*Norbert Shema*

Systems Analysis and Design

Husson University.

***Smart Grocery Shopping App.***

Table of Contents

[Change Log 3](#_Toc187411246)

[Project Selection and Description 4](#_Toc187411247)

[SDLC Framework 4](#_Toc187411248)

[Stakeholder Analysis and Requirements Identification 4](#_Toc187411249)

[Modeling the Project 5](#_Toc187411250)

[Designing the Project 5](#_Toc187411251)

[Quality Assurance 5](#_Toc187411252)

[Implementation and Handoff 6](#_Toc187411253)

[Summary and Reflection 6](#_Toc187411254)

**Project Iteration Submission Cycle**

Iteration 01:

* Change Log
* Project Selection & Description
* SDLC Framework
* Summary & Reflection

Iteration 02:

* Updates to previous work based on instructor feedback
* Change Log
* Stakeholder Analysis& Requirements Identification
* Summary & Reflection

Iteration 03:

* Updates to previous work based on instructor feedback
* Change Log
* Modeling the Project
* Summary & Reflection

Iteration 04:

* Updates to previous work based on instructor feedback
* Change Log
* Designing the Project
* Summary & Reflection

Iteration 05:

* Updates to previous work based on instructor feedback
* Change Log
* Quality Assurance
* Implementation & Handoff
* Summary & Reflection

# Change Log

Use this section to include brief notes about changes from one iteration to the next. An example is provided below.

***Sample Change Log***

*2025-01-28: Iteration 1*

1. Created initial project document with inclusion of project description and iteration reflection
2. Completed the initial SDLC Framework and the Summary & Reflection for the work completed in first iteration.

*2025-02-05: Iteration 2*

1. Itemized out who the stakeholders would be for the project.
2. Documented at least 5 functional requirements (specific system features or actions) and 3 non-functional requirements (performance, security, usability) for the project.
3. Updated the Summary & Reflection for iteration 2.

*2025-02-11: Iteration 3*

1. Moved the stakeholder analysis and requirement gathering in this main document.
2. Created an entity-relationship diagrams as an appropriate model for the project.
3. Updated the Summary & Reflection for iteration 3.

*2025-02-18: Iteration 4*

1. Made some changes on the ERD to add optionality on some of the entities and adjust the angles to make it more visual.
2. Updated the document for easy readability.
3. Completed the physical diagram and the project designing.
4. Updated the Summary & Reflection for iteration 4.

*2025-04-03: iteration 5*

1. Updated the document for easy readability.
2. Completed the Quality Assurance Plan, Implementation Plan and, Handoff Plan.
3. Updated the Summary & Reflection for iteration 4.

# Project Selection and Description*.*

**Project Overview:**

The Smart Grocery Shopping App is designed to help users streamline their grocery shopping experience by ensuring they purchase exactly what they need for their recipes. The app will allow users to input or select recipes, automatically generating shopping lists with the required ingredients. The app will also provide the options to budget and diets planning according to history of shopping.

**Intended Business and User Community:**

This app aims to serve busy individuals, home cooks, and health-conscious shoppers who struggle with meal planning and grocery shopping.

**App usage**

I envision this app will be used by integrating smart recommendations, recipe-based shopping lists, and real-time inventory tracking, the app ensures users buy only what they need, reducing food waste and saving time. It is envisioned as a user-friendly mobile application that simplifies meal preparation while enhancing the shopping process.

**Project Interest**

I am interested in this project because as aspiring busy professional and someone who is learning how to cook, this project will help me to understand what I eat and how to make certain recipes. It will also allow me to track my diet as it could be hard for busy individuals to know what they eat and stay healthy. Overall, the concept aligns with the growing trend of digital grocery shopping which I am eager to learn how it works through this project.

# SDLC Framework

The Software Development Life Cycle (SDLC) will provide a structured approach to developing the **Smart Grocery Shopping App**, this will be ensuring that each phase contributes to building a functional, user-friendly, and scalable solution of the smart grocery app. Below is a high-level explanation of how each SDLC phase applies to this project.

**Planning:**

This phase involves defining the objectives and feasibility of the app. Since the app is designed for busy individuals, home cooks, and health-conscious users, the planning stage will focus on understanding their needs. Key questions include what features will be essential (such as recipe-based shopping lists, budgeting tools, and diet tracking), identifying competitors, and estimating costs and timelines. A feasibility study ensures that the app can successfully address the problem of grocery shopping inefficiencies.

**Analysis:**

During the analysis phase, detailed requirements are gathered from potential users. Since the app focuses on meal planning, budgeting, and diet tracking, research will be conducted on how users interact with grocery shopping apps. Functional requirements such as smart recommendations, shopping history tracking, and inventory management will be documented. Non-functional requirements, such as performance, security, and scalability, will also be defined to ensure a smooth user experience.

**Design:**

The design phase translates requirements into a blueprint for development. User interface and user experience designers will create prototypes and wireframes to ensure the app is user-friendly and visually appealing. The system structure will be designed to support real-time inventory tracking, recipe integrations, and budgeting tools. Additionally, the database structure will be planned to store user shopping history, dietary preferences, and inventory details.

**Development:**

In this phase, the actual coding and app development will take place. Developers will build the front-end (mobile app interface) and back-end (database, API integration, and recommendation predictions). Features such as recipe-based shopping lists, barcode scanning, and AI-powered meal planning will be implemented iteratively. Agile development practices may be used to continuously improve and test the app as new features are built.

**Testing:**

The app will go through extensive testing to ensure functionality, usability, and security. Unit testing will verify that each component (e.g., shopping list generation, budget tracking) works correctly. Integration testing will ensure smooth interactions between features such as recipe selection and grocery list creation. User testing will also happen in this stage to get real user feedback on whether the app meets their grocery shopping needs efficiently.

**Deployment:**

After successful testing, the app will be deployed for public use. A gradual rollout may be planned to monitor initial user feedback and address any immediate concerns. The database will be configured to handle real-time updates and secure user data. Marketing efforts will also begin to attract users to the app.

**Maintenance & Support:**

Once live, the app requires continuous monitoring and updates. Bug fixes, performance improvements, and feature enhancements (such as integrating more dietary tracking tools) will be released based on user feedback. Security patches will be regularly applied to protect sensitive shopping and diet data. Future updates may also include grocery store partnerships and AI-driven personalized recommendations to keep users engaged.

# Stakeholder Analysis and Requirements Identification

1. **Stakeholder Analysis**

Key stakeholders for the Smart Grocery Shopping App include the following groups:

1. **End Users:**
2. Busy Professionals: The users are specifically individuals with tight schedules who value efficiency in grocery shopping and meal planning.
3. Home Cooks: People interested in cooking meals at home, requiring recipe-based shopping assistance.
4. Health-Conscious Shoppers: Users focused on good and strict dietary tracking and nutrition goals.
5. **Developers and Designers:**

Developer and designers are simply the stakeholders who are responsible for building, designing, and maintaining the app, ensuring it is functional, user-friendly, and scalable.

1. **Project Managers:**

These are the stakeholders who will be overseeing the app's development, ensuring adherence to deadlines, budgets, and quality standards of the application.

1. **Retail Partners (Future Stakeholders):**

These the future stakeholders, they include specifically the grocery stores and supermarkets that may integrate their inventory systems with the app to provide real-time stock updates for the customers.

1. **Investors and Sponsors:**

Finally, the stakeholders in form of investors and sponsors will be providing financial backing for the app's development and marketing efforts.

**Stakeholder Matrix/Map**

**Internal Stakeholders External Stakeholders Ranks for stakeholders**

Developers/Designers End Users Retail Partners 1- Most Interest

Project Managers Investors/Sponsors 2-Some Interest

End Users 3-Least Interest

A-Most Influence

B-Some Influence

C- Little Influence

***Stakeholders Ranks***

End Users 1A

Developers/Designers 2A

Project Managers 2A

Retail Partners 2B

Investors/Sponsors 1B

A screenshot of a grid with text

Description automatically generated

Figure 1

The figure 1 above, show a visual representation of the stakeholders involves in the Smart Grocery App categorized and based on their level of influence and interest.

1. **Requirements Gathering**
2. ***Functional Requirements***
3. Recipe-Based Shopping List Generation:

Users will be able to input or select recipes, and the app generates a detailed shopping list of required ingredients.

1. Budget Planning and Tracking:

Users will be able to set a budget, and the app provides insights into how much they are spending per shopping trip.

1. Dietary Preference Tracking:

Users will be able to set a dietary preference (e.g., vegetarian, gluten-free) and receive recommendations aligned with their goals.

1. Smart Recommendations:

Users will also be able to get personalized suggestions for recipes or items based on previous shopping.

1. Inventory Updates:

The app will also integrate with grocery store systems to show availability of items at the stores.

1. ***Non-Functional Requirements***
2. Performance:

The app should be able to load within 3 seconds and handle up to 1000 active users without performance degradation.

1. Security:

All user data (e.g., shopping history, dietary preferences) must be encrypted during storage and transmission.

1. Usability:

The interface must be intuitive and accessible to a diverse user base, with a focus on simplicity and ease of navigation.

**Prioritization of Requirements**

Below shows a table with listed functional and non-functional requirements, prioritized.

|  |  |  |
| --- | --- | --- |
| Requirement | Priority | Reasoning |
| Recipe-Based Shopping List Generation | High | Core feature and primary functionality of the app. |
| Budget Planning and Tracking | Medium | Important for attracting users focused on financial control. |
| Dietary Preference Tracking | High | Aligns with health-conscious user needs. |
| Smart Recommendations | Medium | Enhances user experience but can be developed iteratively. |
| Inventory Updates | Low | Dependent on partnerships; not critical for initial launch. |
| Performance (Non-Functional) | High | Crucial for user retention and smooth operation. |
| Security (Non-Functional) | High | Protects user data and builds trust with the user base. |
| Usability | High | This enhances engagement on the app and users can use it in the first place. |
|  | | |

# Modeling the Project*.*

**Entity-Relationship Diagram (ERD)**

This ERD models the key entities and their relationships in the **Smart Grocery Shopping App** system. It shows how the user data, recipes, ingredients, and shopping lists interact:

* Users create shopping lists and save recipes based on their dietary preferences.
* Recipes are made up of multiple ingredients, stored in the system's database.
* Shopping lists are generated based on recipes and contain shopping list items, which track specific ingredients and their quantities.
* Inventory connects to grocery store systems to check ingredient availability and pricing, enabling real-time updates for users.

**Entities list:**

1. User
2. Recipe
3. Ingredient
4. ShoppingList
5. ShoppingListItem
6. Inventory

**Entities description.**

* User: Represents individuals using the app, storing their personal details, preferences, and budgets.
* Recipe: Contains meal preparation instructions and details, linking to ingredients needed for the recipe.
* Ingredient: Represents items used in recipes or shopping lists, and this where we have different attributes like name, price etc.
* ShoppingList: Tracks user-generated lists of items to purchase.
* ShoppingListItem: Represents individual ingredients/items in a shopping list, including quantity and references.
* Inventory: Tracks item availability and pricing in stores for real-time grocery updates.

**Entity relationships:**

User and Recipe:

* A user can save multiple recipes.
* Each recipe belongs to one user.

Recipe and Ingredient:

* A recipe can include multiple ingredients.
* An ingredient can be used in multiple recipes.

User and ShoppingList:

* A user can create multiple shopping lists.
* Each shopping list belongs to one user.

ShoppingList and ShoppingListItem:

* A shopping list can contain multiple shoppingList items.
* Each shoppingList item belongs to one shopping list.

ShoppingListItem and Ingredient:

* Each shoppingListItem corresponds to one specific ingredient.
* A specific ingredient can appear in multiple shopping list items.

Inventory and Ingredient:

* Each store's inventory tracks the availability and price of specific ingredients.
* Many stores can store the same ingredients.

A diagram of a recipe

Description automatically generated

Figure 2

The figure above shows the Entity-Relationship Diagram (ERD).

# Designing the Project*.*

The Smart Grocery Shopping App is built using a three-tier architecture, which means it has three main parts: the front-end (what the user sees and interacts with), the back end (where the app does all the heavy lifting), and the database (where everything is stored).

1. Front-End Interface
   * For the user interface, I’ll use HTML, CSS, and JavaScript to create the layout and functionality of the app.
   * The front end is built with React.js, making the app dynamic and responsive, so users can easily browse recipes, create shopping lists, and check ingredient availability.
2. Back-End Logic
   * On the back end, I’ll use Node.js with Express.js to handle all the logic and make the app run smoothly.
   * This part makes sure things like user logins, creating shopping lists, and checking ingredient availability all work behind the scenes.
3. Database
   * For storing everything like user info, recipes, shopping lists, and inventory, I’ll use MySQL.
   * It’ll store all the data in a way that’s quick to access and makes sure everything is connected properly, like which ingredients belong to a recipe or which items are in a shopping list.
4. Application Server
   * I’ll use Apache (or Nginx) to host the app and manage all the web traffic.
   * This will make sure that when users open the app or perform an action, the requests go to the right place and get handled correctly.
5. Networking & Security
   * I’ll rely on Cisco networking gear to ensure data is securely transferred between the app and the database.
   * For extra security, the app will use JWT-based authentication to keep users’ accounts safe and SSL encryption to protect all data during transactions.

When the user interacts with the front-end (such as adding a recipe or creating a shopping list), the app sends a request to the back end via API calls. The back end, built with Node.js, processes these requests and either fetches or updates the data from the MySQL database. The Apache or Nginx server ensures that the app’s content loads smoothly and manages the communication between the front-end and back-end. The database stores all the user data, recipes, and inventory updates, allowing the app to display accurate, real-time information. Cisco networking gear ensures that everything stays connected, allowing users to securely and reliably interact with the app.

**Full Database Physical ERD (P-ERD)**



Figure 3

The table above in figure 3 we can see that after the entities, relationships, and rules formalized, a listing of attributes has been drafted, as preparation for normalization. The listing below also includes the primary or foreign keys listed.

A diagram of a diagram

Description automatically generated

Figure 4

The figure 4 above show the full physical ERD for the model.

# Quality Assurance*.*

To make sure the Smart Grocery Shopping App runs smoothly and delivers a great experience, we’ll test it at different levels:

* **Unit Testing**: We’ll check individual features (like logging in or adding a recipe) to make sure they work correctly on their own.
* **Integration Testing**: Since different features interact (e.g., selecting a recipe should generate a shopping list), we’ll test how well these connections work.
* **User Acceptance Testing (UAT)**: We’ll have real users try out the app to see if it meets their needs and feels intuitive.
* **Performance Testing**: Verifies the app's responsiveness and stability under different loads.
* **Security Testing**: Ensures data protection through encryption and secure authentication methods.

**Key Test Scenarios.**

1. ***Login and Authentication***
   * **Scenario:** User attempts to log in with valid and invalid credentials.
   * **Required Input:** Correct and incorrect username/password combinations.
   * **Expected Results:** Successful login with valid credentials; error message for invalid credentials.
2. ***Recipe-Based Shopping List Generation***
   * **Scenario:** User selects a recipe to generate a shopping list.
   * **Required Input:** A chosen recipe with predefined ingredients.
   * **Expected Results:** The app generates an accurate shopping list with all required ingredients.
3. ***Budget Tracking and Alerts***
   * **Scenario:** User sets a budget and makes purchases.
   * **Required Input:** A budget amount and added items with prices.
   * **Expected Results:** The app provides real-time budget tracking and alerts if the budget is exceeded.
4. ***Dietary Preference Filtering***
   * **Scenario:** User sets dietary preferences (e.g., vegetarian, gluten-free) and searches for recipes.
   * **Required Input:** Selected dietary preference.
   * **Expected Results:** The app filters recipes and shopping suggestions accordingly.
5. ***Inventory and Availability Updates***
   * **Scenario:** The app integrates with grocery store databases to check item availability.
   * **Required Input:** Selected store location and shopping list.
   * **Expected Results:** The app displays real-time stock availability for selected items.

# Implementation and Handoff*.*

**Implementation plan.**

Launching the app isn’t just about flipping a switch. We’ll roll it out carefully to avoid issues and make sure users have a smooth experience. The implementation of the Smart Grocery Shopping App follows a structured process to ensure a smooth rollout and adoption.

**Deployment Steps:**

1. **Server Setup:**
   * Configure cloud hosting (e.g., AWS, Azure) to support back-end services.
   * Set up databases and API gateways for secure communication.
2. **Application Deployment:**
   * Deploy front-end and back-end components.
   * Ensure proper API integration with grocery store systems.
3. **Data Migration:**
   * Import existing recipe and grocery inventory datasets.
   * Verify data accuracy and consistency post-migration.
4. **Final System Checks:**
   * Conduct pre-launch system testing to validate all functionalities.
   * Perform a security audit and optimize app performance.
5. **User Training & Documentation:**
   * Provide user guides, video tutorials, and FAQs.
   * Offer live training sessions for early adopters.

**Handoff plan.**

Once the app is live, it needs ongoing support to stay functional and relevant. Here’s how that can be handled. It has been broken down into documents in different areas for easy delegation.

***Documentation for Handoff:***

1. **Technical Documentation:**
   * API documentation for developers.
   * Database schema and ERD for database administrators.
   * System architecture and server setup guides.
2. **User Documentation:**
   * Step-by-step user manuals and troubleshooting guides.
   * FAQs and instructional videos for app navigation.

***Post-Deployment Support:***

1. **Bug Tracking & Fixes:**
   * Implement a ticketing system like Trello for tracking issues.
   * Assign a support team for ongoing maintenance.
2. **User Feedback Collection:**
   * Conduct surveys and app store reviews analysis.
   * Use feedback for future feature enhancements.
3. **Regular Updates & Security Patches:**
   * Release periodic updates for feature improvements.
   * Apply security patches to protect user data and maintain compliance.

# Summary and Reflection

**Iteration 1:**

Through this iteration 1, I was able to select the project and defined it as well as s the Intended Business and User Community. I was also able to come up with a SDLC framework for the project. The project “The Smart Grocery Shopping App” is about and designed to help users streamline grocery shopping through recipe-based shopping lists, budgeting tools, and diet tracking. The question I came up with is how the app would be hosted and become available for the users such as in App store and play store.

**Iteration 2:**

During the second iteration, I conducted a stakeholder analysis and identified the various stakeholders involved in the "Smart Grocery Shopping App," including actual shoppers and those responsible for developing the app. Additionally, I outlined both functional and non-functional requirements through the requirements-gathering process. One question I have out of curiosity is how APIs can be designed to enhance accuracy and provide location-based recommendations in the future.

**Iteration 3:**

Through this iteration 3, I completed the Entity-Relationship Diagram (ERD) model for the project. I was able to make a list of needed entities and their structure business rules relationships. I was also able to visualize the ERD for the model.

Throughout the process, I had a few questions, such as including the necessary attributes for the entities generated. I designed ERD so that I can quickly add them. In contrast to the physical diagram, where we add attributes and create a cross-reference table for M: N relationships. For example, the entities Recipes and Ingredients have many-to-many relationships. So overall I had concerns of how to model the system visually well without over doing it since we won’t need to necessarily make a database for the project.

**Iteration 4:**

In this iteration, I focused on designing the Physical **Entity-Relationship Diagram (P-ERD)** and defining the **system architecture** for the Smart Grocery Shopping App. I identified the necessary entities, their attributes, and relationships, ensuring the design was simple yet functional, given that a full database may not be needed, and we can always add a necessary attribute.

For the system architecture, I structured it with a three-tier approach, including the front-end, back-end, and database. One challenge I faced was balancing the level of detail in both the ERD and the architecture without overcomplicating them and mostly using the tools or frameworks that I know or aware of for easy understanding. I also had to consider how to implement real-time inventory updates and whether the database should be scalable or kept simple for the project's current scope.

Didn’t purposely change the stakeholder map to remove the gridlines because the tool I used required a license for full experience. I will be using Draw.io to recreate the map/matrix and updating it in the next iteration 5.

**Iteration 5:**

In this final iteration, I went through the whole project and double checked the four previous iterations before for clarity. In this final one, I completed the quality assurance and the implementation handoff on how the project should be deployed and managed daily form both ender users and different stakeholders. Through the test scenarios I can always add on one depend on the user’s feedback in the handoff- post-deployment support to keep improving the app according to users need.

I am confident in the developing of the app and can see how both the backend and frontend will come together to get the application running while also remaining open for updates. For recap, by integrating intelligent recipe suggestions, shopping list automation, real-time inventory tracking, and budget control, the app should enhance user convenience and efficiency. With thorough testing, structured deployment, and ongoing support, the Smart Grocery Shopping App is poised to offer a seamless and efficient grocery shopping experience.