Mission: Computer Science Question Input and Delivery Template Manager (C-SQID-TM)

DID: Software Requirements Specification (SRS)

PREPARED by: Team Two(
Charles Varga
Ryan Appleby
Amrithya Balasubramanian
Chris DeVoe
Danada Amalage Don
Aleya Mayo
)

DATE: 12-DEC-2020

REVISION LOG

REVISION	BY	DESCRIPTION	DATE
a	Team	Initial version	13-OCT-2020
b	Ryan Appleby	Requirements Update	12-DEC-2020

1. Introduction **4**
1.1 Purpose 4
1.2 Scope 4
1.3 Definitions, acronyms, and abbreviations 4
1.4 References 4
1.5 Overview 4

2. Overall Description **4**
2.1 Product perspective 4
2.2 Product functions 5
2.3 User characteristics 6
2.4 Constraints 6
2.5 Assumptions and dependencies 7

3. Specific Requirements **7**
3.1 External Requirements 7
3.1.1 User interfaces 7
3.1.2 Hardware Interfaces 7
3.1.3 Software Interfaces 7
3.1.4 Communications interfaces 7
3.2 Functional Requirements 7
3.3 Performance Requirements 8
3.4 Configuration Requirements 8
3.5 Security Requirements 8
3.6 Error Handling Requirements 9
3.7 Qualification Provisions 9

4. Appendices **10**

5. Index **10**

1. Introduction

1.1 Purpose

This Software Requirement Specification (SRS) pertains to the development of the Computer Science Question Input and Delivery Template Manager (C-SQID-TM) application. This document contains the requirements of the application, as well as any constraints and assumptions made to the development to said application. The intended audience includes the engineers tasked with developing the application and its database as well as the test conductors responsible for the verification and acceptance of the application.

1.2 Scope

The C-SQID-TM application can interface with the backed C-SQID-TM database to store, retrieve, and update the information within the database. The user can manually enter information to both store and retrieve information. The user may also submit a local document to be parsed and stored in the database. In addition, the application allows the user to input values into retrieved questions and save said questions locally without storing them in the database, and will allow the user to export this output to a separate file. The primary user for this application would be Computer Science professors and teaching assistants.

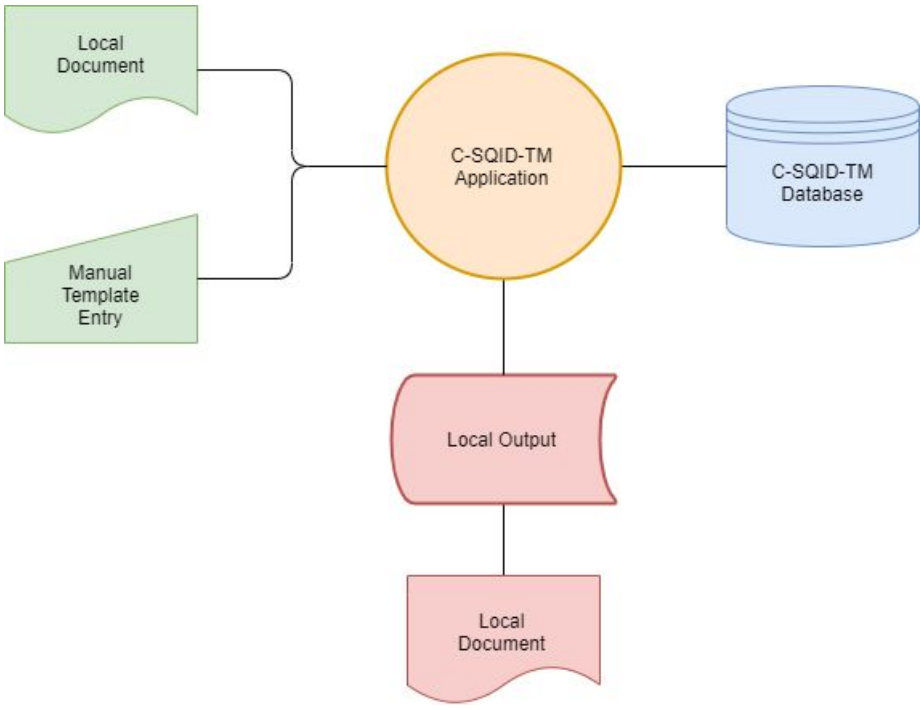

Figure 1. C-SQID-TM Interface

The software outlined in this SRS is responsible for interfacing with the user's local machine as well as the C-SQID-TM database in order to store and retrieve information. The software's interface diagram is shown in Figure 1.

1.3 Definitions, acronyms, and abbreviations

C-SQID-TM	Computer Science Question Input and Delivery Template Manager
TA	Teaching Assistant
SQL	Structured Query Language

1.4 References

IEEE Std 830-1998, Recommended Practice for Software Requirements Specifications.

1.5 Overview

The rest of the SRS is organized as follows: Section 2 describes the project as a whole. This section includes the purpose of the project (2.1), its capabilities (2.2), the persons authorized to utilize the application (2.3), the constraints that dictate the development of the project within certain operational limits (2.4), and the assumptions and dependencies of the application (2.5). Specific requirements for the application to operate are described in section 3. Requirements include external (3.1), functional (3.2), performance (3.3), configuration (3.4), security (3.5), and error handling (3.6), as well as a qualification matrix (3.7).

2. Overall Description

2.1 Product perspective

The main purpose of the C-SQID-TM is to provide professors at UMBC a web application that generates homework and test questions and stores them in a database. Inside the database is a variety of previously used (or new) question templates given by professors to be used to populate new, slightly different questions. This web application is similar to a product called LeetCode, which is a website that allows people to practice coding problems ranked based on level of difficulty.

The web application shall be implemented with the help of Flask. Flask provides a framework that is useful in terms of building an efficient and easily maintainable web application. This Python API connects to HTML files which in return allows the user to view the front end of the web app.

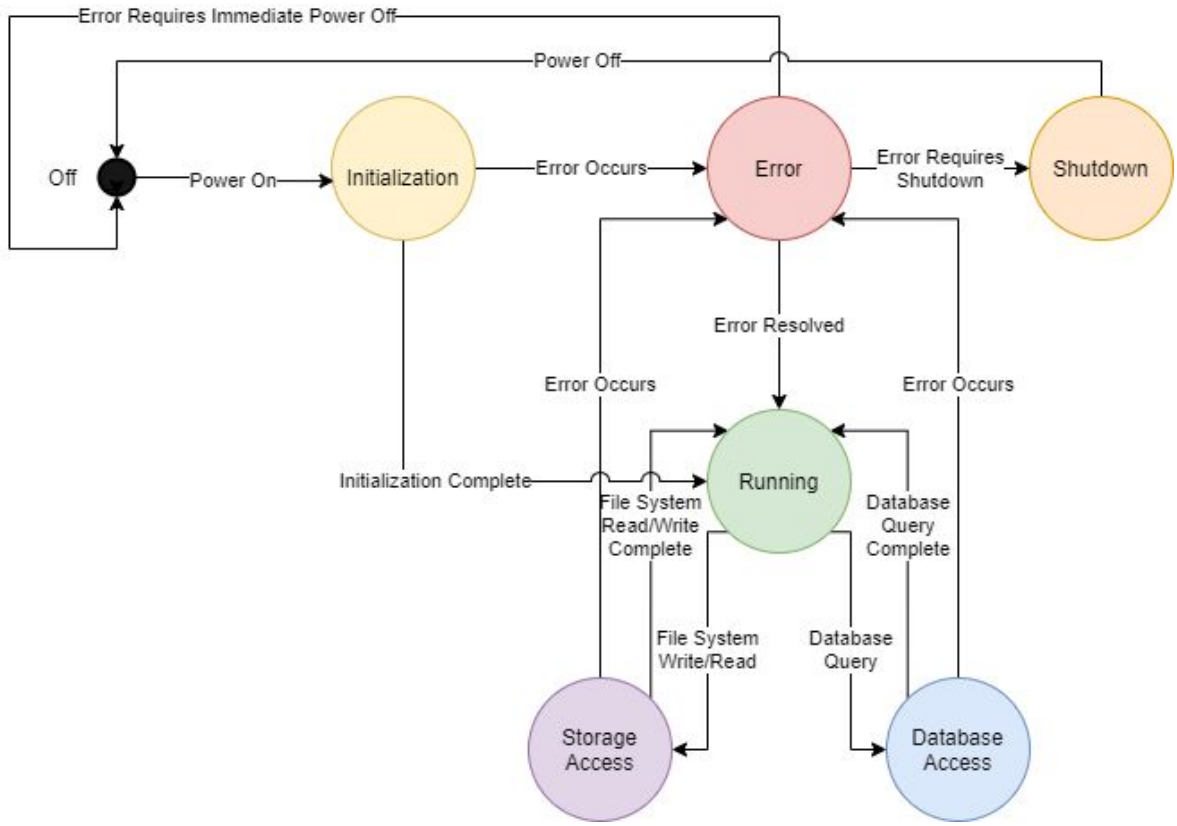

Figure 2. C-SQID-TM State Diagram

The user (a UMBC professor or admin) shall be able to make queries from the web application to connect to the backend built in Flask; which in return, sends the query to a database that holds the vast variety of questions.

2.2 Product functions

For the project different users must have different levels of permissions, this would help to keep out inappropriate or low-quality question templates from the database. An example would be that a professor of the university must have more access to the applications database and functions than an authorized user.

In the application itself a professor or TA shall be able to search and access the database of question templates through the project's web application to see the different types, constraints and difficulties of the searched templated questions. The search of the database shall show the user the different types of templated questions, the different possible constraints the templated question might have on the students as well as a difficulty to allow the user to see how difficult the templated question would be.

The commands that can be issued and the types/tags for the search are summarized in Table 1 and 2 respectively.

Table 1. Available Commands relevant to the C-SQID-TM

Command	Description
Search Templates	Would allow for searching of the database
Access Template	Would allow for accessing the question template
Submit Template	Would allow for the submission of question templates
Save Template	Would allow for the downloading of question templates
Edit Template	Would allow for the editing of question templates
Delete Template	Would allow for the deletion of a question template
Fill Template	Would allow for the filling out of a question template

Table 2. Available Types/Tags relevant to the C-SQID-TM

Command	Description
User Type	Admin / Professor / TA / Guest
Template Difficulty	# representing the difficulty of the templated question
Template Type	String(s) representing the type(s) of the templated question
Template Constraints	String(s) representing the possible constraints on the problem.

2.3 User characteristics

There are a few different types of users that will be interacting with the system. There will be the professor or TA, a guest and the administrators. Below are bullets points that go into more detail on what each type of user is expected to do.

- A Professor and TA shall be able to use the search functions to find templated questions. They should also be able to fill out, download and upload new templated questions to the database.

- An Admin shall have access to upload, edit, download, fill out and delete templates from the database. They also have the ability to directly access the backend database, add new users, and perform database maintenance tasks.

- A Guest shall not have access to the database or front-end software beyond the initial login screens at all.

2.4 Constraints

The purpose of this web application is to provide a cross-platform, online interface to upload, pull, and manage question bank templates from a centralized source database. There are limitations and user commitments regarding the usability of the templates produced. The constraint factors are summarized below:

- An uploaded template shall require users to manually fill in blanks within the question banks. Since the product is not meant to be a means of version control, multiple users/professors cannot edit the templates collaboratively.
- Students shall not be able to download templates or question banks uploaded by professors without the required permissions.
- The question difficulty classification feature (if implemented) may not provide accurate data sets as the application will not support question answering features within the application as of the current status of the project. Any performance metrics and/or feedback shall be self-reported by the students and may consequently prove biased.

Authentication and security constraints:

- Creation of administrative professor accounts cannot be strongly validated by the application. Professors/instructors shall be added by other pre-authorized staff. The application does not support security features to implement academic identity verification through formal means.
- Even if students do not have share or download permissions, question bank templates and questions can still leak. Professors shall manually enforce academic integrity for sensitive documents and incorporate appropriate protocols to conduct secure release of the documents if this application will be used for examination purposes.

Proprietary control:

- Professors may be limited in the way they can use question banks/templates committed to the system by the management setting the committing professor/staff has set. Their submission shall be treated as intellectual property and their proprietary rights will be protected by limiting the use of their submitted template by other professors according to whatever terms of use conditions are set as constraints.

2.5 Assumptions and dependencies

This application is built as a web application using the Flask framework and stylistic scripting languages such as HTML and CSS. As such, not many dependencies are present from the user end other than an internet connection. This includes assuming that users have an adequate web browser. A Google account/myUMBC login info is also required for user authentication. The application will also require a connection with a remote database, either structured like SQL or unstructured like NoSQL. The exact database platform to be used shall be determined further in the design process.

3. Specific Requirements

3.1 External Requirements

Set forth is the description of the external scope of the C-SQID-TM application; Documenting its interactions, both software and hardware based on anything extra to the internal application design.

3.1.1 User interfaces

The application does not have its own graphical user interface. The user interaction with the application will take place through a user's web browser, interfacing with the C-SQID-TM's HTML web pages, the Flask API, and React. Through the browser, the user will be able to query our database, receive and upload question templates, and prepare templates for assignments.

3.1.2 Hardware Interfaces

The software will interface with the user's own device. The software will be able to access the user's disk storage when requested, allowing the user to either upload a properly formatted list of new question templates to be added to the database, or download a set of templates to their device for later use.

3.1.3 Software Interfaces

The software interfaces with the user's web browser in order to display both HTML and JavaScript content. In addition, our software interfaces with database software that will allow administrators to manage and maintain the database through an external tool.

3.1.4 Communications interfaces

The software does not have any external communication interfaces.

3.2 Functional Requirements

3.2.1 CST_F_1: The application shall store question templates in a database for later access and use.

3.2.2 CST_F_2: The application shall allow the user to query the database to retrieve question templates.

3.2.3 CST_F_3: The application shall categorize question templates to allow the user to query the database for a specific type of question.

3.2.4 CST_F_4: The application shall provide a way to store question difficulty for each question template.

3.2.5 CST_F_5: The application shall allow the user to upload a file of question templates, specifically .txt files, that the software will then parse through and add each question to the database.

3.2.6 CST_F_6: The application shall allow the user to save question templates to their devices.

3.2.7 CST_F_7: The application shall allow the user to fill out question templates with data.

3.2.8 CST_F_8: The application shall provide a way to manually enter a new question template in a text box and add it to the database.

3.2.9 CST_F_9: The application shall allow the user to update information for question templates already added to the database.

3.2.10 CST_F_10: The application shall recommend restrictions for assignments, based on the type of question chosen, in an effort to prevent students from using concepts that may make the problem trivial in lower-level courses.

3.3 Performance Requirements

3.3.1 CST_P_1: Application shall be able to perform main tasks such template retrieval and database management without letting the application hang for more than ten seconds on an average internet connection.

3.3.2 CST_P_2: Application shall access the correct user’s choice of template and let the user make appropriate changes without changing the template document.

3.4 Configuration Requirements

3.4.1 CST_C_1: The web application shall connect to the database automatically whenever the user accesses the web interface.

3.5 Security Requirements

3.5.1 CST_S_1: The application shall possess user authentication capabilities to maintain the confidentiality of the question templates written to the interface.

3.5.2 CST_S_2: The application shall transmit user credentials between the client and the server in a manner that preserves confidentiality thereof.

3.5.3 CST_S_3: Availability of the question template management and test creation features of the application shall be reserved to professors and TAs employed at the receiving university.

3.6 Error Handling Requirements

3.6.1 CST_E_1: The application shall operate continuously, notwithstanding attempted denials of service.

3.6.2 CST_E_2: The application shall continue to accept requests for question template retrieval in the event of an internal error.

3.6.3 CST_E_3: The application shall continue to allow question template uploads in the event of an internal error.

3.6.4 CST_E_4: The application shall dispose of unused memory in a secure manner that prevents leakage.

3.6.5 CST_E_5: The application shall raise exceptions in the event of errors that compromise confidentiality, integrity, or availability.

3.7 Qualification Provisions

The verification methods used to determine whether a requirement has been met are as follows:

- Inspection - Requirement satisfied via an exterior examination of the software.
- Demonstration - Requirement satisfied via user interaction with the software.
- Test - Requirement satisfied via a controlled test of the software given pre-defined inputs.
- Analysis - Requirement satisfied through the use of simulations and modelling.
- Special qualification methods - Any special qualification methods not listed above.

Requirement	Inspection	Demonstration	Test	Analysis	Special
CST_F_1			X		
CST_F_2			X		
CST_F_3	X				
CST_F_4	X				
CST_F_5			X		

CST_F_6			X		
CST_F_7			X		
CST_F_8			X		
CST_F_9			X		
CST_F_10			X		
CST_P_1		X			
CST_P_2			X		
CST_C_1		X			
CST_S_1		X			
CST_S_2		X			
CST_S_3			X		
CST_E_1			X		
CST_E_2			X		
CST_E_3			X		
CST_E_4			X		
CST_E_5			X		