## Detailed Test Plan

**A. Functional Tests**

1. **Customer Facing Interfaces**:
   * **Web Bank, Mobile Banking, Web Store**:
     + **Usability Testing**: Ensure the interfaces are user-friendly and intuitive.
     + **Functionality Testing**: Verify all functionalities such as user login, policy purchase, claim submission, and payment processing.
     + **Test Cases**:
       - Assert logged in bank customer should have access to all digital channels domains
       - Assert registered insurance sales partners have access to Insurance sales SaaS A services
       - Assert non-bank customer have access to only the Webstore and Insurance core services
       - Assert Bank Employees have acess to both Insurance Sales microservices domain but also all Internal Sales Tools digital services
       - Confirm user ID access restriction works for all logged in users.
2. **Internal Tools**:
   * **Insurance Sales SaaS, Health Declaration**:
     + **Functionality Testing**: Ensure the tools support the sales process and health declaration accurately.
     + **Test Cases**:
       - Sales workflow end to end functional and non-functional tests
       - Health data input, validation, and submission.
       - Integration with backend systems for data storage and retrieval.
       - Assert internal Sales Tools are not accessibily to unauthorized users.
3. **Backend Systems**:
   * **API Gateway**:
     + **API Testing**: Validate all API endpoints for correct data handling, security, and performance.
     + **Test Cases**:
       - API response validation for various requests.
       - Security testing for API endpoints (e.g., authentication, access tokens).
       - Load testing to ensure performance under high traffic.
       - API integration and dataShake integration with other CRM providers (SalesForce, Zendesk, HubSpot)
   * **Insurance Domain Microservices**:
     + **Microservices Testing**: Ensure each microservice functions correctly and integrates well with others.
     + **Test Cases**:
       - Individual microservice functionality (dev unit tests )
       - Inter-microservice communication and data exchange. (end to end tests)
       - Error handling and recovery mechanisms.

**B. Integration Testing**

1. **Component Integration**:
   * **End-to-End Scenarios**: Verify the integration between customer interfaces, internal tools, and backend systems.
   * **Test Cases**:
     + Policy purchase from web bank to backend processing and confirmation.
     + Claim submission from mobile banking to backend validation and approval.
     + Data flow from health declaration to insurance domain microservices.
2. **External Systems**:
   * **Integration Testing**: Ensure seamless interaction with external systems and UI.
   * **Test Cases**:
     + Data exchange with external systems (e.g., third-party services).
     + Integration with external UI for consistent user experience.
     + Validation of data consistency and integrity across systems.

**C. Performance Testing**

1. **Load Testing**:
   * **Objective**: Ensure the system can handle peak usage without performance degradation.
   * **Test Cases**:
     + Simulate high user traffic on customer interfaces.
     + Stress test backend systems with large data processing.
     + Monitor system performance metrics (e.g., response time, throughput).
2. **Stress Testing**:
   * **Objective**: Determine the system’s breaking point and ensure it recovers gracefully.
   * **Test Cases**:
     + Overload the system with excessive requests.
     + Monitor system behavior under extreme conditions.
     + Validate system recovery and data integrity post-stress.

**D. Security Testing**

1. **Vulnerability Assessment**:
   * **Objective**: Identify and mitigate security vulnerabilities.
   * **Test Cases**:
     + SQL injection, cross-site scripting, and other common vulnerabilities.
     + Data encryption and secure data transmission.
     + Security of API endpoints and backend systems.
2. **Access Control**:
   * **Objective**: Ensure proper access control mechanisms are in place.
   * **Test Cases**:
     + Role-based access control for different user types.
     + Unauthorized access attempts and system response.
     + Audit logging and monitoring of access events.

**E. User Acceptance Testing (UAT)**

1. **End-User Scenarios**:
   * **Objective**: Validate the system with real-world scenarios to ensure it meets user expectations.
   * **Test Cases**:
     + Complete user journey from policy purchase to claim settlement.
     + User feedback on system usability and functionality.
     + Validation of business requirements and user needs.

**3. Reporting and Documentation**

* **Test Reports**:Tests reports in Wikipage in Jira
* **Bug Tracking**: Bug tracking with Xray in Jira
* **Test Summary**: Test Summary and documentation in pdf in Shared test folder in Onedrive

**4. Continuous Testing and Automation**

* **Automation**: RobotsFramework
* **Manual Tests**: QA Team
* **Unit Tests**: Developers