SHAREBOX Application

PRODUCT DESIGN SPECIFICATION

Version <1.1>

March 16 2025

Lead Developer: Nini Ola

Revision HIstory

|  |  |  |  |
| --- | --- | --- | --- |
| Date | Description | Author | Comments |
| February 3 2025 | Version 1 | Nini Ola | First version |
| April 6 2025 | Version 1.1 | Nini Ola |  |

TABLE OF CONTENTS

## **Table of Contents**

* **Revision History** ......................................................................................... 2
* **Introduction** ................................................................................................. 3
* **Purpose** ....................................................................................................... 3
* **System Design** ............................................................................................ 4
* **Architecture Design** .................................................................................... 5  
  + Hardware Architecture .............................................................................. 5
  + Software Architecture ............................................................................... 6
  + Security Architecture .............................................................................. 14
  + Communication Architecture ................................................................. 14
* **Sequence Diagrams** ................................................................................... 16
* **Use Cases** .................................................................................................. 20
* **Database Design** ...................................................................................... 23
* **API Reference** ............................................................................................ 24
* **Class Diagrams** ......................................................................................... 29
* **User Interface Design** .............................................................................. 31

Introduction

1.1 Purpose of the Document

The purpose of this design document is to provide a comprehensive overview of the necessary information required to define the hardware and software architecture for the ShareBox system. This document outlines the system’s design principles, key components, interfaces, and security considerations to ensure an efficient, scalable, and secure food distribution solution. It aims to communicate the technical details of the ShareBox App to the intended audiences and stakeholders.

1.2 Intended Audience

This document is intended for future stakeholders who may be interested in the Sharebox application and its integration across various sectors. The target audience includes:

* Government Officials – Individuals seeking to integrate the Sharebox application or web extension into county websites or community portals to support food accessibility and resource-sharing initiatives.
* Donation Centers – Organizations that wish to participate in or manage local Share Box units, helping to streamline donation intake and distribution.
* Public Libraries – Community hubs that can serve as trusted locations for hosting Sharebox units or supporting digital access to the platform.
* Software Developers – Engineers responsible for implementing and maintaining the Sharebox web extension, mobile application, and server-side infrastructure.
* Embedded Systems Engineers – Professionals involved in developing the ESP-based microcontroller firmware, ensuring reliable sensor and locking mechanism integration.
* Project Stakeholders – Individuals and organizations invested in understanding the system’s design goals, functionality, and constraints, including sponsors and community advocates.
* Hardware Designers – Engineers and designers tasked with constructing and optimizing the physical Share Box units, including structural design and electronic lock systems.

**2.1 General Overview**

The Sharebox system is designed to optimize the current item donation process for all donation centers, whether Sharebox-certified or not. It provides a secure, automated, and accessible method for distributing food to those in need. The system leverages WiFi-enabled ESP controllers to manage smart storage boxes, integrates machine learning models to ensure donation quality, and features a user-friendly interface. It also offers tax benefits to donors while enabling remote authentication and secure access through a web extension.The design prioritizes ensuring that only authorized users can access the food boxes, maintaining stable and fault-tolerant system operation Supporting multiple food boxes across various locations and Providing a simple and intuitive interface for both administrators and users.

2.2 Design Guidelines & Approach

The Share Box system follows a modular and microservice architecture. The system is divided into three main components: the hardware layer, the software layer, and the communication layer. The Hardware Layer consists of the electronic locking mechanism controlled by a WiFi-enabled ESP microcontroller. It follows a Sensor-Signal-System design, where sensors detect interaction, signals are processed by the ESP, and the system responds with appropriate locking or unlocking actions. The Software Layer is Built using a microservice architecture, this layer is composed of several independent components: A user-friendly interface for administrators, receivers and donors.A machine learning model API responsible for verifying the quality and condition of donations.A Flask integration module that serves as the bridge between the front-end interface and the backend machine learning services.Lastly Firebase is used as the central backend service for: User Authentication, Realtime Database and Cloud Storage. The communication layer manages data exchange between the software and hardware components using RESTful APIs and MQTT protocols to ensure real-time synchronization, low-latency control, and secure communication. The communication layer facilitates seamless communication across multiple all relevant components This architecture allows for scalability, modular reuse, independent updates, and user-friendly experience across all app functionalities.

## 

2.3 Assumptions

## Stable Wi-Fi Connectivity The ESP controllers are assumed to have consistent access to a Wi-Fi network for real-time communication with the Sharebox application and Firebase Realtime Database.

## Controlled Access by Trusted Users The system is intended for use by authorized individuals who must have registered and verified accounts. Verification includes providing a valid ZIP code and a photo ID.

## Power Supply Availability Each food donation box is expected to have access to a reliable power source, either through a rechargeable battery or a direct power adapter, to ensure continuous operation of the ESP controller and electronic locking mechanism.

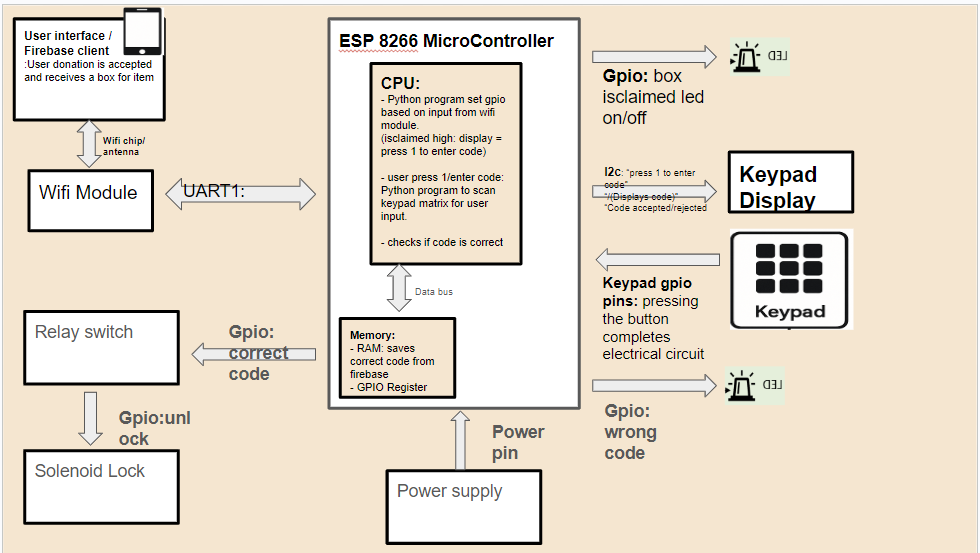
3. Architecture Design

The Share Box system is composed of Various Software, Hardware, Security,and Communication components that work seamlessly together. The hardware component consists of the The ESP controller (8266) as the core hardware component that connects the web application with the physical locking mechanism of the food bank box. It acts as a bridge between the cloud-based system and the physical hardware, handling authentication, communication, and access control. The keypad is a crucial input device that allows users to enter an access code. It communicates with the microcontroller (ESP8266 ), which verifies the code and controls the solenoid lock securing the donations. And lastly A stable power source (battery or adapter) is used to operate the ESP controller and locking mechanism.

3.1 Hardware Architecture

|  |  |  |  |
| --- | --- | --- | --- |
| Component | Hardware Interaction | Processing | Performance |
| ESP8266 Controller | Phone/server/UIWE: wifi module  Relay switch: GPIO pin  4x4 Keypad: GPIO input pins  UART pin  Power supply  Led indicator: input pins | Handles communication between the web app and the lock mechanism.  Stores and verifies the access code.  Controls the locking/unlocking mechanism.   * The User interface business logic sends the phone number of the user claiming the box and sends it over HTTP or MQTT. * The ESP stores the valid access code in its memory (EEPROM) until it expires or is used. * The user enters the phone number on the keypad which Uses gpio mapping to determine the user input using the keypad gpio pins * The ESP reads the input and compares it with the stored code.  Step 3: Code Verification & Lock Control  * If the entered code matches, the ESP activates the relay module, which unlocks the solenoid lock * If the code is incorrect, the ESP rejects the attempt and triggers the LED indicator | The ESP8266 microcontroller is chosen due to its built-in WiFi capabilities, low power consumption, and GPIO control, which makes it ideal for the Electronic lock mechanism and share box system |
| 4x4 Keypad | User input Gpio pins: gpio mapping  Esp controller: Gpio input pins | Keypad (4x4 Matrix) or Touchscreen   * Users can manually enter the code. * Uses GPIO for communication. * The esp controller uses gpio mapping to determine the input * Gpio mapping | 4 Rows (R1-R4) → Connected to GPIO input pins.   * 4 Columns (C1-C4) → Also connected to GPIO pins. * The microcontroller scans the columns while grounding each row one by one to detect a keypress. |
| Relay | Esp controller cpu  Solenoid lock | Once the microcontroller CPU detects the correct code it sends a signal to the relay switch |  |
| Solenoid lock | Relay : high voltage from gpio pin | Solenoid Lock: Uses a magnetic field to lock/unlock the box. Requires a relay to switch it on/off. |  |
| Power supply | ESP Controller, Relay, Solenoid Lock | 1️ Converts AC to DC (if adapter)  2️Supplies 5V to ESP & relay, 12V to solenoid  3️Powers up the entire system  5V/2A Power Adapter or Li-Ion Battery + Voltage Regulator   * ESP8266 operates at 3.3V (can use AMS1117 regulator). * Solenoid locks require 12V (use step-down buck converter if needed). |  |

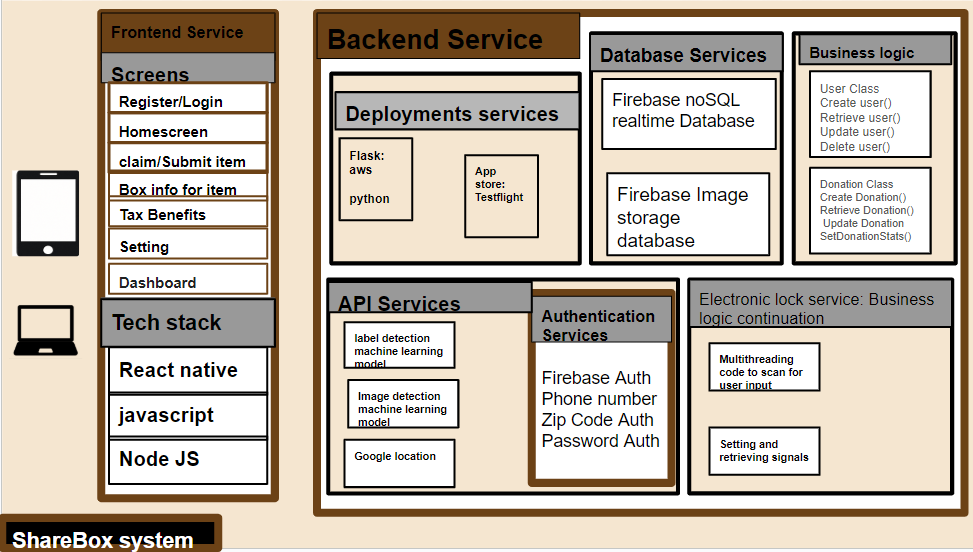
Hardware Architecture Diagram



3.2 Software Architecture

The software architecture follows a hybrid of 2 layers: a front end layer and a backend micro service layer.The Sharebox system software layer follows a microservice architecture, consisting of the User Interface (UI): A simple, user-friendly platform for administrators and donors to interact with the system, Machine Learning Model API: Verifies the quality of donations and ensures items meet requirements before approval.Flask Integration: Facilitates communication between the front-end UI and the backend machine learning models and the Firebase Integration, which handles real-time data synchronization and storage of user data, donation records, and box statuses. This modular software design enables seamless updates, scalability, and efficient user interaction.

|  |  |  |  |
| --- | --- | --- | --- |
| SOFTWARE COMPONENT layers | Structure | Technology & dependencies | Interactions |
| Quality Assurance Object detection model - Quality assurance layer | Microservices   * Data preprocessing * Data training and testing * Integration into app | * python | Input:  Output:  Processing |
| UIWE- User interface layer | Microservices   * Login * Homepages * Profiles * settings | Reat native  Node js | Input:  Output:  Processing |
| Esp8266 controller software layer | * Gpio mapping * Code generation * Access code verification * Box status | Python | Input: Access code verification Routes request to Box Control Service Box unlock status  Output:  Processing |
| Firebase: data storage layer | NoSQL database  Microservices:   * Authentications services * Logging analytics service * Photo saving and retrieval | Realtime database | Input: User authentication request  Output:Verifies with Firebase Auth  Processing: Access token or error  Email & password Validates with Firebase Auth Access token or error |

Software Architecture Diagram

3.3 Security Architecture:

|  |  |  |
| --- | --- | --- |
| **Component** | **Description** | **Purpose** |
| **Password-Protected Accounts** | Requires users to log in with a unique username and password. | Ensures only authorized users can access personal or system data. |
| **Secure Passcode-Controlled Share Boxes** | Hardware or digital lockers with passcode verification before access. | Protects physical or digital items from unauthorized access or tampering. |
| **Secure Database** | Databases protected with encryption, access controls, and user-level security. | Safeguards stored data from breaches, leaks, and unauthorized manipulation. |

3.4 Communication Architecture:

| **From** | **To** | **Protocol/Method** | **Purpose** |
| --- | --- | --- | --- |
| **Mobile App** | **Firebase Realtime Database** | HTTPS ( Firebase SDK) | Store and retrieve user inputs, item data, and system state in real-time. |
| **Firebase Realtime Database** | **ESP32/ESP8266 Microcontroller** | HTTPS | Send control commands (e.g., lock/unlock box, accept/reject donation). |
| **ESP Microcontroller** | **Relay Module/Solenoid Switch** | GPIO + Relay Driver Circuit | Physically actuates the locking/unlocking mechanism on a smart box. |
| **ESP Microcontroller** | **Firebase (status updates)** | HTTPS | Reports status (e.g., box opened, item received) back to the database. |
| **Firebase Authentication** | **Mobile App** | Firebase Auth SDK | Authenticates user login and manages secure sessions. |
| **Firebase Storage** | **Mobile App** | Firebase Storage SDK | Handles image upload/download for items (e.g., pictures of donations). |

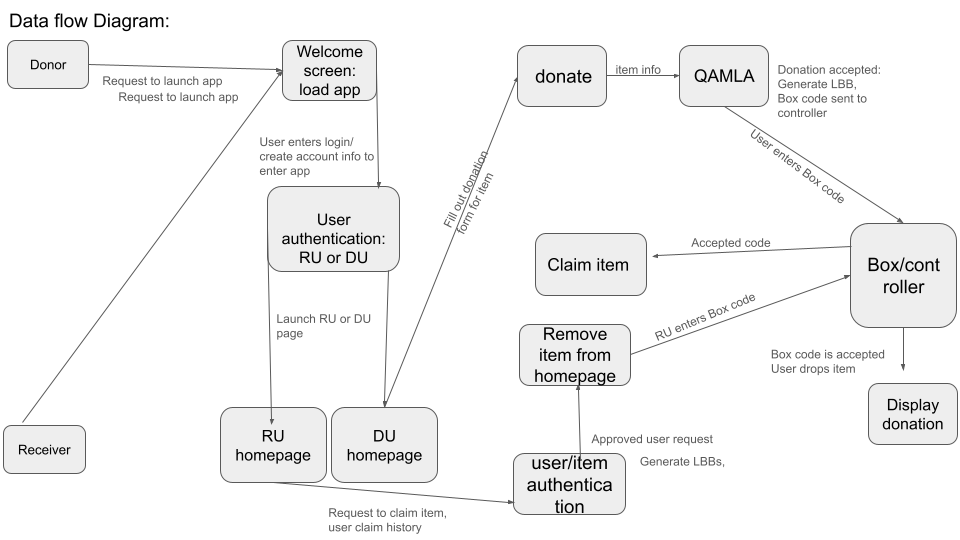
4. System Design:

4.1 Use Cases: Clearly define the various ways users interact with the system.

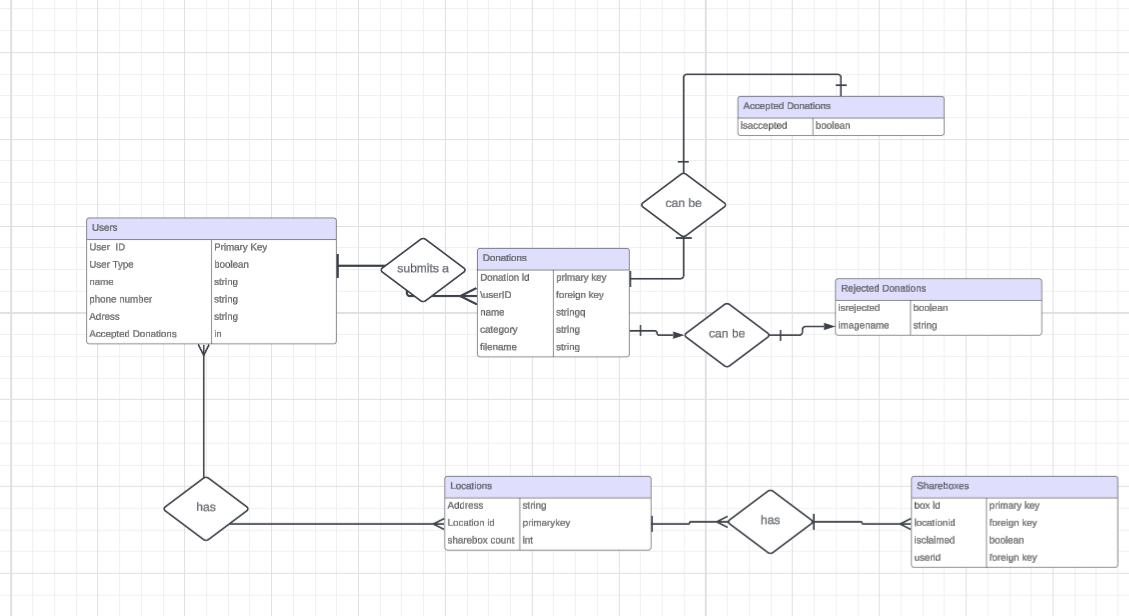
|  |  |  |
| --- | --- | --- |
| use case |  |  |
| precondition |  |  |
| Trigger |  |  |

4.1 USE-CASES DIAGRAM

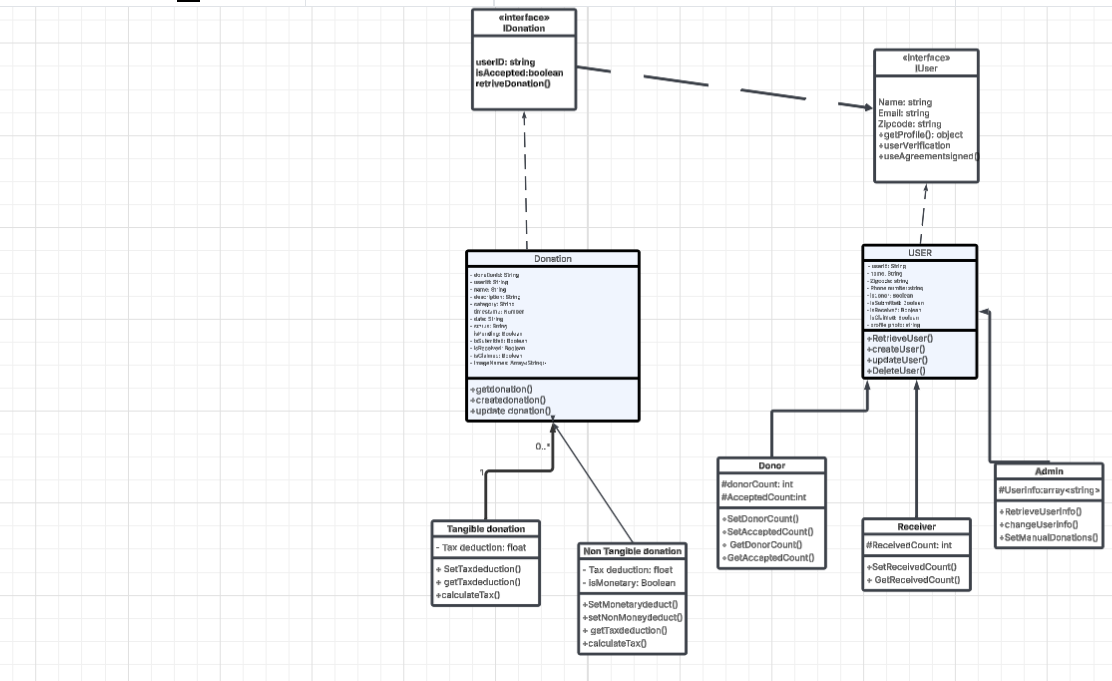
4.2 DATA FLOW DIAGRAM



4.4 DATABASE DESIGN



4.5 CLASS DIAGRAM



4.6 APPLICATION PROGRAM INTERFACE

|  |  |  |
| --- | --- | --- |
| **API / Tool** | **Purpose** | **Usage in Sharebox App** |
| **Firebase Realtime Database API** | Store and retrieve categorized donation items | /donations/{Category}/{donationId} entries, with fields like name, description, etc. |
| **Firebase Storage API** | Upload and retrieve item images | Stores images in stuff/{fileName}, retrieves them to display on HomeScreen |
| **Firebase Authentication API** | Manage user accounts and secure access | (If used) Authenticates users and ties items to userId |
| **Item Classification API (Text-based)** | Classifies items using name and description | Determines if the item is a shirt, can of food, etc., based on input text |
| **Image Classification & Object Detection API** | Inspects item images for type and quality issues | Detects tears in clothing, damaged packaging, missing parts, etc. |
| **Google Maps API** | Location services | Can be added to show drop-off locations for donations |

4.7 User Interface Design: Describe the design principles and layout of the user interface. The layout of the user interface follows a simple and straightforward design that allows all the functionality needed for a successful interaction with the application

|  |  |
| --- | --- |
| Design Principle | Description |
| Simplicity & Clarity | The UI should be intuitive and easy to navigate, with clear labels and minimal distractions. |
| Accessibility | Ensure usability for all users, including those with disabilities (e.g., high contrast mode, voice navigation). |
| Consistency | Maintain a uniform design language across all screens (colors, fonts, buttons). Use Material Design or iOS guidelines. |
| Responsiveness & Adaptability | The interface should work seamlessly on mobile devices, tablets, and desktops. |
| Error Prevention & Feedback | Provide clear error messages and confirmations (e.g., “Invalid Code” with guidance on next steps). |
| Aesthetic & Emotional Design | Use friendly, visually appealing colors and animations to create a positive user experience. |