**Pollinator Garden Planner**

Developer’s Guide

North Carolina State University

Department of Applied Ecology

CSC 492 Team 5:

Kevin Bartushak

Noel Drotor

Tammy Le

Carolyn Thompson

North Carolina State University

Department of Computer Science

December 14, 2018

[**Introduction**](#_2zb4lr3x9m2s) **2**

[**Project Setup**](#_7l68u6rqobop) **3**

[**Development Tools**](#_3gz7xkie7owa) **4**

[**File Structure**](#_dl2sluhg0n9a) **5**

[**Build Application**](#_aabjci7g82o7) **6**

[**Development Use Cases**](#_onmw7d8hh9vy) **7**

[**Run Tests**](#_gs5hof7jwtn3) **9**

Introduction

This Developer Guide contains information for future design teams and sponsors who want to extend or integrate the Pollinator Garden Planner with other systems. The guide will aid in the understanding of the project’s file structure, development tools, most common development use cases, and how to build the application.

# 

Project Setup

The team recommends that you use Visual Studio Code as a source code editor for this project, but any preferred editor/IDE can be used. The following setup will be described with the context of using Visual Studio Code.

Steps

1. Download Visual Studio Code at <https://code.visualstudio.com/>.
2. Create a directory on your local file system for the project
3. Navigate to [github repo link] and select “Clone or Download” and copy the repo link
4. In Visual Studio Code, sect File 🡪 Add Folder to Workspace 🡪 Select the newly created directory.
5. In Visual Studio Code, Select Terminal 🡪 New Terminal
6. In the terminal window, type:

git config --global user.name <github userID>  
  
 git clone <URL from github link copied earlier>

Version Control

For Windows Users, we recommend using Git Extensions, a toolkit aimed at making working with Git on Windows more intuitive. This may be a good option for you if you want a more visual representation of the project branches. Git Extensions can be downloaded here: <https://sourceforge.net/projects/gitextensions/>.

Development Tools

The following instructions describe the commands for installing development tools used in the project.

Installation

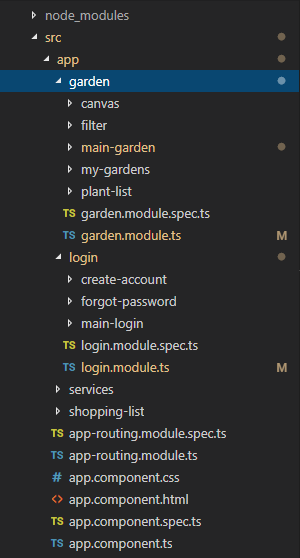
The root of the project folder contains two batch script files that may only be run on Windows machines. These files can be run from the file explorer or through the command prompt window by switching to the root of the project folder.

1. Setup.bat should be run one time at the beginning of the semester. Anytime you want to run the application, run the file run.bat.

Using Angular

The front-end layer of the project uses Angular 2. To familiarize yourself with the module and component structure of Angular, we recommend completing this tutorial: <https://angular.io/tutorial>.

File Structure



**Figure 1. A Glimpse of the Project Folder Structure**

Figure 1 above shows the project directory and file structure of the web application. Although the team has not implemented all the requested features of the web application, they have created directories for each of the front-end modules specified in the design: login, garden, and shopping list. Each module directory contains .css, .html, .spec.ts, and .ts files in which the components specified in the design are created. The module directories live inside the pollinator-garden directory. App.js, package.js, and all the .js files for each route in the web application live inside the api directory. The pollinator-garden and api directories live in the root directory of the project. As described in the Resources Needed section, the configuration files that exist for the project are the dependencies required for the application. The Angular web application is connected to the database through Node.js. The team has installed Express, Body-parser, and MySQL packages in Node.js; they are located in the package.json file in api directory as dependencies.

Build Application

Steps

1. In Visual Studio Code, Select Terminal -> New Terminal
2. In the terminal window, type:

ng serve

3. Click link: <http://localhost:4200/> in the terminal to launch Pollinator Garden Planner website.

Failure

If the above command fails with the message “Port 4200 already in use”:

1. Open cmd.exe as administrator
2. In the terminal window, type:

netstat -ano | findstr :4200

3. Note the PID number located in the right-most column beside “LISTENING”. In the terminal window, type:

taskkill /PID [insert PID number] /F

Replace “insert PID number” with the PID you noted in the above step. Now you should be able to launch the application as normal with “ng serve”.

Prevention

To avoid the above error, terminate the batch job in the Visual Studio Code terminal when you are done using the website. To do so, type on the keyboard:

Ctrl + C

You will be prompted “Terminate batch job (Y/N)?” Type “Y” and enter.

Development Use Cases

Add/Edit Table in MySQL Database

Although all currently planned tables for existing use cases have been created to the database, new fields might need to be added to existing tables, or new tables might need to be created for future functionality.

Using MySQL Workbench, you can add tables and edit fields via the GUI interface.

A tutorial for creating/editing tables in the Workbench can be found here: <https://dev.mysql.com/doc/workbench/en/wb-table-editor.html>.

Add REST API Route & Angular Service

To retrieve data from the database, we have an API running in Express. In the *routes* directory of the *api* folder, we define the REST endpoints for each class in the application.

1. If needed, create a new route file in the *routes* folder. Otherwise, edit an existing one.
2. Create a new endpoint function with a unique path.
3. Construct the SQL query string and query the database.
4. Pass back the JSON data and HTTP status of the request.

To handle 404 errors, we created a middleware function that lives in the app.js file. To call this function from any other function in the API project, simply call return next(). This will exit the current function and proceed to the next function in the middleware queue, which is the 404 error handler.

To use this new API endpoint, you’ll also need to create an Angular service that will call the endpoint. These services live in the *services* directory of the Angular project.

1. Create a new service using the Angular command line interface:

ng generate service services/<name>

1. Use an existing service as a template for imports and other setup.
2. Create functions for each endpoint to retrieve or store data from the database.
3. In your Angular component class, create an instance of the service in the component constructor.
4. Create a component function that calls the service function.

Create Data Model

Another part of the service components is the data classes that the front end uses. At the top of each service, we declare any classes that are relevant to the service. This ensures that data received from/passed to the service are valid objects. You can use these classes in your Angular components by importing them.

Create modules/components in Angular

When more features of the program are required, a new Angular feature module will be necessary. Roughly, each feature module should contain one set of functionality for the program.

For example, our *garden* module contains the components dealing with creating a garden interface, which is the main feature of our project. For more information on feature modules, read the [Angular tutorial](https://angular.io/guide/feature-modules).

Create a feature module:

1. Use the Angular command line interface:

ng generate module <name>

1. Create necessary components:

ng generate component <module-directory>/<name>

Creating a component will generate four files in a new directory:

* *component.ts* The TypeScript file, containing the logic of the component.
* *component.spec.ts* The Karma/Jasmine unit test file (see more below).
* *component.html* The HTML template.
* *component.css* The style sheet.

# 

Run Tests

Karma/Jasmine Unit Tests

The Angular framework comes with the Karma Jasmine unit testing framework, which we used to create white box test cases for the application.

To run these tests,

1. Open a Node command prompt and navigate to the Angular project directory.
2. Run the following command: ng test --code-coverage
3. Once the project is built, a new web browser will appear and run the tests.

To see the coverage report, navigate to the Angular project directory. In the *coverage* folder, open the index.html file in your web browser.

Postman API Tests

Using Postman, we created a collection of API calls and tests for our application. In order to run the test collection, follow these steps:

1. Import the Postman collection, located in the *api* folder of the git repository.
2. Import the Postman environment, in the same folder.
3. Once in Postman, click the “Runner” button in the top left corner.
4. Select the “Garden API” collection and environment.
5. Hit “Start Run.”

To see the tests that are being run for each call, click on the call and select the “Tests” tab. The right side of the screen has code snippets that are quick and easy for adding simple tests to calls.