Software Requirements Specification

for

URI Schedule Maker

Version 1.0 approved

Prepared by Brendan Chadwick, Sean Creamer, Baheem Ferrell, Brian Hopkins, and Paul Perry

Team 1

May 27, 2020

Table of Contents

Revision History

Name	Date	Reason For Changes	Version
Team 1	5/30/2020	Initial Creation of Document	1.0.0
Team 1	6/26/2020	Combined our Stimulus/Response	1.1.0
Team 1	7/10/2020	Included Functional Requirements	1.2.0

1. Introduction

Purpose

This is a document detailing the specifics of a schedule making utility for University of Rhode Island (URI) students. This document is intended for the use of software developers working on the utility, URI systems administrators and URI students looking for information on how the utility works.

1.1 Document Conventions

1.2 Project Scope

This schedule making utility will be a tool available to all URI students. Currently URI students must build a schedule class by class by browsing the catalog for individual courses and there is no schedule builder incorporated into the e-campus system students must use for selecting classes. This utility will enable users to input their major, areas of interests and what classes they have already completed, and it will suggest to them a schedule for the upcoming semester based upon this information.

1.3 References

URI API Documentation - *https://api.uri.edu/#/docs/gettingstarted*

2. Overall Description

2.1 Product Perspective

The utility will be an entirely new software tool available to the students of URI. The utility will be a standalone desktop application. It will interact with URI's e-campus system through the API provided by the university for development of apps to interact with e-campus. The utility will interact with the e-campus system to gain access to the university's course catalog and to get information about what classes a student has already completed.

2.2 User Classes and Characteristics

The only intended user class for this software utility is URI students. The utility will function exclusively with URI's e-campus system and with a database of only URI's major requirements and as such will not have much use to any groups other than students at URI. URI students are technologically well-versed and almost all have access to their own private PC's.

2.3 Operating Environment

This utility will be a standalone desktop app and because it will be written in the Python programming language it will be able to work on all user PC's. The only other application with which the software must interact is URI's e-campus system.

2.4 Design and Implementation Constraints

Going forward there are two major concerns that may pose possible constraints. Firstly, there is the issue of URI's policies and regulation, it may be difficult to gain access to E-campus and any other internal data for obvious privacy and security reasons. Secondly, assuming that access is granted to the data there is the issue of combining it, considering the universities size.

2.5 Assumptions and Dependencies

We assume that the utility will be able to interact with the e-campus system and pull the course catalog from the upcoming semester. We also assume that through the use of the e-campus API our utility will be able to pull information about individual students which may not be true.

The utility will depend on the Python programming language and the availability of the API that URI has published for development of apps which interact with e-campus.

3. System Features

3.1 Course Selection by Major

3.1.1 Description

As a student, I need the suggestion to include the appropriate classes to my major of choice so that I don't enroll in courses, which have no meaning for my degree.

3.1.2 Stimulus/Response Sequences

3.1.2.1 Major

Stimulus: The student enters their major.
Response: The class finder shall suggest classes that make sense and scenario for selection.

3.1.2.2 Minor

Stimulus: The student enters their minor.
Response: The class finder shall suggest classes that make sense and scenario for selection.

3.1.3 Functional Requirements

3.1.3.1 The system must give the user a choice for all courses required by the major or minor.

3.1.3.2 Must update UI to bold possible classes available to take

3.1.3.3 The system must recommend certain classes that can fit in both major and minor, if possible.

3.2 Completed Courses

3.2.1 Description

As an upperclassman, I need to be able to exclude courses that I've already taken so that the schedule maker will not include those courses in its results. (High priority)

3.2.2 Stimulus/Response Sequences

Stimulus: The user checks off a class that has been previously taken
Response: The course/schedule maker will not include/consider that course in its results.

Stimulus: The user leaves a course unmarked
Response: The course/schedule maker will include/consider that course in its results.

3.2.3 Functional Requirements

3.2.3.1 The system must list all courses needed to graduate under whatever major is chosen

3.2.3.2 Must update UI to show which courses are marked as completed and which courses yet to be taken by the user.

3.2.3.3 Must be able to pass a course prerequisite check if the course is marked as completed

3.2.3.4 Must be able to save desired time slots for classes

3.3 User Login

3.3.1 Description

As a student, I need to be able to login and have the schedule generator load all of my academic information such as my past coursework, major, minor and preferences so that the generator can recommend a feasible schedule for me. (High priority).

3.3.2 Stimulus/Response Sequences

3.3.2.1 Returning User

Stimulus: The student launches the program
Response: The schedule generator shall greet the student with a welcome screen that prompts the user for their login information. Once entered, a check will be performed to make sure that the user entered valid information.

3.3.2.2 First Time User

Stimulus: The student launches the program, and selects the option new user
Response: The schedule generator prompts the student to fill out a form of preferences which include questions such as time of day for classes, maximum amount of credits they're comfortable taking, types of general education courses they're interested in, etc. The system then saves the users data for future use in its database.

3.3.3 Functional Requirements

3.3.3.1 The system must list all courses needed to graduate under whatever major is chosen

3.3.3.2 Must update UI to show which courses are marked as completed and which courses yet to be taken by the user.

3.3.3.3 Must be able to pass a course prerequisite check if the course is marked as completed

3.3.3.4 Must be able to save desired time slots for classes

3.4 Rate My Professor Integration

3.4.1 Description

The system must allow the user to view the professors ranking as well as comments on the professor from the website “Rate My Professor” (Medium priority)

3.4.2 Stimulus/Response Sequences

3.4.2.1 Selected “view professor” and there is a Rate My Professor page

Stimulus: User first clicked on a course and then clicked on “view professor” under the professor's name.
Response: The app will then redirect the user to a new window which will pull up the professor’s page on “Rate My Professor”.

3.4.2.2 **Selected “view professor” and there is not a Rate My Professor page**

Stimulus: User attempted to “view professor” however the website “Rate My Professor” does not have a listing of the professor.
Response: The system will prompt a small error indicating to the user that no results were found when attempting to search for the professor on the site.

3.4.3 Functional Requirements

3.4.3.1.1 The system must be able to allow the user to view a Professor.

3.4.3.1.2 The system must check whether or not the Professor has a listing on the site.

3.4.3.1.3 If there is a listing the system must be able to redirect the user to “Rate My Professor”.

3.4.3.1.4 If there is not a listing the system must be able to display that no results were found.

3.5 Switching of Majors

3.5.1 Description

The student chooses to pick another class and the system must allow the user to see a list of class which collides with the new updated major (High priority).

3.5.2 Stimulus/Response Sequences

3.5.2.1 Changing your major

Stimulus: The user requests a major change.
Response: App temporarily changes the requirements to what is necessary for the new request of requirements for classes taken and classes necessary to solve the new major requirements.

3.5.2.2 Changing your minor

Stimulus: The user requests a minor change.
Response: App temporarily changes the requirements to what is necessary for the new request of requirements for classes taken and classes necessary to solve the new minor requirements.

3.5.3 Functional Requirements

3.5.3.1 The system must be able to temporarily store the old requirements taken and print out the list if it counts towards a new major or minor.

4. Data Requirements

<This section describes various aspects of the data that the system will consume as inputs, process in some fashion, or create as outputs.>

4.1 Logical Data Model

<A data model is a visual representation of the data objects and collections the system will process and the relationships between them. Include a data model for the business operations being addressed by the system, or a logical representation for the data that the system itself will manipulate. Data models are most commonly created as an entity-relationship diagram.>

4.2 Data Dictionary

<The data dictionary defines the composition of data structures and the meaning, data type, length, format, and allowed values for the data elements that make up those structures. In many cases, you're better off storing the data dictionary as a separate artifact, rather than embedding it in the middle of an SRS. That also increases its reusability potential in other projects.>

4.3 Reports

<If your application will generate any reports, identify them here and describe their characteristics. If a report must conform to a specific predefined layout you can specify that here as a constraint,

perhaps with an example. Otherwise, focus on the logical descriptions of the report content, sort sequence, totaling levels, and so forth, deferring the detailed report layout to the design stage.>

4.4 Data Acquisition, Integrity, Retention, and Disposal

<If relevant, describe how data is acquired and maintained. State any requirements regarding the need to protect the integrity of the system's data. Identify any specific techniques that are necessary, such as backups, checkpointing, mirroring, or data accuracy verification. State policies the system must enforce for either retaining or disposing of data, including temporary data, metadata, residual data (such as deleted records), cached data, local copies, archives, and interim backups.>

5. External Interface Requirements

<This section provides information to ensure that the system will communicate properly with users and with external hardware or software elements.>

5.1 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.>

5.2 Software Interfaces

<Describe the connections between this product and other software components (identified by name and version), including other applications, databases, operating systems, tools, libraries, websites, and integrated commercial components. State the purpose, formats, and contents of the messages, data, and control values exchanged between the software components. Specify the mappings of input and output data between the systems and any translations that need to be made for the data to get from one system to the other. Describe the services needed by or from external software components and the nature of the intercomponent communications. Identify data that will be exchanged between or shared across software components. Specify nonfunctional requirements affecting the interface, such as service levels for responses times and frequencies, or security controls and restrictions.>

5.3 Hardware Interfaces

<Describe the characteristics of each interface between the software and hardware (if any) components of the system. This description might include the supported device types, the data and control interactions between the software and the hardware, and the communication protocols to be used. List the inputs and outputs, their formats, their valid values or ranges, and any timing issues developers need to be aware of. If this information is extensive, consider creating a separate interface specification document.>

5.4 Communications Interfaces

<State the requirements for any communication functions the product will use, including e-mail, Web browser, network protocols, and electronic forms. Define any pertinent message formatting. Specify communication security or encryption issues, data transfer rates, handshaking, and synchronization mechanisms. State any constraints around these interfaces, such as whether e-mail attachments are acceptable or not.>

6. Quality Attributes

6.1 Usability

<Specify any requirements regarding characteristics that will make the software appear to be "user-friendly." Usability encompasses ease of use, ease of learning; memorability; error avoidance, handling, and recovery; efficiency of interactions; accessibility; and ergonomics. Sometimes these can conflict with each other, as with ease of use over ease of learning. Indicate any user interface design standards or guidelines to which the application must conform.>

6.2 Performance

<State specific performance requirements for various system operations. If different functional requirements or features have different performance requirements, it's appropriate to specify those performance goals right with the corresponding functional requirements, rather than collecting them in this section.>

6.3 Security

<Specify any requirements regarding security or privacy issues that restrict access to or use of the product. These could refer to physical, data, or software security. Security requirements often originate in business rules, so identify any security or privacy policies or regulations to which the product must conform. If these are documented in a business rules repository, just refer to them.>

6.4 Safety

<Specify requirements that are concerned with possible loss, damage, or harm that could result from use of the product. Define any safeguards or actions that must be taken, as well as potentially

dangerous actions that must be prevented. Identify any safety certifications, policies, or regulations to which the product must conform.>

6.5 [Others as relevant]

<Create a separate section in the SRS for each additional product quality attribute to describe characteristics that will be important to either customers or developers. Possibilities include availability, efficiency, installability, integrity, interoperability, modifiability, portability, reliability, reusability, robustness, scalability, and verifiability. Write these to be specific, quantitative, and verifiable. Clarify the relative priorities for various attributes, such as security over performance.>

7. Internationalization and Localization Requirements

<Internationalization and localization requirements ensure that the product will be suitable for use in nations, cultures, and geographic locations other than those in which it was created. Such requirements might address differences in: currency; formatting of dates, numbers, addresses, and telephone numbers; language, including national spelling conventions within the same language (such as American versus British English), symbols used, and character sets; given name and family name order; time zones; international regulations and laws; cultural and political issues; paper sizes used; weights and measures; electrical voltages and plug shapes; and many others.>

8. Other Requirements

<Examples are: legal, regulatory or financial compliance, and standards requirements; requirements for product installation, configuration, startup, and shutdown; and logging, monitoring and audit trail requirements. Instead of just combining these all under "Other," add any new sections to the template that are pertinent to your project. Omit this section if all your requirements are accommodated in other sections. >

Appendix A: Glossary

URI – University of Rhode Island

Appendix B: Analysis Models

<This optional section includes or points to pertinent analysis models such as data flow diagrams, feature trees, state-transition diagrams, or entity-relationship diagrams. You might prefer to insert certain models into the relevant sections of the specification instead of collecting them at the end.>