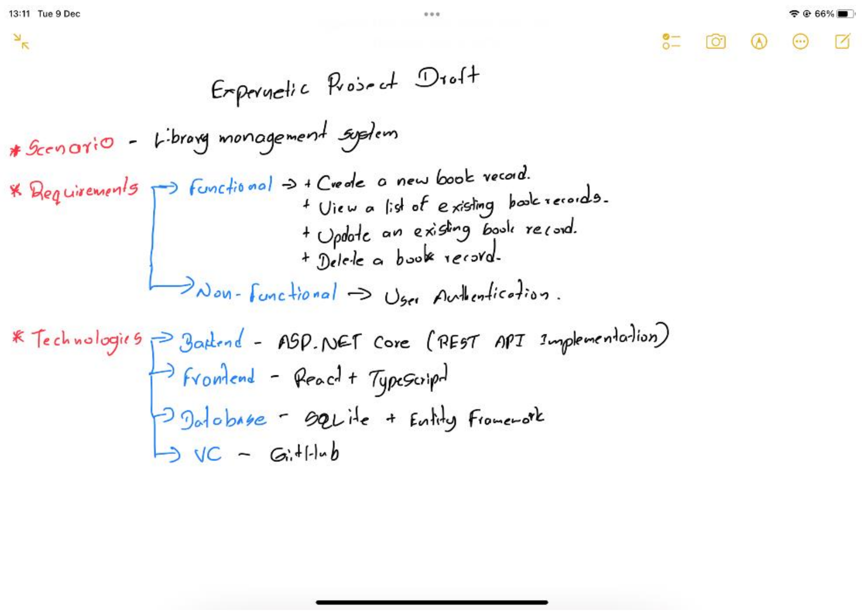
Expernetic – Project

Project Drafting

As the first step I did examine the requirement document carefully and then identify the functional and non-functional requirements, technologies for backend, frontend, database and version control.



Expernetic Project Draft

* Scenario - Library management system

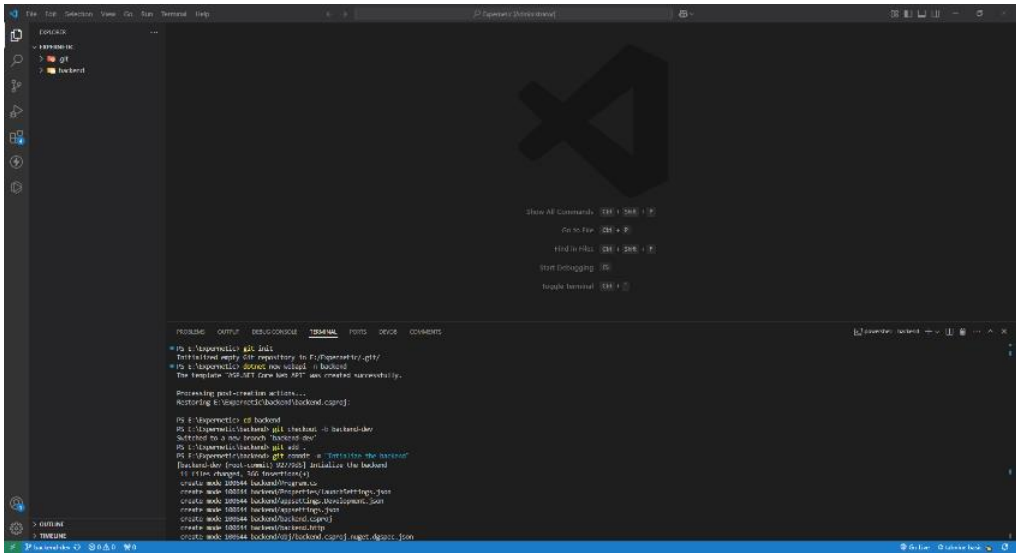
* Requirements → Functional → + Create a new book record.
+ View a list of existing book records.
+ Update an existing book record.
+ Delete a book record.

→ Non-Functional → User Authentication.

* Technologies → Backend - ASP.NET Core (REST API Implementation)
→ Frontend - React + TypeScript
→ Database - SQLite + Entity Framework
→ VC - GitHub

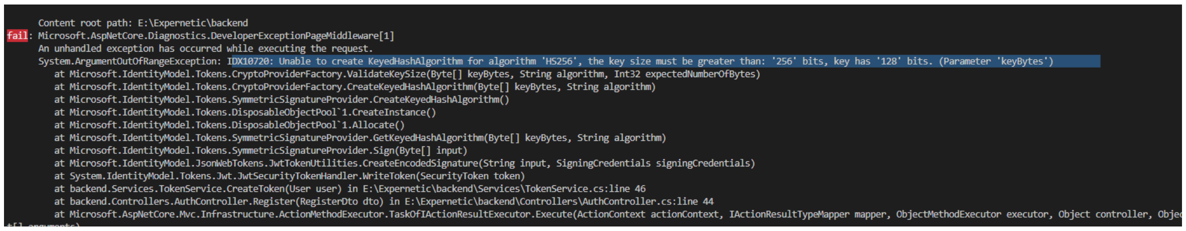
Backend Development

Initialize the ASP.NET Core backend and github for version control



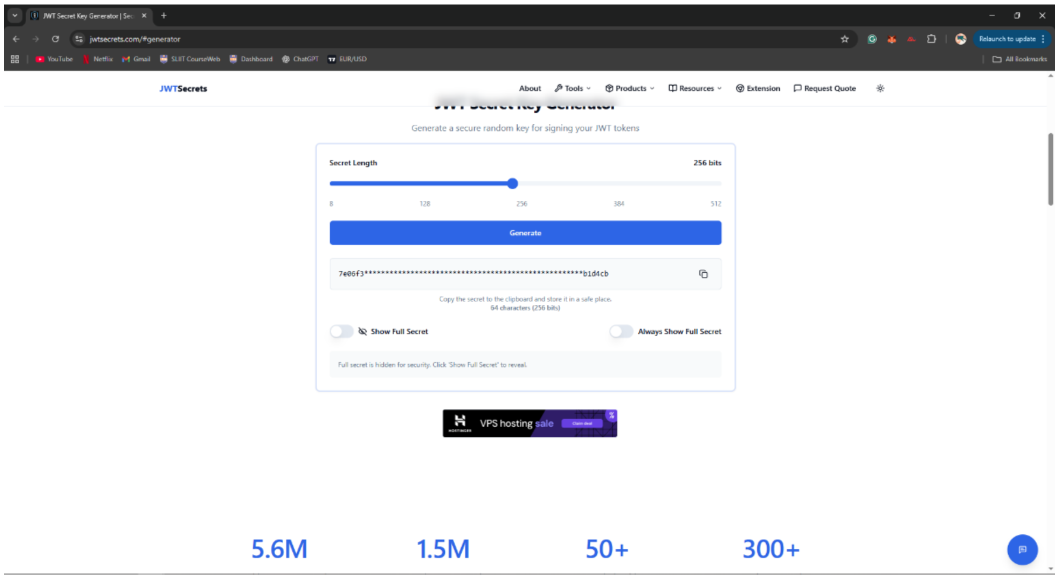
```
PS E:\Expernetic> git init
Initialized empty Git repository in E:/Expernetic/.git/
PS E:\Expernetic> dotnet new webapi -n backend
The template "ASP.NET Core Web API" was created successfully.

Processing post-creation actions...
Restoring E:\Expernetic\backend\backend.csproj:

PS E:\Expernetic> cd backend
PS E:\Expernetic\backend> git checkout -b backend-dev
Switched to a new branch 'backend-dev'
PS E:\Expernetic\backend> git add .
PS E:\Expernetic\backend> git commit -m "Initialize the backend"
[backend-dev (root-commit) 9277985] Initialize the backend
 11 files changed, 366 insertions(+)
 create mode 100644 backend/Program.cs
 create mode 100644 backend/Properties/launchSettings.json
 create mode 100644 backend/appsettings.Development.json
 create mode 100644 backend/appsettings.json
 create mode 100644 backend/backend.csproj
 create mode 100644 backend/backend.http
 create mode 100644 backend/obj/backend.csproj.nuget.dgspec.json
```

Pop-up a backend issue that says JWT secret key length is not enough



```
      Content root path: E:\Expernetic\backend
fail: Microsoft.AspNetCore.Diagnostics.DeveloperExceptionPageMiddleware[1]
      An unhandled exception has occurred while executing the request.
      System.ArgumentOutOfRangeException: IDX10720: Unable to create KeyedHashAlgorithm for algorithm 'HS256', the key size must be greater than: '256' bits, key has '128' bits. (Parameter 'keyBytes')
         at Microsoft.IdentityModel.Tokens.CryptoProviderFactory.ValidateKeySize(Byte[] keyBytes, String algorithm, Int32 expectedNumberOfBytes)
         at Microsoft.IdentityModel.Tokens.CryptoProviderFactory.CreateKeyedHashAlgorithm(Byte[] keyBytes, String algorithm)
         at Microsoft.IdentityModel.Tokens.SymmetricSignatureProvider.CreateKeyedHashAlgorithm()
         at Microsoft.IdentityModel.Tokens.DisposableObjectPool`1.CreateInstance()
         at Microsoft.IdentityModel.Tokens.DisposableObjectPool`1.Allocate()
         at Microsoft.IdentityModel.Tokens.SymmetricSignatureProvider.GetKeyedHashAlgorithm(Byte[] keyBytes, String algorithm)
         at Microsoft.IdentityModel.Tokens.SymmetricSignatureProvider.Sign(Byte[] input)
         at Microsoft.IdentityModel.JsonWebTokens.JwtTokenUtilities.CreateEncodedSignature(String input, SigningCredentials signingCredentials)
         at System.IdentityModel.Tokens.Jwt.JwtSecurityTokenHandler.WriteToken(SecurityToken token)
         at backend.Services.TokenService.CreateToken(User user) in E:\Expernetic\backend\Services\TokenService.cs:line 46
         at backend.Controllers.AuthController.Register(RegisterDto dto) in E:\Expernetic\backend\Controllers\AuthController.cs:line 44
         at Microsoft.AspNetCore.Mvc.Infrastructure.ActionMethodExecutor.TaskOfIActionResultExecutor.Execute(ActionContext actionContext, IActionResultTypeMapper mapper, ObjectMethodExecutor executor, Object controller, Obje
```

Use a JWT secret key generating website to generate a secured 256bits key as the solution



jwtsecrets.com/#generator

JWT Secret Key Generator

Generate a secure random key for signing your JWT tokens

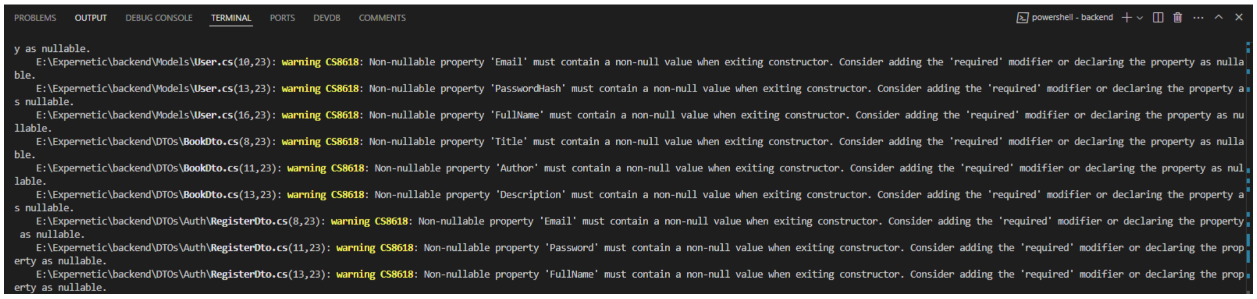
Secret Length 256 bits

```
7e06f3****************************************************b1d4cb
```

Copy the secret to the clipboard and store it in a safe place.
64 characters (256 bits)

Full secret is hidden for security. Click 'Show Full Secret' to reveal.

Pop-up nullable reference type warning and fix it using declared properties with non-nullable string.

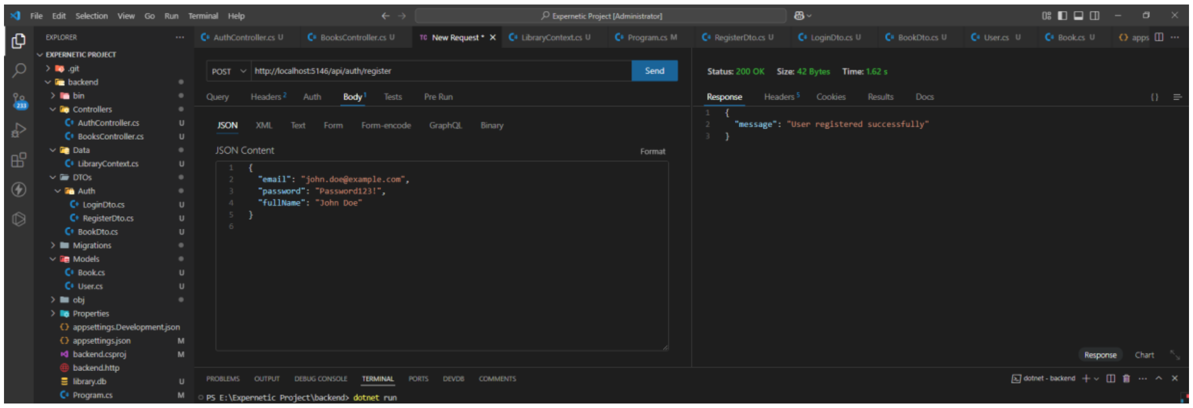


```
y as nullable.
    E:\Expernetic\backend\Models\User.cs(10,23): warning CS8618: Non-nullable property 'Email' must contain a non-null value when exiting constructor. Consider adding the 'required' modifier or declaring the property as nulla
ble.
    E:\Expernetic\backend\Models\User.cs(13,23): warning CS8618: Non-nullable property 'PasswordHash' must contain a non-null value when exiting constructor. Consider adding the 'required' modifier or declaring the property a
s nullable.
    E:\Expernetic\backend\Models\User.cs(16,23): warning CS8618: Non-nullable property 'FullName' must contain a non-null value when exiting constructor. Consider adding the 'required' modifier or declaring the property as nu
llable.
    E:\Expernetic\backend\DTOs\BookDto.cs(8,23): warning CS8618: Non-nullable property 'Title' must contain a non-null value when exiting constructor. Consider adding the 'required' modifier or declaring the property as nulla
ble.
    E:\Expernetic\backend\DTOs\BookDto.cs(11,23): warning CS8618: Non-nullable property 'Author' must contain a non-null value when exiting constructor. Consider adding the 'required' modifier or declaring the property as nul
lable.
    E:\Expernetic\backend\DTOs\BookDto.cs(13,23): warning CS8618: Non-nullable property 'Description' must contain a non-null value when exiting constructor. Consider adding the 'required' modifier or declaring the property a
s nullable.
    E:\Expernetic\backend\DTOs\Auth\RegisterDto.cs(8,23): warning CS8618: Non-nullable property 'Email' must contain a non-null value when exiting constructor. Consider adding the 'required' modifier or declaring the property
 as nullable.
    E:\Expernetic\backend\DTOs\Auth\RegisterDto.cs(11,23): warning CS8618: Non-nullable property 'Password' must contain a non-null value when exiting constructor. Consider adding the 'required' modifier or declaring the prop
erty as nullable.
    E:\Expernetic\backend\DTOs\Auth\RegisterDto.cs(13,23): warning CS8618: Non-nullable property 'FullName' must contain a non-null value when exiting constructor. Consider adding the 'required' modifier or declaring the prop
erty as nullable.
```

Backend API endpoints Testing with Thunder Client

01.Register a new user [http://localhost:5146/api/auth/register]

JSON body includes email, password and full name. Got the response with 200 OK status with customize successful message.



```
POST http://localhost:5146/api/auth/register
{
  "email": "john.doe@example.com",
  "password": "Password123!",
  "fullName": "John Doe"
}
```

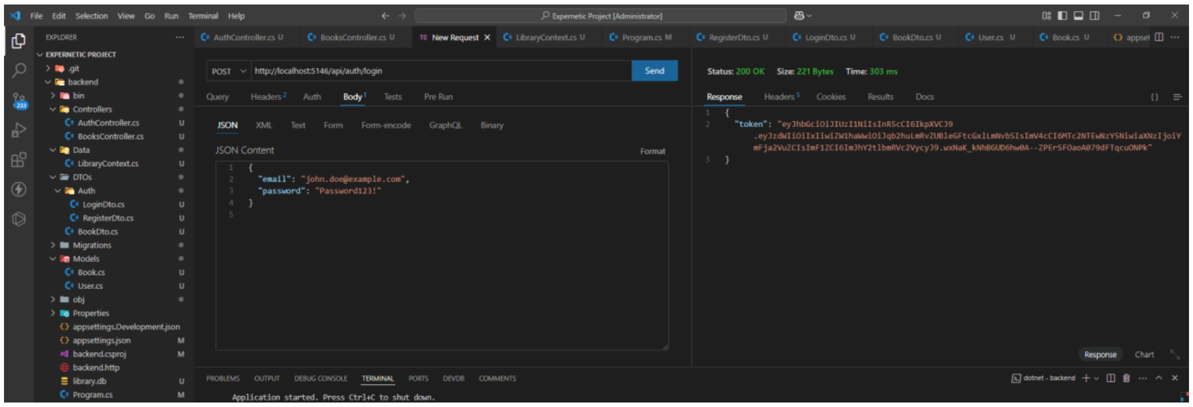
Status: 200 OK Size: 42 Bytes Time: 1.62 s

```
{
  "message": "User registered successfully"
}
```

```
PS E:\Expernetic Project\backend> dotnet run
```

02.Login [http://localhost:5146/api/auth/login]

Try to login with registered credentials and return 200 OK status with token as the response.



```
POST http://localhost:5146/api/auth/login
{
  "email": "john.doe@example.com",
  "password": "Password123!"
}
```

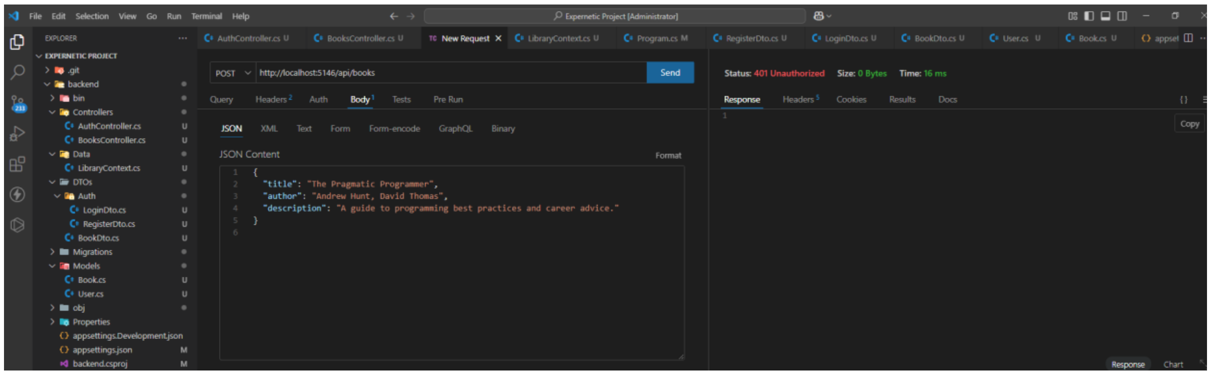
Status: 200 OK Size: 221 Bytes Time: 303 ms

```
{
  "token": "eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9
      .eyJzdWIiOiIxIiwiZW1haWwiOiJqb2huLmRvZUBleGFtcGxlLmNvbSIsImV4cCI6MTc2NTEwNzY5NiwiaXNzIjoiY
      mFja2VuZCIsImF1ZCI6ImJhY2tlbmRVc2VycyJ9.wxNaK_kNhBGUD6hw0A--ZPErSFOaoA079dFTqcuONPk"
}
```

```
Application started. Press Ctrl+C to shut down.
```

03.Create a new book without token[http://localhost:5146/api/books]

Use POST request with book title, author and description without token to ensure the authorization and return 401 Unauthorized status.

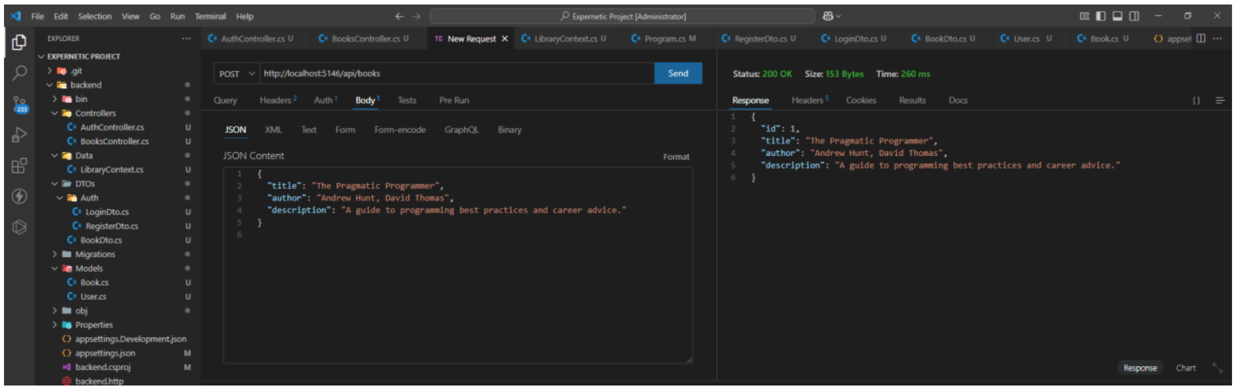


```
POST  http://localhost:5146/api/books
{
  "title": "The Pragmatic Programmer",
  "author": "Andrew Hunt, David Thomas",
  "description": "A guide to programming best practices and career advice."
}
```

```
Status: 401 Unauthorized   Size: 0 Bytes   Time: 16 ms
```

04.Create a new book with token[http://localhost:5146/api/books]

Use POST request with book title, author and description with token to ensure the authorization and return 200 OK status and return the id, title, author and description as response.

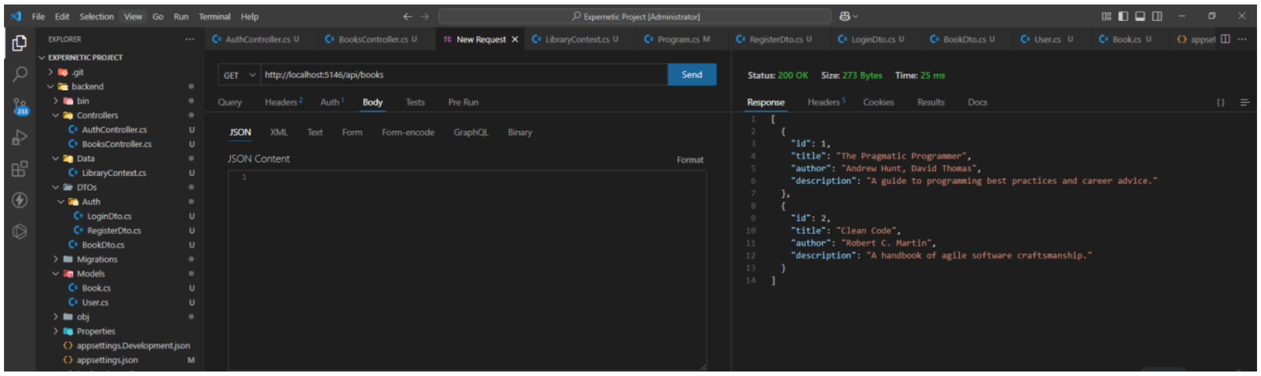


```
POST  http://localhost:5146/api/books
{
  "title": "The Pragmatic Programmer",
  "author": "Andrew Hunt, David Thomas",
  "description": "A guide to programming best practices and career advice."
}
```

```
Status: 200 OK   Size: 153 Bytes   Time: 260 ms
{
  "id": 1,
  "title": "The Pragmatic Programmer",
  "author": "Andrew Hunt, David Thomas",
  "description": "A guide to programming best practices and career advice."
}
```

05.Get all books[http://localhost:5146/api/books]

Use GET request to retrieve all the available books.

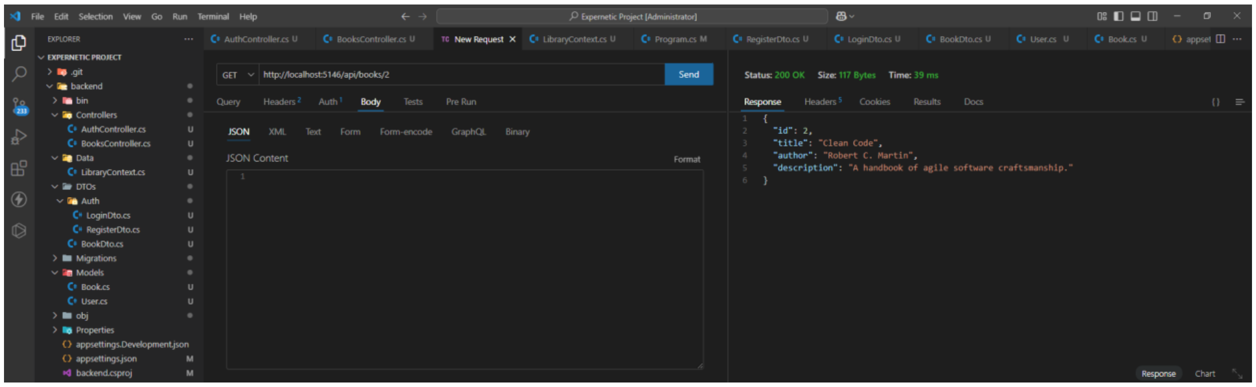


```
GET  http://localhost:5146/api/books
```

```
Status: 200 OK   Size: 273 Bytes   Time: 25 ms
[
  {
    "id": 1,
    "title": "The Pragmatic Programmer",
    "author": "Andrew Hunt, David Thomas",
    "description": "A guide to programming best practices and career advice."
  },
  {
    "id": 2,
    "title": "Clean Code",
    "author": "Robert C. Martin",
    "description": "A handbook of agile software craftsmanship."
  }
]
```

06.Get a specific book by ID [http://localhost:5146/api/books/2]

Use GET request along with specific book ID to retrieve a specific book and return the book details in response with 200 OK status.



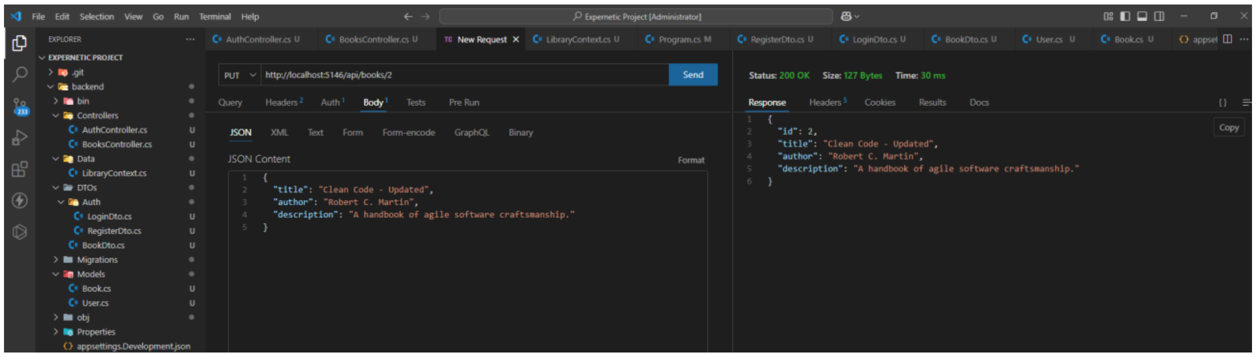
GET http://localhost:5146/api/books/2

Status: 200 OK Size: 117 Bytes Time: 39 ms

```
{
  "id": 2,
  "title": "Clean Code",
  "author": "Robert C. Martin",
  "description": "A handbook of agile software craftsmanship."
}
```

07.Update a specific book by ID [http://localhost:5146/api/books/2]

Use PUT request to update a specific book title and return the updated details in response with 200 OK status.



PUT http://localhost:5146/api/books/2

JSON Content

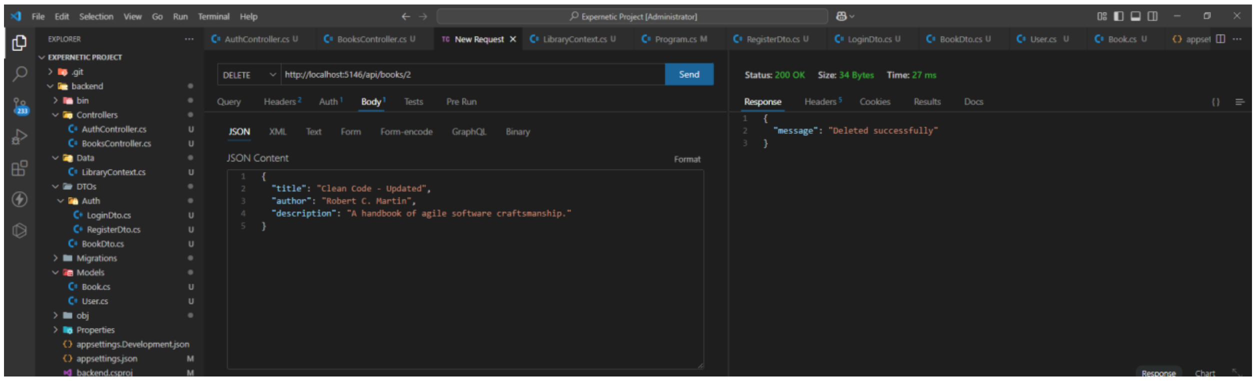
```
{
  "title": "Clean Code - Updated",
  "author": "Robert C. Martin",
  "description": "A handbook of agile software craftsmanship."
}
```

Status: 200 OK Size: 127 Bytes Time: 30 ms

```
{
  "id": 2,
  "title": "Clean Code - Updated",
  "author": "Robert C. Martin",
  "description": "A handbook of agile software craftsmanship."
}
```

08.Delete a specific book by ID [http://localhost:5146/api/books/2]

Use DELETE to request the test the delete endpoint with specific book id and return 200 OK status message with customized successful message.


```
DELETE http://localhost:5146/api/books/2
```

JSON Content

```json
{
  "title": "Clean Code - Updated",
  "author": "Robert C. Martin",
  "description": "A handbook of agile software craftsmanship."
}
```

Status: 200 OK Size: 34 Bytes Time: 27 ms

```json
{
  "message": "Deleted successfully"
}
```

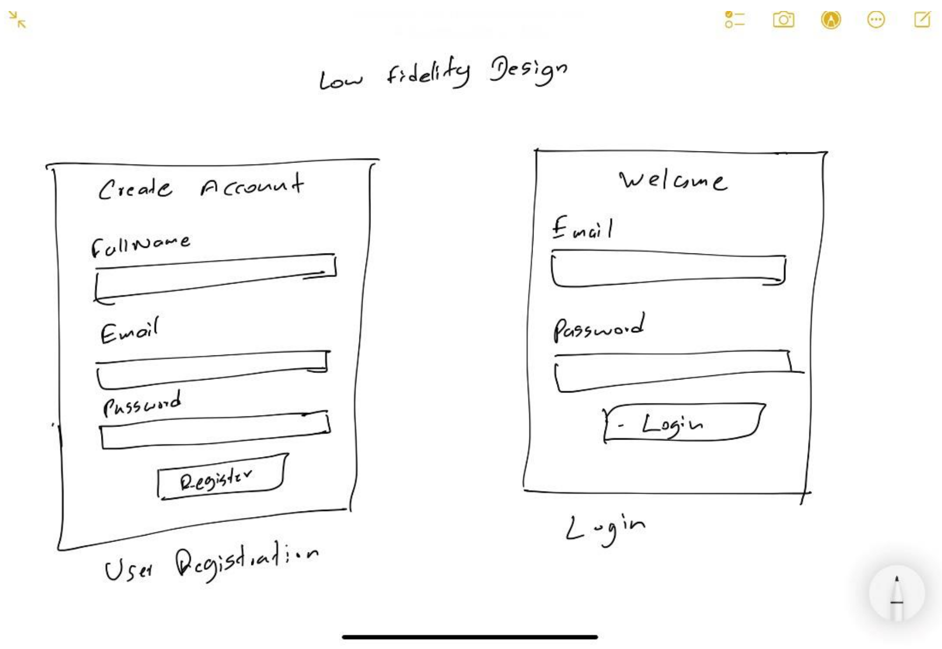
Frontend Development

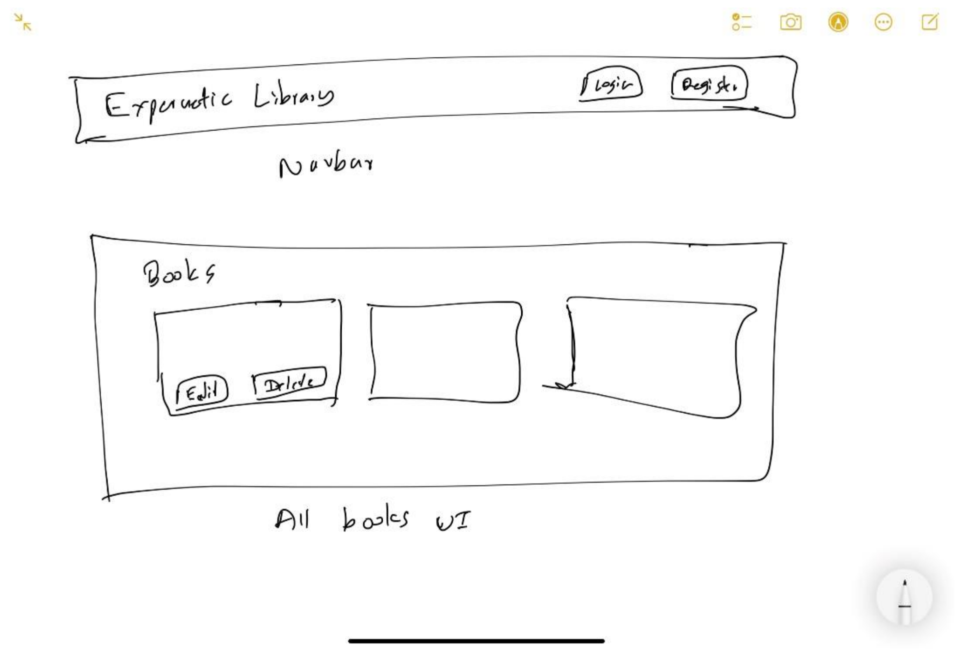
After successfully initializing frontend (React + TypeScript) in new branch (frontend-dev), then setup the Tailwind CSS for the styling. Below mentioned resources help to set up the frontend with Tailwind.

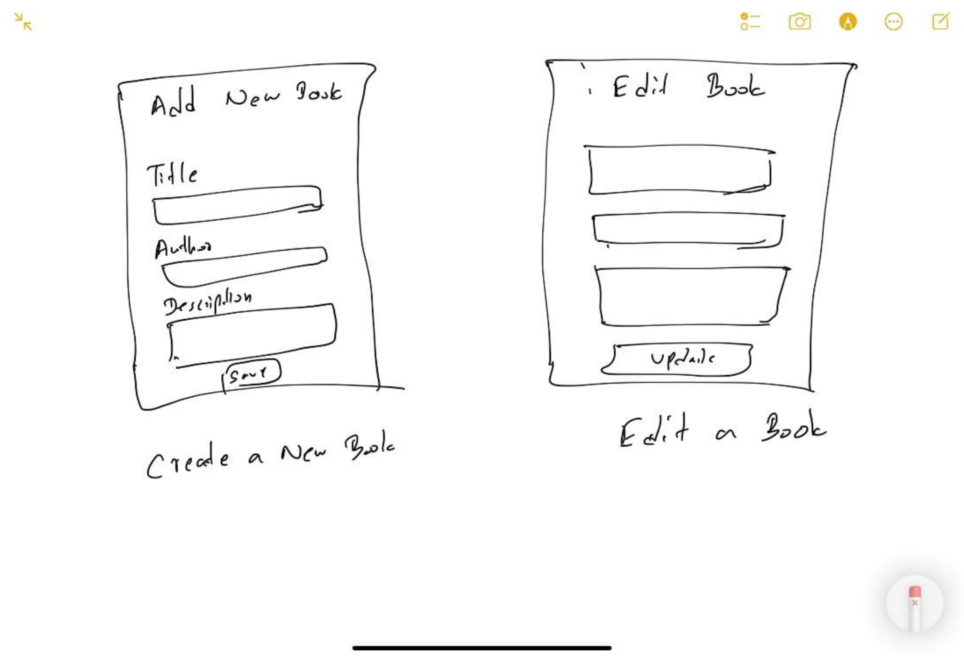
Tailwind Official Document - https://tailwindcss.com/docs/installation/using-vite

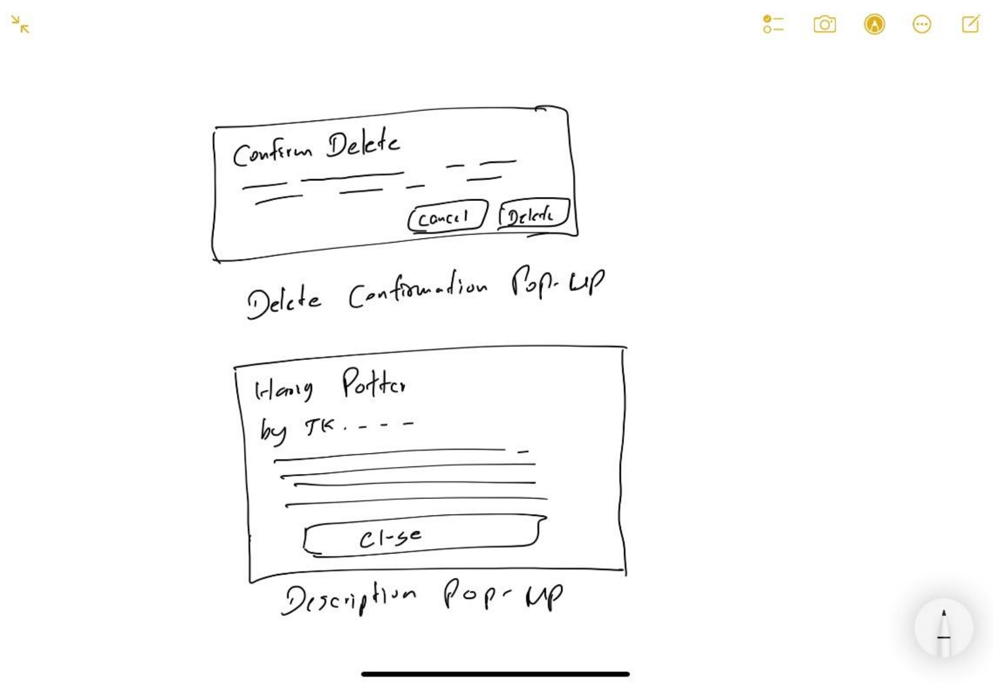
Low Fidelity Design

The next step came up with low fidelity design for user registration, login, view all books, create a new book, update a book, delete book, navbar, book description pop-up window and delete confirmation pop-up modal.




Expressatic Library
Login
Register
Navbar
Books
Edit
Delete
All books UI

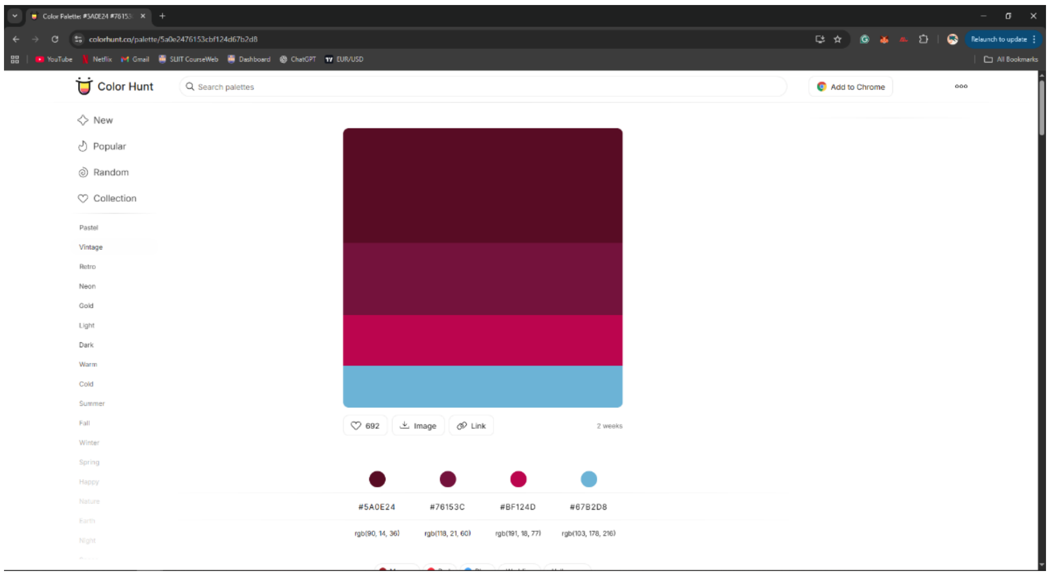

Add New Book
Title
Author
Description
Save
Create a New Book
Edit Book
Update
Edit a Book



Color Palette

Before moving to the High-Fidelity design, I came with a color palette using color hunt web site.

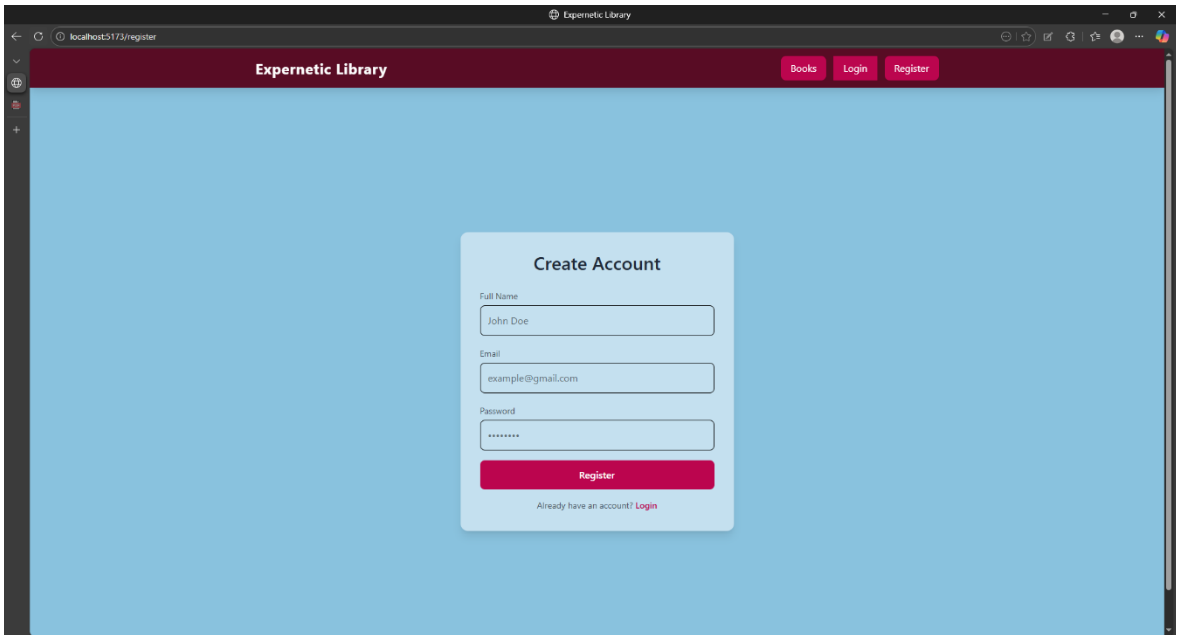
Website - https://colorhunt.co/palette/5a0e2476153cbf124d67b2d8

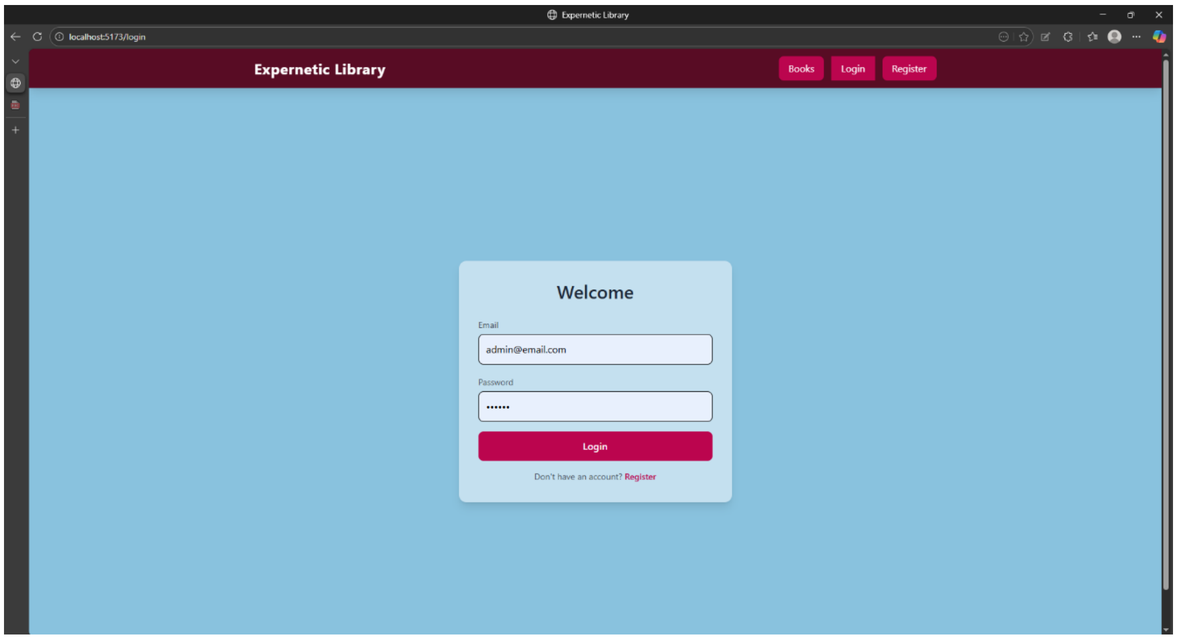


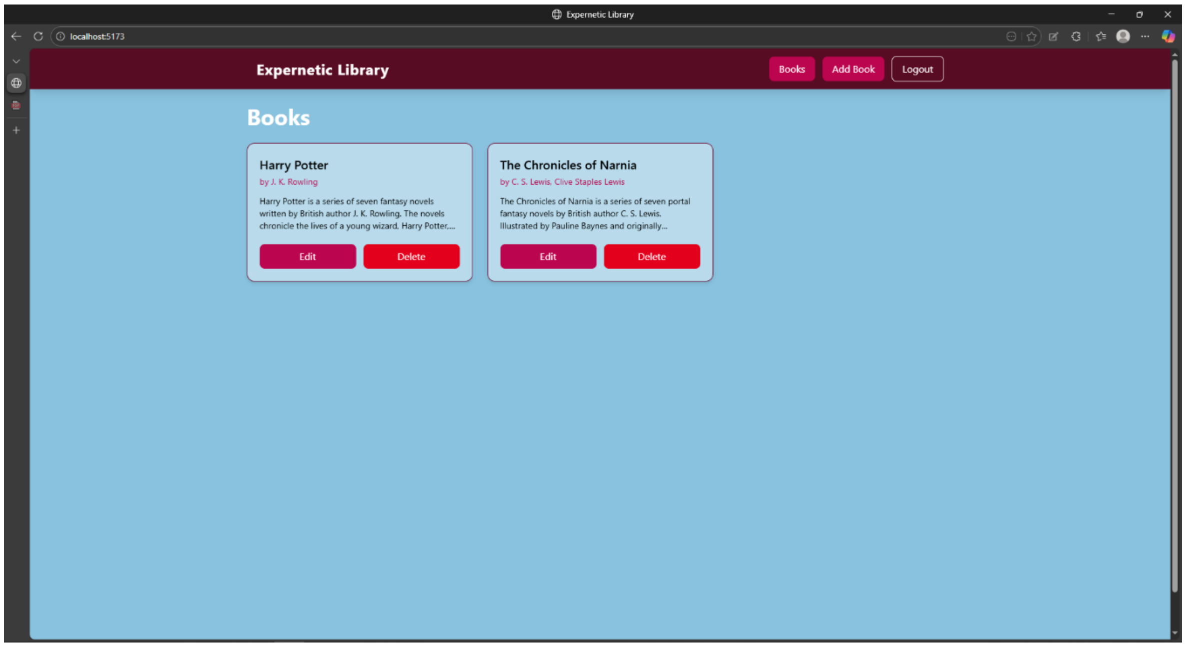
#5A0E24	#76153C	#BF124D	#67B2D8
rgb(90, 14, 36)	rgb(118, 21, 60)	rgb(191, 18, 77)	rgb(103, 178, 216)

High Fidelity Design

AS the final step developed the designed UIs with Tailwind Stylings.







Books

Harry Potter

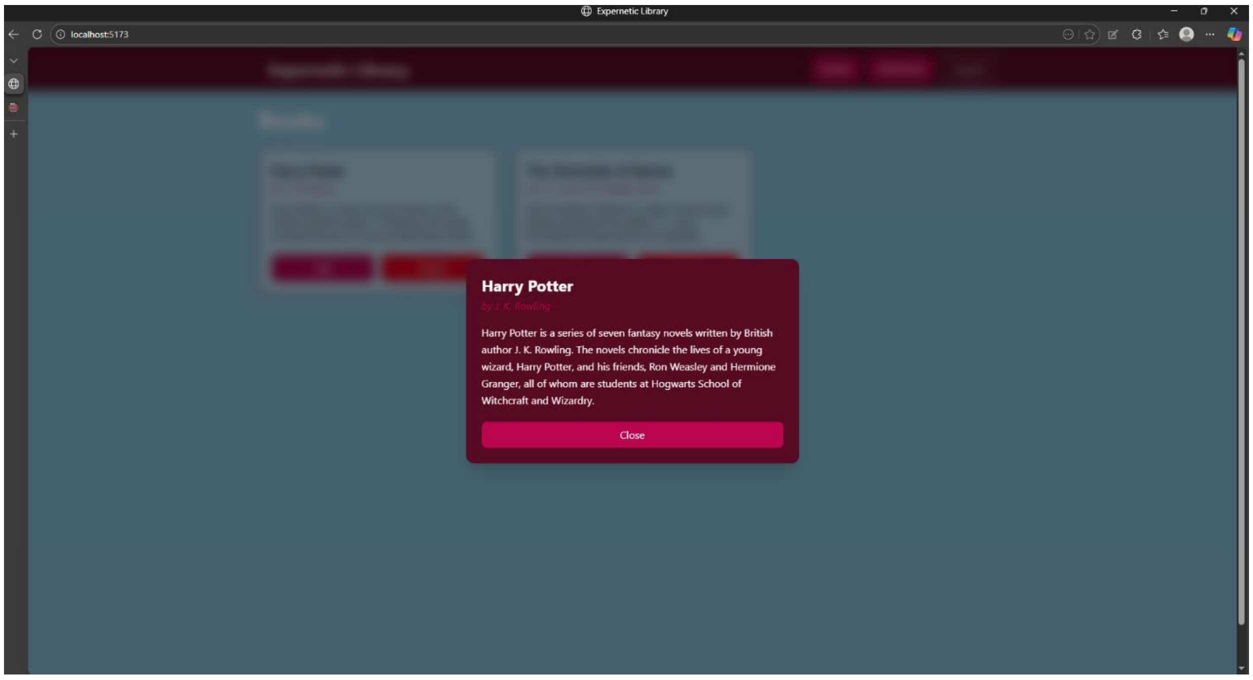
by J. K. Rowling

Harry Potter is a series of seven fantasy novels written by British author J. K. Rowling. The novels chronicle the lives of a young wizard, Harry Potter,...

The Chronicles of Narnia

by C. S. Lewis, Clive Staples Lewis

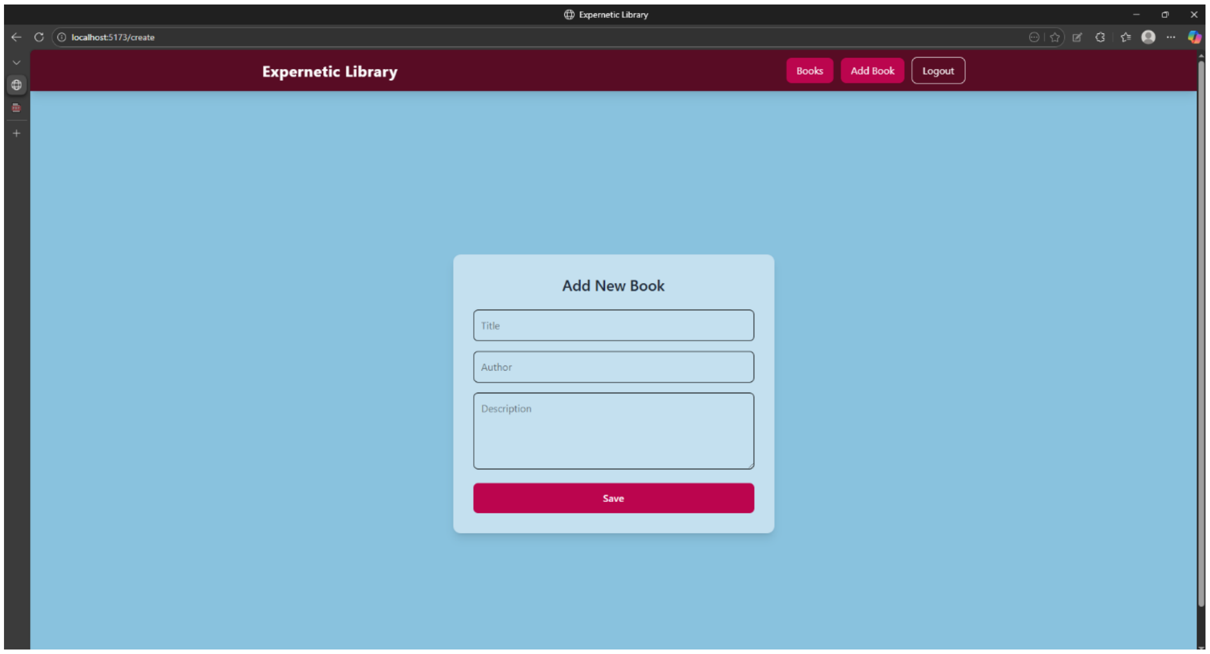
The Chronicles of Narnia is a series of seven portal fantasy novels by British author C. S. Lewis. Illustrated by Pauline Baynes and originally...

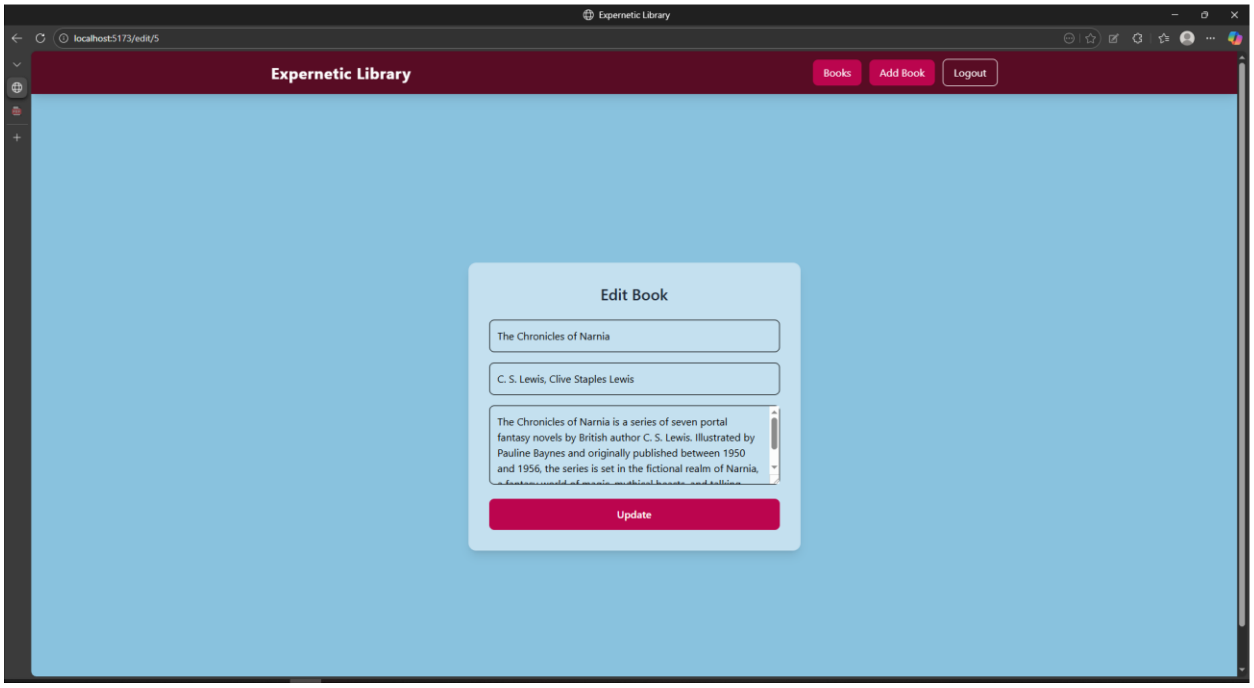


Harry Potter

by J. K. Rowling

Harry Potter is a series of seven fantasy novels written by British author J. K. Rowling. The novels chronicle the lives of a young wizard, Harry Potter, and his friends, Ron Weasley and Hermione Granger, all of whom are students at Hogwarts School of Witchcraft and Wizardry.

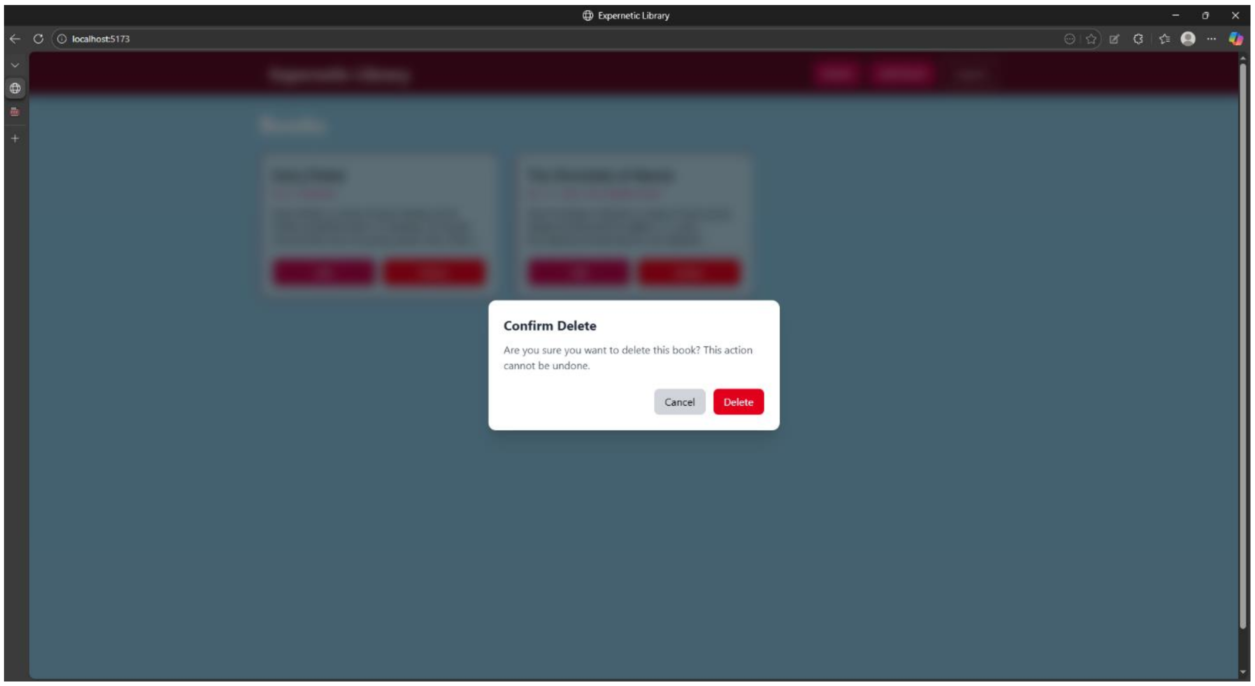




The Chronicles of Narnia

C. S. Lewis, Clive Staples Lewis

The Chronicles of Narnia is a series of seven portal fantasy novels by British author C. S. Lewis. Illustrated by Pauline Baynes and originally published between 1950 and 1956, the series is set in the fictional realm of Narnia,



Confirm Delete

Are you sure you want to delete this book? This action cannot be undone.

Application Overview

A full-stack library management system built with ASP.NET Core Web API (backend) and React + TypeScript (frontend) featuring JWT-based authentication. Users can register, login, view books, and perform CRUD operations on books after authentication.

Getting Started

These steps are also available in the github README file.

Backend Setup

1. Install .NET 8 SDK.
2. Navigate to the backend – [Terminal command – cd backend].
3. Install Dependencies and restore packages – [command - dotnet restore].
4. Manually create "appsettings.json" file in backend root directory.

   (replace placeholders with your own values)

```
{
 "ConnectionStrings": {
  "DefaultConnection": "Data Source=YOUR_DATABASE_FILE.db"
 },
 "Jwt": {
  "Key": "YOUR_SECRET_JWT_KEY",
  "Issuer": "YOUR_ISSUER",
  "Audience": "YOUR_AUDIENCE",
  "ExpiryMinutes": 60
 },
 "AllowedHosts": "*"
}
```

5. Manually create "appsettings.Development.json" file in backend root directory.

```
{
 "Logging": {
  "LogLevel": {
   "Default": "Information",
   "Microsoft.AspNetCore": "Warning"
  }
 }
}
```

6. Apply Migrations. (Run below commands in backend terminal)

```
dotnet ef migrations add InitialCreate
dotnet ef database update
```

7. Run the application – [command - dotnet run].

Frontend Setup

1. Navigate to the frontend folder – [cd frontend].
2. Install dependencies – [npm install].
3. Run the frontend development server – [npm run dev].

Additional Features

1. Implement Registration and Login using JWT.
2. Protected backend routes with [Authorize].
3. Frontend routes are protected with PrivateRoute.
4. Implemented a pop-up box for book description.
5. Implemented a delete confirmation dialog box.