Mobile Development Project for the Guadalajara Food Bank (BAMX)

Diego Iván Morales Gallardo

October 24, 2024

Contents

1 Problem Definition 5

2 General Objective 5

3 Specific Objective: Inventory Management and Logistics 5

4 Methodology for Acquiring User Information 6

5 Team Members and Their Roles 6

6 Empathy Map 7

7 Customer Journey Map 7

8 Persona: María Hernández 8

9 User Story 9

9.1 User Authentication 9

9.2 User Registration 9

9.3 Differentiated Interface (Administrator) 10

9.4 Differentiated Interface (Standard User) 10

9.5 Product Registration 10

9.6 Search for Existing Products 10

9.7 Product Entry Record 10

9.8 Product Exit Record 11

9.9 Real-Time Inventory Viewing 11

9.10 Product Management (Administrator) 11

9.11 Operation History 11

9.12 User Modification 12

9.13 Report Generation 12

9.14 Low Inventory Notifications 12

10 Functional Requirements 12

11 Non-Functional Requirements 13

12 Activity Diagram 14

13 Project Management Plan 15

13.1 Project Scope 15

13.1.1 Problem Definition and Solution 15

13.1.2 Measurable Benefits of the Solution 15

13.1.3 Beneficiaries of the Solution 15

13.1.4 Deliverables Included in Scope 15

13.1.5 Out of Scope Deliverables 16
13.2 Project Timeline 16
13.2.1 Week 1: Project Initiation and Requirements Gathering 16
13.2.2 Week 2: System Design and Planning 16
13.2.3 Week 3: Design Review and Approval 17
13.2.4 Weeks 4-7: Development 17
13.2.5 Week 8: User Acceptance Testing (UAT) 17
13.2.6 Week 9: System Integration and Testing 18
13.2.7 Week 10: Finalization and Deployment 18
13.3 Project Effort Estimation 18
13.3.1 Function Point Breakdown 19
13.3.2 Effort per Function Point 19
13.3.3 Effort Distribution by Team Member 19
13.3.4 Total Project Effort 19
13.4 Project Cost 19
13.4.1 Cost Breakdown 20
13.4.2 Contingency Buffer 20
13.4.3 Project Cost Summary 20
13.5 Risk Matrix 20
13.6 Contingency Matrix 23

14 Database Description 25
14.1 Users 25
14.2 Products 25
14.3 Donors 25
14.4 Inventory 25
14.5 Operation History 25
14.6 Notifications 26
14.7 Categories 26
14.7.1 Reference of the Categories that BAMX Manages 26
14.8 Reports 26
14.8.1 Relationship between tables 26
14.9 Database Diagram 27

15 Test Strategy 27
15.1 Priorities In Our Testing 27
15.1.1 Fluency 27
15.1.2 Tediousness 27
15.1.3 Business Scenarios 28
15.1.4 Stability 28
15.1.5 Confidentiality 28
15.1.6 Functionality 28

15.1.7 Data 28
15.2 Testing Types In Our Testing 28
15.2.1 Regression Testing 28
15.2.2 Dogfooding 28
15.2.3 Violate Data Format Rules 28
15.2.4 Usability Testing 28
15.2.5 Security Testing 29
15.2.6 Compatibility Testing 29
15.2.7 Configuration Testing 29
15.3 Test Schedule 29
15.3.1 Sprint Structure and Activities 29
15.3.2 Sprint Breakdown (10 working days) 29
15.3.3 Focus Areas for Each Sprint 30
15.3.4 Post-Sprint Activities 30
15.4 Test Planning 31
15.4.1 Post-Sprint Planning Alignment 31
15.4.2 Documentation of Testing Scope 31
15.4.3 Task Organization and Prioritization 31
15.4.4 Defining Communication Protocols 31
15.4.5 Test Case Management 31
15.4.6 Automation Strategy 32
15.4.7 Documentation of Agreements and Tasks 32
15.5 Agile Testing 32
15.5.1 New Feature Testing 32
15.5.2 Regression Testing 33
15.5.3 Confirmation Testing 33
15.6 Roles and Responsibilities 34
15.7 Testing Tools 34
15.8 Test Environments 34
15.8.1 DEV Environment 34
15.8.2 SIT Environment (System Integration Testing) 35
15.8.3 STG Environment (Staging) 35
15.8.4 Justification 35
15.9 Test Automation 35
15.9.1 Unit/Component Level 35
15.9.2 Integration Level 36
15.9.3 UI Level 36
15.10 Life cycle of a Ticket 36
15.10.1 Implementation in Our Project 36
15.10.2 Definition of Done Guideline 37
15.11 Test Metrics 37
15.11.1 Test Coverage 37

15.11.2 Ratio of Failures to Defects 38
15.11.3 Time Taken to Execute Tests 38
15.11.4 Trend Metrics (Over Multiple Sprints) 38
15.12 Test Monitoring and Reporting 38
15.12.1 Sprint Review Meetings 38
15.12.2 Metrics from Project/Test Management Tools 38
15.12.3 Metrics from Continuous Integration (CI) Tools 39
15.12.4 Manual Data Preparation 39
15.12.5 Reporting 39

1 Problem Definition

The Guadalajara Food Bank (BAMX) faces serious challenges in optimizing its operations due to its reliance on manual processes, particularly in managing its inventory. This traditional approach not only consumes considerable time, but also increases the risk of human errors that can affect the accuracy and efficiency of food distribution. These issues not only hinder operational flow, but also limit BAMX's ability to maximize the impact of its resources in fighting food insecurity in the most needy communities.

In addition, the lack of an automated and centralized system for inventory management makes it difficult to access real-time data, which is crucial for making quick and effective decisions in an environment where demand is constant and needs are urgent. This situation highlights the need for a technological solution that can address these operational deficiencies.

The problem, therefore, lies in inefficiency and potential waste of resources due to the lack of advanced technological tools that can support BAMX operations more effectively. Without an adequate system, BAMX runs the risk of not being able to optimally serve vulnerable communities, thereby reducing its ability to combat food insecurity in Guadalajara and its surrounding areas. Solving this problem is essential to ensure that BAMX resources are used efficiently and effectively, maximizing their impact on the community.

2 General Objective

The main objective of this project is to develop a comprehensive mobile application designed to optimize the operations of the Guadalajara Food Bank (BAMX). This application seeks not only to facilitate the communication and management of the various support programs that BAMX implements, but also to significantly contribute to the reduction of food insecurity in the vulnerable communities that depend on its services. In the long term, the goal is that the efficiency achieved through this application will ultimately eliminate the very need for BAMX's existence, having fulfilled its mission of eradicating hunger in the communities served.

3 Specific Objective: Inventory Management and Logistics

One of the key specific objectives is to develop a robust and efficient system for inventory management. This system must allow for detailed recording, accurate management, and continuous monitoring. The system will be designed to handle large volumes of data and provide real-time information on the status of inventory, allowing BAMX to make informed and strategic decisions to maximize the impact of its resources. This objective also includes the implementation of automatic alerts for products that are about to run out, thus ensuring that food is distributed in a timely manner.

4 Methodology for Acquiring User Information

The methodology for acquiring user information was based on a virtual meeting session conducted via Zoom with key stakeholders of Banco de Alimentos Guadalajara (BAMX). During this session, the project team had the opportunity to gain an initial and fundamental understanding of the issues facing the organization, focusing particularly on inventory management and food distribution logistics. The session was crucial in directly capturing the experiences and needs of BAMX staff, allowing the team to begin defining the scope and specific objectives of the project.

In this single session, time was spent listening in detail to BAMX's current workflow, from food rescue to distribution in communities. Specific limitations and challenges associated with the reliance on manual processes and lack of advanced technological tools were discussed. The interaction allowed the project team to quickly map critical processes and begin to identify areas of improvement that would be key in the development of the mobile app.

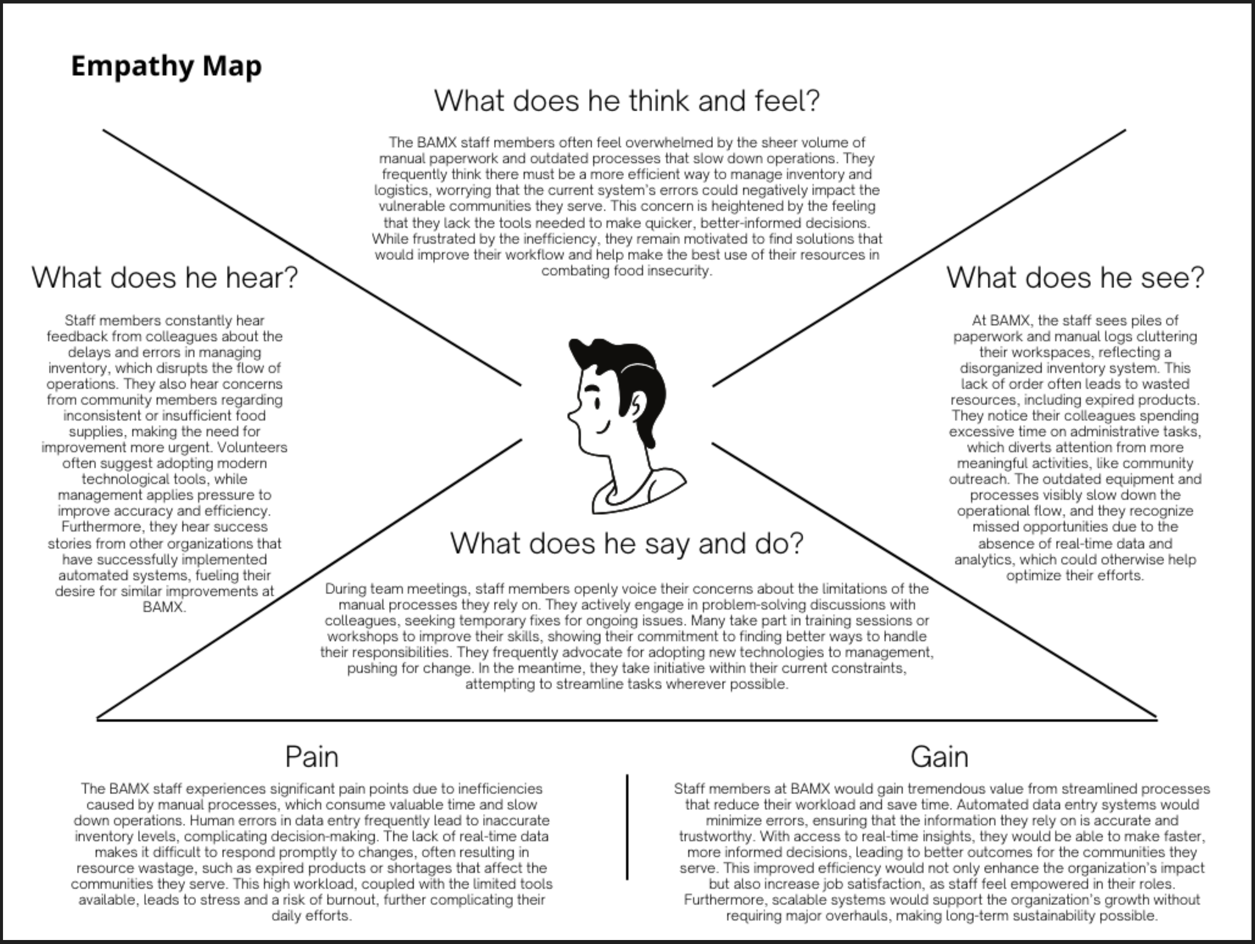
To ensure that the information collected was accurate and comprehensive, Zoom's recording feature was used, allowing the team to subsequently review the details discussed during the session. This recording became a valuable resource for the development of the application's functional and non-functional requirements, ensuring that the proposed solutions were aligned with BAMX's real needs and effectively responded to the operational challenges identified in the meeting.

5 Team Members and Their Roles

- **Luis Daniel García Espinosa:** Responsible for the implementation of the user interface (UI) and user experience (UX). Luis Daniel will be responsible for ensuring that the application is intuitive, easy to use, and meets end-user expectations in terms of design and functionality.
- **Victor Jaziel Coronado Flores:** Responsible for UI/UX design using Figma. Jaziel will work closely with Luis Daniel to develop visual prototypes and ensure that the application design is consistent, attractive, and functional.
- **Milan Alejandro De Alba Gómez:** Responsible for endpoint and API development. Milan will design and implement the programming interfaces necessary for the application to interact with backend systems, ensuring that data is managed and transmitted securely and efficiently.
- **Jorge Antonio Arizpe Cantú:** Responsible for database management. Jorge will be responsible for designing and implementing the database architecture, ensuring it is scalable, secure, and capable of handling large volumes of data efficiently.
- **Diego Iván Morales Gallardo:** Responsible for documentation and testing. Diego will manage the project, ensuring that deadlines and objectives are met, as well as overseeing the quality of the product by performing extensive testing and generating detailed documentation.

6 Empathy Map

The empathy map provides an in-depth view of the BAMX staff's emotional and cognitive experiences as they manage the food bank's daily operations. It captures key insights into what the staff thinks, feels, hears, and sees regarding their current inventory and logistics processes. The map also highlights how these employees express their frustrations, particularly with outdated manual processes, and showcases potential opportunities for improvement through the introduction of modern technological solutions. Lastly, it emphasizes both the pain points they endure and the gains that could be achieved by streamlining their operations with automated systems.



The BAMX staff members often feel overwhelmed by the sheer volume of manual paperwork and outdated processes that slow down operations. They frequently think there must be a more efficient way to manage inventory and logistics, worrying that the current system's errors could negatively impact the vulnerable communities they serve. This concern is heightened by the feeling that they lack the tools needed to make quicker, better-informed decisions. While frustrated by the inefficiency, they remain motivated to find solutions that would improve their workflow and help make the best use of their resources in combating food insecurity.

Staff members constantly hear feedback from colleagues about the delays and errors in managing inventory, which disrupts the flow of operations. They also hear concerns from community members regarding inconsistent or insufficient food supplies, making the need for improvement more urgent. Volunteers often suggest adopting modern technological tools, while management applies pressure to improve accuracy and efficiency. Furthermore, they hear success stories from other organizations that have successfully implemented automated systems, fueling their desire for similar improvements at BAMX.

At BAMX, the staff sees piles of paperwork and manual logs cluttering their workspaces, reflecting a disorganized inventory system. This lack of order often leads to wasted resources, including expired products. They notice their colleagues spending excessive time on administrative tasks, which diverts attention from more meaningful activities, like community outreach. The outdated equipment and processes visibly slow down the operational flow, and they recognize missed opportunities due to the absence of real-time data and analytics, which could otherwise help optimize their efforts.

During team meetings, staff members openly voice their concerns about the limitations of the manual processes they rely on. They actively engage in problem-solving discussions with colleagues, seeking temporary fixes for ongoing issues. Many take part in training sessions or workshops to improve their skills, showing their commitment to finding better ways to handle their responsibilities. They frequently advocate for adopting new technologies to management, pushing for change. In the meantime, they take initiative within their current constraints, attempting to streamline tasks wherever possible.

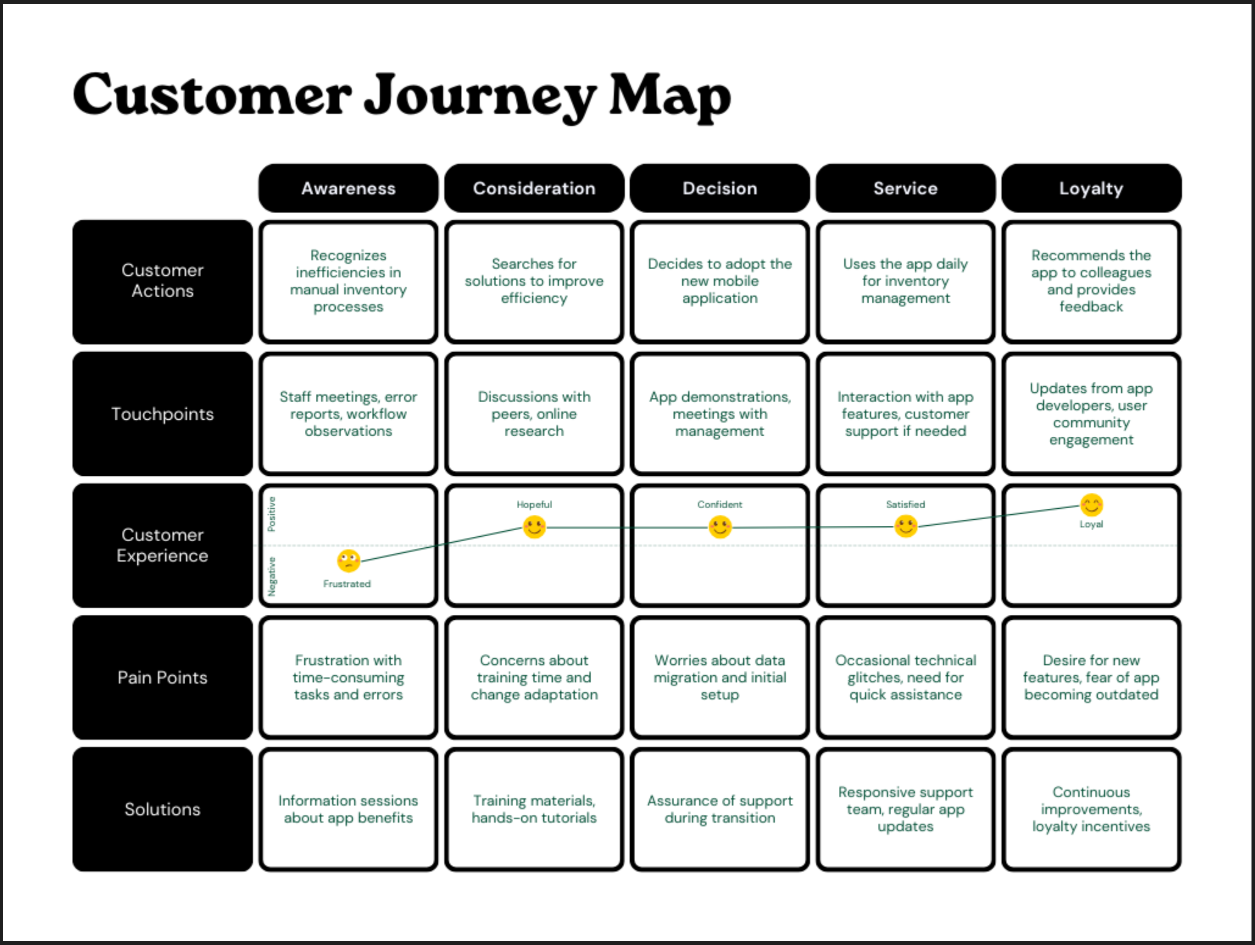
The BAMX staff experiences significant pain points due to inefficiencies caused by manual processes, which consume valuable time and slow down operations. Human errors in data entry frequently lead to inaccurate inventory levels, complicating decision-making. The lack of real-time data makes it difficult to respond promptly to changes, often resulting in resource wastage, such as expired products or shortages that affect the communities they serve. This high workload, coupled with the limited tools available, leads to stress and a risk of burnout, further complicating their daily efforts.

Staff members at BAMX would gain tremendous value from streamlined processes that reduce their workload and save time. Automated data entry systems would minimize errors, ensuring that the information they rely on is accurate and trustworthy. With access to real-time insights, they would be able to make faster, more informed decisions, leading to better outcomes for the communities they serve. This improved efficiency would not only enhance the organization's impact but also increase job satisfaction, as staff feel empowered in their roles. Furthermore, scalable systems would support the organization's growth without requiring major overhauls, making long-term sustainability possible.

Figure 1: Empathy Map of BAMX staff, detailing their thoughts, emotions, and operational challenges.

7 Customer Journey Map

The customer journey map outlines the process that BAMX staff undergoes from recognizing inefficiencies in their current manual processes to becoming long-term users of a new mobile application for inventory management. This journey spans several phases, including awareness, consideration, decision, service, and loyalty. Each phase identifies specific customer actions, touchpoints, emotions, and pain points, with corresponding solutions aimed at improving the staff's experience at each stage. By understanding this journey, the project team can better design the mobile app to meet BAMX's operational needs effectively.



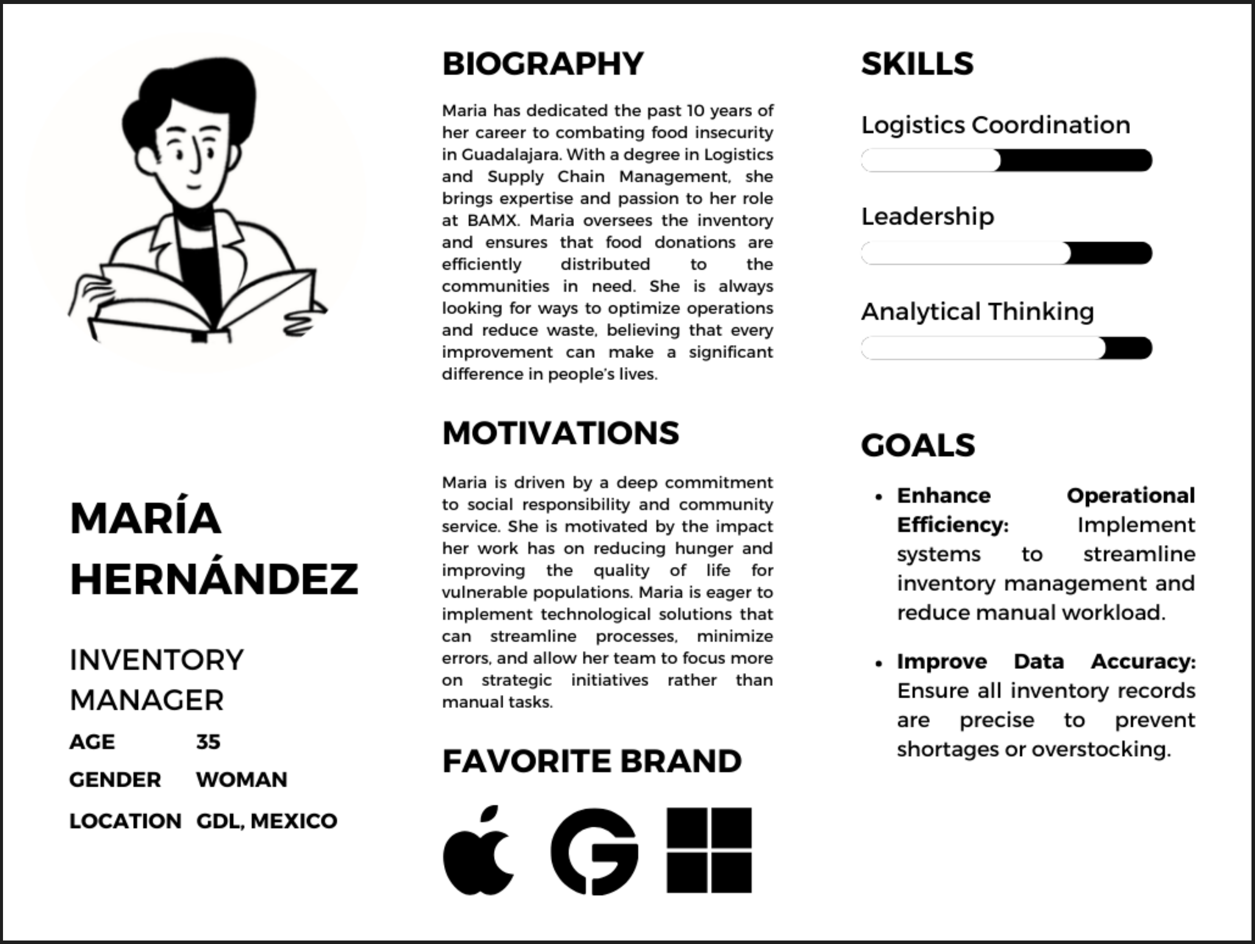
Customer Journey Map

	Awareness	Consideration	Decision	Service	Loyalty
Customer Actions	Recognizes inefficiencies in manual inventory processes	Searches for solutions to improve efficiency	Decides to adopt the new mobile application	Uses the app daily for inventory management	Recommends the app to colleagues and provides feedback
Touchpoints	Staff meetings, error reports, workflow observations	Discussions with peers, online research	App demonstrations, meetings with management	Interaction with app features, customer support if needed	Updates from app developers, user community engagement
Customer Experience	Frustrated	Hopeful	Confident	Satisfied	Loyal
Pain Points	Frustration with time-consuming tasks and errors	Concerns about training time and change adaptation	Worries about data migration and initial setup	Occasional technical glitches, need for quick assistance	Desire for new features, fear of app becoming outdated
Solutions	Information sessions about app benefits	Training materials, hands-on tutorials	Assurance of support during transition	Responsive support team, regular app updates	Continuous improvements, loyalty incentives

Figure 2: Customer Journey Map showing the stages of adoption for the BAMX mobile application.

8 Persona: María Hernández

The persona of María Hernández, an inventory manager at BAMX, provides a detailed profile of her role, motivations, and challenges. With over 10 years of experience in logistics and supply chain management, María is dedicated to improving the efficiency and accuracy of food distribution at BAMX. Her goals focus on enhancing operational efficiency and improving data accuracy within the inventory system. This persona helps to humanize the target user and guides the design decisions for the mobile app to ensure it meets her needs for streamlining processes and minimizing manual tasks.



MARÍA HERNÁNDEZ

INVENTORY MANAGER

AGE	35
GENDER	WOMAN
LOCATION	GDL, MEXICO

BIOGRAPHY

Maria has dedicated the past 10 years of her career to combating food insecurity in Guadalajara. With a degree in Logistics and Supply Chain Management, she brings expertise and passion to her role at BAMX. Maria oversees the inventory and ensures that food donations are efficiently distributed to the communities in need. She is always looking for ways to optimize operations and reduce waste, believing that every improvement can make a significant difference in people's lives.

MOTIVATIONS

Maria is driven by a deep commitment to social responsibility and community service. She is motivated by the impact her work has on reducing hunger and improving the quality of life for vulnerable populations. Maria is eager to implement technological solutions that can streamline processes, minimize errors, and allow her team to focus more on strategic initiatives rather than manual tasks.

FAVORITE BRAND

SKILLS

Logistics Coordination

Leadership

Analytical Thinking

GOALS

- **Enhance Operational Efficiency:** Implement systems to streamline inventory management and reduce manual workload.
- **Improve Data Accuracy:** Ensure all inventory records are precise to prevent shortages or overstocking.

Figure 3: Persona of María Hernández, BAMX Inventory Manager, highlighting her goals, motivations, and challenges.

9 User Story

9.1 User Authentication

- *As a user, I want to be able to enter my username and password to access the application, so that only I can see and manage the relevant information.*
- **Acceptance Criteria:**
  - The system must validate the username and password before allowing access.
  - If the credentials are incorrect, a clear error message must be displayed.

9.2 User Registration

- *As a new user, I want to be able to register for the application by providing my full name, email, username, and password, in order to access and use the functionalities of the application.*

- **Acceptance Criteria:**
  - The system must validate that all required fields are complete and correct before creating an account.
  - Once registered, the new user should be redirected to the login screen with their account created.

9.3 Differentiated Interface (Administrator)

- *As an administrator, I want to access an interface that allows me to view and manage advanced features, so I can efficiently manage the system and users.*
- **Acceptance Criteria:**
  - Administrators must have access to all advanced features specific to inventory and user management.
  - Only administrators can modify permissions for other users.

9.4 Differentiated Interface (Standard User)

- *As a standard user, I want to see a simplified interface with basic options, so I can record product entries and exits without complications.*
- **Acceptance Criteria:**
  - Standard users should see only basic options such as recording product entries and exits.
  - The system should prevent a standard user from accessing administration functions.

9.5 Product Registration

- *As a user, I want to be able to register a new product in the system by providing its name, category and unit of measure, to keep the inventory up to date.*
- **Acceptance Criteria:**
  - The user should be able to register a product only if all the required details are provided (name, category, unit).

9.6 Search for Existing Products

- *As a user, I want to be able to search for an existing product in the inventory before recording an entry or exit, to ensure that I am managing the correct product.*
- **Acceptance Criteria:**
  - The user should be able to search for an existing product by name in the search bar.

9.7 Product Entry Record

- *As a user, I want to record a new product entry specifying the quantity and the donor, so that the inventory reflects the donations received correctly.*

- **Acceptance Criteria:**
  - The system must allow recording an entry only if the quantity and the donor are entered.
  - The operation must automatically record the date and time of the entry.

9.8 Product Exit Record

- *As a user, I want to be able to record a product exit specifying the quantity, to keep the inventory updated according to the deliveries made.*
- **Acceptance Criteria:**
  - The system must allow recording an exit only if the correct quantity of the product is entered.
  - The operation must automatically record the date and time of the exit.

9.9 Real-Time Inventory Viewing

- *As an administrator, I want to be able to view inventory in real-time with the ability to search and filter products to efficiently manage stock.*
- **Acceptance Criteria:**
  - Administrators should be able to view updated inventory with search and filter options.
  - Products that are below the minimum quantity should be highlighted.

9.10 Product Management (Administrator)

- *As an administrator, I want to be able to modify information for an existing product, including its category, and units, to keep inventory properly updated.*
- **Acceptance Criteria:**
  - Administrators should be able to modify information for an already registered product.
  - The system should verify that the modified fields are valid before applying the changes.

9.11 Operation History

- *As an administrator, I want to be able to see a history of all operations performed on the inventory, to track inputs and outputs and make informed decisions.*
- **Acceptance Criteria:**
  - Administrators should be able to see a complete history of all operations with details such as date, time, and user.
  - The history should be filterable by user.

9.12 User Modification

- *As an administrator, I want to be able to modify a user's information, including their permissions, to ensure that they only have access to the necessary functions.*
- **Acceptance Criteria:**
  - Administrators should be able to modify any user's information, including permissions.
  - The system should validate that the changes are correct before saving them.

9.13 Report Generation

- *As an administrator, I want to generate detailed inventory and operations reports for a specific date range, to have a complete overview of warehouse activities.*
- **Acceptance Criteria:**
  - Administrators should be able to select a date range and generate a detailed inventory and operations report.
  - The report should be available for download in an accessible format (such as TXT).

9.14 Low Inventory Notifications

- *As an administrator, I want to receive notifications when a product category is below its minimum quantity, so I can take preventive actions before products run out.*
- **Acceptance Criteria:**
  - The system should generate an automatic notification when a category is below its minimum quantity.
  - Notifications should be visible in a list accessible from the user interface.

10 Functional Requirements

1. **Intuitive and Easy-to-Use Interface:** The system must have an intuitive and easy-to-navigate user interface, designed so that BAMX employees can operate the system without extensive training, facilitating rapid adoption of the system.
2. **Scalability:** The system must be scalable to handle growth in inventory and user volume without losing efficiency, allowing BAMX to adapt to increases in operational capacity without requiring significant changes to the system infrastructure.
3. **Color Palette Aligned to the Organization:** The application interface must use a color palette that is aligned with the visual identity of BAMX, ensuring that the design reflects the brand and culture of the organization.
4. **User Authentication:** The application must allow users to authenticate using a username and password and handle login errors such as incorrect username or password.
5. **User Registration:** There should be functionality that allows new users to register by providing their full name, email, username, and password, with validation to ensure all fields are correctly filled out before

proceeding.

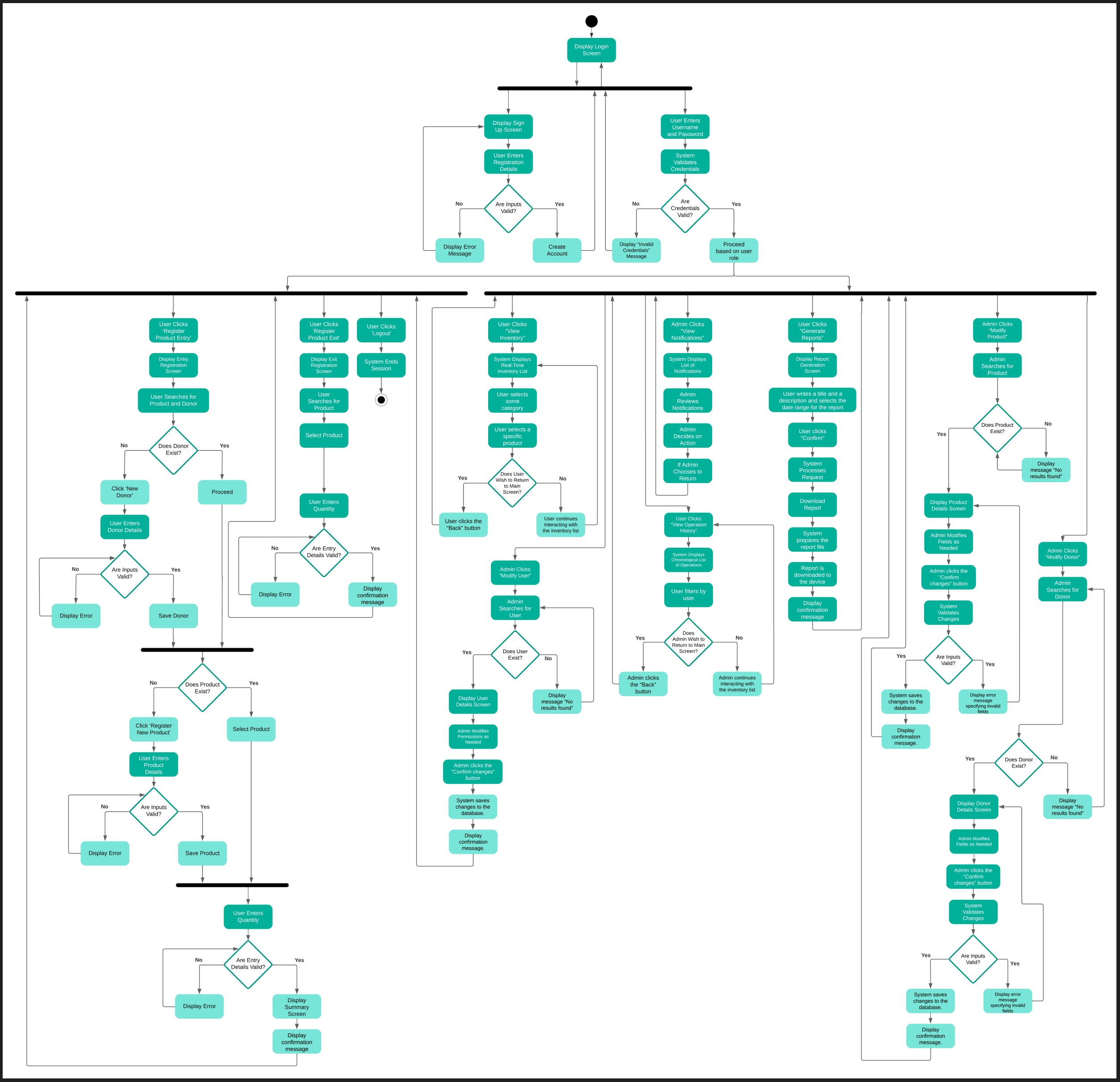
6. **Differentiated Interface Management (Administrator/User):** The application should change the interface and available functionalities based on the type of user (administrator or standard user). Only the administrator can grant additional permissions to other users.
7. **Product Registration:** Users should be able to register new products in the system by specifying details such as name, category, and unit type (units, kilograms, or liters).
8. **Inventory Input and Output Registration:** The application should allow users to register product inputs and outputs from inventory. Each operation should record the quantity, the corresponding unit of measure, the donor (for inputs), and the date and time of the operation.
9. **Real-Time Inventory Management:** Administrators should be able to view inventory in real time, with the ability to filter products by different categories, as well as identify products that are below the established minimum inventory.
10. **Operation History:** The application should maintain a detailed history of all operations (inputs and outputs), including information such as date, time, user who performed the operation, product, donor (if applicable), and quantity handled.
11. **User and Permission Management:** Administrators should be able to modify user information, including updating permissions to allow them access to additional features.
12. **Report Generation:** The application should allow administrators to generate reports based on a selected date range, which will include information on the current status of the inventory and the history of operations.

11 Non-Functional Requirements

1. **Data Security:** The application must implement security measures to protect user data and inventory information, including encryption of passwords and sensitive data, as well as protection against SQL injection attacks.
2. **Performance:** The application must be able to handle inventory queries, transaction logging, and report generation without excessive wait times. The response time for any operation must not exceed 10 seconds under normal usage conditions.
3. **Scalability:** The system must be designed to scale efficiently as the number of users and amount of data grow, without sacrificing performance or availability.
4. **Availability:** The application must be available to users at all times, ensuring that BAMX employees can access and operate the system at all times.
5. **Compatibility:** The app must be compatible with both Android and iOS operating systems, and must function consistently on a variety of mobile devices with different screen resolutions.
6. **Usability:** The user interface must be intuitive and easy to navigate, designed so that BAMX employees can operate the app without extensive training.
7. **Error Handling:** The app must include robust error handling, clearly informing users when a problem occurs and, if possible, suggesting corrective actions.
8. **Energy Efficiency:** The app must be optimized to minimize power consumption, ensuring that it runs efficiently on resource-constrained devices and does not drain the battery quickly.

12 Activity Diagram

The activity diagram illustrates the flow of various processes within the BAMX inventory management system. It details user interactions such as login, product registration, donor information handling, and inventory input/output operations. Each process is represented by decision points and sequential steps that guide the user through the system's functionalities. This diagram serves as a crucial visual aid in understanding the application's workflow, ensuring that each action leads to the correct operational outcome. It also helps identify potential points for error handling and system validation.



Display Login Screen
Display Sign Up Screen
User Enters Registration Details
Are Inputs Valid?
No
Yes
Display Error Message
Create Account
User Enters Username and Password
System Validates Credentials
Are Credentials Valid?
Display "Invalid Credentials" Message
Proceed based on user role
User Clicks 'Register Product Entry'
Display Entry Registration Screen
User Searches for Product and Donor
Does Donor Exist?
Click 'New Donor'
Proceed
User Enters Donor Details
Display Error
Save Donor
Does Product Exist?
Click 'Register New Product'
Select Product
User Enters Product Details
Save Product
User Enters Quantity
Are Entry Details Valid?
Display Summary Screen
Display confirmation message
User Clicks 'Register Product Exit'
Display Exit Registration Screen
User Searches for Product
User Clicks 'Logout'
System Ends Session
User Clicks "View Inventory"
System Displays Real-Time Inventory List
User selects some category
User selects a specific product
Does User Wish to Return to Main Screen?
User clicks the "Back" button
User continues interacting with the inventory list
Admin Clicks "Modify User"
Admin Searches for User
Does User Exist?
Display User Details Screen
Display message "No results found"
Admin Modifies Permissions as Needed
Admin clicks the "Confirm changes" button
System saves changes to the database.
Display confirmation message.
Admin Clicks "View Notifications"
System Displays List of Notifications
Admin Reviews Notifications
Admin Decides on Action
If Admin Chooses to Return
User Clicks "View Operation History"
System Displays Chronological List of Operations
User filters by user
Does Admin Wish to Return to Main Screen?
Admin clicks the "Back" button
Admin continues interacting with the inventory list
User Clicks "Generate Reports"
Display Report Generation Screen
User writes a title and a description and selects the date range for the report
User clicks "Confirm"
System Processes Request
Download Report
System prepares the report file
Report is downloaded to the device
Admin Clicks "Modify Product"
Admin Searches for Product
Display Product Details Screen
Admin Modifies Fields as Needed
Admin clicks the "Confirm changes" button
System Validates Changes
Display error message specifying invalid fields
Admin Clicks "Modify Donor"
Admin Searches for Donor
Does Donor Exist?
Display Donor Details Screen

Figure 4: Activity Diagram illustrating the workflow of the inventory management system, including user actions, decision points, and system responses.

13 Project Management Plan

13.1 Project Scope

13.1.1 Problem Definition and Solution

- **Problem:** The Guadalajara Food Bank (BAMX) faces significant difficulties in optimizing its operations due to the reliance on manual processes, particularly in inventory management. This situation not only limits operational efficiency, but also reduces the potential impact of the resources that BAMX can mobilize to combat food insecurity in vulnerable communities.
- **Solution:** The proposed solution is to develop a functional prototype of a mobile application that effectively integrates inventory management. This application should be designed to streamline daily operations, reduce errors associated with manual processes, and improve data accuracy and availability. Ultimately, the application should increase the efficiency and effectiveness of BAMX, allowing for better management of resources.

13.1.2 Measurable Benefits of the Solution

- **Operational Efficiency:** Implementing this application will significantly reduce the time required to complete tasks related to inventory management, and minimize errors that often occur in manual processes. This will allow BAMX staff to devote more time and resources to other critical areas of their operation.
- **Better Resource Management:** Access to real-time data on inventory will provide a solid basis for more effective decision making. This will allow BAMX to allocate resources more strategically, maximizing the impact of its interventions and ensuring that food reaches those who need it most.

13.1.3 Beneficiaries of the Solution

BAMX Staff: The primary beneficiaries of this solution will be BAMX employees, who will find it easier and more efficient to manage inventory and the distribution of food to beneficiaries. The application will improve their ability to fulfill the food bank's mission and contribute to a more productive work environment.

13.1.4 Deliverables Included in Scope

- **Requirements Documentation:** Developing a comprehensive list of functional and non-functional requirements for the application. This documentation will serve as a guide during project development and ensure that all user expectations and needs are met.
- **Design Documents:** Developing UI/UX design prototypes that represent how the application will look and function. Additionally, architectural design documents will be included, covering the database schema and system architecture, ensuring that the technical design is aligned with the project goals.
- **Prototype Development:** Creating a functional prototype that includes the basic implementation of core functionalities such as data input and output, user interfaces, and data security features. This prototype will be the basis for future testing and iteration, ensuring that all essential functions are well-integrated.

- **Testing and Quality Assurance:** Development of detailed test plans, creation of specific test cases and generation of error reports. Functional testing will be performed on all application modules to ensure that they meet the required quality standards.
- **Deployment and Demonstration:** Deployment of the prototype in controlled test environments, followed by a presentation and demonstration to BAMX stakeholders. This phase will ensure that the prototype is ready to be evaluated and validated before its full implementation.

13.1.5 Out of Scope Deliverables

Full Production Deployment: Full-scale deployment in a production environment is out of scope for this project. Implementation of advanced features and additional customizations beyond the basic functionality required for the prototype are also excluded. Also, long-term maintenance and support after the prototype phase will not be addressed at this stage.

13.2 Project Timeline

The project will be executed following a detailed 10-week plan, ensuring that each stage is completed thoroughly and aligned with the overall goal of delivering a functional, operational prototype.

13.2.1 Week 1: Project Initiation and Requirements Gathering

Activities:

- Conduct a project kickoff meeting with all key stakeholders to establish a common understanding of the project's goals, scope, and expectations.
- Detailed collection of functional and non-functional requirements through interviews, surveys, and analysis of BAMX's needs.
- Comprehensive documentation of the collected requirements, which will serve as a reference for all subsequent phases of the project.
- Establishment of specific project objectives, definition of key deliverables, and success criteria that will be used to assess the achievement of objectives.

Responsible: Diego, with the active participation of all team members.

13.2.2 Week 2: System Design and Planning

Activities:

- Creating detailed user interface (UI) and user experience (UX) designs for the application, ensuring that they meet the established requirements and are intuitive for end users.

- Designing the system architecture, including the development of a database schema and defining the API structures necessary for the integration of the different system components.
- Selecting and defining the technology stack and tools to be used during the development of the project, ensuring that they are the most appropriate for the needs of BAMX.
- Developing a detailed project plan, including identifying key milestones, defining specific tasks, and creating a schedule that ensures the completion of the project within the established time frame.

Responsible: Luis Daniel (UI/UX), Jaziel (UI/UX), Jorge (Database), Milan (APIs), under the supervision of Diego.

13.2.3 Week 3: Design Review and Approval

Activities:

- Detailed review and refinement of UI/UX designs to ensure they meet your specific expectations and needs.
- Finalization of system architecture and design documents, which will serve as a guide for the technical development of the project.
- Beginning of the configuration of the development environments, ensuring that all team members have access to the necessary tools and resources to begin the implementation of the prototype.

Responsible: Diego, with the participation of Luis Daniel (UI/UX), Jorge (Database) and Milan (APIs).

13.2.4 Weeks 4-7: Development

Activities:

- Implementation of the UI/UX design.
- Development and testing of the features to ensure they meet the functional and non-functional requirements established during the requirements gathering phase.
- Conducting thorough unit testing on each component to identify and fix potential bugs before full integration into the system.

Responsible: Milan (APIs), Jorge (Database), Luis Daniel (UI/UX), under Diego's supervision.

13.2.5 Week 8: User Acceptance Testing (UAT)

Activities:

- Organizing and conducting test sessions within the team (Dogfooding), to validate the functionality and

usability of the prototype.
- Collecting detailed feedback from team members, identifying areas for improvement and necessary adjustments to the prototype.
- Implementing suggested adjustments and improvements to ensure that the prototype meets the success criteria defined at the start of the project.

Responsible: Diego, with the collaboration of all team members.

13.2.6 Week 9: System Integration and Testing

Activities:

- Integration of all developed functionalities and components into a cohesive and functional prototype that represents the preliminary version of the full application.
- Performing integration tests to ensure that everything works together smoothly, without conflicts or errors.
- Identifying and correcting any significant errors or inconsistencies that are detected during integration testing, ensuring that the prototype is ready for the next phase of user acceptance testing.

Responsible: Diego (Testing), Milan (APIs), Jorge (Database), Luis Daniel (UI/UX).

13.2.7 Week 10: Finalization and Deployment

Activities:

- Finalization of the prototype, including implementation of improvements identified during acceptance testing.
- Preparation for deployment of the prototype in a test environment, ensuring that all technical and security aspects are fully covered.
- Conducting a final review of the prototype and presenting a full demo to BAMX stakeholders, ensuring they are fully satisfied with the outcome before deployment.

Responsible: Diego and all team members.

13.3 Project Effort Estimation

The project effort estimation for the mobile application developed for the Guadalajara Food Bank (BAMX) is based on the Function Point Analysis (FPA) methodology, which measures the complexity and functionality of the system in terms of function points (FP). This analysis provides an objective way to estimate the effort required for the development, testing, and deployment of the system by breaking down its components and assessing their complexity.

13.3.1 Function Point Breakdown

The system has a total of **53 function points (FP)**, which are distributed among various components of the application, including user authentication, inventory management, user registration, and product tracking. Each component has been evaluated based on its type (e.g., external input, external output, internal logic file) and the complexity associated with processing data and interacting with the system.

13.3.2 Effort per Function Point

Based on industry standards and the complexity of the project, we estimate that each function point will require approximately **20 hours** of effort. This time includes tasks such as UI/UX design, development, testing, debugging, integration, and deployment. Given the total of **53 FP**, the estimated effort for the project is:

- **Effort per FP:** 20 hours
- **Total Function Points:** 53 FP
- **Total Effort Estimate:** 53 FP × 20 hours = **1,060 hours**

This effort will be distributed across the project team members based on their roles and responsibilities.

13.3.3 Effort Distribution by Team Member

- **Luis Daniel (UI/UX Implementation):** Responsible for implementing the user interface and user experience. Estimated effort: **200 hours**.
- **Jaziel (UI/UX Design):** Responsible for the UI/UX design in Figma, collaborating closely with Luis Daniel to ensure smooth implementation. Estimated effort: **160 hours**.
- **Milan (API and Endpoint Development):** In charge of creating and securing the API and endpoints, facilitating communication between the frontend and backend. Estimated effort: **220 hours**.
- **Jorge (Database Management):** Manages database architecture and ensures secure and efficient data handling. Estimated effort: **180 hours**.
- **Diego (Documentation, Testing, and Project Management):** Oversees the project, manages documentation, conducts testing, and ensures project milestones are met. Estimated effort: **300 hours**.

13.3.4 Total Project Effort

The total estimated effort for the entire project is **1,060 hours**.

13.4 Project Cost

The project cost has been calculated based on the total estimated effort and the hourly rate for the development team. We assume a rate of **$100 MXN per hour**, which reflects the cost of labor for each team member, including UI/UX design, development, and project management.

13.4.1 Cost Breakdown

- **Total Effort:** 1,060 hours
- **Hourly Rate:** $100 MXN/hour
- **Total Cost Estimate:** 1,060 hours × $100 MXN/hour = **$106,000 MXN**

13.4.2 Contingency Buffer

To account for potential risks such as delays, technical challenges, and additional testing, a **contingency buffer of 15%** has been added to the total cost. This buffer ensures the project can absorb unforeseen complexities without significantly impacting timelines or deliverables.

- **Contingency Buffer (15%):** $15,900 MXN
- **Total Project Cost (Including Contingency):** $106,000 MXN + $15,900 MXN = **$121,900 MXN**

13.4.3 Project Cost Summary

- **Total Effort:** 1,060 hours
- **Cost per Hour:** $100 MXN/hour
- **Base Project Cost:** $106,000 MXN
- **Contingency (15%):** $15,900 MXN
- **Total Project Cost: $121,900 MXN**

This estimation ensures that the project remains on budget while delivering a high-quality solution to BAMX, with room for adjustments if necessary.

13.5 Risk Matrix

N°	C/O	Type of risk	Risk	I	P	C	Value	Level
1	Technical	Lack of experience	Limited knowledge in React Native	3	3	9	3	High
2	Schedule	Tight deadlines	Insufficient time for thorough testing	3	3	9	3	High

3	Experience	Learning curve	Steep learning curve for new tools (e.g., Firebase, Expo)	3	3	9	3	High
4	Scope	Scope creep	Adding new features mid-project	3	2	6	3	High
5	Technical	Integration issues	Difficulty integrating different components	3	2	6	3	High
6	Technical	Debugging and troubleshooting	Difficulty in identifying and fixing bugs	3	2	6	3	High
7	Schedule	Inadequate testing resources	Limited tools and time for adequate testing	3	3	9	3	High
8	Technical	Dependency management	Issues with third-party libraries or APIs	2	3	6	2	Medium

9	Technical	Performance issues	Poor performance due to inefficient code	2	2	4	2	Medium
10	Schedule	Team fatigue	High workload leading to burnout	3	2	6	3	High
11	Technical	Version control problems	Issues with Git (e.g., merge conflicts, loss of work)	3	2	6	3	High
12	Technical	Insufficient testing coverage	Missed bugs due to inadequate test cases	3	3	9	3	High
13	Scope	Incomplete features	Not all planned features implemented in time	3	3	9	3	High
14	Technical	Deployment challenges	Difficulty in deploying the prototype for demo	3	2	6	3	High

15	Schedule	Poor communication	Misalignment between team members due to poor communication	3	2	6	3	High

Table 1: Risk Matrix detailing potential risks, their impact (I), probability (P), consequence (C), calculated risk value, and risk level.

13.6 Contingency Matrix

Risk ID	Component / Output (WBS)	Contingency Plan
1	Development	Plan intensive React Native training sessions for the entire team at the start of the project.
2	Testing	Establish a detailed testing schedule and prioritize critical tests from the outset.
3	Development	Allocate more time for the learning curve during the initial weeks and request external support if necessary.
4	Scope Management	Implement a strict change control process and obtain approvals before adding new features.
5	Development	Break down integration tasks into smaller steps and conduct frequent unit tests to catch issues early.
6	Debugging	Set up a daily code review routine and allocate extra time for debugging during each sprint.
7	Testing	Prioritize automated tests to maximize coverage with available resources and reduce manual effort.

8	Development	Maintain an up-to-date log of all dependencies and run frequent local tests to ensure external libraries or APIs do not disrupt development. If major issues arise, consider using more stable alternatives.
9	Performance	Perform code reviews focusing on efficiency and performance optimization before implementation.
10	Team Management	Implement a workload balancing policy and allow for adequate breaks to avoid burnout.
11	Version Control	Conduct training on best practices for Git and establish merge reviews to reduce conflicts and prevent work loss.
12	Testing	Continuously review and expand test cases throughout the project to ensure broader coverage and avoid undetected issues.
13	Scope Management	Clearly define priorities and ensure that critical items are completed first, leaving additional features for the end.
14	Prototype Preparation	Conduct extensive testing in a local or staging environment for the demo, ensuring the prototype functions properly. Prepare a local version of the prototype with simulated data to demonstrate functionality without requiring full deployment.
15	Team Communication	Implement quick daily stand-up meetings and use project management tools to maintain clear and effective communication among team members.

Table 2: Contingency Matrix outlining contingency plans for identified risks.

14 Database Description

14.1 Users

user_id: Primary key that uniquely identifies each user.
full_name, email, username, password: Contact information and authentication credentials.
role: Enum that defines whether a user is ”admin” or ”user”.
creation_date, last_session: Timestamps to record when the account was created and when the user last logged in.

14.2 Products

product_id: Primary key that identifies each product.
product_name: Name of the product.
category: Relates products to the categories table.
unit_type: Enum that indicates whether it is handled in units, kilograms, or liters.
current_quantity, minimum_quantity: Values indicating the current quantity of the product and the minimum quantity before triggering a notification.
creation_date, modification_date: Timestamps that record when the product was created and when it was last modified.

14.3 Donors

donor_id: Primary key to identify each donor.
donor_name, contact: Donor identification and contact information.
creation_date: Timestamp that records when the donor entry was created.

14.4 Inventory

inventory_id: Primary key to identify an inventory operation.
operation_type: Enum that can be ”input” or ”output”, indicating whether the operation was to add or remove products.
product_id: Relationship to the products table.
quantity: Quantity of products entered or withdrawn.
user_id: Relationship with the user table that indicates who performed the operation.
donor_id: Optional relationship to record the donor in case of an input operation.
operation_date: Timestamp that records when the operation was performed.

14.5 Operation History

operation_id: Primary key that identifies each operation.
user_id, product_id: Relationships that identify the user who performed the operation and the product involved.

operation_type: Similar to the inventory table, it specifies whether it was an input or output.
quantity: Quantity of products involved.
operation_date: Records the date and time of the operation.

14.6 Notifications

notification_id: Primary key that identifies a notification.
product_id: Related to the product to which the notification corresponds.
message: Text describing the notification.
notification_date: Timestamp of when the notification was generated.
status: Enum indicating whether the notification is ”pending” or ”resolved”.

14.7 Categories

category_id: Primary key to identify a category.
category_name: Descriptive name of the category.

14.7.1 Reference of the Categories that BAMX Manages

Category	Products
Basic Basket (6 varieties)	Eggs, rice, pasta, sugar, oats, soy, beans, lentils, oil, chick-peas.
Fruits and Vegetables (4 varieties)	14 kilograms
Meat, Cold Cuts, and Dairy (2 varieties)	Ham, sausage, milk, cheese, yogurt, butter, cream, meat.
Groceries (4 varieties)	Dressing, canned goods, cereal, cookies, ketchup, sliced bread, general pantry items.
Non-food Items (1 kilogram)	Cleaning and personal hygiene products.

14.8 Reports

report_id: Primary key for each generated report.
date_range: Time period covered in the report.
generation_date: Timestamp of when the report was generated.
user_id: Relationship to the users table indicating who generated the report.
file: Path or name of the file generated for the report.

14.8.1 Relationship between tables

- **Products** is related to **Categories**, **Inventory**, **Notifications** and **Operation History**.
- **Users** is related to **Inventory**, **Operation History** and **Reports**.

- **Donors** is related to **Inventory**.
- **Notifications** is related to **Products**.

14.9 Database Diagram

The database diagram provides a clear representation of the structure and relationships between the key tables in the BAMX inventory management system. It outlines the primary and foreign keys, along with the attributes stored in each table, which include users, products, donors, inventory, operation history, notifications, categories, and reports. The diagram helps to visualize how the different entities interact with each other, ensuring a smooth flow of data across the system.

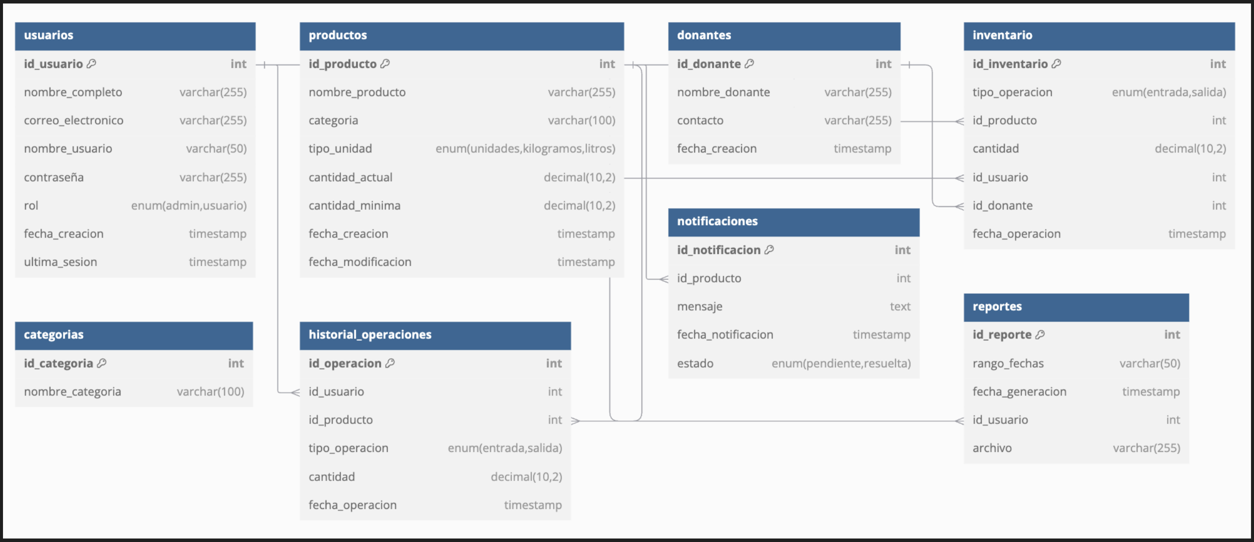


Figure 5: Database diagram illustrating the relationships between users, products, donors, inventory, and other key tables.

15 Test Strategy

15.1 Priorities In Our Testing

15.1.1 Fluency

Ensuring a smooth user interface is critical for BAMX inventory managers to operate the application without interruptions, especially in situations where quick access to information is needed to distribute food to vulnerable communities.

15.1.2 Tediousness

Reducing tediousness is essential to avoid user frustration. Designing straightforward and efficient workflows will allow BAMX employees to focus on strategic tasks without wasting time on repetitive processes.

15.1.3 Business Scenarios

Focusing on the most critical business scenarios ensures that the most utilized functionalities are well-covered and tested, maximizing the application's effectiveness in optimizing BAMX operations.

15.1.4 Stability

Stability is essential to ensure that the application functions reliably and is available when users need it, maintaining operational continuity, especially during high-demand periods.

15.1.5 Confidentiality

Protecting the confidentiality of sensitive data is key to ensuring that only authorized individuals have access to information, complying with data protection regulations, and maintaining user trust.

15.1.6 Functionality

Ensuring that all key functionalities, such as inventory management and expiration alerts, work correctly is fundamental to the success of the project, guaranteeing that the application meets its purpose.

15.1.7 Data

Proper data management is essential to avoid issues like data loss, duplication, or corruption, ensuring the integrity and reliability of the system.

15.2 Testing Types In Our Testing

15.2.1 Regression Testing

Testing to ensure that new changes do not negatively affect the existing functionality of the application.

15.2.2 Dogfooding

Internal testing where the development team uses the application in real-world scenarios to identify usability issues.

15.2.3 Violate Data Format Rules

Testing the system's ability to handle improper data formats and ensuring it provides appropriate error handling.

15.2.4 Usability Testing

Evaluating the user interface and experience to ensure it is intuitive, efficient, and easy to use for BAMX staff.

15.2.5 Security Testing

Ensuring the application properly protects sensitive data and adheres to security protocols, preventing unauthorized access.

15.2.6 Compatibility Testing

Testing the application across different devices, operating systems, and environments to ensure consistent performance.

15.2.7 Configuration Testing

Verifying that the application functions correctly with different configurations and system settings, ensuring flexibility.

15.3 Test Schedule

15.3.1 Sprint Structure and Activities

- **Sprint Duration**: 1 Sprint = 1 Week. With the limited time of 3 remaining weeks, we will complete two sprints before the prototype presentation.
- **Sprint Planning**: Each sprint will begin with a **Sprint Planning session** where the entire development team will participate in selecting and prioritizing **user stories**, **tasks**, and **bug tickets** based on urgency and impact on the project.
- **Test Planning**: Following Sprint Planning, a **Test Planning session** will be held with the Quality Engineer (QE) to define the specific testing activities for that sprint.

15.3.2 Sprint Breakdown (10 working days)

First Sprint (Days 1-5):

- **Static Testing Activities**:
  - **Ticket review & clarification**: QE will review and clarify any questions regarding the tickets selected during Sprint Planning.
  - **Code review**: An initial code review will be conducted to identify potential issues before the code is executed.
  - **Kickoff & technical reviews**: A kickoff meeting will be held to align expectations and ensure all team members are on the same page.
  - **Design review & clarification**: QE will ensure that the UI/UX and architectural designs are clear and feasible for implementation.
  - **Test case design**: During this phase, test cases will be designed to cover the functionalities planned for the sprint.

  - **Test execution preparation**: Testing environments and necessary data will be prepared for the test execution in the second half of the sprint.

Second Sprint (Days 6-10):

- **Dynamic Testing Activities**:
  - **Test case execution**: QE will execute the designed test cases.
  - **Test reporting**: Detailed reports on the testing status will be generated, including results and any bugs found.
  - **Dogfooding**: The application will be used internally by the team to identify practical issues and improve the user experience.

15.3.3 Focus Areas for Each Sprint

Sprint 1 (Weeks 1):

- **User Stories**: Focus on completing the core inventory management functionalities.
- **Testing Focus**:
  - **Functional Testing** of implemented functionalities.
  - Initial **Stress Testing** to ensure the system can handle basic load.
  - **Violate Data Format Rules** to ensure the application handles malformed data correctly.

Sprint 2 (Weeks 2):

- **User Stories**: Finalization and refinement of functionalities, with an emphasis on usability and system stability.
- **Testing Focus**:
  - **Dogfooding** to identify final improvements from the user's perspective.
  - **Security Testing** to ensure the system is protected against common vulnerabilities.

15.3.4 Post-Sprint Activities

- **Final Prototype**: At the end of the second sprint, the team will prepare the final prototype, integrating all components and ensuring the system is ready for presentation.
- **Demonstration and Feedback**: An internal demonstration will be conducted, followed by feedback collection to fine-tune details before the final presentation.

15.4 Test Planning

15.4.1 Post-Sprint Planning Alignment

- After the Sprint Planning meeting, the QE will conduct a Test Planning session. This is crucial to ensure that all testing tasks are well-defined and aligned with the sprint's goals and priorities.
- The session will begin with a review of the user stories, tasks, and bug tickets selected during Sprint Planning, focusing on understanding the testing requirements for each.

15.4.2 Documentation of Testing Scope

- The QE will document all tickets or tasks that require testing within the sprint. This includes identifying the specific functionalities, integrations, and components that need to be tested.
- The QE will define the testing scope for each ticket, ensuring that all key aspects of the project, such as inventory management and alert systems, are covered.

15.4.3 Task Organization and Prioritization

- The QE will organize the testing tasks, establishing a timeline and priority for each. For example:
  - **High priority**: Testing essential functionalities like inventory management.
  - **Medium priority**: Performing stress testing on the system to handle large volumes of data.
  - **Low priority**: Conducting exploratory testing towards the end of the sprint if time permits.

15.4.4 Defining Communication Protocols

- Even though there is only one QE, it's essential to maintain clear communication with the rest of the development team. This might include:
  - Regular updates on the status of testing and any issues encountered.
  - Using project management tools to report bugs and request clarifications from developers.

15.4.5 Test Case Management

- The QE will be responsible for designing, documenting, and managing test cases. This includes:
  - Defining the structure and format for test case documentation, ensuring they are clear and comprehensive.
  - Keeping a record of test cases and their status in a test management tool or a well-organized document.

15.4.6 Automation Strategy

- The QE will assess which tests can be automated and how to integrate them into the CI/CD pipeline, considering available resources and time.
- The focus will be on automating regression tests for critical functionalities, using the most appropriate tools (e.g., Jest, Maestro).

15.4.7 Documentation of Agreements and Tasks

- The QE will document all decisions, tasks, and agreements made during the Test Planning session. This documentation will include:
  - The final list of testing tasks and the timeline for their execution.
  - The priorities set for testing and any specific processes to be followed.
- This documentation will be shared with the development team to ensure that everyone is aligned and understands the testing approach for the sprint.

15.5 Agile Testing

In our project, we will follow an agile approach to testing, integrating three key testing activities into each sprint: **New Feature Testing**, **Regression Testing**, and **Confirmation Testing**. Below is a detailed plan on how we will implement each of these activities:

15.5.1 New Feature Testing

1. **Identifying New Features**
   - At the start of each sprint, during planning, we will identify the new features to be developed.
   - The acceptance criteria specified in the user stories will be reviewed to ensure they are clear and complete.
2. **Designing Manual Test Cases**
   - The QE will design detailed manual test cases based on the acceptance criteria for each new feature. These test cases will focus on validating that the feature meets user expectations and the specified requirements.
3. **Test Execution**
   - Once the new feature is ready for testing, the QE will execute the manual test cases, verifying that everything works correctly as specified.
   - Any bugs or defects found will be reported immediately for correction.
4. **Test Automation**

- In parallel, automated tests for these new features will be developed. These automated tests will be integrated into the regression test suite to ensure that new features are not affected by future changes.

15.5.2 Regression Testing

1. **Impact Analysis**
   - At the start of each sprint, an impact analysis will be conducted to identify which existing features might be affected by the new developments or changes.
   - Regression testing will be prioritized based on the impact analysis, ensuring that the most critical functionalities are tested first.
2. **Executing Regression Tests**
   - Most of the regression tests, which will be automated, will be executed in the Continuous Integration (CI) pipeline to quickly detect any defects introduced by new changes.
   - The QE will review the results of these tests and take immediate action if any failures are detected, ensuring that existing functionalities continue to work as expected.
3. **Maintaining Automated Tests**
   - The automated regression tests will be continuously maintained and updated to include new tests related to features added or modified in each sprint.
   - This will ensure that the regression test suite remains relevant and effective in detecting regressions throughout the development process.

15.5.3 Confirmation Testing

1. **Identifying Bugs and Defects**
   - During development and test execution, any bugs or defects detected will be thoroughly documented.
   - Once a bug has been fixed by the developers, the QE will conduct confirmation testing to verify that the fix effectively resolves the issue.
2. **Using Existing Test Cases**
   - The QE will reuse existing test cases whenever possible to validate bug fixes. This allows for quick verification and ensures consistency in testing.
3. **Designing New Test Cases (if necessary)**
   - If a bug fix involves complex changes or has a broad impact, the QE will design new test cases specifically to address these situations and ensure that no unwanted side effects occur.

4. **Automating Confirmation Testing**

   - Whenever possible, confirmation tests will also be automated and integrated into the CI pipeline. This will ensure that bug fixes are continuously validated with each new build.

15.6 Roles and Responsibilities

Tests	**Devs**	**QE**
Unit tests	Primary	Secondary
Integration tests	Primary	Secondary
Infrastructure tests	Primary	Secondary
Performance tests	Primary	Primary
UI tests	Secondary	Primary
Security tests	Secondary	Primary

- **Primary**: The main person(s) in-charge of a test activity.
- **Secondary**: Supplementary role in-charge of reviewing, assisting, and pairing with the primary person(s) in managing the test activity.

15.7 Testing Tools

Testing type	**Testing tool**
Unit Testing	Jest
Integration Testing	Jest / **React Native Testing Library**
UI Testing	**React Native Testing Library** / **Maestro**

15.8 Test Environments

```
Dev -> SIT -> STG
```

15.8.1 DEV Environment

- **New Feature Testing and Exploratory Testing**:
  - New feature testing and exploratory testing will primarily take place in the DEV environment. This is where developers actively work on implementing new features.

  - The QE will conduct initial tests here to ensure that new functionalities are working as expected before moving them to more formal testing environments.

15.8.2 SIT Environment (System Integration Testing)

- **Test Automation Execution**:
  - The SIT environment will be primarily used for test automation execution. This environment is set up to allow integration and regression tests to be run in isolation, without interference from active development.
  - Additionally, the QE can use this environment for isolated manual testing if needed, especially to validate the integration between different system components.

15.8.3 STG Environment (Staging)

- **Product Demos and External Engagements**:
  - The staging (STG) environment will be used for product demos and external engagements, such as presentations to stakeholders or client reviews.
  - This environment is a close replica of what would be a production environment, allowing demos to be conducted in conditions similar to a live environment, although this project will not be deployed to production.

15.8.4 Justification

- **Focus Limited to the First 3 Environments**: Since this project will not be deployed to production, there is no need for UAT (User Acceptance Testing) or Production environments. The primary goal of the project is to ensure that functionalities and the system are fully developed and validated within controlled environments.
- **Finalization in STG**: The staging (STG) environment will serve as the final environment where all necessary validations and demos will be conducted. This environment provides a close enough replica of a production environment, allowing stakeholders to see the product in a near-final state without the need for a live deployment.

15.9 Test Automation

15.9.1 Unit/Component Level

We will write most of our automated tests at the unit level using Jest to validate that each component and function in the React Native application works correctly in isolation.

15.9.2 Integration Level

If unit tests are not sufficient to cover certain functionalities, we will use Jest along with React Native Testing Library to write integration tests. These tests will validate how different components and services interact together.

15.9.3 UI Level

For UI tests, we will implement automated tests using React Native Testing Library. These tests will ensure that the user interface functions correctly and meets the end-user expectations. We will also test the application with Maestro to check that it behaves as it should on real devices.

15.10 Life cycle of a Ticket

Each ticket in a sprint will generally follow this lifecycle:

```
Backlog -> In design -> Ready for dev -> Dev In Progress -> Ready for testing -> Tested -> Done
```

15.10.1 Implementation in Our Project

1. **Backlog**:
   - Tickets start in the *Backlog* column, where all tasks, user stories, and bugs that have not yet been prioritized for a specific sprint are collected.
2. **In design**:
   - Tickets selected for the sprint move to the *In design* column, where the development team and the QE collaborate to clearly define the requirements, acceptance criteria, and necessary technical design.
   - During this stage, the QE will perform static testing activities, such as reviewing requirements and creating preliminary test cases.
3. **Ready for dev**:
   - Once the ticket design is complete and approved, it moves to the *Ready for dev* column. This indicates that the ticket is ready for developers to begin working on its implementation.
4. **Dev In Progress**:
   - Developers work on the ticket while it is in the *Dev In Progress* column. Here, the code is written, unit tests are performed, and integration with other parts of the system takes place.
5. **Ready for testing**:
   - When developers complete their work and consider the ticket ready for testing, they move it to the

Ready for testing column.
- The QE reviews the ticket and begins test execution, evaluating whether it meets the defined acceptance criteria. This may include functional, integration, and automated tests as appropriate.

6. **Tested**:

   - If the ticket passes all tests, it moves to the *Tested* column. This indicates that the ticket has been validated by the QE and meets all specified requirements.
   - If the tests fail, the ticket will move back to *Dev In Progress* with detailed notes on the issues found.

7. **Done**:

   - Once the ticket has passed all tests and has been reviewed, it moves to the *Done* column. This indicates that the ticket has been successfully completed and requires no further work.

15.10.2 Definition of Done Guideline

The QE will use the product team's established *Definition of Done* guideline to evaluate whether a ticket can be considered complete. This includes verifying that all tests have passed, that there are no open issues, and that all acceptance criteria have been met.

15.11 Test Metrics

15.11.1 Test Coverage

- **Unit Level**:
  - **Tool**: We will use **Jest** to run unit tests. At the end of each run, Jest will provide a coverage report showing what percentage of the code is covered by tests.
  - **Goal**: We will focus on achieving reasonable coverage (e.g., 70%), ensuring that the critical parts of the code are tested. We won't aim for 100% coverage but will prioritize covering the most important areas.
- **Integration Level**:
  - **Tool**: We will use **Jest** along with **React Native Testing Library** to create tests that verify how different components of the application interact.
  - **Process**: Instead of trying to cover every possible interaction, we will focus on testing the most critical workflows of the application, such as the interaction between the frontend and any backend APIs or services.
- **UI Level**:
  - **Tool**: We will use **React Native Testing Library** for UI tests and run these tests on Maestro to see how the app behaves in a real-world-like environment.
  - **Goal**: We will ensure that the main screens and key functionalities, such as forms or action buttons, are covered by UI tests. We will manually assess how these tests perform in Maestro on real devices.

15.11.2 Ratio of Failures to Defects

- **Simple Tracking**: Each time a test fails, we will record whether the failure is due to a defect in the code or an issue with the test itself. This will help us identify if our tests are effective at catching bugs.
- **Process**: We will use a simple spreadsheet to track the number of failed tests and associated defects. This will allow us to see patterns and adjust our tests if necessary.

15.11.3 Time Taken to Execute Tests

- **Basic Measurement**: At the end of each test run (unit, integration, and UI), we will note the total time it took to execute all the tests.
- **Goal**: If we notice that tests are taking too long, we will review which tests are the slowest and consider optimizing them or running them more efficiently. This will also help us identify if the tests are impacting the speed of development.

15.11.4 Trend Metrics (Over Multiple Sprints)

- **Tracking Over Time**: Throughout the sprints, we will maintain a simple record of how test coverage, execution time, and the ratio of failures to defects evolve.
- **Data Visualization**: We will use basic charts in a spreadsheet to visualize these metrics over time. This will allow the team to see improvements or identify areas that need more attention, such as an increase in detected defects or a decrease in test coverage.

15.12 Test Monitoring and Reporting

15.12.1 Sprint Review Meetings

We will share test monitoring and reporting data during each Sprint Review meeting. This will ensure that the entire team is informed about testing progress, issues encountered, and the overall quality of the project.

15.12.2 Metrics from Project/Test Management Tools

We will use a project management tool (e.g., Jira) to track task-level metrics and trends. This will include:

- **Task Completion**: Monitoring the status of testing tasks, such as whether unit, integration, or UI tests have been completed.
- **Defect Tracking**: Tracking the number of defects found during testing and their resolution status.

15.12.3 Metrics from Continuous Integration (CI) Tools

We will use a Continuous Integration tool (e.g., GitHub Actions, CircleCI) to gather automation metrics, including:

- **Test Coverage**: Reviewing the automated reports generated by Jest to understand the percentage of code covered by tests.
- **Build Success/Failure**: Monitoring how often tests pass or fail during the CI process, helping us identify any recurring issues with the code.

15.12.4 Manual Data Preparation

For certain metrics that require manual effort, we will prepare the data as follows:

- **Requirement Traceability**: Manually mapping test cases to user stories or requirements to ensure that all critical functionalities are covered.

15.12.5 Reporting

We will compile all the gathered metrics into a simple report to be presented during the Sprint Review meeting. This report will include:

- **Test Coverage Summary**: A summary of how much of the codebase is covered by tests.
- **Defect Summary**: An overview of the defects found, their severity, and their current status.
- **Performance Trends**: Any noticeable trends in test execution times, defect detection, or test coverage over the past sprints.