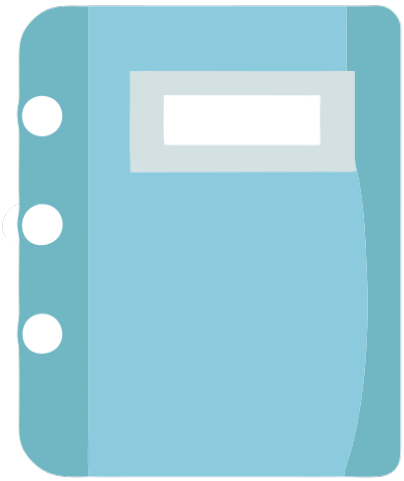

Acknowledgement Book

A Digitized Solution for Paper Acknowledgement Book

Version 1.0

Software Documentation

For more, visit: www.peonbook.subharti.org

Contents

Introduction

1 Overview

The **Acknowledgement Book** is an integrated software solution developed to manage and streamline the physical transfer of documents, files, and papers or any other physical entity between departments within an organization. In many organizations, handling physical documents is an essential task, often requiring a well-defined process to ensure that documents are correctly transferred and acknowledged upon receipt. This software aims to replace the traditional, paper-based acknowledgment system by providing a digital platform that records and tracks the movement of documents between departments.

This system ensures transparency, accountability, and traceability in document transfers. It assists both the sender and the receiver, while also ensuring that the peons or office assistants responsible for delivery are included in the acknowledgment process. Each document or file transferred will be logged into the system, and every step of its journey from the originating department to the receiving department will be tracked, recorded, and acknowledged digitally, thereby minimizing the risk of lost or misplaced documents.

2 Project Background

In many organizations, the physical transfer of files, papers, and official documents plays a critical role in daily operations. However, existing manual processes for managing these transfers often lack efficiency and traceability. Problems such as lost or delayed documents, missing acknowledgements, and difficulties in tracking a document's location between departments frequently arise, leading to operational slowdowns and compromised productivity.

The Acknowledgement Book was conceived to address these issues by creating a centralized, digital platform for recording and monitoring all document transfers. The software ensures that both the sender and the receiver properly acknowledge the handover of documents, creating a clear, verifiable chain of custody. This system will assist in preventing miscommunication between departments, minimizing the risk of losing important paperwork, and providing a clear record of every transfer for auditing and review purposes.

3 Scope

The scope of the Acknowledgement Book system is to manage the transfer of physical documents within an organization. It tracks and monitors document movement from the time it leaves one department until it is received and acknowledged by another. The system is particularly focused on the physical handling of papers and files, ensuring that accountability is maintained throughout the transfer process.

The Acknowledgement Book is not designed for electronic document management or cloud storage systems. It strictly handles physical documents and ensures that their transfer between departments is accurately recorded. Future expansions could potentially include integration with electronic document management systems for hybrid document flows, but this version is focused solely on physical document logistics.

System Architecture

1 System Overview

The Acknowledgement Book system is an application developed using Flutter for the front-end, Node.js for the backend, and SQL Server as the database. It follows a client-server architecture where the mobile app interacts with the backend to manage document transfers between departments. The system is designed to track and manage the flow of physical documents, such as files and papers, by logging their transfer and acknowledgment at every stage of the process.

This application is accessible to all users without role-based access control. The focus is on maintaining traceability of documents as they move between departments. Users are required to register with the system and can then log in to manage the sending, receiving, and tracking of documents.

Key Architectural Layers

- **Presentation Layer (Flutter)**: The mobile interface where users can register, log in, send documents, receive documents, and track their status.
- **Business Logic Layer (Node.js)**: Handles core application logic, including processing document transfers, acknowledgments, and user actions.
- **Data Layer (SQL Server)**: Responsible for persisting data related to users, documents, and their movement history.

2 Key Components

The **Acknowledgement Book** system is composed of the following key components:

1. **Registration Module**:
    - **Functionality**: Allows new users to register with the system.
    - **Component Description**: Users provide personal details such as name, department,center, and contact information to create an account in the system. Once registered, they can access all functionalities (login, send, receive, track).
    - **Interactions**: Sends user data from the Flutter app to the Node.js backend, which stores the new user details in the SQL Server database.
2. **Login Module**:
    - **Functionality**: Facilitates user authentication to access the system.
    - **Component Description**: Users log in by providing their registered credentials. Once authenticated, they can manage document transfers, deliveries, and tracking.
    - **Interactions**: Verifies credentials against the SQL Server database and grants access upon successful authentication.

3. **Send Module**:
    - **Functionality**: Allows users to initiate the transfer of documents, objects etc. to another department.
    - **Component Description**: Users have to create a new entry for the document or object to be sent by filling deatils as To, From, Content and specify the Content Code which can be automatically generated or entered manually, and initiate the transfer. The system logs the transfer, updates the document's status as "Pending".
    - **Interactions**: Sends document details and transfer information to the backend for logging in the SQL Server database.
4. **Deliver Module**:
    - **Functionality**: Manages the delivery of documents between departments.
    - **Component Description**: Peons (or office assistants) use this module to mark the delivery of a document. This module tracks the physical movement of the document.
    - **Interactions**: Updates the document's delivery status in the SQL Server database, confirming its transfer from one department to another.
5. **Receive Module**:
    - **Functionality**: Enables users in receiving departments to acknowledge receipt of a document.
    - **Component Description**: Users can confirm the receipt of a document, updating its status in the system. This ensures accountability and logs the successful handover.
    - **Interactions**: The database is updated with the document's receipt status once acknowledged by the receiving department.
6. **View All Module**:
    - **Functionality**: Provides users with a comprehensive view of all documents associated with their department.
    - **Component Description**: Users can filter and view the status of documents (e.g., sent, received, pending). This gives users insights into the document flow and their department's document management status.
    - **Interactions**: Fetches document data from the backend, which retrieves it from the SQL Server database, and displays it in the app.
7. **Track Module**:
    - **Functionality**: Tracks the current status and movement history of a document as current user or department having the document or object and can also track the cycle number of the file.
    - **Component Description**: Users can trace a document's journey, including timestamps, department handovers, and delivery details.
    - **Interactions**: Queries the backend to retrieve tracking data and history, which is fetched from the SQL Server database and displayed to the user.

3 Interactions and Data Flow

The system ensures a seamless interaction between the client app (Flutter), the backend server (Node.js), and the SQL Server database. These interactions manage everything from user registration and document transfers to tracking and acknowledgment.

Key Interactions:

- **Registration and Login**: New users register their details, which are stored in the SQL Server database. Upon login, credentials are verified against the database to allow access.
- **Sending Documents**: A user initiates the sending of a document, which is logged in the database. The recipient is notified, and the document status is updated.
- **Receiving Documents**: The recipient acknowledges the document via the app, triggering an update to the document's status in the database.
- **Delivering Documents**: Delivery information, such as timestamps and destination departments, is logged as peons hand over the documents.
- **Tracking Documents**: Users query the system for document status and history, which the backend retrieves from the database for display.

Data Flow Diagram: The data flow diagram (DFD) can represent how data moves between the app, server, and database:

- **User inputs** (registration, document send, receive, tracking)
- **Data exchanges** (registration details, document metadata, tracking status)
- **Database operations** (insert new users, update document status, retrieve tracking history)

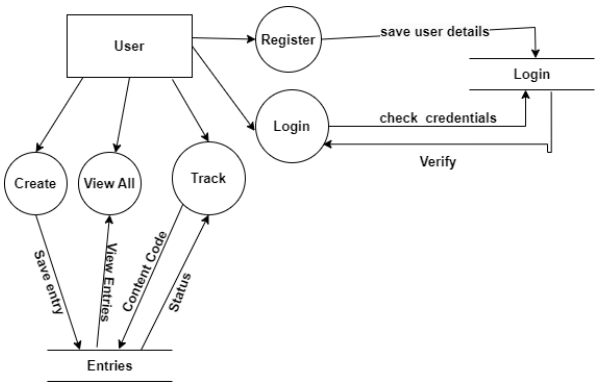

Figure 1: DFD for Office Users

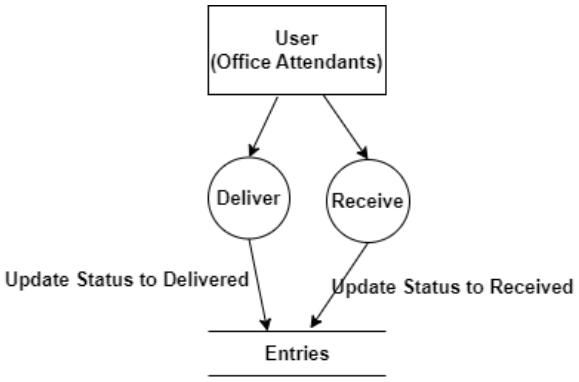

Figure 2: DFD for Office Assistants

Functional Requirements

1 Functional Overview

The *Acknowledgement Book* system is designed to manage the process of sending, receiving, delivering, tracking, and acknowledging documents between departments within an organization. The core functionalities of the system include the following:

1.1 Login

Description: The login functionality ensures secure access to the system by authenticating users through their credentials. Each user (e.g., administrative staff, delivery personnel) must log in with a unique username and password.

Purpose: To provide access control, ensuring that only authorized personnel can use the system. The login feature also associates actions such as sending, receiving, and tracking documents with specific users.

1.2 Send Document

Description: This feature allows users to send a document to another department. The sender creates the entry for the document, chooses the department it should be sent to, generates the content code, fills in all the details, and confirms the dispatch. The document is then added to the system's dispatch queue.

Purpose: To streamline the dispatch process, ensuring that the right document reaches the intended department efficiently.

1.3 Receive Document

Description: The receiving department acknowledges the receipt of the dispatched document once it has been delivered by the delivery personnel. The receiver confirms the receipt in the system, which updates the document's status to "Received."

Purpose: To confirm the successful delivery of the document, ensuring that it has reached the intended recipient.

1.4 Deliver Document

Description: This functionality allows delivery personnel to update the system regarding the status of the dispatched document. They can mark it as Delivery By <Name of Delivery Personnel>.

Purpose: To provide real-time updates on the delivery status, allowing for better tracking and accountability in document handling.

1.5 Track Document

Description: This feature allows users (both senders and receivers) to track the current status of a document in real-time. The document's lifecycle, from dispatch to delivery and acknowledgment, can be monitored through this functionality.

Purpose: To provide transparency in the document dispatch process and ensure accountability for all stages of delivery.

1.6 View All Documents

Description: This functionality provides users with an overview of all the documents they have interacted with. Users can see a history of documents they've sent, received, or are tracking, along with their current status.

Purpose: To allow users to review and manage the documents associated with them, ensuring they can easily track and audit the flow of documents.

Non-Functional Requirements

Non-functional requirements define the quality attributes, performance benchmarks, and other criteria that the *Acknowledgement Book* system must meet to ensure smooth operation, scalability, security, and usability. Unlike functional requirements, they describe *how* the system performs its tasks rather than *what* tasks it performs. These requirements are essential for maintaining system performance and user satisfaction across various use cases.

1 Performance Requirements

Description: The system must operate efficiently, providing quick response times to user actions, regardless of the volume of data being processed. The performance of critical features, such as document tracking and status updates, should be consistent across all user devices and network conditions.

- **Response Time**: All user actions (e.g., sending, receiving, and tracking documents) should be completed within 1-2 seconds under normal load conditions.
- **Concurrency**: The system should support at least 100 concurrent users without noticeable degradation in performance.
- **Data Processing**: The system must be capable of handling large volumes of document-related data, ensuring rapid retrieval and updates.
- **Uptime**: The system must maintain a minimum uptime of 99.5%, ensuring continuous availability of critical functions.

2 Scalability

Description: The *Acknowledgement Book* system must be scalable to accommodate increasing numbers of users, documents, and departments as the organization grows. It should support vertical and horizontal scaling to meet future demands without compromising performance.

- The system should handle a *10x increase* in the number of documents processed per day without requiring significant infrastructure changes.
- *Elastic scaling* of the server should be possible to handle peaks in demand (e.g., end-of-quarter paperwork spikes).
- Database architecture should be optimized to support growth in the number of records without significant changes to query performance.

3 Security

Description: Security is critical to ensure the confidentiality, integrity, and availability of sensitive documents and user data within the system. The system must implement robust authentication, authorization, and encryption mechanisms to safeguard all interactions and data transmissions.

- **Authentication**: Strong password policies and optional two-factor authentication (2FA) for users to prevent unauthorized access.
- **Authorization**: Access controls ensure that users can only interact with documents relevant to their department and role.
- **Audit Logs**: The system should log all critical actions, such as document dispatch, delivery, and acknowledgment, with timestamps and user information for auditing purposes.
- **Data Backup**: Regular backups of the database must be scheduled to ensure that no data is lost in the event of system failure.

4 Reliability and Availability

Description: The system must be highly reliable to ensure continuous operation and avoid any loss of critical data. This includes robust error handling, failover mechanisms, and minimal downtime for maintenance or updates.

- **Fault Tolerance**: The system should be capable of recovering from common failures such as server crashes or network interruptions without data loss or prolonged downtime.
- **Redundancy**: Data redundancy should be employed to ensure that no single point of failure exists within the system infrastructure.
- **Scheduled Downtime**: Any planned downtime for system updates or maintenance should be minimal and communicated in advance to users.
- **Recovery Time**: In the event of system failure, recovery must be completed within 1 hour, ensuring that critical document management activities can continue without significant disruption.

5 Usability

Description: The system must be user-friendly and intuitive to ensure that users with varying levels of technical expertise can easily perform their tasks. A simple and clear user interface (UI), combined with a responsive design, is essential for a smooth user experience.

- **Intuitive Design**: The interface should be designed in such a way that users can quickly learn how to use the system without extensive training.
- **Responsive Layout**: The UI must be responsive and accessible across various devices (desktops, tablets, mobile phones), ensuring that users can interact with the system from anywhere.
- **Error Handling**: Clear and user-friendly error messages should guide users to resolve issues independently without requiring technical support.
- **Accessibility**: The system should follow accessibility standards (e.g., WCAG 2.0) to ensure it is usable by people with disabilities.

Design Specifications

1 System Design

The *Acknowledgement Book* system is designed using a combination of **Flutter**, **Node.js**, and **SQL Server**, ensuring a robust, scalable, and maintainable architecture. The design follows key architectural patterns that prioritize performance, security, and modularity.

1.1 Frontend - Flutter

The frontend of the system is built using **Flutter**, a cross-platform framework. The application leverages **state management** to manage the application's state efficiently across multiple pages. This structure ensures a clean separation between UI components and the logic controlling them.

- **Pages and Utilities**: The design separates the application's pages (e.g., Login, Send Document, Track Document) from utility functions, providing a modular structure that enhances maintainability and ease of updates.
- **HTTP Requests**: The system interacts with the backend by making **REST API calls via HTTP requests** to retrieve and update data dynamically.

1.2 Backend - Node.js

The backend is powered by **Node.js**, chosen for its asynchronous, event-driven architecture, which supports high concurrency and scalability.

- **Password Hashing**: User credentials are securely stored using robust password hashing mechanisms to protect against unauthorized access.
- **Rate Limiting and CORS**: The system employs **rate limiting** to protect against abuse and **Cross-Origin Resource Sharing (CORS)** to ensure secure communication between the frontend and backend.
- **Multithreading Support**: To enhance performance under load, the Node.js backend is designed with **multithreading** to efficiently handle multiple requests concurrently.
- **Routes and Models**: The code is organized into separate **routes** and **models**, ensuring a clear structure for handling requests and interacting with the database. Routes handle incoming API requests, while models define the schema for data stored in the database.

1.3 Database - SQL Server

The **SQL Server** database is used for persistent data storage. It handles all document-related data, including users, departments, and tracking information. The database is designed with efficient indexing, ensuring rapid query responses and supporting high performance even as the dataset grows.

This design ensures a modular and secure system that is scalable and optimized for performance. The separation of concerns across different technologies (Flutter, Node.js, SQL Server) enhances the system's flexibility and maintainability.

2 Database Design

2.1 ER Diagrams and Schemas

The database for the *Acknowledgement Book* system is designed using a relational model, with entities representing departments, users, documents, and document statuses. Each entity has defined relationships to ensure data integrity and efficiency. The key features of the database design include:

- **Documents Table**: Holds information about each document, including the sender, receiver, status, timestamps, and tracking information.
- **Departments Table**: Contains details about each department, allowing the system to manage document flow between various units within the organization.
- **Users Table**: Stores user credentials and profiles, including information about their roles and departments.
- **Status Table**: Tracks the status of documents, with entries such as "Sent," "Received," "Delivered," etc.

The database design follows normalization principles to eliminate redundancy and improve data integrity. The database schema includes **primary keys**, **foreign keys**, and **indexed columns** to enhance performance during query execution.

3 API Design

3.1 Endpoints and Formats

The system provides a set of RESTful APIs that allow for interaction between the client application (frontend) and the server. These APIs enable the system to manage documents, users, and departments effectively. Each API follows a standardized request/response format (*JSON*), ensuring compatibility and ease of integration with future enhancements or third-party services.

3.1.1 Key API Endpoints

- **POST /login**: Authenticates users based on their credentials.
- **POST /sendDocument**: Sends a document to a department, updating the system's record.
- **GET /trackDocument/{id}**: Retrieves the current status of a specific document by its ID.
- **GET /viewAllDocuments**: Provides a list of all documents with their statuses for the logged-in user.
- **PUT /updateDeliveryStatus**: Updates the delivery status of a document.

Each API is designed with error handling in mind, providing meaningful responses for both successful and failed requests, ensuring users receive clear feedback in case of issues (e.g., invalid document ID, unauthorized access).

Future Enhancements

The *Acknowledgement Book* system is designed with future growth and functionality in mind. The following features are proposed for future releases to improve usability, enhance performance, and support additional functionalities.

- **Role-Based Access Control (RBAC)**: Introduce role-based access control, allowing certain features to be restricted to specific user roles (e.g., admin, department head).
- **Document Encryption**: Add end-to-end encryption for sensitive documents during transfer between departments to enhance security.
- **Notification System**: Implement real-time notifications for users when a document has been dispatched, received, or delivered.
- **Advanced Search and Filters**: Enable users to search documents by various parameters (e.g., department, date, status) and apply filters to narrow down the document list.
- **Mobile Offline Mode**: Allow users to access certain features of the application in offline mode and sync data when an internet connection becomes available.
- **Custom Reporting**: Add the ability for department heads to generate custom reports on document handling efficiency, pending tasks, and document history.

These proposed enhancements aim to evolve the system to better meet the needs of the organization and ensure the *Acknowledgement Book* remains a vital tool for document management.