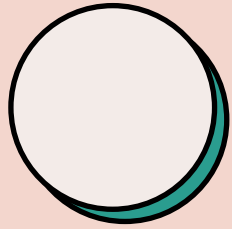


Note Reader

Emmett Fitzharris

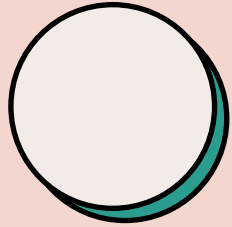


Presentation Structure



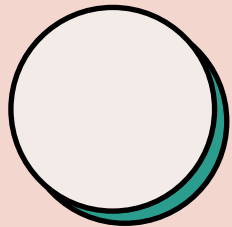
Introduction

A brief overview of the concept



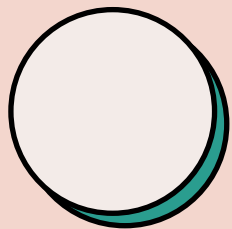
Research Phase

A short recap of the research phase of the project



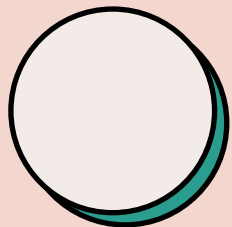
Implementation

Review the actual implementation, and the changes from the original plan



Difficulties and Challenges

What went wrong, and what was learned



The Future of NoteReader

What's next for the project

Introduction

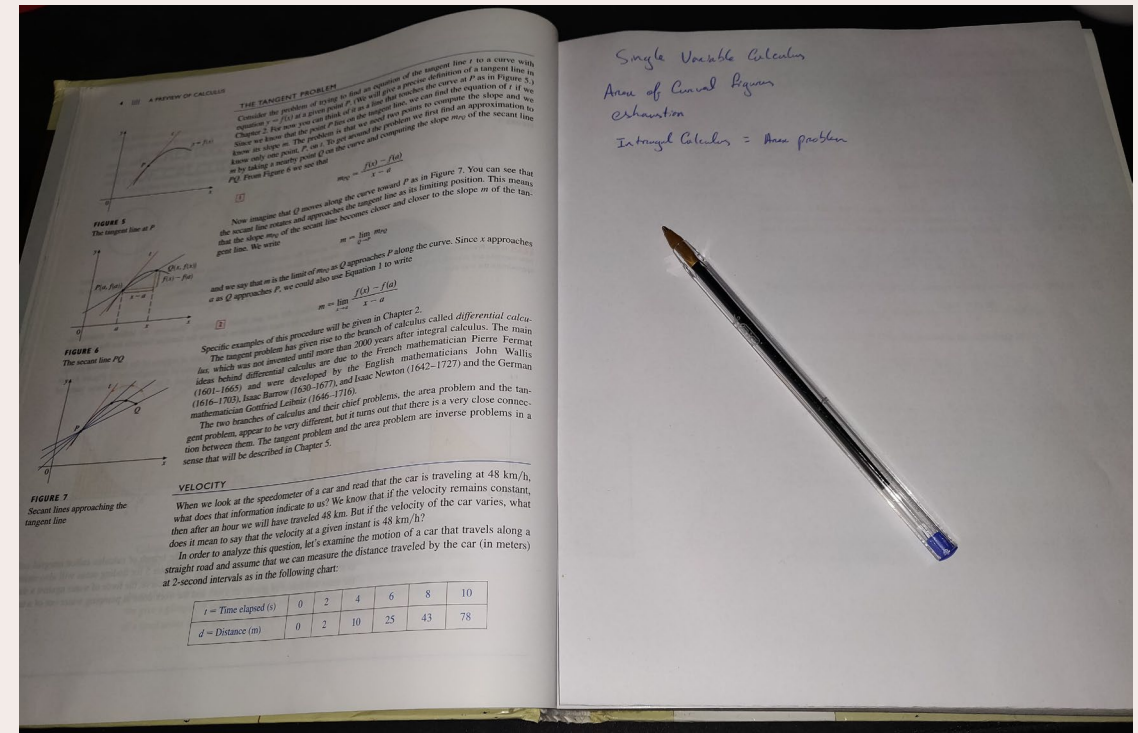
Picture a book with a blank sheet of paper slotted between each page.

As you read, you take notes.

Flip to the next page, a new sheet awaits you.

When it's time to revise, you go back to that page and your notes on the topic are right there.

This is the feeling Note Reader wants to achieve.



Project Summary

- An all-in-one interface for document reading and note-taking.
- Each page of notes is directly tied to a page of the source document.
- The core interface of NoteReader is the Split-View.

Left Pane:	Right pane:
“true-to-intent” rendering” of a wide range of common file types, including PDFs, Word Documents, PowerPoint slides, and eBooks.	Text editor with Markdown and HTML rendering, to allow for richly formatted notes.

Background Research

E. M. Stacy and J. Cain, "Review note-taking and handouts in the digital age. 2015

- Advocates for a "skeletal" handout structure to provide cognitive scaffolding, while encouraging engaging with material without full transcription.
- The above would be well supported by a split-pane application like Note Reader.

D. Sun and Y. Li, "Effectiveness of digital note-taking on students' performance in declarative, procedural and conditional knowledge learning," International Journal of Emerging Technologies in Learning, vol. 14, pp. 108–119, 2019

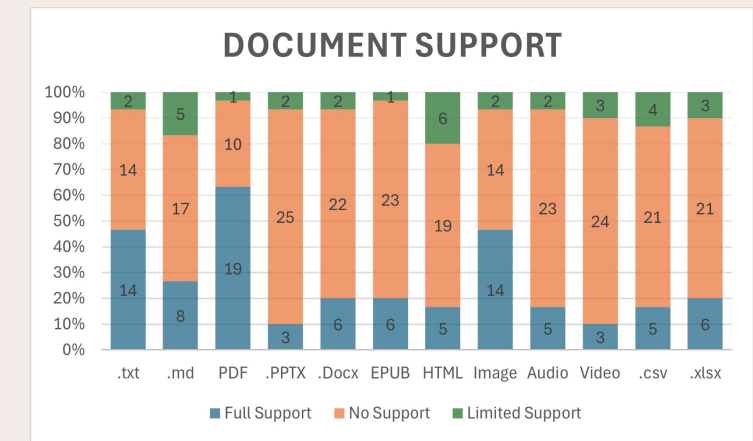
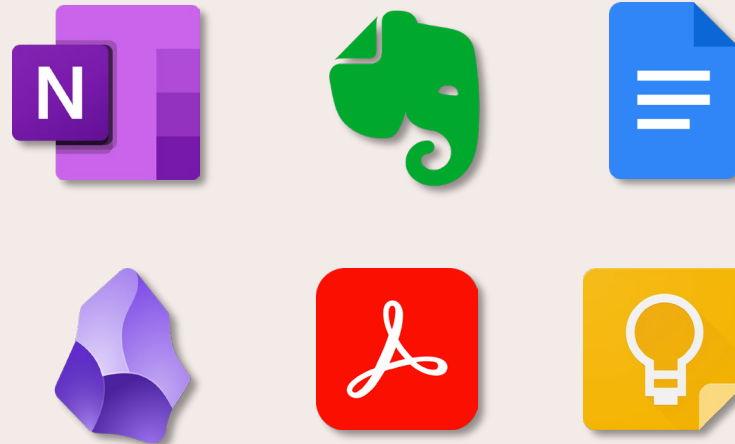
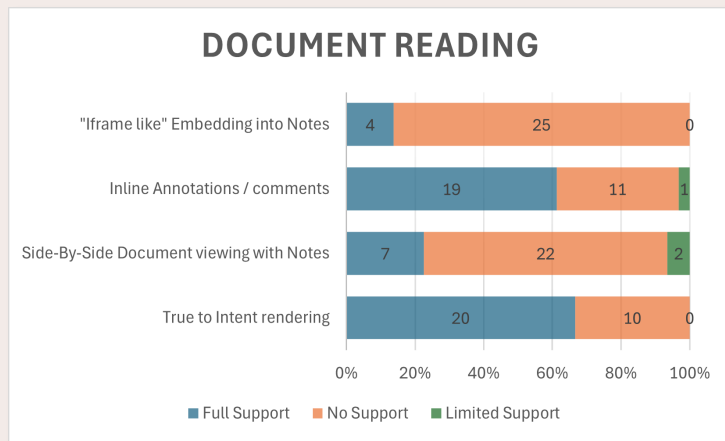
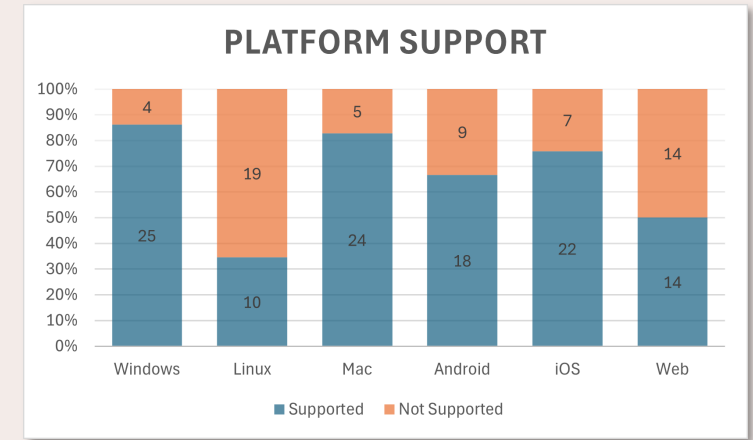
- Shows evidence of small performance benefits to digital note taking, especially in declarative knowledge.
- Structure of Note Reader application may further support conditional knowledge, by ensuring context remains intact.

B. Artz, M. Johnson, D. Robson, and S. Taengnoi, "Taking notes in the digital age: Evidence from classroom random control trials," Journal of Economic Education, vol. 51, pp. 103–115, 4 2020.

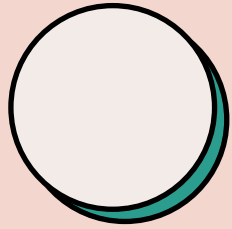
- No statistically significant difference found between digital and handwritten notes on retention.
- Topics dependant on diagrams/mathematical elements may give advantage to traditional note taking.

Existing Solutions

32 programs were reviewed, to determine current levels of support offered by existing solutions. The programs were categorized and checked against a list of features.

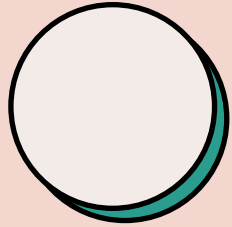
[illegible]

Current Issues



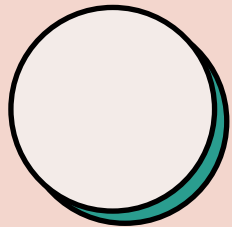
Fragmentation of Tools

Document readers have weak support for note taking, and vice versa.



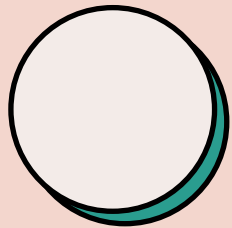
Limited Multi-Platform Support

Linux support is rare. Cross functionality is a challenge.



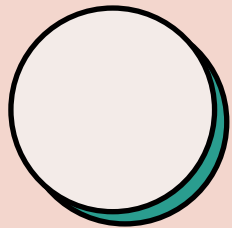
High Cost of Subscription Services

Especially for cloud storage and accessing data on multiple devices.



Inadequate File Type Support

Power Point support is especially rare



Manual Organization and Storage

Loss of Context, misplaced documents, wasted time.

Problem Definition

Despite the availability of numerous note-taking applications and document readers, there exists a significant gap in the seamless integration of these two functionalities.

The primary objective is to develop a cohesive system that allows users to read and annotate documents, create linked notes, and access these materials in an organized and intuitive manner, without destructively modifying the source material.

Unique Value Propositions

Side-by-Side
document
reading and note
taking

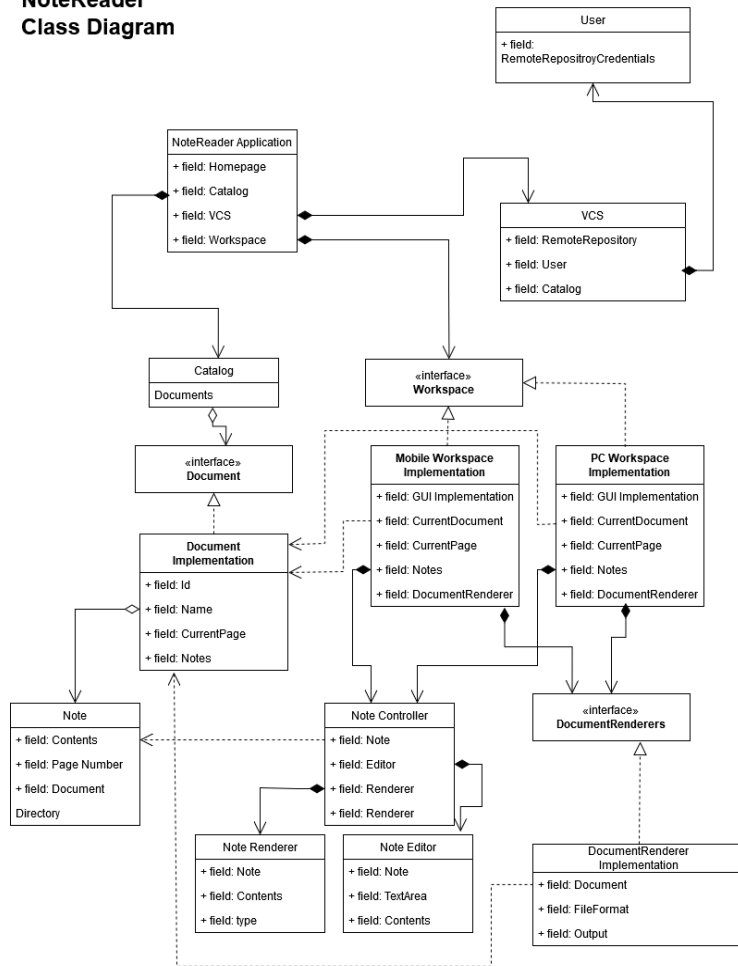
Support a wide
Range of
commonly used
document types

Multi-Platform
Design approach

Git-based file
Synchronization

Implementation Approach

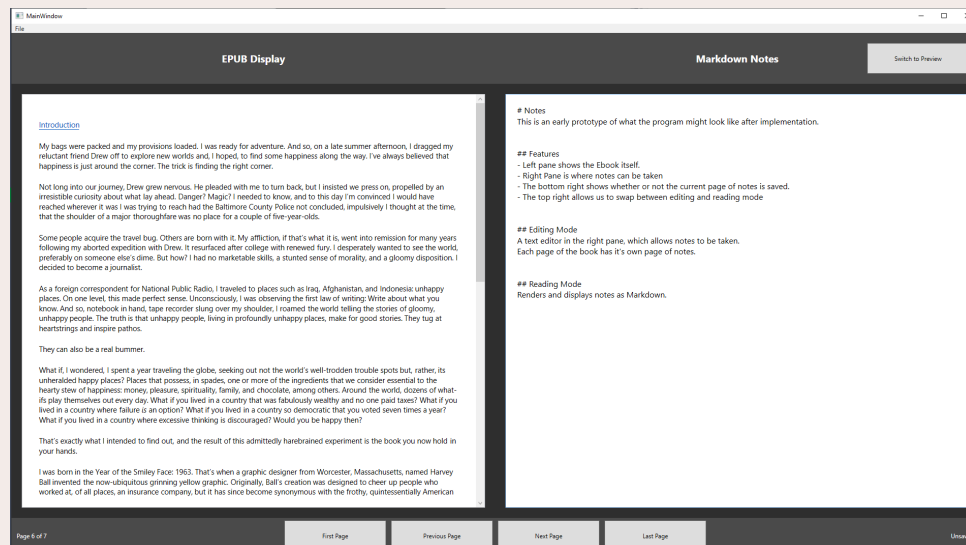
**NoteReader
Class Diagram**



- User centric design approach
- Agile based project management
- Core application built in Kotlin Multiplatform
- Abstract GUI interfaces to ensure consistency across platforms.

Early Prototypes

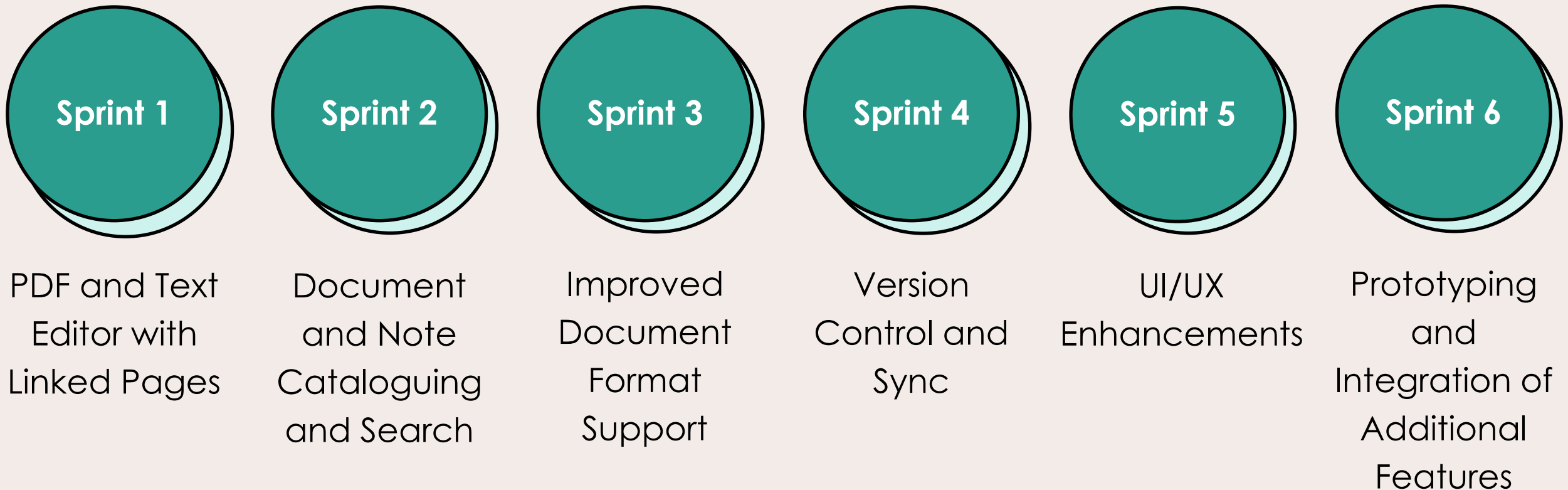
Desktop Application



Mobile Application



Implementation Plan Schedule



Actual Implementation Schedule



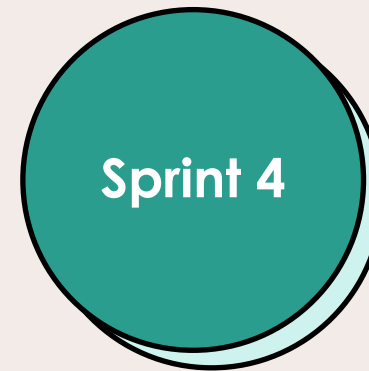
Document
Rendering (PDF)
and Linked
Note Editor



Focus on File
System and
Program State



Expanded file
format support
(EPUB, DOCX,
DOC, PPTX,
HTML)



UI/UX
Refinement
and Theming



Prototype
Integration and
Documentation

Functional Requirements

The system shall provide a split-view interface to allow users to view documents and take notes simultaneously

Status: Fulfilled

The system shall support the import and display of PDF, EPUB, DOCX, and PPTX files

Status: Fulfilled

The system shall allow users to link notes to specific pages or sections of the source document

Status: Fulfilled

The system shall support searching and filtering of notes and documents using tags, keywords, and categories

Status: Deferred

The system shall support exporting notes in plain text format for use in other applications

Status: Fulfilled

The system shall enable users to sync notes to cloud storage providers like GitHub using version control

Status: Deferred

The system shall allow users to tag and categorize notes for efficient organization and retrieval

Status: Deferred

Non-Functional Requirements

Performance:

The system shall load and display documents without noticeable wait times on standard modern devices

Status: Partially Fulfilled

Usability:

The user interface shall be intuitive, with clear user experience flows

Status: Fulfilled

Scalability:

The system shall support large document files and offer alternative synchronization methods for these files.

Status: Partially Fulfilled

Portability:

The system shall support cross-platform functionality on Windows, macOS, Linux, and mobile devices

Status: Deferred

Reliability:

The system shall gracefully handle unexpected crashes to prevent data loss by saving notes in real-time

Status: Fulfilled

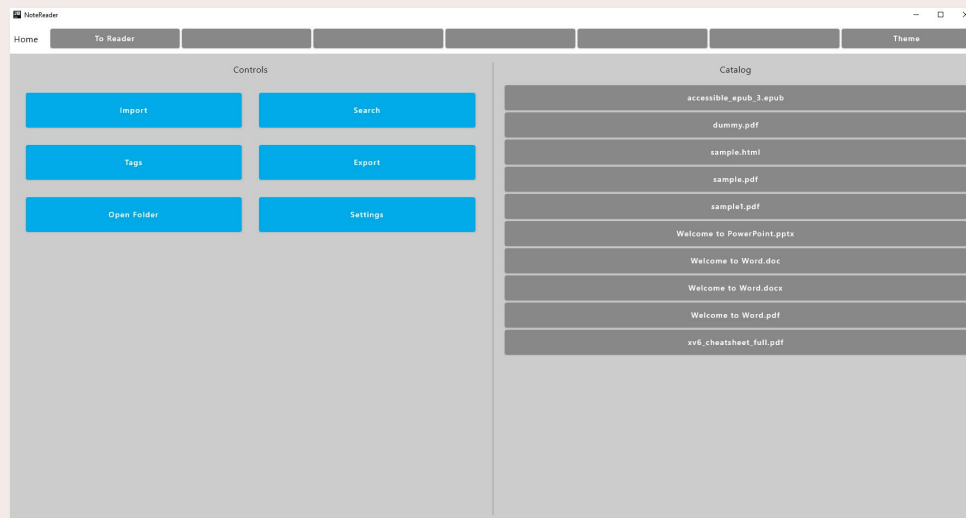
Maintainability:

The system shall be modular to allow for easy updates and addition of new features

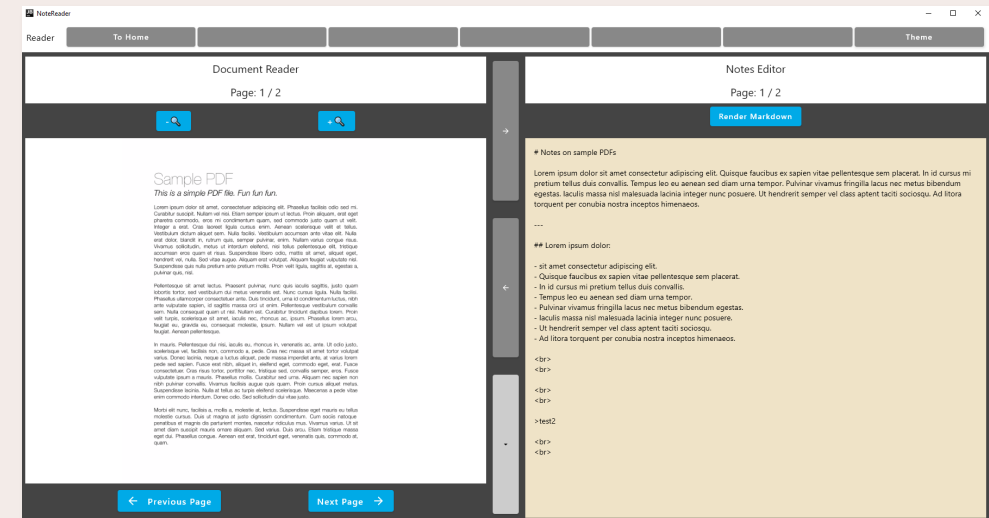
Status: Fulfilled

Implemented Prototype

Home Page

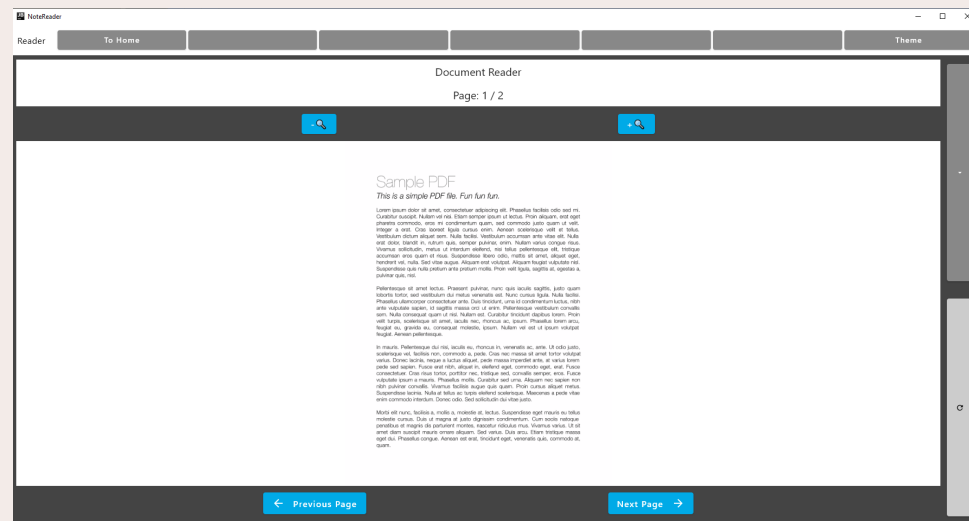


Reader View

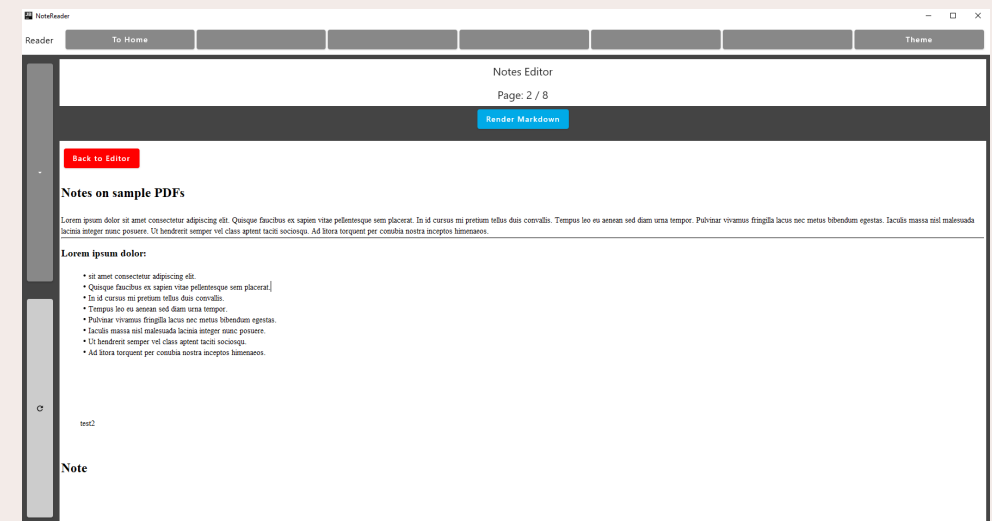


Implemented Prototype

Reader Focus Mode



Notes Markdown Focus Mode



Challenges Encountered

Framework Immaturity:

- Kotlin Multiplatform first stable release November 2023
- Limited documentation and community support
- Third-party plugins and libraries often experimental or poorly supported
- Java cross-compatible, giving access to Java Libraries, but not trivial to implement



Document Rendering:

- Native per-format support involved significant development cost
- Fallback to PDF conversion approach adopted

Time Constraints:

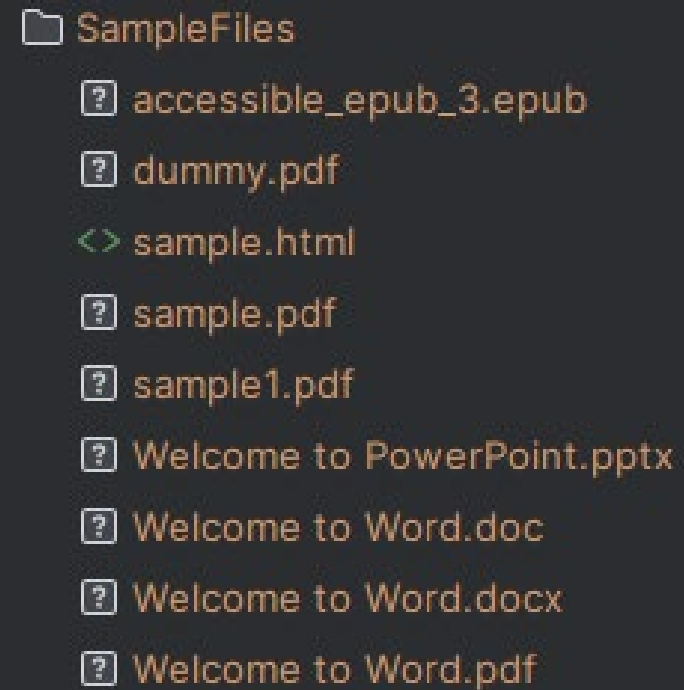
- Scope reigned in, deferring some requirements to ensure delivery of MVP with core features

Testing

A range of sample files selected to represent typical academic study material

Manual testing validated:

- Document Import and Rendering
- Note Creation and Retrieval
- Session Persistence
- File System Inspection



```
SampleFiles
accessible_epub_3.epub
dummy.pdf
sample.html
sample.pdf
sample1.pdf
Welcome to PowerPoint.pptx
Welcome to Word.doc
Welcome to Word.docx
Welcome to Word.pdf
```

Discussion and Evaluation

- The completed NoteReader prototype delivers a focused and practical solution to the challenge of fragmented note-taking and document reading workflows.
- Supports all targeted document formats, with split view interface for notes.
- Document-page-to-note linking, real time saving and session persistence
- Sprint plan was adapted to challenges encountered in implementation
- Some requirements were deferred to later development cycles after re-evaluation

Conclusion

- There is a clear gap in the features offered by currently available applications.
- Existing tools are fragmented and none treat both note taking & document reading as first class priorities.
- NoteReader has potential to address these shortcomings, while supporting cognitive benefits.
- Cutting-edge frameworks provide useful features, but may introduce new limitations ecosystem support.
- The project achieved a core Minimum Viable Product, which addresses the problem definition.
- The project acts as a solid proof of concept and illustrates the value of the premise.
- There are several areas identified for further development and enhancement.
- The architecture is built to accommodate easy addition of new features and platforms.

The Future of NoteReader

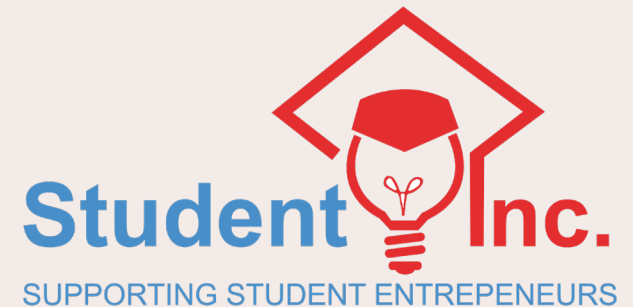
- Catalogue System
- Git Integration
- Cross-Platform Expansion
- Optical Character Recognition
- Handwriting Support
- Improved UI

Direct Integration with
Open-Source Learning
Management Systems



Startup Accelerator

Rubicon



Thank You!

Questions and comments welcome