System Requirements Specification

Index

For

Insurance Management Application

Version 1.0

TABLE OF CONTENTS

Table of Contents

INSURANCE POLICY MANAGEMENT
System Requirements Specification
SPRING BOOT RESTFUL APPLICATION

1 Project Abstract

The **Insurance Policy Management** is an application with a backend implemented using Spring Boot with a MySQL database. The application aims to provide a comprehensive platform for managing and organizing all insurance policies for a company.

The **Insurance Policy Management** project presents developers with a vital task: to design and implement a comprehensive set of test cases using Junit&Mockito to validate the functionality of the application.
Your task is to develop a robust suite of test cases that thoroughly evaluate the insurance policy management activities under various scenarios, ensuring accurate results and error-free performance.
The test suite aims to ensure the accuracy and reliability of the system, providing confidence in its performance and enhancing customer satisfaction.

<u>Following is the requirement specifications</u>:

	Insurance Policy Management
Modules	
1	Insurance Policy
Insurance Policy Module Functionalities	
1	Get all policies
2	Get policy details by id
3	Create a new policy
4	Update a policy by id
5	Delete a policy by id

2 REST ENDPOINTS

Rest End-points exposed in the controller along with method details for the same :

2.1 INSURANACECONTROLLER

URL Exposed		Purpose
1. /api/policies		Fetches all the policies
Http Method	GET	
Parameter	-	
Return	List<InsurancePolicyDTO>	
2. /api/policies/{id}		Fetches a policy by id
Http Method	GET	
Parameter 1	Long (id)	
Return	InsurancePolicyDTO	
3. /api/policies/		Creates a new policy
Http Method	POST **The policy data to be created should be received in @RequestBody**	
Parameter	-	
Return	InsurancePolicyDTO	
4. /api/policies/{id}		Updates a policy by id
Http Method	PUT **The policy data to be updated should be received in @RequestBody**	
Parameter 1	Long (id)	
Return	InsurancePolicyDTO	
5. /api/policies/{id}		Deletes a policy by id
Http Method	DELETE	
Parameter 1	Long (id)	
Return	-	

3 TEMPLATE CODE STRUCTURE

3.1 PACKAGE: com.insurancepolicy

Resources

insurancePolicyManagementApplication (Class)	This is the Spring Boot starter class of the application.	Already Implemented

3.2 PACKAGE: com.insurancepolicy.test

Resources

InsurancePolicyTests	➔This class needs to contains Junit & Mockito test cases to verify the correctness of the methods in the **InsurancePolicyController** and **InsurancePolicyServiceImpl** classes ➔Make sure the test cases you write achieves 100% code coverage. .	Need to implement

3.3 PACKAGE: com.insurancepolicy.repository

Resources

Class/Interface	**Description**	**Status**
InsurancePolicyRepository (interface)	● Repository interface exposing CRUD functionality for insurance policy Entity.	Already Implemented

3.4 PACKAGE: com.insurancepolicy.service

Resources

Class/Interface	**Description**	**Status**
InsurancePolicyService (interface)	● Interface to expose method signatures for insurance policy related functionality. ● Do not modify, add or delete any method.	Already implemented.

3.5 PACKAGE: com.insurancepolicy.service.impl

Class/Interface	**Description**	**Status**
InsurancePolicyServiceImpl (class)	● Implements InsurancePolicyService. ● Do not modify, add or delete any method signature	Already Implemented.

3.6 PACKAGE: com.insurancepolicy.controller

Resources

Class/Interface	Description	Status
insurancePolicyController (Class)	● Controller class to expose all rest-endpoints for insurance policy related activities. ● May also contain local exception handler methods	Already Implemented

3.7 PACKAGE: com.insurancepolicy.dto

Resources

Class/Interface	Description	Status
InsurancePolicyDTO (Class)		Already Implemented

3.8 PACKAGE: com.insurancepolicy.entity

Resources

Class/Interface	Description	Status
InsurancePolicy (Class)		Already Implemented

3.9 PACKAGE: com.insurancepolicy.exception

Resources

Class/Interface	Description	Status
NotFoundException (Class)	● Custom Exception to be thrown when trying to fetch, update or delete the insurance policy info which does not exist.	Already implemented.

4. 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 Application menu (Three horizontal lines at left top) → Terminal →New Terminal.
3. This editor Auto Saves the code.
4. 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.
5. 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.
6. To execute and run test cases:

```
mvn clean install exec:java -Dexec.mainClass=" com.insurancepolicy.InsurancePolicyManagementApplication" -DskipTests=true
```

7. 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.