This application is written with the idea of an insurance company using SMS to communicate with customers. This version is at a Beta/Demo level. This means that it has the basic functionality in place. But there are a number of features that could be added, existing features that could be fleshed out and probably a number of edge cases that are not handled properly.

This guide will go through setting up and using it, familiarizing you with the features along the way.

This was written and run on a system using:

1. Visual Studio 2022
2. Microsoft SQL Server 2022 Developer
3. SQL Server Management Studio 20.1

It also is set to run on .Net 8

Once you clone the solution, open up SSMS (SQL Server Management Studio) and connect to local machine (or the server you want to run it on). Open up the SMSCommunicationDB.sql script in the Documents folder of the solution. Run the script. It will create the SMSCommunication database and insert the initial data needed to run the application.

A screenshot of a computer

Description automatically generated

There are four main projects in the solution.

Two apis:

1. BreeceWorks.CommunicationWebApi
2. BreeceWorks.SMSCoreWebApi

Two UIs:

1. BreeceWorks.CommunicationHub
2. BreeceWorks.DemoSMS

Each of these have appsettings.json files which have the following connection string:

"Server=BreeceWorksPC1;Database=SMSCommunication;Trusted\_Connection=True;MultipleActiveResultSets=true;encrypt=false;"

You will need to update to connect to your SQL Server. This will include changing the server’s name to your server’s name. And, if you choose to use a user name and password, you will need to replace the “Trusted\_Connection=True;” section with the user name and password.

Once the connection strings are updated, right click on the main solution file, then click on “Configure Startup Projects”. Select “Multiple startup projects” and set the four main projects to “Start” leaving the “BreeceWorks.Shared” library project to “None”.

A screenshot of a computer

Description automatically generated

Now you should be able to run the solution. You will see two UI pages and two Swagger pages for the APIs.

The heart of the solution are the two APIs. That’s where the main work has gone into. The two UI projects are there to demonstrate how the APIs can be utilized.

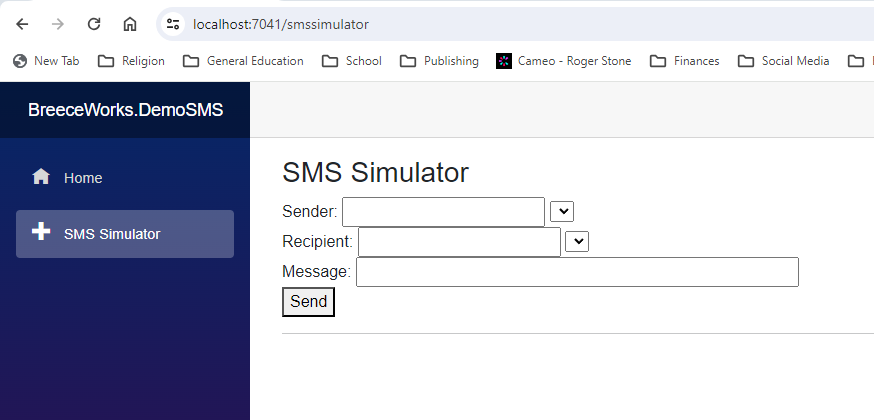
The BreeceWorks.CommunicationWebApi is where all of the business logic resides. When put into production, this would only be utilized within the network and would require authorization from anything trying to call it. I put in a simple form of authorization requiring an api key in the header of any requests. This can, of course, be changed to something more advanced and I probably will in later versions.

The BreeceWorks.SMSCoreWebApi is the publicly facing API. It calls your SMS service and provides webhooks for your SMS service. Currently, it has controllers interacting with the BreeceWorks.Demo project and Twillio. The Twilio webhooks in the controller are set up to use Twilio’s authentication methods to make sure any calls are actually coming from them. Other SMS services will have their own authentication methods you can use in the webhooks.

I stubbed in a simple example of adding an authorization handler to the non-webhook methods in the API. Currently, the Demo and Twilio SMS controllers send the attachment directly in the SMS message. You can set up the authorization handler for the SMSAttachment controller to use an IP address whitelist to only allow the SMS service(s) to access the AttachmentDownload method. You may choose to a link to the attachment instead. This would allow for maximum compatibility with the various phone service carriers. However, this means the AttachmentDownload method would have to be completely open to the public.

Since this is a document on the Beta/Demo version of this solution, it will primarily focus on walking through using the UIs and will only go into a few key parts of the APIs when they come up.

The first thing you will want to do after starting the solution is to go to the BreeceWorks.DemoSMS page and click on the SMS Simulator link. That way, it will begin showing messages received by the customer. It does not retrieve previously sent messages, but only displays messages it received after loading the page. It is kind of hacky in that it was thrown together with minimal design. It is simply there to see how everything works without having to actually send SMS messages.



Next you will need to go to the BreeceWorks.CommunicationHub page and click on the “Case Management” link. Then click on the “Open New Case” button.

A screenshot of a computer

Description automatically generated

A screenshot of a computer

Description automatically generated

If there are operators (agents) in the system, you can add one by clicking the “Add Primary Contact” button then selecting one from the drop down menu. Right now, you haven’t created any. But you can still create the case. You can add an operator later who can send messages to the customer, as well as assigning the case to different operators (agents).

Fill out the form and click on “Open New Case”.

It will take you to the Case Management page.

A screenshot of a computer

Description automatically generated

Back on the SMS Simulator, you will see the request SMS.

A screenshot of a computer

Description automatically generated

Type “Yes” in the Message text box and click “Send”.

A screenshot of a computer

Description automatically generated

Go back to BreeceWorks.CommunicationHub and click on the “Customer Management” link.

A close-up of a computer screen

Description automatically generated

Click on the “Communication” link, then click on the “Open Case Communication” link for the current case. It will load the message history for that case from the database.

A screenshot of a computer

Description automatically generated

A screenshot of a computer

Description automatically generated

Since there are no operators (agents) in the system, you will need to create one and add it to the case so they can send messages to the customer.

Click the “Operator Management” link and then click the “Create Operator” button.

A screenshot of a computer

Description automatically generated

A screenshot of a computer

Description automatically generated

Enter the required fields and click the “Submit” button.

Create another operator so there are multiple operators in the system.

Now, you can add an operator to the existing case as well as assign one from the beginning when you create new cases.

Click on the “Case Management” link, then click on the “Get Case Details” button. Then select an operator and click on the “Reassign Case” button to reassign it from no one to the selected operator.

A screenshot of a computer

Description automatically generated

A screenshot of a computer

Description automatically generated

If you scroll down the case details, you will now see “Primary Contact” information.

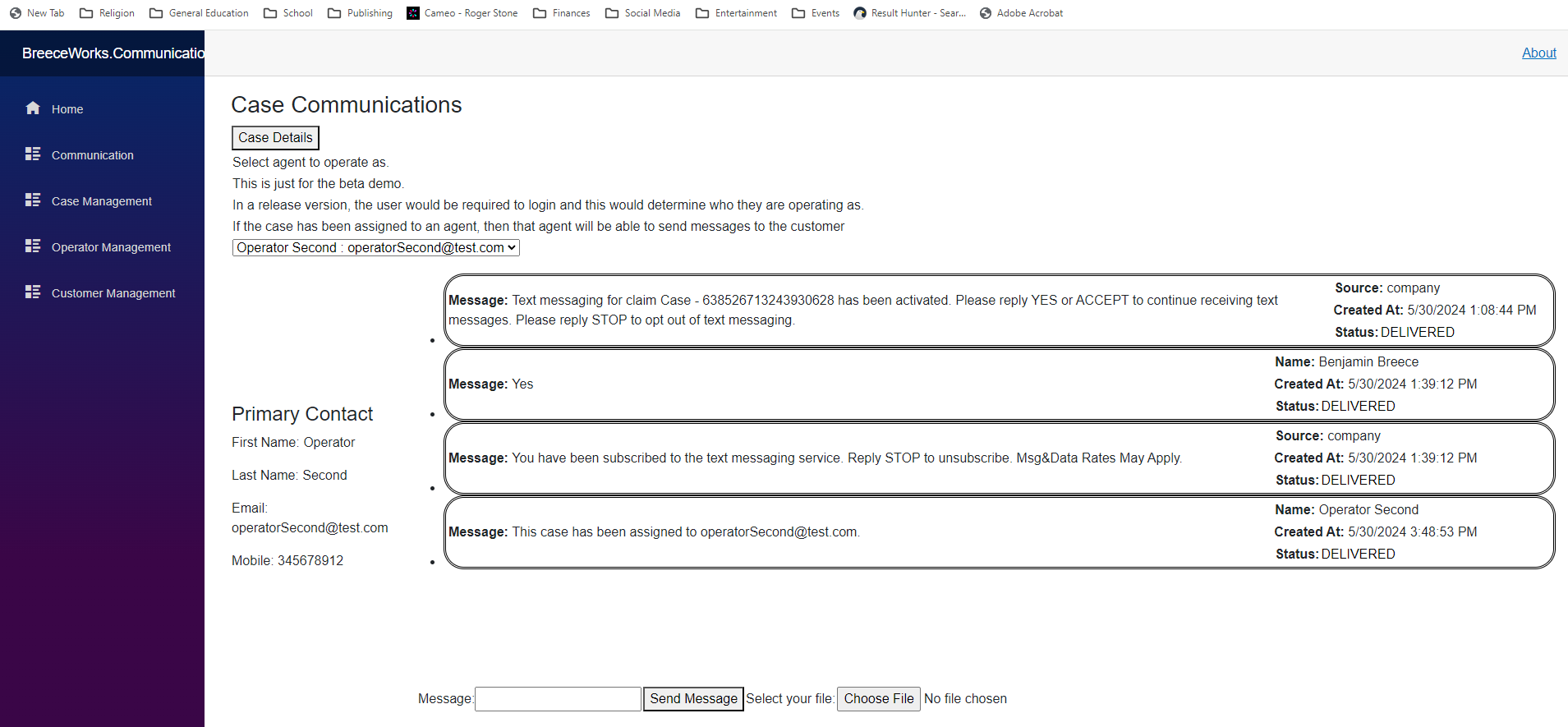
A screenshot of a contact page

Description automatically generated

Go back to the “Communication” link and click on the “Open Case Communication” button. You will now see a list of operators you can act as along with the “Primary Contact” information for that case.

The idea is that only the operator assigned to the case can send messages to the customer. Since this is a Beta, I have not implemented logging in functionality. When that is done, it will automatically display the message functionality if you are logged in as the assigned operator. Otherwise, it will hide that functionality.

For now, select the operator from the dropdown menu that matches the one assigned.



Type a message and select a file to send as an attachment. Then click “Send Message”

The new message will appear on the communication list.

A screenshot of a computer

Description automatically generated

Go back to the BreeceWorks.DemoSMS window and you will see the new message there as well.

A screenshot of a computer

Description automatically generated

When you create new cases, you will be able to assign operators at creation and the same or different customers based on the email and mobile number you put in for them. All of the conversations will show up in the SMS Simulator as long as you keep the page open (the next version will allow for loading existing conversations from the database when you go to the page, like the Communication Hub. When sending a message from the SMS Simulator, you select the sender (which customer phone the message is coming from) and the recipient (which of the company phone numbers it is going to) from the dropdown lists.

A screenshot of a computer

Description automatically generated

The project was written with the idea of utilizing the Twilio SMS service.

When you are ready to begin using the actual SMS service, you can deactivate the DemoSMS numbers by running the Set Demo Phone Numbers Inactive.sql script.

Then you fill out and run the Set Twilio Values.sql script. You will also need to set up your Incoming webhook in your Twilio account. Naturally, in order for Twilio to call your webhook, it will need to be publicly accessible. You will need to deploy it to a server for long term use. If you don’t have a server set up during Dev, you can use Ngrok (<https://ngrok.com/> ) to create a temporary publicly accessible endpoint for your api while running it on your local machine.

If you want to switch to another SMS service, you will just need to add another controller that implements the ISMSController interface to the SMSCoreWebApi. Then you can add the new numbers and config values using database scripts. Then set the IsActive flag for the old numbers to false. The IsActive flag is only utilized when assigning a company number to a new case. This allows for a smoother transition. Customers with open cases will still be able to continue to send texts, while new cases will be assigned numbers from the new service. Once there are no more open cases with the old numbers, you can terminate the old SMS service without causing any hardship to the customers.

I linked my phone to my PC so I can take screenshots of the SMS messages received on my phone and then ran the scripts to deactivate the DemoSMS numbers and set up the number from my Twilio account.

I then opened a new case. This screenshot shows both the messages on my phone and the CommunicationHub side by side.

A screenshot of a computer

Description automatically generated

A screenshot of a computer

Description automatically generated