

**VIJAY M**

**Day 13**

**date -21/08/25**

**1. only user with admin access can insert customer and bank account details.**

**2. create an account balance api, which receives username as input and provides the balance amount across all the accounts.**

**- 2.1 logged in user can view the balance amount .**

**source code:**

**AuthController.cs**

```
using Microsoft.AspNetCore.Identity.Data;
```

```
using Microsoft.AspNetCore.Mvc;
```

```
using Microsoft.Extensions.Configuration;
```

```
using Microsoft.IdentityModel.Tokens;
```

```
using Miniproject.Models;
```

```
using Newtonsoft.Json;
```

```
using System.IdentityModel.Tokens.Jwt;
```

```
using System.Security.Claims;
```

```
using System.Text;
```

```
namespace Miniproject.Controllers
```

```
{
```

```
    [Route("api/[controller]")]
```

```
    [ApiController]
```

```
    public class AuthController : ControllerBase
```

```
    {
```

```
        private readonly HttpClient _httpClient;
```

```

private readonly IConfiguration _configuration;

public AuthController(HttpClient httpClient, IConfiguration configuration)
{
    _httpClient = httpClient;
    _configuration = configuration;
}

[HttpPost("login")]
public async Task<ActionResult> Login([FromBody] Models.LoginRequest loginRequest)
{
    string fastApiUrl = "http://127.0.0.1:8000/login"; // FastAPI URL

    var content = new StringContent(
        JsonConvert.SerializeObject(loginRequest),
        Encoding.UTF8,
        "application/json"
    );

    try
    {
        var response = await _httpClient.PostAsync(fastApiUrl, content); // POST request

        if (response.IsSuccessStatusCode)
        {
            var responseBody = await response.Content.ReadAsStringAsync();

            var loginResponse = JsonConvert.DeserializeObject<LoginResponse>(responseBody);

```

```

        var token = GenerateJwtToken(loginResponse);

        return Ok(new
        {
            message = "Login successful",
            token = token,
            data = loginResponse
        });
    }

    else
    {
        var errorResponse = await response.Content.ReadAsStringAsync();

        return Unauthorized(new { message = "Invalid credentials", error = errorResponse });
    }
}

catch (Exception ex)
{
    return StatusCode(500, new { message = "An error occurred while processing your request", error =
ex.Message });
}
}

private string GenerateJwtToken(LoginResponse user)
{
    var jwtSettings = _configuration.GetSection("JwtSettings");

    var secretKey = jwtSettings["SecretKey"];

    // ✔ Validate key length (must be at least 256 bits = 32 characters)

```

```
        if (string.IsNullOrEmpty(secretKey) || secretKey.Length < 32)
        {
            throw new Exception("Secret key must be at least 32 characters long.");
        }

        var key = new SymmetricSecurityKey(Encoding.UTF8.GetBytes(secretKey));
        var creds = new SigningCredentials(key, SecurityAlgorithms.HmacSha256);

        var claims = new[]
        {
            new Claim(ClaimTypes.Name, user.Name),
            new Claim(ClaimTypes.Role, user.Role)
        };

        var token = new JwtSecurityToken(
            issuer: jwtSettings["Issuer"],
            audience: jwtSettings["Audience"],
            claims: claims,
            expires: DateTime.Now.AddMinutes(Convert.ToDouble(jwtSettings["ExpiryMinutes"])),
            signingCredentials: creds
        );

        return new JwtSecurityTokenHandler().WriteToken(token);
    }

}
```

## BankAccountController.cs

```
using Microsoft.AspNetCore.Authorization;
```

```
using Microsoft.AspNetCore.Mvc;
```

```
using Microsoft.EntityFrameworkCore;
```

```
using Miniproject.Context;
```

```
using Miniproject.Models;
```

```
using System;
```

```
using System.Linq;
```

```
using System.Security.Claims;
```

```
namespace Miniproject.Controllers
```

```
{
```

```
    [ApiController]
```

```
    [Route("[controller]")]
```

```
    public class BankAccountController : ControllerBase
```

```
    {
```

```
        private readonly AppDbContext context;
```

```
        public BankAccountController(AppDbContext context)
```

```
        {
```

```
            this.context = context;
```

```
        }
```

```
        [HttpPost("add")]
```

```
        [Authorize(Roles = "Admin")]
```

```
        public IActionResult AddBankAccount([FromBody] BankAccountCreateDto dto)
```

```

{

    var customer = context.Customers.FirstOrDefault(c => c.Id == dto.CustomerId);

    if (customer == null)
    {
        return NotFound(new { message = $"Customer with ID {dto.CustomerId} not found" });
    }

    var bankAccount = new BankAccount
    {
        AccountNumber = dto.AccountNumber,
        Amount = dto.Amount,
        CreatedAt = dto.CreatedAt,
        CustomerId = dto.CustomerId
    };

    context.BankAccounts.Add(bankAccount);

    context.SaveChanges();

    BankAccountCreateDto bankdto = new BankAccountCreateDto { AccountNumber =
bankAccount.AccountNumber, Amount = bankAccount.Amount, CreatedAt = bankAccount.CreatedAt, CustomerId =
bankAccount.CustomerId };

    return Ok(new { message = "Bank account added successfully", bankdto });
}

[HttpGet("all")]

public IActionResult GetAllAccounts()
{

```

```
        var accounts = context.BankAccounts.Include(a => a.Customer).ToList();

        return Ok(accounts);
    }
}
```

```
[HttpGet("by-customer/{customerId}")]
```

```
public IActionResult GetAccountsByCustomer(int customerId)
```

```
{
```

```
    var accounts = context.BankAccounts
```

```
        .Where(a => a.CustomerId == customerId)
```

```
        .ToList();
```

```
    if (accounts == null || accounts.Count == 0)
```

```
    {
```

```
        return NotFound(new { message = $"No accounts found for customer ID {customerId}" });
```

```
    }
```

```
    return Ok(accounts);
```

```
}
```

```
[HttpGet("balance")]
```

```
[Authorize]
```

```
public IActionResult GetUserTotalBalance()
```

```
{
```

```
    var username = User.FindFirst(ClaimTypes.Name)?.Value;
```

```

        if (string.IsNullOrEmpty(username))
        {
            return Unauthorized(new { message = "Invalid user token" });
        }

        var customer = context.Customers.FirstOrDefault(c => c.Name == username);

        if (customer == null)
        {
            return NotFound(new { message = $"Customer '{username}' not found." });
        }

        // Calculate total balance for this customer
        var totalBalance = context.BankAccounts
            .Where(a => a.CustomerId == customer.Id)
            .Sum(a => a.Amount);

        return Ok(new
        {
            name = username,
            totalBalance
        });
    }

    [HttpGet]
    public IActionResult Index()
    {

```



```

        return Ok("BankAccount API is running.");
    }
}
}

```

## CustomerController.cs

```

using Microsoft.AspNetCore.Authorization;

using Microsoft.AspNetCore.Mvc;

using Miniproject.Context;

using Miniproject.Models;

using System.Linq;

namespace Miniproject.Controllers
{
    [ApiController]

    [Route("[controller]")]

    public class CustomerController : ControllerBase
    {
        private readonly AppDbContext context;

        public CustomerController(AppDbContext context)
        {
            this.context = context;
        }

        [HttpPost("add")]

        [Authorize(Roles = "Admin")]

        public IActionResult AddCustomer([FromBody] CustomerCreateDto customerDto)
    }
}

```

```

{
    var customer = new Customer
    {
        Name = customerDto.Name,
        Age = customerDto.Age
    };

    context.Customers.Add(customer);

    context.SaveChanges();

    return Ok(new { message = "Customer added successfully", customer });
}

```

```

[HttpGet("all")]
public IActionResult GetAllCustomers()
{
    var allCustomers = context.Customers.ToList();

    return Ok(allCustomers);
}

```

```

[HttpGet("{id}")]
public IActionResult GetCustomerById(int id)
{
    var customer = context.Customers.FirstOrDefault(c => c.Id == id);

    if (customer == null)
    {

```

```

        return NotFound(new { message = $"Customer with ID {id} not found" });
    }

    return Ok(customer);
}

[HttpGet]

public IActionResult Index()
{
    return Ok("Customer API is running.");
}
}
}

```

#### **program.cs**

```

using Microsoft.EntityFrameworkCore;
using Microsoft.IdentityModel.Tokens;
using Miniproject.Context;
using System.Text;

var builder = WebApplication.CreateBuilder(args);

// Add services to the container

builder.Services.AddControllers();

// Learn more about configuring Swagger/OpenAPI at https://aka.ms/aspnetcore/swashbuckle

builder.Services.AddEndpointsApiExplorer();

builder.Services.AddSwaggerGen();

const string connectionString= "Data Source=PTPLL605;" +

    "Initial Catalog=sampled;" +

    "Integrated Security=True;" +

    "TrustServerCertificate=True;";

```

```

builder.Services.AddDbContext<AppDbContext>(x=>x.UseSqlServer(connectionstring));

builder.Services.AddControllers()

    .AddJsonOptions(options =>

    {

        options.JsonSerializerOptions.ReferenceHandler =
System.Text.Json.Serialization.ReferenceHandler.IgnoreCycles;

        options.JsonSerializerOptions.MaxDepth = 32; // optional, default is 32

    });

builder.Services.AddAuthentication("Bearer")

    .AddJwtBearer("Bearer", options =>

    {

        var jwtSettings = builder.Configuration.GetSection("JwtSettings");

        options.TokenValidationParameters = new TokenValidationParameters

        {

            ValidateIssuer = true,

            ValidateAudience = true,

            ValidateLifetime = true,

            ValidateIssuerSigningKey = true,

            ValidIssuer = jwtSettings["Issuer"],

            ValidAudience = jwtSettings["Audience"],

            IssuerSigningKey = new SymmetricSecurityKey(Encoding.UTF8.GetBytes(jwtSettings["SecretKey"]))

        };

    });

builder.Services.AddHttpClient();

var app = builder.Build();

// Configure the HTTP request pipeline.

if (app.Environment.IsDevelopment())

{

    app.UseSwagger();

```

```
        app.UseSwaggerUI();
    }

    app.UseHttpsRedirection();

    app.UseAuthorization();

    app.MapControllers();

    app.Run();
```

## **FastApi**

### **main.py**

```
from sqlalchemy import create_engine, Column, Integer, String
from sqlalchemy.orm import sessionmaker, Session, declarative_base
from fastapi import FastAPI, Depends, HTTPException, status
from pydantic import BaseModel
from typing import List

import hashlib # Use bcrypt for secure password hashing
import uvicorn

from fastapi.responses import RedirectResponse

url = "sqlite:///./user.db"

engine = create_engine(url, connect_args={"check_same_thread": False})

SessionLocal = sessionmaker(autocommit=False, autoflush=False, bind=engine)

Base = declarative_base()

app = FastAPI()

class User(Base):

    __tablename__ = "user"

    Id = Column(Integer, primary_key=True, index=True)

    Name = Column(String(100), nullable=False)

    Password = Column(String(100), nullable=False)
```

```

        Role = Column(String(100), nullable=False)

Base.metadata.create_all(bind=engine)

class Users(BaseModel):

    Name: str

    Password: str

    Role: str

class UserResponse(BaseModel):

    Name: str

    Role: str

    class Config:

        orm_mode = True

class LoginRequest(BaseModel):

    name: str

    password: str

def get_db():

    db = SessionLocal()

    try:

        yield db

    finally:

        db.close()

def hash_password(password: str) -> str:

    return hashlib.sha256(password.encode('utf-8')).hexdigest()

def verify_password(stored_hash: str, password: str) -> bool:

    return stored_hash == hash_password(password)

@app.post("/add_user")

def add_user(user: Users, db: Session = Depends(get_db)):

```

```

hashed_pw = hash_password(user.Password)

u = User(Name=user.Name, Password=hashed_pw, Role=user.Role)

db.add(u)

db.commit()

db.refresh(u)

return RedirectResponse(url="/get_users", status_code=303)

@app.get("/get_users", response_model=List[UserResponse])

def get_users(db: Session = Depends(get_db)):

    return db.query(User).all()

@app.get("/get_userbyId/{id}", response_model=UserResponse)

def get_user(id: int, db: Session = Depends(get_db)):

    u = db.query(User).filter(User.Id == id).first()

    if not u:

        raise HTTPException(status_code=404, detail="User not found")

    return u

@app.post("/login", response_model=UserResponse)

def user_login(login_request: LoginRequest, db: Session = Depends(get_db)):

    name = login_request.name

    password = login_request.password

    u = db.query(User).filter(User.Name == name).first()

    if u is None or not verify_password(u.Password, password):

        raise HTTPException(status_code=401, detail="Invalid username or password")

    return UserResponse(Name=u.Name, Role=u.Role)

if __name__ == "__main__":

    uvicorn.run("user:app", host="127.0.0.1", port=8000, reload=True)

```

## Output

https://localhost:7146/api/Auth/login

## Responses

### Curl

```
curl -X 'POST' \
  'https://localhost:7146/api/Auth/login' \
  -H 'accept: */*' \
  -H 'Content-Type: application/json' \
  -d '{
    "name": "thalapathy",
    "password": "12345"
  }'
```

### Request URL

https://localhost:7146/api/Auth/login

### Server response

#### Code

#### Details

200

#### Response body

```
{
  "message": "Login successful",
  "token": "eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9.eyJ1IjoiYWR5b3R5IiwiaWF0IjoxNjU0MjUwMDA",
  "data": {
    "name": "thalapathy",
    "role": "Admin"
  }
}
```

https://localhost:7146/Customr/add





https://localhost:7146/Customer/add

POST



https://localhost:7146/Customer/add

Params

Authorization ●

Headers (9)

Body ●

Pre-request Script

Tests

Settings



none



form-data



x-www-form-urlencoded



raw



binary

JSON ▾



```
1 {  
2   "name": "dskjnkjhg",  
3   "age": 25  
4 }
```

Body

Cookies

Headers (4)

Test Results

Pretty

Raw

Preview

Visualize

JSON ▾



```
1 {  
2   "message": "Customer added successfully",  
3   "customer": {  
4     "id": 6,  
5     "name": "dskjnkjhg",  
6     "age": 25  
7   }  
8 }
```

Unauthorized

#### Curl

```
curl -X 'POST' \
  'https://localhost:7146/BankAccount/add' \
  -H 'accept: */*' \
  -H 'Content-Type: application/json' \
  -d '{
    "accountNumber": "string",
    "amount": 0,
    "createdAt": "2025-08-22T11:57:46.458Z",
    "customerId": 0
  }'
```

#### Request URL

<https://localhost:7146/BankAccount/add>

#### Server response

##### Code

##### Details

401

*Undocumented*

Error: response status is 401

##### Response headers

```
content-length: 0
date: Fri, 22 Aug 2025 11:57:48 GMT
server: Kestrel
www-authenticate: Bearer
```

<https://localhost:7146/BankAccount/add>

POST https://localhost:7146/B



https://localhost:7146/BankAccount/add

POST



https://localhost:7146/BankAccount/add

Params

Authorization ●

Headers (9)

Body ●

Pre-request Script

Tests

S



none



form-data



x-www-form-urlencoded



raw



binary

JSON



```
1 {
2   "accountNumber": "1005",
3   "amount": 1000,
4   "createdAt": "2025-08-22T11:57:46.458Z",
5   "customerId": 2
6 }
```

Body

Cookies

Headers (4)

Test Results

Pretty

Raw

Preview

Visualize

JSON



```
1 {
2   "message": "Bank account added successfully",
3   "bankdto": {
4     "accountNumber": "1005",
5     "amount": 1000,
6     "createdAt": "2025-08-22T11:57:46.458Z",
7     "customerId": 2
8   }
9 }
```

https://localhost:7146/BankAccount/all

#### Curl

```
curl -X 'GET' \
  'https://localhost:7146/BankAccount/all' \
  -H 'accept: */*'
```

#### Request URL

<https://localhost:7146/BankAccount/all>

#### Server response

##### Code

##### Details

200

##### Response body

```
[
  {
    "id": 1,
    "accountNumber": "10005",
    "amount": 1000,
    "createdAt": "2025-08-22T06:50:45.204",
    "customerId": 1,
    "customer": {
      "id": 1,
      "name": "rajan",
      "age": 23
    }
  },
  {
    "id": 2,
    "accountNumber": "1005",
    "amount": 10000,
    "createdAt": "2025-08-22T06:53:26.257",
    "customerId": 1,
    "customer": {
      "id": 1,
      "name": "rajan",
      "age": 23
    }
  }
]
```

<https://localhost:7146/BankAccount/balance>



https://localhost:7146/BankAccount/balance

GET



https://localhost:7146/BankAccount/balance

Params

Authorization ●

Headers (7)

Body

Pre-request Script

Tests

Set

Type

Bearer Token



Token

The authorization header will be automatically generated when you send the request.

[Learn more about authorization](#)



Store your secrets with end-to-e

Body

Cookies

Headers (4)

Test Results

Pretty

Raw

Preview

Visualize

JSON



```
1 {
2   "name": "rajan",
3   "totalBalance": 11200
4 }
```