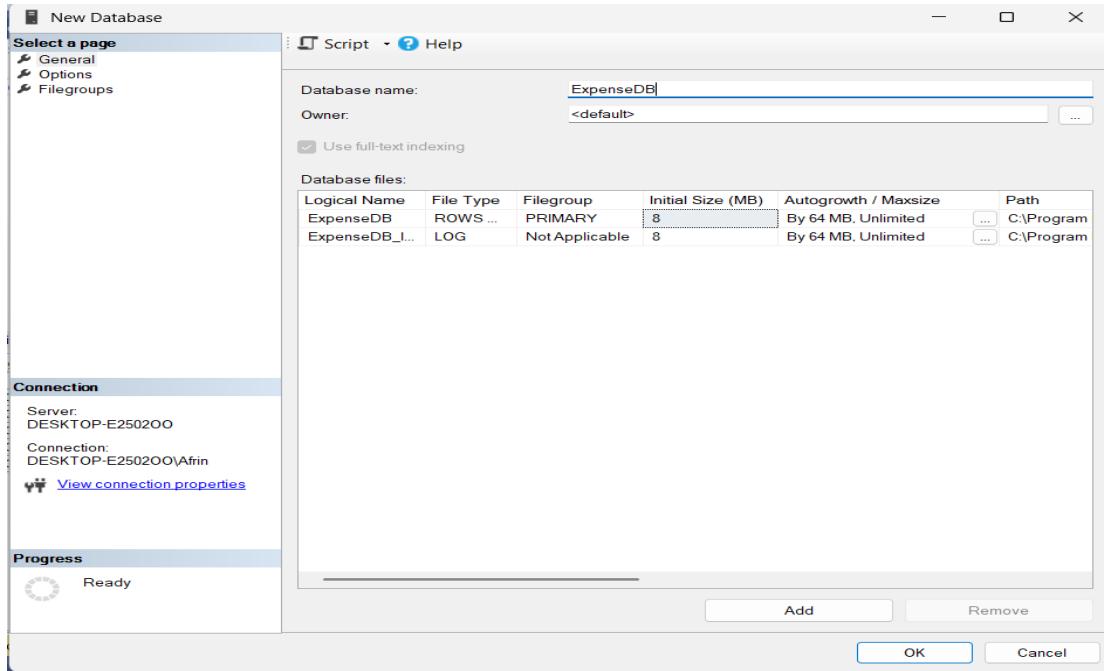


Expense Tracker Project

React & Asp.net Project

Create a database in SQL server Management Studio:



Our database name is Database Name -> ExpenseDB

Install Visual Studio

ASP.NET Core Web API Create a New ASP.NET Core Web API Project

1. Open Visual Studio and select Create a new project.
2. Choose ASP.NET Core Web API and click Next.
3. Enter the project name and location, then click Create.
4. Select the latest .NET version and ensure Use controllers is checked.
5. Click Create to generate the project structure.

Install Required

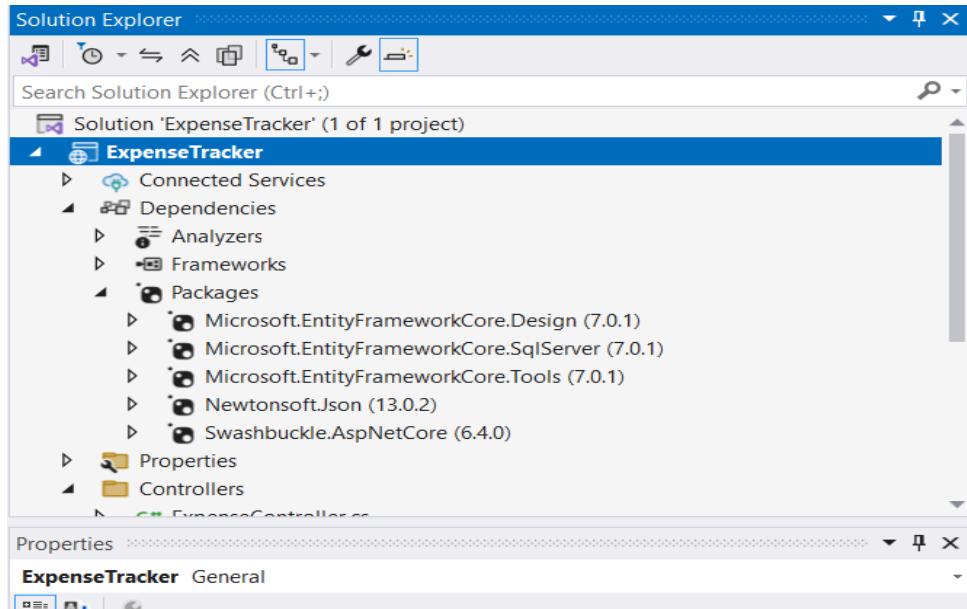
Packages Open Manage NuGet Packages by right-clicking your project Install the packages below

Microsoft.EntityFrameworkCore.Design

Microsoft.EntityFrameworkCore.SqlServer

Microsoft.EntityFrameworkCore.Tools

Newtonsoft.Json



Check the package requirements

Create model folder. give name Models

First Step Select the Model Folder Right Click and Create the Class **Expense.cs**

```
using System.ComponentModel.DataAnnotations;
using System.ComponentModel.DataAnnotations.Schema;
namespace ExpenseTracker.Model
{
    public class Expense
    {
        [Key]
        public int Id { get; set; }
        public DateTime Date { get; set; }
        public string Description { get; set; }
        public string Category { get; set; }
        [Column(TypeName = "decimal(18,2)")]
        public decimal Amount { get; set; }
        public string PaymentMethod { get; set; }
    }
}
```

}

}

The screenshot shows the Visual Studio IDE interface with the following details:

- Solution Explorer:** Shows the solution named "ExpenseTracker" with its projects: "ExpenseTracker" (selected), "Connected Services", "Dependencies", "Analyzers", "Frameworks", "Packages" (containing Microsoft.EntityFrameworkCore, Microsoft.EntityFrameworkCore.Fz, Microsoft.EntityFrameworkCore.Fr, Microsoft.EntityFrameworkCore.Fr, Newtonsoft.Json, and Swashbuckle.AspNetCore), and "Properties".
- Code Editor:** Displays the file "Expense.cs" containing the code for the "Expense" entity class. The class has properties: Id, Date, Description, Category, Amount, and PaymentMethod. The "Description" property is underlined in red, indicating a potential issue.
- Status Bar:** Shows the zoom level at 128%, and the current position as Line 10, Column 10.
- Task List:** Located at the bottom left, it lists tasks such as "2025122813113...italCreate.cs", "ExpenseController.cs", "Program.cs", "appsettings.json", and "ExpenseDbContext.cs".
- Properties Window:** Located on the right side of the interface.

ExpenseDbContext Class:

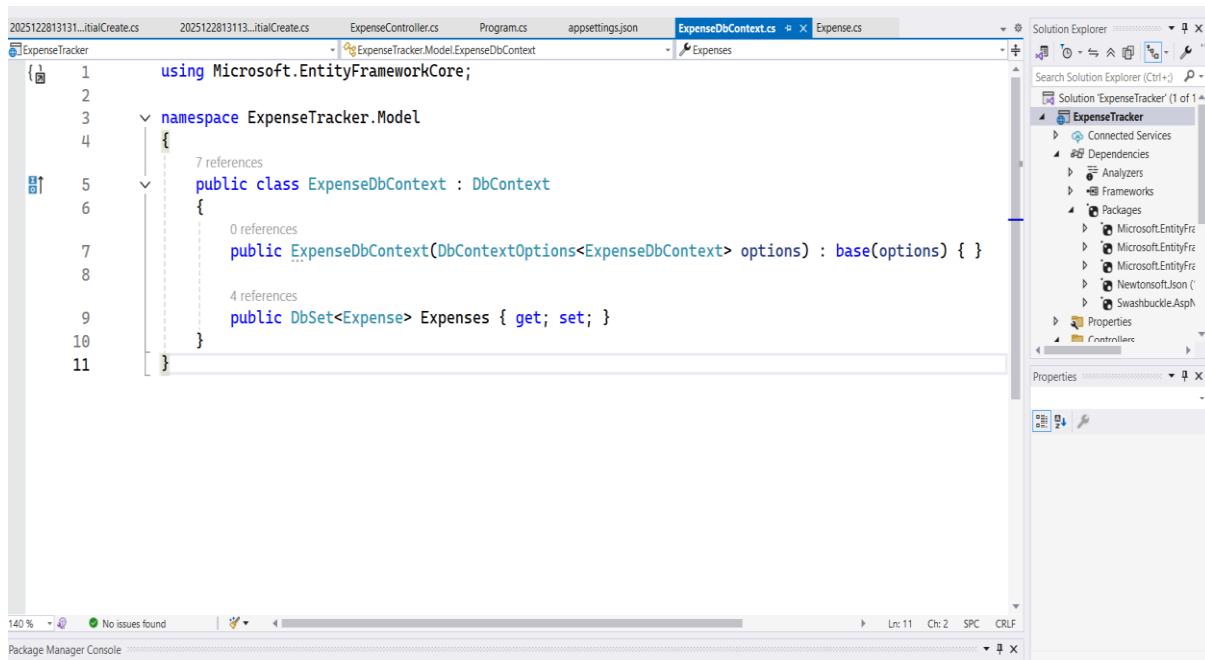
Create another model class – **ExpenseDbContext.cs**

```
using Microsoft.EntityFrameworkCore;
```

```
namespace ExpenseTracker.Model

{
    public class ExpenseDbContext : DbContext
    {
        public ExpenseDbContext(DbContextOptions<ExpenseDbContext> options) :
        base(options) { }

        public DbSet<Expense> Expenses { get; set; }
    }
}
```



```
202512281313...itaiCreate.cs 202512281313...itaiCreate.cs ExpenseController.cs Program.cs appsettings.json ExpenseDbContext.cs Expense.cs
ExpenseTracker
1  using Microsoft.EntityFrameworkCore;
2
3  namespace ExpenseTracker.Model
4  {
5      public class ExpenseDbContext : DbContext
6      {
7          public ExpenseDbContext(DbContextOptions<ExpenseDbContext> options) : base(options) { }
8
9          public DbSet<Expense> Expenses { get; set; }
10     }
11 }
```

Search Solution Explorer (Ctrl+F) Solution Explorer
Solution ExpenseTracker (1 of 1)
ExpenseTracker
Connected Services
Dependencies
Analyzers
Frameworks
Packages
Microsoft.EntityFrameworkCore
Microsoft.EntityFrameworkCore
Newtonsoft.Json
Swashbuckle.AspNetCore
Properties
Controllers

Properties

Configure the Database Connection – (appsettings.json)

```
{
    "Logging": {
        "LogLevel": {
            "Default": "Information",
            "Microsoft.AspNetCore": "Warning"
        }
    },
    "AllowedHosts": "*",
    "ConnectionStrings": {
        "DefaultConnection": "Server=DESKTOPE2502OO;Database=ExpenseDB;Trusted_Connection=True;TrustServerCertificate=True;MultipleActiveResultSets=true"
    }
}
```

```
2025122813131...italCreate.cs 2025122813113..italCreate.cs ExpenseController.cs Program.cs appsettings.json* ExpenseDbContext.cs Expense.cs
Schema: https://www.schemastore.org/appsettings.json
1   < Logging> {
2     < LogLevel> {
3       < Default>: "Information",
4       < Microsoft.AspNetCore>: "Warning"
5     }
6   },
7   < AllowedHosts>: "*",
8   < ConnectionStrings> {
9     < DefaultConnection>: "Server=DESKTOP-E250200;Database=ExpenseDB;
10    [Trusted_Connection=True;TrustServerCertificate=True;MultipleActiveResultSets=true]"
11  }
12}
13}
14
```

Program.cs

Establish the Database Connection

Add these Context inside the Program.cs file

```
// Database Connection
builder.Services.AddDbContext<ExpenseDbContext>(options =>
  options.UseSqlServer(builder.Configuration.GetConnectionString("DefaultConnection")));

// Allow React to talk to this API
app.UseCors(policy => policy.AllowAnyHeader()
  .AllowAnyMethod()
  .SetIsOriginAllowed(origin => true)
  .AllowCredentials());
```

Full code structure of program.cs -

```
using Microsoft.EntityFrameworkCore;
using ExpenseTracker.Model;

var builder = WebApplication.CreateBuilder(args);

// Add services to the container.
builder.Services.AddControllers();
builder.Services.AddEndpointsApiExplorer();
builder.Services.AddSwaggerGen();

// Database Connection
builder.Services.AddDbContext<ExpenseDbContext>(options =>
    options.UseSqlServer(builder.Configuration.GetConnectionString("DefaultConnection")));

var app = builder.Build();

// Allow React to talk to this API
app.UseCors(policy => policy.AllowAnyHeader()
    .AllowAnyMethod()
    .SetIsOriginAllowed(origin => true)
    .AllowCredentials());

if (app.Environment.IsDevelopment())
{
    app.UseSwagger();
    app.UseSwaggerUI();
}

app.UseHttpsRedirection();
app.UseAuthorization();
app.MapControllers();
app.Run();
```

```

    \ using Microsoft.EntityFrameworkCore;
    \ using ExpenseTracker.Model;

    var builder = WebApplication.CreateBuilder(args);

    // Add services to the container.
    builder.Services.AddControllers();
    builder.Services.AddEndpointsApiExplorer();
    builder.Services.AddSwaggerGen();

    // Database Connection
    builder.Services.AddDbContext<ExpenseDbContext>(options =>
        options.UseSqlServer(builder.Configuration.GetConnectionString("DefaultConnection")));

    var app = builder.Build();

    // Allow React to talk to this API
    app.UseCors(policy => policy.AllowAnyHeader()
        .AllowAnyMethod()
        .SetIsOriginAllowed(origin => true)
        .AllowCredentials());
    ...

    \ if (app.Environment.IsDevelopment())
    {
        \ app.UseSwagger();
        \ app.UseSwaggerUI();
    }

    app.UseHttpsRedirection();
    app.UseAuthorization();
    app.MapControllers();
    app.Run();

```

Add Controller

Rightclick

Controller -> Add-> new scaffold item -> mvc empty controller-> **ExpenseController.cs**

ExpenseController.cs

```

using Microsoft.AspNetCore.Mvc;
using Microsoft.EntityFrameworkCore;
using ExpenseTracker.Model;

namespace ExpenseTracker.Controllers
{
    [Route("api/[controller]")]
    [ApiController]
    public class ExpenseController : ControllerBase

```

```
{  
    private readonly ExpenseDbContext _context;  
  
    public ExpenseController(ExpenseDbContext context)  
    {  
        _context = context;  
    }  
  
    [HttpGet]  
    [Route("GetExpenses")]  
    public async Task<IEnumerable<Expense>> GetExpenses()  
    {  
        return await _context.Expenses.ToListAsync();  
    }  
  
    [HttpPost]  
    [Route("AddExpense")]  
    public async Task<Expense> AddExpense(Expense objExpense)  
    {  
        _context.Expenses.Add(objExpense);  
        await _context.SaveChangesAsync();  
        return objExpense;  
    }  
  
    [HttpPatch]  
    [Route("UpdateExpense/{id}")]  
    public async Task<Expense> UpdateExpense([FromRoute] int id, [FromBody] Expense objExpense)  
    {  
        // Ensure the ID in the object matches the ID in the URL  
        objExpense.Id = id;  
        _context.Entry(objExpense).State = EntityState.Modified;  
        await _context.SaveChangesAsync();  
        return objExpense;  
    }  
}
```

```

[HttpDelete]
[Route("DeleteExpense/{id}")]
public async Task<bool> DeleteExpense(int id)
{
    var expense = await _context.Expenses.FindAsync(id);
    if(expense != null)
    {
        _context.Expenses.Remove(expense);
        await _context.SaveChangesAsync();
        return true;
    }
    return false;
}
}

```

Below the code output:

```

using Microsoft.AspNetCore.Mvc;
using Microsoft.EntityFrameworkCore;
using ExpenseTracker.Model;

namespace ExpenseTracker.Controllers
{
    [Route("api/[controller]")]
    [ApiController]
    public class ExpenseController : ControllerBase
    {
        private readonly ExpenseDbContext _context;

        public ExpenseController(ExpenseDbContext context)
        {
            _context = context;
        }

        [HttpGet]
        [Route("GetExpenses")]
        public async Task<IEnumerable<Expense>> GetExpenses()
        {
            return await _context.Expenses.ToListAsync();
        }
    }
}

```

```
[HttpPost]
[Route("AddExpense")]
0 references
public async Task<Expense> AddExpense(Expense objExpense)
{
    _context.Expenses.Add(objExpense);
    await _context.SaveChangesAsync();
    return objExpense;
}

[HttpPatch]
[Route("UpdateExpense/{id}")]
0 references
public async Task<Expense> UpdateExpense([FromRoute] int id, [FromBody] Expense objExpense)
{
    // Ensure the ID in the object matches the ID in the URL
    objExpense.Id = id;
    _context.Entry(objExpense).State = EntityState.Modified;
    await _context.SaveChangesAsync();
    return objExpense;
}
```

```
[HttpDelete]
[Route("DeleteExpense/{id}")]
0 references
public async Task<bool> DeleteExpense(int id)
{
    var expense = await _context.Expenses.FindAsync(id);
    if (expense != null)
    {
        _context.Expenses.Remove(expense);
        await _context.SaveChangesAsync();
        return true;
    }
    return false;
}
```

Apply Migrations and Create the Database

Tools->NuGet Package Manager->Package Manager Console

Run command

Run these commands in the Package Manager Console –

```
Add-Migration InitialCreate
```

```
Update-Database
```

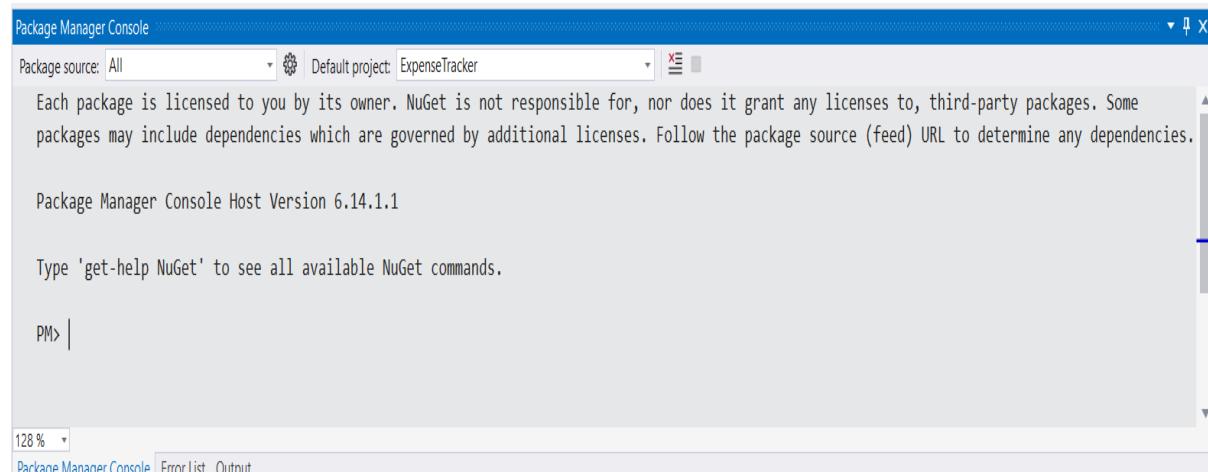
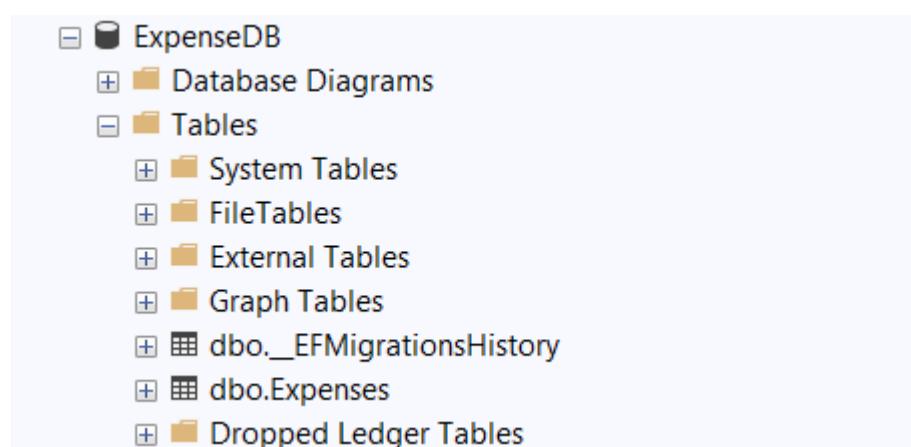


Table will be created in SQL our database inside table –



RUN THE SERVER -

API CONNECTION WILL OPEN.

The screenshot shows the Swagger UI interface for the ExpenseTracker API. At the top, there's a header with the Swagger logo and a dropdown menu labeled "Select a definition" set to "ExpenseTracker v1". Below the header, the title "ExpenseTracker 1.0 OAS" is displayed, along with the URL "https://localhost:7182/swagger/v1/swagger.json". The main content area is organized into sections: "Expense" and "WeatherForecast". The "Expense" section contains four methods: "GET /api/Expense/GetExpenses" (blue button), "POST /api/Expense/AddExpense" (green button), "PATCH /api/Expense/UpdateExpense/{id}" (green button), and "DELETE /api/Expense/DeleteExpense/{id}" (red button). The "WeatherForecast" section contains one method: "GET /WeatherForecast" (blue button). At the bottom, there's a "Schemas" section with a "DateOnly" schema entry.

React Frontend –

Open vs code terminal creates the react application

```
npx create-react-app frontend
```

```
npm start
```

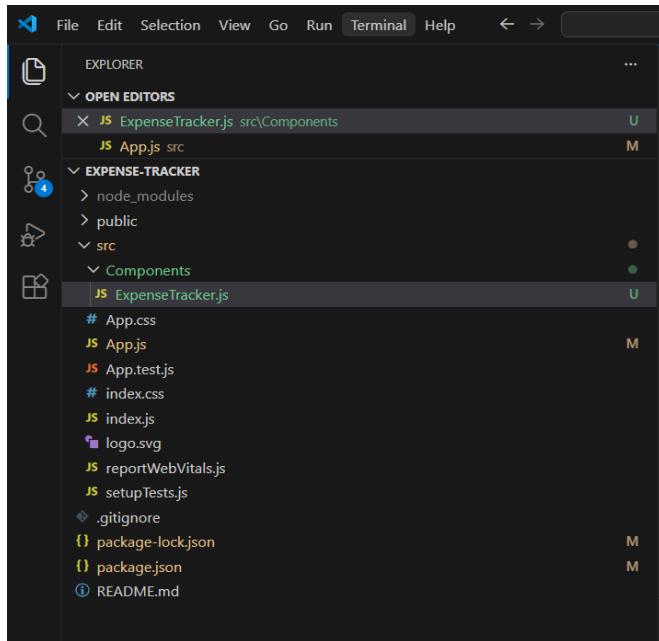
Install the bootstrap command

```
npm i bootstrap (or) npm list bootsrap
```

Fetch The Api/Backend Connection Install the Command

```
npm i axios
```

Open a New Component folder -> give **ExpenseTracker.js** file



App.js

```
import 'bootstrap/dist/css/bootstrap.min.css';

import ExpenseTracker from './Components/ExpenseTracker';

function App() {
  return (
    <div className="App">
      <ExpenseTracker />
    </div>
  );
}

export default App;
```

```
JS ExpenseTracker.js U JS App.js M X
src > JS App.js > ...
1 import 'bootstrap/dist/css/bootstrap.min.css';
2 import ExpenseTracker from './Components/ExpenseTracker';
3
4 function App() {
5   return (
6     <div className="App">
7       <ExpenseTracker />
8     </div>
9   );
10 }
11
12 export default App;
13
```

ExpenseTracker.js file

Import Statements –

```
import axios from "axios";
import { useEffect, useState } from "react";
```

State Variables –

```
function ExpenseTracker() {
  const [id, setId] = useState("");
  const [date, setDate] = useState("");
  const [description, setDescription] = useState("");
  const [category, setCategory] = useState("");
  const [amount, setAmount] = useState("");
  const [paymentMethod, setPaymentMethod] = useState("");
  const [expenses, setExpenses] = useState([]);
```

Component Load –

```
useEffect(() => {  
  (async () => await Load())();  
}, []);
```

Read Operation –

```
async function Load() {  
  const result = await axios.get("https://localhost:7182/api/Expense/GetExpenses");  
  setExpenses(result.data);  
}
```

Create Operation –

```
async function save(event) {  
  event.preventDefault();  
  if (!date || !amount) return alert("Please fill in Date and Amount");  
  await axios.post("https://localhost:7182/api/Expense/AddExpense", {  
    date, description, category, amount, paymentMethod  
  });  
  alert("Expense Saved");  
  clear();  
  Load();  
}
```

Update Operation –

```
async function update(event) {  
    event.preventDefault();  
    await axios.patch("https://localhost:7182/api/Expense/UpdateExpense/" + id, {  
        id, date, description, category, amount, paymentMethod  
    });  
    alert("Expense Updated");  
    clear();  
    Load();  
}
```

Delete Operation –

```
async function deleteExp(id) {  
    if (window.confirm("Are you sure you want to delete this record?")) {  
        await axios.delete("https://localhost:7182/api/Expense/DeleteExpense/" + id);  
        Load();  
    }  
}
```

Edit Operation –

```
function edit(exp) {  
    setId(exp.id);  
    setDate(exp.date.split('T')[0]);  
    setDescription(exp.description);  
    setCategory(exp.category);  
    setAmount(exp.amount);  
    setPaymentMethod(exp.paymentMethod);  
}
```

Clear Operation –

```
function clear() {  
    setId(""); setDate(""); setDescription(""); setCategory(""); setAmount("");  
    setPaymentMethod("");  
}
```

Calculating Total Expenses –

```
const totalAmount = expenses.reduce((sum, item) => sum + parseFloat(item.amount || 0), 0);
```

UI Design code –

```
return (  
    <div style={{  
        backgroundColor: "#f8f9fa",  
        minHeight: "100vh",  
        padding: "40px 0",  
        color: "#343a40"  
    }}>  
        <div className="container">  
            {/* Header section */}  
            <div className="mb-5">  
                <h2 style={{ fontWeight: "700", color: "#2c3e50", borderLeft: "5px solid #4e73df",  
                paddingLeft: "15px" }}>  
                    Expense Tracker  
                </h2>  
                <p className="text-muted">Tracking my expenses and manage my finances.</p>  
            </div>  
    
```

```
<div className="row g-4">
  {/* Sidebar Form */}
  <div className="col-lg-4">
    <div className="card border-0 shadow-sm p-4" style={{ borderRadius: "12px" }}>
      <h5 className="mb-4" style={{ fontWeight: "600" }}>{id ? "Update Transaction" : "New Transaction"}</h5>
      <form>
        <div className="mb-3">
          <label className="small fw-bold text-muted mb-1">DATE</label>
          <input type="date" className="form-control bg-light border-0" value={date} onChange={(e)=>setDate(e.target.value)} />
        </div>
        <div className="mb-3">
          <label className="small fw-bold text-muted mb-1">CATEGORY</label>
          <select className="form-select bg-light border-0" value={category} onChange={(e)=>setCategory(e.target.value)}>
            <option value="">Select Category</option>
            <option value="Food&Groceries">Food & Groceries</option>
            <option value="Rent">Rent</option>
            <option value="Utilities">Utilities</option>
            <option value="Health">Health</option>
            <option value="Education">Education & Fees</option>
            <option value="Transportation">Transportation</option>
            <option value="Maintenance">Maintenance & Repairs</option>
            <option value="Entertainment">Entertainment</option>
            <option value="Shopping">Shopping & Lifestyle</option>
            <option value="EMI & Loans">EMI & Loans</option>
            <option value="Investments">Investments</option>
            <option value="Others">Other</option>
          </select>
        </div>
      </form>
    </div>
  </div>
</div>
```

```

        </div>

        <div className="mb-3">
            <label className="small fw-bold text-muted mb-1">AMOUNT (INR)</label>
            <input type="number" className="form-control bg-light border-0"
placeholder="0.00" value={amount} onChange={(e)=>setAmount(e.target.value)} />
        </div>

        <div className="mb-3">
            <label className="small fw-bold text-muted mb-1">DESCRIPTION</label>
            <textarea className="form-control bg-light border-0" rows="2"
value={description} onChange={(e)=>setDescription(e.target.value)}></textarea>
        </div>

        <div className="mb-4">
            <label className="small fw-bold text-muted mb-1">PAYMENT
METHOD</label>
            <input type="text" className="form-control bg-light border-0"
placeholder="Bank Transfer / Card" value={paymentMethod}
onChange={(e)=>setPaymentMethod(e.target.value)} />
        </div>

<div className="d-grid gap-2">
    {!id ? (
        <button className="btn btn-primary shadow-sm" onClick={save} style={{ backgroundColor: "#4e73df", border: "none", padding: "12px" }}>Add Entry</button>
    ) : (
        <button className="btn btn-dark shadow-sm" onClick={update} style={{ padding: "12px" }}>Update Entry</button>
    )}
    <button type="button" className="btn btn-outline-secondary btn-sm border-0"
onClick={clear}>Reset Form</button>
</div>
</form>
</div>
</div>

```

```

/* Table Area */

<div className="col-lg-8">
  /* Summary Banner */
  <div className="card border-0 shadow-sm p-4 mb-4" style={{ backgroundColor: "#ffffff", borderRadius: "12px" }}>
    <div className="row align-items-center">
      <div className="col border-end">
        <span className="small text-muted text-uppercase fw-bold">Total Expenditures</span>
        <h3 className="mb-0" style={{ color: "#e74c3c", fontWeight: "700" }}>$ {totalAmount.toLocaleString(undefined, {minimumFractionDigits: 2})}</h3>
      </div>
      <div className="col ps-4">
        <span className="small text-muted text-uppercase fw-bold">Active Records</span>
        <h3 className="mb-0" style={{ color: "#2c3e50", fontWeight: "700" }}>{expenses.length}</h3>
      </div>
    </div>
  </div>
</div>

/* Data Table */

<div className="card border-0 shadow-sm" style={{ borderRadius: "12px", overflow: "hidden" }}>
  <div className="table-responsive">
    <table className="table table-hover mb-0">
      <thead style={{ backgroundColor: "#f1f4f8" }}>
        <tr>
          <th className="p-3 small text-muted">TRANSACTION</th>
          <th className="p-3 small text-muted">CATEGORY</th>
          <th className="p-3 small text-muted">AMOUNT</th>
        </tr>
      </thead>

```

```

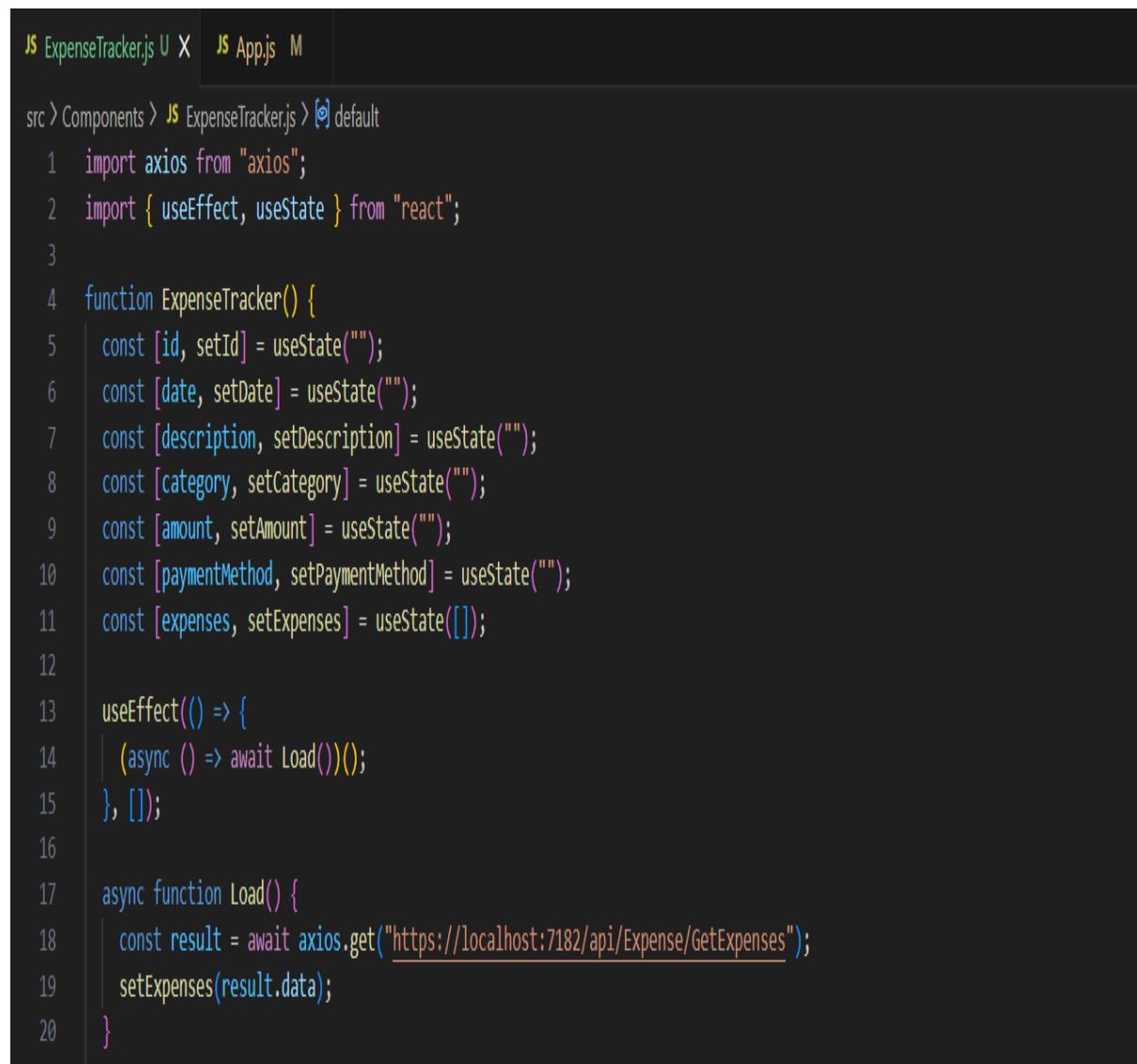
<th className="p-3 small text-muted text-end">CONTROLS</th>
</tr>
</thead>
<tbody className="bg-white">
  {expenses.map((exp) => (
    <tr key={exp.id}>
      <td className="p-3">
        <div className="fw-bold">{exp.description || "Unspecified"}</div>
        <div className="text-muted small">{new Date(exp.date).toLocaleDateString()} | {exp.paymentMethod}</div>
      </td>
      <td className="p-3">
        <span className="badge bg-light text-secondary border px-3 py-2 fw-normal">
          {exp.category}
        </span>
      </td>
      <td className="p-3 fw-bold">
        ${parseFloat(exp.amount).toFixed(2)}
      </td>
      <td className="p-3 text-end">
        <button className="btn btn-sm btn-outline-primary me-2 border-0" onClick={() => edit(exp)}>Edit</button>
        <button className="btn btn-sm btn-outline-danger border-0" onClick={() => deleteExp(exp.id)}>Delete</button>
      </td>
    </tr>
  )))
</tbody>
</table>
</div>
</div>

```

```
</div>
</div>
</div>
</div>
);
}

export default ExpenseTracker;
```

In VS Code -



The screenshot shows the VS Code interface with the tab bar at the top. The active tab is 'ExpenseTracker.js' (highlighted in yellow), and the other tab is 'App.js'. The code editor below displays the 'ExpenseTracker.js' file. The code is a functional component named 'ExpenseTracker' that uses React's useState and useEffect hooks to manage state and load data from an API.

```
src > Components > ExpenseTracker.js > [?] default
1 import axios from "axios";
2 import { useEffect, useState } from "react";
3
4 function ExpenseTracker() {
5   const [id, setId] = useState("");
6   const [date, setDate] = useState("");
7   const [description, setDescription] = useState("");
8   const [category, setCategory] = useState("");
9   const [amount, setAmount] = useState("");
10  const [paymentMethod, setPaymentMethod] = useState("");
11  const [expenses, setExpenses] = useState([]);
12
13  useEffect(() => {
14    (async () => await Load())();
15  }, []);
16
17  async function Load() {
18    const result = await axios.get("https://localhost:7182/api/Expense/GetExpenses");
19    setExpenses(result.data);
20  }
}
```

```

async function save(event) {
  event.preventDefault();
  if (!date || !amount) return alert("Please fill in Date and Amount");
  await axios.post("https://localhost:7182/api/Expense/AddExpense", {
    date, description, category, amount, paymentMethod
  });
  alert("Expense Saved");
  clear();
  Load();
}

async function update(event) {
  event.preventDefault();
  await axios.patch("https://localhost:7182/api/Expense/UpdateExpense/" + id, {
    id, date, description, category, amount, paymentMethod
  });
  alert("Expense Updated");
  clear();
  Load();
}

async function deleteExp(id) {
  if (window.confirm("Are you sure you want to delete this record?")) {
    await axios.delete("https://localhost:7182/api/Expense/DeleteExpense/" + id);
    Load();
  }
}

```

```

function edit(exp) {
  setId(exp.id);
  setDate(exp.date.split('T')[0]);
  setDescription(exp.description);
  setCategory(exp.category);
  setAmount(exp.amount);
  setPaymentMethod(exp.paymentMethod);
}

function clear() {
  setId(""); setDate(""); setDescription(""); setCategory(""); setAmount(""); setPaymentMethod("");
}

const totalAmount = expenses.reduce((sum, item) => sum + parseFloat(item.amount || 0), 0);

```

```

return (
  <div style={{
    backgroundColor: "#f8f9fa",
    minHeight: "100vh",
    padding: "40px 0",
    color: "#343a40"
  }}>
    <div className="container">
      /* Header section */
      <div className="mb-5">
        <h2 style={{ fontWeight: "700", color: "#2c3e50", borderLeft: "5px solid #4e73df", paddingLeft: "15px" }>
          Expense Tracker
        </h2>
        <p className="text-muted">Tracking my expenses and manage my finances.</p>
      </div>

      <div className="row g-4">
        /* Sidebar Form */
        <div className="col-lg-4">
          <div className="card border-0 shadow-sm p-4" style={{ borderRadius: "12px" }}>
            <h5 className="mb-4" style={{ fontWeight: "600" }}>{id ? "Update Transaction" : "New Transaction"}</h5>
            <form>
              <div className="mb-3">
                <label className="small fw-bold text-muted mb-1">DATE</label>
                <input type="date" className="form-control bg-light border-0" value={date} onChange={(e)=>setDate(e.target.value)} />
              </div>
              <div className="mb-3">
                <label className="small fw-bold text-muted mb-1">CATEGORY</label>
              </div>
            </form>
          </div>
        </div>
      </div>
    </div>
  </div>
)

```

```

<select className="form-select bg-light border-0" value={category} onChange={(e)=>setCategory(e.target.value)}>
  <option value="">Select Category</option>
  <option value="Food&Groceries">Food & Groceries</option>
  <option value="Rent">Rent</option>
  <option value="Utilities">Utilities</option>
  <option value="Health">Health</option>
  <option value="Education">Education & Fees</option>
  <option value="Transportation">Transportation</option>
  <option value="Maintenance">Maintenance & Repairs</option>
  <option value="Entertainment">Entertainment</option>
  <option value="Shopping">Shopping & Lifestyle</option>
  <option value="EMI & Loans">EMI & Loans</option>
  <option value="Investments">Investments</option>
  <option value="Others">Other</option>
</select>
</div>
<div className="mb-3">
  <label className="small fw-bold text-muted mb-1">AMOUNT (INR)</label>
  <input type="number" className="form-control bg-light border-0" placeholder="0.00" value={amount} onChange={(e)=>setAmount(e.target.value)} />
</div>
<div className="mb-3">
  <label className="small fw-bold text-muted mb-1">DESCRIPTION</label>
  <textarea className="form-control bg-light border-0" rows="2" value={description} onChange={(e)=>setDescription(e.target.value)}></textarea>
</div>
<div className="mb-4">
  <label className="small fw-bold text-muted mb-1">PAYMENT METHOD</label>
  <input type="text" className="form-control bg-light border-0" placeholder="Bank Transfer / Card" value={paymentMethod} onChange={(e)=>setPaymentMethod(e.target.value)} />
</div>

```

```

<div className="d-grid gap-2">
  {id ? (
    <button className="btn btn-primary shadow-sm" onClick={save} style={{ backgroundColor: "#4e73df", border: "none", padding: "12px" }}>Add Entry</button>
  ) : (
    <button className="btn btn-dark shadow-sm" onClick={update} style={{ padding: "12px" }}>Update Entry</button>
  )}
  <button type="button" className="btn btn-outline-secondary btn-sm border-0" onClick={clear}>Reset Form</button>
</div>
</form>
</div>
</div>

/* Table Area */
<div className="col-lg-8">
  /* Summary Banner */
  <div className="card border-0 shadow-sm p-4 mb-4" style={{ backgroundColor: "#fffffe", borderRadius: "12px" }}>
    <div className="row align-items-center">
      <div className="col border-end">
        <span className="small text-muted uppercase fw-bold">Total Expenditures</span>
        <h3 className="mb-0" style={{ color: "#e74c3c", fontWeight: "700" }}>${totalAmount.toLocaleString(undefined, {minimumFractionDigits: 2})}</h3>
      </div>
      <div className="col ps-4">
        <span className="small text-muted uppercase fw-bold">Active Records</span>
        <h3 className="mb-0" style={{ color: "#2c3e50", fontWeight: "700" }}>{expenses.length}</h3>
      </div>
    </div>
  </div>
</div>

```

```

/* Data Table */


<div className="table-responsive">
    <table className="table table-hover mb-0">
      <thead style={{ backgroundColor: "#f1f4f8" }}>
        <tr>
          <th className="p-3 small text-muted">TRANSACTION</th>
          <th className="p-3 small text-muted">CATEGORY</th>
          <th className="p-3 small text-muted">AMOUNT</th>
          <th className="p-3 small text-muted text-end">CONTROLS</th>
        </tr>
      </thead>
      <tbody className="bg-white">
        {expenses.map((exp) => (
          <tr key={exp.id}>
            <td className="p-3">
              <div className="fw-bold">{exp.description || "Unspecified"}</div>
              <div className="text-muted small">{new Date(exp.date).toLocaleDateString()} | {exp.paymentMethod}</div>
            </td>
            <td className="p-3">
              <span className="badge bg-light text-secondary border px-2 py-2 fw-normal">
                {exp.category}
              </span>
            </td>
            <td className="p-3 fw-bold">
              ${parseFloat(exp.amount).toFixed(2)}
            </td>
            <td className="p-3 text-end">
              <button className="btn btn-sm btn-outline-primary me-2 border-0" onClick={() => edit(exp)}>Edit</button>
              <button className="btn btn-sm btn-outline-danger border-0" onClick={() => deleteExp(exp.id)}>Delete</button>
            </td>
          </tr>
        )));
      </tbody>
    </table>
  </div>


```

```

    </tbody>
  </div>
  </div>
  </div>
  </div>
  </div>
);
}

export default ExpenseTracker;

```

Output –

```

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS

Compiled successfully!

You can now view expense-tracker in the browser.

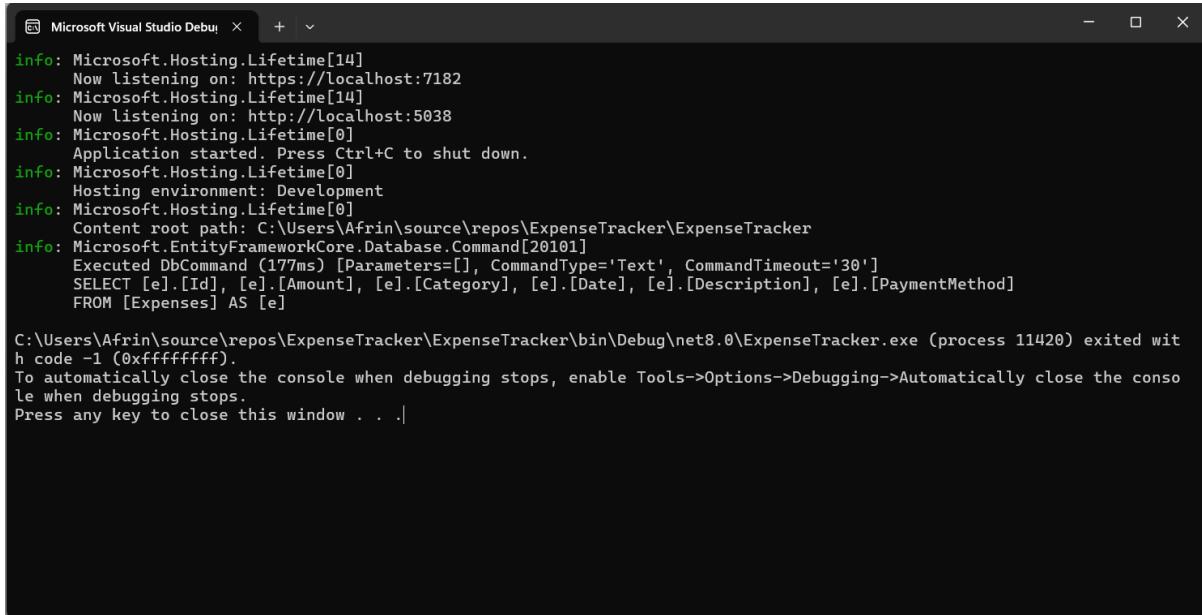
Local:          http://localhost:3000
On Your Network: http://192.168.8.103:3000

Note that the development build is not optimized.
To create a production build, use npm run build.

webpack compiled successfully

```

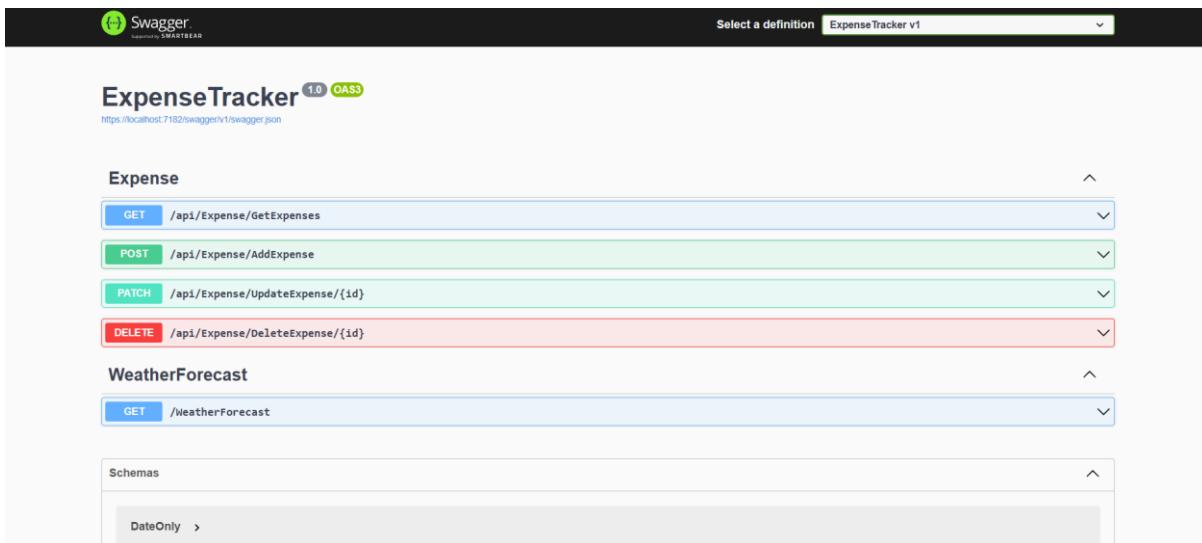
Backend server running :



```
Microsoft Visual Studio Debug + ^ - X

info: Microsoft.Hosting.Lifetime[14]
      Now listening on: https://localhost:7182
info: Microsoft.Hosting.Lifetime[14]
      Now listening on: http://localhost:5038
info: Microsoft.Hosting.Lifetime[0]
      Application started. Press Ctrl+C to shut down.
info: Microsoft.Hosting.Lifetime[0]
      Hosting environment: Development
info: Microsoft.Hosting.Lifetime[0]
      Content root path: C:\Users\Afrin\source\repos\ExpenseTracker\ExpenseTracker
info: Microsoft.EntityFrameworkCore.Database.Command[20101]
      Executed DbCommand (177ms) [Parameters=[], CommandType='Text', CommandTimeout='30']
      SELECT [e].[Id], [e].[Amount], [e].[Category], [e].[Date], [e].[Description], [e].[PaymentMethod]
      FROM [Expenses] AS [e]

C:\Users\Afrin\source\repos\ExpenseTracker\ExpenseTracker\bin\Debug\net8.0\ExpenseTracker.exe (process 11420) exited with code -1 (0xffffffff).
To automatically close the console when debugging stops, enable Tools->Options->Debugging->Automatically close the console when debugging stops.
Press any key to close this window . . .
```



The screenshot shows the Swagger UI interface for the ExpenseTracker API. At the top, it says "Select a definition" and "ExpenseTracker v1". Below that, it displays the "Expense" section with four operations: GET /api/Expense/GetExpenses, POST /api/Expense/AddExpense, PATCH /api/Expense/UpdateExpense/{id}, and DELETE /api/Expense/DeleteExpense/{id}. The "DELETE" operation is highlighted with a red background. Below the "Expense" section is the "WeatherForecast" section with one operation: GET /WeatherForecast. At the bottom left is a "Schemas" section containing a "DateOnly" schema.

Developed Page –

Expense Tracker

Tracking my expenses and manage my finances.

New Transaction

DATE dd-mm-yyyy

CATEGORY Select Category

AMOUNT (INR) 0.00

DESCRIPTION

PAYMENT METHOD Bank Transfer / Card

Add Entry

Reset Form

TOTAL EXPENDITURES \$31,649.00

ACTIVE RECORDS 5

TRANSACTION	CATEGORY	AMOUNT	CONTROLS
Monthly Grocery 12/20/2025 Cash	Food&Groceries	\$1500.00	Edit Delete
Netflix Subscription 12/27/2025 UPI	Entertainment	\$149.00	Edit Delete
Electricity Bill 12/2/2025 Debit Card	Utilities	\$3000.00	Edit Delete
Apartment Rent 12/2/2025 UPI	Rent	\$12000.00	Edit Delete
School Fees 12/29/2025 Debit Card	Education	\$15000.00	Edit Delete

Adding Details –

Expense Tracker

Tracking my expenses and manage my finances.

New Transaction

DATE 15-12-2025

CATEGORY Health

AMOUNT (INR) 400

DESCRIPTION Went to hospital for fever

PAYMENT METHOD Cash

Add Entry

Reset Form

After clicking Add Entry Button –

The screenshot shows the 'Expense Tracker' application interface. On the left, there is a form for 'New Transaction' with fields for DATE (15-12-2025), CATEGORY (Health), AMOUNT (INR) (400), DESCRIPTION (Went to hospital for fever), and PAYMENT METHOD (Cash). A blue 'Add Entry' button is at the bottom of the form. On the right, a table displays 'ACTIVE RECORDS' (5) with columns for TRANSACTION, CATEGORY, AMOUNT, and CONTROLS (Edit, Delete). A modal window titled 'localhost:3001 says' shows the message 'Expense Saved' with an 'OK' button. At the top, the browser address bar shows 'localhost:3001'.

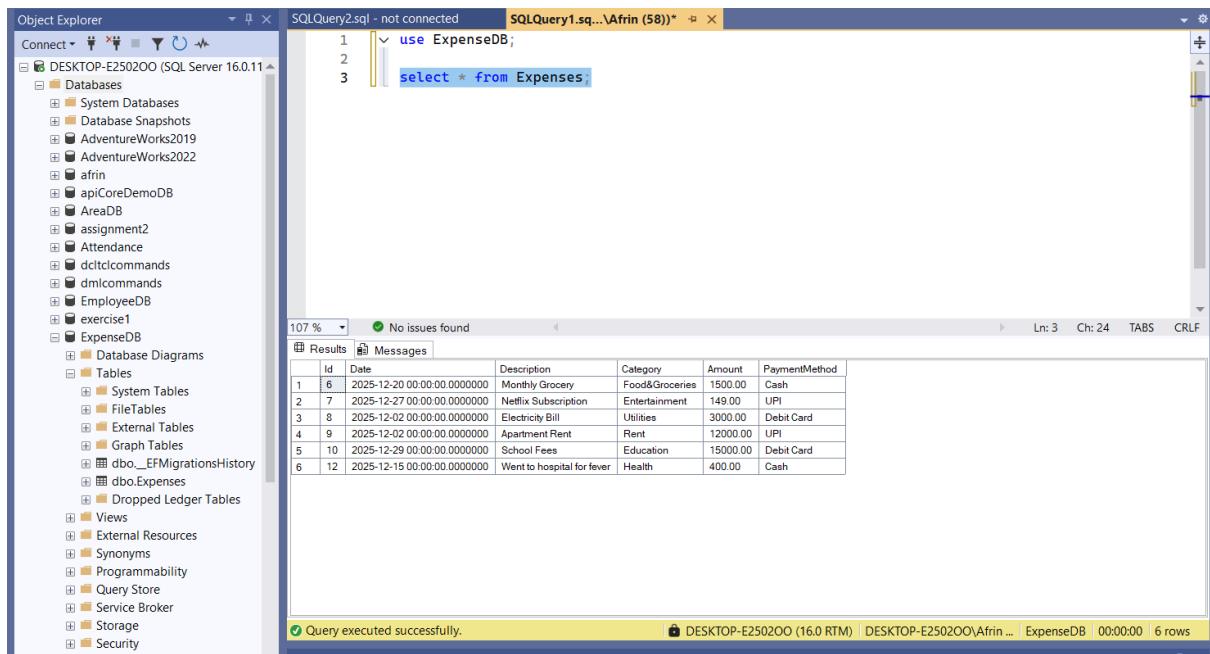
TRANSACTION	CATEGORY	AMOUNT	CONTROLS
Monthly Grocery 12/20/2025 Cash	Food&Groceries	\$1500.00	Edit Delete
Netflix Subscription 12/27/2025 UPI	Entertainment	\$149.00	Edit Delete
Electricity Bill 12/2/2025 Debit Card	Utilities	\$3000.00	Edit Delete
Apartment Rent 12/2/2025 UPI	Rent	\$12000.00	Edit Delete
School Fees 12/29/2025 Debit Card	Education	\$15000.00	Edit Delete

Updated Form –

The screenshot shows the 'Expense Tracker' application interface after adding a new transaction. The total expenditures are now \$32,049.00. The table on the right shows 'ACTIVE RECORDS' (6) with the new entry 'Went to hospital for fever' added. The new entry has a blue background.

TRANSACTION	CATEGORY	AMOUNT	CONTROLS
Monthly Grocery 12/20/2025 Cash	Food&Groceries	\$1500.00	Edit Delete
Netflix Subscription 12/27/2025 UPI	Entertainment	\$149.00	Edit Delete
Electricity Bill 12/2/2025 Debit Card	Utilities	\$3000.00	Edit Delete
Apartment Rent 12/2/2025 UPI	Rent	\$12000.00	Edit Delete
School Fees 12/29/2025 Debit Card	Education	\$15000.00	Edit Delete
Went to hospital for fever 12/15/2025 Cash	Health	\$400.00	Edit Delete

Updated Table in SQL –



The screenshot shows the Object Explorer on the left with the database 'ExpenseDB' selected. In the center, two query panes are open: 'SQLQuery2.sql' and 'SQLQuery1.sq...Afrin (58)*'. The 'SQLQuery1' pane contains the SQL command:

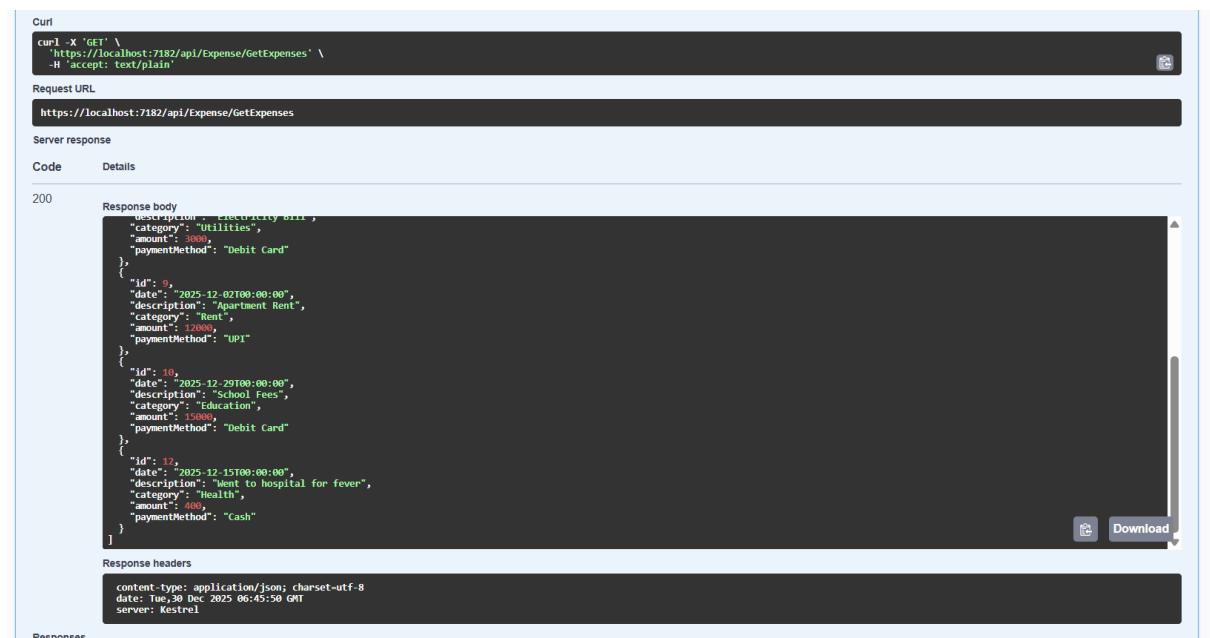
```
1 | use ExpenseDB;
2 |
3 | select * from Expenses;
```

The results pane shows the 'Expenses' table with 6 rows of data:

Id	Date	Description	Category	Amount	PaymentMethod	
1	6	2025-12-20 00:00:00.0000000	Monthly Grocery	Food&Groceries	1500.00	Cash
2	7	2025-12-27 00:00:00.0000000	Netflix Subscription	Entertainment	149.00	UPI
3	8	2025-12-02 00:00:00.0000000	Electricity Bill	Utilities	3000.00	Debit Card
4	9	2025-12-02 00:00:00.0000000	Apartment Rent	Rent	12000.00	UPI
5	10	2025-12-29 00:00:00.0000000	School Fees	Education	15000.00	Debit Card
6	12	2025-12-15 00:00:00.0000000	Went to hospital for fever	Health	400.00	Cash

At the bottom, a message indicates: 'Query executed successfully.'

In Swagger –



The screenshot shows the Swagger UI interface. At the top, there is a 'curl' command:

```
curl -X 'GET' \
  'https://localhost:7182/api/Expense/GetExpenses' \
  -H 'accept: text/plain'
```

Below it, the 'Request URL' is displayed as:

```
https://localhost:7182/api/Expense/GetExpenses
```

The 'Server response' section shows the JSON data returned by the API:

```
200
{
  "expenses": [
    {
      "category": "Electricity Bill",
      "date": "2025-12-02T00:00:00",
      "description": "Electricity Bill",
      "amount": 3000,
      "paymentMethod": "Debit Card"
    },
    {
      "id": 9,
      "date": "2025-12-02T00:00:00",
      "description": "Apartment Rent",
      "category": "Rent",
      "amount": 12000,
      "paymentMethod": "UPI"
    },
    {
      "id": 10,
      "date": "2025-12-29T00:00:00",
      "description": "School Fees",
      "category": "Education",
      "amount": 15000,
      "paymentMethod": "Debit Card"
    },
    {
      "id": 12,
      "date": "2025-12-15T00:00:00",
      "description": "Went to hospital for fever",
      "category": "Health",
      "amount": 400,
      "paymentMethod": "Cash"
    }
  ]
}
```

Below the JSON, the 'Response headers' are listed:

```
content-type: application/json; charset=utf-8
date: Tue, 30 Dec 2025 06:45:50 GMT
server: Kestrel
```

Editing Details –

Update Transaction

DATE
15-12-2025

CATEGORY
Health

AMOUNT (INR)
750

DESCRIPTION
Went to hospital for fever

PAYMENT METHOD
Cash

Update Entry

Reset Form

After clicking Update Entry –

The screenshot shows a web browser window with the URL `localhost:3001`. A modal dialog box is displayed in the center, stating "Expense Updated" with an "OK" button. Below the modal, the main content area shows a table of transaction records. The table has columns: TRANSACTION, CATEGORY, AMOUNT, and CONTROLS. The transactions listed are:

TRANSACTION	CATEGORY	AMOUNT	CONTROLS
Monthly Grocery 12/20/2025 Cash	Food&Groceries	\$1500.00	Edit Delete
Netflix Subscription 12/27/2025 UPI	Entertainment	\$149.00	Edit Delete
Electricity Bill 12/2/2025 Debit Card	Utilities	\$3000.00	Edit Delete
Apartment Rent 12/2/2025 UPI	Rent	\$12000.00	Edit Delete
School Fees 12/29/2025 Debit Card	Education	\$15000.00	Edit Delete
Went to hospital for fever 12/15/2025 Cash	Health	\$400.00	Edit Delete

Updated Form -

TOTAL EXPENDITURES		ACTIVE RECORDS	
\$32,399.00		6	
TRANSACTION	CATEGORY	AMOUNT	CONTROLS
Monthly Grocery 12/20/2025 Cash	Food&Groceries	\$1500.00	Edit Delete
Netflix Subscription 12/27/2025 UPI	Entertainment	\$149.00	Edit Delete
Electricity Bill 12/2/2025 Debit Card	Utilities	\$3000.00	Edit Delete
Apartment Rent 12/2/2025 UPI	Rent	\$12000.00	Edit Delete
School Fees 12/29/2025 Debit Card	Education	\$15000.00	Edit Delete
Went to hospital for fever 12/15/2025 Cash	Health	\$750.00	Edit Delete

In Sql Server –

```
SQLQuery2.sql - not connected SQLQuery1.sq...\\Afrin (58)* □ ×
1 1 use ExpenseDB;
2 2
3 3 select * from Expenses;
```

107 % ▾ No issues found

Results Messages

	Id	Date	Description	Category	Amount	PaymentMethod
1	6	2025-12-20 00:00:00.0000000	Monthly Grocery	Food&Groceries	1500.00	Cash
2	7	2025-12-27 00:00:00.0000000	Netflix Subscription	Entertainment	149.00	UPI
3	8	2025-12-02 00:00:00.0000000	Electricity Bill	Utilities	3000.00	Debit Card
4	9	2025-12-02 00:00:00.0000000	Apartment Rent	Rent	12000.00	UPI
5	10	2025-12-29 00:00:00.0000000	School Fees	Education	15000.00	Debit Card
6	12	2025-12-15 00:00:00.0000000	Went to hospital for fever	Health	750.00	Cash

In Swagger –

The screenshot shows the Swagger UI for an API endpoint. At the top, there's a "Responses" section. Below it, under "Curl", is a command to run a GET request to "https://localhost:7182/api/Expense/GetExpenses" with an accept header of "text/plain". The "Request URL" is also shown as "https://localhost:7182/api/Expense/GetExpenses". Under "Server response", there are tabs for "Code" (selected) and "Details". The "Code" tab shows a status code of 200 and a "Response body" containing JSON data. The JSON data represents a list of expense entries:

```
curl -X 'GET' \
  'https://localhost:7182/api/Expense/GetExpenses' \
  -H 'accept: text/plain'

Response body
[{"id": 1, "date": "2025-12-02T00:00:00", "description": "Electricity Bill", "category": "Utilities", "amount": 3000, "paymentMethod": "Debit Card"}, {"id": 9, "date": "2025-12-02T00:00:00", "description": "Apartment Rent", "category": "Rent", "amount": 12000, "paymentMethod": "UPI"}, {"id": 10, "date": "2025-12-29T00:00:00", "description": "School Fees", "category": "Education", "amount": 15000, "paymentMethod": "Debit Card"}, {"id": 12, "date": "2025-12-15T00:00:00", "description": "Went to hospital for fever", "category": "Health", "amount": 750, "paymentMethod": "Cash"}]
```

At the bottom right of the response body area, there are "Copy" and "Download" buttons.

Deleting Data –

The screenshot shows a web browser window with a modal dialog box. The dialog box has a title "localhost:3001 says" and the message "Are you sure you want to delete this record?". It contains two buttons: "OK" (highlighted in blue) and "Cancel". Behind the modal, the main page displays a table of transaction records. The table has columns: TRANSACTION, CATEGORY, AMOUNT, and CONTROLS. The data in the table is as follows:

TRANSACTION	CATEGORY	AMOUNT	CONTROLS
Monthly Grocery 12/20/2025 Cash	Food/Groceries	\$1500.00	Edit Delete
Netflix Subscription 12/27/2025 UPI	Entertainment	\$149.00	Edit Delete
Electricity Bill 12/2/2025 Debit Card	Utilities	\$3000.00	Edit Delete
Apartment Rent 12/2/2025 UPI	Rent	\$12000.00	Edit Delete
School Fees 12/29/2025 Debit Card	Education	\$15000.00	Edit Delete
Went to hospital for fever 12/15/2025 Cash	Health	\$750.00	Edit Delete

Updated form after Deleting –

TOTAL EXPENDITURES	ACTIVE RECORDS		
\$31,649.00	5		
TRANSACTION	CATEGORY	AMOUNT	CONTROLS
Monthly Grocery 12/20/2025 Cash	Food&Groceries	\$1500.00	Edit Delete
Netflix Subscription 12/27/2025 UPI	Entertainment	\$149.00	Edit Delete
Electricity Bill 12/2/2025 Debit Card	Utilities	\$3000.00	Edit Delete
Apartment Rent 12/2/2025 UPI	Rent	\$12000.00	Edit Delete
School Fees 12/29/2025 Debit Card	Education	\$15000.00	Edit Delete

In SQL Server –

SQLQuery2.sql - not connected SQLQuery1.sq...\\Afrin (58)*

```

1  use ExpenseDB;
2
3  select * from Expenses;

```

107 % No issues found Results Messages Ln: 3 Ch: 1 TABS CRLF

	ID	Date	Description	Category	Amount	PaymentMethod
1	6	2025-12-20 00:00:00.0000000	Monthly Grocery	Food&Groceries	1500.00	Cash
2	7	2025-12-27 00:00:00.0000000	Netflix Subscription	Entertainment	149.00	UPI
3	8	2025-12-02 00:00:00.0000000	Electricity Bill	Utilities	3000.00	Debit Card
4	9	2025-12-02 00:00:00.0000000	Apartment Rent	Rent	12000.00	UPI
5	10	2025-12-29 00:00:00.0000000	School Fees	Education	15000.00	Debit Card

In Swagger –

Curl

```
curl -X 'GET' \
'https://localhost:7182/api/Expense/GetExpenses' \
-H 'accept: text/plain'
```

Request URL

```
https://localhost:7182/api/Expense/GetExpenses
```

Server response

Code Details

200 Response body

```
[{"id": 1, "date": "2025-12-01T00:00:00", "description": "Netflix Subscription", "category": "Entertainment", "amount": 149, "paymentMethod": "UPI"}, {"id": 2, "date": "2025-12-02T00:00:00", "description": "Electricity Bill", "category": "Utilities", "amount": 3000, "paymentMethod": "Debit Card"}, {"id": 3, "date": "2025-12-02T00:00:00", "description": "Apartment Rent", "category": "Rent", "amount": 12000, "paymentMethod": "UPI"}, {"id": 4, "date": "2025-12-29T00:00:00", "description": "School Fees", "category": "Education", "amount": 15000, "paymentMethod": "Debit Card"}]
```

Response headers

Download