## 1. Introduction

# 1.1 Purpose of this Document

The purpose of this Software Requirements Document (SRD) is to outline the structure of the UBCcards flashcard project for the S17 COSC 448 Directed Studies course under Dr. Patricia Lasserre. Client-oriented requirements describe the system from the client’s perspective. These requirements include a description of the different types of users served by the system. Developer-oriented requirements describe the system from a software developer’s perspective. These requirements include a detailed description of functional, data, performance, and other important requirements.

# 1.2 Scope of the Development Project

The scope of the UBCcards application includes its distinct features, its benefits, and limitations. Its distinct features are:

* multi-platform access on a unified code base
* centralized resource of study information gathered through crowdsourcing
* tools for users to moderate the information

It is intended to provide an efficient comprehensive study tool to students at UBC by:

* providing a simple and portable web application, as smartphones are ubiquitous with students
* reducing individual burden for producing a comprehensive set of flashcards

However, UBCcards will not:

* provide the study content for the application
* moderate the information contained
* be responsible for inaccuracy of the content

# 1.3 Definitions, Acronyms, and Abbreviations

|  |  |
| --- | --- |
| Term | Definition. Acronym, Abbreviation |
| JavaScript | The programming language used in the UBCcards project |
| MEAN Stack | A full stack package consisting of MongoDB, Express, Angular and Node. |
| MongoDB | A NoSQL database program |
| ExpressJS | A web application framework for Node.js |
| AngularJS | A front-end web application framework |
| Node.js | A JavaScript runtime environment for executing server side code |
| REST | REpresentational State Transfer, an architectural style for approaching communications in web services |

# 1.4 References

Austin, V. (n.d.). *SoftwareRequirementsDocument* [Doc]. Austin: University of Texas. http://web.stonehill.edu/compsci/CS400/SoftwareRequirementsDocument.doc

# 1.5 Overview of Document

Section one contains:

* a general introduction of the purpose and scope of the document
* definition of terms used in the document
* references used in the document

Section two contains:

* a client-oriented view of the system
* personas and characteristics of users who will interact with the system
* a high-level description of functional, data, and non-functional requirements
* a set of use cases describing how users will interact with the system

Section three contains:

* a software developer-oriented view of the system
* a detailed description of functional requirements including input, processing and output for each major feature
* a detailed description of data requirements
* a detailed description of non-functional requirements

Section four contains:

* a timeline of expected deliverables

## 2. User-Oriented Requirements

# 2.1 User Personas

Student (Steve): Steve is a first-year biology student with heavy time restraints. He has a full course load and a part-time job. He is comfortable with technology, and always has a smartphone and laptop with him. He wants to do well in school, but has difficulty managing time.

Teacher (Dr. Tom): Dr. Tom is a chemistry instructor at UBC Okanagan. He is passionate for beakers and chemistry-related things, and enjoys when students are engaged with the content. He prefers pen and paper over computers, but is willing to try new things.

Administrator (Ted): Ted is an upper level math student with a programming background. He volunteered to maintain the UBCcards application for the Qualitative Sciences Course Union at UBC Okanagan. He is adept with computers, and has no trouble debugging the code of others.

# 2.2 Product Interfaces

UBCcards is a digital form of the traditional flash card. Instead of carrying hundreds of individual cards, they will be digitally managed and accessible to many users.

The UBCcards application interfaces with several hardware and software components:

* Smartphone: Android, iOS, Blackberry, or any other mobile device that supports a JavaScript-compliant browser
* Computer: Windows, OSX, Linux, or any other OS that supports a JavaScript-compliant browser
* Wifi/Mobile Data: This will be used by the application to communicate with the server
* Server: A Linux-based server that runs the MEAN stack.

# 2.3 Functional Requirements

UBCcards will improve the study experience for students by meeting the following functional requirements, indicated as required (R), desired (D), or optional (O):

* FR0(R): The system will allow students to add flashcards to the database. This will contain the question, answer, and tags relevant for searching.
* FR1(R): The system will allow students to navigate through the list of cards. Sets of cards can selected for a particular class, and further refined by search terms.
* FR2(D): The system will allow students to flag incorrect or inappropriate flash cards, and provide tools to modify the content.
* FR3(D): The system will allow students to rate the flash card for content. Sets of cards can be refined based on the rating of cards.
* FR4(O): The system will allow the student to login and view their submitted cards, and track their activity.
* FR5(O): The system will allow teachers to add courses to the application, approve users for the course, and track their usage.
* FR6(O): The system will include gamification concepts to encourage student usage. Give points for adding cards or modifying cards, and level up or provide badges as incentives.
* FR7(O): The system will allow for guest usage options, which allows the user to only view the cards, but not create, edit, rate or flag.
* FR8(D): The system will allow users to roll back to a previous version of the flashcard, in case of vandalism.
* FR9(D): The system will count the number of flags raised on a flashcard, and remove it from the library when 3 people have flagged it.
* FR10(D): The system will list the cards that have been flagged for review, so the information can be corrected. The user who submitted or edited the card will also be displayed.
* FR11(D): The system will allow administrators to remove users and delete all of their content.

# 2.4 Data Requirements

UBCcards will improve the study experience for students by meeting the following data requirements, indicated as required (R), desired (D), or optional (O):

* DR1(R): Student → UBCcards: The user will provide the question, answer, and search tags that define a flashcard.
* DR2(R): UBCcards → Student: The application will serve a set of flashcards as specified by the user’s parameters (i.e. course or search term).
* DR3(D): Student → UBCcards: The user will provide a username and password to access their individual account.
* DR4(O): Teacher → UBCcards: The user is able to create new courses, and can decide whether it will be open, or if he needs to approve the students.
* DR5(O): Server → Device database: The server will provide an offline copy of the database for the user’s device. This is limited in size due to web standards.

# 2.5 Non-functional Requirements

UBCcards will improve the study experience for students by meeting the following non-functional requirements, indicated as required (R), desired (D), or optional (O):

* NFR1(R): Accommodate devices with varying screen sizes. Though, a minimum size will be required.
* NFR2(D): Minimize the data throughput will serving the flashcard content
* NFR3(R): Protect user information as per any relevant privacy acts
* NFR4(R): Not access any restricted information unless given express permission

# 2.6 Use Cases

Steve is waiting for the bus, and wants to review a test later that day. He takes out his smartphone and runs the UBCcards application. The application loads, and shows him the list of classes he’s selected. He picks Biology 225, and the flashcards load. After moving through a few cards, he remembers that the test is only on Chapter 5. He goes to the search bar, and types in ‘Chapter 5”. A set of cards containing only those from Chapter 5 are displayed.

Steve has just finished class for Math 225, and he wants to add in some flashcards to UBCcards. He goes to the computer in the library, and loads up the application website. Once loaded, he logs in, then selects Math 225 from the list of courses. He clicks the button to create a card, and inputs in the necessary information – question, answer, and search tags. Now finished, he clicks submit, and the flash card is added to the library.

Steve is flipping through some flashcards outside his next class, while waiting for the previous class to exit. He notices that one of the flashcards is not correct – Jupiter is a planet, not a plant. He clicks the button to edit the card, and fixed the mistake. He saves the card, and continues studying.

Steve is on the bus, stuck in traffic due to Kelowna’s ridiculous road construction. He notices that a flashcard is full of profanity. He is studying Math, not English, so the profanity is definitely inappropriate. He clicks the button to flag the flashcard, and selects ‘inappropriate’. He will no longer see the card while studying.

Dr. Tom wants to add his course to the UBCcards application. He logs into his account, and clicks the button to create a new course. He inputs the course name, and selects the box for selective join, the clicks submit. He tells his class to sign up for the application, and he will approve them once they try to join. He marks them off the class list as they join, so he can appropriately add on participation marks.

Ted has noticed a user posting all inappropriate flashcards. He pulls up the user’s profile, and can choose to either remove the cards manually, or remove the user and all their associated cards. He decides to remove the user.

## 3. Developer-Oriented Requirements

# 3.1 Functional Requirements

UBCcards will implement the application by meeting the following data requirements, indicated as required (R), desired (D), or optional (O):

* FR0(R): The system will allow students to add flashcards to the database. This will contain the question, answer, and tags relevant for searching.

|  |  |
| --- | --- |
| Inputs | The flashcard will be entered through a user interface in the application. The data will conform to the following information:  (Question, String), (Answer, String), (Tags, String), (Rating, int), (Flags, int), (Creator, String) |
| Processing | The input will be restricted to the normal character set, and will be sanitize before processing. The data will be input into the appropriate table in the database |
| Outputs | A Boolean indicating the success of the operation |

* FR1(R): The system will allow students to navigate through the list of cards. Sets of cards can selected for a particular class, and further refined by search terms.

|  |  |
| --- | --- |
| Inputs | The user will select a class, and input search terms (if desired). The input will be restricted to the normal character set, and will be sanitize before processed. A call is made to the database with the class and search terms (if given) |
| Processing | The database will search the database for cards in that class, and have text matching to the search terms (if given). |
| Outputs | The cards that fulfill the search terms |

* FR2(D): The system will allow students to flag incorrect or inappropriate flash cards, and provide tools to modify the content.

|  |  |
| --- | --- |
| Inputs |  |
| Processing | The data will be input into the appropriate table in the database |
| Outputs | A Boolean indicating the success of the operation |

* FR3(D): The system will allow students to rate the flash card for content. Sets of cards can be refined based on the rating of cards.

|  |  |
| --- | --- |
| Inputs | The user will click on a rating for the card, and input search terms (if desired). The input will be restricted to the normal character set, and will be sanitize before processed. |
| Processing | A call is made to the database with the class and search terms (if given). The database will search the database for cards in that class, and have text matching to the search terms (if given). |
| Outputs | The cards that fulfill the search terms |

# 3.2 Data Requirements

UBCcards will implement the application by meeting the following data requirements, indicated as required (R), desired (D), or optional (O):

* DR1(R): The system will include a database schema that will contain the different types of users (student, teacher, and administrator), the flashcard information, and the associated classes.
* DR2(R): The system will conform to all expected internet protocols to ensure standard communication.
* DR3(R): The system will use open source and accessible libraries to ensure the potential of future expansion

# 3.3 User Interface Requirements

UBCcards will implement the application by meeting the following user interface requirements, indicated as required (R), desired (D), or optional (O):

* UIR1(R): The system will conform to the standards of modern browsers to ensure communication and compatibility
* UIR2(R): The system will display materials and interactive sections in a manner expected for their role.

# 3.4 Hardware Interface Requirements

UBCcards will implement the application by meeting the following hardware interface requirements, indicated as required (R), desired (D), or optional (O):

* HIR1(R): The system will run on a UBC-hosted server
* HIR2(R): The system will require continuous internet access
* HIR3(R): The system will need sufficient CPU and memory resources to handle many simultaneous users
* HIR4(R): The system will require sufficient storage space for the flashcards and associated data.

# 3.5 Software Interface Requirements

UBCcards will implement the application by meeting the following software interface requirements, indicated as required (R), desired (D), or optional (O):

* SIR1(R): The system will implement the MEAN stack (MongoDB, ExpressJS, AngularJS and Node.js) for communication between the server and user
* SIR2(R): The system will need to interface with any JavaScript-compliant browser.
* SIR3(R): The system requires a server with a standard Linux installation
* SIR4(D): The system will be deployed as a packaged webapp to the respective app stores of Android and iOS
* SIR5(R): The system must sanitize inputs to prevent malicious code from executing

# 3.6 Performance Requirements

UBCcards will implement the application by meeting the following performance requirements, indicated as required (R), desired (D), or optional (O):

* PR1(R): The system will need to allow multiple simultaneous users, and allow for future expansion.
* PR2(R): The system will minimize the information being transferred to reduce the bandwidth of the server and user, and minimize the response times

## 4. Timeline and deliverables

|  |  |
| --- | --- |
| Date | Deliverables |
| Friday, July 7 | * Draft project plan for review with Dr. Lasserre |
| Monday, July 10 | * Updated project plan based on feedback * Setup ZenHub for Agile project management |
| Monday, July 17 | * Paper prototypes for mobile and desktop * Deploy stack onto server and implement basic functionality (Node.js and Express - requests/responses) * Create unit tests for basic server functions * Database schema drawn up |
| Monday, July 24 | * Implement database schema, handlers and API (MongoDB) * Develop RESTful API for communications * Create unit tests for bulk of back end |
| Monday, July 31 | * Develop Angular MVC front end * Integrate the front and back end * Have a working prototype of the application |
| Monday, August 7 | * Add graphics and design to the UI * Implement any additional features * Compile the final report |
| Thursday, August 10 | * Submit final report |