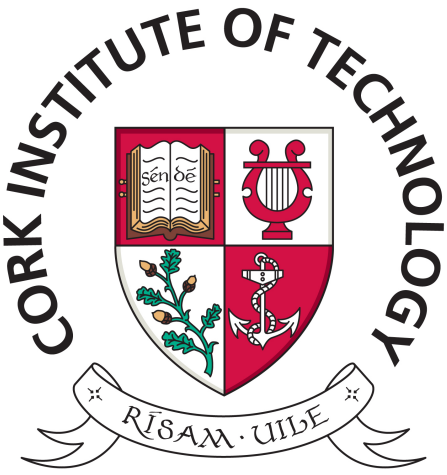


Cornell Notes: A Student Focused Learning Platform with Gamification Elements

by

Steve Walsh

This thesis has been submitted in partial fulfillment for the degree of Bachelor of Science in Software Development

in the
Faculty of Engineering and Science
Department of Computer Science

May 2020

Declaration of Authorship

I, Steve Walsh, declare that this thesis titled, 'Cornell Notes: A student Focused Note taking SaaS Application' and the work presented in it are my own. I confirm that:

- This work was done wholly or mainly while in candidature for an undergraduate degree at Cork Institute of Technology.
- Where any part of this thesis has previously been submitted for a degree or any other qualification at Cork Institute of Technology or any other institution, this has been clearly stated.
- Where I have consulted the published work of others, this is always clearly attributed.
- Where I have quoted from the work of others, the source is always given. With the exception of such quotations, this project report is entirely my own work.
- I have acknowledged all main sources of help.
- Where the thesis is based on work done by myself jointly with others, I have made clear exactly what was done by others and what I have contributed myself.

Signed: _____

Date: _____

CORK INSTITUTE OF TECHNOLOGY

Abstract

Faculty of Engineering and Science
Department of Computer Science

Bachelor of Science

by Steve Walsh

The goal of this project is to provide an online platform for students that encourages them to take, review and share notes by utilizing the Cornell Note Taking System and gamification elements.

The project stems from a number of years of observation that few students actively take notes during lectures. Furthermore, many students lose time when it came to trying to create notes when reviewing material for exams. Prior use of the Cornell Note Taking System over a number of years was found to be beneficial to learning outcomes. Research for the project found that there were no modern applications that directly support this system. This project therefore seeks to implement a platform that I believe could help many future students.

Using Cornell Notes as the basic note taking technique, the project develops a software solution where students can create, edit and review notes. The platform then modernises and extends the Cornell technique by implementing a review system along with gamification through a points-based badge system to provide both a measure of success to the student and assist in engagement by rewarding students who create, share or review their notes.

Acknowledgements

After an intensive period of researching and writing, this note is the is to acknowledge those that assisted me during the completion of this project. I would first like to thank my project supervisor John Creagh who was there to provide supported if required throughout the first section of the final year project and also Byron Treacy who offered fantastic advice and constant support as my Term 2 project supervisor during the implementation phase.

I would also like to thank my fellow students/friends who helped me stay focused and motivated one another to succeed.

Contents

Declaration of Authorship i

Abstract ii

Acknowledgements iii

List of Figures x

List of Tables xiv

Abbreviations xv

1 Introduction 1
- 1.1 Motivation . . . 2
- 1.2 Contribution . . . 3
- 1.3 Structure of This Document . . . 5

2 Background 7
- 2.1 The Cornell Note Taking System . . . 7
  - 2.1.1 Template Design . . . 8
  - 2.1.2 5 R's for taking Effective Notes . . . 9
    - 2.1.2.1 Record . . . 9
    - 2.1.2.2 Reduce . . . 9
    - 2.1.2.3 Recite . . . 11
    - 2.1.2.4 Reflect . . . 11
    - 2.1.2.5 Review . . . 12
  - 2.1.3 Online Resources . . . 13
  - 2.1.4 Studies . . . 13
- 2.2 Software as a Service (Saas) . . . 14
  - 2.2.1 History . . . 14
  - 2.2.2 Architecture . . . 15
  - 2.2.3 Characteristics . . . 16
    - 2.2.3.1 Ease of use . . . 16
    - 2.2.3.2 Customisation . . . 17
    - 2.2.3.3 Accelerated Feature Delivery . . . 17

2.2.3.4 Licensing Model . . . 18
2.2.3.5 Scalability . . . 18
2.2.4 Adoption Drivers . . . 18
2.2.5 Adoption Challenges . . . 19
2.3 Gamification . . . 19
2.3.1 Psychology of Gamification . . . 20
2.3.1.1 Deterdinga and Dixon . . . 20
2.3.1.2 B.J.Fogg . . . 20
2.3.2 Elements of Gamification . . . 22
2.3.3 Design Rules . . . 22
2.4 Competitor analysis . . . 23
2.4.1 Websites . . . 24
2.4.1.1 Office 365 Online . . . 24
2.4.1.2 Google Docs . . . 25
2.4.1.3 EverNote . . . 25
2.4.2 Applications . . . 26
2.4.2.1 Ulysses . . . 26
2.4.2.2 Bear . . . 27
2.4.2.3 Notability . . . 27
2.4.3 Summary . . . 29

3 Cornell Notes: A Student Focused Note Taking SaaS Application 30
3.1 Problem Definition . . . 30
3.2 Objectives . . . 31
3.3 Functional Requirements . . . 31
3.3.1 User Accounts . . . 31
3.3.2 Study Groups . . . 31
3.3.3 Traditional Notes . . . 32
3.3.4 PDF Support . . . 32
3.3.5 Cornell Note Taking System . . . 32
3.3.6 Collaboration . . . 32
3.3.7 Gamification . . . 32
3.3.8 Interface . . . 33
3.4 Non-Functional Requirements . . . 33
3.5 Users . . . 33
3.5.1 Persona 1 . . . 33
3.5.1.1 Goals . . . 33
3.5.1.2 Use Cases . . . 34
3.5.2 Persona 2 . . . 34
3.5.2.1 Goals . . . 34
3.5.2.2 Use Cases . . . 34

4 Implementation Approach 35
4.1 Architecture . . . 35
4.2 Front End . . . 36
4.2.1 Selected Technology . . . 37
4.2.2 Main Components . . . 37

4.2.2.1 Homepage - Login/Register . . . 38
4.2.2.2 Main Container . . . 38
4.2.2.3 Cornell Notes Component . . . 38
4.2.2.4 Creating Review Modal . . . 39
4.2.2.5 Completing Review Component . . . 39
4.2.2.6 Badges Modal . . . 40
4.2.2.7 Header . . . 40
4.2.2.8 Sidebar . . . 41
4.2.3 Other Components . . . 41
4.2.3.1 My Account Modal . . . 41
4.2.3.2 Font Editing Toolbar . . . 42
4.2.3.3 PDF Viewer Component . . . 42
4.2.3.4 Alert and confirmation Modals . . . 42
4.2.4 Imported Libraries . . . 42
4.3 Back End . . . 43
4.3.1 Selected Technology . . . 43
4.3.2 APIs . . . 44
4.3.2.1 Users . . . 45
4.3.2.2 Groups . . . 45
4.3.2.3 Notes . . . 45
4.3.2.4 Cornell Notes . . . 46
4.3.2.5 Folders . . . 46
4.3.2.6 Tags . . . 46
4.3.2.7 Reviews . . . 47
4.3.2.8 PDFs . . . 48
4.3.2.9 Badges . . . 48
4.3.3 Database . . . 52
4.3.4 Selected technology . . . 52
4.3.5 Database Tables . . . 52
4.3.6 Deployment Service . . . 57
4.4 Risk Assessment . . . 57
4.4.1 Personal Skills . . . 57
4.4.2 Development Challenges . . . 58
4.5 Methodology . . . 59
4.6 Implementation Plan Schedule . . . 60
4.7 Evaluation . . . 62
4.8 Prototype . . . 62
4.8.1 Front End Prototype . . . 62
4.8.2 Back End Prototype . . . 65

5 Implementation **67**
5.1 Project Changes before Implementation . . . 68
5.1.1 1. Relational vs Document-Store Database . . . 68
5.2 2. Sprint Structure . . . 69
5.2.1 Sprint 1 - Week 1 . . . 72
5.2.1.1 Feature . . . 72
5.2.1.2 Tasks . . . 72

5.2.1.3 Challenges . . . 73
5.2.1.4 Retrospective . . . 73
5.2.2 Sprint 2 - Week 2 . . . 74
5.2.2.1 Feature . . . 74
5.2.2.2 Tasks . . . 74
5.2.2.3 Challenges . . . 75
5.2.2.4 Retrospective . . . 75
5.2.3 Sprint 3 - Week 3 . . . 76
5.2.3.1 Feature . . . 76
5.2.3.2 Tasks . . . 76
5.2.3.3 Challenges . . . 77
5.2.3.4 Retrospective . . . 77
5.2.4 Sprint 4 - Week 4 . . . 78
5.2.4.1 Feature . . . 78
5.2.4.2 Tasks . . . 78
5.2.4.3 Challenges . . . 79
5.2.4.4 Retrospective . . . 79
5.2.5 Sprint 5 - Week 5 . . . 80
5.2.5.1 Feature . . . 80
5.2.5.2 Tasks . . . 80
5.2.5.3 Challenges . . . 80
5.2.5.4 Retrospective . . . 81
5.2.6 Sprint 6 - Week 6 . . . 82
5.2.6.1 Feature . . . 82
5.2.6.2 Tasks . . . 82
5.2.6.3 Challenges . . . 82
5.2.6.4 Retrospective . . . 83
5.2.7 Sprint 7 - Week 7 . . . 84
5.2.7.1 Feature . . . 84
5.2.7.2 Tasks . . . 84
5.2.7.3 Challenges . . . 84
5.2.7.4 Retrospective . . . 85
5.2.8 Sprint 8 - Week 8 . . . 86
5.2.8.1 Feature . . . 86
5.2.8.2 Tasks . . . 86
5.2.8.3 Challenges . . . 87
5.2.8.4 Retrospective . . . 87
5.2.9 Sprint 9 - Week 9 . . . 89
5.2.9.1 Feature . . . 89
5.2.9.2 Tasks . . . 89
5.2.9.3 Challenges . . . 90
5.2.9.4 Retrospective . . . 90
5.2.10 Sprint 10 - Week 10 . . . 92
5.2.10.1 Feature . . . 92
5.2.10.2 Tasks . . . 92
5.2.10.3 Challenges . . . 92
5.2.10.4 Retrospective . . . 93

5.2.11 Sprint 11 - Week 11 . . . 93
5.2.11.1 Feature . . . 93
5.2.11.2 Tasks . . . 93
5.2.11.3 Challenges . . . 94
5.2.11.4 Retrospective . . . 94
5.2.12 Sprint 12 - Week 12 . . . 94
5.2.12.1 Feature . . . 94
5.2.12.2 Tasks . . . 94
5.2.12.3 Challenges . . . 95
5.2.12.4 Retrospective . . . 95
5.3 Difficulties encountered . . . 95
5.3.1 Vue.js Knowledge . . . 96
5.3.2 Covid-19 . . . 96
5.4 Learnings . . . 97
5.4.1 Vue.js . . . 97
5.4.2 Golang . . . 97
5.4.3 Project Management . . . 97
5.5 Actual Solution Approach . . . 99
5.5.1 Architecture of solution . . . 99
5.5.2 Risk Assessment . . . 99
5.5.3 Development Methodology . . . 100
5.5.4 Implementation Schedule . . . 100
5.5.5 Evaluation Approach . . . 100

6 Testing and Evaluation 102
6.1 Service Testing . . . 102
6.1.1 Back End Service Testing . . . 103
6.1.2 Front End Service Testing . . . 104
6.1.3 Database Service Testing . . . 106
6.2 Platform Evaluation . . . 107

7 Discussion and Conclusions 108
7.1 Solution Review . . . 108
7.2 Project Review . . . 109
7.3 Conclusion . . . 110
7.4 Future Work . . . 111
7.4.1 Vue.JS Structure Change . . . 111
7.4.2 Customised Reviews . . . 111
7.4.3 PDF Support . . . 112
7.4.4 Enhanced Collaboration . . . 112
7.4.5 Enhanced Gamification . . . 112

Bibliography 113

A Code Snippets 117

B Wireframe Models **124**

List of Figures

2.1 Sample Cornell Note Taking System Template! 8

2.2 Example of Answer section of note taken using the Cornell Note Taking System [1] 10

2.3 Example of Cue section of note taken using the Cornell Note Taking System [1] 10

2.4 Hermann Ebbinghaus' Forgetting Curve! 12

2.5 University of Maine and Cornell University promoting the method 13

2.6 SaaS illustration 16

2.7 SaaS Application Teamwork Desk's Custom Fields 17

2.8 Gamification is becoming mainstream 20

2.9 Games and gamification classification 21

2.10 B. Fogg's Behavioural Model 21

2.11 MDA Framework mapping 22

2.12 Word Templates for Cornell Note Taking System 24

2.13 Google Doc's Collaborative Editing Feature 25

2.14 Bear Mac App's interface showing Tags and Markdown 27

2.15 Notability - Drawing Cornell Note Taking System Template 28

2.16 Competitor Analysis Feature Comparison Table 29

4.1 Architecture Component Overview 36

4.2 APIs to support Users 45

4.3 APIs to support Groups 45

4.4 APIs to support Notes 46

4.5 APIs to support Cornell Notes 47

4.6 APIs to support Folders 48

4.7 APIs to support Tags 48

4.8 APIs to support Reviews 49

4.9 APIs to support PDFs 49

4.10 Points earned by creating items 50

4.11 Points earned by sharing items 50

4.12 Badges that can be earned 51

4.13 Database Table for Users and Folders 53

4.14 Database Table related to Notes 53

4.15 Database Table related to Groups 53

4.16 Database Table related to Reviews 54

4.17 Database Table related to PDFs and Tags 54

4.18 Database Table related to Badges 54

4.19 Database Tables relationship diagram 56

4.20 Implementation Sprint Schedule 61

4.21 Wire frame sketch of front end displaying the main Cornell Notes editor, the PDF viewer and the platform header 63

4.22 Wire frame sketch of front end displaying the sidebar with navigation routers for Notes, Reviews and Tags 64

4.23 Vue.JS prototype showing partial header and sidebar 64

4.24 Back End API Routes for Folder Requests 65

4.25 Back End function for Folder API Request Create 65

4.26 Back End function for Folder API Request Get 66

4.27 Postman API request to Back End for Folder information which show folders name returned along with correct HTTP status code 66

5.1 New Sprint structure 70

5.2 Left: Old Sprint structure, Right: New Sprint structure 71

5.3 Tasks for Sprint 1 72

5.4 Top Left - Vue.js service, Bottom Left Go Back End service, right - database service 74

5.5 Tasks for Sprint 2 74

5.6 Tasks for Sprint 3 76

5.7 Sprint 3 - initial implementation for Login(right) and Register(left) Modals. 77

5.8 Tasks for Sprint 4 78

5.9 Sidebar showing closed folders and displaying all tags(left) and sidebar showing one open folder, it's items and their icons(right) 79

5.10 Tasks for Sprint 5 80

5.11 Original Tasks for Sprint 6 82

5.12 Cornell Note taking feature 83

5.13 Add New note Modal(left) and edit note modals (right) 83

5.14 Tasks for Sprint 7 84

5.15 Basic Note taking Feature 85

5.16 Tasks for Sprint 8 86

5.17 Cornell Note Review 87

5.18 Cornell Note Review Summary 88

5.19 Tasks for Sprint 90

5.20 Badges and Rewards feature 91

5.21 Score notification for earning points 93

5.22 Left: Old Sprint structure, Right: New Sprint structure 101

6.1 GetFolders API test using Postman showing successful HTTPS 200 response 104

6.2 Spread Sheet showing APIs Implementation Result including reasons if not implemented 105

6.3 Left: Sign up error - user already exists, Right: add note errors - empty note name and trying to create a new folder with the same title as existing folder 105

6.4 Testing cornell_users table in database using MySQL Workbench 106

6.5 Functional and Non-functional requirements pass/fail with comments . . . 107

A.1 Front End - main.js - project imported libraries and App creation function 118

A.2 Front End - Badges.Vue - progress bar and scoring components 119

A.3 Front End - routes.js - shows the routes handled 119

A.4 Front End - styles.js - Main styling colors for platform 120

A.5 Front End - user.js - Section of API request for Logging in User 120

A.6 Front End - store.s - Section of Vuex store for shared component properties 121

A.7 Back End - API request Middle ware function 121

A.8 Back End - User Model Struct 122

A.9 Back End - Sample of API routes handled 122

A.10 Back End - Sample of API Request 123

A.11 Back End - Sample of DB Transaction Request 123

List of Tables

Abbreviations

SaaS	**S**oftware **a**s **a** **S**ervice
MDA	**M**echanics **D**ynamics **A**esthetics
UI	**U**ser **I**nterface
API	**A**pplication **P**rotocol **I**nterface
JSON	**J**ava **S**cript **O**bject **N**otation
AWS	**A**amazon **W**eb **S**ervices
CRUD	**C**reate **R**etrieve **U**pdate **D**elete
CRM	**C**ustomer **R**elationship **M**anager
ISO	**I**reate **S**etrieve **O**pdate
ASP	**A**pplication **S**service **P**rovider
IT	**I**information **T**echnology
LAN	**L**ocal **A**rea **N**etwork
HTTPS	**H**yper **T**ext **T**ransfer **P**rotocol **S**ecure
CORS	**C**ross **O**rigin **R**esource **S**haring
TLS	**T**ransport **L**ayer **S**ecurity
SSL	**S**ecure **S**ocket **L**ayer
CSS	**C**ascading **S**tyle **S**Sheet

Chapter 1

Introduction

Cornell Notes is a SaaS based note taking and learning platform that enables students to utilise the Cornell Note Taking Method, developed at Cornell University in the 1940s by Walter Pauk [2], to construct study notes individually or collaboratively as part of a group.Users are also able to create fully customised reviews from their notes in a area of the platform to assist with their revision. Cornell Notes implements gamification through a points based badge system to provide both a measure of success to the user and assist in engagement.

I firmly believe that two of the most persistent and widespread problems for students today related to taking notes during classes are the lack of student engagement and motivation, and the lack of guidance in how to take effective notes to be later expanded upon and reviewed. I have witnessed these first hand throughout secondary school and during each of the semesters leading up to this project during my time at CIT, having myself experienced the difficulty of trying to take note for a subject or on a topic that doesn't particularity interest me. This lead to challenges when it came time to review a topic and I saw that it is more difficult to learn anything if you are not motivated and engaged in the process and through others noticed the frustration that occurred due to not having implemented an organised note taking system when it came time to study for an exam.

The gamified aspects of Cornell Notes is designed and implemented to try and alleviate part of these problems for students by providing native tools within the application to motivate the student when it comes to reviewing their notes by automatically generating relevant questions when it comes time to study and by providing a scoring system to entice the student to create and review notes.

The core features of Cornell Notes are firstly its natively integrated Cornell Note Taking template that can be used to take effective notes and also provides the ability to generate review notes based on the questions added. Furthermore Markdown support is provided to allow the student an method of generating clean looking notes withing needing to move away from the keyboard. Most of the note taking applications that I have used all focus primarily on the taking of the notes but do not tend to have any features specifically centred around the reviewing of notes or method that can be used to study those notes in a more effective method than just reading them, Cornell Notes has a core section called Review that can be used to generated questions based on what is added to the Cue section of the Cornell Template and will provide options to the reviewer to view the answer related to the Cue or the note's summary. Collaboration plays a large part within Cornell Notes and students will also have the ability to be able to generate notes that are shared between their peers to build a larger grouping of questions to be able to practice when studying a topic.

During my work placement I gained experience working within a SaaS company and from this decided to use the SaaS framework for this project where Cornell Notes will be accessible via any online browser, the front-end is written using Vue.JS, the back-end API endpoints are all written using Golang. These two separate instances are run through containers using Docker and connect to an SQL database which provides long term persistence.

1.1 Motivation

There are four main elements that motivated me to select this idea for my final year project. The first being that I have personally been using the Cornell Note Taking Method throughout my time at CIT and found it to be an effective method for me in helping with both building a collection of review notes and when it comes to reviewing the aforementioned notes. I created notes usually either during or after each lecture but unfortunately as there was not an application or service that provided me with a comprehensive tool to create notes using the Cornell Note Taking System, I was forced to utilise various templates or create templates and import them into existing note taking applications such as Evernote, OneNote, Bear, Google Docs and Notability. During the testing of each of these different programs and methods to try and replicate the Cornell Note Taking System, I encountered various challenges that prevented me from being able to conveniently able to create notes using this system.

The second being that I have noticed during my time at CIT that not many students appear to engage actively in class by taking notes and that many people I have encountered struggle to create effective study material. I used circulate my notes with the class when it came to preparing for tests as many of those in my class didn't have any curated notes of their own to reference when studying. I also used to promote this method when I was a PALS Study Leader and assisting first years with learning different techniques to effectively create notes and review material during my second year at CIT. During these times I received very positive feedback from other students on the Cornell Note Taking System and this has been a driving factor in what has inspired me to build the Cornell Notes platform for this project.

From initially researching the effectiveness of taking notes both in class [3] and from reading material/ notes [4] that it is known that students that participate in taking notes during class or creating review material after experience better retention of material and also in particular that even "individuals who have poor working memory, can still take effective notes if they use a note-taking strategy"[5] which has contributed greatly to my motivation as the third element.

The final reason that has motivated me to create this platform using the SaaS distribution model is my new found interest in SaaS after having the opportunity during my third year internship to work for a local Cork SaaS based company Teamwork and having been exposed to the advantages of using this model when it comes to real time collaboration, redundancy, upkeep and improvement lead me to want to incorporate this distribution into my final year product. I am interested in increasing my knowledge in two of the main programming languages used by Teamwork being Vue.JS and Golang which I will be utilising to implement the front end and back end of the Cornell Notes platform respectively.

1.2 Contribution

There are dozens of currently available applications that provide the option for users to take notes however as most are designed to be used by a range of user types they tend to implement feature that are more generalised such as Evernote, Google Docs and OneNote where templates are required to utilise the Cornell Note Taking Method. From examining some of the most popular note taking applications targeted at students, many are platform specific such as Bear or Ulysses which are only available on MacOS and iOS and do not natively provide a methodology for note taking or review and do not support the integration of templates that would provide the ability to utilise available note taking systems.

Cornell notes is designed to fill the gaps between these popular note taking applications by providing a platform that will include the the core features of these applications being:

- Nested Folder Structure for collections
- Tag support for searching
- Standard note taking tools
- Note sharing

Cornell Notes will also include the following that will enhance the platform to being focused on students:

- Markdown editor
- Cornell structured note taking system
- Review system
- Student focused syntax
- Gamification elements to promote engagement
- Cross Platform Availability

There are obvious benefits from creating a platform that will assist students in reviewing content for their modules including feature that promote teamwork among peers to create collections and quizzes based on collaboratively taken notes that is directly integrated with the platform they used to take the notes. Time is a precious commodity for students and anything that will help reduce the time it takes to create effect notes, switching between applications, creating templates, keeping focus along with motivation and promoting sharing information with others will benefit students.

In terms of the contribution this project will have to my learning, the technical stack I have selected to use for the implementation phase which include languages like Golang for the back-end, Vue.JS for the front-end, SQL for the database and services such as Docker and Jenkins for deployment are ones that I have been exposed to from my internship but now want to have the opportunity to get hands on with and also increase my knowledge of. By creating a full SaaS platform this project will contribute greatly to my understanding of web based applications that utilise REST APIs to send information between components of a micro service architecture to facilitate a very quick, clean

looking scale-able service. This will also enhance my language specific programming skills, of which front-end being one particular area that I am looking forward to getting better at during the progression of this project as it is an area I find more challenging than others which I have tried.

1.3 Structure of This Document

Chapter 2: Provides the reader with research focused content on the following topics:

- The Cornell Note Taking System
- Software as a service
- Gamification

This chapter concludes with a competitor analysis on various applications and solutions that provide different functionality that will be present in Cornell Notes.

Chapter 3: Design architecture and strategy is investigated and established for the project looking at the overall problem statement, definition and the project definition. The projects functional and non functional requirements and then outlined and cussed and use cases are presented.

Chapter 4 : The implementation approach to the project is discussed in this chapter, identifying the how the objectives are solved with different technologies and any difficulties which arose during development. Beginning with a description of the overall architecture chosen for the project and then describing the different components that will be implemented in the development of the front end and back end of the platform. The implementation approach, methodology and evaluation steps are also outlined in this chapter concluding with summary of prototypes creating during this research phase of the project.

Chapter 5 : The fifth chapter of this report will follow the progress completed during the implementation phase of this project. The structure will be to outline any changes that might have been made related to any of the plans outlined in previous chapters and then will outline the work completed during each of the Sprints that were used to structure and manage the tasks required to be completed.

Chapter 6 : The sixth chapter focuses on how features were tested during the implementation phase and how I evaluated the progress and completion of the project.

Chapter 7 : The seventh and final chapter as part of this phase of the final year project discusses the challenges encountered since the commencement of this project along with conclusions generated.

Chapter 2

Background

2.1 The Cornell Note Taking System

The Cornell Note Taking System was developed by Walter Pauk, and education professor and study centre director at Cornell University in the 1940s. Walter developed the system as a new method to enable students studying at Cornell to be able to better organize their notes by providing a systematic framework for condensing and organising them.

The Cornell Note Taking System was later introduced to a much wider audience with the release of Walter's book, 'How to Study in College first Edition' in 1961, which is widely regarded as the bench mark to which similar texts are compared [2]. How to Study in College has since been revised, re-released and is current on its eleventh edition(2014), the Cornell Note Taking Method remains present in the book not only being focused on with a dedicated chapter but also being used throughout the entire book as a template for every section throughout the book[6].

When developing the Cornell Note Taking System Walter derived inspiration from the SQ3R System, a note taking system designed around five R's that together would provide an effective framework and foundation for students to create organised study notes [6] . In conjunction with developing these five R's , a template design was created to assist the student in creating organised notes for effective studying and are later described in this section.

2.1.1 Template Design

The goal of the template is to create an organised section of notes that allows the student to quickly attach verbal meaning to information in an organised format to assist both when taking notes in class and when summarizing and reviewing them at a later date. The basis of the template is the division of a page into a minimum of two sections, one for Cue statements and the second for Answers with the option a third section that is used for a summary of the overall section. A sample template can be seen in Figure 2.1

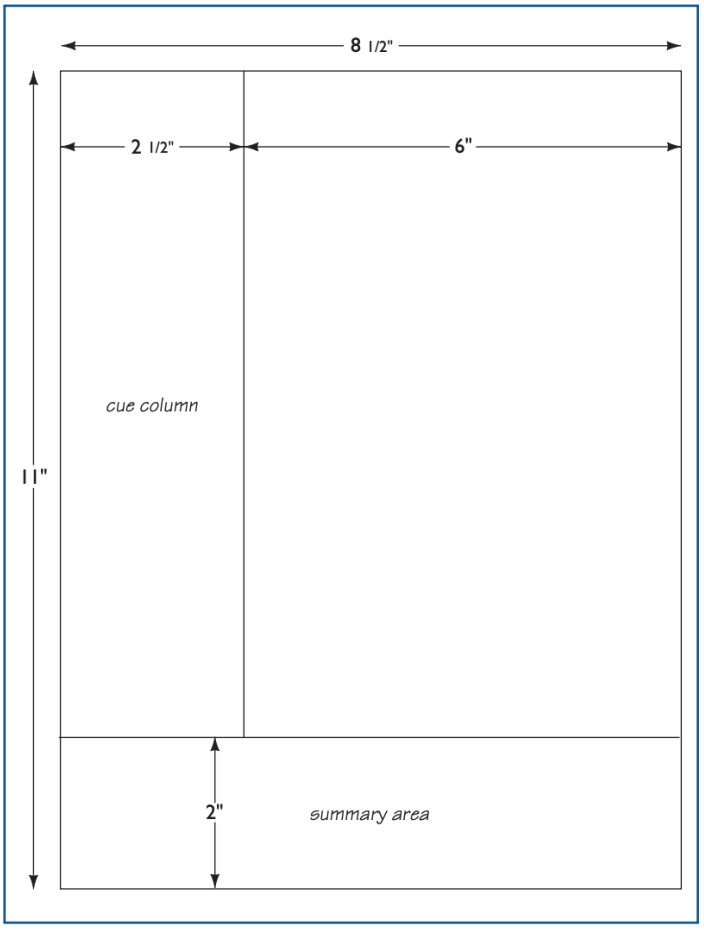


FIGURE 2.1: Cornell Note Taking System Template Sample[7]

The measurements given in Figure 2.1 for the size of each section are a guidance and not a rule as they can be edited to suit a students preference however there should remain a clear divisor between the cue column and the Answer section on the right and again between both of these and the Summary section; should be included.

The main section on the right is used to take notes in class or for providing answers/ information when creating review notes that are related to each of the Cues. To help

create organised notes it is recommended to use 'telegraph notes' to omit any non-critical words which both saves space but also makes the it much cleaner looking.

Example: The following sentence:

"Redis is an open source, in-memory data structure store which is used as a database, cache and message broker to store information in a key-value pair system"

Could be be shortened to:

"Open source, in-memory data structure. Stores: Key-value pair, Uses: DB, cache, message broker"

With added emphases in the form of colored, highlighted or bold-ed font for key worlds of sections such as "Stored, Uses" to catch the students eye when reviewing the Answer section.

2.1.2 5 R's for taking Effective Notes

2.1.2.1 Record

During class the only section that will be filled is the main Notes section on the right hand side, the cues and summary sections are filled after class or when creating review notes.

Any type of information can be entered into this section from paragraphs, definitions, graphs, equations, figures and contains the 'meat' of the lesson information. It is recommended to try and omit any information that is not critical to the learning by being selective in the information that is noted down and Figure 2.2 below illustrates what the main section might look like once filled. You can see the use of both colors and diagrams in the figure which helps separate the points but also make it appear cleaner and easier to read.

2.1.2.2 Reduce

It is not a goal to try and always create the most detailed as possible notes during a lecture as this can detract from the students attention and ability to great notes covering the entire lessons contents and thus it is recommended to always review to reduce notes after the lesson. The goal of the reduce step is to narrow down the most important facts for the body section by extracting key words, definitions, phrases and formatting the information to help extract questions based on each of the information points. This

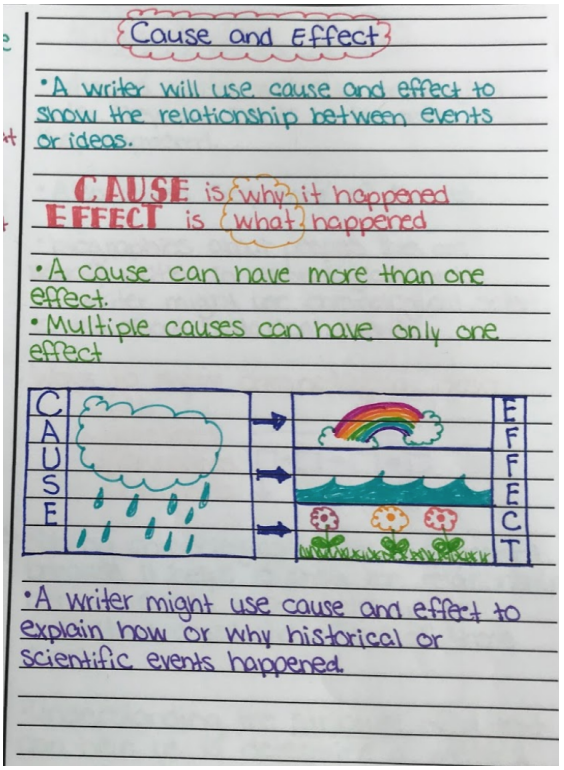


FIGURE 2.2: Example of Answer section of note taken using the Cornell Note Taking System [1]

helps enable the student to utilise the time they are investing in taking notes more efficiently and provide a framework that will later be used for revision.

These key words, phrases and potential exam questions are added to the Cue section of the template as displayed below in figure 2.3 where you can see that there are key definitions for 'cause' and 'effect' along with examples and signal words.

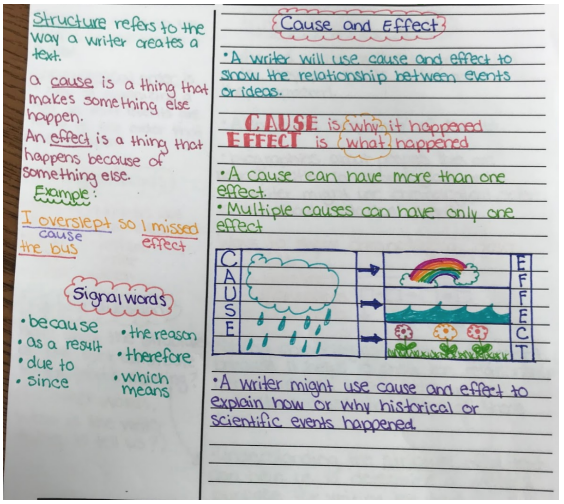


FIGURE 2.3: Example of Cue section of note taken using the Cornell Note Taking System [1]

These act as memory cues so that when students are reviewing the material, they will assist in recalling the ideas or facts related to the information covered in the note. The questions help clarify the meanings of the facts and ideas and help prepare for potential exam questions. They provide a tool that the student can use when reviewing the material during subsequent Rs in the system, particularily recite and review.

The answer section is also completed during this phase to add notes related to each of the Cues that will provide the required information in a concise form. The Answers section helps inform the student of the explanation behind any Cue which they may have forgotten during later sections of recitation and revision

2.1.2.3 Recite

Recitation is a very powerful and important process in the retention of information. It involved not just reading or writing the information but also stating out loud in the students own words, the facts, ideas and ideas they are trying to learn. This is an effective way to learn because hearing your thoughts help you sharpen your thinking process, and stating your ideas and the facts in your own words challenges you to think about the meaning of the information.

The Cornell Note Taking System assists with recitation by providing the template which allows the student to check the summary section first. Using the summary they can determine if they can recall the information contained in the note or if they should review it. In the instance where the student wants to review the material they can then, using the Cue column with key words and questions, start reciting information about each topic and should they not know and answer or want to confirm they answer they can uncover the Answer section which offers all the relevant information they need to know about each topic.

2.1.2.4 Reflect

Reflection involves pondering or thinking about the information that was learning to reinforce deeper learning by relating ideas and facts to other learning and knowledge through a process known as scaffolding [8]. It is recommended to pause after reciting information on each page and contemplating the following questions:

- How do the ideas and facts fit into prior learning and knowledge?
- How can they be applied?

- What is important about knowing these facts and ideas?
- What is the significance of these facts and ideas

By answering these questions the student both helps create the foundation for the final step, review.

2.1.2.5 Review

The Fading, or Decay, theory proposes the idea that memories fade due both the passage of time and a decrease in retrieval of those memories[9]. It was deduced from the idea of the Forgetting Curve which was proposed in the 19th century by German psychologist Hermann Ebbinghaus deduced which is a mathematical formulae that describes the rate at which information is forgotten after its learned. This formulae which is represented in graph form in Figure 2.4 shows after within just 24 hours of obtaining information a person retains just 25% of that information and a recent study undertaken in the Psychology Department of the University of Amsterdam has confirmed this to be still accurate [10].

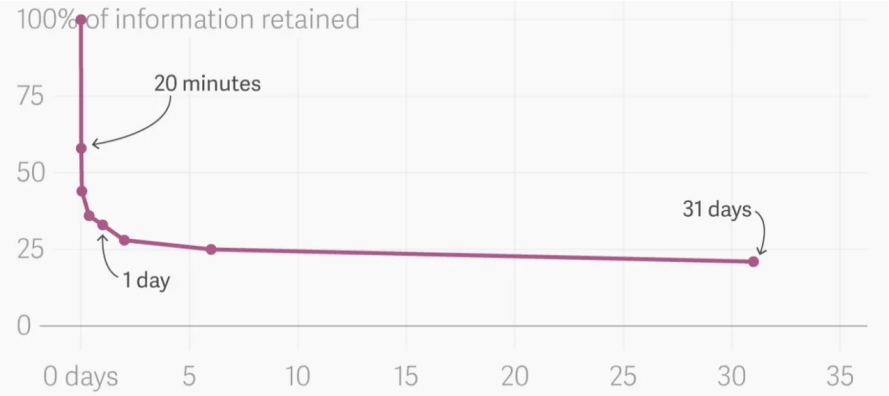


FIGURE 2.4: Hermann Ebbinghaus' Forgetting Curve

The review section of the Cornell Note Taking System's template allows the student to create a short summary of the page of sections notes providing them with an easy accessible sample for them to review. The summary should be a subsection of the cue or main sections created in the students own words where if they fail to recall the information the summary when reviewing can quickly read the cues and subsequently if required the main section to regain any lost knowledge.

2.1.3 Online Resources

The Cornell Note Taking System is recommended by thousands of universities around the world and a search of the .edu domain for 'Cornell Note System' shows up with over 6 million results with most providing guidelines and templates to assist in implementing the system such as the examples in Figure 2.5

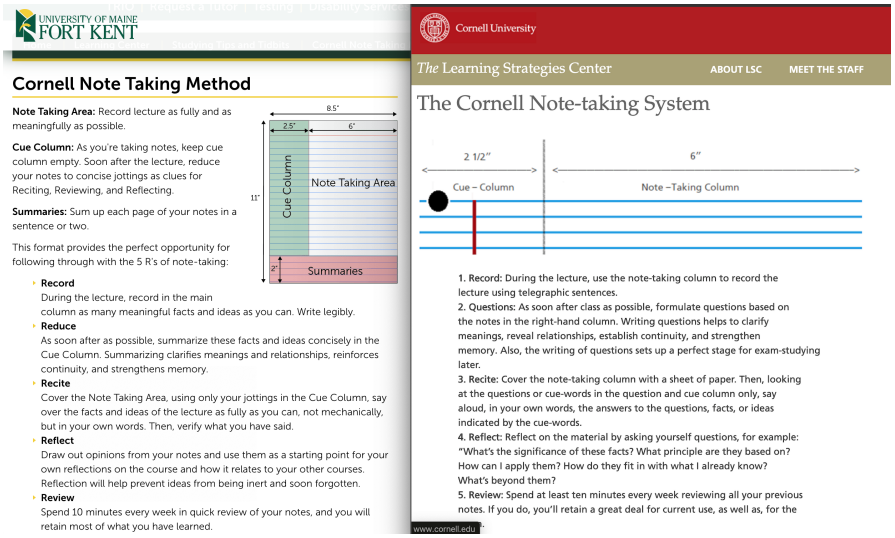


FIGURE 2.5: University of Maine and Cornell University promoting the method

Most examples encountered cover the same core elements outlining the template, the five R's and outline the benefits of both taking effective notes and effectively studying them after lessons. Often there are links to downloadable templates that students could print and utilise if taking notes offline using traditional pen and paper methods and others at times provide links to digital templates for platforms such as Microsoft Word or Google Docs.

2.1.4 Studies

In 1994 Carol Backman, a math teacher conducted a study on students in her classroom on finding an effective note taking system to increase the number of students in her class that take notes. She allowed her students to create and develop there own templates and sets of notes and then utilised surveys to gather information from the students and extract commonalities amongst the notes. Some of these included"definitions, diagrams, sample problems, rules or steps to do a problem, and a list of items that gave them trouble" [11]Backman observed an increase in both note-taking and in student grades. Over 30 years later a similar study was conducted in a math classroom in Minnesota where a teacher Michael Swenson undertook the same experiment with slightly different approach

The results of these correlate directly with other guidance provided in various 'Effective Study' guides such as in A Handbook For Classroom Instruction that Works where the authors discovered that when students are trying to directly record everything that is heard, verbatim, that the recording process takes up a lot of the student's working memory leaving them without as much room to actually analyse the information information. [12].There are additional studies that show an increase in student achievement by using a guided note taking system that promotes active thinking over writing verbatim [13], [14], [15].

Studies that have specifically targeted the Cornell Note taking System also correlate with these results by shows the existence of relationship between Cornell note-taking method instruction and EFL learners' grammar learning. It strongly confirms the conclusions drawn by Bui [5] and Chang[4] on how justified the note-taking process is in itself. Where it was concluded that improvement on the post-test of both experimental and control group clearly shows that note taking does improve test performance and that it is even more significant if done in an organized way such as Cornell method. [16]

2.2 Software as a Service (Saas)

SaaS is a software distribution model allows for software vendors to to service clients though hosting a web based infrastructure rather than shipping a disk or providing a downloaded application that the client would host internally. Today, the term SaaS is often used interchangeable with "cloud computing" and has an interesting history over the past, almost 60 years from when it was first introduced in the 1960s and today due to the internet and modern global communication technologies, has become one of the most popular way to distribute software for numerous companies.

2.2.1 History

In the 1950s, computers were physically huge machines and extremely expensive. Few small and medium-sized businesses could afford to invest in a computer to take advantage of its processing power as the cost versus the time they would have some one using it did not equate as they would typically be using it for up to 8 - 12 hours a day being the standard opening time of a business. There was a realisation that whilst any individual user would make inefficient use of a computer compared to its cost, a large group of users would not due to their pattern of interaction which allows for the processing power of the computer to be used all the time as when the activity of one user stops, it allow for the activities of other users to be processes. The concept

was called Time-Sharing and first noted in 1954 by John Backus, a computer scientist responsible for the creation for FORTRAN [17], and the first commercial time-sharing system was introduced in 1963 by Dartmouth University as the Dartmount Time Sharing System allowing students to have free access to computing power [18]. In the mid 1960s companies such as IBM began offering time-sharing through a subsidiary known as The Service Bureau Corporation, Tymshare which allowed clients of small, medium and even large companies, educational institutions, and government agencies to connect to their host machines and take advantage of the processing power they were able to provide which at this time were primarily used for CRM, payroll and accounting systems.

From the 1970s to 1990s the decreasing cost of computer hardware lead to the ability for companies to start providing more employees with an individual computer and as the demand for time-sharing systems decreased the model adapted into a model that shifted the focus to LANs. In these in-house systems, applications were hosted on local machines and critical business data was kept separately on a central server. Employees connected to the LAN to access these applications and data.

This was seen as an early form of cloud computing and in order to manage these LANS companies began to implement IT departments, managers and build up their IT infrastructure which for many was a costly experience, particularly initially when all the hardware would need to be purchased. This along with issues such as the cost of hard-drive space and the rise of bloatware appearing on computers after installing applications lead to the need for another solution to be introduced. With the rise of the internet in the 1990s ability for data and applications for be stored remotely became attractive to numerous companies and lead to the founding of some of the biggest SaaS companies available today such as SalesForce and Google who offered this service and were known as Application Service Providers (ASP) before later being called Software as a service in 2005 by John Koenig for the SDForum Software as a Service Conference [19] which is still used as the term for companies that use the internet to deliver software to the end user.

2.2.2 Architecture

The majority of SaaS applications are based on a multi-tenant architecture. With this model, a single version of the application, with a single configuration (operating system, network and hardware), is used for all users ("tenants"). To help support scalability, the application can be installed on multiple machines which is called horizontal scaling. Often a second version of the application is set up to offer a select group of users access to pre-release versions of the applications for testing purposes or a beta features section

FIGURE 2.6: SaaS illustration [20]

might be provided where users can choose to activate or not knowing they might not function fully. This is completely contrasted with traditional software, where multiple physical copies of the software — each potentially of a different version, with a potentially different configuration, and often customized — are installed across various user sites [21].

There are two main varieties of SaaS [21]:

1. Vertical SaaS: Software which answers the needs of a specific industry e.g., software for the healthcare, agriculture, real estate, finance industries.
2. Horizontal SaaS: The products which focus on a software category e.g., marketing, sales, developer tools, HR, but are industry neutral rather than specific

2.2.3 Characteristics

2.2.3.1 Ease of use

One could say that the singular most important factor in software adoption is ease of use.The International Organization for Standardization publishes research and recommendations covering human interaction with computers and software. ISO standard 9241 defines usability as: "The extent to which a product can be used by specified users to achieve specified goals with effectiveness, efficiency, and satisfaction in a specified context of use."[22] More simply, it comes down to how easy is it for the user to get the job done. As all that is needed to connect to a SaaS system is an internet enabled device a huge focus is typically put on the user interface to ensure compatibility across screen sizes and to allow for different inputs. This tends to result in an easy to navigate

graphical user interface design along with the inclusion of tutorials and demonstration data that can be used to help users learn how to use the software.

2.2.3.2 Customisation

Systems architects in the SaaS area face a constant back and forth with the product teams and sales teams, to decide where do they draw the line between standardization and customization. Henry Ford said that "Any customer can have a car painted any color that he wants, so long as it is black." but after after General Motors started offering customers a variety of colors Ford changed their tune and also started providing customers with the colors they asked. In SaaS applications the ability to provide custom fields, alter the color, base configuration and settings to suit their individual needs allows the distributors to widen their target users and also provide users to tailor the application to specifically suit their individual needs. Teamwork Desk, a ticket management software, provide the use of custom fields allowing users to customise different input fields and types for their support system ensuring they receive all the information they require from their user as seen in Figure 2.7

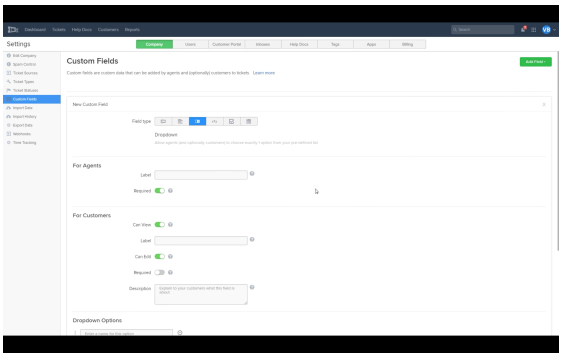


FIGURE 2.7: SaaS Application Teamwork Desk's Custom Fields

2.2.3.3 Accelerated Feature Delivery

SaaS distributors help businesses receive new features, updates and bug fixes as soon as they are ready without the need to get new disks or install updates, saving both time and resources for the customer.[23]. As there is a single version of the software provided to the customer the distributor also saves time and resources by not needing to maintain old releases which compounds to providing the distributor with greater resources continue upgrading. The ability to provide at-will updates also allows distributors to deploy features that might not be completely bug free at launch and fix or change any aspect at any time post launch.

2.2.3.4 Licensing Model

Unlike traditional software applications, which are conventionally sold as a perpetual license with a once off up-front cost often with an optional ongoing support fee, SaaS providers typically price applications using a subscription fee model, most commonly a monthly fee or an annual fee so the users are now renting the software rather than owning it [24]. This allows for the users to budget better and removes the traditional high setup costs of acquiring the required systems or infrastructure to host the service internally.

2.2.3.5 Scalability

Since SaaS systems are cloud-based, they are able to scale easily and can be quickly integrated with other similar systems. Users are no longer required to need to buy additional software or servers — only a new SaaS subscription or alternative tier has to be enabled. This helps a business quickly and easily scale their number of users allowed in cases of employee growth or to alleviate the issue of the 'Noisy Neighbour' in cloud computing which is a term used when there is a subset of the main users that tend to monopolizes bandwidth, disk I/O, CPU and other resources, which can negatively affect other users' cloud performance [23], depending on the SaaS platform they are using. This provides huge value to a business as they are able to scale without the large upfront cost which typically would accompany upgrading internal IT infrastructure in the form of machines, services etc.

2.2.4 Adoption Drivers

The characteristics above are but a few of the most comment of the key elements that driver users to SaaS applications and below are a few of the other offerings that attract SaaS

- Improved User Experience
- Security through HTTPS standardisation
- Cost savings and budgeting through licensing model
- Accessibility via any browser
- Less management required

2.2.5 Adoption Challenges

Like for all decisions there are some compromises that need to be made when choosing SaaS as a model as a user

- Lack of control for changes that might be introduced
- Internet connection typically required
- Data security as stored via third party can be a concern
- Performance might be lower than an on-site solution

2.3 Gamification

Gamification can be defined as the implementation of game design elements in real-world contexts for non-gaming purposes[25]. The central idea is to take the building blocks of games, and to implement them, often with the goal of motivating specific behaviours, within the gamified situation that is selected. Many authors appear to see gamification as an innovative and promising concept that can be applied within a variety of contexts[26] and there have been numerous studies undertaken that was resulted in the conclusion that for the majority of users that gamifying aspects of applications did increase both use of and competency in using those applications[27][28]. One study on over 1000 participants that added a badge based reward system on an online learning tool discovered a highly significant positive effect on the quantity of participants' contributions, without a corresponding reduction in their quality, as well as on the period of time over which participants engaged with the tool. Participants noted that they enjoyed being able to earn badges, and indicated a strong preference for having them available in the user interface.[28].

Typically gamification is most frequently used as a clever way to promote a business, product or application or increase time spend using a product or application. For example, players are able to earn badges, discounts, and other rewards for visiting real-world shops and using their "checking-in" feature on the mobile phone application FourSquare. Gartner predicted that by 2014, 70% of the Global 2000 organisations would have a gamified application shown in Figure 2.8[29].

Numerous Games that are designed to promote positive lifestyle changes are starting to appear as well. Chore Wars, Habitica and EpicWin encourage players to complete daily chores, while websites like Fitocracy helps gamify fitness goals or Google Powermeter

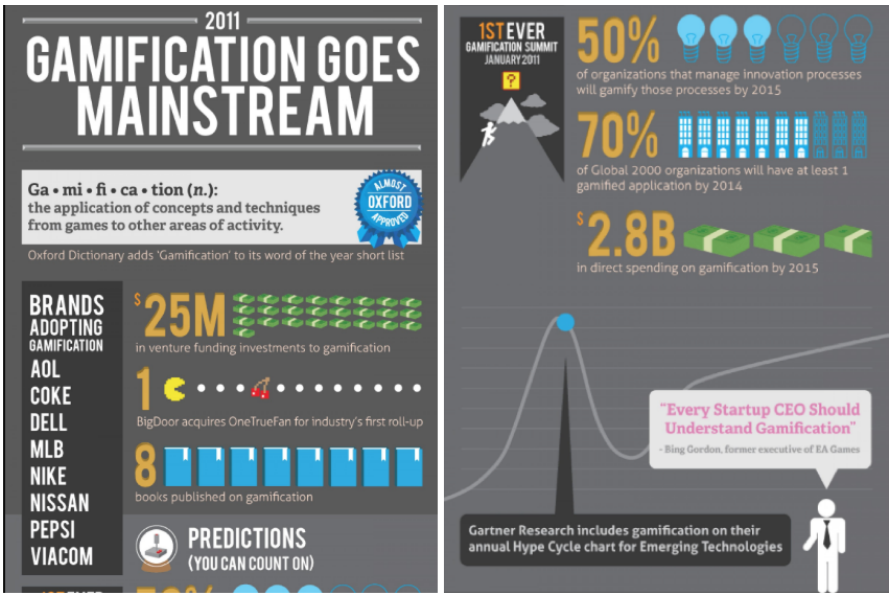


FIGURE 2.8: Gamification is becoming mainstream

which encourage household reductions in energy consumption by displaying progress bars and collectible badges.

2.3.1 Psychology of Gamification

2.3.1.1 Deterdinga and Dixon

There are many classifications that games can be broken down into by different authors, Sebastian Deterding and Dan Dixon, authors of "From Game Design Elements to Gamefulness: Defining Gamification" classify games as serious games, gamification, and playful interactions with the majority contained more than a single classification as can be seen in Figure 2.9 where it can be seen that gamification falls in the area between the playing/gaming and parts/whole and where elements over- lap [30].

2.3.1.2 B.J.Fogg

B. J. Fogg, a professor of Stanford University, states that there are three core elements that make a game successful[32]:

1. Motivation : Emotional investment
2. Ability: Difficulty to complete an action
3. Triggers: Cue to complete the action

FIGURE 2.9: Gamification is becoming mainstream [31]

Fogg also proposes that the three elements must converge at the same time in order for a behaviour to occur, see Figure 2.10. All three elements need to be present in order for the behavior to occur as if they are not all present, the player can become discouraged and potentially lose interest in the game. Small and basic behaviors can be elicited from a player with low motivation as long as the ability is present and a clear trigger exists. Complex behaviors tend to require more motivation and more ability. However, in the absence of a trigger, the desired behavior might not occur. High motivation and a good trigger coupled with low ability may result in no action being taken so it is crucial that a gamified application must offer both a worthwhile experience for the player to ensure they will want to engage with it, and offer a measure of success [32].

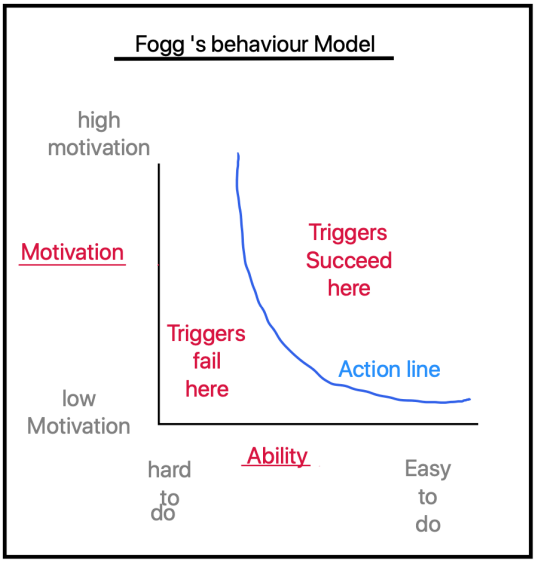


FIGURE 2.10: B. Fogg's Behavioural Model

2.3.2 Elements of Gamification

As described in the "Ten Ingredients of Great Games" by Read and Reeves [33] games can be broken into the following elements: Self-representation with avatars; narrative context; ranks, and levels; feedback; reputations, three-dimensional environments; marketplaces and economies; competition under rules that are explicit and enforced; parallel communication systems that can be easily configured; teams; time pressure.

Each of these elements can be found outside of games, and taken in isolation, none of them would be readily identified as 'gameful', let alone game-specific. Also, there is serious variation between the different game genres and digital versus non-digital games – avatars are common in action and role-playing games, but not necessarily in strategy video games or card games. In addition, how game elements are perceived can also be a matter of role, whether this be designer or user.

One example, the MDA model [26] suggests that developers work with mechanics to create aesthetics, whereas players experience aesthetics dynamically by interacting with the game and in so doing, infer knowledge about the games mechanics [30]. Mechanics are the base components of a game - they include its rules, each basic action the player can take and the algorithms and data structures in the game engine. These are expressed to the player as in game dynamics are the run-time behavior of the mechanics acting on the player input combined with the mechanics interaction with the other mechanics in place. Aesthetics refer to the emotional responses evoked in the player. These are represented in Figure 2.11.

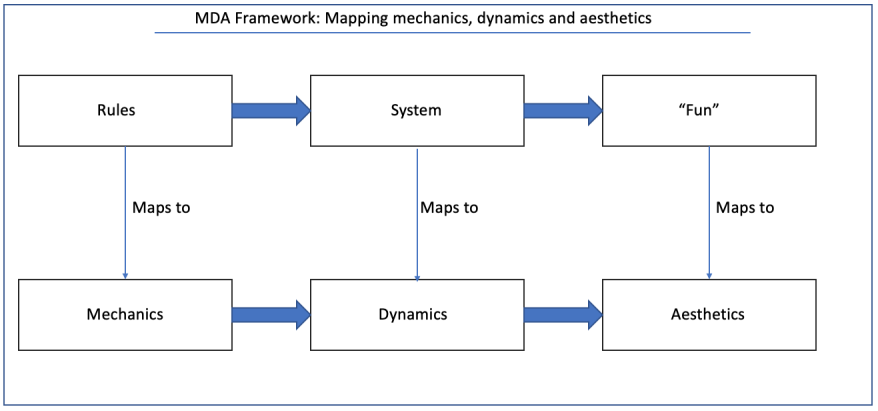


FIGURE 2.11: MDA Framework mapping

2.3.3 Design Rules

The Player journey is the conceptual path that a player follows through the game. There needs to be a beginning, middle, and an end so that the player can achieve a level of

mastery within the game. The beginning of the gamification application usually involves a process which easily transitions the player into the game, such as by incorporating a tutorial.

Secondly, the middle section should have scaffolding which allows the players to learn. Scaffolding is a teaching method that enables a student to solve a problem, or achieve a goal through a gradual removal of assistance. Elements of training are essential, allowing the player to understand what the next step that they should take.

The final step, the end, should allow the player to gain some mastery, achieve a skill or provide a reward to entice the player to move on to a higher and more difficult level. Balance is key within games in order for players to progress, and play-test the game to ensure abilities or difficulties are not too high.

It is also essential that the training provided is sufficient for the player to overcome challenges to ensure it remains fun for the player at all times. An example of balance within a game is the "Pass Go" section in a game of Monopoly where the player receives $200 to ensure the player can continue to purchase locations and properties (objective) and carry on playing the game, keeping the economy in balance within the game.

2.4 Competitor analysis

There is an almost limitless amount of resources available to students when it comes to taking notes and while there's a lot of good that can be said about the old-fashioned pen and paper method of taking notes, it comes with several downsides: paper notes are hard to search, fragile, and have no backup. Digital note taking applications go a long way towards overcoming these difficulties, while making it even quicker and easier for students to take notes. For the purpose of this analysis the focus is concentrated on comparable digital platforms that currently offer similar feature to Cornell Notes in the form of websites and applications that provide the ability for students to take notes, if they provide a system for taking notes, collaborative features, ability to use Markdown and if there are aspects of gamification to motivate the student. For services that are available both as an browser application and downloaded application, the comparison is related to the browser version.

2.4.1 Websites

2.4.1.1 Office 365 Online

Microsoft began providing its Office 365 suite of applications free to selected educational intuitions in 2012[34] and offered to all students since 2013 [35] which includes a note taking focused application OneNote but also provides Word, a document editing application for which there are Cornell Note Taking System templates available as seen in Figure 2.12

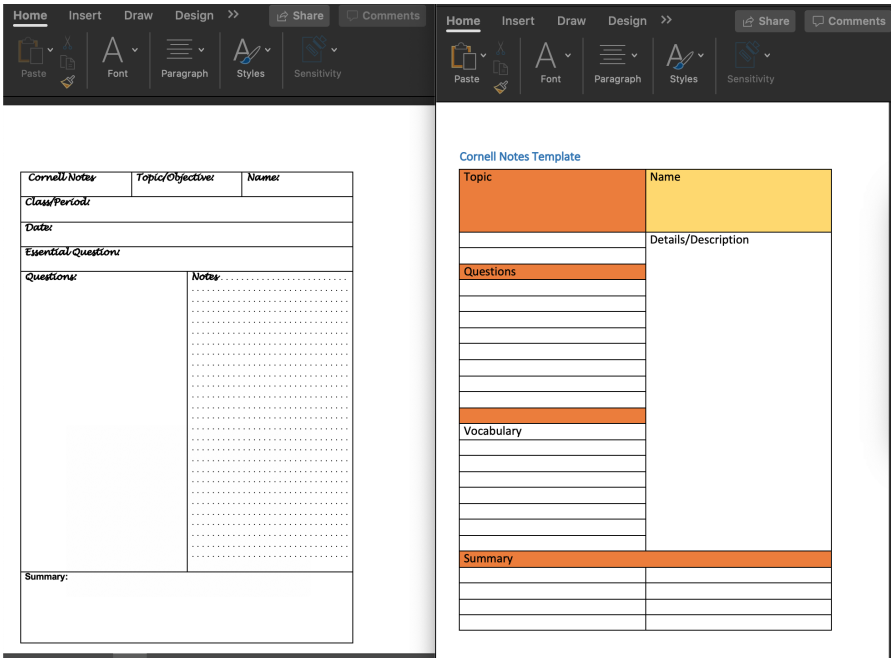


FIGURE 2.12: Word Templates for Cornell Note Taking System

Both Word and OneNote have backup ability with Microsoft's OneDrive Cloud application that comes part of Office 365 but are not able to be integrated with other popular cloud services such as Dropbox or Google Drive. Neither application provide any aspect of gamification.

Office 365 Online does offer collaborative editing using word where multiple students could edit the same document when taking notes but it does not provide a feature where students can combine notes for review.

The support for Markdown format when creating notes is possible through the installation of third party packages into the downloadable OneNote application so is not available via OneNote online nor available in Word.

2.4.1.2 Google Docs

Google Docs is a word processor that is included as part of Google's free, web-based software suite. It is compatible with documents created in other editors such as Word and it allows users to create and edit their files online while collaborating with other users in real-time.

The use and promotion of Google Docs for as a tool for students to be able to take notes individually or collaboratively has been published numerous times since its release in 2006 [36][37] due to its powerful collaboration features where like Word in Office 365 online, numerous students can collaborate to edit a singular document simultaneously such as in Figure 2.13 where both Julia's and Javier's cursors can be seen and also there is a comment section on the right where they can leave extra notes or ask each other questions

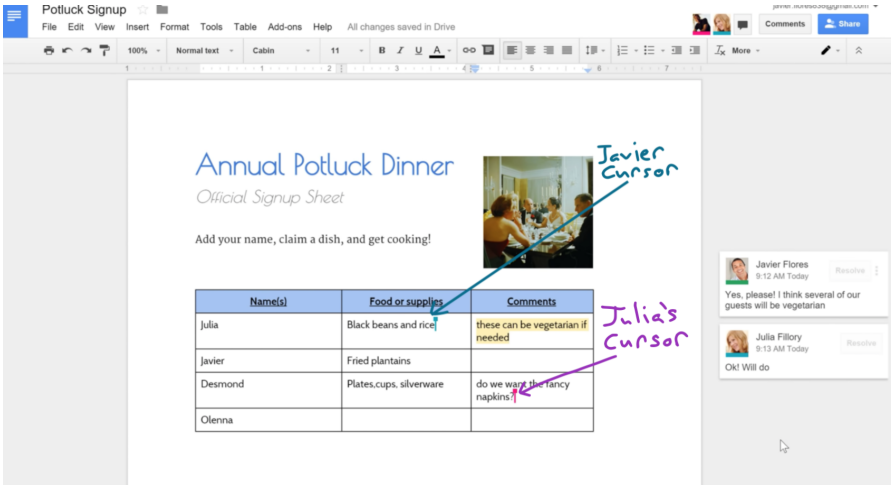


FIGURE 2.13: Google Doc's Collaborative Editing Feature

Google Docs is directly integrated with the Google Drive Cloud system for backups and there are add-ons available through Google Docs online that will facilitate the use of Markdown.

Students can also use third party templates to provide the Cornell Note Taking Systems structure to their documents like in Word but also lacks any aspects of gamification withing Google Docs to help promote use or motivate when the time comes for studying.

2.4.1.3 EverNote

Evernote is an application designed for note taking, organizing, task management, and archiving. It allows it's users to create notes, which can be text,photographs,

drawings, or saved content from websites. Notes are stored in notebooks and can be tagged,searched, edited, annotated, given attachments,backed up and exported.

Evernote is regularly featured in online comparisons of the best note taking applications for students due to its powerful note taking features and useful Tag system for searching through notes [38][39].

Evernote uses Google Drive for backing up however with third party services such as Zapier a student can utilise iCloud or OneDrive as alternative Cloud interactions for backup.

Evernote provides the ability to share notes with other students however it does not currently provide the option for two or more users to edit the same document simultaneously. If two students where to try and both edit the same shared note then two copies of the note would be created, one with all the edits from one of the users an the other would be placed in a newly created folder called 'Conflicting Changes'[40].

Native Markdown support is not offered within Evernote however it does provide its own auto-formatting shortcuts for creating lists, links, tables and more[40].

2.4.2 Applications

2.4.2.1 Ulysses

Ulysses is a Markdown-based text editor for MacOS and iOS devices released in 2010 and has undergone multiple iterations since adding to its suite of features including customisation via themes, clean distraction free interface which offers Keyboard Navigation, daily support goals and sharing capabilities [41].

Ulysses provides incentive and motivation to the user by the introduction of goals who's progress can be tracked but doesn't provide a community based goal center. Backup is supported either offline to the Macs or iOS devices file system or through iCloud for an online option.

The only features missing from Ulysses compared to Cornell Notes would be the additional of a system for note taking or template integration and the lack of collaborative editing features..

2.4.2.2 Bear

Bear has emerged as a favorite writing application among many students I've spoken to, and after using it for a few days it goes without saying why. A premium note-taking application ,and like Ulysses, is available on macOS as well as iOS.

Bear lacks many of the features of more complex note taking applications such as Ever-Note or the Office 365 suite in order to provide a sleeker and quicker experience to its users. It offers native Markdown support, a comprehensive Tag based folder structure that makes searching for notes a very quick and a clean interface as seen in Figure 2.14

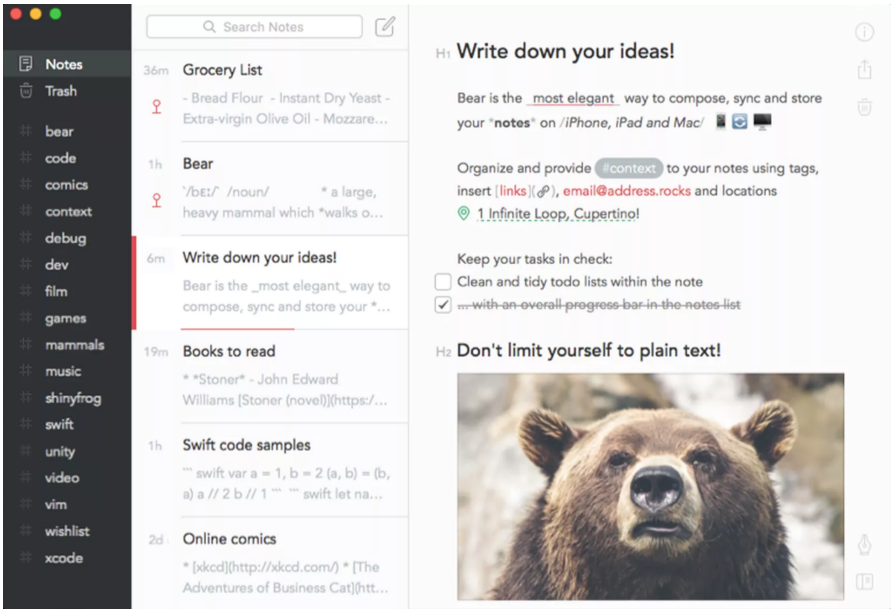


FIGURE 2.14: Bear Mac App's interface showing Tags and Markdown

Like Ulysses, backup is supported either offline to the Macs or iOS devices file system or through iCloud for an online option. Notes can be shared

Whilst the contents of notes can be exports or shared, notes themselves cannot making collaborative editing not possible when using Bear and there is no support for templates to allow for the Cornell Note Taking System to be implemented using Ulysses.

2.4.2.3 Notability

Unlike the previous two applications, Notability is only available on iOS devices and is optimised to be used with the Apple Pencil on an iPad as a physically notebook replacement which is regularly features as the Editors Choice for Note taking application on the iOS Application Store.

Notability provides powerful search features that can even search through handwriting along with wide range of note-taking and sketching tools to capture every detail during a class. Users are able to add and annotate PDFs in Notability and record and include audio into their notes, a feature that none of the other platforms analysed offered.

Aspects of gamification are omitted from Notability, as are collaborative editing but students do have the option to export or share any notes.

In order to facilitate taking notes using the Cornell Note Taking System a user would need to draw their own template which can be duplicated for new classes.

As Notability is an iOS device, backup is facilitated through the iCloud backup service but notes can be exported to other Cloud platforms including OneDrive Google Drive and Dropbox.

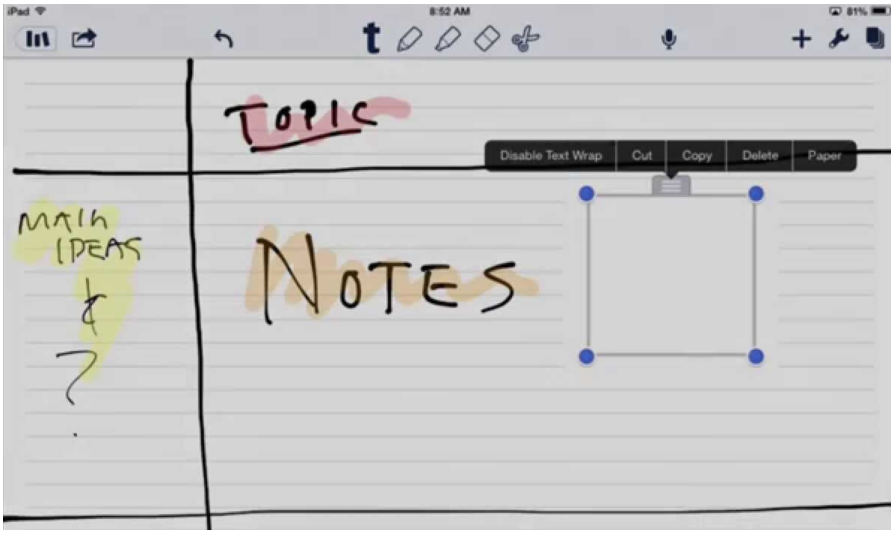


FIGURE 2.15: Notability - Drawing Cornell Note Taking System Template

2.4.3 Summary

Between all of the platforms analysed and compared to Cornell Notes almost all features within Cornell Notes are available with the exception of gamified elements to help with motivation. The closest feature within the selection of platforms used would be the trackable goal setting ability within Ulysses. In order to more clearly display the features contained by these applications and how they compare to Cornell Notes I have included a the table below in Figure 2.16 which contains the features of Cornell Notes and which of the analysed platforms contains those. I have not included the numerous features available within each of these platforms that is not present within Cornell Notes.

Platform	Tags/Labels Support	Collaborative Editing	Cornell System Support	Markdown Support	Gamification Elements	Dedicated Review Support	Backup Support
Cornell Notes	✓	✓	✓	✓	✓	✓	✓
OneNote	✗	✗	✓ *	✗	✗	✗	✓
Word	✗	✓	✓ *	✓	✗	✗	✓
Google Docs	✗	✓	✓ *	✗	✗	✗	✓
EverNote	✓	✗	✓ *	✗	✗	✗	✓
Ulysses	✓	✗	✗	✓	✗	✗	✓
Bear	✓	✗	✗	✓	✗	✗	✓
Notability	✗	✗	✓ *	✗	✗	✗	✓

✓ * = Available using third part support

FIGURE 2.16: Competitor Analysis Feature Comparison Table

Chapter 3

Cornell Notes: A Student Focused Note Taking SaaS Application

Taking notes is an important aspect within classroom learning, and students who take more lecture notes in general are higher achievers. Traditional note taking methods and current mainstream note taking applications lack gamification elements and integrated note taking systems. Numerous studies have concluded that using note taking systems and gamifying learning elements contribute greatly to a students ability to retain information[13], [28]. There are no student focused applications available to students that support note taking systems and included gamification elements to promote engagement, that also support the core services of mainstream note taking applications such as collaboration though sharing, PDF support, and are available on a variety of devices and operating systems. Developing a platform that supports students note taking by utilising a world renowned note taking system which contains gamification elements to further promote engagement could provide a beneficial tool to students to both increase their engagement during a lecture and promote revision through the Cornell Note Taking system and the gamification elements that promote learning through collaboration and competition.

3.1 Problem Definition

Improve a students experience of note taking by providing a platform that includes traditional note taking services and also contains gamification elements to promote collaboration and revision through a tested note taking system, The Cornell Method, that

will be available on a large range of devices as a SaaS application.

3.2 Objectives

Introducing a new note taking system that will continue to be used by a student can be challenging requires the setting of some clear, achievable goals. These objectives will help provide both direction and motivation to implement the solution by helping to identify appropriate actions to be undertaken to succeed and enable tracking of progress.

1. Improve engagement by providing fast and simple interactivity with the application
2. Improve notes taken through a tested note taking system.
3. Encourage use and social engagement through collaboration and gamification elements
4. Provide existing note taking services such as cloud backup, standard note taking, folders, authentication.
5. Provide popular new services for taking notes like markdown

3.3 Functional Requirements

The functional requirements below include functions performed by specific screens within Cornell Notes and outlines work-flows performed by the system.

3.3.1 User Accounts

A user can create a new account or log in using their Google Cloud account. An email will be sent to a user when they first login and use Cornell Notes. The login screen will provide an option to reset a users password by sending an email to their account with details on how to reset it. A user can change their in application user name within the application.

3.3.2 Study Groups

A user can create a study group and invite other users to the study group using an email address. The name of the study group can be changed along with the color associated with it. Users can leave a study group or delete it.

3.3.3 Traditional Notes

The system will provide traditional note taking services through a text box that offers traditional word processing tools and also supports markdown. Notes can be stored in a folder structure

3.3.4 PDF Support

The system will support uploading of PDFs that can be displayed along side the note taking section.

3.3.5 Cornell Note Taking System

The Cornell Note taking system is supported through a custom interface that will use linked text boxes for Cues and related Answer information, these will be individually indexed and support being tagged allowing to be utilised in the gamification section for revision. An optional summary section can also be added to each of the Cornell Notes

3.3.6 Collaboration

Users within study groups can share notes between users that can be altered and expanded upon or used for revision through the review feature.

3.3.7 Gamification

Users can compete during revision by taking quizzes constructed by displaying randomised Cues from their study groups or individually from their own notes which will allow them to select a number of questions to receive and include a note name or tag name to select which notes. The quiz will display the number of correct and incorrect answers upon completion that can be seen in the application.

These reviews can be shared among other users creating a quiz based competition between users who can earn various badges as a result of achieving various goals to be outlined later in chapter 4.

3.3.8 Interface

The interface allows the user to view all their folders with notes, access their account settings, access the revision and study groups sections. Users cannot see notes of other users.

3.4 Non-Functional Requirements

UI : The user interface will be easy to navigate by the user. The UI should be portable between devices of different screen size and operating systems. Icons on the website should clearly indicate what function they will perform when selected.

Performance : Numerous users should be able to use the platform simultaneously without degrading performance.

3.5 Users

There are two main user types within Cornell Notes, students that want to use the application independently and those that will use its collaboration features within a group. Both will have many of the same objects and goals with a few differences that required other functionality to carry out their tasks. The two personals outlined below were created with both these roles in mind to help identify a selection of use cases.

3.5.1 Persona 1

Name & Sinead
Age & 20
College & Cork Institute Of Technology
Course & Mechanical Engineering
Role & Using collaboration features

3.5.1.1 Goals

- Sinead wants to be able to take notes with her friends
- Sinead is very competitive and wants to unlock as many badges as she can, quicker than her friends.

3.5.1.2 Use Cases

- As a user, I want to be able to sharing my notes with my classmates
- As a user, I want to be able to create effective notes that can immediately be used for revision.
- As a user, I want to use a system that rewards me for creating notes and reviewing them.

3.5.2 Persona 2

Name & Frank
Age & 22
College & Cork Institute Of Technology
Course & Analytical Chemistry and Instrumentation
Role & Using platform independently

3.5.2.1 Goals

- Frank wants to try out a new note taking system to see if it can help him make better notes.
- Frank currently takes notes using pen and paper but recently bought a laptop to start typing so that he can better backup and navigate through his notes.
- Frank is interested in the review aspect to be able to easily build revision material.

3.5.2.2 Use Cases

- As a user, I want to be able to use markup and a note taking system to take better notes.
- As a user, I want to be able to easily access, view and edit all my notes.
- As a user, I want to be able to tag these notes to quickly retrieves them for revision.
- As a user, I want to be able to create useful reviews based on these notes for revision.

Chapter 4

Implementation Approach

Identifying the best available and practical technologies to implement the design and functionality of the Cornell Notes platform enabled the projects requirements to be met. Understanding the requirements for the selected technologies would each fit into the overall architecture and function as a cohesive unit meant researching various technologies.

This chapter will outline the four core units that will comprise Cornell Notes micro service architecture as seen in Figure 4.1along with each of the main components that will need to be implemented, the risks associate with the choice of technologies used and others that could arise during the implementation phase and the methodology that will be used to achieve the goal of creating the platform passes the requirements outlined during Chapter 3

4.1 Architecture

To create the Cornell Notes Platform there are four core units that construct the architecture including:

1. Front End
2. Back End
3. Database
4. Deployment Service

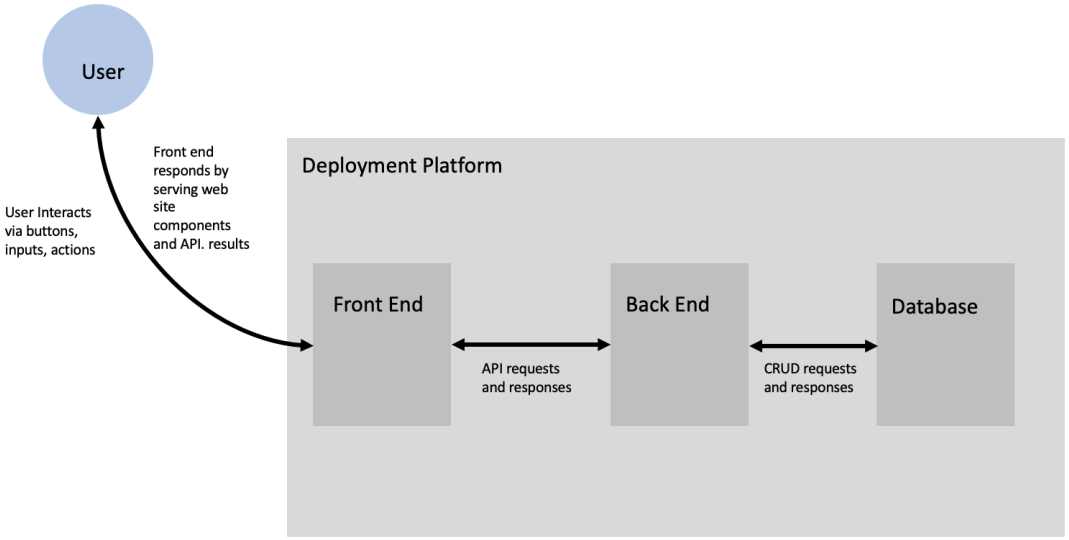


FIGURE 4.1: Architecture Component Overview

Separate front end and back end units for this project were selected for the following reasons:

Modularity : Modularity is the primary reason for selecting a separate front end and back end for this project as it provides massive benefits during development as it can be divided into smaller sections which helps create smaller code sections to assist with readability and error detection during both development and testing.

Re-usability : Whilst the scope outlined in this project for the user interactive section of the platform to be through a web browser creating a back end unit that serves API requests for Cornell Notes allows for future extension such as by developing an Android or iOS based application that then would only requirement front end components to be developed.

Responsiveness : Utilising a separate front end to serve and handle all the user interactive options for the UI allows for the back end to only send raw data, through JSON, which improves efficiency on the back end but also on the front end as the whole page is not being updated with every request but instead sections within components on each page.

4.2 Front End

A user of the Cornell Notes platform will be able to communicate with the platforms back end through a web based front end in order to create notes and reviews and completed

individual and group based reviews in order to earn badges that will be presented in an Trophy section on the platform for the user to see.

4.2.1 Selected Technology

The technology selected to be used in the development of the front end is Vue.JS which is a progressive JavaScript framework designed for building client side view for applications and provided the following benefits:

Quick to Adopt : As Vue.JS has a component library that is based on HTML, CSS and JavaScript it does not have a large learning curve that would need to be overcame in order to develop a great looking and functioning front end environment for Cornell Notes.

Vue.JS has a great two-way binding system by design allowing for data to be altered either from the JavaScript or the template which I found much easier to understand when researching other front end frameworks that does not compromise on efficiency as outlined below.

Comprehensive documentation : The Vue documentation is freely available online, extremely detailed, relatively easy to understand, and includes a variety of code examples to assist developers. It also has a good, growing community of developers that can offer help when needed which is ideal when selecting a framework with which I have no previous experience in using.

There are also a collection of official Vue.JS libraries that are outlined on their website that provide functionality such as routing, deployment, API and data management some of which will be directly implemented in this project.

Speed : Vue.JS utilises a virtual DOM in order to improve efficiency by generating a version of the DOM in-memory each time a user changes a state and compare it to the actual DOM, so that the front end then can update only the part that needs to be updated instead of re-rendering everything on the page to improve load times.

4.2.2 Main Components

Cornell Notes front end will consist of the following main pages:

4.2.2.1 Homepage - Login/Register

The default page when the user navigates to Cornell Note will present a page that will outline the various features of the application through a auto scrolling gallery on the left and contain a section to allow the user to either create a new account to get started using the platform or to log into an existing account

Features

- About Cornell Notes Section
  - Gallery of Images showing platform with overlapping text information
- Register/Login
  - Create new account button to open modal
  - Input for user to log into existing account

4.2.2.2 Main Container

The platforms main content container is used to display the Cornell Notes Component, the PDF Viewer and Reviews. It will provide options to split the container between the Cornell Notes Component and the PDF viewer allowing users to utilise both when taking notes.

4.2.2.3 Cornell Notes Component

This component will mimic the Cornell Note taking template in digital form allowing the user to type in auto colored Cues and fill in answers using standard text and markdown along with adding a optional summary section at the end of each note.

This component will also provide the option to hide the font editing menu and also the answer section to allow for quick revision and more focused note taking.

Features

Main Div composed of three sections which are arranged following the Cornell Note taking templates format of the Cues on the left hand side, the answer section on the right and beneath both is the optional summary section

1. Cues : User can enter Cues for material such as questions or definition words
2. Answers : User can enter answers for the cues in text form and import images such as PNG/JPEG
3. Summary : Hidden section by default with option to add using button at top of main component

4.2.2.4 Creating Review Modal

A user will have the option of creating a Review in the application and this component will assist the user in tailoring the questions they want to include in their review from existing notes. It will provide a drag and drop interface which also supports check boxes that the user utilise to move either individual questions or entire lists of questions from their Notes into a Quiz-like review that can be completed to earn points that accumulate to unlock various achievement badges in the application

Features

- Input to add a name for the review
- Options to add tags for the review
- Share option to send review to friends
- Drag-able area that displays list of Notes by Folder, Notes have the option to be expanded to show individual Cues within them
- Preview area where Cues/ Notes can be dragged into to display the list which will compose the review
- A create button to finalise the review that was built in the modal

4.2.2.5 Completing Review Component

Reviews share the same template design as a Cornell Note consisting of both the Cue and Answer sections that by default will be displayed with the Cues showing one at a time with an options to skip or view the answer Points can be accumulated by the user for getting correct answers and completing reviews.

Features

- Main Section shows Cue
- List of buttons for next, skip, show answer along with option
- Completing review shows how many you skipped and the questions you needed to display the answers for to determine points and inform user of areas to focus on

4.2.2.6 Badges Modal

As a major part of the Cornell Notes platform is the addition of gamification to promote and assist with reviewing material there is a dedicated section where the user can view their accumulated points in the application along with any individual and group badges they have accumulated during their time using the platform.

Features

- Current Points gauge displaying how many until next major badge
- List of badges, locked badges are hidden, earned badges are displayed and include a short summary informing how they were earned
-

4.2.2.7 Header

One of the two main navigational components in the application the header will provide a button to hide or display the second main navigational component being the sidebar.
Features

- Cornell Notes Icon
- Hide/display button for sidebar
- Search bar to search through notes
- Badges Icon to navigate to Badges page previously mentioned
- My Account button to open the users Account modal

4.2.2.8 Sidebar

The second of the main navigational components which provides the user an option of selecting from their collection of notes, PDFs and reviews which will then be displayed in the main page. to the user

Features

- Nested folder lists containing Notes and associated PDFs
- Nested reviews group by topic/type
- List of tags to quickly search

4.2.3 Other Components

Along with the above components required to facilitate the core functionality of the application the following components are also required in order to ensure the platform contains features users may have become accustomed to using in other applications such as:

4.2.3.1 My Account Modal

The My Account modal allows the user to edit information about their account including:

- Name
- Email
- Password
- School
- Course
- Avatar

4.2.3.2 Font Editing Toolbar

One of the goals I wanted to achieving during the design and implementation of this project is to allow for users to use the Markdown framework to be able to style their notes but to prevent limiting the users who might adopt Cornell Notes as a study tool to those that are used to markdown I have chosen to include a traditional Front editing toolbar that will facilitate customising the following features related to the font used including:

- Font Type
- Font Size
- Font Color
- Bold / Italic / Underline
- Heading 1 - 3

4.2.3.3 PDF Viewer Component

The PDF viewer provides the options for users to view PDF documents in the platforms Main Container. Users can then utilise functions such as copy and paste to quickly transfer information from the PDF document into the Cornell Notes Component for their notes.

4.2.3.4 Alert and confirmation Modals

A selection of confirmation and alert modals will be implemented to support the platforms previously referenced features.

4.2.4 Imported Libraries

Vue.JS has a wide range of natively supported libraries that will be taken advantage of to enhance both the functionality and UI design of the platform while also reducing the learning curve related to using a newer language. Some of the main libraries that will be utilised in the implementation of the front end are:

- Vuetify

  - A component framework that aims to provide reusable, semantic and clean components while supporting all the modern browsers and is compatible with Vue CLI-3

- Vuex
  - A state management pattern and library that serves as a centralized store for all the components in an application providing rules ensuring that the state can only be mutated in a predictable fashion which will make it easier to manage the different states of components in the application.

- Vue-CLI
  - A command-line utility that supports and provides a range of build tools, which it will then install and configure.

4.3 Back End

The purpose of the platforms back end is to serve API requests from the Front End and communicates with the Database using CRUD requests in order to facilitate the transfer of data related to the users need to the user during their use of the Cornell Notes platform.

4.3.1 Selected Technology

The technology selected to be used in the development of the back end is Golang, a statically typed, compiled programming language designed at Google which was chosen for the following reasons.

Efficiency : A major benefit of utilising Golang as the back end in a micro service architecture is due to the speed as which it runs to help mitigate the delay caused by the API response time between the front and back end services. Golang has a small syntax and concurrency model which makes it a really fast programming language that compiles to machine code and its compilation process is very fast. Golang also eliminates a dependency on servers by linking all the required dependency libraries into a single binary file.

Simplicity : Golang is, in my opinion, a beautiful looking programming that offers a clean, readable typeset that greatly assists when reviewing code. Golang also offers simple approaches to many important and often complicated units in a language including:

- Concurrency is a first-class language feature.
- There is no need to explicitly define class hierarchies, as interfaces are implicitly satisfied
- Golang supports a reduced language keyword set, for example there's only one loop construct being the For loop
- No Project files
- Supports built in garbage collection

Personal Interest : During my work placement in third year I was exposed to the Golang programming language while working as a Technical Intern on the back end development team at Teamwork and one of the reasons for selecting Golang as the back end for this project is to help improvement my knowledge of this language.

4.3.2 APIs

Cornell Notes will utilise APIs for the transfer of information between the front end and back end services and the Golang Gorilla Mux Web Routing framework will be employed to assist with performing more efficient API routing than the native net/http router that Golang provides.

There are a total of eight different entities that will need to be supported by the back end through multiple APIs that are outlined below.

1. Users
2. Notes
3. Cornell Notes
4. Folders
5. Tags
6. Reviews
7. PDFs
8. Badges

4.3.2.1 Users

The user entity will contain data related to the currently logged in user including their name, password, school and course information. The following APIs shown in Figure 4.2 will support the requests for creating, retrieving and editing user data between the front end and back end.

Type	Name	Reason	Input	Return
GET	GetUser	Retrieve single user	userID, email, password,	User
GET	GetUsers	Retrieve list of users	ListUsers	List of users
POST	RegisterUser	Create new user account	Name, password, email,dateTime	Status Code
PUT	EditUser	Edit user account details	Name, course, college, year, dateTime, user_ID	Status Code

FIGURE 4.2: APIs to support Users

4.3.2.2 Groups

In order to facilitate the sharing of note and reviews among different users which is a core part of the collaboration goals within the platform, the ability to create groups is required and Figure 4.3 displays a selection of the APIs that will be developed to support these features.

Type	Name	Reason	Input	Return
GET	GetUserGroups	Gets a list of groups that a user is part of	User_ID	listofGroups
POST	CreateGroup	Create a new group	groupName, user_ID	Status Code
PUT	EditGroupName	Change name of group	Group_ID, newName,	Status Code
PUT	AddGroupMember	Add new user to group	User_ID, group_ID	Status Code
DELETE	DeleteGroup	Deletes Group	Group_ID, User_ID	Status Code

FIGURE 4.3: APIs to support Groups

4.3.2.3 Notes

As well as supporting Cornell Notes the user will also have the option of adding traditional notes within the platform which are supported by the following APIs seen in Figure 4.4

Type	Name	Reason	Input	Return
GET	GetNotes	GetsUsersNotes	User_ID, tags	Notes
POST	CreateNote	Create new note	User_ID, Type, Tag	Note_ID
PUT	EditNoteBody	Edit note body	User_ID, body, note_ID, dateTime	Status Code
PUT	EditNoteName	Edit name of the note	User_ID, name, note_ID, dateTime	Status Code
PUT	AddNoteToFolder	Add note to a folder	Note_ID, Folder_ID, dateTime	Status Code
PUT	EditNoteFolder	Change folder of note	Note_ID, Old_Folder_ID, dateTime, new_Folder_ID, dateTime	Status Code
PUT	EditNoteTag	Change tag(s) attached to note	Note_ID, dateTime, Tags	Status Code
DELETE	DeleteNote	Delete single note	Note_ID, user_ID, dateTime	Status Code
DELETE	DeleteNotes	Delete multiple notes	Note_IDs, user_ID, dateTime	Status Code

FIGURE 4.4: APIs to support Notes

4.3.2.4 Cornell Notes

The most important entity in the application is the support of the notes that will contain all the information related to cornell notes from the list of cues and their associated answers and tags to the overall summary and tags tied to the entity. This requires the most APIs due to the complexity of the entity as displayed in Figure 4.5

4.3.2.5 Folders

Cornell Notes will provide a nested folder structure which can store Notes, Cornell Notes, PDFs and Reviews and the APIs shown in Figure 4.6 will be used to create and modify folders within the application.

4.3.2.6 Tags

Tags are an optional choice that can be applied to Notes, Cornell Notes, PDFs and reviews to assist the users in searching for them using the Search bar that will appear in the header of Cornell Notes when the user is logged in. Figure 4.7 outlines some of the APIs that will be used to facilitate this.

Type	Name	Reason	Input	Return
GET	GetCornellNotes	Gets all Users Cornell Notes with option to filter by tag	User_ID, tags	Notes
GET	GetCornellNotesByFolderID	Gets all Users Cornell Notes per folder	User_ID, Folder_Id	Notes
POST	CreateCornellNote	Create new note	User_ID, cornell_note_ID, Tag, dateTime	Status Code
POST	CreateCornellCue	Create new Cue which also creates the associated answer	User_ID, cornell_note_ID, Tag, dateTime	Status Code
POST	CreateCornellSummary	Create new note	User_ID, cornell_note_ID, Tag, dateTime	Status Code
PUT	EditCornellNoteCue	Edit name of the note	User_ID, name, cornell_note_ID, dateTime	Status Code
PUT	EditCornellNoteAnswer	Edit Answer in the note	User_ID, name, cornell_note_ID, , Cue_id dateTime	Status Code
PUT	EditCornellNoteName	Edit note body	User_ID, body, cornell_note_ID, dateTime	Status Code
PUT	EditCornellNoteSummary	Edit name of the note	User_ID, name, cornell_note_ID, dateTime	Status Code
PUT	AddCornellNoteToFolder	Add note to a folder	cornell_Note_ID, Folder_ID, dateTime	Status Code
PUT	EditCornellNoteFolder	Change folder of note	cornell_Note_ID, Old_Folder_ID, dateTime, new_Folder_ID, dateTime	Status Code
PUT	EditCornellNoteTag	Change tag(s) attached to note	cornell_Note_ID, dateTime, Tags	Status Code
DELETE	DeleteCornellNote	Delete single note	cornell_Note_ID, user_ID, dateTime	Status Code
DELETE	DeleteCornellNotes	Delete multiple notes	Note_IDs, user_ID, dateTime	Status Code

FIGURE 4.5: APIs to support Cornell

4.3.2.7 Reviews

Reviews can be created, edited, shared and edited by a user. The below APIs in Figure 4.8 represent a list of some of the functions that will be implemented to allow for users to create a review from a list of Cornell Note Cues and their respective answers.

Type	Name	Reason	Input	Return
GET	GetFolderNotes	Gets all notes in a folder	Folder_ID	List notes
GET	GetFolderParentFolder	Gets Folders Parent Folder	Folder_ID	Folder_ID
GET	GetFolderPChildFolder	Gets Folders child Folder	Folder_ID	Folder_ID
PUT	EditFolderParentFolder	Change Folders Parent Folder	Folder_ID, dateTime	Status Code
PUT	EditFolderPChildFolder	Change Folders child Folder	Folder_ID, dateTime	Status Code
DELETE	DeleteFolder	Delete a folder	Folder_ID, dateTime	Status Code

FIGURE 4.6: APIs to support Folders

Type	Name	Reason	Input	Return
GET	GetTags	Gets all users tags	User_Id	Listof tags
GET	GetTagNotes	Gets all notes per tag		
POST	CreateTag	Create new tag	Name	Tag_Id
PUT	EditTag	Edit tag name/color/ note_ID	Tag_ID, name, color, note_ID, dateTime	Status Code
PUT	AddTagToNote	Adds tag to note	Tag_ID, note_ID, dateTime	Status Code
PUT	Remote tag from note	Removes tag from note	Tag_ID, note_ID, dateTime	Status Code
DELETE	DeleteTag	Deletse tag	Tag_ID, user_ID, dateTime	Status Code

FIGURE 4.7: APIs to support Tags

4.3.2.8 PDFs

To enable the PDF viewer component to function correctly the following APIs in Figure 4.9 will be created.

4.3.2.9 Badges

The Badge features of Cornell Notes will not only require APIs for support but will also require additional functions that will be run intermittently to determine if a user is due to be awarded a new badge. To facilitate this the user will earn points for creating new Cornell Notes or reviews, adding Cues, sharing and completing reviews to help promote user engagement.

An outline for the points awards for creating each of these elements can be seen in Figure 4.10

Users are also able to earn bonus points for achieving certain requirements such as creating or sharing a specific number of notes or reviews as referenced in Figure 4.11

Type	Name	Reason	Input	Return
GET	GetReview	Returns review details	Review_ID	Name, users, highest_score
GET	GetReviewHighestScore	Returns user with highest score on review	Review_ID	highest_score
POST	CreateReview	Creates new section for reviewing notes	User_ID, dateTime	Status Code
PUT	AddReviewNote	Adds note to a review section	Review_ID, user_ID, Note_ID, dateTime	Status Code
PUT	RemoveReviewNote	Removes note from a review	Review_ID, user_ID, Note_ID, dateTime	Status Code
PUT	AddReviewUser	Adds user to a review section	Review_ID, user_ID, Note_ID, dateTime	Status Code
PUT	RemoveReviewUser	Remove users from review section	Review_ID, user_ID, Note_ID, dateTime	Status Code
POST	AddReviewScore	Add a score for a user to a review sectoin	Review_ID, user_ID, score_ID, dateTime	Status Code
PUT	EditReviewScore	Edits score for a user on a review	Review_ID, user_ID, score_ID, dateTime	Status Code
Delete	DeleteReview	Delete review section	eview_ID, user_ID, dateTime	Status Code

FIGURE 4.8: APIs to support Reviews

Type	Name	Reason	Input	Return
GET	GetPDF	Gets PDF in folder	folderID	PDF
POST	AddPDF	Adds a new pdf	PDF	Status Code
PUT	MovePDF	Move PDF between folders	PDF_ID, folder_ID, dateTime	Status Code
DELETE	DeletePDF	Deletes a PDF	PDF_ID, dateTime	Status Code

FIGURE 4.9: APIs to support PDFs

The accumulation of these points will assist in unlocking a variety of badges that are divided by either a score limit or for special instances such as for creating first Cornell Note or review or sharing them. Figure 4.12 displays the current badges that will be implemented for users to unlock.

How	Points
Create Note	25
Create Cornell Note	25
Create Review	25
Sharing	50

FIGURE 4.10: Points earned by creating items

Name	Requirement	Bonus Points
Cornell Notes		
	Create first	1000
	Create 10	100
	Create 100	200
	Create 100	400
	Create 500	1500
	Share first	250
	Share 10	400
	Share 50	500
Reviews		
	Create first	1000
	Create 10	100
	Create 100	200
	Create 100	400
	Create 500	1500
	Share first	250
	Share 10	400
	Share 50	500
	Completing First review	250

FIGURE 4.11: Points earned by sharing items

Badge	Unlock requirement
Getting started	Create first Cornell Note
Team Player	Sharing your first Note
Team Leader	Share 20 Notes
Novice	Earn 2000 Points
Padawan	Earn 1500 Points
Look out Brian Cox!	Earn 2500 Points
Taking things seriously	Earn 5000 points
Teacher better watch out!	Earn 1000 Points

FIGURE 4.12: Badges that can be earned

4.3.3 Database

A separate database will also be implemented as part of the micro service architecture to store all the information required by the platform during its operation. In the below section the Database type and reasons for selection are outlined along with a draft of the expected tables and their relationships.

4.3.4 Selected technology

The database type of choice for the project is going to be RethinkDB which is a open-source, scalable JSON database built for real time web activities. It offers the functionality to continuously push updated query results to applications in real time which dramatically reduces the time and effort necessary to both build and serve requests. The following reasons assisted with my selection of this database type:

Open Source Community : An important factor in selecting RethinkDB as the database choice to store Cornell Notes user and platform information is due to it being an open source project with a large developer community that I have found helpful when interacting with in the past.

JSON Format First : RethinkDb offers a JSON format first architecture where it stores its data in JSON format which, due to the architecture approach take to use JSON based APIs to send information between the front end and back end will help improve response times due to the removal of requirement to reformat the data between requests.

Push over Poll : Another efficiency benefit that Rethink offers alongside supporting JSON first is the ability to configure the database to push updated query results rather than waiting for them to be polled and requested which will be useful when it comes to the collaborative feature of sharing Notes and Cornell Notes where if one user adds a new Cue, rather than other users needing to request those changes, the changes will be pushed from the database through the back end and UI for the user.

4.3.5 Database Tables

The Figure 4.19 was created from the list of relationship requirements below to assist in determining the most appropriate structure of the database tables and their relationships to each other to be implemented

The following tables in the below Figures from Figure 4.13 to Figure 4.18 which outline the Table Name, its elements and their respective types and keys where:

- PK denotes Primary Keys
- PF denotes Foreign Keys
- CK denotes compound Keys

Table Name	Users	
Field	**Value**	**Key**
User_ID	UUID	PK
FirstName	String	
LastName	String	
Password	HashedString	
College	String	
Course	String	
Profile_Pic	BinaryObject	
Date	DateTime	

Table Name	Folders	
Field	**Value**	**Key**
Folder_ID	UUID	PK
Item_ID	UUID	FK
Item_Type	String	
Name	String	
Date	DateTime	

FIGURE 4.13: Database Table for Users and Folders

Table Name	Note	
Field	**Value**	**Key**
Note_ID	UUID	PK
Name	String	
Body	String	
Folder_ID		FK
Date	DateTime	

Table Name	Note Users	
Field	**Value**	**Key**
Note_ID	UUID	CK
User_ID	UUID	CK
Date	DateTime	

FIGURE 4.14: Database Table related to Notes

Table Name	Groups	
Field	**Value**	**Key**
GroupID	UUID	PK
Name	String	
Date	DateTime	

Table Name	Group Users	
Field	**Value**	**Key**
GroupID	UUID	CK
User_ID	UUID	CK
Date	DateTime	

FIGURE 4.15: Database Table related to Groups

Using the above tables along with the following relationship requirements Figure 4.19 was designed to outline the database relationships as a diagram.

Table Name	Review Users	
Field	**Value**	**Key**
Review_ID	UUID	CK
User_ID	UUID	CK

Table Name	Review Notes	
Field	**Value**	**Key**
Review_ID	UUID	CK
Note_ID	UUID	CK
Date	DateTime	

Table Name	Review Scores	
Field	**Value**	**Key**
Review_ID	UUID	CK
User_ID	UUID	CK
Score_ID	UUIS	CK
Date	DateTime	

FIGURE 4.16: Database Table related to Reviews

Table Name	PDFs	
Field	**Value**	**Key**
PDF_ID	UUID	PK
Name	String	
PDFBody	BinaryObject	
Folder	Folder_ID	FK
Date	DateTime	

Table Name	Tags	
Field	**Value**	**Key**
Tag_ID	UUID	PK
Item_ID	UUID	FK
Item_Type	String	
Date	DateTime	

FIGURE 4.17: Database Table related to PDFs and Tags

Table Name	Badges Users	
Field	**Value**	**Key**
Badge_ID	UUID	CK
User_ID	UUID	CK
Date	DateTime	

Table Name	Badges	
Field	**Value**	**Key**
Badge_ID	UUID	PK
Score	Int	
Icon	picture	

FIGURE 4.18: Database Table related to Badges

- Users can have zero or more folders
- Users can have zero or more badge users
- Users can have zero or more review scores
- Users can have zero or more groups users
- Users can have zero or more note users
- Users can have zero or more Cornell note users
- Notes users can have one or more Notes
- Cornell Notes users can have one or more Cornell Notes
- Cornell Note can have zero or more Cornell Note Cues
- Cornell Note Cues can have one Cornell Note
- Notes can have one or more note users

- Notes can have zero or more tags
- Groups can have one or more users
- PDFs can have one folder
- PDFs can have zero or more tags
- Badge user can have one or more badges
- Badge users can have one or more badges
- Badges can have one or more badge users
- Reviews can have one Group
- Reviews can have one review user
- Reviews can have one or more tags
- Reviews can have one folder
- Reviews can have zero or more review scores
- Reviews can have zero or more tags
- Tags can have zero or more Notes
- Tags can have zero or more Cornell Notes
- Tags can have zero or more Reviews
- Folders can have zero or more Reviews
- Folders can have zero or more notes
- Folders can have zero or more Cornell Notes
- Folders can have zero or more PDFs
- Folders can have zero or more Folders

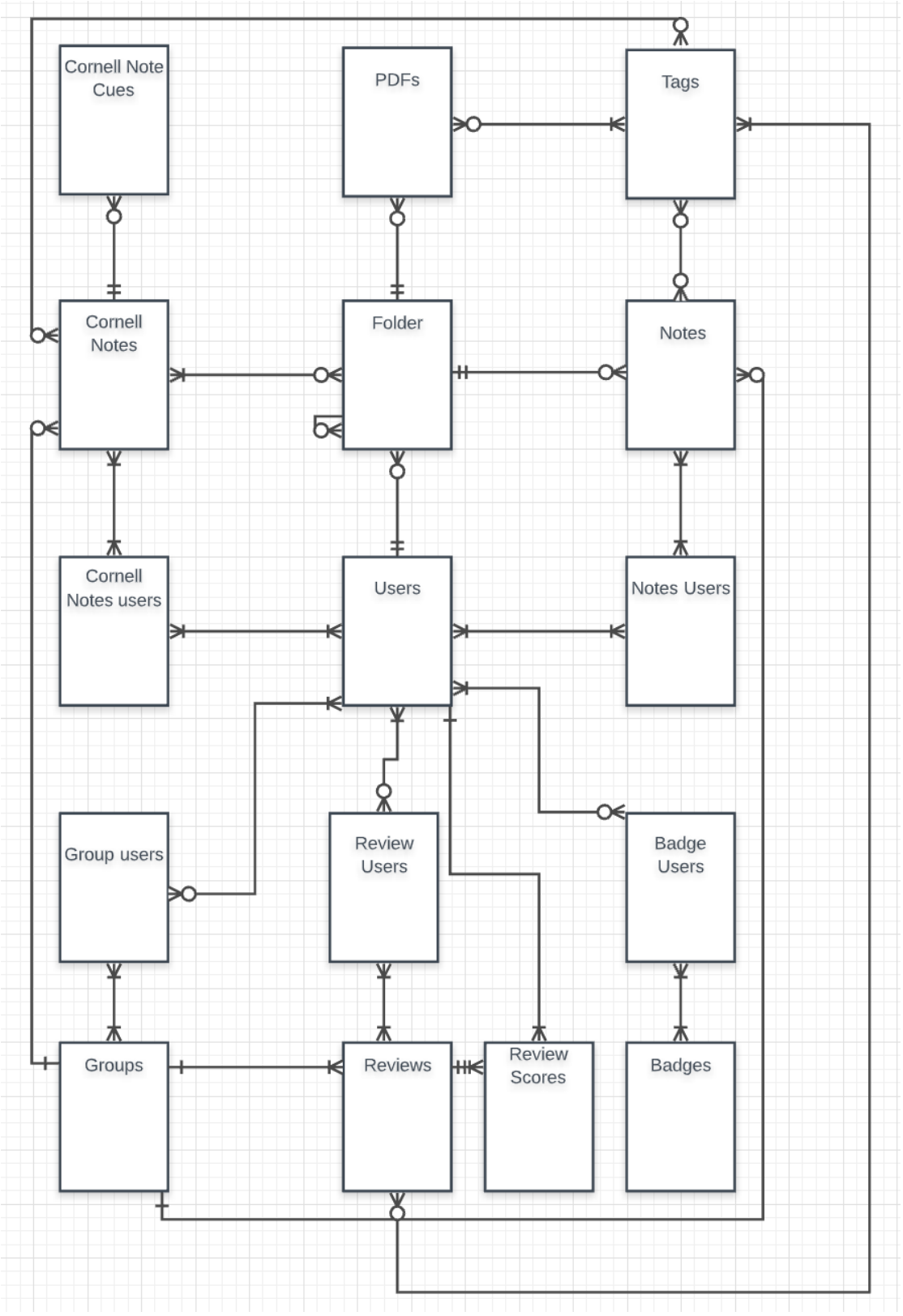


FIGURE 4.19: Database Tables relationship diagram

4.3.6 Deployment Service

To ensure that Cornell Notes is available to users using any web based browser hosting it on an external platform is required and researched platforms to provide these facilities include AWS and Digital Ocean. The viability of hosting the platform from a local server should any challenges arise in relation to utilising a deployment service was also researched.

4.4 Risk Assessment

Undertaking any large project involves a number of risks with many of the standard risks associated with software development being increased due to the lack of prior experience completing such a project that I have. I have identified what I believe will be the two main areas of risk associated with this project which are related to personal skills and challenges and risks that could occur directly during development along with any mitigation's that can be utilised to help reduce risk in each of these areas

4.4.1 Personal Skills

The biggest areas of risk associated with this project are related to my personal skills including programming language knowledge, time management and project execution. For this project I have selected to use the front end framework language Vue.JS, having not been exposed to any JavaScript framework, or even JavaScript itself, during my time at CIT represents a large risk related to experience as there are many beginner issues that I might encounter when developing the front end. To help lower this risk I have started an online course related to learning the basics of Vue.JS on a website called Udemy, that I plan to complete during the January break in order to acquire the necessary Vue.JS skills to develop the ambitious front end UI that I was for the project.

Another programming language risk is related to my choice of language of the back end of the project being Golang, another language that I was not exposed to directly during my time in college which poses as risk in terms of the lack of potential support should issues arise during the course of the development. Similar to the approach for my Vue.JS skills risk, I am spending time during the January month off practicing Golang and attempting to create multiple prototypes that will act as a foundation for the back end.

The final personal skill that I believe represents a potential risk is time management skills, specifically related to estimating time management with relation to completing

different tasks around the upcoming college semester. From completing the third year group project and this first section of the final year project I have learned that due to my minimal experience that it can be extremely challenging when trying to correctly estimate appropriate time frames to complete different tasks due the lack of knowledge around how to complete those tasks in terms of programming knowledge as a times, a steep learning curve might need to be overcame in order to ensure a feature functions correctly when it might have been assumed to be an easy process to complete beforehand. During the two previous projects referenced I used different methodologies in terms of tracking project progression and for task management including traditional Agile Methodologies using strict time based sprints to modified Agile frameworks that incorporate flexible time boxes for their sprints and Kanban Boards for task management and found that the latter method that allows for more flexibility when it comes to completing tasks produced better results both in terms of mental dexterity and achieved results at the end of the projects. During the later sections of methodology Implementation Plan Schedule I will outline the methods that I can to utilise to try and mitigate this risk

4.4.2 Development Challenges

To understand what risks could be associate with the completion of this project I also looked at other factors that pose risk during the development that would result on the application not functioning as intended. Hardware failure that could occur in my primary development machine, Code editor errors or coding errors could all negatively impact the development of this project and services such versioning control through GitHub and regular iterating testing will be utilised to reduce these risks from causing any fatal issues.

Using an easy to read language such as Golang for the back end will assist with detecting errors and the use of commenting will be employed to assist with understand complex sections of the code bases that will be generated.

The use of external deployment services such as Amazon through container management service Docker also poses a risk as it requires the project services images to be ran externally limiting the direct control. To assist with this I will also be ensuring to get a working version of the project implemented that can run directly from my local machine.

4.5 Methodology

The goal of Cornell Notes is to promote not only note taking but effective note taking and reviewing notes so to understand the depths for this project research was conducted into the history and rules that encompass the Cornell Note Taking System to ensure that its method would be replicated in an authentic way during the design and implementation of this project. To assist with increasing user engagement and further promote use of the platform.

During this semester I selected to complete a Game Development Elective which greatly assisted in learning about the methods used in games to engage users which I was able to use as a foundation for my research into applying gamification elements to non-game products. Other non standard gamification elements were then researched to try and determine what would be an appropriate solution to implement to entice users to create notes and review and also share them with other users which resulted in the design of a badge based trophy system that users can unlocked by completing various requirements.

During the research phase I tested out numerous different note taking solutions as documented in Chapter 2 to determine what gap in the market Cornell Notes could fill in terms of features that are almost required due to becoming commonplace in note taking apps to what features would help make Cornell Notes stand out and provide its own unique benefits to its users in the form of a tool that not only supports taking notes but actively promotes and supports reviewing notes too.

In order to acquire the development skills required to achieve the ambitious goals outlined for this project I will be completing an online course related to Vue.JS in January 2020 along with practicing Golang and trying to develop numerous prototypes to help when starting the project in the second semester of this college year. As referenced during the risk section previously ensuring that familiarity with these languages is available to me by the time it comes to developing the platform will be essential to ensure the timely completion of the project. I will also need to find guidance from other online tutorials, documentation, and look to books available from the library and a few that I have bought related to Golang, Vue.JS and efficient software development.

The project management approach that I believe will offer the best support for the completion of this project is a modified version of the Scrum framework related to the Agile Methodology. Agile is all about change and as it can be used both during the building stage to ensure all areas are correctly developed, It can also be used when looking at any changes in behaviours the user will face as they move throughout the platform and progress to learning better note taking and reviewing skills. Changing behaviours is challenging however I believe the use of gamification elements throughout

the platform will assist users with modifying their behaviours for the better and promote more efficient note taking and reviewing techniques. The version of Agile that will be implemented will include rough time based sprints that are supported by a large product backlog divided by service area along with tasks ordered by priority. Each of the sprints will begin with a retrospective as to what has been completed in the previous sprint along with any tasks that were unable to be completed and an outline of what tasks will be attempted during the course of the upcoming sprint.

This differs from the traditional Scrum framework in two main areas, the first being that the sprint retrospective will be completed at the beginning of each sprint rather than at the end and there is flexibility for times when tasks are not completed by the end of a sprint, if tasks are being completed quicker than expected, or if I change any requirements as that I will be open to completely reorganising the Implementation approach schedule and the second in that as this project is not a group project there are not roles of project owner, scrum master or daily scrum meetings with others that will be required. Instead there will be daily checks that I will conduct to monitor progress and will be meeting with my Term Two project supervisor weekly to outline what tasks have been completed since the last meeting and update them on the overall project progress.

4.6 Implementation Plan Schedule

To ensure the implementation stage of the project runs smoothly and is completed on time, I will use the modified Agile process referenced in previous sections. This will allow me to assess the direction of the project throughout the development life-cycle. I have broken down the schedule into size separate sprints which will initially have an estimated time box of two weeks each . These iterations will allow me to both stay focused and have a functioning chunk of software each fortnight. This incremental approach will allow for design changes and testing to take part allowing for the early detection of bugs or logical errors. Using this adaptable plan will enable modifications to occur depending on feedback received from my term two project supervisor. Figure 4.20 outlines the overall area that will be completed during each of the six sprints along with what the expected tasks to be completed from the backlog are.

Sprint	Estimated Dates	Service	Tasks
Sprint 1	Semester 2 Week 1 - 2	Back End Front End Database Deployment	Backend : Create API router Backend : Create DB CRUD Frontend : Project skeleton Frontend : API router Deploying Services remotely
Sprint 2	Semester 2 Week 3 – 5	Back End	Create APIs Create Badge Functions
Sprint 3	Semester 2 Week 6 - 7	Front End	Login/ Register Page Sidebar Header Notes
Sprint 4	Semester 2 Week 8 - 9	Front End	Cornell Notes component
Sprint 5	Semester 2 Week 10 - 11	Front End	Reviews Badges
Sprint 6	Semester 2 Week 11 - 13	Front end Testing	Markdown editor Cleaning up code and remaining tasks

FIGURE 4.20: Implementation Sprint Schedule

4.7 Evaluation

My implementation approach outlined previously will assist with evaluation the progression of the project during implementation during the Sprint reviews which will entail comparing completed tasks, which since are arranged by priority will ensure the most important and core features of the project will be completed early to ensure the core functionality of creating, sharing and reviewing Notes will be completed early.

Having created a list of the required APIs will assist in evaluating the progress and success of the back end as all the previously outlined APIs are required for full functionality of the application.

4.8 Prototype

Time management issues that occurred during this project limited the amount of prototyping that was implemented during this research phase and below I outline the two main sections where physical prototypes were implemented in the front end and back end to inspect the viability of utilising my selected technologies to ensure they would function as intended.

4.8.1 Front End Prototype

Front end prototyping was initiated by creating paper prototype diagrams to determine what layout the UI would provide to the user. The following Figure 4.21 which represents the main Cornell Notes editor, the PDF viewer and the platform header is the result of multiple iterations of paper prototyping which allowed for quick revisions as to the placement of different components within the design.

Figure 4.22 illustrates the sidebar which will be used by the user to navigate between Notes and reviews. It has the option of being hidden or expanded by the user allowing them to focus on the main container displaying their Note, Cornell Note, review or PDF should they choose.

I also implemented a basic Vue.JS application using both the Vuetify and NUXT libraries referenced earlier in this chapter for the Front End imported libraries which offer a fantastic, modern looking material design for components and different routing options that will be used to connect to the Back End. Figure 4.23

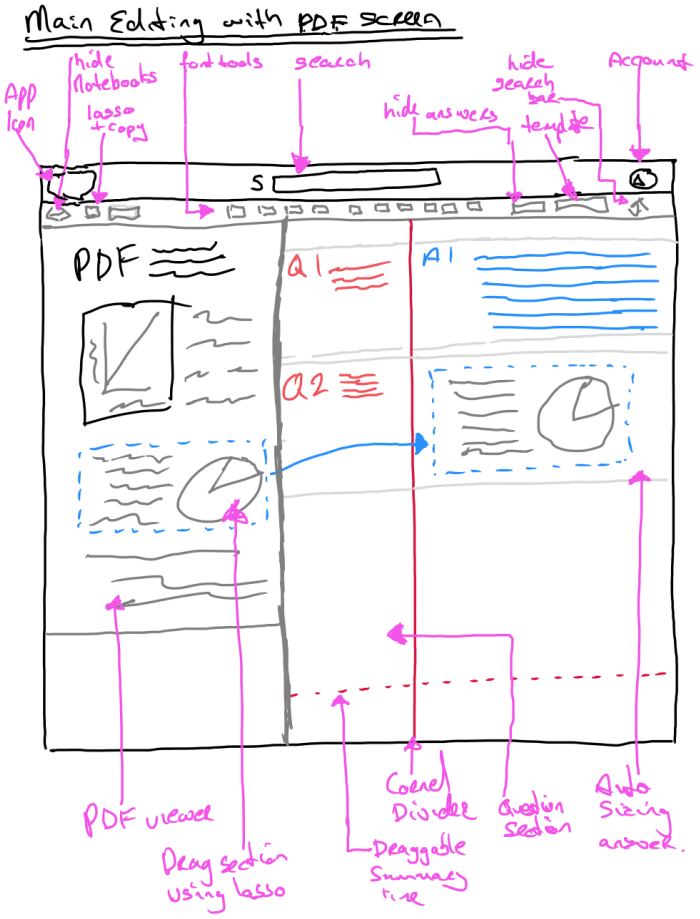


FIGURE 4.21: Wire frame sketch of front end displaying the main Cornell Notes editor, the PDF viewer and the platform header

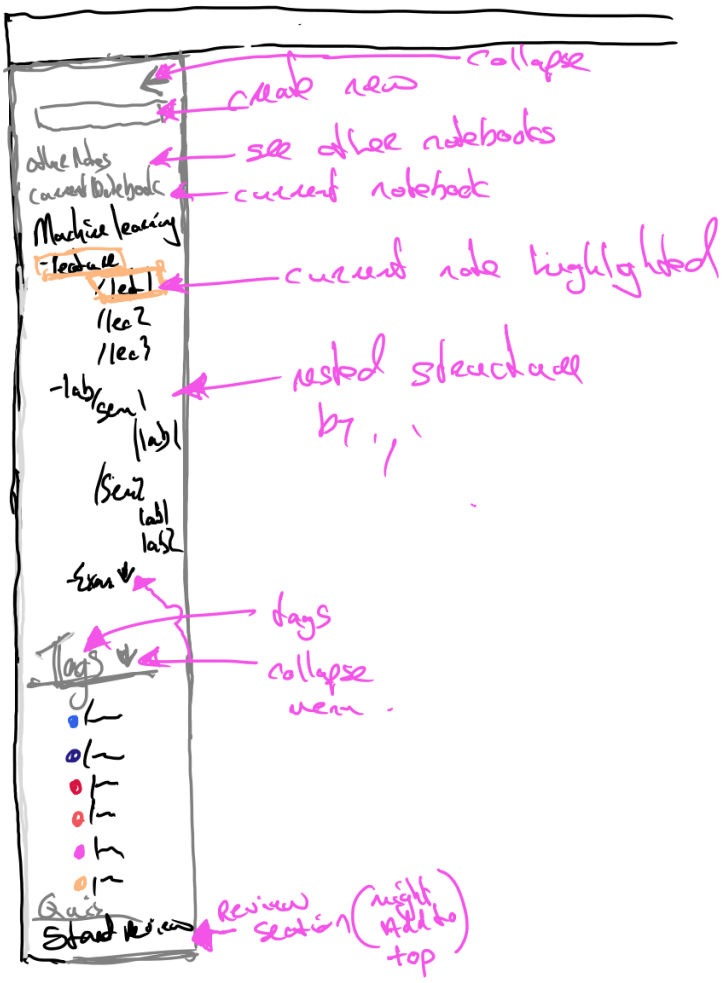


FIGURE 4.22: Wire frame sketch of front end displaying the sidebar with navigation routers for Notes, Reviews and Tags

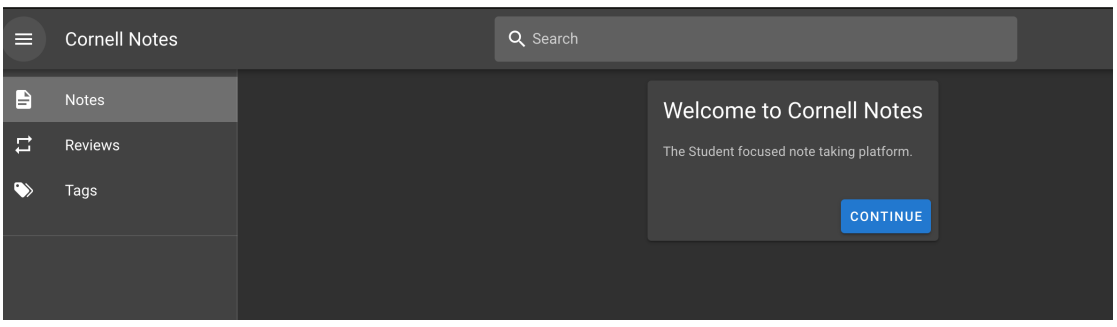


FIGURE 4.23: Vue.JS prototype showing partial header and sidebar

4.8.2 Back End Prototype

As a prototype for the Back End I created a Golang application that utilises the Gorilla Mux routing framework to provide functionality to serve API requests and also creating basic REST API for the Folder entity to allow for creation, retrieval, updating and deletion. Figure 4.24 outlines the routes that were created and this will act as a template for which future API requests use to add more complex routes for the APIs outlined during the description of the Back End APIs earlier in this chapter.

```
/*
 * FOLDERS
 */
router.HandleFunc("/folders", api.GetFolders).Methods("GET")
router.HandleFunc("/folder", api.CreateFolder).Methods("POST")
router.HandleFunc("/folders/{id}", api.GetFolder).Methods("GET")
router.HandleFunc("/folders/{id}", api.UpdateFolderName).Methods("PUT")
router.HandleFunc("/folders/{id}", api.DeleteFolder).Methods("DELETE")
```

FIGURE 4.24: Back End API Routes for Folder Requests

In order to serve each of these URL requests the different functions were create that implement the standard Golang handler interface. This interface takes in a responseWriter and pointer to a request as input parameters and can then be used to access the parameters of an API request that is sent using the route referenced in Figure 4.25 but also respond by sending http status requests or information such as Folder Name or description as seen in Figure 4.26.

```
// CreateFolder : creates and returns folder
func CreateFolder(w http.ResponseWriter, r *http.Request) {
    w.Header().Set("Content-Type", "application/json")
    var folder m.Folder
    _ = json.NewDecoder(r.Body).Decode(folder)
    folder.ID = strconv.Itoa(rand.Intn(1000000))
    m.Folders = append(m.Folders, folder)
    json.NewEncoder(w).Encode(&folder)
}
```

FIGURE 4.25: Back End function for Folder API Request Create

To ensure that these functions worked as intended and were able to serve the requests I utilised the application Postman, an API testing tool, to verify that the routers created would return the correct information as confirmed in Figure 4.27.

```go
// GetFolders : returns all folders
func GetFolders(w http.ResponseWriter, r *http.Request) {
    w.Header().Set("Content-Type", "application/json")
    json.NewEncoder(w).Encode(m.Folders)
}

// GetFolder : returns folder by id
func GetFolder(w http.ResponseWriter, r *http.Request) {
    w.Header().Set("Content-Type", "application/json")
    params := mux.Vars(r)
    for _, item := range m.Folders {
        if item.ID == params["id"] {
            json.NewEncoder(w).Encode(item)
            break
        }
        return
    }
    json.NewEncoder(w).Encode(&m.Folder{})
}
```

FIGURE 4.26: Back End API Routes for Folder Requests Create

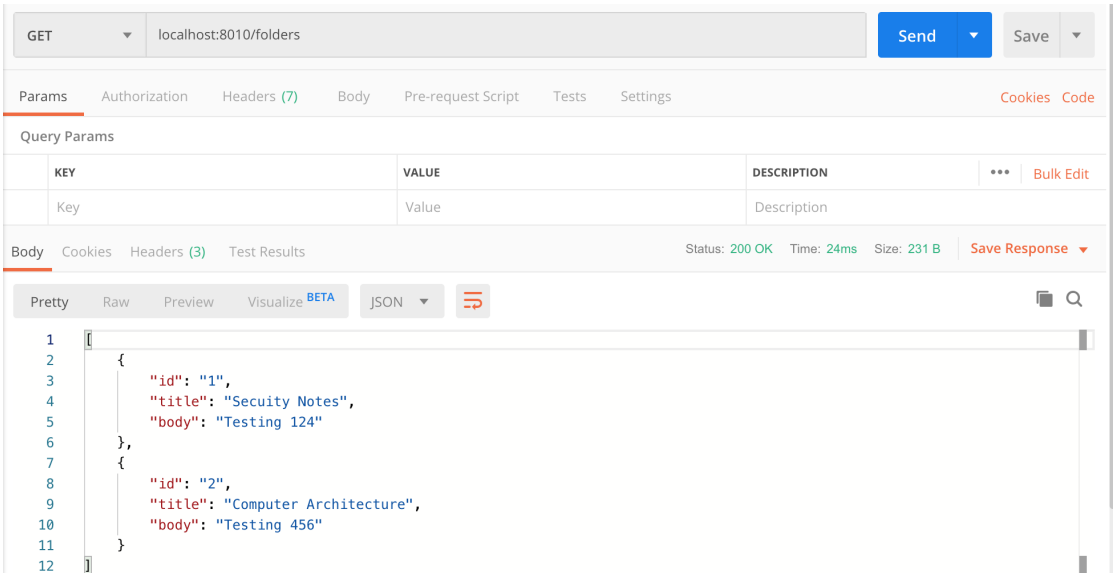


FIGURE 4.27: Postman API request to Back End for Folder information which show folders name returned along with correct HTTP status code

Chapter 5

Implementation

Implementing this project meant drawing on everything learned over the last four years while also exploring and learning several new technologies such as Go, Vue.JS, Web based authentication, HTTPS Requests and separating services. This made the implementation phase both a challenging and worthwhile experience.

Dwight D Eisenhower once quoted "Good planning without good work is nothing" and throughout this chapter I will outline the implementation I did of the research and planning undertaken and documented in the previous chapters which I believe meets the main criteria I set out for the platform in previous sections to provide a service that is: available on any device that allows students to create notes and study guides that implement the Cornell Note Taking System, provides a system for easily reviewing material and incentivizes students to create and review notes using gamification elements.

During the three week period between semester 1 and 2, during which I was able to start practicing with the languages and frameworks selected and creating various sample prototypes to get a better understanding of these languages I learned a lot about what sections of the project would require more work than initially planned for and adjusted some aspects of the project's planning implementation as detailed in the next section. There were also various challenges that I encountered throughout the development of Cornell Notes that range of feature specific issues or language specific issues that I was confronted with during individual Sprints to external factors like the Global Pandemic that caused changes to my college workload and time available leading to some features being removed or slimmed down.

There were two main compromises that I found were required before the commencement of semester two and numerous others that were found during various Sprints. I will outline the changes decided on before the implementation in the next section before going

into each of the Sprints individually documenting what work was required, challenges or roadblocks detected, changes made as a result of different issues and show images and code snippets of some sections developed.

5.1 Project Changes before Implementation

5.1.1 1. Relational vs Document-Store Database

During chapter 4, reThinkDB was mentioned as the database choice due to being open-source, lightweight and built for real time web activities which I had believed would have been suitable for this project. During the brief break between semesters and building basic prototypes when I was learning Vue.JS, I realised that the requirement of learning and implementing another new framework that I would need to communicate with reThinkDB would put me under much stricter time constraints for each feature in the Sprints, which I was learning were already poorly estimated based on the time I had spent being able to practice with Vue.JS during the 3 week semester break.

Instead of utilising reThinkDB I opted for selecting a standard SQL relational database, with which I had experience writing different CRUD operations for during my second year at CIT and during my third year placement. I knew that this would incur its down disadvantages when it came to real time updates for collaboration, requiring the additional of listeners to the database which was one of the main reasons for initially choosing reThinkDB as a solution for this. However it would still provide me with the ability to develop the required calls between the Go Back End and the Database at a faster rate and allow me more time to focus on learning the Idiosyncrasies of Vue.js that I needed.

The overall platform layout would remain unchanged as the tables I displayed in chapter 4 had been drafted using a relational DB structure with the idea that during the break, when I had the time available to learn more about document store databases, that I could modify these respectively. I did however end up creating additional tables to the 17 previously outlined to achieve the goals of the system including folder_items to store both notes and cornell notes to further separate out individual tables and their responsibilities. The SQL instance is initialised by a Go service that creates the required tables along with populating the database with sample information to assist with development and testing.

5.2 2. Sprint Structure

One of the major risks outlined during the projects Risk Assessment in chapter 4 was the personal skills required to be learned in order to fulfill the implementation of this project. During the 3 week period of learning Vue.js between semesters I realised that the volume of information required to be learned and practiced for the different areas of the Front End would require me to constantly be spending time researching and testing different approaches during each Sprint. This would conflict with my initial Sprint plan to spend the first 2 Sprints, which would have amounted to 4 weeks, focusing solely on the Back End API requests. I decided then to shift the Sprint structure from bi weekly Sprints and were separated by Back End and Front End into weekly Sprints that were feature based. This meant I would be completing a total of 12 Sprints during the semester rather than 6. The new structure I researched and created with these new learning's can be seen in Figure 5.1 and 5.2 , the latter also displays the previous Sprint structure for comparison.

This new structure should provide an adequate amount of time working between both the Back End and with Vue.js on the Front End without compromising my ability to retain and recall the new information learned about Vue. In the following sections I will go into further detail about each of the Sprints focusing particularly on the following areas:

- Sprint Feature and goal
- Sprints tasks
- Outline of challenges encountered
- Sprint Retrospective - Learning's, tasks carried to next Sprint

Sprint / Week	Feature	Tasks Front End	Tasks Back End
Sprint 1	Services set up	Vue project Vuex store Vue Router Vuetify Vuelidate Icons Folder Structure Landing page	Go Project Gorilla Mux Router GoSQL connection Models Structs Database - Go Project - Create tables - Insert Sample Data
Sprint 2	Middleware & Authentication	Support HTTPS TLS Certification	CORs middleware Cookies
Sprint 3	Authentication	Sign in/sign up buttons Sign in/sign up modals Sign in/sign up requests Sign in/sign up validation Vuex - user	API Requests Validate users Logging system DB calls
Sprint 4	Sidebar	Dashboard Header Sidebar Vuex - sidebar - folders Displaying folders Displaying folder items Different icons per item type Displaying tags	API requests DB calls
Sprint 5	Cornell Notes	Add Note component Main component layout Header – basic info Summary Cues Answers Display options Vuex – current note	API requests DB calls
Sprint 6	Cornell Notes	Header – tags Update note Vuex – notes	API requests DB calls
Sprint 7	Notes	Header Update Note Note body Markdown	Support markdown API requests DB – more sample data
Sprint 8	Reviews	Add review Main component - Cue - Answer - Summary details Header	API requests DB calls
Sprint 9	Badges	Badges modal - Hidden - Earned - Current scores Notification for points	Listeners API requests DB calls
Sprint 10	Account/ Sharing	My Account Modal - Edit details - Log out - Delete account Friends section - Add - Edit - Share notes/ reviews	
Sprint 11	Deployment	Dockerise service	Dockerise service
Sprint 12	Testing	Clean up remaining errors	Clean up remaining errors

FIGURE 5.1: New Sprint structure

Sprint	Estimated Dates	Service	Tasks
Sprint 1	Semester 2 Week 1 - 2	Back End Front End Database Deployment	Backend : Create API router Backend : Create DB CRUD Frontend : Project skeleton Frontend : API router Deploying Services remotely
Sprint 2	Semester 2 Week 3 – 5	Back End	Create APIs Create Badge Functions
Sprint 3	Semester 2 Week 6 - 7	Front End	Login/ Register Page Sidebar Header Notes
Sprint 4	Semester 2 Week 8 - 9	Front End	Cornell Notes component
Sprint 5	Semester 2 Week 10 - 11	Front End	Reviews Badges
Sprint 6	Semester 2 Week 11 - 13	Front end Testing	Markdown editor Cleaning up code and remaining tasks

Sprint / Week	Feature	Tasks Front End	Tasks Back End
Sprint 1	Services set up	Vue project Vuex store Vue Router Vuetify Vuelidate Icons Folder Structure Landing page	Go Project Gorilla Mux Router GoSQL connection Models Structs Database - Go Project - Create tables - Insert Sample Data
Sprint 2	Middleware & Authentication	Support HTTPS TLS Certification	CORs middleware Cookies
Sprint 3	Authentication	Sign in/sign up buttons Sign in/sign up modals Sign in/sign up requests Sign in/sign up validation Vuex - user	API Requests Validate users Logging system DB calls
Sprint 4	Sidebar	Dashboard Header Sidebar Vuex - sidebar - folders Displaying folders Displaying folder items Different icons per item type Displaying tags	API requests DB calls
Sprint 5	Cornell Notes	Add Note component Main component layout Header – basic info Summary Cues Answers Display options Vuex – current note	API requests DB calls
Sprint 6	Cornell Notes	Header – tags Update note Vuex – notes	API requests DB calls
Sprint 7	Notes	Header Update Note Note body Markdown	Support markdown API requests DB – more sample data
Sprint 8	Reviews	Add review Main component - Cue - Answer - Summary details Header	API requests DB calls
Sprint 9	Badges	Badges modal - Hidden - Earned - Current scores Notification for points	Listeners API requests DB calls
Sprint 10	Account/ Sharing	My Account Modal - Edit details - Log out - Delete account Friends section - Add - Edit - Share notes/ reviews	
Sprint 11	Deployment	Dockerise service	Dockerise service
Sprint 12	Testing	Clean up remaining errors	Clean up remaining errors

FIGURE 5.2: Left: Old Sprint structure, Right: New Sprint structure

5.2.1 Sprint 1 - Week 1

5.2.1.1 Feature

As the first Sprint for this phase of the project, the goal is to set up the core services needed which are listed below and also create an efficient structure for mapping the different models and components in the different services.

1. The Front End Vue.Js service
2. The Back End Go service
3. The GoSQL database service

5.2.1.2 Tasks

Figure 5.3 shows a close up view from the previously outlined in Figure 5.1, this outlines most the packages that will be required by the Vue.JS service to be able to change routes based on URL changes, update information across components and state using Vuex store, service API requests utilising Vue router and to assist with the creation of custom components and styling of existing components the Vuetify framework was also imported.

Sprint / Week	Feature	Tasks Front End	Tasks Back End
Sprint 1	Services set up	Vue project Vuex store Vue Router Vuetify Vuelidate Icons Folder Structure Landing page	Go Project Gorilla Mux Router GoSQL connection Models Structs Database - Go Project - Create tables - Insert Sample Data

FIGURE 5.3: Tasks for Sprint 1

Go Back End Service: Setting up a bare-bones Go project took little time including setting up the basic Gorilla Mux API router which allowed the project to be exposed to external API requests. The majority of the time was spent related to the back end was related to the time it took to create the structs used to ensure they mapped correctly with the DB tables values and the Front End models to allow for quick decoding and mapping into the relevant structs. I created basic API instances to test the connect from the Front End and to the database to ensure all three services would be able to transfer information as intended.

Vue.js Front End Service: In the middle of learning about Vue.js there was a large update to the technology that changed how projects were created and managed, this

change from Vue CLI 2 to Vue CLI 3 ended up simplifying some of the set up steps as I was able to add Vue router, Vuex and Vuetify directly from the Vue CLI setup terminal meaning that when the project was then initiated that I only needed to install a couple of other packages such as material design, icon support and some component options in order to get the core sections of the Front End up and running and have a basic landing page appearing in my browser that could send a single API request to the Back End to confirm a successful connection between them.

Go Database ServiceA significant amount of time during this Sprint was spent on setting up the database for the platform as I knew that by ensuring that the tables were created and working as intended and as sample data would be added that it would be ease the testing of both the UI and API requests. I created functions that would run SQL queries from a Go map which allowed me to efficiently add new SQL calls when it came to both creating the tables and creating the data for each of these.

5.2.1.3 Challenges

There were no technical challenges that I encountered during this Sprint. I found that using the new Vue CLI 3 for the setup saved quite a bit of time in comparison to using the older Vue CLI 2 which I used to create the prototypes in semester 1.

I did encounter some time challenges when it came to entering in all the sample data for the database and creating the Go structs, I had anticipated that both would only take 1 - 2 hours where as it actually ended up taking closer to 4 hours when I counted the testing time to ensure it was functioning as intended.

5.2.1.4 Retrospective

The main goals for this Sprint being setting up the three main services were all completed, Figure 5.4 displays the terminal output when starting each of the services.

All bar one of the tasks set out was completed giving 14/15, the time it took to add in and test all the sample data for the database took more time than expected, as discussed in the challenges section, and was not all added by the time Sprint 2 commenced.

FIGURE 5.4: Top Left - Vue.js service, Bottom Left Go Back End service, right - database service

5.2.2 Sprint 2 - Week 2

5.2.2.1 Feature

When creating prototypes I learned that a middleware would be required by my Back End service to allow for secure connections between the Front End and Back End service using HTTPS which would then support CORS requests. This was needed in order to allow users to access the page without seeing an 'insecure site' error which blocks access entirely on some modern browsers. This is what prompted me to set the first Sprint after the services are initially set up to handle secure connections as outlined in Figure 5.5.

Sprint / Week	Feature	Tasks Front End	Tasks Back End
Sprint 2	Middleware & Authentication	Support HTTPS TLS Certification	CORs middleware Cookies

FIGURE 5.5: Tasks for Sprint 2

5.2.2.2 Tasks

The goal of this Sprint is to create the required CORS middleware within the Back End to support HTTPS. Rolling over from the previous Sprint I also needed to finish the sample data entry so I to achieve these goals the following Sprint tasks were created:

1. Getting Authorised SSL/TLS Cert to put the 's' in HTTPS for Back End

2. Add CORS middleware to Back End
3. Add cookie functionally to requests for Authorisation
4. Add request logging functionality for testing
5. Finish Adding Sample Data to Database service

5.2.2.3 Challenges

I had encountered most of the major challenges during when creating prototypes and was able to utilize LetsEncrypt to create valid SSL/TLS certificates without any issues this time around.

Creating there cookie authentication was time consuming to get correct so that it would only respond to requests sent with a secret key that the Front End possessed, Postman became instrumental here in testing and increasing the pace at which I would test new queries.

5.2.2.4 Retrospective

The main goals for this Sprint being setting implementing middleware to allow for secure HTTPS connection, CORS authentication and cookie creation for user sessions. I also was able to add in the remaining sample data to the database service which was the leftover task from Sprint 1.

5.2.3 Sprint 3 - Week 3

5.2.3.1 Feature

User Registration and Login are the goals for this Sprint. Implementing this early will allow me to ensure that the cookies I created during the last Sprint functions as expected going forth into the project and also creates necessary standard functionally these days in an application allowing a user to be able to persist data and return to use the platform on another device or multiple users on a single device.

Sprint / Week	Feature	Tasks Front End	Tasks Back End
Sprint 3	Authentication	Sign in/sign up buttons Sign in/sign up modals Sign in/sign up requests Sign in/sign up validation Vuex - user	API Requests Validate users Logging system DB calls

FIGURE 5.6: Tasks for Sprint 3

5.2.3.2 Tasks

Figure 5.6 outlines the main tasks to be completed for the Back End and Front End services,

Go Back End Service: In Chapter 4, I outlined four API requests that would be required for users to be able to handle including GetUsers, RegisterUser, LoginUser and EditUser. These were relatively quick to add in Go and to handle the requests which I tested initially using Postman before the UI requests would be created.

Vue.js Front End Service: The majority of the time during this Sprint was focused on the Front End, primarily implementing Vuex for user state management allowing for a user to be able to utilize different global and environmental state attributes such as the Back End APIs URL, user specific detail like ID, userLoggedIn and more.

For the GUI, buttons where added to the main dashboard for registration and login, each with opened separate modals to prompt user to enter the relevant information as seen in Figure 5.7 before sending the requests to the Back End

I also added a carousel gallery element to the dashboard that will later be injected with images of the applications main features including its modernisation of the cornell note taking system, its reviews and badges each also containing text with information about each feature.

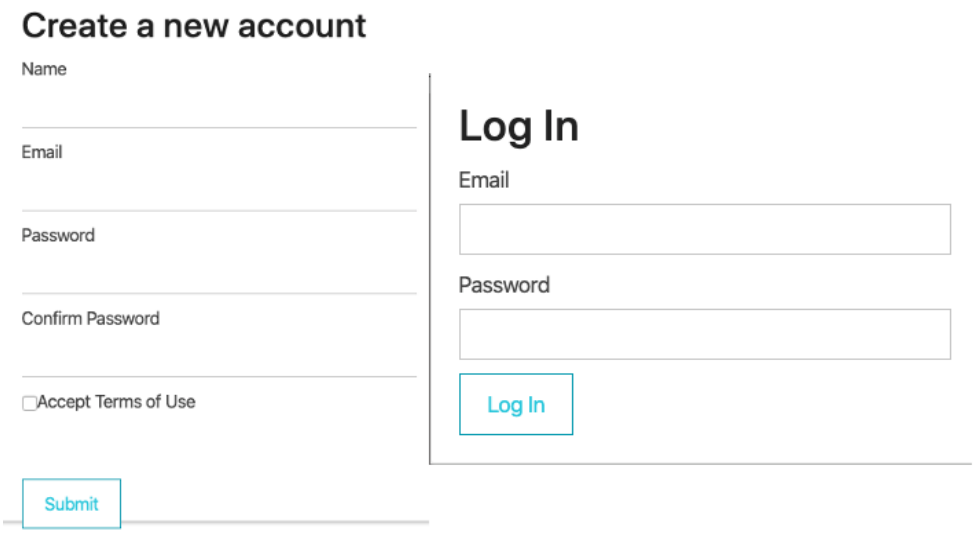


FIGURE 5.7: Sprint 3 - initial implementation for Login(right) and Register(left) Modals

5.2.3.3 Challenges

Still being relatively new to the Vue.js framework and JavaScript, there were numerous small challenges I faced that ended up taking a far greater amount of time than anticipated around styling of reactive components, the importance 'this.' and its way of passing events through the parent and child components for the modals.

I utilised the Vuetify's framework accommodations for sizes to help.I also encountered issues when adding the carousel and adding components to the main dashboard, at the moment the application looks extremely basic. I spent time looking at different templates online and tried to recreate but with my limited knowledge Vue I was unable to making it appear to complex. I have added to my Write backlog of tasks to update this should there be time later during the project and also to update the styling of the modals, including the 'forgot password' option that was missed during this Sprint but as the API was created will be added later.

5.2.3.4 Retrospective

Whilst I was displeased with the design of the dashboard and modals after implementation the functionality that was required was achieved through the completion of the Sprint Tasks. As mentioned in the previous section, I do need to add the 'forgot password' UI elements and API request.

5.2.4 Sprint 4 - Week 4

5.2.4.1 Feature

This week's Sprint contains and ambitious plan to complete one of the most important parts of the application, the sidebar. This will allow students to navigate between different folders and notes which will then open in applications main container. Folders will have an option to hide or display its items using a drop down style button. Each folder can contain either standard notes, cornell notes or PDFs, each of which is denoted by the item name and a unique icon allowing the student to quickly identify what type of item they are selecting. There will be a button on the header to open and close the sidebar and as I will be unitising Vuetify's Navbar component to create the sidebar, it will provide the option to overlay on smaller screens and close when unfocused automatically.

5.2.4.2 Tasks

Figure 5.8 outlines the tasks for be completed for this Sprint. It contains the most amount of tasks and work so far out of the Sprints up to this point. I anticipated that I would be able to allocate more time to this project which I thought necessary due to the amount of work that would be required on the Front End elements here particularly around utilising some of Vuex's features including the way it requires indirect access of its variables through mapmutations and strict mutator functions code guidelines. I knew that putting in the work here to ensure that I was learning the required structure for Vuex would pay of as it would be required in many other areas of the application that would be developed during the next few Sprints.

Sprint / Week	Feature	Tasks Front End	Tasks Back End
Sprint 4	Sidebar	Dashboard Header Sidebar Vuex - sidebar - folders Displaying folders Displaying folder items Different icons per item type Displaying tags	API requests DB calls

FIGURE 5.8: Tasks for Sprint 4

Vue.js Front End Service: The main tasks are related to the Front End where I needed to create the sidebar, facilitate the folders with different items and icons and the tags and be able to pass the open/close status between the sidebar and header component, which I needed to also create and add some components on, to determine when

to hide or display it. This would involve learning more about Vuex mutations and implementing them rather than passing information between parent and child components using params and events like the login and register modals. As I needed a basic dashboard for the login and register buttons added in the last Sprint, that task was already completed.

Go Back End Service: Here I needed to support requests to create and get folders along with their respective items. For the tag section I would need to return all the tags the student has added between their notes along with filtering notes by tag name.

5.2.4.3 Challenges

Similar to the last Sprint, the majority of my issues were related to my knowledge of Vue.JS, whilst this time I encountered new challenges more related to the Vuex's requirements I was able to overcome most issues at a much quicker pace than previous and did not encounter any issues, in this Sprint, that I could not overcome .

5.2.4.4 Retrospective

There was a large amount of learning in this week related to Vue.JS and the frameworks within it that I was using, it was a very beneficial and successful spring in that regard but also as I was able to complete all of the tasks and Figure 5.9 shows the end result.

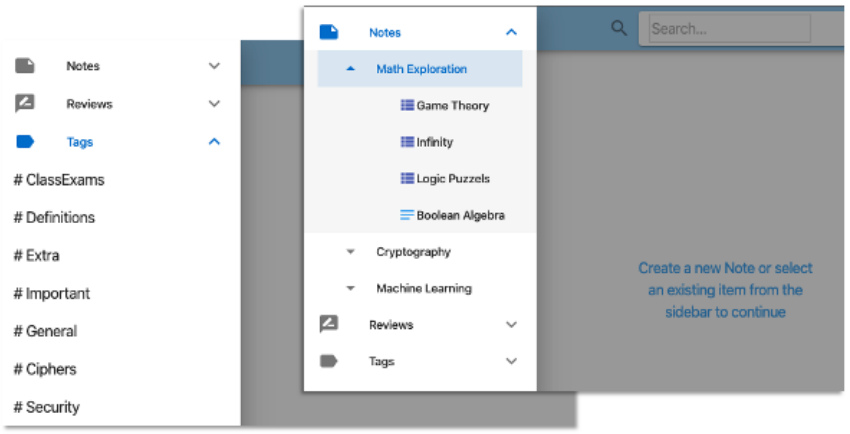


FIGURE 5.9: Sidebar showing closed folders and displaying all tags(left) and sidebar showing one open folder, it's items and their icons(right)

5.2.5 Sprint 5 - Week 5

5.2.5.1 Feature

During this Sprint and the following Sprint I shall be implementing the cornell note taking section of the platform. This Sprint will mostly consist of additions to the Vue.js section of the project adding the main components needed to display this feature to the student user. Using this feature, the student will be able to add new notes and edit existing notes by adding multiple cues and answers along with a dedicated summary section that will appear right at the top of the component, just below its header as a convenient modern option allowing students to quickly review using the summary to determine if they want to learn more by reviewing the cues and answers beneath. There also will be buttons within the components header that will modify the component arrangement in the display allowing the student to hide answers, view a PDF along side their note and as the summary is an optional part of the cornell note taking system, there will be also button in the components header allowing the student to display or hide this section.

5.2.5.2 Tasks

As the trend developing of increasing tasks per subsequent Sprint, this task follows suit with again, the most ambitious Sprint to date. It will also consist of primarily developing on the Vue.js section of the project as there are many small components that are contained within this feature. Figure 5.10 outlines the different components that will need to be added. In addition there is also an EditNote component required allowing the student to edit details such as the notes title, folder, tags or delete the note.

Sprint / Week	Feature	Tasks Front End	Tasks Back End
Sprint 5	Cornell Notes	Add Note component Main component layout Header – basic info Summary Cues Answers Display options Vuex – current note	API requests DB calls

FIGURE 5.10: Tasks for Sprint 5

5.2.5.3 Challenges

As I progressed through this Sprint I quickly realised that the time I had estimated for its completion was not going to be enough as I encountered numerous challenges when

creating some of the components that required a large amount of time to be invested in researching different JavaScript and Vue.js elements to be able to overcome and caused many of the tasks for this Sprint remain uncompleted by the time the week ended.

When creating the main Cornell note component comprised of the header, cues, answers, summary and PDF section, I encountered a multitude of issues that had not presented themselves during previous Sprints whereby the sizing would not conform to the dimensions that I had planned. This, along with a series of other unusual CSS related styling issues forced me to change how the element would be designed and implemented, using quite a lot of time to be able to follow the design I had planned to modernise and digitalize the Cornell Note Taking System by providing the options to add/ remove cues along with the new format for the summary section.

There is also an issue that I have not solved by the end of this Sprint related to how Vuex is updating when the Cues and Answers details are changing where when a new Cue is being added between the time the previous is sending its request to update and receiving confirmation from the back end that it was saved, that new data quickly added to the new cue can be overwritten. I invested more time into researching Vue component state and how to update components better through the use of the different life cycle hooks offered per component and hopefully will resolve this in the next Sprint.

5.2.5.4 Retrospective

Due to the challenges encountered during this Sprint there were many tasks that had to be pushed to Sprint 6 which include:

- Header - Display Options
- AddNote Modal
- API calls

In addition, I also have to fix the Cue overwriting bug mentioned in the challenges section. There is a possibility that the structure of these Sprints require altering if I cannot catch up over the following two Sprints. I had allowed for some time in Sprint 7, where I would be working on the Note section. This was planned due to the expectation of having numerous other assignments due from other modules allowing for less time to be spent on this project, however I believe that I will probably need to add in some part of this component into Sprint 7.

5.2.6 Sprint 6 - Week 6

5.2.6.1 Feature

This Sprint will consist of continuing working on the Cornell Note feature of the application. Due to the challenges encountered during the previous Sprint and the uncompleted tasks my original sprint structure for this Sprint and the following Sprint, needed to be changed. This Sprint will focus on completing the tasks that were uncompleted from the last Sprint and there will be time allocated to try and resolve the issue related to cues being overwritten.

5.2.6.2 Tasks

Outlined below are the list of tasks that I will be implementing during this Sprint, as previously mentioned it consists of uncompleted tasks from the last Sprint along with some of the tasks originally planned for this Sprint, which can be seen in Figure 5.11.

- Header - Display Options, tags
- AddNote Modal
- EditNote Modal
- Vuex Store for notes
- Bug: syncing issue on front end when adding new Cues
- API calls

Sprint / Week	Feature	Tasks Front End	Tasks Back End
Sprint 6	Cornell Notes	Header – tags Update note Vuex – notes	API requests DB calls

FIGURE 5.11: Original Tasks for Sprint 6

5.2.6.3 Challenges

I was able to make a moderate amount of progress during this Sprint and complete the majority of the tasks outlined previously. I did encounter a few issues when developing the addNote and Edit Note modals related to the way I was managing note tags. I was able to overcome these issues however I did spend more time than I had estimated.

5.2.6.4 Retrospective

Overall this was a very successful Sprint in that I was able to almost completely catch up with some of the delays incurred as a result of the challenges experienced during previous Sprints. As a result of this and the previous Sprint the Cornell Note taking feature has been almost completely implemented as planned which can be seen in Figure 5.12.

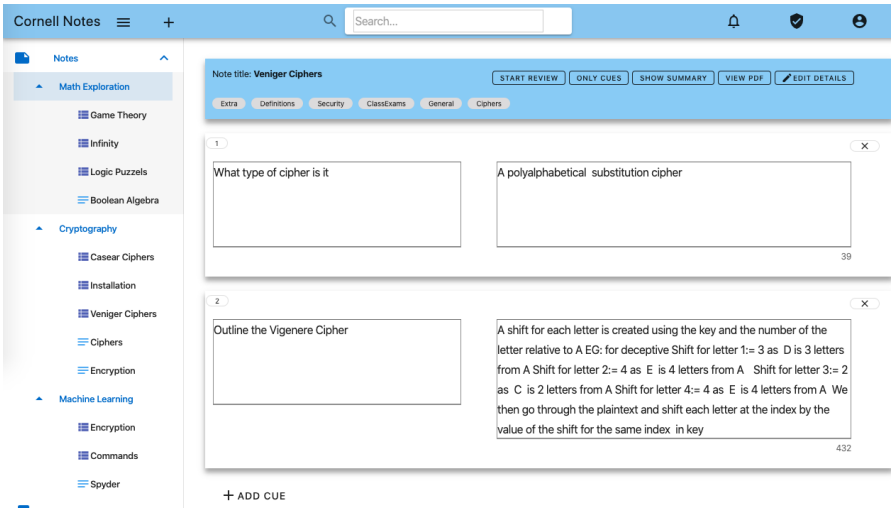


FIGURE 5.12: Cornell Note taking feature

A summary option was also implemented, when the summary button in the components header is selected a new text field appears directly beneath the header, in a convenient place to quickly become informed about the contents of a note without being required to read each of the cues. An example of how how notes can be added and their details modified can be viewed in Figure 5.13 which displays both the addNote and EditNote modals.

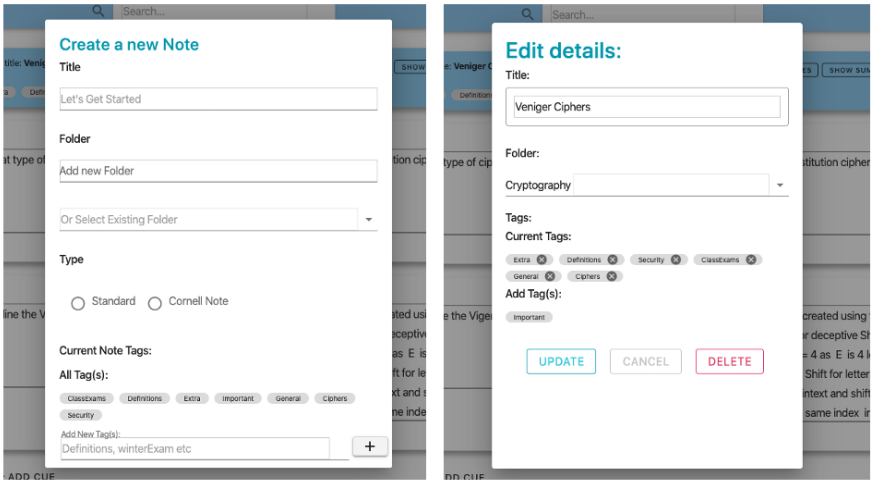


FIGURE 5.13: Add New note Modal(left) and edit note modals (right)

5.2.7 Sprint 7 - Week 7

5.2.7.1 Feature

This Sprint will focus on taking advantage of some of the features implemented previously such as the addNote and editNote components and apply them to the standard note feature. This will allow students to be able to create standard notes like any other note taking application. I will also be trying to implement markdown into the platform. This will provide students with a convenient and quick way to stylise their documents and should enrich the overall user experience of the platform.

5.2.7.2 Tasks

I anticipated that the majority of the time allocated for this Sprint will be spent implementing the Markdown feature as the Note components will be extending on the foundation I laid out in the past Sprints creating the different modals for adding and editing Cornell Notes. Figure 5.14 outlines the main tasks to be completed this week.

Sprint / Week	Feature	Tasks Front End	Tasks Back End
Sprint 7	Notes	Header Update Note Note body Markdown	Support markdown API requests DB – more sample data

FIGURE 5.14: Tasks for Sprint 7

5.2.7.3 Challenges

The biggest challenge encountered during this Sprint was in implementing a live markdown feature. During my research I had investigated a few options for markdown including writing a custom markdown library which was not selected as the time it would take would be a large project in itself or importing existing markdown libraries such as Markdown.js or Vue.markdown. Due to the time constraints that would be associated with option one I had selected to choose to import an existing markdown library, similar to importing Vuex, or Vuetify which I used elsewhere however when trying to develop this features I learned that these libraries do not provide the functionality that I wanted for this platform. Although it was possible to write using markdown format, there was no live preview option and the only way to display the stylised text to the user would be in a separate panel which would not suit the overall UI. I spent four days investigating other options and researched trying to build a basic custom markdown feature. The results concluded that there would not be enough time to implement a even a simple

custom markdown feature due to the amount of learning that would involved related to the JavaScript knowledge required.

5.2.7.4 Retrospective

I would mark this Sprint overall as a fail in terms of tasks completed. The standard note feature has been implemented as seen in Figure 5.15 however I came across the markdown issue that could not be overcame and resulted in the removal of that feature from the application. I am disappointed that its not going to be a part of the final project but the time it would take to invest in implementing this at this stage would negatively impact other sections of this project.

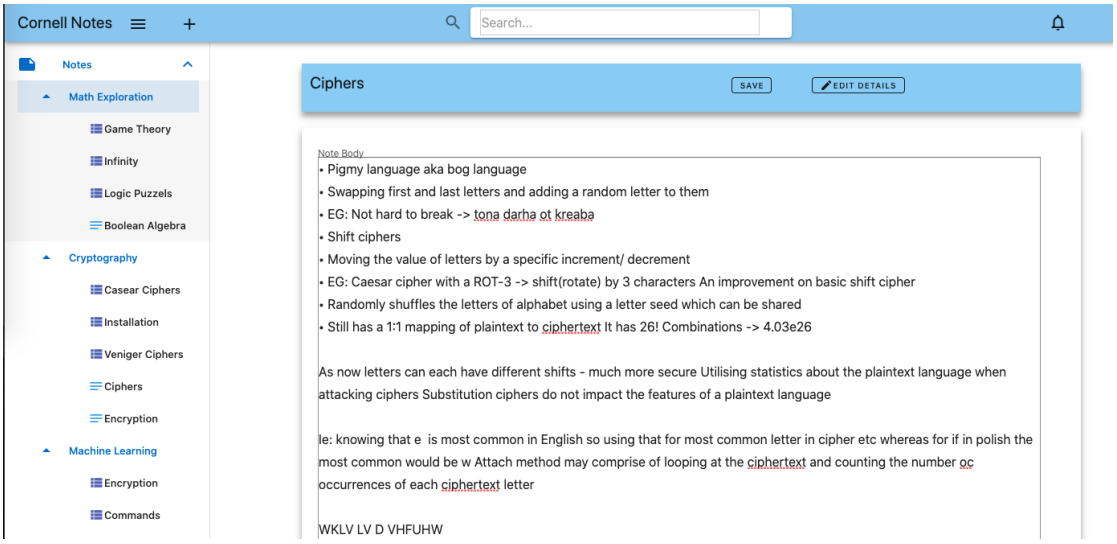


FIGURE 5.15: Basic Note taking Feature

5.2.8 Sprint 8 - Week 8

5.2.8.1 Feature

A core part of the Cornell Note taking Systems methodology is associated with reviewing the material that was documented using the system. The review feature of this platform is designed to provide a quick, easy to use and beneficial tool to allow students to review and test their knowledge of the notes they took. The goal if the review feature is to display Cues to the student along with options to show the notes summary, the Cues Answer or move to the next Cue when they are comfortable that they have a good understanding of the current Cue. When a review is completed they are provided with a summary that informs them of various statistics related to the review such as number of Cues reviewed, number of times they viewed the summary or Cue answer.

5.2.8.2 Tasks

There has been some external factors that have directly impact the about of time that has been allocated to the implementation of the tasks outlined in Figure 5.16. Due to the Covid-19 closure of the CIT campus and movement restrictions and having some vulnerable family members I have stripped back sections of this feature and others in later Sprints.

Sprint / Week	Feature	Tasks Front End	Tasks Back End
Sprint 8	Reviews	Add review Main component - Cue - Answer - Summary details Header	API requests DB calls

FIGURE 5.16: Tasks for Sprint 8

For this feature I will be focusing on the implementation of the features mentioned above and I will not be creating the custom addReview modal that would have provided options to combine Cues and Answers from different Cornell Notes into a single review. If time allows towards the end of this project then I will add this back in however for now, I believe the most important aspect to align with the goal of this project is to provide a review feature for a Cornell Note which I should be able to implement within the new time frame allotted.

5.2.8.3 Challenges

Due to the modification of tasks related to this Sprint, i was able to complete the review feature without encountering any difficulties.

5.2.8.4 Retrospective

I am pleased with the implementation of the review feature. It provides a quick and convenient way for students to quickly review the notes they have documented and provides some basic review statistics to inform the student how well they may know the material. Figure 5.17 displays what the student is presented with when they select to begin a review and Figure 5.18 displays the summary of the review to the student when they complete a review.

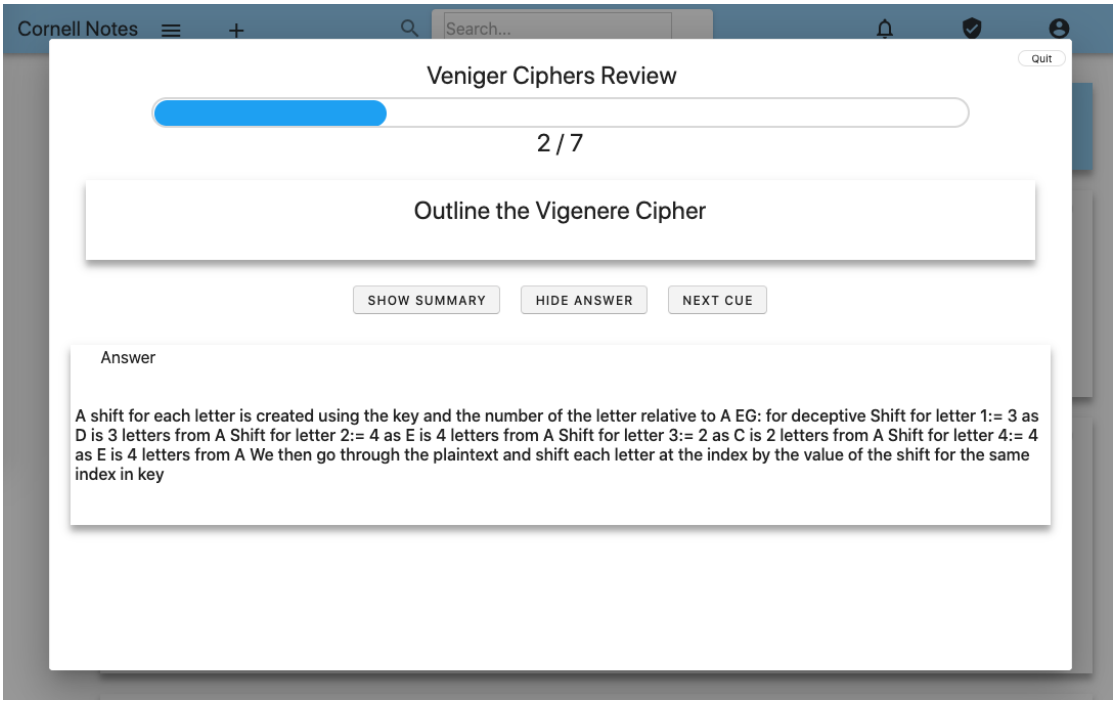


FIGURE 5.17: Cornell Note Review

I do hope that there will be some extra time in the coming weeks to be able to implement the custom AddReview section to be able to create customised reviews based on Cues across a variety of Cornell Notes. In the meantime I am pleased that the the base functionality allowing for a review to be conducted is ready.

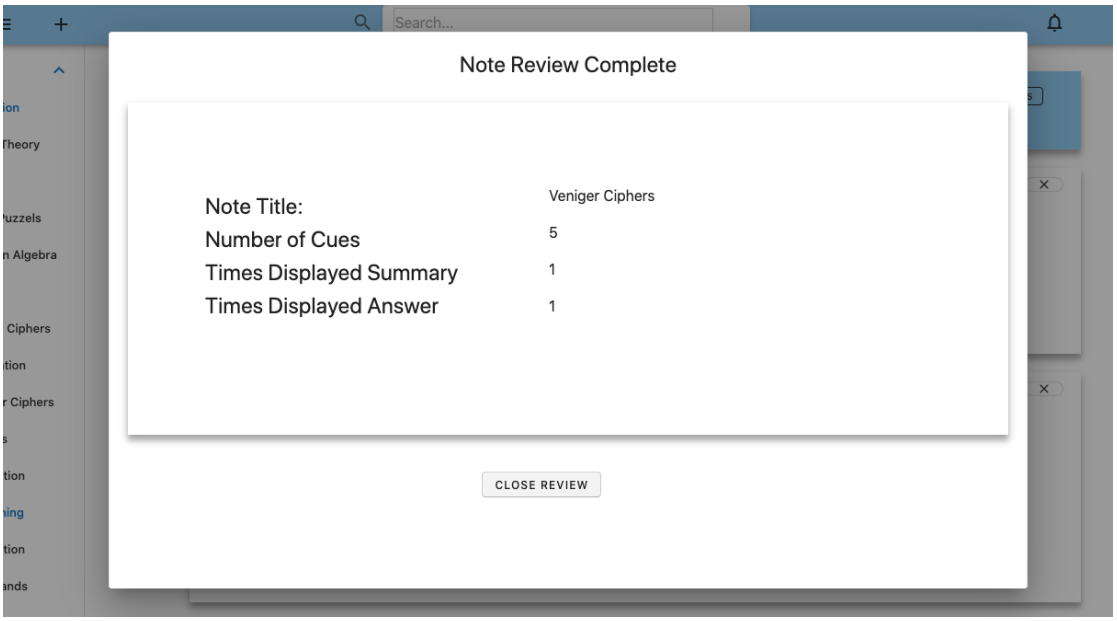


FIGURE 5.18: Cornell Note Review Summary

5.2.9 Sprint 9 - Week 9

5.2.9.1 Feature

The gamification elements within Cornell notes is designed and a motivational tool to reward students for creating notes, completing reviews and sharing material with other students. The goal of this Sprint is to implement the badges and rewards system that will provide each student with an overview of their use of the platform. Students will also be prompted when they earn points though a basic notification system.

Similar to the previous Sprint there are changes that I have had to make due to the ongoing changes with coursework and addition of unexpected assignments as a result of the changes implemented due to Covid-19. The notification system that had been planned initially is going to be altered to focus on the score modal and not on creating a notification drop down with a history of notifications.

5.2.9.2 Tasks

Figure 5.19 outlines the main tasks for this Sprint. Most of the work to be completed is related to the front end, adding in the review container, outlining the statistics and implementing badges that can be unlocked. I have limited the badges that will be available to be unlocked to the following:

1. Note Reviewed
2. Total number of Cues reviewed
3. Times displayed summary
4. Times displayed Cue Answer

I will also be implementing the notification system that will notify the user that they have been awarded points when they:

1. Create a new Note
2. Create a new Cornell Note w/ at least one Cue
3. Complete a Review
4. Share a Note
5. Share a Cornell Note

Sprint / Week	Feature	Tasks Front End	Tasks Back End
Sprint 9	Badges	Badges modal - Hidden - Earned - Current scores Notification for points	Listeners API requests DB calls

FIGURE 5.19: Tasks for Sprint 9

5.2.9.3 Challenges

External issues to this project prevented me from being able to spend the required time to complete this Sprint. The Badges and Rewards component was created and a basic style applied. I was able to write the API required to get the students scores and unlocked badges. I was not able to implement any of the updating or adding features in the form of the score modals.

I also encountered a resizing issues related to the badges that I was not able to resolve by the end of this Sprint. What should have been a simple enough component ended up taking far more time than I anticipated from the already reduced time allotment. I had been using the same CSS styling methods as in previous components but the badges refused to appears horizontally and then wrap vertically dependant on the width of the container. At the end of this Sprint they just appear vertically. It does not look good and I do want to get that resolved should time allow.

5.2.9.4 Retrospective

This Sprint contained some successes and some failures. Figure 5.20 displays the current state of the badges and rewards feature at the end of this Sprint where students can quickly see their unlocked badges highlighted in Blue and badges left to unlock in grey along with information about their current level based on their score. How many points needed to reach the next level and other user statistics related to badges that can be unlocked.

I realise that the reduced time I have to work on this project will result in either me not completing the tasks I had planned for initially or in the requirement for changes. I have selected to modify and reduce the final stage of this project to allow for it to get completed.

The Sprint structure for the final three sprints in week 10, 11 and 12 respectively will focus on the completion of the following:

- Sprint 10: Completing the Badges and Rewards Section

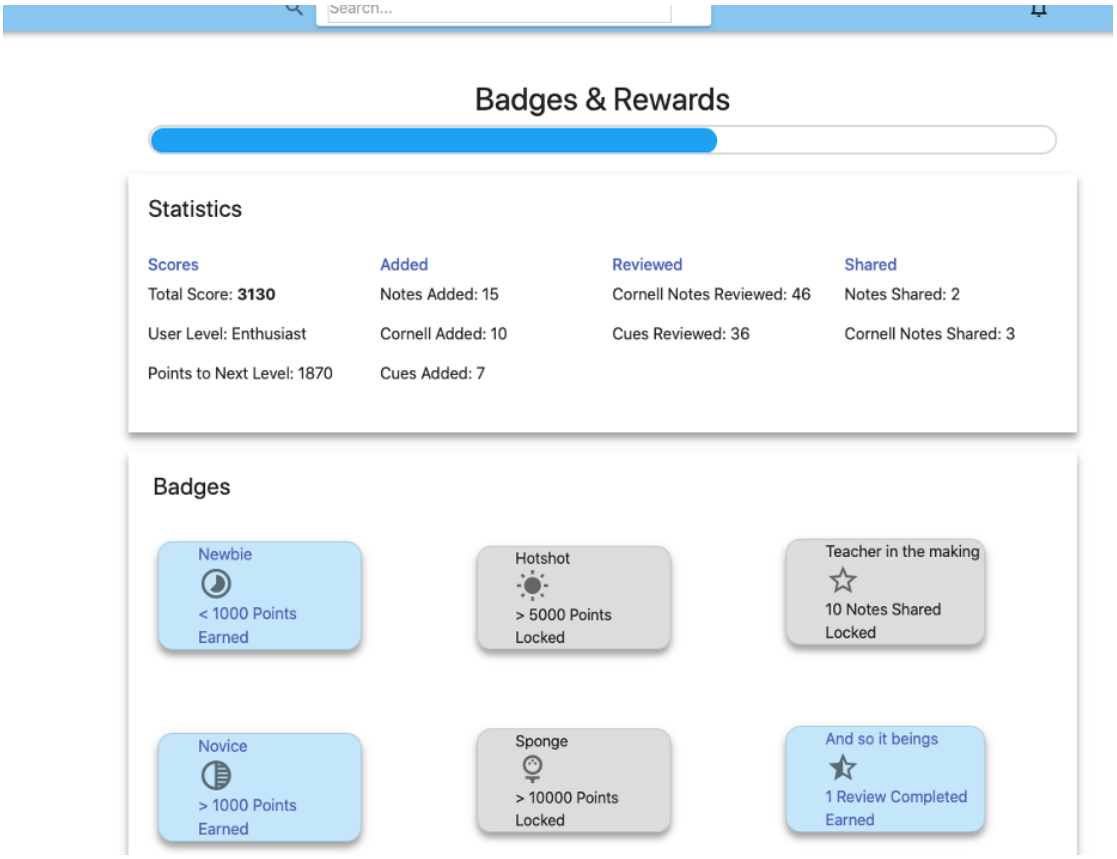


FIGURE 5.20: Badges and Rewards feature

- Sprint 11: Basic Note Sharing Feature
- Sprint 12: Account Management Feature

As a result of this I will not longer be focusing on the following tasks/features.

- Markdown
- PDFs
- Customised Reviews
- Adding project to Docker

5.2.10 Sprint 10 - Week 10

5.2.10.1 Feature

The goal of the current sprint is to complete the badges and rewards component by implementing the functions required to notify the student when they have earned points by using different features of the platform and create the required APIs to communicate any score changes with the back end.

5.2.10.2 Tasks

Unlike previous Sprints there will not be a figure displaying the current tasks due to the Sprint structure changing as referenced during the previous Spring retrospective. Instead the tasks to be completed during this Sprint are as follows:

- Adding temporary notification modal
- Adding points API:
  - Adding Note
  - Adding Cornell Note
  - Completing Review
  - Sharing Note
  - Sharing Cornell Note
- Updating badges and reward through Vuex

To try and ease the process in implementing these notification style pop ups that will inform the student how many points they are earning for what task I am going to utilise an external notification library called Snotify that provides customised toast notifications for Vue.js applications.

5.2.10.3 Challenges

Changing the structure of the Sprint allowed me to complete this Sprint without encountering any major difficulties. I was able to implement the snotify notifications library which saved me time in creating a customised notification system.

5.2.10.4 Retrospective

The new Sprint structure has worked well, I've been able to complete the score notifications and necessary API to update the database with a students new score. Figure 5.21 illustrates what the student is presented with when they are after earning points by utilising different features of the platform.

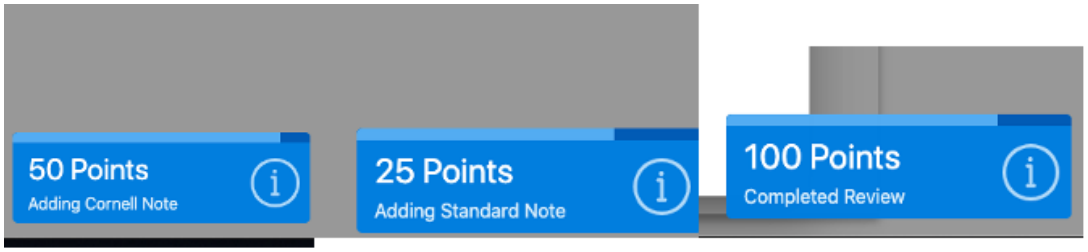


FIGURE 5.21: Score notification for earning points

5.2.11 Sprint 11 - Week 11

5.2.11.1 Feature

For the revised Sharing feature I will be focusing on the sharing of Cornell Notes between students. Students will have the option of adding classmates to their account and any Cornell Note will then be able to be shared and will appear within a new shared folder.

5.2.11.2 Tasks

The tasks that will be required here are to implement the classmates feature that will appear in the MyAccount section, here students will be able to enter their classmates email address to be invited. The implementation of an invitation email will not be completed during this Sprint and will be added to my future work. The following tasks are to be completed:

- create sharing component to MyAccount
- create addClassMate modal
- create share Cornell Note functionality
- Display shared notes in sidebar
- add Score for sharing notes

5.2.11.3 Challenges

This sprint was completed without any major challenges. There were some styling issues that arose related to the sharing component but the tasks outlined were completed.

5.2.11.4 Retrospective

Although the current implementation of the sharing feature does not exactly mirror the collaboration feature I outlined during the research phase in that shared notes are currently copied rather than a single note being able to be modified by multiple users simultaneously. This is another feature that I have moved into the future works category as the implementation when researched further in this stage resulted in the realisation that it would take far more than than could be allocated within the scope of the sprints. The current sharing feature does meet the goal of providing students with the ability to collaborate with others on notes and assists with the gamification elements as you can see any linked students current score level.

5.2.12 Sprint 12 - Week 12

5.2.12.1 Feature

Like most applications, the account management section will provide the option to view and modify basic information about the user. They will have the ability to change their name and password along with logging out.

5.2.12.2 Tasks

Along with implementing the account management section of the platform I will be attempting to fix any small bugs that are still present as this is the final Sprint.

The goal is to implement the standard account management features that are present in most applications including:

- Editing personal details
- Updating password
- Deleting Account
- Logging Out

The remaining bugs within the system that I would like to try and fix include:

- Filter by tags in sidebar stopped working - Not too sure when as it was not tested sine implemented and was working
- Badge styling - make dynamic as only appearing vertically
- Resizing from tablet to phone does not look good for Cues and Answers, they overlap instead of being on top of each other.

5.2.12.3 Challenges

Time was definitely an issue here. The Sprint's main tasks related to the account management feature were implemented however the remaining bugs that were planned to be further investigated and fixed were not all completed.

At the end of this Sprint the following issues are still present

- Filter by tags in sidebar stopped working - Not too sure when as it was not tested sine implemented and was working
- Badge styling - make dynamic as only appearing vertically

5.2.12.4 Retrospective

My comfort and ability to program using the Vue.JS framework has definitely increased over the last few Sprints. Issues that previously took me a significant amount of time to fix around Vuex state management and implementing error handing for the different requests we able to be added much quicker this time. It is unfortunate that such a level of ability was not available at the beginning of this project to help with adding the the features that were removed or altered to be less complex but I have most certainly met my goal of learning a front end framework.

5.3 Difficulties encountered

Throughout the implementation of the project there were many different challenges encountered. These challenges ranged from small issues such as CSS styling difficutlies in arranging components, inter service issues such as the CORs middleware issue discovered in Sprint 2 to quite large issues that directly impacted the overall features of the

platform and the approach taken during the implementation. In this section I am going to summarise some of the main challenges that impacted the completion and how I faced each of these challenges.

5.3.1 Vue.js Knowledge

One of the biggest challenges that I hoped to overcome during the implementation of this project was to gain a better understanding of a front end framework that would allow me to great GUIs for any application I plan on developing in the future. I had selected Vue.js as I had some brief exposure to it during my work placement in year 3 and from reading about it, has assumed that it would be something I could learn during the few weeks in between semesters. Due to having a smaller break in between semesters than planned for I was forced to continue learning foundation and intermediate aspects of Vue.js whilst undergoing the implementation which caused an increase in time per task completed related to the front end. I encountered far more issues than I had anticipated which lead to some delays in completing features and quite a large change to the complexity of some of the front end features.

Features that were directly impacted by my knowledge of Vue.js are as follows:

1. Markdown: removed due to lack of ability to create live markdown
2. PDF: removed as unable to render PDFs properly to display alongside notes
3. Tags: do not filter as expected

5.3.2 Covid-19

I think it is important to mention that in the middle of this projects implementation that access to the college campus and many other resources were removed due to the Corona Virus Pandemic. This has been one of the largest challenge to overcome during the project as it drastically impacted the amount of time that I would have available to work on this project. As well as the time it would take to be able to get support due to new responsibilities that I had to take on to assist vulnerable family members and due to the changing of coursework with the replacement of final exams that I had been preparing for already and the introduction of multiple new assignments that required many hours to complete, which had not been accounted for during the research phase. As a result of this, there were significant changes made to the final stage of the project, including numerous features modified or removed from the implementation plan and

instead moved to future work. This can be seen during the Sprints challenges and retrospective sections above where upon analysis it can be seen that time management was one of the biggest issues that resulted in features either not being complete or requiring alteration in order to be added.

5.4 Learnings

5.4.1 Vue.js

Experience and knowledge of Vue.js and JavaScript was a challenge that I mentioned often during the Sprints and as a major challenge that impacted significant sections of the overall project. Regardless of this, over the last three months developing with Vue.js I have gained a huge amount of understanding as to the foundations of its framework. I have been able to successfully create a multi-page application, reactive components and in some parts even created a UI that I am reasonably please with.

As I was learning Vue.js throughout the implementation of this project, I did make many decisions related to component styling, formatting along with file and directory choices that given the new knowledge I have since acquired, I would most certainly change.

Since prior to this projects implementation I had zero hands on experience, I would consider the knowledge gained as a result of completing this project to have been one of the most rewarding benefits.

5.4.2 Golang

One of the goals of this project was to increase my knowledge of the Golang programming language and having committed approximated six thousand lines of code to GitHub for the back end in Go, I have most certainly met that goal. The modification to the implementation schedule that moved the Go API from the first two Sprints to instead be spread across all twelve Sprints was a great decision as I had constant practice.

5.4.3 Project Management

Having completed the implementation of the research I feel that I have learned a lot of important lessons around the planning of a project and in particular around prototyping and working with what you know. One of the biggest challenges that I faced in this project was lack of knowledge in one of the main programming languages that I was

using. I had chosen to do this as I wanted to leave CIT with an understanding of at least one GUI related language however I realise now that for completing a project were time management is so important and subject to wide changes that it would have been a wiser decision to have chosen to complete this project in a language I was familiar with.

5.5 Actual Solution Approach

One of the foundations rules of an Agile workflow is to be able to respond to change. Throughout the implementation of this project there have been many different aspects that were modified as referenced during the Sprint overviews and in the challenges encountered section. In this section I will directly compare the following sections that I had outlined during the research phase with how those features/ sections were actually implemented over this semester.

1. Architecture of solution
2. Risk Assessment
3. Development Methodology
4. Implementation Schedule
5. Evaluation Approach

5.5.1 Architecture of solution

There was one significant change to the overall architecture of the platform and that is the removal of the deployment stage. This was a result of both missed deadlines for some tasks in Sprints and also the time allotment change that was required due to Covid-19. This resulted in not porting the services to Docker and hosting directly through Amazon as initially planned for.The remaining components of the architecture being the Front End, Back End and Database were each implemented as intended.

5.5.2 Risk Assessment

When comparing the Risk assessment conducted during the research phase to the main challenges encountered throughout the implementation phase of this project there are some direct correlations. The biggest risk I had proposed after my investigated was related to my personal knowledge of the languages and frameworks chosen for this project, with specific reference to Vue.js. This was indeed one of the biggest challenges that I faced and did result in the modification of numerous aspects of the project due to not having the ability to implement some complex features with the knowledge I had gained of the Vue.js framework/

I had also referenced getting the project hosted on Amazon via Docker containers as a risk, I had initially assumed that the risk might be associated with external factors

that i could not control such as licensing or costs and my proposed solution was to ensure that I had a version of the project that would be able to run locally so that I could still demo the project once completed. The local version of the platform has been implemented successfully allowing me to test and run the project throughout the implementation phase.

5.5.3 Development Methodology

The overall methodology that I had planned for in Chapter 4 remained unchanged throughout this implementation phase. The plan I had set in place was to utilise a modified version of an Agile based workflow where I would complete Sprints based on a back log. I had planned that this would allow for quick and easy modification of the Sprints to react to any challenges encountered. This was important as throughout the Sprint sections outlined in this chapter, there were multiple times that I had to change the structure or Sprints based on the remaining backlog of tasks to complete.

5.5.4 Implementation Schedule

In the opening of this Chapter I mentioned how during the break between semesters and project phases that I had opted to. modify the original Implementation Schedule for a new revised one. This resulted in expanding the six original Sprints into twelve new shorter Sprints. Figure 5.22, displayed again below, shows a brief comparison of both schedules and the format of the tasks associated with each Sprint.

5.5.5 Evaluation Approach

The evaluation outline described in Chapter 4 focused primarily on the project managements evaluation through Sprint management and API completion. I found that the Sprint approach that I planned for worked quite well and although the contents of some of the Sprint's tasks were altered, the structure of the Sprints remained the same. Each Sprint began on a Monday with an overview of the tasks to be completed. During the completion of these tasks I noted any issues to be reviewed during the Sprint retrospective which I conducted each Sunday. I found this to be a useful method for trying to keep accountable during each Sprint.

Having now completed the implementation phase I realise that this did not account for specific testing of different sections. The next Chapter will go into more detail about the specific tools and methods that I used to test the different services implemented

Sprint	Estimated Dates	Service	Tasks
Sprint 1	Semester 2 Week 1 - 2	Back End Front End Database Deployment	Backend : Create API router Backend : Create DB CRUD Frontend : Project skeleton Frontend : API router Deploying Services remotely
Sprint 2	Semester 2 Week 3 – 5	Back End	Create APIs Create Badge Functions
Sprint 3	Semester 2 Week 6 - 7	Front End	Login/ Register Page Sidebar Header Notes
Sprint 4	Semester 2 Week 8 - 9	Front End	Cornell Notes component
Sprint 5	Semester 2 Week 10 - 11	Front End	Reviews Badges
Sprint 6	Semester 2 Week 11 - 13	Front end Testing	Markdown editor Cleaning up code and remaining tasks

Sprint / Week	Feature	Tasks Front End	Tasks Back End
Sprint 1	Services set up	Vue project Vuex store Vue Router Vuetify Vuelidate Icons Folder Structure Landing page	Go Project Gorilla Mux Router GoSQL connection Models Structs Database - Go Project - Create tables - Insert Sample Data
Sprint 2	Middleware & Authentication	Support HTTPS TLS Certification	CORs middleware Cookies
Sprint 3	Authentication	Sign in/sign up buttons Sign in/sign up modals Sign in/sign up requests Sign in/sign up validation Vuex - user	API Requests Validate users Logging system DB calls
Sprint 4	Sidebar	Dashboard Header Sidebar Vuex - sidebar - folders Displaying folders Displaying folder items Different icons per item type Displaying tags	API requests DB calls
Sprint 5	Cornell Notes	Add Note component Main component layout Header – basic info Summary Cues Answers Display options Vuex – current note	API requests DB calls
Sprint 6	Cornell Notes	Header – tags Update note Vuex – notes	API requests DB calls
Sprint 7	Notes	Header Update Note Note body Markdown	Support markdown API requests DB – more sample data
Sprint 8	Reviews	Add review Main component - Cue - Answer - Summary details Header	API requests DB calls
Sprint 9	Badges	Badges modal - Hidden - Earned - Current scores Notification for points	Listeners API requests DB calls
Sprint 10	Account/ Sharing	My Account Modal - Edit details - Log out - Delete account Friends section - Add - Edit - Share notes/ reviews	
Sprint 11	Deployment	Dockerise service	Dockerise service
Sprint 12	Testing	Clean up remaining errors	Clean up remaining errors

FIGURE 5.22: Left: Old Sprint structure, Right: New Sprint structure

Chapter 6

Testing and Evaluation

Different testing methods were undertaken for each of the different services. This Chapter will outline the different testing approaches utilized in each of the three core services of the platform and their results.

The evaluation of the functional and non-functional requirements outlined in Chapter 4 will also be discussed in this chapter excluding those related to features or aspects of the project that were modified or removed as outlined during Chapter 5. This formed the majority of the testing through ensuring that core features to the completion of the projects goals were completed.

6.1 Service Testing

To test the functionality of the Go Back End Service's APIs that would be requested by the Vue.js Front End Service, tests were created that would check the HTTP response codes that were returned against a series of correct and incorrect requests. As HTTP status codes are divided into the five categories, each containing a range of codes, the following HTTP status codes were focused on:

1. 1xx informational response – the request was received, continuing process
   (a) 100 Continue
2. 2xx successful – the request was successfully received, understood, and accepted
3. (a) 200 OK
   (b) 201 Created

4. 4xx client error – the request contains bad syntax or cannot be fulfilled
    (a) 400 Bad Request
    (b) 401 Unauthorized
    (c) 403 Forbidden
    (d) 404 Not Found
    (e) 405 Method Not Allowed
5. 5xx server error – the server failed to fulfil an apparently valid request
    (a) 500 Internal Server Error
    (b) 505 HTTP Version Not Supported

Handling these then status codes meant that the Back End APIs should be able to withstand the majority of bad requests that would be sent within the scope of this application. It would also ensure that the Front End Service would be able to display appropriate error messages to the student along with handling of successful responses.

6.1.1 Back End Service Testing

Manual and automated tests were created to check the success or failure of the the APIs added to the Back End. The manual aspects of testing was completed through the use of Postman, an API management tool where API requests can be configured and sent, along with measuring different information about the returned response. Figure 6.1 illustrates one of the initial API requests tested the ǵetFoldersŕequest that is used when the student log into the platform to populate the sidebar with folder names and their contained notes. Postman offered a quick and convention options to set custom cookies along with a duplicate option allowing to copy working APIs, modify a component of it and test to ensure the correct HTTP response would be provided. Approximately sixty percent, which is around 25, of the APIs created for the Back End service were tested using Postman, this resulted in over fifty different Postman requests being created and run. I was able to automate some of the testing done via Postman further by utilising its Collection Runner component which allowed for the combining of multiple requests and their attached tests to be ran in sequence within a collection. This provided the ability to quickly run a subsection or all of the tests added within Postman with a single button press.

Being aware that in the majority of modern day applications automated testing through test suites developed along suit their main application resulted in me also adding tests to

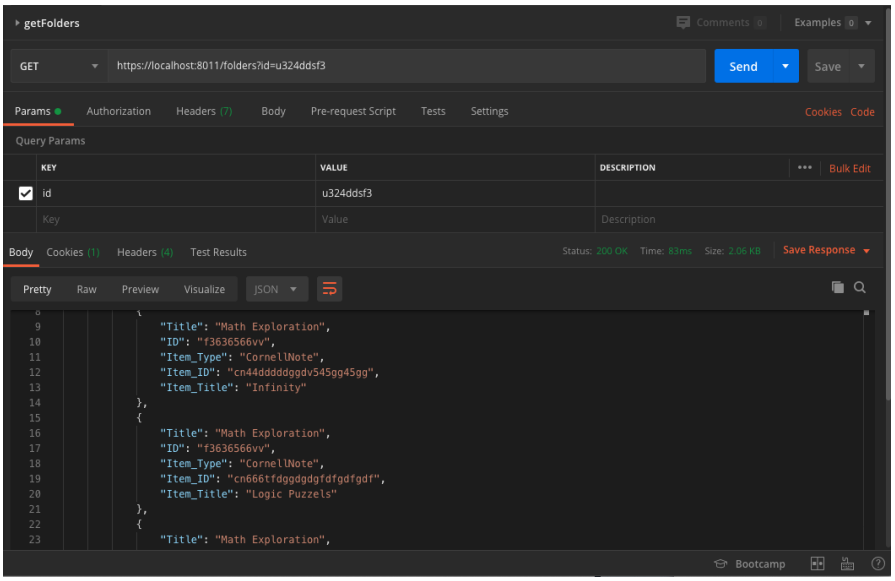


FIGURE 6.1: GetFolders API test using Postman showing successful HTTPS 200 response

the back end in Go. Having being advised that this is a preferred and usually more time efficient manor of testing services, automated tests using Golangs gotest package was also investigated and added. This allowed me to create a small number of automated tests for some of the most important parts of the back end that I could run with a single command. The disadvantage of implementing test suites for each API in the Back End is that it takes much more time than writing an API via Postman. That is the reason why for the scope of this project, Postman was the primary tool for testing APIs.

In Chapter 4, 62 different APIs were planned to be implemented, with the modification of the Review component to be based purely on the Front End along with the removal of the PDF viewing feature, this reduced the final count of APIs implemented to be 40 and the following Figure 6.2 compared the original API outlined in the previous chapters with those implemented including reasons.

6.1.2 Front End Service Testing

Similar to the Back End Service testing, the Front End functions testing primarily centered around the testing of HTTP response codes so that either the correct platform information could be returned to the student or an appropriate error code along with the validation of any user inputs within the application. These were handled within the Vue.js through the implementation of conditional v́-alertćomponents that would be displayed to the student if they made an incorrect requests or attempted to submit an incomplete form. Examples of these are displayed in Figure 6.3 where on the left hand side you can see the alerts presented to the student when trying to sign up if they

API Category	Function	Added	Reason if not implemented		API Category	Function	Added	Reason if not implemented
User	GetUser	✓			Folder	GetFolder	✓	
	GetUsers	✓				GetFolder	✓	
	RegisterUser	✓				GetParentFolder	✕	
	EditUser	✓				EditParentFolder	✕	single structure implemented
Groups	GetUserGroups	✓				DeleteFolder	✓	
	CreateGroup	✓				GetFolderByTag	✓	
	EditGroupName	✓				GetFolderItems	✓	
	AddGroupMember	✓			Review	GetReview	✕	
	DeleteGroup	✓				CreateReview	✕	
Notes	GetNotes	✓				RemoveReviewNote	✕	
	CreateNote	✓				AddReviewUser	✕	
	EditNoteBody	✕				RemoveReviewUser	✕	
	EditNoteName	✕				AddReviewScore	✕	
	EditNoteTag	✕	Changed to updateNote			EdiitReviewScore	✕	
	AddNoteToFolder	✕	changed to addFolderItem			DeleteReview	✕	Moved to Front End Only
	DeleteNote	✓			Tags	GetTags	✓	
	DeleteNotes	✕	Not required			CreateTag	✓	
CornellNote	GetCornellNote	✓				AddTag	✓	
	CreateCornellNote	✓				UpdateTag	✓	
	UpdateCornellNote	✓				DeleteTag	✓	
	DeleteCornellNote	✓			Badges	GetUserBadges	✓	
	GetCornellCues	✓				GetUserStats	✓	
	AddCornellCue	✓				UpdateUserStats	✓	
	UpdateCornellCue	✓			Account	GetUser	✓	
	DeleteCornellCue	✓				RegisterUser	✓	
	CreateCornellSummary	✕	Not needed as added by default			UpdateUser	✓	
	UpdateCornellSummary	✓				DeleteUser	✓	
	AddCornellNoteTag	✕				LoginUser	✓	
	EditCornellNoteTag	✕			PDF	GetPDF	✕	
	EditCornellNoteTag	✕	Handled using UpdateCornellNote			AddPDF	✕	Removed feature as unable
	EditCornellNoteAnswer	✕	Handled using UpdateCornellCue			DeletePDF	✕	to implement

FIGURE 6.2: Spread Sheet showing APIs Implementation Result including reasons if not implemented

already have an account. On the right hand side of Figure 6.3, alerts are presented to the student for trying to add a note with an empty note title and trying to create a new folder that already exists.

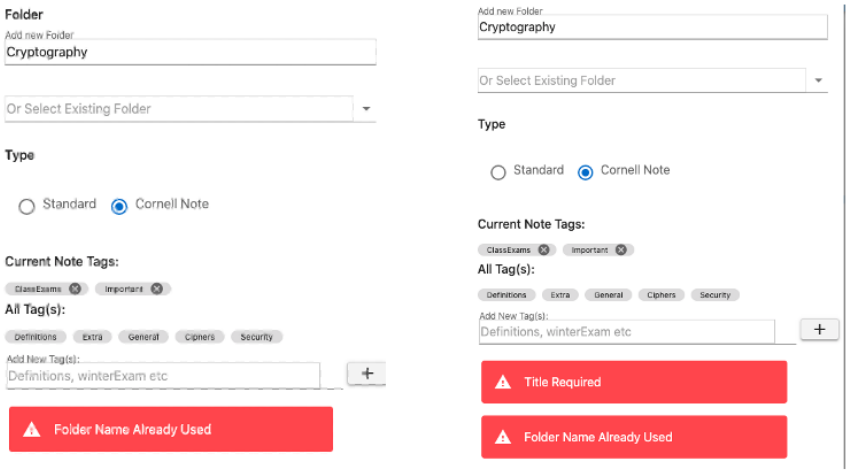


FIGURE 6.3: Left: Sign up error - user already exists, Right: add note errors - empty note name and trying to create a new folder with the same title as existing folder

To help debug the Front End services the official Vue.JS browser extension was utilised which provides additional functionality into a browsers inspect function. The extension provided the ability to view how data was being passed around the Vue project like a traditional IDE debug mode would allow. This saved a huge amount of time when it came to testing the success of the different functions added to the Front End service,

particularly when it came to managing the Vuex state configurations changing values as properties within different components get updated.

6.1.3 Database Service Testing

To test the database design and any changes that were implemented throughout the implementation phrase MySQL Workbench, a database modelling, visualising and management tool was utilised to create a range of tests for each table. This provide two main advantages. The first, being that it was a quicker option for checking and modifying the query structure when writing the database queries that were added to the Back End than testing in a production environment and secondly that it provided the option to store and run multiple queries in sequence to create collections of tests per table that could each be run with a single button click.

Each set of tests was created with queries that would pass or fail dependant on their inputs and format. This allowed for the determination of any database design issues that had not been expected when initially planning the design of the database in Chapter 4. Figure 6.4 outlines the tests for the cornell_users table where it run a range of queries that each have a different alteration to inspect different sections of the original query for any issues.

A large amount of sample data was also created and added to the project allowing for quick and easy testing of each of the twenty database table's designs. The Go project that initialised the SQL database contained functions to automate the creation, population of data and tearing down of tables so that the database could be altered and rebuild as needed.

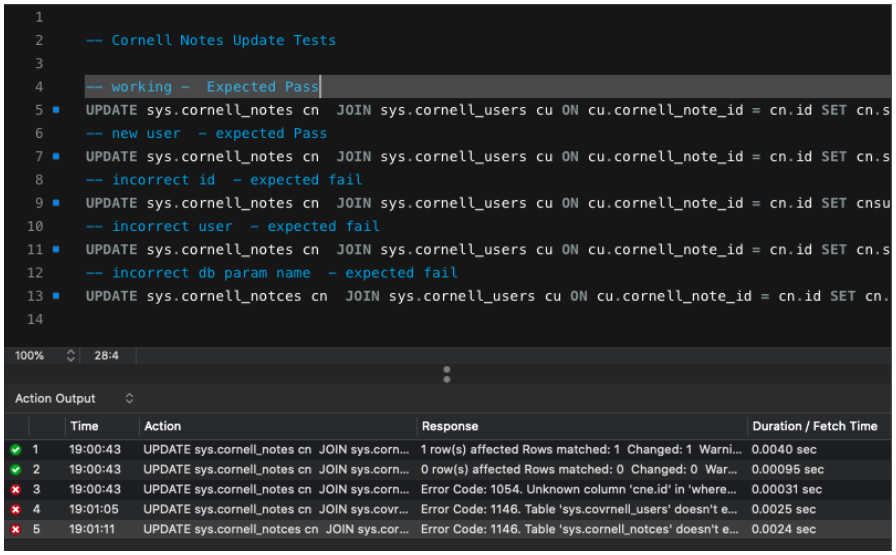


FIGURE 6.4: Testing cornell_users table in database using MySQL Workbench

6.2 Platform Evaluation

In Chapter 3, the following tasks were marked as objectives in the completion of this project:

1. Improve engagement by providing fast and simple interactivity with the application
2. Improve notes taken through a tested note taking system.
3. Encourage use and social engagement through collaboration and gamification elements
4. Provide existing note taking services such as cloud backup, standard note taking, folders, authentication.
5. Provide popular new services for taking notes like markdown

When evaluating the current state of the platform, the majority of the objectives mentioned above have been achieved with only Markdown and cloud Back up of the features referenced above having failed to be added to the platform.

Along with the set of objectives to be completed, a list of functional and non-functional required were outlined. These would be used as a measure of completion for various components and for the overall success of the projects implementation. Figure 6.5 displays a summarised version of the requirements from Chapter 3 along with the result of testing each component related to each requirement.

Feature	Functional Requirement	Pass/Fail	Notes	Feature	Functional Requirement	Pass/Fa	Notes
User Account	User can create an account	Pass		Collaboration	Multiple users can edit the same note simultanously	Fail	Not added to sprint tasks
	User can log in using Google A	Fail	Not added to sprint tasks	Gamification	Users can compete using quizzes	Fail	Gamification elements simplied due to time constaints
	User can reset password	Pass			Users can earn badges	Pass	
	User can edit account details	Pass			Users can gain points by completing tasks	Pass	
Groups	User can creat group	Pass		Interface	Users can view all folders and no	Pass	
	User can add others to group	Pass			User can accesss badges page	Pass	
	Group Details can be chaned	Pass			User can move between sections	Pass	
Standard Notes	User can create standard note	Pass			User can accesss account page	Pass	
	user can edit standard notes	Pass			User can accesss notes page	Pass	
Cornell Notes	User can utilise the Cornell Note taking system	Pass					
	User can add cue and answer	Pass					
	User can add summary	Pass		Feature	Non-Functional Requirement	Pass/Fa	Notes
	User can edit cues	Pass		UI	User Interface will be easy to nav	Pass	
	User can edit answer	Pass			UI Portable between devices	Pass	
	User can edit summary	Pass			Icons should indicate where to go	Pass	
	User can delete cue	Pass		Performance	Components should load quickly	Pass	
	User can hide summary	Pass			Multiple users should not degreate performance	Pass	
	User can hide answers	Pass					
	User can edit Notes Tags	Pass					

FIGURE 6.5: Functional and Non-functional requirements pass/fail with comments

Chapter 7

Discussion and Conclusions

7.1 Solution Review

The ultimate goal at the beginning of this academic year was to create a platform that I would have found beneficial to my learning during my time at CIT.While not all of the features proposed and planned during the research phase were able to be implemented, the platform does provide the core functionality that I wanted to develop within the platform and meets most of the objectives outlined during the research phase of this project being:

1. Improve engagement by providing fast and simple interactivity with the application
2. Improve notes taken through a tested note taking system.
3. Encourage use and social engagement through collaboration and gamification elements
4. Provide existing note taking services such as cloud backup, standard note taking, folders, authentication.
5. Provide popular new services for taking notes like markdown

Learning a new language and framework for this project has been a great experience and although it has resulted in a GUI that might not be up to the standards of modern day note taking applications, it offers quick as smooth response to user input, it supports all devices through dynamic resizing and provides feedback through alerts and notifications to assist in the user experience.

I am particularly pleased with the implementation of the Cornell Note component withing the platform. During Sprints 5 and 6, where I focused on this component, many

hours were spent trying to not only create a feature to support the Cornell Note Taking System by providing options for the Cues, Answers and Summary, but also to take advantage of digitising the system by providing different viewing options that make it easier to write Cues in smaller screened devices during lectures by hiding the Answer section and displaying the summary at the top of the note since it can take advantage of dynamic resizing to expand based on content length unlike traditional paper note books.

Although external factors, discussed during Sprint 8, lead to the reduction of complexity with relation to the creation of the Review component within the platform, the current state does support reviewing notes that were taken and further enhancement to this feature will be discussed during the later section, Future Work.

7.2 Project Review

This project has resulted in a large amount of learning for me on both technical skills and personal skills levels. The amount of knowledge I gained from researching the Cornell Note Taking System, particularly around the five R's of revision, outlined in Chapter 2, definitely benefited me when applying them to exams during the first semester. Similarly, having the opportunity to learn more about applying gamification elements helped me learn that it is an area that I would enjoy working with in the future after reading multiple reports and journals documenting its effectiveness.

Having now completed the implementation phase and evaluation there has been some key learning's that I can take away as a result of trying to implement the tasks prepared during the research phase. Looking back, where I to undertake this project again, for the scope of the college assessment I would not use Vue.js. I have found Vue.js to be a very interesting framework and having the opportunity to learn JavaScript and and how to create a responsive GUI has been a great experience, however the amount of time that was invested into learning this framework could have been utilised better had they been put into feature implementation. The majority of the smaller time consuming issues that I faced during the Sprints were directly related my inexperience with the framework and JavaScript as a language. This resulting in some features complexity being reduced and some being cut completely lead me to rethinking how I would redo this project many times.

I would have created the platform using an Android application. Although this would result in not meeting one of the main goals I had planned in the platform supporting all devices, being already familiar with Android application development means that I would have had more time to focus on the implementation of features rather than spending

so much time learning Vue.js and related fixing bugs. One of the major benefits to the platform I developed is that as the Vue.js services is separated from the Golang Back End, when I do make an Android application, it can take advantage of utilising the already implemented and tested APIs to speed up production.

In terms of the Back End services that were implemented. I have pleased with the functionality that is provided. The API requests were built to handle different types of results and service the appropriate responses. I would like to have had more time to invest in implementing different testing and monitoring services but as it was not ported onto Docker I feel that the Postman tests completed along with the Golang test suits are were sufficient to help me ensure that different component functions correctly.

7.3 Conclusion

Developing a student focused note taking SaaS based platform meant researching the benefits of note taking system, student learning and gamification along with comprehending the real value behind what the platform could benefit to all involved. It was an interesting experience getting to implement a project of such a large scale based on so many months of such research and investigation. The research I conducted helped re-enforce the benefits I believed this platform would provide to students once completed and released and I am pleased at the learning's I have gained through this implementation phase. I found it extremely interesting to learn about areas of the Cornell Note Taking System I had not reviewed as much such as the 5 R's discussed in Chapter 2. Having been able to apply these learning's to my current workload at the time helped assure me of the benefits that my project could provide once completed and acted as a great source of motivation. Behaviours and psychology play a major part in the reasoning behind a personals motivation, engagement and their drive to succeed. I found it really interesting learning about these aspects when researching gamification elements and constructing the Badges and Rewards framework.

The experience was invaluable in terms of the lessons learned around the planning of tasks and goals. Deciding to create a full Front End for my application through Vue.JS served to confirm what areas of programming I felt most comfortable working in. I found that I had a much more enjoyable time developing and debugging the Back End services over the Front End which helps provide an area to focus on in the future.

The overall solution implemented provides a platform that allows students to create, review and share notes as intended. They have the ability to create notes in a traditional block format or utilise the Cornell Note Taking System. The added gamification elements

help provide triggers for motivation and a measure of achievement to promote both the use of the platform and good studying habits in general. I am disappointed that some of the features were unable to be completed like the PDF support, and others who's complexity was reduced. Overall, however I am pleased that the goals I set out to achieve, and the problem I hoped to address and implement a solution for was developed to the level it has been.

7.4 Future Work

During Chapters 4's Sprints outline, there were multiple features that were either removed or altered due to the various challenges encountered and as mentioned, the majority of those would be added to this section to be added over the months following my completion of this project. There are many features that I learned about during this phase that I believe would fit well within this platform and many sections of the current platform that I would alter having learned so much more about Vue.JS over the course of this project than when I started and choose to make decisions around component structure, styling and state management. Below are the main sections that I plan to work on;

7.4.1 Vue.JS Structure Change

During the Sprints, I encountered various challenges around implementing different Vue.JS features and also many issues around its styling. Through investigating solutions to overcome these challenges I gained knowledge about different frameworks that I could have utilised to assist with my styling shortcomings along with different tips that I had not seen when initially researching Vue.JS during the research phase. By redoing the Front End through the implementation of some of these UI Kit frameworks that work alongside Vuetify, I could have reduced the amount of work that was put in to styling the different components along with better separation of components. I feel that having learned so much about Vue.JS throughout this project that by redoing the Front End using the learning's I now have will allow me to implement a much better looking UI.

7.4.2 Customised Reviews

One of the features that was altered due to some of the challenges encountered was the reduction of complexity to the review component. Initially it was planned to be able to

create customised reviews that would have Cues from multiple different Notes so that a student could prepare a more detailed review based on a collection of notes on a topic. I would still like to implement this feature over the coming months.

7.4.3 PDF Support

A challenge that was unable to be overcome within the time frame of this project was the functionality to allow for a side by side view of the notes and PDF so that users would be able to conveniently take notes based on PDFs in a single window. I was unfortunately unable to complete this feature due to a variety of challenges discussed during Chapter 5 but as I believe this would be a core feature that I would want to have available at the time of this platforms alpha release, it is one that will be investigated in more detail to be included.

7.4.4 Enhanced Collaboration

My long term goal for Cornell Notes would be that students could simultaneously edit a single note together by adding Cues and Answer and also have an option to compere their notes. This should provide a learning tool to help students take better notes by viewing others and also help with collaborating during lectures when taking the notes. It was something that was deemed to complex for the time frame of this project during the research phase but it was added to the list of features I wanted to include. I would like Cornell Notes to eventually becomes a platform where people can view and learn from notes taken by people they don't even know, like a Reddit for notes where people can up vote notes or reviews they think are well created to help new learners make the most out of their notes.

7.4.5 Enhanced Gamification

There are many different elements related to gamification that I wanted to include in this project. The elements that were included in the final submission, I believe, reflects many of the basic gamified applied however there are so many other elements that could be added to this platform. My initial idea was to create a quiz like system for the reviewing that would allow students to complete against each other. This was also deemed to complex for the time that would be allowed during this project but it is a feature that I also want to implement into the platform before it would be released.

Bibliography

[1] TeacherMom101, "How i use cornell notes effectively in my laguage arts classroom," May 2018. [Online]. Available: https://www.teachermom101.com/2018/05/how-i-use-cornell-notes-effectively-in.html

[2] G. a. Kerstiens, "Studying in college, then & now: An interview with walter pauk," *Journal of Developmental Education*, vol. 21, pp. 20 – 24, 1998.

[3] K. A. Kiewra, "How classroom teachers can help students learn and teach them how to learn," *Theory Into Practice*, vol. 41, pp. 71– 80, 2012.

[4] W.-C. Chang, "The effects of note-taking skills instruction on elementary students' reading," *The Journal of Educational Research*, vol. 4, pp. 278 – 291, 2014.

[5] D. C. Bui, "Note-taking with computers: Exploring alternative strategies for improved recall," *Journal of Educational Psychology*, vol. 105, no. 2, pp. 299 – 309, 2013. [Online]. Available: https://psycnet.apa.org/record/2012-27380-001

[6] F. P. Robinson, *Effective study.* Harper and Row, 1941.

[7] W. Pauk and R. J. Q. Owens, *How to study in college.* Wadsworth Cengage Learning, 2014.

[8] A. L. Costa and B. Kallick, *Learning and leading with habits of mind: 16 essential characteristics for success.* Association for Supervision and Curriculum Development, 2018.

[9] M. Berman, "In search of decay in verbal short term memory," *Journal of Experimental Psychology: Learning, Memory, and Cognition*, vol. 35, pp. 317–333, 2010.

[10] ——, "Replication and analysis of ebbinghaus' forgetting curve," *PLOS One*, 2015. [Online]. Available: https://doi.org/10.1371/journal.pone.0120644

[11] C. A. Backman, "Finding an effective note-taking system for math students," *na*, 1994.

[12] J. E. P. Robert J. Marzano, Debra J. Pickering, *A Handbook for Classroom Instruction that Works*. Pearson, 2004.

[13] J. Faber, J. Morris, and M. Lieberman, "The effect of note taking on ninth grade students' comprehension," *Reading Psychology*, vol. 21, 01 2000.

[14] S. K. M. McLeskey, T. Haydon, "A review of the effectiveness of guided notes for students who struggle learning academic content," *Preventing School Failure: Alternative Education for Children and Youth*, 2011.

[15] J. Boyle, "Strategic note-taking for inclusive middle school science classrooms," *Remedial and Special Education*, vol. 34, pp. 78–90, 03 2011.

[16] G. Z. M. Davoudi, N. Moattarian, "Impact of cornell note-taking method instruction on grammar learning of iranian efl learners," *na*, 2015.

[17] J. Backus, "The history of fortran i, ii, and iii," *History of programming languages*, p. 25–74, Jan 1978.

[18] J. G. Kemeny and T. E. Kurtz, *The Dartmouth Time-Sharing Computing System. Final Report*. na, na.

[19] J. Koenig, *na*. na, 2005.

[20] Priti, "What is saas all about? software as a service (saas)," Jan 2019. [Online]. Available: http://www.datacenterprofessionals.net/profiles/blogs/what-is-saas-all-about-software-as-a-service-saas

[21] C. L. Brun, "The benefits of multi-tenancy to manage it & communication expenses." [Online]. Available: https://blog.cimpl.com/the-benefits-of-multi-tenancy

[22] "Usability - iso 9241 definition." [Online]. Available: https://www.w3.org/2002/Talks/0104-usabilityprocess/slide3-0.html

[23] M. Rouse and N. Rando, "What is noisy neighbor (cloud computing performance)? - definition from whatis.com," Dec 2014. [Online]. Available: https://searchcloudcomputing.techtarget.com/definition/noisy-neighbor-cloud-computing-performance

[24] M. Levinson, "Software as a service (saas) definition and solutions," May 2007. [Online]. Available: https://www.cio.com/article/2439006/software-as-a-service--saas--definition-and-solutions.html

[25] M. Sailer, J. U. Hense, S. K. Mayr, and H. Mandl, "How gamification motivates: An experimental study of the effects of specific game design

elements on psychological need satisfaction," Dec 2016. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S074756321630855X

[26] G. Zichermann, *Gamification by Design.* OReilly Media, 2011.

[27] A. Matallaoui, J. Koivisto, J. Hamari, and R. Zarnekow, "How effective is gamification? a systematic review on the effectiveness of gamification features in exergames," *Proceedings of the 50th Hawaii International Conference on System Sciences (2017)*, 2017.

[28] P. Denny, "The effect of virtual achievements on student engagement," *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems - CHI 13*, p. 763–772, 2013.

[29] Gartner, "Gartner predicts over 70 percent of global 2000 organisations will have at least one gamified application by 2014," *Gartner Predicts Over 70 Percent of Global 2000 Organisations Will Have at Least One Gamified Application by 2014*, Nov 2011. [Online]. Available: https://web.archive.org/web/20141201115316/http://www.gartner.com/newsroom/id/1844115

[30] S. Deterding, D. Dixon, R. Khaled, and L. Nacke, "From game design elements to gamefulness," *Proceedings of the 15th International Academic MindTrek Conference on Envisioning Future Media Environments - MindTrek 11*, 2011.

[31] L. Jennings, "Wellness works, a wellness platform promoting social, interaction, encouraging achievement and monitoring progression," 2017.

[32] B. J. Fogg, "Behavior model." [Online]. Available: https://www.behaviormodel.org/

[33] B. Reeves and J. Reeves, "Ten ingredients of great games," *na*, 2010.

[34] Microsoft, "Official microsoft blog," *Official Microsoft Blog*, May 2012. [Online]. Available: https://blogs.microsoft.com/blog/2012/05/24/more-than-4-5-million-catholic-school-students-to-receive-microsoft-office-365-for-education/

[35] ——, "Microsoft offical blog," *Microsoft Offical Blog*, Mar 2013. [Online]. Available: https://blogs.microsoft.com/blog/2013/03/11/college-students-work-together-without-being-together-with-office-365-and-skydrive/

[36] huffington post, "Use google docs to collaborate on class note taking," *https://www.huffpost.com/*, Dec 2017. [Online]. Available: https://www.huffpost.com/entry/use-google-docs-to-collab_b_844192?guccounter=1&guce_referrer=aHR0cHM6Ly93d3cuZ29vZ2xlLmNvbS8&guce_referrer_sig=AQAAANzywb_

bhJ78vkWceRBDU5mPk2FYfO65cPrYaGgRvmmBhP2LF1JAvwdcYfJO84yQwbv_
cgvDE_4mmCQZa0NWWmLXN84tzD3LtKkOKFEuLio04qwglaqNBpF4-pMlmnGu1kOuhbKzP8
IoX

[37] W. Gordon, "Use google docs for powerful, collaborative school note taking," Jun 2011. [Online]. Available: https://lifehacker.com/use-google-docs-for-powerful-collaborative-school-note-5788145

[38] R. Kee, "The 10 best note-taking apps in 2019 – evernote, notion, and more," Jul 2019. [Online]. Available: https://collegeinfogeek.com/best-note-taking-apps/

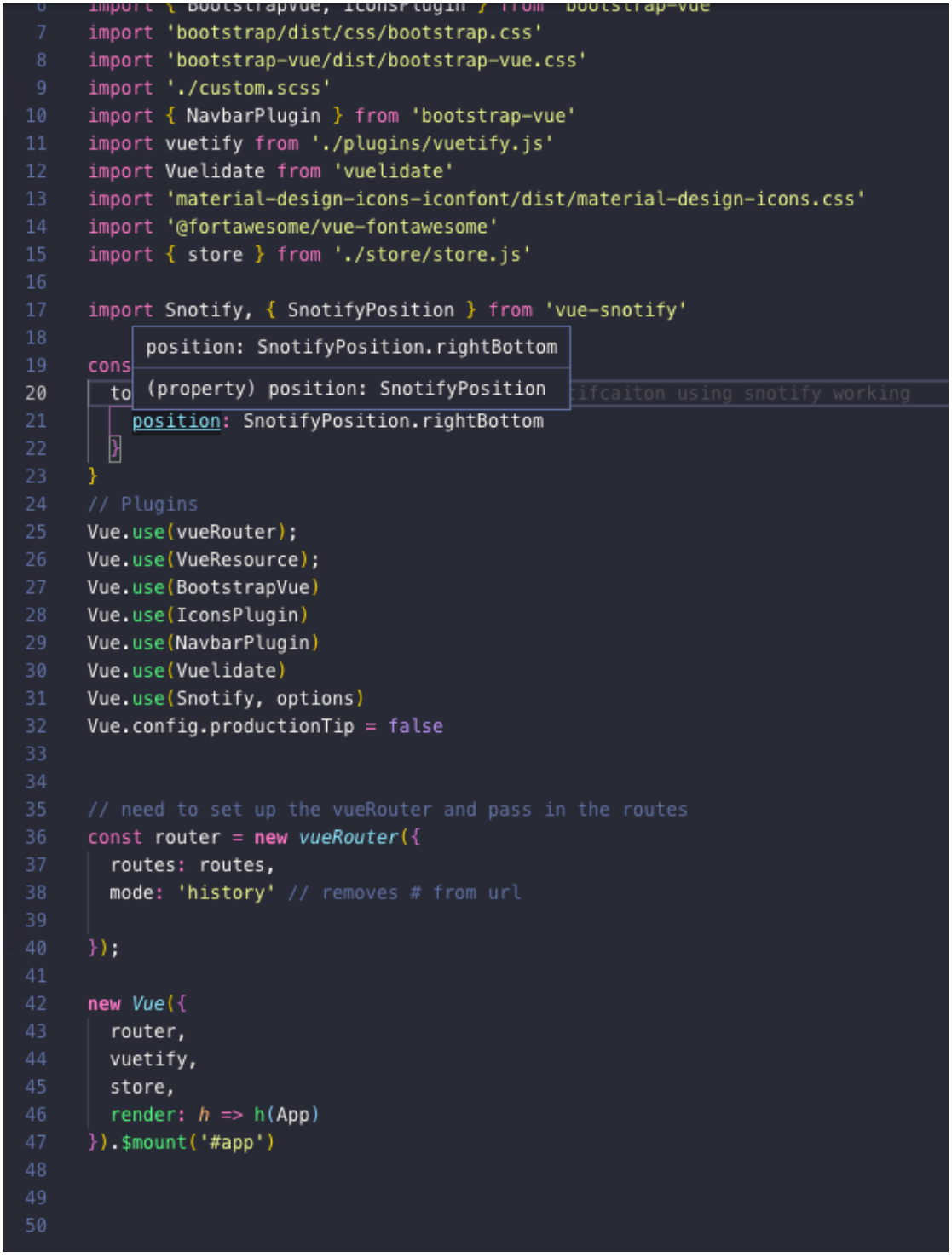
[39] F. D'Alessio, "Top 10 note-taking apps for 2017," Mar 2017. [Online]. Available: https://medium.com/the-mission/top-10-note-taking-apps-for-2017-609c3fb64115

[40] "Evernote - the note taking application." [Online]. Available: https://help.evernote.com/

[41] "Ulysses - the ultimate writing app for mac, ipad and iphone." [Online]. Available: https://ulysses.app/features/

Appendix A

Code Snippets



```
7   import 'bootstrap/dist/css/bootstrap.css'
8   import 'bootstrap-vue/dist/bootstrap-vue.css'
9   import './custom.scss'
10  import { NavbarPlugin } from 'bootstrap-vue'
11  import vuetify from './plugins/vuetify.js'
12  import Vuelidate from 'vuelidate'
13  import 'material-design-icons-iconfont/dist/material-design-icons.css'
14  import '@fortawesome/vue-fontawesome'
15  import { store } from './store/store.js'
16
17  import Snotify, { SnotifyPosition } from 'vue-snotify'
18
19  cons
20    to                                                    tifcaiton using snotify working
21      position: SnotifyPosition.rightBottom
22    }
23  }
24  // Plugins
25  Vue.use(vueRouter);
26  Vue.use(VueResource);
27  Vue.use(BootstrapVue)
28  Vue.use(IconsPlugin)
29  Vue.use(NavbarPlugin)
30  Vue.use(Vuelidate)
31  Vue.use(Snotify, options)
32  Vue.config.productionTip = false
33
34
35  // need to set up the vueRouter and pass in the routes
36  const router = new vueRouter({
37    routes: routes,
38    mode: 'history' // removes # from url
39
40  });
41
42  new Vue({
43    router,
44    vuetify,
45    store,
46    render: h => h(App)
47  }).$mount('#app')
48
49
50
```

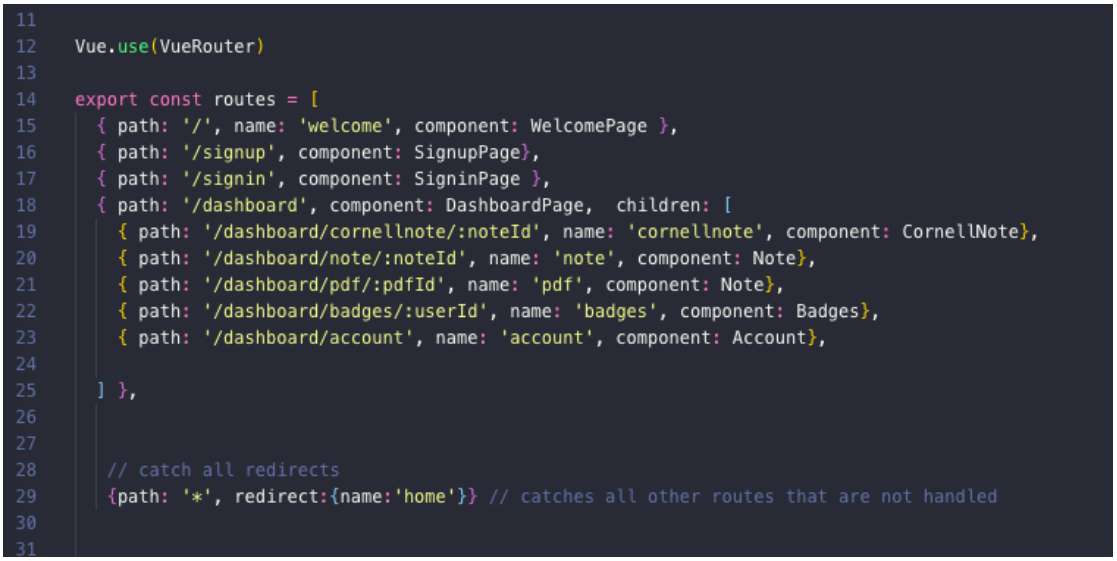
position: SnotifyPosition.rightBottom

(property) position: SnotifyPosition

FIGURE A.1: Front End - main.js - project imported libraries and App creation function

```
You, a month ago | 1 author (You)
<template>
  <div>
    <v-container>
      <div id="badgeDisplay">


        <!-- Title   -->
        <v-row id="titleRow">
          <h2 class="titleHeading">Badges & Rewards</h2>
        </v-row>
        <!-- progress -->
        <v-row id="progressRow">
          <div class="progressBar">
            <div class="loading-bar">
              <div class="percentage" :style="{'width' : percentage + '%'}">
              </div>
            </div>
          </div>
        </v-row>

        <!--  Stats -->
        <v-row id="statsRow">
          <h4 id="statsHeading">Statistics</h4>

          <v-col id="statScores">
            <v-row><h6>Scores</h6></v-row>
            <v-row><p>Total Score: <strong>{{userStats.Points}}</strong></p></v-row>
            <v-row><p>User Level: {{userStats.UserLevel}}</p></v-row>
            <v-row><p>Points to Next Level: {{userStats.PointsToNextLevel}}</p></v-row>
          </v-col>

          <v-col id="statCreated">
            <v-row><h6>Added</h6></v-row>
            <v-row><p>Notes Added: {{userStats.NotesCreated}}</p></v-row>
            <v-row><p>Cornell Added: {{userStats.CornellNotesCreated}}</p></v-row>
            <v-row><p>Cues Added: {{userStats.CuesCreated}}</p></v-row>
          </v-col>

```

FIGURE A.2: Front End - Badges.Vue - progress bar and scoring components



```

Vue.use(VueRouter)

export const routes = [
  { path: '/', name: 'welcome', component: WelcomePage },
  { path: '/signup', component: SignupPage},
  { path: '/signin', component: SigninPage },
  { path: '/dashboard', component: DashboardPage,  children: [
    { path: '/dashboard/cornellnote/:noteId', name: 'cornellnote', component: CornellNote},
    { path: '/dashboard/note/:noteId', name: 'note', component: Note},
    { path: '/dashboard/pdf/:pdfId', name: 'pdf', component: Note},
    { path: '/dashboard/badges/:userId', name: 'badges', component: Badges},
    { path: '/dashboard/account', name: 'account', component: Account},

  ] },


  // catch all redirects
  {path: '*', redirect:{name:'home'}} // catches all other routes that are not handled

```

FIGURE A.3: Front End - routes.js - shows the routes handled

```
import Vue from "vue"
import Vuetify from "vuetify"

Vue.use(Vuetify)

export default new Vuetify({
  icons: {
    iconfont: 'md',  // 'mdi' || 'mdiSvg' || 'md' || 'fa' || 'fa4'
  },
  theme: {
    dark: false,
  },
  themes: {
    light: {
      primary: "#4682b4",
      secondary: "#b0bec5",
      accent: "#8c9eff",
      error: "#b71c1c",
    },
    dark: {
      primary: "#4682b4",  // TODO: change all these
      secondary: "#b0bec5",
      accent: "#8c9eff",
      error: "#b71c1c",
    },
  },
})
```

~/Documents/Projects/FYP/FYP_frontend • Contains emphasized items

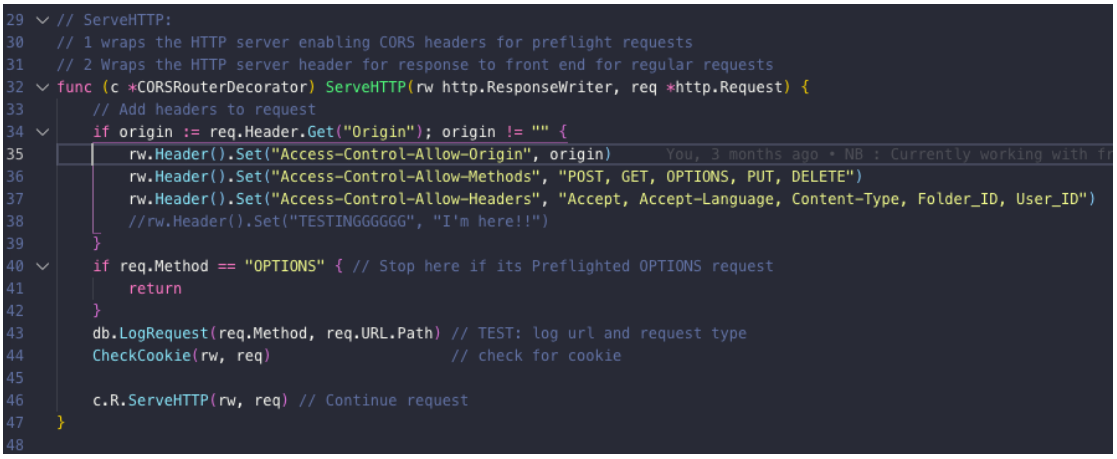
FIGURE A.4: Front End - styles.js - Main styling colors for platform

```
//console.log(formData)
const fyp_backend = this.$store.getters.backendUrl
const dataType = 'register'
var url = fyp_backend + dataType
//console.log(url)
this.$http.post(url, this.user)
  .then(response =>{
    console.log(response.body.UUID);
    this.userCreated = true
    // Go to Dashboard
    router.replace('dashboard')
  }, error => {
      if (error.status == 409){
        console.log("User Already Exists");
        this.alert.message = "Email already in use: Please Sign in or use another email to register"
        this.showAlert = true // display alert
```

FIGURE A.5: Front End - user.js - Section of API request for Logging in User

```
  },
  getters: {
    sidebar: state => state.sidebar,
    backendUrl: state => state.backendUrl,
    userId: state => state.loggedInUser,
    selectedItemId: state => state.selectedItemId,
    selectedItemType: state => state.selectedItemType,
    cn: state => state.cn,
    showNote: state => state.showNote,
    showCn: state => state.showCn,
    note: state => state.note,
    folders: state => state.folders,
    currentFolder: state => state.currentFolder,
    tags: state => state.tags,
    userStats: state => state.userStats,
  },
  mutations: {
    loginUser (state, value) {
      state.userId = value
    },
    toggleSidebar (state) {
      state.sidebar = !state.sidebar // toggle sidebar open/close
    },
    showNote (state) {
      state.showCn= false
      state.showNote = !state.showNote // toggle display note
```

FIGURE A.6: Front End - store.s - Section of Vuex store for shared component properties



```
// ServeHTTP:
// 1 wraps the HTTP server enabling CORS headers for preflight requests
// 2 Wraps the HTTP server header for response to front end for regular requests
func (c *CORSRouterDecorator) ServeHTTP(rw http.ResponseWriter, req *http.Request) {
    // Add headers to request
    if origin := req.Header.Get("Origin"); origin != "" {
        rw.Header().Set("Access-Control-Allow-Origin", origin)      You, 3 months ago • NB : Currently working with fr
        rw.Header().Set("Access-Control-Allow-Methods", "POST, GET, OPTIONS, PUT, DELETE")
        rw.Header().Set("Access-Control-Allow-Headers", "Accept, Accept-Language, Content-Type, Folder_ID, User_ID")
        //rw.Header().Set("TESTINGGGGGG", "I'm here!!")
    }
    if req.Method == "OPTIONS" { // Stop here if its Preflighted OPTIONS request
        return
    }
    db.LogRequest(req.Method, req.URL.Path) // TEST: log url and request type
    CheckCookie(rw, req)                    // check for cookie

    c.R.ServeHTTP(rw, req) // Continue request
}
```

FIGURE A.7: Back End - API request Middle ware function

```go
// UserStats - all stats for user scores
You, a month ago | 1 author (You)
type UserStats struct {
    UserID              string `db:"user_id" json:"UserID"`
    Points              int    `db:"points" json:"Points"`
    NotesCreated        int    `db:"notes_created" json:"NotesCreated"`
    CornellNotesCreated int    `db:"cornell_notes_created" json:"CornellNotesCreated"`
    CuesCreated         int    `db:"cues_created" json:"CuesCreated"`
    NotesShared         int    `db:"notes_shared" json:"NotesShared"`
    CornellNotesShared  int    `db:"cornell_notes_shared" json:"CornellNotesShared"`
    ReviewsCompleted    int    `db:"reviews_completed" json:"ReviewsCompleted"`
    CuesReviewed        int    `db:"cues_reviewed" json:"CuesReviewed"`
}
```

FIGURE A.8: Back End - User Model Struct

```go
/*
 * Tags
 */
router.HandleFunc("/tags", api.GetTags).Methods("GET", "OPTIONS")
router.HandleFunc("/tag", api.CreateTag).Methods("POST", "OPTIONS")
router.HandleFunc("/tag", api.UpdateTag).Methods("PUT", "OPTIONS")
router.HandleFunc("/tag", api.DeleteTag).Methods("DELETE", "OPTIONS")

/*
 * Cornell Notes
 */
router.HandleFunc("/cornellnote", api.GetCornellNote).Methods("GET", "OPTIONS")
router.HandleFunc("/cornellnote", api.CreateCornellNote).Methods("POST", "OPTIONS")
router.HandleFunc("/cornellnote", api.UpdateCornellNote).Methods("PUT", "OPTIONS")
router.HandleFunc("/cornellnote", api.DeleteCornellNote).Methods("DELETE", "OPTIONS")
router.HandleFunc("/cornellnote/cue", api.AddCornellNoteCue).Methods("POST", "OPTIONS")
router.HandleFunc("/cornellnote/cue", api.UpdateCornellNoteCue).Methods("PUT", "OPTIONS")
router.HandleFunc("/cornellnote/cue", api.DeleteCornellNoteCue).Methods("DELETE", "OPTIONS")
router.HandleFunc("/cornellnote/summary", api.UpdateCornellNoteSummary).Methods("PUT", "OPTIONS")

/*
 * Notes
 */
router.HandleFunc("/note", api.GetNote).Methods("GET", "OPTIONS")
router.HandleFunc("/note", api.SaveNote).Methods("POST", "OPTIONS")
router.HandleFunc("/note", api.UpdateNote).Methods("PUT", "OPTIONS")
router.HandleFunc("/note", api.DeleteNote).Methods("DELETE", "OPTIONS")

/*
 * Badges & Scores
 */
router.HandleFunc("/badges", api.GetBadges).Methods("GET", "OPTIONS")
router.HandleFunc("/badges/stats", api.GetUserStats).Methods("GET", "OPTIONS")
```

FIGURE A.9: Back End - Sample of API routes handled

```go
// UpdateCornellNoteCue - update cornell cue
func UpdateCornellNoteCue(w http.ResponseWriter, r *http.Request)
    var cue m.CornellCue
    // Get Cue details from body and user id from header
    userID := r.Header.Get("user_id")
    err := json.NewDecoder(r.Body).Decode(&cue)
    db.Check(err)
    cue.DateEdited = time.Now()
    // Update DB with new Cue information
    res := q.UpdateCornellNoteCue(cue, userID) // run db query

    w.Header().Set("Content-Type", "application/json")
    json.NewEncoder(w).Encode(res)
}
```

FIGURE A.10: Back End - Sample of API Request

```go
func UpdateCornellNoteCue(cue m.CornellCue, userID string) string {

    res := ""
    conn := db.CreateConn()
    tx, err := conn.Begin()
    db.Check(err)
    stmt, err := tx.Prepare("UPDATE cornell_cues cc " +
        "JOIN sys.cornell_users cu ON cc.cornell_note_id = cu.cornell_note_id " +
        "SET cc.cue = ?, cc.answer = ?  " +
        "WHERE cu.user_id = ? AND cc.id = ?;")
    if err != nil {
        fmt.Println("UpdateCornellNoteCue Error 1", err)
        tx.Rollback()
        return "Error"
    }
    defer stmt.Close()
    if _, err := stmt.Exec(cue.Cue, cue.Answer, userID, cue.ID); err != nil {
        fmt.Println("UpdateCornellNoteCue Error 2", err)      You, a few seconds ago • Uncommitted changes

        tx.Rollback() // return an error too, we may want to wrap them
        return "Error"
    }
    stmt, err = tx.Prepare("UPDATE cornell_notes cn JOIN cornell_users cu ON cn.id = cu.cornell_note_id " +
        "SET cn.date_edited = ? WHERE cn.id = ? AND cu.user_id = ?")
    if err != nil {
        fmt.Println("UpdateCornellNoteCue Error 3", err)

        tx.Rollback()
        return "Error"
    }
    defer stmt.Close()

    if _, err := stmt.Exec(cue.DateEdited, cue.CornellNoteID, userID); err != nil {
        fmt.Println("UpdateCornellNoteCue Error 4", err)

        tx.Rollback() // return an error too, we may want to wrap them
        return "Error"
    }
```

FIGURE A.11: Back End - Sample of DB Transaction Request

Appendix B

Wireframe Models