Software Requirements Specification

for

SchMate

Prepared by Team Blackcat

3rd February 2022

Table of Contents

Table of Contents ii
Revision History ii
1. Introduction 1
1.1 Purpose 1
1.2 Document Conventions 1
1.3 Intended Audience and Reading Suggestions 1
1.4 Product Scope 1
1.5 References 1
2. Overall Description 2
2.1 Product Perspective 2
2.2 Product Functions 2
2.3 User Classes and Characteristics 2
2.4 Operating Environment 2
2.5 Design and Implementation Constraints 2
2.6 User Documentation 2
2.7 Assumptions and Dependencies 3
3. External Interface Requirements 3
3.1 User Interfaces 3
3.2 Hardware Interfaces 3
3.3 Software Interfaces 3
3.4 Communications Interfaces 3
4. System Features 4
4.1 System Feature 1 4
4.2 System Feature 2 (and so on) 4
5. Other Nonfunctional Requirements 4
5.1 Performance Requirements 4
5.2 Safety Requirements 5
5.3 Security Requirements 5
5.4 Software Quality Attributes 5
5.5 Business Rules 5
6. Other Requirements 5
Appendix A: Glossary 5
Appendix B: Analysis Models 5
Appendix C: To Be Determined List 6

Revision History

Name	Date	Reason For Changes	Version

1. Introduction

1.1 Purpose

<Identify the product whose software requirements are specified in this document, including the revision or release number. Describe the scope of the product that is covered by this SRS, particularly if this SRS describes only part of the system or a single subsystem.>

Choosing a university course can be a painstaking process and making the right decision can potentially dictate one's future path.

Our web application SchMate aims to ease the process by assisting students in making more informed decisions for their higher education, more specifically their undergraduate studies. This is achieved through creating a one stop portal for students to find out about the schools and courses that are related to their interests. Students can also do a side by side comparison for the courses that they are interested in which will help them in making a decision.

1.2 Document Conventions

<Describe any standards or typographical conventions that were followed when writing this SRS, such as fonts or highlighting that have special significance. For example, state whether priorities for higher-level requirements are assumed to be inherited by detailed requirements, or whether every requirement statement is to have its own priority.>

This document will generally be in Times New Roman font in normal text, size 12. The headings will be in Times font, size 14.

In the stimulus/response sequence under the system features, the users and systems are underlined to show the distinction and transitions between user and system responses.

1.3 Intended Audience and Reading Suggestions

<Describe the different types of reader that the document is intended for, such as developers, project managers, marketing staff, users, testers, and documentation writers. Describe what the rest of this SRS contains and how it is organized. Suggest a sequence for reading the document, beginning with the overview sections and proceeding through the sections that are most pertinent to each reader type.>

This document is intended for project managers and developers. This will give them an idea of the whole project and its various aspects. The document has 5 major components, namely Introduction, Overall Description, External Interface Requirements, System Features and Other NonFunctional Requirements. The user can read all the components in order.

1.4 Product Scope

<Provide a short description of the software being specified and its purpose, including relevant benefits, objectives, and goals. Relate the software to corporate goals or business strategies. If a separate vision and scope document is available, refer to it rather than duplicating its contents here.>

SchMate allows users to find out about potential courses that might interest them based on their inputs. Users can then select up to 3 courses and compare them side by side to make a better and more informed decision.

1.5 References

<List any other documents or Web addresses to which this SRS refers. These may include user interface style guides, contracts, standards, system requirements specifications, use case documents, or a vision and scope document. Provide enough information so that the reader could access a copy of each reference, including title, author, version number, date, and source or location.>

2. Overall Description

2.1 Product Perspective

<Describe the context and origin of the product being specified in this SRS. For example, state whether this product is a follow-on member of a product family, a replacement for certain existing systems, or a new, self-contained product. If the SRS defines a component of a larger system, relate the requirements of the larger system to the functionality of this software and identify interfaces between the two. A simple diagram that shows the major components of the overall system, subsystem interconnections, and external interfaces can be helpful.>

Currently, students go to the official university websites to look for courses that they are interested in before applying. There are some websites that show general information about the school or just hyperlinks that lead users to the official university site. There has not any easy way for users to make comparisons between schools and courses.

SchMate is a follow-on member of such websites and aims to cover this gap in university hunting. This is done by implementing more useful functions such as including a comparison interface which aligns features of the schools selected in a side by side view.

2.2 Product Functions

<Summarize the major functions the product must perform or must let the user perform. Details will be provided in Section 3, so only a high level summary (such as a bullet list) is needed here. Organize the functions to make them understandable to any reader of the SRS. A picture of the major groups of related requirements and how they relate, such as a top level data flow diagram or object class diagram, is often effective.>

2.2.1 Course Catalogue feature: Users can browse through the entire list of courses and schools.
2.2.2 Select Interests: Users can select the various field that interests them. The system will recommend the users a list of courses that are relevant to their inputs.
2.2.3 Filter and Sorter: Filter and sort the courses based on the options given.
2.2.4 Course Comparison: Users can select a few courses and compare them side by side
2.2.5 Course and School Display: System displays more specific information about the course ans school.
2.2.6 Contact Team form: Users can submit their queries or suggestions to the developers via the Contact Team form.

2.3 User Classes and Characteristics

<Identify the various user classes that you anticipate will use this product. User classes may be differentiated based on frequency of use, subset of product functions used, technical expertise, security or privilege levels, educational level, or experience. Describe the pertinent characteristics of each user class. Certain requirements may pertain only to certain user classes. Distinguish the most important user classes for this product from those who are less important to satisfy.>

2.3.1 Graduated or graduating junior college students (High priority). Occasional to Frequent user of the Course and School Displayer and Comparison feature.
2.3.2 Graduated or graduating polytechnic students who wish to further their studies (High priority). Occasional to Frequent user of the Course and School Displayer and Comparison feature.
2.3.3 Parents of the above-mentioned students (Medium priority). Occasional to Frequent user of the Course and School Displayer.

2.4 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.>

The web application will operate on web browsers such as Google Chrome, Mozilla FireFox, Microsoft Edge and Safari.

2.5 Design and Implementation Constraints

<Describe any items or issues that will limit the options available to the developers. These might include: corporate or regulatory policies; hardware limitations (timing requirements, memory requirements); interfaces to other applications; specific technologies, tools, and databases to be used; parallel operations; language requirements; communications protocols; security considerations; design conventions or programming standards (for example, if the customer's organization will be responsible for maintaining the delivered software).>

2.5.1 Users should have a computer device.
2.5.2 The computer devices are installed with the common web browsers (see 2.4).
2.5.3 Users must have internet access to visit the website.
2.5.4 The website will be made available in English
2.5.5 The website is in Singapore's context.

2.6 User Documentation

<List the user documentation components (such as user manuals, on-line help, and tutorials) that will be delivered along with the software. Identify any known user documentation delivery formats or standards.>

2.7 Assumptions and Dependencies

<List any assumed factors (as opposed to known facts) that could affect the requirements stated in the SRS. 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, unless they are already documented elsewhere (for example, in the vision and scope document or the project plan).>

Assumptions:

1. Users are in Singapore.
2. Users are able to read and understand English.
3. Users have an internet connection.
4. Users have a computer device.
5. Users have the compatible web browsers installed and working.
6. The information extracted from the government dataset and the universities websites are accurate and up to date.

Dependencies:

1. SchMate depends on the external API data set provided by the government.
2. SchMate depends on SQLite database which will be stored on a cloud server.

3. External Interface Requirements

3.1 User Interfaces

<Describe the logical characteristics of each interface between the software product and the users. This may include sample screen images, any GUI standards or product family style guides that are to be followed, screen layout constraints, standard buttons and functions (e.g., help) that will appear on every screen, keyboard shortcuts, error message display standards, and so on. Define the software components for which a user interface is needed. Details of the user interface design should be documented in a separate user interface specification.>

SchMate will have a navigation bar at the top of the website for all web pages. It will serve as a guide for the user to know which tab they are on and aid in navigating to other tabs of the website. Users will interact with buttons and drop-down lists to utilise functions within the web page.

3.2 Hardware Interfaces

<Describe the logical and physical characteristics of each interface between the software product and the hardware components of the system. This may include the supported device types, the nature of the data and control interactions between the software and the hardware, and communication protocols to be used.>

SchMate will require a device that is connected to the internet for it to retrieve data from APIs. The device must also be able to support desktop web browsers.

3.3 Software Interfaces

<Describe the connections between this product and other specific software components (name and version), including databases, operating systems, tools, libraries, and integrated commercial components. Identify the data items or messages coming into the system and going out and describe the purpose of each. Describe the services needed and the nature of communications. Refer to documents that describe detailed application programming interface protocols. Identify data that will be shared across software components. If the data sharing mechanism must be implemented in a specific way (for example, use of a global data area in a multitasking operating system), specify this as an implementation constraint.>

SchMate is designed to work with devices that can support desktop web browsers. It is designed to work with SQLite as the database, React as the library for the user interface, Node.js as the runtime environment to run JavaScript outside of a browser, and Express.js as the back end web framework for Node.js.

3.4 Communications Interfaces

<Describe the requirements associated with any communications functions required by this product, including e-mail, web browser, network server communications protocols, electronic forms, and so on. Define any pertinent message formatting. Identify any communication standards that will be used, such as FTP or HTTP. Specify any communication security or encryption issues, data transfer rates, and synchronization mechanisms.>

3.4.1 The device will need to be connected to the internet to make API calls to retrieve information to display on the web page.
3.4.2 SchMate will need to send the information from the Contact Team form to the relevant university's email through Simple Mail Transfer Protocol (SMTP), Post Office Protocol (POP), or Internet Message Access Protocol (IMAP).
3.4.3 The communication standard to be used is HTTP.

4. System Features

<This template illustrates organizing the functional requirements for the product by system features, the major services provided by the product. You may prefer to organize this section by use case, mode of operation, user class, object class, functional hierarchy, or combinations of these, whatever makes the most logical sense for your product.>

4.1 Welcome Page (System Feature 1)

<Don't really say "System Feature 1." State the feature name in just a few words.>

4.1.1 Description and Priority

<Provide a short description of the feature and indicate whether it is of High, Medium, or Low priority. You could also include specific priority component ratings, such as benefit, penalty, cost, and risk (each rated on a relative scale from a low of 1 to a high of 9).

It shows the logo and name of our website as well as a short description. As we scroll down, more information about the website and the team will be shown. It has a "Let's Get Started" button for users to navigate to the Interests page (see 4.3). It also contains the navigation bar. It has High priority.

4.1.2 Stimulus/Response Sequences

<List the sequences of user actions and system responses that stimulate the behaviour defined for this feature. These will correspond to the dialog elements associated with use cases.>

Let's Get Started Button

1. User clicks on Let's Get Started button.

2. System redirects the user to the Select Interests webpage which displays the different areas of interest.

4.1.3 Functional Requirements

<Itemize the detailed functional requirements associated with this feature. These are the software capabilities that must be present in order for the user to carry out the services provided by the feature, or to execute the use case. Include how the product should respond to anticipated error conditions or invalid inputs. Requirements should be concise, complete, unambiguous, verifiable, and necessary. Use "TBD" as a placeholder to indicate when necessary information is not yet available.>

<Each requirement should be uniquely identified with a sequence number or a meaningful tag of some kind.>

4.1.3.1 The system must display the welcome message.
4.1.3.2 The system must display the "Let's Get Started" button.
4.1.3.2.1 The user must be able to click on the "Let's Get Started" button.
4.1.3.3 The user must be able to scroll down to view more information about the website.
4.1.3.3.1 The system must show the team member's profile at the bottom of the page.

4.2 Navigation Bar (System Feature 2)

<Don't really say "System Feature 2." State the feature name in just a few words.>

4.2.1 Description and Priority

<Provide a short description of the feature and indicate whether it is of High, Medium, or Low priority. You could also include specific priority component ratings, such as benefit, penalty, cost, and risk (each rated on a relative scale from a low of 1 to a high of 9).>

The navigation bar allows users to navigate to the various functions provided by the website. The functionalities include Home, Select Interests, Course Catalogue and Contact Team. The navigation bar will also be present at the top of all webpages. This has high/low priority.

4.2.2 Stimulus/Response Sequences

<List the sequences of user actions and system responses that stimulate the behavior defined for this feature. These will correspond to the dialog elements associated with use cases.>

Home

1. User clicks on the Home button.
2. System redirects the user to the home page of the website.

Interests Button

1. User clicks on Select Interests button.
2. System redirects the user to the Select Interests webpage which displays the different areas of interest.

Course Catalogue Button

1. User clicks on Course Catalogue Button.
2. System redirect users to Course Catalogue webpage which displays the relevant courses and schools based on the filter applied.

Contact Team Button

1. User clicks on Contact Team Button.
2. System redirect users to Contact Team webpage which provides users with the contact information of the website administrators.

4.2.3 Functional Requirements

<Itemize the detailed functional requirements associated with this feature. These are the software capabilities that must be present in order for the user to carry out the services provided by the feature, or to execute the use case. Include how the product should respond to anticipated error conditions or invalid inputs. Requirements should be concise, complete, unambiguous, verifiable, and necessary. Use "TBD" as a placeholder to indicate when necessary information is not yet available.>

<Each requirement should be uniquely identified with a sequence number or a meaningful tag of some kind.>

4.2.3.1 The system must be able to display the navigation bar that leads the user to the other functionalities.

4.2.3.2 The system must allow users to click on the functionalities.

4.2.3.3 The system must redirect users to the respective functions that they have clicked.

4.3 Select Interests (System Feature 3)

<Don't really say "System Feature 3." State the feature name in just a few words.>

4.3.1 Description and Priority

<Provide a short description of the feature and indicate whether it is of High, Medium, or Low priority. You could also include specific priority component ratings, such as benefit, penalty, cost, and risk (each rated on a relative scale from a low of 1 to a high of 9).>

The Select Interests feature will display a page of potential areas of interest users might have such as Technology, Literature and Engineering, in the form of a card. Hovering the mouse over the card will display intriguing information about the course category and courses related to it. Users can select course categories. Based on the user's inputs, the system will recommend a list of courses and schools in the Course Catalogue (see 4.4).

4.3.2 Stimulus/Response Sequences

<List the sequences of user actions and system responses that stimulate the behavior defined for this feature. These will correspond to the dialog elements associated with use cases.>

User chooses areas of interest

1. User hovers the mouse over the card that interests them.
2. System displays intriguing information regarding the interest as well as courses related.
3. User clicks on the cards that they are interested in.
4. System applies a dark filter/tick over the cards to indicate that the card has been selected.
5. User is satisfied with the selection and clicks the "Recommend Course(s)" button.
6. System redirects user to Course Catalogue with applied filter according to the selected cards.

4.3.3 Functional Requirements

<Itemize the detailed functional requirements associated with this feature. These are the software capabilities that must be present in order for the user to carry out the services provided by the feature, or to execute the use case. Include how the product should respond to anticipated error conditions or invalid inputs. Requirements should be concise, complete, unambiguous, verifiable, and necessary. Use "TBD" as a placeholder to indicate when necessary information is not yet available.>

<Each requirement should be uniquely identified with a sequence number or a meaningful tag of some kind.>

4.3.3.1 The system must display all the course categories.
4.3.3.2 When the user hovers the mouse on the card, the card must display the relevant information.
4.3.3.3 The system must apply a dark filter over the card when the user clicks on the card.

4.3.3.3.1 Similarly, the system has to undo the dark filter when the card is clicked a second time.

4.3.3.4 If user does not select any course categories and tries to click on "Show courses" button, the system will display error message "Minimum of 1 course category needs to be selected".

4.3.3.5 The system must be able to redirect the user to the Course Catalog when "Recommend Course" button is clicked.

4.3.3.5.1 The system must already apply the appropriate filter according to the card selection.

4.4 Course Catalogue (System Feature 4)

<Don't really say "System Feature 4." State the feature name in just a few words.>

4.4.1 Description and Priority

<Provide a short description of the feature and indicate whether it is of High, Medium, or Low priority. You could also include specific priority component ratings, such as benefit, penalty, cost, and risk (each rated on a relative scale from a low of 1 to a high of 9).>

This feature allows users to view the various courses that are a match for their interests, selected in the "Select Interest" feature. It has High Priority.

4.4.2 Stimulus/Response Sequences

<List the sequences of user actions and system responses that stimulate the behaviour defined for this feature. These will correspond to the dialog elements associated with use cases.>

"Interested" Checkbox

1. User checks the "Interested" checkbox.
2. System stores this selection into a list of "Interested" courses.
3. System displays the checkbox as checked.

More Details

1. User clicks more details, which is displayed beside each entry in the catalogue.
2. System redirects the user to the respective course's webpage which provides more information regarding the course (number of years of study, annual school fees, etc).

Compare Courses

1. User clicks Compare Courses.
2. System redirects the user to the Course Comparison page, which shows side-by-side comparison of the user's "interested" courses.

4.4.3 Functional Requirements

<Itemize the detailed functional requirements associated with this feature. These are the software capabilities that must be present in order for the user to carry out the services provided by the feature, or to execute the use case. Include how the product should respond to anticipated error conditions or invalid inputs. Requirements should be concise, complete, unambiguous, verifiable, and necessary. Use "TBD" as a placeholder to indicate when necessary information is not yet available.>

<Each requirement should be uniquely identified with a sequence number or a meaningful tag of some kind.>

4.4.3.1 The system must display all courses related to user's "Select Interested" topic.
4.4.3.2 The checkbox must be an interactive element.
4.4.3.3 The checkbox should change from empty to ticked when a user clicks on it.
4.4.3.4 The checkbox should change from ticked to empty when a user clicks on it.
4.4.3.5 If user selects less than 2 courses and tries to click on "Compare courses" button, the system will display error message "Minimum of 2 courses need to be selected".
4.4.3.6 If user selects more than 3 courses and tries to click on "Compare courses" button, the system will display error message "Maximum of 3 courses can be selected".
4.4.3.7 "more details" must be an interactive element.
4.4.3.8 "more details" must redirect user to the correct course details webpage.
4.4.3.9 "Compare Courses" must redirect user to the Course comparison webpage.
4.4.3.10 System must correctly keep track of user's "interested" courses (up to 6).
4.4.3.11 System must give user a warning message when user selects more than 6 "interested" courses.
4.4.3.12 The system must be able to fetch the API from the government data successfully.
4.4.3.13 The system must be able to query the database successfully.

4.5 Catalogue Filter (System Feature 5)

<Don't really say "System Feature 5." State the feature name in just a few words.>

4.5.1 Description and Priority

<Provide a short description of the feature and indicate whether it is of High, Medium, or Low priority. You could also include specific priority component ratings, such as benefit, penalty, cost, and risk (each rated on a relative scale from a low of 1 to a high of 9).>

This feature allows users to filter the courses that have been listed to them in the Course Catalogue after they have selected their desired course according to metrics such as location. It has Medium Priority.

4.5.2 Stimulus/Response Sequences

<List the sequences of user actions and system responses that stimulate the behavior defined for this feature. These will correspond to the dialog elements associated with use cases.>

User selects intended filter

1. User clicks on the drop-down list of a specific filter. (eg. Location, Price Range)
2. System displays options in the drop-down list.
3. User clicks on the filter in the drop-down list. (eg. Central, East, West, North)
4. System records selected filter.
5. User clicks on the "Filter" button.
6. System queries database to find and display the corresponding results on Course Catalogue.

4.5.3 Functional Requirements

<Itemize the detailed functional requirements associated with this feature. These are the software capabilities that must be present in order for the user to carry out the services provided by the feature, or to execute the use case. Include how the product should respond to anticipated error conditions or invalid inputs. Requirements should be concise, complete, unambiguous, verifiable, and necessary. Use "TBD" as a placeholder to indicate when necessary information is not yet available.>

<Each requirement should be uniquely identified with a sequence number or a meaningful tag of some kind.>

4.5.3.1 The system must display the filter at the top of the webpage.
4.5.3.2 The user must be able to click on/give an input for each of the filter options.
4.5.3.3 The system must be able to filter the courses and schools based on the user's inputs.
4.5.3.4 The system must be able to query the database successfully.

4.6 Catalogue Sorter (System Feature 6)

<Don't really say "System Feature 6." State the feature name in just a few words.>

4.6.1 Description and Priority

<Provide a short description of the feature and indicate whether it is of High, Medium, or Low priority. You could also include specific priority component ratings, such as benefit, penalty, cost, and risk (each rated on a relative scale from a low of 1 to a high of 9).>

This feature allows users to sort the courses that have been listed to them in the Course Catalogue after they have selected their desired course according to metrics such as school fees. It has Medium Priority.

4.6.2 Stimulus/Response Sequences

<List the sequences of user actions and system responses that stimulate the behavior defined for this feature. These will correspond to the dialog elements associated with use cases.>

User selects intended sorter

1. User clicks on the criteria drop-down list. (eg. School Fees)
2. System displays options in the drop-down list.
3. User clicks on the criteria in the drop-down list.
4. System records selected criteria for sorting.
5. User clicks on the "Sort" button.
6. System queries database to sort the courses on Course Catalogue.

4.6.3 Functional Requirements

<Itemize the detailed functional requirements associated with this feature. These are the software capabilities that must be present in order for the user to carry out the services provided by the feature, or to execute the use case. Include how the product should respond to anticipated error conditions or invalid inputs. Requirements should be concise, complete, unambiguous, verifiable, and necessary. Use "TBD" as a placeholder to indicate when necessary information is not yet available.>

<Each requirement should be uniquely identified with a sequence number or a meaningful tag of some kind.>

4.6.3.1 The system must display the sorter at the top of the webpage.
4.6.3.2 The user must be able to click on/give an input for each of the sorting options.
4.6.3.3 The system must be able to sort the courses and schools based on the user's inputs.
4.6.3.4 The system must be able to query the database successfully.

4.7 Comparison Between Courses and Schools (System Feature 7)

<Don't really say "System Feature 7." State the feature name in just a few words.>

4.7.1 Description and Priority

<Provide a short description of the feature and indicate whether it is of High, Medium, or Low priority. You could also include specific priority component ratings, such as benefit, penalty, cost, and risk (each rated on a relative scale from a low of 1 to a high of 9).>

This feature allows users to compare the various courses that they have selected in the Course Catalogue. It provides a "table" view where the selected courses are displayed side

by side. Users must choose a minimum of 2 courses for comparison and can only select a maximum of 3 courses. It has High Priority.

4.7.2 Stimulus/Response Sequences

<List the sequences of user actions and system responses that stimulate the behavior defined for this feature. These will correspond to the dialog elements associated with use cases.>

Comparing Courses

1. System queries the database and fetches from API to display selected courses information.

Compare More Courses

1. User clicks on the "Compare more courses" button to select more courses for comparison.
2. System redirects the user to the "Course Catalogue" page.

4.7.3 Functional Requirements

<Itemize the detailed functional requirements associated with this feature. These are the software capabilities that must be present in order for the user to carry out the services provided by the feature, or to execute the use case. Include how the product should respond to anticipated error conditions or invalid inputs. Requirements should be concise, complete, unambiguous, verifiable, and necessary. Use "TBD" as a placeholder to indicate when necessary information is not yet available.>

<Each requirement should be uniquely identified with a sequence number or a meaningful tag of some kind.>

4.7.3.1 The system must display the courses selected in the "Course Catalogue" feature side-by-side.

4.7.3.1.1 The features must be aligned horizontally to allow for easy comparison for the user.

4.7.3.2 The system must display the key information of the courses selected, such as QS World University Ranking, Course Duration, and Annual School Fees.

4.7.3.3 The user must be able to click on a "Compare more courses" button to select more courses for comparison.

4.7.3.4 The system must be able to fetch the API from the government data successfully.

4.7.3.5 The system must be able to query the database successfully.

4.8 Courses and Schools Displayer (System Feature 8)

<Don't really say "System Feature 8." State the feature name in just a few words.>

4.8.1 Description and Priority

<Provide a short description of the feature and indicate whether it is of High, Medium, or Low priority. You could also include specific priority component ratings, such as benefit, penalty, cost, and risk (each rated on a relative scale from a low of 1 to a high of 9).>

This feature allows users to learn more about the course selected from the "Course Catalogue" feature. It has Medium Priority.

4.8.2 Stimulus/Response Sequences

<List the sequences of user actions and system responses that stimulate the behavior defined for this feature. These will correspond to the dialog elements associated with use cases.>

Course Displayer

1. System queries the database and fetches the API to display detailed information about the selected course, such as the school, course curriculum, annual school fees, and QS World University Ranking.
2. User clicks on the "View other courses" button.
3. System redirects the user to the "Course Catalogue" page.

4.8.3 Functional Requirements

<Itemize the detailed functional requirements associated with this feature. These are the software capabilities that must be present in order for the user to carry out the services provided by the feature, or to execute the use case. Include how the product should respond to anticipated error conditions or invalid inputs. Requirements should be concise, complete, unambiguous, verifiable, and necessary. Use "TBD" as a placeholder to indicate when necessary information is not yet available.>

<Each requirement should be uniquely identified with a sequence number or a meaningful tag of some kind.>

4.8.3.1 The system must display detailed information about the course as selected from the "Course Catalogue" page.

4.8.3.2 The user must be able to click on the "View other courses" button to get more details about another course.

4.8.3.3 The system must be able to fetch the API from the government data successfully.

4.8.3.4 The system must be able to query the database successfully.

4.9 Contact Team (System Feature 9)

<Don't really say "System Feature 9." State the feature name in just a few words.>

4.9.1 Description and Priority

<Provide a short description of the feature and indicate whether it is of High, Medium, or Low priority. You could also include specific priority component ratings, such as benefit, penalty, cost, and risk (each rated on a relative scale from a low of 1 to a high of 9).>

This feature allows users to contact the development team to report bugs with the system, or give feedback and suggestions on improving it.

4.9.2 Stimulus/Response Sequences

<List the sequences of user actions and system responses that stimulate the behavior defined for this feature. These will correspond to the dialog elements associated with use cases.>

Contact Team Page

1. System displays a page with an input box for users' to input feedback/suggestions.
2. User types in feedback into the input box and contact details, then clicks the "Submit" button.
3. System sends feedback and contact details to the development team, then displays a message to indicate the successful delivery of feedback.

4.9.3 Functional Requirements

<Itemize the detailed functional requirements associated with this feature. These are the software capabilities that must be present in order for the user to carry out the services provided by the feature, or to execute the use case. Include how the product should respond to anticipated error conditions or invalid inputs. Requirements should be concise, complete, unambiguous, verifiable, and necessary. Use "TBD" as a placeholder to indicate when necessary information is not yet available.>

<Each requirement should be uniquely identified with a sequence number or a meaningful tag of some kind.>

4.9.3.1 The system must display a page with an input box for users to type their feedback and contact details.

4.9.3.2 The user must be able to click on the "Submit" button to submit their feedback to the development team.

4.9.3.3 The system must display a message to indicate success upon submission of user feedback.

5. Other Nonfunctional Requirements

5.1 Performance Requirements

<If there are performance requirements for the product under various circumstances, state them here and explain their rationale, to help the developers understand the intent and make suitable design choices. Specify the timing relationships for real time systems. Make such requirements as specific as possible. You may need to state performance requirements for individual functional requirements or features.>

5.1.1 The system must load and display the website within 15 seconds when the user accesses it.
5.1.2 The system must be able to navigate to other pages of functionalities within 2 seconds.
5.1.3 The system must have little to no downtime.

5.2 Safety Requirements

<Specify those requirements that are concerned with possible loss, damage, or harm that could result from the use of the product. Define any safeguards or actions that must be taken, as well as actions that must be prevented. Refer to any external policies or regulations that state safety issues that affect the product's design or use. Define any safety certifications that must be satisfied.>

5.2.1 The system must be able to have an option to turn the website's contents to "Dark Mode", to allow users to reduce the strain on their eyes from long hours of computer usage.

5.3 Security Requirements

<Specify any requirements regarding security or privacy issues surrounding use of the product or protection of the data used or created by the product. Define any user identity authentication requirements. Refer to any external policies or regulations containing security issues that affect the product. Define any security or privacy certifications that must be satisfied.>

5.3.1 The submission of the Contact Team form must be secure to prevent any user emails from being leaked.
5.3.2 The database containing the email addresses of users who send the Contact Team form must also be secure.

5.4 Software Quality Attributes

<Specify any additional quality characteristics for the product that will be important to either the customers or the developers. Some to consider are: adaptability, availability, correctness, flexibility,

interoperability, maintainability, portability, reliability, reusability, robustness, testability, and usability. Write these to be specific, quantitative, and verifiable when possible. At the least, clarify the relative preferences for various attributes, such as ease of use over ease of learning.>

5.4.1 Correctness: The system must be regularly updated with the most recent data of the University Courses and GES survey to ensure accuracy of information for the users.
5.4.2 Usability: SchMate is available in simple English and will be foolproof even for completely new users.

5.5 Business Rules

<List any operating principles about the product, such as which individuals or roles can perform which functions under specific circumstances. These are not functional requirements in themselves, but they may imply certain functional requirements to enforce the rules.>

5.5.2 Developers are to manage the website.

6. Other Requirements

<Define any other requirements not covered elsewhere in the SRS. This might include database requirements, internationalization requirements, legal requirements, reuse objectives for the project, and so on. Add any new sections that are pertinent to the project.>

Appendix A: Glossary

<Define all the terms necessary to properly interpret the SRS, including acronyms and abbreviations. You may wish to build a separate glossary that spans multiple projects or the entire organization, and just include terms specific to a single project in each SRS.>

Appendix B: Analysis Models

<Optionally, include any pertinent analysis models, such as data flow diagrams, class diagrams, state-transition diagrams, or entity-relationship diagrams.>

Appendix C: To Be Determined List

<Collect a numbered list of the TBD (to be determined) references that remain in the SRS so they can be tracked to closure.>

Source: http://www.frontiernet.net/~kwiegers/process_assets/srs_template.doc