Software Requirements Specifications for Scholarli

College of Engineering and Computer Science

Florida Atlantic University

CEN4010 – Principles of Software Engineering

Dr. Safak Kayikci

Prepared By: Juan Rodriguez, Kevin Tran, Thau Tran-Nguyen

Revisions Page

Name	Date	Version
Thau Tran-Nguyen	6/20/2024	1.0
Juan Rodriguez	6/22/2024	1.0
Kevin Tran	6/22/2024	1.0

Table of Contents

1. INTRODUCTION

1.1 Purpose

The purpose of this Software Requirements Specifications (SRS) document is to outline the requirements for the development of Scholarli. Scholarli is a mobile application that is tasked with providing users a new and efficient way to plan and organize their semester assignments, quizzes, readings and more. This document serves as a guide for the development team, users and investors involved in the project.

1.2 Scope

The name of this project is Scholarli. Scholarli is aimed towards students in colleges, universities or high school. This project will efficiently organize several types of assignments over the semester. It has a function that will leave notification reminders for students to study or start an assignment. Our aim is to provide these students with a fully customizable experience depending on how much they want Scholarli to aid them, students can minimize involvement to only notify students once or twice about a certain assignment or to notify them multiple times based on their lifestyle. The main objective is to give students a minimal maintenance assignment tracking resource that can be flexible and customizable.

Scholarli allows users to track multiple courses and their respective assignments, quizzes, and exams.

Scholarli will provide users the option to mark an event with identifiable tag altering the rate at which Scholarli will notify them.

- Homework: On the assigned date, one day before due date, and on due date
- Readings: On assigned date, one day before due date, and on due date
- Lab: On assigned date, three days before due date, and on due date
- Essay: On assigned date, three days before due date, and on due date
- Quiz: Study reminders each day 7 days before quiz, on quiz date
- Exam: study reminders each day 10 days before exam, increase notification frequency as exam date approaches, date of exam

These are the preset frequency, but users can alter it based on their preferences; overlapping assignments due on the same day can also be marked with a priority tag for better task organization. Our objective with these features is to help students track all school related work with smart reminders.

1.3 Definitions, acronyms, and abbreviations

- **SRS**: Software Requirements Specifications
- **API**: Application Programming Interface
- **OS**: Operating System
- **HTTPS**: Hypertext Transfer Protocol Secure
- **Celery task scheduler**: A tool used in software development to schedule and manage periodic tasks, ensuring they are performed at the correct times.
- **SendGrid email API**: An API provided by SendGrid for sending emails from applications in a reliable and scalable manner.
- **GUI**: Graphical User Interface
- **IDE**: Integrated Desktop Environment

1.4 References

Schedule IT: https://github.com/anxalivn/Schedule-IT

myHomework App: https://myhomeworkapp.com/

Notion: https://www.notion.so/templates/assignment-tracker-2

MYSTUDYLIFE: https://mystudylife.com/

Excel Assignment Schedule: https://create.microsoft.com/en-us/template/assignment-schedule-3445dcd8-2047-4602-8056-ea5c30d95b19

Celery task scheduler: https://docs.celeryq.dev/en/main/userguide/periodic-tasks.html

SendGrid email API: https://sendgrid.com/en-us

When and how to study for a quiz or test: https://www.asundergrad.pitt.edu/study-lab/study-skills-tools-resources/seven-day-test-prep-plan

1.5 Overview

This document will detail the interface, protocols, requirements and performance of Scholarli. It will also provide an overview of the application's reliability, availability, security, maintainability, portability.

2. SPECIFIC REQUIREMENTS

2.1 External Interface Requirements

2.1.1 User Interfaces

Logical characteristics:

Registration and Login: When users open the app, there will be a registration and log in page. It will display two textboxes for the user to enter the log in information, which is their email address and password. If the user does not have an account, there will be another button for a user to register.

Semester Page, View Semester: Upon logging in, there will be a page display all of the user added semesters. Users can add semesters by clicking the plus on the top right of the page. If the user wish to delete a semester, they can swipe left on the semester. That will prompt the user to confirm the deletion by typing "CONFIRM" in the textbox. Clicking the plus button to add a semester will pop-up a screen to name the semester and set the duration of the semester.

Display All Courses, Add/Remove Courses: By clicking a semester, the user will be greeted by a courses page. This page will display all the user added courses for that semester. Users can add a course by clicking the plus button on the top right corner of the page. Similar to semester deletion, users can delete a course by swiping left and confirming. Upon clicking "Display All Courses," user will be greeted with a half-page calendar of all the courses color coded (which is chosen by the user). Below the calendar will be the upcoming deadlines of assignments for all courses. There will be an option to convert the page to a PDF format for printing.

Adding a Course: When users add a course, there will be a pop-up that allows the user to name the course and set a color for the course. More options will be added once there are feedback and demand for more features.

Display One Course: When users click on one course, the calendar for that course will occupy half the page and the upcoming assignments. There will be an option to convert the page to a PDF format for printing.

Add Assignments: To access add assignments, users will need to navigate to a course page. From this page, users can click the add assignments button on the top right corner of the page. This will pop-up a page for users to add the assignment name and type. The form will adjust accordingly to the type that the users choose. Options of assignments include homework, quiz, exam, reading, laboratory, essay and more upon users' feedback and requests.

Edit Assignments: Users have the option to edit assignments once created. They can change all aspects of an assignment. From this page, they are also able to set the status of an assignment from "Not Started, In Progress, Completed."

Notifications: Users will receive constant notifications. For several types of assignments, users will get notifications to start an assignment or to study for an upcoming quiz or exam within the appropriate timeframe.

Optimization:

The application will be optimized to ensure rapid loading times. This is an application intended to be used on mobile devices, so allowing rapid access is crucial for effective use. We will aim to minimize use of resources and use efficient data storing techniques.

2.1.2 Hardware Interfaces

This requires a computer, tablet or smartphone. An internet connection is helpful for device syncing, but it is not necessary.

2.1.3 Software Interfaces

This will be a mobile application on Android/iOS and a web application that can be used on all major web browsers (Chrome, Firefox, Microsoft Edge).

2.1.4 Communications Protocols

This project will be utilizing the networking services provided by the different platforms (android, iOS, web browsers) operating systems. Communications between the users and servers will be encrypted using HTTPS to ensure secure data transmission and user privacy. Email notifications will also be utilized to notify users, this will require an email service API such as SendGrid and a task scheduler such as Celery. We will also be using the built-in notification features found on both mobile OS's.

2.2 Functional Requirements

1. **Validity check on inputs:**

- The system should validate that input for creating/signing into an account to ensure they are valid account details
- Invalid account details when signing in/creating an account should lead to an error message

2. **Exact sequence of operations:**

- Once account details are validated the user should be taken to a semester home page where users can see their already entered semesters (fall 2023, spring 2024) or a blank page where they can enter these semesters.
- Then the user can select a semester to enter another page detailing the classes listed under it.

- Next, the user can access a Calander view for these classes where all assignments, quizzes, and exams are displayed, here we can change the view based on month, week or day. A button to add assignments is also present on this page
- Next, the user can add assignments and add information such as (name, due date, type, subject, status and reminder frequency)
- Next, users can select and print out specific assignment calendars for one or multiple courses.

3.**Responses to abnormal situations:**

- Overflow: the application will handle overflow by indicating any overlapping due dates and prompting the user to rank assignments based on priority to better provide reminders.
- Communication Facilities: The application will attempt to reconnect to required services if connection is interrupted only when those services are **required.**
- Error Handling and recovery: If services are unavailable an error message will be displayed, and error will be logged.

4. **Effect of parameters:**

- The application should allow users to enter a variety of descriptive tasks based on their coursework.
- The application should allow users to customize due dates down to the minute they are due.

5. **Relationship of Outputs to Inputs:**

- Input/Output Sequences: users' input (assignment type, due date, priority) will directly affect the reminder frequency and times.
- Formulas for Input to Output Conversion: The application will use a predefined formula to calculate when reminders should be sent out, variables that will contribute to this equation will come from user input.

2.3 Non-Functional Requirements

2.3.1 Reliability

This app is available offline when the user downloads it. Everything is stored locally, and it synchronizes when connected to the internet. The system will achieve 99.9% uptime to ensure that students can access their assignment tracker faultlessly. The system will ensure that the security of data is intact through regular backups to prevent data corruption or loss.

2.3.2 Availability

The system will be available 24/7, except during scheduled maintenance hours.

2.3.3 Security

The system will implement multi–factor authentication for security.

2.3.4 Maintainability

There will be an option for users to submit a bug report, and the development team will resolve them. Documentation will be provided, alongside user manuals, system architecture, and API documentation to help with maintenance and future developments of the software.

2.3.5 Portability

Users can access the app anytime on their smartphone or tablet. An internet connection is not needed. System will be on multiple platforms such as Windows, MacOS, IOS, and Android.

2.3.6 Performance

The system will be able to process at least 100 transactions per second during peak usage periods.

2.4 Design Constraints

This application must run on the two main operating systems found on mobile devices (IOS/Android OS). Application will also be able to be accessed on all major web browsers. Devices must also be able to access network connectivity to allow for full functionality. Notifications must also be enabled to allow the notification feature to properly work on mobile devices.

3. ADDITIONAL MATERIAL

Android Studio: This is an integrated development environment for android operating systems. This is a good tool to use for developing the android application for Scholarli. It has many useful GUI tools to aid in the app's development.

Xcode: This is an integrated development environment for iOS applications. It can help with the development of the iOS version of Scholarli.

Qt Creator: This is a cross-platform IDE. It can be used to develop Android applications. It may be useful if Scholarli ever expands to making a Linux compatible application.