

UMGC Capstone Project Proposal Management System (CaPPMS)

Programmer Guide

Version 2.0

Prepared By:

Bereket Tamrat

Ephrem Kefle

Kathryn Stewart

Marc Bueno

Tarun Lava

Yonas Mekete

Programmer Guide Approvals

Name	Signature	Date
Professor Dr. Mir Assadullah		
Project Manager Kathryn Stewart		

Revision History

Date	Version	Description
10/20/2020	1.0	Initial Programmer Guide
11/3/2020	2.0	Updates based on professor's feedback.

Table of Contents

Table of Contents	4
1. Introduction	6
1.1. Scope.....	6
1.2. Definitions, Acronyms and Abbreviations	6
2. Context	7
2.1. Architecture Guiding Principles	7
2.2. Architecture Constraints	8
2.3. Architecture Assumptions.....	10
3. Development Environment Configuration	10
3.1. Front End	10
3.2. Back-End.....	15
3.3. File Structure.....	18
4. Data	19
4.1. Storage	20
4.2. Data Integrity	20
3.3 Data Archiving	20
5. Software	21
5.1. Business Logic	21
5.2. User Interface.....	22
6. Technical Positions	23
7. Coding Standards	24
8. Known Issues	25
9. References	26

List of Figures

Figure 1 Architectural Design.....	7
Figure 2 npm Included in node.js Installation	11
Figure 3 node.js Version Command	12
Figure 4 npm Version Command.....	12
Figure 5 Angular Install Command	12
Figure 6 Angular Version Command.....	12
Figure 7 Angular Version Output	13
Figure 8 Add to PATH.....	14
Figure 9 Create Directory	14
Figure 10 Code . Command.....	14
Figure 11 Front-End Start Command	15
Figure 12 Front-End Start Output	15
Figure 13 Select PostgreSQL Components.....	16
Figure 14 Create New Database	16
Figure 15 Name the New Database	16
Figure 16 Query to Insert FAQ.....	17
Figure 17 Application.Properties File Location.....	17
Figure 18 Application.Properties File Contents.....	17
Figure 19 Front-End File Structure.....	18
Figure 20 Back-End File Structure	19
Figure 21 Back-End File Structure	22
Figure 22 Front-End File Structure.....	23

List of Tables

Table 1 Acronyms.....	6
Table 2 Coding Standards	24

1. Introduction

This Programmer's Guide provides the key principles for developing and upgrading the UMGC Capstone Project Proposal Management System (CaPPMS). This document covers the important features that needed for design, implementation, and maintenance.

The key intent of this document is to ensure that the current and future teams have sufficient information to understand and use the software. The document is aimed specifically at software developers who may have different levels of familiarity with software to facilitate use and future modifications.

1.1. Scope

This programmer's guide is to help provide a consolidated location to document the software development processes. This will assist current and future developers working on this system understand how the application was created and how updates can be made. This document covers an overview of the architecture, programming guidelines, and software development libraries available for development,

1.2. Definitions, Acronyms and Abbreviations

Below are the terms and abbreviations used in this document.

Table 1 Acronyms

Acronym	Definition
CaPPMS	Capstone Project Proposal Management System
ERD	Entity Relationship Diagram
PM	Project Manager
PMI	Project Management Institute
SDLC	Software Development Life Cycle
SRS	Software Requirement Specification
SWEN	Software Engineering
UMGC	University of Maryland Global Campus
WBS	Work Breakdown Structure

2. Context

The architectural design below represents Spring Boot exporting a REST API using Spring (Web) MVC. Spring Boot will interact with a PostgreSQL Database using Spring Data and Hibernate as JPA Provider. Clients, in the architecture below, interact with the REST API using HTTP request / response roundtrips, displaying data on the components (COMP.). The routing mechanism is used to browse through pages with Angular connecting via Hibernate as JPA Provider.

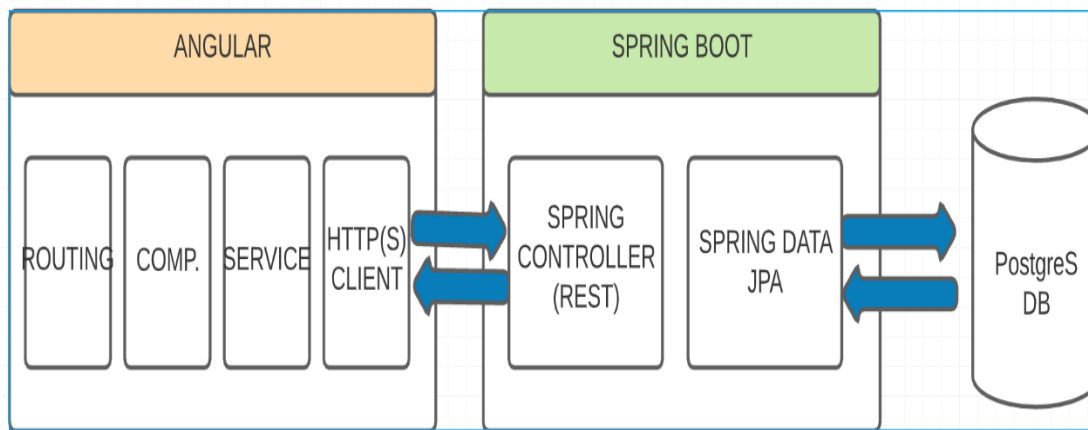


Figure 1 Architectural Design

2.1. Architecture Guiding Principles

Architecture principles establish the general rules and guidelines for the use and deployment of all IT resources and assets. They form the basis for making future IT decisions. On that note, the CaPPMS is designed with the following main principles in mind:

1. Separation of concerns (SoC): separate your application into different sections, and each section will address a separate concern. This is addressed in the CaPPMS through the use of the Model-View-Controller (MVC) design pattern. The application materializes this by using the Spring Data framework.
2. Single Responsibility principle: Every module within an application could only answer a single question and be responsible for a single functionality. The CaPPMS strives to have a well-designed class diagram in which each class has a well-defined responsibility. This does not mean that the CaPPMS follows anti-patterns such as the “Fear of Adding

Classes”. In fact, the design aims to achieve a perfect balance when it comes to number of single-responsibility classes.

3. Principle of Least Knowledge (Or Law of Demeter): An object should have limited knowledge about other objects. This is to avoid getting your objects tangled with each other, especially in applications that have multiple layers as it is the case of CaPPMS.
4. Don't repeat yourself: The idea behind the DRY (Don't Repeat Yourself) is carefully followed in the CaPPMS design and implementation, i.e., functionality should be implemented in only one component rather than being duplicated in any other component or parts of the application. This was achieved, when needed, through the use of static methods with generic functionality and cross-application classes using or following the idea behind Aspect Oriented Programming (AOP).
5. Minimize upfront design: This principle, also referred to, as YAGNI ("You ain't gonna need it") was used, not only while coding, but mainly during the design of the underlying data model of the CaPPMS. The idea here was to avoid the Big Design Upfront or BDUF, which is clearly aligned with the fact that “things change” as prescribed in most if not all agile methodologies. By minimizing upfront design, while following good design and programming practices, the CaPPMS achieves a good tradeoff between maintainability, performance and readability. Some say that this principle can lead to risks. Still, one can follow this principle by building proof of concepts (PoCs), as it is been the case in the CaPPMS, for demonstrating to the stakeholders the advantages of delaying the decision and the benefits of using an alternative course of action.

2.2. Architecture Constraints

The CaPPMS architecture was designed with modern approaches in an attempt to minimize risks and maximize the cost-benefit of the application. One the constraints that our architecture design takes into account is the available time to development such an application as well as budgetary constraints. Time constraints and a relatively good understanding of what was to be designed and implemented led us to choose a hybrid development approach, using waterfall and agile methodologies together during the SDLC. Agile methodologies such as SCRUM are prescribed for software development in which little to none is known upfront as well as some time flexibility is allowed, always focused on customer's feedback and taking into account that

requirements change. The 12-week timeframe to design and develop the CaPPMS from scratch led us to have a strict scope management which in its turn have to postpone some artifacts and implementation to future versions of the system. An example of such is the Security subsystem of the CaPPMS. In the current version, it was assumed that privileges were basically granted to professors and other stakeholder manually, i.e., directly into the database tables. Although this constraint is related to an important aspect of any software, namely, security, the CaPPMS does offer the underlying basic infrastructure so a more elaborate security model be built upon.

At this point, the CaPPMS has not had to deal with budgetary constraints when it comes to the tools used during the SDLC. Basically, open source tools and minimum cost database management system is in use. This tends to ensure compliance to industry standards as well as good system scalability, maintainability and readability. We anticipate, however, that more advanced tools will be used in order to work with the analytics portion of the CaPPMS in the future. Depending on the tools to be used it may be costly. This is something that the current development is not taking in to consideration in this version of the CaPPMS. On the other hand, one of the constraints the CaPPMS team is having to deal with right now is the limited use of software and code quality measurement tools. The initial idea was to make full use of SonarQube for continuous inspection of code quality to perform automatic reviews with static analysis of code to detect bugs, code smells, and security vulnerabilities. Even though free trial can be used in desktop environments, meeting with the DevOps Teams have revealed that production version could be costly. Hence our limitation in the use of such a tool at the time of this writing.

Last but not least, although the CaPPMS team has used modern development tools which should promote a responsive web application, focus, in this version is given to the web functionalities rather than mobile functionalities. Of course, this version is laying the foundation upon which mobile functionalities could be added or adjusted in future version of the system. In these lines, one of the development paradigms used nowadays is focused on the mobile first design and development approach (<https://www.invisionapp.com/inside-design/mobile-first-design/>). This version of the CaPPMS, due to the aforementioned reasons, was focused on a more traditional development strategy.

2.3. Architecture Assumptions

In order to deliver the best product while still considering a solid foundation for future development, the CaPPMS is development with the following assumptions:

- Priority was giving to designing and implementing based on what is known about the systems requirements at the present time. In other words, parts of the system were designed, implemented and tested based on a more specific approach versus a potential generalist approach that could or could not be true in the future.
- Production environment will have enough resources to host the application, the database management system and adjust to the systems scalability.
- DevOps members may be required to interact with the database via pgAdmin as not all CRUD GUIs will have been developed by the end of this version. Examples include, but are not limited to, Frequently Asked Questions (FAQ) table, Status and Usr_types tables and the Accounts tables. All efforts are and will be in place to complete as much as possible of the aforementioned GUIs and minimize the need to direct interaction with the RDBMS.

Although the team does not anticipate the need of a Database Administrator (DBA), the DBMS used in the CaPPMS offers capabilities that may require specialized maintenance from time to time. Estimation of tablespace growth, schema management, backup and recovery among tasks that this version assumes that will be provided as needed, by DevOps or another group of IT professional.

3. Development Environment Configuration

3.1. Front End

The CaPPMS application was built using Angular 10 as a front-end, Spring boot 2 RESTful API as a backend, connected to PostgreS via a hibernate JPA Provider. Front-End

The tools and technologies used to build the front-end are:

- Integrated Development Environment (IDE) -Visual Studio Code - Angular App development
- Angular 10

- Angular CLI
- NodeJS and NPM
- Bootstrap 4
- HTML5

To configure the front-end code on a local machine for building and development, complete the following steps. These steps assume a Windows development environment.

1. Install node.js
 - a. Navigate to <https://nodejs.org/en/>
 - b. Press download for the version 'Recommended for most users'
 - c. Press download for the version 'Recommended for most users'
 - d. Save the install file
 - e. Open the install file and follow the instructions
 - f. Note that npm is selected for installation as part of node.js.

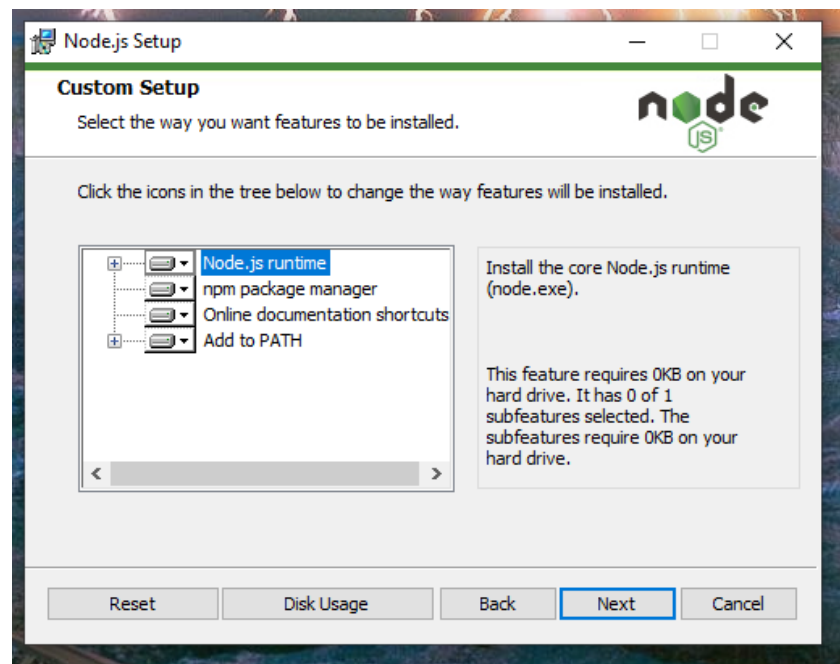
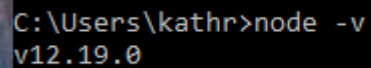


Figure 2 npm Included in node.js Installation

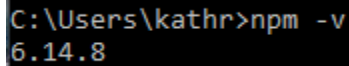
2. Open a command prompt and confirm node.js installation
 - a. `> node -v`



```
C:\Users\kathr>node -v
v12.19.0
```

Figure 3 node.js Version Command

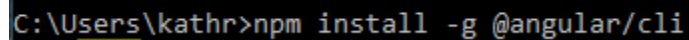
3. From the open command prompt confirm npm installation
 - a. > npm -v



```
C:\Users\kathr>npm -v
6.14.8
```

Figure 4 npm Version Command

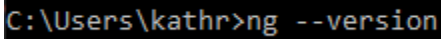
4. From the open command prompt install angular/cli
 - a. > npm install -g @angular/cli



```
C:\Users\kathr>npm install -g @angular/cli
```

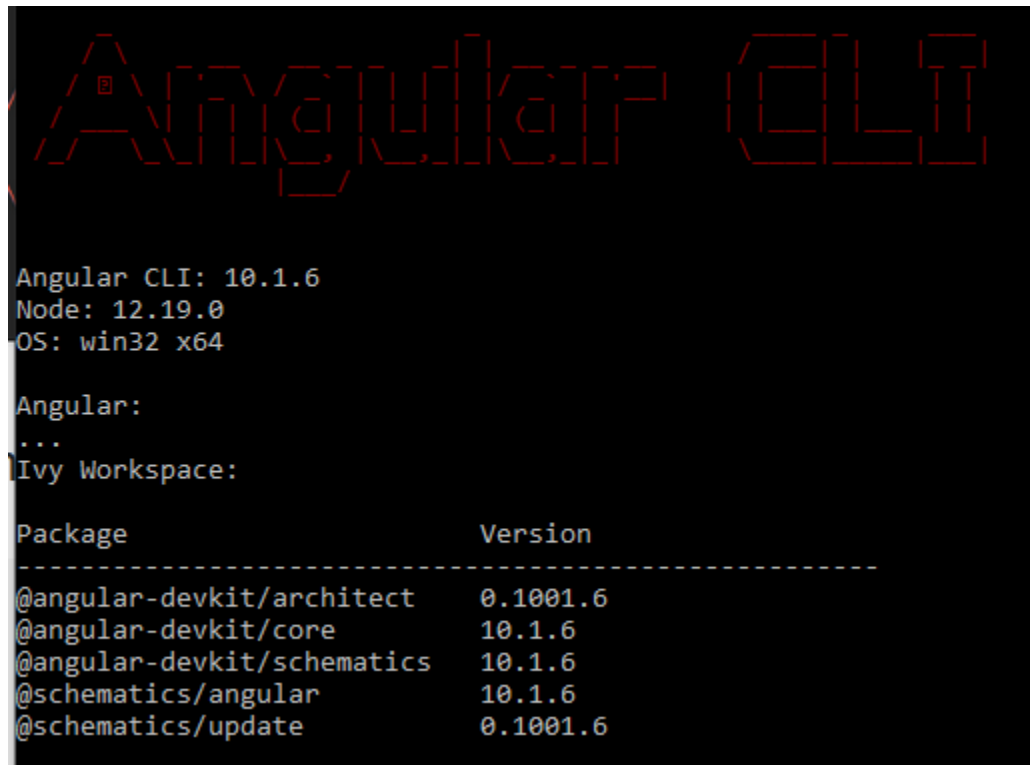
Figure 5 Angular Install Command

5. From the open command prompt confirm angular cli installation. Additional information regarding these commands can be found here: <https://cli.angular.io/>
 - a. > ng --version



```
C:\Users\kathr>ng --version
10.2.3
```

Figure 6 Angular Version Command

A terminal window with a black background and red text. At the top, 'Angular CLI' is displayed in a large, stylized, outlined font. Below this, the following text is shown: 'Angular CLI: 10.1.6', 'Node: 12.19.0', and 'OS: win32 x64'. Then, 'Angular:' is followed by '...' and 'Ivy Workspace:'. At the bottom, a table lists packages and their versions.

```
Angular CLI: 10.1.6
Node: 12.19.0
OS: win32 x64

Angular:
...
Ivy Workspace:

Package                                  Version
-----
@angular-devkit/architect                0.1001.6
@angular-devkit/core                     10.1.6
@angular-devkit/schematics               10.1.6
@schematics/angular                      10.1.6
@schematics/update                        0.1001.6
```

Figure 7 Angular Version Output

6. Install VS Code
 - a. Navigate to <https://code.visualstudio.com/download>
 - b. Press the button to download the installer for Windows
 - c. Save the install file
 - d. Open the install file and follow the instructions
 - e. Ensure the Add to PATH check box is selected, plus any other features desired by the developer.

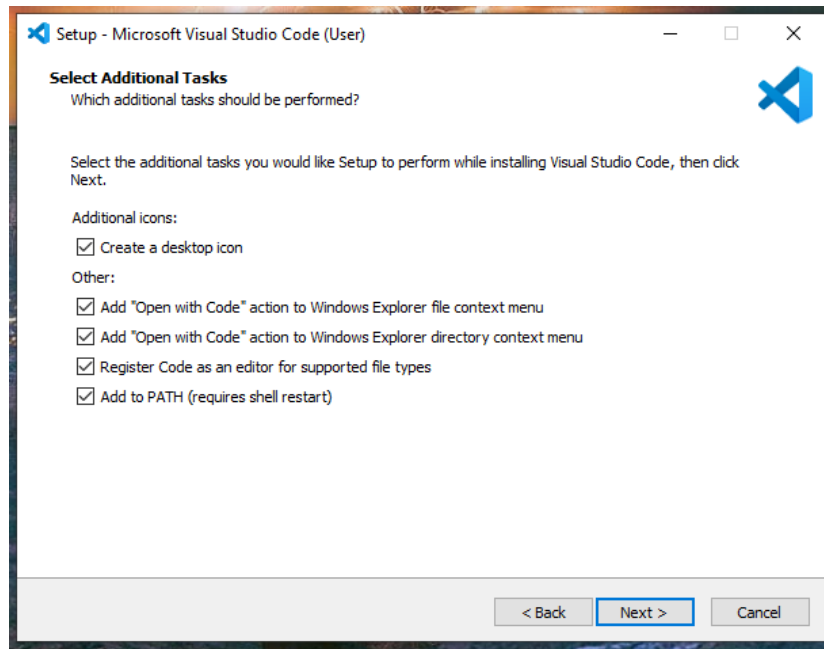


Figure 8 Add to PATH

7. Create a folder to store the project and navigate to that folder
 - a. `> mkdir CaPPMS_FrontEnd`
 - b. `> cd CaPPMS_FrontEnd`

```
C:\Users\kathr\Documents>mkdir CaPPMS_FrontEnd
C:\Users\kathr\Documents>cd CaPPMS_FrontEnd
C:\Users\kathr\Documents\CaPPMS_FrontEnd>
```

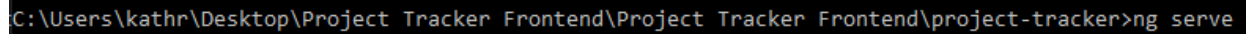
Figure 9 Create Directory

8. Flag the directory to indicate its use for code
 - a. `> code .`

```
C:\Users\kathr\Documents\CaPPMS_FrontEnd>code .
```

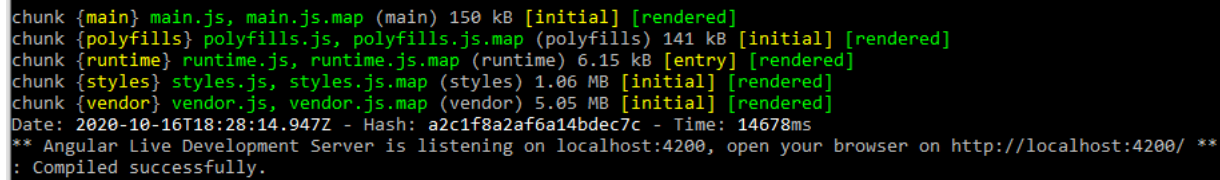
Figure 10 Code . Command

9. Open VS Code and the folder that was just created
 - a. File -> Open Folder
10. Connect to GitHub and clone the repository to this folder.
11. To run the front-end open a terminal window and run this command:
 - a. `> ng serve`



```
C:\Users\kathr\Desktop\Project Tracker Frontend\Project Tracker Frontend\project-tracker>ng serve
```

Figure 11 Front-End Start Command



```
chunk {main} main.js, main.js.map (main) 150 kB [initial] [rendered]
chunk {polyfills} polyfills.js, polyfills.js.map (polyfills) 141 kB [initial] [rendered]
chunk {runtime} runtime.js, runtime.js.map (runtime) 6.15 kB [entry] [rendered]
chunk {styles} styles.js, styles.js.map (styles) 1.06 MB [initial] [rendered]
chunk {vendor} vendor.js, vendor.js.map (vendor) 5.05 MB [initial] [rendered]
Date: 2020-10-16T18:28:14.947Z - Hash: a2c1f8a2af6a14bdec7c - Time: 14678ms
** Angular Live Development Server is listening on localhost:4200, open your browser on http://localhost:4200/ **
: Compiled successfully.
```

Figure 12 Front-End Start Output

12. Navigate to <http://localhost:4200/> to view the program.

3.2. Back-End

Server-side tools and technologies include:

- IDE – Eclipse - Spring boot API development
- Maven 3.3
- Spring Boot 2.2.1
- Spring Framework 5.2
- JDK 1.8
- Spring Data JPA (Hibernate)
- Embedded Tomcat 8.5
- Postgres Database

To configure the back-end code on a local machine for building and development, complete the following steps.

1. Install PostgreSQL
 - a. Download from <https://www.enterprisedb.com/downloads/postgres-postgresql-downloads>
 - b. Ensure that all the options are selected for installation

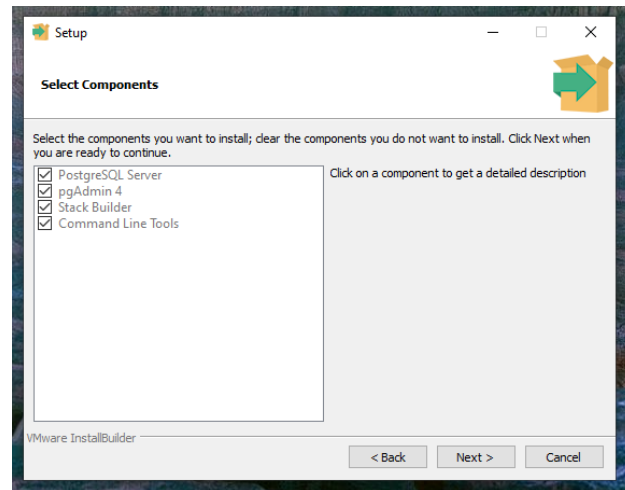


Figure 13 Select PostgreSQL Components

- c. Open pgAdmin 4
- d. Create a database called Project Tracker

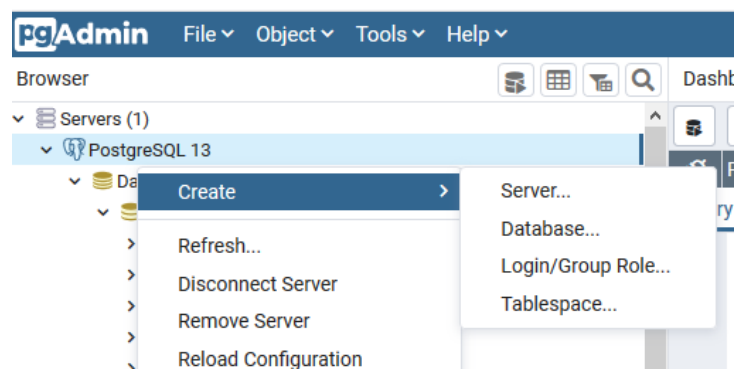


Figure 14 Create New Database

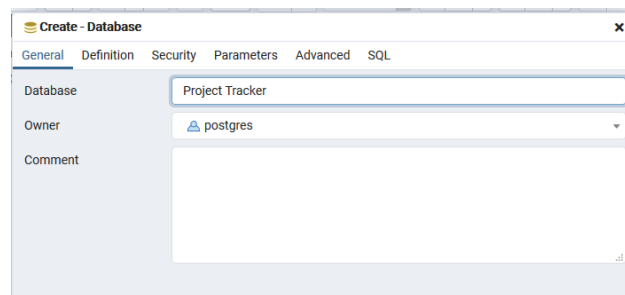


Figure 15 Name the New Database

- e. Enter the FAQ by right clicking on the FAQ table, selecting Scripts, and INSERT Script. Repeat as many times as needed to enter all the FAQs.

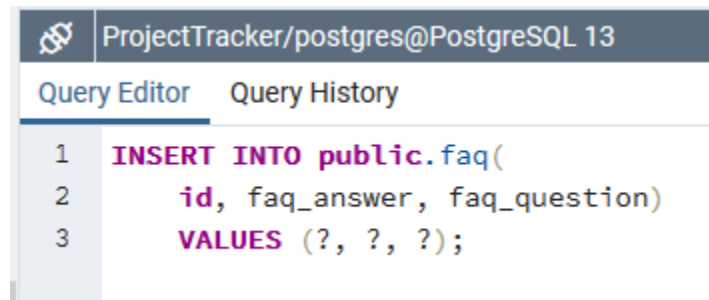


Figure 16 Query to Insert FAQ

2. Using the IDE of choice (i.e. eclipse) import the code from GitHub
3. Open the application.properties folder and update the username and password for the newly created database

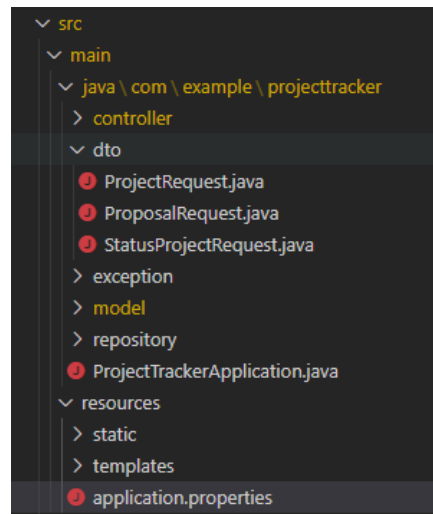


Figure 17 Application.Properties File Location

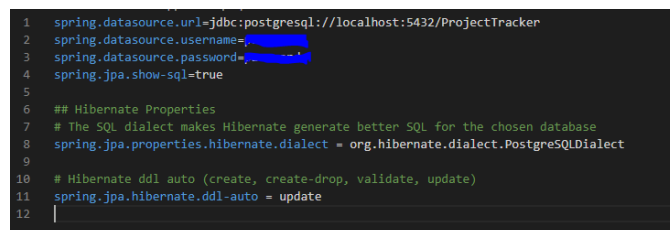


Figure 18 Application.Properties File Contents

4. Right click the project and Build
5. Right click and run the project

3.3. File Structure

There are two separate repositories of code, one for the front-end and one for the back-end.

The front-end code is contained in an angular project with the following structure:

about
create-project
faq
home
login
newadmin
project-details
project-list
project-request
proposal-table
update-project
app-routing.module.ts
app.component.css
app.component.html
app.component.spec.ts
app.component.ts
app.module.ts
faq.service.spec.ts
faq.service.ts
faq.spec.ts
faq.ts
project-request.spec.ts
project-request.ts
project.service.spec.ts
project.service.ts
project.spec.ts
project.ts
user.spec.ts
user.ts

Figure 19 Front-End File Structure

The back-end code is contained in a java project with the following structure:

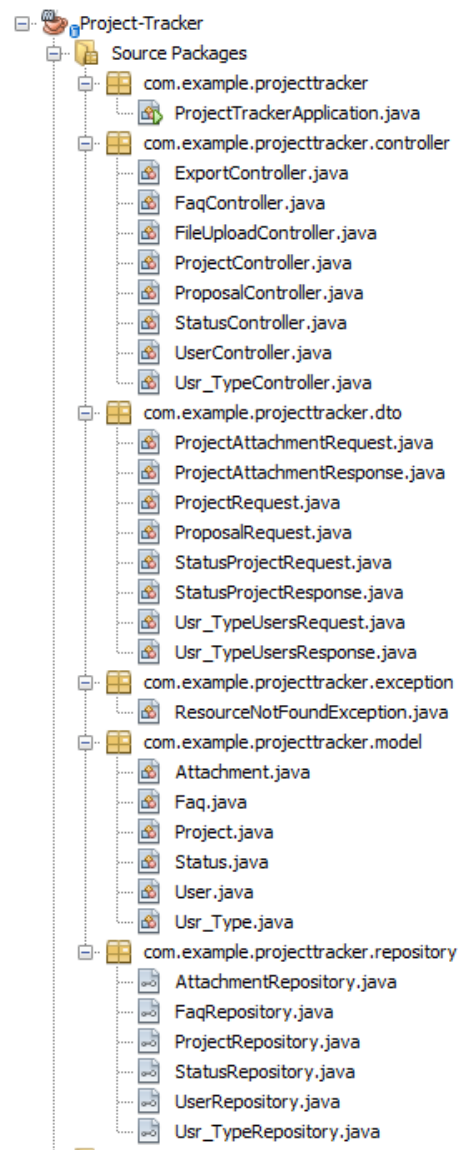


Figure 20 Back-End File Structure

4. Data

In this section of the Programmer's guide the data storage, security, and integrity of the CaPPMS web application will be discussed.

4.1. Storage

The database selected to be used for CaPPMs is PostgreSQL since it is easy to use, open source, works well with Hibernate, has a user-defined data type, a lot of community support, make use of Stored procedures, and It supports Atomicity, Consistency, Isolation, Durability (ACID).

4.2. Data Integrity

The database server will be protected from database security threats by a firewall to deny access to traffic by default. Only traffic coming from specific application or web servers can access the data. The firewall will also protect the database from initiating outbound connections unless there is a specific need to do so. On top of the firewall a web application firewall will also be used to protect attacks such as SQL injection attacks directed at a web application to exfiltrate or delete data from the database.

Exception handling mechanism is also used to protect data source information. The number of people accessing the database will be kept to the minimum possible number as a security measure. Only bare minimum privileges will be given to administrators to do their job, and only during periods while they need access.

3.3 Data Archiving

Archiving data is the process of moving or storing data that is no longer actively used to a separate data storage device for long-term retention of that data. Data archiving consists of older data that is still important and useful for future reference or which must be saved for regulatory compliance purposes. Data archival process is a part of information lifecycle management (ILM). Every additional component in ILM is archived on continuous basis as the relevant data reaches its time to be archived which can be based on so many months old or, preferably, because the data is no longer in active use. The preferred methods chosen are exporting archivable data to an external table (tablespace) based on a defined policy and compressing the external table and store it at a third-party data storage facility like a cloud storage service.

5. Software

The CaPPMS software modules are organized in terms of packaging and layering. The sections below detail the repositories for the business layer and the presentation layer in terms of the package hierarchy.

The code repositories are available on GitHub. They are located at:

<https://github.com/umgc/umgc.idea.tracker> and <https://github.com/umgc/umgc.idea.tracker.ui>.

5.1. Business Logic

Business logic is the part of the application that retrieves, processes, transforms and manages the data. It applies any business rules or policies and ensures consistency and validity of data.

The backend hierarchy starts with standard Java namespace compliant structure and then is divided into sub packages for control. The sub packages for the business layer are:

- controller
- dto
- exception
- model

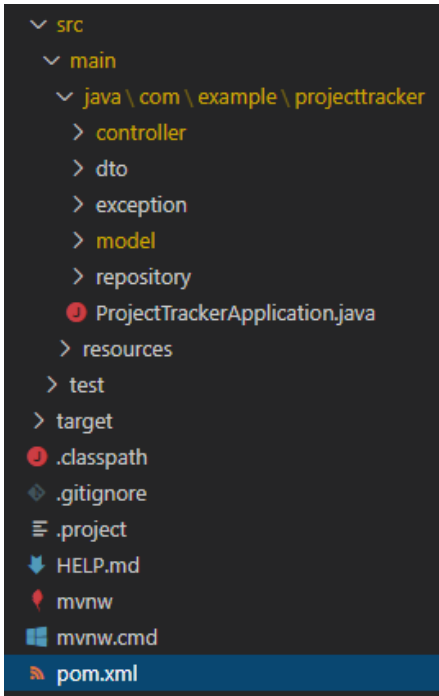


Figure 21 Back-End File Structure

5.2. User Interface

The frontend hierarchy starts with standard Angular compliant structure and then is divided into sub packages for the different GUIs. The sub packages for the presentation layer are broken into the following sections:

- about
- create-project
- faq
- home
- login
- newadmin
- project-details
- project-list
- update-project

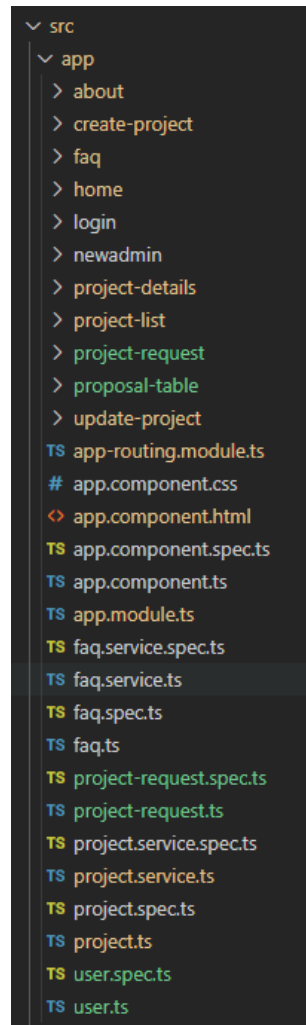


Figure 22 Front-End File Structure

6. Technical Positions

The design rationale shows the reasoning and argument behind making decisions when designing a system. The design intends to achieve the goal that the designer has in mind to fulfill the required function. The Spring framework is selected to build the CaPPMS web application since it addresses most of the infrastructure functionalities of enterprise applications. We are using PostgreSQL as a database. Using a spring boot framework with PostgreSQL gives us the following advantages:

- Spring boot allows developers to develop enterprise applications using POJOs (Plain Old Java Object) which eliminates the need for an enterprise container such as an

application server and Enterprise Java Beans while giving the option of using a robust servlet container. In short, no need to use complex EJB (Enterprise Java Beans) which drastically simplifies development and maintenance and the overall architecture

- Spring framework provides an abstraction layer on existing technologies like servlets, jsp, jdbc, etc. to simplify the design, development, and maintenance processes.
- Spring has existing technologies like the ORM framework which facilitates an easy link to a database without any special data access code. Hibernate can be used as a JPA provider in this case.
- Spring Web MVC framework offers a well-designed web MVC framework compared to the legacy web framework. The MVC design pattern promotes a separation of concerns among User Interface, Business Logic and Data layer.
- Spring application also can be used for the development of different types of applications, like standalone applications, standalone GUI applications, and Web applications.
- Using a RESTful API provides a great deal of flexibility. Data is not tied to resources or methods, so REST can handle multiple types of calls, return different data formats.
- PostgreSQL database offers robustness to full SQL compliance to the architecture while keeping the solution cost-effective.
- Spring Boot and Angular form a powerful tandem that works great for developing web applications with a minimal footprint. In this project, we will use Angular for creating a JavaScript-based frontend.

7. Coding Standards

The CaPPMS development will use the industry naming and coding standards listed in the table below.

Table 2 Coding Standards

Standard	URL
Google Java Style Guidelines	https://google.github.io/styleguide/javaguide.html

8. Known Issues

Any issues will be raised and tracked on GitHub. The issues are located at:

<https://github.com/umgc/umgc.idea.tracker/issues>.

9. References

Assadullah, M. (2020). SWEN 670 9040 Software Engineering Project (2208). Retrieved from <https://learn.umgc.edu/d2l/le/content/515759/viewContent/18965508/View>