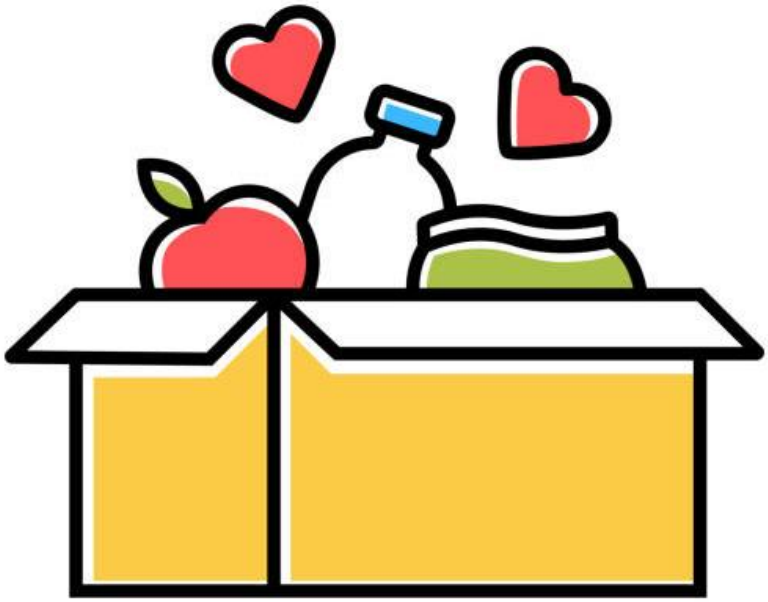
WeAllocate: A Resource Optimizer

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

List of Figures

List of Tables

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.

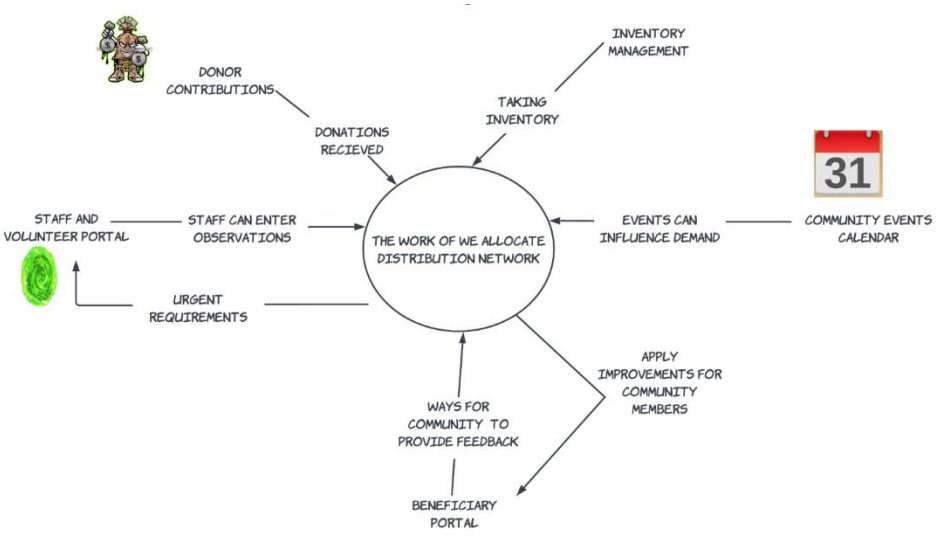
Staff and Volunteers: directly interacting with beneficiaries and handling the day-to-day operations of the Food and Wellness Center. Their insights, observations, and feedback are needed for refining the WeAllocate system. They can also identify anomalies or urgent requirements that might not be immediately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



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

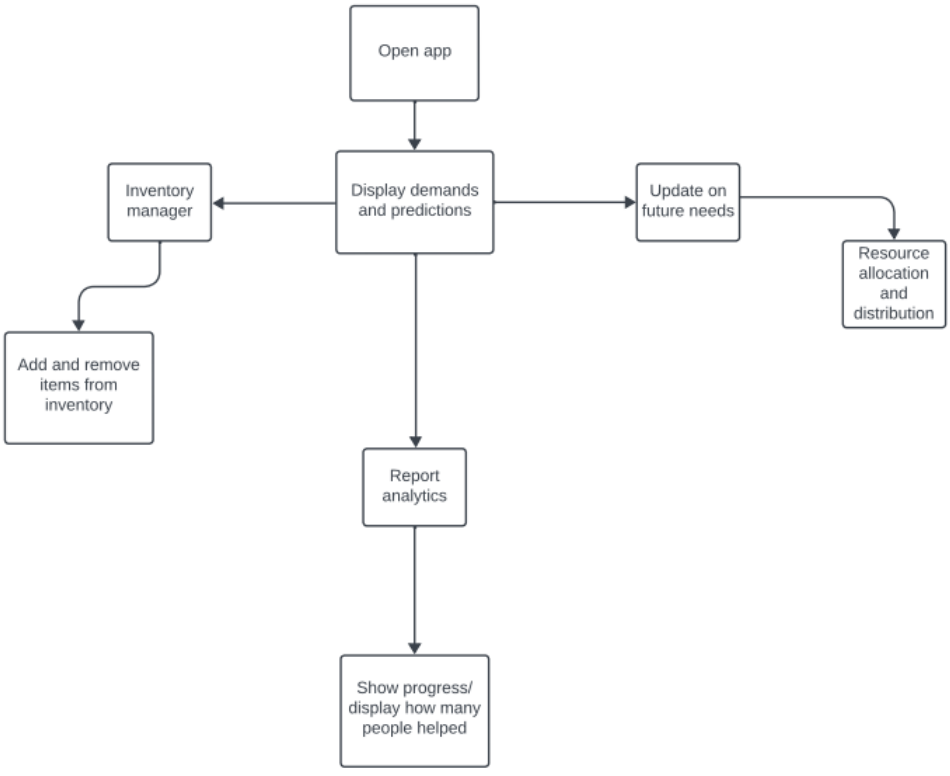
solutions often fall short in addressing the unique demands of food banks. They may lack features for predicting demand based on community changes, might not integrate efficiently with IMAN's existing workflow, and could be a templated solution meeting needs but not exceeding them. The tailored approach 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

1. *Special food in demand*

2. *A lot of food was taken from the food bank and the inventory dropped*
3. *Food is expiring*
4. *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.

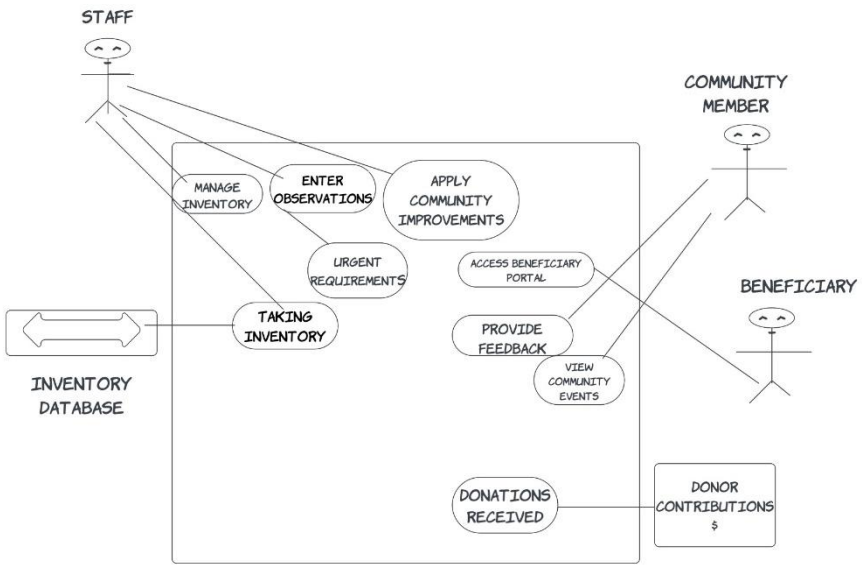
 represents Donor contributions

 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

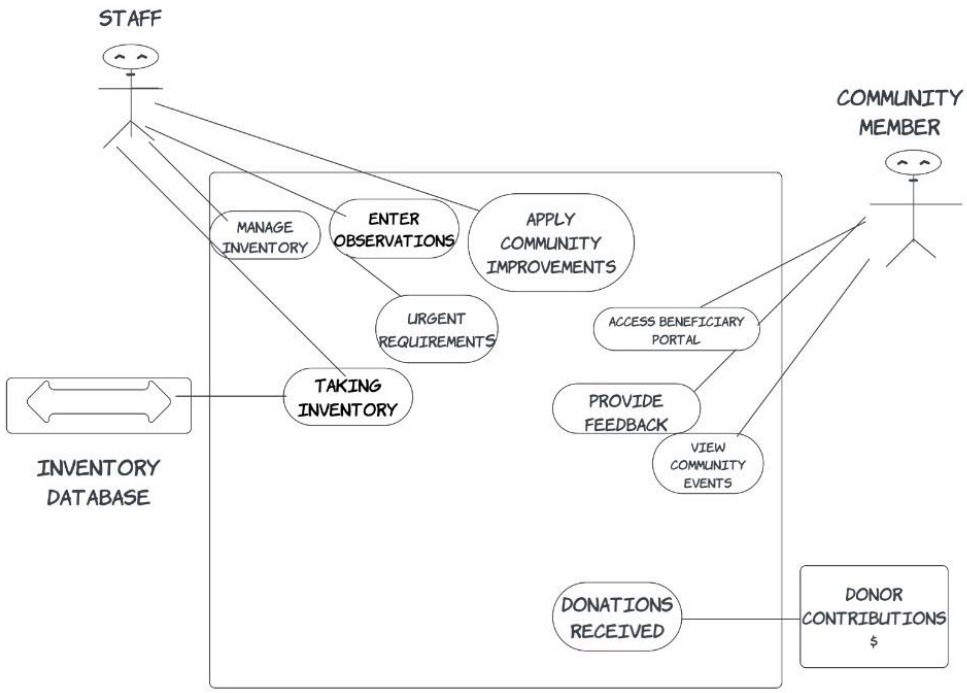


Figure 1 - Sample Use Case Diagram from Robertson and Robertson

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.

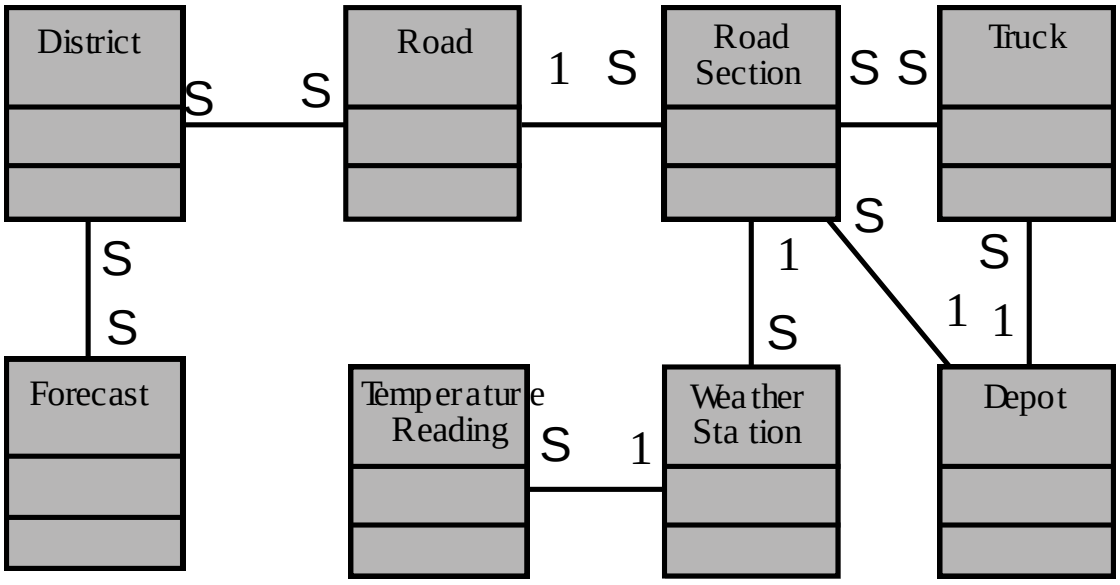
ID# FR-011 - Event Listing and Details

- **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.



Considerations

Are there any data or object models for similar or overlapping systems that might be a useful starting point? Is there a domain model for the subject matter dealt with by this system?

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**

12 Performance Requirements

12a Speed and Latency 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**

12b 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**

12c Capacity 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**

13 Dependability Requirements

13a Reliability 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**

13b 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**

13c Robustness 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

13d 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**

14 Maintainability and Supportability Requirements

14a Maintenance 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.

14b 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
    - 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.

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

14c Adaptability 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
    - 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.

14d 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

14e Longevity 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:**
    - 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.

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:**
    - 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.

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
    - 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.

15 Security Requirements

15a Access 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**

15b Integrity Requirements

Test 1: Verify that WeAllocate logs all user activity.

Test 2: Verify that WeAllocate audit logs are tamper-proof.

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**

15c Privacy 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**
    - 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.

15d 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**

15e Immunity 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**

16 Usability and Humanity Requirements

16a Ease 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**
    - 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.

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: Efficiecy**
    - 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.

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**
    - 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.

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:**
    - 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.

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**

16b 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**

- 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.

16c Learning 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**
  - 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.

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**
  - 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.

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**

16d 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**
    - Test 1: Verify that WeAllocate uses terminology that is familiar to its users.
    - Test 2: Observe users as they interact with WeAllocate and identify any areas where the product uses unfamiliar or confusing terminology.

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.

16e 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**
    - Test 1: Verify that WeAllocate can be fully navigated using the 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**
    - 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.

16f 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

16g Training 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.
- **Acceptance Test:**
    - Test 1: Verify that all new users of WeAllocate complete the WeAllocate Overview and WeAllocate Fundamentals training modules.
    - Test 2: Observe users as they use WeAllocate to complete tasks. Identify any areas where users are struggling or making mistakes. This may indicate that the training needs to be improved.

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**

17 Look and Feel Requirements

17a Appearance 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
    - Test 1: Verify that WeAllocate uses the WeAllocate logo, colors, and fonts consistently throughout the product.
    - Test 2: Conduct a user survey to assess users' perceptions of WeAllocate's visual identity. The survey should ask users whether they find WeAllocate to be visually appealing and consistent with the WeAllocate 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**

17b 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**
    - 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.

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**

18 Operational and Environmental Requirements

18a Expected 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**

18b 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**
    - 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.

18c Productization 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:**
    - 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.

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**
    - 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.

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**
    - Test 1: Verify that the WeAllocate subscription management system allows users to subscribe to and manage their WeAllocate subscriptions successfully.
    - Test 2: Observe users as they subscribe to and manage their WeAllocate subscriptions. Identify any areas where the subscription management system is confusing or difficult to use.

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**
    - 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.

18d 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**
    - 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.

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**
    - 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.

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**
    - 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.

19 Cultural and Political Requirements

19a 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**
  - 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.

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:**
  - Test 1: Verify that WeAllocate has been reviewed by a team of cultural sensitivity experts.
  - Test 2: Observe users from different cultural backgrounds as they use WeAllocate. Identify any areas where the product is culturally insensitive or offensive.

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**
    - 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.

19b 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:**
    - 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.

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**
    - 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.

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**
    - Test 1: Verify that WeAllocate has been reviewed by a legal team and that it complies with all applicable laws and regulations.
    - Test 2: Observe users as they use WeAllocate to perform tasks that are subject to laws and regulations. Identify any areas where the product does not comply with the applicable laws and regulations.

20 Legal Requirements

20a 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:**
    - 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.

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**
    - 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.

20b 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:**
    - 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.

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:**
    - 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.

21 Requirements Acceptance Tests

SV: Every requirement must have one or more acceptance tests associated with it, to confirm that the requirement has been met. At this point these tests are not yet completely specified – A one- or two-sentence description of each test will suffice. Note that some tests may verify more than one requirement, and that some requirements may require multiple tests for their confirmation.

21a Requirements – Test Correspondence Summary

SV: The following sample table is available from the CS 440 web site as "Sample Requirement Test Correspondence Table.xlsx" It is recommended that you work with the table in Excel, and then drag it into the document when it is completed. Depending on the number of requirements and/or tests included, it may be necessary to use multiple tables, and/or use landscape mode. Every row and every column of the table

should include at least one X. Below the table list the ID #, name, and short description of each individual acceptance test.

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														
Test 3			X	X																
Test 4					X	X														
Test 5																				
Test 6																				
Test 7																				
Test 8																				
Test 9																				
Test 10																				
Test 11																				
Test 12																				
Test 13																				
Test 14																				
Test 15																				

Table 1 - Requirements - Acceptance Tests Correspondence

21b Acceptance Test Descriptions

SV: Provide a brief description of each acceptance test. Detailed test specifications will appear in a separate document, which may be referenced here when available.

ID # - Name

Description: Your description here . . .

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.

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.

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.

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.

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.

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.

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.

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.

Efficiecy

- 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

22 Design Goals

SV: Identify the important design goals that are to be optimized in the proposed design.

Content

Design goals are important properties of the system to be optimized, and which may affect the overall design of the system. For example computer games place a higher priority on speed than accuracy, and so the physics engine for a computer game may make some rough approximations and assumptions that allow it to run as fast as possible while sacrificing accuracy, whereas the physics calculations performed by NASA must be much more rigorously correct, even at the expense of speed.

Note an important difference between design goals and requirements: Requirements include specific values that must be met in order for the product to be acceptable to the client, whereas design goals are properties that the designers strive to make "as good

as possible", without specific criteria for acceptability. (Note also that the same property may appear in both a requirement and a design goal, so a design goal may be to make the system run as fast as possible, with a requirement that says any speed below a certain specified threshold is unacceptable.)

Your text goes here . . .

23 Current System Design

*SV: **IF** the proposed new system is to replace an existing system, then the current system should be described here. Otherwise insert a brief statement that there is no pre-existing system.*

Your text goes here . . .

24 Proposed System Design

This section will make heavy use of class diagrams, and also sequence and deployment diagrams where noted. However don't overlook finite state, activity, communication, or other diagram types as needed for effective communication.

24a Initial System Analysis and Class Identification

SV: Perform grammatical and similar analyses to identify the most import and obviously needed classes, and to organize them into an initial class structure. An initial class diagram is appropriate, containing few if any internal details.

Your text goes here . . .

24b Dynamic Modelling of Use-Cases

SV: *Insert sequence diagrams of (at least the most important) use-cases, as a means of identifying other needed classes.*

Content

Include sequence diagrams of each important use-case here. This is a first step towards identifying preliminary objects. (If the sequence diagram would be too big to fit, then it can either be broken down into pieces or a communication diagram can be used in its place.)

Your text goes here . . .

24c Proposed System Architecture

SV: Identify the Software Architecture to be applied to this project, such as Client-Server, Repository, MVC, etc., along with justification for the choice.

Your text goes here . . .

24d Initial Subsystem Decomposition

SV: A slightly more detailed class diagram, showing the classes identified in sections 24a, 24b, and 0 above, partitioned into subsystems. For each subsystem provide a brief description of the subsystem, including its key responsibilities. There should still be few if any internal details.

Your text goes here . . .

25 Additional Design Considerations

SV: The sections listed here do not need to be presented in the order given, and may not all be relevant for any particular project. Those that are relevant can help identify additional classes that are needed as a result.

25a Hardware / Software Mapping

SV: This is particularly important for distributed systems, such as those employing a client-server architecture. Use a deployment diagram to indicate which subsystems are mapped onto which piece(s) of hardware, and what communication subsystems need to be added to the system as a result.

Your text goes here . . .

25b Persistent Data Management

SV: Document the classes and perhaps subsystems necessary to store persistent data when the system shuts down, and to restore that data when the system starts back up again.

Reiterate key data structures and information as necessary for the understanding of this design phase. Refer the reader back to the data dictionary in section I7c above to avoid undue repetition, while reviewing only the most relevant items here.

Your text goes here . . .

25c Access Control and Security

SV: Identify the access control and security concerns for this system, and the new classes and/or subsystems that must be added to handle those concerns.

Your text goes here . . .

25d Global Software Control

SV: Identify the global software control concerns for this system, and the new classes and/or subsystems that must be added to handle those concerns.

Your text goes here . . .

25e Boundary Conditions

SV: Identify the boundary condition concerns for this system, and the new classes and/or subsystems that must be added to handle those concerns. In particular consider startup, shutdown (normal or abnormal), and the creation and/or maintenance of any configuration files, databases, or similar supporting data files.

Your text goes here . . .

25f User Interface

SV: Include a preliminary user interface design here, possibly as a rough sketch or other mockup, in order to identify additional classes needed to implement the interface.

The final user interface design will normally be developed by appropriate experts in that area. However it is appropriate to include an initial design here, including possibly a low- or high- fidelity sketch/mockup, in order to identify key classes necessary to implement the user interface, such as forms and dialog windows. It may also go towards addressing usability and/or look-and-feel requirements, and/or identifying other overlooked components.

Your text goes here . . .

25g Application of Design Patterns

SV: Any design patterns applied as a result of previous sections should have been addressed there, and identified as such at the time. Use this section to document only the additional design patterns that were not previously covered elsewhere. (If any.)

Your text goes here . . .

26 Final System Design

SV: Include here the final version of the overall system design, incorporating all the subsystems and classes added as a result of additional design considerations. Multiple diagrams may be needed, possibly starting with an overall package diagram showing all the different subsystems and the (important) classes contained within each one. Still not a lot of internal details.

Your text goes here . . .

27 Object Design

This section documents the internal details of each class, to the extent that they can be designed at this time. Included should be the class interfaces (public method signatures and responsibilities) and constraints. It is probably best to break this section up into subsections corresponding to subsystems as documented above, and/or by (Java) packages if those are designed. It may also be appropriate to address additional design pattern considerations here, but not to the point of being redundant of previous documentation.

Certain methods, such as simple getters, setters, and constructors are not always documented, unless there is something special about them such as in the Singleton or Factory Method design patterns.

27a Packages

SV: If the design involves assigning classes to packages (.e.g Java packages), then the packages to be created should be documented here.

Your text goes here . . .

27b Subsystem I

Your text goes here . . .

27c Subsystem II

Your text goes here . . .

27d etc.

Your text goes here . . .

IV Project Issues

28 Open Issues

SV: Issues that have been raised and do not yet have a conclusion.

Content

A statement of factors that are uncertain and might make significant difference to the product.

Motivation

To bring uncertainty out in the open and provide objective input to risk analysis.

Examples

Our investigation into whether the new version of the processor will be suitable for our application is not yet complete.

The government is planning to change the rules about who is responsible for gritting the motorways, but we do not know what those changes might be.

Considerations

Are there any issues that have come up from the requirements gathering that have not yet been resolved? Have you heard of any changes that might occur in the other

organizations or systems on your context diagram? Are there any legislative changes that might affect your system? Are there any rumors about your hardware or software suppliers that might have an impact?

Your text goes here . . .

29 Off-the-Shelf Solutions

SV: Discussion of products or components currently available that could either be incorporated into the new solution or simply used instead of developing (parts of) the new solution. The distinction between sections 35 a, b, and c is subtle, and not very important.

Your text goes here . . .

29a Ready-Made Products

SV: Products available for purchase that could be used either as part of a solution or instead of (a part of) a solution.

Content

List of existing products that should be investigated as potential solutions. Reference any surveys that have been done on these products.

Motivation

To give consideration to whether a solution can be bought.

Considerations

Could you buy something that already exists or is about to become available? It may not be possible at this stage to make this determination with a lot of confidence, but any likely products should be listed here.

Also consider whether some products must not be used.

Your text goes here . . .

29b Reusable Components

SV: Similar to 35a, but for components such as libraries or toolkits instead of fully blown products.

Content

Description of the candidate components, either bought from outside or built by your company, that could be used by this project. List libraries that could be a source of components.

Motivation

Reuse rather than reinvention.

Your text goes here . . .

29c Products That Can Be Copied

SV: Products that could legally be copied would typically be past projects developed by the same development group, provided there were no restrictions that would prevent their reuse.

Content

List of other similar products or parts of products that you can legally copy or easily modify.

Motivation

Reuse rather than reinvention.

Examples

Another electricity company has built a customer service system. Its hardware is different from ours, but we could buy its specification and cut our analysis effort by approximately 60 percent.

Considerations

While a ready-made solution may not exist, perhaps something, in its essence, is similar enough that you could copy, and possibly modify, it to better effect than starting from scratch. This approach is potentially dangerous because it relies on the base system being of good quality.

This question should always be answered. The act of answering it will force you to look at other existing solutions to similar problems.

Your text goes here . . .

30 New Problems

SV: The proposed new system certainly has its benefits, but it could also raise new problems. It is a good idea to identify any such potential problems early on, rather than being surprised by them later.

30a Effects on the Current Environment

SV: Could the new system have any adverse effects on the working environment, e.g. the way people do their jobs?

Content

A description of how the new product will affect the current implementation environment. This section should also cover things that the new product should not do.

Motivation

The intention is to discover early any potential conflicts that might otherwise not be realized until implementation time.

Examples

Any change to the scheduling system will affect the work of the engineers in the divisions and the truck drivers.

Considerations

Is it possible that the new system might damage some existing system? Can people be displaced or otherwise affected by the new system?

These issues require a study of the current environment. A model highlighting the effects of the change is a good way to make this information widely understandable.

Your text goes here . . .

30b Effects on the Installed Systems

SV: Could the new system have any adverse effects on other hardware or software systems?

Content

Specification of the interfaces between new and existing systems.

Motivation

Very rarely is a new development intended to stand completely alone. Usually the new system must coexist with some older system. This question forces you to look carefully at the existing system, examining it for potential conflicts with the new development.

Your text goes here . . .

30c Potential User Problems

SV: Could the new system have any adverse effects on the users of the software? Could users possibly have a negative response to the new system?

Content

Details of any adverse reaction that might be suffered by existing users.

Motivation

Sometimes existing users are using a product in such a way that they will suffer ill effects from the new system or feature. Identify any likely adverse user reactions, and determine whether we care about those reactions and what precautions we will take.

Your text goes here . . .

30d Limitations in the Anticipated Implementation Environment That May Inhibit the New Product

SV: Are there any (physical) limitations in the expected environment that could inhibit the proposed product? (e.g. weather, electrical interference, radiation, lack of reliable power, etc.)

Content

Statement of any potential problems with the new automated technology or new ways of structuring the organization.

Motivation

The intention is to make early discovery of any potential conflicts that might otherwise not be realized until implementation time.

Examples

The planned new server is not powerful enough to cope with our projected growth pattern.

The size and weight of the new product do not fit into the physical environment.

The power capabilities will not satisfy the new product's projected consumption.

Considerations

This requires a study of the intended implementation environment.

Your text goes here . . .

30e Follow-Up Problems

SV: Basically any other possible problems that could occur.

Content

Identification of situations that we might not be able to cope with.

Motivation

To guard against situations where the product might fail.

Considerations

Will we create a demand for our product that we are not able to service? Will the new system cause us to run afoul of laws that do not currently apply? Will the existing hardware cope?

There are potentially hundreds of unwanted effects. It pays to answer this question very carefully.

Your text goes here . . .

31 Migration to the New Product

SV: This section only applies when there is an existing system that is being replaced by a new system, particularly when data must be preserved and possibly translated / reformatted. Otherwise just write "Not Applicable" under section 38 and remove sections 38a and 38b.

31a Requirements for Migration to the New Product

SV: These are a list of requirements relevant to the migration procedures. For example a requirement that the two systems be run in parallel for a time until the client is satisfied with the new system and the users know how to use it.

Content

A list of the conversion activities. Timetable for implementation.

Motivation

To identify conversion tasks as input to the project planning process.

Considerations

Will you use a phased implementation to install the new system? If so, describe which requirements will be implemented by each of the major phases.

What kind of data conversion is necessary? Must special programs be written to transport data from an existing system to the new one? If so, describe the requirements for these programs here.

What kind of manual backup is needed while the new system is installed?

When are each of the major components to be put in place? When are the phases of the implementation to be released?

Is there a need to run the new product in parallel with the existing product?

Will we need additional or different staff?

Is any special effort needed to decommission the old product?

This section is the timetable for implementation of the new system.

Your text goes here . . .

31b Data That Has to Be Modified or Translated for the New System

*SV: This section specifically addresses **data** that must be preserved and/or translated / reformatted during the migration process.*

Content

List of data translation tasks.

Motivation

To discover missing tasks that will affect the size and boundaries of the project.

Fit Criterion

Description of the current technology that holds the data.

Description of the new technology that will hold the data.

Description of the data translation tasks.

Foreseeable problems.

Considerations

Every time you make an addition to your dictionary (see section 5), ask this question: Where is this data currently held, and will the new system affect that implementation?

Your text goes here . . .

32 Risks

SV: Consideration of the potential risks that could cause the project to fail / underperform.

All projects involve risk—namely, the risk that something will go wrong. Risk is not necessarily a bad thing, as no progress is made without taking some risk. However, there is a difference between unmanaged risk—say, shooting dice at a craps table—and managed risk, where the probabilities are well understood and contingency plans are made. Risk is only a bad thing if the risks are ignored and they become problems. Risk management entails assessing which risks are most likely to apply to the project, deciding a course of action if they become problems, and monitoring projects to give early warnings of risks becoming problems.

This section of your specification should contain a list of the most likely risks and the most serious risks for your project. For each risk, include the probability of that risk becoming a problem. Capers Jones's Assessment and Control of Software Risks

(Prentice-Hall, Englewood Cliffs, N.J., 1994) gives comprehensive lists of risks and their probabilities; you can use these lists as a starting point. For example, Jones cites the following risks as being the most serious:

• Inaccurate metrics

• Inadequate measurement

• Excessive schedule pressure

• Management malpractice

• Inaccurate cost estimating

• Silver bullet syndrome

• Creeping user requirements

• Low quality

• Low productivity

• Cancelled projects

Use your knowledge of the requirements as input to discover which risks are most relevant to your project.

It is also useful input to project management if you include the impact on the schedule, or the cost, if the risk does become a problem.

Your text goes here . . .

33 Costs

SV: An estimate of what it will cost to complete this project. Think not only in terms of dollars, but also time, resources, lost opportunities, etc.

For details on how to estimate requirements effort and costs, refer to Appendix C Function Point Counting: A Simplified Introduction

The other cost of requirements is the amount of money or effort that you have to spend building them into a product. Once the requirements specification is complete, you can use one of the estimating methods to assess the cost, expressing the result as a monetary amount or time to build.

There is no best method to use when estimating. Keep in mind, however, that your estimates should be based on some tangible, countable artifact. If you are using this template, then, as a result of doing the work of requirements specification, you are producing many measurable deliverables. For example:

- *Number of input and output flows on the work context*

- *Number of business events*
- *Number of product use cases*
- *Number of functional requirements*
- *Number of nonfunctional requirements*
- *Number of requirements constraints*
- *Number of function points*

The more detailed the work you do on your requirements, the more accurate your deliverables will be. Your cost estimate is the amount of resources you estimate each type of deliverable will take to produce within your environment. You can create some very early cost estimates based on the work context. At that stage, your knowledge of the work will be general, and you should reflect this vagueness by making the cost estimate a range rather than a single figure.

As you increase your knowledge of the requirements, we suggest you try using function point counting—not because it is an inherently superior method, but because it is so widely accepted. So much is known about function point counting that it is possible to make easy comparisons with other products and other installations' productivity.

It is important that your client be told at this stage what the product is likely to cost. You usually express this amount as the total cost to complete the product, but you may also find it advantageous to point out the cost of the requirements effort, or the costs of individual requirements.

Whatever you do, do not leave the costs in the lap of hysterical optimism. Make sure that this section includes meaningful numbers based on tangible deliverables.

Your text goes here . . .

34 Waiting Room

SV: This is a place to record ideas or wishes that will not be included in the current release of the product, but which might be worth reconsidering at a later date.

Requirements that will not be part of the next release. These requirements might be included in future releases of the product.

Content

Any type of requirement.

Motivation

To allow requirements to be gathered, even though they cannot be part of the current development. To ensure that good ideas are not lost.

Considerations

The requirements-gathering process often throws up requirements that are beyond the sophistication of, or time allowed for, the current release of the product. This section holds these requirements in waiting. The intention is to avoid stifling the creativity of your users and clients, by using a repository to retain future requirements. You are also managing expectations by making it clear that you take these requirements seriously, although they will not be part of the agreed-upon product.

Many people use the waiting room as a way of planning future versions of the product. Each requirement in the waiting room is tagged with its intended version number. As a requirement progresses closer to implementation, then you can spend more time on it and add details such as the cost and benefit attached to that requirement.

You might also prioritize the contents of your waiting room. "Low-hanging fruit"—requirements that provide a high benefit at a low cost of implementation—are the highest-ranking candidates for the next release. You would also give a high waiting room rank to requirements for which there is a pent-up demand.

Your text goes here . . .

35 Ideas for Solutions

SV: When developing requirements only, it is not the role of the business analyst to dictate the implementation of the solution. However they can pass along any ideas they have here as suggestions to the developers. For CS 440 this report includes system and object design, so this section would make suggestions for implementation and testing that would come after design, such as the use of a particular language, IDE, library, or other tools.

When you gather requirements, you focus on finding out what the real requirements are and try to avoid coming up with solutions. However, when creative people start to think about a problem, they always generate ideas about potential solutions. This section of the template is a place to put those ideas so that you do not forget them and so that you can separate them from the real business requirements.

Content

Any idea for a solution that you think is worth keeping for future consideration. This can take the form of rough notes, sketches, pointers to other documents, pointers to people, pointers to existing products, and so on. The aim is to capture, with the least amount of effort, an idea that you can return to later.

Motivation

To make sure that good ideas are not lost. To help you separate requirements from solutions.

Considerations

While you are gathering requirements, you will inevitably have solution ideas; this section offers a way to capture them. Bear in mind that this section will not necessarily be included in every document that you publish.

Your text goes here . . .

36 Project Retrospective

SV: At the conclusion of the (CS 440) project, reflect back on what worked well and what didn't, and how the process could be improved in the future.

Content

At the end of every project you should reflect upon what methods were used that worked out well and should be repeated in the future, and also what methods did not work out well and should be avoided. Any recommendations, suggestions, or ideas for how to do things better in the future should also be documented

Motivation

To learn from experience, and to continually strive for process improvement.

Considerations

When things don't go well, it is important to distinguish whether the methods themselves were poor, or simply poorly implemented in this particular case, or whether they just weren't right for this particular project / group of engineers.

Your text goes here . . .

V Glossary

SV: The glossary is a more complete and inclusive dictionary of defined terms than that found in section I.7.a, the latter of which only covered the most important key terms needed to understand the report.

The glossary defines terms that may not be familiar to all readers. This is especially important if the document is expected to reach a wide and varied audience, such as school children. The glossary may be placed at either the beginning or the end of the document.

Flotsam: *Any part of a ship or its cargo found floating on the water, whether it was deliberately or accidentally lost by its original owners.*

Jetsam: *Any part of a ship or its cargo that is deliberately cast off (jettisoned) by its original owners, generally in order to lighten the ship, whether it floats or sinks.*

Your text goes here . . .

VI References / Bibliography

This section describes the documents and other sources from which information was gathered. This sample bibliography was generated using the "Insert Citation" and "Bibliography" buttons in the "Citations & Bibliography" section under the "References" tab of MS Word. Creating new citations will not update this list unless you click on it and select "Update Field". You may need to reset the style for this paragraph to "normal" after updating.

[1] Robertson and Robertson, Mastering the Requirements Process.

[2] A. Silberschatz, P. B. Galvin and G. Gagne, Operating System Concepts, Ninth ed., Wiley, 2013.

[3] J. Bell, "Underwater Archaeological Survey Report Template: A Sample Document for Generating Consistent Professional Reports," Underwater Archaeological Society of Chicago, Chicago, 2012.

[4] M. Fowler, UML Distilled, Third Edition, Boston: Pearson Education, 2004.

VII Index

This section provides an index to the report. The sample below was generated using the "Mark Entry" and "Insert Index" items from the "Index" section on the "References" tab, and can be automatically updated by right clicking on the table below and selecting "Update Field". To remove marked entries from the document, toggle the display of hidden paragraph marks (the paragraph button on the "Home" tab), and remove the tags shown with XE in { curly braces. }