**SRS Document Development**

Mitchell Rozonkiewiecz

University of Arizona Global Campus

CST499: Capstone for Computer Software Technology

Dr. Butler

October 7, 2024

Software Requirements Specification

for

Class Tracking System

Version 1.0 approved

Prepared by Mitchell Rozonkiewiecz

UAGC CST 499

Oct 6, 2024

Table of Contents

Table of Contents iii

Revision History iii

1. Introduction 1

1.1 Purpose 1

1.2 Document Conventions 2

1.3 Intended Audience and Reading Suggestions 2

1.4 Product Scope 2

1.5 References 2

2. Overall Description 3

2.1 Product Perspective 3

2.2 Product Functions 3

2.3 User Classes and Characteristics 4

2.4 Operating Environment 4

2.5 Design and Implementation Constraints 4

2.6 User Documentation 5

2.7 Assumptions and Dependencies 5

3. External Interface Requirements 6

3.1 User Interfaces 6

3.2 Hardware Interfaces 6

3.3 Software Interfaces 7

3.4 Communications Interfaces 7

4. System Use Cases 7

4.1 Use case name and identifier 8

4.2 Withdraw money from ATM (U2) 8

4.3 Deposit money into ATM (U3) 9

5. Other Nonfunctional Requirements 9

5.1 Performance Requirements 9

5.2 Safety Requirements 9

5.3 Security Requirements 9

5.4 Software Quality Attributes 9

6. Other Requirements 10

7. System Requirements Chart 10

Appendix A: Glossary 10

Appendix B: Analysis Models 10

Appendix C: To Be Determined List 10

References 11

Revision History

|  |  |  |  |
| --- | --- | --- | --- |
| **Name** | **Date** | **Reason For Changes** | **Version** |
| Mitch Rozonkiewiecz | 2024-10-05 | Week 1 draft, includes sections 1 through 3 | 1.0 |
|  |  |  |  |
|  |  |  |  |

# Introduction

## Purpose

This Software Requirements Specification (SRS) aims to document the requirements for a new Class Registration and Tracking system. There is nothing to start with or build upon, so the registration and tracking site must be designed and built from scratch. The SRS will describe the minimal viable product (MVP) for Version 1.0 of the system. The system needs to allow a school or business to offer classes to potential students, giving the students the opportunity to sign up for the classes offered.

The system needs to be loaded with the class information, including a description, credit hours, cost, which semester it will be offered (spring/summer/fall), max class size, location and room number, as well as the day(s) and time of the classes. The students will then search this information to find the class(es) they are interested in. The search results should be sortable by the student based on location, availability, day of week, etc.

If the student is new to the school or business, they will also need the ability to create a new account. Creating a new account will include selecting a user ID and password for future logins. The student will also enter their full name legal name for the class credit(s), address, phone number, email address, and a credit card to pay for the class(es) selected. When selecting the user ID, the system needs to verify it is not already in use, as duplicates are not allowed. The password will need to be checked for complexity; the rules for the required items, such as upper/lower case, numeric, and special characters, will be determined later.

Once the account information is entered and an unused user id is selected, the student can then log in to select classes, register, and pay for them to reserve a spot in the class.

## Document Conventions

None currently

## Intended Audience and Reading Suggestions

This SRS will contain information needed by each group working on the project. This will be the engineering team members, project manager, product owner, marketing staff, individuals representing the public end users, engineering team testers, and technical document writers. “It is standard practice in the industry to prepare a Software Requirements Specification (SRS) document at the beginning of the project to ensure clear communication between all the stakeholders involved” (Krishna et al., 2024, pg. 475).

## Product Scope

The product aims to provide an easy-to-use application for students to register for and attend desired class(es). Key to the product is providing a search capability that provides users with several approaches to filtering and sorting the available options. Every student will have different criteria when searching for a class. One user might want to search by the day or week they are off work, or time of day if they work nights, and another student by search for the final class needed for a certification. Once the user has narrowed their options, seeing which classes are open and which have a waiting list will allow them to make their final selection. Ensuring the site can easily select and register for a class once found will be key to users recommending it to their friends.

## References

Abdeen, W., Chen, X., & Unterkalmsteiner, M. (2023). An approach for performance requirements verification and test environments generation. *Requirements Engineering*, *28*(1), 117–144. <https://doi.org/10.1007/s00766-022-00379-3>

Krishna, M., Gaur, B., Verma, A., & Jalote, P. (2024). Using LLMs in Software Requirements Specifications: An Empirical Evaluation. *2024 IEEE 32nd International Requirements Engineering Conference (RE), Requirements Engineering Conference (RE), 2024 IEEE 32nd International, RE*, 475–483. <https://doi.org/10.1109/RE59067.2024.00056>

Tsui, F., Karam, O., & Bernal, B. (2018). *Essentials of software engineering*. Jones & Bartlett Learning. <https://uagc.instructure.com/courses/137971/modules/items/7033225>

Wikipedia. (July 5, 2024). Usage share of web browsers. <https://en.wikipedia.org/wiki/Usage_share_of_web_browsers>

# Overall Description

## Product Perspective

This SRS will cover the initial release of the new class tracking application. The initial thought is that this will be a new product, but as requirements are gathered, the team will review existing products to determine if any off-the-shelf functionality already exists.

## Product Functions

There are several major functions the site needs to provide.

1. New user registration that includes account and profile creation.
2. Each new user should have a unique ID associated with a password. The system should guard against two users using the same ID for registration.
3. Profiles must include some key information about the applicant including name, phone, email, and any other information you may see necessary.
4. Post registration, users can login to the system at any time using the ID and the password created during the registration process.
5. Online courses run through three semesters per year (spring/summer/fall), and students can list the courses that will be offered during any semester, as not all courses will be offered in every semester.
6. Each course should have a maximum number of enrollment that may be different depending on the course.
7. If a user wants to enroll into a course and the course is full, the student can add themselves onto a waiting list.
8. A user can also cancel the enrollment from any course that they are enrolled in, and the system should inform the first in the waiting list (if any) that they can enroll into the class.

## User Classes and Characteristics

There will be several classes to implement the needed functionality. One step is for users to register their user ID, profile information, and their credit card so that a user class will be required. Users intend to visit the website to search for and register for classes, so teachers will need to register the class(es) they will be teaching and when. To register for these classes, some type of cart will be required for users to add classes to until they are ready to register and pay. For classes that are full, a waiting list class will be needed to track and notify the user if someone else drops out.

## Operating Environment

The application must work for all users if you expect them to register for classes. The end user should be able to use Windows, macOS, and/or Linux operating systems and their associated browsers. In addition to this, some users might only have mobile devices, such as smartphones or tablets. A reactive web browser interface would be the best approach. A summary of the figures and data tables, StatCounter shows that as of May 2024, Chrome, Safari, Edge, and Firefox (in that order) control nearly 92% of all browser usage (Wikipedia, 2024). Once the application is fully functional, a determination can be made for support for additional browsers is needed.

## Design and Implementation Constraints

While there is no policy for which database to use, in order to reduce costs for schools, an open source database could be considered. There is sensitive data being stored, including Personally Identifiable Information (PII) as well as credit card data (PCI), a database with row and column level encryption should be used. Consider using a paid database with paid support due to the criticality of the application. Server hardware and possible network encryption hardware to offload TLS encryption should also be considered.

## User Documentation

Care and consideration need to be taken with documentation. As a public-facing web application, hackers will do all they can to break the application and access insecure data. Do not document or expose possible points of attack if not required. We would strongly recommend hiring an external penetration testing company to ensure the application isn’t going to leak user credentials or credit card information. If it got out the application leaked user data, no one would use the software to register again, and the business or school could go out of business.

## Assumptions and Dependencies

As previously mentioned, sensitive information needs to be stored, so even though it will be more costly on an ongoing basis, a commercial database such as IBM Db2 with row and column level security should be considered. This knowledge might not exist within the current team and might need to be hired externally.

Also, as a publicly facing site, penetration testing should be performed for the project. Hiring a whitehat company to test the application before production deployment should be performed.

# External Interface Requirements

## User Interfaces

Some registry systems already exist, such as Canvas. A good user experience and possibly lower cost alternative could be used to get users to use your company. With the desired browser and hardware support, having a responsive interface that makes finding and selecting the desired classes easier would be the first step. Think about the issues you encounter when using a website. You want to ensure your site doesn’t repeat those issues.

Users can have disabilities that you do not understand if you don’t have them. I have learned there are a lot more color-blind individuals than you’d expect. Do not rely on only color to get the user’s attention. Many individuals have trouble with objects that flash. This is part of Section 508 compliance. Section 508 measures how easily a user with disabilities can use a product or website. This includes not requiring the highest screen resolutions but rather something in the middle that can be zoomed in or out.

## Hardware Interfaces

The registry system must display the interface on computers, tablets, and smartphones. The site will likely be hosted at a hosting company. Any natively compiled modules must be built for that specific operating system. The design should attempt to reduce, if not eliminate, the need for native code to remove this dependency.

## Software Interfaces

As mentioned, the registry system will need to interface to other components, at least a database. The database will store all the class information, such as a description, day of week, time of day, credit hours, cost, and pre-requisite classes. The teacher information for the class will also be needed. User IDs and their profiles containing physical mailing as well as email addresses, previous classes to validate pre-requisites, and possible balance due. The user’s credit card information will also be needed to register for new classes. When the user checks out, the list of classes will need to be tracked, and updates will need to be sent to the student if later canceled. If they were added to a waitlist, updates as to changes in position or entry in the class will need to be communicated.

Another critical interface will be between the registry system and banks to verify the user’s credit card information and send in charges when the credit card is used to reserve a spot in the class. Care is needed to ensure this is a trusted connection using encryption.

## Communications Interfaces

The registry system will need to use several communication standards. The user interface will be a browser-based application that will use HTTP. If a user is browsing the website to see if any desired classes are available, that would be okay. As soon as the user needs to register for a class, the application will require the user to log in and switch to using HTTPS at that time to protect the integrity of the transaction. This means that TLS protocols will encrypt/decrypt all network communications. This will be especially important when the website communicates to the bank to verify that the credit card can be charged for the order. “Performance aspects were not considered equally by the requirements engineers when writing the SRS documents, which shows the lack of knowledge in the inter-dependency relation between different aspects as shown by PRO-TEST” (Abdeen et al., 2022, pg. 136). The performance of the interfaces together must be considered and measured. If the application is too slow, the student will search for other options and might not return.

# System Use Cases

The overall use case diagram should be here.

The text description of each use case should follow.

## Use case name and identifier

1. **Objective** - What is the ultimate objective of the use-case. What is it trying to achieve? What was the source of the use-case requirement?
2. **Priority** – The overall priority of this use-case (Low, Medium, High)
3. **Source** – Who is the main source of this use case. Who cares most about this functionality? This should be the one person you would ask if there is a question about this use-case. (Make up a name and cite their: John Smith (End-user) here.)
4. **Actors** - Who is involved in the use-case? Which actors/stakeholders?
5. **Flow of Events** 
   1. **Basic Flow** - flow of events normally executed in the use-case
   2. **Alternative Flow(s)** - a secondary flow of events due to infrequent conditions
   3. **Exception Flow(s)** - Exceptions that may happen during the execution of the use case
6. **Includes** - other use case IDs that are referenced in steps in the flow of events.
7. **Preconditions** - Any condition that must be satisfied before the use case begins. If the condition is “User is logged in”, then the first step of the use case is NOT “User logs in”. They are already logged in if that is a pre-condition!
8. **Post conditions** - The conditions that will be satisfied after the use case successfully completes
9. **Notes/Issues** - Any relevant notes or issues that need to be resolved

## Withdraw money from ATM (U2)

1. **Objective** – The customer is withdrawing money from the ATM and the system will debit the customer’s account.
2. **Priority** – High
3. **Source** – Carl Gnome (marketing)
4. **Actors** – Customer, central bank computer
5. **Flow of Events** 
   1. **Basic Flow**
      1. Customer chooses the checking option on the ATM
      2. Customer chooses the amount of money needed
      3. Customer confirms the choice
      4. System validates the amount
      5. System asks central bank computer to debit the customer’s account
      6. System issues money to the user
   2. **Alternative Flow 1** – At step 5.1.4 the amount is not a multiple of $20
      1. An error message is displayed telling the customer they must use multiple of $20.
      2. Return to step 5.1.2
   3. **Alternative Flow 2** – At any step the user presses “cancel”
      1. System returns to the main menu
   4. **Alternative Flow 3** - At step 5.1.5 bank computer returns a failed status, “insufficient funds”
      1. An error message is shown to the user
      2. Return to step 5.1.2
   5. **Exception Flow 1** –
      1. Database is locked due to backup in progress. System executes use case U5
6. **Includes**
   1. U5 – Exception occurs
7. **Preconditions** – User is logged in
8. **Post conditions** – Money has been returned to the user and their account balance has been updated.
9. **Notes/Issues** - None

## Deposit money into ATM (U3)

….

# Other Nonfunctional Requirements

## 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.>

## 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.>

## 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.>

## 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.>

# 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.>

You may not have any.

# System Requirements Chart

*< Include a* ***table*** *in this section with the following columns:*

***ID*** *– Unique requirement ID*

***Priority*** *– Priority of this requirement*

***Type*** *– Functional(F) or Non-functional(NF)*

***Source*** *– Who is most interested in this requirement (John Smith – Customer). For this project you can make it up, in reality you’ll want to capture this as you capture the requirements.*

***Contained in Use Case(s****) – Which use cases reference this requirement or which use cases when executed will perform this requirement. There may be a few functional requirements without a use-case and the non-functional requirements generally will NOT be part of a use-case (so put N/A).*

***Description*** *– The description of the requirement. “The system shall …. “*

*>*

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.>

**Don’t do any of these for CS421 SRS. You will create these models during the high level design deliverable.**

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.>

List here any open questions or things you know still need to be done to the SRS, but haven’t been addressed yet. (It’s okay to have things like that, especially in this CS421 project because we don’t have time to do everything.)

# References

Abdeen, W., Chen, X., & Unterkalmsteiner, M. (2023). An approach for performance requirements verification and test environments generation. *Requirements Engineering*, *28*(1), 117–144. <https://doi.org/10.1007/s00766-022-00379-3>

Krishna, M., Gaur, B., Verma, A., & Jalote, P. (2024). Using LLMs in Software Requirements Specifications: An Empirical Evaluation. *2024 IEEE 32nd International Requirements Engineering Conference (RE), Requirements Engineering Conference (RE), 2024 IEEE 32nd International, RE*, 475–483. <https://doi.org/10.1109/RE59067.2024.00056>

Tsui, F., Karam, O., & Bernal, B. (2018). *Essentials of software engineering*. Jones & Bartlett Learning. <https://uagc.instructure.com/courses/137971/modules/items/7033225>

Wikipedia. (July 5, 2024). Usage share of web browsers. <https://en.wikipedia.org/wiki/Usage_share_of_web_browsers>