**Software Requirements and Design Document**

**For**

**Group 5**

Version 2.0

**Authors**:

Adrian Balbuena

Evan Foglia

Raghav Gundavarapu

Prachi Kapur

Javier Roque

# Overview

*Give a general overview of the system in 1-2 paragraphs (similar to the one in the project proposal).*

The system is a CRUD application that allows users to enter specific data, which will be read and stored by the system. The data stored from users may be requested and displayed in a table, with functionality allowing for filtering, sorting, and querying the table. Users may also make changes to the data they have input, which includes deleting the submitted data. The system requires users to login to perform all of these actions.

# Functional Requirements

*List the* ***functional requirements*** *in sentences identified by numbers and for each requirement state if it is of high, medium, or low priority. Each functional requirement is something that the system shall do. Include all the details required such that there can be no misinterpretations of the requirements when read. Be very specific about what the system needs to do (not how, just what). You may provide a brief design rationale for any requirement which you feel requires explanation for how and/or why the requirement was derived.*

1. *Allow users to login and/or register in a secure fashion; high priority*
2. *Send and retrieve data to and from the backend; high priority*
3. *Create a reactive form when users input data to give feedback on the input; high priority*
4. *Include a home page to welcome users and an info page to inform users about the application and COVID-19; medium priority*
5. *Store the data collected from users in an SQL database; medium priority*
6. *Allow users to edit information while viewing the table; low priority*

# Non-functional Requirements

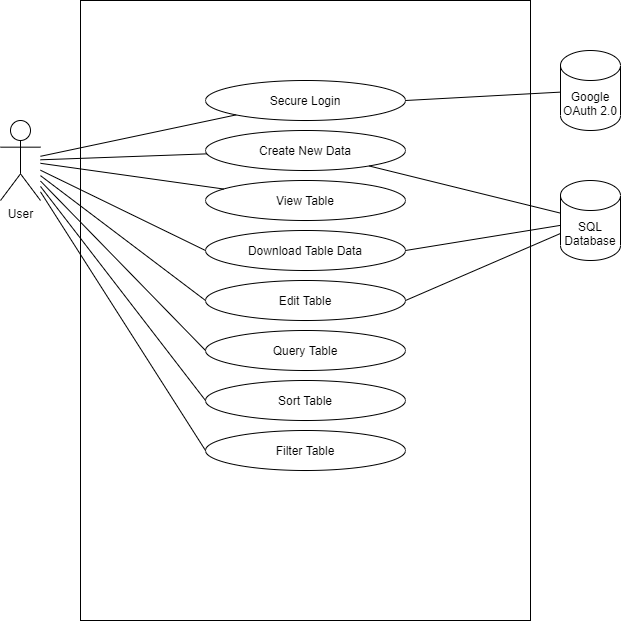
*List the* ***non-functional requirements*** *of the system (any requirement referring to a property of the system, such as security, safety, software quality, performance, reliability, etc.) You may provide a brief rationale for any requirement which you feel requires explanation as to how and/or why the requirement was derived.*

1. *Security: users must log in to access the application, and their login information will be safely secured in Google Firebase; by logging in, it can be determined whether a user is tampering with the data*
2. *Accessibility: the application should be capable of running on any device; to do this, the code will be containerized*
3. *Reliability: the data collected must be reliable, as data that is either biased or poorly given may produce invalid results when analyzed*
4. *Regulatory: users may report a collection of data to prevent others from tampering with it, so the data may be reverted to a previous version or deleted if it was maliciously created to begin with*

# Use Case Diagram

*This section presents the* ***use case diagram*** *and the* ***textual descriptions*** *of the use cases for the system under development. The use case diagram should contain all the use cases and relationships between them needed to describe the functionality to be developed. If you discover new use cases between two increments, update the diagram for your future increments.*

***Textual descriptions of use cases****: For the first increment, the textual descriptions for the use cases are not required. However, the textual descriptions for all use cases discovered for your system are required for the second and third iterations.*

**

* Secure Login: The user must log into the application to access any of its functionality. A login is verified by the account information stored in Google Firebase.
* Create New Data: The user fills out the Angular reactive form to input data, which is stored in the SQL database.
* Delete Data: The user may delete data that they have added to the database.
* View Table: The user may view the data collected in a table.
* Download Table Data: The user may request to download all the table’s data, which is provided by the SQL database.
* Edit Table: The user may edit the data found in the table, which is updated in the database (though the former data is kept in case the new data must be reverted to an older version).
* Query Table: The user may search for specific individuals in the data.
* Sort Table: The user may sort all the queried data by a certain variable.
* Filter Table: The user may remove data from the query based on certain characteristics.

# Class Diagram and/or Sequence Diagrams

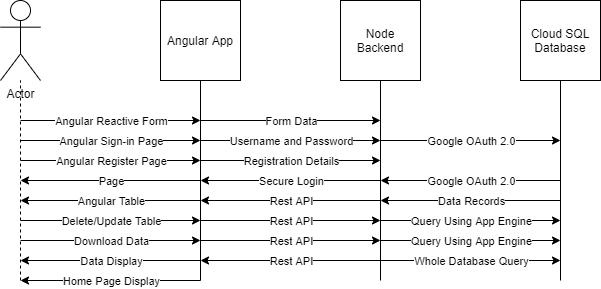
*This section presents a high-level overview of the anticipated system architecture using a* ***class******diagram*** *and/or* ***sequence diagrams****.*

*If the main* ***paradigm*** *used in your project is* ***Object Oriented*** *(i.e., you have classes or something that acts similar to classes in your system), then draw the* ***Class Diagram******of the entire system and Sequence Diagrams for the three (3) most important use cases in your system.***

*If the main* ***paradigm*** *in your system is* ***not Object Oriented*** *(i.e., you* ***do not*** *have classes**or anything similar to classes in your system) then only draw* ***Sequence Diagrams****,* ***but for all the use cases of your system.*** *In this case, we will use a modified version of Sequence Diagrams, where instead of objects, the lifelines will represent the functions in the system involved in the action sequence.*

***Class Diagrams*** *show the* ***fundamental objects/classes*** *that must be modeled with the system to satisfy its requirements and* ***the relationships*** *between them. Each class rectangle on the diagram* ***must also include the attributes and the methods of the class*** *(they can be refined between increments). All the* ***relationships between classes and their multiplicity*** *must be shown on the class diagram.*

*A* ***Sequence Diagram*** *simply depicts* ***interaction******between objects*** *(or* ***functions -*** *in our case - for non-OOP systems) in a sequential order, i.e. the order in which these interactions take place. Sequence diagrams describe how and in what order the objects in a system function.*

**

# Operating Environment

*Describe the environment in which the software will operate, including the hardware platform, operating system and versions, and any other software components or applications with which it must peacefully coexist.*

*As a web application, the software can run on any device capable of accessing the web. The application should work on modern browsers, but not necessarily on older ones that are incompatible with the features of Angular 9. The application will run using npm (Node Package Manager), and Node will be used as the application’s backend.*

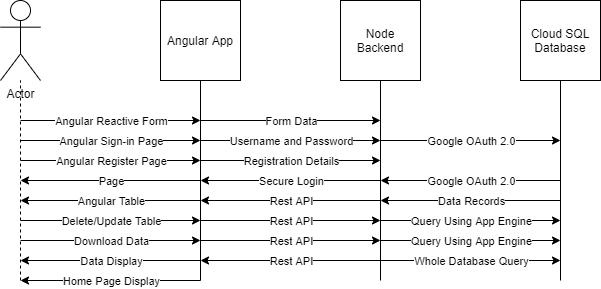
# Assumptions and Dependencies

*List any assumed factors (as opposed to known facts) that could affect the requirements stated in this document. These could include third-party or commercial components that you plan to use, issues around the development or operating environment, or constraints. The project could be affected if these assumptions are incorrect, are not shared, or change. Also identify any dependencies the project has on external factors, such as software components that you intend to reuse from another project.*

Assuming client access to Google Firebase.

Assuming Google Cloud Platform (GCP) will be on free trial.

Assuming forms are reactive and not template driven.

Assuming backend will have increased functionality with Angular.

Assuming SQL database is normalized.