# Document 6

## Use case diagram

A Use Case Diagram is a graphical representation in Unified Modeling Language (UML) that shows the interactions between actors (users or external systems) and use cases (functionalities or services) of a system. It assists in capturing high-level functional requirements of a system. Use case diagrams are developed to:

* Define System Scope: It assists in understanding the system boundary and what functionalities it offers.
* Identify Key Interactions: It indicates various users or systems interacting with the system.
* Improve Communication: It gives a precise and straightforward method for describing system behavior to stakeholders such as developers, testers, and business analysts.
* Support Requirement Analysis: It assists in collecting and analyzing system requirements effectively.
* Assist in System Design: It can be used as a guide by developers to get functionalities right.

Use case diagram help in the requirement gathering phase to document functional requirements, it visualize system interactions for better understanding to align business analysts, developers, and testers on system functionality.

## Use case diagram and specification

1. Login with Biometric/2FA
2. Browse Insurance Plans
3. Purchase Insurance Policy
4. Manage Existing Policy
5. Claim Insurance
6. Renew Policy
7. Process Payment
8. Send Confirmation Notification

### 1. Use Case Name:

Insurance Policy Management

### 2. Use Case Description:

This use case outlines the process of managing insurance policies within the FNB insurance banking solution, including purchasing, renewing, and claiming policies with secure authentication and automated processing.

### 3. Actors:

**Primary Actors**: Customer, Insurance Agent

**Secondary Actors**: Bank System, Insurance Provider System, Payment Gateway

### 4. Basic Flow:

* Customer logs in via Biometric/2FA.
* System verifies authentication.
* Customer browses available insurance plans.
* Customer selects a plan and enters details.
* System validates details and calculates premium.
* Customer makes payment via the Payment Gateway.
* System confirms policy purchase and sends confirmation.
* Customer can later manage, renew, or claim the policy.

### 5. Alternate Flow:

* If the customer already has an active policy, they can directly manage or renew it.
* If biometric authentication fails, the system prompts for manual login.

### 6. Exceptional Flows:

* If payment fails, the system notifies the customer and allows retry.
* If invalid details are entered, the system prompts the customer to correct them.
* If insurance provider services are down, the request is logged for later processing.

### 7. Pre-Conditions:

* Customer must have an active FNB banking account.
* Customer must be registered for online banking.

### 8. Post-Conditions:

* Policy is successfully issued, renewed, or managed.
* Customer receives confirmation and policy documents.
* Payment details are securely recorded.

### 9. Assumptions:

* Customers are familiar with online banking and insurance services.
* Insurance provider APIs are available and functional.

### 10. Constraints:

* System must comply with financial and insurance regulations.
* Real-time updates depend on third-party integrations.

### 11. Dependencies:

* Integration with the BKI insurance system.
* Secure payment processing through a payment gateway.
* Secure authentication mechanisms (biometric, 2FA).

### 12. Inputs and Outputs:

* **Inputs**: Customer details, policy selection, payment details.
* **Outputs**: Policy confirmation, payment receipt, insurance documents.

### 13. Business Rules:

* Customers can only purchase policies if they meet eligibility criteria.
* Policy claims must go through verification before approval.
* Payments must be successfully processed before policy issuance.

### 14. Miscellaneous Information:

* The system must support multiple types of insurance policies.
* Notifications should be sent via email and SMS.
* Customer support should be available for claim-related queries.



**Activity diagram**

An Activity Diagram is a UML (Unified Modeling Language) behavioral diagram that visually represents the flow of activities in a system or process. It models the sequence of actions, decision points, parallel processes, and interactions between different system components. Activity diagrams are created to model Business and System Workflows which helps in visualizing the step-by-step execution of a process. it is user to analyze Use Cases which represents how different actions occur within a use case. Activity Diagram improve Communication which gives clear representation of business logic to stakeholders, developers, and testers. Activity Diagram identify Decision Points and Exceptions which Helps in understanding conditional flows, loops, and error handling.

it also support System Design and Development where it is helpful to developers in understanding process logic before coding.



# Document 7

**Screens and pages**

## Login/Sign Up Page mock up screen



## Register page mock up screen



## Home page mock up screen



## Contact Dashboard page screen mockup



## Policy Dashboard page mock up screen



## Cover detail page mock up screen



## Payment Page mock up screen



## Policy history mock up screen page



## Log out page mock up screen



# Document 8

## Experience with Microsoft Visio:

Microsoft Visio was a crucial tool used for the development of UML diagrams and visually modeling the workflows of the system in the FNB insurance banking solution project. I employed Visio to design use case diagrams, activity diagrams, and flowcharts, which aided in demonstrating system functionality, user interactions, and business processes clearly. The drag-and-drop functionality, comprehensive library of UML elements, and formatting capabilities of the tool made it easy to construct well-structured diagrams. Visio's alignment and layering capabilities made sure that diagrams were properly organized and professional in nature. On the whole, Visio was extremely useful for system modeling and requirement documentation, simplifying communication of technical specifics to stakeholders and the development team.

## Experience Using Axure RP:

Axure RP was used to develop interactive wireframes and prototypes of the FNB insurance banking portal. Using the tool's capability to create clickable prototypes enabled me to mimic actual user behavior, giving stakeholders a test-drive experience of the system UI/UX prior to development. I used Axure dynamic panels, conditional logic, and interactive elements to demonstrate various user flows, including policy purchases, renewals, and claims. This interactive method facilitated easier early feedback, usability problem identification, and iterative refinement of the design. Axure's robust prototyping capabilities bridged the gap between business needs and user experience, resulting in a well-designed and user-friendly digital platform.

# Document 9

**BA experience**

1. Requirement gathering:
As the Business Analyst for the FNB Insurance Banking Solution, I had a tight timeline to gather requirements. The challenge? The client would be unavailable for a while, so I had to quickly identify alternative points of contact to ensure no delays. To prioritize efficiently, I applied the MoSCoW technique, categorizing requirements into Must-Have, Should-Have, Could-Have, and Won’t-Have. This helped the team stay focused on critical features like real-time updates, high-security authentication, and self-service options. During validation, I used FURPS to ensure all requirements met Functionality, Usability, Reliability, Performance, and Supportability standards. While reviewing, I noticed duplicate requirements, which we immediately removed to maintain clarity. To refine the details, we introduced prototyping, allowing stakeholders to visualize features like biometric authentication, 2FA, and real-time notifications. This approach helped us gather more precise requirements and ensure alignment with business goals. By combining structured prioritization, validation, and prototyping, we streamlined the requirement-gathering process—laying the foundation for a secure and seamless insurance banking experience.

2. Requirement Analysis:
As the project Business Analyst, Requirement Analysis was my next step—ensuring all the requirements that were gathered were properly formatted and well-understood by the team. In order to depict the requirements graphically, I have used UML diagrams and deconstructed the main system features. I also applied Activity Diagrams to describe the process flows so that each part of the user flow was correctly captured. Once the diagrams were completed, I presented them to the development and test teams. There were some reservations and proposed changes from some members of the teams based on technical viability and improved system design, as anticipated. As a BA, I was careful to incorporate their feedback with the required adjustments while maintaining congruence with business goals. Having completed the visual models, I proceeded to document the requirements. I drafted the Business Requirements Specification (BRS) to record high-level business requirements and the Software Requirements Specification (SRS) to detail the system's functional and non-functional features. Through a combination of visual modeling, collaboration with the team, and extensive documentation, we had a thoroughly analyzed set of requirements—a strong foundation for development.

3. Design:
Once the requirement analysis was done, my subsequent task was to make sure the design phase synchronized with business objectives. Based on the Use Case Diagrams, I started writing test cases, making sure each feature was well-tested. To keep the client in the loop, I shared the design and solution documents with them, taking their feedback and making adjustments accordingly. In the meantime, I carefully crafted positive and negative test cases so that we didn't miss any edge cases. One missing test case would lead to significant problems down the line, so I made sure we left nothing behind. After the test cases were finalized, I proceeded to develop test data to mimic real-world conditions when testing. To ensure traceability, I went ahead and refreshed the Requirements Traceability Matrix (RTM)—ensuring each requirement had a traceable test case and nothing was left untested. With systematic test case design, client engagement, and strict traceability, we established a solid foundation for an uneventful development and testing process.

4. Development:
As we entered the Development Phase, my work as a Business Analyst transitioned to facilitating easy team collaboration. To get everybody on the same page, I conducted JAD (Joint Application Development) sessions, where the business and technical teams came together to discuss things. But some team members didn't like the method or even want to cooperate. I solved the problem by sitting down one-on-one and telling them why their ideas mattered and how if they resisted, it would hurt the project. By clearing their doubts and establishing a good team atmosphere, I got everybody working on the same wavelength. Throughout development, the technology team frequently had questions about the requirements, which I explained immediately. I also referred to the UML and Activity Diagrams to assist developers in coding each unit properly, ensuring it was aligned with the business requirements. Having regular meetings with the technical team and the client was another challenge. Some of the team members were not able to attend every time, so I recorded the sessions and distributed them to the ones who missed out. To also ensure that things remained clear, I held follow-up one-on-one meetings with them. By keeping communication clear, solving disagreements, and ensuring ongoing cooperation, I ensured that development remained on track, creating a foundation for successful implementation.

5. Testing:
With the needs clearly established, my second assignment as a Business Analyst was to assist the testing process to make the transition from development to deployment as seamless as possible. I started by creating test cases from use cases, making sure that every business scenario was well covered. The test cases were used to verify if the system behaved as it should. To catch issues early, I also performed high-level testing, verifying core functionalities prior to formal testing. In order to run test scenarios successfully, I asked for test data from the client to ensure we employed real-world inputs for proper validation. As we tested, I regularly updated the Requirements Traceability Matrix (RTM) to monitor coverage and ensure that each requirement was being tested. After testing was done, I coordinated the sign-off process with the client, showing test output and that all key issues were resolved. I then readied the client for User Acceptance Testing (UAT), walking them through the system and making them feel confident to validate it against business requirements. Through a disciplined testing methodology, we had a high-quality, well-tested system ready for deployment.

6. Deployment:
As the Business Analyst, the last stage of the project—Deployment—was important to facilitate a seamless transition. My primary task was to send the Requirements Traceability Matrix (RTM) to the client, ensuring it was attached to the project closure document correctly for final verification. Secondly, I collaborated with the technical team to prepare and distribute end-user guides, making sure that users had concise, step-by-step instructions on how to use the system. To facilitate adoption, I scheduled and conducted training sessions, ensuring the end users recognized the functionalities of the new system. It was vital that all major stakeholders were present, so I pursued follow-up actively, sent reminders, and managed schedules to get their full presence. By synchronizing documentation, training, and communication, I made the deployment process smooth, properly documented, and smoothly transferred to the users—completing the project successfully.