Midstate Shuttle Service

Team Revan

Nicolas Ross

Phoo Myint

Alexis Jacobson

Nathan Brock

[**Requirements**](#_3grusxe2opu1) **2**

[Overview](#_66p3rp46w2y4) 2

[Functionality](#_sj93588ph7xf) 2

[**Team**](#_5vs4mtkak9cy) **2**

[Rules](#_rdgkj41x8kl4) 2

[Code of Conduct](#_wjeo1aobzqx4) 3

[Coding Standards](#_x9rh9tnw1j3y) 3

[**Administration**](#_df0j1zismqhc) **4**

[Configuration](#_bvge1uv6odav) 4

[Logging](#_gc8ldzjshcoe) 4

[Continuous Integration and Deployment](#_o2hyrr405z1u) 4

[**Technologies**](#_axp8i6t90q7x) **5**

[**Architecture and Project Structure**](#_w9mm5szf8zhb) **5**

[Shuttle Service](#_33kns9x050ig) 5

[**ERD**](#_z91j033pvmzp) **6**

[RegisterModel](#_x8bi5v9urri) 6

[Location](#_7jcyaclxg73l) 6

[CheckIn](#_kgv64lex5mn9) 6

[Routes](#_i0lpi8pqmf1l) 7

[Bus](#_6qu3nt1n0jll) 7

[Driver](#_zgg39qw4n3xw) 7

[Feedback](#_m94uvjws2k66) 7

[Message](#_ol0jpjv6c9jn) 7

[**Wireframes**](#_8g9ftko89gfw) **8**

[Home Page](#_nvgsomsrsqee) 8

[Request Page](#_90x5zt75nl8u) 9

[Check In Page](#_hh0uka7ocutk) 10

[Message Page](#_a4yeuivr0mdt) 11

[Admin Dashboard](#_rvmx3k201xb5) 11

# Requirements

## Overview

Midstate Shuttle Service is an application created for Midstate and the Grant offices. This application, which began to make it easier for riders to register for rides, turned into a way to do that and help the admins and dispatch easily update routes and easily get data they needed for the grant.

The application currently has two types of users: riders(guests) and our admins(dispatch). Currently, riders only have the ability to see the main pages. Those pages include the home page, our registration page, check in page, and messages to dispatch page. Admins have access to those pages as well as a fully formed dashboard. In this dashboard they can add, edit, remove any items to the database, send messages to riders and get any data that they were to need.

## Functionality

Riders should be able to view all the base pages, they should be able to request rides and check-in. Finally, they should be able to message dispatch about any problems they are experiencing.

Admins should be able to sign in. After that they can have full control over all the aspects of the services. This is not limited to changing the route schedule, adding locations, and updating check ins.

# Team

## Rules

* Meeting Attendance
  + Meet once a week.
  + Virtual
  + Everyone should be there, outside of schedule conflicts.
* Handling Absences
  + Messages should be sent in the team channel in a timely manner.
* Communication
  + Utilize teams.
  + Be active.
    - If you have a question post it in the chat.
  + Dail Standups need to be done by 12 pm CST.

## Code of Conduct

* Respect
* Collaboration
* Open Communication
* Responsibility
* Inclusivity
* Adherence to Deadlines
* Professionalism
* Continuous Improvements

## Coding Standards

* Commenting and Documentation
  + Comment code when writing it.
  + If you comment something out, note that it needs to stay. If it doesn’t remove it
  + Any text documents need to be shared to the teams channel so everyone has those available
* Coding Style
  + Team found what worked best and we continued with that. Once one was decided upon we all pivoted to that
* Log Handling
  + Logging all forms.
* Error Handling
  + Don’t merge code to master that has errors
  + Comment out code that doesn’t work if we need to push your branch to main branch.
* Version Control Practices:
  + Use Git for all control systems.
  + Add messages to all commits. Describe what was done and any hiccups.
* Code Review Process
  + Add to teams channel to let everyone know it needs to be peer reviewed.
  + Check out branch, test, and make comments
  + Whoever’s branch it was then makes changes and merges

# Administration

## Configuration

Parts of the application were configured in the appsettings.json. This was used both for configuring the connection to database as well as configuring to the Entra Id portion of azure and our default directory.

## Logging

This Application currently logs whenever a form was being submitted. These logs can be found in wwwroot/logfiles. This is one area that I think we should have improved on. Logging is something, Nic, thinks he needs more time with on hope to properly utilize in a project.

## Continuous Integration and Deployment

GitHub is where the project is currently being kept. This is where all the branches are being stored and pushed to. Later the buildswere being deployed and published through Azure. This allowed us to properly deploy.

# Technologies

* ASP.NET Core MVC
* ASP.NET Identity
* Entity Framework Core
* Azure SQL Server
* SMTP2GO API
* FluentAPI

ASP.NET was the framework that the whole team was most familiar with, it was a relatively quick and simple decision. We used Identity early before having to switch to what we thought was SSO, then found that it was also the solution to use for that. Identity helped with the authorization and authentication for this whole solution

Entity Framework was another option that was chosen due to Nathan’s familiarity with it, he honestly handled 90 percent of that throughout the whole semester and helped with allowing that to work. SQL Server was our database of chose because of how well it integrated with Microsoft and Visual Studio, as well as the team being most comfortable with it.

Git was used for source control while GitHub was used to host the repository.

Login was handled through Microsoft Entra, this allowed us to utilize their separate login form to help with security of the application.

Finally, there were two Apis that were utilized throughout the project. SPTP2Go was utilized for sending emails to riders. FluentAPI was used in entity framework.

# Architecture and Project Structure

The main application can be found in ShuttleService/.

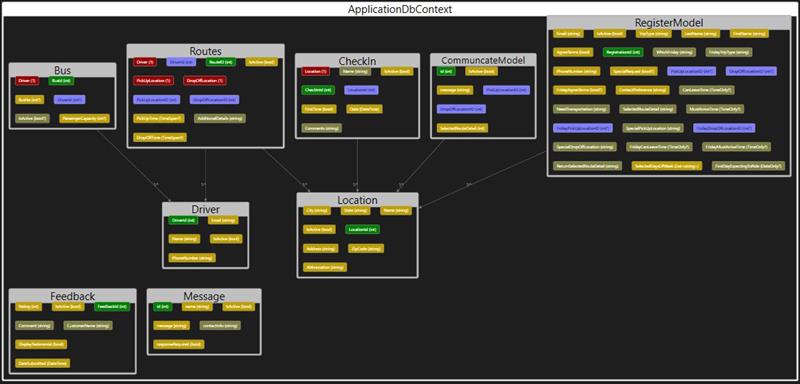
While Documentation can be found in Documentation/.

## Shuttle Service

We viewed this project as 2 separate applications: the one that the rider will see and be using, and the one that admins and dispatch will be using.

Utilizing MVC and Entity Framework helped us shape the structure into 3 layers. Views, Models, Controllers. In there we added both Services and Migrations. We believed this was the best way to keep everything separate and allowed us to keep everything together.

# ERD



## RegisterModel

This represents the riders signing up for the shuttle, there was so much that went into it that it is by far and away the biggest table used.

## Location

This represents the locations that the admins would want to have in creating routes.

## CheckIn

This shows rider check ins. This is the most important table in the entire application. The client needed this data to send back to WATEA for grant purposes.

## Routes

This is to hold current route schedule.

## Bus

Shuttles that are being used.

## Driver

Drivers that are driving, we kept this here because for future updates there is the potential of having drivers be able to log in so this would help bridge that gap.

## Feedback

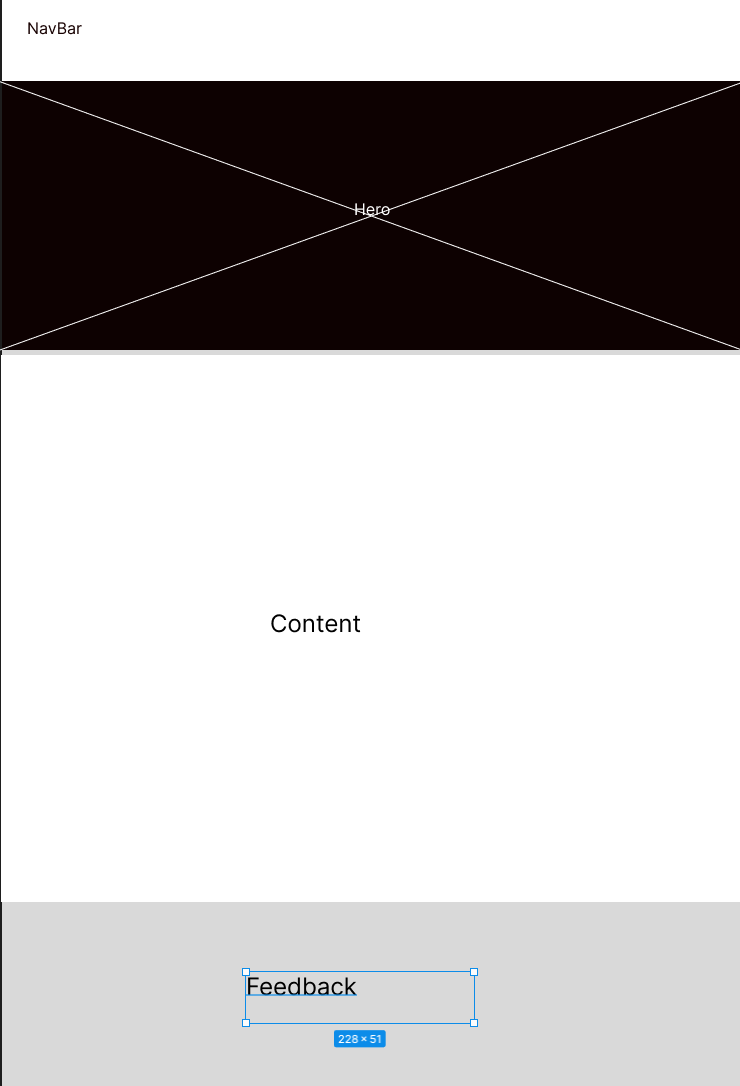
One of Two standalone tables. This was to allow us to store and show any feedback that was given from users.

## Message

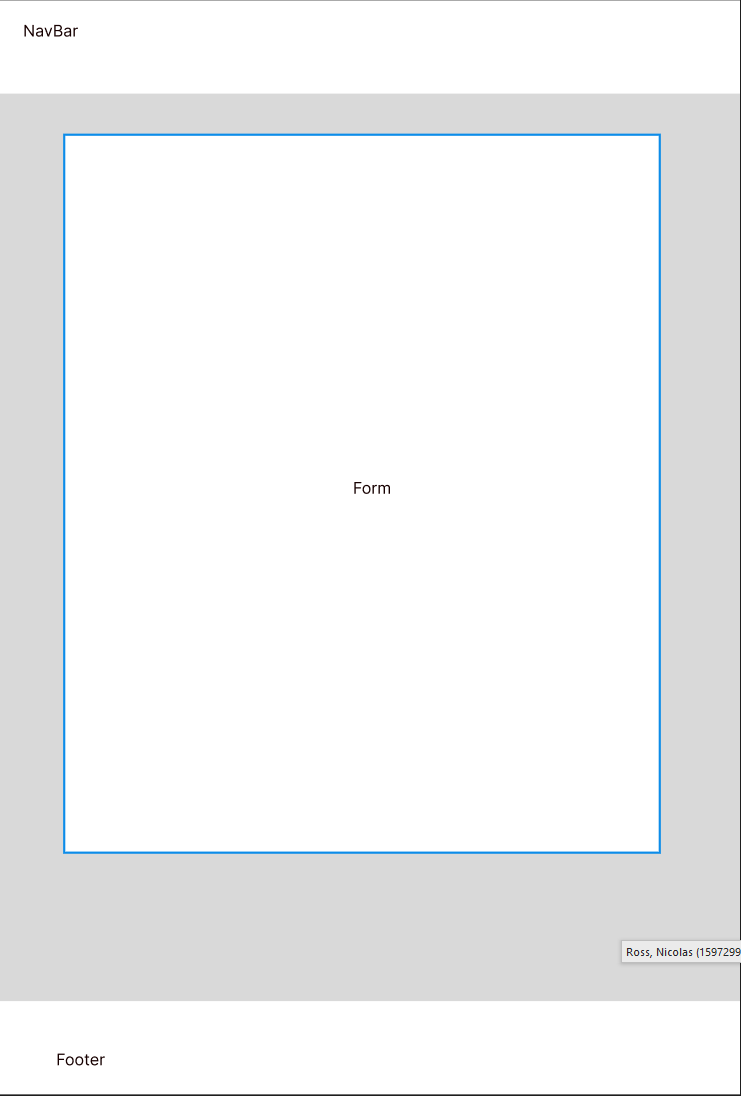
The second of the standalone tables, was for riders to send messages to dispatch and admins directly through the application.

# Wireframes

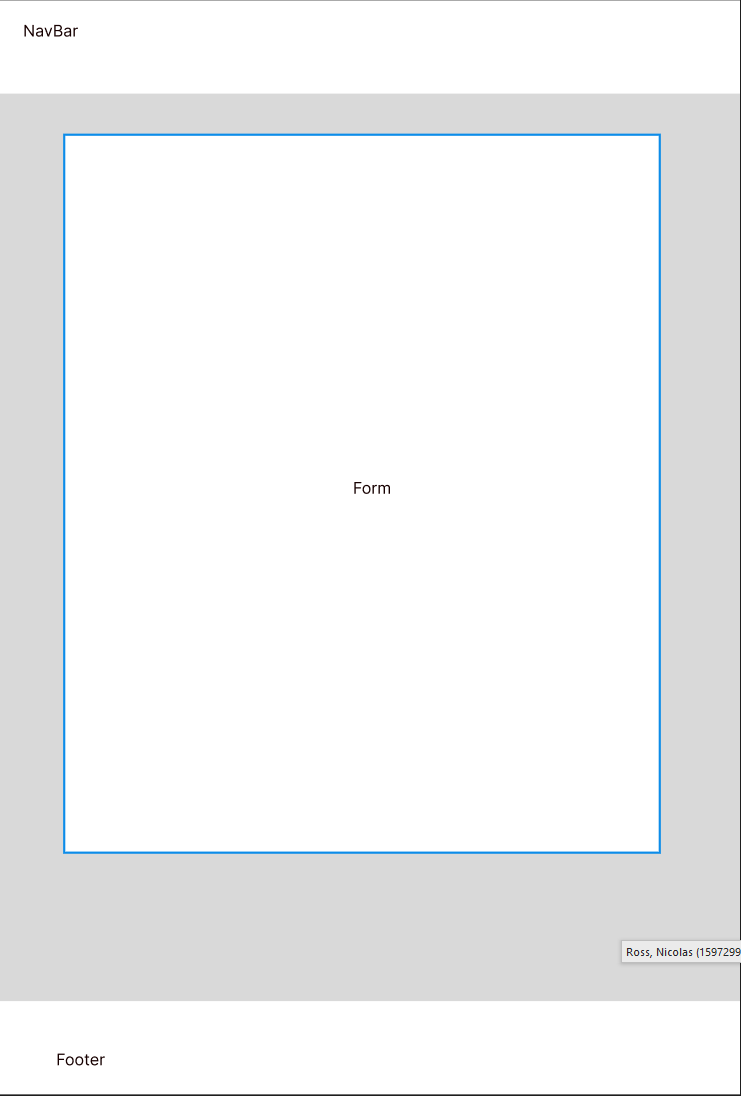
## Home Page



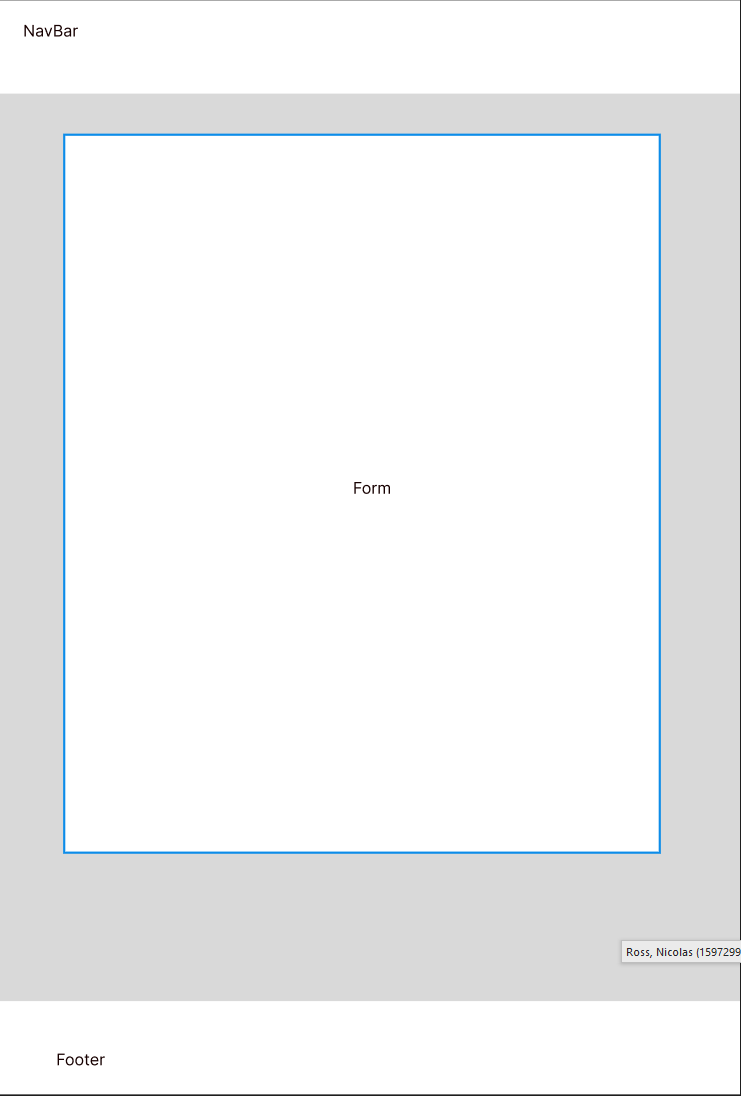
## Request Page



## Check In Page



## Message Page



## Admin Dashboard

