4DocNo: 001.C.4:1

X-Note System Architecture Design 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
04/17/2019	1.0	Add the runtime, deployment and physical requirement part	Shixuan Gu
04/18/2019	1.0	Add some diagrams.	Jingyu Li
04/20/2019	1.0	Add the subsystem part	Yu Fan

Contents

1. Introduction

1) Purpose

The goal of writing this System Architecture Design is to describe and define the system architecture in the XNote software. Based on the previous requirements specification and system analysis, various models are used to demonstrate the system structure in detail for the subsequent software implementation. The document will define the design objectives of the system architecture, describe the system structure, subsystem definition, software and hardware deployment,data management, software control, boundary conditions, etc.This document is used by the development team to define and implement the architecture and design of the system, and develop the software based on this document.

2) Range of Application

The document is used for the software: XNote
The features, subsystems, models, and so on associated with the software conform to the content of this document.

3) Definition

The terms involved in this document are defined in the project vocabulary (glossary.docx)

4) References

"Object-oriented software engineering -- using UML, patterns and Java" (3rd edition), tsinghua university press, 2011

5) Overview

This document includes four parts: introduction, current system architecture, system architecture design objectives, the proposed software system architecture. Current system architecture analyzes the current online note-taking software and points out its shortcomings. System architecture design objective section is combined with software requirements, and the objectives of the system design are listed. The proposed software system architecture gives the system architecture and subsystem decomposition, and express the system object design, software and hardware deployment, data management, software component control, boundary conditions, etc. in the way of combining text and model diagram The various parts of this document are closely related, complementary and contrasting, presenting the software architecture together.

2. Current System Architecture

The existing note-taking software, like One Note, doesn’t support much functions, like producing mind maps based on user’s notes, show the user their notes in time order and compat notes in markdown form. So it is inefficient to use this software, which imposes restrictions on the scale of the users. That is why a new kind of note-taking software is needed.

3. System Architecture Design Objectives

The design objectives of the system architecture are as follows:
(1) High availability: as a note taking software, the software should support various kinds of note form and can be used in different platforms.
(2) Security: the system is registered as a real-name system, and important personal information is kept in the database, so system security must be guaranteed.
(3) High performance: the system may deal with different kinds of notes in one document, so it requires high performance.
(4) Scalability: when the scale of the system is small in the initial stage, relatively high-specification hardware is not required.But it should be taken into account as the expansion of a system as the number of users increases.

4. Proposed Software System Architecture

4.1 Overview

Architecture of the system and its adoption:
1) Three-layer architecture
The system uses three levels to organize the subsystem: the first layer is the user interface layer, providing an interface to interact with the user, namely the software UI. The second layer is the application logic layer .This layer is mainly responsible for controlling and implementing system functions, including UserManagement, NoteManagement, and PreviewManagement three subsystems, which are responsible for user information, note editing and control. The third layer is the cloud service and local service layer. Mainly responsible for note synchronization and note search and note storage.
The three-layer architecture is relatively mature, which separates the interface and application logic, has low coupling, strong flexibility and expansibility, and is convenient for development and division of process. Therefore, this software adopts this structure.
2) Model-View-Controller mode (MVC)
The MVC pattern applies to interactive systems. In the above three-tier architecture, the system has been divided into views (the first tier),the controller (level 2) and the model (level 3) three parts.
This system is developed independently, and the third party class library will be used to support the realization of interactive communication between the two parties.
This system contains the following subsystems:
(1) UI: responsible for the user interface section
(2) UserManagement: responsible for user information management, control login, registration, personal data modification and other functions
(3) NoteManagement: responsible for the note creating, modification, deleting and the note list generation.
(4) PreviewManagement: responsible for the mindmap file, PDF file, HTML file and Text file generation and offers the user to preview them.
(5) CloudService: responsible for the note cloud synchronization and offer the user to download the cloud files.
(6) LocalService: responsible for the note storage and note search.

4.2 User Case View

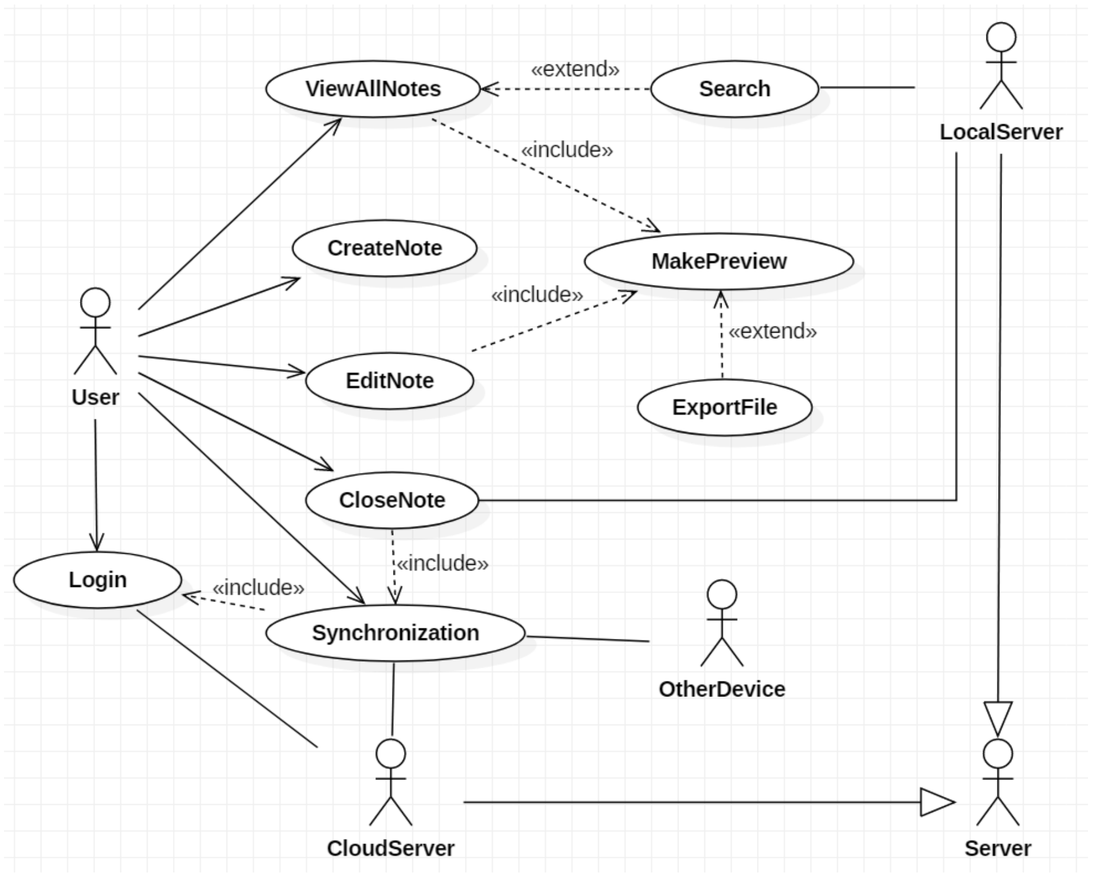

Figure 1: Use case diagram

4.3 System logic View

4.3.1 System Architecture

This system adopts three-layer architecture and client/server mode. At the top is the UI layer, the client section users interact directly. The middle layer is the application logic layer, which contains the subsystems to realize the system functions. The third layer is the public service layer, achieving data storage and management, mobile terminal and server communication functions.

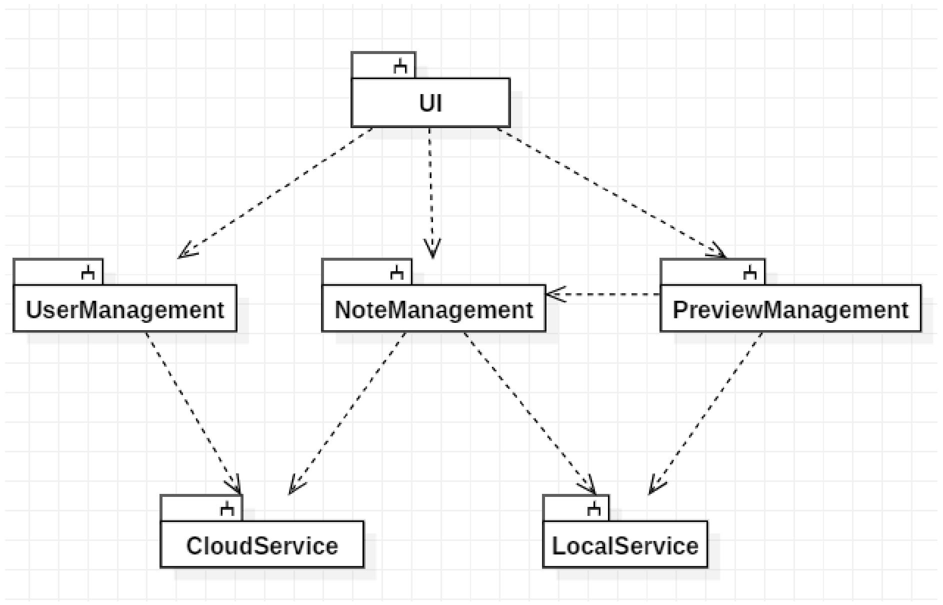

Figure 2: System decomposition

4.3.2 Subsystem

(1) UI System
Function: interaction with the user.
Service: The UI layer offers various kinds of input boxes, buttons and text boxes, which are used for information display and interaction with the user.
Class diagram:

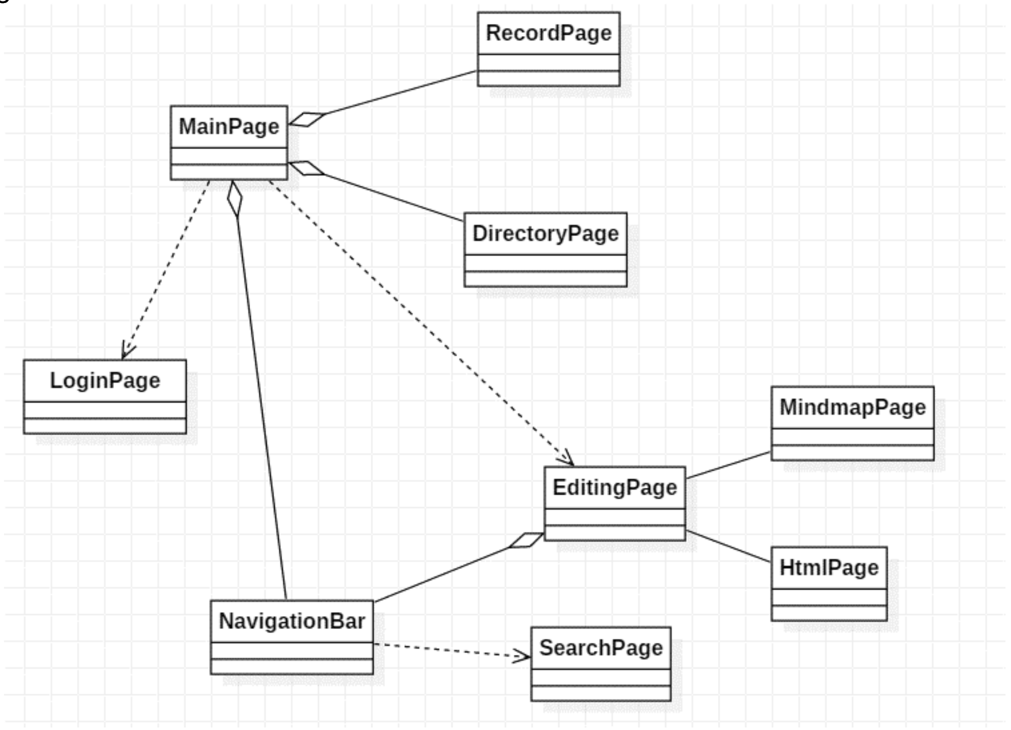

Figure 3: Class diagram of subsystem UI

Component diagram:

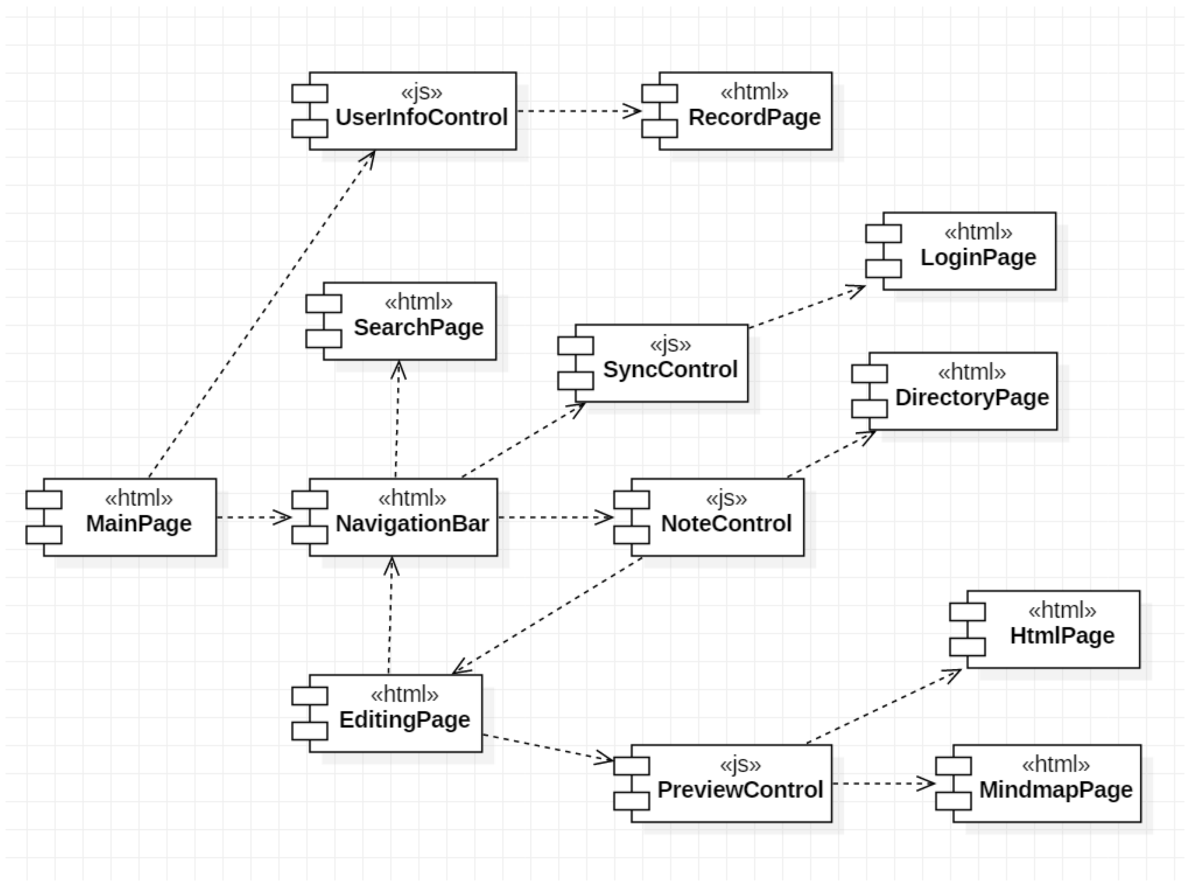

Figure 4: Component diagram of UI

(2) UserManagement System
Function: user information management, control login, register, and personal information modification.
Service:
public void register()
public int submitUserInfo(String username, String password, String mail, String phoneName, String passwordConfiramation)
public int login(String username, String password)
public void show PersonalInfo()
public int updatePersonalInfo(String userName, String password, String mail, String phoneNumber, String passwordConfirmation)
Class diagram:

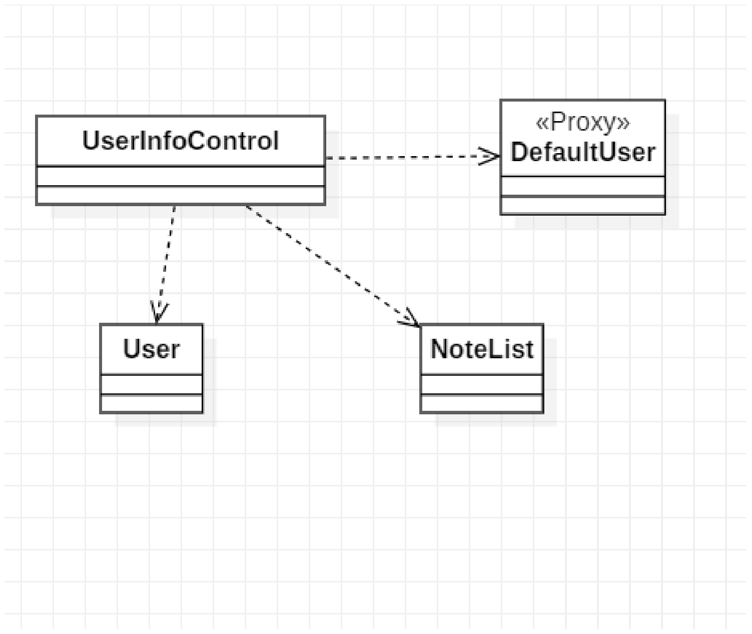

Figure 5: class diagram of subsystem user management

Component Diagram:

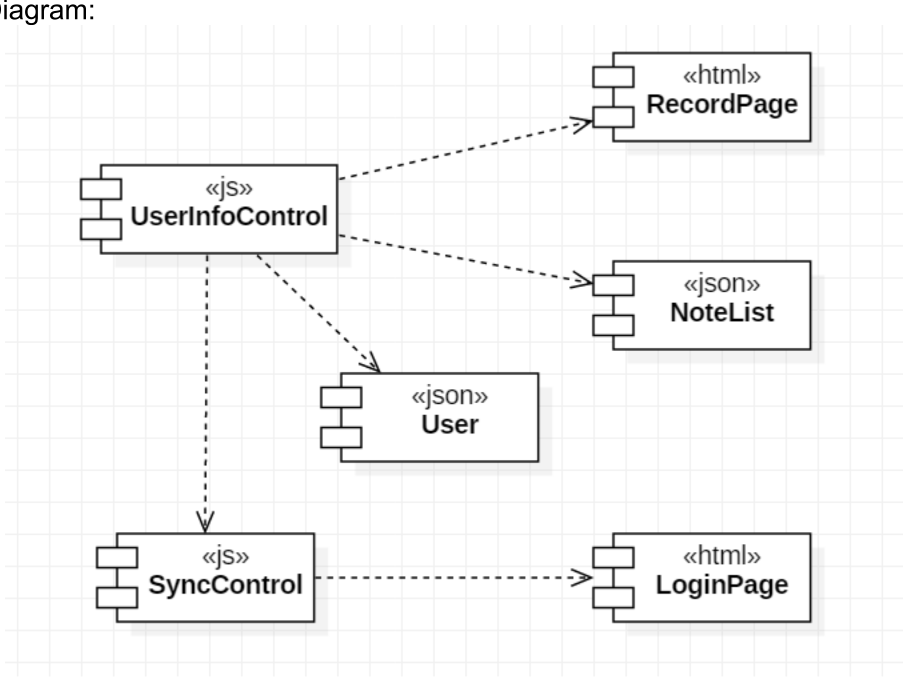

Figure 6: Component diagram of subsystem user management

(3) NoteManagement System
Function: user note management, including creating note, modifying note, deleting note and generating the list of when the user creates the note and the last time of the note modification.
Service:
public void createNote()
public void modifyNote(String NoteTitle, String modificationContent, String modifyTime)
public void deleteNote(String NoteContent, String NoteTitle)
public void modifyNoteList(String NoteTitle, String modifyTime, String NoteList)
public void viewNote(String NoteTitle, String NoteContent)

public void viewNoteList(String NoteList)
Class diagram:

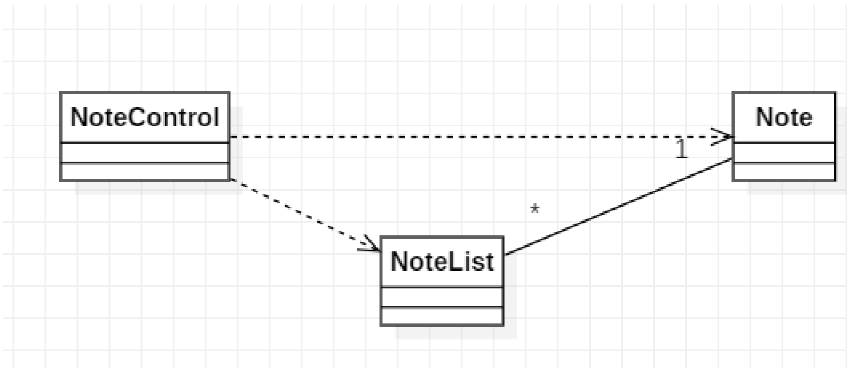

Figure 7: Class diagram of subsystem note management

Component diagram:

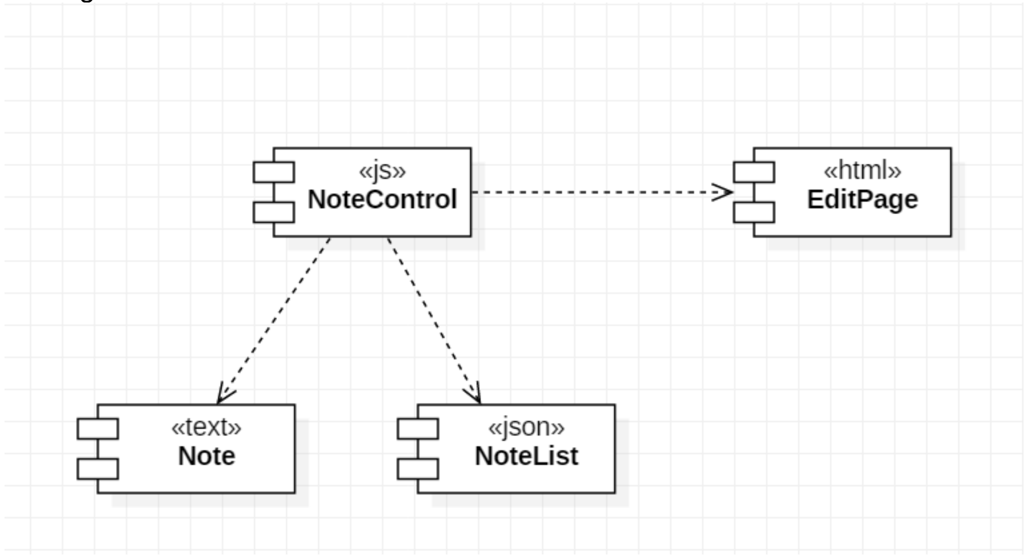

Figure 8: Component diagram of subsystem note management

(4) PreviewManagement System
Function: based on the content of the note and the note list, the software can generate mindmap, pdf file, html file, and text file automatically.
Service:
public void MindmapExporter(String NoteContent)
public void PDFExporter(String NoteContent)
public void TextExporter(String NoteContent)
public void HTMLExporter(String NoteContent)
Class diagram:

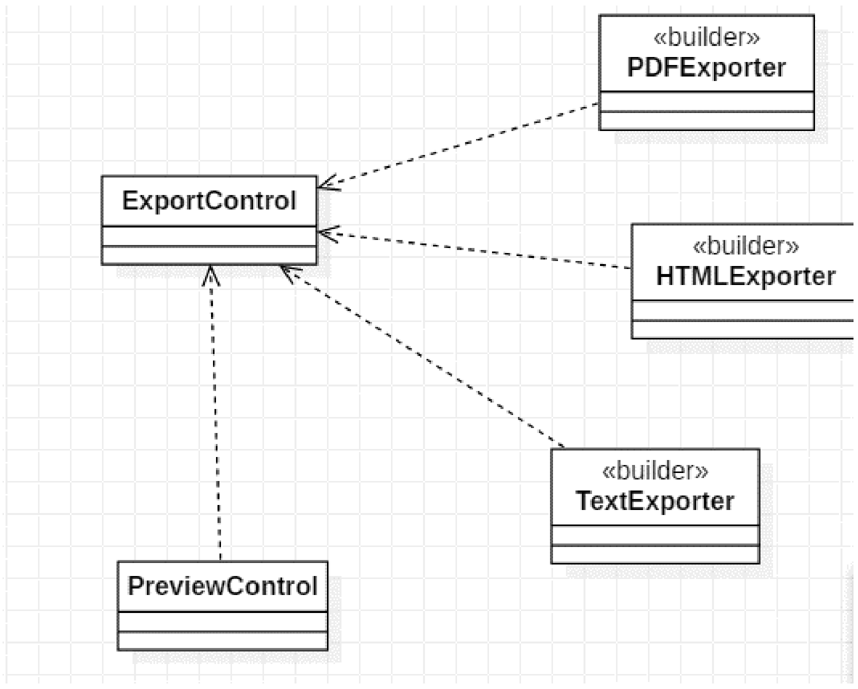

Figure 9: class diagram of subsystem preview management

Component diagram:

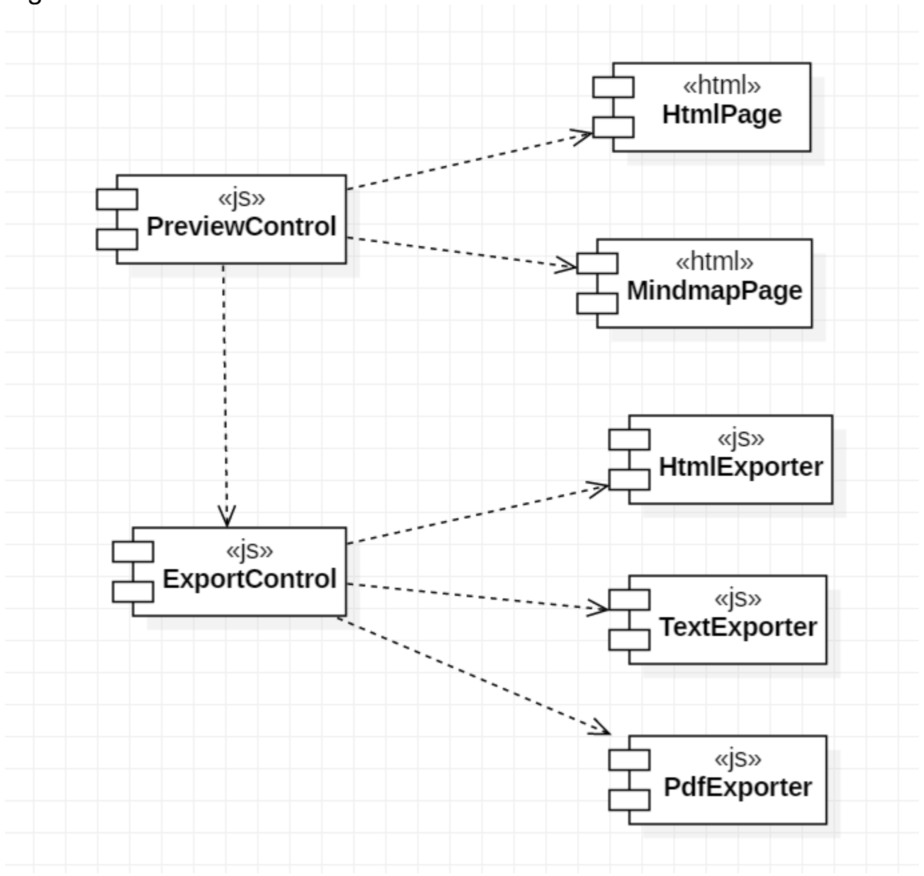

Figure 10: Component diagram of subsystem preview management

(5) Cloud Service

Function: synchronize user's note, note list and study record to the cloud, and offer user the function to download cloud note and notelist.
Service:
public void syncNote(String NoteContent, String NoteTItle)
public void syncNoteLIst(String NoteList)
public void syncStudyRecord(String modifyTime, String NoteTItle)
public String downloadNote(String NoteTitle, String Notecontent)
public String downloadNoteList(String NoteList)
public String downloadStudyRecord(String modifyTime, String NoteTitle)
Class diagram:

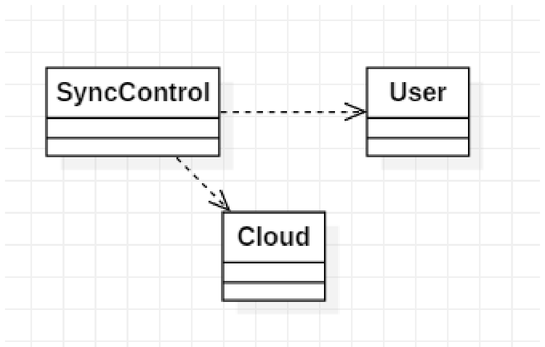

Figure 11: Class diagram of subsystem cloud service

Component diagram:

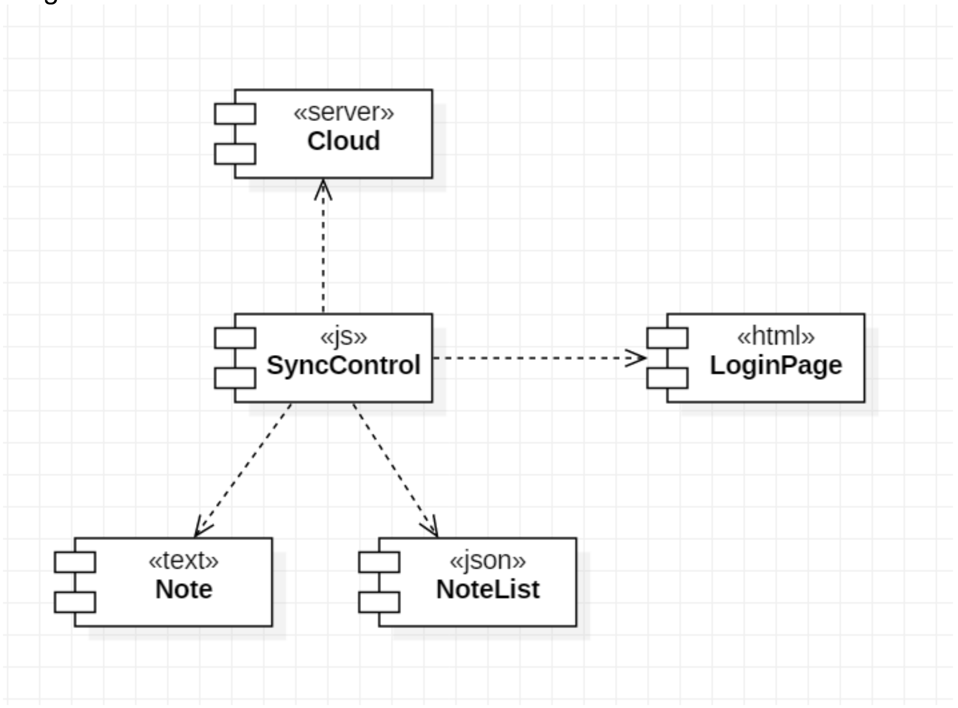

Figure 12: Component diagram of subsystem cloud service

(6) Local Service
Function: allow user searching keywords in all his/her notes and return all the results containing the keywords.
Service:
public String inputKeywords(String Keywords)
public String Search(String Keywords, String Note, String NoteList)

public void showResults(String Results)
Class diagram:

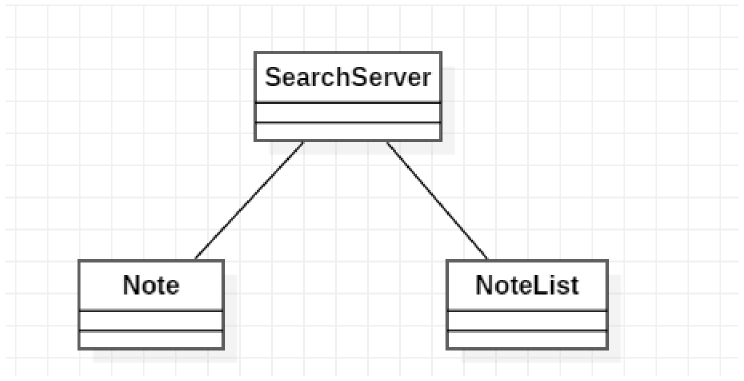

Figure 13: Class diagram of subsystem local service

Component diagram:

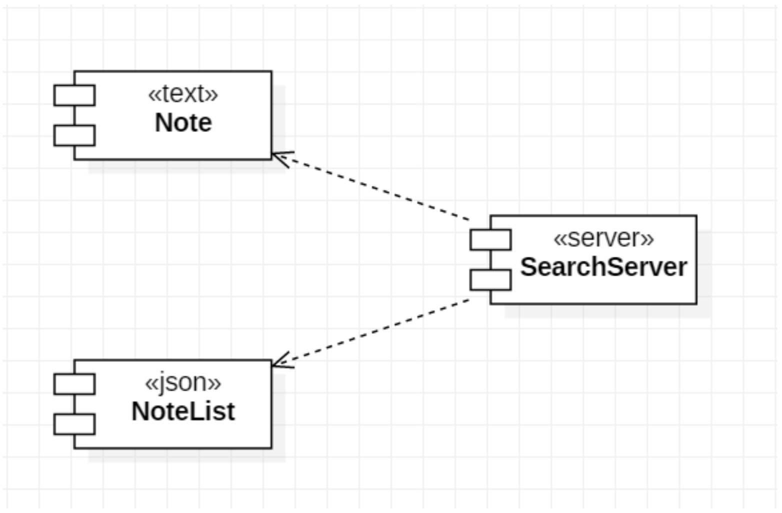

Figure 14: Component diagram of subsystem local service

4.3.3 Use Case

Our central use case in this system is the editingNote, makePreview and exportFile. In makePreview, **Command Pattern** is used to generate either html view or mind map view. In exportFile, **Builder Pattern** is used to export different types of files. In addition, in case the user of our App cannot login for some reasons, we set a local **proxy** user to act the same as the logged user, except for synchronization.

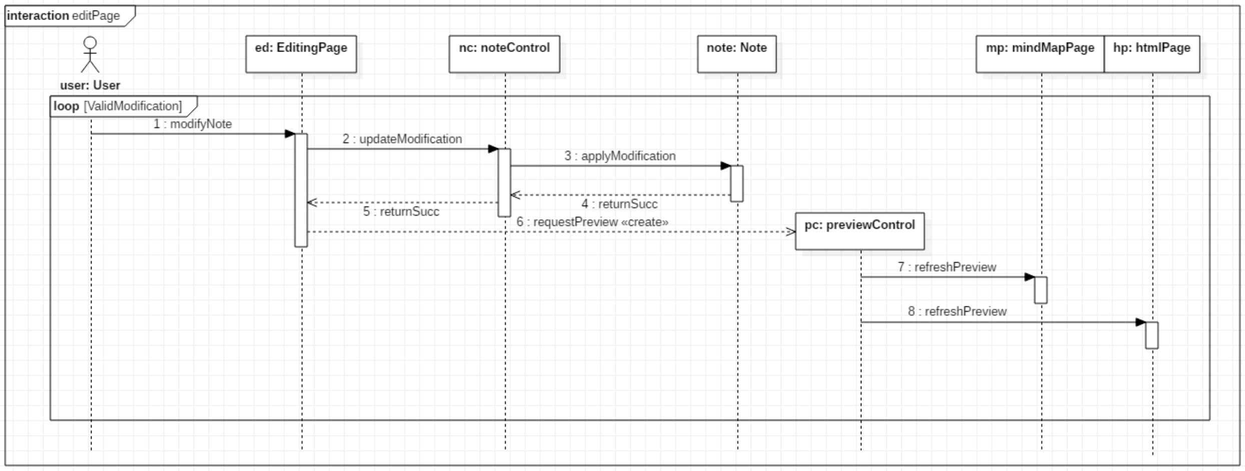

Figure 15: Realization of use case EditNote

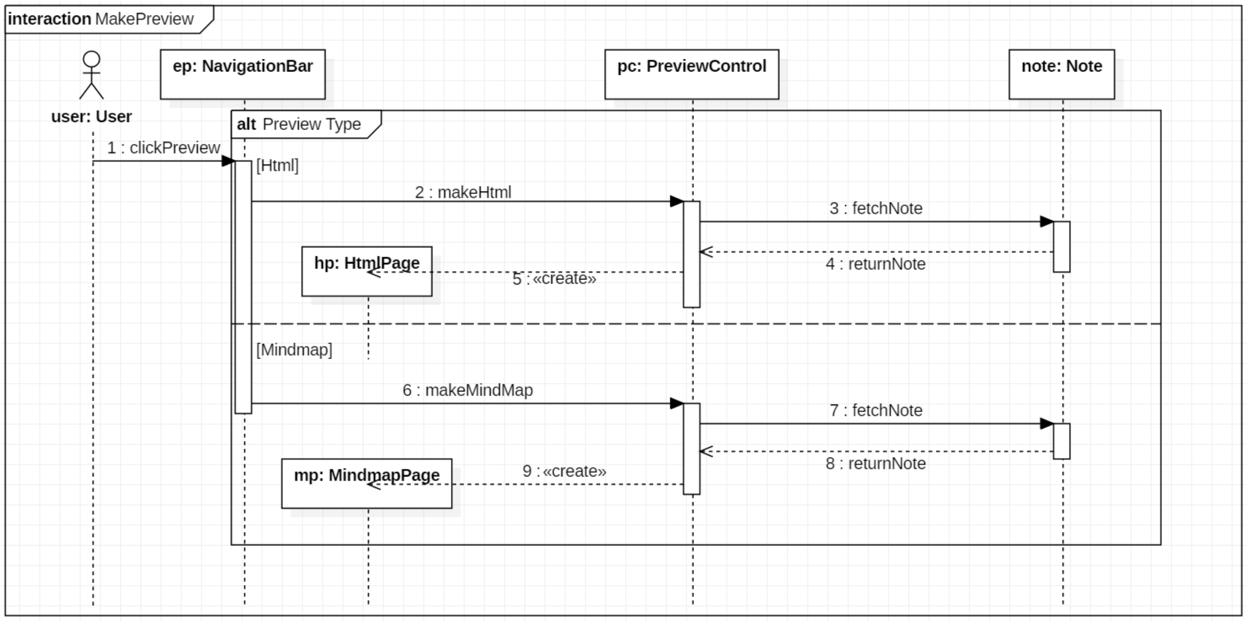

Figure 16: Realization of use case MakePreview

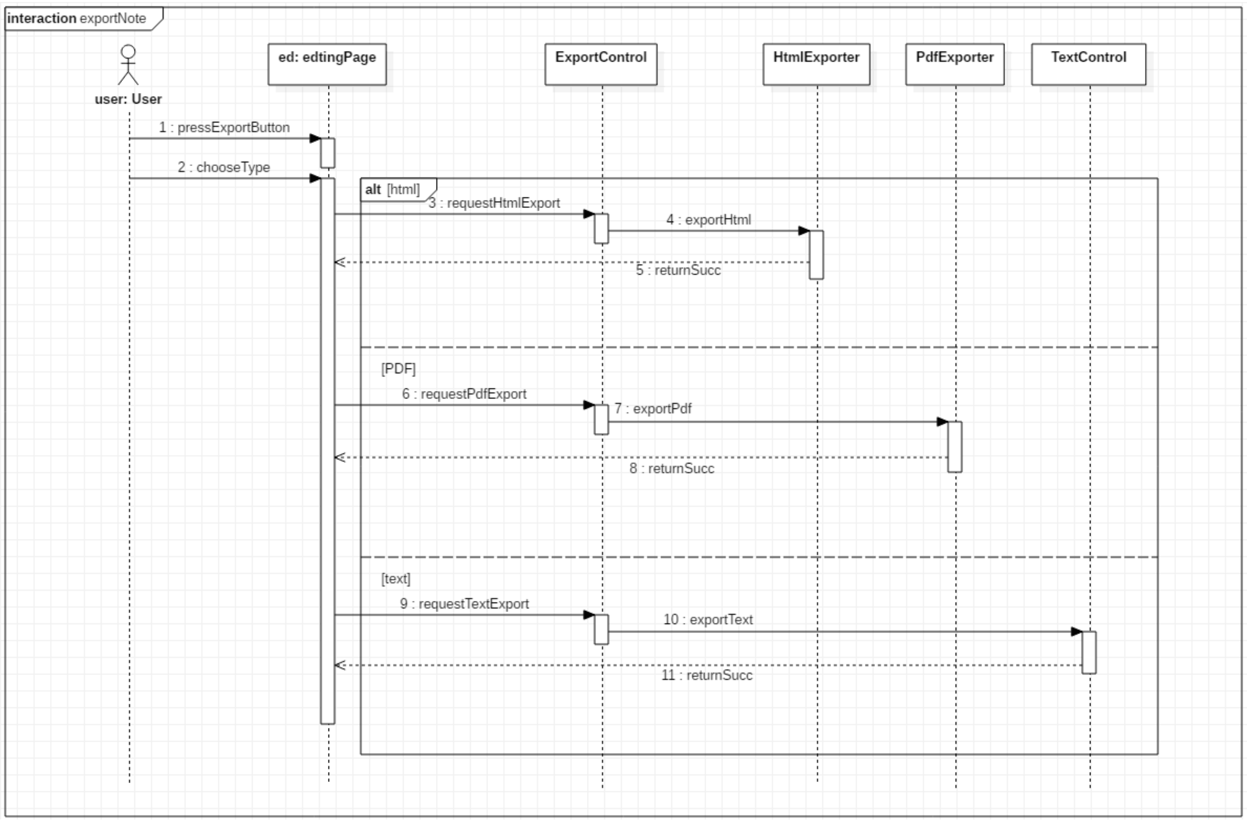

Figure 17: Realization of use case ExportFile

4.4 Subsystem Collaboration

The core use case editing note’s subsystem collaboration is as figure showing.

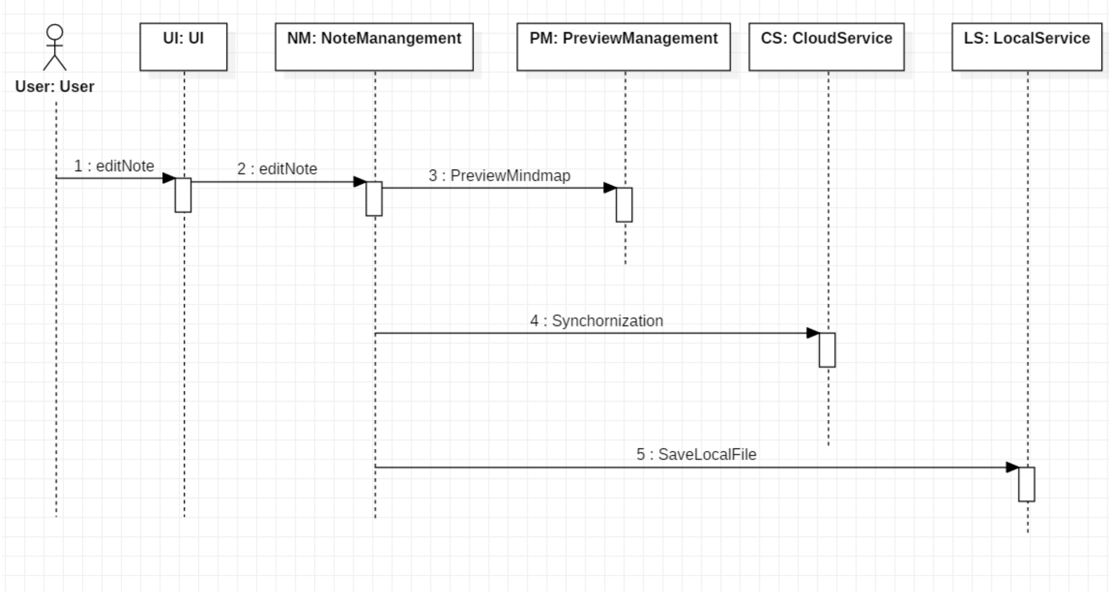

Figure 18:core use case editingNote's subsystem collaboration

4.5 System Process View

Since we will use Electron framework which creates two processes at runtime: Main Process to run the main app and Render Process to show UI, so we have two processes. We believe that some but not all the controllers should be assigned a thread.

For example, NoteControl, SyncControl and SearchServer should be assigned a thread so that synchronization can be automatically triggered when user is editing a note. UserInfoControl does not have a thread as it can be activated by the SyncControl. PreviewControl does not have a thread either, as it is activated by NoteControl. Those two control don't need to be working all the time.

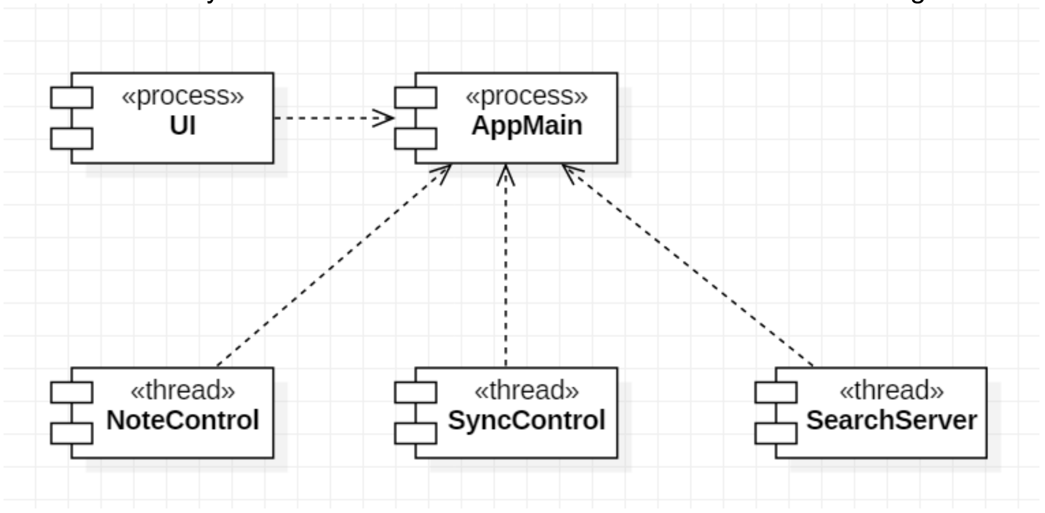

Figure 19: System runtime diagram

4.6 System Implementation View

1) System developing environment
   Developing environment: windows.
   Developing language: javascript
2) System developing model
   In this system, we made the definition of relevant software file according to the component graph, so the system developing model may comply with the component graph in the previous section.

4.7 System Physical View

The system is a PC application, so the main App runs on a desktop computer. There will be a cloud server to ensure the cloud storage feature. Local service including local storage and search will be deployed on the same computer where the App runs.

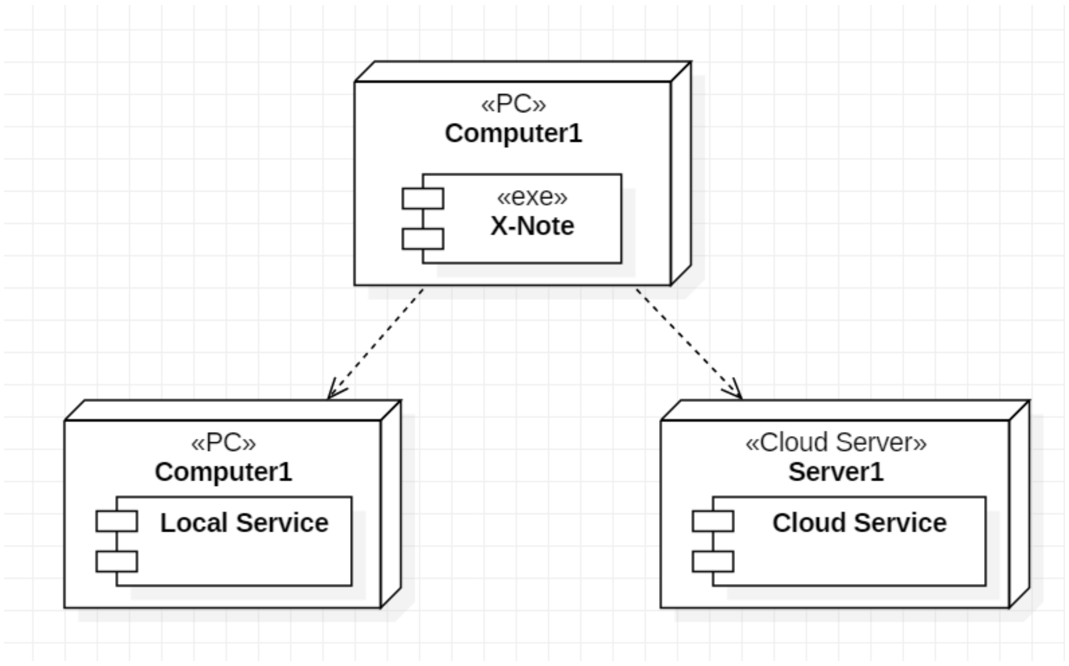

Figure 20: System deployment diagram

Hardware support required:
1) User machine: A computer with windows operation system and at least 1G memory and 1G hard drive space is required.
2) Network: network connection is required and the bandwidth don't need to be very large. The synchronization function will require certain amount of data transformation with the cloud servicer, but since the text file is transmitted, no much bandwidth resource is required.

4.8 Edge Condition Design

We consider the edge condition by defining start and end cases.
1) Launch the application
   (1) User clicks on the icon of the application.
   (2) System launches MainPage.
   (3) MainPage launches NoteControl.
   (4) MainPage launches UserInfoControl.
   (5) MainPage launches LoginPage.

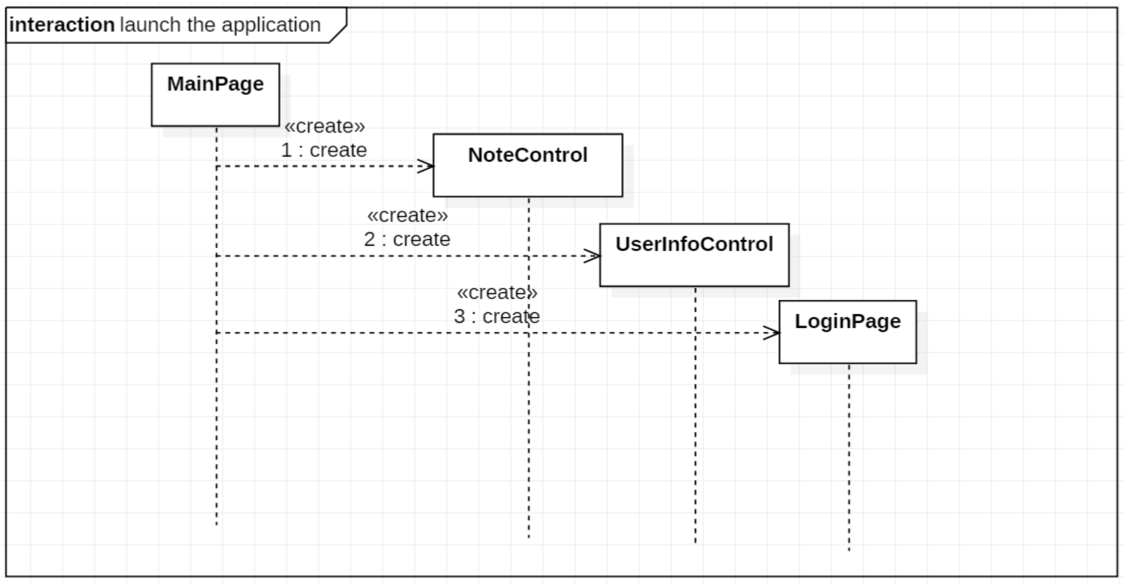

Figure 21: Launch the App

2) Close the application
   (1) User closes the application.
   (2) NoteControl is ended.
   (3) UserInfoControl is ended.
   (4) Mainpage is ended.

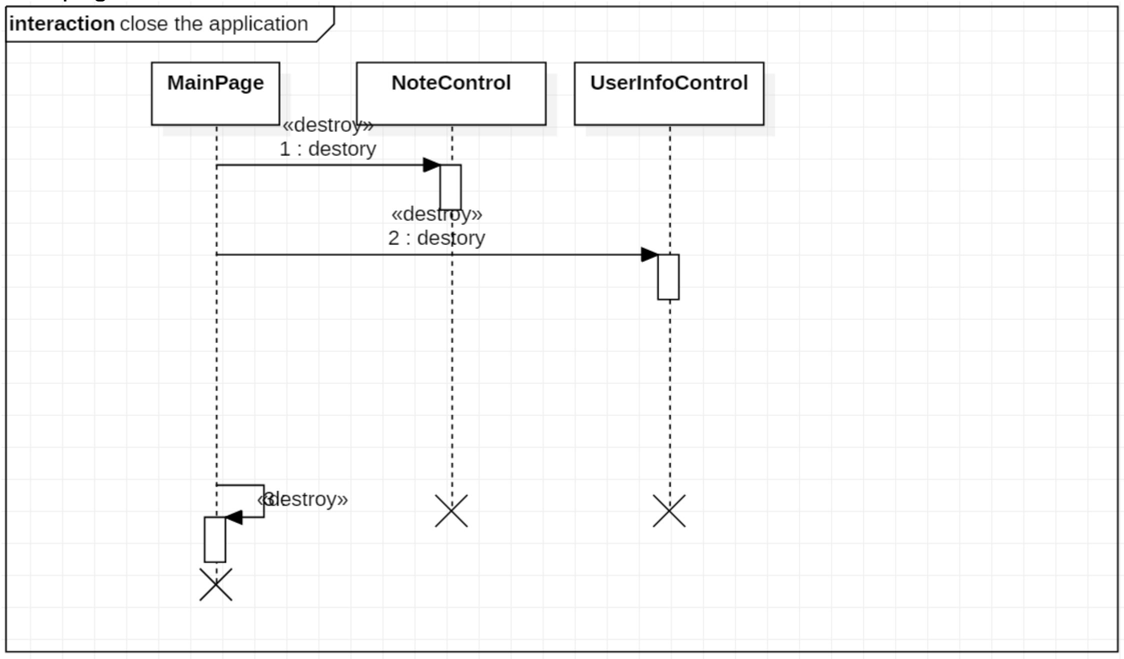

Figure 22: Close the App

3) Exception Condition
   When the system has an exception condition, the application may be relaunched.

4.9 Data Management Design

Since our system does not require a rather huge amount of data, we do not specify the data management here.

The designed entry class diagram is showed below:

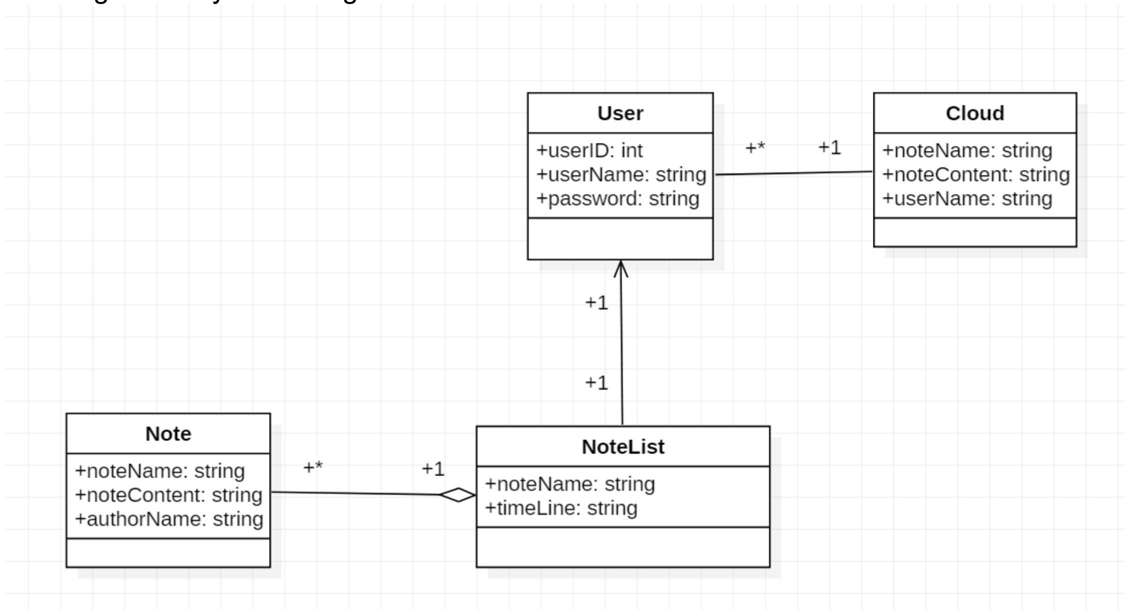

Figure 23: Design class diagram of entity

4.10 Other Design

(1) Visit control and privacy design

Users should submit their username and password before access the notes of the account. Only those who pass the checking procedure can access the whole data and continue their synchronization.

	User Information	Notes
User	Modify user information	Modify existing notes and synchronize

(2) Reliability design

To ensure the reliability of our system, we decide to set a backup function. The existing notes will not be covered until their third synchronization of modification.