
required:
true   },
category: {
type: String,
required:
true
  },   date: {
type: Date,
```

```
default:
Date.now

}
}
);
module.exports = mongoose.model('Expense', expenseSchema);
```

• **Models-user.js:-** const mongoose = require("mongoose");

```
const userSchema = new mongoose.Schema({
name: { type: String, required: true },   email: {
type: String, required: true, unique: true },
password: { type: String, required: true },
});

module.exports = mongoose.model("User", userSchema);
```

4. Routes-authRouter.js:- const express = require("express"); const router = express.Router(); const { register, login } = require("../controllers/authController");

```
router.post("/register", register);
router.post("/login", login);

module.exports = router;
```

• **Routes-expenseRoutes.js:-**

```
const express =
```

```
require('express'); const router =
express.Router();
const Expense = require('../models/Expense');

// @route GET /api/expenses router.get("/", async (req, res) => {   try {
const expenses = await Expense.find().sort({ date: -1 });
console.log("Fetched expenses count:", expenses.length);
res.json(expenses);   } catch (err) {     console.error("❌ Error fetching
expenses:", err);     return res.status(500).json({ message: "Server Error",
error: err.message });
  }
});

// @route POST /api/expenses
router.post("/", async (req, res) => {   try {
const { title, amount, category } =
req.body;

   if (!title || !amount || !category) {       return
res.status(400).json({ message: "All fields are required" });
   }

   const newExpense = new Expense({
title,
     amount,
category,
   });

   const savedExpense = await newExpense.save();
res.status(201).json(savedExpense);
  } catch (err) {     console.error("❌ Error adding
expense:", err.message);     res.status(500).json({ error:
err.message });
  }
});

// ☑ @route DELETE
/api/expenses/:id router.delete("/:id",
async (req, res) => {   try {
   const deleted = await
Expense.findByIdAndDelete(req.params.id);     if (!deleted) {
return res.status(404).json({ message: "Expense not found" });
```

```
    }
    res.json({ message: "Expense deleted", id: req.params.id });
  } catch (err) {     console.error("❌ Error deleting expense:", err);
return res.status(500).json({ message: "Delete Failed", error: err.message
});
  }
});

module.exports = router;
```

5. **.env code:-** PORT=5000
   MONGO_URI=mongodb+srv://soumyaghosh:1C6llmDF3uvMUbYW@cluster0.oqsg6oy.mongodb.net/?retryWrites=true&w=majority&appName=Cluster0
   JWT_SECRET=SDHFJHFBUIFVHFV

6. **Server.js:-**
```
const express =
require("express"); const
mongoose =
require("mongoose"); const
cors = require("cors"); const
dotenv = require("dotenv");

dotenv.config();

const app = express();

// Middleware
app.use(cors());
app.use(express.json());

// Routes const authRoutes =
require('./routes/authRoutes'); app.use('/api',
authRoutes); app.use("/api/expenses",
require("./routes/expenseRoutes"));

// DB connection
```

```
mongoose.connect(process.env.MONGO_U
RI, {   useNewUrlParser: true,
useUnifiedTopology: true,
})
.then(() => console.log("MongoDB connected ☑"))
.catch((err) => console.error("MongoDB connection error ❌:", err.message));

// Server start const PORT = process.env.PORT || 5000;
app.listen(PORT, () => console.log(Server running on port ${PORT}
🚀 ));
```

6.CONCLUSION

The Smart Expense Tracker is a game-changer for managing your finances. With its intuitive interface and powerful features, it helps you track expenses, set budgets, and achieve financial stability. By using this tracker, you'll gain clarity on your spending habits, make informed financial decisions, and reach your financial goals faster. Take control of your finances today with the Smart Expense Tracker!

7. FUTURE SCOPE & FURTHER ENHANCEMENTS

❖ Future scope:-

Smart Expense Tracker has significant potential for future enhancement and scalability. Some possible areas for future development include:

1. Mobile Application Integration:

Developing Android and iOS apps for on-the-go expense tracking can improve accessibility and user engagement.

2. Bank Account & UPI Integration:

Linking with bank accounts or UPI platforms can allow automatic transaction imports, reducing manual

entry. 3. AI-Based Insights:

Implementing artificial intelligence to analyze spending patterns and provide smart budgeting tips or savings suggestions.

4. Expense Reminders & Alerts:

Setting reminders for bill payments or alerts for exceeding budget limits can promote better financial discipline.

5. Multi-Currency & International Use:

Adding support for various currencies and exchange rates for users managing international finances.

❖ Further enhancement:-

1)Receipt Scanning: Enable users to scan receipts and automatically extract expense data.
2)Predictive Analytics: Use machine learning to predict future expenses and provide proactive budgeting advice.
3)Customizable Dashboards: Offer personalized dashboards with widgets and charts tailored to individual user needs.
4)Multi-User Support: Allow multiple users to share expenses and track joint finances.
5)Expense Forecasting: Provide users with forecasted expenses for upcoming bills and events.
6)Spending Insights: Offer detailed insights into spending habits, including trends and patterns.
7)Budgeting Goals: Allow users to set and track progress towards specific budgeting goals.

8. BIBLIOGRAPHY

1) www.w3schools.com

2) www.youtube.com

3) www.pexels.com

4) www.codepen.io

5) www.google.com

6) www.googlefont.com

7) www.react.our

8) www.codingworld.com