**COMP 3059 – Capstone Project I**

**Software Requirements Analysis and Design Assignment**

This assignment is an overview to gather the software needs with requirements analysis and help to proceed with the design.

The requirements analysis helps to break down functional and non-functional requirements to a basic design view to provide a clear system development process framework. It involves various entities, including business, stakeholders and technology requirements.

The design is the activity following requirements specification and before programming. Software design usually involves problem solving and planning a software solution.

To work on this assignment you could use the references and a sample template given below. The sample template can be customised to suit the nature of your project.

Reference Readings/Example:

<http://www.uacg.bg/filebank/acadstaff/userfiles/publ_bg_397_SDP_activities_and_steps.pdf>

[www.cse.msu.edu/~chengb/RE-491/Papers/SRSExample-webapp.doc](http://www.cse.msu.edu/~chengb/RE-491/Papers/SRSExample-webapp.doc)

<https://nces.ed.gov/pubs2005/tech_suite/part_2.asp>

Reference template:

[www.tricity.wsu.edu/~mckinnon/cpts322/cpts322-srs-v1.do](http://www.tricity.wsu.edu/~mckinnon/cpts322/cpts322-srs-v1.doc)c

**System Design Document**

**Project Name: ResTaskest**

**Team 30**

Document Version: 1.0

Date: October 31th, 2019

**Table Of Contents**

**1.0 Introduction**

1.1 Purpose

1.2 Scope

1.3 Definitions, Acronyms, and Abbreviations

**2.0 System Overview**

2.1 Project Perspective

2.2 General Constraints

2.3 User Characteristics

2.4 Assumptions and Dependencies

**3.0 Function Requirements**

3.1 Product Functions

3.2 Use Cases

*3.2.1 Management Use Cases*

*3.2.1 Employee Use Cases*

3.3 Data Modeling and Analysis

*3.3.1 Normalized Data Model Diagram*

*3.3.2 Activity Diagrams*

*3.3.3 Sequence Diagrams*

*3.3.4 UML Class Diagram*

3.4 Process Modeling

*3.4.1 Data Flow Diagram*

**4.0 Non-Functional Requirements**

**5.0 Logical Database Requirements**

**6.0 References**

**7.0 Approval**

# 1.0 Introduction

## Purpose

The purpose of this document is to provide a clear structure of ResTaskest, and to identify the essential requirements for the success of the project. Specifically, this document describes all the features, functions, and interfaces of the system by pointing out the what the system provides and how the users can benefit from that. Also, this document will cover knowledge about the underlying layers of the system.

This document provides both stakeholders and developers with an overall context of creating the Restaurant Management Application. Therefore, by going through this document, stakeholders will possibly have a clear picture of components created in the application. Additionally, this document can assist our team in drawing a developing path that team members can follow and achieve our objectives

## Scope

The ResTaskest is a creation of a contemporary restaurant management application. Our application target to local medium-sized and large-sized restaurants over Ontario, which need a comprehensive system to assist restaurant manager to handle restaurant’s resources such as restaurant’s staff, restaurant’s inventory, and sale’s summary. Since the ResTaskest does not aim to be a POS system, payment handling, and customer service are not included in the deliverables.

In detail, the ResTaskest application offers a variety of functions for restaurant’s manager and kitchen’s staffs (Back-of-house and Front-of-house staff) to effectively and efficiently govern the restaurant resources as well as communicate to others by simplifying the daily processes of running a restaurant and cutting-off tedious manual steps. One of the main functions that ResTaskest offers is restaurant’s inventory management ,which will assist BOH staff in checking restaurant’s resources and sends a report to the restaurant’s manager at the end of the day. Another system’s function is scheduling, which maintains the connection between the manager and staffs so that they can together come up with a more appropriate schedule. Regarding how the scheduling function works, the application will help staffs to send their availability to their manager via an activity called Availability Report. Once the manager receives the availability report from staffs, he/she can effectively schedule staffs with a respect to their expectations. Later on when the manager submit their completed schedule to the system, staffs can immediately access to that schedule by using View Schedule function. The system ,also, provides order handling function, which would help FOH and BOH to communicate accurately and quickly to decrease the chance of miscommunication between two departments. One more important thing about the system is that the system will record all the data using MongoDB which is a cross-platform NoSQL database program that will be hosted on the Google Cloud Platform. Hence, users can access to their data from anywhere by using any devices. As far as the platform support is concerned, ResTaskest will be web-based application that will be transferred to native for iOS and Android users by using NativeScript.

**1.3** **Definitions, Acronyms, and Abbreviations**

|  |  |
| --- | --- |
| **Terms** | **Definitions** |
| **User** | Restaurant manager and/or staffs |
| **Stakeholder** | Person who interest in the project but are not developer |
| **FOH** | Front-of-house employees who can interact with system by submitting their daily work hours, order status, availability, and end-of-day sale report. |
| **BOH** | Back-of-house employees who can interact with system by submitting their daily work hours, order status, availability, and inventory info. In return, they receive their paystubs and schedule which are generated by restaurant manager |
| **Staff** | Back-of-house and front-of-house staff of the restaurant |
| **Restaurant manager** | Restaurant manager is the one who generates documents such as staff schedule, staff paystub. In return, everyday, he/she will receive the sales report, inventory levels, and staff work hours |
| **Sale summary** | The end-of-day sale report which is summarized by FOH employees and submitted to restaurant manager |
| **Chat box** | A live chat room where users could communicate in real - time |
| **Dashboard** | A graphical user interface which provides at-a-glance views of key information |

**System Overview**

##### 2.1 Project Perspective

The ResTaskest application is a contemporary web application,

which will provide local restaurants in Ontario and around Canada a robustly assistance in managing diverse restaurant’s resources ( such as inventory, staff, … etc). Thanks to that assistance, the restaurant’s manager can put more attention to improve the business development process. ResTaskest is created to be a supportive tool that will be used with the POS system to help restaurant’s managers comprehensively manage their businesses.

## 2.2 **General Constraints**

*List of constraints:*

* **Limitations in tool selection**: Since we do not have any stakeholder or industry partner yet, budget is quite a big issue when it comes to picking up which technology to employ. For example, we plan to host our server on a cloud computing plat form, but none of them are free. Therefore, using open-source products will always be our top priority.
* **Lack of experience:** This is the first time that our team collaborate with each other to solve a real world problem. Therefore, experience is the most crucial thing that we lack. However, it is a good opportunity for us to combine all the knowledge that we have learned from the very first semester to build something up and sharpen our skills.
* **Platform support still limits:** The ResTaskest project is a web-based system, and we plan to use NativeScript to make it native for users. However, the experience when using native version of ResTaskest may not as good as when we use the web version.
* **Payment Handling is not included:** The scope of this system is to focus on the management part of the restaurant. Hence, payment and customer handling are not included in the system
* **Difficulty to write documentation for the system:** Since English is not our first language, we have to try a lot to write a clear and comprehensive documentation for users

## 2.3 **User Characteristics**

|  |  |
| --- | --- |
| Users | Authority |
| **Restaurant manager** | Entirely directly access to the system, the manager has authority to make changes in resources of the system |
| **Back-of-house staff** | Limited access to some areas of the system that relates to their role in the restaurant. (such as check inventory, view schedule… etc) |
| **Front-of-house staff** | Limited access to some areas of the system that relate to their role in the restaurant. (such as sale summary, view schedule… etc) |

## 2.4 Assumptions and Dependencies

* The following assumptions were made during the initiation of the project by our team of developer:
* Our team members are willing to learn new technology which is NativeScript, Docker, and Kubernetes
* Those mentioned industrial specialists are ready to help whenever they can
* The developing process cost will not go over the planned budget so that we can keep our product’s price reasonable
* There is enough time allocated to team members to learn new technology
* Number of project stakeholders will change since we are still approaching to more industrial experts.

## 3.0 **Functional Requirements**

***3.1 Product Functions***

### 3.1.1 **Profit Visualization**

* I**ntroduction**: The profit visualize function is to help restaurant mangers see how their business grows over a period of time
* **Inputs**: This function takes the operation costs such as the total wages of employees, total spending on materials, and the restaurant earning which is based on the sale report
* **Processing:** System will calculate the profit by subtracting the total earning by the operation costs
* **Outputs:** A chart of profit in a specific period of time

### 3.1.2 **Inventory Management**

* I**ntroduction**: Inventory Management will help BOH employees to easier compile the inventory at the end of the day, and help manager to better manage the restaurant’s inventory
* **Inputs**: List of needed inventory for the next business days
* **Processing:** System will take the list of items that BOH employees submit and send it to the manager
* **Outputs:** Managers will see a list of items which include item name, quantity so that they can decide which items to be ordered

### 3.1.3 **Sale Summary**

* I**ntroduction**: A sale report which includes the total earning summarized by the FOH employees at the end of the day
* **Inputs**: List of orders and prices
* **Processing:** System will summarize the before-tax and net earning of the business based on the number of orders that are paid using cards. FOH employees, after that, have to calculate the cash earning in that day, add it to report to receive the final net earning. Finally, FOH employees will submit this report to the manager
* **Outputs:** A detailed list of sales in a business day for manager

**3.1.4 Staff Work Hours Calculation**

* I**ntroduction**: The total work hours of staffs is automatically calculated by the system based on the clock in and clock out time that staffs submitted
* **Inputs**: staffs’ in and out time
* **Processing:** System will summarize the total work hours of staff based on the number of their shifts and the duration of those shifts
* **Outputs:** The total work hours of staffs within a specific time

**3.1.5 Staff Paystub Generator**

* I**ntroduction**: This function will use the staff work hours to generate paystub that includes gross earning, the amount of tax that employees have to pay as well as the vacation earning in a pay-period
* **Inputs**: staff work hours, vacation earning percentage
* **Processing:** System will generate a document in pdf format based on the information submitted
* **Outputs:**  A completed, identified paystub for staffs

**3.1.6 Availability Submission**

* I**ntroduction**: This function will allow employees to inform manager their availability for the upcoming weeks
* **Inputs**: employees’ availability
* **Processing:** System will generate a weekly calendar with available employees on each day
* **Outputs:**  A weekly calendar with available employees on each day

**3.1.7 Scheduling**

* I**ntroduction**: Scheduling function will help manager to create a weekly schedule for emloyees
* **Inputs**: employees’ availability
* **Processing:** Manager is managed to create a weekly schedule based on the employees’s availability
* **Outputs:**  A comprehensive weekly schedule for employees

**3.1.8 Order Viewer**

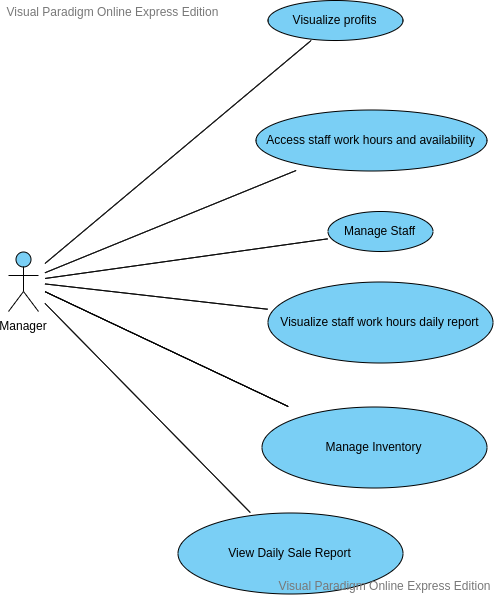
* I**ntroduction**: Order viewer is a feature that can help to avoid the miscommunication between FOH and BOH, and thus it decreases the chance of making mistakes when preparing orders
* **Inputs**: food and drink submitted from FOH employees
* **Processing:** stuffs submitted by FOH employees will then be placed in an organized form including order status, allergy alert, time placed, extra stuffs to add.
* **Outputs:**  A form with a modifiable order status field for BOH employees

**3.1.9 Schedule Viewer**

* I**ntroduction**: This function will use the schedule prepared by the manager to find out the shifts for a specific employee and display them to that employee
* **Inputs**: schedule prepared by the manager, employee id
* **Processing:** A function will take scheduled prepared by the manager and the employee id to through the big schedule and find out shifts for that specific employee. Finally, the function will return the list of shifts in that week in a weekly calendar format for the employee
* **Outputs:**  A weekly calendar with assigned shifts for employee

## 3.2 Use Cases

### 3.2.1 Management Use Cases

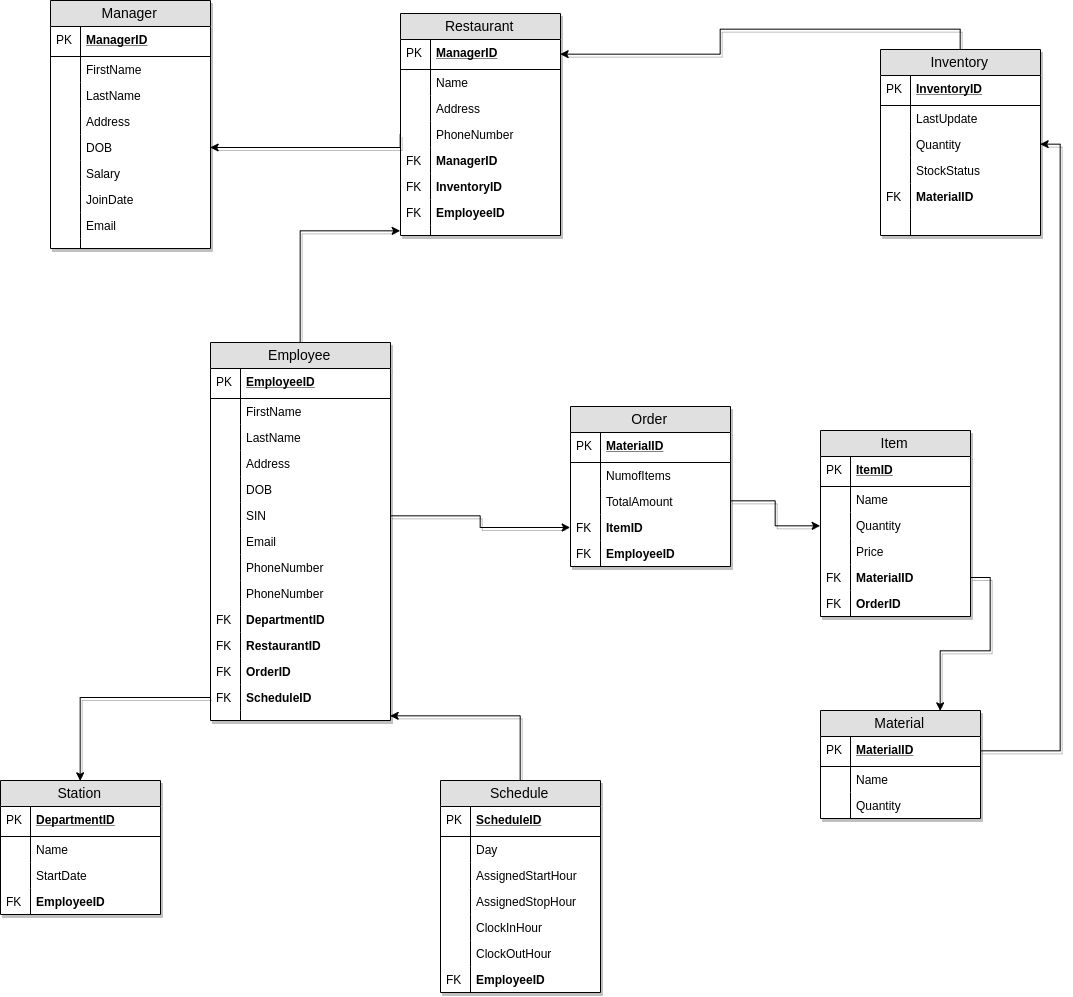


### 3.2.2 **Employees** Use Cases

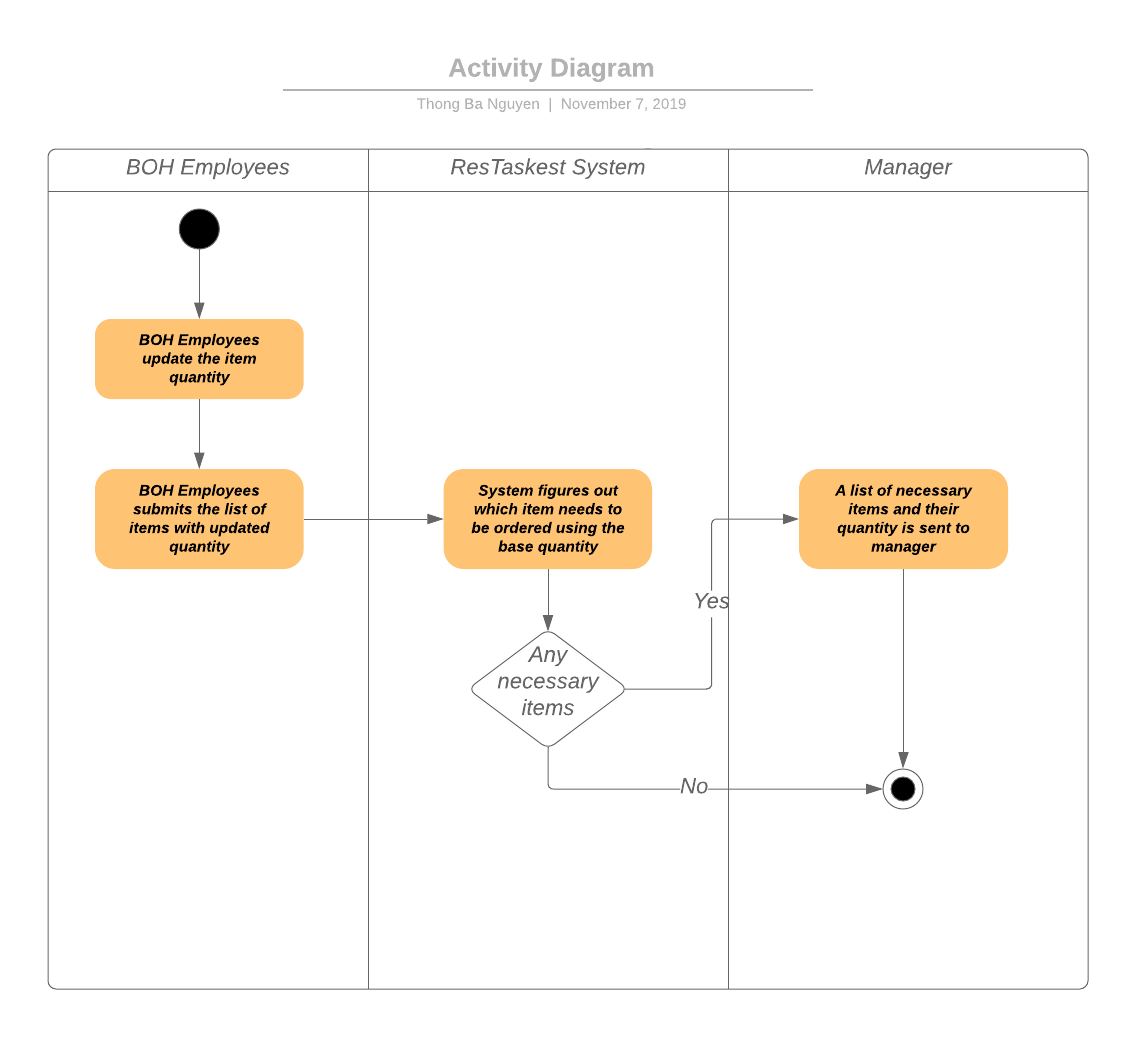


**3.3 Data Modelling and Analysis**

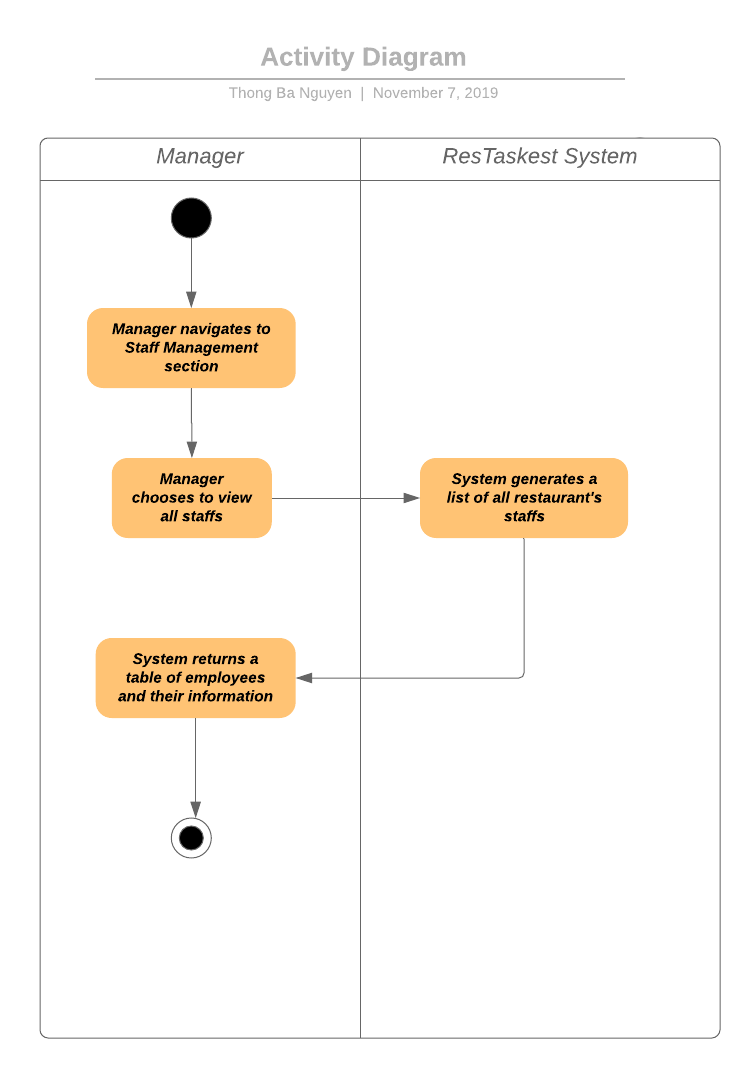
***3.3.1 Normalized Data Model Diagram***



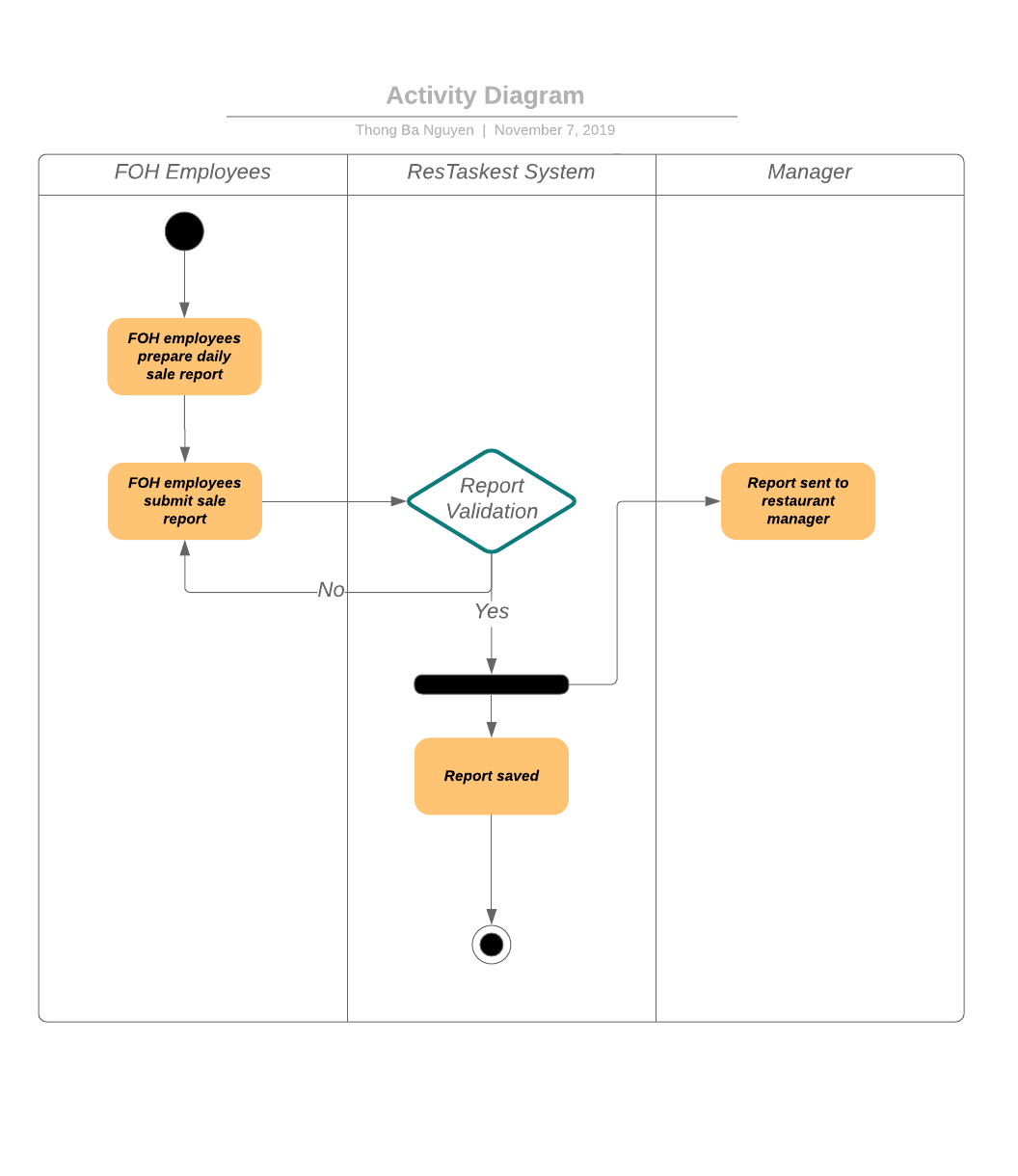
**3.3.2 Activity Diagrams**

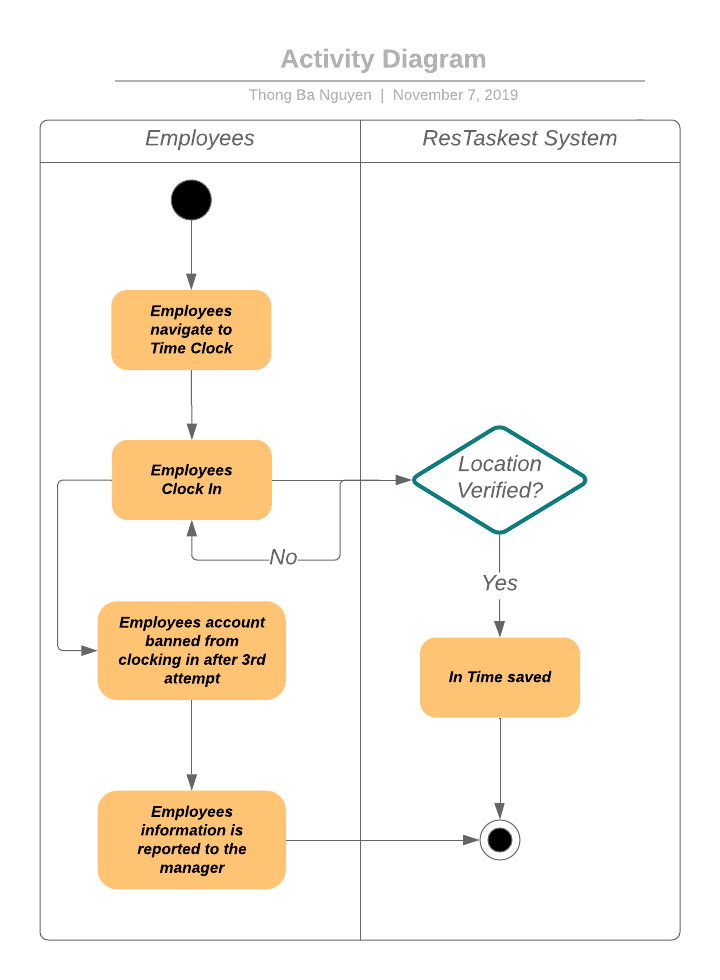


*Description: Activity Diagram when back-of-house employee compiles inventory*



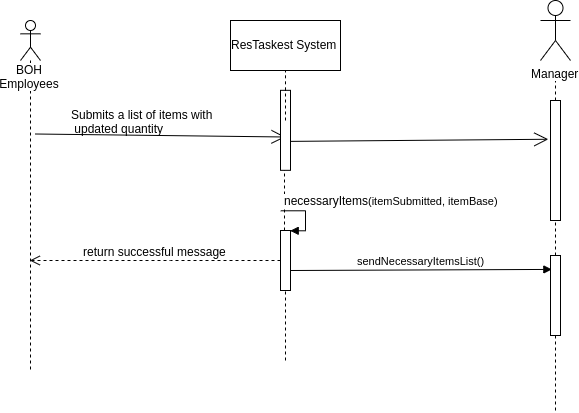
*Description: Activity Diagram when manager retrieves a list of all restaurant staffs*

*Description: Activity Diagram when front-of-house employees submit their end-of-day sale report to manager*

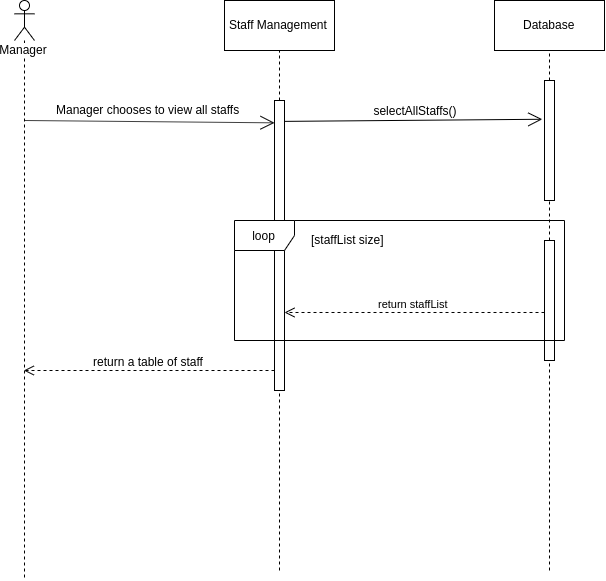


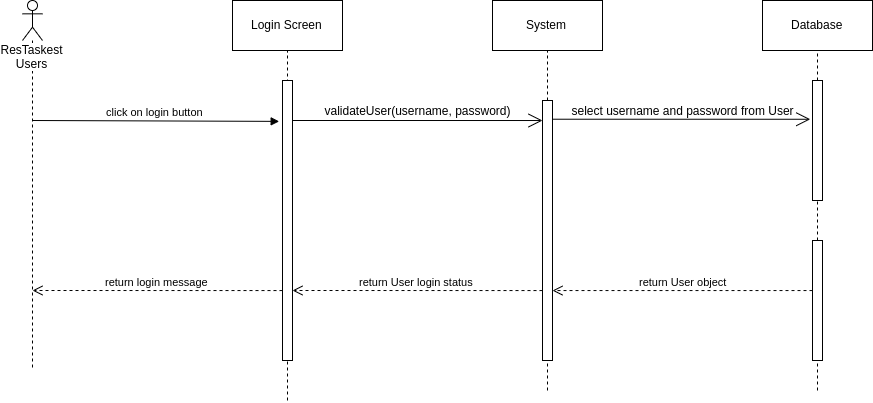
*Description: Activity Diagram when employees clock in their shift*

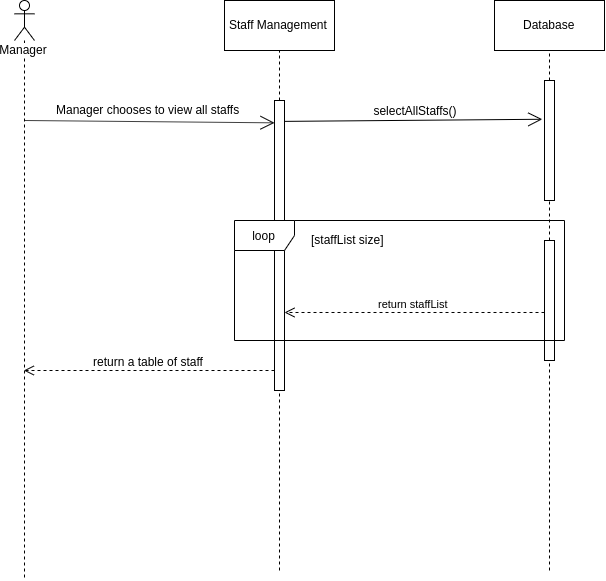
**3.3.3 Sequence Diagrams**



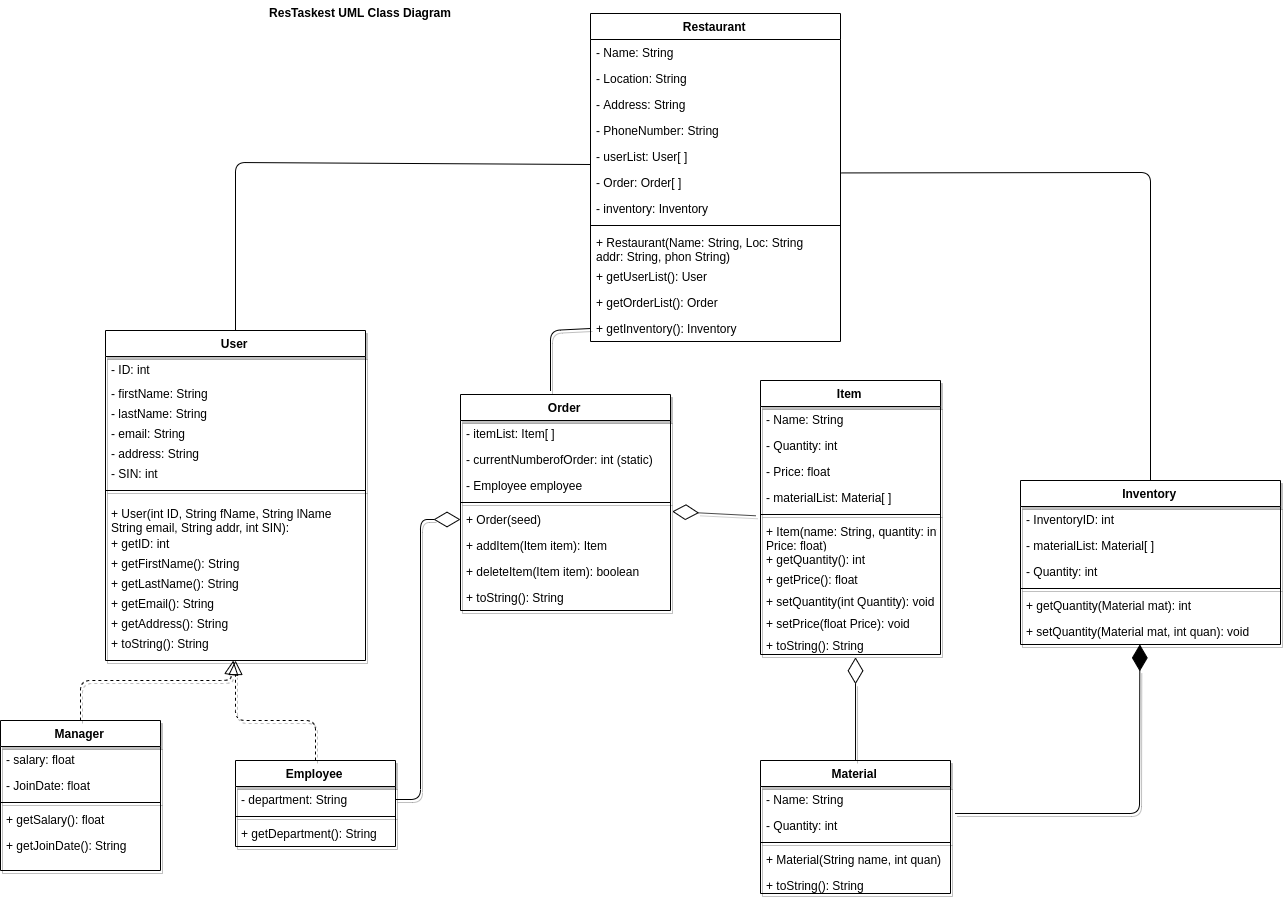
*Description: Sequence Diagram when back-of-house employees compile inventory*

*Description: Sequence Diagram when manager retrieves the list of all restaurant staffs*



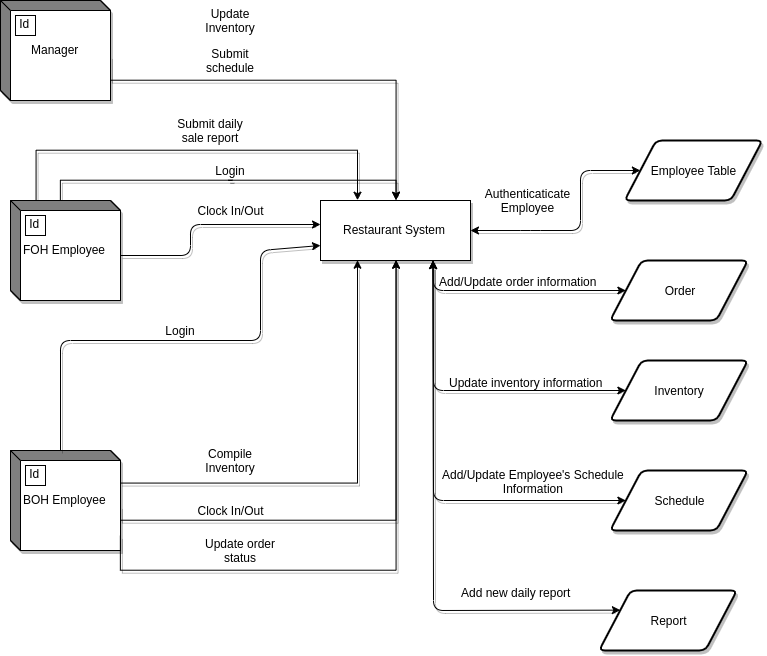
*Description: Sequence Diagram when ResTaskest users attempt to login to the system*

**3.3.4 UML Class Diagram**



**3.4 Process Modelling**

**3.4.1 Data Flow Diagram**

*Description Level 1 DFD for the system*

## 4.0 Non-Functional Requirements

* **Usability**: the system should be usable by every body. The user should be able to operate on the functionality easily. A guidance or training must be provided to users so that they can adapt to the system as well as possible
* **Extensibility:** the system should allow users to conveniently upgrade to meet their business requirements as their business grow. The system design must take advantages of using microservices to scale the application horrizontally, and thus decrease the resources needed.
* **Reliability:** the system should be upgradable with no or at least a minimal downtime which may not exceed 2 minutes to not cause any negative influence on the business
* **Performance:** the elapsed time when users send any request to the system must be minimal and not over 1 second
* **Security:** the system must safeguard user’s information by appropriately selecting who is in the system can have access to what resources
* **Portability:** the system should be responsive so that users can access using different platforms
* **Maintainability:** the design of the system should be well-structured, and the system’s codebase must be readable and easy to maintain
* **Scalability:** System must be scalable to handle multiple users using the system simultaneously

## 5.0 Logical Database Requirements

* The system should store all data about order, employees, and material with a well-defined Schema for NoSQL database.
* Only administrator can have the access to the system database
* User’s personal information such as email, password must be highly protected, and encrypted
* In the future, if the system extends to have an employee auto-deposit function, the last four digit numbers of user’s account have to be hidden
* All the unnecessary data have to frequently be cleaned to save database storage

## 6.0 **References**

1. Software Development Process – actitvities and steps

Link: <https://www.uacg.bg/filebank/acadstaff/userfiles/publ_bg_397_SDP_activities_and_steps.pdf>

2. Software Requirement Specification (SRS)[***https://personal.utdallas.edu/~chung/RE/Presentations06F/Team\_2.pdf***](https://personal.utdallas.edu/~chung/RE/Presentations06F/Team_2.pdf)

3. SRS Example – Web App www.cse.msu.edu/~chengb/RE-491/Papers/SRSExample-webapp.doc

**7.0 Approval**

The signatures below indicate their approval of the contents of this document.

|  |  |  |  |
| --- | --- | --- | --- |
| Project Role | Name | Signature | Date |
| Project Manager | Thong Nguyen | Thong | Nov 10, 2019 |
| Project Developer | Quang Pham | Quang | Nov 10, 2019 |
| Project Developer | Tu Nguyen | Tu | Nov 10, 2019 |
| Project Developer | Thanh Quan | Thanh | Nov 10, 2019 |