Youth Homelessness Project

Summer 2022

Team:

Allen Youngblood

Altamush Mirza

Chris Hall

Derek DiLeo

James Purvis

Jean Dalmont

# 

[**Overview**](#_sakqq2txb49w) **3**

[**Production Stack of Project**](#_kjbimlbn8onk) **3**

[Software](#_gcn48vgkbr4a) 4

[Spring Boot](#_kanza66p0p6t) 4

[AWS](#_d4be4lk38a1t) 4

[MySQL](#_cjc3pvbvw4r1) 4

[Database configuration](#_exevz9iybuc2) 5

[Endpoints and REST API](#_f3fvj6wpocnj) 5

[Notes on localhost deployment](#_ipg0m7o7bhpb) 6

[Notes on AWS deployment](#_vm0anqtywya3) 6

[Testing with TestNG](#_j0xjv3n4sf05) 6

[**Installing and Configuring the Project**](#_ryapi9ug7yta) **7**

[Download The Project](#_myw5ayvqkju2) 7

[Install Necessary Software](#_5h0p1llsg1hg) 8

[Configure MySQL Database for local testing](#_tgv83h3hwsy0) 9

[Running the Project](#_syad1zaruw1) 10

[**Troubleshooting**](#_dy1obcltue18) **10**

[Update Dependencies](#_5rbcjaw24gu8) 11

# 

# Overview

The **Youth Homelessness Project** is built as a Java Spring MVC web application to collect data through a voluntary survey hosted by the City of Orlando. This data is used to build real time reports for the city about instances of homelessness, food insecurity, and issues with household dependents such as child and elder care primarily among college aged youth. In exchange for this anonymous information, survey takers are provided a dynamic list of resources that they can access in the greater Orlando area based on their survey responses.

In an example case, someone who (as an example) expresses food insecurity while not showing needs in the area of housing or childcare, would be shown resources specifically for their food insecurity and in their area. This benefits the participants by providing them with information about appropriate resources for their needs while also collecting information about the current needs of participants as a whole. This, in turn, can be used by the city to address needs and allow for better insight into what types of social welfare programs would have the largest benefit to the community.

# Production Stack of Project

This project is bootstrapped using Spring Initializer with the following Spring Boot Starters and Maven dependencies: Web, Data JPA, Security, Thymeleaf, Validation, Devtools, Lombok, TestNG, and MySQL Java Connector. The entities use Lombok annotations to minimize boilerplate code and use Hibernate Validator for Java Bean validation. The project leverages the power of Spring Data Java Persistence API (JPA) repositories and Hibernate Object-Relational Mapping (ORM) which removes the need for the hard-coded (and often error-prone) SQL queries for data persistence and retrieval. These queries are passed via the MySQL Java Connector to the MySQL database indicated in the application.properties file located in the resources directory. Thanks to Spring Web Starter, the app comes bundled with an Apache Tomcat server and can be spun up by simply running the main method within the AcademicSuccessApplication class in the package root of the app. When an HTTP request is made to a valid endpoint (i.e. one that is defined in a controller class), the appropriate view will be returned to the user via Thymeleaf / HTML templates. Otherwise, the user will be presented with one of four Thymeleaf error pages- depending on the HTTP status code.

Most communication between the front end Thymeleaf templates and the controller classes are done via data transfer objects (DTOs). Similarly, most communication between the controllers and JPA repository classes (which communicate with the MySQL database) are made via service implementation classes that rely on constructor-based dependency injection of their respective repository classes at runtime. The predefined methods provided by the JPA repositories allows the app to easily perform standard create, read, update, and delete (CRUD) operations. Automated tests were written using the TestNG framework to ensure proper HTTP functionality of our home and login controller classes by asserting response codes are equal to 200 (OK).

## Software

* AWS
* Java v1.8
* Spring Boot
* Spring Web
* Spring Data JPA
* Spring Security
* Apache Maven
* Lombok
* MySQL
* TestNG

### Spring Boot

Spring boot was used to get the Youth Homelessness Project application up and running quickly by simplifying the process of selecting and downloading the necessary components and dependencies which are both up to date and compatible with each other.

### AWS

AWS is used both for cloud based deployment of the database and the containerized application to allow for dynamic traffic handling as well as to build reports using “Amazon Quicksight”. While not required, Amazon allows the project to be hosted on a cloud based platform and allow dynamic hosting, powerful security, and lower deployment costs

### MySQL

The application uses MySQL for data persistence in both development and production environments. During development, the database is hosted on a local MySQL instance but in production, it is deployed on an AWS RDS instance running with MySQL. Rather than writing SQL queries manually, we leverage Spring Data JPA methods and Hibernate ORM to CRUD the data.

#### Database configuration

### 

The database consists of nine tables- each corresponding to a separate entity declared in the models package of the application and was designed to achieve third normal form (3NF) to avoid unnecessary duplication of data. The entity classes make use of annotations to call a number of predefined methods in the JavaX Persistence, Lombok, and Hibernate Validator packages. The JavaX Persistence methods determine how the entity classes are mapped to their respective database tables via ORM (and vice-versa). The Lombok annotations are used to avoid the need for numerous lines of boilerplate code by inferring getters and setters for each field within a class as well as hash code, equals, and toString methods for each class. Lastly, Hibernate Validator annotations are called where necessary to ensure (or validate) sufficient data is present when an object of said class is instantiated and persisted to the database.

Although this is subject to change, the application currently has three types of users, Admins, Employees, and Students- with each getting their own respective table in the database to store information such as firstName, lastName, username and password (which is encrypted with BCrypt). In addition to a one-to-one relationship with Session entities, Survey entities have a List<Question> which corresponds to a one-to-many relationship and, likewise, Question entities contain a List<Option>. Resource entities, which are returned to users based on their survey and responses, have a name, description and contain boolean fields related to their type (food, housing, or dependent). Additionally, each Resource entity has an Address which correlates to a one-to-one relationship. Session entities store information related to each individual survey submission. These data include the Student who took the survey (subject to change), start and finish timestamps and the need-based scores for each resource category calculated by their responses. These scores determine the resources returned to the user, and will also be used by the city to determine what resources are needed most and (ideally) where to deploy them. Finally, every entity has an id field of type Long which functions as the primary key and is automatically generated by the JavaX Persistence API using the GeneratedValue annotation.

### Endpoints and REST API

To adhere to the separation of concerns design principle, the logic for the REST endpoints provided by this application is spread across ten separate controllers located in the controllers package and indicated by springframework.stereotype.Controller annotations. Admin actions can be found in the AdminController, Employee actions can be found in EmployeeController, logic pertaining to Surveys can be found in the SurveyController, and so on. To communicate with the database, each controller accesses methods in the associated service(s) which are made available via field-based dependency injection at runtime. These services then call their respective JPA repositories which can, in turn, access the database.

#### Notes on localhost deployment

To deploy the application on your localhost, clone the [GitHub repository](https://github.com/derekdotdev/YouthHomelessnessProject), install MySQL Workbench and set up a connection (note: the username and password in the resources / application.properties file will need to match your chosen username and password). Next, open the studentsuccess.sql file in the root directory of the repository and run the CREATE DATABASE command under step 1. You should now have a database called studentsuccess. Now, import the project into your favorite IDE and run it. At this point, Hibernate will create a table for each entity in the models package. Lastly, uncomment the code in steps 2 and 3 in the studentsuccess.sql file and execute to populate users and dummy data to the database. Lastly, go to localhost:8080/ and try logging in. Username / password combinations are admin/admin, employee/employee, and student/student.

#### Notes on AWS deployment

We choose to use AWS for our cloud deployment. Most organizations today have moved most, if not all, of their IT infrastructure to a cloud services provider. AWS dominates the market right now, so that seemed like the most logical choice. We deployed our application to a service called elastic beanstalk in AWS. It’s fast, easy to use and eliminates the manual work of having to provision servers, load balancers etc.. All that has to be done is take the latest jar file from the project, and configure the security and server settings. Upload the jar to the elastic beanstalk service, and test to make sure it’s working properly. We also set up an AWS RDS instance as our database to push and pull data from. The web app was set up to communicate with the RDS instance in the configuration file.

### Testing with TestNG

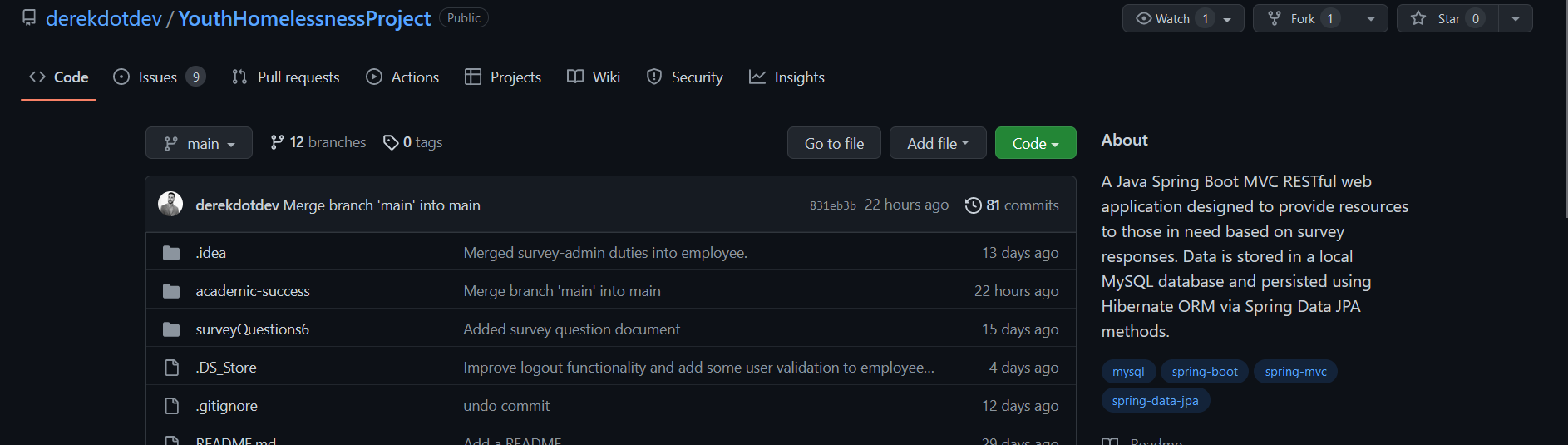
TestNG was set up to be our testing framework.Spring boot has some out of the box testing functionality, but TestNG has enhanced features, and allows our testing to be more organized. With the testng.xml file, organizing and controlling our tests are easy. Once the test cases are written, the xml file can be configured to run tests in a certain order, skip certain tests, run different test cases at the same time (parallel execution) etc.. The major advantages of this may not be apparent if you only have a few test cases, but when they scale up as the application grows, it’s really helpful to have a powerful testing framework that is organized and easy to control. At this stage in development, the only test cases that we have written are to test the rest endpoints to ensure that they are working and accessible.

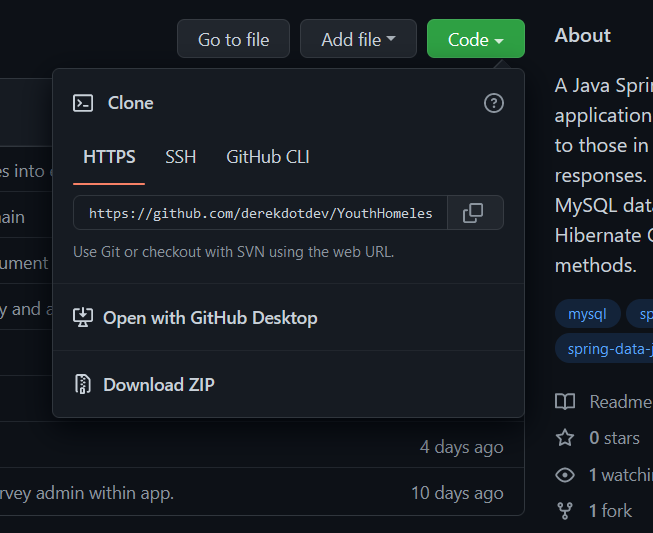
# 

# Installing and Configuring the Project

## Download The Project

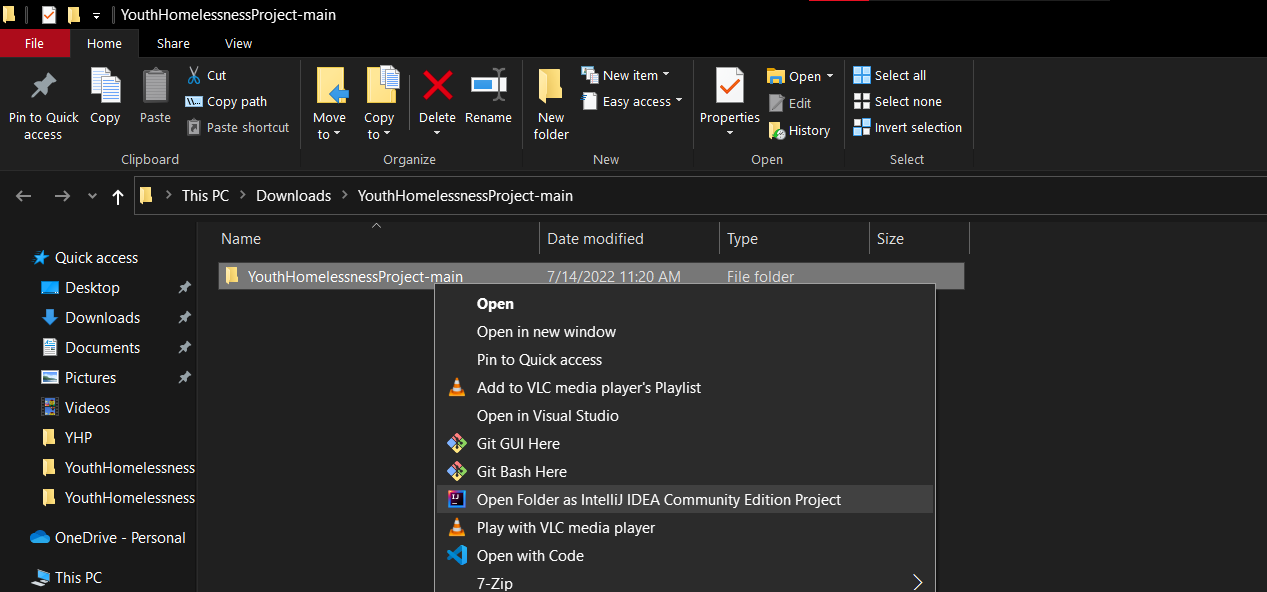
1. Using your web browser, Navigate to (“<https://github.com/derekdotdev/YouthHomelessnessProject>”), this is the main page of the Project as of 07/2022. **Future team leaders: please FORK this repository to your GitHub and use the FORKED repository as your new main.** This way, past contributors may continue to track the application as it progresses.





1. Select the green code button and select your preferred download option, If you select “Download ZIP” be sure to unzip the file into your desired location.

An alternative method would be to clone the Repository using git with you log in credentials, for more information on this see (“<https://docs.github.com/en/repositories/creating-and-managing-repositories/cloning-a-repository>”)



1. Navigate to your now extracted project folder and right click on the folder. From the drop down menu. Select your preferred IDE to open the project. For this guide, we will be using IntelliJ, but you can use Eclipse or any other IDE you wish.

## Install Necessary Software

While the Maven Wrapper inside of the project will handle most of the dependencies, you will want to make sure you have the appropriate plugins.

If using Intellij - Install the plugin JPA buddy under Preferences > Plugins

If using Eclipse - Install the plugin Spring tools via Help > Eclipse Marketplace…

In addition to these, you will want to have MySQL and IDE installed. In this guide we are using Intellij, however you can use any IDE you would like, just be sure to install the correct plugins to operate the system.

For a guide on how to install Intellij, check out -

<https://www.jetbrains.com/help/idea/installation-guide.html>

For a guide on how to install Eclipse check out -

<https://www.eclipse.org/downloads/packages/installer>

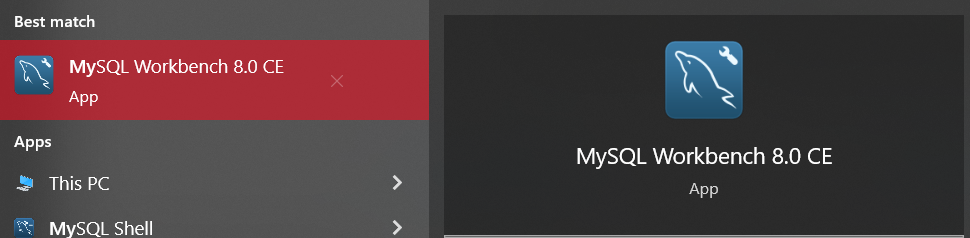
For a guide on how to install MySQL check out -

<https://dev.mysql.com/doc/mysql-installation-excerpt/5.7/en/>

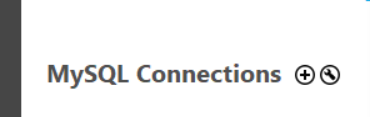
### 

## Configure MySQL Database for local testing

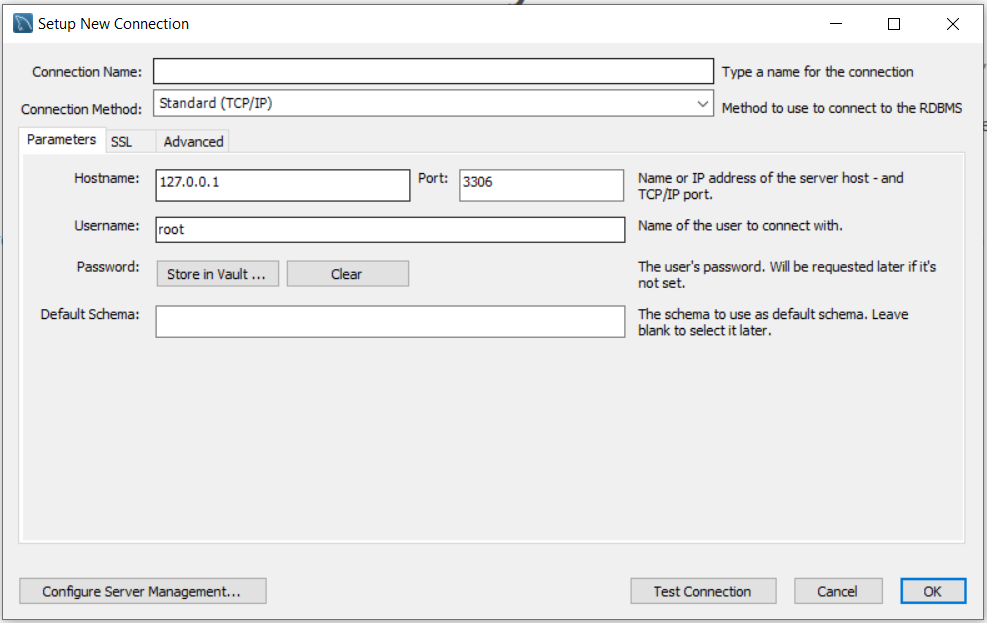
NOTE: We will be creating the database using MySQL workbench. If you don’t have MySQL workbench, reinstall MySQL and select workbench as a package to install.



1. Open MySQL workbench on your machine



1. Once open Click the + option next to MySQL connections

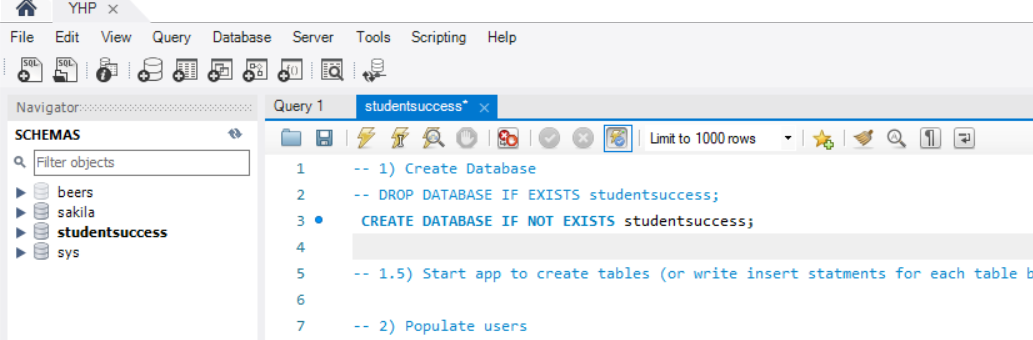


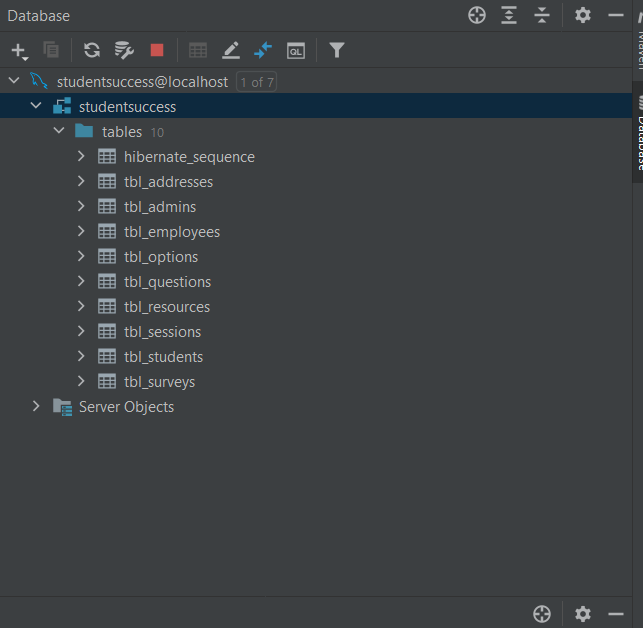
1. To start, you will want to set up your new connection by adding a connection name, hostname, port, username, and password.

These values can be anything, but you will need to change the appropriate values in the application.properties file, most notably your username and password.

It's always a good idea to test your connection first and if everything performs as expected, click okay and connect to your new connection by double clicking the connection panel on the MySQL main page. You will be prompted to enter your password to access the connection. Be sure to update lines 3 & 4 of the resources / application.properties file to match these credentials!

1. Select File > Open SQL Script... and navigate to the folder you placed the project in. Once there, select “studentsuccess.SQL” from the main project directory to open the SQL query.



1. You will see a list of commented out SQL statements, uncomment the top line “CREATE DATABASE studentsuccess;” and click the run option. This will create the database studentsuccess
2. Next, open your project in your IDE of choice. NOTE: you may need to run “mvn clean” and “mvn install” in the folder containing your POM.XML file to get the [dependencies](#_5rbcjaw24gu8) operating. Navigate to the project’s main java directory and right-click the file: AcademicSuccessApplication and run it. At this point, Hibernate will automatically create a table in the database for each entity in the project. Now, you can either navigate back to your MySQL workbench or verify using the integrated DB window in intellij.
3. Uncomment and execute the INSERT statements (the same query from step 5) to populate some users, questions, and resources. This can be done either in the IDE or MySQL workbench.

### 

## Running the Project

With the application still running and the default user data inserted from the previous steps, you should now be able to use the application. Open a browser, type localhost:8080 into the address bar and press Enter. This should take you to the home route which is defined in the HomeController. From here, you can choose to log in as a student, employee or admin using student/student, employee/employee, and admin/admin user/password combinations.

* Students take surveys and are provided with appropriate resources based on the needs indicated by their survey responses.
* Employees can currently create, edit and delete survey questions and resources. In the future, Employees will be able to view data and generate reports based on submitted surveys.
* Admins are presently only responsible for creating and editing Employee and Admin users.

At this time, there exists a handful of resources and zero questions in your local database. So, you can login as an employee to manually create some.

Each resource has a name, description, address, resource category(s) and numeric degree associated with the resource category(s). Think of this number as the severity of need.

Each question can have up to five options and each option can be given a numeric value from -5 to 5 (inclusive) and can be associated with one, two, or all three resource types. When the survey scores are calculated, the numeric value of the chosen option for every question is added to the total score for each associated resource(s). If a selected option is associated with food resources and has a value of 3, 3 points are added to the food score of the survey. Finally, all resources with a degree less than or equal to the calculated resource category score are returned to the user.

# Troubleshooting

This section aims to address some potential problems you may experience when installing the application for the first time. For simplicity, we are only covering IntelliJ and Eclipse IDEs.

If you have just downloaded the project, please review this document in its entirety to ensure your system is properly configured, then view the Update Dependencies section to ensure you have the latest Maven dependencies on your machine.

#### Runtime Exceptions when launching the application

**Problem**: java.sql.SQLSyntaxErrorException : Unknown database ‘studentsuccess’

**Solution**: No database was found. Please refer to the [Configure MySQL Database for local testing](#_tgv83h3hwsy0) section.

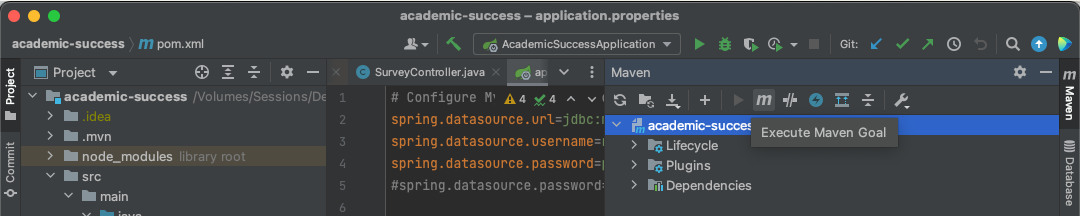
**Problem**:java.sql.SQLException: Access denied for user.

**Solution**: Make sure MySQL is installed and running on your machine and that the username and password chosen when setting up your local MySQL connection is configured under spring.datasource.username and spring.datasource.password in the resources / application.properties file. (see [Configure MySQL Database for local testing](#_tgv83h3hwsy0): Step #3)

### Update Dependencies

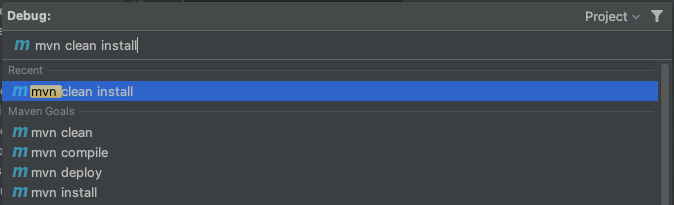
To update dependencies in Eclipse (with m2eclipse plugin), right-click your project and select Maven > Update Project or simply click Project Menu > Update Maven Project

To update project dependencies in IntelliJ, click the Maven tool in the top-right corner of the editor window. If you do not see it, click View > Appearance > Tool Window Bars.

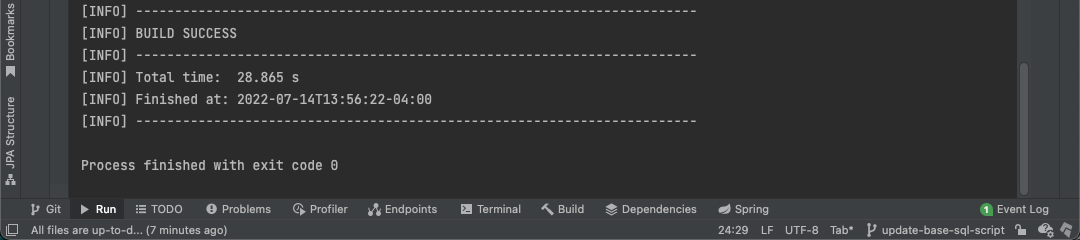
****

Locate and click on Execute Maven Goal and execute **mvn clean install**

(**Note:** this has the same effect as mvn clean followed by mvn install)



Depending on your internet connection, this process will complete in seconds or minutes.



### 