



StudyBug Academic Planner Software Design Document

McMaster University, 2022 Fall, COMPSCI 4ZP3 - Capstone Project

GROUP 13

Name(s): Camisha Mortensen, mortensc, mortensc@mcmaster.ca
Yifan Jiang, jiany25, jiay25@mcmaster.ca
Mike Dai, daiy29, daiy29@mcmaster.ca

Date: 11/30/2022

Table of Content

Table of Content	2
1 INTRODUCTION	4
1.1 Purpose	4
1.2 Target Audience	4
1.3 Scope	4
1.3.1 Goals	5
1.3.2 Objectives	5
1.3.3 Benefits	6
1.4 Overview	6
1.4.1 SDD Overview	6
1.4.2 SSD Organization	6
1.5 Reference Material	7
1.6 Definitions, Acronyms and Abbreviation	7
2 SYSTEM OVERVIEW	8
2.1 Functionality	8
2.2 Context	9
2.3 Design Consideration	9
2.3.1 Software Development Tools	9
2.3.2 Software Development and Support Team	10
2.3.3 Product Perspective and References	10
2.3.4 User Classes and Characteristics	11
2.3.5 Operating Environment	11
2.3.6 Design and Implementation Constraints	11
2.3.7 External Interface Requirements	11
2.3.8 User Documentation	12
2.3.9 Assumptions and Dependencies	13
3 SYSTEM ARCHITECTURE	13
3.1 Architectural Design	13
3.1.1 Development Environment	13
3.1.2 Activity Diagram	13
3.1.3 Module Architecture	14
3.2 Decomposition Description	15
3.3 Design Rationale	18
4 DATA DESIGN	19
4.1 Data Description	19
4.2 Data Dictionary	19
5 COMPONENT DESIGN	22
User Class	22
Overview Class	22
Course Class	23

Software Design Document

Task Class	24
Todolist Class	24
Calendar Class	25
6 HUMAN INTERFACE DESIGN	26
6.1 Overview of User Interface	26
6.2 Screen Images	28
6.3 Screen Objects and Actions	32
Table 6. Screen objects and actions	34
7 REQUIREMENTS MATRIX	34

1 INTRODUCTION

1.1 Purpose

The purpose of this Software Design Document(SDD) is to document the system design, including architecture design, data design, component design and human interface design, of StudyBug Academic Planner(SAP). The SSD demonstrates the approaches of software development that achieves the requirements of StudyBug Academic Planner. This SDD serves as a development guideline, technical reference and traceback checkpoint for the development team and technical support team. This document is not the final version of SDD and will be incremented and iterated, keeping track of all milestones and updates, throughout the software life cycle of StudyBug Academic Planner.

1.2 Target Audience

The target audiences to this SDD includes but are not limited to:

- software development team
- software maintenance team
- technical support team
- software project supervisor
- software project reviewer

The assumptions on audiences of this SDD:

- the audiences shall have technical knowledge in the fields of computer science and software engineering
- the audiences shall know how to use computers that run mainstream operating systems including macOS, Windows, Linus and know how to use mainstream browsers including Chrome, Edge, Firefox
- the audiences shall understand English language

1.3 Scope

StudyBug Academic Planner is a web application intended to allow university students to plan their upcoming semesters within the current academic year and to plan their overall degree as well. It will offer features for users to input details about each of their courses, assignments, tests and deadlines and calculate grades. Users will be able to view their courses and goals in an overall dashboard view as well as a calendar view by month, week, day, etc. and check them off a To Do List, tracking their grades and progress as they go. Additionally, students will be able to generate a weekly study plan based off of their schedule and

assignment priority/difficulty. The strategy for this product is to combine and expand upon the best features of similar Calendar, Time Management and Academic Planning applications in one place and for students to be able to use the application throughout their entire degree. The web application will be developed using the MERN Stack including MongoDB, Express, React and Node.

1.3.1 Goals

The following goals must be achieved by the software deliverable:

Meet Core Requirements

The core functional and non-functional requirements proposed in Software Requirements Specification for Academic Planner Program must be met by the software deliverable.

User Friendly

The using approaches, interactive logic human-computer interface must be easy to use and demonstrate a satisfactory user experience.

Extensibility

The system must be extensible for future new features in the maintenance phase of the software life cycle. New features should not impact the existing software architecture and functions. Upgrades should not be made upon the sacrifice of current user benefit and user experience.

Traceability

The system must be reasonable to analyze and easy to maintain from the maintenance technicians' perspectives. All development processes should be effectively and completely documented and all aspects reflected in the software deliverable should be able to trace back to the corresponding documentation.

Open to Feedback

The development and maintenance of the system must be open to user feedback. Necessary upgrades must be made based on constructive feedback in feasible ways.

1.3.2 Objectives

The objectives throughout the software development of StudyBug Academic Planner:

- Form development team
- Document Software Requirements Specification(SRS)
- Document Software Design Document
- Implement software prototype
- Implement full system functionalities
- Test and debug code
- Final delivery of software
- Software project showcase in McMaster Faculty of Engineering Expo

1.3.3 Benefits

StudyBug Academic Planner is a software intended to help university students better manage their schedules and plan their degrees. Correctly and effectively use this software will significantly help university students do better in time management, schedule organization and self discipline and present a better academic performance. Potential users including but are not limited to university faculty administrative staff, instructors, teaching assistants, students' parents and university non-faculty staff are also able to benefit from this software in their daily planning.

StudyBug Academic Planner is a capstone project and is supervised under McMaster University, Faculty of Engineering, Computer Science Program (Year of 2022/2023). Development team members are able to learn approaches to perform the entire software development process and understand the software life cycle in practice. Development team members will also demonstrate their understanding of the computing and software industry, their proficiency of technical knowledge and their skills in programming.

1.4 Overview

1.4.1 SDD Overview

The SDD consists of 7 sections including Introduction, System Overview, System Architecture, Data Design, Component Design, Human Interface Design and Requirement Matrix, and the additional Appendices.

1.4.2 SSD Organization

Introduction

An overall introduction of the SDD of StudyBug Academic Planner from aspects of purpose, target audiences, scope, overview, reference material, definition, acronyms and abbreviations. This section helps audiences to effectively read this SDD and obtain the information they need.

System Overview

A general description of the functionalities, context and design considerations of the software project. This section helps audiences understand the software from a top-level perspective.

System Architecture

The specific set of descriptions of the system architecture design, the insights to the decomposition of the subsystem and the underlying design rationale. This section helps audiences understand the structure of the software system in detail and let them know the reasons why the system is designed in this way.

Data Design

The specific set of descriptions of how the information domain of the system is transformed into data structures in the back-end database and an alphabetical list data

dictionary of the system entities with attributes, methods and parameters. This section provides audience understanding of the data structures and useful tools in developing the database management system.

Component Design

The summary of algorithms design for each object member function in Procedural Description Language(PDL) or pseudocode. This section provides the development team the necessary information and guideline of how the system should be implemented.

Human Interface Design

The specific set of descriptions of the human-computer interface including the description to interactive logics, the visual demonstrations of the system screen with images. This section helps the audience know how to use this software as a user and helps the development team know how to present the software appearance and deploy the front-end website.

Requirement Matrix

The set of cross-references that trace components in Section 5 Component Design to requirements in Section 4 System Features and Section 5 Other Nonfunctional Requirements in SRS for APP. This section helps audiences know whether and how the software requirements are met by the software development.

Appendices(Optional)

Additional supporting documents to the SDD.

1.5 Reference Material

- Capstone Project - Group Information and Project Description
- Software Requirements Specification for Academic Planner Program
- [ANSI/IEEE Std 1016-1987,IEEE Recommended Practice for Software Design Descriptions.](#)
- [ANSI/IEEE Std 1016-1993,IEEE Guide to Software Design Descriptions.](#)
- [Atlanta Regional Commission – MSAA, System Design Document.](#)
- [Software Design Document, Testing, Deployment And Configuration Management, And User Manual of the UUIS.](#)
- [Web Content Accessibility Guidelines 2.](#)
- [Software Design Document \(SWDD\) Template.](#)
- Object Oriented And Classical Software Engineering 8th Edition

1.6 Definitions, Acronyms and Abbreviation

**Table entries are listed in chronological order of the first appearance in this SDD.*

Terms / Acronyms and Abbreviations	Definition / Full Word
SDD	Software Design Description
SAP	StudyBug Academic Planner
MERM Stack	MERN Stack, including MongoDB, Express, React, Node, is a JavaScript software technology stack for website and web applicant development.
SRS	Software Requirements Specification
PDL	Procedural Description Language
UI	User Interface

Table 1. Acronyms and Abbreviation

2 SYSTEM OVERVIEW

2.1 Functionality

The core feature of StudyBug Academic Planner is providing a planner that allows students to monitor courses in a top-down manner:

- Users will be able to add the details for each of their current courses including, lecture schedule, assignments, tests, and syllabus information.
- Added assignments, labs, and tests will include weights, deadlines, and relevant links.
- The application will generate corresponding tasks based on course information such as a lecture review task, a reading task, or a note-organizing task.
- Students can return to the course homepages they create throughout the semester

to record assessment marks, monitor in-progress grades and calculate the minimum mark requirements for outstanding assessments depending on their goals.

- Course details and study tasks will be viewable on the Calendar
- The same tasks will also be viewable on the To-Do list in more detail where users can check them off
- Each course will be clearly highlighted in a unique color across the application
- Users will be able to add non-academic tasks to the calendar and To-Do list
- Task on the calendar and To-Do list will be taggable and filterable

StudyBug Academic Planner also provides a rich set of course-management and time-management features, including:

- The application will generate a suggested weekly study plan in order to help students decide what they should study, when and for how long based on their schedule and assignment priority/difficulty. This can be automatically added to the calendar and To-Do list.
- Users will also be able to access degree planning section from the Dashboard where they can track and view the progress of their entire degree

2.2 Context

The concept of StudyBug Academic Planner originates from a set of time management strategies employed by one of the development team members for several years through his undergraduate career. StudyBug Academic Planner is the computerization of the handwriting version planner and is intended to perform the whole set of academic planning logics in a more effective and reliable way, and help more university students on the other hand. The software strategy is using the MERN Stack to implement a web application that runs on modern web browsers, which is cross-platform and cross-device.

2.3 Design Consideration

2.3.1 Software Development Tools

Tool	Description
Google Docs, Microsoft Office365	Collaboration Document Editing Platform
Figma	User Interface Design Platform

Chrome, Edge, Safari, Firefox	Software Running and Testing Browser
macOS, Windows, Linux, iOS, Android	Software Running and Testing Platform
VisualStudio, WebStorm	Integrated Development Environment

Table 2. Software Development Tools

2.3.2 Software Development and Support Team

If you have any troubleshooting or feedback to any aspect of StudyBug Academic Planner, please contact the software development and support team

Name	Role	Email
Camisha Mortensen	software development and support team member	mortensc@mcmaster.ca
Mike Dai	software development and support team member	daiy29@mcmaster.ca
Yifan Jiang	software development and support team member	jiany25@mcmaster.ca

Table 3. Software Development and Support Team

2.3.3 Product Perspective and References

The application is a new and self-contained academic planner that helps students to plan their upcoming semester and manage their time. Similar applications include, Google Calendar, Outlook Calendar, Google Keep and TickTick. There are many other generalized planners, To-Do list organizers and habit trackers available online as well as internal academic planners such McMaster's MyTimeTable or Avenue to Learn. However, there are

few, free solutions available for students to manually track their progress in classes and in their degrees and organize their study time around the rest of their life. This application aims to provide a solution to this problem.

2.3.4 User Classes and Characteristics

- University students are the primary audience of the application. They are aged 18-25 and come from a diverse background, they may be international students from anywhere in the world, they may be studying any major, they may be any gender, English may not be their first language however they are very tech savvy and familiar with similar applications. They may be juggling multiple major responsibilities such as multiple classes, work, extracurriculars and social events and use the application up to multiple times daily in order to keep track of their lives.
- Non university students are also an important user class. They are younger than the primary audience at around 12-17 (middle and high school aged). This group is equally diverse and tech savvy as the university students however they may not have quite as many responsibilities in life to keep track of and the degree planning section is likely to be of no use to them for now. However the application can still be a powerful tool for those in this group who want to learn to plan and track their academics and build good time management skills for the future.
- Parents of students may also be users of the product when helping their child organize their time. This is the oldest user class at 30+ and likely the least tech savvy however they are still assumably familiar with similar applications and should have no issues guiding someone through the application

2.3.5 Operating Environment

The application will be available on all modern web browsers (chrome, firefox) and will be optimized for both a desktop and mobile view, therefore there are minimal hardware limitations for the project. As a standard web application, the HTTP protocol will be used for communication and sensitive user data will be encrypted.

2.3.6 Design and Implementation Constraints

- The application will not interact with any other application to automatically fill out any of the course information, users will be required to do this themselves. This keeps the product accessible to students from any institution.
- The application will only be available in English.
- The application will be developed using the most up to date version of all technologies in the MERN stack.

2.3.7 External Interface Requirements

User Interfaces

All major functionality of the application will require the user to interact with the interface. The application will be designed responsively to ensure a pleasant viewing

experience on both mobile and desktop browsers. Users can always expect a standard navigation bar across the top of the website with links to the Homepage, Dashboard, Calendar, To-Do List, and an Account button which upon being clicked reveals a Settings Page, Account options and a Sign In/Out option. The GUI will take accessibility into account ensuring all pages are in line with Web Content Accessibility Guidelines.

Hardware Interfaces

This product should be able to run on as many types of devices as possible, and thus hardware should not be a limitation. There is very minimal hardware usage for this program outside of the basic components of a computer or handheld device. At the bare minimum, the supported device types will be mobile phones, tablets, as well as most computers.

Software Interfaces

This project will run as a web application using the MERN stack. Most of the application should be written in the most up-to-date version of Javascript. [The development team reserves the right to reevaluate the technologies, libraries and frameworks used to create this application during implementation in January 2023 should the following not suit their needs.]

- MongoDB: This is a cross-platform document-oriented NoSQL database program. It uses JSON-like objects with optional schemas. This will be used to store and retrieve information needed for the application; an example would be course information.
- Express.js: Express.js is to be used alongside Node.js as a framework. As we are using HTTP, it will be used as middleware, and assist in routing and specifying HTTP verbs and URL patterns for the web application.
- React.js: React will be used for designing the user interface for the web application. It is dynamic, simple to use, and scalable to various devices.
- Node.js: Node.js is used for server development and can be run on many types of operating systems, such as Windows, Linux, Unix, Mac OS, making this ideal for allowing our application to be flexible and usable on most devices. It allows for dynamic web content to be displayed in the browser running on Javascript. Some of the key features are that it's asynchronous, has no buffering, and is scalable.

We will host this application on Amazon Web Services.

Communications Interfaces

This application will use the HTTP protocol for communication, as it will be a web application. This application will feature functions such as forms, email authentication, electronic forms, and web navigation which will be handled by Node.js through HTTP request methods. Sensitive information such as passwords, will be encrypted and stored in our database using a verifiable standard such as AES.

2.3.8 User Documentation

- Homepage: Users who are not signed into any account will land on a homepage which gives an overview of the application including an about section and highlights of key features

- New user pop-up guides: As a new user navigates the application for the first time they will be prompted with small dismissible pop-ups that guide them through the flow of the application and all of the major features in the expected order

2.3.9 Assumptions and Dependencies

- We assume that the user is enrolled in classes where they have access to all of the details of the course and will be able to provide these to the application
- We assume that the user speaks English (ESL will be sufficient)
- We assume that the user has an email address
- We assume that the user is familiar with how to use a computer and navigate a web browser

3 SYSTEM ARCHITECTURE

3.1 Architectural Design

3.1.1 Development Environment

Software	Description
Github	Version control repository
Heroku/Amazon Web Services	Cloud computing platform
MongoDB	Database
NodeJS	Programming language
ExpressJS	Programming framework for Node
React	Programming framework for Web UI
JavaScript	Programming language

Table 4. Development Environment

The development environment will use the stack listed above. This suits the needs of Studybug.

3.1.2 Activity Diagram

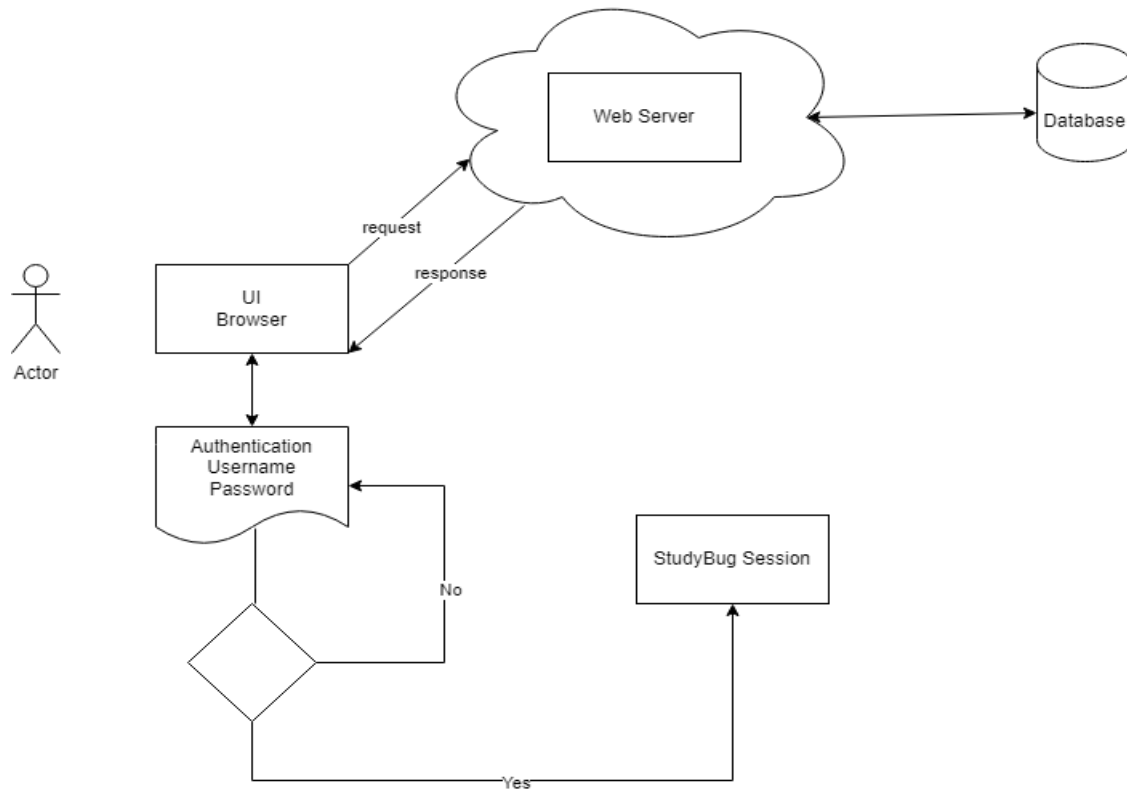


Figure 1. Activity Diagram

3.1.3 Module Architecture

The system is to be built upon modules revolving around the user's academic planner. As such, the relevant data will be stored in a user's "profile". At the highest level, we will have the user and all sub-modules will interact directly with the user's profile. Higher level subsystems will include a user sign-in module, which will be responsible for user registration, login, and other related features such as password resets, as well as an academic overview module. Submodules of this include a calendar and ToDo module, which may contain information about courses and tasks. These modules will be interacting with each other. Users will be able to input their courses and tasks which will be saved to their profile, and these can also be added to their profile's calendar and ToDo list. These modules will provide all the functions required for the product. A more detailed decomposition will be provided in the following section.

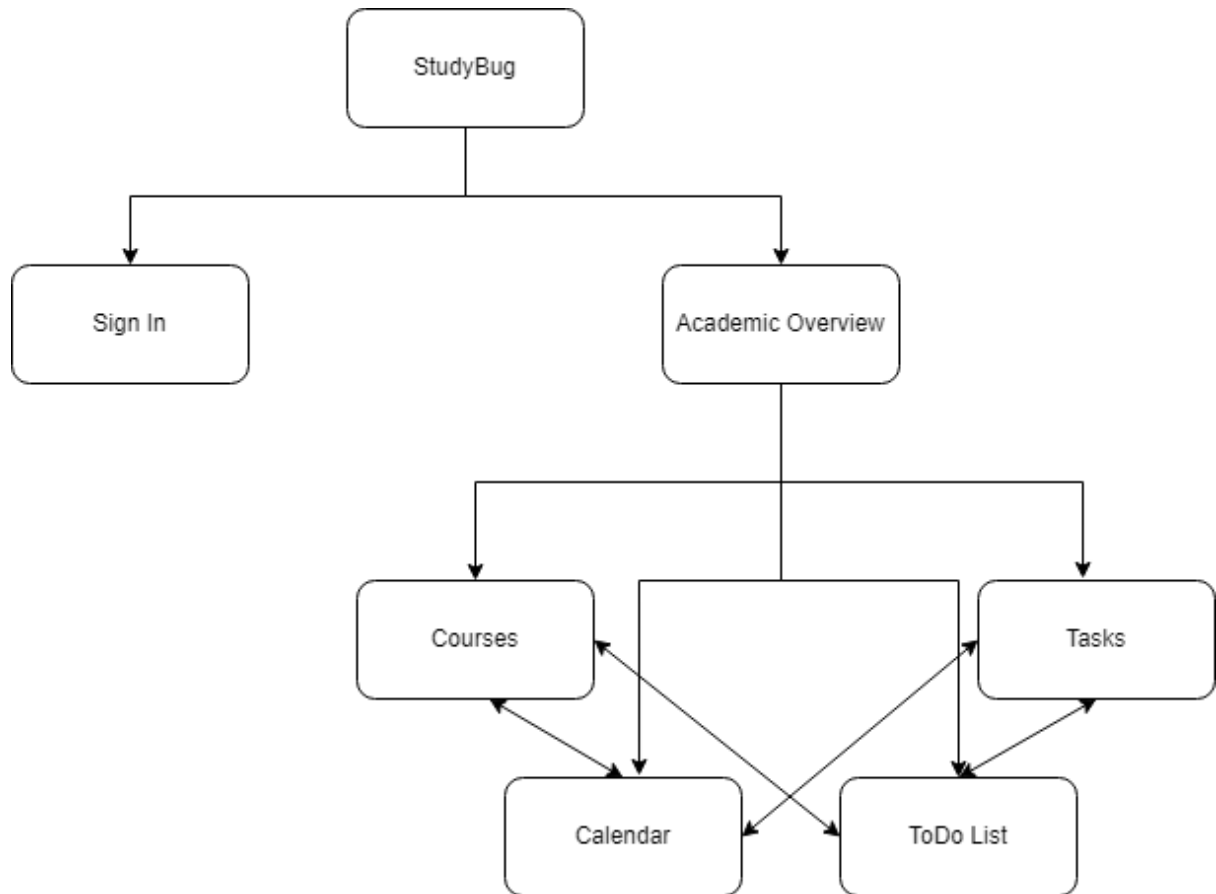
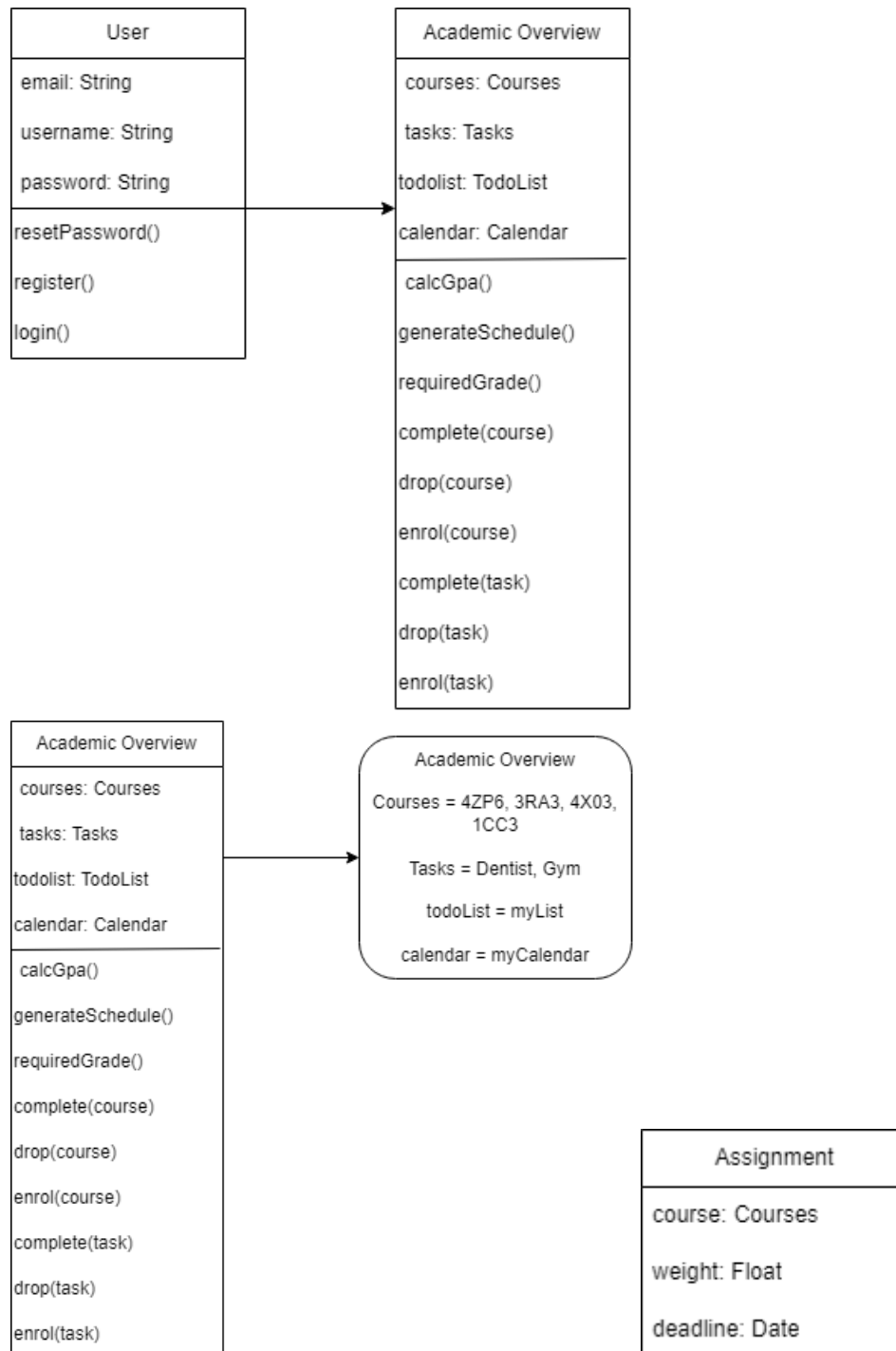


Figure 2. Module Overview

3.2 Decomposition Description

A decomposition of the subsystems are shown, as well as object diagrams:

Software Design Document



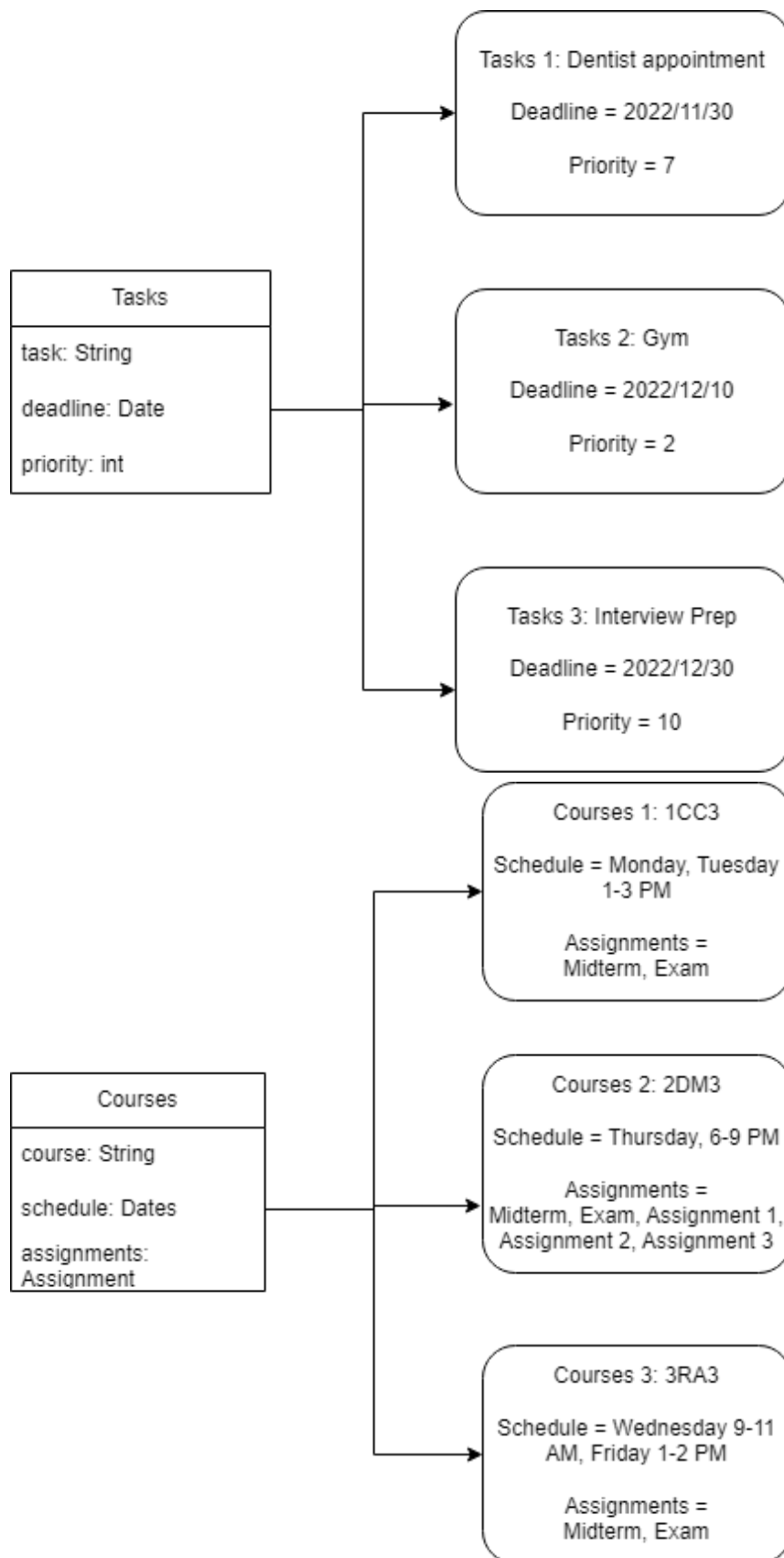


Figure 3. Decomposition of the subsystems - object diagrams

Sequence diagrams for user actions are as follows:

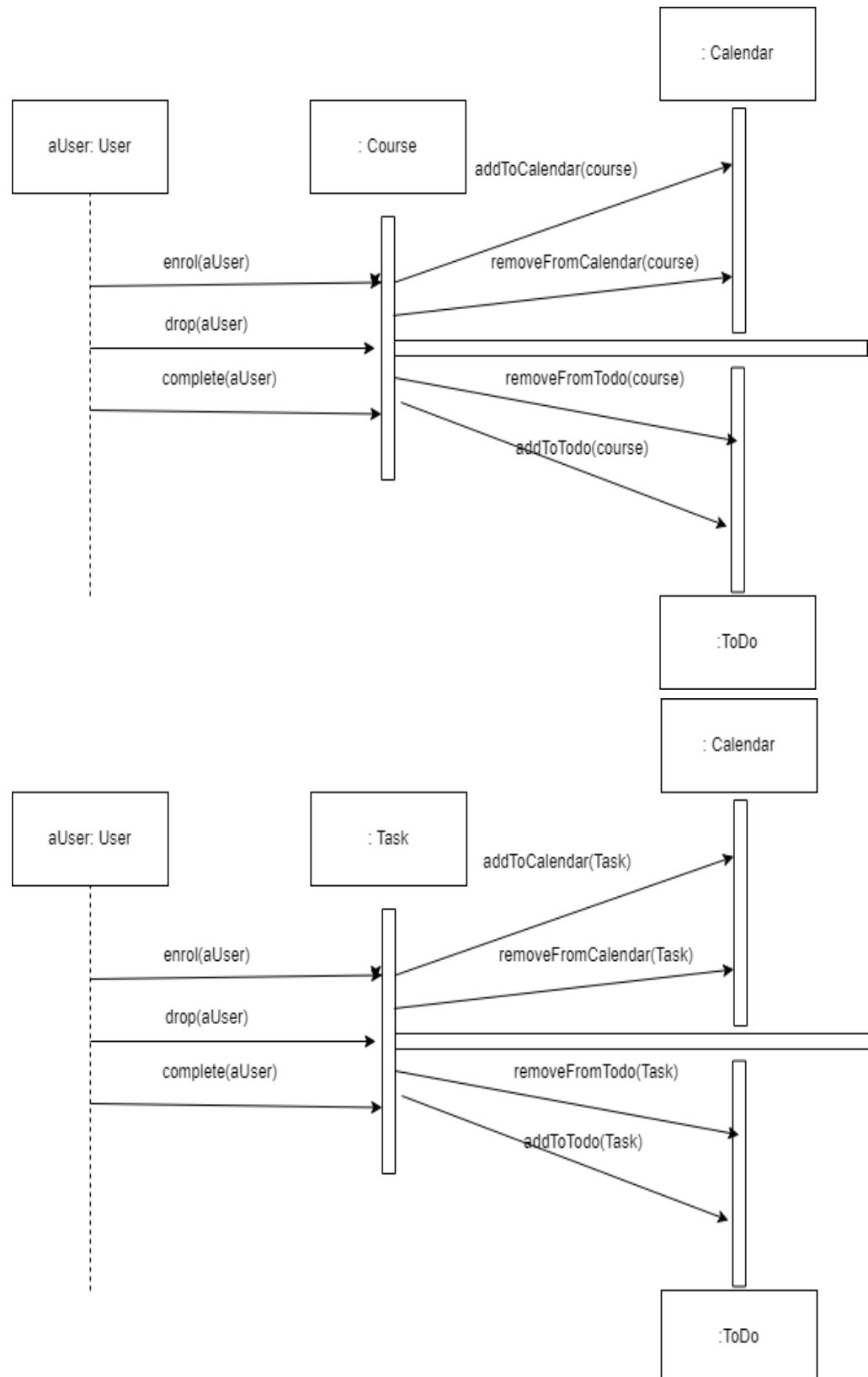


Figure 4. Decomposition of the subsystems - Sequence diagrams for user actions

3.3 Design Rationale

The development environment was chosen due to the stack serving all the requirements of the program. A top down system as such with the user and academic overview as the top

level makes the most sense, as every additional function implemented will be a feature of the academic overview. This allows for flexibility in implementing additional modules if needed in the future.

4 DATA DESIGN

4.1 Data Description

Explain how the information domain of your system is transformed into data structures. Describe how the major data or system entities are stored, processed and organized. List any databases or data storage items.

Users will enter information into the front end React browser application. React allows the data to be quickly updated and will rerender when anything changes. On the backend, the data is then Organized into three key structures. Users, Courses, and Tasks (see section 4.2 for detailed descriptions of these objects). Then, using Node.js, a JavaScript runtime, and Express, a Node.js framework to create a REST API, write middleware and handle requests the data can be passed and stored in MongoDB, a NoSQL database, as part of a Collection in a json format.

4.2 Data Dictionary

Attribute Name	Data Type	Required	Description/Methods	Example
USER OBJECT				
UserId	String	Yes	Unique Identifier for each User. Automatically assigned by MongoDB	507f191e810c19729de860ea
Email	String	Yes	User email. The application may communicate to the User via this address. Validation method runs on account creation. Must be of the form “example@domain”	mortensc@mcmaster.ca
Password	String	Yes	User password. Validation method runs on account creation. must be more than 12 characters, and contain a combination of upper and lowercase letters, numbers and symbols.	AeYr6m9kJyF8!
Username	String	Yes	The application will refer to the User by their username. Could be their real name or a nickname. Validated on account creation. Must be non-empty.	Camisha

Software Design Document

Courses	List of Courses	No	List of all User courses	[CourseId, CourseId, CourseId, CourseId, CourseId]
Tasks	List of Tasks	No	List of all User tasks (both academic and non-academic)	[TaskId, TaskId, TaskId, TaskId]
COURSE OBJECT				
CourseId	Number	Yes	Unique Identifier for each Course. Automatically assigned by MongoDB	5bf142459b72e12b2b1b2cd
Course Code	String	No	Shorter string abbreviation for the name of the Course. Must be less than 15 characters.	SWENG-3RA3
Course Name	String	Yes	Name of course. Validated upon creation. Must be less than 75 characters.	Software Requirements and Security Considerations
Course Color	Color	Yes	Unique color to identify the course across the application. Can be entered in any standard color format (hex, rgb). Final value will be converted to and stored as a hexadecimal.	#EE6352
Relevant Documents	List of Documents	No	List of Documents related to the course.	[Syllabus.jpeg, Reference.png]
Prof Name	String	No	Name of the primary professor of the course. Must be less than 50 characters.	Richard Paige
Prof Email	String	No	Professor email. Validation method runs on account creation. Must be of the form "example@domain"	paigeri@mcmaster.ca
Teaching Assistant Name	String	No	Name of Teaching Assistant. A single course could have multiple TAs. Must be less than 50 characters	Rachel Johnson
Teaching Assistant Email	String	No	Teaching Assistant email. Validation method runs on	johnsr12@mcmaster.ca

			account creation. Must be of the form “example@domain”	
Tasks	List of Tasks	Yes	List of all tasks associated with the particular Course.	[TaskId, TaskId, TaskId, TaskId]
TASK OBJECT				
TaskId	Number	Yes	Unique Identifier for each Task. Automatically assigned by MongoDB	507f1f77bcf86cd799439011
Name	String	Yes	Name of the Task. Must be less than 75 Characters	Assignment 4
Due Date	Date	Yes	Date of the Task. Could be the time of a lecture or the deadline of the assignment. Date validation occurs upon entry.	October 31st 2022
Relevant Documents	List of Documents	No	List of documents related to the particular task	[AssignmentDesc.pdf, Reference2.png]
Weight	Number	No	For academic tasks, a number that represents the weight of the task in the overall Course.	10%
Type	Lecture, Lab, Exam, Assignment, Non-Academic	Yes	Represents the Type of the Task. A task could be academic and belong to a Course, such as a Lecture or an Assignment. Or it could be Non-Academic entered by the user on the To Do or Calendar page such as a shift at work or a doctor’s appointment.	Assignment
Priority	Number	No	A numerical value assigned to a task	3
Earned	Number	No	A numerical value representing the earned grade for the task. Can be updated after the creation of the Task.	89%

Table 5. Data Dictionary

5 COMPONENT DESIGN

User Class

CLASS USER:

BEGIN

CONSTRUCTOR User(email, username, password)

BEGIN

user_email <- email

user_username <- username

user_password <- password

END

METHOD ResetPassword()

BEGIN

reset password

END METHOD

END CLASS

Overview Class

CLASS OVERVIEW:

BEGIN

CONSTRUCTOR OVERVIEW(username)

BEGIN

user_username <- username

CREATE courses <- []

CREATE tasks <- []

CREATE calendar

CREATE todolist

END

Software Design Document

```
METHOD CalcGPA()  
BEGIN  
    calculate gpa from course in courses  
END METHOD
```

```
METHOD generateSchedule()  
BEGIN  
    for task in tasks:  
        add to calendar  
        add to todoList  
    for course in courses:  
        add to calendar  
END CLASS
```

Course Class

```
CLASS COURSE:  
BEGIN  
    CONSTRUCTOR COURSE(name, schedule, assignments)  
    BEGIN  
        course_name <- name  
        course_schedule <- schedule  
        course_assignments <- assignment  
        isCompleted <- False  
    END  
    METHOD setCompleted()  
    BEGIN  
        isCompleted is True  
    END METHOD
```

END CLASS

Task Class

CLASS TASK:

BEGIN

CONSTRUCTOR TASK(name, deadline, priority)

BEGIN

course_name <- name

course_deadline <- deadline

course_priority <- priority

isCompleted <- False

END

METHOD setCompleted()

BEGIN

isCompleted is True

END METHOD

END CLASS

Todolist Class

CLASS TODOLIST:

BEGIN

CONSTRUCTOR TODOLIST()

BEGIN

create todoList <- []

END

METHOD addToList(activity)


```
BEGIN
    append activity to todoList
END

METHOD removeFromList(activity)
    BEGIN
        remove activity from todoList
    END
END

METHOD getFromList(index)
    BEGIN
        get index of todoList
    END
END CLASS
```

Calendar Class

```
CLASS CALENDAR:
    BEGIN
        CONSTRUCTOR CALENDAR()
        BEGIN
            create customCalendar <- []
        END

        METHOD addToCalendar(activity)
            BEGIN
                append activity to customCalendar
            END
```

```
METHOD removeFromCalendar(activity)
```

```
    BEGIN
```

```
        remove activity from customCalendar
```

```
    END
```

```
METHOD getFromCalendar(index)
```

```
    BEGIN
```

```
        get index of customCalendar
```

```
    END
```

```
END CLASS
```

6 HUMAN INTERFACE DESIGN

6.1 Overview of User Interface

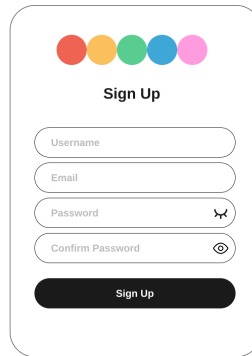
There are 6 core functionalities that our application will allow users to make use of.

- Register/Login
 - Upon launching the app the user will arrive at a landing page that will give them the option between Registering a new account (for first time users) and Logging into an existing account (for returning users).
 - The design of the Register and Login pages will be standard and familiar to users who have used similar applications.
 - The Register page will allow users to enter a username, email and password and then confirm that password. Emails will need to be in standard “example@domain” form. Passwords will need to be at least 12 characters long and include a combination of numbers, symbols, uppercase and lowercase letters. In the case that either of these conditions are not met or the password does not match the confirmation, the user will be given feedback through an error message highlighted in red which will alert them of what they are doing wrong.
- Add Courses
 - Once inside the application the first thing new users will be expected to do is navigate to the courses page and begin adding the details for all of their current or upcoming courses.
 - They will click the “Add a Course” button which will display a form where they can enter information about the course outline, class schedule, contact information, tags, assignments, exams etc. Each section will have clear instructions on what information the user is intended to provide and display

the answers that have already been entered, in the case of any bad inputs the form will give feedback through an error message highlighted in red.

- Once courses have been added, when the user returns to this page they will be able to quickly reference all of the details they have already entered and update them if anything changes and when they earn more marks. They will also be able to track their progress as they work through assignments and view all of their grades displayed graphically. They will also be able to calculate their current averages and predict final marks based on their remaining coursework.
- View Calendar
 - Once classes have been added, users may choose to view their schedule on the calendar page. This page will look and function in a way that users who are familiar with similar apps will find recognizable.
 - Users will be able to adjust the view of the calendar to suit their needs by making changes to the filtering menu on the side of the screen. They can choose between a monthly, weekly or daily view. They can filter which class' tasks are visible and which type of task is visible. Tasks associated with each class will be highlighted in a unique color. Users will also be able to add non-academic tasks to the calendar from this page (ex: work schedule, doctor's appointment) and edit existing tasks through a pop up form.
- View To Do List
 - Alternatively, users can review their tasks and class schedule in a To Do list view. This page will offer the same filtering menu and editing capabilities as the Calendar page as well as the ability to sort the tasks in different orders (chronologically, priority, etc.)
 - Users will be able to see both an expanded and compacted view of their tasks and be able to easily mark them as completed.
- View Dashboard
 - Users can also get an overview of all their course and task information on the dashboard view. The page features a weekly calendar view, a summary of current marks in all classes, quick links to recently referenced resources, degree planning information and any notes the user has written for themselves.
- Add Degree Plan
 - For users who would like to do some long term planning, the degree planning section will allow them to track course requirements for the completion of their full degree

6.2 Screen Images



Sign Up

Username

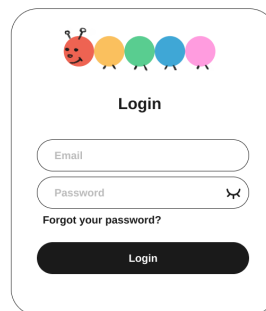
Email

Password

Confirm Password

Sign Up

Image 1. Screen Image - Sign Up Page



Login

Email

Password

Forgot your password?

Login

Image 2. Screen Image - Login Page

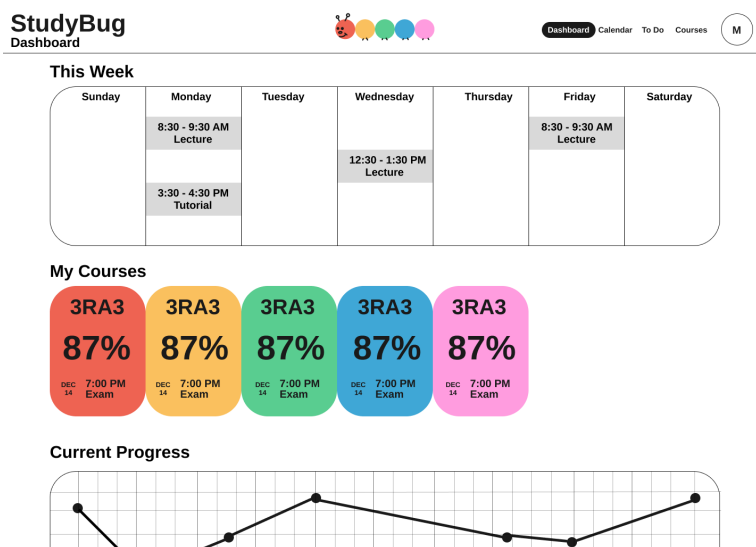


Image 3. Screen Image - Dashboard

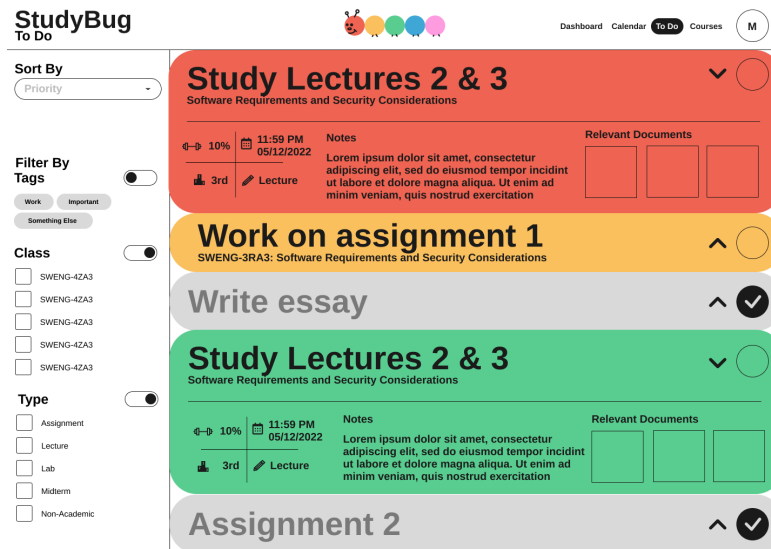


Image 4. Screen Image - To Do list

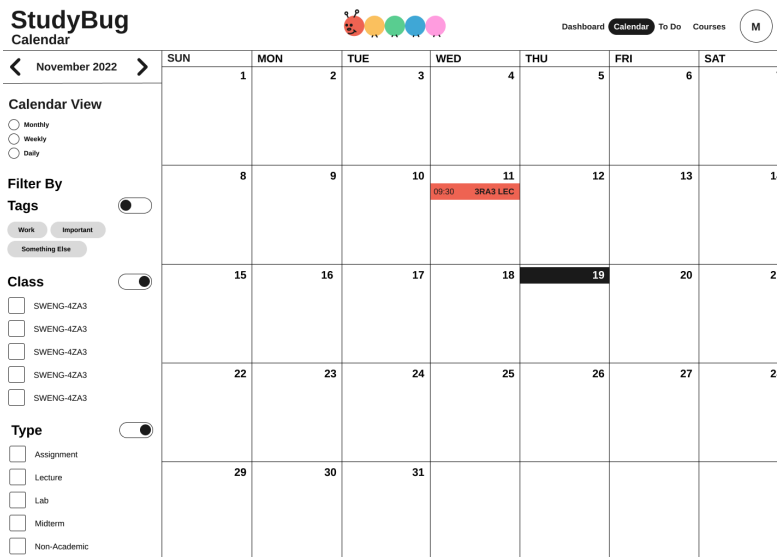


Image 5. Screen Image - Calendar



Image 6. Screen Image - Empty Course List Page

The screenshot shows the StudyBug interface. At the top, the StudyBug logo is on the left, a user profile icon labeled 'Misha's Courses' is in the center, and navigation links for 'Dashboard', 'Calendar', 'To Do', 'Courses', and a profile icon 'M' are on the right. Below the header, a list of five courses is displayed, each in a colored rounded rectangle. Each course entry includes the course ID, a brief description, the current grade (92% for all), and a table of upcoming assignments.

Course ID	Description	Current Grade	Upcoming Assignments
SWENG-3RA3	Software Requirements and Security Considerations	92%	DEC 14 11:59 PM Assignment 5 NOV 3 8:30 AM Lecture
COMPSCI-4ZP6A	Capstone Project	92%	DEC 14 11:59 PM Assignment 5 NOV 3 8:30 AM Lecture
MATH-2DM3	Discrete Mathematics	92%	DEC 14 11:59 PM Assignment 5 NOV 3 8:30 AM Lecture
COMPSCI-3FA3	Functional Programming	92%	DEC 14 11:59 PM Assignment 5 NOV 3 8:30 AM Lecture
STATS-3Y03	Probability and Statistics	92%	DEC 14 11:59 PM Assignment 5 NOV 3 8:30 AM Lecture

Below the course list, there is a black circular button with a white plus sign.

Image 7. Screen Image - Course List Page

StudyBug

Dashboard
Calendar
To Do Courses
M

Add a Course

Basics

Course Code

SFWENG 3RA3

Course Color

#FF9CDF

Course Name

Software Requirements

Syllabus

Upload Document

Administrators

Professor Name

Dr. Richard Paige

Professor Email

paigeri@mcmaster.ca

Teaching Assistant Name

Richard Paige

Teaching Assistant Email

paigeri@mcmaster.ca

Add Teaching Assistant

Weekly Schedule

Please enter your weekly class schedule including lectures, tutorials, labs, etc.

Sunday	Monday	Tuesday	Wednesday	Thursday	Friday	Saturday
	8:30 - 9:30 AM Lecture		12:30 - 1:30 PM Lecture		8:30 - 9:30 AM Lecture	
	3:30 - 4:30 PM Tutorial					

Day of the Week

Wednesday

Time

8:30 AM

Type

Lecture

Add Weekly Event

Add Tasks

Task Name

Assignment 3

Due Date

20/10/2022 5:00 PM

Task Weight

10%

Task Type

Assignment

Task Priority

???

Earned*

84%

Something explaining the units of the priority and what it's being used for.

Only fill in this field if you have completed this task and earned a mark.

Relevant Documents

Upload Document

Add Task

All Added Tasks

Assignment 3	16/11/2022 5:00 PM	↔ 10%	✍ Assignment	👤 ???
Assignment 2	01/11/2022 5:00 PM	↔ 10%	✍ Assignment	👤 ???
Assignment 1	84%	20/10/2022 5:00 PM	↔ 10%	✍ Assignment
Midterm 1	31/10/2022 8:30 PM	↔ 25%	✍ Midterm	👤 ???

Image 8. Screen Image - Adding Course Page

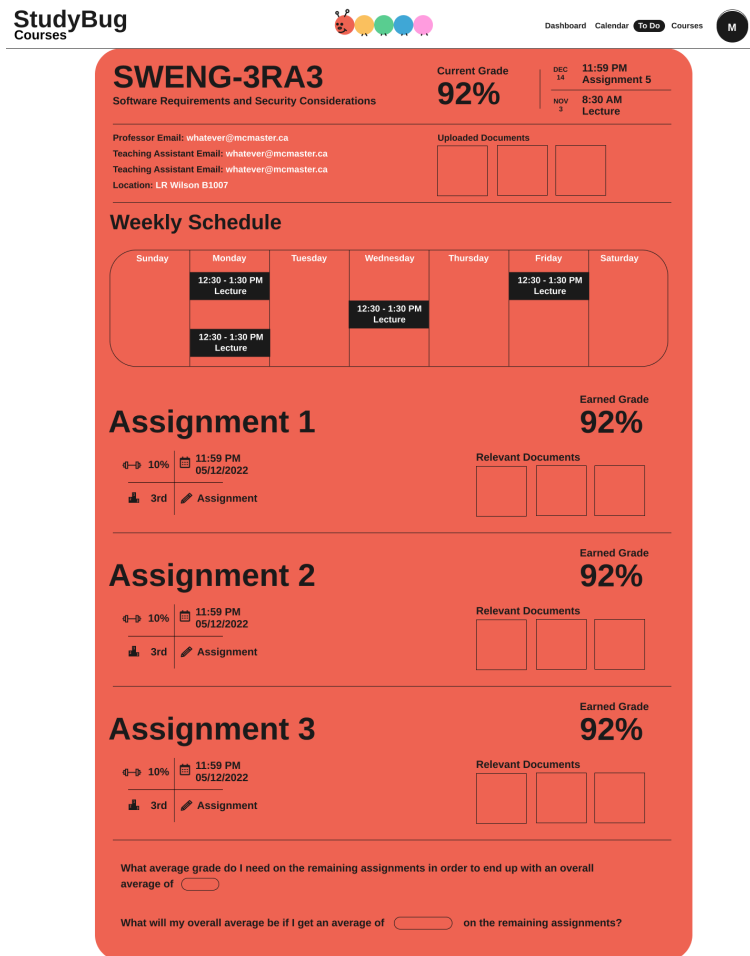


Image 9. Screen Image - Course Page

6.3 Screen Objects and Actions

A discussion of screen objects and actions associated with those objects.

NAME	LOCATION	DESCRIPTION/ACTION
TO SIGN UP	Landing	Links new users from the Landing page to the Sign Up page
TO LOGIN	Landing	Links returning users to the Login page
LOGIN BUTTON	Login	Logs the user in
SIGN UP BUTTON	Sign Up	Creates an account and signs the user in
NAVIGATION BAR	All Signed In Pages	Contains links to the Dashboard, To Do, Calendar and Courses pages and Settings Menu
ADD A COURSE BUTTON	Courses	Adds a new Course, pops up the

		Add Course Form
ADD COURSE FORM	Courses	Allows users to enter information about each course such as schedule, administrator contact information, relevant documents, assignments, exams, etc.
COURSE PREVIEW	Courses	Displays a snapshot overview of information about each course. Clicking takes the user to the full inner class page.
INNER COURSE PAGE	Courses	Displays all course information. Users are able to edit inputs as they earn grades, and track their progress.
FILTER MENU	To Do & Calendar	Allows users to filter the content seen on the by class, chronology, priority, type, tag, etc. Also includes a button to allow the addition of non-academic tasks.
ADD TASK	To Do & Calendar	Allows the user to add a non-academic task to either the To Do list or the Calendar
TO DO TASK	To Do	To Do list version of a task. Appears in a filterable/sortable list on the To Do page. Can be expanded or compacted, edited or marked as completed
EDIT TASK	To Do & Calendar	Pops up a form that allows the user to update the information in a given task
CALENDAR	Calendar	Displays all tasks in a filterable Calendar view. Can be shown in a Monthly, Weekly or Daily format. Clicking on Tasks will also open the editing menu. Clicking on blank space will create a new task in that time slot.
CALENDAR PREVIEW	Dashboard	Displays the calendar view of the current week.
COURSES PREVIEW	Dashboard	Displays an overview of the current courses

DEGREE PLANNING BUTTON	Dashboard	Pops up a form allowing the user to enter their degree planning information.
CURRENT PROGRESS PREVIEW	Dashboard	Displays a graph detailing the current progress in all courses

Table 6. Screen objects and actions

7 REQUIREMENTS MATRIX

**Please refer to:*

Software Requirements Specification for Academic Planner Program(del2.pdf)

Requirements Matrix Project Name: StudyBug Academic Planner					
Software Requirements Specification (SRS)				Software Design Document (SDD)	
Priority	Requirement ID	Main Requirement	Sub-requirement ID	Object Class	Components
High	1	Sign Up	SIGN-2	USER	user_email <- email user_username <- username user_password <- password
			SIGN-3	USER	User(email, username, password)
	2	Log In	LOGIN-1	USER	user_email <- email user_username <- username user_password <- password
			LOGIN-2	USER	User(email, username, password)
			LOGIN-3	USER	ResetPassword()
			LOGIN-4	USER OVERVIEW	User(email, username, password) OVERVIEW(username)
	3	Navigation Bar	NAV-1	OVERVIEW	OVERVIEW(username)
			NAV-2 NAV-3	CALENDAR	CALENDAR()

Software Design Document

			NAV-4 NAV-5	USER	User(email, username, password)
	4	Dashboard	DASH-1 DASH-2	OVERVIEW	OVERVIEW(username)
			DASH-3 DASH-4	OVERVIEW	CalcGPA()
			DASH-5 DASH-6 DASH-7	OVERVIEW	OVERVIEW(username)
			DASH-8 DASH-9	OVERVIEW COURSE	OVERVIEW(username) COURSE(name, schedule, assignments)
			DASH-10 DASH-11 DASH-12 DASH-13	OVERVIEW CALENDAR	OVERVIEW(username) CALENDAR()
			DASH-14	OVERVIEW TODOLIST	OVERVIEW(username) TODOLIST()
	5	To-do List	TODO-1 TODO-2 TODO-3 TODO-4	TODOLIST TASK	TODOLIST() TASK(name, deadline, priority)
			TODO-5	TODOLIST TASK CALENDAR	removeFromList(activity) setCompleted() getFromCalendar(index)
	6	Timetable	TTABLE-1 TTABLE-2 TTABLE-3 TTABLE-4	CALENDAR TASK	CALENDAR() TASK(name, deadline, priority)
			TTABLE-5	CALENDAR TASK TODOLIST	removeFromCalendar(activity) setCompleted() getFromList(index)
	7	Calendar	CAL-1 CAL-2 CAL-3 CAL-4	CALENDAR TASK	CALENDAR() TASK(name, deadline, priority)
			CAL-5	CALENDAR TASK TODOLIST	removeFromCalendar(activity) setCompleted() getFromList(index)
	8	Course Registration	COURSE-1	COURSE	course_name <- name course_schedule <- schedule course_assignments <- assignment isCompleted <- False
			COURSE-2	COURSE	COURSE(name, schedule, assignments)

Software Design Document

			COURSE-3	CALENDAR	addToCalendar(activity)
	9	Automatically Generate Tasks	GENETASK-1	TASK TODO LIST	TASK(name, deadline, priority) addToList(activity)
	10	Modify Tasks	TASK-1	TASK	course_name <- name course_deadline <- deadline course_priority <- priority isCompleted <- False
			TASK-4	TASK	setCompleted()
Medium	11	Course List	COURSELIST-1	COURSE	course_name <- name course_schedule <- schedule course_assignments <- assignment isCompleted <- False
			COURSELIST-2 COURSELIST-3	OVERVIEW COURSE	OVERVIEW(username) COURSE(name, schedule, assignments)
	12	Individual Course Page	COURSEPAGE-1 COURSEPAGE-2 COURSEPAGE-3 COURSEPAGE-4 COURSEPAGE-5 COURSEPAGE-6 COURSEPAGE-7	COURSE	To be updated
	13	Course Grade	RECORDGRADE-1 RECORDGRADE-2 RECORDGRADE-3 RECORDGRADE-4 RECORDGRADE-5 RECORDGRADE-6	COURSE	To be updated
	14	Calculate Grade	CALCULATEMIN-1 CALCULATEMIN-2 CALCULATEMIN-3 CALCULATEMIN-4 CALCULATEMIN-5	COURSE	To be updated
	15	Course Detail	COURSEDETAIL-1 COURSEDETAIL-2 COURSEDETAIL-3 COURSEDETAIL-4 COURSEDETAIL-5	COURSE	To be updated
	16	Contact Faculty	CONTACT-1 CONTACT-2 CONTACT-3 CONTACT-4	COURSE	To be updated
	17	Upload Course Outline	OUTLINE-1 OUTLINE-2 OUTLINE-3 OUTLINE-4	COURSE	To be updated

Low	18	Generate Study Plan	GENERATETT-1 GENERATETT-2 GENERATETT-3 GENERATETT-4 GENERATETT-5	OVERVIEW TASK CALENDAR TODO LIST	generateSchedule() TASK(name, deadline, priority) addToList(activity) addToCalendar(activity)
-----	----	---------------------	--	---	--

Table 7. Requirements Matrix