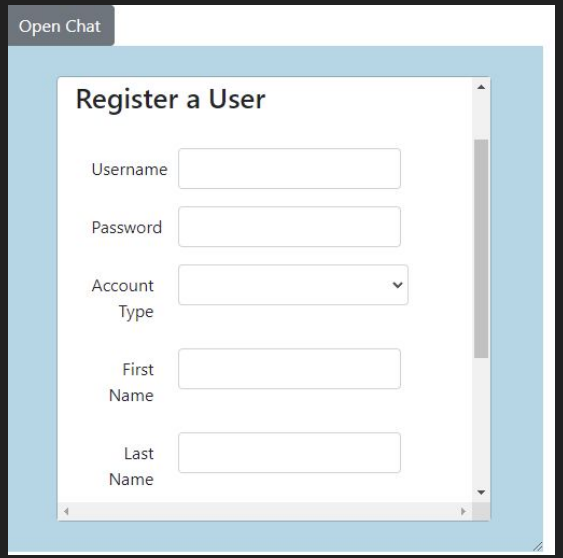
# Secure Chat

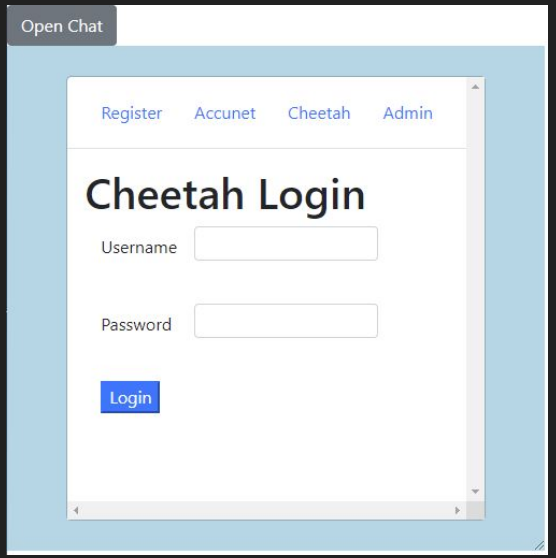
Ben Eger, Ethan Holland, Mercedes Thompson, Nicole Gerber

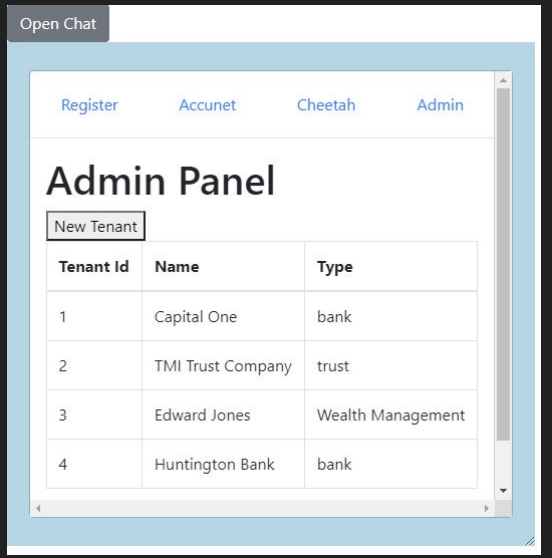
# New Functionality

- Resources are protected by JWT Token
  - Tokens are managed by axios
  - Passed through header securely
- User Registration
  - Form to register new users w/ information
- User Login
  - Authenticated login pages for each client
- Admin Panel
  - Manage Tenants
    - Create
    - Delete

# Aesthetics

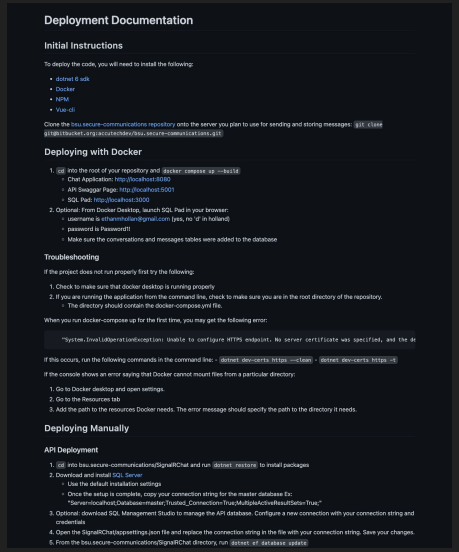



- Added global rules for components and pages to follow

# Documentation



## Deployment Documentation

### Initial Instructions

To deploy the code, you will need to install the following:

- dotnet 8 sdk
- Docker
- NPM
- Vue-cli

Clone the bsu.secure-communications repository onto the server you plan to use for sending and storing messages: `git clone git@bitbucket.org:accutechdev/bsu.secure-communications.git`

### Deploying with Docker

1. `cd` into the root of your repository and `docker compose up --build`
   - Chat Application: http://localhost:8080
   - API Swagger Page: http://localhost:5001
   - SQL Pad: http://localhost:3000
2. Optional- From Docker Desktop, launch SQL Pad in your browser:
   - username is ethanmhollan@gmail.com (yes, no 'd' in holland)
   - password is Password1!
   - Make sure the conversations and messages tables were added to the database

### Troubleshooting

If the project does not run properly first try the following:

1. Check to make sure that docker desktop is running properly
2. If you are running the application from the command line, check to make sure you are in the root directory of the repository.
   - The directory should contain the docker-compose.yml file.

When you run docker-compose up for the first time, you may get the following error:

```
"System.InvalidOperationException: Unable to configure HTTPS endpoint. No server certificate was specified, and the de
```

If this occurs, run the following commands in the command line: - `dotnet dev-certs https --clean` - `dotnet dev-certs https -t`

If the console shows an error saying that Docker cannot mount files from a particular directory:

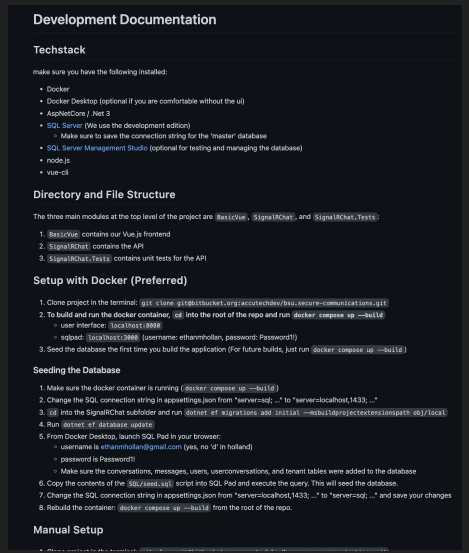
1. Go to Docker desktop and open settings.
2. Go to the Resources tab
3. Add the path to the resources Docker needs. The error message should specify the path to the directory it needs.

### Deploying Manually

#### API Deployment

1. `cd` into bsu.secure-communications/SignalRChat and run `dotnet restore` to install packages
2. Download and install SQL Server
   - Use the default installation settings
   - Once the setup is complete, copy your connection string for the master database Ex: "Server=localhost;Database=master;Trusted_Connection=True;MultipleActiveResultSets=True;"
3. Optional- download SQL Management Studio to manage the API database. Configure a new connection with your connection string and credentials
4. Open the SignalRChat/appsettings.json file and replace the connection string in the file with your connection string. Save your changes.
5. From the bsu.secure-communications/SignalRChat directory, run `dotnet ef database update`

Deployment



## Development Documentation

### Techstack

make sure you have the following installed:

- Docker
- Docker Desktop (optional if you are comfortable without the ui)
- AspNetCore / .Net 3
- SQL Server (We use the development edition)
  - Make sure to save the connection string for the 'master' database
- SQL Server Management Studio (optional for testing and managing the database)
- node.js
- vue-cli

### Directory and File Structure

The three main modules at the top level of the project are `BasicVue`, `SignalRChat`, and `SignalRChat.Tests`:

1. `BasicVue` contains our Vue.js frontend
2. `SignalRChat` contains the API
3. `SignalRChat.Tests` contains unit tests for the API

### Setup with Docker (Preferred)

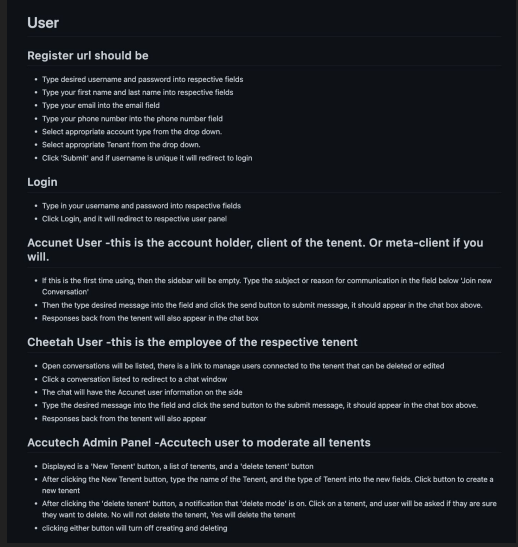
1. Clone project in the terminal: `git clone git@bitbucket.org:accutechdev/bsu.secure-communications.git`
2. **To build and run the docker container, `cd` into the root of the repo and run `docker compose up --build`**
   - user interface: `localhost:8080`
   - sqlpad: `localhost:3000` (username: ethanmhollan, password: Password1!)
3. Seed the database the first time you build the application (For future builds, just run `docker compose up --build`)

### Seeding the Database

1. Make sure the docker container is running (`docker compose up --build`)
2. Change the SQL connection string in appsettings.json from "server=sql; ..." to "server=localhost,1433; ..."
3. `cd` into the SignalRChat subfolder and run `dotnet ef migrations add initial --msbuildprojectextensionspath obj/local`
4. Run `dotnet ef database update`
5. From Docker Desktop, launch SQL Pad in your browser:
   - username is ethanmhollan@gmail.com (yes, no 'd' in holland)
   - password is Password1!
   - Make sure the conversations, messages, users, userconversations, and tenant tables were added to the database
6. Copy the contents of the `SQL/seed.sql` script into SQL Pad and execute the query. This will seed the database.
7. Change the SQL connection string in appsettings.json from "server=localhost,1433; ..." to "server=sql; ..." and save your changes
8. Rebuild the container: `docker compose up --build` from the root of the repo.

### Manual Setup

Development



## User

### Register url should be

- Type desired username and password into respective fields
- Type your first name and last name into respective fields
- Type your email into the email field
- Type your phone number into the phone number field
- Select appropriate account type from the drop down.
- Select appropriate Tenant from the drop down.
- Click 'Submit' and if username is unique it will redirect to login

### Login

- Type in your username and password into respective fields
- Click Login, and it will redirect to respective user panel

### Accunet User -this is the account holder, client of the tenent. Or meta-client if you will.

- If this is the first time using, then the sidebar will be empty. Type the subject or reason for communication in the field below 'Join new Conversation'
- Then the type desired message into the field and click the send button to submit message, it should appear in the chat box above.
- Responses back from the tenent will also appear in the chat box

### Cheetah User -this is the employee of the respective tenent

- Open conversations will be listed, there is a link to manage users connected to the tenent that can be deleted or edited
- Click a conversation listed to redirect to a chat window
- The chat will have the Accunet user information on the side
- Type the desired message into the field and click the send button to the submit message, it should appear in the chat box above.
- Responses back from the tenent will also appear

### Accutech Admin Panel -Accutech user to moderate all tenents

- Displayed is a 'New Tenent' button, a list of tenents, and a 'delete tenent' button
- After clicking the New Tenent button, type the name of the Tenent, and the type of Tenent into the new fields. Click button to create a new tenent
- After clicking the 'delete tenent' button, a notification that 'delete mode' is on. Click on a tenent, and user will be asked if thay are sure they want to delete. No will not delete the tenent, Yes will delete the tenent
- clicking either button will turn off creating and deleting

User

# Mentor Feedback

- Kade recommended the use of JWT tokens and axios to keep the front end and data secure
  - Axios allows us to use the JWT token in the frontend without the risk of an insecurity

- Said to automatically run seed the database so we don’t have to manually run migrations
  - It’s been added to update the database on startup

# Client Feedback

- Scott said that he really like what we have so far and is thinking of all kinds of ways to implement it into their systems

- Scott also said there probably won’t be someone monitoring the chat all day, so an sms/email notification that a message has been received would be a good feature to have

# Next Iteration

- Cheetah Panel
  - Manage open conversations
  - Reply to users by opening their conversation
  - Close conversations after their complete
  - Filters open and closed tickets
- Testing
- SMS/Email alert that a message has been received
- Polishing
  - Bug-fixing
  - Refactoring
  - Messages displayed with timestamp
  - Aesthetic fixes

# Random Slide