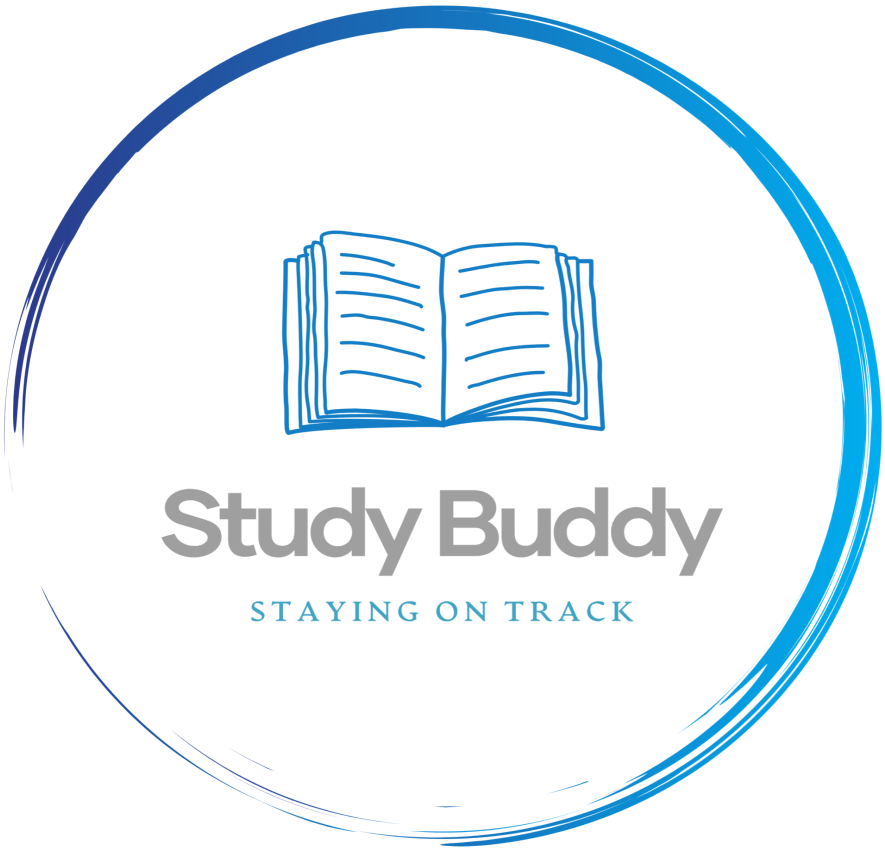


Study Buddy

Online Study Tracker

CM2020 Agile Software Projects - Team 31 - September 2021

Table of Contents

Background

Project Proposal

Study Buddy is an App that assists students with organising their studies. Students can create their own accounts and add the UoL/Coursera modules they are currently studying.

- The App will create an overview of the Modules in a list and provide the students with a personalised overview including tasks, deadlines and exam dates among others.
- The App will also be able to send reminders in an email or pop-up message form.
- The App will provide the student with general information about the module itself in a module page.
- Upon finishing the module students can remove it from their list and add the ones they chose for the next term.

Team Principles

The team consists of 5 individuals with heavy workload in their personal as well as professional life. Having already had experiences of project successes and failures in our professional lives we initiated working on the Final Assignment by defining our team principles.

Trust among the team had already been established during the Midterm Assignment phase and we had built up the confidence of a high performing team.

What we learnt about how we work together was that we had to be very careful around planning and attending in order to accommodate each other. Moreover, we have a big experience gap within the team when it comes to programming. When it comes to principles we agreed the following during our first meeting:

Alignment
Life happens! We know where we are in the process, what are the next steps and we inform each other as soon as possible if we are unavailable.

Accountability
We know exactly which tasks are assigned to us and we finish the meetings shortly telling each other what we will do until we meet again. We are not afraid to speak up, take on daunting tasks and ask for assistance.

Assistance
The team is here to cover for each other. We communicate difficulties as soon as possible so that we can get help.

Key Objectives

The objectives of the Online Study tracker stem from a quite diffused issue among students that in order to map their current status and identify a strategy to optimize next actions have to spend a consistent amount of time integrating information that is normally sparse across different documents, applications and even information systems in some cases.

- Reduce time students spend on non-value adding tasks, thus freeing up time for studying.
- Provide overview of key information about courses and study paths.
- Assist the student with planning the term by providing a progress monitor, overview of actions to be taken, schedules as well as reminders.

The system must provide a simple process to collect information about courses along with any specific deadlines, descriptions, documents and provide the set of instruments needed to enrich such information such as tasks and finally provide access to an environment that allows to respond to the following key questions:

- In which week we are
- when are things due
- when is this due
- how much time do I have

MVP Scope

Based on the outcome of our Research we decided to focus on the top requested features for our Minimum Viable Product, for the product we deliver to be impactful.

Moreover, we will focus on students/users currently studying BSc Computer Science at the University of London, as this has been our sample for the research.

Components in Scope

- Front End for responsive Website as well as Mobile access
- REST API Back End Service to handle Front End and Data
- Data Layer

Modules in Scope

- User Module
- Course Module
- Task Module
- Calendar Module

Features in Scope

- Register
- Log In
- Add Course
- Remove Course
- Course Overview
- Monitoring Progress
- Search functionality
- Writing Reports
- Submitting Assignments
- Handle Task Due Dates
- Add Calendar Events
- Remove Calendar Events
- Calendar Overview
- Google Calendar Integration
- Outlook Calendar Integration
- Time zone handling

Out of Scope

- Security
- Administration
- Analytics
- SEO
- System Monitoring
- Integration with Coursera Platform (Coursera requires affiliation program and a lengthy process to get access to their API, which we will not be able to in the provided time frame. https://building.coursera.org/developer-program/)

Planning

Team

Composition

Our team consists of 5 individuals with different professional backgrounds and specialist skills. Having already worked together for the first half of the term we had built rapport and trust allowing us to work in an even better way. We decided to keep our weekly check in meetings on Mondays at 17:30 CEST for 30 mins, extending whenever necessary. The choice of day and time was in order to accommodate constraints around working patterns, as well as time zone differences since the team was residing in 3 continents.

Roles and Responsibilities

Having a good understanding of what the task was and building on the experience we had accumulated by working together the first half of the term we met to distribute the roles and amend the RACI model we had already utilised.

The RACI model is a straightforward tool used for identifying roles and Responsibilities and avoiding confusion during a project. The Acronym RACI stands for:

- Responsible: The person who does the work to achieve the task. They have responsibility for getting the work done or decision made. As a rule this is one person.
- Accountable: The person who is accountable for the correct and thorough completion of the task. This must be one person and is often the project executive or project sponsor. This is the role that responsible is accountable to and approves their work.
- Consulted: The people who provide information for the project and with whom there is two-way communication. This is usually several people, often subject matter experts.
- Informed: The people kept informed of progress and with whom there is one-way communication. These are people that are affected by the outcome of the tasks, so need to be kept up-to-date.

Source: https://www.projectsmart.co.uk/raci-matrix.php

For each planned activity we discussed what our roles and responsibilities are, how the sub-teams will be formed and who is accountable for reaching out to team members for input as well as for delivering the final piece of work for each activity. The agreed split can be seen below:

Domain	PM (Andreas)	PO (Max)	Tech Lead (Robert)	UX (Swalin)	QA (Jorge)
Process	A/R	C	I	I	I
Product	C	A/R	C	C	C
Front-End	I	A	C	R	I
Back-End	I	C	A/R	I	I
Testing	I	I	C	C	A/R

The above does not mean that the team members were limited to their roles. The team was working in a very collaborative manner with our Slack channel being very active and team members assisting each other.

Project Phases

In an effort to optimise the process as well as equally distribute the areas of responsibility we took a holistic approach and split the project into 6 Phases.

1. Planning
2. System Analysis
3. System Design
4. Development
5. Testing
6. Implementation

The first 3 Phases of the Project include the activities the team must perform in order to reach the Project Proposal milestone (Midterm Assessment) and the rest of the Phases correspond to the delivery of the MVP of the product (Final Assessment).

Project Management

In order to manage this agile development project we were faced with the question which strategy we would choose. The whole team already had experience using scrum, kanban and waterfall and we had a very interesting dialogue on what would be the best approach for the task in hand. We chose a hybrid, combining elements of both scrum and kanban along with a waterfall tool. The rationale explained below.

Scrum

In order to utilise our resources (time) more efficiently we created a backlog of tasks which then got bundled up and assigned to two sprints.

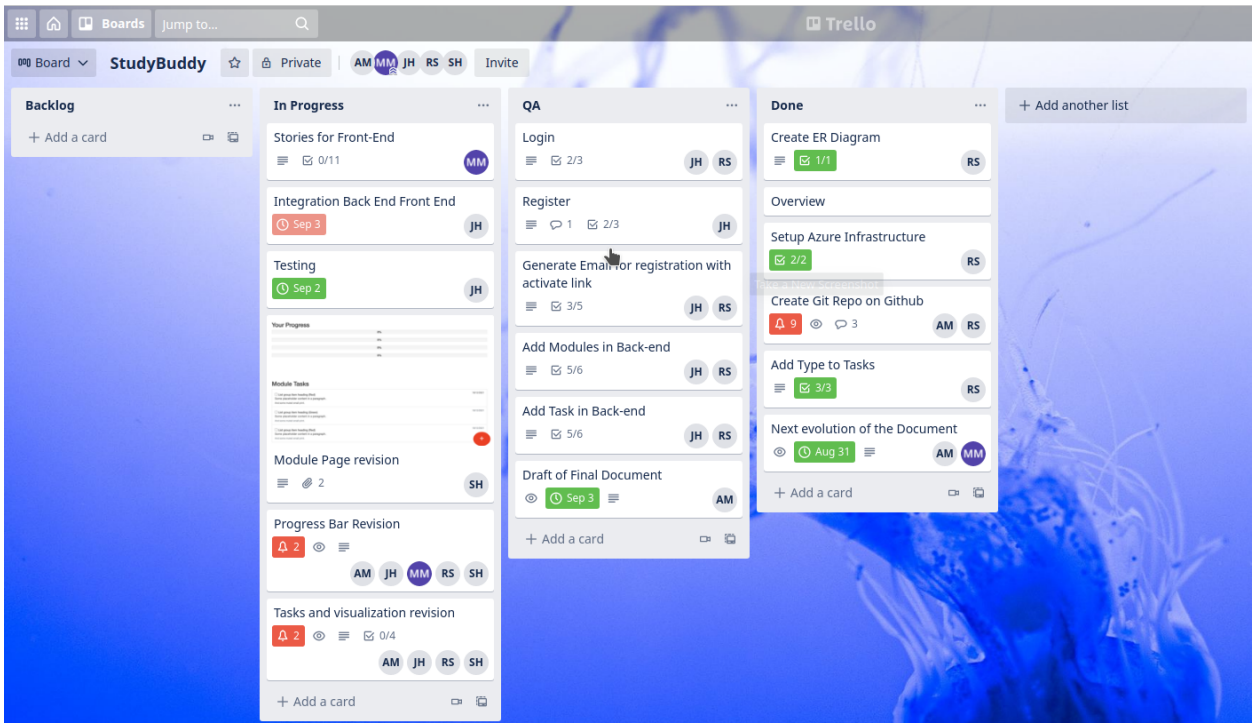
Sprint 1: Log-in & Sign-Up
Spring 2: Overview & Add Module

With this split of sprints we made sure that we would be working towards delivering a working product after each iteration.

Kanban

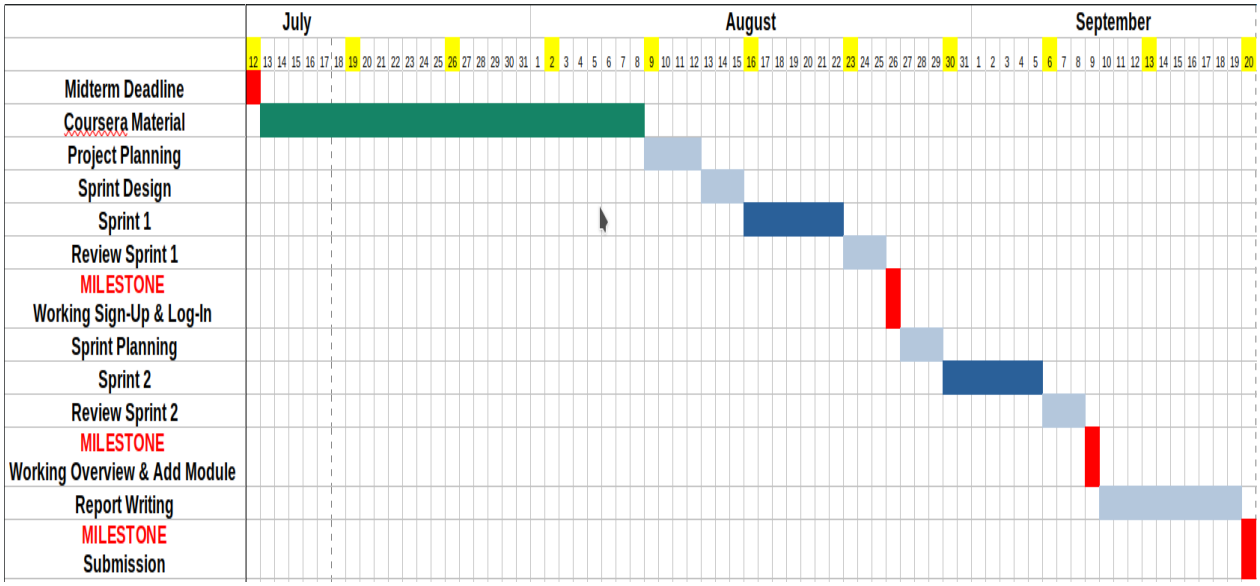
An element of Kanban that we decided to implement was a Trello Board. This was important in order for the team to have a working workflow and visualise the bigger picture. Although the Scrum Sprints are indeed self sufficient there was a need for all involved parties to avoid tunnel vision and maintain an understanding of the whole process. The steps on the Board were 4:

1. Backlog
2. In Progress
3. QA / Testing
4. Done



Waterfall

Last but not least, although the setup was sufficient we realised that both systems do have limitations when it comes to dealing with time and overall project deadlines. Although the deadlines are set for each task separately, an overall time planning became imperative if we were to adhere to the strict deadline of our final submission. We used a traditional Gantt chart to make our general planning and plotted the Sprints, as well as revisions in it, along with Milestones and buffer time to account for Risks.



Key Requirements

Functional Requirements with acceptance criteria

Below is a use case with key requirements defined as user stories using the S.M.A.R.T. paradigm. (Simple, Measurable, Achievable, Realistic, Trackable)
The priority column will be used during execution to create a timeline of activities.

Requirement	Description
UC 01	As a student/user I want to be able to register and create login credentials, works when I am able to provide an email and I am then guided through the selection of the password by providing a minimum of 8 characters, containing one capital, one number and a special character. Last step, I want to receive a confirmation email that allows me to activate the account.

UC 02	As a user I want to be able to log into my created account, which is linked to the email address I provided which works when I use my registered email address and selected password and I can see my personalized dashboard.
UC03	After the first log in the user should be prompted to start adding modules, works when after the main page is reached for the first time the user is shown a message inviting to set up his/her first module by providing the minimum set of information i.e. Title and Short Description.
UC 04	For each module added the user should be able to set events such as deadlines to be notified, works when for each module created the user has access to an editing functionality that allows to add event based information along with details about notification rules.
UC 05	As a user I want to be able to easily change the progress of my modules, works when either form the main page or from the module details page I can change the progress through a fader
UC 06	As a student i want to be able to search easily through search engine and display list with all Modules, works when user enter the search term and the user can further filter the list based on various parameters as this benefits the user in reducing searching time

Non-Functional Requirements

Non-functional requirements are quality characteristics that specify how our application should behave. These are list of all our non-functional requirements

Performance, availability, response time, reliability and maintainability.

Requirements	NFR Description
UC 1	As a student, each request I submit should be processed within 10 seconds on the application.
UC 2	As a student, I want to receive Emails with a latency of no greater than 2 hours.
UC 3	As an Impatient student, I want to load the website within 3

	seconds when the number of simultaneous users is greater than 1,000.
UC 4	As a student, I want to be able to run your study tracking application on the latest version of Windows.
UC 5	As a student, I want the site to be available 99.999 percent of the time when I try to access it, so that I don't get frustrated and find another site to use.

Research

Persona (Example of a user)

Michelle is a Part-Time working Student that needs reliable application to keep track of her studying progress and can easily access all day long 24/7. Michelle does not have a reliable Management Interface which preserves track of her progress, schedule, due dates, since the whole process is done manually through (paper and pen) as she is responsible for, writing all the information by hand on paper and mistakes often happen.

So Michelle wants to be able to have an overview of her registered Modules (courses). Also she needs an efficient way to keep track of her exam, assignments due dates and deadlines.

User Questionnaire

We utilized an online questionnaire as our approach and methodology for gathering data from possible end-users as:

1. It's fast and efficient in gathering data from participants who are located in remote locations.
2. It is appropriate for the nature of our intended end-users (E-learning Students), who are overwhelmed with the unformatted class dates, assignments, and examinations, and who are constantly forgetting a lecture or upcoming exam.
3. With few resources, it is possible to reach a large number of end-users.

Noting that our major goal and purpose in performing such research is to produce an easy-to-use application with an interactive interface. And to obtain a knowledge of the requirements, needs and behaviors of diverse end-users.

The following questions were asked to each user:

- Please select your age range
- Please Select your Learning Style
- Are you a full-time student or do you have a (part/full time) job?
- How Many modules are you studying in each term?
- How do you track your Studying progress?
- In case of using a third party app. to track your study progress, how frequent do you use it?
- How much time do you typically spend on tracking your study progress rate / Weekly? How interested would you be in a Study tracking app to keep tracking your learning progress?
- What are the most important features the Study tracker application should have?

- What kind of user-interface would you prefer the Study tracker application to be on?

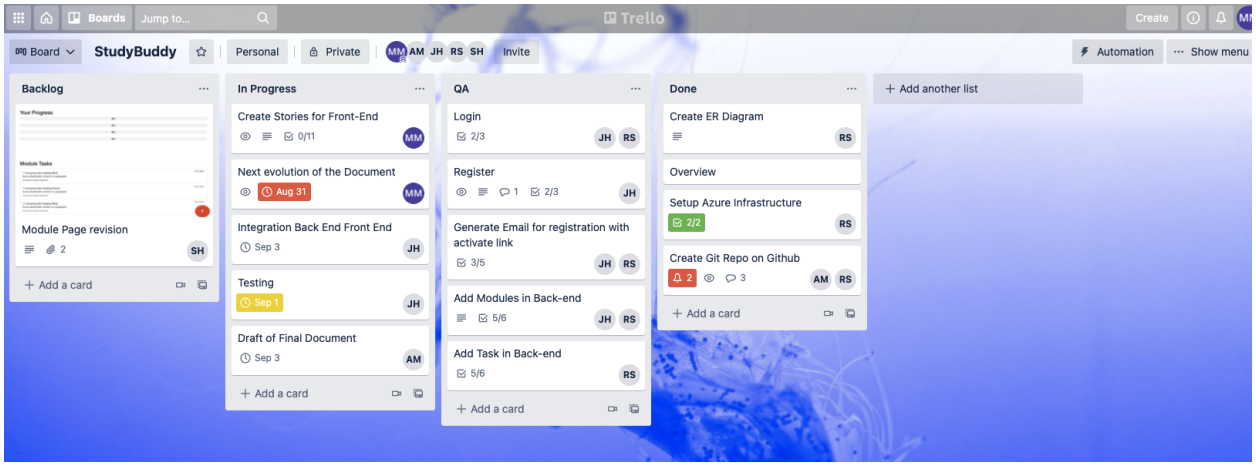
User groups

Based on our Questionnaire analysis and the responses of more than 40 participants to our questionnaire, we have concluded that our target consumers are as follows.

User group	Background	Use of the system	Environment	Main goals
Students	**Age:** 18+ Gender: All genders	**Usage:** All year round **User skills**: Beginner **Attitude:** Positive since it simplifies the way data is displayed and provides a better user interface **Number of users:** All Students	Technical environment: To have access to the device and an internet connection	Students Ability to log in/Register using the website, access to the Modules section to track his Studying Progress and his due dates.

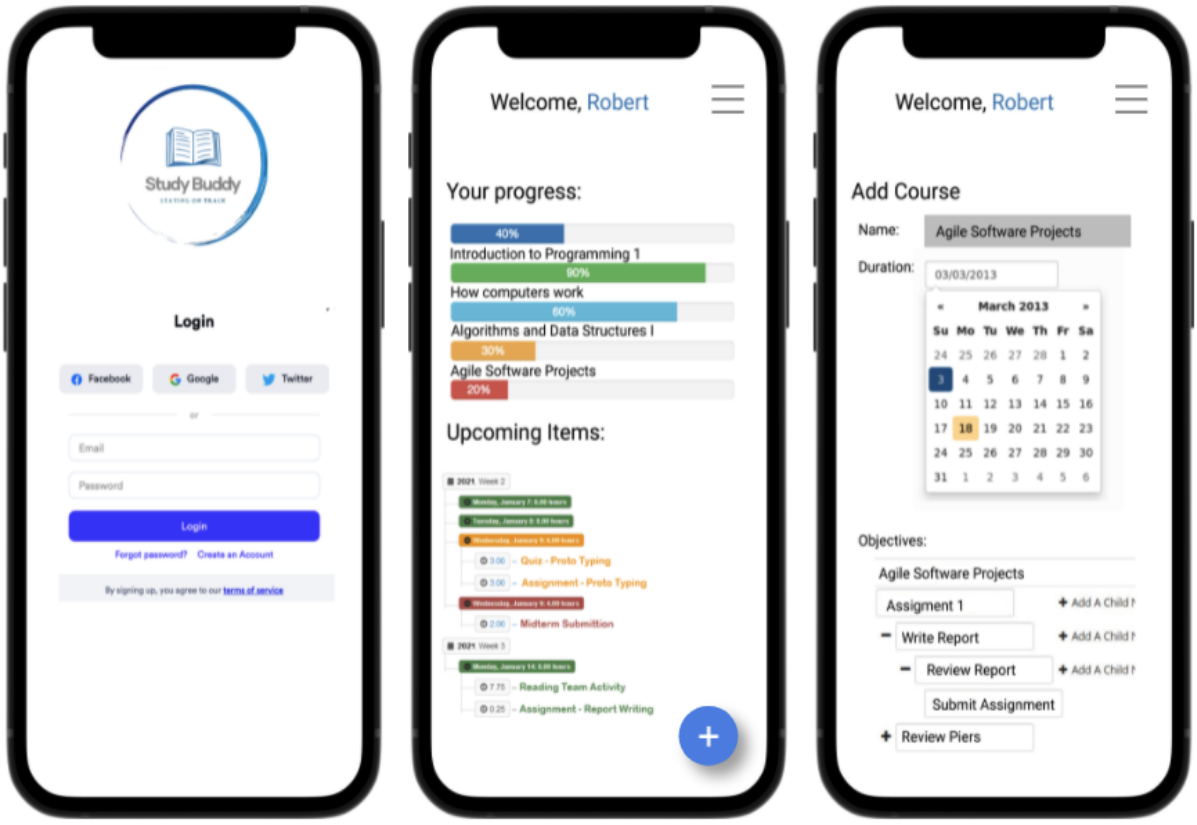
Prototyping and Iteration

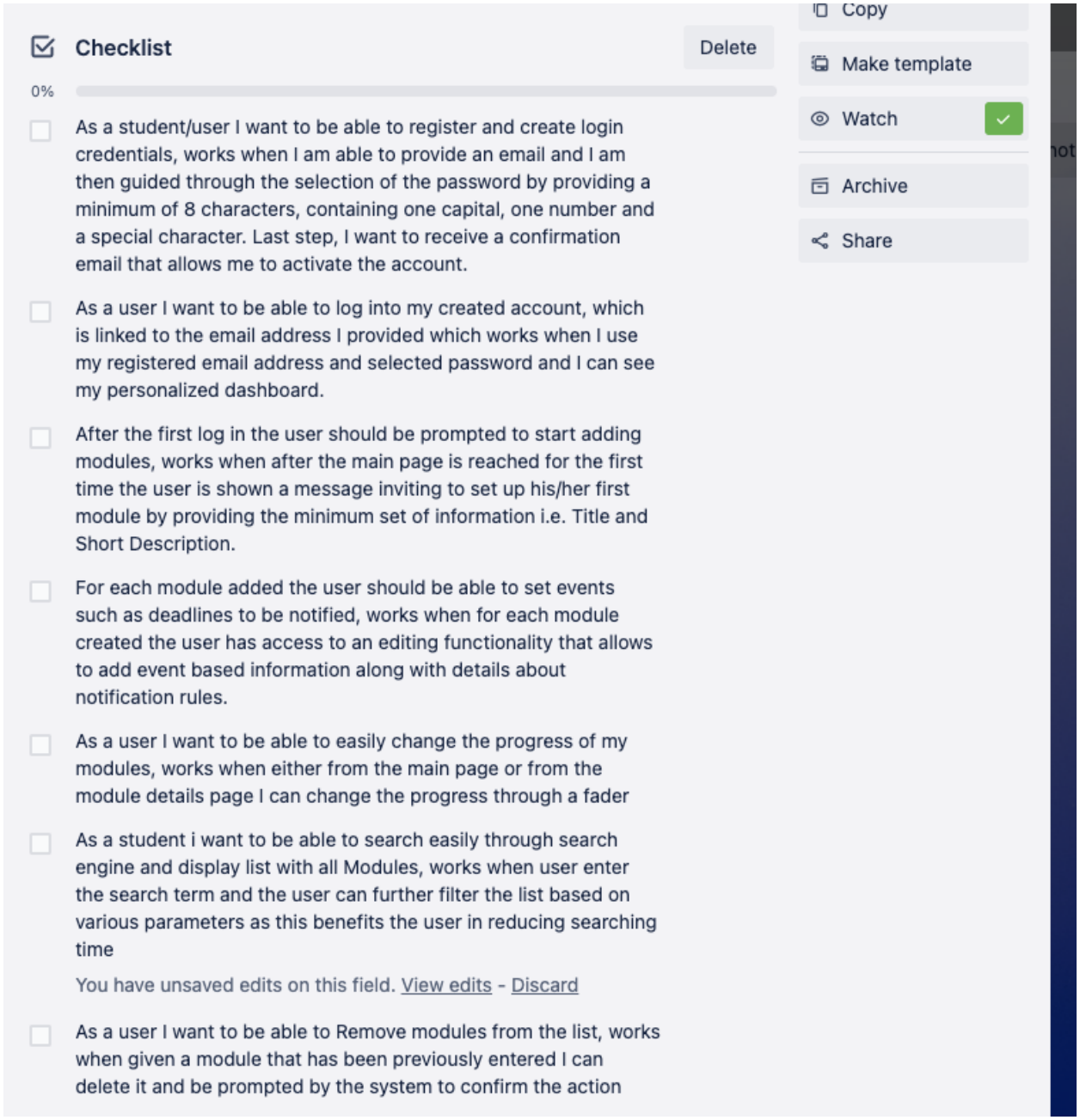
The prototype was built through two scrum iterations, below a screenshot of the kanban board taken as one of the first iteration was almost completed:



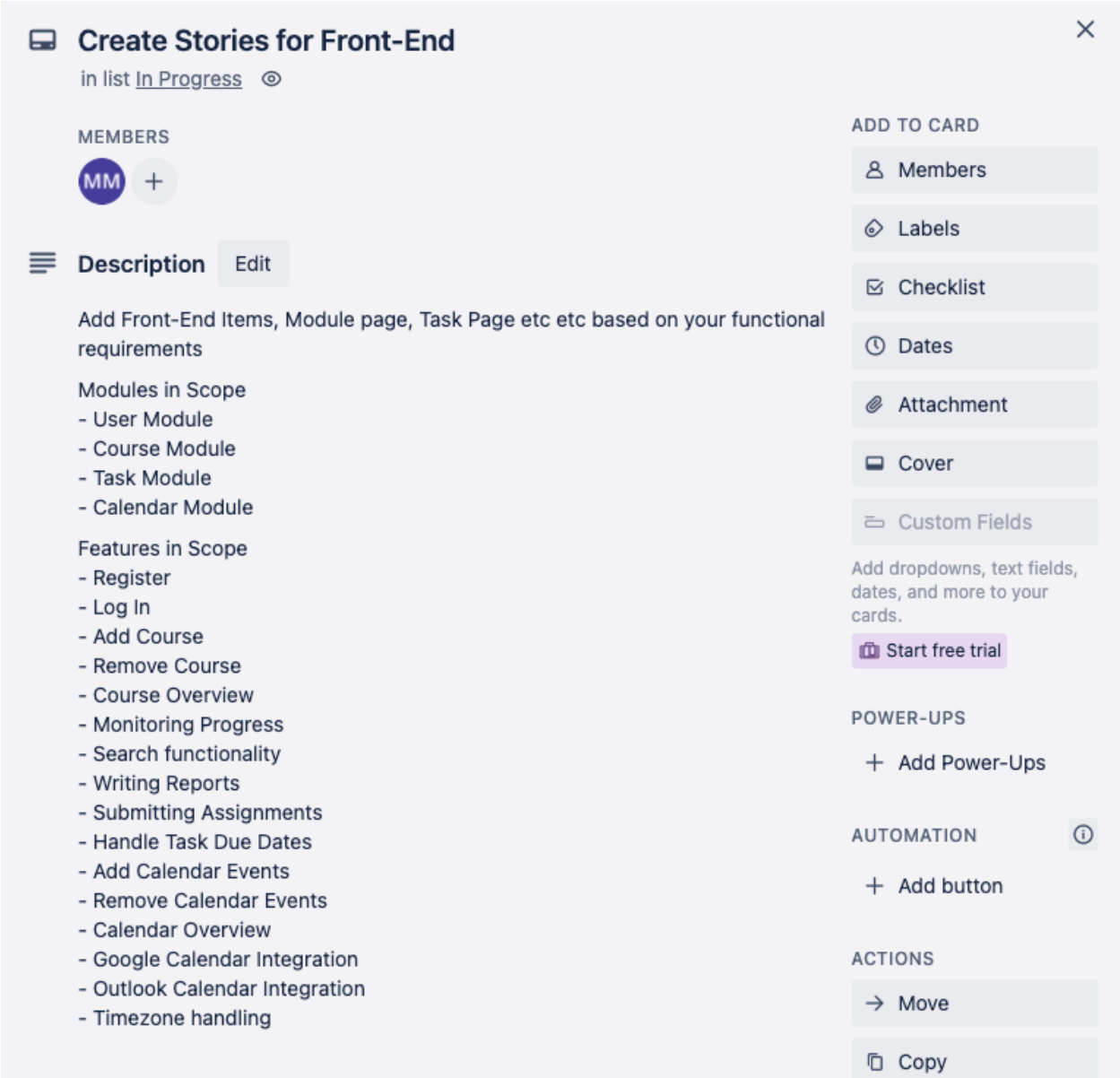
The activities were organized as user stories configured as tasks for each card as shown in the screenshot below.

Starting from the prototype submitted for Midterm we built an initial list of user stories that described the general functioning and the inner functioning of the system for all actors and all processes identified, during the following iterations discrepancies and issues where discovered and an incremental evolution approach was used in order to reach a level of functionality and usability that was satisfactory.
The team as described earlier has adopted a clear role model therefore it was fairly easy to understand the points that needed additional work and refine the level of granularity of description when needed.
The final step was to make sure that the system was compliant with the initial requirements and in line with the subsequent change requests therefore able to go through a final acceptance.


Study Buddy
Login
Facebook
Google
Twitter
Email
Password
Login
Forgot password?
Create an Account
Welcome, Robert
Your progress:
40%
Introduction to Programming 1
90%
How computers work
60%
Algorithms and Data Structures I
30%
Agile Software Projects
20%
Upcoming Items:
Quiz - Proto Typing
Assignment - Proto Typing
Midterm Submittion
Reading Team Activity
Assignment - Report Writing
Welcome, Robert
Add Course
Name:
Agile Software Projects
Duration:
03/03/2013
March 2013
Objectives:
Agile Software Projects
Assigment 1
Write Report
Review Report
Submit Assignment
Review Piers



In order to align the developments with the scope drafted for Midterm the scope features were detailed and used as guidelines as shown in the screenshot below.



Github was used to version control the code.

Design

Front-End

The design phase will utilize the user needs analysis in order to make the most effective product possible. This phase will focus on the specificity of the Study Buddy app (Study Tracking) and how it can be made in the most user-friendly way possible, as well as optimizing its efficiency. It is crucial in this phase that all targeted users (Middle Age Students) can use this app efficiently.

Design Purpose

Our Design major goal is to keep track of the necessary information to properly describe the architecture of our system in order to provide direction and guidance to the development team on the architecture of the system to be developed.

Our Front-end Designs are incrementally and iteratively produced during the *SDLC* (system development life cycle), based on the particular and targeted user feedback.

Design Languages

After several meetings and discussions, we made the decision to employ these front-end languages.

1. **HTML5** will be used for the structure of the page and will consist of a series of different elements to show the intended structure of the page. It also provides a number of benefits that will support our project.

    a. The browser would be *unsure and unable* to display the content correctly, without HTML elements and structure.
    b. It's Ability to Integrate with other languages.
    c. Simple to use

2. **CSS3** will describe how the HTML elements on the site are displayed and presented onto the screen. It also provides a number of benefits that will assist our project.

    a. Using CSS can *save and reduce the amount of work*. This can be done by controlling the layout of multiple web pages all at one time.
    b. Multiple Browser Support
    c. Speed

3. **JavaScript**, one of the most important and popular of front end development languages due to:-

a. Its ability to *create dynamic and responsive web pages*. Not only are interactive web pages fantastic for user experience, but this useful addition can also play dividends when it comes to paid advertising.
b. Its capability to give you Extended Functionality.

4. **JQuery** is a JavaScript library that allows us to write less code and create sleeker sites. With a unique ability to simplify programming, it's more favourable due to:-

a. Its ease and user-friendliness.
b. Cross-browser compatibility.
c. Fast page Speed.

5. **JSON** is JavaScript Object Notation. It will be used for serializing and transmitting structured data over a network connection. It's primarily to transmit data between a server and web application. It is a much-more compact way of transmitting sets of data across network connections as compared to XML

Back-End

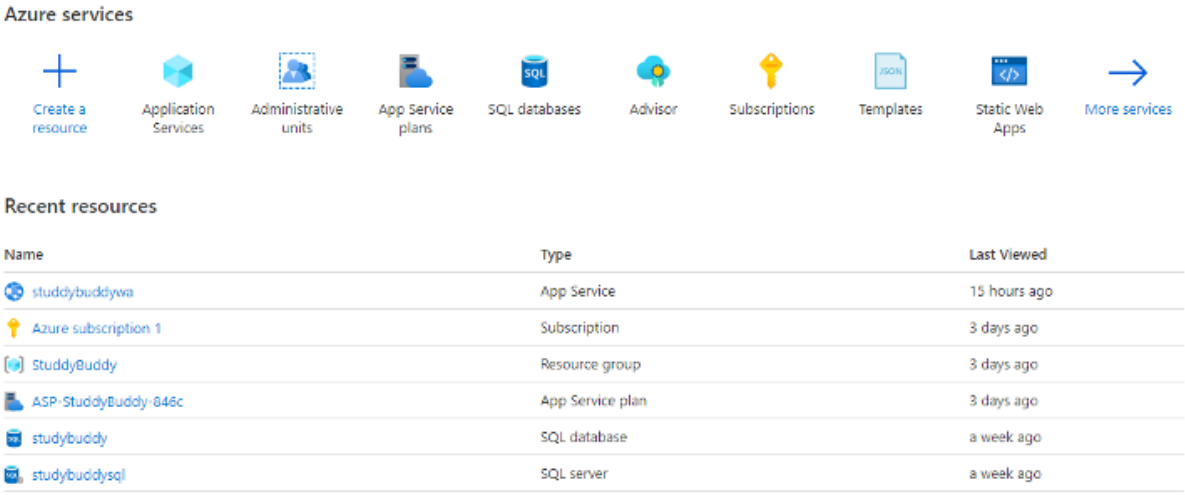
Technologies

Once we decided on the application we are building, we needed to find a way to make this accessible to all users, making it as easy as possible for the evaluator of the software to access it with the least amount of effort. The Azure platform will also be used for the final product we launch to the public.

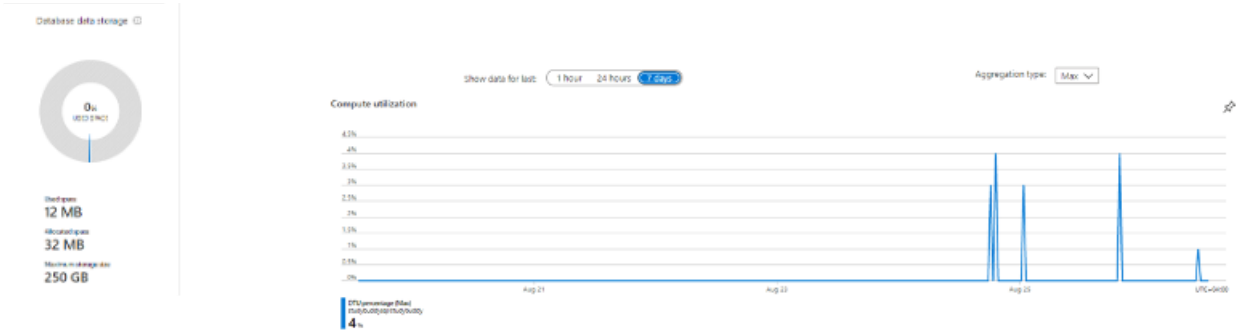
We decided to go with the Microsoft Stack for hosting as this offers us a free trial and, based on the internal skill set of the team, was the quickest go to market strategy to implement.

We evaluated and chose Azure for hosting the application, as illustrated below, as this offered us the ability to create an App Service to host our business layer of the application by using a .NET Core API. Our Data Layer resides in a SQL Server with a SQL database component.

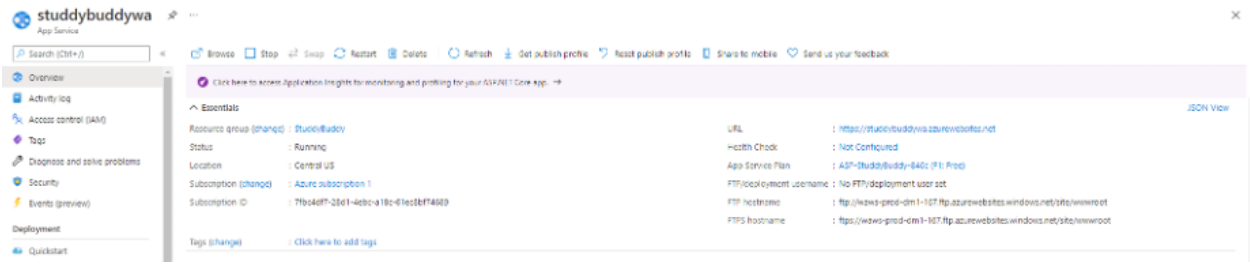
Security and access are all controlled via the resource group in Azure, making it quick and easy for us to manage who has access to our services being hosted in the cloud. We have nth level control down to controlling which countries can access the hosted application and users.



Azure also allows us to monitor traffic to the application and be pre-emotive in attacks to our service or servicing the required load based on the popularity of our application, and we are hoping that the latter will happen, in which case we can easily upscale our solution.



The Azure Dashboard provides us monitor and configuration ability on the fly.



The technologies used for the core component of the application is as follow:

- Hosting
  - Azure Hosting

        - App Service
        - SQL Server
        - SQL Database
        - Resource Group
- Security
    - Resource group control
- Data Layer
    - SQL Server
    - SQL Database
- Back-end REST API service
    - App Service Hosting container
    - .Net Core API, which is platform-independent
- Presentation Layer
    - HTML / Bootstrap CSS / JS / jQuery / JSON

During the evaluation phase of the technologies, we had to make decisions based on the team's expertise and ease of implementation and turnaround time in getting us started as quickly as possible due to tight timelines and deadlines.

Hosting analysis and decisions

We chose to go with Azure as this offered us a free trial period of a year, and team members had experience with the platform. Amazon Web Service and Platform had a cost to it, and limited internal expertise was available. Our assessment was as follow:

- Azure:
    - Expertise available
    - Free 12-month trial
    - Ease of use and accessibility
    - Lots of Platforms available (Unix, Linux, Windows)
    - Security controls
    - Highly configurable
- Amazon Web Services
    - Platform-Specific
    - Lack of internal expertise
- Independent hosting
    - Long setup time
    - Not guaranteed uptime as hosted by team member
    - Security setup and access is complicated and requires cost
    - Not reliable

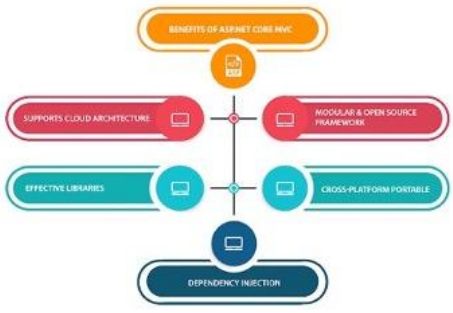
Data Layer Analysis and decisions

We knew from the start that we would be using a structured query language to store and access data, but we need to decide on the technology to use. We choose to go with SQL server as the choice for database and hosting this on an Azure SQL server. Our assessment was as follow:

- MongoDB
  - Transactions using MongoDB are complex
  - MongoDB is not a strong ACID (Atomic, Consistency, Isolation & Durability) compared to many other RDBMS systems.
  - Transactions using MongoDB are complex
  - In MongoDB, there is no provision for Stored Procedures or functions, so you can't implement any business logic at the database level, which you can do in any RDBMS system.
- MySQL
  - Transactions related to the system catalogs are not ACID compliant
  - Sometimes a server crash can corrupt the system catalogue
  - Stored procedures are not cacheable
  - MYSQL tables which are used for the procedure or trigger are most pre-locked.
- Azure SQL
  - Scalability and Beyond
  - Support Direct Integration with Developer Environment
  - High Speed and Minimal Down Time
  - Improved Usability
  - Intelligent Protection with added security
  - Internal Expertise within the Team

Back-end analysis and decisions

We need to find a relevant technology in the market and accessible for us to use and implement to get us started as quickly as possible to prevent delays between the front-end and back-end development. We decided to go with a REST-API service. We developed the signature and methods as a framework available to the front-end to start independently of the parallel back-end development. We decided to go with .NET Core API instead of Node.js for the following reasons:

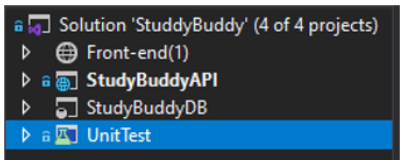


.NET Core API

- Support on Azure
- .NET Core is Cross-Platform
- .NET Core is Open Source
- Mature Framework and Widely Used Programming Languages
- The .NET Core Supports a Wide Range of Application Types
- Increased security with .NET Core
- .NET Core Enables Top App Performance
- .NET Core Enables Flexibility
- .NET Core is Cost-Effective
- .NET Core has a Large Community

The solution for our application consists of the following core components:

- Web Application
- Web REST API Service
- Database Project
- Testing project



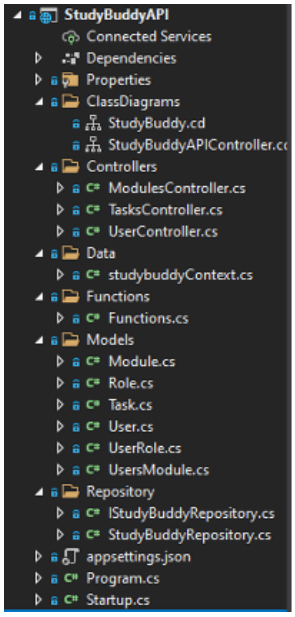
Please refer to the Front-end application section for more details on the structure and technologies used.

In planning for the back-end services, the first thing that was needed was to define the interface and the requirement methods based on the project's use cases, as this was the primary driver in the development of the application.

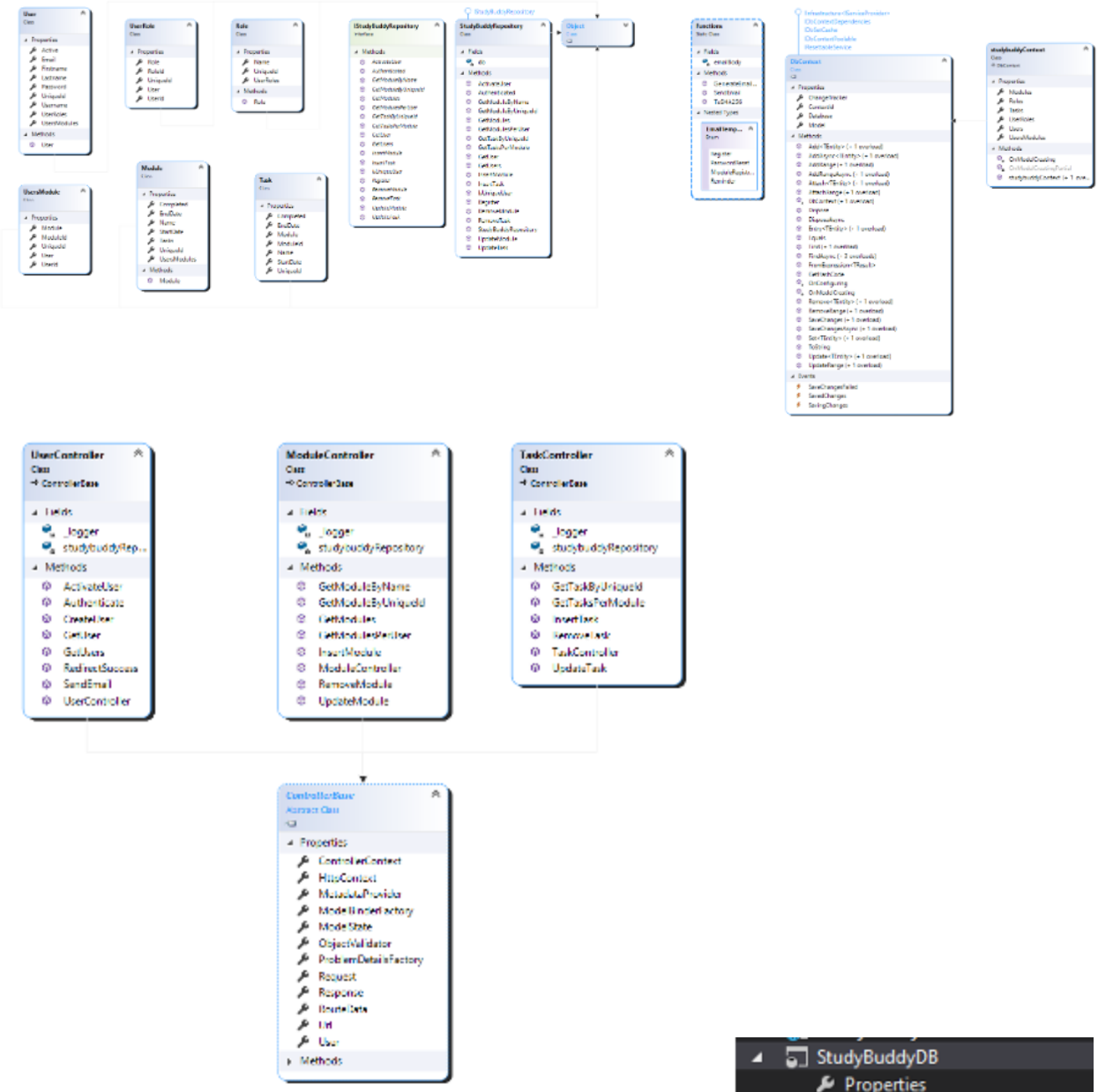
The design phase was the most critical part of the internal session we had as a development team. We used the Trello board to keep track of the development of each item.

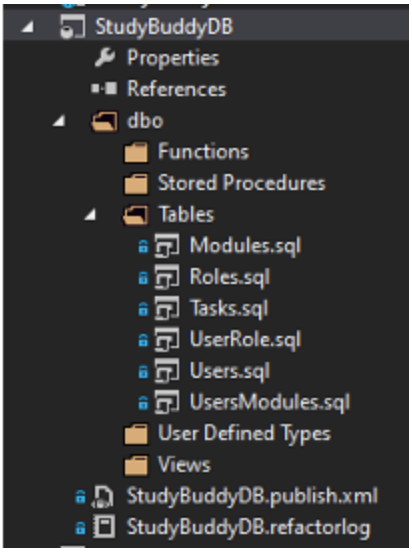
The .NET Core API project (**Rest API Service**) contains the following components:

- Interface with defined methods available for consumption by the front-end service
- Controller classes
  - User Controller (User functionality, CRUD)
  - Module Controller (Module functionality CRUD)
  - Task Controller (Task functionality CRUD)
- Repository Class (Handling Database connections and CRUD)
- Functions (Shared Library for classes to use standard functions)
- Start-up (configuration class to set security, available routes)



As part of the design phase, we created class diagrams to showcase how all the service components will fit together with our modular approach and interface design goals.





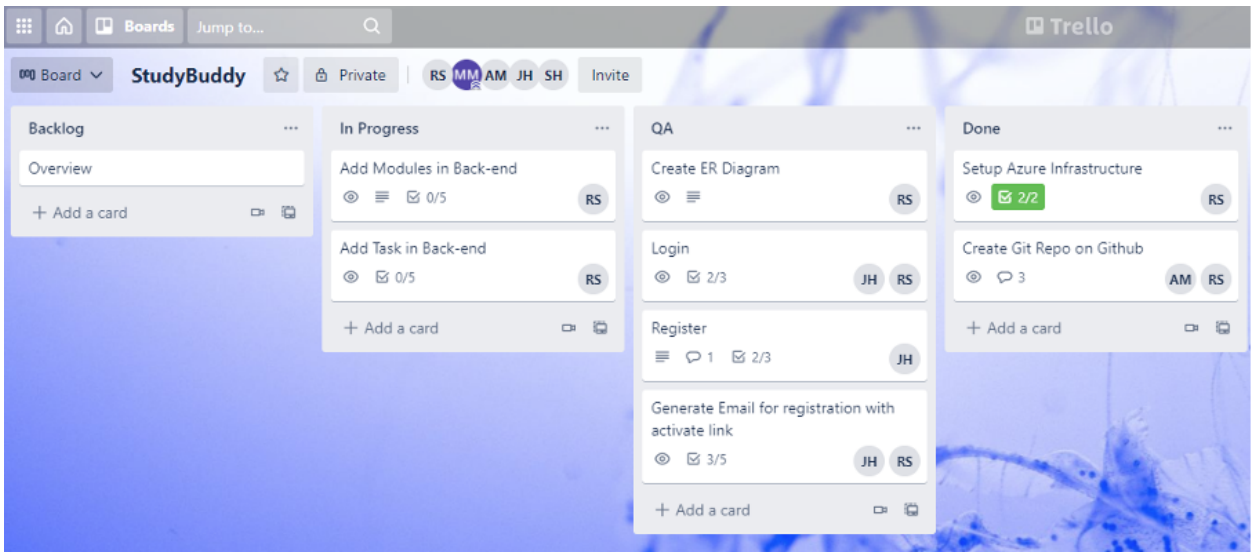
Database

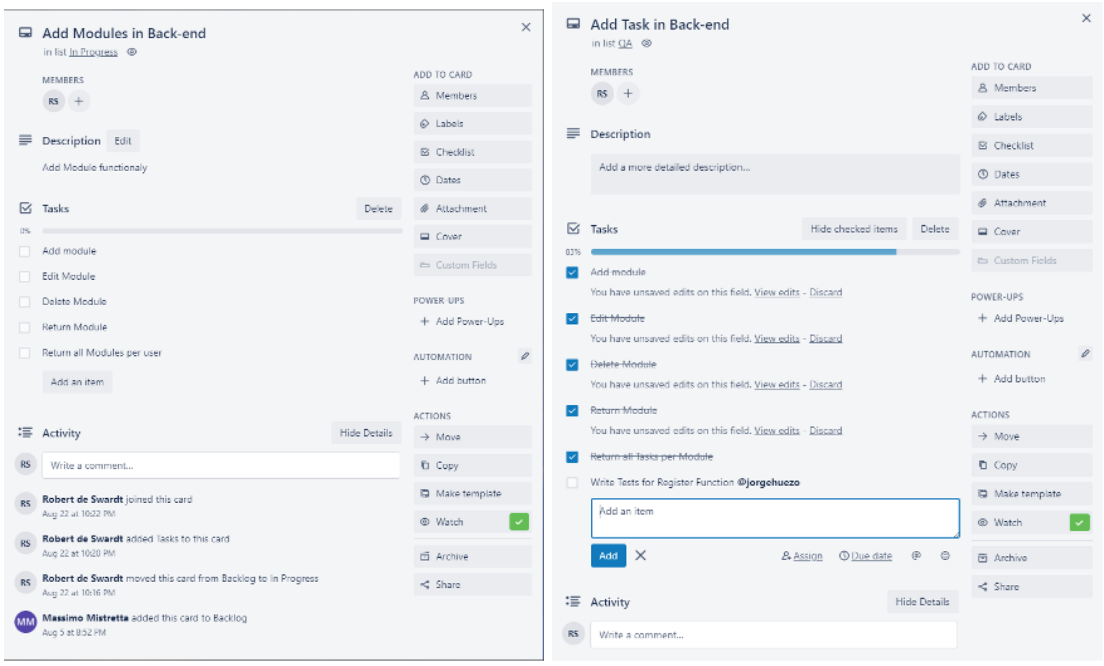
The **Database project** consists of the tables and their table definitions for our applications. The table definition contains the details on the data types and primary keys, and relationships between tables.

We created the project to allow us to implement changes quickly and to publish this to our hosted azure database server as we perform iterations on the project.

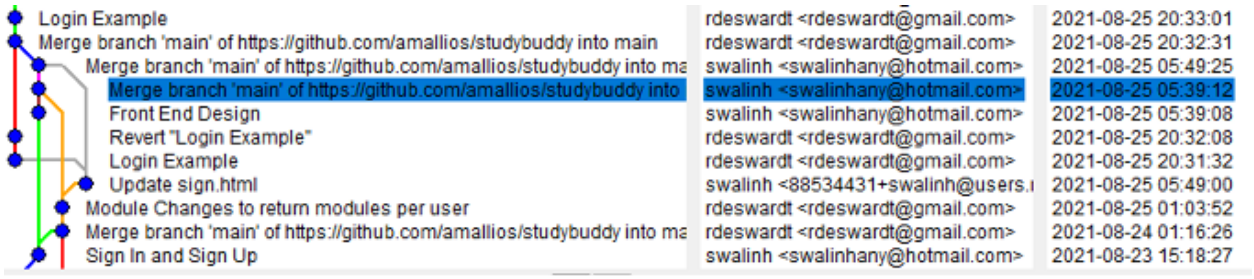
The database design can be seen in the ERD diagram below:

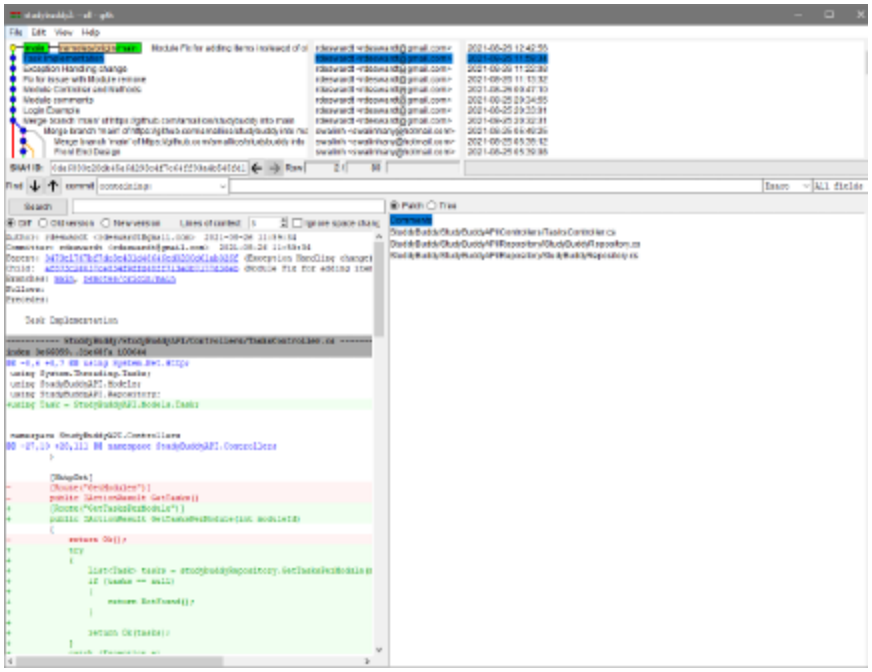
The process that was followed for the development lifecycle was using a sprint with 2-week iterations. We implemented a Trello board to keep track of tasks and progress on individual items. We as a team decided to use GitHub as our choice of repository to allow us all to work remotely on our respective tasks.



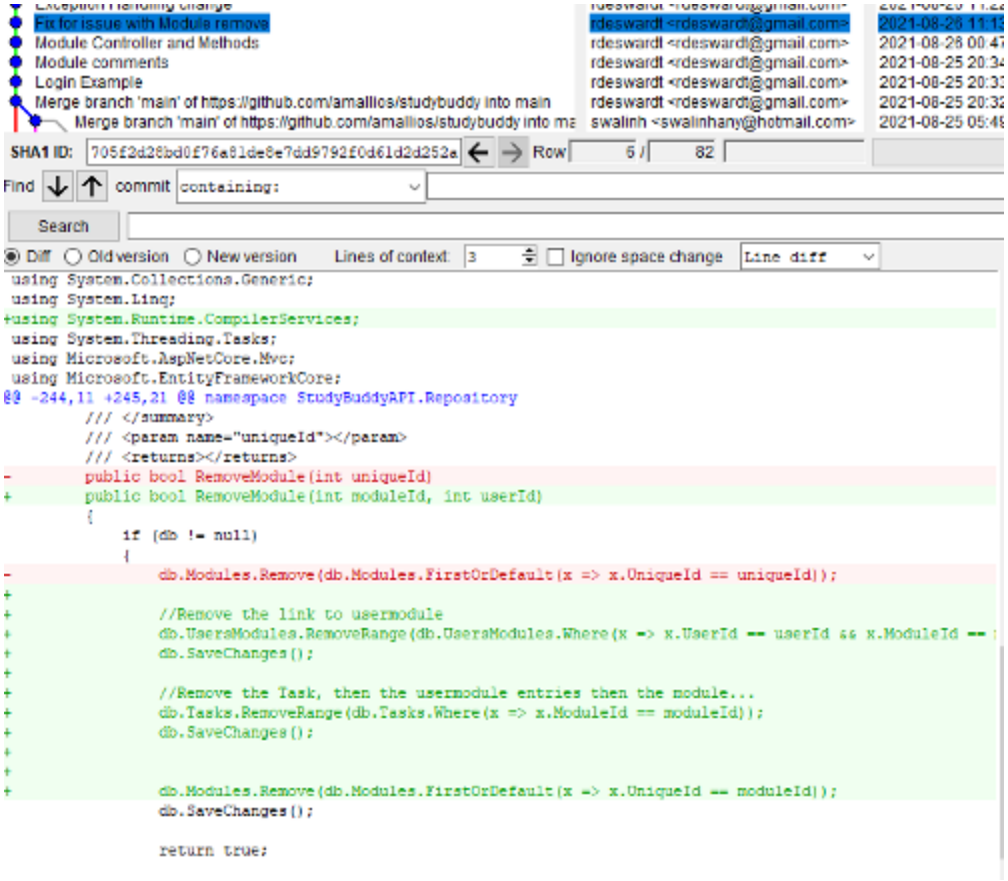


Illustrated below, you can see how two contributors have been working side by side on the project's solution performing their commits and comments on the components that have been worked on.





While testing was underway, the relevant person would perform the fix and commit the change to be re-tested if any issues were logged. Below is an illustration of this:



Testing

The System was developed through 2-week sprints, using test-driven development, to implement its main functionality.

Front-End and Back-End were developed simultaneously then integrated all while being tested separately then together.

First Sprint

In the first sprint, the basic elements to have a working website where users could register and log-in were the focus. The interface to register and log-in users, the databases corresponding to users and their roles and the required methods for the back-end services were created.

As it was decided that test-driven development was the right methodology for this application we started with **Unit testing**. As it involves the lowest-level components of the application we wanted to ensure everything was working as it was being coded and that we could build up a set of unit tests that ensure the application is working well and that no feature being added or changes made to the code or databases breaks the system. Every time a feature would be added a new test would be created and the feature would need to pass it before continuing.

Second Sprint

In the second sprint, the elements to allow users to add modules and track their important dates were the focus. The interface to add modules, the databases corresponding to modules and the required methods for the back-end services were created.

We continue with the test-driven development methodology for this application so we continue with **Unit testing** in order to test the lowest-level components of the application as we want to ensure everything is working as it's being coded and that we could continue to build up a set of unit tests that ensure the application is working well and that no feature being added or changes made to the code or databases breaks the system. Every time a feature would be added a new test would be created and the feature would need to pass it before continuing.

As with this second sprint we approached a functioning application, we needed to integrate more kinds of testing in order to ensure what is being produced meets the originally defined requirements. We designed several functional and usability tests and tested them with real users.

System Development

As per the project plan agreed at the beginning of the process the backlog was made into two sprints which were executed as per the overall Gantt Chart. We focused our efforts in the project planning phase to create a stand alone product after each Sprint and we assigned milestones to avoid splitting our focus.

This chapter of the report is split in two showing the iterative process that the team followed and it contains Front-End, Back-End and Testing chapters for each Sprint.

First Sprint (Login & Sign up Users)

The first sprint was the Log in and Sign up functionalities of Study Buddy, that is the user database, back-end services and front-end design and interactions that deal with the tasks assigned to the first Sprint.

Front-End

During our first sprint, we completed the following tasks related to the front end part of the Log in/Sign up forms:

Task	Estimated Time
Create Prototype Design of the user interface	2 hours
Setting up UI project environment	1 day
Create Design for Navigation bar	1 day
Search for Bootstrap/template that fits wanted design	1 hour
Create Design Sign in / Sign up form page	1 day
Bootstrap Integration with HTML	1 days
Finishing touches on Design and updates on Project Report	1 hour
Preparation and conduction of the UI evaluation questionnaire	1 hour
Analyzing Users feedback	1 hour

Redesigning the system according to the Users feedback	1 day

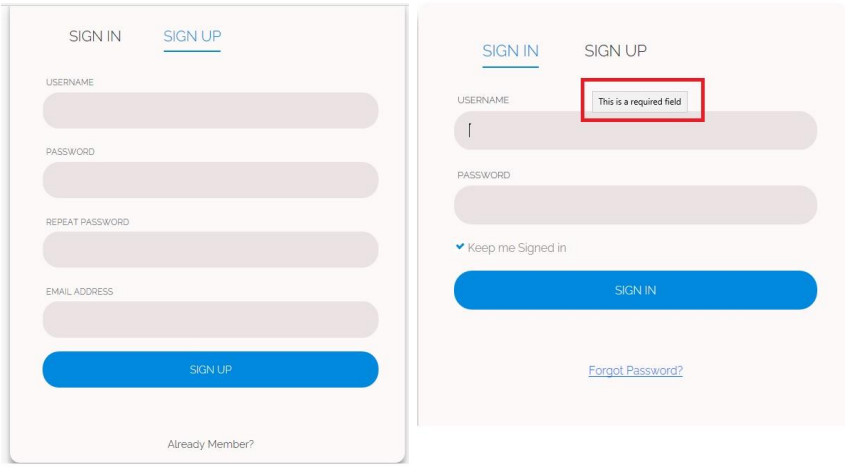
Login / Sign up Design

Our Study Buddy app is primarily designed for use on mobile devices and desktop computers, since this makes it convenient to use when studying at home or checking anticipated time to due dates.

As a result, we interviewed more than 40 users through an online survey and according to their feedback we designed the site to be both visually appealing and user-friendly. We used a Book picture in the middle of the page to indicate that this is the home page.

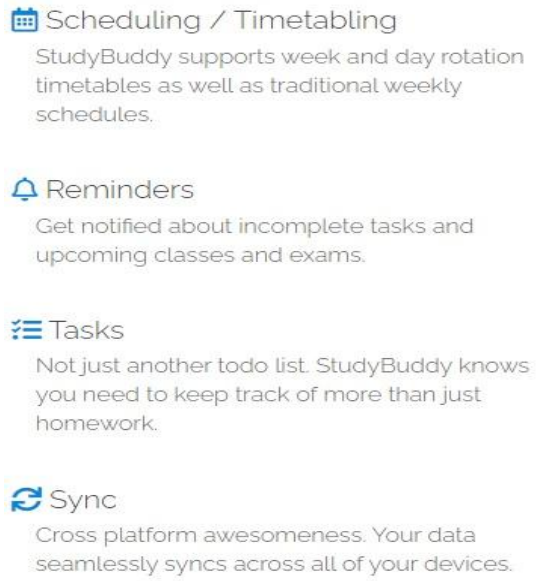
In order to sign in one must type in their email address and their password and then press the signin button which is designed with a Blue large button so that it is easy for anyone to realize and Click.

Also, for convenience use, we have combined our Sign in / Sign up forms onto one single page. By Simply hovering over the required form as shown below.



Icons Design

When designing our icons, we tried to make our icons as self-explanatory as possible. When a user scrolls around the app, they can view the icons and figure out what each function is just by looking at them. Users who scroll quickly or have difficulty reading may prefer icons, therefore we tried to make them as user-friendly as possible.



Required Attribute

Also, we developed the **required** Attribute in sign in / sign up page, required attribute is a Boolean which specifies that the input element must be filled out before submitting the Sign in / sign up Form, to make it easier for the user to see that this field is required before completing the form, as seen below.

Hovering

The hover selector is a pseudo-class that allows you to target an element that is being hovered over by the cursor or mouse pointer. When the mouse is hovering over a word, it is styled in a nice blue color to make it easier for users to see where the cursor is now pointing as shown below.

Media Design

For our design, our team used only a few images and very little animation as we did not have a tremendous amount of experience in design.

Also we also tried to keep the amount of data required to launch the app to a minimum according to user needs while utilising a cellular or network connection.
We just included simple language to let the user sign in and to indicate what the different symbols meant so that the user could be guided. We didn't require many animations because we wanted our software to be simple and straightforward.

Back-End

During the first sprint the tasks that were decided for the back-end was based on the 1st sprint for the front-end. This included the setup of the hosted environment and components required for the front-end to integrate into the back-end. Sprint 1 was also used for scaffolding required to make future changes quickly.

Please reference the back-end section for the hosting component to the project.

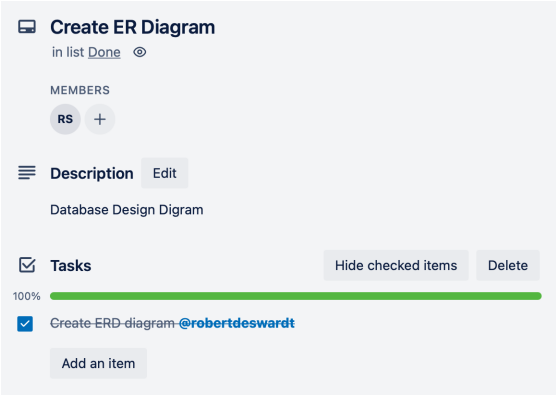
Task	Estimated Time
Create Database	2 days
Create .NET Core Rest API scaffolding	2 day
Create Database Model	4 hours
Create Repository Interface with project required methods	1 day
Create Repository for accessing Database Entities	1 day
Create API methods to be called by the front-end	1 days
Setup database on Azure	2 hour
Setup application service on Azure	2 hour
Publish Project to Azure	1 hour
Testing and Bug fixes	1 day

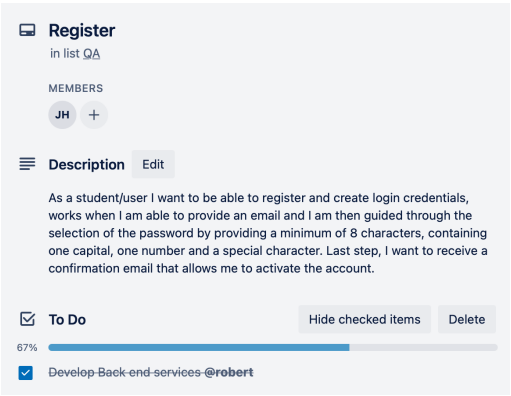
Retrospective

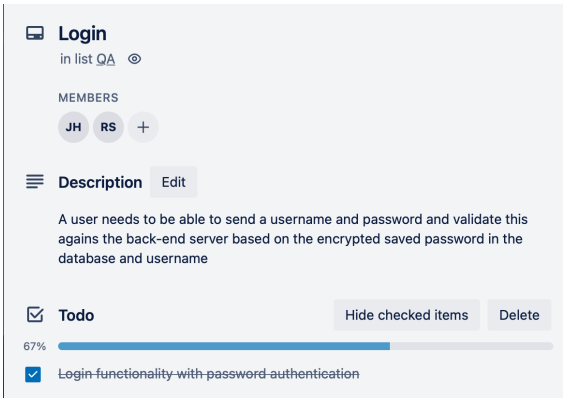
A couple of meetings and discussions were held on the database portion of the project to make sure the design is correct as this would be the driver for both the back-end and front-end services. Some small changes were made to the database but our design allows us to quickly make changes and deploy the changes.

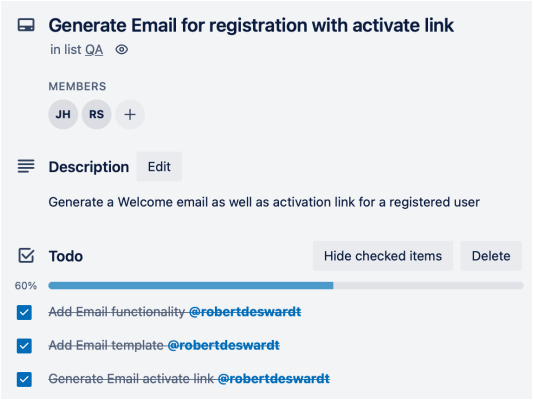
Methods In this spring included the following as per the Trello Board:

- [ ] Register
- [ ] Activate Account
- [ ] Email Activation link and Welcome
- [ ] Login









Testing

We decided to go with a Test Driven Development (TDD) approach for this project as part of the SDLC (Software Development Life Cycle) process.

The main reason why we chose to take this route for development is because this will allow us to get to a working MVP in a much shorter time frame making sure all the features have been tested and working.

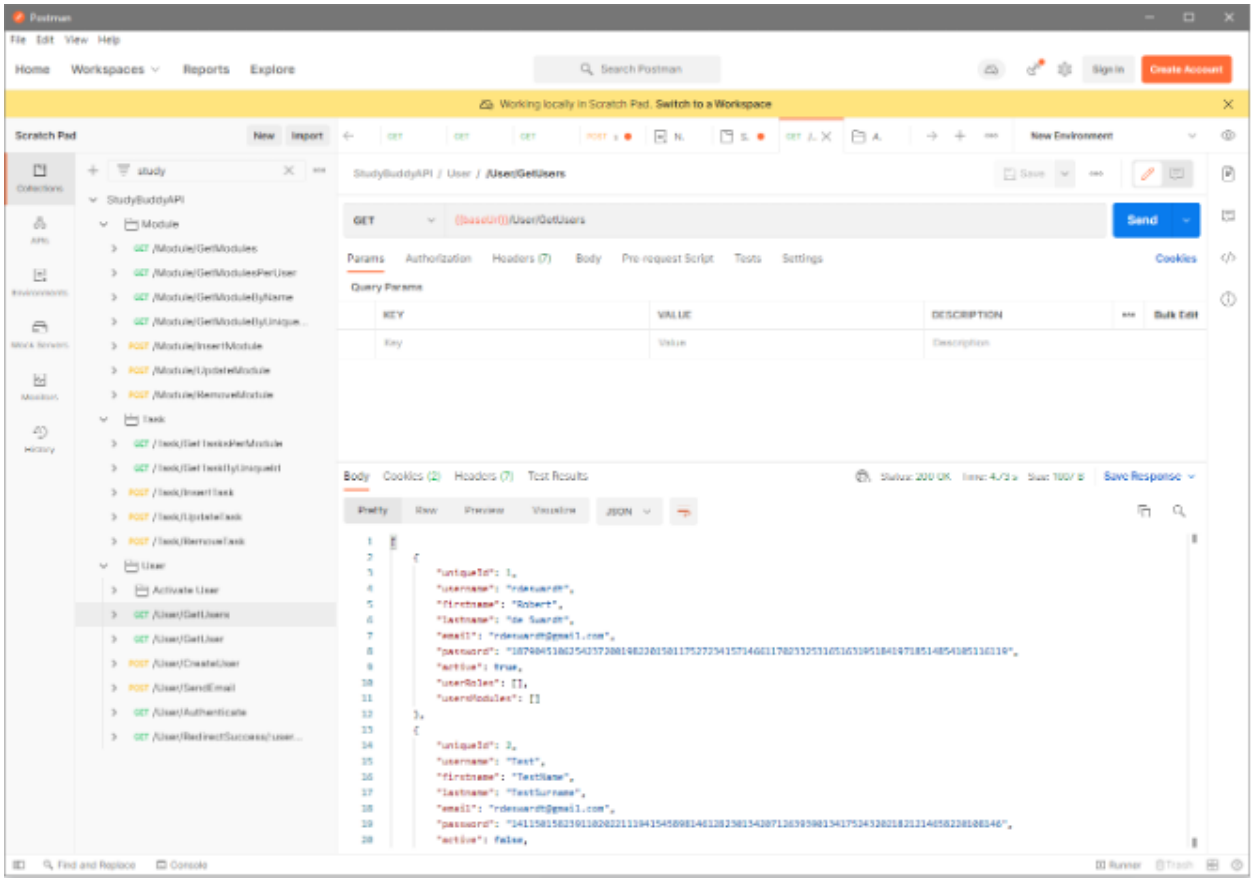
From a developer's perspective, TDD is a testing methodology or a programming discipline. A QA engineer uses this technique to begin defining and writing test cases for each small feature of an application. This method tries to answer a basic question: Is the code correct?

The major goal of this strategy is to only change or write new code when the test fails. As a result, there is less duplication of test scripts. This method is often used in agile development environments. Automated test scripts are written before functional code in a TDD methodology. The steps in the TDD approach are as follows:

- A developer creates an automated test case based on the requirements stated in the documentation.
- These tests are executed, and in some cases, they fail as they are developed before the development of an actual feature.

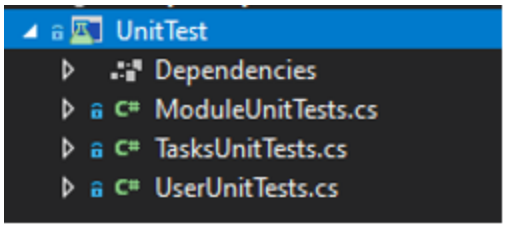
The development team then re-factors the code for the test to pass successfully.

We decided also to use another tool, Postman, to perform integration testing to the REST API service hosted in the cloud, as this can be run on Windows, Linux and iOS platforms, which allowed more members in the team to participate in the testing of the services and application.



The techniques above allowed us to perform component and functional testing on the individual components and therefore have iterative, incremental changes to our application and contribute towards the final projects. For the complete solution to be tested, we implemented a Unit test project with a test runner that tests each component on the back-end and reports if there is an issue with one of the code changes committed.

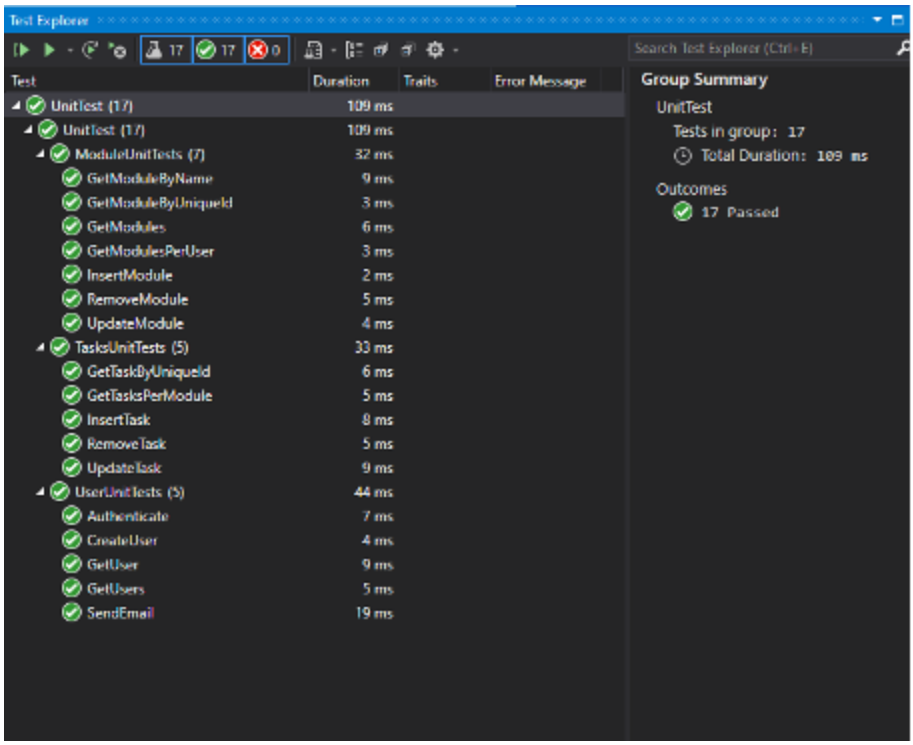
Illustrated below is the Unit test project with the results from the test performed:



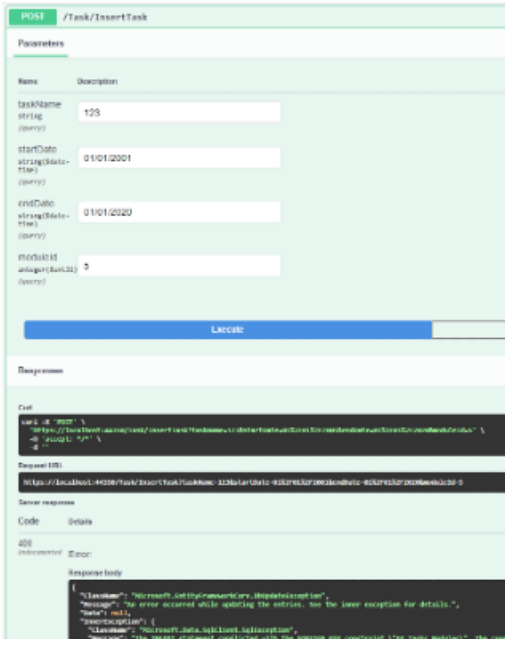
Unit test example:

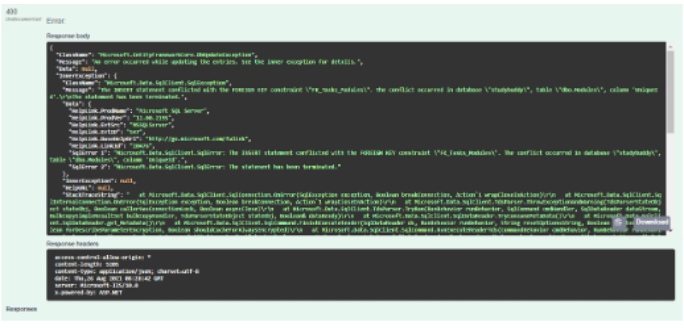
```
namespace UnitTest
{
    [TestClass]
    0 references
    public class UserUnitTests
    {
        IStudyBuddyRepository studybuddyRepository;

        private readonly ILogger<UserController> _logger;

        [TestMethod]
        0 references
        public void GetUsers()
        {
            User testUsers = new User();

            var controller = new UserController(_logger, studybuddyRepository);

            var result = controller.GetUsers() as List<User>;

            Assert.Equals(obj1: result.Count, obj2: 2);
        }
```

Results:



```
[HttpGet]
[Route(template: "GetUsers")]
1 reference
public IActionResult GetUsers()
{
    try
    {
        var users :List<User> = studybuddyRepository.GetUsers();
        if (users == null)
        {
            return NotFound();
        }

        return Ok(users);
    }
    catch (Exception)
    {
        return BadRequest();
    }

}
```





Second Sprint (Overview & Add Module & Add Task)

The second sprint was the Module Overview and Add Module & Add Task functionalities of Study Buddy, that is the module database, back-end services and front-end design and interactions that deal with the tasks assigned to the second sprint.

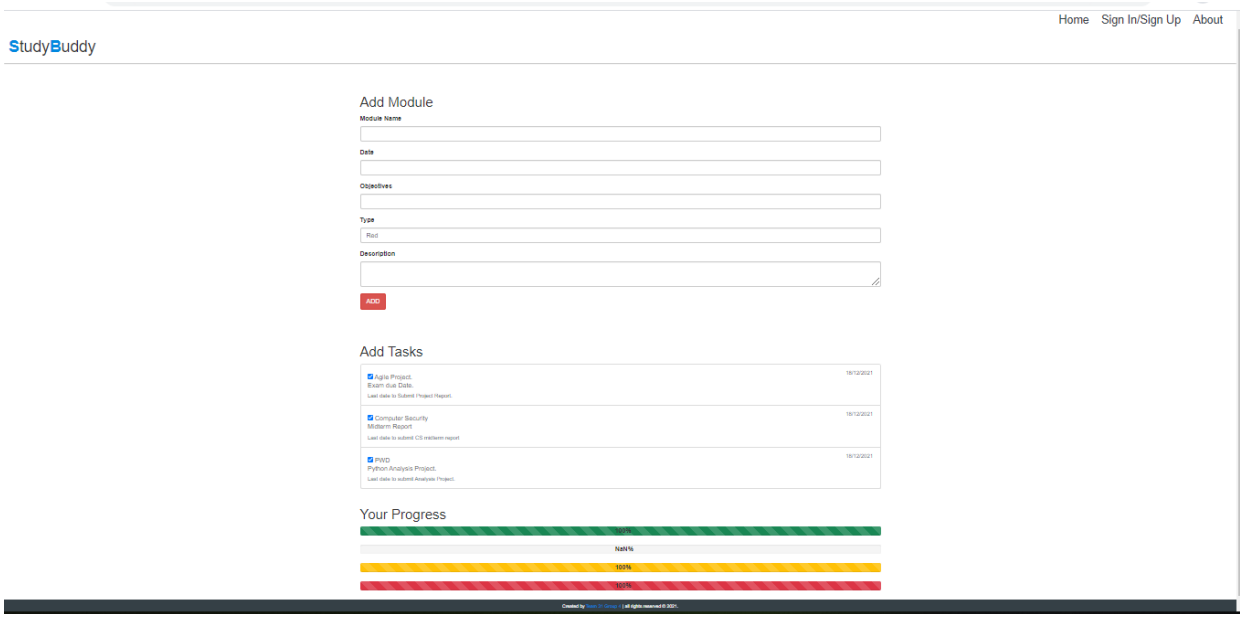
Front-End

During our Second sprint, we completed the following tasks related to the front end part of the Add Modules/Add Tasks page:

Task	Estimated Time
Create Prototype Design for add module / add task page	2 hours
Create Design for Add Module form page	1 day
Create Design for Add Tasks form page	1 day
Create Design for Task progress bar	1 day
Bootstrap Integration with HTML	1 days
Finishing touches on Design and updates on Project Report	1 hour
Preparation and conduction of the UI (add Module) evaluation questionnaire	2 hour
Analysing Users feedback	2 hour
Redesigning the system according to the Users feedback	1 day

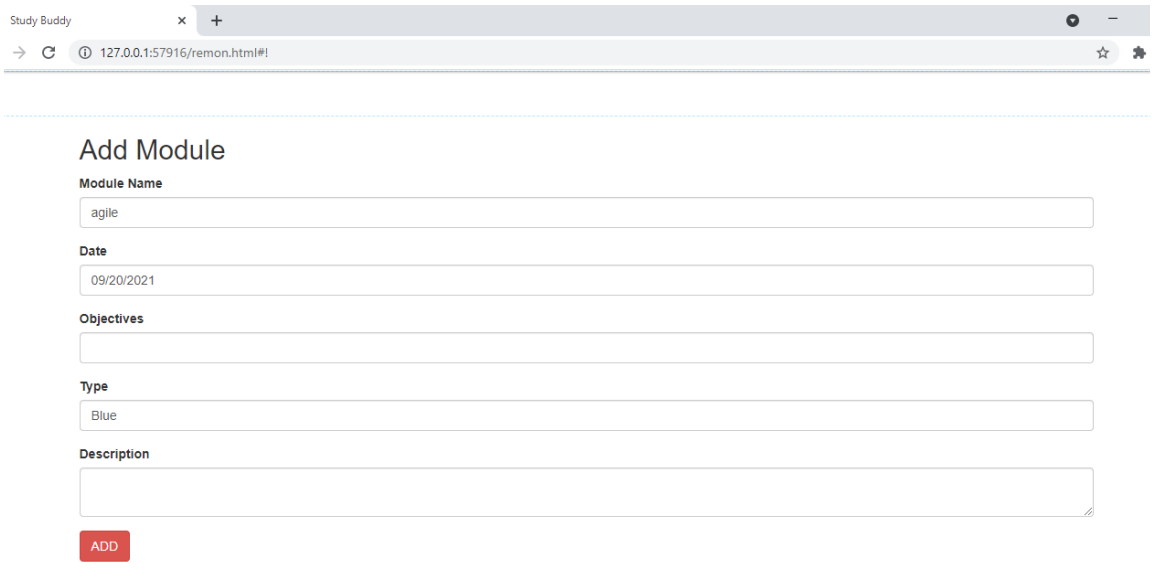
Add Module / Add Tasks Design

After signing in from the main page in your application, it will be redirected to add Module page automatically.



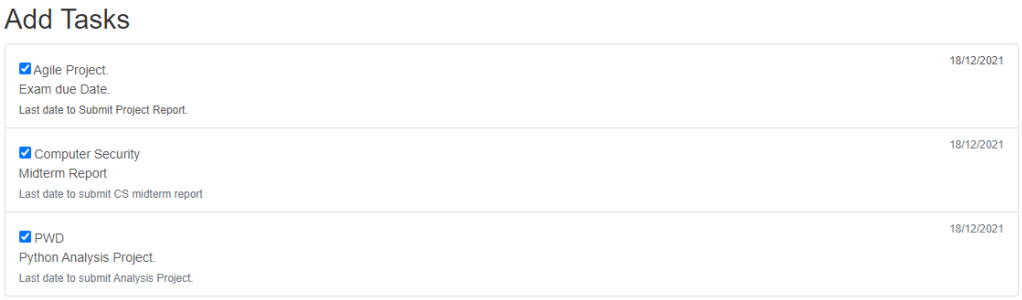
Add Module

Add module view has a form to fill out to add a new module to your Studybuddy website that will remind you of your due dates and deadlines for any assignments and examinations. Each user has the ability to name a new module and specify its goals (exam deadline, Quiz time, assignment due date, etc.). And to add the module name to his reminder list, a user must fill up the following text boxes and press the red button to show its importance .



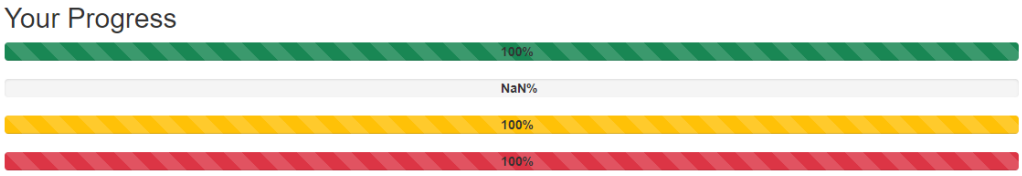
Add Tasks

Our Add Module page's second view will be an Add Task Section, which will be shown as a list of additional tasks linked to each module, each with a distinct colour.
As seen below, the Add Tasks section will display the Module name, Task name, relevant comments, and due date.



Progress bar

While the progress bar will show you how far you've gotten on doing your pending chores, each job will be distinguished by a distinct colour, as illustrated below.
Exams --- > red colour denotes the importance of this task.
Assignment with tiny Weight --- > Green colour stands growth.
Quizzes ---- > the yellow colour should be visible to the user's sight, In order to avoid missing any deadlines.
Incomplete peer review ---- > Blue colour stands for Security, Stability.



Back-End

The second spring required us to complete all the outstanding back-end tasks:

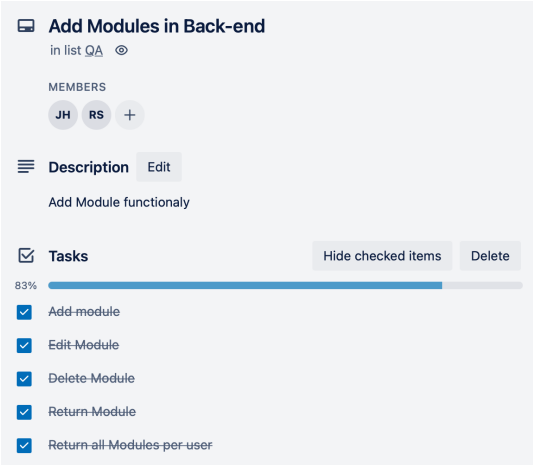
- [ ] Add module functionality
- [ ] Add Task functionally

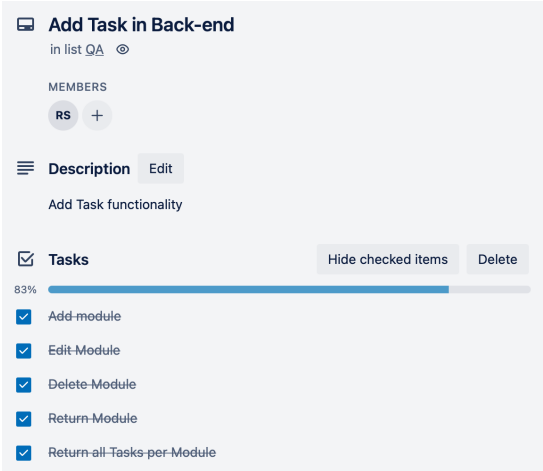
Task	Estimated Time

Create Module section in repository	1 day
Create Task section in repository	1 day
Update interface definitions to include new methods for the Module and Task functionality	1 day
Test CRUD methods for all modules added	2 days
Bug fixes and integration testing	2 days
Method changes based on scope discussions	2 days
Project deployment to cloud - includes iterations	2 days

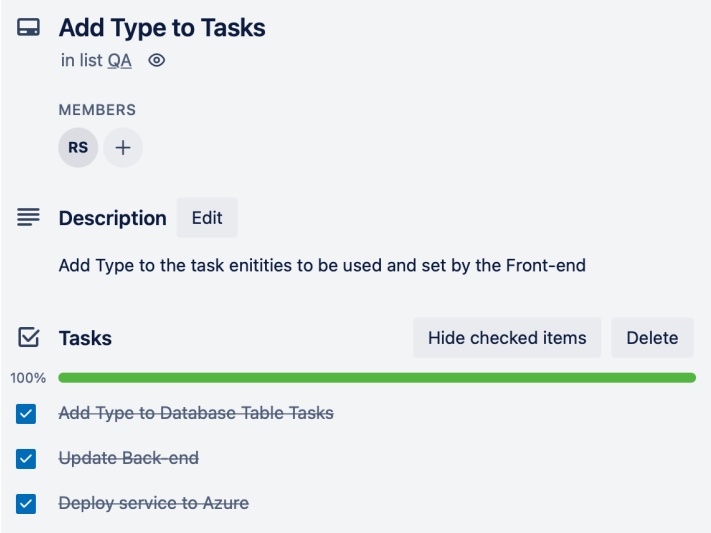
Retrospective

Example mock-up pages were created to assist in calling the services, showing how to perform post requests to the rest API service. Sprint 2 tasks where as follow:





During the testing phase we realised that we require a type for the task items to show the importance of the task for the progress bar indicator.



Testing

As we continued development and entered the second sprint we continued using the same methodologies and systems.

In addition to the unit tests we created several different test procedures to better understand the performance of our system.

Functional Test Procedures

We needed a test procedure that would allow us to see how the users interact with our system. For this we designed the following functional test to ensure that the requirements we had previously defined were being met and that users would not find our system difficult to use.

Test Procedure ID	Objective and Priority	Estimated Duration
1	User can sign up	10 minutes
Startup	Open browser, go to: studybuddy.com/sign	
Relationship to other procedures	Needed in order to be registered on the website and add modules in order to track them	

Procedure				
Step and Test Case	Activities	Examination of result	Actual Result	Test Result
1	Click on the sign up tab	The user should be prompted to fill a form with the necessary data to create an account		
2	Click on the sign up button	The user should receive a message that their account has been successfully created and that a confirmation email has been sent for activation.		
3	Click on the activation link received via email	The user should be brought to their homepage where they can add modules and see their progress		
Stop.				

Test Procedure ID	Objective and Priority	Estimated Duration
2	User can sign in	10 minutes
Startup	Open browser, go to: studybuddy.com/sign	
Relationship to other procedures	Needed in order to be log onto the website and add modules in order to track the	

Procedure				
Step and Test Case	Activities	Examination of result	Actual Result	Test Result
1	Click on the sign in tab	The user should be prompted to fill a form with an email as username and a password		
2	Click on the sign in button	The user should receive a message that their account has been successfully validated and be brought to their modules page		
Stop.				

Test Procedure ID	Objective and Priority			Estimated Duration
3	User can add modules and update progress			10 minutes
Startup	Open browser, go to: studybuddy.com/sign and sign in			
Relationship to other procedures	Only accessible if the user has an account on the website			
Procedure				
Step and Test	Activities	Examination of	Actual Result	Test Result

Case		result		
1	Add module	On their module homepage the user should be able to fill a form with a module's relevant data and add it		
2	Update module	Works when for each module created the user has access to an editing functionality that allows to add event based information along with details about notification rules.		
3	Update module	Works when either form the main page or from the module details page I can change the progress through a fader		
Stop.				

User Test Procedures

Our design philosophy is aligned with making a product that is easy to understand, easy to use and that users feel like it useful. We used the **UMUX** questionnaire to get feedback from users and to ensure the product was usable and accessible.

	Strongly Disagree	1	2	3	4	5	6	7	Strongly agree
This system's capabilities meet my requirements									
Using this systems is a frustrating experience									
This system is easy to use									
I have to spend too much time correction things with this system									

HTML Testing

Finally we tested the correctness of our HTML using a couple different validators. The feedback was brought up to the Front-End developer and adjustments were made

Nu Html Checker

This tool is an ongoing experiment in better HTML checking, and its behavior remains subject to change

Showing results for uploaded file index.html

Checker Input

Show ☐ source ☐ outline ☐ image report Options…

Check by file upload Choose file No file chosen

Uploaded files with .xhtml or .xht extensions are parsed using the XML parser.

Check

Use the Message Filtering button below to hide/show particular messages, and to see total counts of errors and warnings.

Message Filtering

1. **Warning** **The `type` attribute is unnecessary for JavaScript resources.**

   From line 27, column 3; to line 27, column 255

   ```
   ss" />↩↩  <script crossorigin="anonymous" defer="defer" integrity="sha512-NefnY8O7CG7NvgTQOUD9T6bwbNxg9zNYTFW1… type="application/javascript"
   src="https://github.githubassets.com/assets/environment-35e7e763.js"></scri
   ```

2. **Warning** **The `type` attribute is unnecessary for JavaScript resources.**

   From line 28, column 5; to line 28, column 262

   ```
   ript>↩  <script crossorigin="anonymous" defer="defer" integrity="sha512-YduJ/b0BatW4HfH1xupjk1ZeDCB0tPThg4cK…="application/javascript"
   src="https://github.githubassets.com/assets/chunk-frameworks-61db89fd.js"></scri
   ```

{JSON formatter} JSON PARSER XML FORMATTER JSBEAUTIFIER SAVE RECENT LINKS LOGIN

HTML Validator

Sample

```
<!DOCTYPE html>
<html lang="en" data-color-mode="auto" data-light-theme="light" data-dark-theme="dark">
<head>
<meta charset="utf-8">
<link rel="dns-prefetch" href="https://github.githubassets.com">
<link rel="dns-prefetch" href="https://avatars.githubusercontent.com">
<link rel="dns-prefetch" href="https://github-cloud.s3.amazonaws.com">
<link rel="dns-prefetch" href="https://user-images.githubusercontent.com/">
<link rel="preconnect" href="https://github.githubassets.com" crossorigin>
<link rel="preconnect" href="https://avatars.githubusercontent.com">
<link crossorigin="anonymous" media="all" integrity="sha512-y7qmntuinsKc9vzjz7YzWWh8GcPQmvwoF4ULzjX7GgRSvik3SnIR2GIylHwLEF/3nya8RYtrrif1YX5IliJ1wA==" rel="stylesheet" href="https://github.githubassets.com/assets/frameworks-cbbaa69edba29ec29cf6fce3cfb63359.css" />
<link crossorigin="anonymous" media="all" integrity="sha512-5495K7TraDdD0rUjH7go28D4vVQZ5ieXuCpiS3jv75MnUggsnjiA3JsOO5nbFy/PpjvO5yYu7KNfr4weDVEkMQ==" rel="stylesheet" href="https://github.githubassets.com/assets/behaviors-e78f792bb4eb683743d2b5231fb828db.css" />
<link crossorigin="anonymous" media="all" integrity="sha512-sdeTh+U8loxnLuWlh5hUYyRxXIsNrMtJ8ggDYod/VP4vb3vqPN2W+poSiOJQMV6VEoVeHQgxxtT8kg4kIYbTxA==" rel="stylesheet" href="https://github.githubassets.com/assets/github-b1d79387e53c968c672ee5a587985463.css" />
<script crossorigin="anonymous" defer="defer" integrity="sha512-NefnY8O7CG7NvgTQOUD9T6bwbNxg9zNYTFWlTkin6RHuUTZvv7S4Vr91TAtVhBYcyfFsUz6ptnpMxRPtAT48tw==" type="application/javascript" src="https://github.githubassets.com/assets/environment-35e7e763.js"></script>
```

Load Data

Validate HTML

Format HTML

RUN

Download

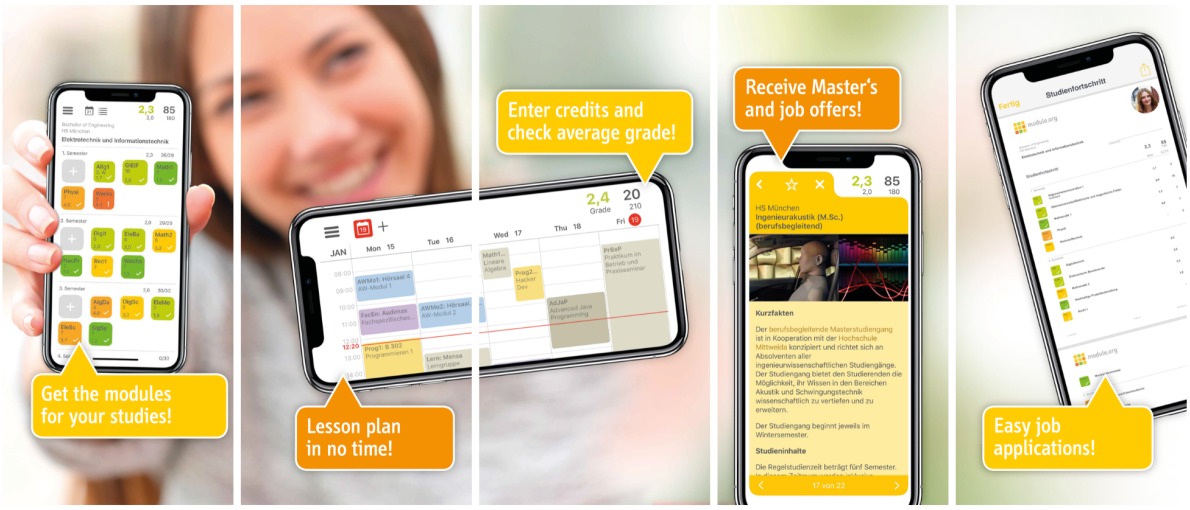
Output

```
Error on->  Line :1334 Column :75
Message :attributes construct error

Error on->  Line :1334 Column :75
Message :Couldn't find end of Start Tag span line 1334

Error on->  Line :1334 Column :93
Message :Opening and ending tag mismatch: td line 1334 and span

Error on->  Line :1334 Column :105
Message :AttValue: " or ' expected

Error on->  Line :1334 Column :105
Message :attributes construct error

Error on->  Line :1334 Column :105
Message :Couldn't find end of Start Tag span line 1334

Error on->  Line :1334 Column :123
Message :Opening and ending tag mismatch: tr line 1332 and span

Error on->  Line :1334 Column :136
Message :AttValue: " or ' expected

Error on->  Line :1334 Column :136
Message :attributes construct error

Error on->  Line :1334 Column :136
Message :Couldn't find end of Start Tag span line 1334

Error on->  Line :1334 Column :152
Message :Opening and ending tag mismatch: html line 3 and span

Error on->  Line :1334 Column :152
Message :Extra content at the end of the document
```

Analysis

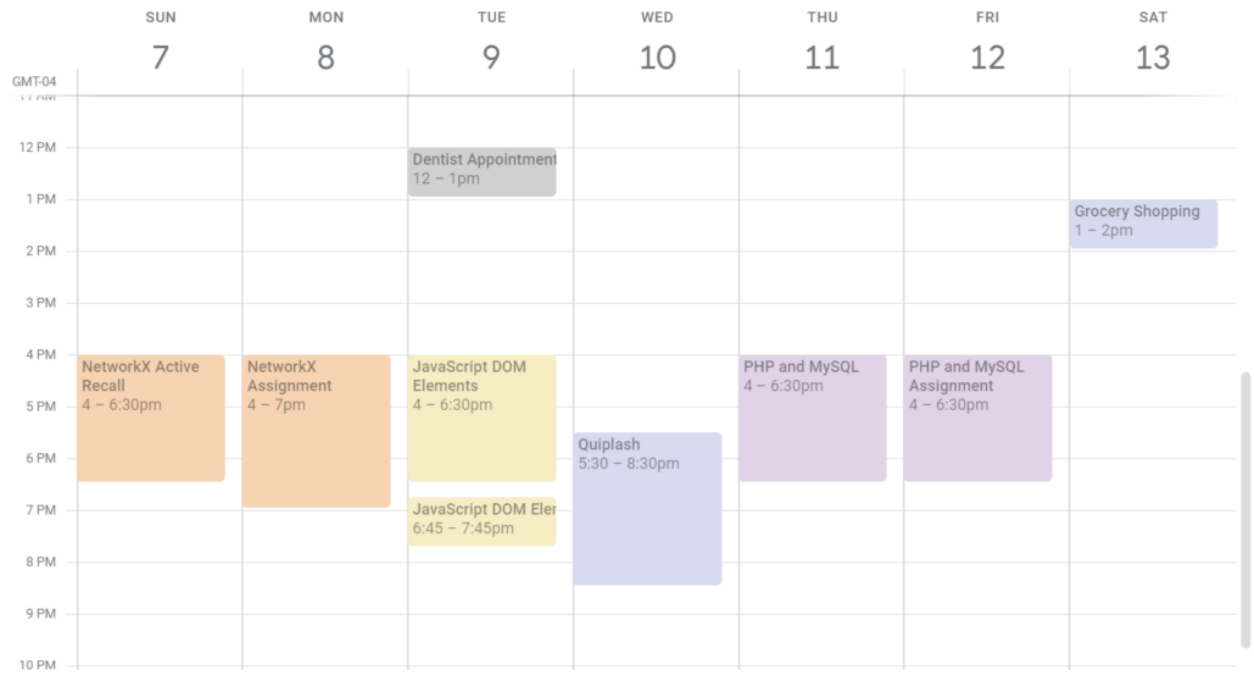
The aim of the application being prototyped is to improve the ability of students to track their progress and has been specifically designed for students of all ages to gain control over their studies in a more efficient way than the traditional general purpose time management and productivity software.

In the following example modules.org allows to create modules and create a calendar for the lessons

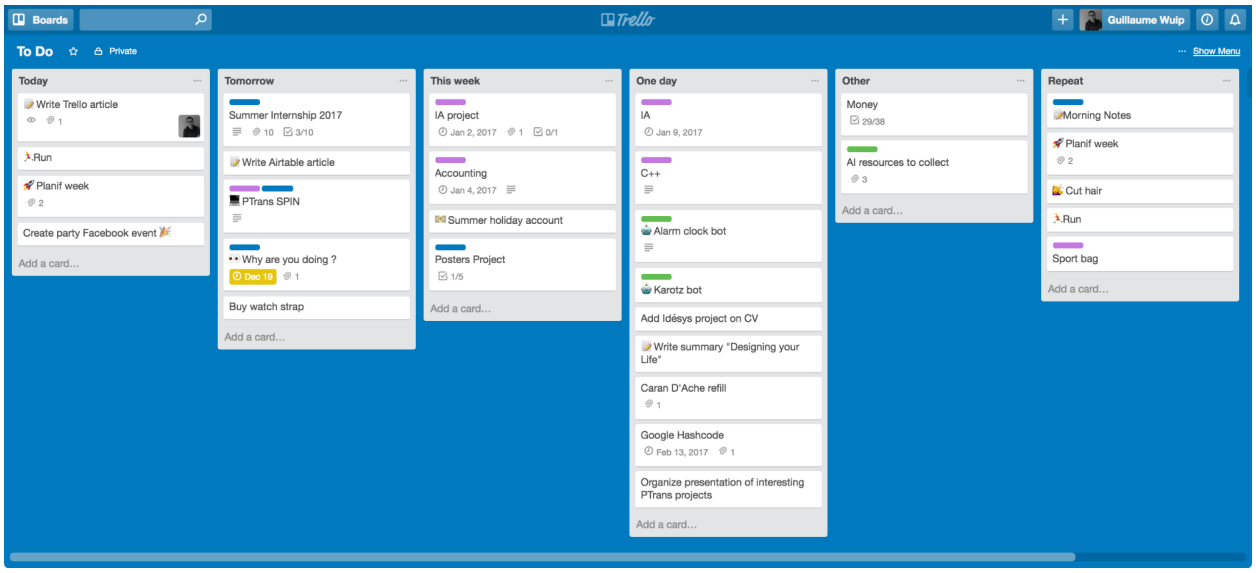


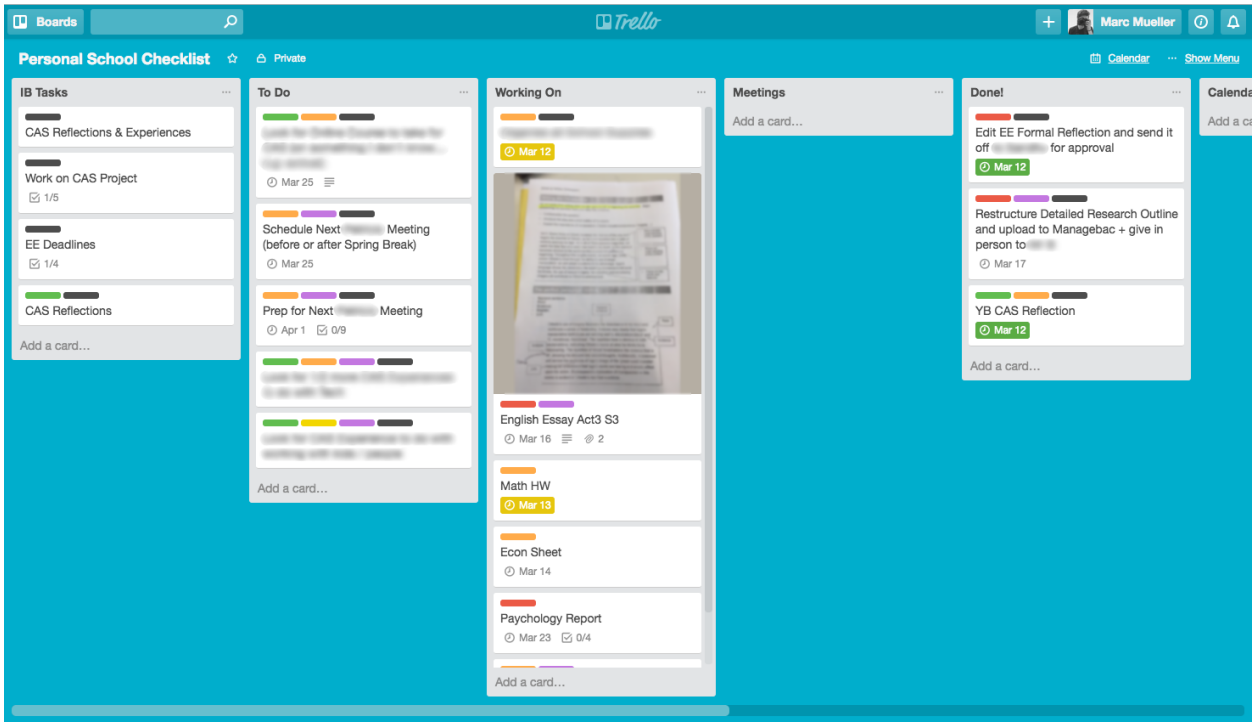
Such functionalities can be also achieved as a combination of multiple production tools, for instance I could set reminders and calendarize events using google calendar and encode the activities as tasks in productivity tool such as trello or similar

The following is a typical view of google calendar :



while Trello could be used for tasks and status





Studybuddy will focus more on the ability to track the progress and make it really easy to stay on top of deadlines and upcoming assignments.
Tools like google calendar and Trello as much as other Microsoft productivity tools might still be of help for people so Study Buddy should provide account sync capabilities (not as part of the MVP being prototyped)

Risk Assessment

No.	Risk	Danger 1-5	Effects	To Prevent	Solutions	Responsible Person
1	Poorly defined project description	3	The team does not entirely understand the project and may spend too much time making important decisions	N/A	Organize meetings with Product Owners as often as it is needed to clear things up	Scrum Master
2	Troubles connecting and integrating between Front-end and back-End development	5	A big part of the ‘A’ requirements cannot be finished	Start early, make it a top priority	Focus all manpower on establishing the integration	Product Owner
3	Workload in other courses	3	Less focus on the project, programmer may be out for a long time	Stick to the plan, organize your homework according to the work schedule	Focus all your energy into finishing what’s due to clear the stress, then come back	Product Owner
4	Incompletion of tasks by the end of the sprint	3	Falling behind the schedule and delaying the work	Try to assess time of tasks as well as possible	Learn from it and use retrospectives to make later sprints more efficient	Product Owner

5	Inefficient estimation of time assigned for the project	3	Incomplete product	Update the backlog and follow the Scrum plan	Re-estimate the requirements and time if needed	Product Owner
6	Product Owners do not have time for the group	3	Crucial information may come late	Try to get regular meetings with them	Call or email project owners if needed and hope for a quick answer	UX designer
7	Lack of communicatio n	2	More chance of misunderstan ding	Try to make it to all meetings	Write all information down and send it to those who couldn't make it	Scrum Master
8	Illness	2	Programmer may not be able to leave home	Try to get enough healthy food and good sleep	Work from home or work up the hours during free time	Scrum Master
9	Inefficient estimation of time assigned for a sprint	2	Failed sprints	Try to give yourselves more time for each task	Hold retrospectives to learn from it and re-evaluate time, maybe spend more time on the project	Scrum Master

10	Over-estimation of skills	2	Implementation of some requirements may take longer time or be done wrong	Gather as much information about the task as possible before diving into programming	Ask someone who may be able to help and explore the material better	Scrum Master
11	Absences	2	Programmer away for some time	Try to schedule other activities according to the work schedule	Work up the hours, e.g. use weekends, respect the working agreement	Scrum Master

Risk occurrences

No.	What happened	Effects	Solution
1	The project's scope was unclear.	Scope Creep effect as the team was unsure about the project's scope.	A few meetings have been arranged to go through the project in further depth and to clarify things up.
2	Team Member was sick	Delayed Tasks consequently delayed schedule	Team Spliced his tasks on other 3 members
3	Lack of communication from the Product Owner	The team got wrong information from Product Owner	The team questioned the Product Owner if everything was clear at the following meeting.
4	Lack of communication between the Product Owners and the team	The meeting was delayed	The team used the time to work on the report while waiting for the Product Owner to arrive

5	Workload in other courses	The team conducted the prototype questionnaire with users, however due to an exam in another course, we were unable to continue working on the project. Several tasks that were supposed to be completed in Sprint 1 were not completed.	The team needed to keep working on the sprint's outstanding assignments.
6	Workload in other courses. There were a lot of significant deadlines.	The whole team decided to focus on finishing other projects	The team did not assign too many tasks to the sprint, so they would definitely manage to finish it
7	Exam session	The team had to put the work on hold while studying for examinations.	The team meticulously scheduled the remaining sprints and calculated the time for the remaining tasks before putting work on hold.
8	Integration phase took way longer time than was expected	Unfinished tasks by the end of the sprint	The team spent its time studying and resolving the problem, deferring less critical duties.

Evaluation

The project was split into 5 domains (Process, Product, Front-End, Back-End and Testing) with the five authors having the accountability for each. We considered it appropriate to evaluate the project from each domain's point of view.

Process

The team spent a significant amount of time considering the process in depth. Since we all happen to be experienced professionals we are used to taking shortcuts in our professional life depending on the size of the task in hand.

During this project we all took a step back and discussed thoroughly how the different Agile strategies fit into each other. We discovered the limitations of each and discussed in length the differences between the theory and practice, exchanging ideas and stories from our respective workplaces.

A Project Manager's role is exactly to facilitate the process, make sure that there are opportunities for the team members to discuss and try to focus people's attention to certain tasks. Not always the most charming role, but having worn this hat we will certainly be able to work better with project managers in the future.

Last but not least, we did have fun. Along with getting to know and appreciate each other's reaction in tense moments, when a milestone was slipping or when we were caught by surprise when underestimating the complexity of certain tasks. All team members were quick to jump in and save the day.

On reflection, from a Project Manager's point of view, I did fail to foresee certain circumstances that could have made the project run smoother. Having known the extent of our time limitations in the summertime I should have built in much more buffer than what I did.

Product

End User Product Evaluation

In order to be easily accessible by the designated end users the application should be developed with a mobile first approach and using the material UI designs as much as possible.

Material UI has been selected as it has gained a lot of traction in the last few years and is being supported by multiple UI frameworks and this should avoid many usability issues as well as speed up the UI implementation.

Users might connect from several time zones, it is fundamental to receive notification in the relevant time zone for each specific user. The time zone should be set automatically according to the device being used to make the connection and/or geo-referencing the IP address from which the connection is being established.

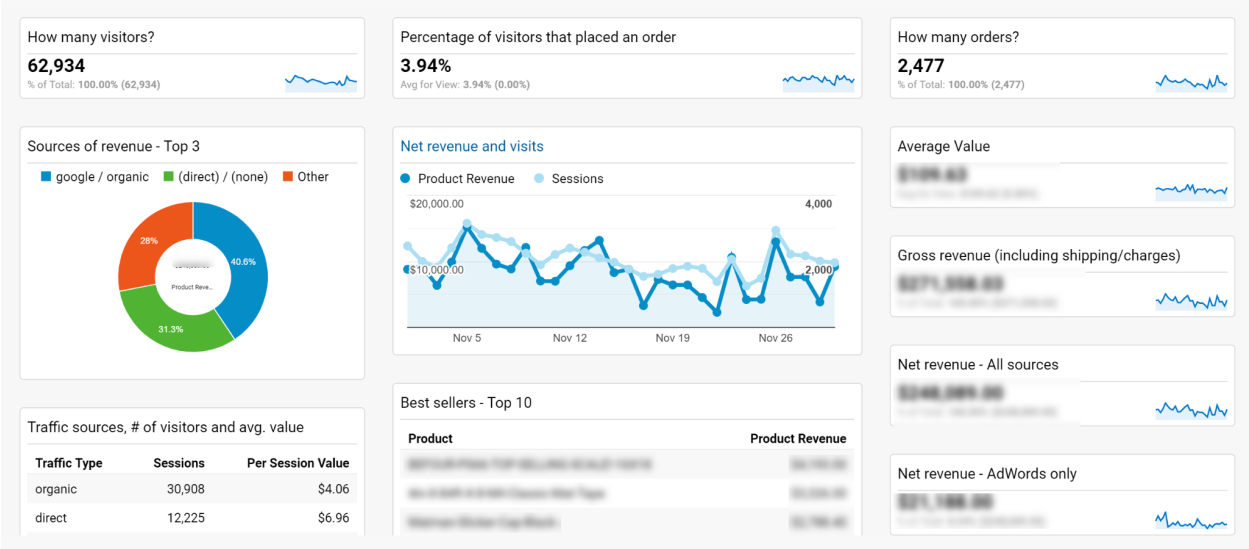
Losing information could be really frustrating therefore the Study buddy application should allow the user to back up the information and store the backup in one of the user's favorite cloud services or locally on the user's machine.

As mentioned in the previous section, integration with 3rd parties application even though not being part of the MVP developments is a critical part of the application since many user expect to be able to export information and make them available to their favorite apps
Implementing an open architecture and providing the ability to sync accounts with most used apps and networks should be part of the evolution plans.

Internal User Product evaluation

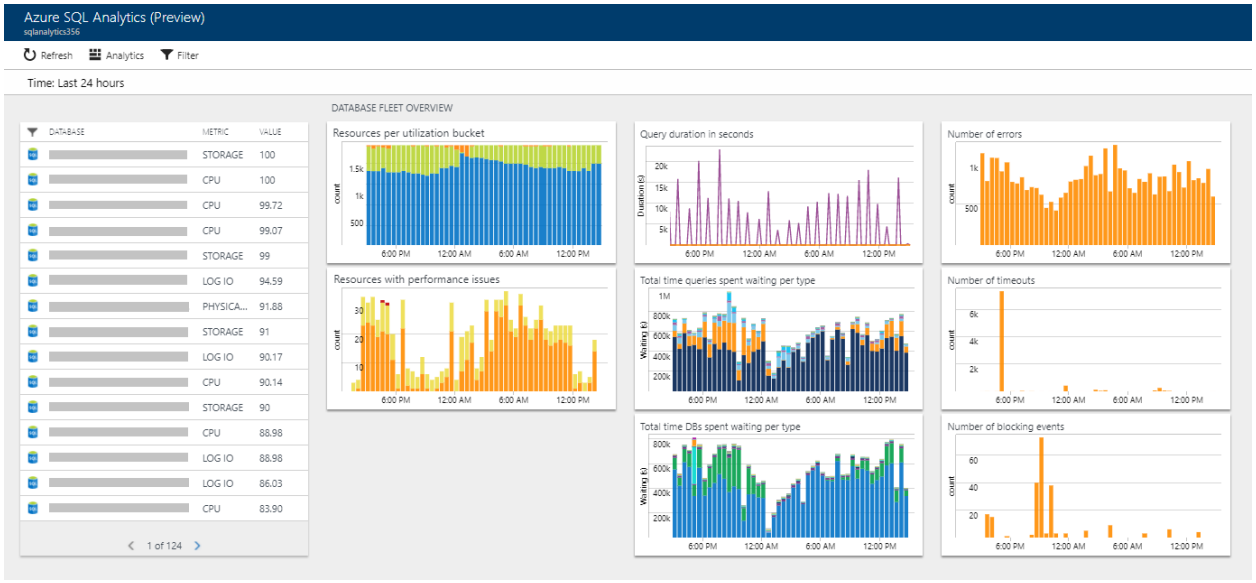
In order to understand how the application is being used overtime the application should be geared with tools for the analysis of traffic, being a web based application the best tool is probably google analytics

Below a screenshot:



From a system perspective the application should be as available as possible to the end users even though it is in MVP therefore should be geared with a minimum set of monitoring services.

Azure, the cloud platform selected to build the prototype provides such services for the different tiers of the stack as shown below:



Front-End

Evaluating Design

Principle 1: Allow easy access to application

Our application relies on widespread easy use and we have to market it that way. If a user has initial trouble getting on the system or trying to make an account they may disregard using it.

Making a paper studying progress list is a pain for many users, but they would rather take a few minutes to make a list than to spend a whole afternoon troubleshooting their new study buddy list app.

Logging in should be something that is very easy to do and not require a lot of cognitive or physical thought. Our log-in Button is located in our Navigation bar in big Black text and it turns into Blue when hovering on the word to make it clear for the user that he will use this feature now.

All it takes to log in is an e-mail address and a password. The log-in was designed to just require an email address and password to log in so that it is easier for the user and allows them to have their own account.

Principle 2: Design for Security

Security is extremely important. Many users can be careless with passwords and not change their passwords very often or use the same password for multiple applications. We have to be prepared for that issue as well as allowing for a user to not have issues accessing the app by creating some obscure password that they won't be able to remember. Being able to log off of the system on the device allows for some sense of security that the lists won't be compromised if someone gets a hold of the user's device.

Principle 3: Design for ease

One of the most significant features of our software is that it allows users to add as many Modules and tasks as they like on a regular basis. Also our software allows users to get to their task list whenever they want from their mobile device and add them in an orderly manner (coloring schema).

Principle 5: Visual Aesthetics

It is critical to have something that is appealing to the eye in order to sell it. One of the reasons for Apple's success has been the aesthetically pleasing nature of some of their software. People appreciate things that look beautiful, and our app is full of them, with a book image on the front and bright graphics throughout. This maintains the user's attention on the app and increases their enjoyment of it. Also we utilized the friendly font text Arial.

Principle 6: Support Diversity of Users

We made every effort to keep our design as straightforward as possible. We could only examine so many variables, so we took gender, age, and a range of user ability levels into account. We believe our design is simple enough for anyone with even a basic grasp of computers to understand.

Back-End

We decided to create a back-end that is not dependent on the technology we will use for the front end. This is entirely server-side, while the front end will be client-side.

The back-end needed to be de-coupled from the front-end, making it possible to develop a front-end, either a website or a mobile application or even a windows application.

For future development, having the back-end de-coupled allows us to provide multiple types of User interface applications for various users who prefer a native application running a mobile device.

Please refer to the back-end section on the reasons for choosing the technology stack.

The biggest problem we had within the team was the experience in back-end technologies, so we had to rely on a single person to develop the back-end and assist with the front-end development.

This was a significant risk as numerous things could happen to the sole person developing the back end. The way this risk was mitigated

- We had incremental check-ins
- Reviews by the team were done frequently
- Training the team on this new technology. .NET Core.
- Individual component testing

We as a team learnt a lot about the options available to us for developing this back end. A lot of research was done, and watching tutorials on how to develop within the .NET Core platform, which is platform agnostic. This was also the first time the team deployed to the Azure cloud, as this was the first deployment done to Azure Hosted.

We have had a lot of learnings, and the next time we will make sure we have more members in the team that can code within the platform itself that would allow us to share the workload better.

Testing

As it was decided early on, we developed StudyBuddy using test-driven development so for the technical aspects the way we would test would be through unit tests. In retrospect, it's easy to see why TDD is such a popular system to create software.

The test set allows one to gain a lot of insight whenever there is a problem or when the app is not working in the expected manner. As the tests are created along the software we ensure that every basic function is performing its task as expected and when we add any new code that this code will be compatible with the rest of the app. In short, TDD worked as expected, allowing us to build a very complete test set that allowed us to code in a very secure and concise manner.

As for the Functional testing we designed 3 procedures that would be testing the main functionality of the app; which means allowing users to sign up to the service and to create their dedicated studies tracker by adding modules with dates to track. Additionally we used a UMUX test to gather more general feedback from users. The results we obtained from users were of great help to design an interface that was accessible and friendly to as many students as possible and specially for our target demographic. They were useful as they allowed us to see how users interact with the system and what parts work well. From these tests we would revise the results as a group and then add to our Kanban the aspects that might need tweaking or changing based on the feedback and assign them to the respective developper.

Conclusion / Summary

Outcomes

The MVP can be used to test the initial idea and could be distributed among a community of students willing to test and provide feedback about the different functional and usability aspects.

Development Practice

The Kanban was selected as the main instrument of collaboration and Trello as the technical tool to configure the activities and track the status of the activities as well as encode the details of such activities.
The team was able to perform two full scrum sprints and deliver a relevant number of functionalities including a number of change requests that have been added along the way while the application was being developed.
The team is composed of experienced individuals so the scope was tailored in order to be feasible within the available time frame and functionalities developed according to the initial plan.
We did not make the prescribed card estimation game but we preferred to leave the choice to each individual on which task to pick and we preferred to maximize the development capabilities in terms of development hours being executed by each member.
Since the resources were fixed in terms of team members and amount of hours to be made available by each we should have achieved the maximum amount of deliverables feasible with the given resources.
Many challenges were faced as a distributed development team, the first was the different time zones that made synchronous communication (such as MS Teams or Google Meet) available only for a short fraction of the day therefore we had to rely heavily on the capabilities provided by asynchronous communication tools such as chat (Slack), Kanban Task management (Trello), Code Versioning (Git) Document Management (Google Drive and Google Docs/Slides)

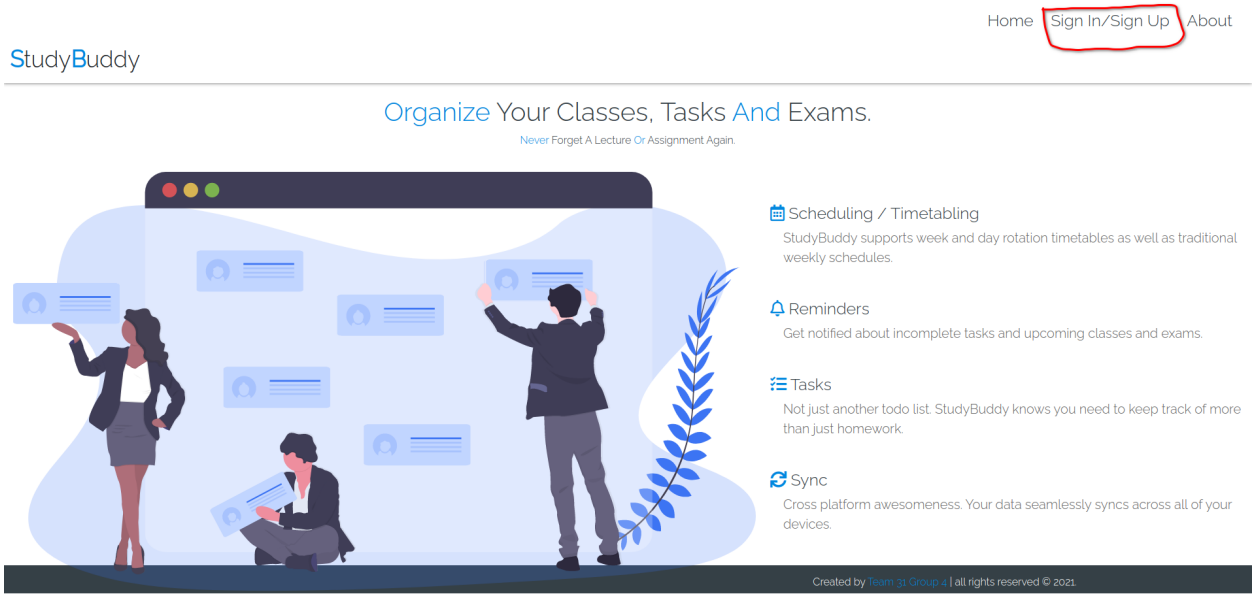
Future Plans

The application has plenty of room for improvement in functional terms but it has been developed in a way that the initial set of features are solid enough to be built upon and not require a full rewrite in order to provide the desired evolutions.
The main area for evolution are related to integration with 3rd parties therefore making the app as part of a larger ecosystem.

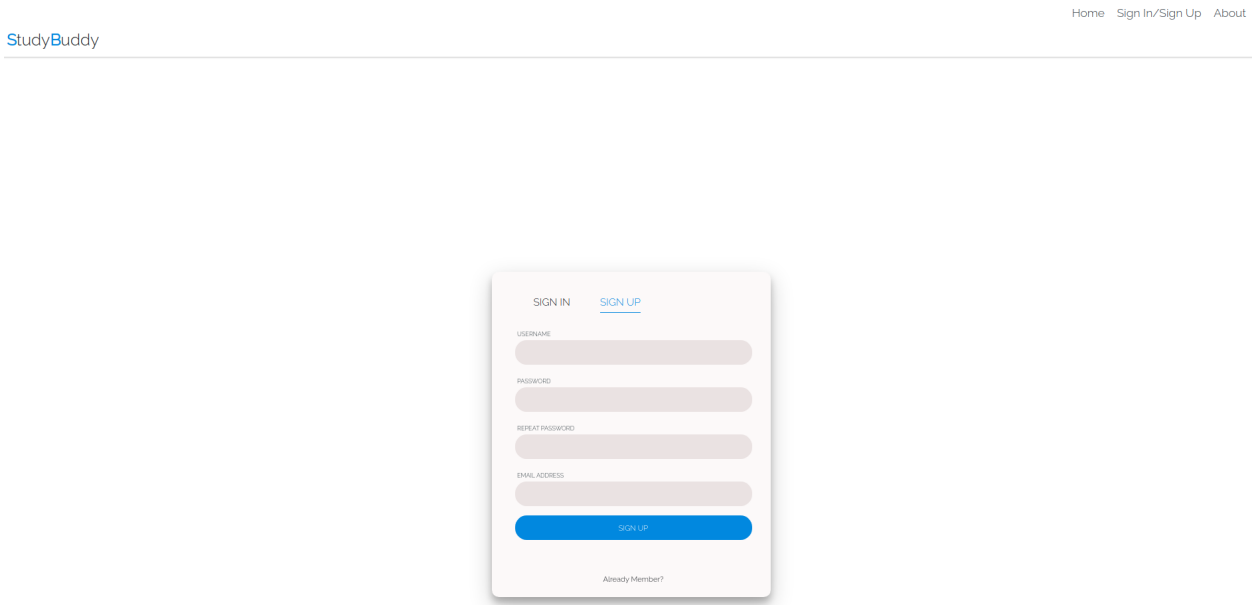
Appendix - User Manual

1. Creating an account

a. Click on Sign In / Sign Up on the landing page:



b. Fill out username, password and email address



c. Verify your email address by clicking the link in the email received

From: From Study Buddy <rdeswardt@gmail.com>
Sent: Thursday, September 16, 2021 9:40 AM
To: Robert de Swardt <r.deswardt@qarar.org>
Subject: Study Buddy Registration

Dear Jarred Lloyd,

Thank you for registering with Study Buddy.

Click here to reset your password using our secure server.

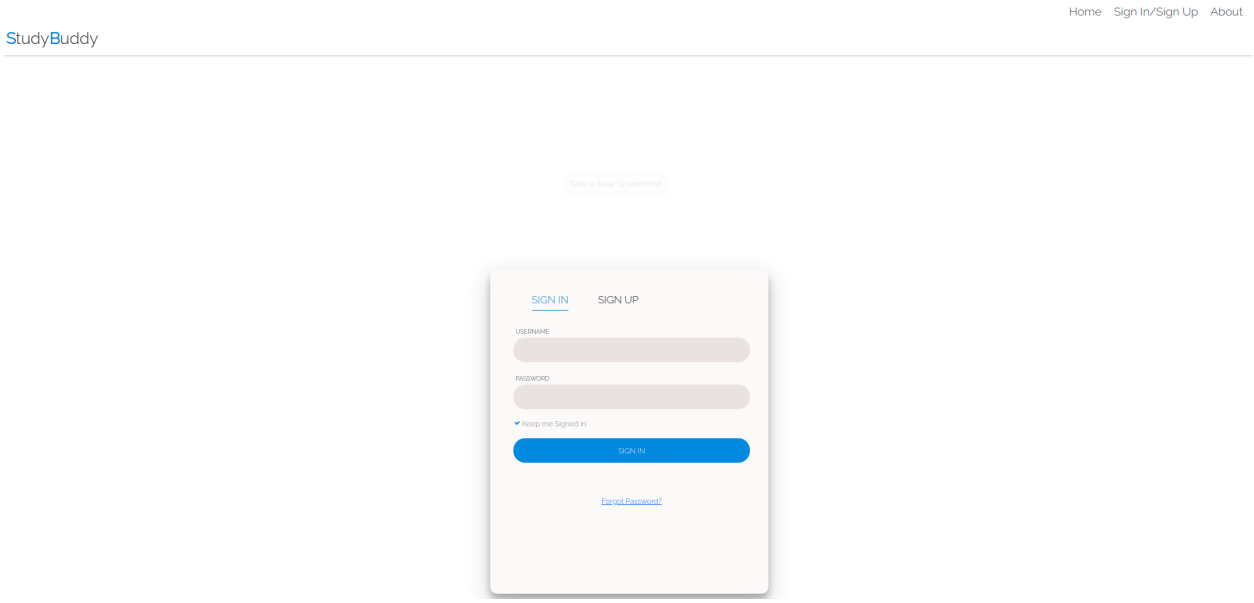
If you did not register with the Study Budy, please ignore this email.

If clicking the link doesn't seem to work, you can copy and paste the link into your browser's address window, or retype it there. After using the link, your account will be activated.

Yours truly,
The Study Buddy Team

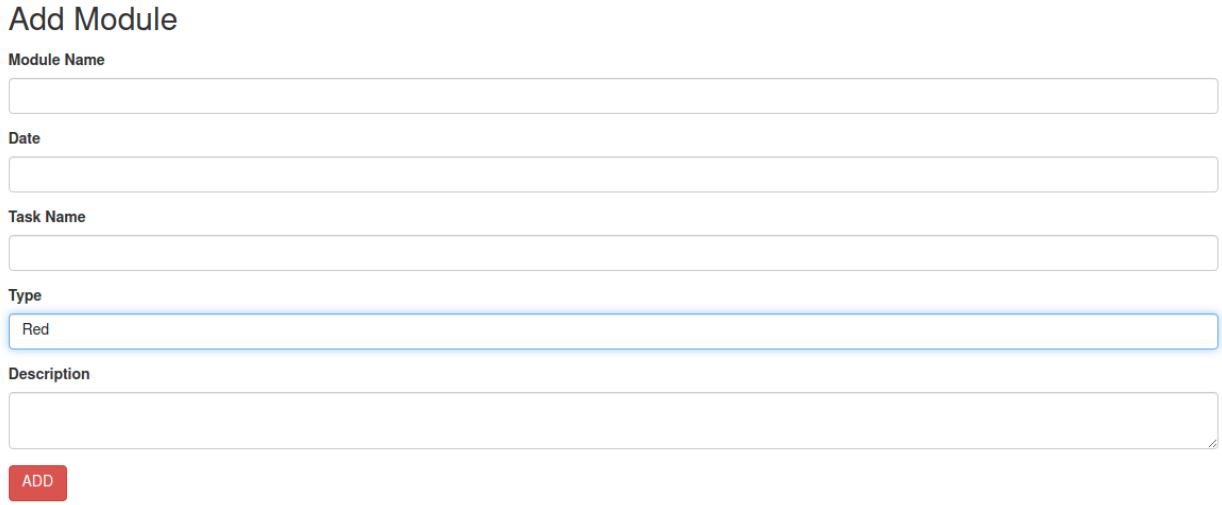
http://www.studybuddy.com
staying on track

d. Log in to your account



2. Overview, Tasks and Modules

a. Add / Remove new modules and tasks



b. Visualise your progress by using the checkboxes.