DocNo: 001.C.4:1

X-Note
System Requirement Specification
Version 1.0

By:
X-Note Developers
2019-03

Group Member:
Jingyu Li
Du Liu
Yu Fan
Qiuxuan Ling
Shixuan Gu

Document Language:
English

Revision History

Date	Version	Description	Author
04/03/2019	1.0	Create and edit three sections	Shixuan Gu
04/06/2019	1.0	Make some format change.	Jingyu Li
04/06/2019	1.0	Get the file integrated.	Shixuan Gu
04/17/2019	1.1	Update class and sequence diagram	Jingyu Li

Contents

1. Introduction

1.1 Purpose

The purpose of writing this document is to describe and constrain the program requirements of Xnote, thus laying a foundation for the subsequent system analysis, design and realization.

This document will give detailed definitions for both functional and nonfunctional requirements, scenario and use cases.

This document is also responsible for the communications with customers as well as the details of customers requirements.

1.2 Application Scope

This document is mainly applied for the software: Xnote.
Features, subsystems and models relevant to Xnote are all corresponding to this document.

1.3 Definition

The definition of the terms involved in this document will be given in the project glossary section.

1.4 Reference

Microsoft BMS System White Book

Bob Hughes and Mike Cotterell 《Software Project Management》

1.5 Overview

This document includes introduction, current system and proposed system.

The current system section analyzes the current note-taking software, points out the deficiency and comes up with the necessity of Xnote.

The proposed system section gives parts of functional requirements for the system and constrains nonfunctional requirements respectively. This part also describes scenario of the system and concludes to the use cases such as note creating, document editing, etc. The initial user interface design is given as well.

2. Current System

2.1 Overview

Nowadays, e-Notes plays a really important role in our daily life, with many companies and individual developers raising their own software.

Unfortunately, almost each of them has its deficiency. For example, OneNote produced by Microsoft suffers a lot from the synchronization problem, especially in China; Markeditor behaves badly in Notes management; Evernote doesn’t support mathematical formula input which is important to college students especially those major in engineering and science.

2.2 Operation Procedure and Data Processing

Users open the software, create notes, type in data need to be recorded, save the file and expect to revise or review next time.

2.3 Deficiency and Limitation

The existing note-taking software has the deficiency and limitations as following:

1. Synchronization is rather inconvenient.
2. Terrible notes management.
3. No support for mathematical formula input.

3. Proposed System

3.1 Overview

The functional requirements of the Xnote system mainly includes note-taking, support for both Markdown and ordinary text format, mind map, etc.

Nonfunctional requirements of the Xnote system includes synchronization, high reliability, quick response, convenient operation, etc.

The system model of Xnote mainly includes specified participants and use cases. The participants mainly refer to the users, and the use cases include note editing, document management, etc.

3.2 Functional Requirements

Xnote provides users a powerful tool to facilitate learning process through note taking and mind maps, especially for students.

As a cross-platform note-taking desktop application, Xnote acts as a pragmatic tool for learners to take notes with both Markdown and ordinary text format, as well as generating a dynamic mind map to show interconnections within and across different notes.

Notes can be revised on different devices and users are supposed to enhance their reviewing by searching for key words, having time-based learning record, tagging, etc. What's more, it can also output pdf files, txt files and so on.

Users can also have a mind map of their notes which can better sort out the clue. What’s more, personalized user interface style can improve the note-reviewing experience, thus scientifically improve the learning efficiency.

3.3 Nonfunctional Requirements

3.3.1 Usability

Normally, e-note users are quite familiar with note-taking software and since Xnote has a rather user-friendly interface, it's safe to assume that users can easily use the product without training. In terms of the system, it needs to ensure the notes won't be loss during the synchronization and correctly produce mind map, etc.

3.3.2 Reliability

Since the only online operation is synchronization which doesn't take us a long time to maintain, we assume the system is 7days times 24 hours for using.The system must run normally always if the hardware has no problem.

3.3.3 Performance

1) The system response time must less than 1 second.The run-time errors appear every month should less than 5.
2) Apart from the synchronization, X-Note can actually work offline. So it’s not necessary to consider the throughput.

3.3.4 Support

The criteria of coding is corresponding to the <Software Engineering>.

3.3.5 Design Constraints

The software is mainly developed by JS language. In the developing phase, different blocks are divided for convenience. The plantability and reusability are also ensured.

3.3.6 Interface

3.3.6.1 User Interface

As a desktop application, the user interface is designed to be pleasing and tidy.

3.3.6.2 Hardware Interface

Xnote requires the hardware that can run browser.

3.3.6.3 Software Interface

Xnote uses the Nutstore Server

3.3.6.4 Telecommunication Interface

TCP/IP, http protocol

3.3.6.5 Law, Copyright and Declaration

This system is constructed by personal coding and open sources, the part using the open sources complies with the corresponding agreement and the copyright of the part does not use the open source belongs to our development team. The system's (includes but unlimited to the software, usage, etc) final interpretation belongs to our development team.

3.3.6.6 General Criteria

Specification of computer software testing GB/T 15532-2008
Specification of computer software development GB 8566-88
Specification of software maintenance GB/T 14079-93

3.4 System Model

3.4.1 Senario

Scenario: Documents creating
Participants: User
Flow events:
1) The user clicks "New File" button to open the Document Creating Page.
2) The user choose the type of notes.
3) The new document is created.

Scenario: Documents Editing
Participants: Users
Flow events:
1) The user double-click the document that he wants to edit.
2) The editing page is opened. The user is free to edit the document and add tags to it.
3) If editing is finished, click "Save" button, then editing will be saved.

Scenario: Preview
Participants: User
Flow events:
1) The user clicks "Preview" button near the file name.
2) The preview will be show in the new window.

Scenario: Export Notes
Participants: User
Flow events:
1) The user right-clicks the file name and choose the format that is going to be exported on.
2) Choose the directory that the exported file will be saved in.
3) The operation will be done as the user clicks "Confirm" button.

Scenario: Key Word Searching
Participants: User
Flow events:
1) The user keys in the key words that he wants to search.
2) The user clicks "Search" button.
3) The system will show files containing the key words in a list.

Scenario: Check Learning Record
Participants: User
Flow events:
1) The user clicks "Record" button.
2) The system will show the statistics including time that the user spends in editing in different ways.

Scenario: Synchronization
Participants: User, cloud
Flow events:
1) The user clicks "Setting" button and switch to the "Synchronization Log in" page.
2) The user keys in the user name and the password and clicks "Log in" button.
3) The user clicks "Synchronization" button.
4) Local contents will be synchronization to the cloud.

Scenario: Generate the Mind Map
Participants: User
Flow events:
1) The user right-clicks the file name and clicks "Mind Map" button.
2) The Mind Map will be showed in the new page.

Scenario: Sort by tags
Participants: User
Flow events:
1) The user clicks "Tag" button.
2) The system will show files sorted by tags.

3.4.2 Use Case

Table 1 Participant Information Table

Participant Name	Participant Interpretation
User	The user of the system, who can create new notes, edit the existing notes, output html, pdf files, etc.

Table 2 Use Case List

Name	Level	Description
ViewAllNotes	User-oriented	Show all notes with info.
Search	Sub-routine	Search for specific notes or content.
CreateNote	User-oriented	Create a new note.
EditNote	User-oriented	Edit a note.
MakePreview	Sub-routine	Make the preview of a note.
CloseNote	User-oriented	Close and save a note.
ExportFile	User-oriented	Export the preview to file.
Login	User-oriented	User login the system
Synchronization	User-oriented/ Sub-routine	Synchronize the notes between the client, server and other devices.

Detailed use case information is showed below:

Use Case Name: ViewAllNotes
Scope: System
Level: User-oriented
Participating Actors: Initiated by *User*
Involved User and Focus: Users: To have a look at his/her existing notes and make a choice to review.
Entry Condition: User clicks on the 'ViewAllNotes' button.
Exit Condition: List all the existing notes.
Main Flow:

1) User clicks on the 'ViewAllNotes' button.
2) The system shows all the existing notes. If *User* chooses a timeline view, then they will be shown in a timeline form. Otherwise, they will be shown in categories.
3) The name and a preview of the note are shown. The preview part is specified in *MakePreview*.
4) The system shows some statistical information, like the use time and frequency of each note by *User*
5) The system recommends some notes that *User* may want to look at.
6) *User* chooses one note.

Extended Flow:

1) User may want to create a new note. Then this use case is over and jumps to CreateNote.
2) User may want to search for the desired note. Then this use case is over and jumps to Search.

Special Requirements: None.
Frequency of Occurrence: Medium. Happens when *User* wants to look for one particular note.

Use Case Name: Search
Scope: System
Level: Sub-routine
Participating Actors: Initiated by *User*. Communicates with *LocalDatabase*.
Involved User and Focus: User: To look for desired notes by search keywords.
Entry Condition: User clicks on the ‘search’ button.
Exit Condition: Search results are shown. The system will show the search results according to keywords and other query conditions.
Main Flow:

1) User clicks the search box.
2) User enters the keywords.
3) User selects from a set of the advanced search options. For example, User may choose to search in one note or all notes. User may also choose to search the title or texts or other levels of elements specified by Markdown.
4) User submits the query.
5) The system sends query to LocalDatabase.
6) The system receives results from LocalDatabase.
7) The system shows the results in an orderly manner.

Extended Flow: No results from search. In this case, the system will show the message of ‘No results’ and asks User to try other keywords or search options.
Special Requirements: None.
Frequency of Occurrence: Medium. Happens when user wants to look for one note or a specific line in a note.

Use Case Name: CreateNote
Scope: System
Level: User-oriented
Participating Actors: Initiated by User
Involved User and Focus: User: User creates a new note.
Entry Condition: User clicks on the ‘New’ button.
Exit Condition: An empty note is created. This use case ends and jumps to EditNote.
Main Flow:

1) User clicks the ‘New’ button.
2) The system opens a window to let User enter the name, tag and category of the note.
3) User chooses the texting format, including Markdown and pure text.
4) The system saves the information and creates a new text file at the local.

Extended Flow: The name is the same as that of an existing note. The system will show the message of ‘Name not unique’ and asks User to choose another name.
Special Requirements: None.
Frequency of Occurrence: Low. Happens anytime when *User* wants to create a new note.

Use Case Name: EditNote
Scope: System
Level: User-oriented
Participating Actors: Initiated by User
Involved User and Focus: User: To edit the source file of an existing note, add new texts or delete some lines.
Entry Condition: The Note must have been created. To edit a note, that note must exist. That means, the note must have been created and have its necessary information like name and format recorded in the system.
Exit Condition: The source file of the note is updated. As User finishes the note, the new data is written to the local source file.
Main Flow:
1) User edits the source file, adding or deleting some texts/codes.
2) The system keeps record of User's modification history to make possible the function of Undo and Redo.
3) The system updates the change to the local source file.
4) The system generate a preview of the typeset note. This will be specified in MakePreview.
5) Return to step 1) if User doesn't finish typing.

Extended Flow: There are some syntax errors in User's texts. In Markdown mode, this may happen. The system checks the syntax of the texts and marks those that is incorrect. Preview will remain unchanged.
Special Requirements: Two modes. In Markdown mode, the syntax will be that of Markdown, while in pure text mode, the level of logic are specified by indents.
Frequency of Occurrence: High. Happens when User types.

Use Case Name: MakePreview
Scope: System
Level: Sub-routine
Participating Actors: Initiated by the system.
Involved User and Focus: System**:** The system will generate a preview of the typeset version of the existing notes, mainly in a HTML format. This use case is included in *ViewAllNotes* and *EditNote*.
Entry Condition: At least one note appears in the front page. The note here means the textual information of a note. Whenever it appears, this use case starts.
Exit Condition: A typeset preview of the note is generated. This preview may be a HTML view or a mind map.
Main Flow:

1) The system detects the textual information or a change of contents of a note in the front page.
2) The system checks if there exist an updated preview for the note.
3) If not, the system generates a typeset preview in HTML format according to the source file of the note.
4) The system saves the preview as a cache to the local.

Extended Flow:

1) There exist some syntax errors in the source file. User will be informed of the error. If there exist an earlier preview, then this one will be shown. Otherwise, No preview will be shown.
2) User may choose to view the note as a mind map. Then instead of a textual HTML view, a mind map will be generated.

Special Requirements: None.
Frequency of Occurrence: High. Happens anytime when *User* modifies one note.

Use Case Name: CloseNote
Scope: System
Level: User-oriented
Participating Actors: Initiated by User. Communicates with LocalDatabase.
Involved User and Focus: User: User finishes editing one note and wants to close it.
Entry Condition: A note is being edited. This note is the object that this use case deals with.
Exit Condition: The modification of the note is saved to local and cloud. In the editing part, the system will automatically updates the source file. In this part, the system will not only do this, but also synchronize with the ones on the cloud.
Main Flow:

1) User clicks the close button of the note or change to other pages of the App.
2) The system updates the local source file.
3) The system saves the generated view as a cache to the local.
4) The system updates the database of LocalDatabase by the source file.
5) The system synchronize the local change to the cloud. This part is specified in Synchronization.

Extended Flow: There are syntax errors in the source file. The system will remind User of the errors, but User can ignore it and still saves the file.
Special Requirements: None.
Frequency of Occurrence: Medium. Happens anytime when *User* finishes a note.

Use Case Name: ExportFile
Scope: System
Level: Sub-routine
Participating Actors: Initiated by User.
Involved User and Focus: User: After User sees the preview of the note, he/she can choose to save the view to the local disk or share it with others.
Entry Condition: A view of the note is successfully generated. The file to be exported is corresponding to the current view displayed to User. If the view cannot be generated, then it cannot be exported.
Exit Condition: A file containing the note's view is created.
Main Flow:

1) User clicks the 'Export' button.
2) User chooses the type of file to be exported, including HTML, pdf, or jpg.
3) The system processes the current view of the note, and generate a file containing it.
4) User chooses to save the file to local or share it through the Internet.

Extended Flow: None.
Special Requirements: None.
Frequency of Occurrence: Low. Happens when *User* finishes typing and wants to export a view file.

Use Case Name: Login
Scope: System
Level: User-oriented/Sub-routine
Participating Actors: Initiated by *User*. Communicates with *CloudServer*.
Involved User and Focus: User: To log in to get a full service.
Entry Condition: No account has been logged. If User wants to change the login account, he/she must first log out.
Exit Condition:

1) The system is in a logged-in status. Customized service like synchronization and detailed statistics will work then.
2) The UI returns to where the login action started. User can log in from any view of the UI and will be still at that view after login.

Main Flow:

1) The system opens a login window.
2) *User* enters the account name and the password.
3) The system transmit the login information to *CloudServer*.
4) *CloudServer* authenticates the login and returns a success message.
5) The system shows that *User* has logged in successfully.

Extended Flow:

1) Connection failure. If the system cannot connect to CloudServer, it will tell User to check the Internet connection and retry.
2) CloudServer returns a failure message. The account name and the password are not valid. The system shows the message of ‘Invalid account’ or ‘Wrong password’ to User and asks User to try again.
3) User has no account. User may have no account before and want to create one. In this case, the system will jump to a sign up window and asks User to fill in a sign up form, including account name, password and email address.

Special Requirements: None.
Frequency of Occurrence: As often as one uses the App. Login is not a Must for receiving local service, which do not need to interact with CloudServer, but we strongly recommend you log in to get a better using experience.

Use Case Name: Synchronization
Scope: System
Level: User-oriented/Sub-routine
Participating Actors: Initiated by *User* or the system. Communicates with *CloudServer* and *OtherDevice*.
Involved User and Focus: Users: To get the edited documents synchronized.
Entry Condition: No note is being edited. If a note is being edited, then synchronization may produce some errors or conflicts. Therefore, this situation is not allowed.
Exit Condition: Data at local and on the cloud is synchronized.This means that the local data and cloud data is the same. Note that databases of LocalServer will not be synchronized as the Search action only happens locally. Only the source files and necessary information will be synchronized.
Main Flow:
1) The system checks this source file and covers its counterpart on the cloud.
2) The system query CloudServer of changes not local and update the changes to local.
3) The CloudServer checks if there is any OtherDevice online and update the changes to it.

Extended Flow:
1) Connection failure. If the system cannot connect to CloudServer, it will tell User to check the Internet connection and retry.
2) User has not been logged in. The system will tell the User to log in and this use case jumps to Login. If User refuses to log in, then this use case ends.
3) One file is edited at the same time from multiple devices.The system will send a ‘Conflict’ message and asks User to choose one version or merge the different versions.

Special Requirements: Databases of *LocalServer* will not be synchronized as the *Search* action only happens locally. Only the source files and necessary information will be synchronized.
Frequency of Occurrence: Medium. Happens when *User* closes one note or want to update local file from other devices.

The detailed use case diagram is showed below.

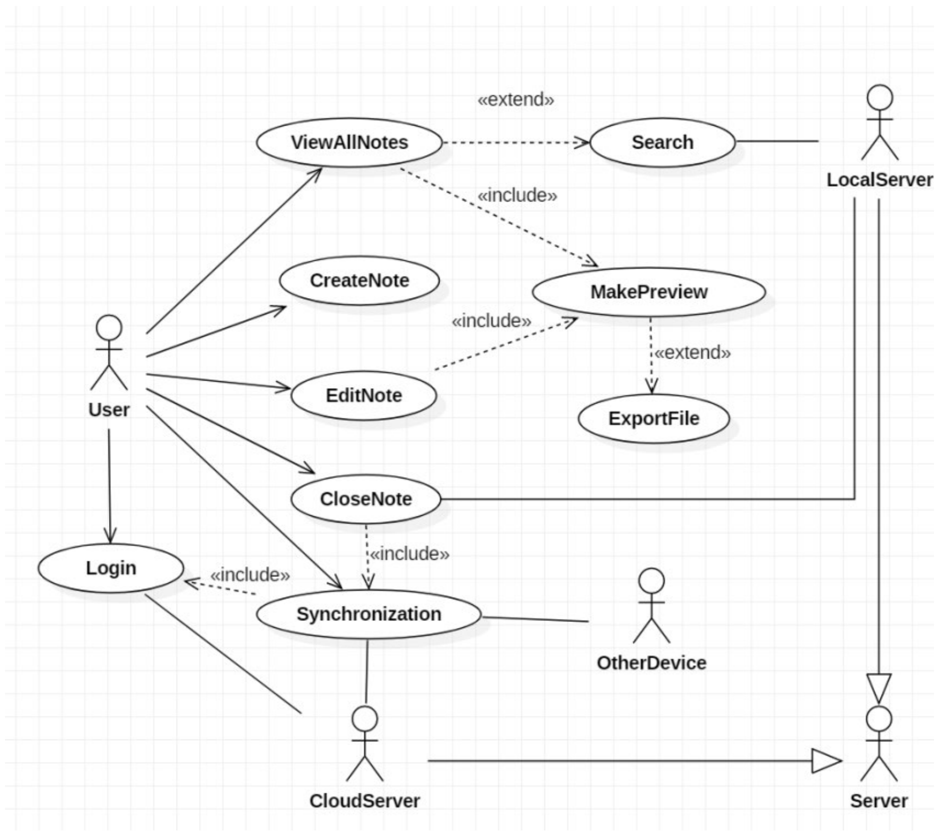

Figure 1: Use case diagram

3.4.3 Object Model

Table: Entity Class Defintion

Entity Class Name	Definition
User	The account info and preference of a user.
Note	A note containing its source text and record.
NoteList	A list of notes, containing only the basic information like its pointer, but not the source text
Cloud	Contains the user information and their notes from different devices.

Table: Boundary Class Definition

Boundary Class Name	Definition
HtmlPage	The page to preview html output of notes
EditingPage	The page to edit notes
DirectoryPage	The page to list existing files
SearchPage	The page to search key words and show results
RecordPage	The page to show learning record
LoginPage	The page to login
NavigationBar	The page to show most operations user may select
MindmapPage	The page to show the mind map
HomePage	The homepage, showing the most information

Table: Control Class Definition

Control Class Name	Definition
NoteControl	Control the process of editing notes and exchange information with SearchServer.
PreviewControl	Control the process of generating previews
SyncControl	Control the process of login and synchronization
UserInfoControl	Control the user account and preference
SearchServer	Control the process of storing and retrieving notes info.

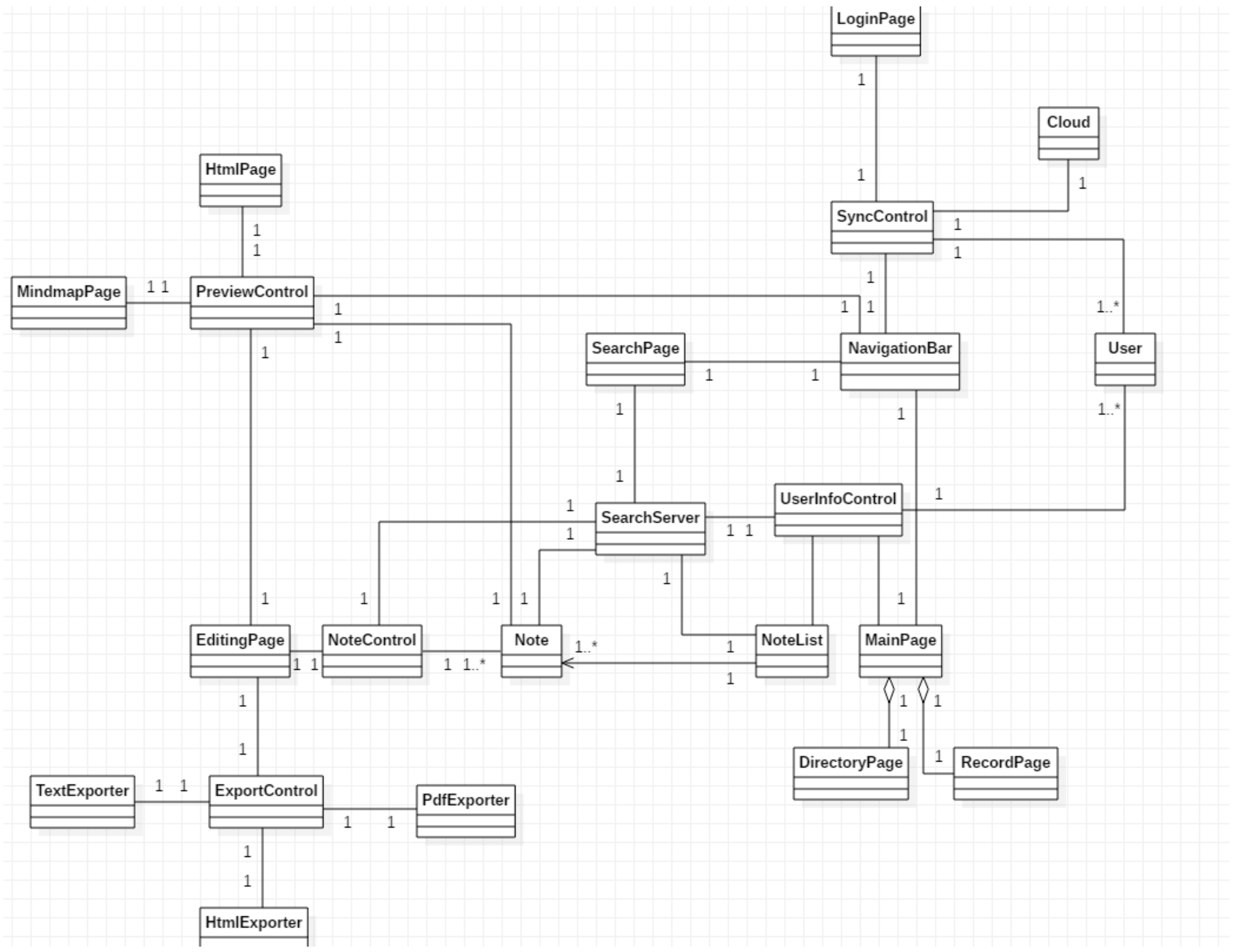

Figure 2: Class diagram

3.4.4 Dynamic Model

1) system sequence diagram and contract

Use Case 1: ViewAllNotes

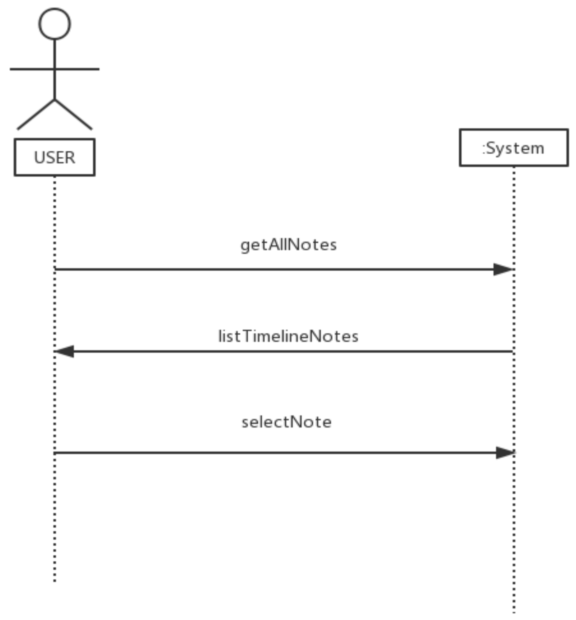

Figure 3: ViewAllNotes system sequence diagram

Use Case 2: Search

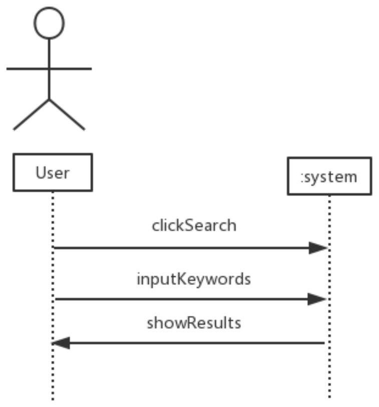

Figure 4: Search system sequence diagram

Use Case 3: CreateNote

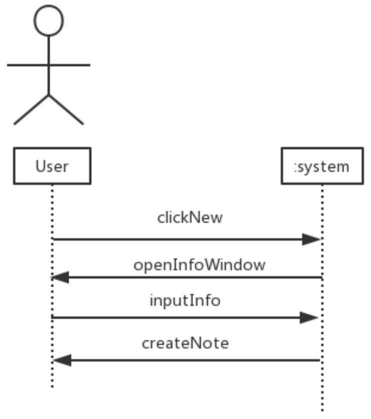

Figure 5: CreateNote system sequence diagram

Contract 1: inputInfo

Opeartion: InputInfo (name: noteName, type: noteType, addtime: addTime, modifieddate: modifiedTime, content: Content)
Cross Reference:
createNote
Entry Condition:
The user clicks "CreateFile" button.
Exit Condition:
1) Create the instance of Note, n.
2) n.noteName is set as name.
3) n.noteType is set as type.
4) n.addTime is set as addtime.
5) n.modifiedTime is set as modifiedtime.
6) n.Content is set as content.

Use Case 4: EditNote

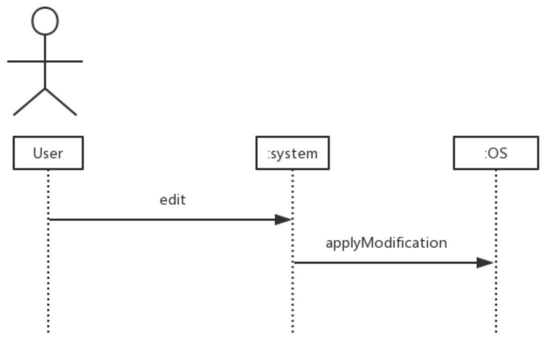

Figure 6: EditNote system sequence diagram

Contract 2: applyModification

Opeartion: applyModification (new_content: content)
Cross Reference: EditNote
Entry Condition:
1) The user is in the editing page.
2) The document that has been modified is doc.

Exit Condition: doc.content is set as new_content

Use Case 5: MakePreview

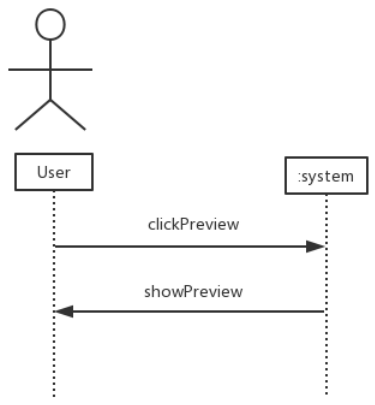

Figure 7: MakePreview system sequence diagram

Use Case 6: CloseNote

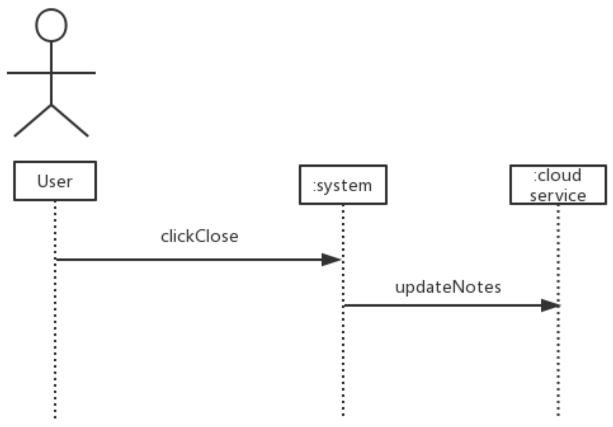

Figure 8: CloseNote system sequence diagram

Use Case 7: ExportFile

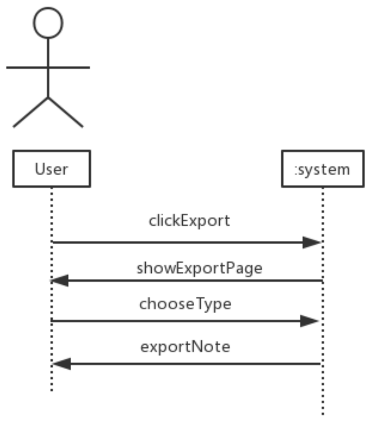

Figure 9: ExportFile system sequence diagram

Use Case 8: Login

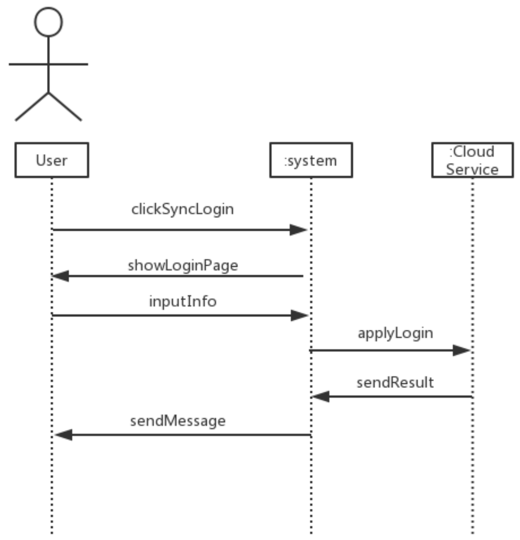

Figure 10: Login system sequence diagram

Contract 3: applyLogin

Opeartion: applyLogin(userid: userid, pwd: password)
Cross Reference: Login
Entry Condition:
The user.userid and user.password aren't empty.
Exit Condition:
1) cloud.userid is set as user.userid.
2) cloud.password is set as user.password.

Use Case 9: Synchronization

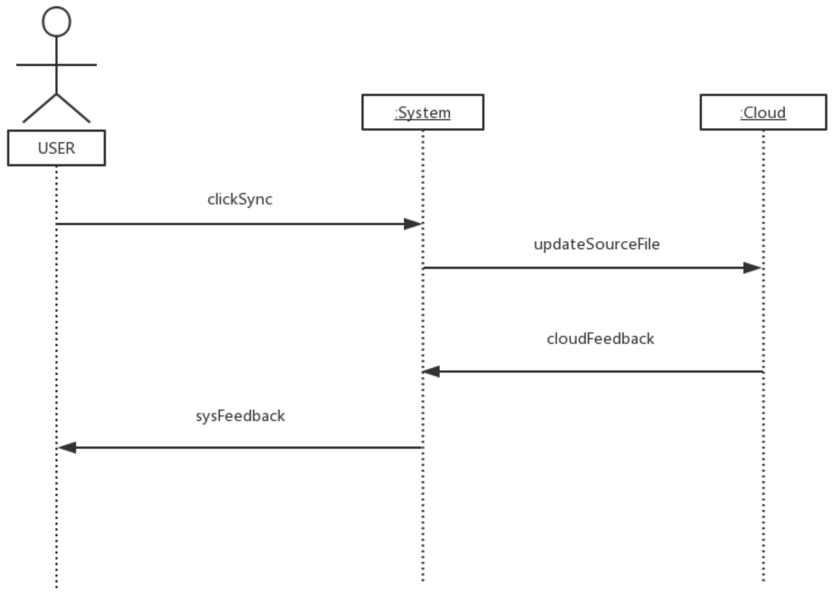

Figure 11: Synchronization system sequence diagram

Contract 4: updateSourceFile

Opeartion: updateSourceFile (userID: UserID, password: Password, content: Newcontent) **Cross Reference:** Synchronization **Entry Condition:** User clicks on the synchronization button. **Exit Condition:** 1) System sends UserID, Password and Newcontent to the cloud. 2) The library of the user is logged in and update the content with New content. 3) The cloud send message to the system and system informs the user of the synchronization state.

2) system sequence diagram

3) Use case realization: ViewAllNotes

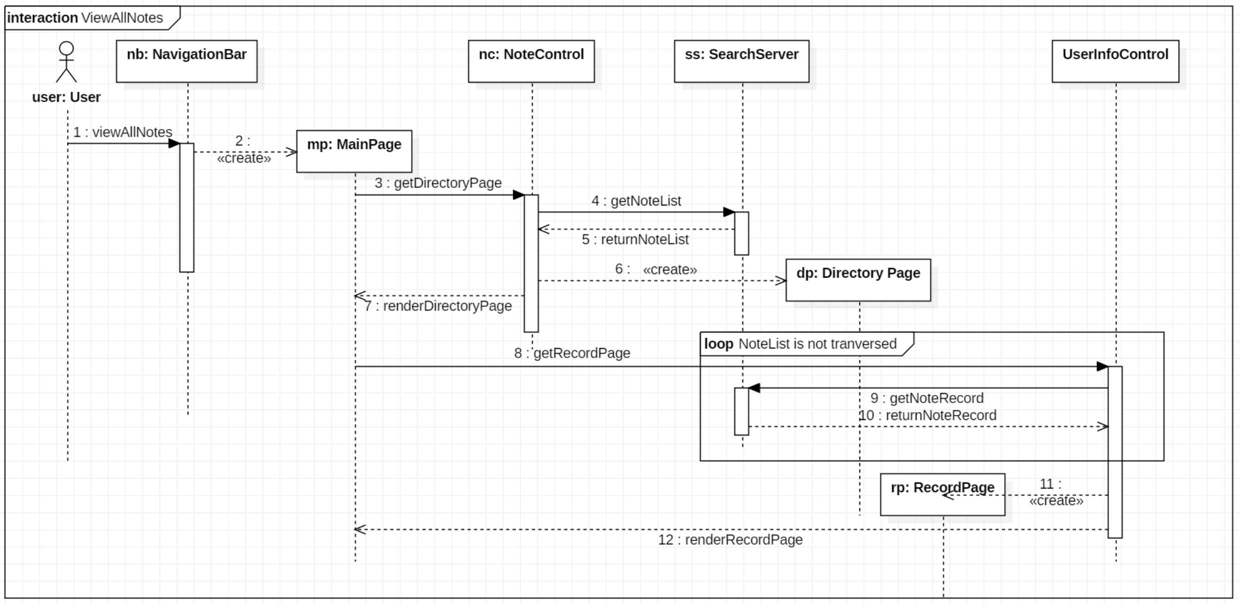

Figure 12: Realization of use case ViewAllNotes

4) Use case realization: Login

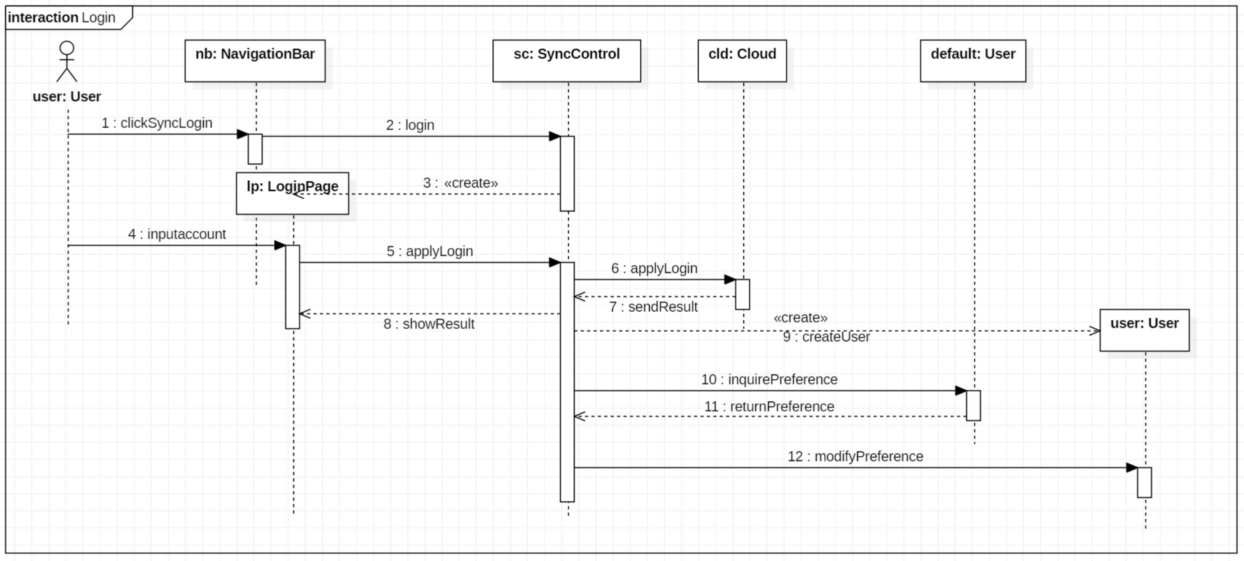

Figure 13: Realization of use case Login

5) Use case realization: Synchronization

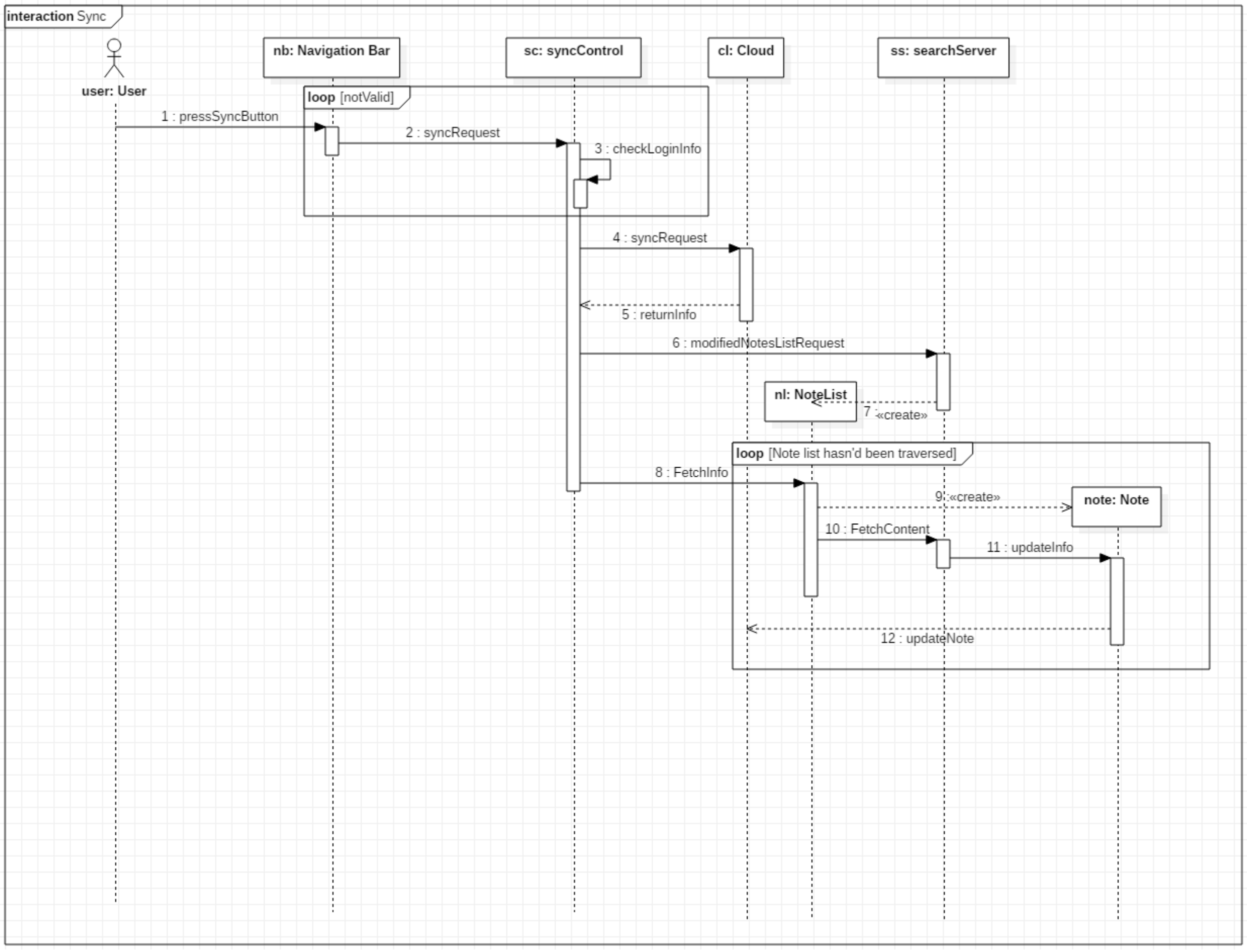

Figure 14: Realization of use case Synchronization

6) Use case realization: CreateNote

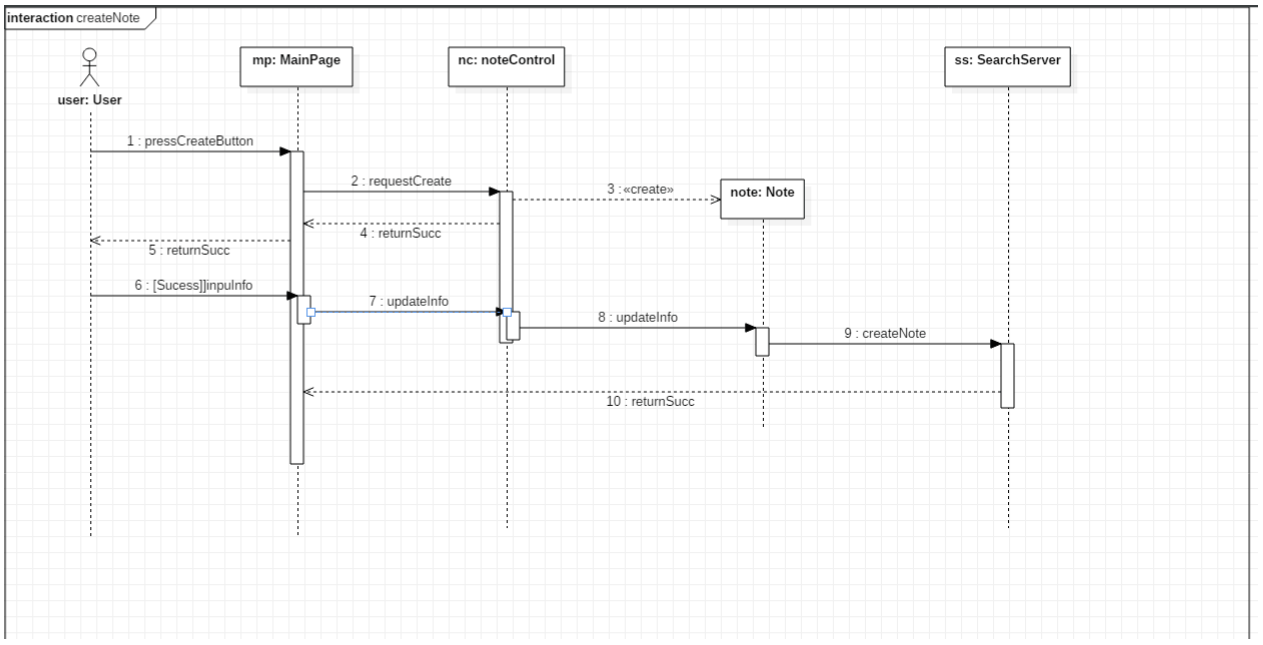

Figure 15: Realization of use case CreateNote

7) Use case realization: EditNote

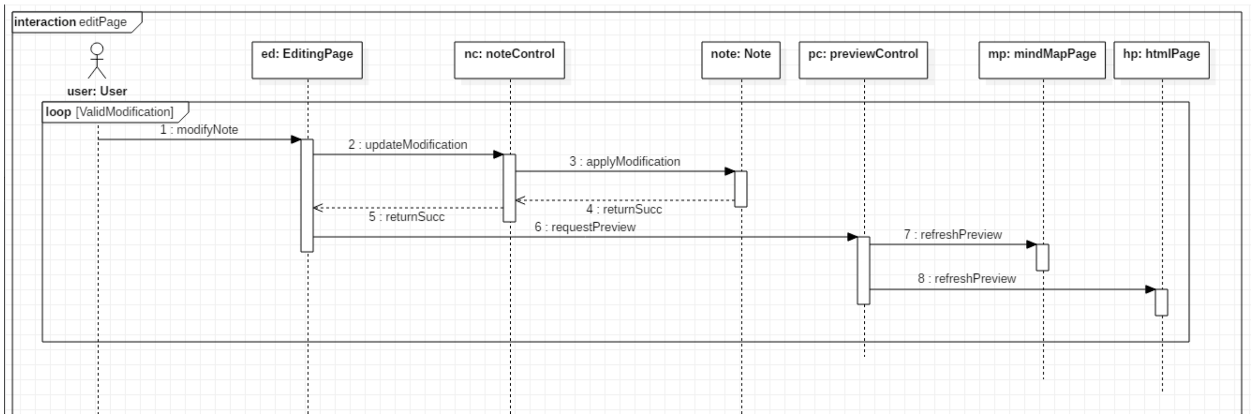

Figure 16: Realization of use case EditNote

8) Use case realization: CloseNote

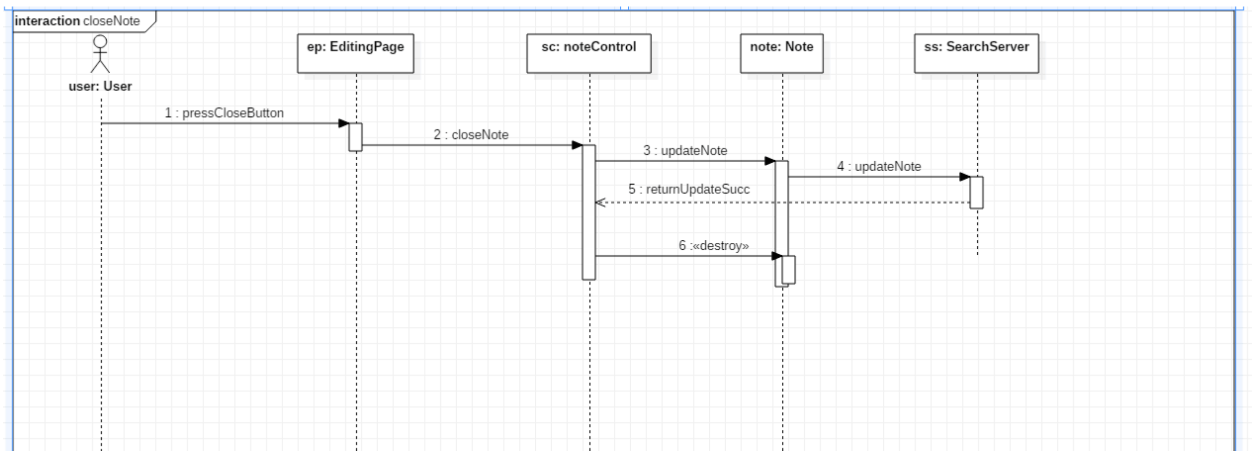

Figure 17: Realization of use case CloseNote

9) Use case realization: MakePreview

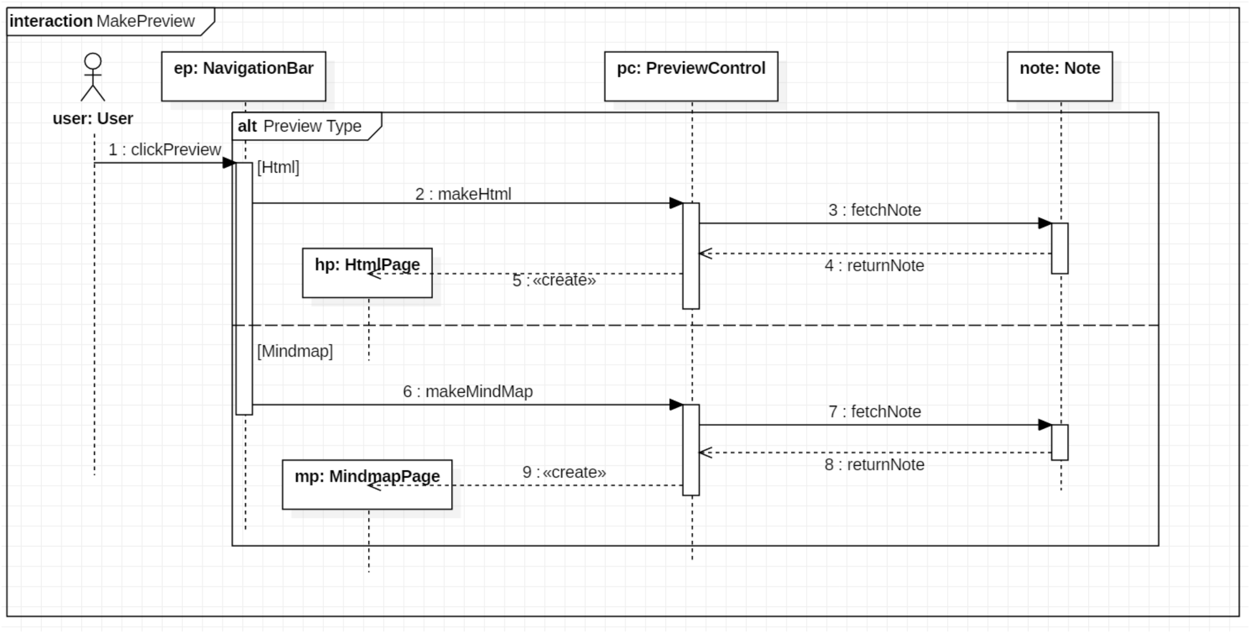

Figure 18: Realization of use case MakePreview

10) Use case realization: ExportFile

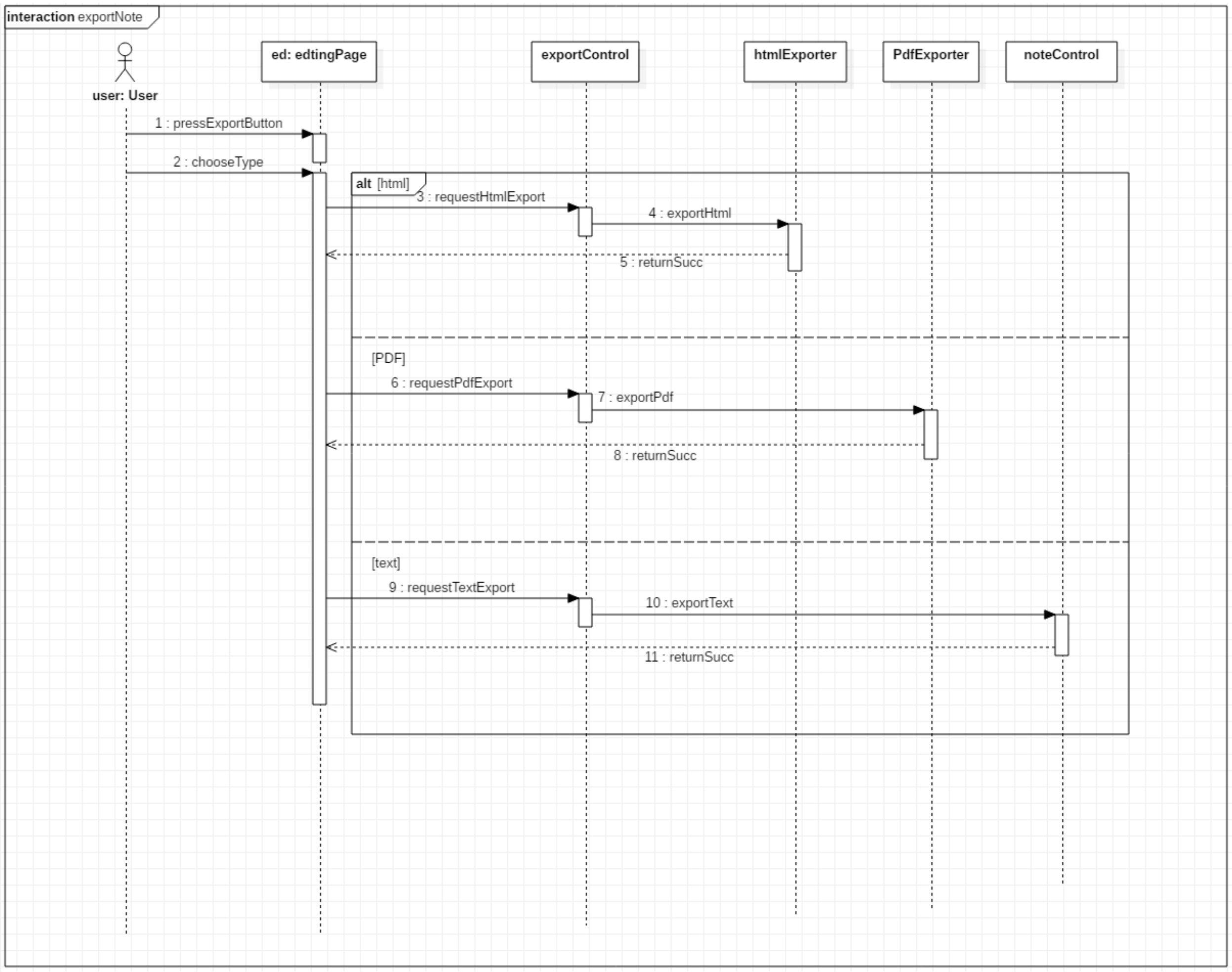

Figure 19: Realization of use case ExportFile

11) Use case realization: Search

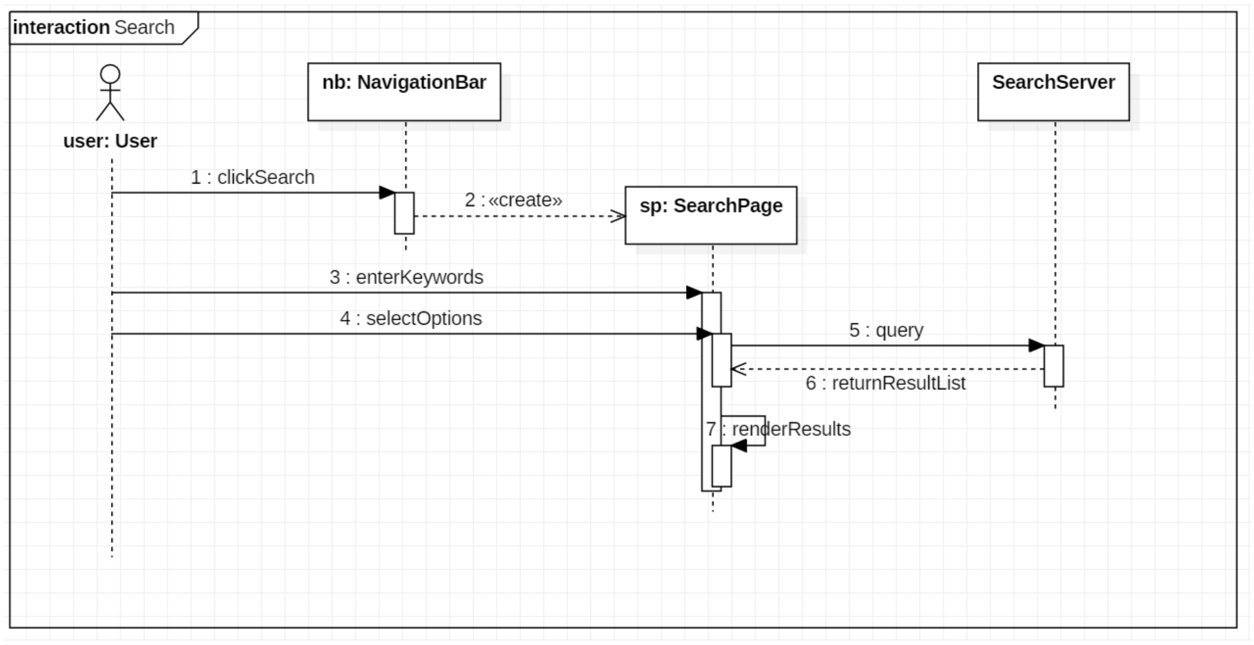

Figure 20: Realization of use case Search

12) State Diagram

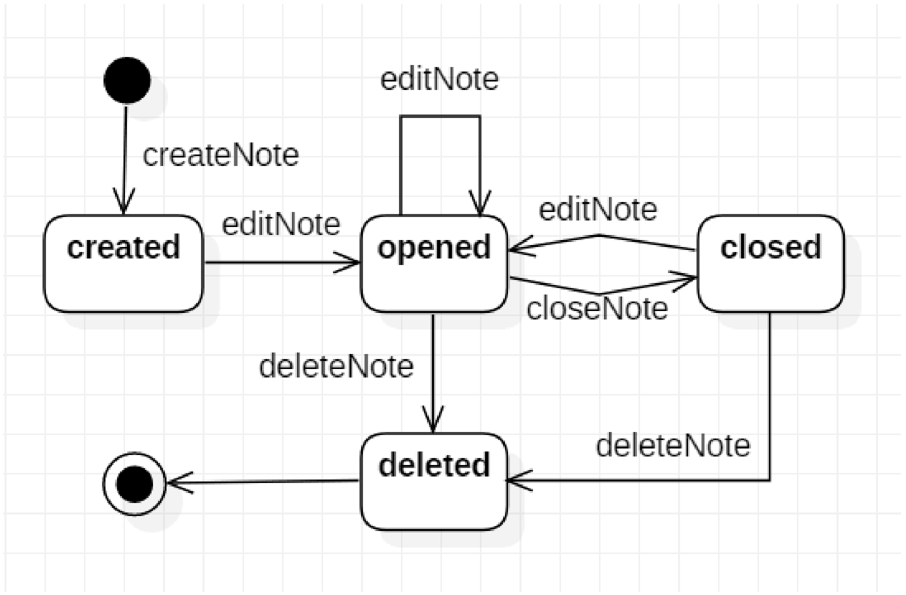

Figure: State Diagram

3.4.5 User Interface

According to the core function of the system, the user interface is supposed to contain HomePage, LoginPage, PreviewPage, EditingPage, DirectoryPage, SearchPage, RecordPage, MindmapPage, etc. The detailed page design is showed below:

1) HomePage: This page displays the name of the software, group name, the login button, etc.
2) LoginPage: This page is responsible for users' login operation. Users can enter the account ID and password to login or register.

3) PreviewPage: When user clicks on one of his/her notes, a hovering preview page will pop out and displays the abstract of the notes selected.
4) EditingPage: This page is mainly for users to edit their notes. It contains save button, synchronization button and so on.
5) DirectoryPage: This page lists all the files created by the user.
6) SearchPage: This page mainly serves as the searching function for users. Users can enter the keyword and will get the result on this page.
7) RecordPage: On this page, users can find their learning record and the editing history.
8) MindmapPage: This page will generate relevant mind maps corresponding to the notes selected by the user.