Software Design

Document

for

<City Pave>

**Version 3 approved**

**Prepared by <city pave Online Application for Street and Highway Pavements Design>**

**<Department of Civil Engineering, CSULA>**

**<5/13/2021>**

# Revision History

|  |  |  |  |
| --- | --- | --- | --- |
| Name | Date | Reason For Changes | Version |
| City Pave | 12/9/2020 | Setting up pages | v1 |
| City Pave | 4/3/2021 | Pages has become live | v2 |
| City Pave | 5/11/2021 | Homepage update | v3 |
|  |  |  |  |

# Table of Contents

[**Revision History**](#_heading=h.g082gap3l8ql) **2**

[**Table of Contents**](#_heading=h.pw9tzsaz7ypy) **3**

[**1. Introduction**](#_heading=h.wydkl3av5ixk) **4**

[1.1 Purpose](#_heading=h.zfhx1pu9zwgp) 4

[1.2 Document Conventions](#_heading=h.n37oeqbdgi6a) 4

[1.3 Intended Audience and Reading Suggestions](#_heading=h.jtvhyg9wq5zf) 4

[1.4 System Overview](#_heading=h.y83pimdhnukv) 4

[**2. Design Considerations**](#_heading=h.8uff5ldgu7bk) **5**

[2.1 Assumptions and Dependencies](#_heading=h.6ofjr35onjaj) 5

[2.3 Goals and Guidelines](#_heading=h.txoe8uvatr9i) 6

[2.4 Development Methods](#_heading=h.xfsugx7uvmtr) 7

[**3. Architectural Strategies**](#_heading=h.1uvp5hgfz2aj) **8**

[**4. System Architecture**](#_heading=h.1cfpif75kup1) **9**

[**5. Policies and Tactics**](#_heading=h.x08w7a88uxo) **11**

[5.1 Choice of which specific products used](#_heading=h.qszta43rw50a) 11

[5.2 Plans for ensuring requirements traceability](#_heading=h.4d8bv86n957u) 11

[5.3 Plans for testing the software](#_heading=h.njfh3cnm06m5) 11

[**6. Detailed System Design**](#_heading=h.6wzpwjetd5yg) **12**

[Assets](#_heading=h.r81tn27lvsrl) 12

[Analyze Pavement Structure](#_heading=h.1vrhnyir8vq3) 12

[Calculate\_LifeCyle\_Cost](#_heading=h.w4fdfg851t9v) 13

[Determine\_Pavement\_Structures](#_heading=h.ivvhj8jmr7nj) 14

[Estimate\_Material\_Cost](#_heading=h.ad5py65sw59r) 15

[Project page](#_heading=h.jtlrgcobma00) 15

[Client](#_heading=h.hb9men411aqb) 16

[Databases](#_heading=h.f5s4lkg9pcs1) 16

[**7. Detailed Lower level Component Design**](#_heading=h.j4uatcyurg3f) **18**

[7.Files](#_heading=h.dbsjb7qn6av0) 18

[7.1 Classification](#_heading=h.5qn97cuq6gyz) 18

[7.2 Processing Narrative (PSPEC)](#_heading=h.s0hl9owl8yv2) 18

[7.3 Interface Description](#_heading=h.qsf19vmaxxq4) 18

[7.4 Processing Detail](#_heading=h.d2eo6mqgbxy8) 18

[7.4.2 Restrictions/Limitations](#_heading=h.r25uk222b0b8) 18

[7.4.3 Performance Issues](#_heading=h.xdfd2ku7rnsw) 18

[7.4.4 Design Constraints](#_heading=h.kdj8mi561qn5) 19

[**8. User Interface**](#_heading=h.5yhlqkj5ozal) **20**

[8.1 Overview of User Interface](#_heading=h.tn3xd0q8ocb8) 20

[8.2 Screen Frameworks or Images](#_heading=h.2zk04swp4qbp) 21

# 1. Introduction

## 1.1 Purpose

The purpose of this document is to inform individuals about what this project is about. Which allows individuals to get a better understanding of the program.

## 1.2 Document Conventions

The higher level requirements for city pave was creating a database to store information that can be accessed by our program. This higher level requirement is a database that stores account information and project details allowing the user to be able to get access to their information in a fast and easy way.

## 1.3 Intended Audience and Reading Suggestions

This document is intended for someone who would like to learn more about City Pave and would like to learn the technical aspects. This document would also be intended for engineers to go over in case they would like to edit the project, see how the project was designed.

## 1.4 System Overview

City pave application will be implemented as a web page application. This web page application will be able to calculate pavement calculations. These calculations range from many types of calculations from pavement layer, pavement cost, life cycle cost and etc following the AASHTO guidelines.

# 2. Design Considerations

This section describes many of the issues which need to be addressed or resolved before attempting to devise a complete design solution.

## 2.1 Assumptions and Dependencies

● Related software or hardware

o Certain software compatibility issues with what the software required.

▪ Caused us to switch to many different types of libraries.

● End-user characteristics

o Trying to figure out what the end user wants in a program.

▪ How do we make it more convenient for these users to use the program

▪ Along with how the end user will use this program to make his/her life easier.

**2.2 General Constraints**

● Hardware or software environment

o Have a web browser that has all the plug in and internet access.

● End-user environment

o He/she needs to be able to have an up to date web browser

● Availability or volatility of resources

o Coding language being updated so some features may not work depending on the version its on

● Standards compliance

o Brand guidelines

● Data repository and distribution requirements

o Cost for website host and database host

● Security requirements (or other such regulations)

o The program has basic encryption process to make sure end users are safe from hackers

● Memory and other capacity limitations

o Based on Third party host for the website

● Performance requirements

o Standard system requirements for being able to use a web browser

● Network communications

o End user has to have a stable internet connect

o Third party host has to be able to maintain server connection

● Verification and validation requirements (testing)

o Use a user base validation

o Encrypt user password

## 2.3 Goals and Guidelines

**Goals**

* Emphasis on following the brand guidelines
* The product should work, look, or "feel" better then existing programs
* The KISS principle
  + Keeping things simple so that it is easier to use and understand.

**Guidelines**

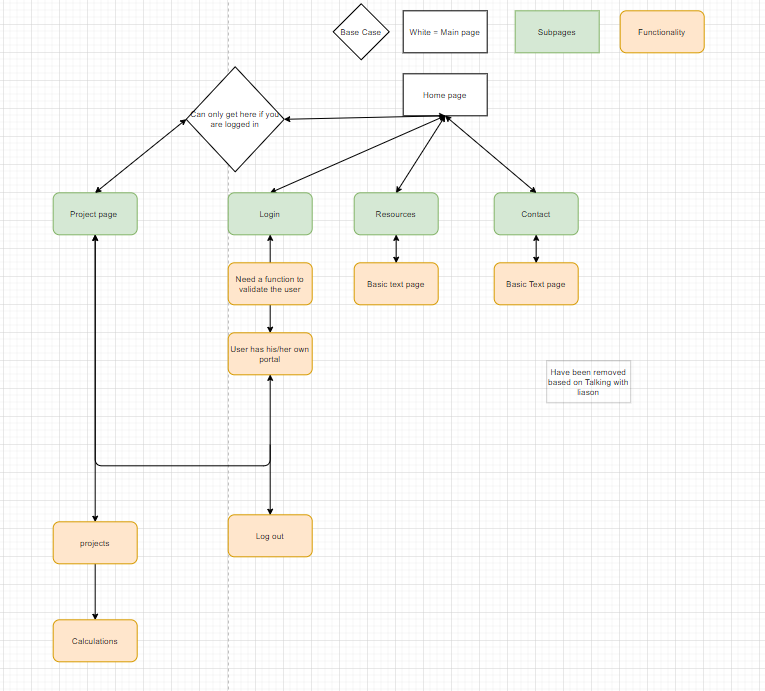
* The site should follow the current style guideline
* The site functions properly as expected
* The site is easy to use

## 2.4 Development Methods

We decided to use the process of comparing working websites and improve on what they have structured. We did this process because this is what one of the group members learned from work. We did this by following a lot of the recommendations that the liaison noted as a site to use as a reference. From there we decided to create an excel sheet and log the good and the bad of the website that we visited.

# 3. Architectural Strategies

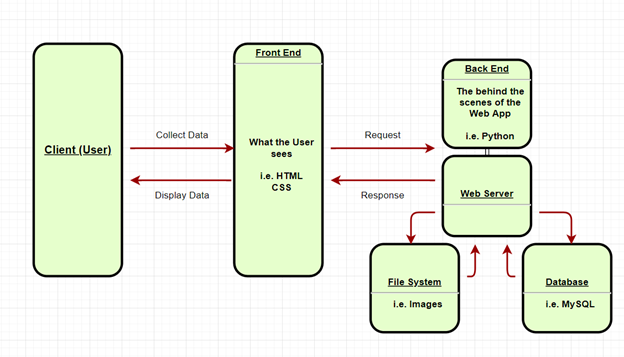
Our program architecture is based on a Model-view-controller pattern. The reason is that our group focused on making different models, views and controllers for different parts of our program. The reason we chose this is because this model works well with web application and since we are creating a web application we believed that this was a perfect fit for us. The programs we used were node.js, sequelize, html, css, javascript, and heroku for web hosting. The diagram belows shows our simple version architectural design.



# 4. System Architecture

****

This is a simple design explanation of how our program works

This diagram shows the higher level of how the website works.

# 5. Policies and Tactics

## 5.1 Choice of which specific products used

**IDE**

· Visual studio

**Programing language**

· Javascript

· Html

· Css

**Database**

· Mysql

**Cloud services**

· Heroku

## 5.2 Plans for ensuring requirements traceability

The way that we ensure requirement traceability is that we use github and everytime we make a push we have a meeting to confirm what we say is pushed to the main branch is working correctly. This helps us ensure that the requirements have traceability. This also allows us to follow what we have finished and created.

## 5.3 Plans for testing the software

**Basic QA testing.**

o Testing and seeing if result comes out as predicted

o See if there is any linking errors between tabs

o See if the ports are being called correct

o Seeing if information is being stored correctly

# 6. Detailed System Design

## Assets

alligator\_high.jpg

· Image for pavement structure

alligator\_low.jpg

· Image for pavement structure

alligator\_moderate.jpg

· Image for pavement structure

transverse\_high.jpg

· Image for pavement structure

transverse\_low.jpg

· Image for pavement structure

transverse\_Moderate.jpg

· Image for pavement structure

Icon.png

· Image for icon for city pave

## Analyze Pavement Structure

· AnalyzePavementStructure.vue

o Form for analyze pavement structure

o Main form page for analyze pavement structure

· CrossSection.vue

o Subcategory for analyze pavement structure

· Loads.vue

o Subcategory for analyze pavement structure

· ResponseLocations(X,Y).vue

o Subcategory for analyze pavement structure

· ResponseLocations(Z).vue

o Subcategory for analyze pavement structure

· Results.vue

o Subcategory for analyze pavement structure

Calculate\_LifeCyle\_Cost

· CalculateLifeCycleCost.vue

o Main program for calculate life cycle cost

· Alternatives.vue

o Sub program for calculate life cycle cost

· AnalysisOptions.vue

o Sub program for calculate life cycle cost

· Results.vue

o Sub program for calculate life cycle cost

o Result page for calculate life cycle cost

· ScenarioInfoLife.vue

o Sub program for calculate life cycle cost

· TrafficData.vue

o Sub program for calculate life cycle cost

· TrafficHourlyDis.vue

o Sub program for calculate life cycle cost

· ValueOfUser.vue

o Sub program for calculate life cycle cost

## Determine\_Pavement\_Structures

· ConditionSurvey.vue

o Sub program for determining pavement structure

· DesignGuidance.vue

o Sub program for determining pavement structure

· DesignParameters.vue

o Sub program for determining pavement structure

· DeterminePavementStructure.vue

o Main program for Determining pavement structure

· LayerCoefficients.vue

o Sub program for determining pavement structure

· Layers.vue

o Sub program for determining pavement structure

· NondestructiveTesting.vue

o Sub program for determining pavement structure

· PavementLayers.vue

o Sub program for determining pavement structure

· PavementStructure.vue

o Sub program for determining pavement structure

· PavementSubStructure.vue

o Sub program for determining pavement structure

· ScenarioInfo.vue

o Sub program for determining pavement structure

· TrafficAndLoading.vue

o Sub program for determining pavement structure

## Estimate\_Material\_Cost

Area.vue

o Sub program for calculate life cycle cost

CrossSection.vue

o Sub program for calculate life cycle cost

EstimateMaterialCost.vue

o Main program for calculate life cycle cost

Results.vue

o Sub program for calculate life cycle cost

o Show the results of the calculations from the other forms

## Project page

Project.vue

· Main program to navigate project

## Client

App.vue

· Navigation bar

http-common.js

· Port information for hosting

main.js

· Sets up Vue to run

prettyJson.js

· Set up function for vue

router.js

· Routing paths for each component

## Databases

conditionLayers.model.js

determine.model.js

index.js

layer.model.js

layerCoefficients.model.js

layerType.model.js

pavementLayers.model.js

porousLayers.model.js

project.model.js

scenarioEst.model.js

user.model.js

# 7. Detailed Lower level Component Design

## 7.Files

## 7.1 Classification

There are not many different types of files; they are all js script files or vue files which are used for forms.

## 7.2 Processing Narrative (PSPEC)

The way that the process works is the files are uploaded to the remote server. From there we access the files using the link Citypave.app . Then we go and interact with the website which calls the file based on what directory it sends us to.

## 7.3 Interface Description

The interface is very simple, it is based on a directory approach. One to freely change parts here and there very seamlessly or even take away parts.

## 7.4 Processing Detail

The way the information gets processed is simply user contact server. Then the server gives information to the user. After that the user gets a new generated page. From there the user gives another input and the server responds.

## 7.4.2 Restrictions/Limitations

The restriction and limitation is that we are reliant on a host server to keep our program running. Another limitation is the funds that we have currently because the program does not generate money in any way. This causes some of the paid features to be off loaded on the individual who decides to host.

## 7.4.3 Performance Issues

There is not a performance issue currently. Only issue would be if the host server goes down.

## 7.4.4 Design Constraints

The design constraints are that we are reliants on software that was not created by us. This possesses an issue because code may interact with other code causing bugs to happen. This is a low chance to happen but can happen if the languages that we use are updated and the syntax that we used are changed.

# 8. User Interface

## 8.1 Overview of User Interface

The User interface revolves around the use of the navbar to navigate the site and the various important pages it offers. The navbar itself presents few logical processes that need to occur in order to access elements of the UI. These logical processes are the use of the login functionality, which is presented on both the homepage and it’s own dedicated page. Logging in then allows the user to look at the projects that have been created by the account they logged into, and of course to log out (all of which are presented on the navbar). If the user hasn’t logged in the option to register is also on the Navbar.

The projects page then allows users to select any of the variety of projects they have created to modify them, delete them, or even create new ones. The projects are presented in the form of cards. Clicking on them will take them to the forms which to fill out. Each one of these forms has fields for the user to fill out, which can be traversed with the tabs seen at the top. There are a multitude of restrictions imposed on this area of the UI as some of the fields have ranges that they can accept, and selecting different options for certain fields will also change the tabs that they can access and must fill out.

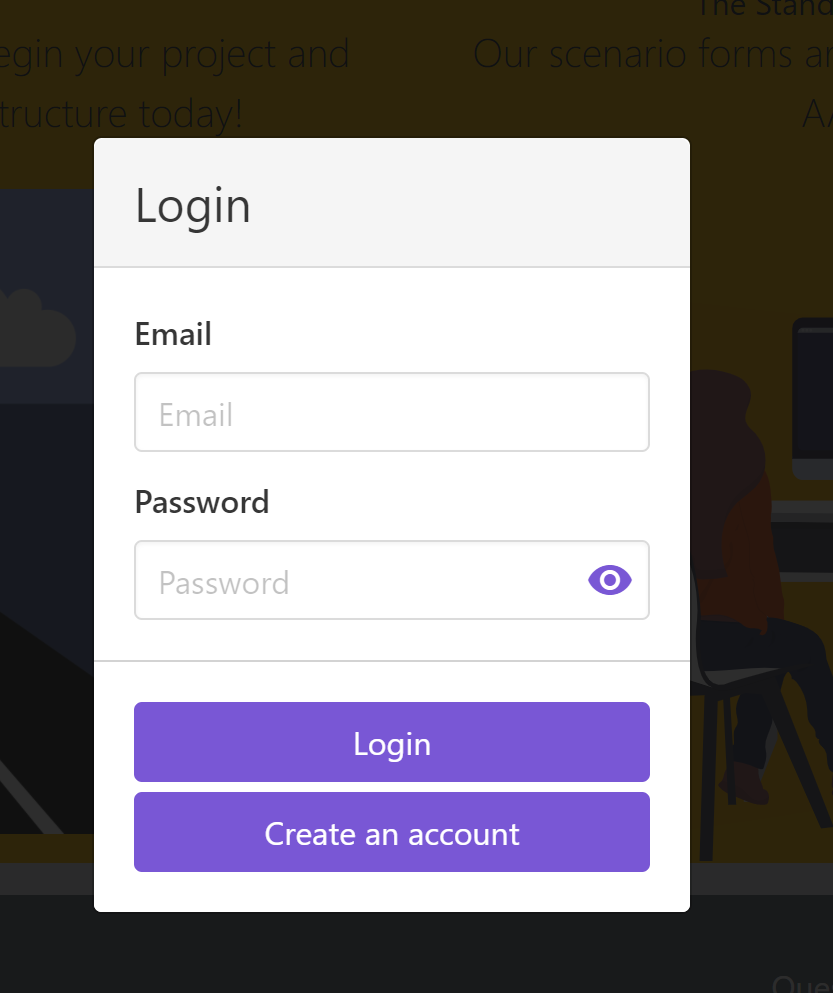
All the restrictions presented so far are requisite for the application to successfully complete its required purposes.

## 8.2 Screen Frameworks or Images

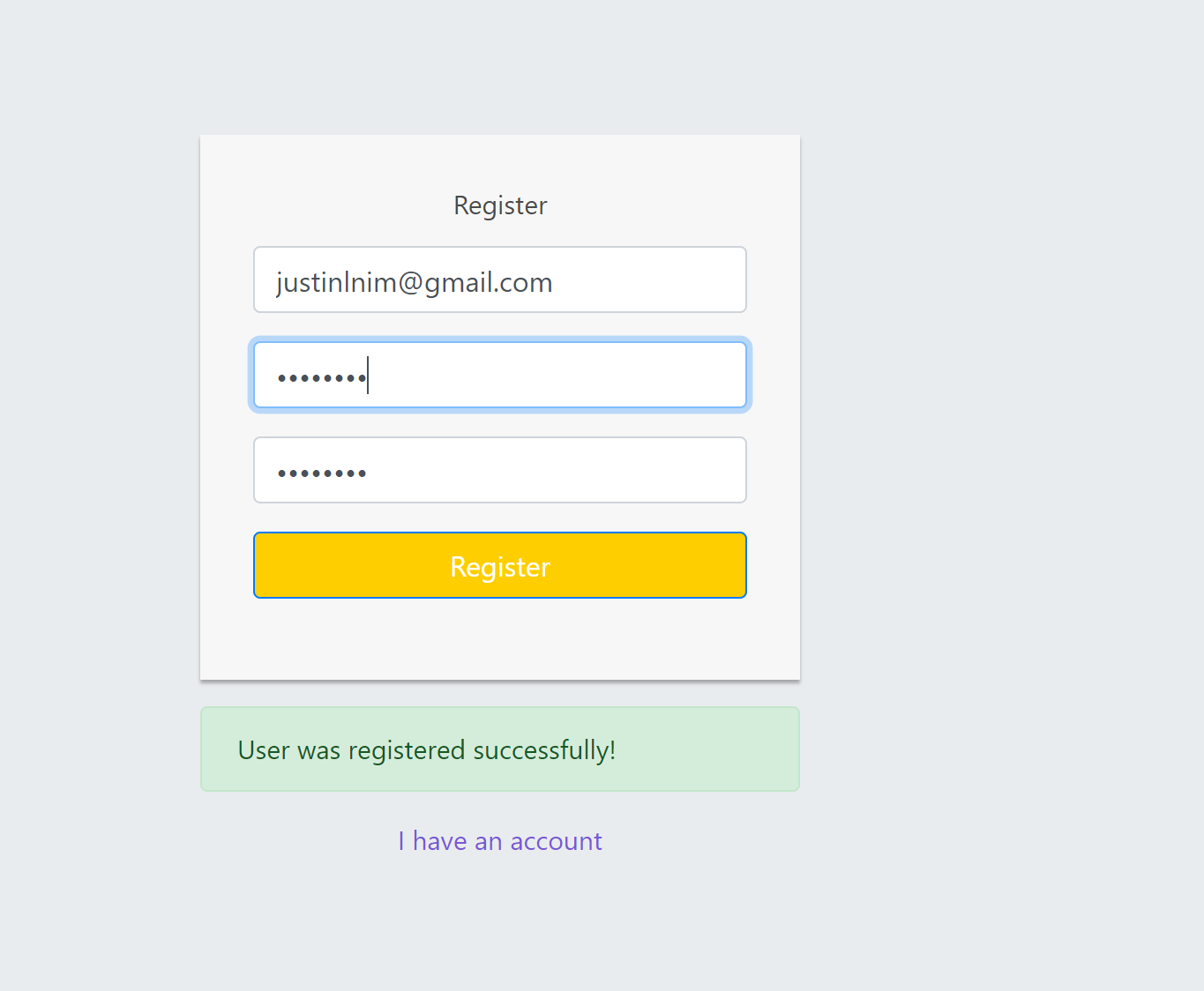
## Home page



## Login page



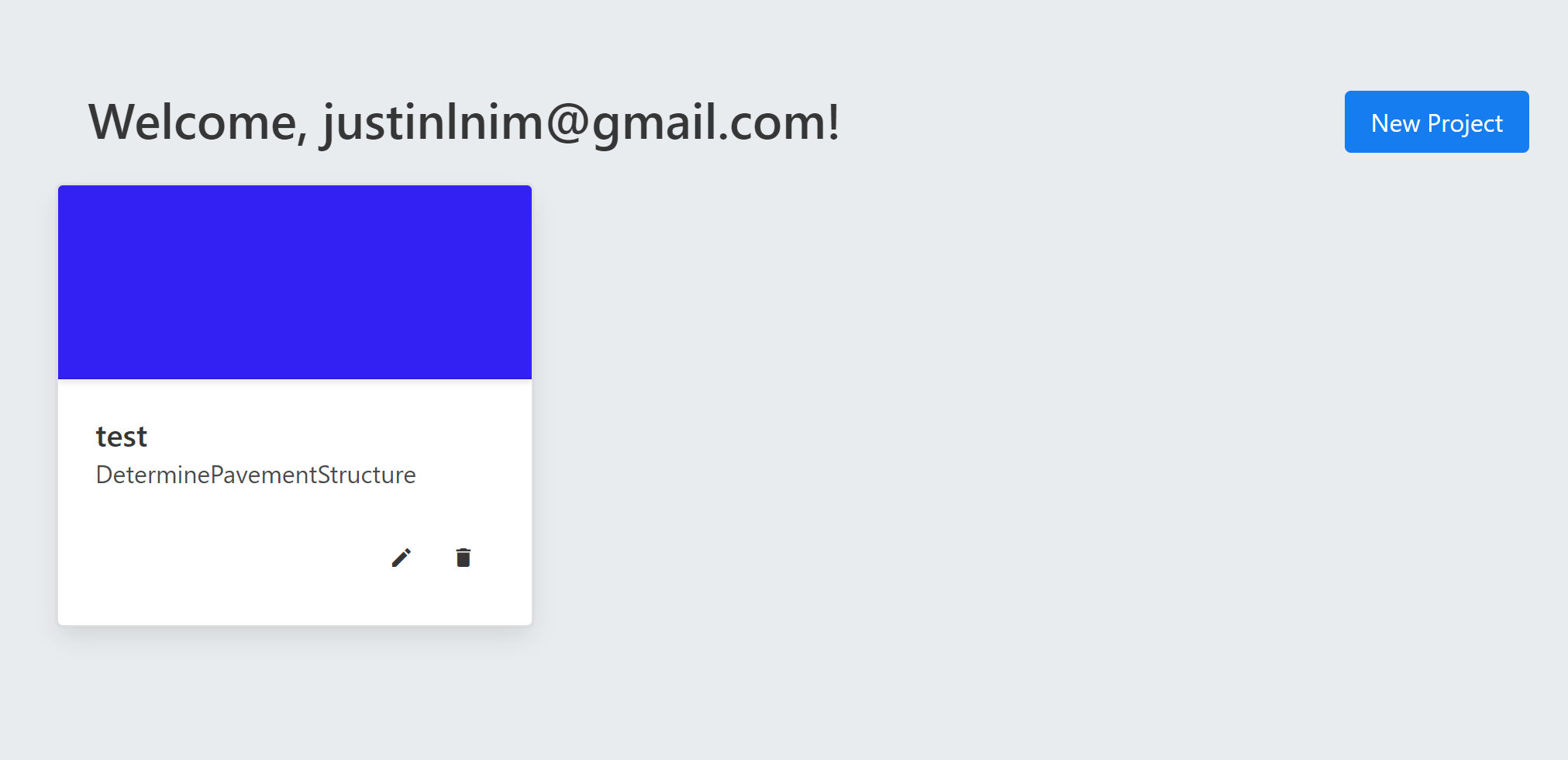
## Register page



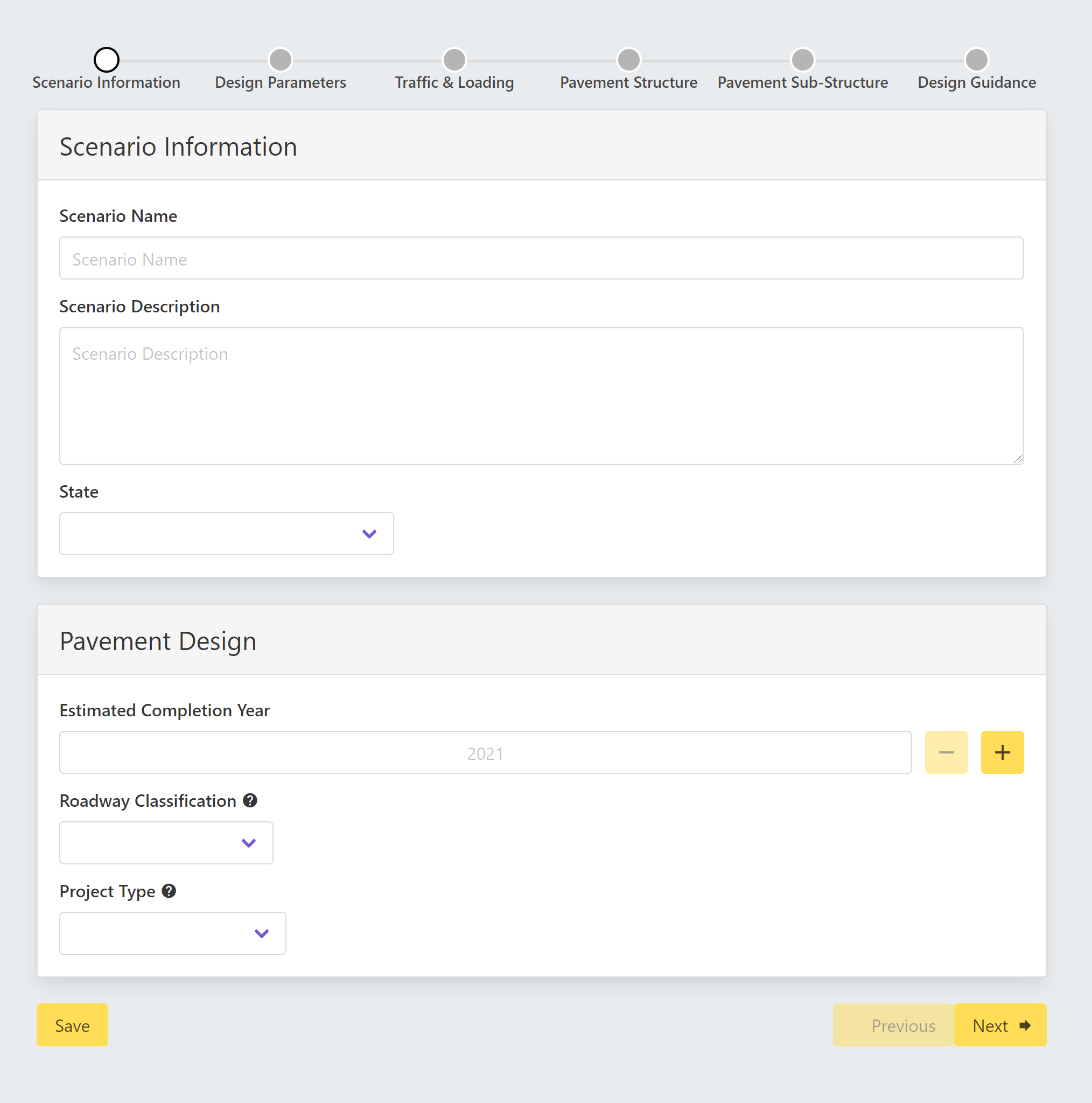
## Project selection page

## 

## Project page



## Example of form pages for the projects



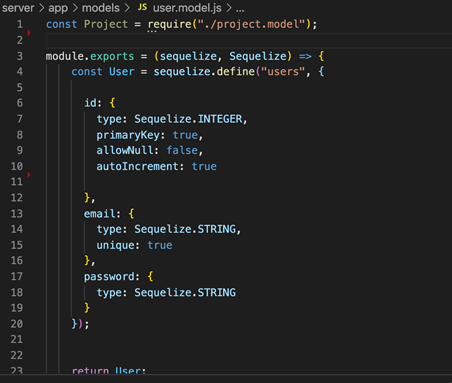
**8.3 User Interface Flow Model**

As mentioned previously, the login functionality gives the user access to the projects page along with the option to log out. It also removes the option to access the login page from the Navbar. Once selecting a project, there are multiple paths the user can take as each project can select up to 4 different scenario types, each of which have their own sets of forms that need to be fulfilled. Of course each of these forms also have their own sets of restrictions, and checkpoints that the user must fill out in order to progress through.

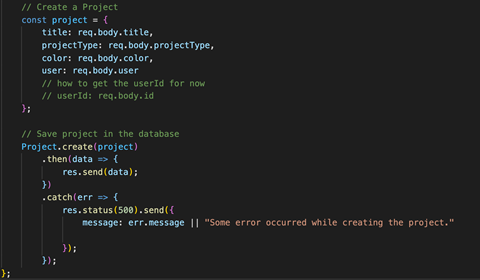
# 9. Database Design

MySQL was used for this project largely due to the relational structure between the users who will use the website and the project tables that will be saved under their accounts.

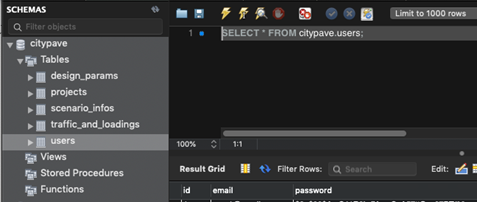
The object-relational mapper Sequelize was used as well to provide the code to build the tables using JavaScript conventional syntax without the need to write SQL style coding to minimize potential bugs. Sequelize was very useful in creating the “models” for which the tables represent such as “users” or “projects”.



Controllers were used to help create the tuples in the tables when users interact with the CityPave application. When the users fill out the forms and create a new account or start a new project, the website collects the information and stores them in the appropriate tables.



The CityPave web application uses Amazon Web Services to host its database. The database is called “citypave”.



# 10. Requirements Validation and Verification

1.1.1 User shall be able to create an account.

1.1.2 User shall be able to add information to the database

1.1.3 User must be able to login with account information

1.1.4.User should be able to validate user information

1.2.1 Number must be inputted

1.2.2 Numbers must be in range

1.2.3 Number data must be able to generate answers

1.2.4 Numbers can change measurement parameters

1.2.5 Calculations must be stored

1.3.1 Project information must be saved

1.3.2 Project must be able to modify information

1.3.3 Project must be able to delete information

1.4.1 Contacts must be able to fill out form

1.4.2 Contact information will be sent to Corresponding email address

1.5.1 Database shall store all user information

1.5.2 Database shall be able to modify stored information

1.5.3 Database shall be able to delete information.

# 11. Glossary

Data Type- Specified defined value

City Council- individuals who work for the city

Contractor- In who perform work based on contracts

Pavement- A paved surface that has been hardened

UX- User experience

Style guide- Color coordinated guidelines to follow

ADA- American disability act

AASHTO-American Association of State Highway and Transportation Officials.

**12. References**

|  |  |
| --- | --- |
| References | |
| Name of reference | Reason |
| Jungsoo Soo Lim Smart Pavement Project program | For Smart Pavement Project(used to create formula for calculations) |
| Vue.js | Creating the front end of the website |
| MySQL | Database for the website |
| CSULA style guide lines | Used to design correctly colored theme |