

1 - Software Install

Android Studio, C#/ASP.NET Visual Studio, MySQL Community Server, Postman

Android Studio - Download this from developer.android.com. The installation is pretty straightforward and doesn't require any additional commands.

Visual Studio - Download and install Visual Studio Community 2017 from <https://www.visualstudio.com/>. The installation is straightforward as well, the only key part is to make sure the checkboxes for C#/ASP.NET during the installation are checked. If Visual Studio is already installed on the machine, launch it, go to Tools -> Get Tools and Features, and ensure that the ASP.NET & Web Development workload is installed.

MySQL Community Server - For a lightweight installation, download MySQL Community Server from <https://dev.mysql.com/downloads/mysql/> and extract the compressed folder to the location you want the MySQL server files to be located. This document refers to that location as [MySQL Directory].

Postman - Postman can send arbitrary HTTP requests and receive HTTP responses, which is useful for testing the Web API without building a user interface. Download the installer from <https://www.getpostman.com/apps> and run it.

2 - Environment Setup

Android Studio:

Once the project is created, select Run and then Run 'app' or Debug 'app' from the Navigation Bar. This will take you to a menu to select your deployment target. If you have an Android device plugged in via USB, you can choose this. Otherwise you can configure a virtual device. If you have not configured a virtual device before, you are prompted to select a model, RAM to use and more (the project was primarily tested emulating a Galaxy Nexus). When you select "OK" an emulator will appear and open the app. Add and edit files through the Android Studio IDE, which is pretty straightforward.

The Android Studio project on the repository may contain user-dependent and/or binary files which are not intended to be tracked, such as in the .idea and .gradle directories. Avoiding adding these files in the first place is a good idea, but if they are pulled and Android Studio starts acting up (e.g. saying the Android SDK version must be selected before running the app), deleting .idea and .gradle, opening Android Studio again, and pressing the "Sync Project with Gradle Files" button helps.

Visual Studio:

Visual Studio is used to develop the Web API (FoodServiceAPI) and Web interface (CapProj) projects.

The Web API project uses the ASP.NET Core 2.0 framework, which is open-source and cross-platform. The Web interface project uses the ASP.NET Framework, distinct from ASP.NET Core 2.0.

Open the solutions pulled from the repository. For running your project, select **IIS Express** from the navigation bar or type **Ctrl+F5** (for running without debugging) or **F5** on your keyboard.

Running the project from Visual Studio launches the IIS Express server on localhost. The Web interface project opens the default browser to the current page being edited. The API project opens the default browser and automatically sends a request to a route specified in the project properties, currently configured to the default *api/values* route. That route does not exist, so the response should be a 404.

The Web API is set to listen for external requests on 127.0.0.1:50576.

Note: The **Web API project must be opened and run with administrative rights** in order to avoid an error. It needs admin rights because it is configured to have IIS Express accept requests to '127.0.0.1' instead of the default 'localhost', which IIS Express treats as two distinct host names during routing. '127.0.0.1' is used because a '10.0.2.2' request sent from the Android emulator is translated to '127.0.0.1', not 'localhost'. An Invalid Hostname error (along with a status code 400) would occur otherwise. Oddly enough, sending requests to 'localhost' directly (i.e. from Postman or the Web server) works OK.

MySQL Server Initial Setup:

Open cmd.exe (may need admin rights for this initial command) and `cd` to [MySQL Directory]/bin. Execute the command:

```
mysqld --initialize-insecure
```

This sets up the MySQL server, creating the root@localhost user with no password (which is okay for local development), storing information in [MySQL Directory]/data, and creating a Windows Eventlog registry key. The command may take a while. It never has to be run again unless you reinstall MySQL Community Server.

To run the MySQL server, `cd` to [MySQL Directory]/bin and execute `mysqld` (admin rights no longer needed). The console will start `mysqld.exe` in the background and block while it is running, though you can exit cmd.exe without shutting down `mysqld.exe`.

To log into the running MySQL server as root@localhost, open up another cmd instance in the bin directory and execute:

```
mysql -u root -p
```

If root has no password, either leave out `-p` or enter a blank password when prompted.

Once logged in, the cmd prompt should be "mysql>". Here you can enter SQL queries, such as `SHOW DATABASES;` to show all the databases on the server. The semicolon is essential, otherwise MySQL will prompt for additional lines until a semicolon is entered. If, even with a semicolon, MySQL is still prompting for new lines, stopping and restarting both `mysqld.exe` and `mysql.exe` seems to fix the issue.

Enter `SHUTDOWN;` to stop the MySQL server. Enter `EXIT;` to log out.

If you want to set the password for the user you are logged in as, execute:

```
SET PASSWORD = 'password';
```

MySQL Database Setup:

To run the MySQL script that creates a database on the MySQL server, do the following:

- Start the MySQL server (launch `mysqld.exe` from cmd)
- Log into the MySQL server as a user with admin privileges, such as root (`mysql -u root -p`)
- Execute `SOURCE [Path to Script]`

[Path to Script] is relative to the working directory you were in when you logged into the MySQL server.

Connect Web API to MySQL Database:

The Web API project is configured to connect to the MySQL database via the `MySql.Data.EntityFrameworkCore` package used to implement a database context that can be used with LINQ. The connection string is in `appsettings.json`.

Before running the project, make sure `mysqld.exe` is running on the port specified in the connection string (can set temporarily with the `-P [port]` option). The default `mysqld.exe` port is 3306.

Postman:

After launching Postman, you can configure and save requests. For example, to create a request to the `api/user/login` route, click New, click Request, set the request's name and select a request collection, and click the Save button. Set GET to POST and enter "`http://127.0.0.1:50576/api/user/login`" for the URL. Go to the Body tab, select the 'raw' bullet, change Text to JSON (`application/json`), and enter the parameters in JSON format:

```
{
  "username": "testbusy",
  "password": "dontsteal"
}
```

Click Send to send the request and wait for a response, shown at the bottom of the window once received. In this case, you should get a session token on success. Click Save to save changes to the request.

Import Postman Collection:

In the repository is a file (`FoodServiceAPI/sms_food_service.postman_collection.json`) that defines a collection of configured requests for all the API routes. To import these into Postman, click the Import button next to the New button and choose that file.

Playing with these requests is probably a good way to grasp how communication with the API does and does not work, and they are handy to have for quick tests.

3 - Github Integration

Getting Started and Git Workflow options:

This depends on our development flow. If we are using the Fork and Pull Model, one person will need to create the master repository on Github. Others will fork the master and clone that fork onto their local machine. They will need to link the clone to the upstream repository using `git remote add upstream http://github.com/nameofrepo` and check for changes before making their own. In this model you

then add changed files using: `add filename` or `add *` to add all. Then `commit -m "Some message with a note about changes."`. After that, on Github you submit a pull request.

If using the Shared Repository Separate Branches model, we will need to create a repository add all of the contributors and clone a copy to our local machines. We will need to create branches for our specific features using: `git checkout -b my-new-feature-branch`. Adding and committing are the same as described above. Then execute, `git push origin my-new-feature-branch`. There will be an option to submit a pull request. If it is approved by the rest of the team, make sure everything is still up to date to avoid merge conflicts and then execute the following three commands: 1. `git checkout master` 2. `git pull origin master` 3. `git merge --no-ff my-feature-branch`. Then delete the branch from local and master repositories.

Android Studio:

This is much easier. The process will be the same for creating and sharing or forking a repository as listed above. However all changes can be managed via the GUI VCS menu. You just have to select preferences, version control, and Github to configure your github account.

There are also multiple tutorials and documents online such as:

https://www.youtube.com/watch?v=_d4fFFAJKVA

Visual Studio:

The process is similar to Android Studio. Microsoft Docs provides a nice guide on how to do this. All we have to do is follow it: <https://docs.microsoft.com/en-us/vsts/git/gitquickstart?tabs=visual-studio>

MySQL:

For now, the only thing we will commit is a database-creation script, which will also insert example data. Later, a user-creation script may be committed that creates the MySQL user that the ASP.NET server will log in as. Early tests can use `root@localhost`.