Software Requirements Specification

for

CAR: Course Analysis and Registration System

Version 1.0

Prepared by Liyan Ibrahim, Juan Piñeros and Akshat Totla

CAR Incorporated

4th March 2022

Revision History

Name	Date	Reason For Changes	Version

1. Introduction

1.1 Purpose

The purpose of this document is to provide the system requirement specification for a course analysis and registration system. It covers the needs and functionalities of the course analysis and registration system in an easy to follow manner for the system's intended users. This will help the developers build an efficient system, a one-stop place for students for everything academic - to track their academic progress, create four year plans, and find new courses for upcoming semesters. This SRS specifies guidelines to develop this system with the aim of its pilot implementation in New York University Abu Dhabi, while keeping the system flexible to connect to other universities' databases.

1.2 Document Conventions

The document follows the IEEE conventions and formatting requirements.
Main Section Titles – Font: Times, Face: Bold, Size: 23
Sub Section Titles – Font: Times, Face: Bold, Size: 17
Text Explanations – Font: Times, Face: Normal, Size: 11

Acronyms and Abbreviations:

- NYUAD: New York University Abu Dhabi
- CAR: Course Analysis and Registration
- HTML: Hypertext Markup language
- HTTPS: Secure Hypertext Transfer Protocol
- HTTP: Hypertext Transfer Protocol
- GUI: Graphical User Interface
- MFA: Multi-Factor authentication

1.3 Intended Audience and Reading Suggestions

The intended client and end user of this system is university students looking to streamline their course planning, monitoring, and registration process, the pilot implementation being around the NYUAD course offering. Other stakeholders are university departments and administrators, given they will interact with course plans exported by the system, and will be able to update the required courses list from the system.

This SRS is directed towards developers, project managers, and testers. Sections 1 and 2 are useful for project managers to refine requirements and Section 2.2 is useful for testers to verify the completion of functional requirements. Section 4 is useful for developers to know the minute details of the system's implementation.

1.4 Product Scope

The software developed is aimed at allowing users to perform common tasks associated with course registration like tracking their academic progress, creating four year plans, finding new courses for upcoming semesters, and finding classes and professor ratings. It would provide a stable, convenient, and user-friendly application platform that is more beneficial compared to other softwares as it is a one-stop place for students for everything academic. This provides a medium for students to specialize their academic plans which directly relates to increasing student satisfaction and ease for finding courses while also saving hours of time students spend creating and updating their four year plans.

1.5 References

NYUAD existing course registration website (Albert): http://albert.nyu.edu/albert_index.html
Multi-Factor Authentication: https://www.cisa.gov/publication/multi-factor-authentication-mfa
Schedge Albert API https://github.com/A1Liu/schedge
Firebase SSO https://firebase.google.com/docs/auth
IEEE 830-1998 Recommended Practice for Software Requirements Specifications Template, Computer Society, 1998 – https://standards.ieee.org/ieee/830/1222/

2. Overall Description

2.1 Product Perspective

This product is a new system that aims to complement existing course registration systems by streamlining the course planning and pre-registration process, first by fulfilling the course planning function that is currently performed by an inconvenient mix of university bulletins, students' individual files, and diverse course planning templates from different departments, and additionally by providing a faster way to search for courses, their availability, and their instructors' ratings. To this purpose, the CAR system will be a web-based system that includes a link to the university course's API, and a database with course requirements and individual course progress.

2.2 Product Functions

Function Number	Function Name	Description
1	Course Suggestion	CAR allows the users to select their major and courses they have taken after logging in, and suggest courses they can take in the coming semesters to be on track to graduate
2	Professor Rating	Users can search for a class and get a link to the professor ratings if available
3	Location specific courses	Users shall be able to identify the semester location to look up a certain course
4	Class Status	Users shall be able to check if the desired class is waitlisted, open, closed or offered for a certain semester, and this information should be updated in real time
5	Study Plan	Users shall be able to get the study plan for their majors and check their process
6	Exporting Plan	Users shall be able to export their four year plans in a format usable by all university departments
7	Saving and Updating 4 Year Plan	Users shall be able to store four-year plans to work on them over different sessions
8	Verifying Classes	Users shall be able to verify whether their class choices for a given semester are valid in terms of prerequisites and time clashes
9	Degree Progress	Users shall be able to check their degree progress
10	Modifying Courses and Requirements	Administrators must be able to modify the course requirements of a degree program

2.3 User Classes and Characteristics

User	Limitations
Students	Can access their individual course schedule, four year plan, degree progress report and university-wide courses offered in any semester (with the corresponding information about it e.g. open, closed, requirement for which major). Most frequent user of the system with low-medium technical expertise, having low privilege levels but high priority
Administrators	Can access any student's course schedule, four year plan, degree progress report and university-wide courses offered in any semester. Administrators are also able to modify the course details, course requirements of a degree program and students' enrollment. Low frequency user of the system with high technical expertise assuming experienced registrars having previously used similar systems having high privilege levels but low priority

2.4 Operating Environment

The project will be a web application designed responsively in order to be accessed from all kinds of devices. As a result, it will have to operate on a server capable of high uptime. However, the volume of data to be managed isn't large, therefore the server's performance is not a concern. As well, the system must retrieve course offering and registration information from the Registrar's system via their API (in the case of our pilot implementation the Schedge NYU Albert API), and outsource authentication to the university's SSO system, if present.

Environment	**Specification**
Operating System	Shall run on any operating system (Windows, MacOS, Linux, Android, iOS, etc.) having a stable internet connection and browsing capability
Web Browser	Shall run on any browser (Chrome, Safari, Firefox, Edge, etc.) as long as it has support for JavaScript and web applications
Mobile Friendly	Shall be able to display information well and run efficiently on mobile devices

Frameworks:

- Frontend Web: JavaScript, CSS3, HTML5, Bootstrap
- Server Implementation: Flask-Python
- Database and SSO: Firebase and SQLite

2.5 Design and Implementation Constraints

- **Timing Constraints:** System must be fully functioning before the next round of course registration occurs (July)
- **Interface Design constraint:** CAR must be viewed and have a consistent design interface across devices with different sizes such as laptops, tablets and smartphones. This means that we must make sure that the user interface is responsive.
- **Technical Constraints:** The system should update changes in Albert live, and be available 24/7. CAR should also include either a way to save personal information and an authentication system or use local storage.
- **Hardware Constraints:** The volume of data managed by this system isn't large, however to ensure scalability making the system as lightweight as possible is encouraged.

2.6 User Documentation

The user documentation for CAR will include:

- Troubleshooting and FAQ page
- Quick start guide for new users

2.7 Assumptions and Dependencies

- The project depends on the performance of the Schedge Albert API to retrieve course data. Changes in API endpoints will require us to rework the data update system.
- The project depends on students having a Google account that can interact with Firebase created under their university email account.

3. External Interface Requirements

3.1 User Interfaces

The user interface is required to be clear, intuitive, and easy to use for students of all skill levels. Upon authentication, students will be able to see their current four-year plan, with a button to search for more courses and another to suggest a new four-year plan. As well, they can access a progress report for their degree. Admins will be able to see a console from which they can change the requirements for obtaining a given degree.

The user interface should be minimalistic and large enough to accommodate the student's courses for their eight semesters, with their professor and ratings information, while making it possible for a user to immediately know the status of their courses as soon as they log in.

3.2 Hardware Interfaces

- CAR runs on the university's intranet and should therefore interface with the hardware components in a similar manner to any other university website.
- CAR is intended to run on multiple operating systems and devices and interactions: protocols should remain consistent.

3.3 Software Interfaces

- CAR should be able to interface with Firebase's SSO in order to offer students one-click sign in through their university Google Accounts, and avoid safety concerns related to storing account information in our server.
- CAR uses calls to the Schedge API, especifically the Courses endpoint, to update periodically the system's database on currently offered and past courses, their assigned instructor, and their number of available seats in order to power the course search functionality.
- Since the product is meant to be used by a variety of users, its web application should interact seamlessly with most modern browsers.
- The product should store data in a SQLite database hosted on the server, and interact with it at any user query.

3.4 Communications Interfaces

- Since the software is a website, it will interact with HTTPS to provide access to the website.
- The software will also allow the user to download information such as four year plans into the users' local computer.
- The system must use API calls to interact with Schedge and retrieve course information, and to retrieve login information from Firebase.

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 Four-Year Plans

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).>

One of the main and unique features of our software is the ability to produce and download customizable four year plans. The four year plans will depend on the major, location and year but will be customizable for every student and keeps the student on track to graduate on time. **Priority: high**

4.1.2 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.>

The system will track the user's previous and current courses. The course list and class offering will be inputted from a database and kept up to date every night by API calls triggered by 00:00 local time.

The user is able to select "generate a four-year plan" in which the system will produce an automatic plan for the student based on their classes, majors, and information on graduating on time with the current class offerings. The user is able to adjust the plan produced and downloaded locally.

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.>

REQ-1: System must be able to generate automatic four-year plans based on student information and current degree requirements

REQ-2: Student should be able to adjust the four-year plans that are generated

REQ-3: System should keep updating the four-year plans based on the new and evolving information about the student and courses

4.2 Course Search

4.2.1 Description and Priority

Users should be able to browse and search for courses offered based on major, location and other information. **Priority: high**

4.2.2 Stimulus/Response Sequences

User clicks on the course search button and sends a query by department, degree, course name, or location.

4.2.3 Functional Requirements

REQ-1: System must allow the user to adjust and filter courses based on major, location and semester of the student

REQ-2: Students should be able to read the course description, professor rating and prerequisites for every class offered, and the current registration status for a course.

REQ-3: System should keep updating the classes as the years and semesters evolve so the user can have an updated version of the courses offered and the description (and information) about the courses.

4.3 Degree Progress

4.3.1 Description and Priority

Given the fact that the system stores information about the student and their progress in classes, the system should provide the student with a degree progress report calculating how much of their degree (based on major/minor) is completed and how much needs to be done. Priority is medium since the student can already use the software to track their courses and conclude the progress (using the four-year plan as well). This feature will just allow a more condensed and additional feature for easy quantification for the students' degree progress. **Priority: Medium**

4.3.2 Stimulus/Response Sequences

Should appear on the main screen as the user enters the application, and should expand with details if requested.

4.3.3 Functional Requirements

REQ-1: System must allow the user to check their degree progress report against the current requirements.

REQ-2: Students should be able to download the report from the website into their local devices.

REQ-3: System should keep updating the degree progress report as the student finishes more classes and progresses in their degree, or as requirements for the degree change.

4.4 Registration Planning

4.4.1 Description and Priority

For the semester immediately following the one in which the application is accessed in, the user must be able to select their major, and courses already taken, and see recommendations for classes to take in the coming semester and see how those fit into their 4 year plans depending upon different requirements
Priority: Medium

4.4.2 Stimulus/Response Sequences

Should appear on the main screen as the user enters the application, user clicks on Registration planning, select their information in terms of major(s), courses completed and then click Suggest for coming semester

4.4.3 Functional Requirements

REQ-1: System shall provide all possible options of majors, minors and previous classes offered from the API for the user to select from

REQ-2: System shall use the four year plans in the database of the user to check the remaining requirements and display the ones offered in the coming semester, while also checking for cores and general electives and displaying those available too

REQ-3: System shall let the users save the classes they are interested in for future reference

4.5 Admin Console

4.5.1 Description and Priority
The users with admin privileges shall have the capability to add or modify different information for all the users including course information, course requirements, and add/remove users while also controlling year progression

4.5.2 Stimulus/Response Sequences

Should appear on the main screen as the user enters the application, and should expand with details if requested. There would be different buttons for viewing and editing each category of information.

4.5.3 Functional Requirements

REQ-1: System should display all the information from the database depending on the category - course information to view/update course time, professor, description, etc, course requirements to view/update requirements related to the course, or view/updates users, that is, students

REQ-2: System should update the database in case the user makes changes to the data, and reflect the changes for all users

5. Other Nonfunctional Requirements

5.1 Performance Requirements

The application must perform well even if a large number of users perform requests at the same time, which will happen around the registration season, where the reliability of the service will be critical.

5.1.1 Response Time

The average response time should be less than 4 seconds

5.1.2 Recovery Time

In case of system failure, the average recovery time should be less than 60 minutes

5.1.3 Capacity

The system should be able to handle 500 concurrent users with the potential to scale to accommodate more in the future

5.1.4 Space

The system should be able to store data of up to 4 years in the operation database, after which the data should be backed up and deleted from the operation database

5.2 Safety Requirements

No private data from the student is shared by the university for the proper functioning of the system, no private data is required from the student to set up a four-year plan or search for courses, and the course data obtained from the API is a public resource. To add a layer of privacy to the data stored in the application, authentication must only be authorized to accounts from the university's domain.

5.3 Security Requirements

Users must be required to authenticate using Firebase SSO, which depends on their university-linked Google Accounts. The system requires a responsible management of the API credentials for the course catalog API and SSO access tokens. No other data stored by the system is particularly sensible, however the course catalog and four year plans shall still be stored securely and reliably.

5.4 Software Quality Attributes

Usability: the product should be easy to use by a wide range of users with a range of skill levels.

Availability: the system's uptime should be as close to 100% as possible. Maintenance, if necessary, should be scheduled far in advance and at least a week clear of any university registration event, where the usage is critical, and should be communicated in advance to the clients.

Reliability: the data stored (such as four-year plans) should be reliably and securely saved and protected against corruption and data loss by at least one backup regularly updated.

Scalability: the system should be able to withstand reasonable surges in requests and maintain service especially around registration times, when the demand for the service is high.

5.5 Business Rules

Each student may only have one active four-year plan, a maximum of two selected majors and three selected minors, and the system can reject said selection if the student does not have space in his plan to obtain the requirements necessary.

Admins can modify the degree requirements for a major or minor an unlimited number of times.

6. Requirement Gathering Techniques

The system requirements for the project were gathered and analyzed by interviewing each other. Since we belong to the group of the target users of students from NYUAD, our internal interviews gave deep, quality insights into the requirements. We were able to put ourselves into the shoes of the users and think from the base up through their perspective.

Through this methodology, we were able to come up with several instances and cases that ensure that the problem we are addressing is being solved. One of the primary user needs we identified was ease of use and navigation across the application. Due to the presence of a lot of information, it is easy to get lost and not navigate efficiently, so we shall create multiple sub-sections as part of the application dedicated to different functions as part of a responsive graphical user interface and also use filtering and sorting tools to view and interact with data as needed. Also, another need was access to class and professor reviews while deciding which classes to enroll in, so we implemented the feature to display those. And noting requirements for administrator, we provided add and update access for course and course requirement information.

Analyzing the requirements from a high-level perspective, we observed that for a functional application, the front-end for displaying the interface and data, and back-end for the database and server need to be well integrated and self-reliant in case of non-functionality of any of the components. This is reflected in the requirements and features.