System Requirements Specification Index

For

Insurance Policy Application

Version 1.0

IIHT Pvt. Ltd.
fullstack@iiht.com

Contents

1 BUSINESS-REQUIREMENT:

1.1 PROBLEM STATEMENT:

The purpose of this application is to provide a platform to find all insurance policies of all types. Where a company can add, edit, delete and search any policy.

The Insurance Policy Application is designed to streamline the management of insurance policies for users.

You are tasked to create a user-friendly platform that allows access to a home page and enables the addition of new insurance policies. Each policy should include fields such as firstName, lastName, amount, interest, tenure, startDate, nominee, and policyName. The application should allow users to view a comprehensive list of all policies. It should also include functionalities to edit and delete any policy. A search feature should be integrated, making it possible for users to locate specific policies by person name or policy name, enhancing the efficiency of managing insurance details.

1.1.1 Insurance Policy Application:

The Insurance Policy Application allows you to:

1. Access the home page.
2. Should be able to add new insurance.
3. It can have basic fields like firstName, lastName, amount, interest, tenure, startDate, nominee and policyName.
4. Should be able to get the list of policies along with options to sort in ascending and descending order in each field.
5. Should be able to edit and delete any policy.
6. Should be able to search for a policy by person name or policy name.

2. Template Code Structure:

2.1 InsurancePolicy Controller

Method Exposed	Purpose
listInsurancePolicies()	Should return page “list-insurancePolicies” with required data.
showFormForAdd()	Should return page “add-insurancePolicy-form” for adding a policy.
saveInsurancePolicy()	Should save a tutor and return “insurancePolicy/list” with required data.
showFormForUpdate()	Should show policy details in page “update-insurancePolicy-form” to edit a tutor.
deleteInsurancePolicy()	Should delete a policy and return “insurancePolicy/list” with required data.
searchInsuracePolicys()	Should search for a policy and return “list-insurancePolicies” with required data.
updateAvailability()	Should show and update the status of insurance policy (Claim Now/Policy Claimed)

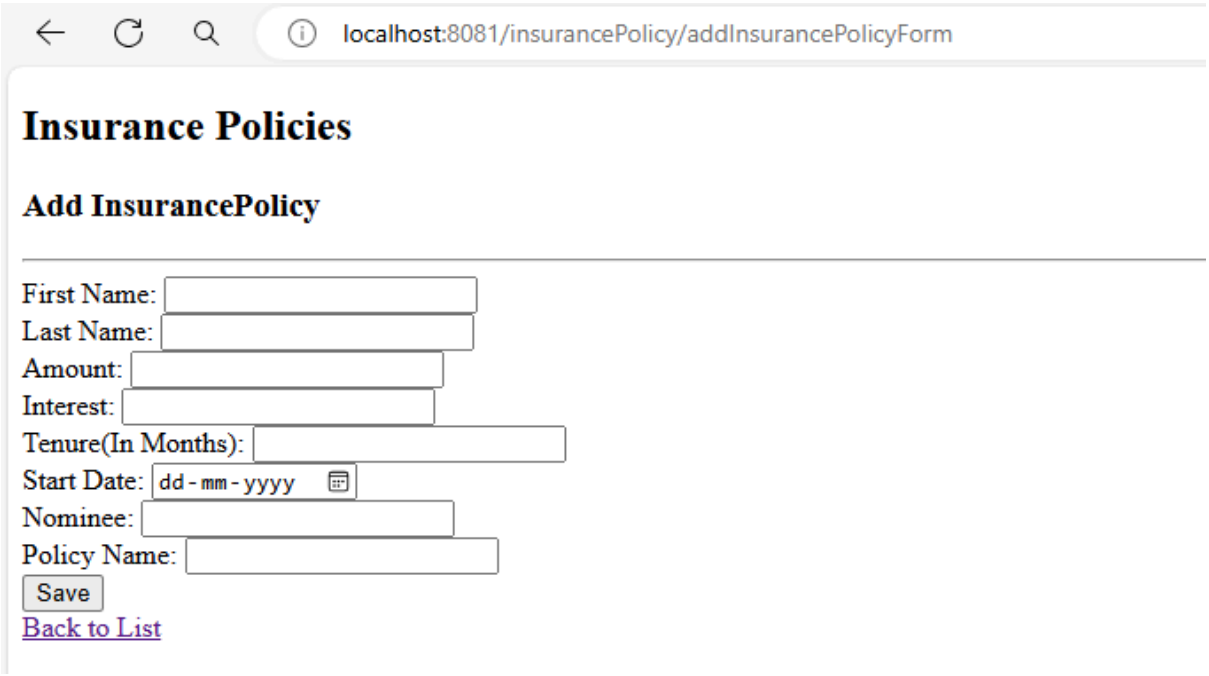
2.2 Resources Available:

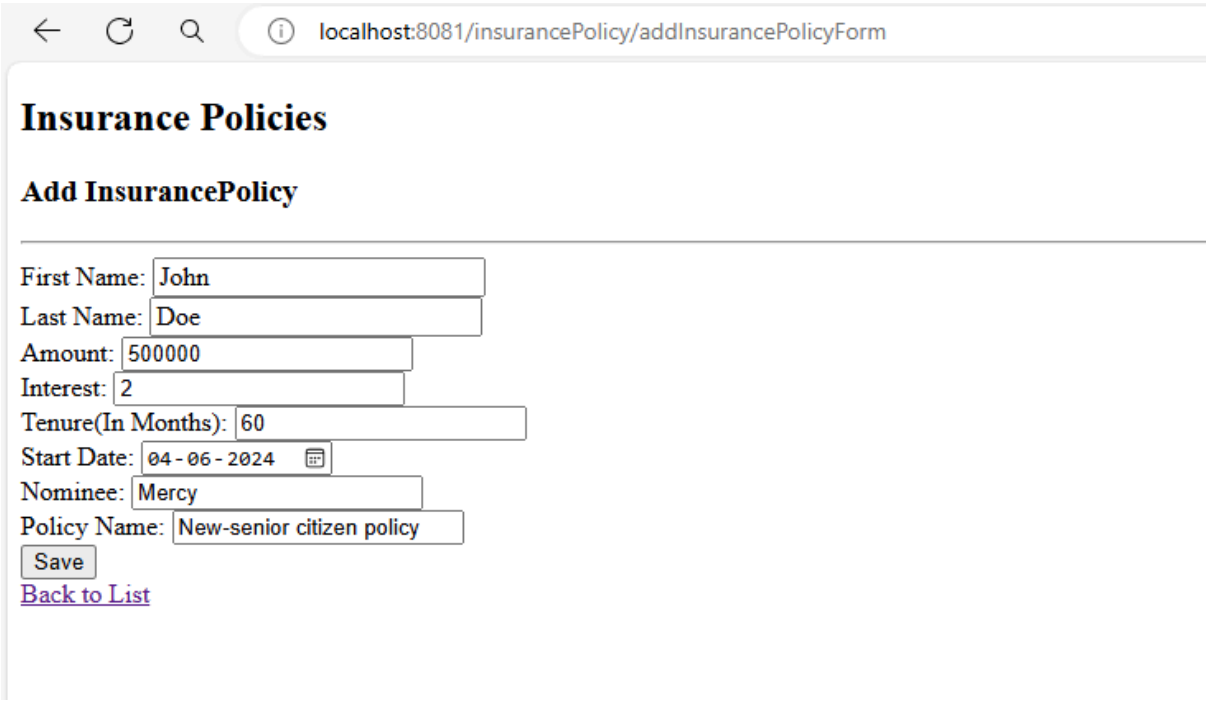
Description	View Pages Name	Remarks
Common UI		
Home Page	list-insurancePolicies	Contains a homepage which shows a list of all policies along with options to add, edit , delete and search a policy.
All policies	list-insurancePolicies	
Add a policy	add-insurancePolicy-form	
Update a policy	update-insurancePolicy-form	
Search a policy	list-insurancePolicies	

3 SUGGESTED WIREFRAMES:

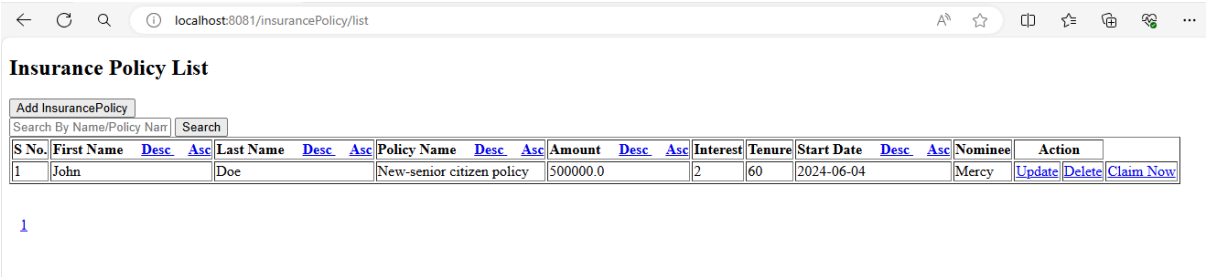
1. Homepage – Visitor Landing Page

2. Create an Insurance

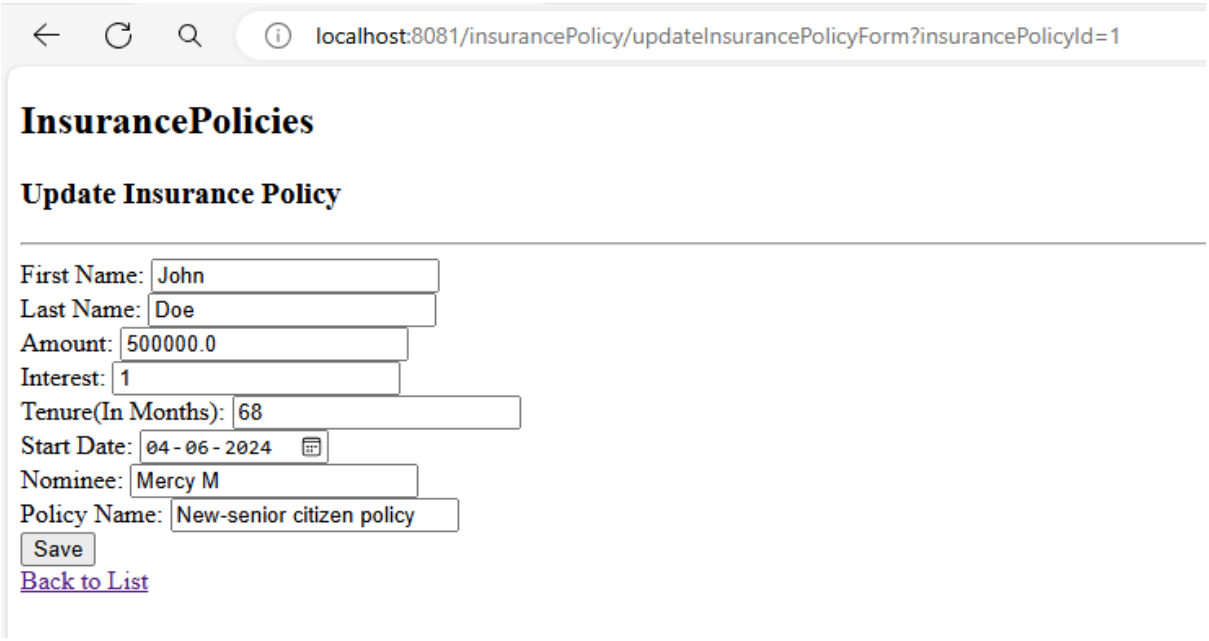


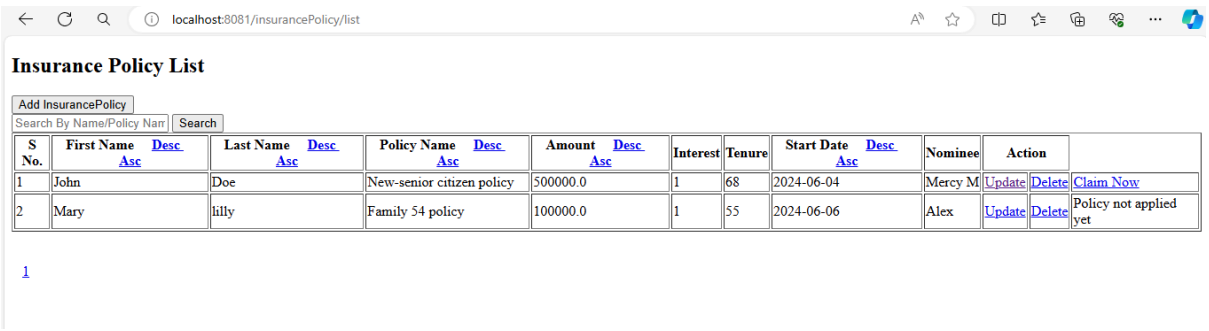


3. Insurance List

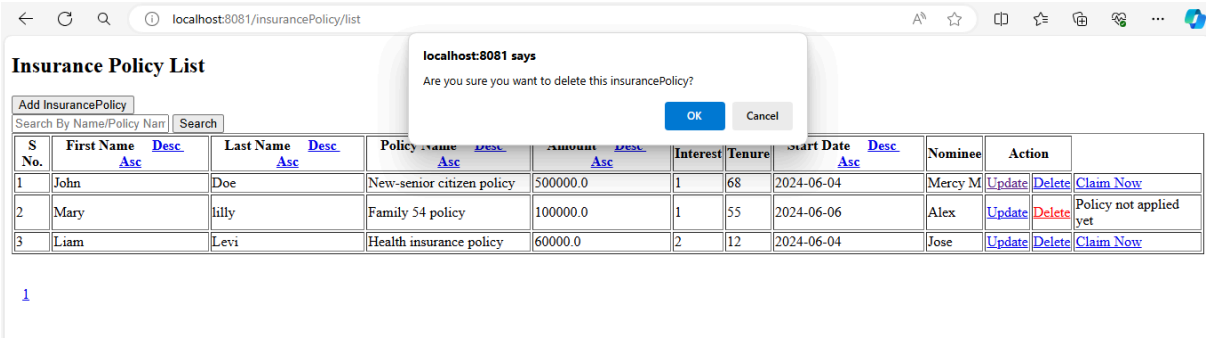


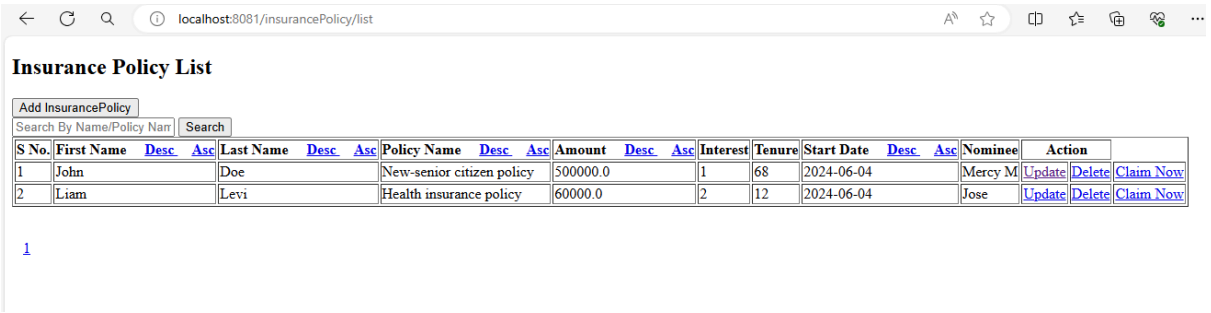
4. Update an Insurance





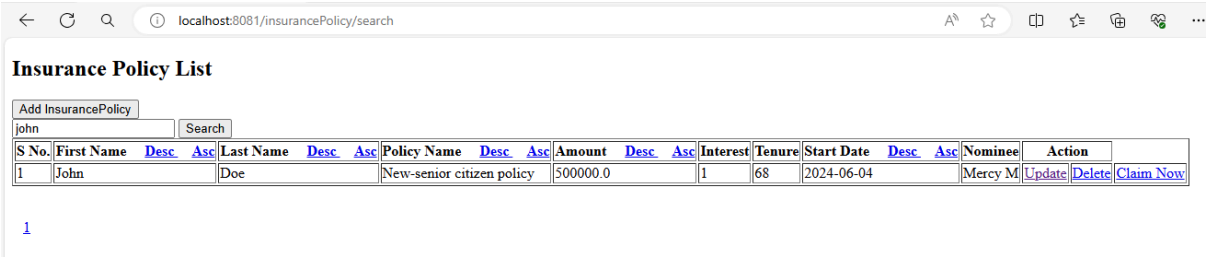
5. Delete an Insurance



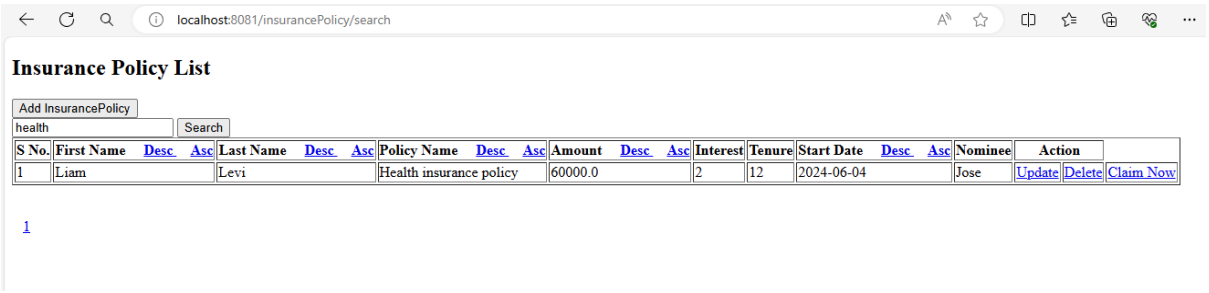


6. Search an Insurance

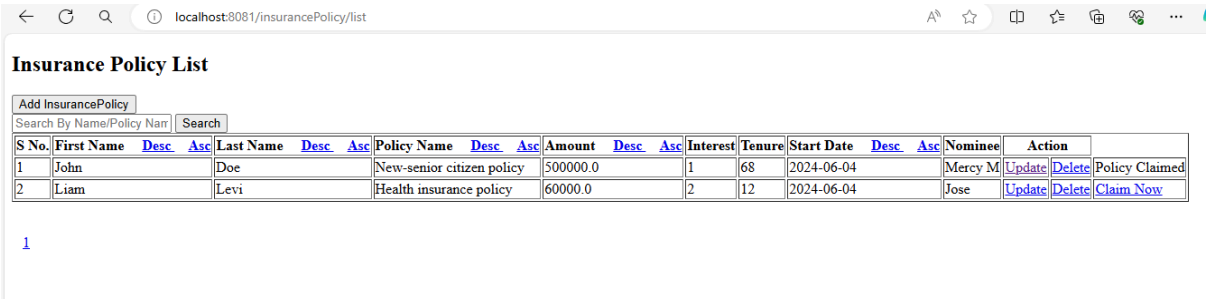
*****By First/Last Name*****



*****By Policy Name*****



7. Claim an Insurance Policy



4 Business Validations

1. Id must be of type id.
2. First name value should not be blank, min 3 and max 50 characters.
3. Last name value should not be blank, min 3 and max 50 characters.
4. Amount value is not null.
5. Interest value is not null.
6. Tenure value is not null.
7. Start date value is not null and in format (yyyy-mm-dd).
8. Nominee should not be blank.
9. Policy Name should not be blank.

5 Considerations

The Code template already contains skeleton methods for service and controller layer. Please write your logic in it.

6 Execution Steps to Follow

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. **To build your project use command:**
   mvn clean package -Dmaven.test.skip
4. **To launch your application:**
   java -jar <your application war file name>
5. **This editor Auto Saves the code**
6. **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.**
7. **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.**

8. **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.**
9. **This is a web-based application, to run the application 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

10. **Default credentials for MySQL:**
    a. **Username: root**
    b. **Password: pass@word1**
11. **To login to mysql instance: Open new terminal and use following command:**
    a. **sudo systemctl enable mysql**
    b. **sudo systemctl start mysql**

       NOTE: After typing any of the above commands you might encounter any warnings.

       >> Please note that these warnings are expected and can be disregarded. Proceed to the next step.

    c. **mysql -u root -p**
       The last command will ask for password which is 'pass@word1'
12. **Mandatory: Before final submission run the following command:**
    mvn test
13. **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.**