A picture containing logo

Description automatically generated

Politecnico di Torino

Projects and Laboratory on Communication Systems

Inventory Management System  
Project Report

Authors:

Andreas Hølleland, Marcus Alexander Tjomsaas

Date:

23/06/2022

Contents

[1 Introduction 1](#_Toc106924853)

[1.1 Overview 1](#_Toc106924854)

[1.2 Requirements 1](#_Toc106924855)

[1.3 Extended functionality 3](#_Toc106924856)

[2 Process and Tools 3](#_Toc106924857)

[2.1 Components 3](#_Toc106924858)

[2.2 Architecture 4](#_Toc106924859)

# Introduction

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

## 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 ware-  
houses 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).

## 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 - 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 the 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.

## Extended functionality

# 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.

## 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 API 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 simple.

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 server-side scripting.

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

## Architecture

Below is a diagram showing the overall architecture and internal communication of the prototype system. Some of this would obviously be different in a real-world scenario due to the tools and setup for this demo.

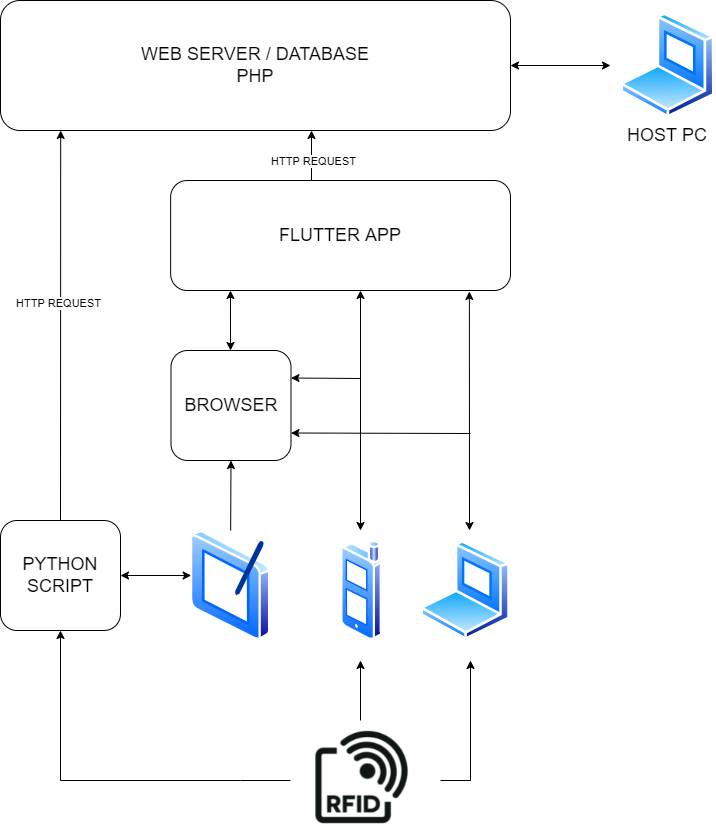


Figure - High level view of architecture

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 use 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.

# Implementation

This section describes the implementations of different sections in the Flutter app. Below is a chart showing the flow between pages for each type of user (user, customer and admin).

Graphical user interface, application, PowerPoint

Description automatically generated

## Login

The login page is used for accounts to access the contents of the app. To be able to login, the user must enter both email and password. When the user has pressed the “Sign-in” button, the email and password is checked in the database for a match. If there is a match, the user is logged in. In the login page, the user can go to the register page if they don’t already have an account.

## Register

The register page is used to add or update accounts. Depending on the account type (No account, customer, admin), the UI will look different as the types have different privileges. Someone without an account will be able to scan their RFID card, enter an email and their password. A customer has the same privileges but can also update already existing accounts. An admin has the privileges of a customer but can modify all the aspects of the account (RFID, email, password, type, customer ID, registered customer ID).

## User

The user page is for users used to borrow and return items. The user can scan an item, and depending on its status, it is either updated to “borrowed” or “returned”. When a user borrows an item, it is automatically registered at the customer that owns the item and the item location is updated to the name of the user.

## Customer

Some customer info

## Admin

Some admin info

## API

Some info about .php files

## Server

Some info about .php files

## Totem

Some info about .php files

Maybe flow of describing should be database -> php filses -> api -> pages.

## 3.x Quality of life implementation

Some quality of life implementations