**Capstone Project**

**Overview**

This document serves as a complete description of the design of my capstone project. The capstone is a stand-alone application that schedules appointments for use across multiple countries. The intended audience of this application may look like a international company that takes appointments with contacts either in/ or outside of their company.

**Table of Contents**

1. Introduction
2. Scope and Requirements
3. Architecture Overview
4. Data Design
5. UI Design
6. Security Considerations
7. Testing Strategy
8. Performance and Scalability
9. Timelines and Milestones
10. Change Log
11. Test Case Results

**Introduction**

The application provides diverse business opportunities for many different customer bases. The solution’s design can streamline scheduling processes such as booking, updating or canceling appointments. Business opportunities for this solution can range anywhere from salons, sales teams, medical teams or other professional services. The business requirements the software solution can fulfill makes the target consumer other businesses in need of scheduling software. Because the system does not require many resources, the solution may best fulfill the needs of small to medium size businesses. The scheduling software will fulfills the needs to store users, customers, and appointment data.

The high-level goals of this application include being user-friendly, salable, secure, as well as being able to validate form information, convert time zone to be most relevant to the user, modify and add data from the user interface, generate reports, and connect to a database.

**Scope and Requirements**

**The application must be a stand-alone Java Application**.

**The application’s source code must display inheritance, polymorphism, and encapsulation.**

Inheritance is expressed in the Stakeholder class, which inherits traits from the contact class. Stakeholders are key actors that the user will be able to make an appointment with. Stakeholders are different from contacts in that they have a classification in what they are and what their relationship with the company will be.

Polymorphism is expressed in the application through through the use of overloaded methods, and overloaded constructors such as the overloaded methods that will be used to create the functionality of the search bars.

Encapsulation is expressed in the application through the structure of most of the java classes throughout the project. Many of which have variables and methods on which they interact with.

**The application must have a search bar and show results from the search.**

The application will contain a search bar to search appointments by appointment id, or name anywhere that the appointment information is viewable.

**The application must have a database component.**

MySql is the foundational data storage used with this application. It serves as an important relational database system to facilitate storage, and retrieval of structured data for the application.

**The application must have exception controls.**

The core use of exception handling within the application will be the handing of SQL exceptions when working with the database, and error detection within the applications forms. A log of log-in attempts will also be contained in the applications documentation.

**The application must have industry appropriate security features.**

The application will express security in the form of user validity, user authentication, and input validation. These key security features will be in use throughout the entirety of the application.

**The application must contain design elements that make the application scalable**.

The application displays many components of a scalable program. The class structure of the program is designed to have as little dependencies as possible, therefore increasing the scalability of the program in terms of flexibility. The code has also been designed with reusable code in mind, and MySql is potentially a very scalable database system to use.

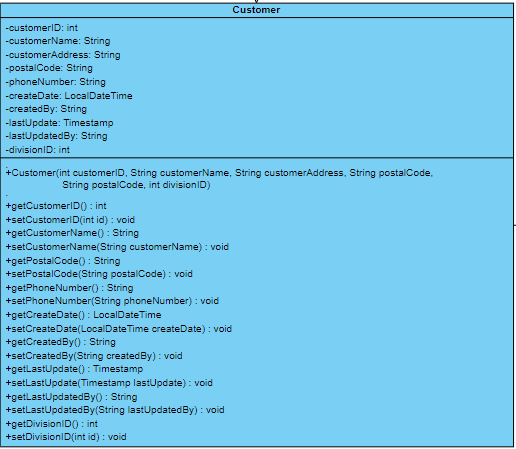
**The application must be user-friendly.**

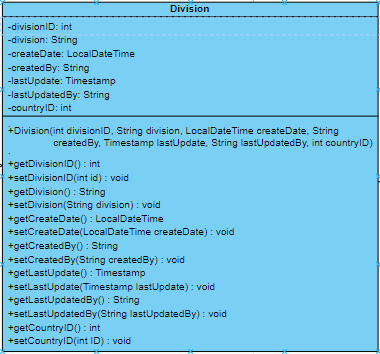
The user interface of the application is user-friendly because it is intuitive, has a consistent design, a minimal learning curve, has effective error handling, and has effective navigation.

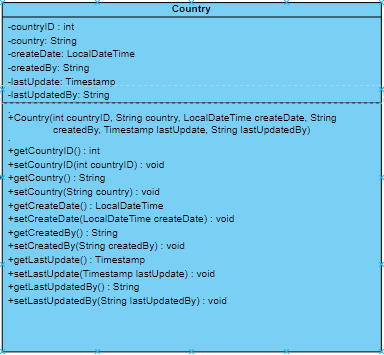
**Architecture Overview**

A full UML diagram is contained in the documents folder of this project, and outlines the structure of the class architecture displayed in the application. The following diagrams are the core classes of the application.

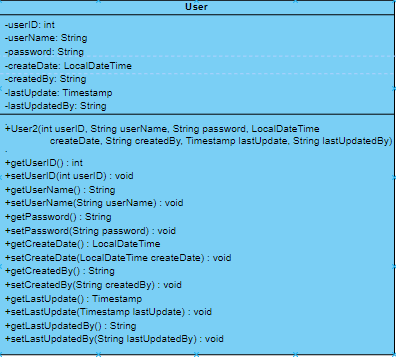


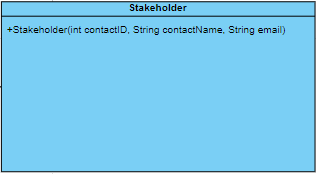
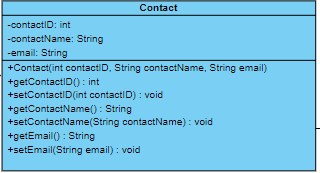




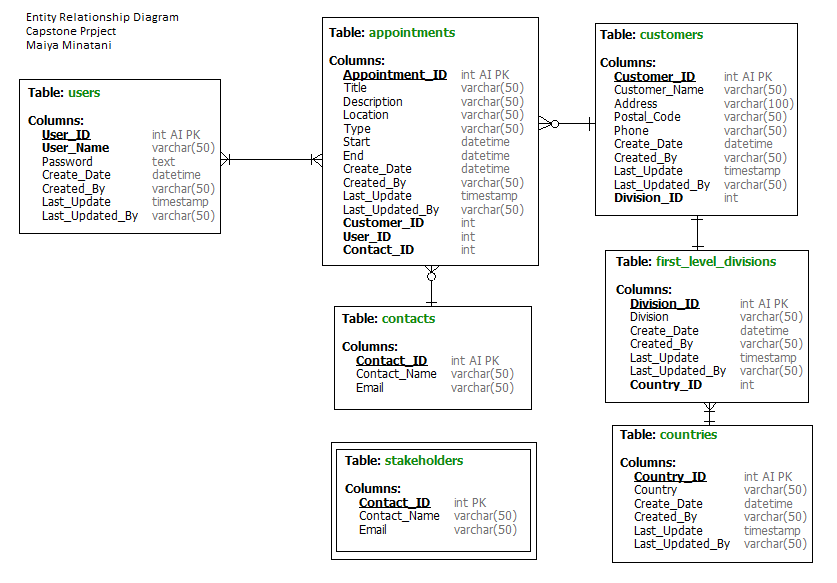
.

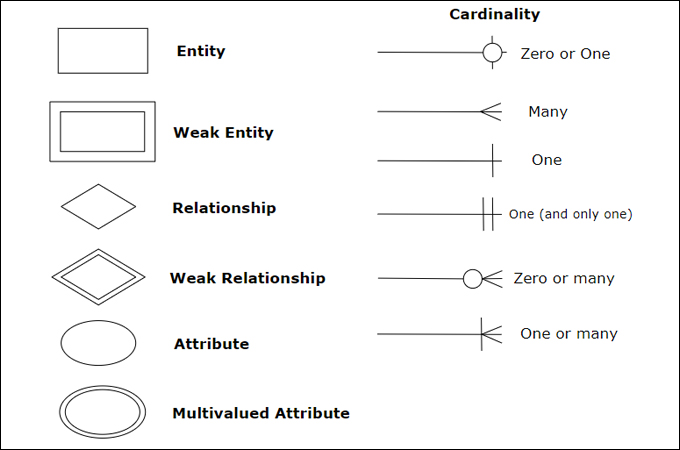






**Data Design**

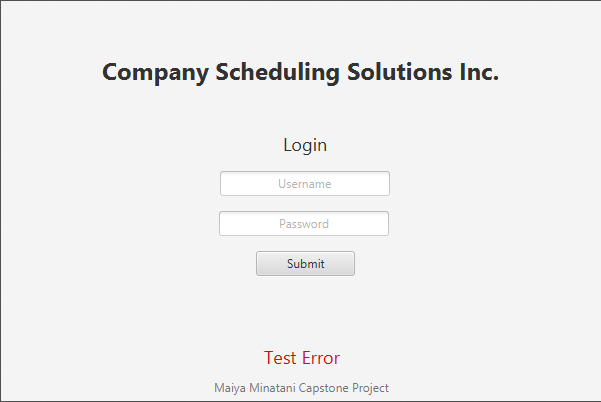




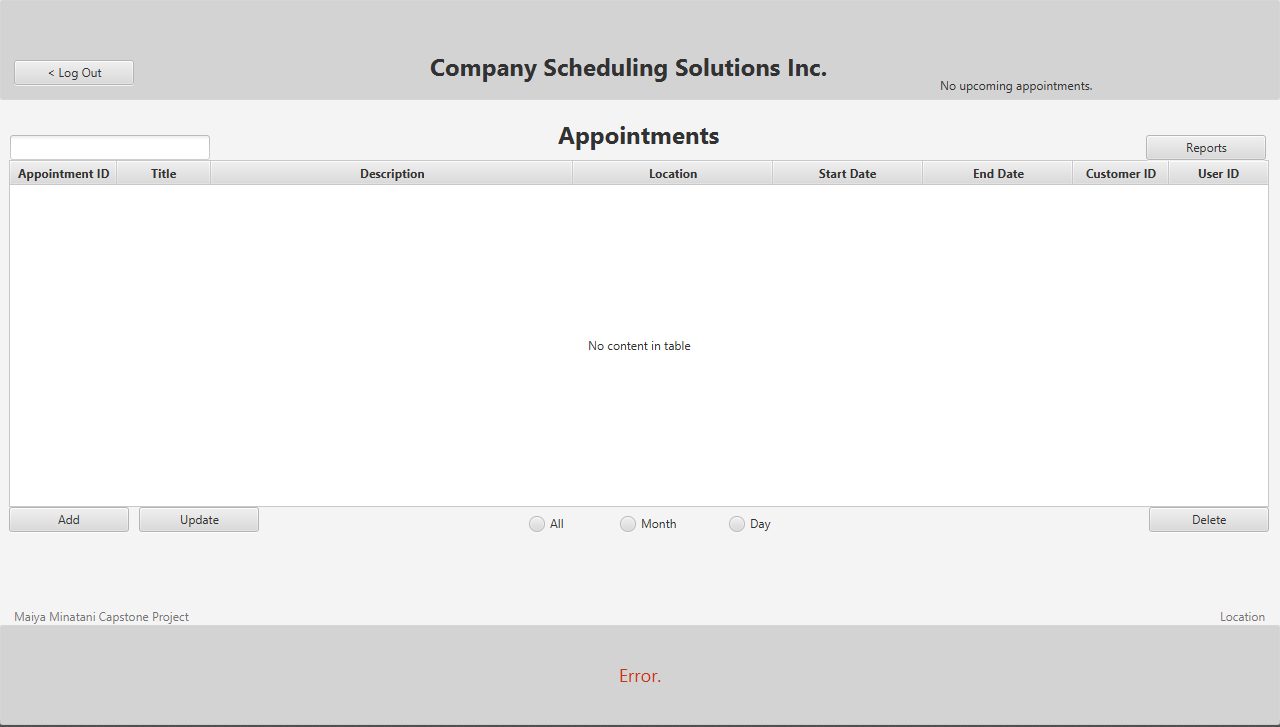
My ERD shows the data architecture prepared within the database (MySQL). The database contains 3 different types of relationships, that being M:M, 1:M, and 1:1.

**UI Design**

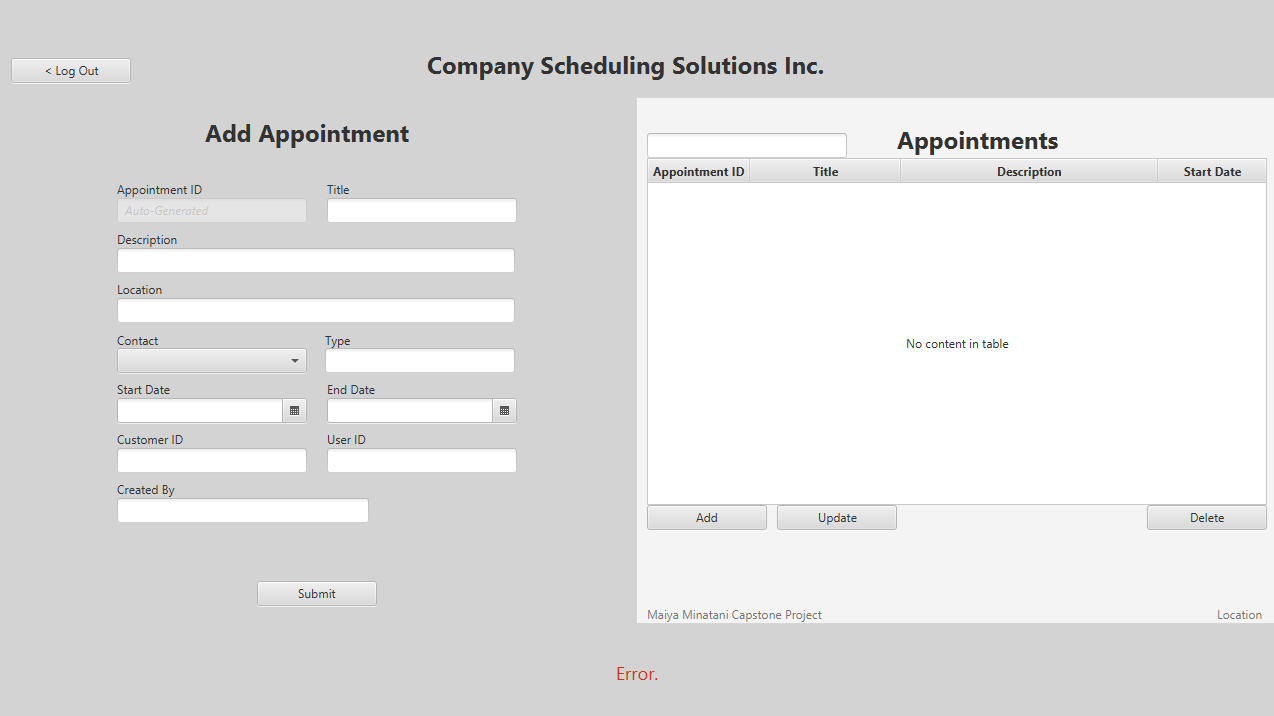
The design of my of my document will be built in SceneBuilder version 20.0.0. This program efficiently builds FXML files, and in turns allows for a quick build of the GUI that will be used in the application. This makes it a useful tool for building mock ups.



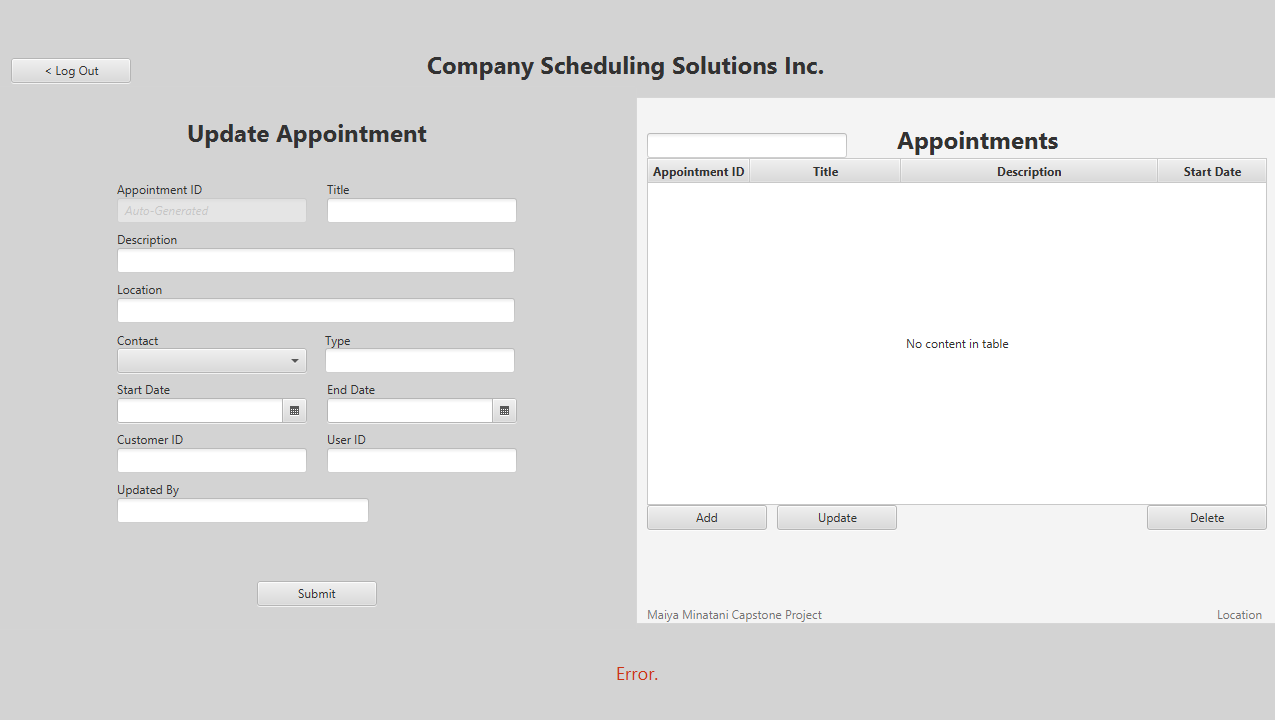
The first scene in the application is the login screen. It allows the user to enter a username and password to gain access to the program. Log-in details are stored in the database, therefore for practical purpose of the project the username/password is either “test/test” or “user/user.”



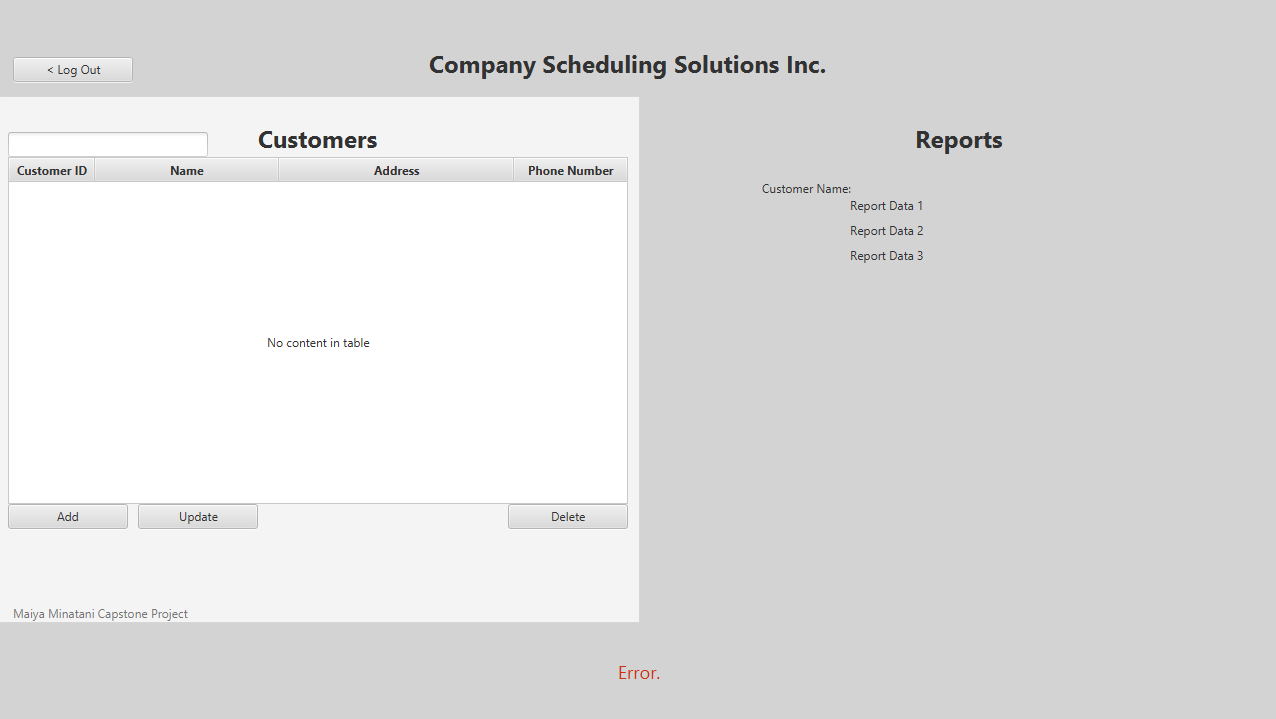
The main screen features a list of appointments that can be filtered by either month or day and shows all appointments by default. The user will also be able to log out, search the appointments, add, update or delete appointments, view reports, be notified of appointments and view the current local location as well as error messages. This screen features the most used functionalities of the application to the user.



The user will be able to go to the add appointment form from the main screen. On this form there is a preview of all the appointments, and a form the user can fill out. The Appointment ID is disabled, and auto generated on clicking the submit button. Clicking the submit button saves the appointment into the database, and directs the user back to the main screen.



The update appointment form has similar functionality to the add appointment form, except the the form opens populated with the data from the appointment the user will be updating.



The structure of the report pages will contain a data table preview, and the featured report data. From this scene the user will be able to update the data, and go back to the main screen.

**Security Considerations**

Security is a significant concern when designing the application, especially because of the potential to handle personal information. The application does through methods of authentication, authorization, input validation, logging and monitoring.

Authentication and Authorization:

The application only allows access to registered users. Users can gain this access by entering their username and password into the login screen before entering into the applications’ systems.This is important as a security measure because it ensures that users can only gain access to information and features they are permitted to.

Logging and Monitoring:

As well as having authentication methods, the application also logs and montors login attempts. This is useful as a security measure to help identify suspicious activity.

Input validation:

The application will also have strong input validation. This is an important security measure because users or bad actors can potentially spoil data without it.

**Testing Strategy**

The testing of this application will consist of a series of acceptance tests. Acceptance tests verify the business requirements of an application by focusing on user behaviors while the entire application is running. This type of testing will be useful to fulfill the requirements of the application because it tests two core components: A) the user-friendliness of the application and how the application interacts with users and B) how the program will react to real-life situations and how the application will be used.

Test Case 1

The purpose of the first test is to verify that the application behaves correctly when validating that appointments for a specific contact do not overlap. The tester will attempt to add new appointments that overlap, and a successful test-case will show errors to user that the attempts overlap, and that application will not accept that input for a new appointment.

Test steps :

1. Add an appointment that occurs between 1:00PM and 2:00PM Eastern Time (This initial appointment should be accepted without issue).
2. Add an appointment that occurs between 1:00PM and 1:30PM Eastern Time.
3. Add an appointment that occurs between 1:30PM and 2:00PM Eastern Time.
4. Add an appointment that occurs between 12:30PM and 1:30PM Eastern Time.
5. Add an appointment that occurs between 1:30PM and 2:30PM Eastern Time.
6. Add an appointment that occurs between 1:00PM and 1:30PM Eastern Time.

A successful test case will have error messages relayed to the user about overlapping appointments for steps 2 through 6.

I expect this test to be unsuccessful the first time it is carried out.

Test Case 2

The purpose of the second test case is to verify the application’s localization ability. This test requires access to both the database, and the the full applications access. The tester will add and edit appointments in local time, verifying the associated eastern time in the database. A successful test case will accept only localized times, and saves in the database as Eastern Time.

Test steps:

1. Add an appointment that occurs between 10:00 AM and 11:00AM Pacific Time.
2. Add an appointment that occurs between 1:00 PM and 2:00 PM Pacific Time.

A successful test case will save the first appointment as starting at 10:00 AM Local time within the application, and within the database will have saved the appointment as 1PM Eastern time, and saving the second appointment starting at 1:00 PM Local in the application, and 4:00 PM Eastern within the database. It will also save the end time of the first appointment within the application as 11:00 AM Local, and 2:00PM Eastern within the database, and it will save the end time of the second appointment as 2:00 PM Local within the application and 6:00 Eastern within the database. Adding these appointments should also not throw any input validation errors within the user interface.

I expect this test to be successful.

**Performance and Scalability**

The performance and scalability of the application is vital and many measures have been implemented to ensure it such as as the scalable architecture design, server infrastructure, and it’s security considerations.

The architecture design and class design is created with scalability and performance in mind because of it’s lack of redundant code, modular build, and potential growth. The application can easily implement new and improved features, and performs very quickly because of its design.

The server utilizes MySQL workbench, which is best used with small t mid-size databases. The potential for it to be refactored to a cloud provider would support a large-scale database, but it is currently able to be horizontally and vertically scaled through MySQL.

It’s security considerations also allow for scalability in that securing all components of the application and it’s database will allow for the continued use of them.

**Timelines and Milestones**

Start Date: September 9, 2023

Design: 20 Days (Milestone 1

Resources:

+ WPS Office

+ Scene Builder

Implementation: 30 Days (Milestone 2)

Dependencies:

+ Design Docs

+ Mock-Ups

+ Stakeholder Feedback

Resources:

+ IntelliJ

+ GitHub

+ MySql WorkBench

+ Scene Builder

+ Design Documents

Testing/Deployment (15 Days)

Dependencies:

+ App functionality

+ Testing Documentation

Resources:

+ Software Application

+ Test Cases

+ Design Documents

+ WPS Office

End Date: November 15, 2023

Milestone 1: Completed September 28, 2023

**Change Log**

\* DATE / DESCRIPTION of anything that changes from these documents,

**Testing Case Results**