Capstone Project 1: Insurance Policy Management Automation Testing

Introduction

Insurance companies rely heavily on web applications for customer onboarding, policy management, claim processing, and premium payments. Manual testing of these features is time-consuming and error-prone. This project automates the functional, regression, and data-driven testing of an **Insurance Policy Management System (IPMS)** using Selenium WebDriver and TestNG.

Problem Statement

Insurance applications deal with multiple modules like **policy issuance, renewal, premium calculation, and claims management**. Errors in these workflows may cause financial losses or compliance issues. A robust automation suite is needed to ensure accuracy and reduce manual efforts.

Objectives

- Automate **end-to-end test scenarios** of insurance policy management.
- Implement **data-driven testing** for different policy types and premium calculations.
- Integrate **cross-browser and parallel testing** with TestNG.
- Generate **detailed Extent Reports** for management review.

Scope

- User Authentication (Login/Logout with role-based access).
- New Policy Creation (Life, Health, Vehicle Insurance).
- Premium Calculation Validation.
- Policy Renewal Process.
- Claims Submission & Status Tracking.
- Payment Gateway Validation.

Tech Stack

- **Language:** Java
- **Automation Tool:** Selenium WebDriver
- **Testing Framework:** TestNG
- **Reporting:** Extent Reports, TestNG HTML Reports
- **Build Tool:** Maven/Gradle
- **Data Source:** Excel/CSV (Apache POI) for data-driven testing
- **CI/CD:** Jenkins (optional)

Functionalities to Automate

1. **Login/Logout Tests** – Verify valid and invalid login attempts.
2. **Policy Creation** – Automate policy creation flow for multiple insurance products using DDT.
3. **Premium Calculation** – Validate premiums based on policy type, age, coverage amount, etc.
4. **Policy Renewal** – Ensure expired policies can be renewed correctly.
5. **Claims Processing** – Submit claims and track status updates.
6. **Payment Gateway** – Validate successful and failed transactions.

Test Scenarios

1. Verify user login with valid/invalid credentials.
2. Validate creation of new Health Insurance policy with correct premium.
3. Verify renewal process for expired Vehicle Insurance policy.
4. Test claim submission for Life Insurance and validate acknowledgment.
5. Test multiple policy creation using **DataProvider** in TestNG.
6. Validate parallel execution of policy and claim workflows in multiple browsers.

Expected Outcomes

- Automated regression test suite for policy management.
- Reusable test scripts for multiple policy types.
- Reduced manual effort and higher test coverage.
- Executable test reports with screenshots for failed cases.

Insurance Domain – Open Source / Free Apps

Unfortunately, there aren't many full-fledged **insurance apps** open-sourced, but you can still simulate flows:

□ Options:

1. **Insurance Project on GitHub (Spring Boot + React)**
   - Sample insurance apps exist on GitHub, e.g., **policy management portals**.
   - Example repos:
     - "Insurance Management System"
   - You can deploy locally (Tomcat/Maven/Node.js) and automate.
2. **Demo Blaze / Parabank with Insurance Twist**
   - **Demo Blaze (e-commerce app)** → can be reused to simulate "policy purchase" like insurance product buying.
   - **Parabank** (from Parasoft) → has *loan* + *account flows*, good for "insurance premium payments".
3. **Custom Insurance Portal (mocked)**
   - Use **Dummy JSON / Mock APIs** (like ReqRes, Mockoon)
   - Build a **basic insurance form site** in HTML/CSS/React (open-source boilerplate).
   - Automate policy creation, claims submission, payments.

---

3. Practical Setup for Your Capstone

- **Insurance Project:**

  Use **GitHub Insurance Management repos** or create a small mock **Insurance Portal** with policy form + claim form + payment page.

□ Recommendation for You:

Pick either a **GitHub insurance app** OR simulate insurance flows on **Demo Blaze / OpenCart** for your **Insurance Automation Project**.
