IMS (Inventory Management System)   
Project Architecture Document

Version 0.1

# REVISION HISTORY

| **Date** | **Version** | **Description** | **Author** |
| --- | --- | --- | --- |
| 10/30/22 | 0.1 | Created initial outline of Project Architecture Document. | Larry Yang |
| 11/4/22 | 0.1 | Added Revision History, Introduction, and Diagrams for General Context, Use Case 3 and Use Case 4. | Priya Rivera |
| 11/5/22 | 0.1 | Updated diagrams and descriptions | Larry Yang |
| 11/6/22 | 0.1 | Edited and reformatted | John Atkins |
|  |  |  |  |
|  |  |  |  |

# TABLE OF CONTENTS

[1. REVISION HISTORY 1](#_Toc118626088)

[2. TABLE OF CONTENTS 1](#_Toc118626089)

[3. INTRODUCTION 2](#_Toc118626090)

[3.1. DEFINITIONS 2](#_Toc118626091)

[4. REASON FOR ISSUE 3](#_Toc118626092)

[5. RELATED DOCUMENTS 3](#_Toc118626093)

[6. OVERVIEW 3](#_Toc118626094)

[6.1. HARDWARE ARCHITECTURE 3](#_Toc118626095)

[6.2. SUPPORTED CONFIGURATIONS 4](#_Toc118626096)

[6.3. SOFTWARE ARCHITECTURE 4](#_Toc118626097)

[6.3.1. Server Module 5](#_Toc118626098)

[6.3.2. Web Server 5](#_Toc118626099)

[6.3.3. IMS Application 6](#_Toc118626100)

[6.3.4. Database Module 6](#_Toc118626101)

[6.4. Client Module 6](#_Toc118626102)

[6.4.1. Installation 6](#_Toc118626103)

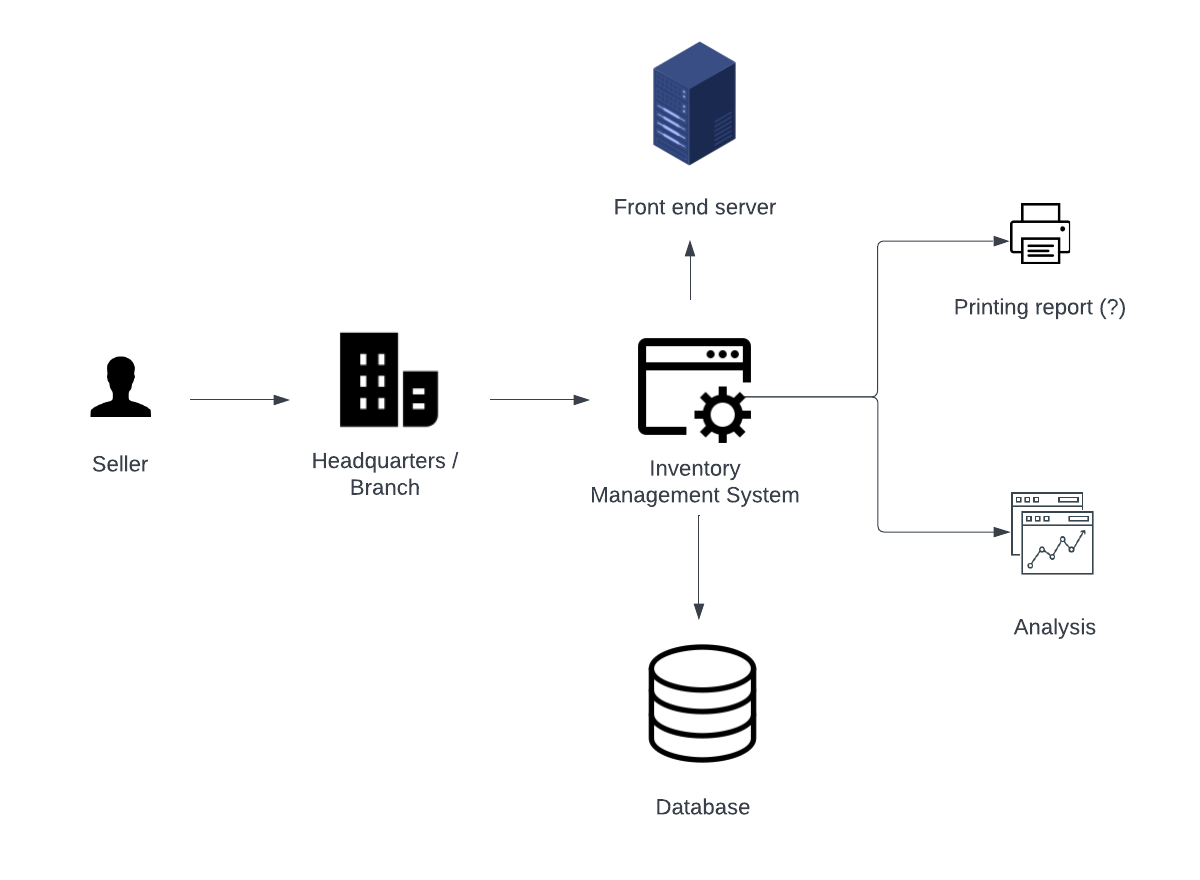


Figure . General IMS Context Diagram

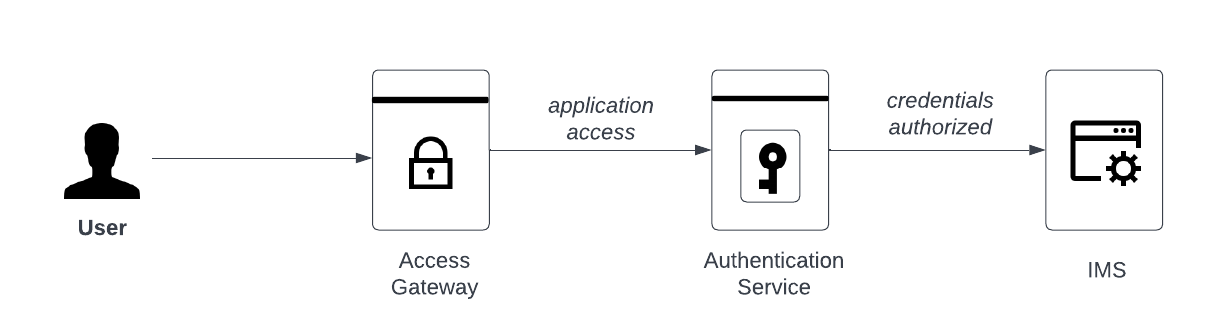


Figure . Login Diagram

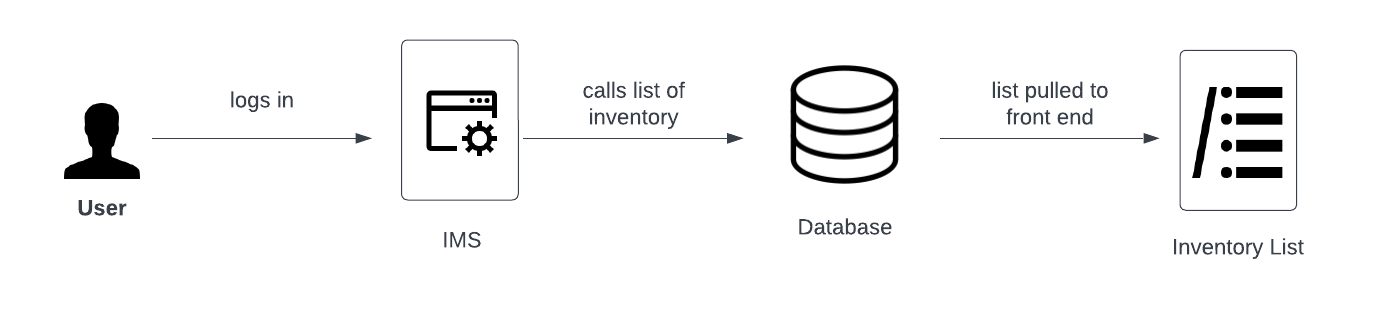


Figure . View List Diagram

# INTRODUCTION

This document defines the software architecture for the Inventory Management System. The architecture is based on the following references:

* [Web Application Architecture: The Latest Guide 2022](https://www.clickittech.com/devops/web-application-architecture/)
* “A Software Architecture for Inventory Management System” (Arsan, et al.).

The Inventory Management system is a business-to-business (B2B) solution that will cater to a seller who acts as a buyer for clients. The seller acquires inventory, negotiates sales with clients, generates purchase orders, takes payment, and ships. Currently the seller manages inventory in spreadsheets, which is cumbersome to use on mobile devices and makes it difficult to determine the current state of inventory. There will also be a read-only view for buyers to review order status. In a future release this view will be integrated with payment and shipping APIs.

The system will provide various interfaces to integrate with the seller’s and buyer’s workflow and processes. For small and medium businesses, this may include manual processes, forms, and spreadsheets.

This document provides details on the IMS objectives, supported configurations, and software architecture.

## DEFINITIONS

|  |  |
| --- | --- |
| API | Application Programming Interface |
| B2B | Business to Business |
| Docker | Docker container system |
| DOM | Document Object Model |
| Flask | Python web framework |
| FOSS | Free and Open-Source Software |
| GUI | Graphical User Interface |
| HTML | Hypertext Markup Language |
| HTTP | Hypertext Transfer Protocol |
| IMS | Inventory Management System |
| JavaScript/JS | JavaScript programming language |
| MVC | Model View Controller design pattern |
| ORM | Object Relational Mapping |
| OS | Operating System |
| React | React JavaScript library |
| RESTful | REpresentational State Transfer |
| PC | Personal Computer |
| Python | Python programming language |
| SaaS | Software as a Service |
| SPA | Single Page Application |
| SQL | Structure Query Language |
| TCP/IP | Transmission Control Protocol/Internet Protocol |

# REASON FOR ISSUE

|  |  |  |
| --- | --- | --- |
| Date | Version | Reason for Issue |
| 11/07/2022 | 1.0.0.0 | First formal release |

The latest released document may be obtained from the version control repository: <https://github.com/larryang/IS663-inventory-management-system/tree/main/documentation>

# RELATED DOCUMENTS

* [IMS Project Plan, Rev. 0.2](https://github.com/larryang/IS663-inventory-management-system/blob/main/documentation/IMS-Project-Plan.docx)
* [IMS Use-Case Specification, Rev. 0.3](https://github.com/larryang/IS663-inventory-management-system/blob/main/documentation/IMS-Use-Case-Specification.docx)
* [A Software Architecture for Inventory Management System](https://github.com/larryang/IS663-inventory-management-system/blob/main/documentation/IMS-Project-Plan.docx)

# OVERVIEW

This document describes the architecture of an inventory management system as an online SaaS. The system is a web-based client/server architecture and follows the MVC software architectural pattern.

## HARDWARE ARCHITECTURE

The server hardware is defined in the IMS Project Plan document. The system will utilize a third-party cloud service provider running Linux OS.

## SUPPORTED CONFIGURATIONS

The application is implemented as a web-based application solution. Figure 4 provides an illustration of the high-level system architecture. The production build (web server, application, SQL database) shall be deployed on the server as Docker images. The server shall support workload balancing and clustering.

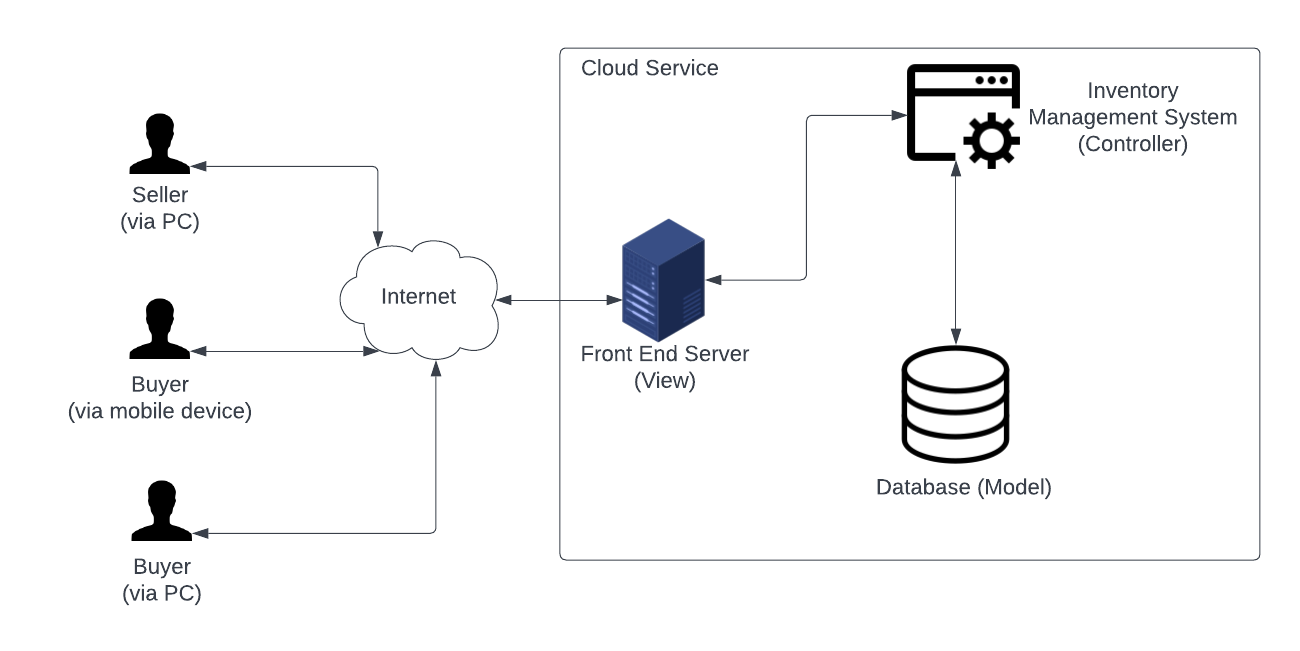


Figure . High-level System Overview

## SOFTWARE ARCHITECTURE

, consisting of three distinct modules: server, client, and database. All three are used during runtime operation of the software.

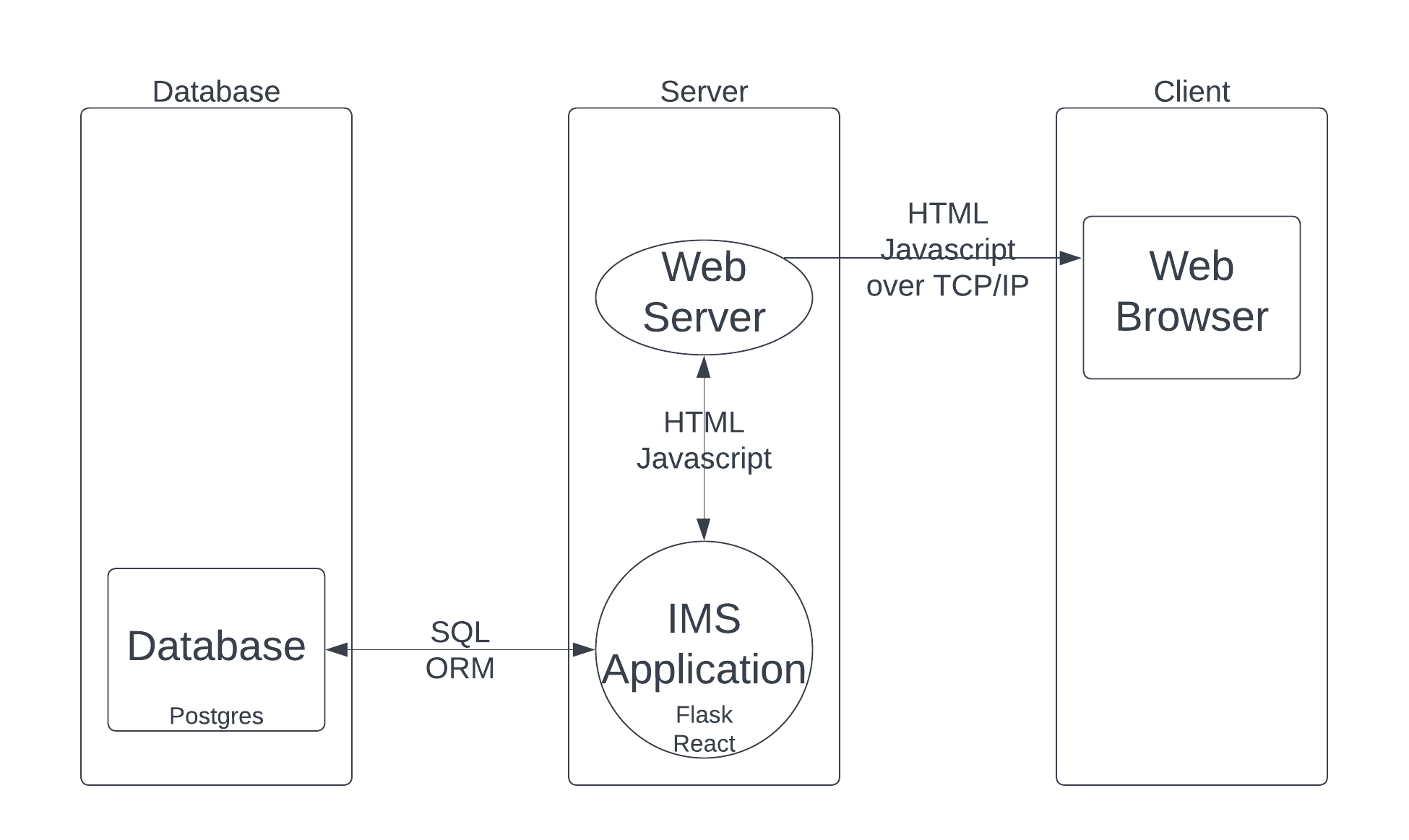


Figure . IMS Software Architecture

### Server Module

The server module is responsible for providing the client module with the information to display on the graphical user interface (GUI) and database access by means of RESTful API. It controls access to the inventory data, provides administrative functions, and generates the dynamic HTML pages. It consists of a web server and the IMS application and provides a RESTful interface for the client.

### Web Server

Software will be served via nginx web server, which uses an asynchronous event-driven approach, and is commonly deployed on Linux in Docker containers.

The software is Free and Open Source Sotware (FOSS), freely available at [nginx.org](https://nginx.org/) and in most Linux software repositories.

### IMS Application

The IMS application software shall be implemented as a Python Flask application. Flask will interact with SQL using ORM (such as SQLAlchemy.

The User Interface shall be implemented using React components, which will provide client-side updates by managing changes to individual components in a virtual DOM. The front end will retrieve and update data using API calls to the database via Flask.

The following programming languages, frameworks, and libraries will be used. These are all FOSS, used widely, and well supported:

* Python: [www.python.org/](https://www.python.org/)
* Flask: [flask.palletsprojects.com/en/2.2.x/](https://flask.palletsprojects.com/en/2.2.x/)
* SQLAlchemy: www.sqlalchemy.org/
* React: [reactjs.org/](https://reactjs.org/)

### Database Module

The database module shall use PostgreSQL, also referred to as Postgres. This is an FOSS relational database management system with support for SQL, freely available at <https://www.postgresql.org/> and in most Linux software repositories.

## Client Module

The client module is responsible for providing the display to the user. It is implemented as HTML documents with JavaScript UI elements. As the user interface by React components, this software will effectively function as a Single Page Application (SPA).

The client’s hardware shall be capable of running a modern web browser such as Chrome ([www.google.com/chrome/](https://www.google.com/chrome/index.html)), Safari ([www.apple.com/safari/](https://www.apple.com/safari/)), Microsoft Edge ([www.microsoft.com/en-us/edge](https://www.microsoft.com/en-us/edge)) and/or Firefox ([www.mozilla.org/en-US/](http://www.mozilla.org/en-US/)).

### Installation

Installation scripts will be implemented for the Linux machine platforms to deploy Docker images.

END OF DOCUMENT