

**Software Requirements**

**Specification**

**for**

**Automated Course**

**Scheduling System → ACSS**

**Version 1.0 approved**

**Prepared by Akhat Suleimenov, Dariga Shokayeva, Madina Yeleukina, Shyngys Karishev**

**NYUAD**

**20.10.2021**

***Copyright © 1999 by Karl E. Wiegers. Permission is granted to use, modify, and distribute this document.***

# Table of Contents

**Table of Contents ii**

**Revision History ii**

1. **Introduction 1**
   1. Purpose 1
   2. Document Conventions 1
   3. Intended Audience and Reading Suggestions 1
   4. Product Scope 1
   5. References 1
2. **Overall Description 2**
   1. Product Perspective 2
   2. Product Functions 2
   3. User Classes and Characteristics 2
   4. Operating Environment 2
   5. Design and Implementation Constraints 2
   6. User Documentation 2
   7. Assumptions and Dependencies 3
3. **External Interface Requirements 3**
   1. User Interfaces 3
   2. Hardware Interfaces 3
   3. Software Interfaces 3
   4. Communications Interfaces 3
4. **System Features 4**
   1. System Feature 1 4
   2. System Feature 2 (and so on) 4
5. **Other Nonfunctional Requirements 4**
   1. Performance Requirements 4
   2. Safety Requirements 5
   3. Security Requirements 5
   4. Software Quality Attributes 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** |
| --- | --- | --- | --- |
|  |  |  |  |
|  |  |  |  |

# Introduction

## Purpose

The purpose of the following document is to specify the software requirements of an automated course scheduling system. Release number: v1.0. The given SRS aims to describe the system requirements of an entire ACSS.

## Document Conventions

The document uses the following conventions:

| NYUAD | New York University Abu Dhabi |
| --- | --- |
| NYU | New York University |
| ACSS | Automated Course Scheduling System |
| IDE | Integrated Development Environment |
| VS | Visual Studio |
| SRS | Software Requirements Specifications |

## Intended Audience and Reading Suggestions

This project aims to build an automated course scheduling system for NYUAD students and it is restricted within the university premises. This project is supervised by professor Mai Oudah and reflects the team's efforts to develop the final project for the Software Engineering course. The rest of the document would present interface requirements, functional and non-functional requirements of ACSS. The table of contents reflects the order in which SRS is structured. This project is designed for NYUAD students and is useful as well as to the Albert system developers and NYUAD staff responsible for the student well-being. While the intended audience includes students, university staff and developers, it is assumed that the SRS document is primarily made for the project builders and professors responsible for the associated course.

## Product Scope

The purpose of the automated course scheduling system is to ease course registration and to create a convenient and easy-to-use application for students, preparing to register for NYUAD courses soon. The system is based on the course information provided on the NYU Albert system with its details of the course category, timings and professors. We will have a scheduling system that would ask students to authorize, import the data from Albert and build a number of possible course schedules based on the students’ preferences and requests. Above all, we hope to provide a comfortable user experience along with a functional and useful application that would enhance students’ course registration experience.

## References

# Software Requirements Specification Document with Example. Retrieved from [*https://krazytech.com/projects*](https://krazytech.com/projects)

1. [*https://www.google.com/search?q=feasibility+study&source=lnms&tbm=isch&sa=X&ved=2ahUKEwjm9eL6quDzAhVFyhoKHTUxBy0Q\_AUoAnoECAEQBA&biw=1440&bih=821&dpr=2#imgrc=l4\_NO3XCTcVjsM*](https://www.google.com/search?q=feasibility+study&source=lnms&tbm=isch&sa=X&ved=2ahUKEwjm9eL6quDzAhVFyhoKHTUxBy0Q_AUoAnoECAEQBA&biw=1440&bih=821&dpr=2#imgrc=l4_NO3XCTcVjsM)
2. SRS Template - IEEE Full version. Software Engineering course supplementary materials, Week 3: Communication

# Overall Description

## 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.>*

The software is a self-contained product, however, it uses the data about the class information from the “Albert” course registration system. It will contain:

**Class details:**

It contains the class time, professor name, which major requirement is the class.

**Schedule:**

## 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.>*

The user may use the search function to search for the desired course.

The user can add courses to the cart.

The system checks if any two classes timings don’t collide with each other.

The user can delete courses from the cart.

1. The students must be able to choose courses
2. The students must be able to see the course description
3. The students must be able to see course availability
4. The students must be able to see different options of a schedule
5. The students must be able to change some of the courses

## 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.>*

Search class.

## 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.>*

Platform:

Database:

Operating system:

## 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).>*

## 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.>*

## 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).>*

# External Interface Requirements

## User Interfaces

* Front-end software: HTML, CSS, JavaScript, Dart.
* Back-end software: Google Firebase, JavaScript, Python*.*

## Hardware Interfaces

Since the system is online, the hardware system will be all the hardware able to connect to the internet. For example, WAN-LAN, Ethernet Cross-Cable, etc. Communication protocols are described in section 3.4.

## Software Interfaces

The following table presents the software we have decided to use for the automated course scheduling online system.

| *Software* | *Description* |
| --- | --- |
| *Operating System* | We have chosen Windows and macOS as both of them are very commonly used by our team members |
| *Development Platform* | As only those who have a nyu.edu account must be able to access our system, we will add authorization that we have chosen to implement using Google Firebase because of its simplicity |
| *IDE* | We will implement the system using Visual Studio IDE because of its simplicity and wide range of functionality |
| *Modeling Web Tool* | We have chosen Creately to build models of our system because it provides an availability to work together on the same projects |
| *HTML, CSS, Dart, JavaScript* | We have chosen to use HTML, CSS, Dart, JavaScript to develop front-end of the system because they are commonly used for that purposes |
| *JavaScript, Python* | We have chosen to use Python and JavaScript to develop back-end of the system because of their simplicity |

## Communications Interfaces

Our system should be compatible with all common web browsers including but not limited to Google Chrome, Safari, Mozilla Firefox, Internet Explorer. The system will use the HTTP protocol for communication over the internet and the TCP/IP protocol suite for communication over the intranet. We will use simple electronic forms for the generate schedule, apply filters, etc. There may appear privacy issues regarding accessing courses offered NYUAD which will be resolved by authorization process. We will use external development tool Firebase to ensure that the authorization process is secure.

# System Features

## Create Schedules

### Description and Priority

The system will have a feature of generating schedules based on the information given by the user including course id or name and user preferences of time, professor, location, etc. All the possible variations of schedules should be generated at once, but the user should see the next or the previous variation of the schedule on the request. The priority of this task is High as it is the main feature of the system.

### Stimulus/Response Sequences

The behavior defined for this feature will be stimulated by the user's click on button after choosing all the courses the user wants to be included in the schedule. Optionally, the user may also choose to apply some filters before the system constructs the schedule.

### Functional Requirements

### REQ-1: The system should show a message dialog if all the possible schedules were already presented.

### REQ-2: The system should show a message dialog if no courses were chosen by the user

REQ-3: The system should show a message dialog if no schedule is possible to construct without the chosen classes clashing or excluding some of the user's preferences.

### REQ-4: The system should show a message dialog if the user adds many (more than 10) courses.

## Import Course Information

### Description and Priority

The system would need to import course information for all the offered NYUAD courses from Albert. The imported course information should at least include the weekdays and time of the particular class. This system feature has high priority and high risk, as the system output would be based on whether the import from Albert was successful or not.

### Stimulus/Response Sequences

The successful student authorization calls for the subsequent import of course information.

### Functional Requirements

### REQ-1: The import should start after successful student authorization

### REQ-2: The imported information should include class timings of each imported course

### REQ-3: The system should be able to check if the import is complete

REQ-4: The system should be able to update the user with the import status (successful/failed)

## Authorization

### Description and Priority

This feature allows users to log in the system. No registration is needed because only NYU emails are valid. Hence, users will be able to access the course scheduling system. It’s a high priority feature, since without it users will not be able to use the system at all.

### Stimulus/Response Sequences

User logins through his NYU email then the system will load the main page with all the courses and filters.

### Functional Requirements

### REQ-1: The system should be able to validate if the email address is of type “nyu.edu”

### REQ-2: The system should be able to log in the user into the system

### REQ-3: The system should be able to output the error message if the credentials are wrong

### REQ-4: The system should be able to check if the credentials are in the database

# Other Nonfunctional Requirements

## Performance Requirements

The system must be interactive and the delays involved should be minimal. So in every action-response of the system, there are no immediate delays. The system should not have delays in the process of showing the course catalogue, and in the stage of picking courses. However, during the stage of creating all the possible schedules, the system may take time to construct all possible schedules with no more than 5 seconds. Switching between schedules should be implemented without any delays. All other functions such as choosing courses, modifying filters, searching courses by names should be implemented without any delays too.

**5. 2** **Safety Requirements**

In case the system accidentally crashes, and all data in the databases get erased, the system will have to restore its past copy of the database from the archival storage. In doing so, the system should recreate a more recent state by reapplying or retrying compromised transaction operations from the backed-up log to the point of failure.

## Security Requirements

In order to keep safe the information of the NYU courses, only NYU users will be able to access the program. All users will have to go through the authentication process at the beginning of the session. The email used for authentication should be an NYU email. Otherwise, the system will not let them authenticate.

## Software Quality Attributes

* The system should be adaptable. The courses saved in the database should be up to date with the courses available at Albert. Hence, if a course is added/deleted at Albert, the system should be updated accordingly.
* A delay of one day may be appropriate.
* The system should be available to use. All NYU students should be able to use it without any difficulties or restrictions.
* The system should be correct. If a course is not on Albert, the course should not be in the system. All information should match the exact information from Albert.
* The schedule created should be valid according to the Albert rules(except for the knowledge of the pre-requisite courses).
* The system should be reliable. The admins should maintain the correct course information all the time.
* The system should be usable. The course scheduling system should satisfy a maximum number of users' needs.

## Business Rules

* All users need a valid NYU email address
* An email is considered valid if the user can prove they have access to the email address associated with the user account.
* All users must add at least one course to the catalogue.
* If the create schedule button is pressed and the course catalogue is not empty, the scheduling system should create a list of schedules and output them in a scrolling window.
* If a course name is pressed, the drop-down window with sessions to pick from(checkboxes) should appear.
* If a filter is activated, the course catalogue should be updated accordingly to show only the courses that fit the filter chosen.

# 

# Appendix A: Glossary

| NYUAD | New York University Abu Dhabi |
| --- | --- |
| NYU | New York University |
| ACSS | Automated Course Scheduling System |
| IDE | Integrated Development Environment |
| VS | Visual Studio |
| SRS | Software Requirements Specifications |