System Guide

CS451R - Angry Nerds, Commerce Bank Project Application

Fall 2021

Version 0.0.1 - Last Updated Dec 8, 2021

[Introduction](#_6n5y8z95q160)

[Getting Started](#_1ra87jezeuo5)

[Technology Used and Development Environment](#_2s2jf41xhtxp)

[Python Virtual Environment](#_5tehlisvqo8d)

[Node.JS](#_rvoi7jsycuk7)

[MySQL Development Database](#_lym4fits43p4)

[App Settings](#_xt1sjc3rzfve)

[Tour of Project](#_yrx29y5scfl4)

[Database](#_mccfm68dhz4r)

[Backend](#_8hlsb1onbcu2)

[Frontend](#_67xbp894d6us)

[Testing](#_j66udjyn8ayx)

[Deployment Procedure](#_io6h4wbpllrg)

# 

# Introduction

This is the system guide for the online banking application built by the Angry Nerds team. This application is a web site that allows customers to view and interact with their banking information. In order to provide future maintenance and development, this development quick start guide is created. It will cover the environments and tools used, as well as provide context to the sections of the program.

# Getting Started

Quick start of development servers, after tools are installed and virtual environments created:

* Make sure that the database is up and running.
* Build and run the backend in your IDE.
* On your terminal of choice run ‘npm start’ inside of the Frontend directory.

# Technology Used and Development Environment

This project uses:

* Python - Version >= 3.8
* C# ASP.NET Core 3 (Can be installed with Visual Studio)
* Node.JS
* MySQL Database (should set up locally for development)
* Visual Studio 2019 (Recommended IDE for C#)
* IDE of choice for Node.JS
* If on windows, Git for Windows (with Git Bash); otherwise, just Git CLI tool and terminal

Git Bash and Visual Studio are simple downloads and installs of the latest compatible version.

Usage of Visual Studio is recommended for the C# project since both the language and IDE are developed together by Microsoft, and the IDE will automatically install project dependencies based on the solution file (.sln).

Python 3.7 has a security issue that was fixed with 3.8, but Python 3.8 is no longer offered for support. Python 3.9 is recommended, but most versions will work. If installing multiple versions of python it is recommended to remove them from your PATH environment variable and use aliases in your ~/.bash\_profile.

## Python Virtual Environment

This guide uses venv, which is standard in python, but any virtual project management method works.

1. Install python version from list of [releases](https://www.python.org/downloads/)
   * You can either add the new installation of python to your PATH variable (an option of the installer)
   * OR not add it to PATH but provide an alias instead (useful for having multiple python versions). Using git bash (for non-windows users, these commands work in terminal without the use of winpty), you can create an alias (for git bash and python 3.9) by adding the following line to ~/.bash\_profile: alias python39="winpty ~/AppData/Local/Programs/Python39/python.exe".
2. Verify installation: python --version
3. Create virtual environment in the folder .venv by running this command in the root of this project directory: python -m venv .venv
4. Activate virtual environment
   * In Git Bash, MacOS, or Linux: source .venv/Scripts/activate
   * In Windows command prompt: ".venv/Scripts/activate.bat"
5. Verify you are in the virtual environment. There should be a (.venv) printed before each command line.
6. (Optional) Run which python or the windows equivalent to see if the python executable points to your virtual environment.
7. In your virtual environment, install your packages: python -m pip install -r requirements.txt
8. Verify the installed packages: python -m pip freeze

## Node.JS

Node.JS is used to develop the frontend portion of the website. Installation of Node.JS can be done by following their instructions. In the project root, running "npm install" will add the required project dependencies locally.

## MySQL Development Database

A local database is used to manage the data used by the tests and project website. Download MySQL client and server and follow the installation steps. You may want to set up a user account with permissions to mirror application (non-root) access.

## App Settings

The file appsettings.Development.json should be added to theBackend folder. It provides confidential information about passwords or secrets for managing JWT tokens and Database Access. It should look like this:

{

"AppSettings": {

"Secret": "THIS IS USED TO SIGN AND VERIFY JWT TOKENS, REPLACE IT WITH YOUR OWN SECRET IT CAN BE ANY STRING"

},

"Logging": {

"LogLevel": {

"Default": "Information",

"Microsoft": "Warning",

"Microsoft.Hosting.Lifetime": "Information"

}

},

"ConnectionStrings": {

"localDB": "server=localhost;database=banking;uid=USERNAME;pwd=PASSWORD;"

}

}

Under the Connection Strings section, the local database username and password should be changed to be your local MySQL credentials.

The database can be set up and filled with development data by running "python DB/db\_setup.py" from the project root. If you need to refresh the database, you can run "python DB/db\_teardown.py; python DB/db\_setup.py"

# Tour of Project

The project consists of four major sections: the database, which maintains system information and state; the backend, which accesses the database and abstracts it out into an API layer; the frontend, which presents the data and controls to the user through the use of the API; and the testing suite, which ensures the API and frontend are behaving as expected.

This tour will dive into each of these components to detail their division of work and how they interact with one another.

## Database

The database is run with MySQL, and this guide assumes you have installed it and set up a root account or user with necessary access (since this is a local development environment, the security of an account is not super significant).

You will need to update the connection string in the appsettings.Development.json file. Most likely, the connection string will require renaming the server to `localhost` and filling in the username and password. Since you do not want your database password to be public, the Developing appsettings is excluded for git using the .gitignore file. If the appsettings format needs to be updated, do so in the App Settings section

Once you have MySQL and Python installed and your connection string updated, you can run the python script in a terminal at the project root to create the development database: `python database/db\_setup.py`.

The folder DB contains initialization sql files, csv files of table data, and the two python files control the creation and removal of the database to ensure a consistent and safe testing environment.

## Backend

The Backend is developed using Visual Studio and C# ASP.NET Core 3. The main dependencies are the ability to query the MySQL database.

* The SQL folder contains parameterized SQL queries that are used to add, get, and update system data.
* The Entities folder contains the user class definition to manage attachment of users to HTTP(S) requests.
* The Services folder defines the User Service, which manages user authentication and other common user tasks (like notifications). It also contains middleware that is used to check if a request has a JWT token and if that token is valid, it adds that user to the request.
* The Models folder contains the input and output class expectations for all of the API endpoints (the conversion between JSON and these class definitions is seamless).
* The Controllers folder defines the api endpoints and the functions that handle those requests for data management (add, get, update, and delete).

Starting the backend is done through Visual Studio's IIS Express development server with their run and debug option.

Opting to not use VS can be done, this requires knowledge of the dotnet CLI tool.

## Frontend

To initialize the frontend of the system, you will need to be inside the Frontend/src directory in a terminal. Within that directory you can run "npm start" and the React development server will start. This dev server will also reload pages every time a file is saved to help speed up prototyping or testing.

Within Frontend/src, there are pages that are the main locations of pages. These import components to render different pieces on the page. App.js defines the possible pages and their references, and public provides a place for all static resources to sit and be referenced from.

## Testing

All tests are written in Python's PyTest module and style. Tests for the API use python's requests library and the Frontend is tested using Selenium.

In order to run the tests, you will need to make sure that you have your workstation environment configured, and both the backend and frontend are running. Once that is done in your terminal of choice go to the main directory of the project and start your python virtual environment. Once there, run command `pytest` and ensure that all tests passed.

# Deployment Procedure

Deployment of this project has no required operating system, but Linux is recommended for widespread support and cheaper server costs.

A MySQL server should be created and maintained, this will require the creation of application and user roles while observing security principles.

The frontend react scripts can be built into static javascript and html files which can be added to the backend C# server as the hosted static resources. Together these can be served with a production server like Apache. Changes should be made to the App Settings json file and the API URL referenced by the frontend fetch requests.

# 