System Requirements Specification Index

For

Tax Management Application

Version 1.0

IIHT Pvt. Ltd.
IIHT Ltd, No: 15, 2nd Floor, Sri Lakshmi Complex, Off MG Road, Near SBI LHO,
Bangalore, Karnataka – 560001, India
fullstack@iiht.com

TABLE OF CONTENTS

Tax Management Application

System Requirements Specification

PROJECT ABSTRACT

In the world of financial management, there's a pressing need to modernize tax handling. Dr. Smith, the CFO of a leading financial institution, challenges a team of developers to create a Fullstack Tax Management Application.

Your task is to develop a digital solution that seamlessly manages tax calculations and related specifications, providing users with an intuitive platform for effective tax management.

BACKEND-DOTNET

1. Business-Requirement:

Tax Management Application is .Net Core web API 3.1 application integrated with MS SQL Server, where it refers to the professional management of various securities and assets to meet specific Tax goals for individuals, institutions, or organizations. This process includes the creation, updating, retrieval, and deletion of tax related properties.

To build a robust backend system that effortlessly handles tax calculations. Here's what the developers need to accomplish:

	Tax Management
Modules	
1	Tax
Tax Module Functionalities	
1	Create an Tax
2	Update the existing Tax
3	Get an Tax by Id
4	Fetch all Insurance Policies
5	Delete an existing Tax

2. Assumptions, Dependencies, Risks / Constraints

2.1 Tax Constraints:

- While deleting the Tax, if Tax Id does not exist then the operation should throw a custom exception.
- While fetching the Tax details by id, if Tax id does not exist then the operation should throw a custom exception.

2.2 Common Constraints

- For all rest endpoints receiving @RequestBody, validation check must be done and must throw custom exception if data is invalid
- All the business validations must be implemented in model classes only.
- All the database operations must be implemented on entity object only
- Do not change, add, remove any existing methods in service layer
- In Repository interfaces, custom methods can be added as per requirements.
- All RestEndpoint methods and Exception Handlers must return data wrapped in **ResponseEntity**

3. Business Validations

Tax Class Entities

- Tax Form Id (long) Not null, Key attribute.
- User Id (int) Not null.
- Form Type (string) is not null, min 3 and max 100 characters.
- Total Tax Amount (decimal) is not null.
- Filling Date (Date)

4. Considerations

- There is no roles in this application
- You can perform the following possible actions

Tax

5. Rest Endpoints

Rest End-points to be exposed in the controller along with method details for the same to be created

5.1 TaxController

<table>
<tr><th>URL Exposed</th><th>Purpose</th></tr>
<tr><td>/create-tax
<table><tr><td>Http Method</td><td>POST</td></tr><tr><td>Parameter 1</td><td>Tax model</td></tr><tr><td>Return</td><td>HTTP Response StatusCode</td></tr></table></td><td>Create Tax</td></tr>
<tr><td>/update-tax
<table><tr><td>Http Method</td><td>PUT</td></tr><tr><td>Parameter 1</td><td>Long Id</td></tr><tr><td>Parameter 2</td><td>TaxViewModel model</td></tr><tr><td>Return</td><td>HTTP Response StatusCode</td></tr></table></td><td>Update a Tax</td></tr>
<tr><td>/get-all-taxes
<table><tr><td>Http Method</td><td>GET</td></tr><tr><td>Parameter 1</td><td>-</td></tr><tr><td>Return</td><td><IEnumerable<Tax >></td></tr></table></td><td>Fetches the list of all Taxes</td></tr>
<tr><td>/get-tax-by-id?id={id}
<table><tr><td>Http Method</td><td>GET</td></tr><tr><td>Parameter 1</td><td>Long (id)</td></tr><tr><td>Return</td><td><Tax></td></tr></table></td><td>Fetches the details of a Tax</td></tr>
<tr><td>/delete-tax?id={id}
<table><tr><td>Http Method</td><td>DELETE</td></tr><tr><td>Parameter 1</td><td>Long (id)</td></tr><tr><td>Return</td><td>HTTP Response StatusCode</td></tr></table></td><td>Delete a Tax</td></tr>
</table>

6. Template Code Structure

6.1 Package: TaxManagement

Resources

Names	Resource	Remarks	Status
Package Structure			
controller	TaxController	Controller class to expose all rest-endpoints for auction related activities.	Partially implemented
Startup.cs	Startup CS file	Contain all Services settings and SQL server Configuration.	Already Implemented
Properties	launchSettings.json file	All URL Setting for API	Already Implemented
	appsettings.json	Contain connection string for database	Already Implemented

6.2 Package: TaxManagement.BusinessLayer

Resources

Names	Resource	Remarks	Status
Package Structure			
Interface	ITaxServices interface	Inside all these interface files contains all business validation logic functions.	Already implemented

Service	Tax Services CS file	Using this all class we are calling the Repository method and use it in the program and on the controller.	Partially implemented
Repository	ITax Repository Tax Repository (CS files and interfaces)	All these interfaces and class files contain all CRUD operation code for the database. Need to provide implementation for service related functionalities	Partially implemented
ViewModels	Tax ViewModel	Contain all view Domain entities for show and bind data. All the business validations must be implemented.	Partially implemented

6.3 Package: TaxManagement.DataLayer

Resources

Names	Resource	Remarks	Status
Package Structure			
DataLayer	TaxDBContext cs file	All database Connection,collection setting class	Already Implemented

6.4 Package: TaxManagement.Entities

Resources

Names	Resource	Remarks	Status
Package Structure			
Entities	Tax ,Response (CS files)	All Entities/Domain attribute are used for pass the data in controller and status entity to return response Annotate this class with proper annotation to declare it as an entity class with **Id** as primary key. Generate the **Id** using the **IDENTITY** strategy	Partially implemented

1. PROBLEM STATEMENT

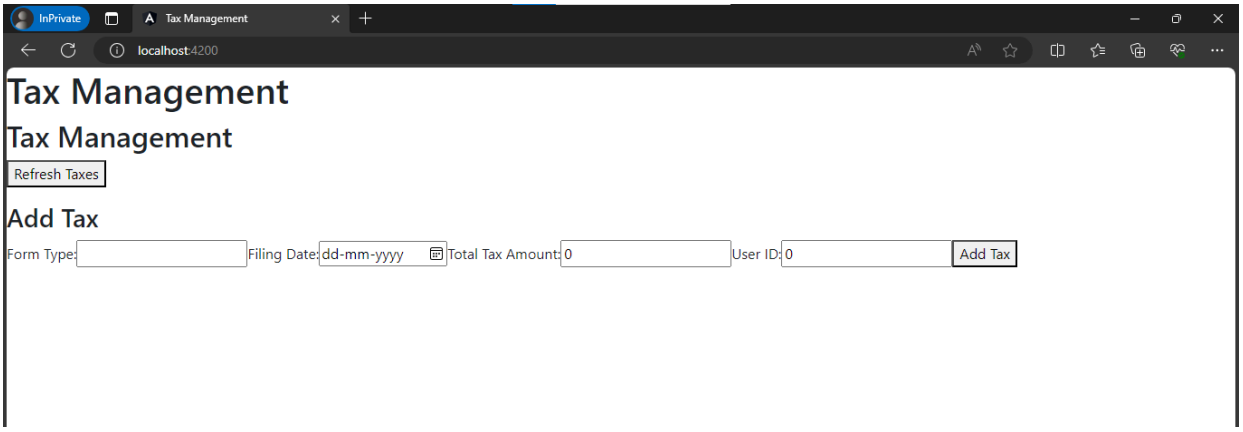
The **Tax Management Application** frontend is a Single Page Application (SPA) built using Angular. Here's what the frontend developers need to achieve:

The frontend should provide a user-friendly interface for users to manage tax-related tasks like: add tax details, update tax details, delete tax and get all taxes.

2. PROPOSED TAX MANAGEMENT WIREFRAME

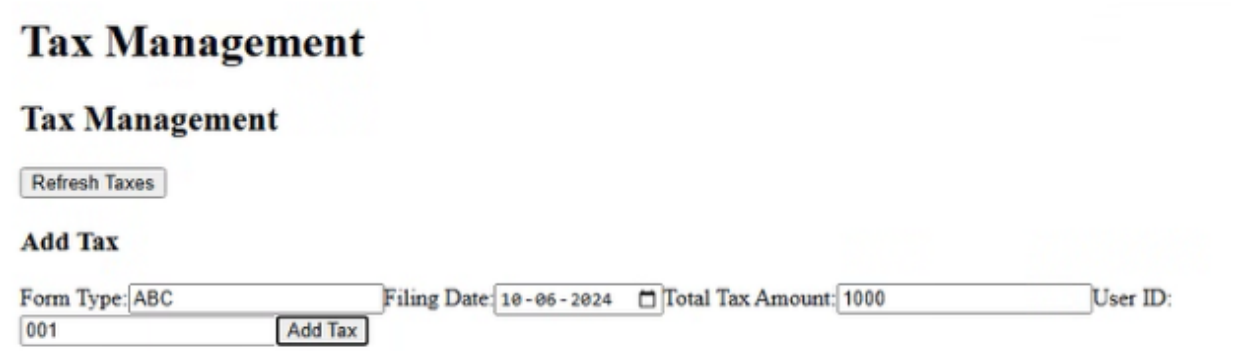
UI needs improvisation and modification as per given use case and to make test cases passed.

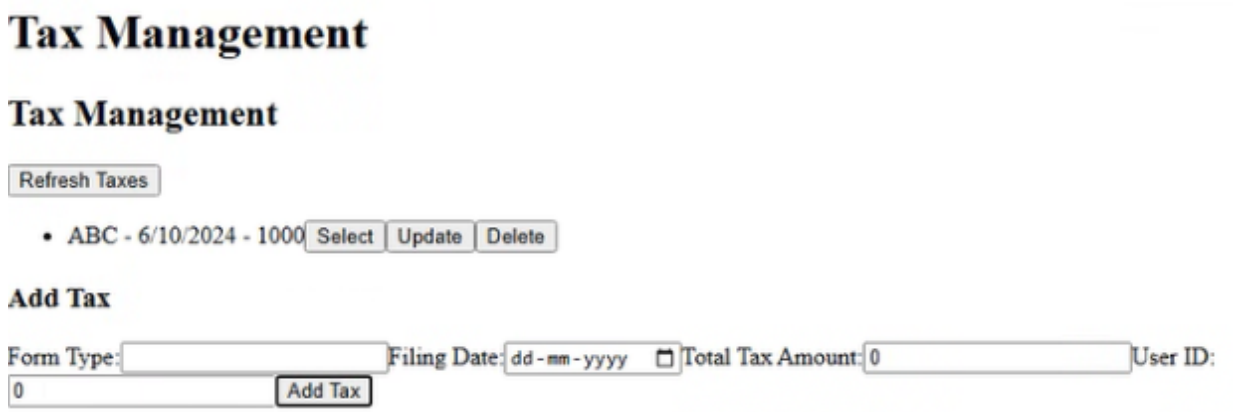
2.1 HOME PAGE



2.2 SCREENSHOTS

***** Add Tax*****





***** Update Tax*****

Tax Management

Tax Management

Refresh Taxes

- ABC - 6/10/2024 - 1000 Select Update Delete

Update Tax

Form Type: ABC Filing Date: 10-06-2024 Total Tax Amount: 1010 User ID: 1 Update Tax

Tax Management

Tax Management

Refresh Taxes

- ABC - 6/10/2024 - 1010 Select Update Delete

Add Tax

Form Type: Filing Date: dd-mm-yyyy Total Tax Amount: 0 User ID: 0 Add Tax

***** Select Tax*****

Tax Management

Tax Management

Refresh Taxes

- ABC - 6/10/2024 - 1000 Select Update Delete

Update Tax

Form Type: ABC Filing Date: 10-06-2024 Total Tax Amount: 1010 User ID: 1 Update Tax

***** Delete Tax*****

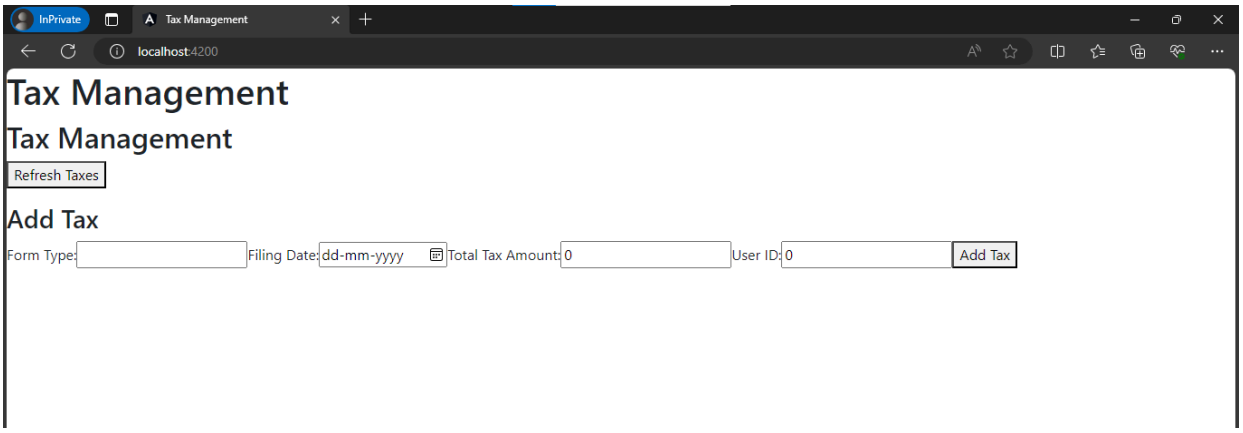
Tax Management

Tax Management

Refresh Taxes

Add Tax

Form Type: Filing Date: mm/dd/yyyy Total Tax Amount: 0 User ID: 0 Add Tax



3. BUSINESS-REQUIREMENT :

As an application developer, develop the Tax Management Application (Single Page App) with below guidelines:

User Story #	User Story Name	User Story
US_01	Home Page	As a user I should be able to visit the Home page as the default page.
US_01	Home Page	As a user I should be able to see the homepage and perform all operations: Acceptance criteria: 1. The page should display the title **"Tax Management"** in **h1**. 2. A **Refresh Taxes** button should be present to reload and display the latest tax entries. **> Tax List**: 1. Should show a list of all policies with “Update” & “Delete” button in each of the taxes. 2. The **Tax List** should display all tax records with the following details for each entry: ● Form Type ● Filing Date ● Total Tax Amount ● User ID 3. Each tax record should have **Select**, **Update**, and **Delete** buttons: ● **Select**: Loads the details of the selected tax record into the input fields for viewing. ● **Update**: Opens a form to edit the existing tax record. ● **Delete**: Removes the selected tax record from the list.

		> Add Tax: 1. The page should display the title **"Add Tax"** in **h3**. 2. Users should be able to add a new tax entry by filling in the following fields: ● **Form Type** (text input) ● **Filing Date** (date picker in `dd-mm-yyyy` format) ● **Total Tax Amount** (numeric input) ● **User ID** (numeric input) 3. There should be an **"Add Tax"** button, which should allow you to add only when all fields are filled with valid data. 4. Once the Add tax button is clicked, it should add the tax details to the tax list immediately. **> Select Tax**: 1. Users should be able to view an existing tax entry using the **Select** button. 2. Clicking the **Select** button next to a tax record should load the details into the input fields for viewing. **> Update Tax**: 1. Users should be able to update an existing tax entry by: ● Clicking the **Update** button next to the respective tax record. ● The details of the selected tax record should load into the **Update Tax** form. ● Editing the fields as necessary and clicking the **"Update Tax"** button to save changes. 2. The **"Update Tax"** button should only allow you to update only when all fields are filled and valid. 3. Once updated, the changes should reflect in the tax list immediately.

		> Delete Tax: 1. Users should be able to delete a tax record by clicking the **Delete** button next to the respective entry. 2. Once deleted, the tax record should be removed from the list, and the display should update automatically. **> Refresh Taxes**: 1. Clicking the **Refresh Taxes** button should reload the tax list and display the most up-to-date records without refreshing the entire page. **** Kindly refer to the screenshots for any clarifications. ****

Execution Steps to Follow For Backend

1. **All actions like build, compile, running application, running test cases will be through Command Terminal.**

2. **To open the command terminal the test takers need to go to the Application menu (Three horizontal lines at left top) Terminal → New Terminal.**

3. **On command prompt, cd into your project folder (cd <Your-Project-folder>).**

4. **To connect SQL server from terminal:**
   (TaxManagement /sqlcmd -S localhost -U sa -P pass@word1)
   - **To create database from terminal -**
     1> Create Database TaxDb
     2> Go

5. **Steps to Apply Migration(Code first approach):**
   - **Press Ctrl+C to get back to command prompt**
   - **Run following command to apply migration-**
     (TaxManagement /dotnet-ef database update)

6. **To check whether migrations are applied from terminal:**
   (TaxManagement /sqlcmd -S localhost -U sa -P pass@word1)

```
1> Use TaxDb
2> Go
1> Select * From __EFMigrationsHistory
2> Go
```

7. **To build your project use command:**
   (TaxManagement /dotnet build)

8. **To launch your application, Run the following command to run the application:**
   (TaxManagement /dotnet run)

9. **This editor Auto Saves the code.**

10. **To test any Restful application, the last option on the left panel of IDE, you can find ThunderClient, which is the lightweight equivalent of POSTMAN.**

11. **To test web-based applications on a browser, use the internal browser in the workspace. Click on the second last option on the left panel of IDE, you can find Browser Preview, where you can launch the application.**

    Note: The application will not run in the local browser

12. **To run the test cases in CMD, Run the following command to test the application:**
    (TaxManagement .Tests/dotnet test --logger "console;verbosity=detailed")
    (You can run this command multiple times to identify the test case status,and refactor code to make maximum test cases passed before final submission)

13. **If you want to exit(logout) and continue the coding later anytime (using Save & Exit option on Assessment Landing Page) then you need to use CTRL+Shift+B - command compulsorily on code IDE. This will push or save the updated contents in the internal git/repository. Else the code will not be available in the next login.**
14. **These are time bound assessments the timer would stop if you logout and while logging in back using the same credentials the timer would resume from the same time it was stopped from the previous logout.**

15. **You need to use CTRL+Shift+B - command compulsorily on code IDE, before final submission as well. This will push or save the updated contents in the internal git/repository, and will be used to evaluate the code quality.**

EXECUTION STEPS TO FOLLOW FOR FRONTEND

1. **All actions like build, compile, running application, running test cases will be through Command Terminal.**
2. **To open the command terminal the test takers, need to go to**

   Application menu (Three horizontal lines at left top) -> Terminal ->New Terminal.

3. **This is a web-based application, to run the application on a browser, use the internal browser in the environment.**
4. **Follow the steps below to install and use Node.js version 18.20.3 using nvm:**
   a. **Install nvm:**
      curl -o- https://raw.githubusercontent.com/nvm-sh/nvm/v0.39.1/install.sh | bash
   b. **Set up nvm environment:**
      export NVM_DIR="$([-z "${XDG_CONFIG_HOME-}"] && printf %s "${HOME}/.nvm" || printf %s "${XDG_CONFIG_HOME}/nvm")" && [-s "$NVM_DIR/nvm.sh"] && \. "$NVM_DIR/nvm.sh"
   c. **Verify nvm Installation:**

      command -v nvm

   d. **Install Node.js Version 18.20.3:**

      nvm install 18.20.3

   e. **Set the installed Node.js version as active:**

      nvm use 18.20.3

5. **You can follow series of command to setup Angular environment once you are in your project-name folder:**
    a. **npm install -> Will install all dependencies -> takes 10 to 15 min**
    b. **npm run start -> To compile and deploy the project in browser. You can press <Ctrl> key while clicking on localhost:4200 to open project in browser -> takes 2 to 3 min**
    c. **npm run test -> to run all test cases. It is mandatory to run this command before submission of workspace -> takes 5 to 6 min**
6. **You need to use CTRL+Shift+B - command compulsorily on code IDE, before final submission as well. This will push or save the updated contents in the internal git/repository, and will be used to evaluate the code quality.**
