**Document 6:**





|  |  |
| --- | --- |
| **Use Case Name** | Database Configuration |
| **Use Case Description** | Configuration the database  |
| **Actors** | Source system, Configuration system |
| **Pre-Conditions** | Configure the database with all properties from the source system |
| **Post-Conditions** | database properties of the Integrated system should be configured with all attributes from source system |
| **Assumptions** | Availability of resources, data types, system performance |
| **Constraints** | Budgetary Constraints, Balancing quality |
| **Dependencies** | Process, version |
| **Inputs and Outputs** | Resources, End users, Hardware resources |
| **Business Rules** | Avoiding data loss or corruptionMaintaining data structureEnsuring service continuitySecuring data before, during, and after migrationEnsuring the budget covers all migration needs |
| **Miscellaneous Information** | system upgrades, cloud migration, and merging data from multiple systems |

|  |  |
| --- | --- |
| **Use Case Name** | Database Integrity |
| **Use Case Description** | Integrating the database  |
| **Actors** | Source system, Target System |
| **Pre-Conditions** | Integrate the database with all properties from the different source systems |
| **Post-Conditions** | database properties of the Integrated system should be compatible with all attributes from all source system |
| **Assumptions** | Availability of resources, data types, system performance |
| **Constraints** | Budgetary Constraints, Balancing quality |
| **Dependencies** | Process, version |
| **Inputs and Outputs** | Resources, End users, Hardware resources |
| **Business Rules** | Avoiding data loss or corruptionMaintaining data structureEnsuring service continuitySecuring data before, during, and after migrationEnsuring the budget covers all migration needs |
| **Miscellaneous Information** | system upgrades, cloud migration, and merging data from multiple systems |

|  |  |
| --- | --- |
| **Use Case Name** | Database Mapping |
| **Use Case Description** | Mapping the database  |
| **Actors** | Source system, Target System |
| **Pre-Conditions** | Mapping the database with all properties from the different source systems |
| **Post-Conditions** | database properties of the Integrated system should be compatible with all attributes from all source system |
| **Assumptions** | Availability of resources, data types, system performance |
| **Constraints** | Budgetary Constraints, Balancing quality |
| **Dependencies** | Process, version |
| **Inputs and Outputs** | Resources, End users, Hardware resources |
| **Business Rules** | Avoiding data loss or corruptionMaintaining data structureEnsuring service continuitySecuring data before, during, and after migrationEnsuring the budget covers all migration needs |
| **Miscellaneous Information** | system upgrades, cloud migration, and merging data from multiple systems |

**Document 7 Screens and pages**

****

****

**Document 8- Tools-Visio and Axure**

Visio is a good tool which I used it for creating diagrams like Swim lane etc. I also added the diagrams created in Visio in PowerPoint presentation. I used Visio tool for my project to create UML diagrams, Activity diagrams, Flowcharts, Gantt Charts,…

Microsoft Visio is a versatile tool on which I had a very good hands on experience in creating a wide range of diagrams and visual representations, such as:

1. **Flowcharts and Process Diagrams**: Visualize workflows and processes.
2. **Organizational Charts**: Display company hierarchy.
3. **Network Diagrams**: Map IT networks and infrastructure.
4. **Floor Plans**: Design layouts for offices and facilities.
5. **Gantt Charts**: Plan and track project timelines.
6. **UML Diagrams**: Model software systems.

Axure RP is the UX tool that gives UX professionals the power to build realistic, functional prototypes. In our project, I used hotspot-based Axure( prototyping tools) with which we linked screens, and add a few screen transitions.

We used Axure because it can easily handle the subtle details and micro-interactions that bring a prototype to life. Axure is a fully integrated application that eliminates the need to use different design tools

**Document 9- BA experience**

My experience as BA in following phases:

**1. Requirement gathering:**

● To gather requirements, we used MOSCOW technique. the MoSCoW technique is a popular method for gathering requirements and prioritizing them. The MoSCoW technique is a framework for categorizing requirements into four categories: Must-Have, Should-Have, Could-Have, and Won't-Have. The MoSCoW technique helps teams focus on the most important requirements first, ensuring that the project's essential goals are met.

 ● Client is not available for some period during this phase. So as a BA i need to source out point of contacts from his side and get the information ASAP. As a business analyst, the initial interactions with a client are crucial for understanding their needs and establishing a successful working relationship.

● I validate the requirements using FURPS technique. The FURPS technique is a model for classifying software quality attributes and validating requirements:

**Functionality:** The capabilities and features of the application, such as the size and generality of the feature set

**Usability:** How easy the program is to use, and whether the documentation is accurate and complete:

**Reliability:** How much system downtime is acceptable, and how can the system recover from failures

**Performance:** The application's response time and throughput, and the limits on the system resources it can use

**Security:** The safety and exploitability of the application

FURPS technique is used to categorize requirements based on their importance, satisfaction, or quality. It was also used to verify the completeness of requirements

* There are many requirements which are duplicated or repeated. We need to remove them immediately: Requirements prioritization involves evaluating and ranking features based on their business value, technical feasibility, and customer needs. Using this technique in our project we made informed decisions, avoiding scope creep and ensuring that the most valuable features are delivered within the project constraints and also maintained redundancy.

● Prototyping is used to give more specific requirements: Prototyping is the technique for obtaining early feedback on requirements by building a working model of the expected product.

We used this technique to try out a product’s mock-up versions before the final product release.

They are excellent for feedback generation.

Based on the prototype performance, we can make necessary adjustments.

**2. Requirement Analysis:**

● we have drawn UML diagrams to visually describe the requirements

● we have also represented as Activity diagrams to describe the process flow

● the represented diagrams are Communicated to team.

 Some team members might not agree with them and we might need to make changes based on their needs. As a BA we need to consider the points and make modifications

● Prepare BRS and SRS : BRS is a standard document that stands for Business Requirements Specification.

The BRS Document is prepared by a team of Business Analysts (BA) who interacts with the clients, document that describes the core product goals or needs the client is willing to achieve with the software or product, which is usually created at the very beginning of the product’s life cycle. The BRS includes all the client’s requirements.

1. Identify stakeholders and gather requirements

2. Define the purpose and scope

3.Document user requirements

4. Create a realistic timeline for the main project milestones

5. Include a brief cost-benefit analysis

The SRS document is created by a team of System Analysts (SA), a technical expert. An SRS or software requirement specification is a document prepared by a team of system analysts to describe the software being developed, the business purpose and functionality of a particular product, and how it performs its core functions. This document should include the following information:

* Purpose of the product.
* Target audience.
* Intended use.
* Product scope.
* Key terms and acronyms.

**3. Design:**

● From the use case diagrams, we prepare test cases

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Test Case** | TC\_001 | Test Case Name | Source system |   |   |
| **Project ID** | PRJ\_001 | Project Name | Datahub |   |   |
| **PM ID** | PM\_001 | Tester ID | T001 |   |   |
| **Test Strategy ID** | TS\_001 | Tester Name | Prabha |   |   |
| **Test Plan ID** | TP\_001 | Date of Test | dd-MM-yyyy |   |   |
| **Test Schedule** | TSCH\_001 |   |   |   |   |
| **Scenario** | Source System functionality |
| **Example** | Checking the functionality of source |
| **Action** | **Inputs** | **Expected Output** | **Actual Output** | **Test Browser** | **Test Result** |
| Check the functionality of the source database | source system | system should be performing as expected | system is configured correctly | Chrome | Pass |

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Test Case** | TC\_002 | Test Case Name | System Configuration |   |   |
| **Project ID** | PRJ\_001 | Project Name | Datahub |   |   |
| **PM ID** | PM\_001 | Tester ID | T002 |   |   |
| **Test Strategy ID** | TS\_001 | Tester Name | Prathi |   |   |
| **Test Plan ID** | TP\_001 | Date of Test | dd-MM-yyyy |   |   |
| **Test Schedule** | TSCH\_001 |   |   |   |   |
| **Scenario** | Source system configuration |
| **Example** | Configure the source system |
| **Action** | **Inputs** | **Expected Output** | **Actual Output** | **Test Browser** | **Test Result** |
| Check the configuration of the source database | source system | system should be configured as expected | system is configured correctly | Chrome | Pass |

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Test Case** | TC\_003 | Test Case Name | System Performance |   |   |
| **Project ID** | PRJ\_001 | Project Name | Datahub |   |   |
| **PM ID** | PM\_001 | Tester ID | T003 |   |   |
| **Test Strategy ID** | TS\_001 | Tester Name | Prabhu |   |   |
| **Test Plan ID** | TP\_001 | Date of Test | dd-MM-yyyy |   |   |
| **Test Schedule** | TSCH\_001 |   |   |   |   |
| **Scenario** | Source system performance |
| **Example** | Check the performance of the source system |
| **Action** | **Inputs** | **Expected Output** | **Actual Output** | **Test Browser** | **Test Result** |
| Check the performance of the source database | source system | system should be performing/ responding as expected | system is executed correctly | Chrome | Pass |

● Communicate with client on design and solution documents

We followed below key Points for Presenting Design

1. Understand client's business model: What is their plan for the next five years? ...
2. Know our audience: Everyone in the room has a voice that should be valued and heard. ...
3. Don't talk over their heads: Speak in standard, non-technical language. ...
4. Use demos and visual aids

This document outlines an overview of the build, its scope, and all related configurations. The design document also include important sections related to security, third-party integrations, and appendices.

● Write negative test cases as well along with positive test cases.

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Test Case** | TC\_004 | Test Case Name | Incorrect DB configuration |   |   |
| **Project ID** | PRJ\_001 | Project Name | Datahub |   |   |
| **PM ID** | PM\_001 | Tester ID | T004 |   |   |
| **Test Strategy ID** | TS\_001 | Tester Name | ABC |   |   |
| **Test Plan ID** | TP\_001 | Date of Test | dd-MM-yyyy |   |   |
| **Test Schedule** | TSCH\_001 |   |   |   |   |
| **Scenario** | Incorrect DB configuration |
| **Example** | Check the performance of the source system |
| **Action** | **Inputs** | **Expected Output** | **Actual Output** | **Test Browser** | **Test Result** |
| While checking the DB configuration, properties are incorrect | source system | system should be configured as expected | system is incorrectly configured with some wrong data types | Chrome | Fail |

● Do not miss a single test case. It might have huge impact on project development in later stages

● Prepare test data for testing

● Update RTM. This is just as we need to make sure that all the requirements are met

1. **Development:**

● Organized JAD sessions: JAD promotes a spirit of partnership and collaboration, so requirements are documented faster and more accurately than with traditional requirements.  The actual JAD session consists of facilitating a workshop where stakeholders can share their views, needs, and expectations while collaborating

● Clarifying queries of tech team during coding: some examples of Clarifying Questions When did the issue start? What were you doing when the issue occurred? Have there been any recent changes to?

● There might be some team members who doesn't agree with the concept or who doesn’t cooperate during JAD sessions. As a BA i handle the situation gently and had one on one discussion with them. Explained how their actions are going to affect the project. Setup healthy environment within the team.

● Referred diagrams to code the Unit

● Conduct regular meetings with technical team and client which is challenging. Some team members might not be available for the meeting. Recording the session and providing that to missed one and having one to one discussion later with that missed person is all i need to do

**5. Testing:**

● Prepare test cases from use cases: For each use case, generate a full set of use-case scenarios. For each scenario, identify at least one test case and the conditions. that will make it "execute." For each test case, identify the data values with which to test.

● Perform high level testing: These test cases define the functionality of a software/product in a broader way without going into deep functionality. Like if we have to write high level test cases for login functionality, we'll test that 'User should be able to login success full with valid credentials'.

● Test data is requested by BA from client: Test data is an integral part of the testing process for a tester. It provides information to the tester to facilitate finding defects and corrective actions. This information is gathered from client.

● Updated RTM

● Take signoff from client: Use a sign-off form or a digital platform that allows the client to sign electronically, and make sure it includes the details of the job, the date and time, and the client's name and contact information. Thank the client for their business, and ask them to sign the form or the platform.

● Prepare client for UAT

**User Acceptance Testing (UAT) Checklist**

* Define the Scope of UAT.
* Prepare the Test Environment.
* Create a UAT Plan.
* Select Testers, Their Roles & Responsibilities.
* Create UAT Test Cases Based on End-To-End Scenarios and Business Cases.
* Prioritize Test Cases.
* Choose a Test Management Tool.
* Review and Approve Test Cases.

**6. Deployment:**

● Forwarded RTM to client which should be attached to project closure document: The Requirements Traceability Matrix (RTM) is a tool or a document that helps project managers establish and track the project progress. Project closure is the last phase of a project. It’s when the project manager verifies that the client, stakeholder or customer has accepted the [project deliverables.](https://www.projectmanager.com/blog/project-deliverable) If the project or product is ongoing after the project, then maintenance must be set up

● Coordinates to complete and share end user manuals: The purpose of this manual is to help you get the software up and running, and to teach you how to configure and operate it.

● Plans and organizes training sessions: To create an organizational training plan, start by assessing the current skills and knowledge within your organization. Identify gaps and training needs, set clear and measurable goals, and design a training program that addresses these needs while aligning with company objectives

● Make sure all the candidates attend the meeting