2022

Keith Bittner

Eastern Oregon University

6/8/2022



Software Requirements Document

Capstone / Employee & Account Management System

Table of Contents

[1 Executive Overview 2](#_Toc105586159)

[1.1 Project Overview 2](#_Toc105586160)

[1.2 Purpose and Scope 2](#_Toc105586161)

[1.3 Application Description 2](#_Toc105586162)

[1.3.1 User Characteristics / Types 2](#_Toc105586163)

[1.3.2 User Functionality 2](#_Toc105586164)

[1.3.3 User Interaction / Assumptions 3](#_Toc105586165)

[2 Non-Functional Requirements 3](#_Toc105586166)

[System Limitations 3](#_Toc105586167)

[Database Execution Speeds 3](#_Toc105586168)

[Security Restrictions 3](#_Toc105586169)

[Memory Requirements 4](#_Toc105586170)

[Organizational Requirements 4](#_Toc105586171)

[Programming Language and Development Environment 4](#_Toc105586172)

[Software Application Programming Interfaces (API) 4](#_Toc105586173)

[External Requirements 4](#_Toc105586174)

[3 Functional Requirements 4](#_Toc105586175)

[3.1 Employee Use Case(s) 5](#_Toc105586176)

[3.1.1 Description 5](#_Toc105586177)

[3.1.2 Characteristics of Activation 5](#_Toc105586178)

[3.1.3 Use Cases 5](#_Toc105586179)

[3.2 Managers / Owner Use Case(s) 13](#_Toc105586180)

[3.2.1 Description 13](#_Toc105586181)

[3.2.2 Characteristics of Activation 13](#_Toc105586182)

[3.2.3 Use Case(s) 13](#_Toc105586183)

# 1 Executive Overview

## Project Overview

This project is designed to produce a working deployable application for Northwest (NW) Lawn Maintenance with the goal of streamlining daily operations while bringing the company into the modern digital age of utilizing mobile devices to conduct daily business.

## Purpose and Scope

The purpose of this documentation is to describe the requirements needed for this application, users of this application, and what behavior the application should exhibit through normal, alternate and exception use cases.

## 1.3 Application Description

### 1.3.1 User Characteristics / Types

Users of this application will include only the following:

* Employees of NW Lawn Maintenance
* Managers / Owner of NW Lawn Maintenance

### 1.3.2 User Functionality

User functionality will depend on which user type is accessing the application. The following is the expected functionality per user group.

#### 1.3.2.1 Employees

Employees of NW Lawn Maintenance will make up most users of this application. Each employee should experience the same following:

* Ability to log in or out of the application via a username and password combination.
* Ability to update their personal information profile for management and the owner of NW Lawn Maintenance.
* Ability to change their password in the event of a lost credentials.
* Ability to view their current work schedule.
* Ability to start, stop, and update scheduled jobs.
* Send referrals through the appropriate referral system.
* Receive alerts based on updated work schedule changes.
* Receive alerts based on updated job scheduling changes.
* Ability to click on a job address and link to Google Maps for directions.

#### 1.3.2.2 Managers / Owner of NW Lawn Maintenance

While utilizing this application the managers and owner of NW Lawn Maintenance while including items in section [1.3.2.1](#_1.3.2.1_Employees), should experience the following additional functionality:

* Ability to modify each employees work schedule.
* Ability to modify customer job scheduling through the drag and drop system.
* Receive alerts when employee attempts to start/stop jobs outside established allowances.

### 1.3.3 User Interaction / Assumptions

There are several assumptions that tie in with proper user interaction of the application. These assumptions are as follows:

* Application is developed ethically with consideration to the intellectual property of Third-Party applications being used in conjunction. (i.e., Google Maps, Firebase Realtime Database).
* Application works only as designed to with users not attempting to force application to produce abnormal behavior.
* Customer Data is being protected and not shared with any Third-Party outside NW Lawn Maintenance.
* Users of this application are and will be only NW Lawn Maintenance employees and the management team.

# 2 Non-Functional Requirements

## System Limitations

The following limitations are what the application should not do. If the application does perform any of the following operations, it should be considered a bug in the system and require further debugging:

* Should not allow a non-user ability to log into application.
* Should not allow non-managers ability to log into the management portion of application.
* Should not allow employees to perform any tasks outside of [Section 1.3.2.1](#_1.3.2.1_Employees).
* Should not allow managers to perform any tasks outside of [Section 1.3.2.2](#_1.3.2.2_Managers_/).

## Database Execution Speeds

Access and execution speeds of the database are reliant on the individual user’s connection speed. Firebase is a real-time database therefore actual write/read execution speed varies.

## Security Restrictions

In the modern era of mobile devices, their respective operating systems (OS) have embedded security mechanisms. This application will be developed with the following restrictions in order to safeguard unwanted intrusions and/or breach of privacy:

* Allow access to the applications sandbox system files.
* Allow access to only the devices locational data.
* Allow access only to the user’s email service.
* Prohibit access to the OS system files.
* Prohibit access to the user’s files.

## Memory Requirements

Actual memory requirements currently vary during development stage. This application will be optimized to utilize minimal system resources regardless of mobile device.

## Organizational Requirements

The following describes the tools that will be used to develop this application.

### Programming Language and Development Environment

* Kotlin (a general-purpose programming language)
* Android Studio (an integrated development environment)

### Software Application Programming Interfaces (API)

The following API’s are required to connect an application written in Kotlin to Firebase:

* Authentication
* Realtime Database
* Cloud Firestore
* Cloud Functions
* Cloud Storage

The following API’s are required for the application to access Google services in Kotlin:

* LocationManager
* LocationProvider
* LocationRequest
* Google Maps

The following API’s are required to provide drag and drop functionality in application:

* RecyclerView
* CardView

The following API is required to provide email referral functionality in application:

* Intent

## External Requirements

This application must be developed within the guidelines established by International Standard Organization (ISO) and the Institute of Electrical and Electronics Engineers (IEEE).

# 3 Functional Requirements

This section describes an overview of the behavior expected based on which user type is utilizing the application.

## 3.1 Employee Use Case(s)

### 3.1.1 Description

The Employee Use Case describes the capability each user has on the application and the systems behavior.

### 3.1.2 Characteristics of Activation

The Employee Use Case is initiated by either NW Lawn Maintenance employees or management.

### 3.1.3 Use Cases

***Use Case 1:*** Launch Application

***Pre-condition:*** Device must be on with application installed

|  |  |
| --- | --- |
| **User** | **System** |
| Tap application icon on device. |  |
|  | Device loads application. |
|  | Application shows *‘Home’* screen. |
| End of use case. |  |

***Alternate Flow:*** Close application

|  |  |
| --- | --- |
| **User** | **System** |
| Tap the *‘Close’* button on application. |  |
|  | Application exits back to device. |
| End of alternate flow. |  |

***Exception Flow:*** Application fails to launch

|  |  |
| --- | --- |
| **User** | **System** |
| Tap application icon on device. |  |
|  | Device loads application. |
|  | Pop up message from device states that application failed to launch. |
| End of exception flow. |  |

***Use Case 2:*** Send Referral

***Pre-condition:*** Application running and on the *‘Home’* screen

|  |  |
| --- | --- |
| **User** | **System** |
| Tap the *‘Referral’* button on application. |  |
|  | Application loads the *‘Referral’* screen. |
| Tap each appropriate box and fill in the required information. |  |
| Tap the *‘Send’* button. |  |
|  | Application loads device email service with a pre-filled message. |
| Confirm information is correct and tap send. |  |
|  | Device email service sends message. |
|  | Application returns to *‘Home’* screen. |
| End of use case. |  |

***Exception Flow:*** Email fails to send

|  |  |
| --- | --- |
| **User** | **System** |
| Tap the *‘Referral’* button on application. |  |
|  | Application loads the *‘Referral’* screen. |
| Tap each appropriate box and fill in the required information. |  |
| Tap the *‘Send’* button. |  |
|  | Pop up message from device states it failed to load email service. |
|  | Application returns to the *‘Referral’* screen. |
| End of exception flow. |  |

***Use Case 3:*** Employee Login

***Pre-condition:*** Application running and on the *‘Home’* screen

|  |  |
| --- | --- |
| **User** | **System** |
| Tap the *‘Employee Login’* button on application. |  |
|  | Application loads the *‘Employee Login’* screen. |
| Enter in username and password. |  |
|  | Application connects to authentication server. |
|  | Application verifies credentials. |
|  | Application loads employee home screen. |
| End of use case. |  |

***Alternate Flow:*** Logout of account

|  |  |
| --- | --- |
| **User** | **System** |
| Click on the *‘logout’* button. |  |
|  | Application disconnects current user from the application. |
|  | Application returns to *‘Home’* screen. |
| End of alternate flow. |  |

***Exception Flow:*** Fail to log into account

|  |  |
| --- | --- |
| **User** | **System** |
| Tap the *‘Employee Login’* button on application. |  |
|  | Application loads the *‘Employee Login’* screen. |
| Enter in username and password. |  |
|  | Application connects to authentication server. |
|  | Application verifies credentials. |
|  | Application returns message stating *“Failed to login”*. |
|  | Application returns to the *‘Employee Login’*. |
| End of exception flow. |  |

***Use Case 4:*** Update Personal Information

***Pre-condition:*** Application running, and employee logged into account

|  |  |
| --- | --- |
| **User** | **System** |
| Tap the *‘Update Information’* button. |  |
|  | Application loads current employee profile. |
| Make changes to information. |  |
| Click on the *‘Update’* button. |  |
|  | Application connects to the database server. |
|  | Application updates appropriate database entries. |
|  | Application sends message the update was successful. |
|  | Application returns to *‘Employee Home’* screen. |
| End of use case. |  |

***Exception Flow:*** Information fails to save

|  |  |
| --- | --- |
| **User** | **System** |
| Tap the *‘Update Information’* button. |  |
|  | Application loads current employee profile. |
| Make changes to information. |  |
| Click on the *‘Update’* button. |  |
|  | Application connects to the database server. |
|  | Application updates appropriate database entries. |
|  | Application sends message the update was unsuccessful. |
|  | Application returns to *‘Update Information’* screen. |
| End of exception flow. |  |

***Use Case 5:*** Reset Password

***Pre-condition:*** Application running, employee logged into account, and at Update Information screen

|  |  |
| --- | --- |
| **User** | **System** |
| Click on the *‘Reset Password’* button. |  |
|  | Application loads password reset screen. |
| Enter in new password. |  |
| Enter in confirmation password. |  |
| Click on *‘Submit’*. |  |
|  | Application connects to database server. |
|  | Application updates authentication credentials. |
|  | Application sends message password reset successful. |
|  | Application returns to *‘Employee Home’* screen. |
| End of use case. |  |

***Exception Flow:*** Password fails to change

|  |  |
| --- | --- |
| **User** | **System** |
| Click on the *‘Reset Password’* button |  |
|  | Application loads password reset screen. |
| Enter in new password. |  |
| Enter in confirmation password |  |
| Click on *‘Submit’*. |  |
|  | Application connects to database server. |
|  | Application updates authentication credentials |
|  | Application sends message the password reset was unsuccessful. |
|  | Application returns to password reset screen. |
| End of exception flow. |  |

***Use Case 6:*** View Schedule

***Pre-condition:*** Application running, schedule menu screen, and employee logged into account

|  |  |
| --- | --- |
| **User** | **System** |
| Click on *‘Current Week’* button. |  |
|  | Application loads employee schedule. |
| End of use case. |  |

***Alternate Flow:*** Return to employee home

|  |  |
| --- | --- |
| **User** | **System** |
| Click on *‘Home’* button. |  |
|  | Application returns to *‘Employee* Home’ screen. |
| End of alternate flow. |  |

***Exception Flow:*** Schedule fails to load

|  |  |
| --- | --- |
| **User** | **System** |
| Click on *‘Current Week’* button. |  |
|  | Application fails to schedule. |
|  | Application sends message schedule failed to load. |
|  | Application returns to *‘Schedule Menu’* screen. |
| Ends of exception flow. |  |

***Use Case 7:*** View Workday

***Pre-condition:*** Application running, schedule menu screen, and employee logged into account

|  |  |
| --- | --- |
| **User** | **System** |
| Click on *‘Current Workday’* button. |  |
|  | Application loads current workday. |
| End of use case. |  |

***Alternate Flow:*** Return to schedule

|  |  |
| --- | --- |
| **User** | **System** |
| Click on *‘Current Workday’* button. |  |
|  | Application loads current workday. |
| Click on *‘Back’* button. |  |
|  | Application loads *‘Schedule Menu’* screen. |
| End of alternate flow. |  |

***Exception Flow:*** Workday fails to load

|  |  |
| --- | --- |
| **User** | **System** |
| Click on *‘Current Workday’* button. |  |
|  | Application fails to load current workday. |
|  | Application sends message fail to load current day. |
|  | Application returns to *‘Schedule Menu’* screen. |
| End of exception flow. |  |

***Use Case 8:*** View Customer Job

***Pre-condition:*** Application running, schedule menu screen, and employee logged into account

|  |  |
| --- | --- |
| **User** | **System** |
| Click on *‘Current Workday’* button. |  |
|  | Application loads current workday. |
| Click on customer job. |  |
|  | Applications loads *‘Customer Job’* screen. |
| End of use case. |  |

***Alternate Flow:*** Return to workday

|  |  |
| --- | --- |
| **User** | **System** |
| Click on *‘Current Workday’* button. |  |
|  | Application loads current workday. |
| Click on customer job. |  |
|  | Applications loads *‘Customer Job’* screen. |
| Click on *‘Back’* button. |  |
|  | Application returns to current workday. |
| End of alternate flow. |  |

***Exception Flow:*** Selected job fails to load

|  |  |
| --- | --- |
| **User** | **System** |
| Click on *‘Current Workday’* button. |  |
|  | Application loads current workday. |
| Click on customer job. |  |
|  | Applications fails to load customer job. |
|  | Application sends message fail to load customer job. |
|  | Application returns to current workday. |
| End of exception flow. |  |

***Use Case 9:*** Start / Stop Customer Job

***Pre-condition:*** Application running, current customer loaded, and employee logged into account

|  |  |
| --- | --- |
| **User** | **System** |
| Click on *‘Start /* Stop’ button. |  |
|  | Application connects to database server. |
|  | Application stores current date and timestamp. |
|  | Application sends message customer job started/stopped. |
| End of use case. |  |

***Exception Flow:*** Fail to start / stop customer job

|  |  |
| --- | --- |
| **User** | **System** |
| Click on *‘Start /* Stop’ button. |  |
|  | Application connects to database server. |
|  | Application fails to store current date and timestamp. |
|  | Application sends message customer job start/stop unsuccessful. |
| End of exception flow. |  |

***Use Case 10:*** Update Customer Job

***Pre-condition:*** Application running, current customer loaded, and employee logged into account

|  |  |
| --- | --- |
| **User** | **System** |
| Click on text box on *‘Customer Job’* screen. |  |
| Enter notes into text box. |  |
| Click on *‘Update’* button. |  |
|  | Application connects to database server. |
|  | Application updates customer information. |
|  | Application sends message update was successful. |
|  | Application returns to current customer job. |
| End of use case. |  |

***Exception Flow:*** Fail to update customer jobs

|  |  |
| --- | --- |
| **User** | **System** |
| Click on text box on *‘Customer Job’* screen. |  |
| Enter notes into text box. |  |
| Click on *‘Update’* button. |  |
|  | Application connects to database server. |
|  | Application fails to update customer information. |
|  | Application sends message update was unsuccessful. |
|  | Application returns to current customer job. |
| End of exception flow. |  |

## 3.2 Managers / Owner Use Case(s)

### 3.2.1 Description

The Manager Use Case describes the capability each user has on the application and the systems behavior. Management has access to all Employee Use Cases in addition to the use cases found in this section. For any of the Employee Use Cases refer to [Section 3.1](#_3.1_Employee_Use).

### 3.2.2 Characteristics of Activation

The Manager Use Case is initiated by the organization’s management only.

### 3.2.3 Use Case(s)

***Use Case 1:*** Manage Employee

***Pre-condition:*** Application running, and manager logged into account

|  |  |
| --- | --- |
| **User** | **System** |
| Click on *‘Manage Employee’* button. |  |
|  | Application loads employee list. |
| Select specific employee. |  |
|  | Application loads specific employee. |
| Modify employee information. |  |
| Click the *’Update’* button. |  |
|  | Application connects to database server. |
|  | Application updates employee. |
|  | Application returns to employee list. |
| End of use case. |  |

***Exception Flow:*** Failed to update employee

|  |  |
| --- | --- |
| **User** | **System** |
| Click on *‘Manage Employee’* button. |  |
|  | Application loads employee list. |
| Select specific employee. |  |
|  | Application loads specific employee. |
| Modify employee information. |  |
| Click the *’Update’* button. |  |
|  | Application connects to database server. |
|  | Application fails to update employee. |
|  | Application sends message employee update failed. |
| End of exception flow. |  |

***Use Case 2:*** Manage Customer

***Pre-Condition:*** Application running, and manager logged into account

|  |  |
| --- | --- |
| **User** | **System** |
| Click on *‘Manage Customer’* button. |  |
|  | Application loads customer list. |
| Select specific customer. |  |
|  | Application loads specific customer. |
| Modify customer information. |  |
| Click the *’Update’* button. |  |
|  | Application connects to database server. |
|  | Application updates customer. |
|  | Application returns to customer list. |
| End of use case. |  |

***Exception Flow:*** Failed to update customer

|  |  |
| --- | --- |
| **User** | **System** |
| Click on *‘Manage Customer’* button. |  |
|  | Application loads customer list. |
| Select specific customer. |  |
|  | Application loads specific customer. |
| Modify customer information. |  |
| Click the *’Update’* button. |  |
|  | Application connects to database server. |
|  | Application fails to update customer. |
|  | Application sends message customer update failed. |
| End of exception flow. |  |

***Use Case 3:*** Add New Employee

***Pre-condition:*** Application running, and manager logged into account

|  |  |
| --- | --- |
| **User** | **System** |
| Click on *‘Add New Employee’* button. |  |
|  | Application loads new employee form. |
| Fill in new employee form information. |  |
| Click on ’Submit’ *button.* |  |
|  | Application connects to database server. |
|  | Application updates database server. |
|  | Application sends message update was successful. |
|  | Application returns to *‘Manager Home’* screen. |
| End of use case. |  |

***Exception Flow:*** Failed to add new employee

|  |  |
| --- | --- |
| **User** | **System** |
| Click on *‘Add New Employee’* button. |  |
|  | Application loads new employee form. |
| Fill in new employee form information. |  |
| Click on ’Submit’ *button.* |  |
|  | Application connects to database server. |
|  | Application fails to update database. |
|  | Application sends message that update was unsuccessful. |
|  | Application returns to new employee form. |
| End of exception flow. |  |

***Use Case 4:*** Add New Customer

***Pre-condition:*** Application running, and manager logged into account

|  |  |
| --- | --- |
| **User** | **System** |
| Click on *‘Add New Customer’* button. |  |
|  | Application loads new customer form. |
| Fill in new customer form information. |  |
| Click on ’Submit’ *button.* |  |
|  | Application connects to database server. |
|  | Application updates database server. |
|  | Application sends message update was successful. |
|  | Application returns to *‘Manager Home’* screen. |
| End of use case. |  |

***Exception:*** Failed to add new customer

|  |  |
| --- | --- |
| **User** | **System** |
| Click on *‘Add New Customer’* button. |  |
|  | Application loads new customer form. |
| Fill in new customer form information. |  |
| Click on ’Submit’ *button.* |  |
|  | Application connects to database server. |
|  | Application fails to update database. |
|  | Application sends message that update was unsuccessful. |
|  | Application returns to new customer form. |
| End of exception flow. |  |

***Use Case 5:*** Manage Schedule

***Pre-condition:*** Application running, and manager logged into account

|  |  |
| --- | --- |
| **User** | **System** |
| Click on *‘Manage Schedule’* button. |  |
|  | Application loads schedule management. |
|  | Application connects to database server. |
| Select date and time slot. |  |
|  | Application loads date and time slot. |
| Select customer. |  |
| Confirm choices. |  |
|  | Application updates schedule. |
|  | Application sends to message schedule update was successful. |
| End of use case. |  |

***Exception Flow:*** Failed to load schedule

|  |  |
| --- | --- |
| **User** | **System** |
| Click on *‘Manage Schedule’* button. |  |
|  | Application loads schedule management. |
|  | Application fails to connect to database server. |
|  | Application sends message failed to load schedule. |
| End of exception flow. |  |

***Exception Flow:*** Failed to save schedule changes

|  |  |
| --- | --- |
| **User** | **System** |
| Click on *‘Manage Schedule’* button. |  |
|  | Application loads schedule management. |
|  | Application connects to database server. |
| Select date and time slot. |  |
|  | Application loads date and time slot. |
| Select customer. |  |
| Confirm choices. |  |
|  | Application fails to update schedule. |
|  | Application sends message schedule update failed. |
| End of exception flow. |  |