University of Nottingham Ningbo China
School of Computer Science

Academic Year 2015/16 Spring Semester

GRP-DT Group Final Report

Enhanced Multimedia Note-taking Research & Support App Development

Supervisor: Dr Dave Towey

Group Members:
Yichen PAN; ID: 6514897
Shihang WANG; ID: 6514904
Yiru JIANG; ID: 6514889
Jiahui DENG; ID: 6514883
Livia FLORENSIA; ID: 6519837

April 29, 2016

Contents

1 Introduction

1.1 Project Description

In 2014, a number of leading researchers at The University of Nottingham Ningbo China (UNNC) embarked on a project to examine how well the university would be able to support student learning in the future. Over 2014-2015, a team of UNNC Computer Science (CS) and Computer Science and Management (CSM) students worked as a software engineering (SE) team (called gp-dave) and investigated the challenge and implemented a solution for the research team. In 2015-2016, a new SE team (called GRP-DT) was required to look again at the needs of research team, and then examine and evolve the software produced by gp-dave.

We worked together to complete a full cycle of the software engineering process, resulting in delivery of a revised tool, as requested by the research project team. We have gone through a complete requirements engineering process to identify the revised SE project requirements and conducted series of testing to ensure the software quality.

1.2 Goals

The target goal of this project was to deliver, on time, a new tool which will support the research team's project. We have been reminded that a target use of this tool is to support not only students taking notes, but, more importantly, the research team's research project: the tool has the dual purpose of both assisting student note-taking, and facilitating the research.

1.3 Background

1.3.1 Research on Note-taking Application

At the beginning, each of us did some research on several note-taking applications as shown below.

1. **OneNote**
   It can be used anywhere like personal computer (PC), mobile devices, and tablets. Since it is synced on-line, it can be shared among many people, and the notes that are accessible to us are kept up to date. It also has a function to clip a part of screen and render it as an image. We can also extract e-mails to our OneNote notebook by sending an email to me@onenote.com (OneNote, 2015).

2. **EverNote**
   With the friendly interface of EverNote, users are more likely to use it recording whatever happens in daily life. Comparing with OneNote, it has more other functions like maps. However, the application (app) does not offer as many formatting options as OneNote (EverNote, 2015).

3. **iDailyDiary**
   It's more like a virtual diary app. It does have the basic essential functions like inserting graphics and URLs. It has a function to export our notes into an HTML format, so that they can become web pages. It is also able to protect our notes with passwords. It supports the characters of many languages. That is all for the free version, but if

users upgrade it to the professional version, it has even more functions than that, such as dictionary, the alignment of picture between text, themes, print function, smileys, tables, audio and video (Splinterware, 2015).

4. **Google Keep**
   It can convert sound recording into text, and it also can search for keywords in the note, even when the keyword is not stored in the form of text (Google Keep, 2015). However, it cannot be used without VPN in mainland China.

After background research on popular note-taking applications listed above, we found none of them could satisfy the requirements of our stakeholders, such as tracking the activities of users, dragging and dropping information from websites and inserting multimedia into notes. In this case, we decided to design a new note-taking application based on requirements from stakeholders.

1.3.2 2014 - 2015 Project

Before we gathered requirements from stakeholders, we participated in a presentation about last year team's note-taking app, conducted by Melissa and Wenyu who were the leaders of last year's project. The last year's app fulfiled many functions specified in last year's Software Requirements Specification (SRS) Document (Lan Hing Po *et al.*, 2014).

- File Manager: Users can decide whether they want to make a quick note or a notebook. If users create a notebook, then the app will create an empty folder which is used to classify note files.
- Open documents and images: Users can open documents, including PDF and Microsoft Office document, and images within the app.
- Access the Internet: Users can search on the Internet within the app and 'drag and drop' information into notes.
- Play Video: Users can watch videos thorough the app.
- Audio and Video recording: The app can be used to record audio and video. There is a button used to start the recording, and it will generate a window for recording. Users can stop the recording by closing the recording window.

The product outcome for gp-dave is called the Notebook 1.0 (shown in Figure 1). They realized some of requirements, but the app was only partially developed. Therefore, Melissa and Wenyu continued working on the note-taking app during the summer holiday as an internship. They added several functions like being able to add tags to the individual notes, and users being able to search for the tagged keywords, making it easier for the users to look for their notes. The product outcome for the summer intern project is called the Notebook 1.1.

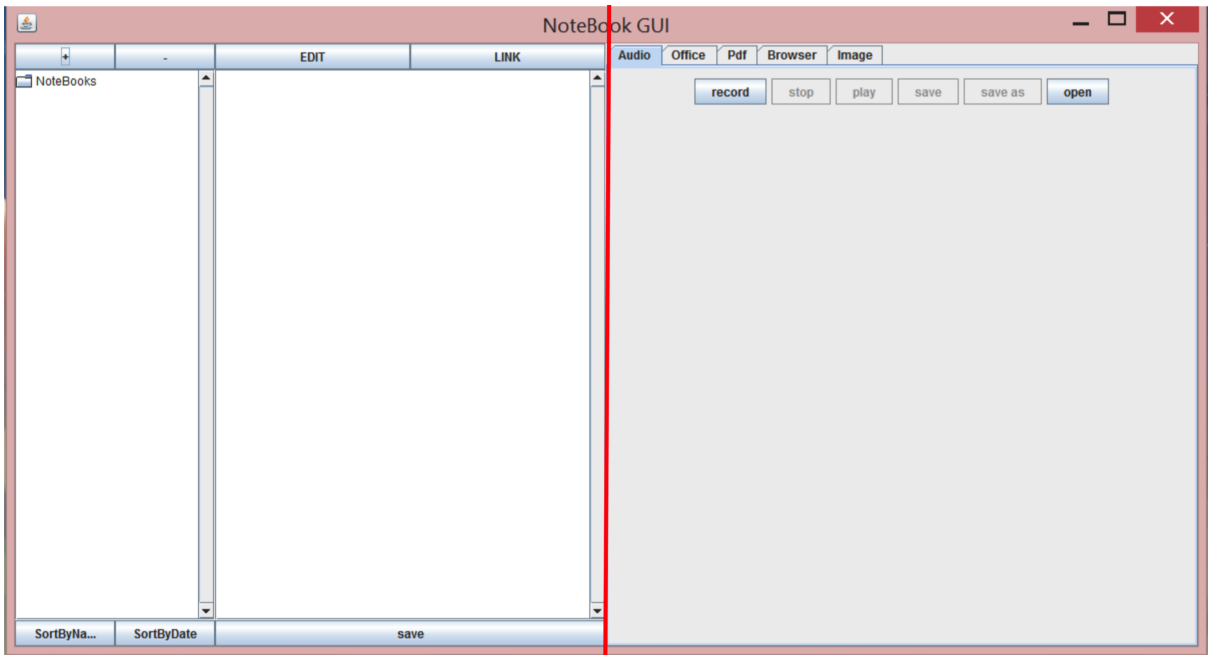


Figure 1: Notebook 1.0

According to the gp-dave team, there were still some requirements not fulfilled in their note-taking app as listed below (Lan Hing Po *et al.*, 2015):

- Text editing functions
- A dictionary, complete with the translator
- Sharing notes

A more detailed evaluation of last year's project and verification of last year's app can be found in **Appendix A**.

1.4 Roles of Each Member

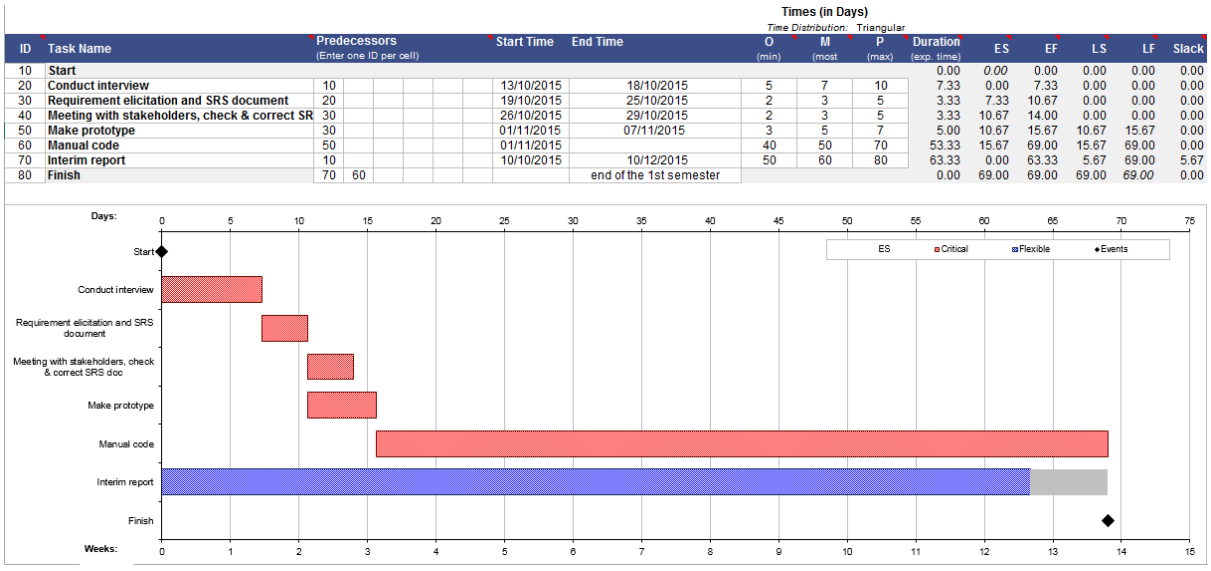
According to Belbin (1981) , a six-person team mainly consists of six kinds of roles, including project manager, technical leader, researcher, implementer, tester, and documentation recorder. The project manager is related to the shaper or coordinator of the whole team. The technical leader is responsible for solving technical issues of the development team. The researcher is more regarded as a specialist, such as a UI designer. The implementer and tester are the team workers for implementing specific tasks and the documentation recorder takes in charge of recording and managing the documentation.

After conducting the Myers Briggs personality tests and knowing the best abilities and weaknesses of each member we delegated roles as following (Briggs and McCaulley, 1992).

Name	Role	Description
Yichen	Team leader	Lead the team
	Scrum master	Arrange daily meetings, track the backlog and motivate the group
	Version manager	Make sure the version of the application updated
	Back designer	Realize the functionalities
Shihang	UI designer	In charge of the interface-related matters
	Back designer	Realize the functionalities
Yiru	Editor	Edit our meeting minutes, agenda, reports, and other documents
	Documenter	Collect, archive and backup all documents
	Front designer	Design the interface
Jiahui	Editor	Edit our meeting minutes, agenda, reports, and other documents
	Sender	Send all group emails including meeting minutes and agenda
	Front designer	Design the interface
Livia	Idea generator	Think of creative solutions for the group project
	Website manager	Maintain the website
	Front designer	Design the interface

1.5 Time Plan

Due to the fact that our project lasted for one whole academic year, it was divided into several stages. Some works have been completed in the autumn semester: conducting interviews with three stakeholders, requirements elicitation, writing the SRS Document, checking the SRS Document with stakeholders and implementing a part of the coding work. The spring semester was dedicated to implementing all functionality to release an executable application of this project. Below is the project schedules of the two semesters. The autumn semester time plan was based on the critical path analysis, but the spring semester time plan was divided into several sprints because we adopted Scrum later.



								Times (in Days)								
								Time Distribution: Triangular								
ID	Task Name	Predecessors (Enter one ID per cell)						Start Time	End Time	O (min)	M (most	P (max)	Duration (exp. time)	ES	EF	LS	LF	Slack
10	Start												0.00	*0.00*	0.00	0.00	0.00	0.00
20	Conduct interview	10						13/10/2015	18/10/2015	5	7	10	7.33	0.00	7.33	0.00	0.00	0.00
30	Requirement elicitation and SRS document	20						19/10/2015	25/10/2015	2	3	5	3.33	7.33	10.67	0.00	0.00	0.00
40	Meeting with stakeholders, check & correct SR	30						26/10/2015	29/10/2015	2	3	5	3.33	10.67	14.00	0.00	0.00	0.00
50	Make prototype	30						01/11/2015	07/11/2015	3	5	7	5.00	10.67	15.67	10.67	15.67	0.00
60	Manual code	50						01/11/2015		40	50	70	53.33	15.67	69.00	15.67	69.00	0.00
70	Interim report	10						10/10/2015	10/12/2015	50	60	80	63.33	0.00	63.33	5.67	69.00	5.67
80	Finish	70	60						end of the 1st semester				0.00	69.00	69.00	69.00	*69.00*	0.00

Figure 2: Critical Path for the Autumn Semester

No.	Event	Time	Deadline
Sprint 1	Video and audio recording function	15/02/2016--28/02/2016	
	Beautify the drag and drop part	15/02/2016--28/02/2016	
	Design the logo image and slogan	15/02/2016--28/02/2016	
	Save and load function	15/02/2016--28/02/2016	
Sprint 2	Track the activities of users and to files.	29/02/2016--13/03/2016	
	Change the appearance	29/02/2016--13/03/2016	
	Drag and drop	29/02/2016--13/03/2016	
	Draft for the final report	29/02/2016--13/03/2016	
	Splitter	29/02/2016--13/03/2016	
	Share notes	29/02/2016--13/03/2016	
Sprint 3	Update the apperance	14/03/2016--27/03/2016	
	Share online	14/03/2016--27/03/2016	
	Make recordings matched with notes	14/03/2016--27/03/2016	
	Write final report	14/03/2016--27/03/2016	
	Print note function	14/03/2016--27/03/2016	
	Export notes to PDF	14/03/2016--27/03/2016	
	Tagging function	14/03/2016--27/03/2016	
Sprint 4	Edit final report	28/03/2016--10/04/2016	
	Super note	28/03/2016--10/04/2016	
	Realise adding comments function	28/03/2016--10/04/2016	
	Note template	28/03/2016--10/04/2016	
	Add margins part	28/03/2016--10/04/2016	
Sprint 5	Testing and release the app	11/04/2016--24/04/2016	29/04/2016
	Open day presentation	11/04/2016--24/04/2016	04/05/2016
	Finish final report	11/04/2016--24/04/2016	29/04/2016

Figure 3: Scrum Sprint Plan for the Spring Semester

1.6 Summary of Work

1.6.1 Autumn Semester

After the first two months, we finished the background research, requirements engineering and produced a complete SRS document (shown in **Appendix B**). During the requirements engineering stage, we adopted plan-driven development approach to make a complete list of user and system requirements. We held three interviews for requirements elicitation, and another meeting with each of them for requirements validation. Then we designed the software based on last year's project and SRS document using UML models (Lan Hing Po *et al.*, 2014). After that, we changed to Scrum development approach to make incremental deliveries so that we could get immediate feedback from stakeholders. In the last month of autumn semester, we went through two sprints and realized 70% of the functionality, including rich-text function, sharing notes, and exporting notes as PDF based on Java Swing.

Figure 4 below shows the product we made by the end of the autumn semester and we named it as Notebook 2.0.

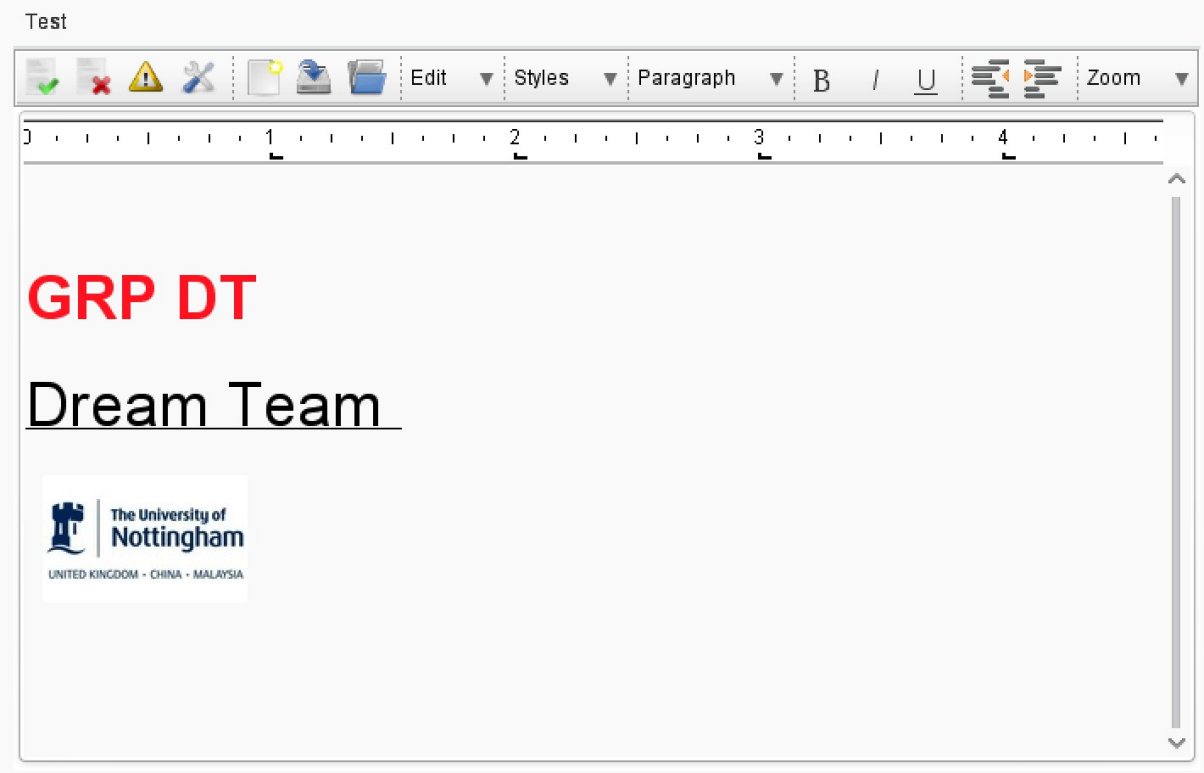


Figure 4: Notebook 2.0

1.6.2 Winter Holiday

At the last week of Autumn semester, we held a meeting with one of our stakeholders, Dr. Smith (the name is pseudonym), to demonstrate our progress and show our product. However, Dr. Smith was not satisfied with the appearance and functionality. He showed a website, Storify.com, where the original idea of developing our note-taking app came from (Storify, 2016). After investigating into Storify.com, we found it is infeasible and difficult to realize what the stakeholders really wanted using Java Swing. In terms of this situation, we went through a complete change management process (shown in section 4.6) to analyse the proposed change and find a feasible solution.

Finally, we decided to use NW.js together with MEAN Stack to develop a new cross-platform application (shown in section 4.3). Since we were not familiar with web programming, we spent the whole winter holiday studying and exploring. In addition, we held online meetings using Skype every week in order to communicate and maintain the progress. At the end of the winter holiday, we finished the framework and basic configuration of the product.

1.6.3 Spring Semester

Since we decided to change the implementation strategies and learnt the web programming knowledge as the preparation for spring semester, we started building functions at the beginning of the semester. Our progress was generally following the Scrum sprint plan (Figure 3) we made. By the end of spring semester, we released a complete new note-taking application, called Quick

Note (Figure 5, 6).

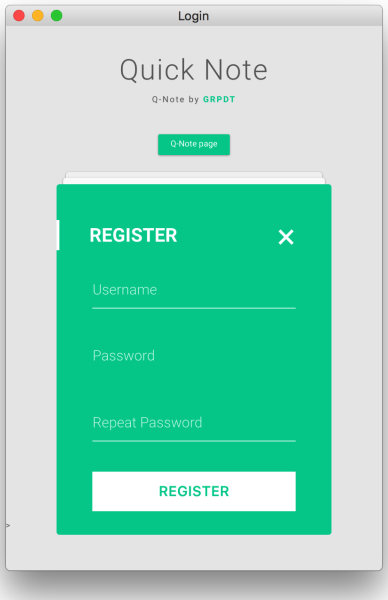


Figure 5: The login Interface of Quick Note

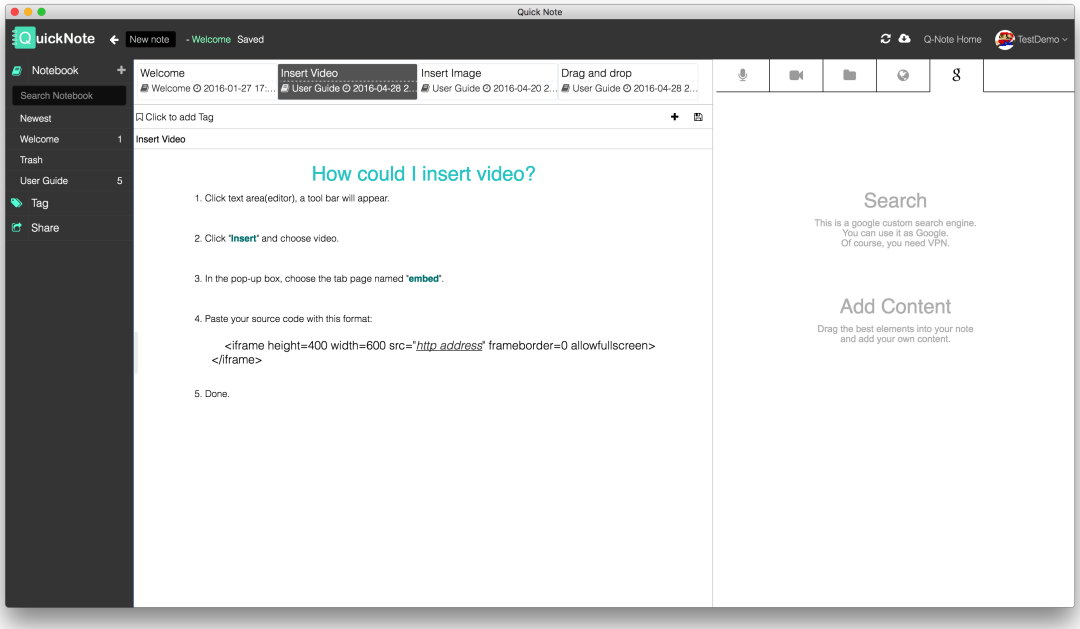


Figure 6: The Main Interface of Quick Note

Although we have realized most of the prioritized requirements from the stakeholders, there are several essential and useful functionalities unfulfilled due to the lack of development time. The checklist of each requirement is listed below: In the last month of spring semester, we started

Requirements	Status
1. The tutors shall be able to track the students' activities.	Implemented
2. The students should be able to easily share the notes with tutors and other students.	Implemented
3. The tutors should be able to view the inserted files in the notes that students shared.	Partially implemented
4. The tutors should be able to extract the useful part of students' notes and integrate them into a super note.	Unimplemented
5. The students should be able to drag information from websites into notes, and together with the URL.	Partially implemented
6. The students should be able to record the lectures when taking notes.	Implemented
7. The students should be able to add tags to each note and search the note classified by tags.	Implemented
8. The students shall be able to put different notes under a single notebook and view them by date order.	Implemented
9. The students shall have access to the internet from within the app using different search engines.	Implemented
10. The tutors should be able to comment on students' notes.	Unimplemented
11. The students shall change the setting options of the app (size of each part).	Implemented
12. The tutors should be able to build the template for notes by themselves for different modules.	Implemented
13. The students should be able to make margins to set a certain structure for their notes.	Unimplemented
14. The students should be able to build tables inside the notes easily.	Implemented
15. The students should be able to open documents within the same window in the app.	Partially implemented
16. The students should be able to create a new note easily.	Implemented
17. The students should be able to change colors of words and make specific words in bold or italic.	Implemented
18. The students should be able to export notes to PDF.	Partially implemented
19. The students should be able to print the notes within the app easily	Implemented

Figure 7: The Checklist of Requirements

testing our application. We tested the application on three different levels: development testing, release testing and user testing (shown in section 5).

2 Requirements Engineering

Requirements engineering refers to the process of defining, documenting and maintaining requirements (Nuseibeh and Easterbrook, 2000). According to Sommerville (2011), there are three essential activities including requirements elicitation, specification and validation.

2.1 Requirements Elicitation

Requirements elicitation is a process where we identify user requirements based on a detailed understanding of the future app. In our project, we referred to 2014 - 2015 project's SRS document and held structured interviews with stakeholders to gather requirements (Lan Hing Po *et al.*, 2014).

2.1.1 Interviews

We conducted one interview with each stakeholder. The main purpose of the interviews was to talk with stakeholders to get further understanding in the functionality of the app and gather the requirements. The main stakeholders are Dr. Smith, Dr. Brown and Dr. Williams (these names are pseudonyms). Before the interviews, the participant information sheet, shown in **Appendix C**, was prepared to help stakeholders better understand the project. The team members and the stakeholders also signed the consent forms (shown in **Appendix D**) under the university regulation based on the ethics checklist (shown in **Appendix E**). The interviews were recorded, and we documented the information as requirements according to the records and interview minutes.

2.2 Requirements Specification

Requirements specification is the process of writing down the requirements in the SRS Document. In our project, after the interviews we produced an SRS Document, which contains lists of user requirements and system requirements as well as a use case diagram and the background information of the project and the GRP-DT team. The detailed user requirements and system requirements can be found in the SRS Document as attached in **Appendix B**. Some explanation of the requirements are elaborated below.

2.2.1 User Requirements

One additional user requirement that the stakeholders requested this year is the app shall enable tutors to track the students' activities. Tutors could use this function to see what notes students are taking in the lecture and what information students are searching on-line, and track students' time on different activities.

2.2.2 System Requirements

This application shall run on any desktop computer or laptop, including Windows 7, 8, 8.1, 10 and Mac OS.

2.3 Requirements Validation

Requirements validation is the procedure for checking whether the requirements meet what the stakeholders want. It is essential because according to Sommerville (2011) fixing a requirements error after delivery may cost up to 200 times of the cost of fixing an implementation error. In our project, we were divided into two separate requirements teams to check for errors and inconsistencies based on a list of criteria shown below:

- Requirements should be necessary.
- Requirements should be feasible.
- Requirements should be complete.
- Requirements should be clear.
- Requirements should be consistent.

After that, we conducted stakeholder validations by holding meetings described as followed.

2.3.1 Stakeholders Validation Meetings

In order to validate that all requirements are recorded correctly, we held one meeting with each stakeholder individually to confirm the requirements, gather the priority of the requirements from the stakeholders, and after that we improved the SRS Document as attached in **Appendix B**.

3 Software Design

3.1 UML Models

The Unified Modelling Language (UML) is a modelling language in software engineering, which is commonly used to visualize the design a the system. In our project, we produced the following UML diagrams to convert requirements specification to object-oriented design.

3.1.1 Use Case Diagram

A use case diagram is one kind of UML diagrams that shows the interaction between the user and the system under different scenarios. The use case diagram, shown in Figure 8, is based on the scenario described in **Appendix B**.

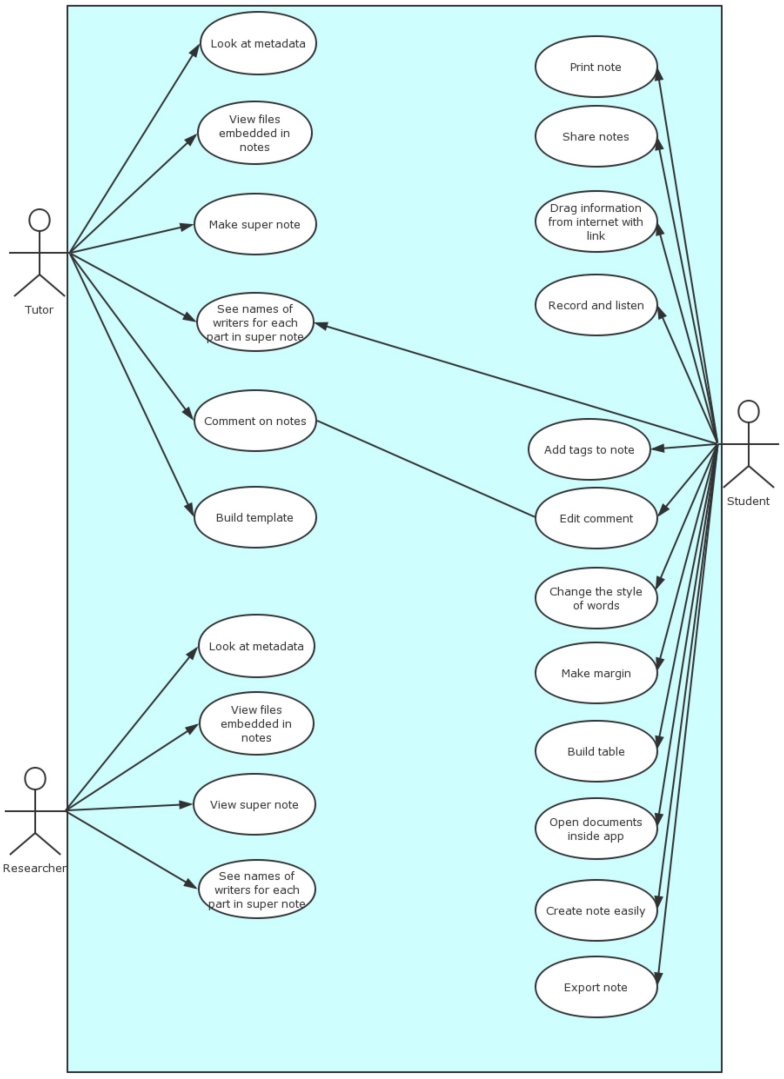


Figure 8: The Use Case Diagram

3.1.2 Class Diagram

A class diagram is one kind UML diagrams that shows the structure of the system and the relationship between object. Figure 9 is a class diagram which contains all necessary classes of our app. 'Note' is placed in the centre because the main function of this project is note-taking.

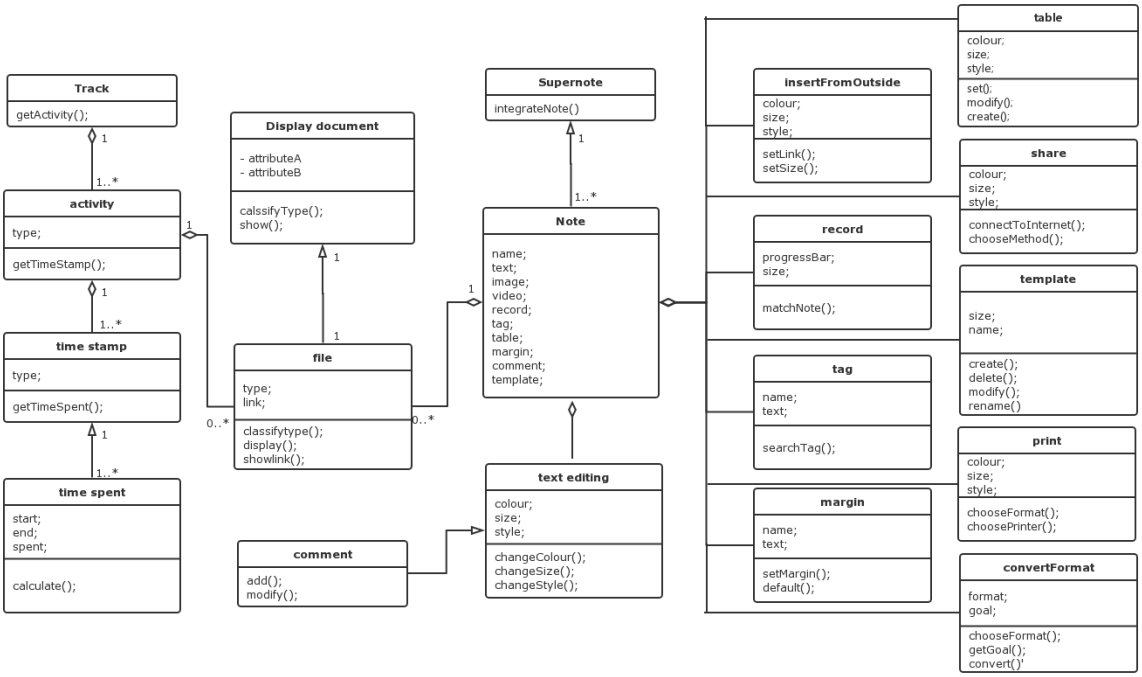


Figure 9: The Class Diagram

4 Implementation

4.1 Original Implementation Strategy: Java Swing

In the middle of development stage, we decided to stop developing based on Java Swing because the shortcomings of it prevent us from progressing. First of all, it can be very slow because of the nature of Java Swing rendering mechanism. Secondly, there is no built-in support for animation, which makes it difficult to make user-friendly interface and unable to satisfy stakeholders, since they want advanced interactive interfaces. Thirdly, because Swing components base on the native components of the operating system, the product looks different on MacOS, Linux and windows. Finally, due to the fact that Swing is a graphic user interface (GUI) widget toolkit for Java, it requires the Java Runtime Environment (JRE) to enable the computer to run the program. The premise of 'Write once, run anywhere' that the computer has to configure JRE in advance is regarded annoying and inconvenient for layman by one of stakeholders (Computer Weekly, 2002).

4.2 Other Potential Implementation Strategies

While we investigated into the potential implementation strategies, we found several practical ways.

4.2.1 Python + QT = PyQT

PyQt is a Python binding of the cross-platform GUI toolkit Qt, which is a widely-used cross-platform application framework (Mobidev, 2015). It is well-designed and has a huge user base, which indicates its power. However, the relative complexity of distribution of applications for Windows and lack of familiarity with Python and QT prevent us from using it.

4.2.2 Adobe Integrated Runtime

Adobe Integrated Runtime (AIR) is a cross-platform environment created by Adobe (Adobe, 2016). It allows developers to use Adobe Flash, Apache Flex, HTML/CSS and AJAX for building Web applications. Although the use of web techniques makes it easier to build GUI and animation, no support for Linux prevents us from using it.

4.3 Final Implementation Strategy

4.3.1 NW.js

NW.js is an application based on Google Chrome core and node.js (NW.js, 2016). It offers an complete assortment of features and provides a good community support. It allows developing cross-platform desktop applications using web programming techniques. After packaging all codes into NW.js and setting up the configuration file, a web application could be easily packaged into a native desktop application. Although NW.js is relatively not stable due to the reason that it is still in the developing stage, it is the best option considering the relatively low learning cost and better capability. The following decision for us to make is which specific set of web techniques should we use and how to build the web application.

4.3.2 MEAN vs LAMP (Linux, Apache, MySQL, PHP)

Concerned with techniques for building dynamic web sites and web applications, there are two popular sets of techniques available. The first one is LAMP stack, which is an archetypal model of web service solution stacks and consists of four open-source components: the Linux operating system, the Apache HTTP Server, the MySQL relational database management system (RDBMS), and the PHP programming language (Rajput, 2015). Due to the popularity and large user-base of LAMP, it is highly secure and constantly updated to incorporate new updates and improvements. Although LAMP provides one of the most efficient and popular ways of developing web applications because of its customization, flexibility and cost effectiveness, it is outdated to some extent and less suitable to our requirements comparing with the second technique, MEAN stack.

MEAN MEAN is a free and open-source JavaScript software stack for building dynamic web sites and web applications. It consists of MongoDB database, the Angular.JS front-end framework, the Express.js Web framework and the Node.js cross-platform runtime environment. Comparing with the traditional LAMP stack, the MEAN stack makes code isomorphic because it is completely JavaScript-powered, which means a single language could be used from top to bottom. This flexibility makes our MEAN-based programming significantly easier and feasible. Moreover, each part of it has its own advantages.

- MongoDB: MongoDB is a document-oriented database that allows storing documents in JavaScript Object Notation (JSON) format (JSON, 2016). It is therefore a format that JavaScript natively understands, which enables the data flows among all the layers without rewriting or reformatting under MEAN stack (MongoDB, 2016).

- Express.js: Express is a minimal and flexible Node.js web application framework (Express.js, 2016). It offers a robust set of features for web and mobile applications. It is good at enabling the simple REST routes, handling automated HTTP header and supporting connect middleware to plug in synchronous functions in order to manage the requests and responses. REST stands for representation state transfer, which is a software architectural style for data communication. It is efficient, scalable, simple and secure.

- Angular.js: Angular.js is an open-source web application framework, which is powerful for building SPA (shown in section 4.3.3) (Angular.js, 2016).

- Node.js: Node.js is an open-source, cross-platform runtime environment for developing server-side Web applications. Navigation among various layers of the LAMP stack can be extremely difficult because of the various configuration files with different syntax. MEAN simplifies this through use of Node.js. Moreover, Node.js is also faster and more scalable than other server side technologies including LAMP because of its non-blocking architecture (Node.js, 2016).

In conclusion, compared with LAMP, MongoDB offers a more flexible, accommodating layer for storing data. Node.js provides a better nexus for running your server, while Express.js helps standardize how to build your websites. On the client, Angular.js provides a simple way of adding interactive functions and AJAX-driven rich components.

4.3.3 How to Build the Web Application: Single-page Application (SPA)

A single-page application (SPA) is a web application or web site that fits on a single web page with the goal of providing a more fluid user experience similar to a desktop application. In the SPA, either all necessary code is retrieved with a single page load, or the appropriate resources are dynamically loaded and added to the page as necessary, usually in response to user actions. SPA feels like a native application because it is fast and responsive. Additionally, it is able to work offline, which means even if the user loses the internet connection, SPA can still work because all the pages are already loaded. According to all metrics of SPA stated above, it fits the requirements of our application perfectly.

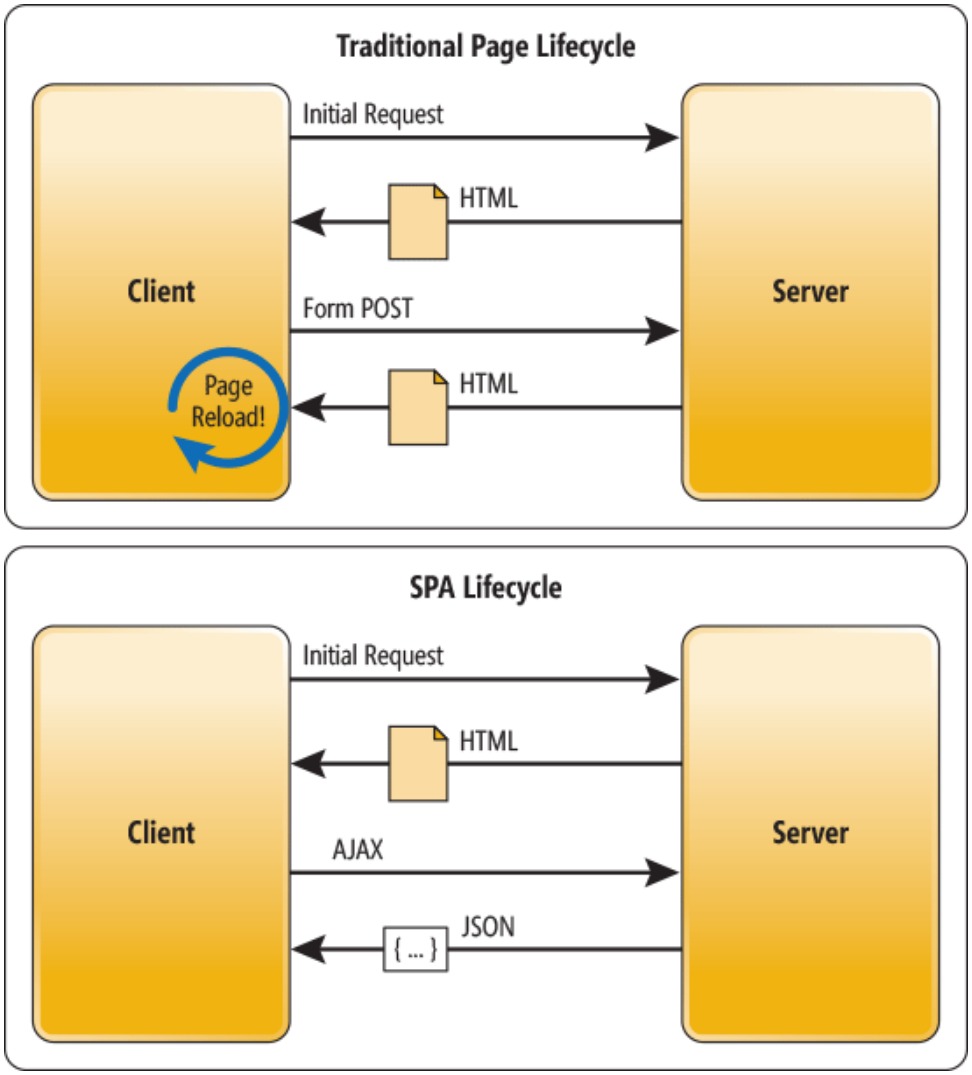


Figure 10: The Traditional Page Lifecycle vs. the SPA Lifecycle. (Wasson, 2013)

4.3.4 Third-party Library

- jQuery: jQuery is a light-weight JavaScript library which simplifies writing JavaScript with easier Application Programming Interfaces (API) (jQuery, 2016). It makes everything much easier, including Document Object Model (DOM) selection, document navigation and manipulation, event handling, CSS manipulation and so on. jQuery has changed the way that millions of people writing JavaScript with its extensibility and flexibility.

- zTree: As a note-taking application, it is necessary to have a clear and easily managed structure of notebooks. Users may create different notes in different notebooks. zTree is a tree plug-ins based on jQuery which supports flexible configuration and outstanding performance and combination of multiple functions (zTree, 2016). Moreover, zTree uses delay loading technique, which allows loading a significant number of notes quickly. It also supports dragging and dropping nodes, flexible checkbox and radio selection function.

- Bootstrap: In order to provide consistent and reusable style, we chose to use bootstrap to ease the development our application. Bootstrap is a powerful and intuitive front-end framework which supports a faster and simpler web development (Bootstrap, 2016). It contains several design templates for forms, buttons, navigation and other interface components. It is compatible with all major web browsers and also supports responsive web design which enables the layout of web pages to adjust dynamically.

- LocalForage: Due to the fact that our application must have offline capabilities such as saving and loading large data locally, it is intuitive and cost-effective to store data into the users browser. However, normal saving mechanisms are fragmented. LocalStorage could realize basic data storage, but it is slow and unable to handle binary blobs (http://mozilla.github.io/localForage/, no date). IndexedDB and WebSQL are asynchronous, fast, and support large data sets, but their APIs are not very straightforward. Moreover, neither IndexedDB nor WebSQL is compatible with all major browsers. The localForage is a JavaScript library developed by Mozilla. It provides a simple storage API that has all the capabilities of APIs mentioned above, but without the steep learning curve. Behind the scenes, localForage uses native browser technologies including IndexedDB, WebSQL, and localStorage to actually store data, but sits on top of this data storage layer and provides a number of methods that can be used for managing data.

- Font-awesome: In order to avoid cumbersome images to present icons and thus save web pace, we use Font-awesome to replace image icons with simple and small-sized fonts. Font Awesome is a font and icon toolkit based on CSS and LESS, which provides scalable vector icons that can instantly be customized (Font-awesome, 2016).

4.4 Group Work

4.4.1 Agile Development: Scrum

Scrum is a flexible, holistic product development strategy where a development team works as a unit to reach a common goal as opposed to a traditional, sequential approach (Scrum Alliance, 2016). The general process has been introduced in the interim report (shown in **Appendix F**) and also shown below.

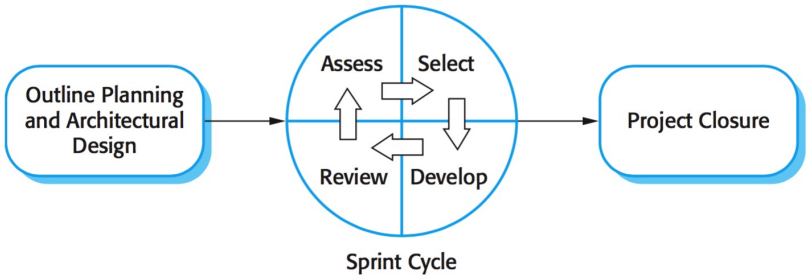


Figure 11: The Scrum process (Sommerville, 2011)

In our development stage, the whole team is divided into four different roles.

1. Product owner: Product owner is responsible for linking the communication gap between the whole team and stakeholders. During two meetings with stakeholders, he takes charge in communicating team status, negotiating priorities, requirements, schedule and problems, and keeping stakeholders interested and satisfied. Additionally, the most important task is to prepare a product backlog, which is a prioritized features list for the product. Through the product backlog, it is possible to convey a vision of individual task to the scrum team and track development progress.

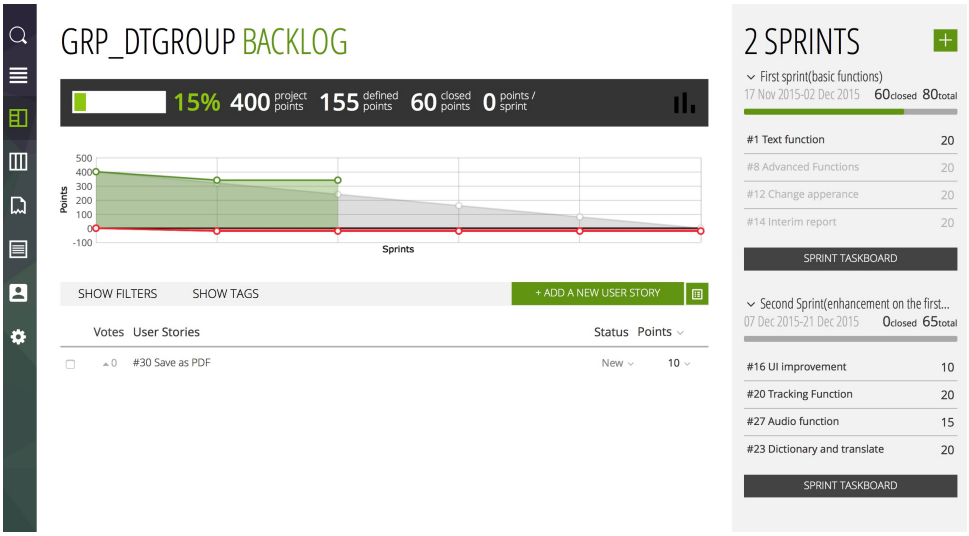


Figure 12: The example backlog for first two sprints: X-axis stands for the whole working period in the first semester, which is divided into 5 sprints as planned. (Taiga, 2016)

2. Development team: The development team consists of two separate teams. The first one is UI-based front-end develop team, which takes in charge of creating and implementing interface components with HTML and CSS. They help ensure the product appears consistent and work properly in different browsers, different operating systems and different devices to realize cross-browser, cross-platform and cross-device. The second team is

back-end develop team, which focuses on creating the framework the site interface will be placed on, data transferring, business logic, database interactions, performance optimization and server configuration and maintenance. The techniques used consist of node.js, express.js, JavaScript and MongoDB. The separation of software systems into front and back ends simplifies development and separates maintenance.

3. Scrum master: The Scrum master is not a traditional team lead or project manager, but serves as a buffer between the team and any distracting influences. He is mainly responsible for coaching the team and facilitating team events to deliver potentially shippable increments of product at the end of each sprint. He also takes in charge of organizing the Scrum board (example shown in Figure 13) and review meeting (example shown in Figure 14) after each sprint, where the project is assessed against the sprint goal determined at the beginning of the sprint.

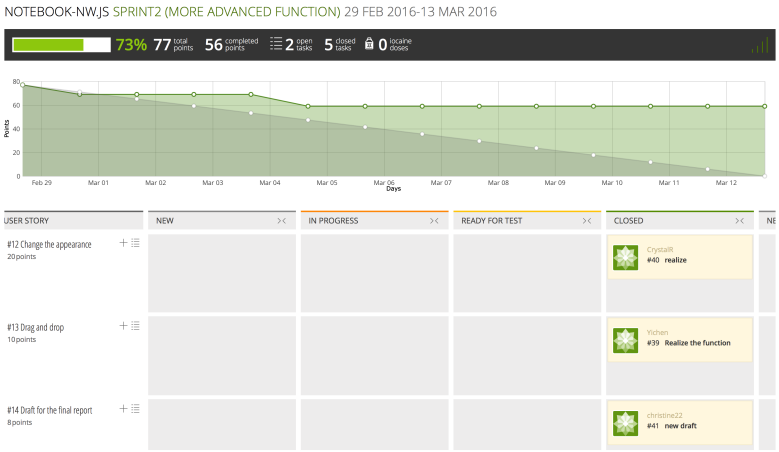


Figure 13: The Example Scrum Board. (Taiga, 2016)

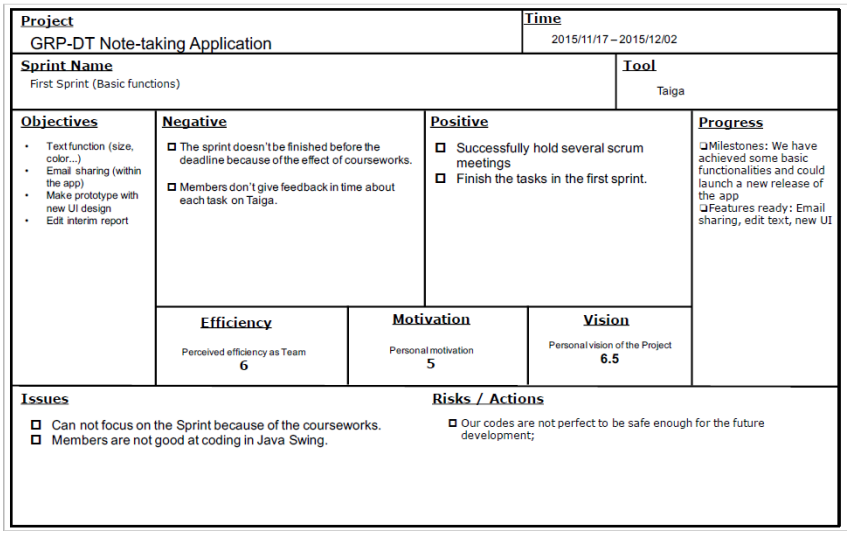


Figure 14: The Sprint Review. (Taiga, 2016)

4.5 SQA (Software Quality Assurance)

According to Sommerville (2011), at the project level, software quality assurance involves specific quality processes, checking the execution of these planned processes, and guaranteeing that the project outputs conform with the standards that are applicable to the project. SQA covers the entire software development process, including processes such as requirements specification, software design, software implementation, source code control, code reviews, software configuration management, software testing, and product integration.

4.5.1 SQA Team

SQA team takes charge of managing the release testing process and checking that the system tests provide coverage of the requirements and that proper records are maintained throughout the testing process. In order to provide an independent check on the software development process SQA team should be separate from development teams.

In our project, due to the small size of the team, we could not set up an entirely separate SQA team as stated above. However, to ensure the objective view of the software, two different SQA teams are set to distinguish from two opposite development teams. This allowed us to check software quality without being influenced by software development issues.

4.5.2 Reviews and Inspections

Reviews and inspections are SQA activities that check the quality of project outcomes. This consists of examining the software based on the SRS documentation and records of the process to discover errors and omissions and to check whether quality standards have been followed.

In our project, the whole process was conducted by the SQA team. During the review stage, the SQA team worked together to inspect the software and its related documentation, looking for potential problems and inconsistency with standards. The results were reported to the Scrum master, and then he made planning decisions based on these assessments. During the inspection stage, the SQA team collaborated to find any problem in the program, including bugs, incompleteness, incorrect logical structure, duplicate code fragments and informal presentations.

4.5.3 Software Measurement and Metrics

Software measurement is concerned with deriving a numeric value for an attribute of a software component so that by comparing these values to each other and to the standards, it is possible to draw conclusions about the quality of software and assess the effectiveness of software processes, tools, and methods. Common software quality attributes are shown in Figure 15.

Safety	Understandability	Portability
Security	Testability	Usability
Reliability	Adaptability	Reusability
Resilience	Modularity	Efficiency
Robustness	Complexity	Learnability

Figure 15: Software Quality Attributes

However, it is often difficult to make direct measurements of many of the software quality attributes. Quality attributes such as learn ability, portability, and usability are external attributes that are concerned with how developers and users experience the software. They are too subjective to be measured objectively (Sommerville, 2011). In order to judge these external attributes, it is necessary to measure some internal attributes of the software, such as its size, complexity and safety.

4.6 Change Management

Change is inevitable for developing a software system. Requirements changes, repairing bugs and adapting system environment are the changes that the software developers should take into consideration. To cope with changes and apply changes to the system in a controlled way, a change management process is necessary and essential. According to Sommerville (2011), the change management process focuses on evaluating costs and benefits of proposed changes, analysing the consequence of not making the changes and deciding which parts in the system need to be changed.

Our change management process consisted of five steps:

1. Report the change.
2. Check the change: checking is necessary because some changes do not require action.
3. Conduct change assessment and costing for valid changes.
4. Estimate the cost of making the change and decide whether it is cost effective.
5. Cope with the approved change and maintain a record of the change.

During the development stage, although we used system prototyping, incremental delivery and re-factoring to support change tolerance and avoidance, we still came across several unavoidable changes.

Java Swing to NW.js As stated in progress section, we had to change our programming environment so that we had to abandon most of our existing product and restart in the middle of the project. Firstly, we reported the changes to our supervisor and discussed potential solutions.

We used the SWOT analysis to list our strengths, weakness, opportunities, and threats in order to help us make the decision (Watson, 2006, p.358). The first priority of the solution is to satisfy the stakeholders' new request, and next is minimize the effects of disruption in the whole process. Based on this principle, we investigated on the available techniques that could replace Java Swing to develop a cross-platform desktop application and estimated the cost and benefits of each solution. Finally, we decided to use NW.js after comparing NW.js with Java Swing.

Comparison between Swing and NW.js

	Java Swing	NW.js
Learning curve	Relatively stable (Only relatively since we are not that good at Swing too)	Steep (although we have learnt the basic of web techniques, we are not as familiar with them as Java.)
Effort	High since we need to redesign totally as well	Higher since we need more effort to learn
Feasibility	Because of the limitations of Swing, we cannot guarantee that we can implement all features.	Because of the nature of the app, we are more confident that we can at least implement some of the features.
User interface of the product	Relatively "ugly"	More user-friendly
Compatibility with the local operating system	Better	Maybe not as good (not sure)
Compatibility across the platform	Better	The layout may differ among different operating systems and display devices.
Motivation	Less motivated since we are a little bit tired of Swing	Maybe more interested in nw.js since it is a new technique.
Risk	Relatively low since at least we have already developed a note-taking app with powerful text-editing function (although stakeholders do not concern about it very much)	High since we are not sure how well we can manage this new technique.
Extension	Relatively limited	Maybe more extensive

Figure 16: Comparsion between JAVA Swing and NW.js

Overall, the software development is in instability, and the way how we manage the change becomes important to process and results.

4.7 Version Control System: Git

4.7.1 Configuration Management

According to Sommerville (2011), configuration management is the process of managing a changing software system. It aims at guaranteeing that team members can access the project code and documents in a controlled way. There are three central configuration management activities:

1. Version management, where support is offered to keep track of the different software components (Sommerville, 2011). In our project, we use a web-based Git repository hosting service called GitHub to manage collaboration. It offers the functionality of Git including all of the distributed revision control and source code management (SCM). More specifically, we used eGit which is a plug-in of Eclipse, our Java programming integrated development environment (IDE), to manage to access to GitHub with the Eclipse.

2. System Integration, where support is offered to help developer define the versions of components to create system version.

3. Problem tracking, where support is offered to allow group members to report bugs and technical issues and inform them who is responsible for what issues. In our project, we use issue tracking function of GitHub, shown in Figure 17, to focus on bugs and features. At first, we have another option, which is a web-based bug-tracker tool called Bugzilla. However, GitHub's issue tracking is more simplistic and easier to use. Moreover, it has the GitHub integration, so we choose it as our problem tracking tool.

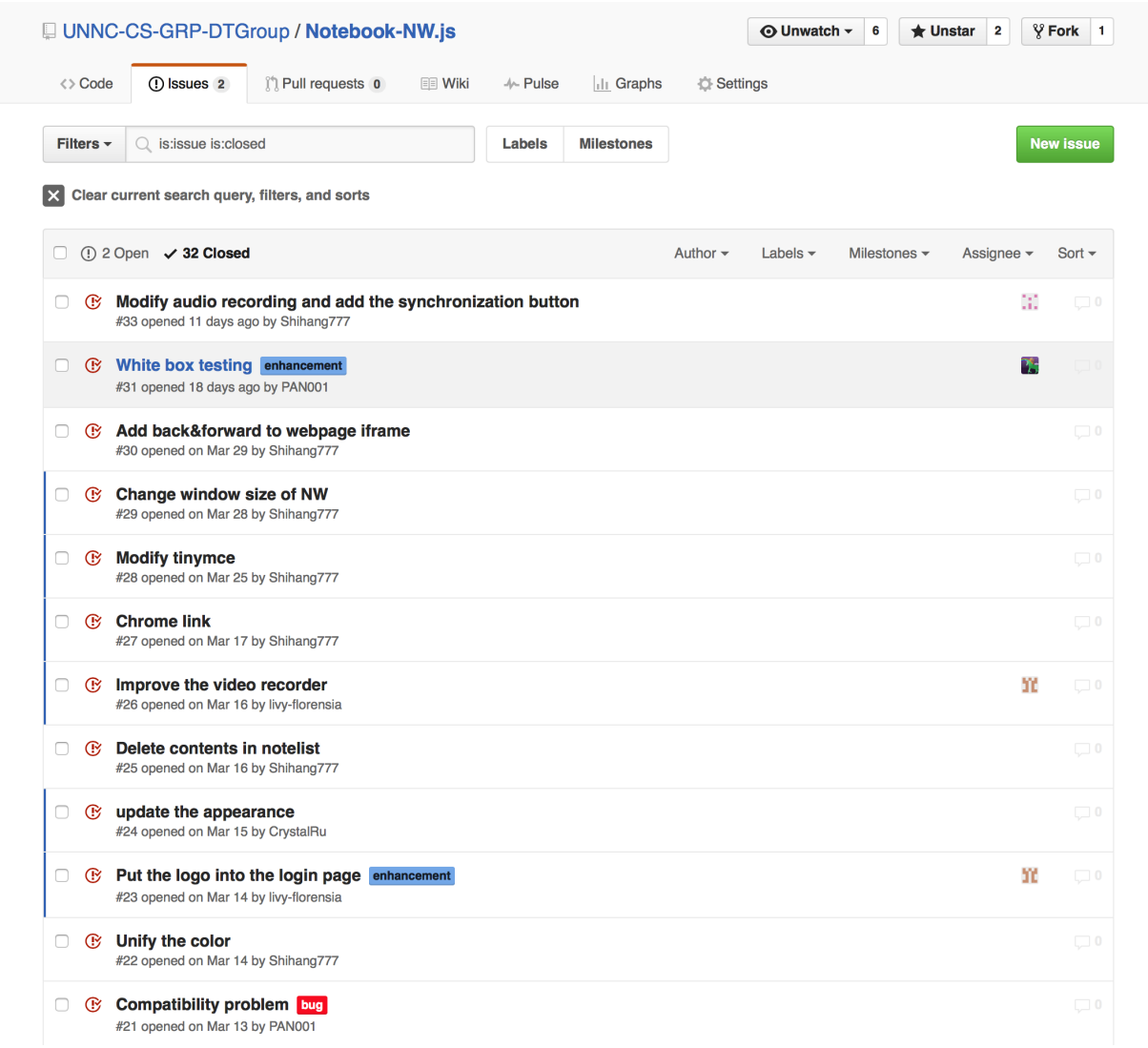


Figure 17: issue tracking of GitHub

4.7.2 Git Branch Model

When we used Github to organize our code at the beginning, we often encountered various collaborative problems such as interruption from unstable or wrong committed codes and frequent commitments. By using branch function of Git and following the branch model below, we managed to work in order.

First of all, branch represents an independent line of development. It is a way to request a brand new working directory, staging area, and project history. In our model, two main branches called master and develop, have infinite lifetime. The origin/master is regarded as the main branch, where the production-ready state is always being reflected by the source code of HEAD. At the same time, the origin/develop branch is considered to be the main branch where the source code of HEAD always reflects the latest delivered development changes for the next

release. The changes should be merged back into master branch as soon as the source code of develop branch reaches a stable point and is ready to be released. These two branches are parallel to each other as shown in Figure 18.

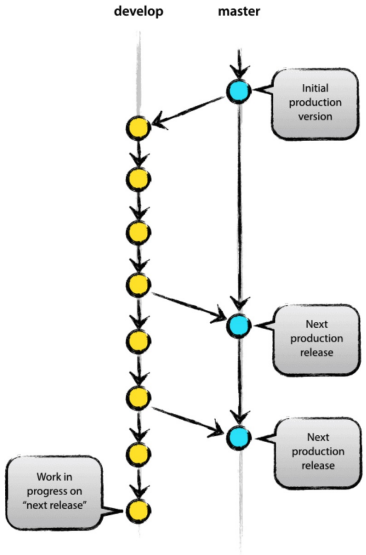


Figure 18: Two Main Branches (Driessen, 2010)

There are also three supporting branches, including feature branch, release branch and hotfix branch. The difference with the main branches is these branches' life time is limited. They work as Figure 19 shows.

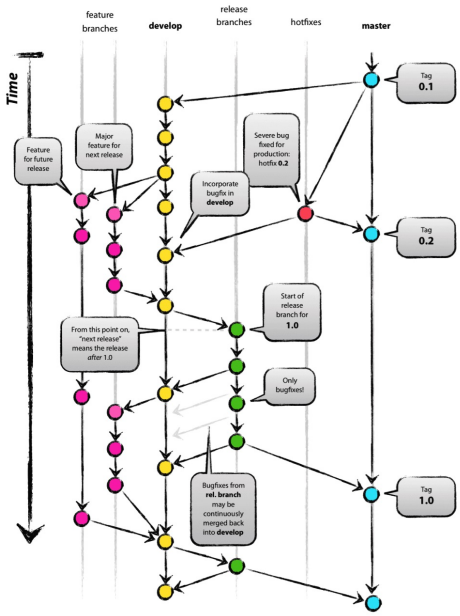


Figure 19: The Branch Model (Driessen, 2010)

In practice, this branch model has turned out to be effective in our project. It helped rearrange

work in order and avoid any potential merging problems. Additionally, it allowed us to get a better understanding of the branching and releasing processes. Figure 20 shows the branches we created.

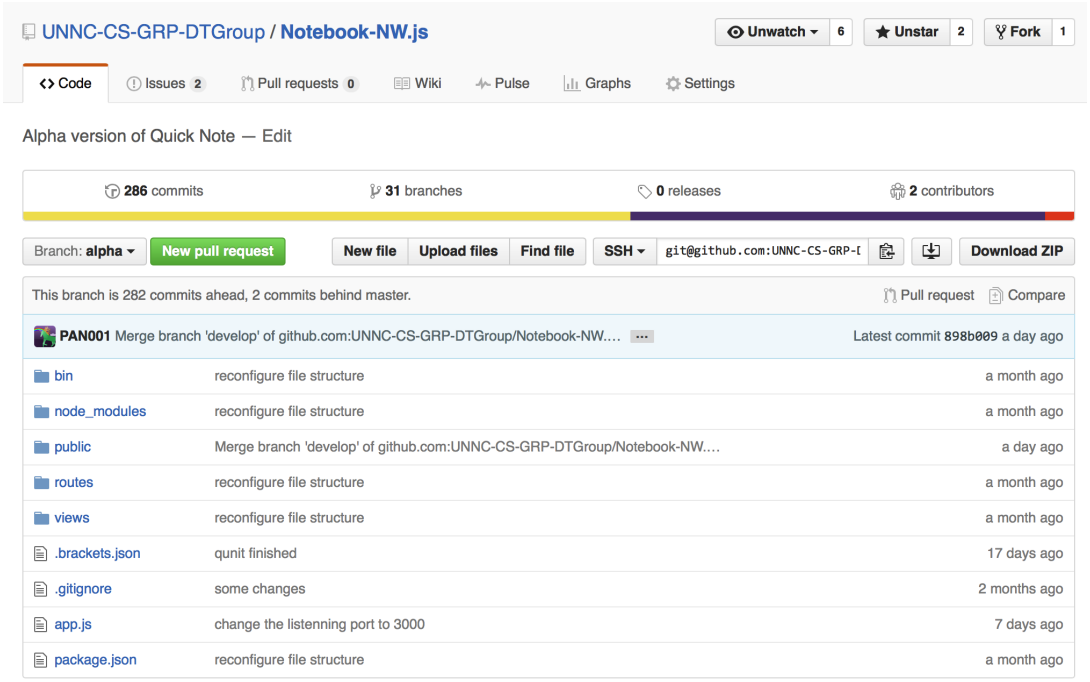


Figure 20: Our Github branches

4.8 Tools

4.8.1 Worktile

Worktile is a working-platform website that is used to manage the small tasks in a relatively huge group project (Worktile, 2015). The tasks are constrained in lists, making them more organized. It also can keep documents and has a calendar to keep track of deadlines. In the first semester, we used it to manage the tasks, share files and track the progress. Later on, after we changed our developing approach from plan-driven to agile, we started to use Taiga instead of Worktile because Taiga is the optimal tool for Scrum.

4.8.2 Taiga

Taiga is a project management platform for agile developers, especially for Scrum methodology (Taiga, 2016). It supports sprints management, burn down charts, product backlog and Kanban.

4.8.3 GitHub

GitHub is used for version control (Github, 2016). It keeps track of our progress, so that when five members of the team are working on the project at the same time, it would not screw up the project. When we want to do some modification, we created a new issue of what we are trying to do, then it gave us a new issue with a number. We created a new branch with that

number, and that was when we start working. When we were done with our modification, we will push to that branch and will merge it to the 'develop' branch later on. It makes modifying our project much more neat and efficient, and we do not have to worry about working at the same time.

4.8.4 Overleaf and TeXStudio

We used Overleaf to write LaTeXreports, since our supervisor told us that LaTeXwas a really good tool for writing documents (Overleaf, 2015). It is really efficient to use if many people are working on the same file at the same time. We also used TeXStudio to write LaTeX documents offline, and also use BibTeX, which is a part of LaTeXused for referencing purpose (TeXStudio, 2016).

4.9 Access Permission Management

First of all, we created a group e-mail account 'DTGroup.nottingham@outlook.com' which was used to register administrator accounts. For every platform we used, there is a default administrator account registered with the group e-mail account with the same user name 'DTGroup'. This account was used to offer the advisor access to the platform and a high-level management above all of us.

For GitHub, we created an organization which is suitable for group project development. The version manager was set to be the owner of the organization to manage it, who had full access to the project, and other members were set to members of the organization who have limited access permission.

4.10 Cooperative and Collaborative Learning

It took a lot of time and energy for us to learn several same things at the same time individually, so we conducted a series of training and knowledge sharing sessions to achieve high efficiency and better performance. More specifically, in each learning process, one or two of us were required to particularly research on one specific topic, and then the 'expert' held the learning session to teach the rest of us in an effective way. We observed that by learning in group, it was much more efficient and productive because of the knowledge sharing. Additionally, it also promoted more sophisticated thinking by discussion of controversial topics. Last but not least, it encouraged healthy evolution and development of our team.

Below is the list of learning sessions that we have held:

- Introduction to Worktile and how to work efficiently in group (shown in **Appendix G**);
- Introduction to Git and GitHub (shown in **Appendix H**);
- Project management and tracking (shown in **Appendix I**);
- Practical application of our GitHub branch model;
- Sharing knowledge of Java Swing;
- Sharing knowledge of software engineering and requirement engineering;

- Sharing knowledge of advanced application of Eclipse and eGit;
- Sharing knowledge of GUI design.

5 Testing

In order to ensure the quality of our application, we went through a complete series of testing processes from development testing, release testing to user testing. The development testing is mainly focuses on finding bugs and defects of our application. The release testing is responsible for testing a complete version of the system before it is released to others and the final user testing aims at deciding whether or not our application should be accepted by the potential users.

We made a detailed and specific testing report (**Appendix J**) which consists of testing cases, testing results, testing assessment for each different testing.

5.1 Developemnt Testing

During development, testing is carried out at three levels of granularity: unit testing, component testing and system testing.

5.1.1 Unit Testing

Unit testing is the process of testing individual functions or methods of the program. Unit testing falls within the scope of white-box testing, which means the internal structure and source code should be tested.

Since testing is costly and time consuming, it is essential that effective unit test cases are prepared. According the suggestions given by Sommerville (2011), our testing cases are designed particularly into two kinds. The first can reflect normal operation of a program and show that the component works. For example, when testing the saving and loading function, the test case is designed to show that the saved note names and note amount are correct as specified. The other kind of test case is based on testing experience of where common problems arise. It uses abnormal inputs to check that these are properly processed and do not crash the component.

Two testing strategies are adopted:

1. Stress testing: Test cases are designed around the limits of the system so that the testing could stress the systems by making demands that are outside the design limits of the software. This kind of test strategy could cause defects to come to light that would not normally be discovered. Example test cases are shown in Figure 21.

2. Partition testing: The input data of a software to test is divided into partitions of equivalent data where test cases can be derived. In principle, test cases are designed to cover each partition at least once. An advantage of this approach is reducing the total number of test cases.

ID	Test case
1	Create “Notebook1” and “Trash”
	Input: Id and title of two notebooks Tests: Test for whether the two notebooks are created successfully Output: “Notebook1” and “Trash” are created successfully and there are 2 objects in Notebook.cache
2	Test title of “Notebook1” and “Trash”
	Input: Titles of the two notebooks Tests: Test for whether the title of two notebooks are corrected when created them Output: The titles are correct
3	Create 100 notebooks.
	Input: Input id of 100 notebooks, from 0 to 99 Tests: Test for whether 100 notebooks are created Output: The 100 notebooks are created successfully
4	Test title of No.18 notebook
	Input: Input expected title of No.18 notebook Tests: Test for whether the real title of No.18 notebook is same as the expected one Output: The real title is same as the expected one
5	Create 1000 notebooks.
	Input: Input id of 1000 notebooks, from 100 to 1099 Tests: Test for whether 1000 notebooks are created Output: The 1000 notebooks are created successfully
6	Test title of No.555 notebook
	Input: Input expected title of No.555 notebook Tests: Test for whether the real title of No.555 notebook is same as the expected one Output: The real title is same as the expected one

Figure 21: Part of the Development Test Cases.

5.1.2 System Testing

In the system testing, some or all components are integrated and the system is tested as a whole. System testing focuses on checking that components are compatible, interact correctly and transfer the right data at the right time across their interfaces. It is a crucial step in SQA because it is the first level where the system is tested as a whole and serves as the prerequisites for following release testing and user acceptance testing. Due to the importance of the system testing, we particularly invited one of year 4 CS students, Jack, to lead the test team. The system testing report could be found in **Appendix K**

5.2 Release Testing

Release testing mainly checks that the system meets the requirements of stakeholders so that helps convince that the software is stable and complete enough to release for daily use. Release testing falls within the scope of black-box testing.

During the release testing stage, one separate testing team focuses on checking that the system delivers its specified functionality, performance, and dependability and it does not fail dur-

ing normal use. Yiru, Jiahui and Livia conducted the release testing, because them three are mainly responsible for the front-end development and seldom accessed the functional codes. The release team applied requirements-based testing, which is a systematic approach to test case design where each requirement is taken into consideration and a set of tests is derived for it. Requirements-based testing is validation rather than defect testing so that the main purpose is to show that our product has properly implemented the requirements specified in the SRS document.

Test Case ID	Test Case
1	Tracking and download the history
	Input: Full URLs in the website address box Keywords in Google search Tests: Test for inputs where the URLs and keywords stand for different information Output: The history can be download and includes the URL typed and keywords or fail
2	View the notes by date order
	Input: Create new notes and edit them Tests: Test for inputs where the new notes are in one notebook Test for inputs where the new notes are in different notebooks Output: The newest edited note is rank first in its notebook and newest folder or fail
3	Add tags to classify notes and search notes by tags
	Input: Tags that already provide: red, blue, yellow and green Tags that created original Tests: Test for adding tags and deleting tags Test for adding more than one tags to one note Test for classifying notes by tags Output: A list of notes that have the same tag or fail
4	Drag and drop
	Input: Drag links Drag pictures Tests: Test for inputs where the links from different website Test for inputs where the pictures with different size Output: The links and pictures can be insert into notes and can open the links by click it or fail
5	Insert multimedia into notes: video, audio and picture
	Input: The source of the multimedia Tests: Test for inputs where the source comes from online resource Test for inputs where the source comes from local storage Output: The multimedia can be embedded into notes and played or fail

Figure 22: Part of the Release Test Cases.

5.3 Uesr Testing

The whole team conducted the user testing in two ways. One is alpha testing, and another is beta testing. For the alpha testing, we invited our three stakeholders as our main users, because they were involved along the application was being developed. We met the stakeholders individually, and got feedback form them. We also invited twenty students from different majors such as International Communication, Computer Science and International Business, to try out

our application. Before the testing, the participant information sheet (shown in **Appendix L**) was prepared to help testers better understand the project and the consent form (shown in **Appendix M**) was signed according to the related university regulations.

With regard to the beta testing, we invited some of our classmates and friends to try our application, because one kind of main potential users of our application is student. We used a online survey questionnaire to collect their feedback, which is shown in **Appendix N**. Their feedback can be found in **Appendix P**.

5.4 Testing Assessment

Based on all tests mentioned above, we concluded multiple testing results as following.

Overall, we have realized most of requirements from stakeholders shown in Figure 23 and provided several new and useful features, such as user logging in and out function, powerful editor and Google customized search engine.

Requirements	Status
1. The tutors shall be able to track the students' activities.	Implemented
2. The students should be able to easily share the notes with tutors and other students.	Implemented
3. The tutors should be able to view the inserted files in the notes that students shared.	Partially implemented
4. The tutors should be able to extract the useful part of students' notes and integrate them into a super note.	Unimplemented
5. The students should be able to drag information from websites into notes, and together with the URL.	Partially implemented
6. The students should be able to record the lectures when taking notes.	Implemented
7. The students should be able to add tags to each note and search the note classified by tags.	Implemented
8. The students shall be able to put different notes under a single notebook and view them by date order.	Implemented
9. The students shall have access to the internet from within the app using different search engines.	Implemented
10. The tutors should be able to comment on students' notes.	Unimplemented
11. The students shall change the setting options of the app (size of each part).	Implemented
12. The tutors should be able to build the template for notes by themselves for different modules.	Implemented
13. The students should be able to make margins to set a certain structure for their notes.	Unimplemented
14. The students should be able to build tables inside the notes easily.	Implemented
15. The students should be able to open documents within the same window in the app.	Partially implemented
16. The students should be able to create a new note easily.	Implemented
17. The students should be able to change colors of words and make specific words in bold or italic.	Implemented
18. The students should be able to export notes to PDF.	Partially implemented
19. The students should be able to print the notes within the app easily	Implemented

Figure 23: The Checklist of Requirements

However, based on the feedback from stakeholders and users, we still found some problems:

- Compatibility problems: Due to the nature of our application, it could run either as a native desktop application or in the web browsers. However, some functionalities could only work in the native desktop application such as audio and video recording while some other functionalities could only work in the web browsers such as inserting multimedia into notes. This compatibility problems may make trouble in the daily usage of the application.
- Difficult to learn: According to feedback from several testers, they found it is difficult to figure out how to use the application at the first time. Partition between notes and

notebooks is not obvious and the media part is confusing. Additionally, when inserting multimedia into notes, it requires source code manipulation instead of choosing files directly, which makes it less user friendly.

- Instability: Several testers reported that during the normal usage of the application, it may crash accidentally, which occasionally result in the loss of progress made before.
- Lack of necessary functionalities: Although the application allows the user to register a new account and log in with it, the application does not provide a way for the user who forgets his password to retrieve it, which makes it inconvenient and troublesome in the daily usage. Moreover, other basic functionalities, such as searching notes by keywords and sorting notes by names, are not realized.

6 Problems Encountered

6.1 Technical Issues

There are several technical issues we have encountered so far, and these problems disrupted our progress to some extent.

6.1.1 Compatibility

Since our application is based on web techniques, it may not work consistently in different operating systems or web browsers due to the reason that how it will work is based on the particular engine of the web browser. We have tried our best to write codes for different environment but some functionalities still could not work under some specific environment.

6.1.2 External Archives

When implementing some functions, we have to use some extended third-party external archives. However, some third-party packages only support Window system which makes it difficult to implement these functions on other operating systems.

6.2 Management Issues

In general, we have not encountered very serious management issues. Our team keeps progressing in a gradual and steady pace without disorderliness under the guidelines from supervisor and the team leader. However, we did meet some problems in management. Below is a summary of management issues encountered by the team leader and scrum masters.

6.2.1 Hierarchy

Due to the fact that we are all of the same age and classmates to each other, there is no distinct hierarchy between the leader and the rest of group members. This is absolutely right in the normal case. However, when it comes to the group project, this makes it rather difficult for the leader to distribute tasks and arrange work in an efficient way. This results in much extra negotiation and potential conflicts which should never occur in the professional development team.

6.2.2 Motivation

Motivation means arranging the work so that group members are encouraged to work as efficiently and effectively as possible. However, there is a period of time when some of the group members are in low morale. As Maslow pointed out, people are motivated by satisfying their needs (Finkelstein, 2006). These needs are mainly divided into five parts as shown in Figure 24. For us, the most important are esteem needs, which means the need to receive respects from others, and self-realization needs, which means the need to feel improved and progressed during the work. To satisfy esteem and self-realization needs, it is beneficial to show group members that they are indispensable and valued by the whole team. Appropriate praise and public recognition of achievements are good ways to realize this. It is also necessary to give them responsibility for their work and organize a training programme to help them promote.

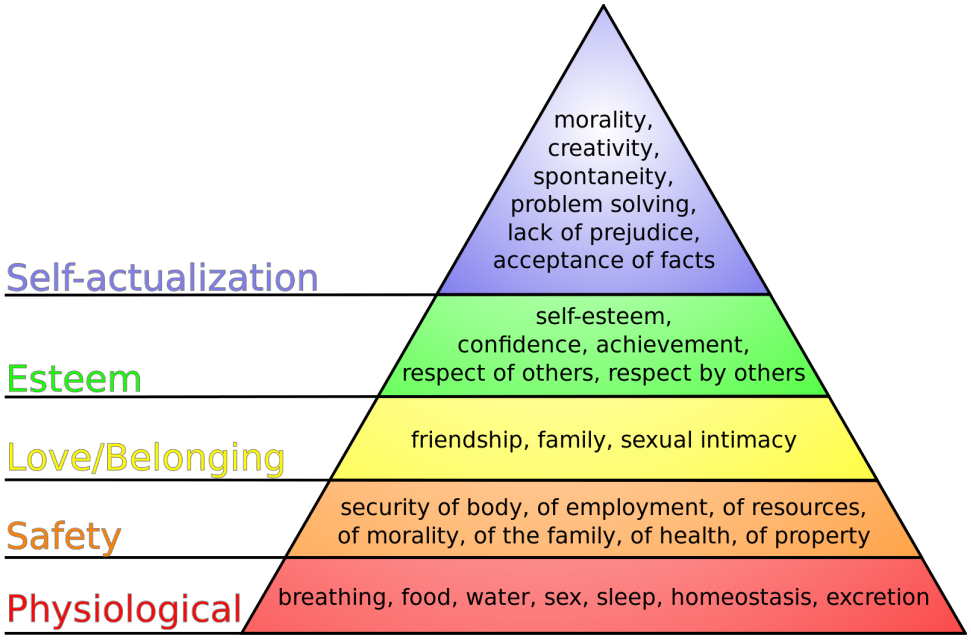


Figure 24: The branch model-Maslow's hierarchy of needs (Finkelstein, 2006)

7 Conclusion

7.1 User manual

In Mac OS, our application Quick Note could be opened as a native Mac OS app and in Windows operating system, Quick Note could be opened as an executable program (.exe file). Moreover, it could also be opened in any web browser in any operating system. A detailed user manual could be found in **Appendix O**.

7.2 Reflection

During the final time of project, our group have had a profound thinking to reflect on our work for learning from the experience and make improvements.

7.2.1 Development Process Reflection

In this part, we used Kolb's Learning Cycle (1975) to structure the reflective writing.

Experience After we spent about one month going through a complete and thorough requirements engineering process, we developed our application for the following two months. At the end of the Autumn semester, we found our product deviating from the stakeholders expectations. Faced with this change from the stakeholders, we went through a complete change management process and restarted our product development as stated above.

Observations and reflections At the first sprint meeting of the Spring semester, we retrospect our agile development strategy of autumn semester and summarized the problems. We noticed that we only held one single meeting with stakeholders during the whole two months'

development period. Whereas, one of the five key agile principles is customer involvement, which requires that customers should be closely involved throughout the development process. Therefore, the lack of communication and getting feedback from stakeholders make it impossible to evaluate the iterations of product developments and the quality of each delivery.

Development of ideas The most essential idea learned from this experience is the importance of agile key principles. We should involve stakeholders in our development process closely and frequently, because the stakeholders' opinions decide how the product should be built. Additionally, we should also embrace the changes, expect the system requirements to change and design the system to accommodate these changes.

Testing ideas in practice Having experienced and analysed the problems we have, we adhered to the principle of agile more firmly. As long as we have made some progress, we set up meetings with stakeholders to examine our product and check whether it is valid. The result is that we are in the right direction of building the product the stakeholders really want.

7.3 Future Plan

Based on the original blueprint of our app at the software design stage, we planned to realize several advanced functionalities during the late development stage. The most promising and piratical one is that the application could classify users' public notes automatically based on machine learning, natural language processing and online learning techniques. By doing this, students could search for public notes using the keyword, such as subject, specific area name, module code. This could not only build a university ranged but major or topic specific knowledge database, but also could provide a user-friendly notes and knowledge sharing environment to facilitate and manage innovation and organisational learning.

Apart from these advanced and useful features to implement, based on the testing assessment and feedback from potential users (section 5.4), it is necessary to improve the interfaces of the application to make it more user friendly and easy to manipulate. Additionally, compatibility and stability problems need to be addresses so that the application could work consistently and property in any operating system or web browser.

References

Adobe. (2016) Available at: http://www.adobe.com/products/air.html (Accessed: 27 April 2016)

Angular.js. (2016) Available at: http://docs.angularjs.cn/api (Accessed: 27 April 2016)

Belbin, R. M. (1981) *Management Teams: Why They Succeedor Fail.* A Butterworth-Heinemann Title

Bootstrap, (2016) Available at: https://wrapbootstrap.com/ (Accessed: 27 April 2016)

Briggs, M. and McCaulley, M. (1992) *Manual: A Guide to the Development and Use of the Myers-Briggs Type Indicator.* Consulting Psychologists Press

Computer Weekly, (2002) *Write Once, Run Anywhere?* Available at: http://www.computerweekly.com/feature/Write-once-run-anywhere (Accessed: 27 April 2016)

Driessen, V. (2010) *A Successful Git Branching Model.* Available at: http://nvie.com/posts/a-successful-git-branching-model/ (Accessed: Dec. 5th, 2015).

EverNote. (2015) Available at: https://evernote.com/penultimate/?noredirect (Accessed: Dec. 5th, 2015).

Express.js. (2016) Available at: http://expressjs.com/ (Accessed: 27 April 2016)

Finkelstein, J. (2006) *Diagram of Maslow's Hierarchy of Needs.* Available at: https://commons.wikimedia.org/wiki/File:Maslow%27_hierarchy_of_needs.svg (Accessed: Dec. 16th, 2015).

Font-awesome. (2016) Available at: http://fontawesome.io/ (Accessed: 27 April 2016)

Github. (2016) Available at: https://github.com/UNNC-CS-GRP-DTGroup/Notebook-NW.js (Accessed: 27 April 2016)

Google Keep. (2015) Available at: https://keep.google.com (Accessed: Dec. 5th, 2015).

http://mozilla.github.io/localForage/ (no date) (Accessed: 27 April 2016)

jQuery. (2016) Available at: http://jquery.com/ (Accessed: 27 April 2016)

JSON. (2016) Available at: http://www.json.org/ (Accessed: 27 April 2016)

Kolb, David A. and Fry, Ronald E. (1975). Towards an applied theory of experiential learning. In Cooper, Cary L. Theories of group processes. Wiley series on individuals, groups, and organizations. London; New York: Wiley. pp. 3358. ISBN 0471171174.OCLC 1103318.

Lan Hing Po, M. L. T. *et al.*, (2014) 'SRS Document', AE2GRP: Software Engineering Group Project.The University of Nottingham Ningbo China. Unpublished assignment.

Lan Hing Po, M. L. T. *et al.*, (2015) 'MULTIMEDIA NOTE-TAKING PROJECT - AE2GRP Final Group Report', AE2GRP: Software Engineering Group Project.The University of Nottingham Ningbo China. Unpublished assignment.

Mobidev. (2015) 'Cross-Platform Development For Desktops: Choosing The Right Technology', Mobidev, 6 August. Available at: https://mobidev.biz/blog/cross-platform_development_for_desktops_choosing_the_right_technology (Accessed: 22 April 2016).

MongoDB. (2016) Available at: https://www.mongodb.org/ (Accessed: 27 April 2016)

Node.js. (2016) Available at: https://nodejs.org/en/ (Accessed: 27 April 2016)

Nuseibeh, B. and Easterbrook, S. (2000) *Proceedings of the conference on the future of Software Engineering.* pp. 35-46.

NW.js. (2016) Available at: http://nwjs.io/ (Accessed: 27 April 2016)

OneNote. (2015) Available at: http://onenote.com:83/?omkt=en-US (Accessed: Dec. 5th, 2015).

Overleaf. (2015) Available at: https://overleaf.com/ (Accessed: Dec. 5th, 2015).

Rajput, M. (2015) *What is the MEAN Stack and Why is it Better than LAMP?* Available at: http://www.programmableweb.com/news/what-mean-stack-and-why-it-better-lamp/analysis/2015/12/22 (Accessed: 22 April 2016).

Scrum Alliance. (2016) *What is Scrum? An Agile Framework for Completing Complex Projects.* Available at: https://www.scrumalliance.org/why-scrum (Accessed: 27 April 2016)

Sommerville, I. (2011)*Software Engineering.* 9th edn. Addison Wesley.

Splinterware. (2015) Available at: http://splinterware.com/products/idailydiary.htm (Accessed: Dec. 5th, 2015).

Taiga. (2016) Available at: https://tree.taiga.io/ (Accessed: Dec. 17th, 2015).

TeXStudio. (2016) Available at: http://texstudio.sourceforge.net/ (Accessed: 27 April 2016)

Wasson, M. (2013) *ASP.NET - Single-Page Applications: Build Modern, Responsive Web Apps with ASP.NET* Available at: https://msdn.microsoft.com/en-us/magazine/dn463786.aspx (Accessed: 22 April 2016).

Watson, T. J. (2006) *Organising and Managing Work.* 2nd edn.The UK:Pearson Education.

Worktile. (2015) Available at: https://worktile.com/ (Accessed: 27 April 2016)

zTree. (2016) Available at: http://plugins.jquery.com/zTree.v3/ (Accessed: 27 April 2016)

Appendices

A. Evaluation Reports

B. Software Requirements Specification Document

C. Participant Information Sheet for Interviews

D. Consent Form for Interviews

E. Research Ethics Checklist Authorization

F. GRP-DT Group Interim Report

G. Introduction to Worktile

H. Introduction to Github and Git

I. Project Management and Tracking

J. GRP-DT Group Testing Report

K. System Testing Reports

L. Participant Information Sheet for User Testing

M. Consent Form for User Testing

N. Quick Note User Testing Questionnaire

O. Quick Note User Manual

P. Quick Note Beta Testing Feedback