SaveFarmer - Dev doc

Tools Used

- *Visual Studio Code*
- *Ionic*
- *Vue*
- *TypeScript*
- *Visual Studio*
- *Azure*
- *Android Studio*
- *Xcode*

Development Installation Instructions

1. ***Install Node.js*** *Download here*
2. ***Install Android Studio*** *Download here* *(Optional, for running an Android emulator)*
3. ***Install XCode*** *Download here* *(Optional, for running iOS emulator. Required a Mac)*

Command Line Setup

1. *Open the command prompt and run the following commands:*

    - ***Install Yarn:*** *npm install --global yarn*
    - ***Install Vue****: yarn global add @vue/cli*
    - ***Install Ionic and Capacitor****: yarn global add -g @ionic/cli @capacitor/assets*
2. *Navigate to the desired directory:*

    - ***Clone the repository****: git clone https://github.com/Sanmeet-EWU/cscd-488-490-project-save-farmers/*
    - ***Change to the project directory source/savefarmer and run the app****:*
      cd savefarmer && yarn install && ionic serve
      (This command should open the app in your browser. If it doesn't, ensure Vue and Ionic are installed correctly.)

Code Editing

Open the ***entire folder*** *in Visual Studio Code. Ensure you have the Vue, TypeScript, and Ionic extensions installed to edit the code within the* `src` *folder.*

Third-Party Plugins

The following plugins are required to run the app:

- ***Chart.js****: npm install chart.js*
- ***chartjs-plugin-datalabels****: npm install chartjs-plugin-datalabels*
  OR yarn add chartjs-plugin-datalabels

Emulator Instructions

Android

In the project terminal, run the following commands:
Ensure Android Studio is installed and open during the first run to build properly.

- ***Add Capacitor Android files****: ionic cap add android*
- ***Build Android****: ionic cap build android*
  (This should open Android Studio and build the emulator)
- ***Run the Emulator****: ionic cap run*
  (This should launch the emulator. Note: It may take some time.)

iOS

In the project terminal, run the following commands:
Ensure you are using a Mac and have Xcode installed.

- ***Add Capacitor IOS files****: ionic cap add ios*
- ***Build IOS****: ionic cap build ios*
  (This will build the ios model but will fail doing a capacitor sync, this is okay.)
- ***Open Xcode****: open existing project from:*
  .cscd-488-490-project-save-farmers/Source/savefarmer/ios/App
  (This will open the project in Xcode)
- ***Build:*** *Select a device to build onto and click the play button to build. (This could build the app on the desired device)*

Frontend Structure

`router` folder: Contains `index.ts`, which configures the app's page routing.
`script` folder: Contains `PostService.ts` and `UserPost.ts`, which handle most of the post functionality and user post settings.
`App.vue` (root file): Located outside all folders. It includes the hamburger menu, toolbar, and button navigation paths.

`main.ts`: Contains the app's entry point and essential configurations.
`page` folder: Contains the website's individual pages.
`component` folder: Contains modular components to reduce page size and improve reusability.

Database setup

*Navigate to the **backend directory**: `Source/FarmerAPI`.*

Open the `FarmerAPI.sln` file in Visual Studio (if installed).

- *This will load the project with the correct setup in Visual Studio.*

For the initial run, execute the following commands to generate the database locally:

```
dotnet ef migrations add InitialCreate
```

```
dotnet ef database update
```

AuthController API Endpoints

- POST `/api/register` - Register a new user (Anonymous)
- POST `/api/login` - Login (Anonymous)
- GET `/api/user/{id}` - Get user by ID (Authorized)
- POST `/api/refresh-token` - Refresh authentication token (Anonymous)
- POST `/api/revoke-refresh-token` - Revoke a refresh token (Authorized)
- GET `/api/current-user` - Get details of the currently authenticated user (Authorized)
- DELETE `/api/user/{id}` - Delete a user by ID (Authorized)
- GET `/api/GetAllUsers` - Retrieve all users (Authorized)
- POST `/api/change-password` - Change user password (Authorized)

PostController API Endpoints

- POST `/api/posts` - Create a new post (Authorized)
- GET `/api/posts` - Get all posts (Anonymous)
- GET `/api/posts/analytic` - Get post analytics (Authorized)
- GET `/api/posts/user/{userId}` - Get all posts by a specific user ID (Authorized)
- PUT `/api/posts/{postId}` - Update a post by ID (Authorized)
- DELETE `/api/posts/{postId}` - Delete a post by ID (Authorized)

Backend Structure

***`Controller`* folder:** *Contains API controllers, named according to the endpoints that the frontend will call.*

***`Domain/Entities`* folder:** *Contains database entity definitions.*

***`Domain/Contracts`* folder:** *Contains database request and response models used for communication between the database and the frontend.*

***`Extensions`* folder:** *Contains critical settings for **JWT**, error handling, and the **ExceptionHandler**.*

***`Infrastructure/Context`* folder:** *Defines how `.NET` will generate the database.*

***`Infrastructure/Mapping`* folder:** *Handles automated mapping between response models and database entities.*

***`Service`* folder:** *Contains most of the database functionality and logic.*

***`appsettings.json`* file:** *Stores settings for both deployment and the local database.*

***`Program.cs`* file:** *Configures the application's build settings and defines how the API is constructed.*

Frontend Deployment - Web

Assuming you have a properly set up Azure account with an active subscription, follow these steps:

1. Fork the project to your personal GitHub repository.
2. Navigate to your Azure main menu and click on **Create a resource**.
3. Search for **Static Web App** and click **Create**.
4. In the **Deployment details** section, select **GitHub** and link it to your account.
5. For **Organization**, select your account name and choose the correct repository for the project.
6. For **Build presets**, select **Vue.js**.
7. In the **App location** field, enter: `./Source/savefarmer`.
8. In **Deployment configuration**, select **GitHub**.
9. Click **Create**.

This process should automatically generate a GitHub workflow and deploy your web app to the Azure-provided link.

Backend Deployment - API

Setting up the API in Azure

I recommend following this video for guidance, then proceed to steps 27-30 as they are not covered in the video.

Prerequisites:

- Ensure you have an Azure account.
- Be logged into Visual Studio.

Steps to Deploy the API

1. **Open the API Project** in Visual Studio.
2. In Visual Studio, **right-click** your API project and select **Publish**.
3. Select **Azure App Service** as the target. For this project, choose **Azure App Service (Windows)**.
4. Click **Next**.
5. Select **Create a new app service** and click **Next**.
6. Select **Skip this step** for the API setup.
7. Click **Finish**.

SQL Database Configuration

1. On the **Publish** page, scroll down to the **SQL Server Database** section.
2. Click the **three dots** (•••) and select **Connect**.
3. Select **Azure SQL Database** and click **Next**.
4. Create a new database.
5. When prompted, create a **username** and **password**—remember these for later use.
6. Click **Next**, change the **Connection String Name** to `"ConnectionString:DefaultConnection"`, and click **Next** followed by **Finish**.
7. Now, click **Publish**.

Configuring Azure Portal

1. Go to the **Azure Portal** and navigate to your **SQL Server**.
2. In **Settings**, go to **Entra ID** (formerly **Nucriseift**).
3. Click **Set Admin**, find yourself, and add yourself as an admin.

SQL Database Permissions

1. Go to the **SQL Database** in Azure and open the **Query Editor**.
2. **Allow your IP address** and log in as the admin.
3. Run the following SQL commands in the query editor:

```
CREATE USER [YourAppServiceName] FROM EXTERNAL PROVIDER;
ALTER ROLE db_owner ADD MEMBER [YourAppServiceName];
```

4. Click **Run**.

Environment Variable Configuration

1. Navigate to your **Web App** in Azure.
2. Go to **Settings** > **Environment Variables**.
3. Click **Add** and enter:
   - **Name:** `JwtSetting__Key`
   - **Value:** `ThisIsA32CharactersLongSecretKeyEXL`
4. Click **Apply**.

Final Step

- Wait about **5 minutes**, then visit your **App Service link**. Your API endpoint should now be working.

Known bugs and future enhancements

Known Bugs and Issues

- **User Deletion Issue:** When deleting a user, the associated posts linked to that user's IP are not deleted.
- **Cypress Deprecation:** Cypress is deprecated as it interferes with the front-end application, and we have decided not to address this issue.

Future Enhancements

- **Transaction System:** Implement a secure and efficient transaction system.
- **Chat System:** Develop a real-time chat system for improved user interaction.
- **Loading Page:** Add a loading page between API requests for a better user experience.
- **Add Testing:** We don’t have any real testing happening in the backend, but for future work, it would be nice to have the test always there.