Politecnico di Torino

Projects and Laboratory on Communication Systems

Inventory Management System
Project Report

Authors:

Andreas Hølleland, Marcus Alexander Tjomsaas

Version: 1.1

Date:

15/07/2022

Contents

1 Introduction

This section aims to give a general overview of the system, explain any assumptions made, and introduce the main aspects of the implementation by presenting the achieved functionality of the system with respect to the given functional requirements.

1.1 Overview

The Inventory Management System (IMS) is a multi-platform application for managing a physical inventory of items and is designed to be as generic as possible to be applicable in a wide variety of use-cases/businesses.

The general idea is that a customer (anything from a large business with warehouses to a single person managing a shop), can purchase this software, register their items with RFID tags and manage their inventory. Users of the software can buy/borrow/return items from any registered customer with the application, which can be accessed through a website on any device, or native applications running on a PC (Windows/Linux/Mac) or a smartphone (Android/Iphone), as well as a dedicated on-site totem with a touch-screen and an RFID reader.

A customer can use the system with only a totem and still be able to get all functionality of the application, the only other requirements for the customer is to have an internet connection and RFID tags for each item. However, the system can also be used completely without a totem, where the only device needed is a phone/pc with RFID/NFC hardware, that can natively run the app (see the mentioned platforms above).

1.2 Requirements

Below is a table of all the given functional requirements extracted from the assignment description slides, showing which ones have been implemented.

There are a few things that has been changed with respect to the assignment brief that were considered vague or unnecessary, in particular the requested features of each individual platform, the account type that are supposed to access them and their possibilities within each interface.

ID	Description	Implemented
FR1	**User account**	YES
FR1.1	Get/return items	YES
FR1.2	Register to one or more customers	YES
FR2	**Customer account**	YES
FR2.1	Add/remove/reset its own users	YES
FR2.2	Add/remove/modify items	YES
FR2.3	Play the role of any of its users	YES
FR2.4	Get/return items for its users	YES
FR2.5	Check items (location, presence/absence)	YES
FR3	**Admin account**	YES
FR3.1	Add/remove/reset customers and users	YES
FR3.2	Play the role of any customer/user	YES
FR4	**Web interface**	YES
FR4.1	Admin/customer/user can use it	YES
FR4.2	Authentication through username/password	YES
FR4.3	Authentication through social profile	NO
FR4.4	Authentication through certificate	NO
FR5	**Totem interface**	YES
FR5.1	Customer/user can use it	YES
FR5.2	Authentication through username/password	YES
FR5.3	Authentication through RFID	YES
FR5.4	Authentication through smartphone	NO
FR6	**Smartphone interface**	YES
FR6.1	Customer and User can use it	YES
FR6.2	First time authentication through username/password	YES
FR6.3	First time authentication through RFID	YES
FR6.4	Log-in authentication through PIN	NO
FR6.5	Log-in authentication through biometric data	NO

Table 1 - Functional requirements

In the given requirements, some of the interfaces/platforms seem to be restricted to certain users and have limited functionality (see FR6.1 for instance). However, in this implementation, each platform runs the same application, and every account type can access the app through any of the supported interfaces and have the same functionality. The only exception here is the web-application, which cannot access the RFID hardware on the device running the browser.

1.3 Extended functionality

Apart from the basic implemented functionality detailed in Table 1, some extra features and improvements has been added to the software, these include:

- The same user interface for every device
- Email verification when registering
- Touch keyboard integrated within the app (mainly for the totem)
- Live search features to navigate the item/user lists more easily
- Password encryption
- Error checking and security features in backend

2 Process and Tools

This section will present the core software components required to run the system, the main tools and frameworks used for the development of these, and a high-level description of the overall system architecture.

2.1 Components

The main components of the system are the:

- Application
- Web server
- Database
- Totem

The application software is written using Flutter and Dart, which has provided a universal framework for designing and building the application for all the previously mentioned interfaces without having to do any platform-specific

tinkering. The services for sending http requests to the web server has also been written in Dart, which is very convenient as it made server integration with the application quite streamlined.

For this prototype, the web server and database are hosted using a program called XAMMP, which is designed for quick testing of clients and websites on a local host before publishing to a remote server. It comes pre-installed with all the software components needed, MariaDB relational database (MySQL), Apache HTTP server Filezilla FTP server, and phpMyAdmin (database admin tool for MariaDB), which is what has been used in this system. PHP has been used for the API and server-side scripting. Email verification is done using a service called *"emailJS"*, allowing emails to be automatically sent from client-side only through their API.

Python has been used for interfacing with the totem's RFID reader as well as communication with the web server. This is due to some issues with the RaspberryPi hardware and integration with the Flutter app (This is explained later).

2.2 Architecture

Below is a deployment diagram showing the overall architecture and internal/external communication of the prototype system.

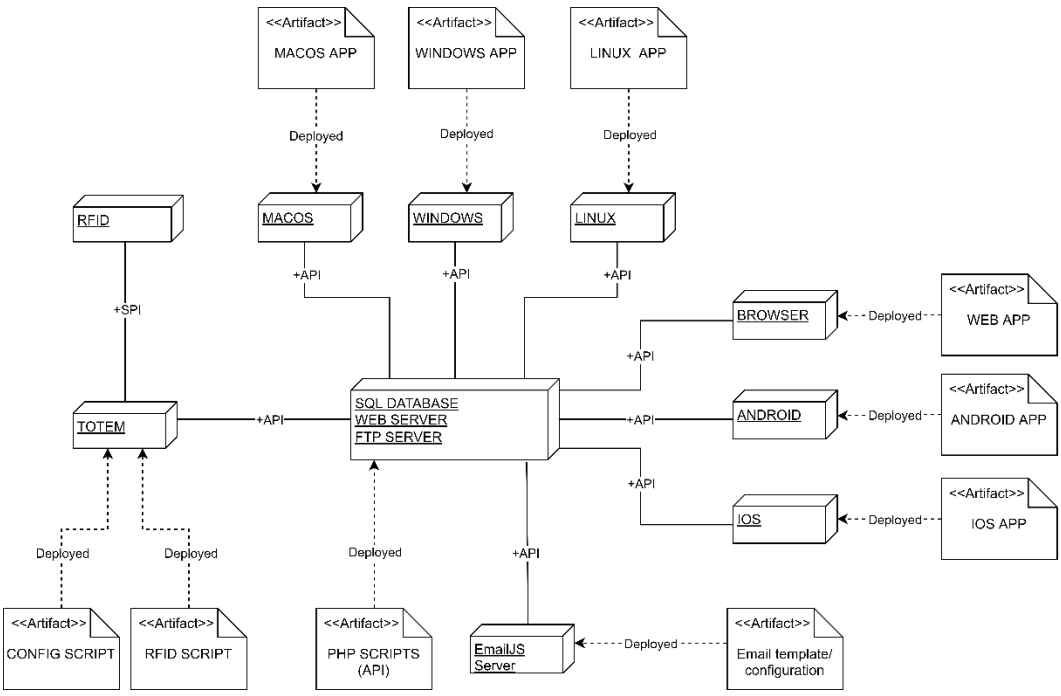

Figure 1 - Deployment diagram of the prototype system

As previously mentioned, there was some issues with building the flutter app for the totem hardware, so as can be seen in Figure 1 the totem is running the Flutter app in the browser, and the python script is reading the RFID module and transmitting information to the database. Since the browser cannot communicate with the module, it uses the database to detect when the RFID has received an input, and if that input came from the same totem. This is discussed in more detail in the following section.

3 Implementation

This section will present some details of the software implementation, which includes the database and server scripting, Flutter app, totem software, and the communication between these components.

3.1 Totem software

3.1.1 Explanation and problems

The goal at first was to make the Flutter application run locally on the raspberry pi hardware, as that would allow the application to communicate directly with the raspberry pi GPIOs using the Dart “rpi_gpio” library. This shouldn’t be a problem as Flutter now has support for compiling native apps for Linux, but there seems to be some issues with Linux compilation and ARM processors.

Another option is using a Flutter engine embedder called flutter-pi which makes it possible to run any Flutter app without booting into the desktop environment, which others have also used previously with rpi_gpio. However, this also didn’t work as there was some trouble with the installation.

Due to the above and time constraints, the app is running on the browser as it would on any other pc/smartphone. However, the challenge was to make sure the RFID module only communicates with the instance of the web-app running on the same totem. This was done by including some initial configuration steps for each totem, where a customer/admin must use a unique RFID chip to pair the totem and app.

3.1.2 Configuration script

This python script first checks for the file “config.txt”, if it doesn’t exist it creates it and waits for the customer/admin to scan an RFID chip to act as a unique ID, otherwise it means the totem has already been configured and gets the ID from the file. Finally, the totem ID is stored in the database through a http request. At this point the ID must also be manually entered in the web-app running on the corresponding totem by a customer/admin through a configuration page to use the

RFID module. To view the totem ID (RFID of the chip associated with it), they can either view the *config.txt* file or scan the RFID chip on a phone through the same configuration page (without linking it to the phone of course).

3.1.3 RFID script

This is script is constantly running on the raspberry, responsible for communicating with the RFID module and the server. It is continuously waiting for an input from the RFID reader, and if a tag is scanned the RFID and the previously generated totem ID is stored in the database through a http request. The RFID is then cleared from the database after a couple of seconds so it can be used again.

3.2 Database and server

3.2.1 Tables

The database is mainly responsible for storing information about every account and their items. It consists of the following three tables:

Accounts

Stores information about all the accounts registered in the system. Each account contains an *ID*, *Username* (email), *Role* (admin, customer, user), *Password* (encrypted), *RFID* (login), *Customer ID*, *Registered Customer ID* (which customers an account is registered to) and a *verified* column (stores verification code until account is verified).

Items

Stores information about all the items registered in the system. Each item consists of an *ID*, *Name*, *Status* (delivered/not), *RFID*, *Description*, *Location*, *Registered Customer ID* (which customer owns it).

Totems

Stores only the basic information about totems used for the totem communication explained previously, which is the *totem_id* and current *RFID*.

3.2.2 PHP scripts

These are the server-side programs acting as the bridge between the application and the database. They are responsible for handling the incoming http requests (API) and retrieving/storing data on the database as well as processing data for security purposes and efficiency.

A connection between the PHP scripts and the MySQL database is needed to retrieve/store information. This is using the PHP script conn.php, which creates the base for all the other PHP scripts.

The general PHP script consists of a variable ($conn) which is used throughout the script to perform MySQL queries. When a http request is sent, usually with some information, a query gathers an object using the SQL commands with the requested information. If the PHP script is supposed to return the result, it's echoed onto the webserver as a json object.

There are some cases where the PHP scripts have some important functionality worth noting:

- *addAccount*.php
  Provides error checking during account registration for already existing accounts/items with the provided username and RFID, and performs a cryptographic hash on the password (also done in *updateAccount.php when changing password*)
- *getAccount.php*
  Verifies incoming hashed password and retrieves the correct account details (depending on the permission of the account requesting the data).
- *addItem.php*
  Provides error checking for already existing accounts/items with the same RFID.
- *getItems.php*
  Gets a list of items depending on the permission of the account requesting the data, e.g., all for admin and only owned items for customer.
- *addTotem.php*
  Stores the incoming totem_id and the RFID in the incoming http requests from the totem scripts.
- *getTotemRFID.php*
  Compares the incoming totem_id from the Flutter application (empty if it's not a configured totem) and retrieves the corresponding RFID (if it exists).

3.3 Flutter application

3.3.1 Services

The web services for communicating with the server is written in Dart and can be found under "flutter_demo/lib/services" in the project source directory.

It is divided into three files:

- account_service.dart
- item_service.dart
- totem_service.dart

Both the account_service.dart and item_service.dart files have CRUD (Create, Remove, Update, Delete) operations. The account_serivce.dart can do additional operations due to required complexity in the frontend. The totem_service.dart only requires one operation, which is retrieving an RFID.

3.3.2 Pages

Login page

This is the startup page. To be able to log in, the user must enter their email and password and press the sign-in button. At this point the program makes a request to "getAccount.php" which checks the database for a match. There is also a button to transition to the "Register" page.

Register page

The register page is used to add or update accounts. The page will have a different layout depending on the type of account accessing it (no account, customer, admin), as the types have different privileges:

- *No account* (Accessed from login page)
  - Scan RFID card/token
  - Enter email/password
- *Customer* (Accessed after login)
  - Create user accounts
  - Update users
  - Edit RFID, email, password of accounts
- *Admin* (Accessed after login)
  - Do the same as customer
  - Create/update customer accounts
  - Edit account type, customer ID, registered customer ID of accounts

After a user has registered for the first time, a random verification code will automatically be generated and sent to their email, and they will be redirected to a verify page when they try to log in.

Verify page

A user with a non-verified email trying to log in will be automatically redirected to this page. Here they must enter the verification code sent previously during registration and can log in normally once they are verified.

User page

This is the home page of the user. The user can scan an item on their phone or a totem, and depending on its current status, the item will be updated to "borrowed" or "returned" in the database. When an item is borrowed, it's registered automatically with the user, and its location is changed to the name of the user. When returned, the status changes, but the location remains the same (name) until it's updated by the customer.

Customer/Admin page

The customer and admin are greeted with the same home page as the user, with the addition of a few buttons redirecting them to some other pages. These include:

- *User list*
  This page displays a list off all users, or if viewed by an admin, both customers and users. From here it's also possible to navigate the list by typing characters in a search field (works for both username and type), as well as add/modify accounts (transitions to the Register page).

- *Item list*
  Same functionality as for the User list, just for items. However, the search bar more advanced, where items will show up if any of its properties (type, description, location, status, RFID) matches the input.

- *Help user*
  Here a customer/admin can log in for any user in case there is no totem, they don't have a phone, forgot password, etc…

- *Play user (default window)*
  This is the same as the previously mentioned *User page*, so admins and customers can use their account for renting/returning items as a normal user would.

- *Configure totem*
  Accessed by clicking the gear symbol in the upper right corner. Here the admin/customer can enter their totems ID after running the configuration script to set up the connection between the RFID module with the instance of the web-app running on the totem. As mentioned previously, they can view the RFID of the associated chip (or any other chip) if they don't want to view the totems *config.txt* file. They can also toggle the on-screen keyboard on and off for the search fields if the customer doesn't need it.

Document history

- 30/06/2022: Version 1

- 15/07/2022: Version 1.1
  - New totem software
  - Database and backend changes
  - Security/efficiency improvements to services and API
  - Email verification
  - Recreated the entire user interface