Mr. Manager

Sale Management Website
Software Architecture Document

Version <1.0>

Revision History

Date	Version	Description	Author
08/07/2021	1.0		Nguyễn Hoàng Long Trần Hoàng Kim Lưu Danh Nhân Lê Hoàng Thịnh Phước

Table of Contents

Software Architecture Document

1. Introduction

Mr. Manager is a website-based system that helps offline stores manage their daily work. This system provides both employees and managers useful, efficient tools for their specific works such as check in, check out, order goods from suppliers, view goods in the warehouse, sell goods, view employee list, modify personal info, etc. With traditional selling way stores, those listed above processes may take a lot of time to manage, store and search. With the help of Mr. Manager, those processes are simplified which makes things faster and easier. Furthermore, this system is website-based so it is very convenient for computers with any configurations.

The introduction of the Software Architecture Document provides a comprehensive architectural view of the Mr. Manager system. The software architecture document is abstracted into many views and components which are explained in detail.

Purpose

This document delivers an inclusive architectural overview of the system by using different architectural views to describe different aspects of the system. It is designed to catch up and deliver the crucial architectural decisions which have been made on the system.

Scope

The Software Architecture Document applies to each static and dynamic of the system. Thus it makes the document complete and consistent.

Under the static behaviour of the system, the document discusses the class diagram and other static architecture designs. The use case realizations and system sequence diagrams are carefully used to show different aspects of the system.

Definitions, Acronyms, Abbreviations

OOP: Object oriented programming

Three-tier architecture:

- Client tier
- Business tier
- Database tier

GUI: Graphical user interface

OS: Operating system

PC: Personal computer

Overview

The document presents a detailed analysis of the architecture of the Mr. Manager system. The further section covers the architectural goals and constraints. The later sections cover the use-case model, logical view, deployment and implementation view respectively.

2. Architectural Goals and Constraints

2.1. Server side

Management system is hosted at the ‘Management’ server. Being a web based application, this client and server OS can be any PC operating system such as Windows, Linux, MacOS. The communication among client and server is REST. MySQL will also be hosted at the ‘Management’ server. The backend of the application uses Python as the main language with Django which is a framework to develop the application.

2.2 Client side

Clients can only access the server through the local network of the store. After successfully accessing and logging in, what features the clients may use depends on whether clients’ role is employee or manager. Clients are expected to use a modern web browser such as: Mozilla FireFox, Google Chrome, Safari, etc to work efficiently with the system. The front end of the application uses Javascript as the main language with ReactJS which is a framework to develop the application.

2.3 Security

Creating accounts for employees is done by managers. After the employee accounts are added to the system, they will be provided with a default password, which can be changed by the user. All the passwords are hashed in order to ensure high security.

2.4 Persistence

In order to maintain this, some measures have been taken such as hashing passwords, using transactions to all database commits, etc.

2.5 Reliability/ Availability

The system will be subjected to several testing operations (unit testing, annual testing, system testing) before being developed in order to make sure that system is reliable. The MySQL database server can respond to many clients at a given moment without lagging.

2.6 Performance

The system responds to any request under standard database and web server script timeouts, and also system performance can depend on availability.

2.7 Development tools

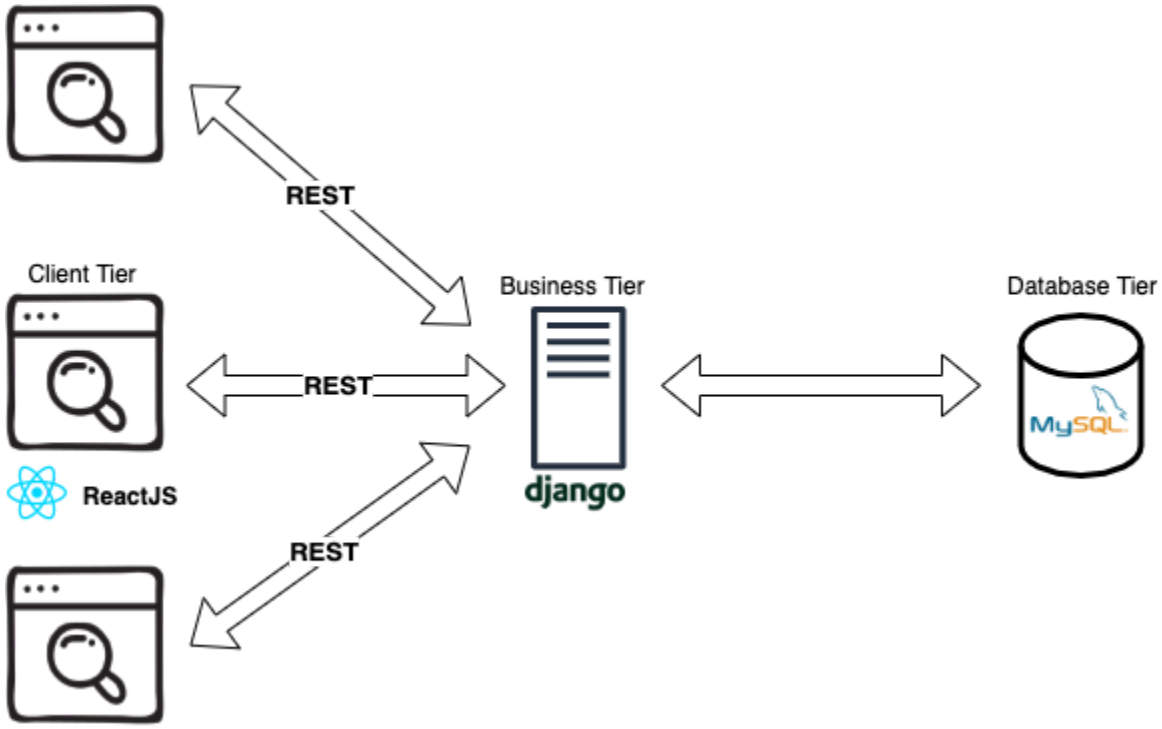
- Database: MySQL
- Programming: Python, Javascript
- Framework: Django, ReactJS
- Diagram: Draw.io, StarUML
- Schedule: Trello, Slack
- Meeting: Zoom

3. Use-Case Model

4. Logical View

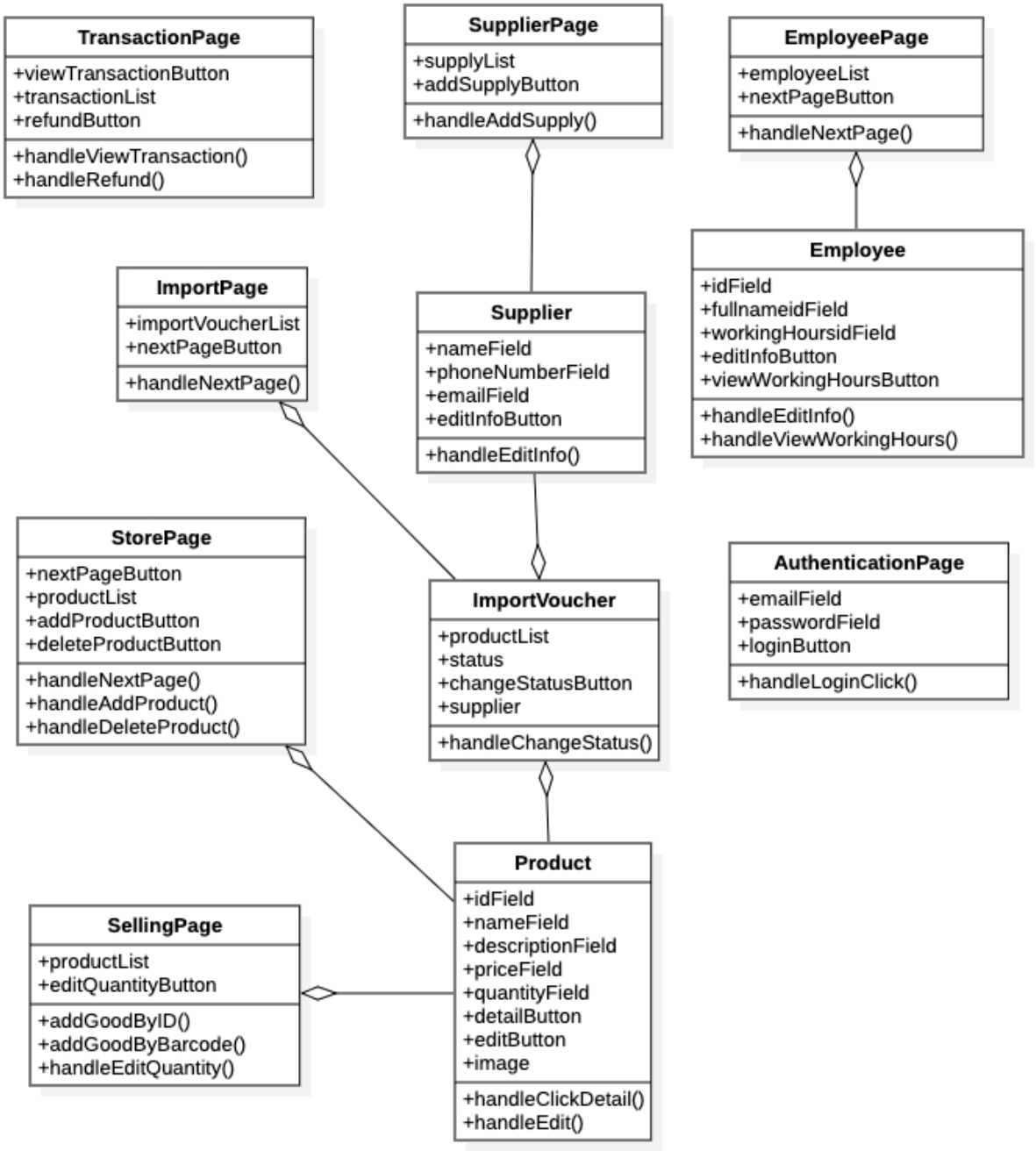
We use 3-tier architecture for our application. It includes Client tier, Business Tier and Database Tier:

- Client tier displays all the features supporting interactions between the user and server. It uses REST to communicate with server tier. Client tier is implemented by JavaScript with ReactJS which is a framework to develop this tier.
- Business tier is the management server which has the function to respond to requests from the client with data to be queried from Database Tier. Business tier is implemented by Python with Django to be a framework to develop this tier.
- Database tier saves all data about history transaction, information of employees and goods. This tier has the function to collect data and responses for queries from the server tier.



4.1 Component: Client tier

GUI Class Diagram



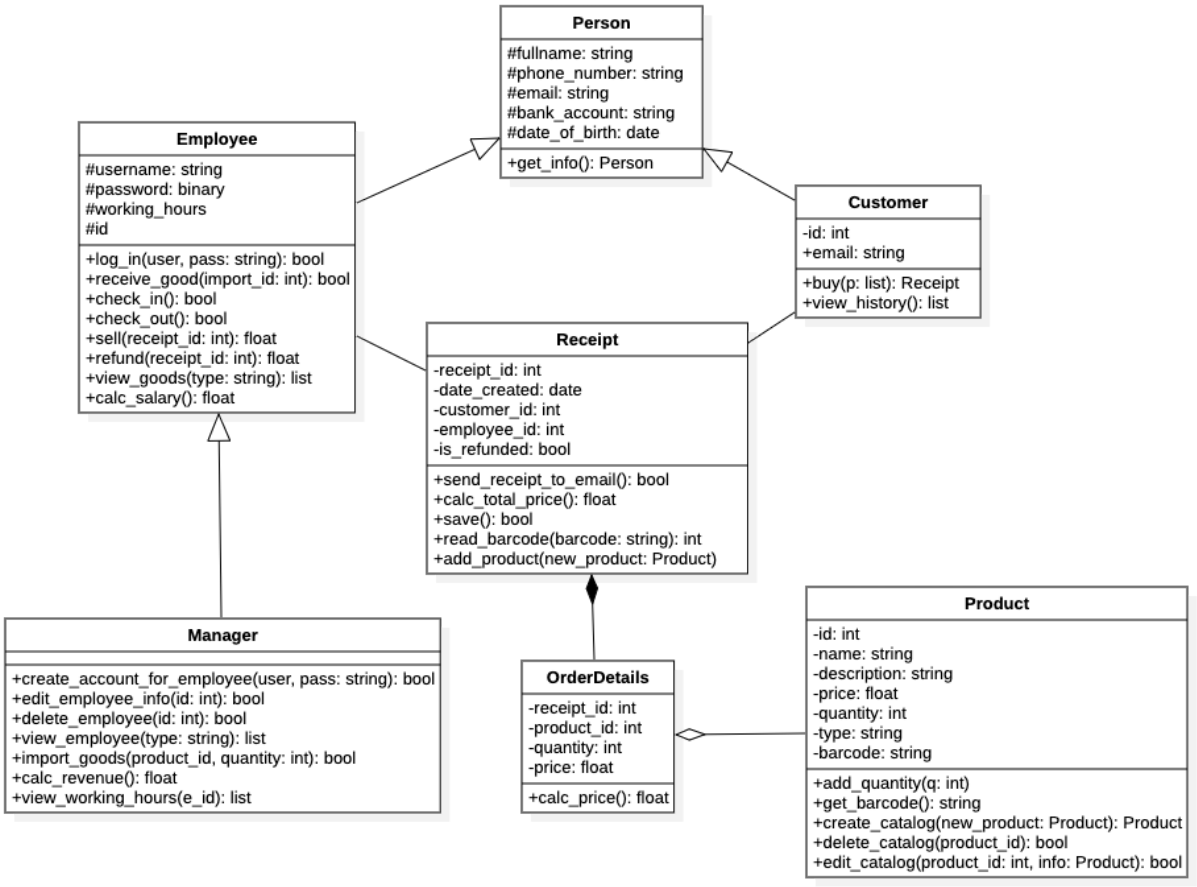
- Authentication page displays when the employee/manager logins. Employees must check in from this page to access the system in order to start working.
- Transaction page saves lots of receipts when the employees sold goods successfully. Employee uses this page to view the transactions in list form, especially for the customers' returned receipts, employees will

mark those transactions as refunded here.

- Supplier page contains many Supplier's information who supply their goods for the store. The manager will add supply if needed on this page as the products of those suppliers appear here.
- The ImportVoucher page contains information of products that the manager orders in the Supplier page. The employees can see the list of ordered products, the suppliers' info to check if the incoming deliveries are right or wrong. Then the employees can update the status of this goods import process.
- Employee page includes employee's information. The manager goes to this page to see the list of his/her employees, the manager can also edit the employees' info here if needed.
- Import page contains a list of imported vouchers which has the goods' import information. The employees can view the vouchers along with its information in list form here.
- Store page contains a list of goods for employees to easily manage their warehouse. On this page, the employees are able to add or delete goods.
- Selling page contains a list of available goods. The employees sell goods by increasing the quantities manually or scanning the barcode for the fast and automatic.

4.2 Component: Business tier

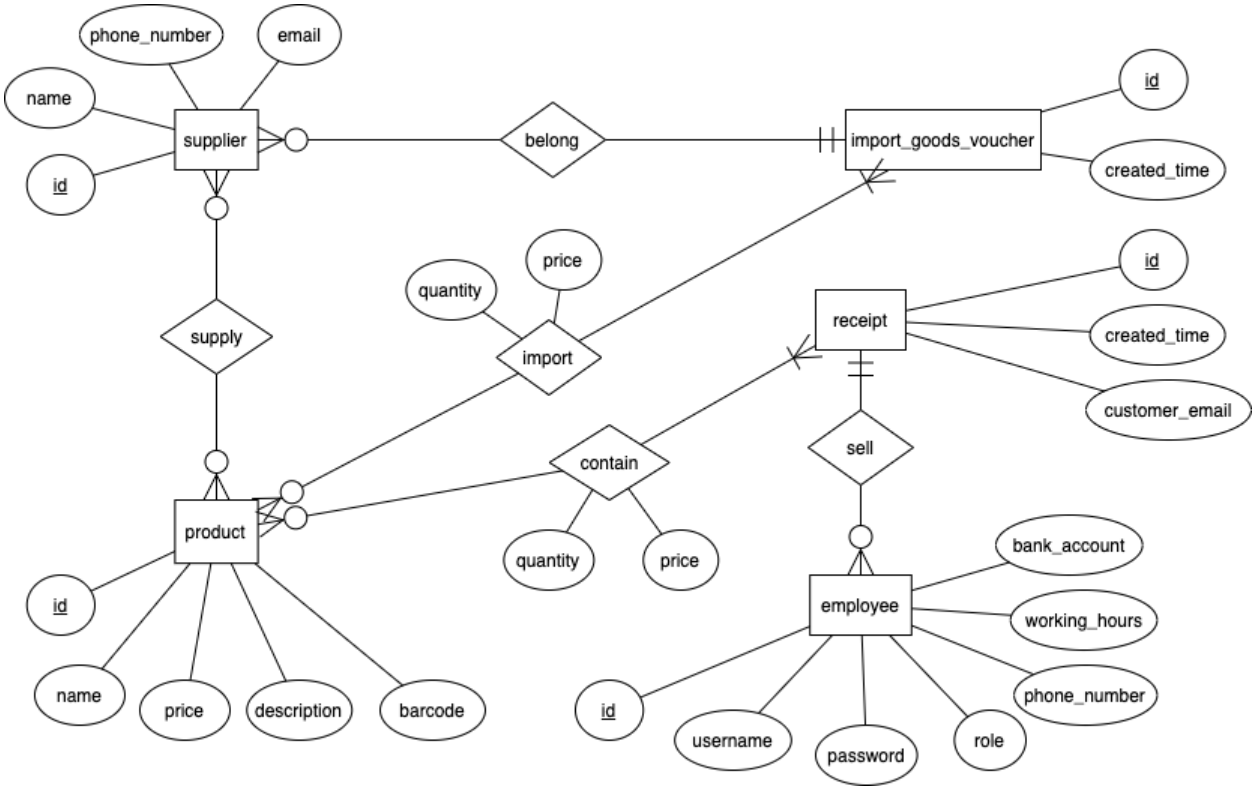
Server Class Diagram



- All employees and customers are derived from the super class called Person. Because each of them has common attributes and behaviors such as Fullname, Phone number, Email, Bank account, Date of birth.
- Receipt class serves the functions for Employee, Customer, OrderDetails. Receipt class has a ‘many to many’ relationship with OrderDetails. It saves necessary information to keep track of transaction history.
- OrderDetails class contains the list of products which are sold in the receipt. The OrderDetails class has a ‘many to many’ relationship with the Product class.
- Product class contains information on available products in the store for the manager to decide whether to import more goods or not.
- Customers are also able to return unwanted products and get their money refunded.

4.3 Component: Database tier

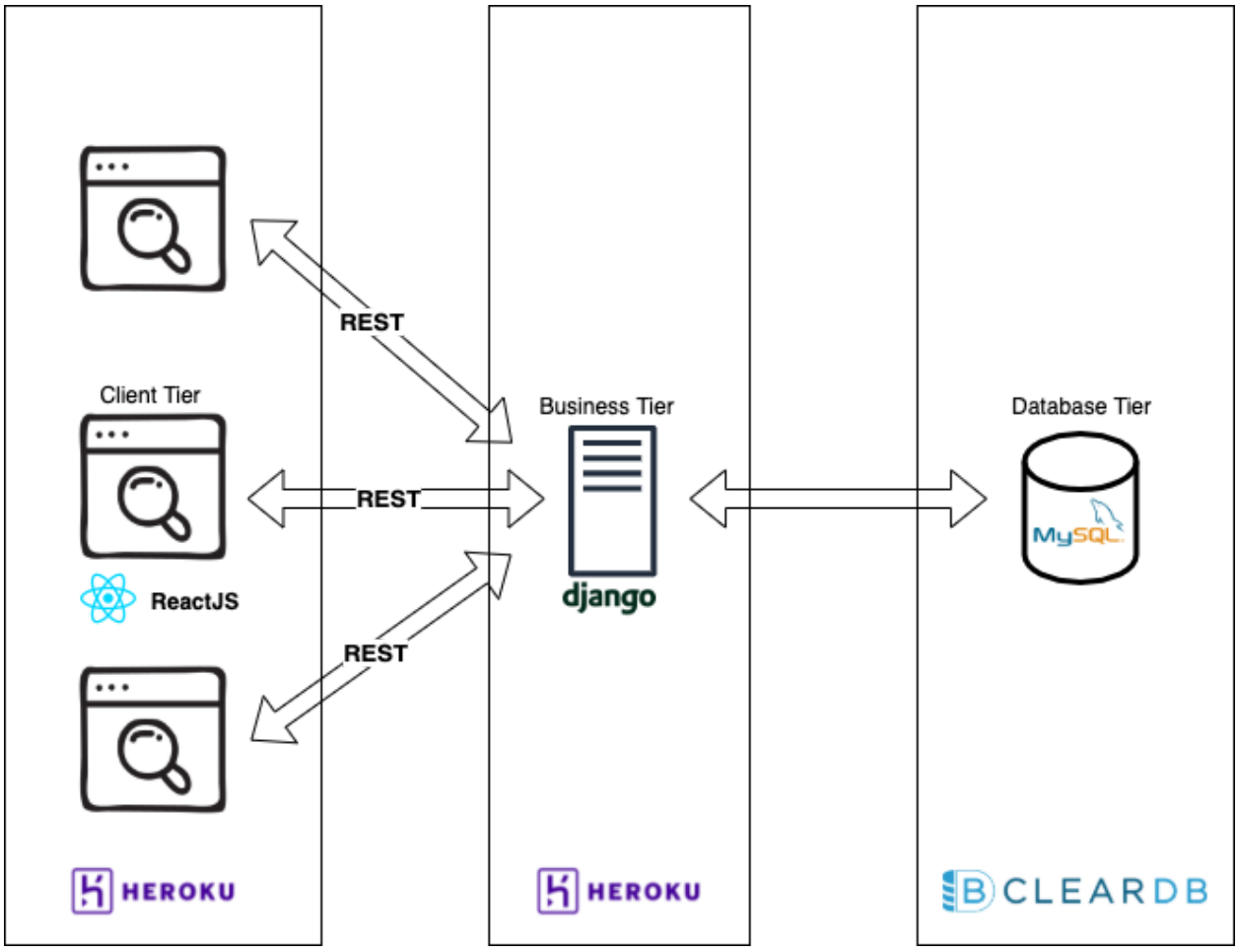
Entity Relationship Diagram



- Product entity is defined by an id which is supplied from a supplier whose entity is also defined by an id. These ids are unique. The result of this import process is an import_goods_voucher with a distinguished id, the date and time created also with the quantities, prices, and suppliers of all the imported products. Many products can be on the same import and from different suppliers, those products are written in an import_goods_voucher. This voucher also stores the id of the imported products' suppliers.
- Employee entity contains personal information of employees to help the manager keep track and manage their employee.
- Receipt entity includes id, customer_email, create_time that is date and time created when the employee sold products successfully. It has a 'contain' relationship that is many to many relationships with the product.
- Supplier entity is used to contain necessary information and ways of contacting suppliers who supply goods to the stores. It has a 'supply' relationship that is many to many relationships with the product.

5. Deployment

Being a web application this system is hosted in a remote server (initially Heroku hosting space). The database will be hosted in the ClearDB hosting services (an add-on of Heroku). All the calculation is processed in the backend except the graphics rendering for security purposes. In case of graphics rendering, all the data sent by the server, will be rendered in the frontend to reduce cpu workload of the server. The front-end application will be also hosted in Heroku.



6. Implementation View

6.1. Overview

Mr. Manager is a web application that follows the three-tier architecture pattern. Main reason to use this pattern is to separate functions into layers thus improving maintainability.

The system is divided into 3 different layers:

- Client tier contains the graphical user interfaces (webpages). The actions of these web pages are handled.
- Business tier contains the main entity classes such as Employee, Receipt, Product, etc.
- Database tier is manipulated by using the object relational mapping in Django framework. Each entity has a corresponding class in the database that provides data access operations to that entity

6.2. Size and Performance

Being a web application, Mr. Manager will be hosted in the free web hosting space heroku.com. Users will not have to set up or install any components to use the application. The size of the application is in the range 50MB to 150MB with all dependencies.
MySQL will be hosted in the heroku database plugins. It provides 5MB for the nonfinancial account (it is enough for the demo). If the customer wants to scale up, we can easily upgrade by paying more money.
The client needs a computer that has a web browser to access the system. All functionalities will be processed in the backend except the graphics rendering.

6.3. Folder structures

```
mr_manager
├── backend
│   ├── main.py
│   ├── controller
│   └── utils
├── frontend
│   ├── index.js
│   ├── pages
│   │   ├── Sell.js
│   │   ├── Dashboard.js
│   │   └── Transaction.js
│   ├── components
│   │   ├── Form.js
│   │   ├── Footer.js
│   │   └── Header.js
│   └── utils
└── README.md
```