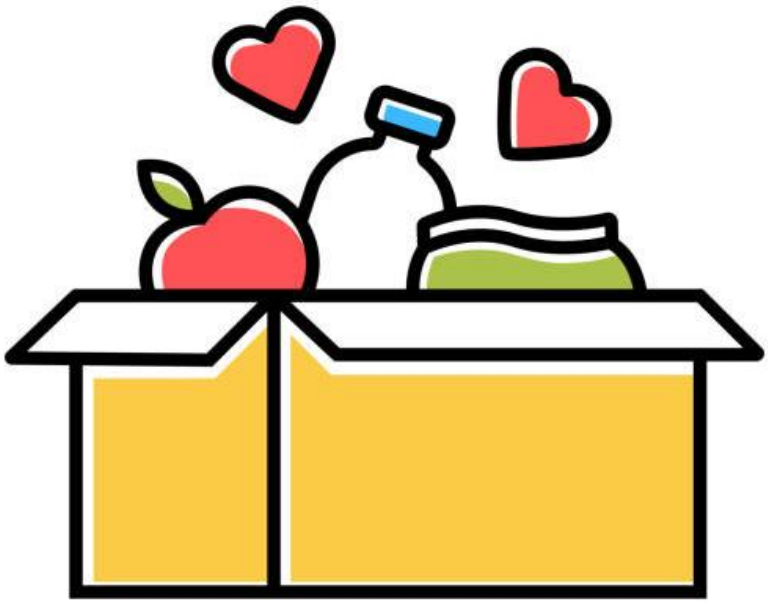
WeAllocate: A Resource Optimizer

Created for IMAN Food and Wellness Center



Prepared by
Bilal Naseer
Umar Khan
Daniyal Khan
David Cardenas
CS 440
at the
University of Illinois Chicago

Fall 2023

[1], [2][3].

Table of Contents

REMOVE OR REPLACE ALL TEXT IN RED ITALICS BEFORE SUBMITTING REPORT ..... **Error! Bookmark not defined.**
How to Use This Document ..... **Error! Bookmark not defined.**
List of Figures 8
List of Tables 9

I Project Description 10

1 Project Overview 10

2 The Purpose of the Project 10

2a The User Business or Background of the Project Effort 10
2b Goals of the Project 11
2c Measurement 12

3 The Scope of the Work 12

3a The Current Situation 13
3b The Context of the Work 13
3c Work Partitioning 16
3d Competing Products 17

4 The Scope of the Product 17

4a Scenario Diagram(s) 17
4b Product Scenario List 18
4c Individual Product Scenarios 18

5 Stakeholders 18

5a The Client 18
5b The Customer 19
5c Hands-On Users of the Product 19
5d Maintenance Users and Service Technicians 19
5e Other Stakeholders 19
5f User Participation 20
5g Priorities Assigned to Users 20

6 Mandated Constraints 21

6a Solution Constraints 21
6b Implementation Environment of the Current System 22
6c Partner or Collaborative Applications 22
6d Off-the-Shelf Software 22
6e Anticipated Workplace Environment 22
6f Schedule Constraints 23
6g Budget Constraints 23

7 Naming Conventions and Definitions 23

7a Definitions of Key Terms 23
7b UML and Other Notation Used in This Document 24
7c Data Dictionary for Any Included Models 25

8 Relevant Facts and Assumptions 26

8a Facts 26
8b Assumptions 26

II Requirements 27

9 Product Use Cases 27

9a Use Case Diagrams 28
9b Product Use Case List 28
9c Individual Product Use Cases 28

10 Functional Requirements 34

11 Data Requirements 35

12 Performance Requirements 37

12a Speed and Latency Requirements 42
12b Precision or Accuracy Requirements 38
12c Capacity Requirements 39

13 Dependability Requirements 39

13a Reliability Requirements 39
13b Availability Requirements 40
13c Robustness or Fault-Tolerance Requirements 41
13d Safety-Critical Requirements 41

14 Maintainability and Supportability Requirements 42

14a Maintenance Requirements 42
14b Supportability Requirements 43
14c Adaptability Requirements 44
14d Scalability or Extensibility Requirements 44
14e Longevity Requirements 45

15 Security Requirements 46

15a Access Requirements 46
15b Integrity Requirements 47
15c Privacy Requirements 47
15d Audit Requirements 48
15e Immunity Requirements 49

16 Usability and Humanity Requirements 50

16a Ease of Use Requirements 50
16b Personalization and Internationalization Requirements 51

16c Learning Requirements 52
16d Understandability and Politeness Requirements 53
16e Accessibility Requirements 54
16f User Documentation Requirements 54
16g Training Requirements 56

17 Look and Feel Requirements 56

17a Appearance Requirements 56
17b Style Requirements 57

18 Operational and Environmental Requirements 58

18a Expected Physical Environment 58
18b Requirements for Interfacing with Adjacent Systems 58
18c Productization Requirements 59
18d Release Requirements 60

19 Cultural and Political Requirements 60

19a Cultural Requirements 60
19b Political Requirements 61

20 Legal Requirements 62

20a Compliance Requirements 62
20b Standards Requirements 62

21 Requirements Acceptance Tests 64

21a Requirements – Test Correspondence Summary 64
21b Acceptance Test Descriptions 64

III Design 79

22 Design Goals 79

23 Current System Design 80

24 Proposed System Design 81

24a Initial System Analysis and Class Identification 81
24b Dynamic Modelling of Use-Cases 85
24c Proposed System Architecture 89
24d Initial Subsystem Decomposition 90

25 Additional Design Considerations 92

25a Hardware / Software Mapping 92
25b Persistent Data Management 93
25c Access Control and Security 112
25d Global Software Control 94
25e Boundary Conditions 96
25f User Interface 98

25g Application of Design Patterns 98

26 Final System Design 100

27 Object Design 102

27a Packages 104
27b Subsystem I 120
27c Subsystem II 120
27d etc. 120

IV Project Issues 106

28 Open Issues 106

29 Off-the-Shelf Solutions 107

29a Ready-Made Products 107
29b Reusable Components 108
29c Products That Can Be Copied 109

30 New Problems 110

30a Effects on the Current Environment 110
30b Effects on the Installed Systems 111
30c Potential User Problems 111
30d Limitations in the Anticipated Implementation Environment That May Inhibit the New Product 112
30e Follow-Up Problems 113

31 Migration to the New Product 114

31a Requirements for Migration to the New Product 114
31b Data That Has to Be Modified or Translated for the New System 115

32 Risks 115

33 Costs 117

34 Waiting Room 118

35 Ideas for Solutions 120

36 Project Retrospective 120

V Glossary 121

VI References / Bibliography 124

VII Index 125

List of Figures

Figure 2 - Sample Use Case Diagram from Bruegge & DuToit (modified)**Error! Bookmark not defined.**

Figure 3 - Sample Use Case Diagram from Robertson and Robertson 28

List of Tables

Table 2 - Requirements - Acceptance Tests Correspondence 64

I Project Description

1 Project Overview

The “WeAllocate” application is a resource allocation optimizer designed to harness the power of machine learning for the betterment of community services. It aims to predict the demand and supply patterns of various food items in local and regional food banks. By doing so, it offers optimized distribution strategies, ensuring maximum reach and efficiency in food allocation. The primary beneficiaries of this tool are local food banks and regional food bank organizations. With this solution, food banks can significantly enhance their distribution processes, ensuring that as many people as possible benefit from the available resources, all while minimizing wastage. This application will be partnered with IMAN: Food and Wellness center based out of Chicago, IL. To accomplish this, machine learning can significantly optimize the resource allocation process for food banks by analyzing historical data, current trends, and other relevant factors.

2 The Purpose of the Project

The primary motivation behind "WeAllocate" is to address the ever-present challenges faced by food banks: unpredictable demand, potential wastage of perishable items, and the need to maximize the impact of available resources. Food banks play a crucial role in supporting communities, especially during times of economic uncertainty or crises. However, they often operate with limited resources and rely heavily on donations, making it imperative to distribute food as efficiently and effectively as possible. By partnering with IMAN, the project seeks to pioneer a new age for resource management within charitable food distribution systems, making them more resilient, efficient, and impactful.

2a The User Business or Background of the Project Effort

Content:

IMAN (Inner-City Muslim Action Network) is a community organization that fosters health, wellness, and healing in the inner-city by organizing for social change, cultivating the arts, and operating a holistic health center. One of their key offerings is the Food and Wellness Center in Chicago, which provides essential food resources and wellness services to the community.

In the city of Chicago, where socio-economic disparities are evident, the Food and Wellness Center plays a pivotal role in ensuring that underserved communities have consistent access to nutritious food. However, like many community-driven initiatives, they face challenges in resource allocation, demand prediction, and efficient distribution.

Enter "WeAllocate", a cutting-edge application designed to revolutionize the way IMAN operates their food bank. WeAllocate seeks to partner with IMAN's Food and Wellness Center to optimize their resource allocation through predictive analytics, ensuring that every individual who approaches them leaves with their needs met and continues to be met.

Motivation:

The driving force of the WeAllocate project lies in the understanding that food banks often grapple with operational challenges. Predicting demand, minimizing wastage, and ensuring equitable distribution are complex tasks that require a blend of ground-level understanding and technological intervention. The WeAllocate application promises to bridge this gap, providing IMAN with a tool that not only streamlines their operations but also amplifies their community impact.

Considerations:

The challenges faced by IMAN's Food and Wellness Center are not unique but are indeed pressing. In a city where many depend on community services for their daily meals, any inefficiency or wastage has dire consequences. The question isn't just about whether there's a problem; it's about the magnitude of its impact on real lives. The need for a solution like WeAllocate is evident. By partnering with IMAN, WeAllocate isn't just providing a technological solution; it's bolstering a community lifeline, ensuring that the noble mission of IMAN - to serve, heal, and uplift - is realized to its fullest potential.

2b Goals of the Project

Content:

The primary goal of the WeAllocate project, from IMAN's perspective, is to enhance the efficiency and reach of their Food and Wellness Center. IMAN strives to ensure that every individual in the community has consistent access to nutritious food and wellness services. WeAllocate aims to empower IMAN with predictive insights to optimize resource allocation, thereby maximizing their impact and ensuring no individual in need is turned away.

Motivation:

As the project progresses, it's essential to keep the core goal at the forefront: improving the lives of the community members served by IMAN. While the development of the WeAllocate software is a significant aspect, it is merely a tool to achieve the larger objective of community betterment. The software's success will be measured not just by its technical prowess but by its tangible impact on IMAN's operations and, by extension, the lives of those it serves.

To ensure the project remains aligned with its goals, meeting sessions should be conducted with both the development team and IMAN's representatives. These sessions will serve as checkpoints, ensuring that the growth of “WeAllocate” aligns with IMAN's mission and objectives.

Examples:

- We want to ensure that every individual who is facing food insecurity is able to get the help they need.

- We aspire to minimize wastage at the Food and Wellness Center, ensuring that resources are utilized optimally and benefit the maximum number of community members

- We aim to streamline the food distribution so that we can continue operating for the best of our community

2c Measurement

Regular assessments will be carried out to ensure that the WeAllocate system aligns with IMAN's operational objectives and delivers tangible benefits. Adjustments will be made as necessary based on the insights gathered from these evaluations.

These will include but not be limited to:

Resource Utilization: Assess the effectiveness of WeAllocate in reducing food wastage at the center. The goal is to ensure optimal use of available resources, minimizing unnecessary losses.

Beneficiary Outreach: Examine the reach and efficiency of the center post-WeAllocate integration. The objective is to serve a broader segment of the community effectively.

Beneficiary Feedback: Periodic feedback will be collected from beneficiaries to gauge satisfaction levels. This will provide insights into the system's success in meeting individual needs.

Operational Streamlining: Evaluate the operational efficiency brought about by WeAllocate. The center should see expedited resource allocation processes, ensuring timely service to beneficiaries.

Return Beneficiaries: Track the frequency of returning beneficiaries as an indicator of the center's effectiveness in meeting community needs.

Inventory Management: Monitor inventory turnover to understand the agility of operations post-WeAllocate integration. The aim is to ensure a responsive and dynamic inventory system that aligns with demand patterns.

3 The Scope of the Work

Within the broader purpose of community support and wellness facilitated by IMAN's Food and Wellness Center of Chicago, the specific "work" addressed by the WeAllocate application is "optimizing food resource allocation and distribution." It narrows its focus to ensuring that food resources are effectively distributed to meet community demands. In this context, WeAllocate will function within the environment of food inventory management, demand prediction, and distribution at IMAN's Food and Wellness Center. We want our software to predict food demand based in data, demographics and other outside factors. Manage and optimize food inventory. Provide inventory information that also allows IMAN to manage distribution and update demands. Finally, we want to generate analytics for assessing impact, identifying trends and allowing the food bank to make more informed decisions.

3a The Current Situation

Content:

Currently, IMAN's Food and Wellness Center manages its food resource allocation through a combination of manual processes and some digital tools. Staff and volunteers assess inventory levels, monitor expiration dates, and gauge community demand based on historical trends, immediate past experiences, and direct feedback from beneficiaries. The distribution process involves manually categorizing food items, prioritizing them based on need and perishability, and then allocating them to beneficiaries as they arrive or through scheduled distributions. There is some level of digitization, such as basic inventory tracking spreadsheets or databases, the majority of decision-making is reliant on human judgment. This involves a significant amount of time, effort, and relies heavily on the expertise of the individuals involved. Moreover, without sophisticated predictive tools, the center might face challenges in anticipating sudden spikes in demand or efficiently handling excess supply. Furthermore, it is also susceptible to the inevitable human error

Motivation:

Understanding this current setup is important for several reasons. Firstly, it provides a baseline against which the improvements brought about by WeAllocate can be measured. Secondly, it offers insights into potential challenges that might be encountered during the implementation of the new system. Users accustomed to manual methods might require training or might have concerns about the new processes. Recognizing the intricacies of the existing system ensures that the transition to WeAllocate is smooth, addressing both the operational challenges and the concerns of the staff and volunteers who will use it.

3b The Context of the Work

Content:

The WeAllocate application is designed to seamlessly integrate into IMAN's Food and Wellness Center's existing workflow. Its primary function is optimizing food resource allocation and distribution. The work context diagram would typically showcase the WeAllocate system at the center, with various external entities it interacts with.

External Entities:

- **Inventory Management System:** Existing databases or systems that track food items, their quantities, and expiration dates.
- **Beneficiary Feedback Portal:** A system or method where beneficiaries provide feedback, which can help in refining predictions and allocations.
- **Donor Systems:** Platforms or methods through which donations (both monetary and in-kind) are received.
- **Community Events Calendar:** Local events can influence demand. Integration with such a calendar can provide predictive insights.

- **Staff and Volunteer Input Portal:** A platform for staff and volunteers to input observations, anomalies, or urgent requirements.

Motivation:

By defining these boundaries and interactions, we ensure that WeAllocate is not developed in isolation but is tailored to fit its operational environment. The interactions with adjacent systems ensure that WeAllocate has all the necessary data for its algorithms and predictions, making it a holistic solution.

Considerations:

- **Beneficiaries**: The primary end-users of the resources allocated by WeAllocate. Their needs can vary based on individual circumstances, seasonality, and community events. Understanding their consumption patterns, preferences, and feedback is crucial for optimizing resource allocation.

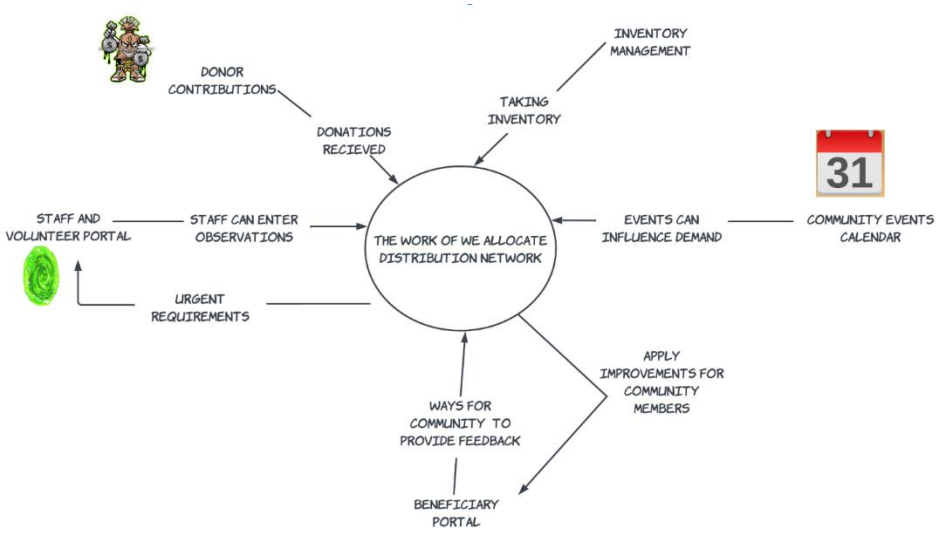
- **Staff and Volunteers**: directly interacting with beneficiaries and handling the day-to-day operations of the Food and Wellness Center. Their insights, observations, and feedback are needed for refining the WeAllocate system. They can also identify anomalies or urgent requirements that might not be immediately evident from data alone.

- **Donors**: Entities or individuals donating food items or funds to IMAN's Food and Wellness Center. They might require reports or insights on how their donations are utilized, which WeAllocate can potentially provide. Understanding donation patterns can also aid in predicting future resources.

- **Community Events**: Local events, holidays, or community gatherings can influence demand. It's essential to factor in these events when predicting resource requirements.

- **Inventory Management**: IMAN's Food and Wellness Center might be using existing systems for inventory management, feedback collection, or donor management. Ensuring seamless integration with these systems is crucial for the holistic functioning of WeAllocate.

WeAllocate can be developed and refined to best suit and meet the objectives of IMAN's Food and Wellness Center.


DONOR CONTRIBUTIONS
INVENTORY MANAGEMENT
TAKING INVENTORY
DONATIONS RECIEVED
STAFF AND VOLUNTEER PORTAL
STAFF CAN ENTER OBSERVATIONS
THE WORK OF WE ALLOCATE DISTRIBUTION NETWORK
EVENTS CAN INFLUENCE DEMAND
COMMUNITY EVENTS CALENDAR
URGENT REQUIREMENTS
APPLY IMPROVEMENTS FOR COMMUNITY MEMBERS
WAYS FOR COMMUNITY TO PROVIDE FEEDBACK
BENEFICIARY PORTAL

3c Work Partitioning

Business Event List

Event Name	Input and Output	Summary
1. Food donations from doners	Doner Contribution (IN)	Receive food donation from donators
2. Track inventory	Inventory Management (in)	Uses data from achieved form the inventory to feed the machine learning algorithm
3. Community Calanders	Community events (in)	These events can influence the demand on which can be used as one of the aspects for the ML algorithm
4. Volunteers & staff observation	Feedback data (in)	Volunteers & staff at station can submit feedback to ML algorithm on whether or not the resources were enough at the even
	Urgent Requirements (out)	Any immediate changes the algorithm picks up will have to be enacted by staff and volunteers
5. Beneficiary feedback	Beneficiary portal (in)	People who took the donation can send in feedback on portions that they receive are enough and take in their input what other location IMAN can hold their donation centers.

3d Competing Products

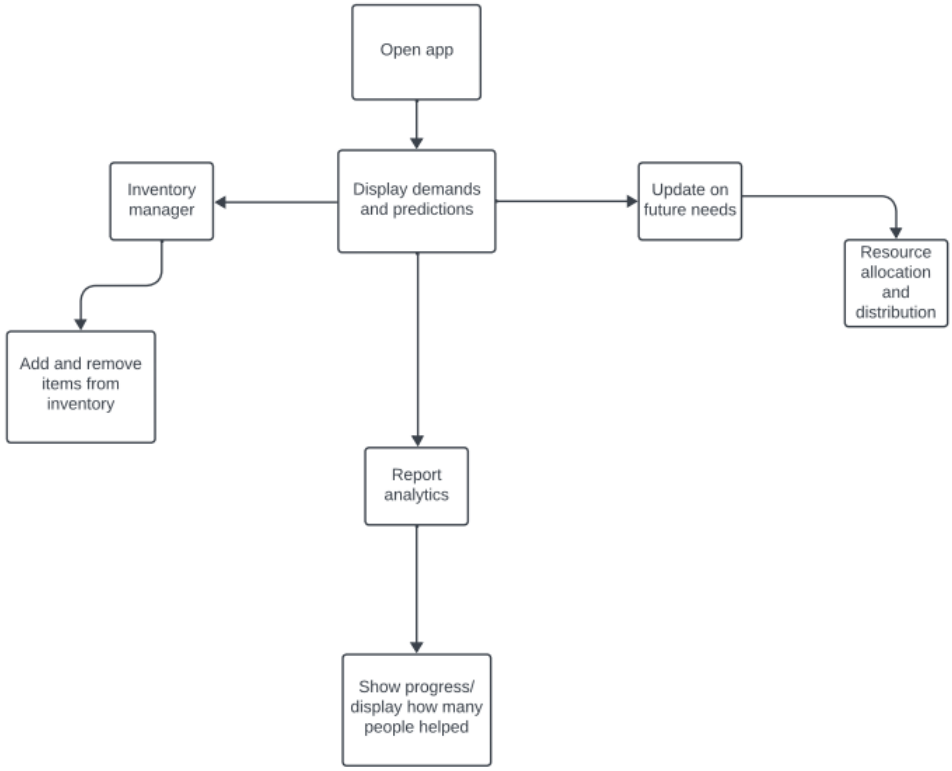
Several inventory management and predictive analytics tools are available that could serve organizations like IMAN's Food and Wellness Center. However, these generic solutions often fall short in addressing the unique demands of food banks. They may lack features for predicting demand based on community changes, might not integrate efficiently with IMAN's existing workflow, and could be a templated solution meeting needs but not exceeding them. The tailored approach of WeAllocate is offered as a specific solution designed to fill and exceed needs with features like beneficiary feedback, community event tracking, and precise integration capabilities.

Our solution is not just another tent to be pitched but instead a foundational pillar for IMAN.

4 The Scope of the Product

This application is a machine learning powered tool which utilizes machine learning algorithms and predicts supply and demand for various sources. This product consists of an optimized distribution strategy that will recommend the best distribution strategy and aim for maximum reach and efficiency in food allocation.

4a Scenario Diagram(s)



4b Product Scenario List

- *Special food in demand*
- *A lot of food was taken from the food bank and the inventory dropped*
- *Food is expiring*
- *A certain spot requires a lot of an item, and the inventory needs to be updated*

4c Individual Product Scenarios

1. In the fall, there are certain food more popular than others. Trail mix is a popular item for people to go pick up and the learning algorithm picked it up and it can now update the IMAN food bank accurately to predict when they will be out of that food based on the trend. Because we can accurately predict when the trail mix will be popular, the IMAN food bank can accurately predict when they will need more.

2. At the IMAN food bank, People took a lot of food from the food bank and their stock is running low. Our program's interface can identify this trend and alert the people at the food bank to restock because the trend calculates that a lot of food will be taken within the next couple of days.

3. When food is inputted into the system, the user at the food bank has to give it an expiration date. This means that it will need to be given out quickly or else it will expire. In the case that it expires, the food bank must remove it from the stock.

4. A certain organization that makes meals during the thanksgiving period required a lot of frozen and canned food. Due to the high demand of these two predicts, the software's inventory is updated and can give a more accurate prediction of how long the supply will last.

5 Stakeholders

5a The Client

Content:

IMAN: Food and Wellness Center of Chicago

IMAN, as the primary client, invests in the WeAllocate system with the aspiration to enhance their food resource allocation efficiency. Their intent is to serve the community better, reduce wastage, and ensure that their beneficiaries receive optimal support.

Though the WeAllocate system is technically for IMAN's internal use, its benefits directly impact the larger community. Therefore, while IMAN is the client, their feedback and satisfaction with the system will also be influenced by the community's experiences.

5b The Customer

The Customer will be the IMAM food bank. This app will allow them to be more efficient and run their operation better. We are only targeting this specific food bank in Chicago but there are many other around the U.S that could benefit from a software like this.

5c Hands-On Users of the Product

- **Food Bank Managers**
    - **User role:** Oversee the operations, inventory management, and distribution processes at the center..
    - **Other user characteristics:** Decision-making skills, community engagement experience, leadership abilities.
- **Volunteers**
    - **Role:** Assist with inventory checks, food distribution, and beneficiary interactions.
    - **Other user characteristics:** Community empathy, collaborative spirit, adaptability to varying tasks.
- **Beneficiaries**
    - **Role:** Receive food allocations, provide feedback on preferences and needs.
    - **Other user characteristics:** Diverse backgrounds, varying levels of technological literacy, different nutritional needs.

Understanding the diverse user base is essential for tailoring WeAllocate's features to their needs. Ensuring usability and accessibility for all users, from managers to beneficiaries, is vital for the system's overall success and efficacy.

5d Maintenance Users and Service Technicians

For the WeAllocate system, there are dedicated maintenance users which comprise System Administrators, who are responsible for the system's overall health, backend configurations, user access management, and timely updates. Database Managers are crucial because they will be ensuring data integrity, conducting regular backups, managing data updates, and overseeing the entirety of information stored within WeAllocate. Additionally, Technical Support Staff form an integral part, assisting in troubleshooting, addressing technical issues, and providing support to primary users like Food Bank Managers and Volunteers. Their interactions and responsibilities, while different from the primary users, are foundational to the system's robustness, security, and efficiency.

5e Other Stakeholders

Sponsor: Includes higher value donor

Role: Financial backing and guiding the project's direction.

Influence: High

Legal Experts: IMAN's Legal Counsel

Role: Ensure compliance with relevant laws and regulations.

Influence: High

5f User Participation

Food Bank Managers: Their deep understanding of day-to-day operations and challenges will be vital.

Contribution expected: Providing business knowledge, offering feedback on feature prototypes, and detailing usability requirements.

Volunteers: Their on-ground experiences will provide invaluable insights into real-world scenarios and challenges.

Contribution expected: Sharing experiences, testing initial user interfaces, and offering feedback on system usability.

System Administrators and Technical Support Staff: Their understanding of the technical infrastructure and potential challenges will help in refining the system's backend.

Contribution expected: Offering technical insights, assisting in system integration planning, and validating technical requirements.

5g Priorities Assigned to Users

- **Key Users:**
    - **Food Bank Managers:** They are at the forefront of operations and make critical decisions based on the insights and recommendations from WeAllocate. Their requirements, feedback, and satisfaction with the system are paramount for the tool's success and adoption.
    - **System Administrators:** Ensuring the system's smooth operation, their role is crucial to the tool's overall functionality, security, and integration with other systems.
- **Secondary Users:**
    - **Volunteers:** While they play a significant role in on-ground operations and provide valuable feedback, the system primarily serves to assist the decision-making

process led by the Food Bank Managers. However, their ease of use and feedback still carry substantial weight.

- **Technical Support Staff:** Their interactions with the system will be more on the troubleshooting and support front, making their priorities secondary to those actively using the system for operations.

- **Unimportant Users:**

  - **Casual Observers:** These might include visitors, potential donors, or other external entities who might get a demonstration of the system but won't interact with it daily. Their requirements or feedback, while appreciated, won't be the driving factor in design decisions.

6 Mandated Constraints

6a Solution Constraints

- **Description:** The WeAllocate system will be a cloud-based application accessible via web browsers.

  Rationale: IMAN does not have high power hardware so a cloud-based system ensures universal access and centralized data storage, without the need for physical installations.

  Fit Criterion: The application should be accessible from any modern web browser (like Chrome, Firefox, Safari) without the need for additional plugins or software installations.

- **Description:** WeAllocate will feature a user-friendly dashboard with interactive data visualization capabilities.

  Rationale: The diverse user base of IMAN, ranging from administrators to volunteers, requires an intuitive interface to quickly understand and act upon allocation data.

  Fit Criterion: Users should be able to understand and navigate the dashboard with minimal training, and the system should support graphical representations like charts and graphs.

- **Description:** WeAllocate will incorporate role-based access controls.

  Rationale: Different stakeholders like administrators, volunteers, and managers have varied access needs. Implementing role-based access ensures that each user only accesses the information and functionalities relevant to their role.

  Fit Criterion: The system should allow the creation of distinct user roles, each with customizable permissions. An administrator should be able to assign and modify these roles.

- **Description:** WeAllocate will be designed with scalability in mind.

  Rationale: As IMAN grows and the number of resources and allocations increases, the system should be able to handle the increased load without performance degradation.

  Fit Criterion: The application should demonstrate consistent performance even with a 50% increase in simultaneous users or data volume.

6b Implementation Environment of the Current System

WeAllocate is designed to operate in a digital environment, predominantly on servers and workstations running whichever system IMAN has setup. It will also have mobile interfaces optimized for both Android and iOS platforms for team members to be able to use. These can be webapps to make transition smoother.

6c Partner or Collaborative Applications

Inventory System: WeAllocate will connect to this system to know about the food stock and its details.

Events Calendar: WeAllocate will check this calendar to know about big events that might increase the number of visitors.

Feedback Tools: If IMAN uses tools like Google Forms to get feedback, WeAllocate should be able to read that feedback.

Donor Systems: WeAllocate will connect to these systems to know about incoming donations.

Microsoft Excel: WeAllocate should be able to use Excel files for easy data handling.

6d Off-the-Shelf Software

Database Management Systems: Tools like PostgreSQL or MySQL will be essential for managing and storing the vast amounts of data WeAllocate will handle.

Cloud Services: Amazon Web Services (AWS) or Google Cloud Platform (GCP) are required for hosting the application and ensuring seamless and scalable performance.

Data Analytics Tools: Integrations with platforms like Tableau or Power BI will be beneficial for generating insights and reports from the collected data.

6e Anticipated Workplace Environment

Busy and Crowded: Food banks often experience a high influx of beneficiaries, volunteers, and staff, leading to a bustling environment.

Varied Tech Proficiency: The environment will have a mix of users, from tech-savvy staff to volunteers who might not be as familiar with digital tools.

Multiple Access Points: The tool might be accessed from both back-office setups (computers) and on-the-floor mobile devices or tablets.

Noise Levels: Given the flow of beneficiaries and operational activities, noise levels can be relatively high, making auditory notifications less effective.

Connectivity Issues: Some areas within the facility might have weaker Wi-Fi signals or connectivity challenges.

6f Schedule Constraints

Given the urgency to enhance food distribution efficiency, WeAllocate aims to be operational before the winter season, when demand spikes. Specific milestones will be set to ensure timely delivery.

- **Pre-Holiday Season Release:** WeAllocate should be fully functional to cater to the increased demand and activities during the holiday season.
- **Beta Testing:** A preliminary version should be available for testing allowing a time window for feedback and improvements before the final release.
- **Training Sessions:** Designated training sessions for staff and volunteers are scheduled. The basic functionalities of WeAllocate need to be ready by this time for effective training.

6g Budget Constraints

Funding for WeAllocate is limited to the budget allocated by IMAN and potential grants. Development choices will be made to ensure the best possible solution within this budget. Fundraising may be needed to meet further needs as well as Maintenace. This includes software development, testing, user training, and support.

7 Naming Conventions and Definitions

7a Definitions of Key Terms

Allocation Algorithm: The specific machine learning method used by WeAllocate to distribute resources within the IMAN food bank.

Dashboard: The user interface of WeAllocate that presents data, allocation results, and system functionalities in an organized manner.

Database: A structured collection of data in WeAllocate, storing information about resources, allocations, users.

IMAN: Abbreviation for "Inner-City Muslim Action Network" The organization for which WeAllocate is being developed to manage and optimize the allocation of its resources.

Resource: Any tangible or intangible item managed by IMAN, such as food items or volunteer hours, that requires allocation via WeAllocate.

System Admin: A user role within WeAllocate with elevated privileges, responsible for system setup, user management, and overall system maintenance.

User Profile: A digital record within WeAllocate that contains data about a specific user, such as user ID, name, contact information, role, and preferences.

WeAllocate: The proposed software system designed to automate and optimize the allocation of resources within the IMAN food bank, ensuring efficient distribution and waste reduction.

7b UML and Other Notation Used in This Document

In the context diagram, there are some images to go along with the entities of this system.

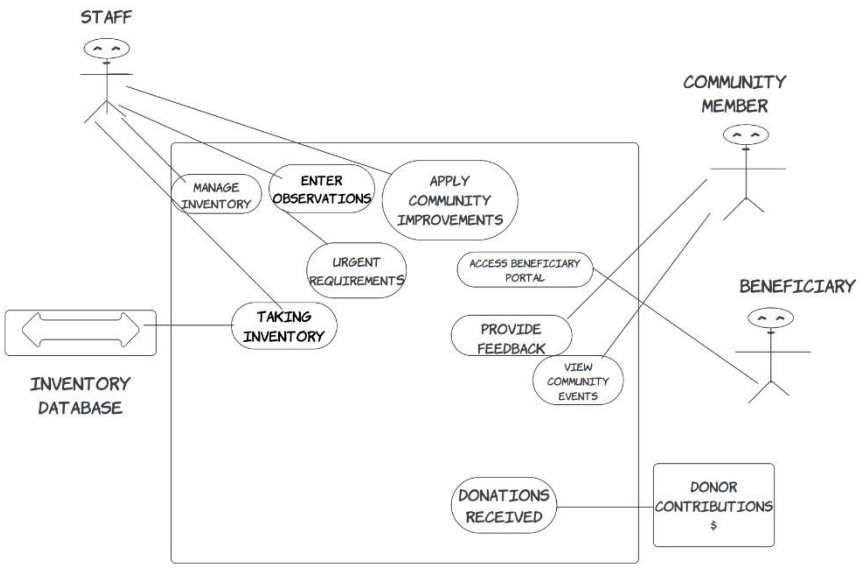
 represents Donor contributions

 represents the staff and volunteer portal

 represents inventory management

 represents the community calendar

USE CASE DIAGRAM



[4]

7c Data Dictionary for Any Included Models

Since there are many elements, there will be numerous data properties to keep track of, that have unique identifiers.

Inventory:

- Item
- Name
- Expiration date
- Quantity
- Category
- Average Demand
- Quantity Available

8 Relevant Facts and Assumptions

8a Facts

Beneficiary Demographics: IMAN's Food and Wellness Center primarily serves the inner-city communities of Chicago, addressing the needs of diverse populations including marginalized groups, low-income families, and individuals from various ethnic backgrounds.

Center's Operations: The Food and Wellness Center operates six days a week, providing both fresh produce and non-perishable food items to beneficiaries. It also offers wellness services and community outreach programs.

Community Engagement: IMAN's Food and Wellness Center, beyond just food distribution, engages with the community through health workshops, nutritional education sessions, and community-building events.

Collaborations: The center collaborates with local farmers, businesses, and other organizations for food sourcing and community programs, ensuring a sustainable supply chain and broader community engagement.

Volunteer Participation: The center's operations are significantly supported by community volunteers. Monthly, they witness active participation from local community members, students, and other individuals who offer their time and expertise.

Existing Infrastructure: IMAN's Food and Wellness Center has a physical infrastructure comprising storage facilities, refrigeration units, and community spaces where beneficiaries and volunteers interact.

8b Assumptions

Technological Proficiency: It's assumed that the primary users of WeAllocate, including volunteers and staff at IMAN's Food and Wellness Center, have basic computer literacy and can navigate through the application with minimal training.

Infrastructure: The center has stable internet connectivity and devices (like computers or tablets) that can run WeAllocate efficiently.

Data Availability: Existing data related to food inventory, beneficiaries, and other relevant metrics are available in a digital format, ready for integration into WeAllocate.

Volunteer Availability: Volunteers will be available for training sessions on how to use the WeAllocate system and will commit to using the system consistently.

Ongoing Support: IMAN has a dedicated team or individual who will act as a point of contact for any software-related issues, feedback, or updates.

Integration with External Systems: WeAllocate can integrate with any existing systems or software that the center might be using, like donor management systems or volunteer coordination tools.

Resource Commitment: The center will allocate necessary resources, both in terms of time and finances, for the successful deployment and adoption of WeAllocate.

Community Engagement: The community will positively receive the introduction of a digital system, understanding it as a move towards efficiency and better service.

II Requirements

9 Product Use Cases

Use Case: Manage Inventory

- **Pre-conditions:** There are items or resources to manage.
- **Post-conditions:** Inventory is updated and organized.
- **Initiated by:** Staff
- **Triggering Event:** Need to oversee and organize available resources or items.
- **Sequence of Events:**
    - Staff will be able to see the inventory in the system.
        - System displays current inventory status.
    - Staff updates inventory.
        - System saves the changes made to the inventory.
- **Alternatives:** Staff adds new items or resources to the inventory.
- **Exceptions:** System fails to update the inventory.

Use Case: View Community Events

- **Pre-conditions:** Events are scheduled.
- **Post-conditions:** Community members are informed about upcoming events.
- **Initiated by:** Community Member
- **Triggering Event:** Curiosity to stay informed about upcoming events.
- **Sequence of Events:**
    - Community members will access the events calendar.
        - System displays upcoming events.
    - Community members view event details.
        - App provides detailed information about the selected events.
- **Alternatives:** Members can set a reminder for the event.
- **Exceptions:** App fails to display events.

Use Case: Access Beneficiary Portal

- **Pre-conditions**: Beneficiary has the necessary credentials.

- **Post-conditions**: Beneficiary accesses relevant information.
- **Initiated by:** Beneficiary
- Triggering Event: Beneficiary would like to provide feedback.
- Sequence of Events:
    - Beneficiary logs into the portal.
    - Beneficiary provides feedback.
        - System records the contribution details.
- **Alternatives:** Beneficiary requests assistance or support through the portal.
- **Exceptions:**
    - Invalid login credential.
    - System fails to record the review.

9a Use Case Diagrams

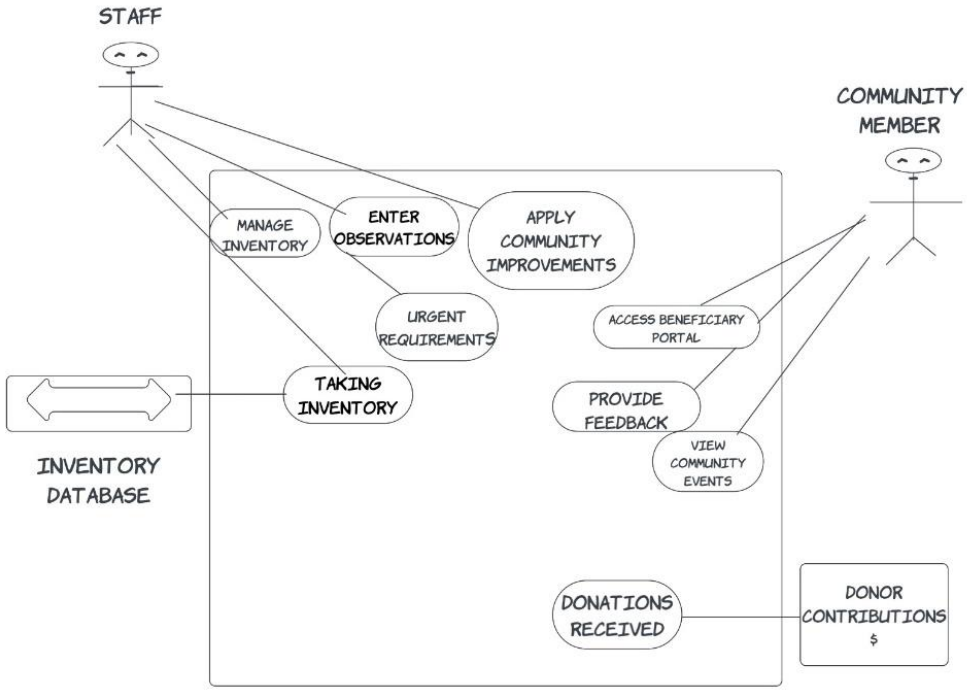


Figure 1 - Sample Use Case Diagram from Robertson and Robertson

9b Product Use Case List

9c Individual Product Use Cases

Use case ID: UC--001 Name: Manage Inventory pre-conditions: There are items or resources to manage. post-conditions: Inventory is updated and organized. Initiated by: Staff Triggering Event: Need to oversee and organize available resources or items. Additional Actors: N/A
Sequence of Events: 1. Staff will be able to see the inventory in the system. • System displays current inventory status 2. Staff updates inventory • System saves the changes made to the inventory
Alternatives: Staff adds new items or resources to the inventory Exceptions: System fails to update the inventory

Use case ID: UC--02 Name: View community events pre-conditions: Events are scheduled post-conditions: Community member is informed about upcoming events Initiated by: Community Member Triggering Event: Curiosity to stay informed about upcoming events Additional Actors: N/A
Sequence of Events: 3. Community member will access the events calendar • System displays upcoming events 4. Community member views event details • App provides detailed information about the selected events
Alternatives: Member can set a reminder for the evebt Exceptions: App fails to display events

Use case ID: UC--03 Name: Contribute

pre-conditions: Donor has monetary or in-kind donation to donate

post-conditions: The donation is received and recorded

Initiated by: Donor

Triggering Event: Donor decides to be a contribution

Additional Actors: Staff/Community members (Anyone can donate)

Sequence of Events:

5. Donor decides to make a contribution
    - System prompts donor for details of the contribution
6. Donor provides details of the contribution
    - System records the contribution details
7. Staff

Alternatives: Donor decides to set up a recurring contribution

Exceptions:

- Contribution details are incorrect or incomplete
- System fails to record the contribution.

Use case ID: UC--04 Name: Access Beneficiary Portal pre-conditions: Beneficiary has the necessary credentials. post-conditions: Beneficiary accesses relevant information. Initiated by: Beneficiary Triggering Event: Beneficiary would like to provide feedback Additional Actors: N/A
Sequence of Events: 8. Beneficiary logs into the portal 9. Beneficiary provides feedback • System records the contribution details 10. Staff
Alternatives: Beneficiary requests assistance or support through the portal Exceptions: • Invalid login credential • System fails to record the review

Use case ID: UC--005 Name: Enter Observations

pre-conditions: Relevant information needed to note down

post-conditions: Observations are recorded

Initiated by: Staff and volunteers

Triggering Event: Needed to record their insights and observations

Additional Actors: N/A

Sequence of Events:

11. Staff accesses the observation section
    - The system provided a section for observations
12. Staff members reports any observation or notes
    - System records the feedback
    - System Emphasizes urgent requirements
13. Staff notes down any urgent

Alternatives: Staff attaches supporting documents or images with observations

Exceptions:

- System fails to save observations

Use case ID: UC--006 Name: Taking/Allocating Inventory pre-conditions: Resources are available to be distributed post-conditions: Resources have been allocated and distributed Initiated by: Staff Triggering Event: Need to distribute resources Additional Actors: N/A
Sequence of Events: 14. Staff identifies resources that need to be distributed • The system lists available resources 15. Staff allocates resources or items • The system lists available resources
Alternatives: Allocation based on priority Exceptions: • System fails to save inventory changes after resources have been allocated

10 Functional Requirements

ID# FR-004 - Predictive Resource Allocation

- **Description:** The system shall utilize machine learning algorithms to analyze historical data and predict future resource demands, optimizing the allocation of food and wellness resources.
- **Rationale:** To anticipate the needs of the community and ensure that resources are distributed efficiently, minimizing waste and ensuring that no individual in need is turned away.
- **Fit Criterion:** The system's predictions should have an accuracy rate of at least 90%, and the resource allocation based on these predictions should result in less than 5% wastage.

- **Acceptance Tests: ML test**

ID# FR-010 - Feedback and Support

- **Description:** The Beneficiary Portal shall include a section for beneficiaries to provide feedback on received resources, events attended, or any other aspect of IMAN's initiatives. Additionally, a support feature should be available for beneficiaries to raise concerns or seek assistance.
- **Rationale:** To ensure that beneficiaries have a voice in the system and can communicate their experiences, suggestions, or concerns.
- **Fit Criterion:** Beneficiaries should be able to submit feedback through a user-friendly interface. The support feature should provide timely responses to beneficiary queries or concerns.
- **Acceptance Tests: Feedback**
  - Test the feedback submission process.
  - Monitor the responsiveness and effectiveness of the support feature.

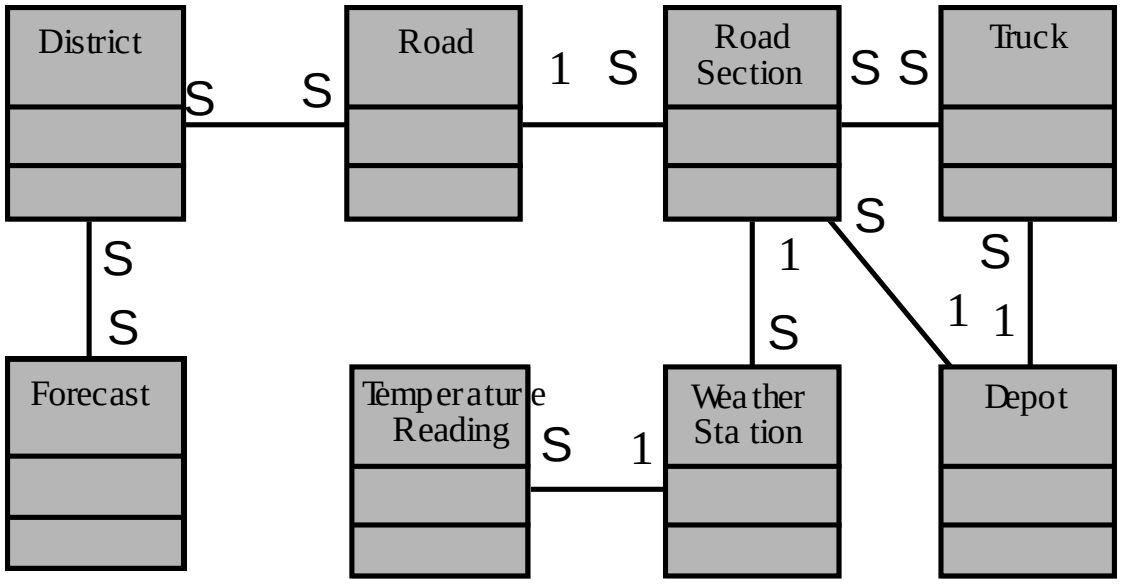
ID# FR-011 - Event Listing and Details

- **Description:** The Beneficiary Portal shall display a comprehensive list of upcoming community events. Each event listing should provide detailed information, including date, time, location, description, and any associated resources or activities.
- **Rationale:** To keep beneficiaries and community members informed about upcoming events and activities organized by IMAN.
- **Fit Criterion:**
- Events should be listed in chronological order.
- Clicking on an event should provide detailed information about that event.
- **Acceptance Tests: Events**

11 Data Requirements

The WeAllocate system will handle various data entities related to its operations. Some of the primary entities include:

- **Inventory:** This encompasses data about items, their names, expiration dates, quantities, categories, average demand, and available quantity.
- **Beneficiary Demographics:** Information about the beneficiaries served by IMAN's Food and Wellness Center, primarily focusing on the inner-city communities of Chicago.



ID# DR-001 - Inventory Data Management

- **Description:** The system shall maintain a comprehensive database of inventory items, tracking their names, expiration dates, quantities, categories, average demand, and available quantity.
- **Rationale:** To ensure efficient resource allocation and minimize wastage by keeping track of available resources and their details.
- **Fit Criterion:** The system should allow for real-time updates to inventory data and reflect changes immediately. Any discrepancies in inventory data should be flagged for review.
- **Acceptance Tests: IDM**

ID# DR-002 - Beneficiary Demographic Data

- **Description:** The system shall maintain data related to the demographics of beneficiaries, capturing essential details that can influence resource allocation decisions.
- **Rationale:** To tailor the resource allocation process based on the specific needs and preferences of the beneficiary demographics.
- **Fit Criterion:** The system should allow for the addition and updating of beneficiary demographic data, ensuring that data remains current and relevant.
- **Acceptance Tests: Demographic**

ID# DR-007 - User Feedback and Surveys

- **Description:** The system shall have mechanisms to capture user feedback, suggestions, and responses to surveys or questionnaires, storing this data for analysis and action.
- **Rationale:** To gather user insights, understand their needs and preferences, and drive improvements based on their feedback.

- **Fit Criterion:** The feedback and survey mechanisms should be user-friendly, and the collected data should be easily accessible for analysis.
- **Acceptance Tests: Feedback**

ID# DR-012 - Monetary and In-Kind Donation Tracking

- **Description:** The system shall track and manage details of both monetary and in-kind donations, ensuring that all specifics like amount/type, date, method, and associated acknowledgment are captured and stored.
- **Rationale:** To maintain a transparent and accurate record of all donations received, facilitating financial reporting, inventory management, and donor acknowledgment.
- **Fit Criterion:** The system should provide real-time updates on donations received and allow for easy retrieval and reporting of donation data.
- **Acceptance Tests: Donation track**

ID# DR-013 - Recurring Donation Management

- **Description:** The system shall manage and track recurring donations, capturing details like frequency, amount, next due date, and any associated payment methods.
- **Rationale:** To ensure that periodic donations are acknowledged, tracked, and managed effectively, fostering consistent support from recurring donors.
- **Fit Criterion:** The system should send reminders or notifications related to upcoming recurring donations and provide tools for donors to modify or cancel their recurring contributions.
- **Acceptance Tests: Donation management**

11a Performance Requirements

ID# PR-001 - User Interface Response Time

- **Description:** Any interface between a user and the WeAllocate system shall have a maximum response time.
- **Rationale:** To provide a smooth user experience and ensure that users can navigate and interact with the system without unnecessary delays.
- **Fit Criterion:** The product shall respond in less than 1 second for 90 percent of the interrogations. No response shall take longer than 2.5 seconds.
- **Acceptance Tests:**

ID# PR-006 - Inventory Load Management

- **Description:** The product shall cater for varying inventory loads, with the ability to handle large numbers of items, categories, and related data.
- **Rationale:** To ensure that the system remains responsive and operational even during high inventory activity periods.

- **Fit Criterion:** The system should be able to handle a specified number of inventory items and related data without degradation in performance.
- **Acceptance Tests: Inventory**

ID# PR-008 - Portal Data Precision

- **Description:** All data entries and feedback submissions in the portal, including comments, ratings, and details, shall be accurate and up-to-date.
- **Rationale:** To ensure accurate data tracking and feedback collection within the portal.
- **Fit Criterion:** Portal entries and feedback should consistently adhere to the provided data and reflect real-time changes.
- **Acceptance Tests: Portal Feedback**

ID# PR-009 - Portal User Load Management

- **Description:** The portal shall cater for varying user loads, with the ability to handle peak loads during specific events or timeframes.
- **Rationale:** To ensure that the portal remains responsive and operational even during high user activity periods.
- **Fit Criterion:** The portal should be able to handle a specified number of simultaneous users without degradation in performance.
- **Acceptance Tests: Load manage**

11b Precision or Accuracy Requirements

ID# PAR-001 - Monetary Amount Precision

- **Description:** All monetary amounts, including donations and financial transactions within the WeAllocate system, shall be accurate to two decimal places.
- **Rationale:** To ensure accurate financial tracking and reporting within the system.
- **Fit Criterion:** Monetary entries and transactions should consistently adhere to the two-decimal-place standard.
- **Acceptance Tests: Donation Precision**

ID# PAR-002 - Beneficiary Data Accuracy

- **Description:** All beneficiary data, including demographic details, feedback, and related entries, shall be captured and stored with utmost accuracy.
- **Rationale:** To ensure that beneficiary data is reliable and can be used for effective decision-making and service provision.
- **Fit Criterion:** Beneficiary data entries should consistently reflect the actual data provided by the beneficiaries without discrepancies.
- **Acceptance Tests: Data Accuracy**

ID# PAR-003 - Inventory Data Precision

- **Description:** Inventory data, including item names, expiration dates, quantities, and other related details, shall be captured and stored with high precision.
- **Rationale:** To ensure effective inventory management and avoid discrepancies that could affect service provision.
- **Fit Criterion:** Inventory data entries should consistently match the actual inventory items without discrepancies.
- **Acceptance Tests: Inventory Precision**

11cCapacity Requirements

Content: This section specifies the volumes that the WeAllocate system must be able to deal with, including user loads, data storage, inventory items, and related data.

Motivation: To ensure that the WeAllocate system is capable of processing the expected volumes, especially during peak usage times or critical operations.

ID# CR-001 - User Load Capacity

- **Description:** The WeAllocate system shall cater for varying user loads, ensuring smooth access and operations for both staff and beneficiaries.
- **Rationale:** To provide a seamless experience for all users, especially during peak times or events.
- **Fit Criterion:** The system should be able to handle a specified number of simultaneous users without degradation in performance.
- **Acceptance Tests: Capacity**

ID# CR-002 - Inventory Data Capacity

- **Description:** The WeAllocate system shall be able to manage and store detailed inventory data, including item names, expiration dates, quantities, categories, average demand, and quantity available.
- **Rationale:** To ensure accurate and efficient inventory management and tracking.
- **Fit Criterion:** The system should be able to handle a specified number of inventory items and related data without degradation in performance.
- **Acceptance Tests: Inventory Capacity**

12 Dependability Requirements

12aReliability Requirements

ID# RR-001 - System Failures

- **Description:** The WeAllocate system shall not fail more than once per month.
- **Rationale:** To provide a consistent and reliable experience for users and to ensure the smooth operation of all system functionalities.
- **Fit Criterion:** Monitor system logs and user reports to track any system failures. The system should not have more than one failure in a 30-day period.
- **Acceptance Tests: Fail**

ID# RR-002 - Data Integrity on Failure

- **Description:** No data shall be lost or damaged in the event of a system failure. The system should have mechanisms in place to recover and restore data to its state before the failure.
- **Rationale:** To ensure the integrity and consistency of data, even in the event of unexpected system disruptions.
- **Fit Criterion:** In the event of a system failure, data should be recoverable to its state before the failure, with no data loss or corruption.
- **Acceptance Tests: Data in System Failure**

ID# RR-003 - Fail-Safe Mechanisms

- **Description:** The WeAllocate system shall implement fail-safe mechanisms to ensure that, in the event of a failure, the system fails safely without causing harm or further disruptions.
- **Rationale:** To prioritize user safety and data integrity, even during unexpected system issues.
- **Fit Criterion:** The system should have mechanisms in place to detect potential failures and shut down or switch to a safe mode without causing further issues.
- **Acceptance Tests: Fail safe**

12b Availability Requirements

ID# AR-001 - System Availability

- **Description:** The WeAllocate system shall be available for use 24 hours per day, 365 days per year.
- **Rationale:** To provide continuous access to the system for both staff and beneficiaries, ensuring that services are available whenever needed.
- **Fit Criterion:** Monitor system uptime and availability. The system should be available for use at all times, with minimal exceptions.
- **Acceptance Tests: System Test**

ID# AR-002 - Maintenance Downtime

- **Description:** The WeAllocate system may have scheduled maintenance downtime during non-peak hours, typically between 2:00 a.m. and 4:00 a.m. on Sundays.
- **Rationale:** Scheduled maintenance is necessary to ensure the health and performance of the system. Downtime during non-peak hours minimizes disruption to users.
- **Fit Criterion:** Monitor scheduled maintenance windows and user impact. Scheduled maintenance should occur during non-peak hours.
- **Acceptance Tests: Downtime**

ID# AR-003 - Uptime Percentage

- **Description:** The WeAllocate system shall achieve a minimum uptime percentage of 99% over a rolling 30-day period.
- **Rationale:** To quantify the system's availability and provide a measurable target for uptime.
- **Fit Criterion:** Monitor system uptime over a rolling 30-day period and calculate the percentage of uptime.
- **Acceptance Tests: UP**

12cRobustness or Fault-Tolerance Requirements

ID# RFT-001 – OfflineMode

- **Description:** WeAllocate must be able to operate in offline mode when network connectivity is unavailable.
- **Rationale:** This requirement ensures that food banks can continue to use WeAllocate even if their internet connection goes down.
- **Fit Criterion:** WeAllocate must be able to operate in offline mode for at least 72 hours.
- **Acceptance Test:** OFFLINE

ID# RFT-002 – Graceful Degradation

- **Description:** WeAllocate must be able to continue to function with reduced functionality in the event of a resource shortage, such as a disk drive failure or a power outage.
- **Rationale:** This requirement ensures that food banks can continue to use WeAllocate even if some of its resources are unavailable.
- **Fit Criterion:** WeAllocate must be able to continue to operate with reduced functionality for at least 24 hours in the event of a resource shortage.
- **Acceptance Test:** GD

ID# RFT-003 – Disaster Recovery

- **Description:** WeAllocate must be able to be recovered from a disaster within 24 hours.
- **Rationale:** This requirement ensures that food banks can quickly resume using WeAllocate after a disaster.
- **Fit Criterion:** WeAllocate must be able to be recovered from a disaster using a variety of methods, including backups, snapshots, and replication.
- **Acceptance Test:** Recovery

12d Safety-Critical Requirements

ID# SCR-001 – Food Safety

- **Description:** WeAllocate must not generate distribution strategies that could compromise food safety.

- **Rationale:** This requirement ensures that the food distributed by food banks using WeAllocate is safe to eat.
- **Fit Criterion:** WeAllocate must comply with all applicable food safety regulations.
- **Acceptance Test: Food Safety**

ID# SCR-002 – User Safety

- **Description:** WeAllocate must not generate distribution strategies that could put users at risk of injury.
- **Rationale:** This requirement ensures that users of WeAllocate are safe from harm.
- **Fit Criterion:** WeAllocate must comply with all applicable safety standards.
- **Acceptance Test:** User Safety test

ID# SCR-003 – Data Security

- **Description:** WeAllocate must protect the privacy and security of user data.
- **Rationale:** This requirement ensures that the data of food banks and their users is protected from unauthorized access, use, disclosure, disruption, modification, or destruction.
- **Fit Criterion:** WeAllocate must comply with all applicable data security standards.
- **Acceptance Test: Protection**

13 Maintainability and Supportability Requirements

13aMaintenance Requirements

ID# MR-001 – Modularity

- **Description:** WeAllocate must be designed in a modular way to facilitate maintenance and updates.
- **Rationale:** This requirement ensures that WeAllocate can be easily modified and updated without impacting other parts of the system.
- **Fit Criterion:** WeAllocate must be divided into a set of well-defined modules, each with its own specific function.
- **Acceptance Test:** Update

ID# MR-002 – Testability

- **Description:** WeAllocate must be designed in a testable way to facilitate maintenance and debugging.
- **Rationale:** This requirement ensures that WeAllocate can be easily tested to identify and fix bugs.
- **Fit Criterion:** WeAllocate must have a comprehensive set of unit tests, integration tests, and system tests.
- **Acceptance Test:** Unit Testing

ID# MR-003 – Documentation

- **Description:** WeAllocate must be well-documented to facilitate maintenance and updates.
- **Rationale:** This requirement ensures that WeAllocate can be easily understood and maintained by developers who did not originally write the code.
- **Fit Criterion:** WeAllocate must have comprehensive documentation that includes design documents, user guides, and API documentation.
- **Acceptance Test:** Update

ID# - WeAllocate System Updates

- **Description**: The WeAllocate system should be designed to allow for regular updates and patches without significant downtime. Any major updates should be scheduled during off-peak hours to minimize disruption.
- **Rationale**: Regular updates are essential to ensure the system remains secure, efficient, and up-to-date with the latest features and improvements.
- **Fit Criterion**: System updates should not result in more than 2 hours of downtime, and users should be notified at least 48 hours in advance of any planned maintenance or updates.
- **Acceptance Tests**: System Update Test, User Notification Test.

13b Supportability Requirements

ID# SR-001 – Help Desk

- **Description:** WeAllocate must be supported by a 24/7 help desk that can help food banks and their users.
- **Rationale:** This requirement ensures that food banks can get help with WeAllocate whenever they need it.
- **Fit Criterion:** The WeAllocate help desk must be staffed by experienced support personnel who can answer questions and resolve issues quickly and effectively.
- **Acceptance Test:** Support

ID# SR-002 – Knowledge Base

- **Description:** WeAllocate must have a comprehensive knowledge base that contains articles and tutorials on how to use WeAllocate.
- **Rationale:** This requirement ensures that food banks and their users can find the information they need to use WeAllocate effectively.
- **Fit Criterion:** The WeAllocate knowledge base must be well-organized and easy to search. It must also be up-to-date with the latest information about WeAllocate.
- **Acceptance Test:** Knowledge

ID# SR-003 – Self-Service Support

- **Description:** WeAllocate should provide self-service support options, such as a FAQ page and a community forum.
- **Rationale:** This requirement ensures that food banks and their users can find help with WeAllocate without having to contact the help desk.
- **Fit Criterion:** The WeAllocate FAQ page and community forum should be well-maintained and contain helpful information about WeAllocate.
- **Acceptance Test:** Self Support

13cAdaptability Requirements

ID# AR-001 – Cross - Platform Support

- **Description:** WeAllocate must be able to run on all major operating systems, including Windows, macOS, and Linux.
- **Rationale:** This requirement ensures that food banks and their users can use WeAllocate regardless of what operating system they are using.
- **Fit Criterion:** WeAllocate must be able to be installed and run on all major operating systems without any additional dependencies.
- **Acceptance Test:** Platform test

ID# AR-002 – Future-Proofing

- **Description:** WeAllocate should be designed to be adaptable to future changes in technology.
- **Rationale:** This requirement ensures that WeAllocate will continue to be useful to food banks even as technology changes.
- **Fit Criterion:** WeAllocate should use standard technologies and interfaces whenever possible.
- **Acceptance Test:** Future

ID# AR-003 – Cloud Support

- **Description:** WeAllocate must be able to be deployed in a cloud environment, such as AWS or Azure.
- **Rationale:** This requirement ensures that food banks can deploy WeAllocate in a way that best meets their needs.
- **Fit Criterion:** WeAllocate must be able to be deployed and managed using standard cloud infrastructure tools.
- **Acceptance Test:** Cloud Test

13d Scalability or Extensibility Requirements

ID# SER-001 – Scalability to 100,000 food banks

- **Description:** WeAllocate must be able to support 100,000 food banks without any performance degradation.

- **Rationale:** This requirement ensures that WeAllocate can scale to meet the needs of a growing number of food banks.
- **Fit Criterion:** WeAllocate must be able to handle 100,000 concurrent users without any performance degradation.
- **Acceptance Test:** Load manage 2.0

ID# SER-002 – Scalability to 500,000 food banks

- **Description:** WeAllocate must be able to support 500,000 food banks without any performance degradation.
- **Rationale:** This requirement ensures that WeAllocate can continue to meet the needs of food banks even as they grow.
- **Fit Criterion:** WeAllocate must be able to handle 500,000 concurrent users without any performance degradation.
- **Acceptance Test:** Load manage 3.0

ID# SER-003 – Elasticity

- **Description:** WeAllocate must be able to scale up or down on demand to meet the changing needs of food banks.
- **Rationale:** This requirement ensures that food banks can use WeAllocate efficiently and cost-effectively.
- **Fit Criterion:** WeAllocate must be able to scale up or down within 15 minutes without any data loss.
- **Acceptance Test:** Elastic

13eLongevity Requirements

ID# LR-001 – 10-Year Support

- **Description:** WeAllocate must be supported for at least 10 years.
- **Rationale:** This requirement ensures that food banks can rely on WeAllocate for the long term.
- **Fit Criterion:** WeAllocate must have a team of developers who are committed to maintaining and updating WeAllocate for at least 10 years.
- **Acceptance Test:** 10-Year

ID# LR-002 – Backward Compatibility

- **Description:** WeAllocate must be backward compatible with previous versions of the product.
- **Rationale:** This requirement ensures that food banks can upgrade to new versions of WeAllocate without disrupting their operations.
- **Fit Criterion:** WeAllocate must be able to import data from previous versions of the product.
- **Acceptance Test: BackWards**

ID# LR-003 – Technology Refresh

- **Description:** WeAllocate must be designed to be compatible with future technology refreshes.
- **Rationale:** This requirement ensures that WeAllocate will remain useful to food banks even as technology changes.
- **Fit Criterion:** WeAllocate must use standard technologies and interfaces whenever possible.
- **Acceptance Test:** Technology refresh

14 Security Requirements

14aAccess Requirements

ID# AR-001 – Role-Based Access Control

- **Description:** WeAllocate must use role-based access control to restrict access to different parts of the system based on user roles.
- **Rationale:** This requirement ensures that only authorized users have access to sensitive data and functionality.
- **Fit Criterion:** WeAllocate must have a set of pre-defined user roles with different permissions.
- **Acceptance Test: ROLES**

ID# AR-002 – Multi-Factor Authentication

- **Description:** WeAllocate must require multi-factor authentication for all users.
- **Rationale:** This requirement adds an extra layer of security to protect user accounts from unauthorized access.
- **Fit Criterion:** WeAllocate must support multiple authentication methods, such as SMS and Google Authenticator.
- **Acceptance Test: 2FA**

ID# AR-003 – Data Encryption

- **Description:** WeAllocate must encrypt all sensitive data, both at rest and in transit.
- **Rationale:** This requirement protects sensitive data from unauthorized access, even if the data is compromised.
- **Fit Criterion:** WeAllocate must use a strong encryption algorithm.
- **Acceptance Test: Data Encrypt**

ID# AR-004 – Audit Logging

- **Description:** WeAllocate must maintain an audit log of all user activity.
- **Rationale:** This requirement allows administrators to track user activity and investigate suspicious activity.

- **Fit Criterion:** WeAllocate must log all user activity, including logins, logouts, and changes to data.
- **Acceptance Test: AUDIT test**

14b Integrity Requirements

ID# IR-001 – Audit Logging II

- **Description:** WeAllocate must log all user activity in a tamper-proof manner.
- **Rationale:** This requirement ensures that all changes to data and system settings are logged and can be audited.
- **Fit Criterion:** WeAllocate must log all user activity, including logins, logouts, and changes to data. WeAllocate audit logs must be tamper-proof and must be retained for a minimum of one year.
- **Acceptance Test: AUDIT 2.0**

ID# IR-002 – Data Integrity

- **Description:** WeAllocate must ensure the integrity of all data, both at rest and in transit.
- **Rationale:** This requirement ensures that data is protected from corruption and tampering.
- **Fit Criterion:** WeAllocate must use checksums or other techniques to ensure the integrity of data at rest and in transit. WeAllocate must also implement mechanisms to detect and prevent data corruption and tampering.
- **Acceptance Test: DATA**

ID# IR-003 – System Integrity

- **Description:** WeAllocate must ensure the integrity of the system itself, including its code, configuration, and data.
- **Rationale:** This requirement ensures that the system is protected from unauthorized modification and corruption.
- **Fit Criterion:** WeAllocate must use digital signatures or other techniques to ensure the integrity of its code and configuration. WeAllocate must also implement mechanisms to detect and prevent unauthorized modification and corruption of the system.
- **Acceptance Test: System Integrity**

14cPrivacy Requirements

ID# PR-001 – User Consent

- **Description:** WeAllocate must obtain user consent before collecting any personal data.
- **Rationale:** This requirement ensures that users understand how their data will be used and that they have a choice about whether or not to share it.

- **Fit Criterion:** WeAllocate must have a privacy policy that clearly explains how user data will be collected and used. Users must be able to opt out of having their data collected and used for marketing purposes.
- **Acceptance Test: User Consent**

ID# PR-002 – Data Minimization

- **Description:** WeAllocate must only collect the personal data that is necessary to provide its services.
- **Rationale:** This requirement ensures that WeAllocate does not collect more data than it needs, and that user data is not used for purposes that users have not consented to.
- **Fit Criterion:** WeAllocate must have a data retention policy that specifies how long user data will be retained and for what purposes. Users must be able to request that their data be deleted.
- **Acceptance Test: Minimize**

ID# PR-003 – Data Security

- **Description:** WeAllocate must implement appropriate security measures to protect user data from unauthorized access, use, disclosure, modification, or destruction.
- **Rationale:** This requirement ensures that user data is safe and secure.
- **Fit Criterion:** WeAllocate must use industry-standard security practices to protect user data. This includes encrypting user data at rest and in transit, using strong authentication mechanisms, and regularly auditing its security systems.
- **Acceptance Test: Data Security**

ID# PR-004 – Compliance with Laws and Regulations

- **Description:** WeAllocate must comply with all applicable laws and regulations related to privacy.
- **Rationale:** This requirement ensures that WeAllocate is operating in a compliant manner.
- **Fit Criterion:** WeAllocate must have a team of lawyers who regularly review the product to ensure that it complies with all applicable laws and regulations related to privacy.
- **Acceptance Test: Law**

14d Audit Requirements

ID# AR-001 – Transaction Logging

- **Description:** WeAllocate must log all transactions in a tamper-proof manner.
- **Rationale:** This requirement ensures that all transactions can be audited for accuracy and completeness.
- **Fit Criterion:** WeAllocate must log all transactions, including the date and time of the transaction, the user who performed the transaction, and the details of the transaction.

WeAllocate transaction logs must be tamper-proof and must be retained for a minimum of seven years.

- **Acceptance Test: Logging Transaction**

ID# AR-002 – Audit Reports

- **Description:** WeAllocate must generate a variety of audit reports, such as transaction logs, user activity reports, and system configuration reports.
- **Rationale:** This requirement makes it easy for auditors to review WeAllocate activity and identify any potential problems.
- **Fit Criterion:** WeAllocate must generate a variety of audit reports, which can be customized to meet the specific needs of the auditor.
- **Acceptance Test: Audit Report**

ID# AR-003 – Audit Support

- **Description:** WeAllocate must provide support for auditors, such as access to audit logs and the ability to run custom queries.
- **Rationale:** This requirement makes it easier for auditors to perform their job and ensure that WeAllocate is operating in a compliant manner.
- **Fit Criterion:** WeAllocate must provide auditors with access to audit logs and the ability to run custom queries. WeAllocate must also have a team of support personnel who can assist auditors with any questions they may have.
- **Acceptance Test: Audit Support**

14eImmunity Requirements

ID# IMR-001 – Virus Protection

- **Description:** WeAllocate must be immune to all known viruses.
- **Rationale:** This requirement ensures that WeAllocate cannot be infected by viruses and that user data is protected from corruption and theft.
- **Fit Criterion:** WeAllocate must be able to detect and prevent all known viruses. WeAllocate must also be able to update its virus definitions on a regular basis.
- **Acceptance Test: Virus**

ID# IMR-002 – Worm Protection

- **Description:** WeAllocate must be immune to all known worms.
- **Rationale:** This requirement ensures that WeAllocate cannot be infected by worms and that user data is protected from corruption and theft.
- **Fit Criterion:** WeAllocate must be able to detect and prevent all known worms. WeAllocate must also be able to update its worm signatures on a regular basis.
- **Acceptance Test: Worm**

ID# IMR-003 – Trojan Horse Protection

- **Description:** WeAllocate must be immune to all known Trojan horses.
- **Rationale:** This requirement ensures that WeAllocate cannot be infected by Trojan horses and that user data is protected from corruption and theft.
- **Fit Criterion:** WeAllocate must be able to detect and prevent all known Trojan horses. WeAllocate must also be able to update its Trojan horse signatures on a regular basis.
- **Acceptance Test: Trojan**

ID# IMR-004 – Sandboxing

- **Description:** WeAllocate must sandbox all untrusted code, such as attachments and plugins.
- **Rationale:** This requirement prevents untrusted code from executing and harming WeAllocate or the user's system.
- **Fit Criterion:** WeAllocate must implement a sandbox that prevents untrusted code from accessing or modifying the system or the user's data. WeAllocate must also test its sandbox regularly to ensure that it is effective.
- **Acceptance Test: Sandbox**

ID# IMR-005 – Security Updates

- **Description:** WeAllocate must be able to receive and install security updates on a regular basis.
- **Rationale:** This requirement ensures that WeAllocate is protected from the latest security threats.
- **Fit Criterion:** WeAllocate must be able to receive and install security updates from the WeAllocate development team. WeAllocate must also notify users when security updates are available.
- **Acceptance Test: Security Update**

15 Usability and Humanity Requirements

15aEase of Use Requirements

ID# EUR-001 – Learnability

- **Description:** WeAllocate should be easy for new users to learn how to use.
- **Rationale:** This requirement ensures that users can be productive with WeAllocate quickly and without extensive training.
- **Fit Criterion:** 80% of new users should be able to complete basic tasks in WeAllocate within 30 minutes of their first use.
- **Acceptance Test: Learning**

ID# EUR-002 – Efficiency

- **Description:** WeAllocate should be efficient to use, allowing users to complete tasks quickly and accurately.

- **Rationale:** This requirement ensures that users can be productive with WeAllocate and avoid wasting time.
- **Fit Criterion:** Experienced users should be able to complete common tasks in WeAllocate in less than 1 minute.
- **Acceptance Test: Efficiency**

ID# EUR-003 – Error Prevention

- **Description:** WeAllocate should help users to avoid making errors.
- **Rationale:** This requirement ensures that users can use WeAllocate confidently and without worrying about making mistakes.
- **Fit Criterion:** WeAllocate should have features that help users to validate their input and prevent them from making common errors.
- **Acceptance Test: Error**

ID# EUR-004 – Memorability

- **Description:** WeAllocate should be easy for users to remember how to use.
- **Rationale:** This requirement ensures that users do not have to constantly re-learn how to use WeAllocate, even if they do not use it frequently.
- **Fit Criterion:** WeAllocate should have a consistent and intuitive user interface. WeAllocate should also provide users with clear and concise help documentation.
- **Acceptance Test: Memorability**

ID# IMR-005 – Satisfaction

- **Description:** WeAllocate should be pleasant and enjoyable to use.
- **Rationale:** This requirement ensures that users are likely to continue using WeAllocate and that they will recommend it to others.
- **Fit Criterion:** WeAllocate should have a visually appealing user interface and should be easy to navigate. WeAllocate should also be responsive and provide users with feedback on their actions.
- **Acceptance Test: Product Survey**

15b Personalization and Internationalization Requirements

ID# PIR-001 – Language Support

- **Description:** WeAllocate should support multiple languages.
- **Rationale:** This requirement ensures that WeAllocate can be used by users from all over the world.
- **Fit Criterion:** WeAllocate should support at least the following languages: English, Spanish, French, German, Chinese, and Japanese.
- **Acceptance Test: Language**

ID# PIR-002 – Currency Support

- **Description:** WeAllocate should support multiple currencies.
- **Rationale:** This requirement ensures that WeAllocate can be used by users from all over the world.
- **Fit Criterion:** WeAllocate should support at least the following currencies: USD, EUR, GBP, CAD, JPY, and CNY.
- **Acceptance Test: Currency**

ID# PIR-003 – Personalization Options

- **Description:** WeAllocate should allow users to personalize their experience.
- **Rationale:** This requirement ensures that users can use WeAllocate in a way that is most comfortable for them.
- **Fit Criterion:** WeAllow should allow users to customize the following settings: language, currency, date and time format, and theme.
- **Acceptance Test: Personalize**

15cLearning Requirements

ID# LER-001 – Onboarding

- **Description:** WeAllocate should provide users with a clear and concise onboarding experience.
- **Rationale:** This requirement ensures that new users can learn how to use WeAllocate quickly and easily.
- **Fit Criterion:** WeAllocate should provide a guided tour of the product and interactive tutorials on how to complete common tasks.
- **Acceptance Test: Onboard**

ID# LER-002 – Help Documentation

- **Description:** WeAllocate should provide comprehensive and helpful documentation.
- **Rationale:** This requirement ensures that users have a resource to reference when they need help learning how to use WeAllocate or troubleshooting problems.
- **Fit Criterion:** WeAllocate should provide documentation that covers all of the product's features and functionality. The documentation should be well-written and easy to understand.
- **Acceptance Test: Help Documentation**

ID# LER-003 – Support

- **Description:** WeAllocate should provide users with access to support in case they need help learning how to use the product or troubleshooting problems.
- **Rationale:** This requirement ensures that users have a way to get help if they are struggling to learn or use WeAllocate.

- **Fit Criterion:** WeAllocate should provide support through a variety of channels, such as email, live chat, and phone support. WeAllocate should also have a knowledge base of common questions and answers.
- **Acceptance Test: WeAllocate support**

15d Understandability and Politeness Requirements

ID# UPR-001 – Use Familiar Technology

- **Description:** WeAllocate should use terminology that is familiar to its users.
- **Rationale:** This requirement ensures that users can understand WeAllocate's messages and instructions without having to learn a new vocabulary.
- **Fit Criterion:** WeAllocate should avoid using jargon and technical terms whenever possible. When technical terms are necessary, WeAllocate should provide clear and concise definitions.
- **Acceptance Test: Tech**

ID# UPR-002 – Provide Clear and Concise Instructions

- **Description:** WeAllocate should provide clear and concise instructions to its users.
- **Rationale:** This requirement ensures that users can understand how to use WeAllocate's features and functionality.
- **Fit Criterion:** WeAllocate's instructions should be written in plain language and should be easy to follow. WeAllocate should also provide screenshots and other visual aids to help users understand the instructions.
- **Acceptance Test:**
    - Test 1: Verify that WeAllocate's instructions are clear and concise.
    - Test 2: Observe users as they follow WeAllocate's instructions to complete tasks. Identify any areas where the instructions are unclear or confusing.

ID# UPR-003 – Avoid Technical Details

- **Description:** WeAllocate should avoid exposing technical details to its users.
- **Rationale:** This requirement ensures that WeAllocate's user interface is simple and easy to use. WeAllocate should hide the details of its internal construction from the user and focus on providing a user-friendly interface.
- **Fit Criterion:** WeAllocate should make sure that the program and information used to make the service run remain hidden from the public.
- **Acceptance Test:**
    - Test 1: Verify that WeAllocate hides the details of its internal construction from the user.
    - Test 2: Observe users as they interact with WeAllocate and identify any areas where the product exposes technical details to the user.

15e Accessibility Requirements

ID# ARE-001 – Screen Reader Support

- **Description:** WeAllocate should support screen readers.
- **Rationale:** This requirement ensures that WeAllocate can be used by blind and visually impaired users.
- **Fit Criterion:** WeAllocate should have all of its text and interactive elements accessible to screen readers. WeAllocate should also provide keyboard shortcuts for all of its features and functionality.
- **Acceptance Test: ScreenRead**

ID# ARE-002 – Keyboard Navigation

- **Description:** WeAllocate should be fully navigable using the keyboard.
- **Rationale:** This requirement ensures that WeAllocate can be used by users with limited mobility or who cannot use a mouse.
- **Fit Criterion:** WeAllocate should have all its interactive elements focusable using the keyboard. WeAllocate should also provide keyboard shortcuts for all its features and functionality.
- **Acceptance Test: Keyboard**

ID# ARE-003 – Color Contrast

- **Description:** WeAllocate should have sufficient color contrast between text and background elements.
- **Rationale:** This requirement ensures that WeAllocate can be used by users with color blindness and low vision.
- **Fit Criterion:** WeAllocate should meet color contrast requirements.
- **Acceptance Test: Color**

ID# ARE-004 – Accessible Documentation

- **Description:** WeAllocate's documentation should be accessible to users with disabilities.
- **Rationale:** This requirement ensures that all users can access WeAllocate's documentation and learn how to use the product.
- **Fit Criterion:** WeAllocate's documentation should be available in multiple formats, such as HTML, PDF, and plain text. WeAllocate's documentation should also be written in plain language and should be accompanied by screenshots and other visual aids.
- **Acceptance Test: Documentation Accessibility**

15f User Documentation Requirements

ID# UDR-001 – User Documentation Maintenance

- **Description:** The WeAllocate development team and support team will be responsible for keeping the user documentation up-to-date.
- **Rationale:** This requirement ensures that the user documentation is accurate and reflects the latest changes to WeAllocate.
- **Fit Criterion:** The user documentation should be updated within one week of any changes to WeAllocate.
- **Acceptance Test:**
    - Test 1: Verify that the user documentation is updated within one week of any changes to WeAllocate.
    - Test 2: Observe users as they use the user documentation to complete tasks. Identify any areas where the documentation is out of date or inaccurate.

Document	Purpose	Audience	Maintenance	Form
User Manual	Provides instructions on how to use WeAllocate's features and functionality.	All users of WeAllocate	Maintained by the WeAllocate development team	HTML, PDF, and plain text
Installation Guide	Provides instructions on how to install and configure WeAllocate.	IT administrators	Maintained by the WeAllocate development team	HTML, PDF, and plain text
Troubleshooting Guide	Provides instructions on how to troubleshoot common problems with WeAllocate.	All users of WeAllocate	Maintained by the WeAllocate support team	HTML, PDF, and plain text
API Documentation	Provides documentation for WeAllocate's REST API.	Developers	Maintained by the WeAllocate development team	HTML, PDF, and plain tex

Knowledge Base	A collection of articles and resources to help users learn about and use WeAllocate.	All users of WeAllocate	Maintained by the WeAllocate support team	HTML, PDF, and plain tex

15gTraining Requirements

ID# TR-001 – User Training

- **Description:** WeAllocate will provide training to users on how to use the application and it's features
- **Rationale:** This requirement ensures that users are able to use WeAllocate effectively and efficiently.
- **Fit Criterion:** All users of WeAllocate must complete the WeAllocate Overview and WeAllocate Fundamentals training modules.

ID# TR-002 – Custom Training

- **Description:** WeAllocate will provide custom training to users on specific roles or use cases.
- **Rationale:** This requirement ensures that users have the training they need to use WeAllocate in the most effective way possible.
- **Fit Criterion:** Custom training will be provided to users upon request.
- **Acceptance Test: User Training**

ID# TR-003 – Training Evaluation

- **Description:** WeAllocate will collect feedback from users on the training they receive.
- **Rationale:** This requirement ensures that the training meets the needs of the users and that it is effective.
- **Fit Criterion:** WeAllocate will collect feedback from users after they complete each training module.
- **Acceptance Test: Train**

16 Look and Feel Requirements

16aAppearance Requirements

ID# APR-001 – Branding

- **Description:** WeAllocate must comply with the WeAllocate brand identity guidelines.
- **Rationale:** This requirement ensures that WeAllocate has a consistent and cohesive visual identity.
- **Fit Criterion:** WeAllocate must use the WeAllocate logo, colors, and fonts consistently throughout the product.
- **Acceptance Test:** Brand

ID# APR-002 – Color Scheme

- **Description:** WeAllocate should use a color scheme that is both visually appealing and accessible.
- **Rationale:** This requirement ensures that WeAllocate is easy to read and use for all users, including those with color blindness.
- **Fit Criterion:** WeAllocate should use a color scheme that meets the WCAG 2.1 AA color contrast requirements.
- **Acceptance Test: Color**

ID# APR-003 – Typography

- **Description:** WeAllocate should use a typography that is both visually appealing and easy to read.
- **Rationale:** This requirement ensures that WeAllocate is easy to read and use for all users.
- **Fit Criterion:** WeAllocate should use a font that is large enough to be easily readable, and that has sufficient line height and kerning.
- **Acceptance Test: Font**

16b Style Requirements

ID# STR-001 – Professionalism

- **Description:** WeAllocate should have a professional and polished appearance.
- **Rationale:** This requirement ensures that WeAllocate is taken seriously by users and that it is perceived as a credible product.
- **Fit Criterion:** WeAllocate should use high-quality graphics and fonts, and the overall design should be clean and uncluttered.
- **Acceptance Test: Survey 2.0**

ID# STR-002 – User Friendliness

- **Description:** WeAllocate should be easy to use and navigate..
- **Rationale:** This requirement ensures that WeAllocate is accessible to all users, regardless of their technical expertise.
- **Fit Criterion:** WeAllocate should use clear and concise language, and the overall design should be intuitive and easy to follow.
- **Acceptance Test: Friendliness**

ID# STR-003 – Visual Appeal

- **Description:** WeAllocate should have a visually appealing and engaging design.
- **Rationale:** This requirement ensures that WeAllocate is enjoyable to use and that it keeps users engaged.
- **Fit Criterion:** WeAllocate should use a color scheme and typography that is both visually appealing and accessible. WeAllocate should also use high-quality graphics and images.
- **Acceptance Test: Visual**

17 Operational and Environmental Requirements

17aExpected Physical Environment

ID# EPE-001 – Device Requirements

- **Description:** Powered by a proper device with specifications that allow it to run.
- **Rationale:** This requirement ensures that WeAllocate can be powered by a variety of devices, including computers, laptops, and power banks.
- **Fit Criterion:** WeAllocate should be able to operate normally when powered by a standard device.
- **Acceptance Test: Device**

17b Requirements for Interfacing with Adjacent Systems

ID# RIA-001 – Interface with Customer Relationship Management

- **Description:** WeAllocate should be able to interface with the customer relationship management (CRM) system to retrieve and update customer data.
- **Rationale:** This requirement ensures that WeAllocate can be integrated with the CRM system to provide a seamless user experience.
- **Fit Criterion:** WeAllocate should be able to retrieve and update customer data from the CRM system in real time.
- **Acceptance Test: CRM**

ID# RIA-002 – Interface with Accounting System

- **Description:** WeAllocate should be able to interface with the accounting system to export financial data.
- **Rationale:** This requirement ensures that WeAllocate can be integrated with the accounting system to streamline financial reporting.
- **Fit Criterion:** WeAllocate should be able to export financial data to the accounting system in a format that is compatible with the accounting system.
- **Acceptance Test: Accounting System**

ID# RIA-003 – Email System

- **Description:** WeAllocate should be able to interface with the email system to send and receive emails.
- **Rationale:** This requirement ensures that WeAllocate can be used to communicate with users via email.
- **Fit Criterion:** WeAllocate should be able to send and receive emails using the email system's servers.
- **Acceptance Test: Email**

17cProductization Requirements

ID# PRE-001 – Distribution Format

- **Description:** WeAllocate should be distributed as a ZIP file.
- **Rationale:** This requirement ensures that WeAllocate is easy to distribute and install.
- **Fit Criterion:** WeAllocate should be distributed as a ZIP file that can be downloaded from the WeAllocate website.
- **Acceptance Test: Zip test**

ID# PRE-002 – Installation Process

- **Description:** WeAllocate should be able to be installed without recourse to separately printed instructions
- **Rationale:** This requirement ensures that WeAllocate is easy to install and use.
- **Fit Criterion:** WeAllocate should have a self-installer that can be used to install the product without the need for any special instructions.
- **Acceptance Test: Install**

ID# PRE-003 – Licensing

- **Description:** WeAllocate should be licensed using a subscription-based model.
- **Rationale:** This requirement ensures that WeAllocate is a sustainable business model.
- **Fit Criterion:** WeAllocate should have a subscription management system that allows users to subscribe to and manage their WeAllocate subscriptions.
- **Acceptance Test: License**

ID# PRE-004 – Support

- **Description:** WeAllocate should have a dedicated support team that can aid users with problems or questions.
- **Rationale:** This requirement ensures that users can get help with WeAllocate when they need it.
- **Fit Criterion:** WeAllocate should have a support portal where users can submit tickets and get help from the WeAllocate support team.
- **Acceptance Test: Ticket**

17d Release Requirements

ID# RPE-001 – Updates

- **Description:** WeAllocate should release small updates to patch bugs or add new features as a patch or minor update.
- **Rationale:** This requirement ensures that WeAllocate releases are small and manageable, and that they do not cause disruption to users.
- **Fit Criterion:** WeAllocate releases should be no larger than 50 MB in size and should not contain any major changes to the product's functionality.
- **Acceptance Test: App Update**

ID# RPE-002 – Release Announcements

- **Description:** WeAllocate will announce new releases on the WeAllocate website and social media channels.
- **Rationale:** This will ensure that users are aware of new releases and can download and install them.
- **Fit Criterion:** WeAllocate will announce new releases on the WeAllocate website, Twitter, and Facebook.
- **Acceptance Test: Social Media**

ID# RPE-004 – Release Notes

- **Description:** WeAllocate will include release notes with each new release.
- **Rationale:** This will provide users with information about the new features and bug fixes in each release.
- **Fit Criterion:** WeAllocate will include release notes with each new release that detail the new features and bug fixes in the release.
- **Acceptance Test: Patch**

18 Cultural and Political Requirements

18a Cultural Requirements

ID# CR-001 – Multilingual support

- **Description:** WeAllocate should support multiple languages, including English, Spanish, French, German, Chinese, Japanese, and Korean.
- **Rationale:** This requirement ensures that WeAllocate is accessible to users from all over the world if needed.
- **Fit Criterion:** WeAllocate should be able to be used in all the supported languages without any errors or glitches.
- **Acceptance Test: Language 2.0**

ID# CR-002 – Cultural Sensitivity

- **Description:** WeAllocate should be culturally sensitive and avoid using any offensive or insensitive language or imagery.
- **Rationale:** This requirement ensures that WeAllocate is respectful of all users, regardless of their cultural background.
- **Fit Criterion:** WeAllocate should be reviewed by a team of cultural sensitivity experts to ensure that it is appropriate for all users.
- **Acceptance Test:**

ID# CRE-003 – Holiday Support

- **Description:** WeAllocate should keep track of public holidays for all countries in the world.
- **Rationale:** This requirement ensures that WeAllocate can be used accurately and effectively by users from all over the world.
- **Fit Criterion:** WeAllocate should be able to calculate deadlines and other dates accurately, considering public holidays for the user's selected country.
- **Acceptance Test: Holidays**

18b Political Requirements

ID# PPR-001 – Sponsors and CEO

- **Description:** WeAllocate should have the executive sponsorship of the CEO.
- **Rationale:** This requirement ensures that WeAllocate has the support of the highest levels of the organization.
- **Fit Criterion:** The CEO should publicly endorse WeAllocate and advocate for its use throughout the organization.
- **Acceptance Test: CEO**

ID# PPR-002 – Integration With Existing Systems

- **Description:** WeAllocate should be integrated with all of the organization's existing systems.
- **Rationale:** This requirement ensures that WeAllocate can be used seamlessly with the organization's other IT systems.
- **Fit Criterion:** WeAllocate should be able to import and export data from all of the organization's existing systems.
- **Acceptance Test: Integration**

ID# PPR-003 – Compliance and Regulations

- **Description:** WeAllocate should comply with all applicable laws and regulations.
- **Rationale:** This requirement ensures that WeAllocate can be used without violating any laws or regulations.

- **Fit Criterion:** WeAllocate should be reviewed by a legal team to ensure that it complies with all applicable laws and regulations.
- **Acceptance Test: Regulate**

19 Legal Requirements

19a Compliance Requirements

ID# CCR-001 – Data Protection

- **Description:** WeAllocate should comply with applicable data protection laws and regulations.
- **Rationale:** This requirement ensures that WeAllocate protects the privacy of its users' data.
- **Fit Criterion:** WeAllocate should be reviewed by a legal team to ensure that it complies with all applicable data protection laws and regulations.
- **Acceptance Test: DATAPROTECT**

ID# CCR-002 – Security

- **Description:** WeAllocate should implement appropriate security measures to protect user data from unauthorized access, use, disclosure, modification, or destruction.
- **Rationale:** This requirement ensures that WeAllocate is secure, and that user data is protected from unauthorized access and use.
- **Fit Criterion:** WeAllocate should be subjected to a security audit to ensure that it meets all applicable security requirements.
- **Acceptance Test: N/A**

ID# CCR-003 – Accessibility

- **Description:** WeAllocate should be accessible to users with disabilities.
- **Rationale:** This requirement ensures that WeAllocate is accessible to all users, regardless of their abilities.
- **Fit Criterion:** WeAllocate should be reviewed by an accessibility expert to ensure that it meets all applicable accessibility requirements.
- **Acceptance Test: Accessibility**

19b Standards Requirements

ID# SRR-003 – Standards

- **Description:** WeAllocate should comply with all applicable industry standards.
- **Rationale:** This requirement ensures that WeAllocate meets the expectations of its target users and that it is interoperable with other products and systems.
- **Fit Criterion:** WeAllocate should be reviewed by a team of industry experts to ensure that it complies with all applicable industry standards.

- **Acceptance Test: Standard**

ID# SSR-002 – Security Standards

- **Description:** WeAllocate should comply with all applicable security standards.
- **Rationale:** his requirement ensures that WeAllocate is secure, and that user data is protected.
- **Fit Criterion:** WeAllocate should be subjected to a security audit to ensure that it meets all applicable security standards.
- **Acceptance Test:**
    - Test 1: Verify that WeAllocate has been subjected to a security audit and that it meets all applicable security standards.
    - Test 2: Observe users as they interact with WeAllocate and how their data is secured. Identify any areas where the product's security is inadequate.

ID# SSR-003 – Development

- **Description:** WeAllocate should be developed according to all applicable development standards.
- **Rationale:** This requirement ensures that WeAllocate is well-designed, maintainable, and scalable.
- **Fit Criterion:** WeAllocate should be reviewed by a team of development experts to ensure that it has been developed according to all applicable development standards.
- **Acceptance Test: DevelopmentTest**
    - Test 1: Verify that WeAllocate has been reviewed by a team of development experts and that it has been developed according to all applicable development standards.
    - Test 2: Observe developers as they maintain and develop WeAllocate. Identify any areas where the product's design or development process makes it difficult to maintain or develop the product.

20 Requirements Acceptance Tests

20a Requirements – Test Correspondence Summary

Test	Requirements																			
	Req 1	Req 2	Req 3	Req 4	Req 5	Req 6	Req 7	Req 8	Req 9	Req 10	Req 11	Req 12	Req 13	Req 14	Req 15	Req 16	Req 17	Req 18	Req 19	Req 20
Test 1	X																			
Test 2	X									X						X			X	
Test 3	X	X					X							X						
Test 4															X					
Test 5			X							X		X			X			X		
Test 6							X	X												
Test 7							X													
Test 8											X		X							X
Test 9								X												
Test 10											X							X		
Test 11																				
Test 12									X			X								X
Test 13																				
Test 14																				
Test 15																				

Table 1 - Requirements - Acceptance Tests Correspondence

20b Acceptance Test Descriptions

ML test

- Test the system's ability to train on historical data and make predictions for future demands.
- Compare the system's predictions with actual demands over a test period to measure accuracy.
- Monitor resource wastage rates when using the system's predictive allocations versus traditional allocation methods.

Feedback

- Test the feedback submission process.
- Monitor the responsiveness and effectiveness of the support feature.

Events

- Test the display of events in the portal.
- Validate that detailed information is provided for each event when selected.

IDM

- Test the system's ability to add, update, and delete inventory items.
- Validate the system's response to discrepancies in inventory data.

Demographic

- Test the system's ability to capture and update beneficiary demographic data.
- Validate the system's use of this data in making allocation decisions.

Feedback

- Test the feedback and survey submission processes.
- Validate the storage and retrieval of this data for analysis.

Donation track

- Test the system's ability to capture real-time donation details for both monetary and in-kind contributions.
- Validate the generation of acknowledgment receipts and donation reports.

Donation management

- Test the system's ability to set up and manage recurring donations.
- Validate the reminder and notification system for upcoming recurring donations.

Inventory

- Test the system's performance under varying inventory loads.
- Validate system stability and response times during peak inventory updates.

Portal Feedback

- Test the portal's ability to capture and display accurate data and feedback.
- Validate feedback submission and retrieval for accuracy.

Load manage

- Test the portal's performance under varying user loads.
- Validate portal stability and response times during peak user activity.

Donation Precision

- Test the system's ability to capture and display monetary amounts with two decimal places.
- Validate the consistency of monetary data entries.

Data Accuracy

- Test the system's ability to capture, store, and retrieve beneficiary data.
- Validate the accuracy of beneficiary data entries against actual data sources.

Inventory Precision

- Test the system's ability to capture, store, and retrieve inventory data.
- Validate the precision of inventory data entries against actual inventory sources.

Capacity

- Test the system's performance under varying user loads.
- Validate system stability and response times during peak user activity.

Inventory Capacity

- Test the system's ability to capture, store, and retrieve inventory data.
- Validate the accuracy and consistency of inventory data entries.

Fail

- Monitor system uptime and performance metrics.
- Validate system stability through stress testing and simulated high-load scenarios.

Data in System Failure

- Simulate system failures and validate data recovery processes.
- Ensure backup and restore mechanisms are functioning correctly.

Fail safe

- Test system's response to potential failure scenarios.
- Validate the effectiveness of fail-safe mechanisms in preventing further disruptions.

System Test

- Monitor system uptime and record any instances of downtime.
- Validate that the system is available for use 24/7, with exceptions for scheduled maintenance.

Downtime

- Schedule and conduct system maintenance during non-peak hours.
- Validate that scheduled maintenance does not significantly disrupt user access.

UP

- Monitor and record system uptime over a rolling 30-day period.
- Calculate and validate that the system achieves a minimum uptime percentage of 99%.

OFFLINE

- Verify that WeAllocate can be started and used in offline mode.
- Verify that WeAllocate can generate optimized distribution strategies in offline mode.
- Verify that WeAllocate can track and analyze the distribution process in offline mode.

Law

- Test 1: Verify that WeAllocate has a team of lawyers who regularly review the product to ensure that it complies with all applicable laws and regulations related to privacy.
- Test 2: Verify that WeAllocate conducts periodic privacy impact assessments to identify and mitigate privacy risks.

GD

- Test 1: Verify that WeAllocate can continue to operate if one of its disk drives fails.
- Test 2: Verify that WeAllocate can continue to operate if its power supply fails for up to 10 minutes.
- Test 3: Verify that WeAllocate can continue to generate optimized distribution strategies with reduced functionality.

Help Documentation

- Test 1: Verify that WeAllocate provides documentation that covers all of the product's features and functionality.
- Test 2: Verify that the documentation is well-written and easy to understand.
- Test 3: Observe users as they use the documentation to complete tasks or troubleshoot problems. Identify any areas where the documentation can be improved for clarity or completeness.

Recovery

- Test 1: Verify that WeAllocate can be recovered from a disk failure within 24 hours.
- Test 2: Verify that WeAllocate can be recovered from a database corruption within 24 hours.
- Test 3: Verify that WeAllocate can be recovered from a hardware failure within 24 hours.

Memorability

- Test 1: Verify that WeAllocate has a consistent and intuitive user interface.
- Test 2: Verify that WeAllocate provides users with clear and concise help documentation.
- Test 3: Observe users as they attempt to complete tasks in WeAllocate after a period of inactivity. Identify any tasks that users have difficulty remembering how to complete.

CEO

- Test 1: Verify that the CEO has publicly endorsed WeAllocate and advocated for its use throughout the organization.
- Test 2: Observe the CEO as they use WeAllocate. Identify any areas where the product does not meet the CEO's needs.

Food Safety

- Test 1: Verify that WeAllocate does not generate distribution strategies that would result in food being stored at an unsafe temperature.
- Test 2: Verify that WeAllocate does not generate distribution strategies that would result in food being cross-contaminated.
- Test 3: Verify that WeAllocate does not generate distribution strategies that would result in food being exposed to allergens.

User Safety test

- Test 1: Verify that WeAllocate does not generate distribution strategies that would require users to lift or move heavy objects.
- Test 2: Verify that WeAllocate does not generate distribution strategies that would require users to work in hazardous environments.
- Test 3: Verify that WeAllocate does not generate distribution strategies that would require users to use dangerous machinery or equipment.

Protection

- Test 1: Verify that WeAllocate encrypts all user data at rest and in transit.
- Test 2: Verify that WeAllocate implements strong authentication and authorization mechanisms.
- Test 3: Verify that WeAllocate has a comprehensive data security plan in place.

Update

- Test 1: Verify that each module of WeAllocate can be independently modified and updated without impacting other modules.
- Test 2: Verify that the documentation for each module of WeAllocate is up-to-date and complete.

Unit Testing

- Test 1: Verify that all unit tests, integration tests, and system tests for WeAllocate pass.
- Test 2: Verify that the test coverage for WeAllocate is at least 90%.

Update

- Test 1: Verify that the documentation for WeAllocate is up-to-date and complete.
- Test 2: Verify that the documentation for WeAllocate is clear and easy to understand.

Support

- Test 1: Verify that the WeAllocate help desk is available 24/7.
- Test 2: Verify that the WeAllocate help desk can be contacted by phone, email, and chat.
- Test 3: Verify that the WeAllocate help desk staff is knowledgeable about WeAllocate and can answer questions and resolve issues quickly and effectively.

Knowledge

- Test 1: Verify that the WeAllocate knowledge base contains articles and tutorials on all aspects of using WeAllocate.
- Test 2: Verify that the WeAllocate knowledge base is well-organized and easy to search.
- Test 3: Verify that the WeAllocate knowledge base is up-to-date with the latest information about WeAllocate.

Zip test

- Test 1: Verify that WeAllocate can be downloaded from the WeAllocate website in ZIP format.
- Test 2: Verify that the WeAllocate ZIP file can be extracted and installed successfully.

Self Support

- Test 1: Verify that the WeAllocate FAQ page contains answers to the most common questions about WeAllocate.
- Test 2: Verify that the WeAllocate community forum is active and that questions are answered promptly.

Platform test

- Test 1: Verify that WeAllocate can be installed and run on Windows 10, macOS 13, and Linux Mint 21.
- Test 2: Verify that WeAllocate can access all of its required resources on all major operating systems.

Future

- Test 1: Verify that WeAllocate uses standard technologies and interfaces whenever possible.
- Test 2: Verify that WeAllocate can be easily updated to support new technologies and interfaces.

Cloud Test

- Test 1: Verify that WeAllocate can be deployed and managed using AWS CloudFormation.
- Test 2: Verify that WeAllocate can scale to meet the needs of food banks of all sizes.

DATAPROTECT

- Test 1: Verify that WeAllocate has been reviewed by a legal team and that it complies with all applicable data protection laws and regulations.
- Test 2: Observe users as they interact with WeAllocate and how their data is handled. Identify any areas where the product does not comply with the applicable data protection laws and regulations.

Load manage 2.0

- Test 1: Verify that WeAllocate can handle 100,000 concurrent users without any performance degradation.
- Test 2: Verify that WeAllocate can process 100,000 requests per second without any performance degradation.

BackWards

- Test 1: Verify that WeAllocate can import data from previous versions of the product.
- Test 2: Verify that WeAllocate can run existing WeAllocate workflows without any changes.

Accessibility

- Test 1: Verify that WeAllocate's documentation is available in multiple formats, such as HTML, PDF, and plain text.
- Test 2: Verify that WeAllocate's documentation is written in plain language and is accompanied by screenshots and other visual aids.
- Test 3: Observe users with disabilities as they use WeAllocate's documentation. Identify any areas where the documentation can be improved for accessibility.

Standard

- Test 1: Verify that WeAllocate has been reviewed by a team of industry experts and that it complies with all applicable industry standards.
- Test 2: Observe users as they use WeAllocate to interact with other products and systems. Identify any areas where the product is not interoperable or does not meet the expectations of the target users.

Load manage 3.0

- Test 1: Verify that WeAllocate can handle 500,000 concurrent users without any performance degradation.
- Test 2: Verify that WeAllocate can process 500,000 requests per second without any performance degradation.

Elasticity

- Test 1: Verify that WeAllocate can scale up from 100,000 to 500,000 concurrent users within 15 minutes without any data loss.
- Test 2: Verify that WeAllocate can scale down from 500,000 to 100,000 concurrent users within 15 minutes without any data loss.

Technology refresh

- Test 1: Verify that WeAllocate uses standard technologies and interfaces whenever possible.
- Test 2: Verify that WeAllocate can be easily updated to support new technologies and interfaces.

Roles

- Test 1: Verify that WeAllocate has a set of pre-defined user roles with different permissions.
- Test 2: Verify that users can only access the parts of the system that are allowed by their role.

2FA

- Test 1: Verify that WeAllocate requires multi-factor authentication for all users.
- Test 2: Verify that WeAllocate supports multiple authentication methods.

Data Encrypt

- Test 1: Verify that WeAllocate encrypts all sensitive data at rest using a strong encryption algorithm.

- Test 2: Verify that WeAllocate encrypts all sensitive data in transit using a strong encryption algorithm.

AUDIT test

- Test 1: Verify that WeAllocate logs all user activity.
- Test 2: Verify that WeAllocate audit logs are tamper-proof.

AUDIT 2.0

- Test 1: Verify that WeAllocate logs all user activity in a tamper-proof manner.
- Test 2: Verify that WeAllocate audit logs are retained for a minimum of one year.

DATA

- Test 1: Verify that WeAllocate uses checksums or other techniques to ensure the integrity of data at rest and in transit.
- Test 2: Verify that WeAllocate implements mechanisms to detect and prevent data corruption and tampering.

System Integrity

- Test 1: Verify that WeAllocate uses digital signatures or other techniques to ensure the integrity of its code and configuration.

- Test 2: Verify that WeAllocate implements mechanisms to detect and prevent unauthorized modification and corruption of the system.

User Consent

- Test 1: Verify that WeAllocate has a privacy policy that clearly explains how user data will be collected and used.
- Test 2: Verify that users are able to opt out of having their data collected and used for marketing purposes.

Minimize

- Test 1: Verify that WeAllocate only collects the personal data that is necessary to provide its services.
- Test 2: Verify that WeAllocate has a data retention policy that specifies how long user data will be retained and for what purposes.
- Test 3: Verify that users are able to request that their data be deleted.

Data Security

- Test 1: Verify that WeAllocate encrypts user data at rest and in transit using a strong encryption algorithm.
- Test 2: Verify that WeAllocate uses strong authentication mechanisms.
- Test 3: Verify that WeAllocate regularly audits its security systems.

Logging Transaction

- Test 1: Verify that WeAllocate logs all transactions in a tamper-proof manner.
- Test 2: Verify that WeAllocate transaction logs are retained for a minimum of seven years.

10-Year

- Test 1: Verify that WeAllocate has a team of developers who are responsible for maintaining and updating the product.
- Test 2: Verify that WeAllocate has a roadmap for future development and support.

Audit Report

- Test 1: Verify that WeAllocate can generate a variety of audit reports, including transaction logs, user activity reports, and system configuration reports.

- Test 2: Verify that WeAllocate audit reports can be customized to meet the specific needs of the auditor.

Audit Support

- Test 1: Verify that WeAllocate provides auditors with access to audit logs.
- Test 2: Verify that WeAllocate provides auditors with the ability to run custom queries.
- Test 3: Verify that WeAllocate has a team of support personnel who can assist auditors with any questions they may have.

Virus

- Test 1: Verify that WeAllocate can detect and prevent all known viruses.
- Test 2: Verify that WeAllocate can update its virus definitions on a regular basis.

Worm

- Test 1: Verify that WeAllocate can detect and prevent all known worms.
- Test 2: Verify that WeAllocate can update its worm signatures on a regular basis.

Trojan

- Test 1: Verify that WeAllocate can detect and prevent all known Trojan horses.
- Test 2: Verify that WeAllocate can update its Trojan horse signatures on a regular basis.

Sandbox

- Test 1: Verify that WeAllocate implements a sandbox that prevents untrusted code from accessing or modifying the system or the user's data.
- Test 2: Verify that WeAllocate tests its sandbox regularly to ensure that it is effective.

Security Update

- Test 1: Verify that WeAllocate can receive and install security updates from the WeAllocate development team.
- Test 2: Verify that WeAllocate notifies users when security updates are available.

Learning

- Test 1: Verify that 80% of new users can complete basic tasks in WeAllocate within 30 minutes of their first use.
- Test 2: Observe new users as they attempt to complete basic tasks in WeAllocate and identify any areas where the product can be improved for usability.

Efficiency

- Test 1: Verify that experienced users can complete common tasks in WeAllocate in less than 1 minute.
- Test 2: Observe experienced users as they complete common tasks in WeAllocate and identify any areas where the product can be improved for efficiency.

Error

- Test 1: Verify that WeAllocate has features that help users to validate their input and prevent them from making common errors.
- Test 2: Observe users as they attempt to complete tasks in WeAllocate and identify any common errors that users make.

Product Survey

- Test 1: Conduct a user satisfaction survey to assess users' opinions of WeAllocate' s usability.
- Test 2: Observe users as they use WeAllocate and identify any areas where the product can be improved for satisfaction.

Language

- Test 1: Verify that WeAllocate can be translated into at least the following languages: English, Spanish, French, German, Chinese, and Japanese.
- Test 2: Verify that WeAllocate can be used in different languages without any errors or bugs.

Currency

- Test 1: Verify that WeAllocate can display and calculate prices in different currencies.
- Test 2: Verify that WeAllocate can process payments in different currencies.

Personalize

- Test 1: Verify that WeAllocate allows users to customize the following settings: language, currency, date and time format, and theme.
- Test 2: Verify that WeAllocate persists the user's personalized settings across sessions.

Onboard

- Test 1: Verify that WeAllocate provides a guided tour of the product and interactive tutorials on how to complete common tasks.
- Test 2: Observe new users as they complete the onboarding experience and identify any areas where the experience can be improved for clarity and conciseness

WeAllocate support

- Test 1: Verify that WeAllocate provides support through a variety of channels, such as email, live chat, and phone support.
- Test 2: Verify that WeAllocate has a knowledge base of common questions and answers.
- Test 3: Open a support ticket and observe how long it takes to receive a response. Verify that the response is helpful and resolves the issue.

ScreenRead

- Test 1: Verify that WeAllocate is compatible with popular screen readers, such as JAWS and NVDA.
- Test 2: Observe blind and visually impaired users as they use WeAllocate. Identify any areas where the product can be improved for accessibility.

Keyboard

- Test 1: Verify that WeAllocate can be fully navigated using the keyboard.
- Test 2: Observe users with limited mobility or who cannot use a mouse as they use WeAllocate. Identify any areas where the product can be improved for accessibility.

Color Contrast

- Test 1: Verify that WeAllocate meets the WCAG 2.1 AA color contrast requirements.
- Test 2: Observe users with color blindness and low vision as they use WeAllocate. Identify any areas where the product can be improved for accessibility

User Training

- Test 1: Verify that custom training is provided to users upon request.
- Test 2: Observe users as they use WeAllocate to complete tasks after receiving custom training. Identify any areas where the training was not effective. This may indicate that the training needs to be improved.

Training Eval

- Test 1: Verify that WeAllocate collects feedback from users after they complete each training module.
- Test 2: Analyze the feedback to identify areas where the training can be improved.
- Test 3: Implement improvements to the training based on the feedback.

Color

- Test 1: Verify that WeAllocate's color scheme meets the WCAG 2.1 AA color contrast requirements.

- Test 2: Observe users with color blindness as they use WeAllocate. Identify any areas where the product's color scheme makes it difficult for users to read or use the product.

Font

- Test 1: Verify that WeAllocate uses a font that is large enough to be easily readable, and that has sufficient line height and kerning.
- Test 2: Observe users as they read the text in WeAllocate. Identify any areas where the typography makes it difficult for users to read the text.

Survey 2.0

- Test 1: Conduct a user survey to assess users' perceptions of WeAllocate's professionalism. The survey should ask users whether they find WeAllocate to be professional, polished, and credible.
- Test 2: Observe users as they use WeAllocate. Identify any areas where the product's design makes it seem unprofessional or untrustworthy.

Friendliness

- Test 1: Conduct a user survey to assess users' perceptions of WeAllocate's user-friendliness. The survey should ask users how easy they find it to use and navigate WeAllocate.
- Test 2: Observe users as they use WeAllocate to complete tasks. Identify any areas where the product's design makes it difficult for users to complete tasks or understand the product.

Visual

- Test 1: Conduct a user survey to assess users' perceptions of WeAllocate's visual appeal. The survey should ask users how visually appealing and engaging they find WeAllocate.
- Test 2: Observe users as they use WeAllocate. Identify any areas where the product's design makes it visually unappealing or disengaging.

Device

- Test 1: Verify that the software can run properly on the device given.

CRM

- Test 1: Verify that WeAllocate can retrieve and update customer data from the CRM system in real time.
- Test 2: Observe users as they use WeAllocate to interact with customer data. Identify any areas where the integration with the CRM system is not seamless or user-friendly.

Accounting System

- Test 1: Verify that WeAllocate can export financial data to the accounting system in a format that is compatible with the accounting system.
- Test 2: Observe users as they use WeAllocate to export financial data. Identify any areas where the integration with the accounting system is not seamless or user-friendly.

Email

- Test 1: Verify that WeAllocate can send and receive emails using the email system's servers.
- Test 2: Observe users as they use WeAllocate to send and receive emails. Identify any areas where the integration with the email system is not seamless or user-friendly.

Install

- Test 1: Verify that the WeAllocate self-installer can be used to install the product successfully without the need for any special instructions.
- Test 2: Observe untrained users as they install WeAllocate. Identify any areas where the installation process is confusing or difficult.

Ticket

- Test 1: Verify that the WeAllocate support portal is accessible to users.
- Test 2: Submit a ticket to the WeAllocate support team and verify that you receive a response within 24 hours.

App Update

- Test 1: Verify that WeAllocate releases are no larger than 50 MB in size.
- Test 2: Observe users as they upgrade to a new version of WeAllocate. Identify any areas where the upgrade process is disruptive or causes problems for users.

Social Media

- Test 1: Verify that new releases of WeAllocate are announced on the WeAllocate website.
- Test 2: Verify that new releases of WeAllocate are announced on Twitter and Facebook.

Patch

- Test 1: Verify that each new release of WeAllocate includes release notes.
- Test 2: Verify that the release notes for each new release of WeAllocate detail the new features and bug fixes in the release.

Language 2.0

- Test 1: Verify that WeAllocate can be translated into all the supported languages.

- Test 2: Observe users from different language groups as they use WeAllocate. Identify any areas where the language support is inadequate or confusing.

Holidays

- Test 1: Verify that WeAllocate can calculate deadlines and other dates accurately, taking into account public holidays for all countries in the world.
- Test 2: Observe users from different countries as they use WeAllocate to manage their tasks and deadlines. Identify any areas where the product does not accurately account for public holidays.

Integration

- Test 1: Verify that WeAllocate can import and export data from all of the organization's existing systems.
- Test 2: Observe users as they use WeAllocate to import and export data from the organization's existing systems. Identify any areas where the integration is not seamless or user-friendly.

Accessibility

- Test 1: Verify that WeAllocate has been reviewed by an accessibility expert and that it meets all applicable accessibility requirements.
- Test 2: Observe users with disabilities as they use WeAllocate. Identify any areas where the product is not accessible or difficult to use

III Design

21 Design Goals

- **User Simplicity**:
    - **Objective**: To create an interface that is intuitive and easy to navigate for all user groups, including volunteers, staff, and management at the IMAN Food and Wellness Center.
    - **Justification**: Given the diverse technological literacy among users, a simple user interface ensures that all users can efficiently interact with the system without extensive training. This simplicity should extend to all aspects of the application, from dashboard navigation to data input and retrieval.
    - **Implementation Strategy**: Utilize user-centered design principles, conduct usability testing with actual users from IMAN, and incorporate feedback iteratively. The design will prioritize clear visual elements, straightforward navigation, and minimalistic layouts to reduce cognitive load.
- **Security**:

    - **Objective**: To ensure data integrity, confidentiality, and availability. This involves protecting sensitive information related to food bank operations, donor details, and beneficiary information.
    - **Justification**: Security is paramount, considering the sensitive nature of the data handled by WeAllocate. Any breach could compromise the trust of donors, beneficiaries, and the organization.
    - **Implementation Strategy**: Implement robust authentication and authorization mechanisms, encrypt sensitive data, and conduct regular security audits. The system will adhere to best practices in cybersecurity and data protection laws.
- **Flexibility**:
    - **Objective**: To design a system adaptable to changing needs and requirements, including varying scales of operation, different types of resources, and evolving organizational processes.
    - **Justification**: IMAN's Food and Wellness Center's needs may evolve over time, and the system should be able to accommodate these changes without requiring a complete redesign.
    - **Implementation Strategy**: Build a modular system architecture, allowing for components to be updated or replaced independently. Utilize APIs for integration with other systems, ensuring that WeAllocate can adapt to new workflows or data sources.
- **Scalability**:
    - **Objective**: To ensure the system can handle increased loads, whether from a growing number of users, more extensive data processing, or expanded operational scope.
    - **Justification**: As IMAN grows and serves a broader community, WeAllocate must be capable of scaling up to meet increased demands without performance degradation.
    - **Implementation Strategy**: Use cloud-based infrastructure for hosting, leverage scalable database solutions, and design the backend to manage increased traffic and data volumes efficiently. Regular stress testing will be conducted to anticipate and address scalability issues.
- **Efficiency**:
    - **Objective**: To optimize the system's performance in terms of resource allocation computations, data processing speed, and overall operational efficiency.
    - **Justification**: Efficiency in processing and resource allocation is crucial for a system like WeAllocate, where timely and accurate distribution of resources directly impacts community services.
    - **Implementation Strategy**: Optimize algorithms for resource allocation, ensure efficient database queries, and streamline data flows within the system. The use of machine learning models will be fine-tuned for both accuracy and speed, balancing the need for precise predictions with the necessity for quick processing.

22 Current System Design

The existing system at IMAN's Food and Wellness Center primarily revolves around manual processes supplemented by basic digital tools. The current workflow involves several key components:

- **Manual Resource Allocation**:
  - **Description**: Staff and volunteers assess inventory levels, monitor expiration dates, and allocate food based on perceived demand. This process is largely reliant on human judgment and experience.
  - **Tools Used**: Basic inventory tracking is conducted using spreadsheets or simple databases.
  - **Challenges**: The manual nature of the process makes it time-consuming, prone to human error, and challenging to adapt quickly to changing demands.
- **Inventory Management**:
  - **Description**: Physical checks of inventory are conducted to gauge available resources, with data entry into spreadsheets or basic databases.
  - **Challenges**: This method lacks real-time updates and predictive capabilities, leading to potential inefficiencies in resource distribution and challenges in handling sudden spikes in demand.
- **Beneficiary Interaction and Feedback**:
  - **Description**: Direct feedback is collected from beneficiaries during distribution, providing insights into their needs and satisfaction.
  - **Challenges**: Feedback collection is ad-hoc and not systematically integrated into the decision-making process.
- **Donation Management**:
  - **Description**: Donations, both monetary and in-kind, are tracked manually or through simple digital tools.
  - **Challenges**: There is a lack of integration between donation management and resource allocation processes, potentially affecting the optimization of resources.
- **Decision Making**:
  - **Description**: Decisions regarding food allocation are made based on staff and volunteer experiences, historical trends, and immediate past experiences.
  - **Challenges**: Decision-making is largely reactive rather than proactive, without the support of predictive analytics.

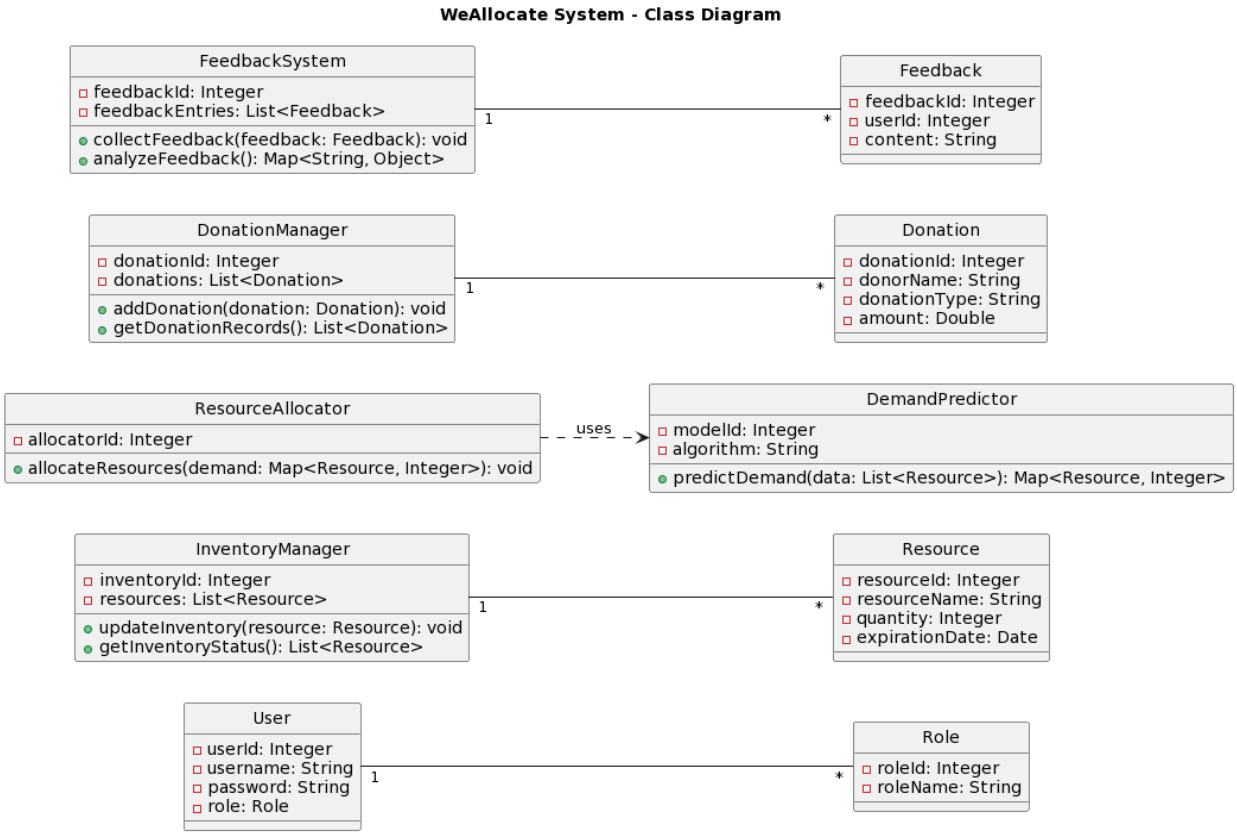
WeAllocate is proposed as a comprehensive solution to replace and significantly upgrade this existing system. By integrating advanced machine learning algorithms and modern digital tools, WeAllocate aims to address the inefficiencies, reduce manual labor, and enhance the precision of demand prediction and resource allocation. The introduction of WeAllocate is anticipated to revolutionize the way IMAN's Food and Wellness Center operates, making it more efficient, responsive, and data-driven in its mission to serve the community.

23 Proposed System Design

23a Initial System Analysis and Class Identification

- **Class User**
  - Attributes:
    - **userId**: An integer representing the unique ID of a user.
    - **username**: A string indicating the user's name.
    - **password**: A string for the user's password.
    - **role**: An instance of the **Role** class, indicating the user's role.
  - Relationship: Each **User** is associated with one or more **Role**s (denoted by "1" -- "*").
- **Class Role**
  - Attributes:
    - **roleId**: An integer for the unique role ID.
    - **roleName**: A string representing the name of the role.
- **Class Resource**
  - Attributes:
    - **resourceId**: An integer for the unique resource ID.
    - **resourceName**: A string for the name of the resource.
    - **quantity**: An integer indicating the amount of the resource.
    - **expirationDate**: A **Date** object representing when the resource expires.
- **Class InventoryManager**
  - Attributes:
    - **inventoryId**: An integer for the inventory's unique ID.
    - **resources**: A list of **Resource** objects.
  - Methods:
    - **updateInventory(resource: Resource)**: A method to update the inventory, returns no value (**void**).
    - **getInventoryStatus()**: Returns a list of **Resource** objects, indicating the current inventory status.
  - Relationship: Each **InventoryManager** is associated with one or more **Resource**s (denoted by "1" -- "*").
- **Class DemandPredictor**
  - Attributes:
    - **modelId**: An integer for the unique model ID.
    - **algorithm**: A string indicating the algorithm used for demand prediction.
  - Methods:
    - **predictDemand(data: List<Resource>)**: Returns a map of **Resource** to integer values, predicting the demand based on the given data.
  - Relationship: The **ResourceAllocator** uses (**..>**) the **DemandPredictor**.
- **Class ResourceAllocator**
  - Attributes:
    - **allocatorId**: An integer for the allocator's unique ID.
  - Methods:
    - **allocateResources(demand: Map<Resource, Integer>)**: A method to allocate resources based on the provided demand, returns no value (**void**).
- **Class DonationManager**
  - Attributes:
    - **donationId**: An integer for the unique donation manager ID.
    - **donations**: A list of **Donation** objects.

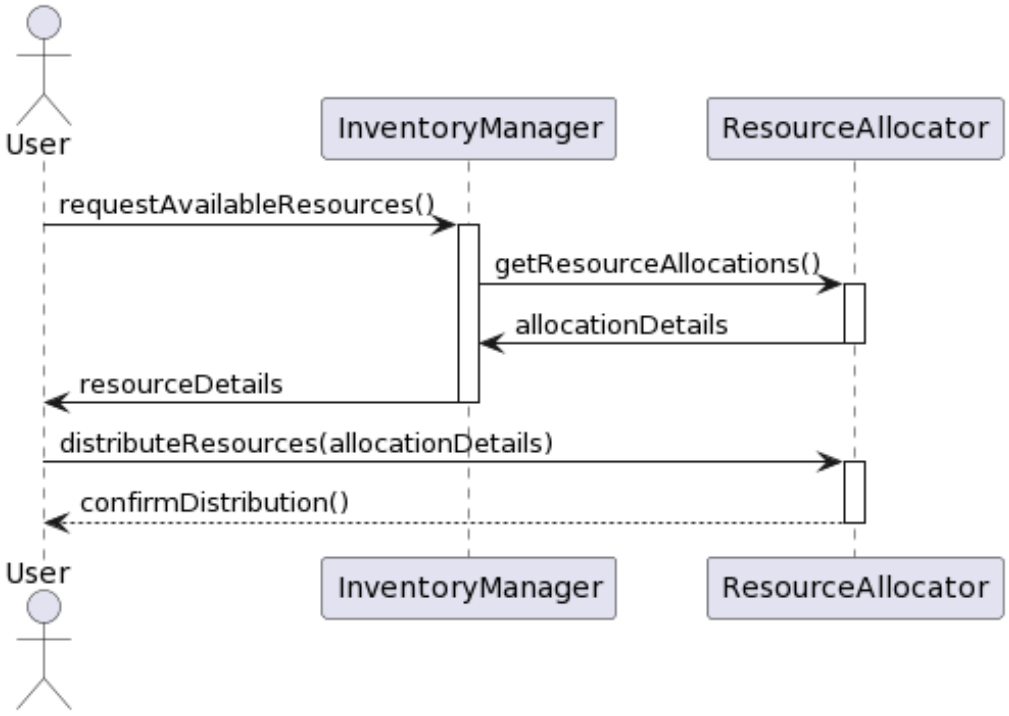
    - Methods:
        - **addDonation(donation: Donation)**: Adds a donation, returns no value (**void**).
        - **getDonationRecords()**: Returns a list of **Donation** objects.
    - Relationship: Each **DonationManager** is associated with one or more **Donation**s (denoted by "1" -- "*").
- **Class Donation**
    - Attributes:
        - **donationId**: An integer for the unique donation ID.
        - **donorName**: A string indicating the name of the donor.
        - **donationType**: A string representing the type of donation.
        - **amount**: A double value indicating the amount of the donation.
- **Class FeedbackSystem**
    - Attributes:
        - **feedbackId**: An integer for the unique feedback system ID.
        - **feedbackEntries**: A list of **Feedback** objects.
    - Methods:
        - **collectFeedback(feedback: Feedback)**: Collects feedback, returns no value (**void**).
        - **analyzeFeedback()**: Returns a map of strings to objects, representing the analysis of feedback.
    - Relationship: Each **FeedbackSystem** is associated with one or more **Feedback** objects (denoted by "1" -- "*").
- **Class Feedback**
    - Attributes:
        - **feedbackId**: An integer for the unique feedback ID.
        - **userId**: An integer representing the ID of the user who gave the feedback.
        - **content**: A string containing the feedback content.


WeAllocate System - Class Diagram
FeedbackSystem
- feedbackId: Integer
- feedbackEntries: List<Feedback>
+ collectFeedback(feedback: Feedback): void
+ analyzeFeedback(): Map<String, Object>
1
*
Feedback
- feedbackId: Integer
- userId: Integer
- content: String
DonationManager
- donationId: Integer
- donations: List<Donation>
+ addDonation(donation: Donation): void
+ getDonationRecords(): List<Donation>
1
*
Donation
- donationId: Integer
- donorName: String
- donationType: String
- amount: Double
ResourceAllocator
- allocatorId: Integer
+ allocateResources(demand: Map<Resource, Integer>): void
uses
DemandPredictor
- modelId: Integer
- algorithm: String
+ predictDemand(data: List<Resource>): Map<Resource, Integer>
InventoryManager
- inventoryId: Integer
- resources: List<Resource>
+ updateInventory(resource: Resource): void
+ getInventoryStatus(): List<Resource>
1
*
Resource
- resourceId: Integer
- resourceName: String
- quantity: Integer
- expirationDate: Date
User
- userId: Integer
- username: String
- password: String
- role: Role
1
*
Role
- roleId: Integer
- roleName: String

23b Dynamic Modelling of Use-Cases

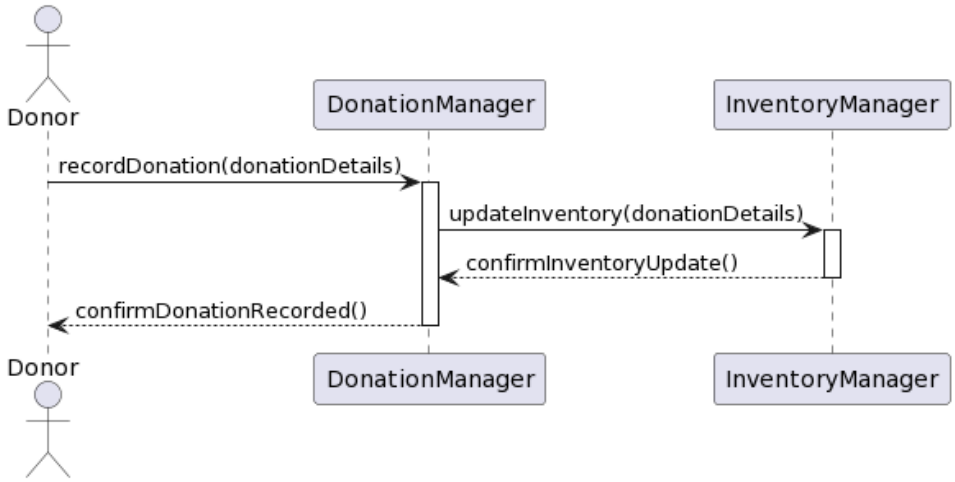
Use Case: User Distributes Items from the Food Bank

In this use case, a user, typically a food bank manager or volunteer, initiates the process of distributing items from the food bank. The user queries the InventoryManager to check the available resources. The InventoryManager, in turn, interacts with the ResourceAllocator to get details on the allocated resources. These allocation details are then passed back to the user. The user proceeds to distribute the resources as per the allocation plan. After distribution, the ResourceAllocator is updated with the distribution confirmation, ensuring that the system accurately reflects the distribution activity. This use case is crucial for the day-to-day operational efficiency of the food bank, ensuring that resources are distributed in a timely and organized manner.



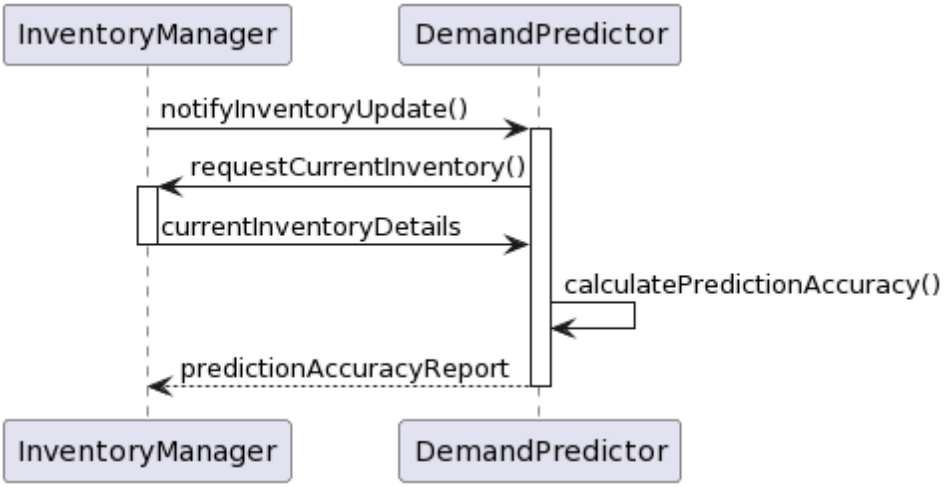
Use Case: Donor Records a Donation

In this scenario, a donor (an individual or an entity) records a donation to the food bank. The donor provides the DonationManager with the details of the donation, which can include both monetary and in-kind contributions. The DonationManager then updates the InventoryManager with the new donation details, effectively increasing the inventory. The InventoryManager acknowledges this update, and the DonationManager confirms back to the donor that the donation has been recorded. This use case is essential for maintaining accurate and up-to-date records of donations, a critical component of food bank operations.



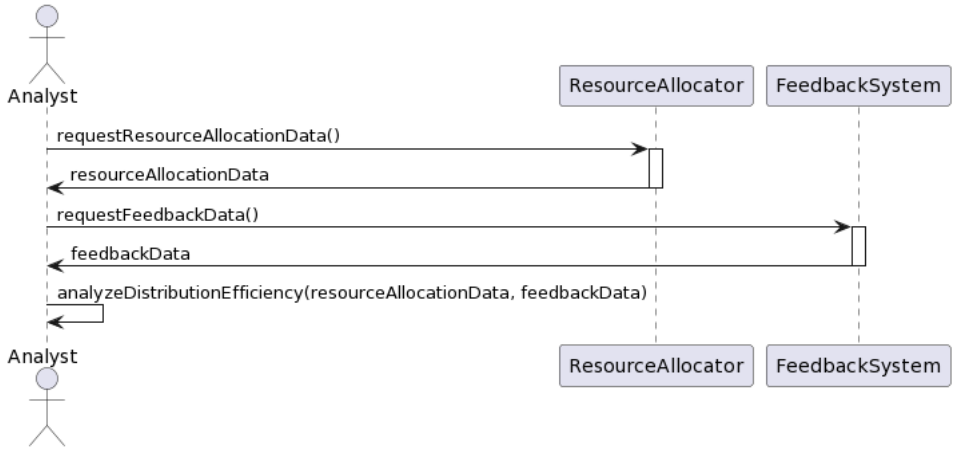
Use Case: Tracking Prediction Accuracy After Inventory Update

This use case focuses on maintaining the accuracy of demand predictions. After an inventory update, the InventoryManager notifies the DemandPredictor of the change. The DemandPredictor requests the current inventory details from the InventoryManager and uses this data to calculate the accuracy of its demand predictions. This process involves comparing predicted demand with actual resource usage. The DemandPredictor then generates a report on prediction accuracy, which is sent back to the InventoryManager. This use case is vital for ensuring the reliability of the demand prediction algorithm, leading to more efficient resource allocation in the future.



Use Case: Analyzing Distribution Efficiency

A supervisor undertakes this use case to evaluate the efficiency of resource distribution within the food bank. The process begins with the Analyst requesting resource allocation data from the ResourceAllocator. This data provides insights into how resources have been allocated and distributed. Simultaneously, the Analyst also requests feedback data from the FeedbackSystem, which gathers beneficiaries' opinions and experiences. Combining these two data sets, the Analyst conducts an analysis to determine the efficiency of the distribution process, identifying areas of success and opportunities for improvement. This use case is key to continuous improvement in the food bank's operations, ensuring that resources reach those in need effectively.



23c Proposed System Architecture

For the WeAllocate project, the proposed software architecture is a **Client-Server Architecture with Microservices**. This architecture is chosen based on several key factors that align with the project's requirements and goals:

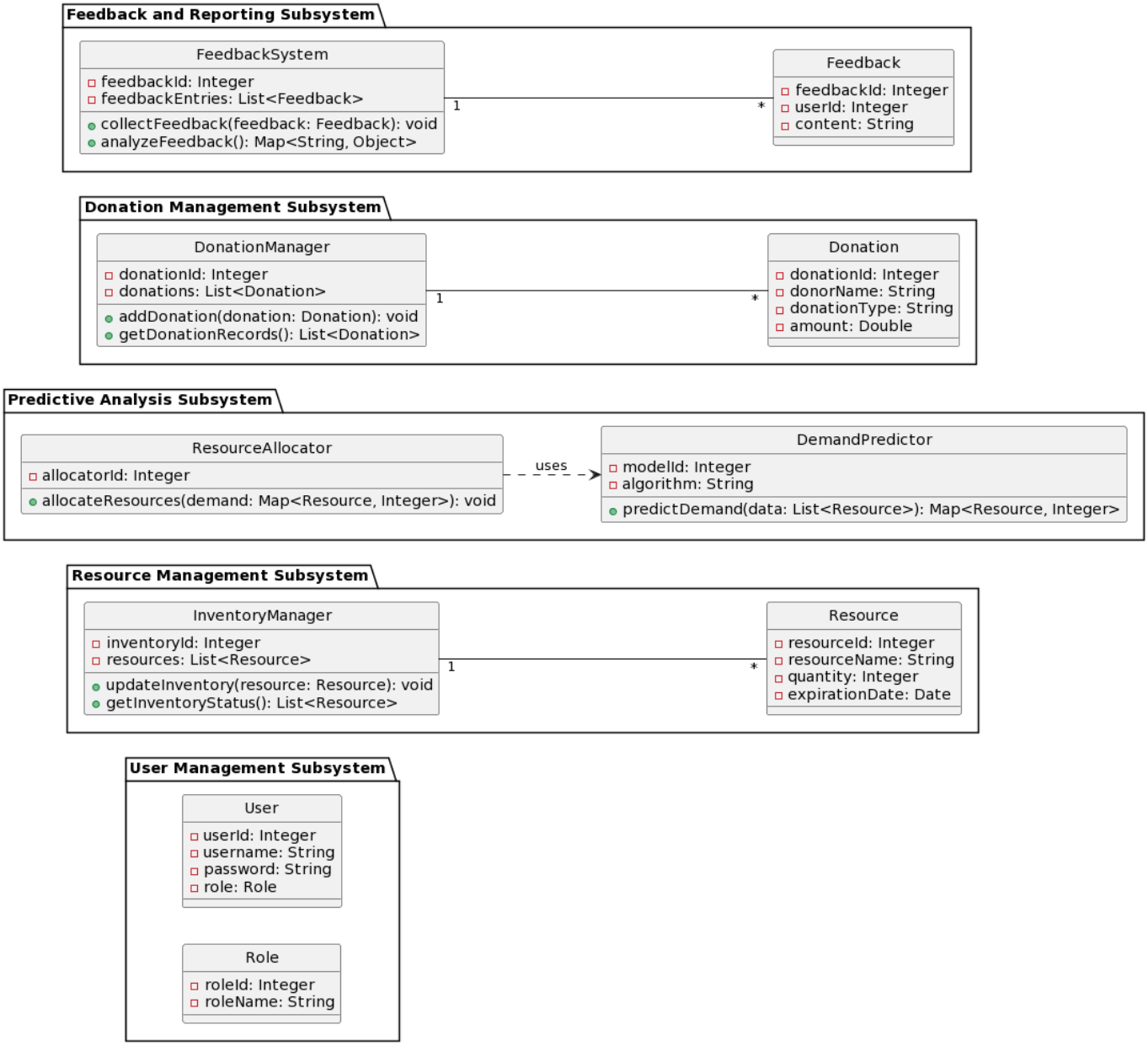
Overview of the Chosen Architecture

- **Client-Server Model**:
  - The system is divided into two main parts: the client (frontend interface) and the server (backend processing and data management).
  - **Client Side**: Provides the user interface where users can interact with the system. This includes food bank managers, volunteers, donors, and analysts.
  - **Server Side**: Handles core functionalities like inventory management, demand prediction, resource allocation, and feedback analysis.
- **Microservices Approach**:
  - The server side is further structured into a set of microservices, each responsible for a specific functionality or domain.
  - For example, separate microservices for Inventory Management, Demand Prediction, Donation Management, and Feedback Analysis.
  - Microservices communicate with each other through well-defined APIs.

Justification for the Choice

The Client-Server Architecture with Microservices is selected for the WeAllocate system to ensure scalability, maintainability, performance, security, and easy integration, aligning with the project's long-term vision and objectives. This architecture will support the diverse and dynamic environment of the IMAN Food and Wellness Center, ensuring the system remains robust and adaptable to future needs.

23d Initial Subsystem Decomposition



1. User Management Subsystem

- **Responsibilities:** This subsystem is responsible for managing all aspects of user data within the WeAllocate system. It handles user identification, authentication, and authorization processes. The system ensures that user roles are clearly defined and managed, allowing for appropriate access control and security measures.

- **Key Components:**

    - **User:** Manages individual user information, including credentials and identification.

- **Role:** Defines different roles within the system, allowing for role-based access and permissions.

2. Resource Management Subsystem

- **Responsibilities:** Central to the inventory management of the food bank, this subsystem tracks and updates the status of various resources. It maintains up-to-date information on resource quantities, types, and expiration dates, ensuring efficient resource allocation and minimizing waste.

- **Key Components:**

  - **Resource:** Represents the food items or other resources managed by the food bank.

  - **InventoryManager:** Handles the inventory data, providing functionalities for updating and retrieving resource information.

3. Predictive Analysis Subsystem

- **Responsibilities:** This subsystem is crucial for forecasting resource demands using data-driven approaches. It uses historical data and predictive models to anticipate future needs, guiding the efficient allocation of resources.

- **Key Components:**

  - **DemandPredictor:** Employs machine learning algorithms to predict future demand for resources.

  - **ResourceAllocator:** Utilizes demand predictions to allocate resources effectively, ensuring optimal distribution.

4. Donation Management Subsystem

- **Responsibilities:** Manages donations received by the food bank, including both monetary contributions and in-kind resources. This subsystem ensures accurate recording of donations and updates the inventory accordingly.

- **Key Components:**

  - **DonationManager:** Oversees the recording and management of all donations.

  - **Donation:** Represents individual donations, encapsulating donor details and donation specifics.

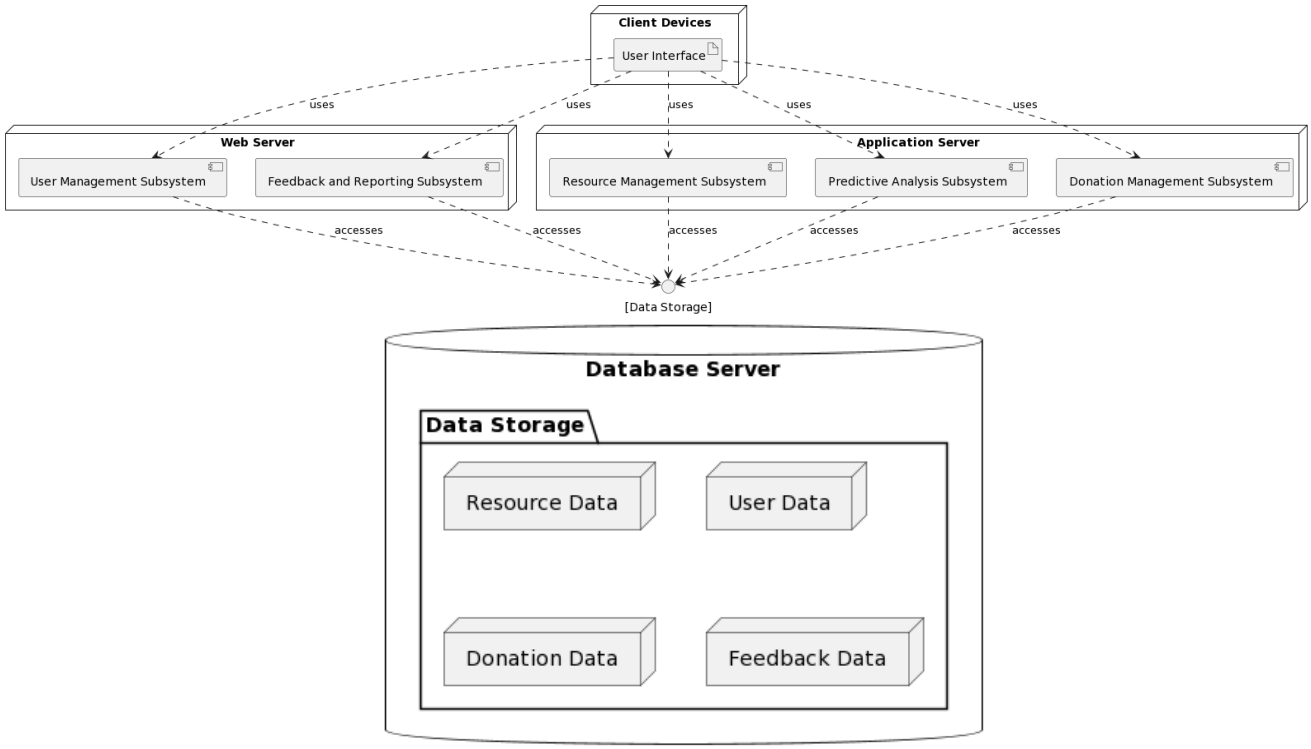
5. Feedback and Reporting Subsystem

- **Responsibilities:** Focused on collecting and analyzing feedback from system users and beneficiaries. This subsystem provides valuable insights into user satisfaction and system performance, driving continuous improvement.

- **Key Components:**
    - **FeedbackSystem:** Gathers and processes feedback from various stakeholders.
    - **Feedback:** Represents individual feedback entries, storing detailed information for analysis.

24 Additional Design Considerations

24a Hardware / Software Mapping



- **Client Devices**: Contains the "User Interface" artifact, representing the front-end part of the system used on various devices (like smartphones, tablets, or PCs).
- **Web Server & Application Server**: These host different components of the WeAllocate system. The Web Server manages user-related interactions and feedback, while the Application Server handles core functionalities like resource management and predictive analysis.
- **Database Server**: Central storage for all critical data, including resources, users, donations, and feedback.
- **Communications**: The "uses" and "accesses" connections show how the User Interface interacts with subsystems and how these subsystems access data storage.

24b Persistent Data Management

- **Database Server Subsystem:**
    - **Responsibilities:** Hosts the Data Storage, which is the primary repository for all persistent data in the system. It ensures data is safely stored and can be retrieved or updated as required.
    - **Key Components:**
        - **Resource Data:** Stores information about the food bank's resources, including quantities, types, and expiration dates.
        - **User Data:** Contains user accounts, authentication details, and role information.
        - **Donation Data:** Records details of all donations received.
        - **Feedback Data:** Captures feedback from system users and beneficiaries.
- **Data Backup and Recovery Mechanism:**
    - **Responsibilities:** Regularly backs up the data from the Database Server to prevent data loss and allows for data recovery in case of system failures.
    - **Implementation:** Automated backup scripts or services, and data replication strategies.
- **Relevant Classes for Persistent Data Management**
    - **InventoryManager:**
        - **Role:** Manages inventory data, ensuring updates are reflected in the Resource Data component of the Database Server.
        - **Data Interaction:** Regularly communicates with the Resource Data to fetch and update inventory status.
    - **Donation Manager:**
        - **Role:** Handles donation records, updating the Donation Data component with new entries.
        - **Data Interaction:** Ensures that every donation is recorded in the Donation Data for persistence.
    - **FeedbackSystem:**
        - **Role:** Gathers feedback and stores it in the Feedback Data component.

        - **Data Interaction:** Processes and archives feedback for analysis and future reference.

    - **User Management Subsystem:**

        - **Role:** Manages user data, including roles and authentication details.

        - **Data Interaction:** Interacts with the User Data component to retrieve and update user information.

- **Conclusion**

  The persistent data management in WeAllocate is designed to ensure that crucial data related to resources, users, donations, and feedback is reliably stored and maintained. The combination of robust database management, regular data backups, and efficient data interaction mechanisms by various system components guarantees data integrity and supports system resilience.

24c Global Software Control

- **Global Software Control Concerns**

    - **System Coordination and Orchestration:**

        - Managing interactions and data flow between various microservices.

        - Ensuring that services work together seamlessly to perform complex tasks.

    - **Error Handling and Exception Management:**

        - Systematic approach to handling errors and exceptions to maintain stability.

        - Logging and notifying relevant stakeholders of critical issues.

    - **Security Management:**

        - Overseeing system-wide security policies.

        - Implementing and enforcing access control and data protection measures.

    - **Performance Monitoring and Optimization:**

        - Monitoring system performance and identifying bottlenecks.

        - Optimizing resource usage and response times.

    - **Data Integrity and Consistency:**

        - Ensuring data remains consistent across various system components.

        - Implementing transactions and rollback mechanisms where necessary.
        - New Classes and Subsystems for Global Software Control
    - **System Orchestrator Class:**
        - **Role:** Acts as a central point for coordinating microservices.
        - **Responsibilities:** Manages service workflows, orchestrates user requests, and ensures the correct sequence of service interactions.
    - **Error Manager Class:**
        - **Role:** Handles global error management and exception handling.
        - **Responsibilities:** Logs errors, triggers alerts, and manages error responses to maintain system integrity.
    - **Security Manager Class:**
        - **Role:** Oversees security across the WeAllocate system.
        - **Responsibilities:** Manages authentication, authorization, and encryption services, ensuring compliance with security protocols.
    - **Performance Monitor Subsystem:**
        - **Components:** Includes monitoring tools and performance analytics.
        - **Role:** Constantly assesses system performance, provides insights into system health, and suggests optimization strategies.
    - **Data Integrity Manager Class:**
        - **Role:** Ensures data consistency across the system.
        - **Responsibilities:** Implements database transactions, manages data locking mechanisms, and oversees rollback operations in case of failures.

- **Conclusion**

  Integrating these classes and subsystems into the WeAllocate architecture is crucial for addressing global software control concerns. The System Orchestrator ensures smooth operation across different microservices, while the Error Manager and Security Manager maintain system reliability and security. The Performance Monitor Subsystem helps in fine-tuning the system for optimal performance, and the Data Integrity Manager guarantees the consistency and reliability of data throughout the application. These components are essential for a resilient, secure, and efficient system that meets the diverse needs of its users.

24d Boundary Conditions

Boundary conditions are critical aspects to consider in the design of the WeAllocate system. They refer to how the system behaves under various start-up and shut-down scenarios, including both normal and abnormal conditions, and how it manages configuration and supporting data files. Addressing these concerns involves introducing specific classes and subsystems to ensure smooth operation under these conditions.

- **Boundary Condition Concerns**
    - **Startup Process:**
        - Ensuring the system initializes correctly with all necessary configurations and data.
        - Loading essential data from databases and setting up the initial state.
    - **Normal Shutdown Process:**
        - Gracefully shutting down the system, ensuring all transactions and operations are properly closed.
        - Saving any necessary state or data that needs to persist between sessions.
- **Abnormal Shutdown Handling:**
    - Detecting system crashes or unexpected shutdowns and ensuring safe recovery.
    - Implementing fail-safes to prevent data corruption and loss.
- **Configuration File Management:**
    - Handling the creation, maintenance, and updating of configuration files.
    - Ensuring that system settings are correctly loaded and applied.
- **Database and Data File Maintenance:**
    - Regular maintenance of databases and data files for performance and integrity.
    - Backup and restoration mechanisms for critical data.
    - New Classes and Subsystems for Boundary Conditions
- **SystemInitializer Class:**
    - **Role:** Manages the startup process.

    - **Responsibilities:** Loads configuration files, initializes database connections, and sets up the initial application state.

- **ShutdownManager Class:**

    - **Role:** Handles the normal shutdown process.

    - **Responsibilities:** Safely terminates processes, saves necessary state, and closes database connections.

- **RecoveryManager Class:**

    - **Role:** Deals with abnormal shutdowns and system recovery.

    - **Responsibilities:** Implements recovery procedures, checks data integrity, and restores the last known good state.

- **ConfigManager Class:**

    - **Role:** Manages configuration files.

    - **Responsibilities:** Reads, writes, and updates configuration settings, ensuring that they are correctly applied throughout the system.

- **DatabaseMaintenance Subsystem:**

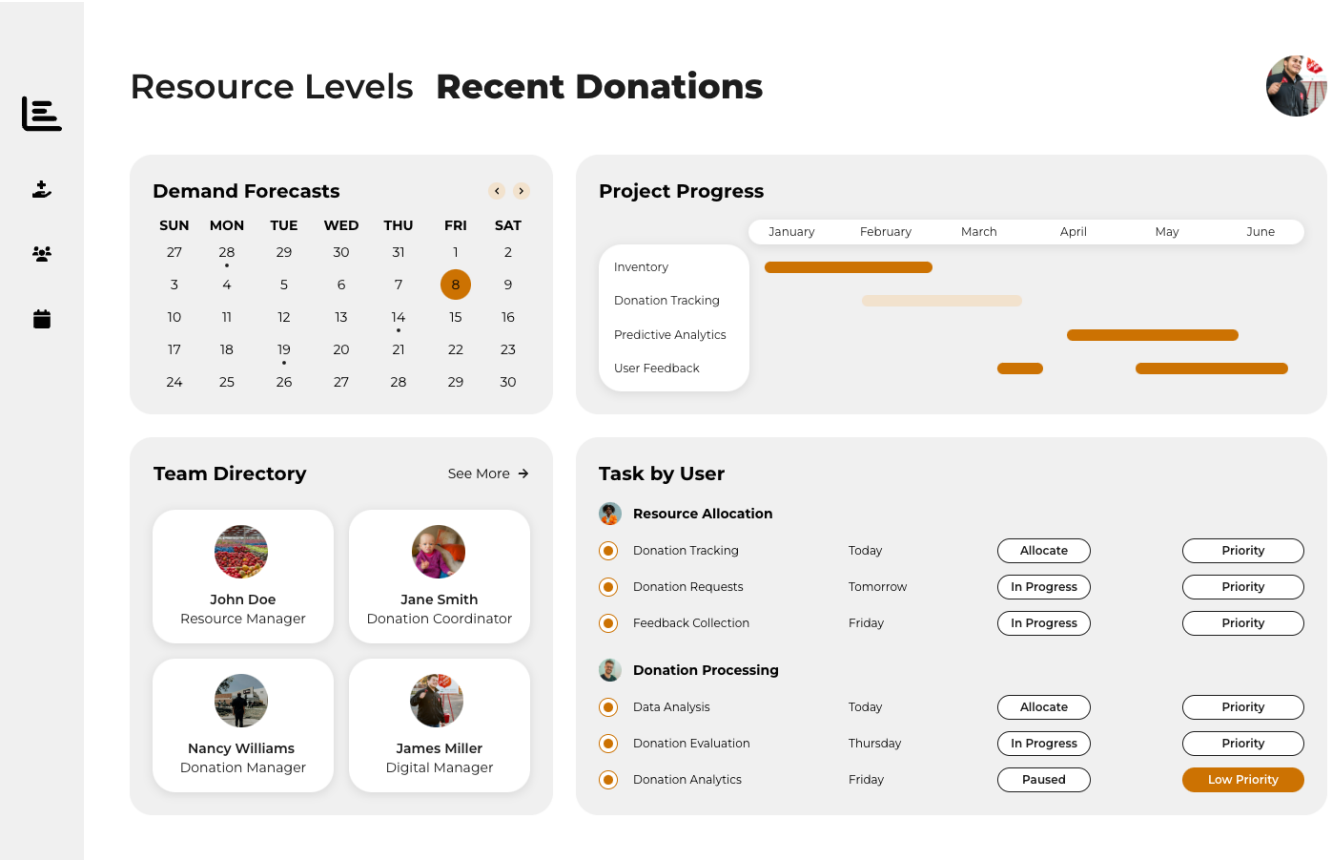
    - **Components:** Includes tools for database optimization, backup, and recovery.

    - **Role:** Regularly maintains the database, performs backups, and ensures data integrity.

- **Conclusion**

  Integrating these classes and subsystems is essential to ensure that WeAllocate can handle various boundary conditions effectively. The SystemInitializer ensures a smooth startup, the ShutdownManager provides a graceful shutdown process, and the RecoveryManager takes charge in case of abnormal terminations. The ConfigManager ensures that all configurations are correctly managed, while the DatabaseMaintenance Subsystem takes care of the underlying data storage. These components together enhance the robustness and reliability of the WeAllocate system, ensuring its resilience in the face of various operational scenarios.

24e User Interface



24f Application of Design Patterns

- **Repository Pattern:**
    - **Application:** Used for data access layers.
    - **Benefits:** Centralizes data access logic, making the code cleaner and more maintainable. It abstracts the data layer, providing a more flexible way to manage database interactions.
- **Factory Pattern:**
    - **Application:** Creating objects without exposing the instantiation logic, particularly useful in the User Management subsystem for creating different types of users or roles.
    - **Benefits:** Enhances flexibility and encapsulation in object creation, making the system more adaptable to changes.
- **Strategy Pattern:**

    - **Application:** Applied in the Demand Predictor subsystem to allow different prediction algorithms to be used interchangeably.

    - **Benefits:** Provides the ability to switch between different algorithms or strategies dynamically, making the system more adaptable and easier to extend.

- **Observer Pattern:**

    - **Application**: Used for implementing notification and alert systems.

    - **Benefits:** Allows various parts of the system to subscribe to and receive updates about certain events (like inventory changes), enhancing responsiveness and interactivity.

- **Decorator Pattern:**

    - **Application:** Enhancing user interface components without modifying their underlying code, especially useful for dynamically adding features or styles.

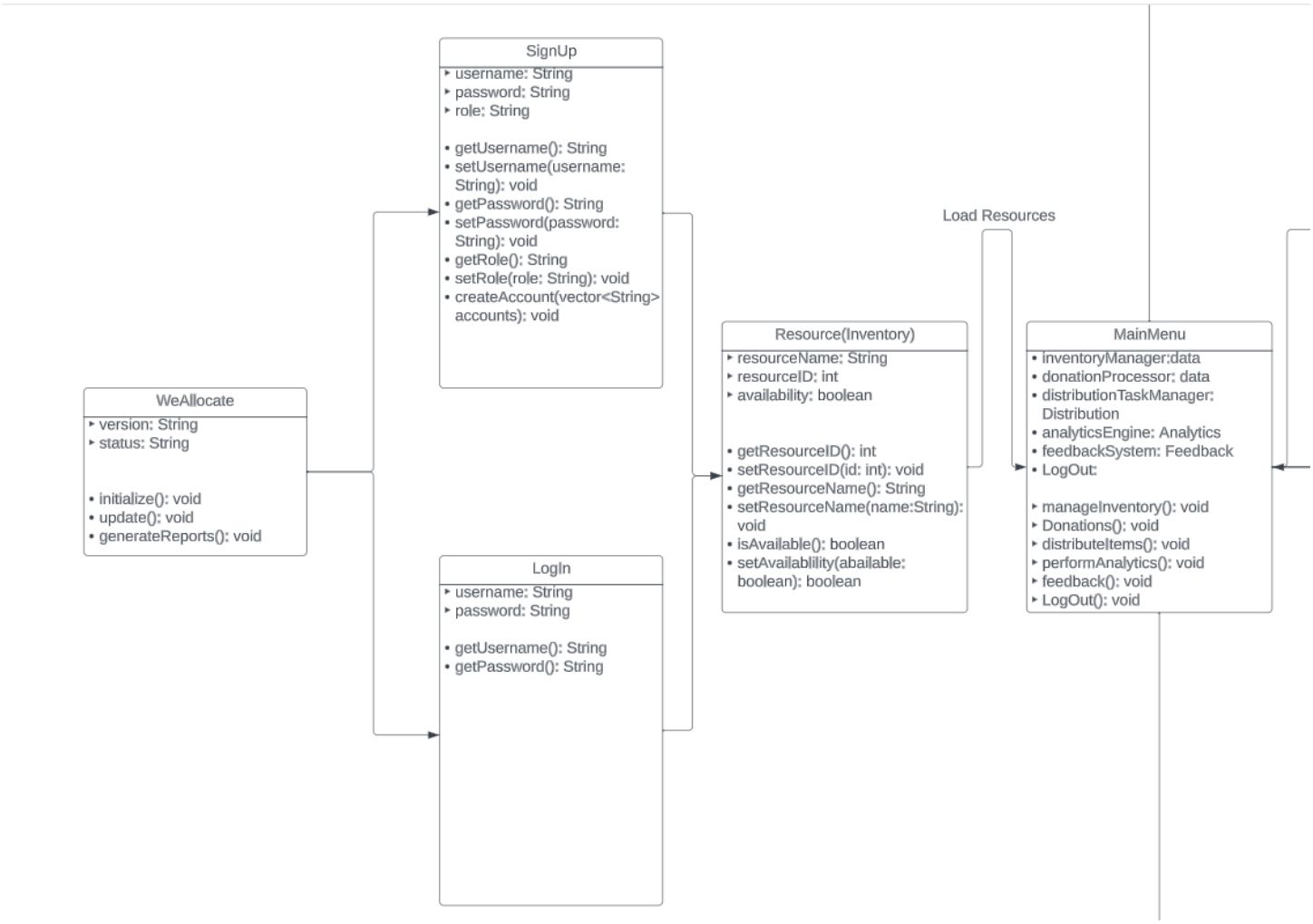
    - **Benefits:** Adds flexibility in UI design and helps in adhering to the Open-Closed Principle, allowing new features to be added without changing existing code.

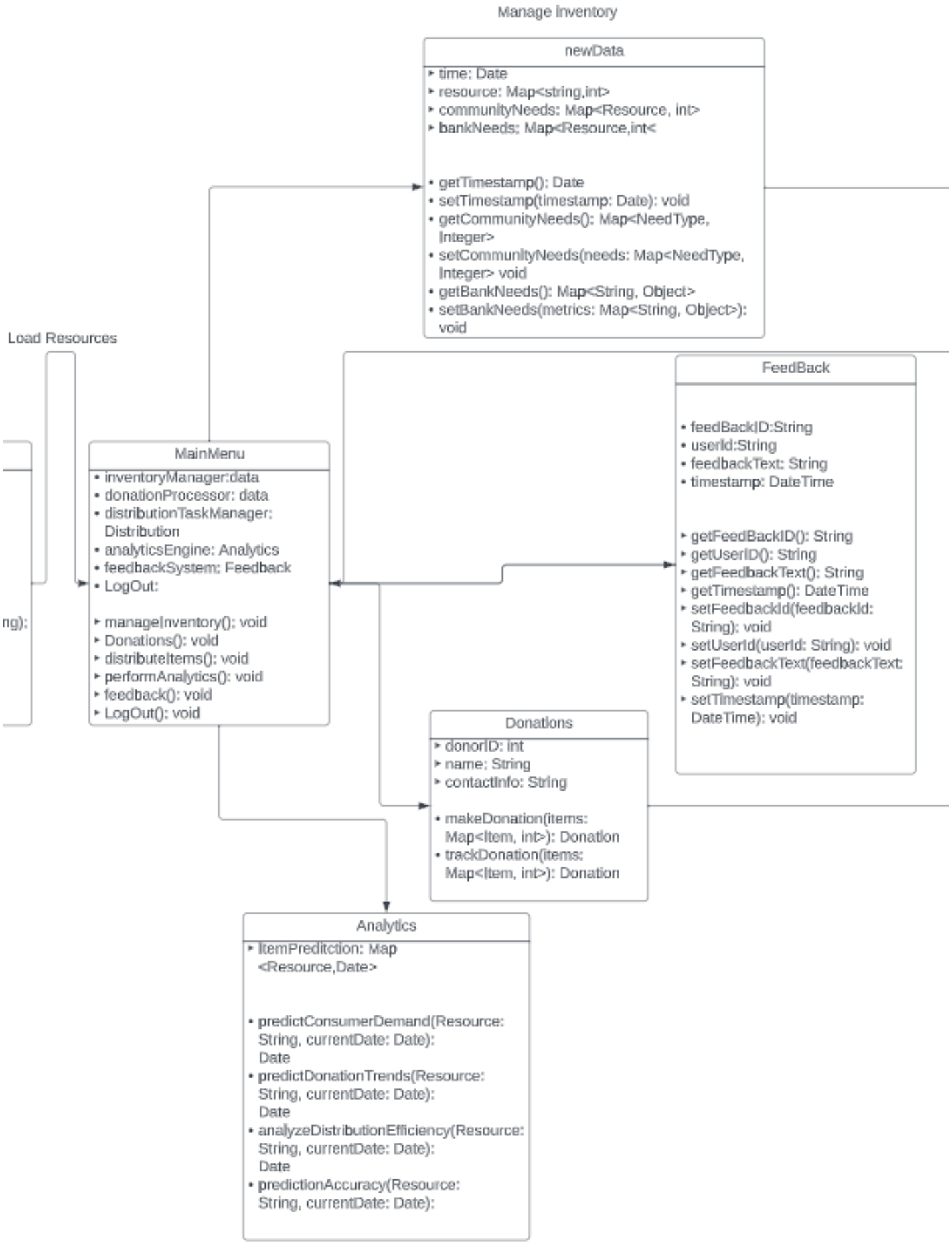
- **Circuit Breaker Pattern:**

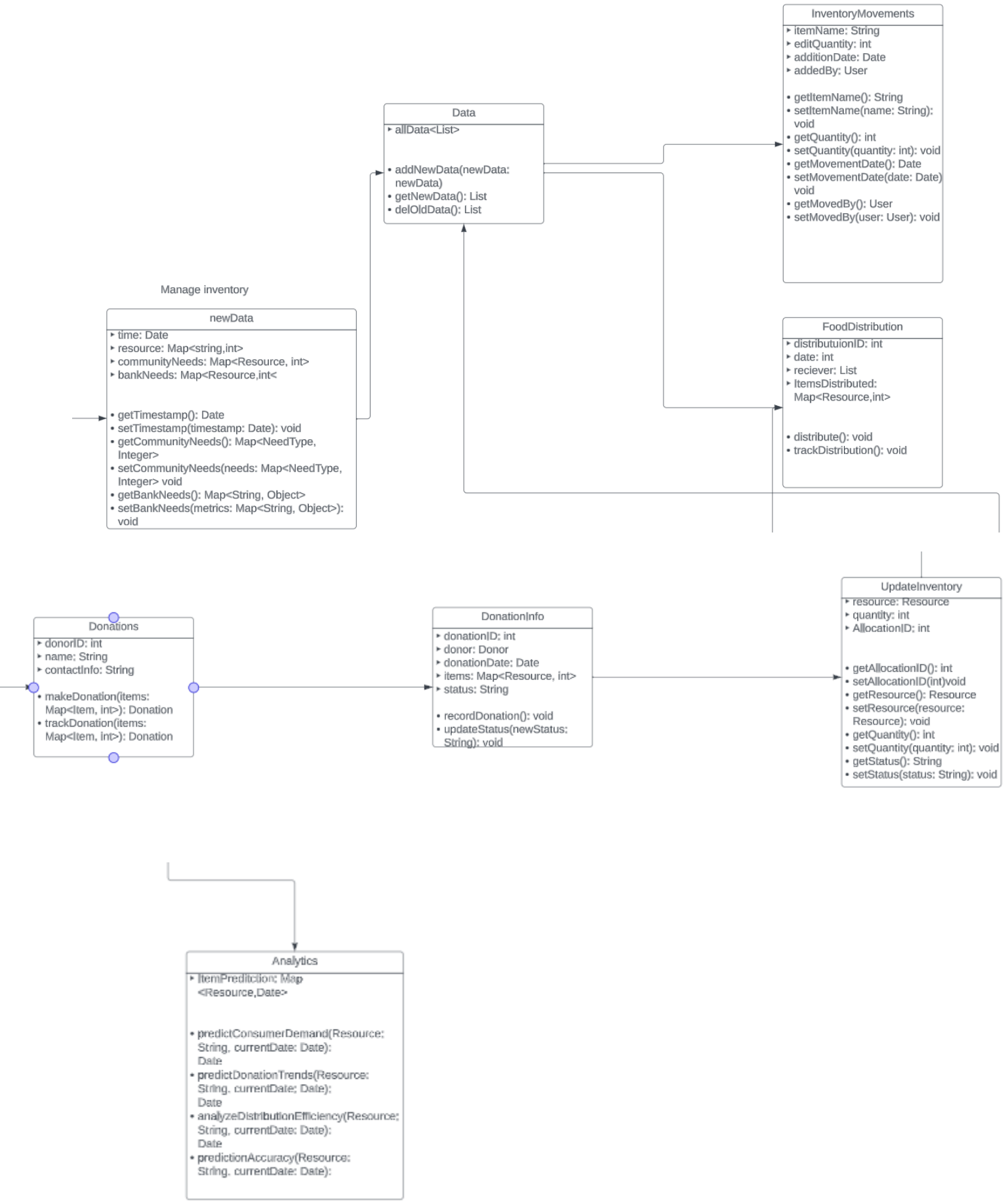
    - **Application:** Used in microservices communication to prevent a network or service failure from cascading to other services.

    - **Benefits:** Enhances the system's resilience and fault tolerance, crucial in distributed architectures like microservices.

25 Final System Design




Manage Inventory
newData
time: Date
resource: Map<string,int>
communityNeeds: Map<Resource, int>
bankNeeds; Map<Resource,int<
getTimestamp(): Date
setTimestamp(timestamp: Date): void
getCommunityNeeds(): Map<NeedType, Integer>
setCommunityNeeds(needs: Map<NeedType, Integer> void
getBankNeeds(): Map<String, Object>
setBankNeeds(metrics: Map<String, Object>): void
Load Resources
MainMenu
inventoryManager:data
donationProcessor: data
distributionTaskManager: Distribution
analyticsEngine: Analytics
feedbackSystem; Feedback
LogOut:
manageInventory(): void
Donations(): void
distributeItems(): void
performAnalytics(): void
feedback(): void
LogOut(): void
FeedBack
feedBackID:String
userId:String
feedbackText: String
timestamp: DateTime
getFeedBackID(): String
getUserID(): String
getFeedbackText(): String
getTimestamp(): DateTime
setFeedbackId(feedbackId: String): void
setUserId(userId: String): void
setFeedbackText(feedbackText: String): void
setTimestamp(timestamp: DateTime): void
Donations
donorID: int
name: String
contactInfo: String
makeDonation(items: Map<Item, int>): Donation
trackDonation(items: Map<Item, int>): Donation
Analytics
ItemPreditction: Map <Resource,Date>
predictConsumerDemand(Resource: String, currentDate: Date): Date
predictDonationTrends(Resource: String, currentDate: Date): Date
analyzeDistributionEfficiency(Resource: String, currentDate: Date): Date
predictionAccuracy(Resource: String, currentDate: Date):


InventoryMovements
itemName: String
editQuantity: int
additionDate: Date
addedBy: User
getItemName(): String
setItemName(name: String): void
getQuantity(): int
setQuantity(quantity: int): void
getMovementDate(): Date
setMovementDate(date: Date) void
getMovedBy(): User
setMovedBy(user: User): void
Data
allData<List>
addNewData(newData: newData)
getNewData(): List
delOldData(): List
Manage inventory
newData
time: Date
resource: Map<string,int>
communityNeeds: Map<Resource, int>
bankNeeds: Map<Resource,int<
getTimestamp(): Date
setTimestamp(timestamp: Date): void
getCommunityNeeds(): Map<NeedType, Integer>
setCommunityNeeds(needs: Map<NeedType, Integer> void
getBankNeeds(): Map<String, Object>
setBankNeeds(metrics: Map<String, Object>): void
FoodDistribution
distributuionID: int
date: int
reciever: List
ItemsDistributed: Map<Resource,int>
distribute(): void
trackDistribution(): void
UpdateInventory
resource: Resource
quantity: int
AllocationID: int
getAllocationID(): int
setAllocationID(int)void
getResource(): Resource
setResource(resource: Resource): void
getQuantity(): int
setQuantity(quantity: int): void
getStatus(): String
setStatus(status: String): void
Donations
donorID: int
name: String
contactInfo: String
makeDonation(items: Map<Item, int>): Donation
trackDonation(items: Map<Item, int>): Donation
DonationInfo
donationID: int
donor: Donor
donationDate: Date
items: Map<Resource, int>
status: String
recordDonation(): void
updateStatus(newStatus: String): void
Analytics
itemPrediction: Map <Resource,Date>
predictConsumerDemand(Resource: String, currentDate: Date): Date
predictDonationTrends(Resource: String, currentDate: Date): Date
analyzeDistributionEfficiency(Resource: String, currentDate: Date): Date
predictionAccuracy(Resource: String, currentDate: Date):

26 Object Design

- **User Management Subsystem**
    - **User Class**
        - **Attributes:** userId: Integer, username: String, password: String, role: Role
        - **Public Methods:** getUserInfo(), changePassword(newPassword: String)
        - **Responsibilities:** Manages user information and authentication.
        - **Constraints:** User passwords are securely hashed and stored.
    - **Role Class**
        - **Attributes:** roleId: Integer, roleName: String
        - **Public Methods:** None (Simple data class)
        - **Responsibilities:** Represents user roles.
    - **UserAuthentication Class**
        - **Attributes:** None
        - **Public Methods:** authenticate(username: String, password: String): User, logout()
        - **Responsibilities:** Handles user authentication and session management.
- **Inventory Management Subsystem**
    - **InventoryManager Class**
        - **Attributes:** inventoryId: Integer, resources: List<Resource>
        - **Public Methods:** updateInventory(resource: Resource), getInventoryStatus(): List<Resource>
        - **Responsibilities:** Manages the inventory of resources.
        - **Constraints:** Ensures resource quantities are updated accurately.
    - **Resource Class**
        - **Attributes:** resourceId: Integer, resourceName: String, quantity: Integer, expirationDate: Date
        - **Public Methods:** None (Simple data class)

        - **Responsibilities:** Represents individual resources in the inventory.
    - **ResourceAllocator Class**
        - **Attributes:** allocatorId: Integer
        - **Public Methods:** allocateResources(demand: Map<Resource, Integer>)
        - **Responsibilities:** Allocates resources based on demand predictions.
- **Donation Management Subsystem**
    - **DonationManager Class**
        - **Attributes**: donmes accurate recording of donations.
    - **Donation Class**
        - **Attributes:** donationId: Integer, donorName: String, donationType: String, amount: Double
        - **Public Methods:** None (Simple data class)
        - **Responsibilities:** Represents individual donations.

26a Packages

- **User Management Package:**
    - **Description:** Contains classes related to user authentication, roles, and permissions.
    - **Classes:** User, Role, UserAuthentication, Permission.
- **Inventory Management Package**:
    - **Description:** Manages the resources and inventory of the system.
    - **Classes:** InventoryManager, Resource, ResourceAllocator.
- **Donation Management Package:**
    - **Description**: Handles the recording and management of donations.
    - **Classes:** DonationManager, Donation.
- **Demand Prediction Package**:
    - **Description:** Provides demand prediction functionality.
    - **Classes:** DemandPredictor, PredictionModel.
- **Feedback and Reporting Package:**
    - **Description:** Deals with user feedback and generates system reports.
    - **Classes:** FeedbackSystem, Feedback, ReportGenerator.
- **Administration Package:**

    - **Description:** Contains classes for system administration and settings.
    - **Classes:** AdminPanel, ConfigurationManager, UserManagement.
- **UI Package:**
    - **Description:** Handles user interface components and interactions.
    - **Classes:** DashboardView, InventoryView, DonationView, PredictionView, FeedbackView, NotificationBar.
- **Utilities Package:**
    - **Description:** Provides utility classes and functions used across the system.
    - **Classes:** Logger, ErrorHandler, ConfigLoader.
- **Communication Package:**
    - **Description:** Manages communication between subsystems or external services.
    - **Classes:** CommunicationManager, RESTClient, MessagingService.
- **Persistence Package:**
    - **Description:** Deals with data storage and retrieval.
    - **Classes:** DatabaseConnector, DataRepository, CacheManager.
- **Security Package:**
    - **Description:** Contains classes related to system security and authorization.
    - **Classes:** SecurityManager, AccessControl, EncryptionProvider.
- **External Integration Package:**
    - **Description:** Handles integration with external services or APIs.
    - **Classes:** ExternalAPIWrapper, ThirdPartyServiceClient.
- **User Management Subsystem**
    - **Responsibilities:** This subsystem is responsible for managing all aspects of user data within the WeAllocate system. It handles user identification, authentication, and authorization processes. The system ensures that user roles are clearly defined and managed, allowing for appropriate access control and security measures.
    - **Key Components:**
        - **User:** Manages individual user information, including credentials and identification.
        - **Role:** Defines different roles within the system, allowing for role-based access and permissions.
- **Resource Management Subsystem**
    - **Responsibilities:** Central to the inventory management of the food bank, this subsystem tracks and updates the status of various resources. It maintains up-to-date information on resource quantities, types, and expiration dates, ensuring efficient resource allocation and minimizing waste.
    - **Key Components:**
        - **Resource:** Represents the food items or other resources managed by the food bank.
        - **InventoryManager:** Handles the inventory data, providing functionalities for updating and retrieving resource information.

- **Predictive Analysis Subsystem**
  - **Responsibilities:** This subsystem is crucial for forecasting resource demands using data-driven approaches. It uses historical data and predictive models to anticipate future needs, guiding the efficient allocation of resources.
  - **Key Components:**
    - **DemandPredictor:** Employs machine learning algorithms to predict future demand for resources.
    - **ResourceAllocator:** Utilizes demand predictions to allocate resources effectively, ensuring optimal distribution.
- **Donation Management Subsystem**
  - **Responsibilities:** Manages donations received by the food bank, including both monetary contributions and in-kind resources. This subsystem ensures accurate recording of donations and updates the inventory accordingly.
  - **Key Components:**
    - **DonationManager:** Oversees the recording and management of all donations.
    - **Donation:** Represents individual donations, encapsulating donor details and donation specifics.
- **Feedback and Reporting Subsystem**
  - **Responsibilities:** Focused on collecting and analyzing feedback from system users and beneficiaries. This subsystem provides valuable insights into user satisfaction and system performance, driving continuous improvement.
  - **Key Components:**
    - **FeedbackSystem:** Gathers and processes feedback from various stakeholders.
    - **Feedback:** Represents individual feedback entries, storing detailed information for analysis.

IV Project Issues

27 Open Issues

As of the current project status, there are several open issues that need to be addressed. These open issues are factors of uncertainty that could have a significant impact on the WeAllocate system's development and operation. It's essential to bring these uncertainties to light to provide input for risk analysis and decision-making. Here are some examples of open issues:

- **System Efficiency:** Is the system good enough to be sustainable, affordable, and easy enough to use for IMAN to not need a technical support team?

- **Requirements Ambiguities:** Some requirements gathered during the initial phase have ambiguities that need to be clarified. Resolving these ambiguities is crucial to ensure that the system meets client expectations.

- **Third-Party Integration:** The system has to be secure yet also be open enough to allow scalability with third-party software/applications

- **Supplier Reliability:** Will the system be able to secure the data of suppliers/donors successfully?

28 Off-the-Shelf Solutions

- **Third-Party Libraries:** We are exploring the use of third-party libraries and frameworks that can expedite the development process. This includes libraries for data analysis, machine learning, and user interface design.

- **Database Management** Systems: Instead of building a custom database management system, we are considering the use of established database solutions like MySQL or PostgreSQL to efficiently manage and store data.

- **Authentication and Authorization:** Utilizing existing authentication and authorization services and frameworks can enhance the security of the system. Solutions like OAuth or OpenID Connect may be viable options.

- **Resource Tracking Tools:** There are commercial resource tracking tools available in the market that offer comprehensive features for managing inventory and resources. We are evaluating whether any of these tools can be integrated into our system.

- **User Interface Frameworks:** We are exploring user interface design frameworks and UI component libraries that can simplify the development of the user interface while ensuring a responsive and user-friendly experience.

- **Machine Learning Models:** Instead of building predictive models from scratch, we are investigating pre-trained machine learning models and libraries that can be fine-tuned for our resource demand prediction needs.

- **Feedback and Analytics Services:** Off-the-shelf feedback collection and analytics services can be leveraged to streamline the gathering and analysis of user feedback, enabling data-driven improvements.

- **Hardware Components:** We are considering the use of off-the-shelf hardware components for certain aspects of the system, such as sensors for monitoring resource conditions.

28a Ready-Made Products

While it may not be possible to make a definitive determination at this stage, considering these products is essential to evaluate their potential suitability. Here are some products that should be investigated:

- **Inventory Management Software:** There are commercial inventory management software solutions available, such as "InventoryPro" and "Zoho Inventory." These products offer comprehensive features for tracking and managing resources.

- **Machine Learning Platforms:** We are looking into machine learning platforms like "TensorFlow" and "scikit-learn" that provide pre-built machine learning algorithms and models that can be integrated into our predictive analysis subsystem.

- **User Feedback Tools:** Products like "SurveyMonkey" and "Qualtrics" offer user feedback collection and analysis capabilities. Investigating these tools may simplify the implementation of the Feedback and Reporting Subsystem.

- **Authentication Services:** Commercial authentication services like "Auth0" and "Okta" provide robust identity and access management solutions. These services can enhance the security of our User Management Subsystem.

- **Database Management Systems:** Established database management systems such as "Oracle Database" and "Microsoft SQL Server" are available for efficient data storage and retrieval.

- **Resource Tracking Hardware:** Some off-the-shelf sensor systems, like "RFID-based inventory tracking systems," can be considered for monitoring resource conditions.

  It's important to note that while these products offer potential solutions, a thorough evaluation of their compatibility with our project requirements, cost-effectiveness, and customization options is necessary. Additionally, we should consider any products that must not be used due to specific project constraints or objectives. Conducting surveys and feasibility studies on these products will be part of our ongoing assessment.

28b Reusable Components

In the context of the WeAllocate project, we are actively exploring the potential of reusable components, which include libraries, toolkits, or components that can be incorporated into our system to expedite development and reduce redundancy. Reusing existing components aligns with the principle of efficiency and avoids reinventing the wheel. Here are some candidate reusable components:

- **Logging Library:** We are considering the integration of a logging library, such as "Log4j" or "SLF4J," to streamline logging and error handling processes within the system.

- **User Authentication Toolkit:** Utilizing a user authentication toolkit, like "Passport.js" for Node.js applications, can simplify user authentication and authorization processes in the User Management Subsystem.

- **Database ORM Framework:** We are exploring Object-Relational Mapping (ORM) frameworks like "Hibernate" (for Java) or "Entity Framework" (for .NET) to enhance database interaction and data management in the system.

- **UI Component Library:** To expedite user interface development, we are evaluating UI component libraries such as "Bootstrap" or "Material-UI" that offer pre-designed UI elements and styles.

- **Data Analysis Libraries:** Reusable data analysis libraries like "Pandas" (for Python) or "Apache Commons Math" (for Java) may be integrated into the Predictive Analysis Subsystem to facilitate data manipulation and modeling.

- **Feedback Collection Widget:** We are considering the incorporation of a feedback collection widget or library that simplifies the process of gathering user feedback in the Feedback and Reporting Subsystem.

- **Resource Tracking SDK:** If available, a Resource Tracking Software Development Kit (SDK) could be utilized to interface with resource monitoring hardware and sensors.

- **Security Framework:** We are investigating security frameworks like "Spring Security" (for Java) or "OWASP Security Libraries" to enhance the security measures in the User Management Subsystem.

28c Products That Can Be Copied

- **Open-Source Authentication Module:** We are exploring open-source authentication modules like "Passport.js" for Node.js. By copying and customizing its authentication logic, we can accelerate the implementation of user authentication in the User Management Subsystem.

- **Reusable UI Components:** Investigating open-source UI component libraries such as "Bootstrap" or "Material-UI" for user interface development. We can copy and adapt UI components to maintain a consistent and visually appealing interface in the User Interface Subsystem.
- **Data Analytics Framework:** Leveraging open-source data analytics frameworks like "TensorFlow" or "Scikit-learn" for predictive analysis in the Predictive Analysis Subsystem. We can adapt machine learning models and algorithms to forecast resource demands accurately.
- **Feedback Collection Widget:** Exploring open-source feedback collection widgets or plugins that can be easily integrated into web applications. By copying and configuring such widgets, we can simplify feedback gathering in the Feedback and Reporting Subsystem.
- **Inventory Management Template:** If available, using existing inventory management templates or solutions from previous projects within our organization as a basis for the Resource Management Subsystem. We can modify and enhance these templates to suit the specific needs of WeAllocate.
- **Database Schema:** Reusing database schemas and structures from past projects with similar data requirements. This can expedite the setup of the database management system in the Resource Management Subsystem.
- **Resource Tracking Algorithms:** Exploring algorithms and resource tracking methods used in other resource management systems. We can adapt and implement these algorithms to optimize resource allocation in the Resource Management Subsystem.

- **Security Framework Configuration:** Utilizing configuration settings and security policies from previous projects that employed security frameworks like "Spring Security." We can copy and tailor these configurations to enhance security in the User Management Subsystem.

29 New Problems

29a Effects on the Current Environment

- **Operational Changes:** The introduction of the WeAllocate system may necessitate changes in the way food bank staff members perform their tasks. For example, the user roles and permissions defined in the User Management Subsystem may alter access to certain functionalities, impacting daily operations.

- **Training Requirements:** Training sessions may be required to familiarize existing staff with the new system's features and functionalities. This could temporarily affect productivity as employees adapt to the changes.

- **Integration with Existing Systems:** We should ensure that the WeAllocate system seamlessly integrates with any existing systems or databases used by the food bank. Incompatibilities could disrupt workflows and data synchronization.

- **Data Migration:** The migration of data from the current inventory management system to WeAllocate may pose challenges. Ensuring data integrity and accuracy during the transition is critical to avoid discrepancies.

- **User Feedback and Acceptance:** User acceptance testing and feedback collection will be essential to identify and address any issues with the new system's usability and performance. Feedback should be gathered from both internal staff and external beneficiaries.

- **Resource Allocation:** The Resource Management Subsystem's changes may impact how resources are allocated and distributed. We should monitor this to ensure that resources reach their intended recipients efficiently.

- **System Performance:** The WeAllocate system must be designed to handle the expected load without degrading performance. Any performance issues could disrupt operations and negatively affect user satisfaction.

- **Security and Privacy:** Security measures must be robust to protect sensitive user data and prevent unauthorized access. Any security breaches could have serious consequences for the food bank and its beneficiaries.

- **Maintenance and Support:** Establishing a system for ongoing maintenance and support is essential to address issues promptly and ensure the system's reliability.

- **Stakeholder Communication:** Effective communication with all stakeholders, including staff, donors, and beneficiaries, will be crucial to manage expectations and address any concerns or resistance to change.

29b Effects on the Installed Systems

- **Data Exchange Interfaces:** WeAllocate will likely need to exchange data with existing systems, such as inventory databases or donor management software. Designing well-defined data exchange interfaces and protocols will be essential to ensure seamless data flow.

- **Compatibility:** Compatibility with existing hardware and software components should be assessed. This includes ensuring that WeAllocate can run on the current server infrastructure and that client devices can access the system without compatibility issues.

- **API Integration:** If there are other systems used by the food bank that provide APIs (Application Programming Interfaces), WeAllocate should be designed to integrate with these APIs to enable data sharing and functionality synchronization.

- **Interoperability:** WeAllocate should be tested for interoperability with commonly used web browsers, operating systems, and mobile devices. Any issues related to platform compatibility should be addressed.

- **Performance Impact:** The new system's deployment should not significantly impact the performance of existing systems. Load testing and performance monitoring should be conducted to ensure that WeAllocate does not overload the infrastructure.

- **Security Implications:** Security measures should be in place to protect sensitive data during data exchange between WeAllocate and existing systems. Encryption, authentication, and access control mechanisms should be considered.

- **Error Handling:** Robust error-handling mechanisms should be implemented to handle communication failures or data inconsistencies between WeAllocate and other systems. Graceful degradation should be a part of the design.

- **Data Synchronization:** In cases where data needs to be synchronized between WeAllocate and existing systems, strategies for data consistency and conflict resolution should be defined.

- **Scalability:** The WeAllocate system should be designed with scalability in mind to accommodate future growth without causing disruptions to existing systems.

- **Documentation:** Thorough documentation of interfaces, protocols, and integration points is crucial for system administrators and support staff to manage the integrated environment effectively.

29c Potential User Problems

- **User Training:** If the WeAllocate system introduces a significant change in workflow or user interface, there may be a need for user training. Users may experience frustration or difficulty adapting to the new system without proper training and support.

- **User Resistance:** Users who are accustomed to existing processes may resist the adoption of the new system. Resistance to change is common, and addressing user concerns and providing clear communication about the benefits of WeAllocate is essential.

- **User Experience:** The user experience (UX) of the WeAllocate interface should be carefully designed to ensure ease of use. Poorly designed interfaces can lead to user frustration and reduced productivity.

- **Accessibility:** Ensure that the WeAllocate system is accessible to users with disabilities. Failure to provide accessible features may lead to exclusion and potential legal issues.

- **Performance:** Users may experience frustration if the system's performance is slow or if there are frequent downtimes. Performance optimization and robust infrastructure are essential to mitigate this issue.

- **Data Privacy and Security:** Users may be concerned about the privacy and security of their data in the new system. Clear privacy policies and security measures should be in place to address these concerns.

- **Feedback Mechanism:** Implement a feedback mechanism within WeAllocate to allow users to report issues, suggest improvements, and provide general feedback. Addressing user feedback can improve the overall user experience.

- **Communication:** Transparent and effective communication with users about the implementation of WeAllocate is vital. Keep users informed about the changes, benefits, and any potential disruptions.

- **User Support:** Provide adequate user support channels, such as helpdesk or online documentation, to assist users in case of issues or questions.

- **User Empowerment:** Involve users in the decision-making process and gather their input during the design and implementation phases to ensure that the system aligns with their needs and preferences.

29d Limitations in the Anticipated Implementation Environment That May Inhibit the New Product

- **Power Supply Reliability:** Ensure that the expected location for deploying the WeAllocate system has a reliable power supply. Unstable or intermittent power sources could lead to system downtime and data loss. Implement backup power solutions if necessary.

- **Environmental Conditions:** Consider the environmental conditions of the deployment area. Extreme weather conditions, such as high humidity, temperature fluctuations, or exposure to the elements, may affect hardware components. Use appropriate enclosures or protective measures to safeguard the system.

- **Physical Space Constraints:** Assess whether the physical space available for deploying the hardware components of the WeAllocate system is sufficient. Ensure that the size and dimensions of the equipment fit within the designated space.

- **Network Connectivity:** Check the availability of stable and high-speed network connectivity at the deployment location. Inadequate network infrastructure can hinder the system's ability to communicate with remote servers or access data in real-time.

- **Security Concerns:** Evaluate the security of the implementation environment to safeguard against unauthorized access or tampering with the system. Implement physical security measures as needed.

- **Environmental Impact:** Consider the environmental impact of the system, such as power consumption and electronic waste. Implement energy-efficient components and responsible disposal practices.

- **Interference and Compatibility:** Assess the potential for electrical interference, radiation, or compatibility issues with other equipment in the environment. Conduct compatibility tests and implement shielding if necessary.

- **Scalability:** Ensure that the implementation environment can accommodate future scalability requirements. Anticipate growth in data volume and user load and plan for expansion as needed.

- **Maintenance Accessibility:** Ensure that the WeAllocate system components are accessible for maintenance and upgrades. Consider ease of access for technicians and the availability of spare parts.

29e Follow-Up Problems

- **Scaling Challenges:** Assess whether the WeAllocate system can effectively scale to accommodate a growing user base and increasing data volumes. Consider load balancing, server capacity, and database scalability to prevent performance bottlenecks.

- **Security Threats:** Continuously monitor and address evolving security threats and vulnerabilities. Implement regular security audits and updates to protect user data and system integrity.

- **Regulatory Compliance:** Stay informed about changes in relevant regulations and compliance requirements. Ensure that the WeAllocate system remains compliant with data privacy, security, and other regulatory standards.

- **User Feedback and Adaptation:** Continuously gather user feedback and adapt the system based on user needs and preferences. Regularly update the user interface and functionality to enhance user satisfaction.

- **Technological Obsolescence:** Plan for the potential obsolescence of hardware or software components used in the system. Keep abreast of technological advancements and consider system upgrades or migrations as needed.

- **Competitive Landscape:** Monitor the competitive landscape and be prepared to adapt to changes in the market. Stay agile and innovative to maintain a competitive edge.

- **Data Management and Retention:** Develop a data management strategy that includes data retention policies and archival procedures. Ensure that historical data remains accessible and compliant with legal requirements.

- **Resource Availability:** Continuously assess the availability of critical resources, such as skilled personnel and infrastructure. Plan for resource gaps and invest in training and development.

- **Business Continuity:** Develop a robust business continuity and disaster recovery plan to ensure minimal disruption in the event of unforeseen disasters or system failures.

- **User Training and Support:** Provide ongoing user training and support to address user inquiries, issues, and challenges. Maintain a responsive support system to assist users effectively.

- **Integration with External Systems:** Regularly test and update integrations with external systems or APIs to ensure seamless communication and data exchange.

- **Feedback Implementation:** Act on user feedback promptly and efficiently to address concerns and improve the system's functionality and usability.

30 Migration to the New Product

30a Requirements for Migration to the New Product

- **Requirements for Migration to the New Product:**

    - **Phased Implementation:** The migration to the WeAllocate system will be conducted in a phased manner to ensure a smooth transition. Each phase will have specific objectives and deliverables.

    - **Data Conversion:** Data from the existing system must be accurately converted and migrated to the WeAllocate system. This includes user accounts, resource information, donation records, and feedback data.

    - **Parallel Operation:** The WeAllocate system will operate in parallel with the existing system for a defined period. This parallel operation will allow users to become familiar with the new system while ensuring that critical operations continue without disruption.

    - **Timetable:** A detailed timetable for the implementation phases, data migration, and parallel operation will be established. The timetable will outline specific milestones and deadlines.

- **Considerations for Migration:**

- **Phased Implementation:** The phased implementation will involve breaking down the migration process into manageable stages. Each phase will be defined by its scope and objectives. The order of migration will prioritize critical functionalities and user groups.

- **Data Conversion:** Special programs or scripts may be required to extract, transform, and load data from the existing system into the WeAllocate database. These programs should ensure data integrity and consistency during the migration process.

- **Manual Backup:** During the migration, a manual backup and recovery plan will be in place to address any unforeseen issues or data discrepancies. This backup plan will ensure that essential data is not lost during the transition.

- **Component Deployment:** The major components of the WeAllocate system, including user management, resource management, predictive analysis, donation management, and feedback systems, will be put in place according to the defined timetable.

- **Parallel Operation:** The parallel operation of the WeAllocate system and the existing system will involve ensuring that data synchronization occurs seamlessly. Users will have access to both systems during this period.

- **Staffing:** Adequate staffing and training will be provided to support the migration process. Staff members will be trained to use the new system and assist users during the transition.

- **Decommissioning:** A plan for decommissioning the old system will be developed. This plan includes archiving necessary data, shutting down old servers, and ensuring a clean exit from the existing system.

- **User Communication:** Effective communication with users is crucial during the migration. Users will be informed about the transition plan, training sessions, and the timeline for full adoption of the WeAllocate system.

- **Data That Has to Be Modified or Translated for the New System**

    - **Data Translation Tasks:**

        - **User Account Data:** User account information, including usernames and roles, will be translated from the existing user database to the WeAllocate user management system. User passwords may need to be encrypted using the new system's encryption methods.

        - **Resource Information:** Data related to food bank resources, such as resource names, quantities, and expiration dates, will be migrated from the current resource database to the WeAllocate resource management system. Data format and units may require translation.

- **Donation Records:** Donation records, which include donor names, donation types, and amounts, will be transferred from the existing donation database to the WeAllocate donation management system. Donation categorization may need to be adjusted to align with the new system's classifications.

- **Feedback Data:** Feedback entries, including user IDs and content, will be moved from the current feedback database to the WeAllocate feedback and reporting subsystem. Data structure and formatting may require modification.

- **Technology Description:**

  - **Current Technology:** The data is currently stored in various databases and systems that are part of the existing infrastructure. Each database may use different data formats and storage mechanisms.

  - **New Technology:** The WeAllocate system utilizes a modern database management system with a defined schema and data storage structure. The new technology offers improved data organization and retrieval capabilities.

- **Data Translation Tasks Description:**

  - User account data will undergo a translation process to ensure that user IDs and roles are accurately mapped to the new system's user management structure.

  - Resource information will be reformatted to fit the data model of the WeAllocate resource management system. Expiration dates may require adjustment to the new date format.

  - Donation records will be migrated while ensuring that donor names and donation types align with the WeAllocate system's conventions.

  - Feedback data will be transferred, and content may be reformatted or sanitized to adhere to the WeAllocate feedback system's standards.

- **Foreseeable Problems:**

  - Data consistency and integrity must be maintained during the translation process to avoid errors or data loss.

  - Handling encrypted passwords during user account migration requires careful consideration of encryption methods and key management.

  - Data format discrepancies between the old and new systems may necessitate data transformation scripts or tools.

- By addressing these data translation tasks and considering potential issues, the WeAllocate project aims to ensure a seamless transition of essential data to the new system while preserving data accuracy and integrity.

31 Risks

- **High Probability, High Impact Risks:**
    - **Data Security Breach (Probability: Moderate, Impact: High):** A security breach leading to unauthorized access to sensitive user data could have a severe impact on the project's reputation and user trust.
    - **Resource Shortage (Probability: Moderate, Impact: High):** Insufficient availability of essential resources, such as server capacity or skilled personnel, may lead to project delays and increased costs.
- **High Probability, Moderate Impact Risks:**
    - **Scope Creep (Probability: High, Impact: Moderate):** Continuous changes in project requirements and scope may disrupt the project timeline and lead to increased development efforts.
    - **Technical Challenges (Probability: High, Impact: Moderate):** Complex technical issues or integration challenges could require additional development time and resources.
- **Moderate Probability, High Impact Risks:**
    - **Regulatory Compliance (Probability: Moderate, Impact: High):** Changes in regulatory requirements related to food bank operations may necessitate significant adjustments to the system, potentially causing delays.
- **Low Probability, High Impact Risks:**
    - **Natural Disasters (Probability: Low, Impact: High):** Natural disasters, such as earthquakes or floods, could disrupt project operations if data centers or critical infrastructure are affected.
- **Low Probability, Moderate Impact Risks:**
    - **Vendor Dependence (Probability: Low, Impact: Moderate):** Dependency on specific vendors for hardware or software components may lead to delays or complications if the vendor faces issues.
- **Mitigation Strategies:**

To address these risks, the WeAllocate project team will implement the following mitigation strategies:

- **Security Measures:** Implement robust security protocols and regularly conduct security audits to minimize the risk of data breaches.
- **Resource Planning:** Develop a resource management plan to ensure the availability of necessary resources and personnel throughout the project.
- **Change Control:** Establish a change control process to manage scope changes effectively and prevent scope creep.
- **Technical Expertise:** Maintain a team of skilled developers and IT professionals to address technical challenges promptly.
- **Regulatory Monitoring:** Continuously monitor regulatory changes and adapt the system to remain compliant.
- **Disaster Recovery:** Implement disaster recovery and data backup procedures to mitigate the impact of natural disasters.
- **Vendor Diversification:** Explore alternative vendors and maintain vendor relationships to reduce dependence on a single supplier.

32 Costs

- **Preliminary Cost Estimate (Early Stage):**
- **Estimated cost range:** $100,000 - $150,000
- **Timeframe:** 2 months
- **Resources:** 2 business analysts, 1 project manager
- **Detailed Cost Estimate (Mid to Late Stage):**
- **Development and coding:** $300,000
- **Testing and quality assurance:** $80,000
- **Deployment and setup:** $40,000
- **Maintenance (first year):** $50,000
- **External software licenses:** $20,000
- **Contingency (10% of total):** $49,000

- **Total estimated cost:** $539,000
- **Timeframe:** 12 months
- **Resources:** 5 developers, 2 testers, 1 deployment specialist, 1 maintenance engineer, 1 project manager

Cost Breakdown:

- Development and coding will involve building the core functionalities of the WeAllocate platform, including user management, resource tracking, predictive analysis, donation management, and feedback/reporting subsystems.
- Testing and quality assurance will ensure that the system functions correctly and meets all requirements.
- Deployment and setup costs include server hosting, domain registration, and initial system configuration.
- Maintenance costs cover ongoing updates, bug fixes, and support for the first year after deployment.
- External software licenses are required for certain third-party tools and libraries used in the project.
- Contingency is included to account for unforeseen challenges or changes in project scope.

33 Waiting Room

Integration with Social Media: Allow users to share their donation activities and achievements on social media platforms. This feature could help promote the WeAllocate platform and encourage more donations.

- **Mobile App:** Develop a dedicated mobile application for users to access WeAllocate on their smartphones. While the current release focuses on web-based access, a mobile app could enhance user convenience.
- **Advanced Analytics:** Implement advanced analytics and data visualization tools for in-depth insights into resource allocation and demand prediction. This feature would require more extensive data processing capabilities.
- **Community Forums:** Create community forums where users can discuss donation strategies, share success stories, and collaborate on charitable initiatives. This could foster a sense of community among users.
- **Gamification:** Introduce gamification elements to make the platform more engaging. Users could earn badges, rewards, or recognition for their contributions and achievements within the system.

- **Integration with Payment Gateways:** Expand the payment options by integrating with additional payment gateways to accommodate users' preferences and international donors.

- **Multi-Language Support:** Provide support for multiple languages to make the platform accessible to a wider global audience.

- **Resource Reservation:** Allow users to reserve specific resources or time slots for donation, especially for high-demand items like blood donations or volunteer shifts.

- **Impact Assessment Tools:** Develop tools to assess the impact of donations, showing users the real-world outcomes of their contributions.

- **AI-Powered Chatbot:** Implement an AI-powered chatbot for user assistance and support, enhancing the overall user experience.

34 Ideas for Solutions

- **Cloud-Based Hosting:** Consider hosting the WeAllocate platform on a cloud infrastructure like Amazon Web Services (AWS) or Microsoft Azure for scalability and reliability.

- **Microservices Architecture:** Implement a microservices architecture to modularize the system and make it easier to develop, deploy, and maintain different components.

- **Database Choice:** Explore NoSQL databases like MongoDB for flexible data storage and retrieval, especially for handling large volumes of resource data.

- **API-First Approach:** Design the system with a robust API that allows easy integration with external services, enabling data sharing and communication with other charitable organizations.

- **User-Friendly Interfaces:** Prioritize user interface (UI) and user experience (UX) design to create intuitive and accessible interfaces for donors, volunteers, and administrators.

- **Continuous Integration and Deployment (CI/CD):** Set up CI/CD pipelines to automate testing and deployment processes, ensuring rapid and reliable releases.

- **Machine Learning Models:** Consider using machine learning models for demand prediction and resource allocation optimization, enhancing the platform's efficiency.

- **Security Measures:** Implement strong security measures, including encryption, user authentication, and authorization, to protect sensitive user data and transactions.

- **Monitoring and Analytics:** Use tools like Prometheus and Grafana for system monitoring and analytics to proactively identify issues and optimize performance.

- **Scalability:** Design the system to be horizontally scalable, allowing it to handle increased user traffic and resource data without performance degradation.

- **Multi-Platform Support:** Ensure cross-browser and cross-platform compatibility to reach a broad user base, including desktop and mobile users.

- **Localization:** Plan for localization and translation support to make the platform accessible to users in different regions and languages.

- **Documentation:** Maintain comprehensive documentation for developers, administrators, and users to facilitate onboarding and troubleshooting.

- **User Feedback Mechanism:** Implement a user feedback mechanism within the platform to gather suggestions and improvements directly from users.

- **Testing Automation:** Invest in automated testing frameworks to ensure robust testing coverage and faster regression testing.

- **User Training Resources:** Develop training materials and resources for users and administrators to maximize the platform's effectiveness.

- **Third-Party Integrations:** Explore integrations with third-party services like payment gateways, social media platforms, and mapping services to enhance functionality.

35 Project Retrospective

- **What Worked Well**:

    - **Clear Requirements Gathering:** The initial phase of gathering requirements was thorough and effective. It helped in establishing a solid foundation for the project.

    - **Agile Development:** Adopting an Agile development approach allowed for flexibility and adaptability in responding to changing requirements and priorities.

    - **User Feedback:** Incorporating mock user feedback at various stages of development ensured that the product met the needs and expectations of its intended users.

    - **Version Control:** Effective use of version control systems, such as Git, streamlined collaboration and code management among team members.

    - **Documentation:** Comprehensive documentation of requirements, design, and implementation details helped in knowledge sharing and onboarding.

- **What Didn't Work Well:**

    - **Lack of Regular Communication:** 3 of 4 group members were very communicative and having 1 group member not taking initiative resulted in many time crunch development sessions due to having to redo work because of the lack of open communication.

- **Scope Creep:** There were instances of scope creep, where additional features and requirements were introduced without proper evaluation of their impact on the project timeline and resources.

- **Resource Constraints:** Limited resources and time constraints posed challenges in meeting ambitious project goals within the desired timeframe mainly from having to split a 4 person work load into a 3 person workload.

- **Recommendations for Improvement:**

    - **Strict Scope Management:** Implement a more rigorous change control process to prevent scope creep and ensure that all changes are properly assessed before implementation.

    - **Resource Allocation:** Prioritize resource allocation and consider factors like team capacity and skill sets when defining project members.

    - **Sprint Planning:** Improve sprint planning and backlog grooming to enhance the Agile development process and prioritize user stories effectively.

    - **Post-Implementation Review:** Conduct post-implementation reviews allowing for continuous improvement.

    - **Project Metrics:** Establish project metrics to measure progress, quality, and adherence to timelines, enabling better project tracking and decision-making.

V Glossary

WeAllocate: An application designed for optimizing resource allocation in food banks, utilizing machine learning to predict demand and supply patterns.

- **IMAN (Inner-City Muslim Action Network):** A community organization based in Chicago, IL, focusing on health, wellness, and social change, operating the Food and Wellness Center.
- **Predictive Analytics:** The use of statistical algorithms and machine learning techniques to identify the likelihood of future outcomes based on historical data.
- **Resource Allocation:** The process of distributing resources, in this context, food items, efficiently to meet community needs.
- **Machine Learning:** A subset of artificial intelligence that enables systems to learn and improve from experience without being explicitly programmed.
- **Demand Prediction:** Anticipating the quantity of resources (food items) needed based on historical data, current trends, and external factors.
- **Beneficiary:** Individuals or groups receiving aid or resources from organizations, in this case, recipients of food from the Food and Wellness Center.

- **Inventory Management:** The systematic tracking and control of quantities of goods, in this context, food items, ensuring optimal usage and minimizing wastage.
- **Community Events Calendar:** A tool or system tracking local events that may influence demand for resources, such as food distribution.
- **Feedback Portal:** A platform for collecting comments and suggestions from beneficiaries, donors, staff, and volunteers to improve operations.
- **Work Context Diagram:** A visual representation showing the interactions between the WeAllocate system and external entities, such as inventory management, community events, and feedback.
- **Competing Products:** Other tools or solutions in the market that offer inventory management and predictive analytics for organizations, potentially serving as alternatives to WeAllocate.
- **Tailored Approach:** A customized solution designed to specifically address the unique needs and challenges of food banks, as opposed to generic, one-size-fits-all solutions.
- **Workflow Integration:** The seamless incorporation of WeAllocate into IMAN's existing processes and systems for efficient operation.
- **Interface:** A point of interaction or communication between a system and its users, often involving a display or control through which users can interact with the system.
- **Inventory:** A detailed list of goods, items, or resources that a system, organization, or entity possesses.
- **Undue:** Unwarranted or excessive.
- **Intuitiveness:** The quality of being easily understood or grasped without explicit instruction.
- **Tailored:** Customized or adapted to specific needs, preferences, or conditions.
- **Hindering:** Acting as an obstacle or impediment; slowing down or restricting progress.
- **Refinement:** The process of improving, enhancing, or making slight adjustments to something.
- **Insights:** Deep or profound understanding; the ability to gain an accurate and deep understanding.
- **Distribution Efficiency:** The effectiveness and accuracy in distributing resources, often measured by comparing predicted and actual distribution outcomes.
- **Resource Allocation:** The process of distributing resources efficiently to achieve specific objectives or goals.
- **Subsystems:** A self-contained, functional unit within a larger system, contributing independently or interdependently to the overall functionality.
- **Subclasses:** In object-oriented programming, a class derived from another class, inheriting attributes and behaviors, allowing for code reuse and the creation of a class hierarchy.
- **Template Design Pattern:** A behavioral design pattern that defines the skeleton of an algorithm in the superclass but lets subclasses override specific steps of the algorithm without changing its structure. It promotes reusability and allows variations in certain steps of an algorithm.
- **Strategy Design Pattern:** A behavioral design pattern that defines a family of algorithms, encapsulates each algorithm, and makes them interchangeable. It allows the client to choose the appropriate algorithm at runtime, promoting flexibility and enabling dynamic selection of algorithms.

- **Meticulously:** In a detailed and careful manner, showing great attention to precision and thoroughness.
- **Pivotal:** Of crucial importance; central or essential to the development or success of something.
- **Repository:** A central location or storage area where data is stored and managed systematically.
- **Proactively:** Taking action to control or prevent a situation rather than responding to it after it has happened.
- **Proficiency:** The skill, ability, or knowledge in a particular subject or activity.
- **Discrepancies:** Differences or inconsistencies between two or more things, especially in data or records.
- **Optimization:** The action of making the best or most effective use of a situation or resource.
- **Fostering:** Encouraging the development or growth of something
- **Accountability:** The state of being accountable or answerable, often involving the responsibility to justify one's actions.
- **Scalability:** The capability of a system to handle an increasing amount of work, or its potential to be enlarged to accommodate that growth.
- **Enrich:** To improve or enhance the quality or value of something.
- **Streamlined Project Execution:** Efficient and well-coordinated implementation of project tasks and processes, minimizing unnecessary complexities for successful outcomes.
- **Retrospectives:** Periodic reviews, often following agile principles, where a team reflects on past successes and challenges to refine and improve their development process.
- **Sprint Planning:** The process of outlining tasks and goals for a development sprint, a short and time-boxed period in agile methodology.
- **Agile Principles:** A set of values and practices in software development that prioritize flexibility, adaptability, and iterative progress.
- **Testing Protocols:** Established procedures and methodologies for systematically evaluating and validating software functionality, emphasizing early implementation in the development phase.

VI References / Bibliography

1. Catania, Patrick J., and Nancy Keefer. “The Marketplace.” Amazon, Board of Trade, 1987, [link](aws.amazon.com/marketplace/solutions/infrastructure-software).

2. “Food Ecosystems.” Inner-City Muslim Action Network, [link](www.imancentral.org/chicago/organizing-advocacy/food-ecosystems/). Accessed 25 Nov. 2023.

3. “Food Pantry Helper.” Food Pantry Helper - Food Pantry Helper, [link](www.foodpantryhelper.com/). Accessed 25 Nov. 2023.

4. Google, Google, [link](cloud.google.com/?utm_source=google&utm_medium=cpc&utm_campaign=na-US-all-en-dr-bkws-all-all-trial-e-dr-1605212&utm_content=text-ad-none-any-DEV_c-CRE_665665924735-ADGP_Hybrid%2B%7C%2BBKWS%2B-%2BMIX%2B%7C%2BTxt_Google%2BCloud%2BGeneral-KWID_43700077212109151-kwd-13487215878&utm_term=KW_google+cloud+services-ST_google%2Bcloud%2Bservices&gad_source=1&gclid=CjwKCAiA04arBhAkEiwAuNOsIoNeRQs3eJLjRImY1wdwly02AVNCQqaM_gb9oCW7WTHOrrWsulRMSxoCbN8QAvD_BwE&gclsrc=aw.ds). Accessed 25 Nov. 2023.

5. “In West Englewood, a New Food Pantry Opens at Corner of Hope and Heartache.” Greater Chicago Food Depository, Greater Chicago Food Depository, 1 Dec. 2022, [link](www.chicagosfoodbank.org/blog/in-west-englewood-a-new-food-pantry-opens-at-corner-of-hope-and-heartache/).

6. “The Number One Food Pantry Solution!” Food Pantry PRO, [link](foodpantrysoftwarepro.com/?gclid=CjwKCAiA04arBhAkEiwAuNOsIuBbQUtrQRNBUrP_pt7QQdrlEKNzKYbpbNT_OzhbdjkhlmRA_3MGxhoCxw8QAvD_BwE). Accessed 25 Nov. 2023.

VII Index

Design 102, 118
Additional Design Considerations 110
Current System Design 102
Design Goals 102
Final System Design 114
Object Design 117
Proposed System Design 103
Glossary 134
Index *138*
Project Description 10
Mandated Constraints 21
Naming Conventions and Definitions 24
Project Overview 10
Relevant Facts and Assumptions 26
Stakeholders 18
The Purpose of the Project 10
The Scope of the Product 17
The Scope of the Work 13
Project Issues 120
Costs 130

Ideas for Solutions 133
Migration To The New Product..... 127
New Problems..... 123
Off-the-Shelf Solutions..... 121
Open Issues..... 120
Project Retrospective 134
Risks..... 128
Waiting Room..... 131
References/Bibliography..... 138
Requirements 28, 38, 65, 77
19 Cultural and Political Requirements 74
21 Requirements Acceptance Tests 79
Data Requirements..... 38
Dependability Requirements..... 45
Functional Requirements 37
Legal Requirements 77
Look and Feel Requirements 69
Maintainability and Supportability 49
Operational and Environmental Requirements 71
Performance Requirements..... 42
Product Use Cases..... 28
Security Requirements..... 54
Usability and Humanity Requirements. 60

Project Description..... 10
Mandated Constraints 21
Naming Conventions and Definitions... 24
Project Overview 10
Relevant Facts and Assumptions 26
Stakeholders..... 19
The Purpose of the Project..... 10
The Scope of the Product..... 17
The Scope of the Work 13
Requirements 28, 39, 65, 77
19 Cultural and Political Requirements 75
21 Requirements Acceptance Tests 79
Data Requirements..... 39
Dependability Requirements..... 45
Functional Requirements 37
Legal Requirements 77
Look and Feel Requirements 69
Maintainability and Supportability 49
Operational and Environmental Requirements 71
Performance Requirements..... 42
Product Use Cases..... 28
Security Requirements..... 54
Usability and Humanity Requirements. 60