Cover Page

NoteConnect

https://github.com/jwhit0/nyu-oop-course-project

Group# 22
Rhan Chen
Joanna Lu
James Whitten

Date of Submission

Table of Work

(Please write x in the boxes to mention what each student achieved in this project)

	Rhan Chen	Joanna Lu	James Whitten
Project Description	x	x	x
Uses Cases Diagram(s)	x		
Sequence Diagrams	x		
Class diagram(s)			x
Implementation	x	x	x
Conclusion	x	x	x

Project Description (One Page)

- General Description, Goals and Benefits

  NoteConnect is an app designed to store text in the form of notes (.txt files, for us). Users are able to store these notes privately, or share them with other users, allowing for collaboration, interaction, and effective sharing of information.

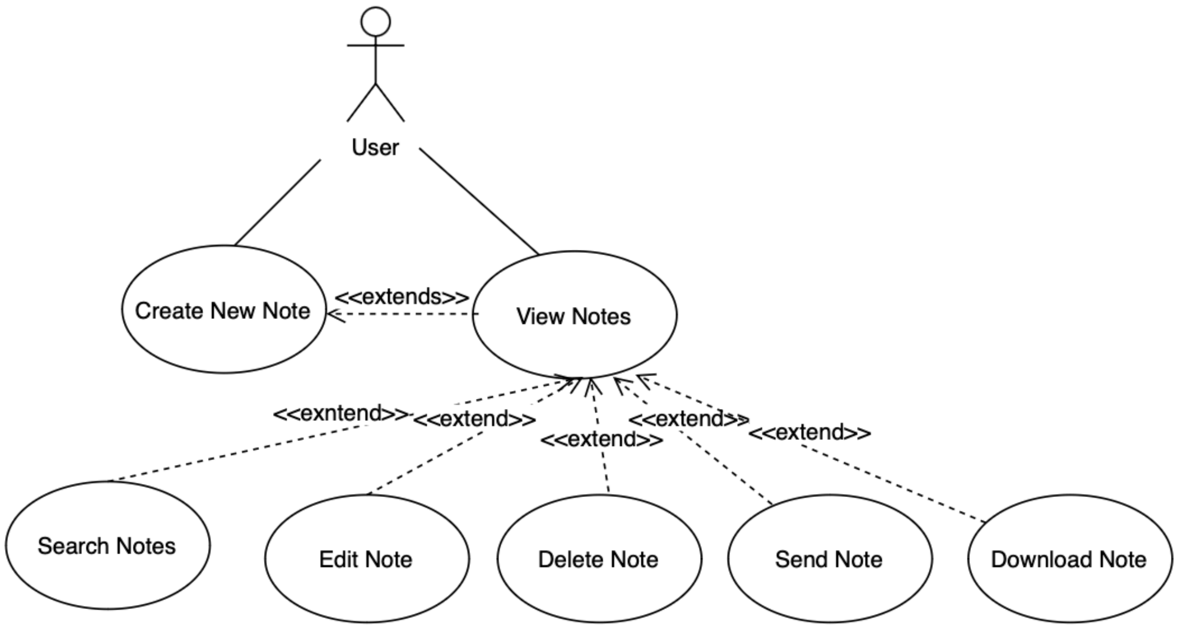
- System input(s) and output(s)
  - Some of this was implemented finally; some was not. Inputs and outputs that are (somewhat) working are denoted with a *
  - Input:
    - user interaction (text, buttons) *
    - user credentials (name, email, password)
    - creating a note (text) *
    - deleting a note (id of note) * (using note text rather than id)
    - media file data, media file size, format of media file
    - Sending notes (note id, credentials of other users)

  Output: textfiles to be stored locally *

- Special requirements (Performance, Interfaces, Constraints, Reliability, if any)

We used the JFrame library to help with GUI implementation and usage – the panels were built with the help of the Professor's code from in class. For file input and output, we used a FileManager class, also developed with class code; the note objects are stored in string form within a .txt file. We also have classes for authentication, media attachments, and User management that we did not end up using due to time constraints. If we had more time, we would also implement a Register/Login GUI that would call this authentication service and allow for network sharing of note objects. As a result of this, we don't have a specific "social media" aspect implemented; however, to reflect the different notes created by different users, our Note class does have a list attribute that stores all Users with access. We had trouble integrating the GUI classes with the Note managing classes, so the GUI classes as of right now just use basic string representation of Note objects in a .txt file. However, given more time, we could have a much more fleshed-out system that allows for simpler and more effective Note, User, and Media management.

- Use Case Diagram



- Use case descriptions

UC Reference Name/Number:	1.create note
Overview	User intends to create a new note file in the NoteConnect app
Related use cases:	Use case 2-save notes extends 1-create note, after creating a new note users can decide when and whether to continue working on the note or select another note to work on
Actors	user

UC Reference Name/Number:	3.search note
Overview	User searches notes by title among the saved notes in the NoteConnect app
Related use cases:	Search Notes extends use case 2-view notes. People can search for Notes with specific keywords and titles.
Actors	User

UC Reference Name/Number:	2.view note
Overview	short description of the processes
Related use cases:	Usecase 3, 4, 5, 6, 7 extends this use case, after selecting view notes users can choose to edit, download, delete, search or send the note to another user.
Actors	user

UC Reference Name/Number:	4.edit saved note
Overview	User intends to edit previously saved note
Related use cases:	Use case extends use case 2: after selecting view notes users can choose to edit the notes
Actors	user

UC Reference Name/Number:	5.delete saved note

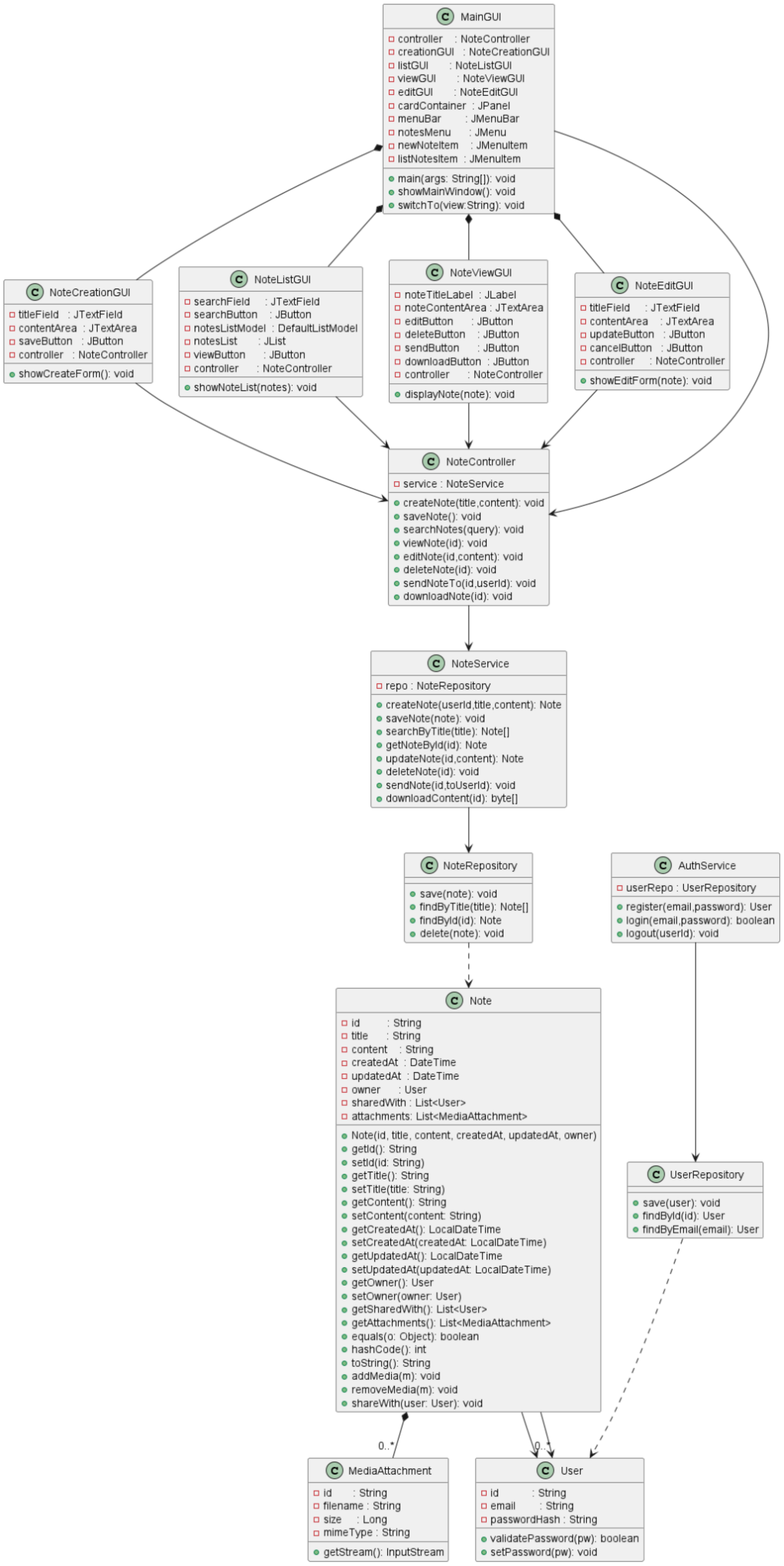
Overview	User intends to delete previously saved note
Related use cases:	Use case extends use case 4: after selecting view notes users can choose to delete the notes
Actors	User

UC Reference Name/Number: 6.send saved notes to other users	
Overview	User intends to send a previously saved note to another user
Related use cases:	Use case extends use case 2: after selecting view notes users can choose to send the notes to another user
Actors	User

UC Reference Name/Number: 7.download note to local machine	
Overview	User intends to download the file to local machine
Related use cases:	Use case extends use case 2: after selecting view notes users can choose to download the notes to local machine
Actors	User

System Design

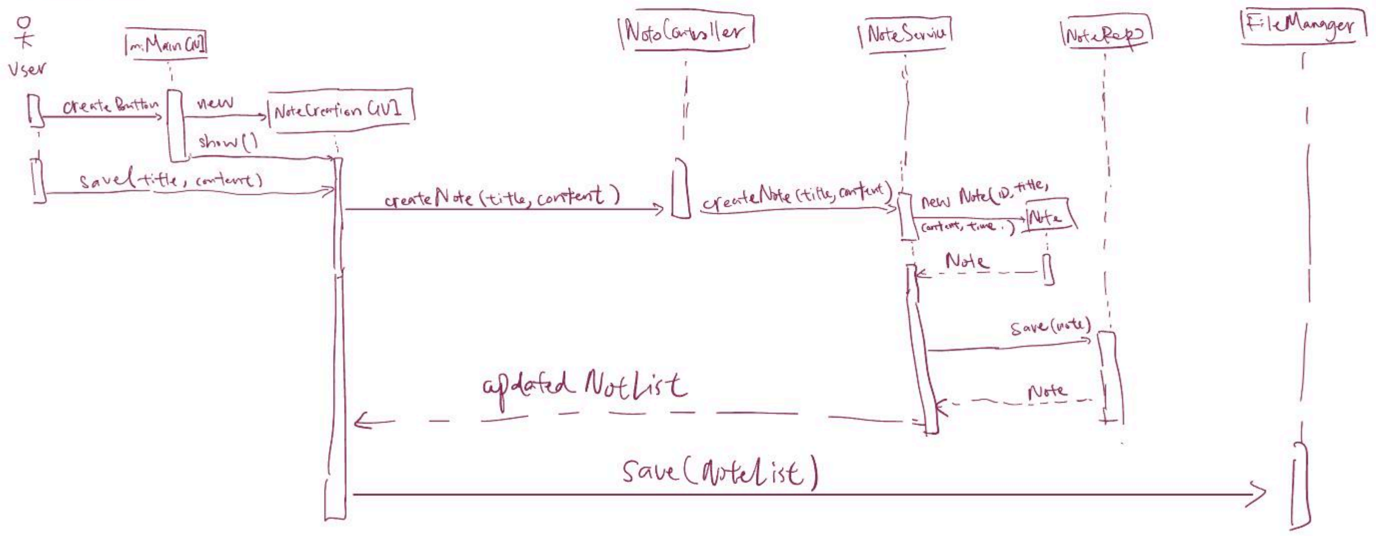
- Class Diagram


MainGUI
controller : NoteController
creationGUI : NoteCreationGUI
listGUI : NoteListGUI
viewGUI : NoteViewGUI
editGUI : NoteEditGUI
cardContainer : JPanel
menuBar : JMenuBar
notesMenu : JMenu
newNoteItem : JMenuItem
listNotesItem : JMenuItem
main(args: String[]): void
showMainWindow(): void
switchTo(view:String): void
NoteCreationGUI
titleField : JTextField
contentArea : JTextArea
saveButton : JButton
controller : NoteController
showCreateForm(): void
NoteListGUI
searchField : JTextField
searchButton : JButton
notesListModel : DefaultListModel
notesList : JList
viewButton : JButton
controller : NoteController
showNoteList(notes): void
NoteViewGUI
noteTitleLabel : JLabel
noteContentArea : JTextArea
editButton : JButton
deleteButton : JButton
sendButton : JButton
downloadButton : JButton
controller : NoteController
displayNote(note): void
NoteEditGUI
titleField : JTextField
contentArea : JTextArea
updateButton : JButton
cancelButton : JButton
controller : NoteController
showEditForm(note): void
NoteController
service : NoteService
createNote(title,content): void
saveNote(): void
searchNotes(query): void
viewNote(id): void
editNote(id,content): void
deleteNote(id): void
sendNoteTo(id,userId): void
downloadNote(id): void
NoteService
repo : NoteRepository
createNote(userId,title,content): Note
saveNote(note): void
searchByTitle(title): Note[]
getNoteById(id): Note
updateNote(id,content): Note
deleteNote(id): void
sendNote(id,toUserId): void
downloadContent(id): byte[]
NoteRepository
save(note): void
findByTitle(title): Note[]
findById(id): Note
delete(note): void
AuthService
userRepo : UserRepository
register(email,password): User
login(email,password): boolean
logout(userId): void
Note
id : String
title : String
content : String
createdAt : DateTime
updatedAt : DateTime
owner : User
sharedWith : List<User>
attachments: List<MediaAttachment>
Note(id, title, content, createdAt, updatedAt, owner)
getId(): String
setId(id: String)
getTitle(): String
setTitle(title: String)
getContent(): String
setContent(content: String)
getCreatedAt(): LocalDateTime
setCreatedAt(createdAt: LocalDateTime)
getUpdatedAt(): LocalDateTime
setUpdatedAt(updatedAt: LocalDateTime)
getOwner(): User
setOwner(owner: User)
getSharedWith(): List<User>
getAttachments(): List<MediaAttachment>
equals(o: Object): boolean
hashCode(): int
toString(): String
addMedia(m): void
removeMedia(m): void
shareWith(user: User): void
UserRepository
save(user): void
findById(id): User
findByEmail(email): User
0..*
0..1
MediaAttachment
id : String
filename : String
size : Long
mimeType : String
getStream(): InputStream
User
id : String
email : String
passwordHash : String
validatePassword(pw): boolean
setPassword(pw): void

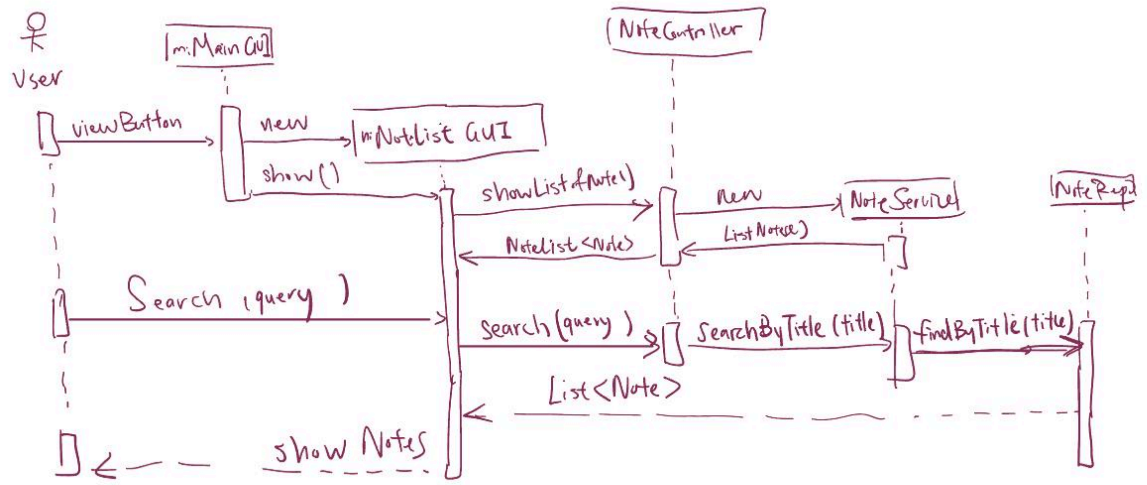
-

- Sequence Diagrams

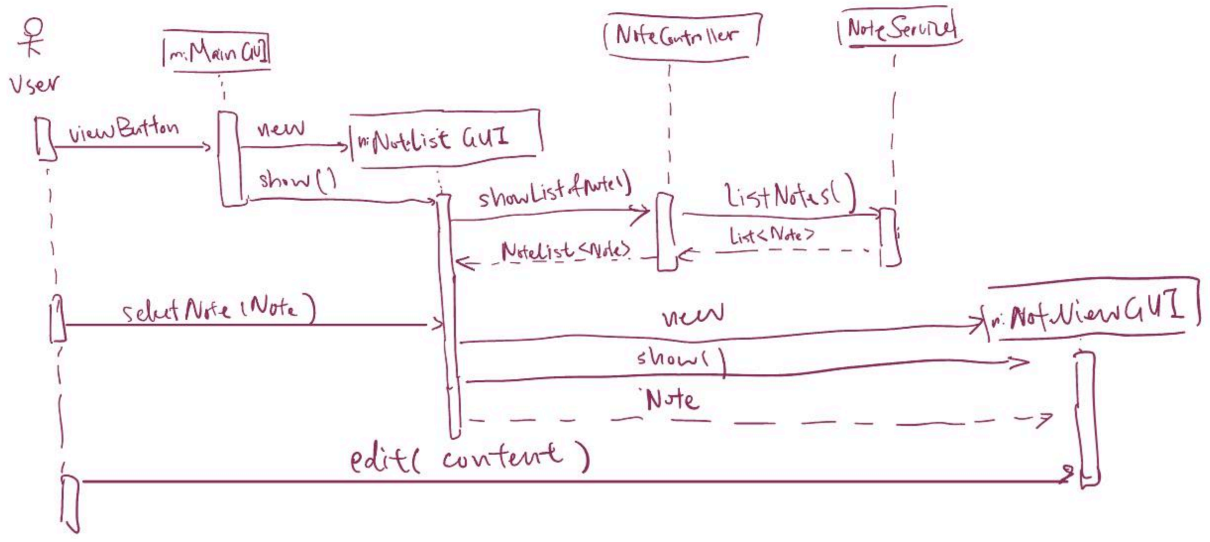
Create New Note



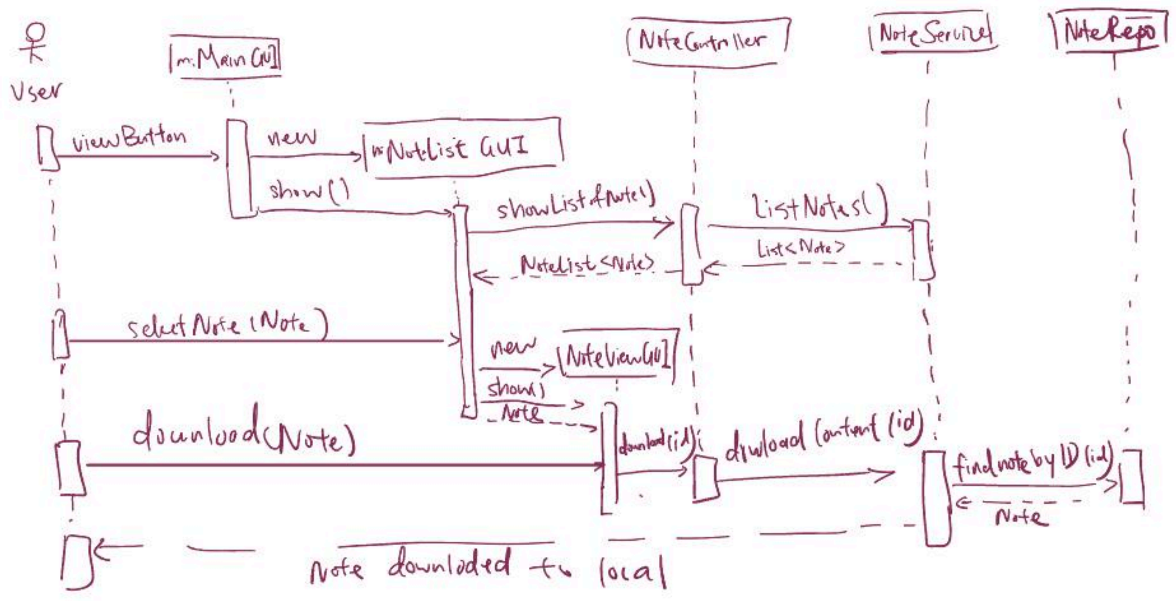
Search Note



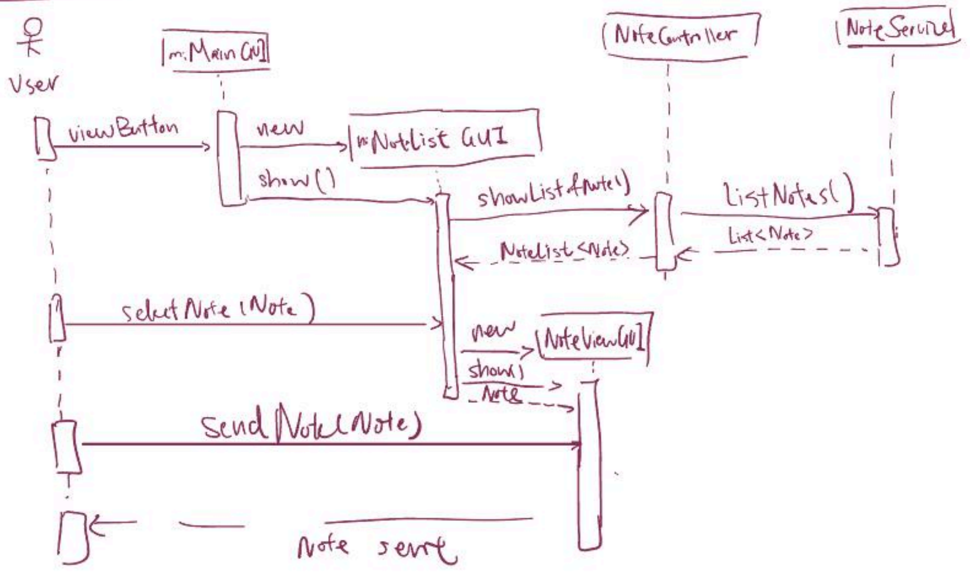
Edit Note



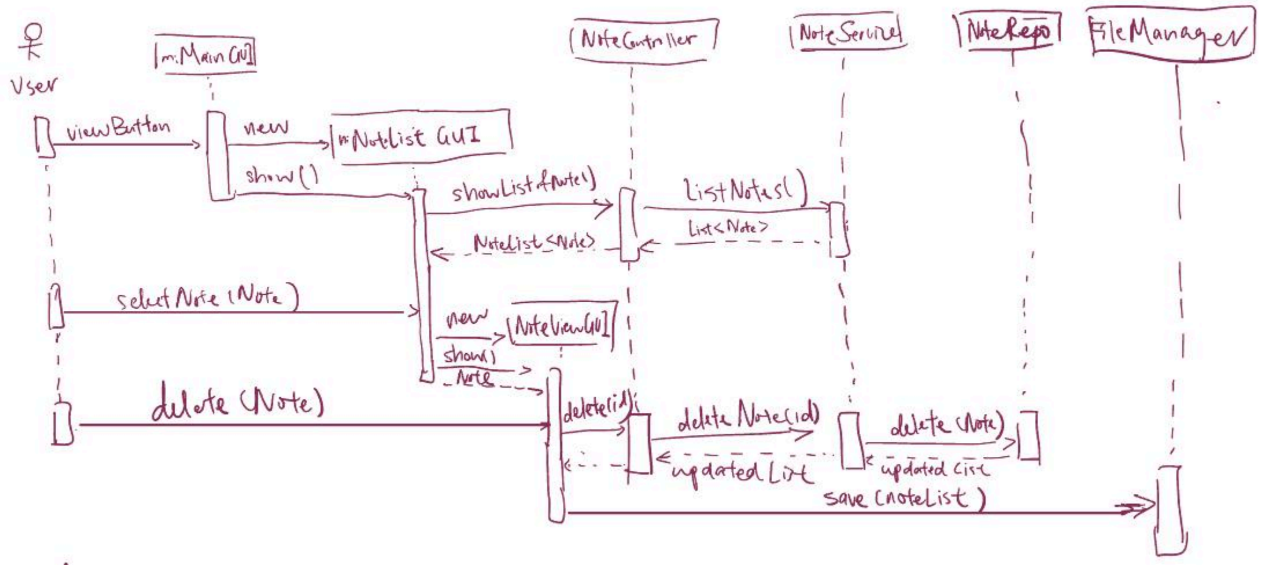
dowloadNote



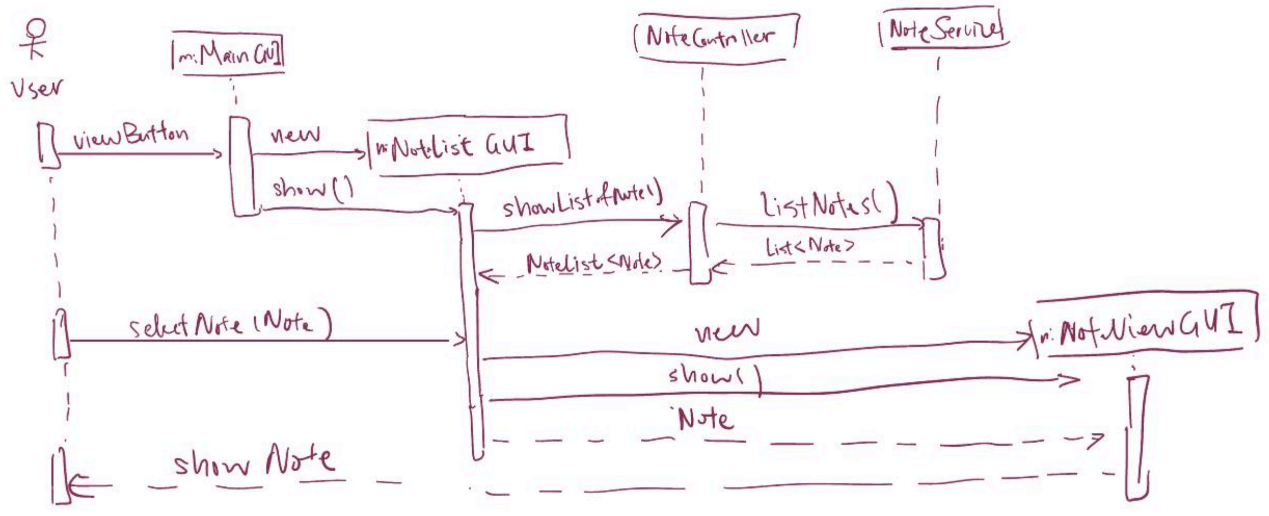
SendNote



deleteNote



ViewNotes



Conclusion

The development of NoteConnect deepened our understanding of object oriented programming and practical application design. Through the implementation of features such as note creation, editing, and deletion, we worked toward goals of usability, collaboration, and effective information management. Although because of the time constraint we couldn’t fully integrate advanced functionalities like authentication and a network-based sharing system, we were able to successfully achieve our primary goal of local note management.

Future improvements would have included the addition of an authentication service and following that, an improved GUI for user registration and login, a complete implementation of the app’s social aspects, and better integration of media attachments within notes. Either way, despite its shortcomings, the current state of NoteConnect serves as a mostly functional prototype that

Some aspects that we could have improved on were:

- communicating earlier into the process so that integrating each of our separate code components would be more fluid and efficient
- Starting off with stronger UML diagrams that would better serve our purposes
- Beginning on the same page in terms of what goals would be more realistically achievable (local file saving, GUI implementation) and what was outside of the scope of this two week long project (authentication, server, network).

Table of Contents

Rhan - Create new, view, send