6-12-2019

N.O.V.I.: Note Organizer for the Visually Impaired

Axel Perez

Cesar Tesen

SANTA CLARA UNIVERSITY

DEPARTMENT OF COMPUTER ENGINEERING

Date: June 12, 2019

I HEREBY RECOMMEND THAT THE THESIS PREPARED UNDER MY SUPERVISION BY

Axel Perez
Cesar Tesen

ENTITLED

N.O.V.I.: Note Organizer for the Visually Impaired

BE ACCEPTED IN PARTIAL FULFILLMENT OF THE REQUIREMENTS FOR THE DEGREE OF

BACHELOR OF SCIENCE IN COMPUTER SCIENCE & ENGINEERING

Thesis Advisor

Department Chair

N.O.V.I.: Note Organizer for the Visually Impaired

by

Axel Perez
Cesar Tesen

Submitted in partial fulfillment of the requirements
for the degree of
Bachelor of Science in Computer Science & Engineering
School of Engineering
Santa Clara University

Santa Clara, California
June 12, 2019

N.O.V.I.: Note Organizer for the Visually Impaired

Axel Perez
Cesar Tesen

Department of Computer Engineering
Santa Clara University
June 12, 2019

ABSTRACT

Visually impaired students face extra challenges when it comes to the basic necessity of note-taking. Current assistive technology is fragmented in function. These students often need to combine solutions such as voice recording lectures, hiring someone to transcribe notes to braille, hiring a reader, etc. The amount of time and money they need for these solutions proves to be a great disadvantage, and we wish to provide an easier solution that will give these students a more independent and productive learning experience.

Our solution is an application that can offer intuitive, convenient, and comprehensive access to notes for the visually impaired. We would implement our solution through an iOS mobile application that would allow users to upload hand written or text notes so that they could be organized and accessed through voice prompts. Users with visual disabilities would then have a platform to store and organize their notes in order to increase their ability to learn independently.

Table of Contents

List of Figures

Chapter 1

Introduction

1.1 Motivation

Assistive technology for the visually impaired has helped them more intuitively interact in the classroom, but there is still a lot to be done. Students with vision disabilities cannot interpret written text in class, nor can they fully utilize notes taken by other students. Even if they obtain notes in an audio format, it is very difficult to organize those notes or find specific sections. Testimonials show that these students spend significantly more time preparing and creating usable notes to keep up with their visually abled peers. It is unfair that it takes so much more time to attain the same level of familiarity with class notes. Blind students should be able to experience education as easily as students without disabilities, and fundamental practices like note taking should not be a hindrance from achieving academic goals.

Blind students rely on assistive technology, in addition to hired readers and transcribers, to keep up with their peers. These students unfortunately have to spend substantial amounts of time to find people willing to share their notes to a note reader who then transcribes the notes into either braille or voice recordings. This preparation process adds weeks to their note taking process, creating a great disadvantage. The most popular current assistive technology for reading text are braille readers/writers, text to speech apps, and simply recording lectures. While these accomplish the goal of giving a voice to visual text, all these systems lack the equivalent of visually scanning for what you are looking for in a body of text, or breaking notes down into sections or topics. If a student wishes to review a certain part in their notes, current solutions only leave the option of listening to an entire transcript over again. A hired reader often helps remedy this issue but in addition to the expense it imposes, it also limits the students ability to be independent in their learning experience.

1.2 Solution

Our solution is to create an audio based application that not only converts in-class textual data to audio, but also organizes and summarizes the notes to better serve blind students. First, our application would gather textual data from our user's captured image. The collected text would then be organized in the application by name, date, and subject. This structure allows us to create a solely voice interface to access notes from several classes and easily find the topics the user is searching for without having to listen to entire recordings. This would be accomplished by developing an audio search tool that would leverage the textual data to better find what the user is looking for. Unlike current solutions, our system would give back the freedom and independence to effectively use notes in an easier, faster manner, without the need of a reader every time one wishes to study.

Chapter 2

Societal Impact

This section details the social impact our application can provide.

2.1 Ethical

Our application gives visually impaired students the ability to more easily achieve academic autonomy. We did this with the motivation that the core elements of the application are already available, but have yet to be put together. Senior design was an opportunity to dedicate time to making that idea a reality, showing that this project is rooted in upholding the first tenet of the ACM Code of Ethics[1]:

> ***Contribute to society and to human well-being, acknowledging that all people are stakeholders in computing.*** This principle, which concerns the quality of life of all people, affirms an obligation of computing professionals, both individually and collectively, to use their skills for the benefit of society, its members, and the environment surrounding them. This obligation includes promoting fundamental human rights and protecting each individual's right to autonomy. An essential aim of computing professionals is to minimize negative consequences of computing, including threats to health, safety, personal security, and privacy. When the interests of multiple groups conflict, the needs of those less advantaged should be given increased attention and priority.

2.2 Social

Globally, there are about 1.3 million people that live with some form of visual impairment[5]. We attempt to bridge the education gap with as many of these people as we can, and we recognize that smartphones are the vehicle we can use to deliver this social good of independent learning.

2.3 Economic

NOVI is built using free and open-source development tools, a design decision made for the consideration of both developers and users. In addition, our application also may enable students and professionals to excel in their respective fields; potentially leading them to greater economic success.

2.4 Usability

Being an accessibility app, high usability was a core concern in the development process. Ensuring good usability helps us achieve our goal of giving more potential students the power to effectively manage their notes. According to the Nielsen Norman Group, a world-class UX research group, usability can be broken down into the following 5 basic quality components:

2.4.1 Learnability

Learnability refers to how easy it is to perform tasks upon the user's first encounter of the design. The voice assist in each page of the application has a guide for what specific commands the user can execute in the current view. In addition to voice assist for the blind, the layout and color palette for the visual user interface is meant to cater to those with mild to severe vision impairment. We do this by providing large buttons for visibility and high color contrast to differentiate between actions easily.

2.4.2 Efficiency

The efficiency of the design is determined by how quickly the user can perform tasks. According to the activity diagram, the user can upload notes or navigate to the notes browser in one press or voice action. Accessing and interacting with a note takes one more step, meaning all core functions are only within a couple of steps from the home menu.

2.4.3 Memorability

Memorability refers to how easily can a user reestablish proficiency after a long period of not using the application. We created the voice assist command guide to help minimize the time required to get acquainted with app navigation. Further UX testing will provide time-based metrics to show whether or not this paired with efficient navigation is enough to ensure memorability.

2.4.4 Errors

Preliminary user testing in earlier versions of the application revealed errors in how we presented prompts, the number of prompts, error handling in the voice assistant, and the flow between actions. We've since then redesigned the user interface to reflect what our first testers expect the behavior of the application to be, resulting in less confusion and better error recovery.

2.4.5 Satisfaction

Satisfaction in how pleasant it is to use the application has improved throughout the multiple initial versions. We would like to use survey data in the future alongside a beta release to track satisfaction with specific modules and qualities of the application.

2.5 Lifelong Learning

We went into this project knowing only the concept and motivation, but none of the iOS development skills. Our understanding of software development in other environments and languages helped lessen the learning curve, and we always found ourselves learning new material we didn't originally plan on tackling. As lifelong learners, we welcomed these challenges and recognize that our project is possible because the field of computer engineering moves so rapidly.

Chapter 3

Requirements

This section details the functional and non-functional requirements as well as design constraints for this project. Requirements were determined by considering the needs of our intended users, how current solutions address their needs, identifying the shortcomings of those solutions, and designing our product to fully serve our intended user's needs.

3.1 Functional Requirements

3.1.1 Critical

- Users will be able to fully utilize the application without having to depend on sight.
- Users will be able to import scanned notes with assistance.
- Users will be able to navigate through previously imported notes.
- The application will be able to dictate notes back to the user.

3.1.2 Recommended

- Users will be able to search notes via voice prompts.
- The application will offer a summary of each scanned note.
- Users will be able to playback desired sections of notes.
- The application will be able to dictate the sections of a specific note.
- Users will be able to go to a specific section of a note via a voice command.
- The application will be able to take lecture videos and PowerPoint slides as inputs to populate notes.
- The application will be able to use haptic feedback as a way to assist the user in navigating through the application.

3.2 Non-Functional Requirements

3.2.1 Critical

- The user interface should be easy to use, especially considering this is an accessibility device.
- The application will be easy to navigate for someone with a vision disability.
- The application will reasonably be able to scan textual images of varying lighting.
- The application will handle data from multiple users.
- The application will reliably keep the users data locally.
- The application will always be available for use.

3.2.2 Recommended

- The application will be able to handle many users simultaneously without severe lag.
- The application will be secure from potential hackers.

3.3 Design Constraints

- The application must run on mobile IOS devices.
- The application must meet accessibility standards.

Chapter 4

Use Cases

The use case diagram in Figure 4.1 outlines the common actions taken by a user when using the NOVI application. We explore the intention and effect of each action.

4.1 Use Case Diagram

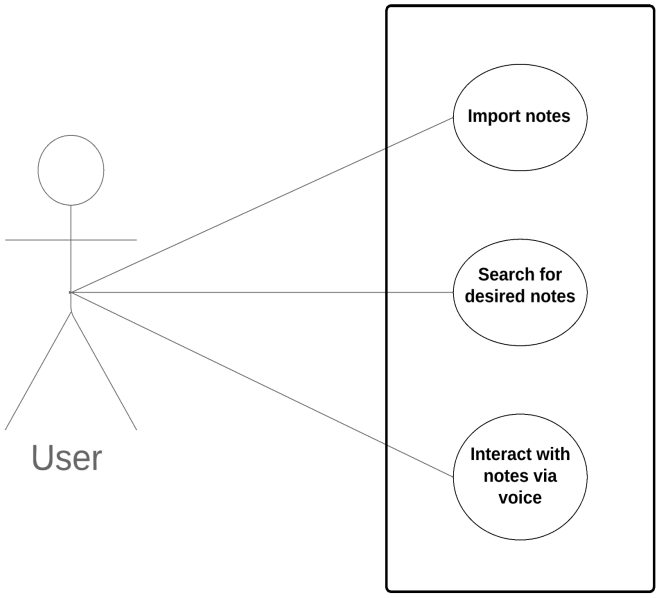


Figure 4.1: Use Case Diagram

4.2 Use Cases

1. Import notes

    - Actors: User
    - Goal: Scan notes and process them to text for future reference.
    - Pre-Conditions: User has notes they wish to import and a suitable environment for a readable picture.
    - Post-Conditions: Notes are ready to be classified with a title and subject.

2. Search for desired notes

    - Actors: User
    - Goal: Find the set of notes the user is looking for.
    - Pre-Conditions: User has a subject or title of notes in mind that they wish to find.
    - Post-Conditions: User can select and interact with the notes they were looking for.

3. Interact with notes via voice

    - Actors: User
    - Goal: Playback notes with pause and stop.
    - Pre-Conditions: User has selected the note they wish to interact with
    - Post-Conditions: User can control the playback of their notes.

Chapter 5

Activity Diagram

Figure 4.1 shows the various options and resulting actions a user can use during interaction with this application. The diagram follows standard UML format.

5.1 Activity Diagram

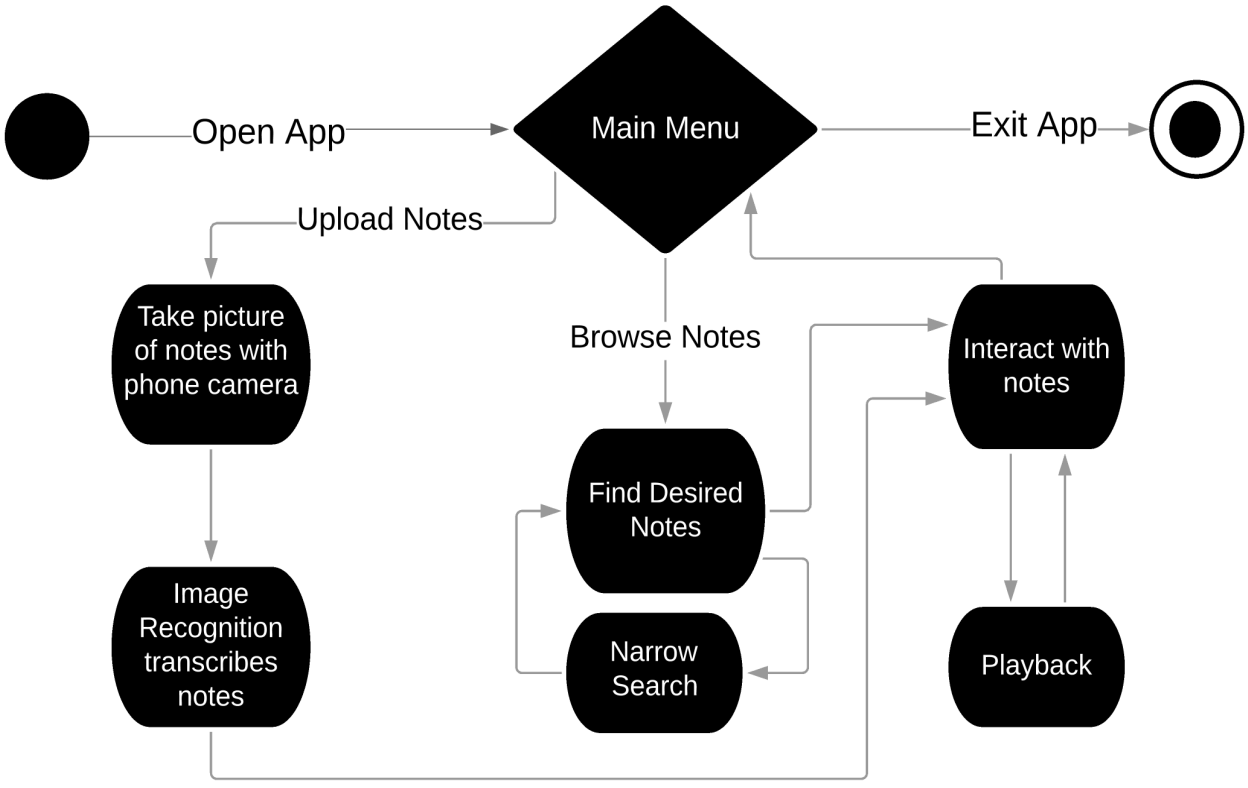


Figure 5.1: Activity Diagram for N.O.V.I

5.2 Main Menu

The main menu is the starting point of the NOVI application, as seen in Figure 5.1. It directs you to the ”Upload Notes”, ”Browse Notes”, and the exit of the application. A visually impaired user can either use the VoiceOver accessibility feature, provided by the iOS device, to interact with the buttons on the main menu or use the buttons large text and distinct colors to determine what button to press. After either ”Upload Notes” or ”Browse Notes” is completed, the application will return to the main menu to await the next user input.

5.3 Upload Notes

The ”Upload Notes” option, seen in Figure 5.1, sends the user to capture an image of their notes, which is then processed to recognize text. The user is then prompted to provide the name and subject of the note that will be saved with the recognized text. The note is now saved and the user can directly interact with it or go back to the main menu.

5.3.1 Image Capture

The user is directed to an image capture view that can be navigated with VoiceOver or just normally depending on the level of the user’s visibility. The user can then either take the photo themselves or ask a friend to assist them. Once the image is taken, the user can decide to retake the picture or proceed to the text recognition phase.

5.3.2 Text Recognition

Once the image capture has finished, the image is processed in the OCR model and the screen displays a loading symbol until the text recognition is completed. The screen then shows the image with rectangles around the strings of text that were detected. This image is not provided specifically for a visually impaired user, but rather for a friend or acquaintance that is assisting our intended user with uploading notes. The user can then decide to either save the note or retake the picture to start the process over again. Once the note text recognition has been accepted, the user is prompted to provide both the name and subject of the note. Finally, the note is saved and the user is directed to the view where they can interact with the note.

5.4 Browse Notes

Browsing notes involves the user finding notes that they wish to interact with and using search filters to more easily find the entry they are looking for. The user may also delete a note entry they no longer wish to store. All notes are stored using Apple’s Core Data framework, meaning privacy is assured as no application besides NOVI is allowed to access the user’s data.

5.4.1 Browse and Search

Voice search allows the user to give their query entirely through voice, after which they will be able to hear the results through VoiceOver. The same voice search button is available from the main menu. Simply dragging one's finger over each entry will trigger VoiceOver to start reading the title of the note and the notebook the note is classified under.

5.4.2 Note Interaction

Currently, voice search allows playback, pause, and stop note dictation. This will allow the user to parse their notes in more palatable chunks or pause in the case that something else requires their attention. VoiceOver currently provides a way to edit the text field through voice, though we are developing a more intuitive interface for this in the next release.

Chapter 6

Conceptual Model

The following screenshots demonstrate the basic functions of the application. Our goal with the UI design is a minimal, simplistic look and feel to accommodate various levels of visual impairment. In addition to a simple UI, we provide voice prompts and haptic feedback to guide users throughout all functions of the application and minimize confusion or error. Our preliminary research showed that the majority of visually impaired iPhone users primarily rely on using VoiceOver to navigate buttons and other visual elements on-screen. We made sure to include all necessary accessibility labels to seamlessly integrate VoiceOver.

6.1 Main Menu

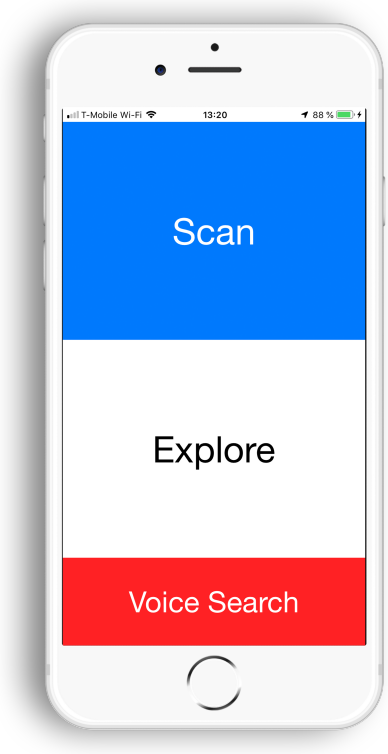


Figure 6.1: Main Menu

6.2 Scan Notes

Figure 6.2: Scanning Interface

6.3 Explore Notes

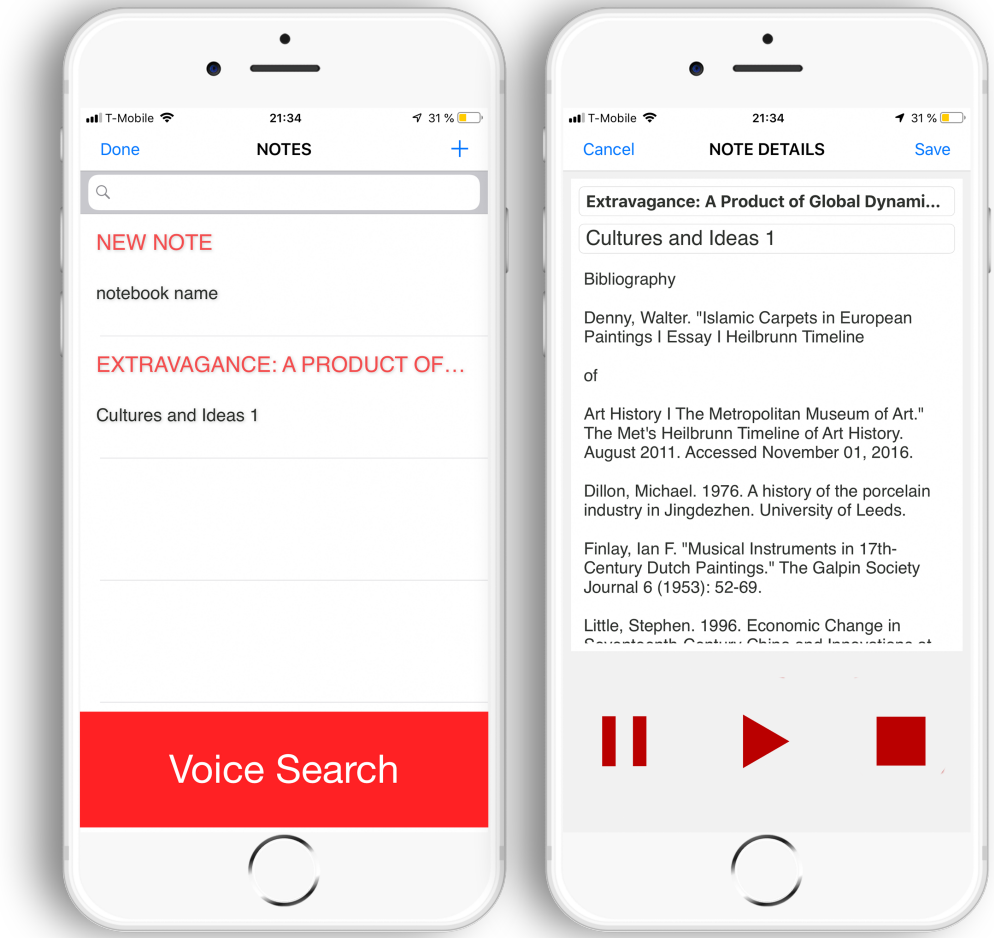


Figure 6.3: Explore Notes Navigation Example

Chapter 7

Technologies Used

The following technologies will be used to build the application.

- **Swift**: A robust and intuitive programming language created by Apple for building apps for iOS devices.
- **Tesseract OCR iOS**: An iOS wrapper of an optical character recognition tool maintained by Google.
- **Core Data**: An object graph and persistence framework provided by Apple that interfaces directly with SQLite.
- **Vision**: A powerful framework from Apple that provides solutions to computer vision challenges through a consistent interface.
- **Natural Language**: A framework designed to provide high-performance, on-device APIs for fundamental natural language processing (NLP) tasks across all Apple platforms.

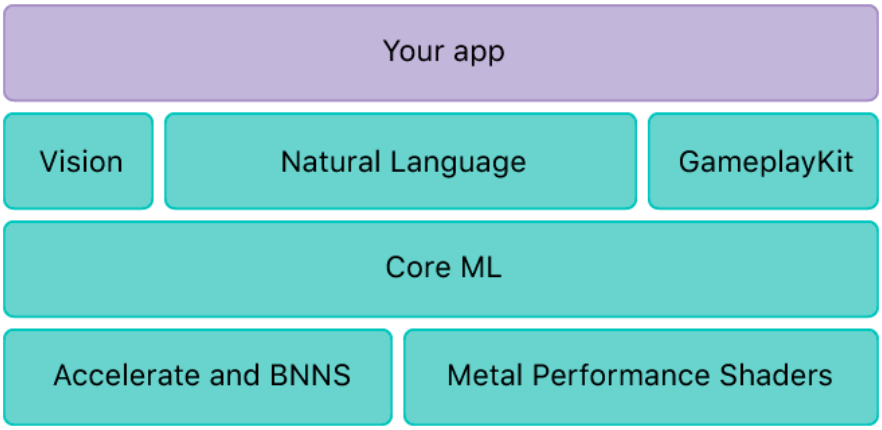


Figure 7.1: Apple ML Stack

Chapter 8

Architectural Diagram

Our system uses a model, view, controller architecture (MVC). Each user sees and interacts with the view, while the controller then makes changes to the model depending on the users actions on the view. The model holds the database and is used to populate the view's user interface. The MVC architecture is very common in mobile phone applications.

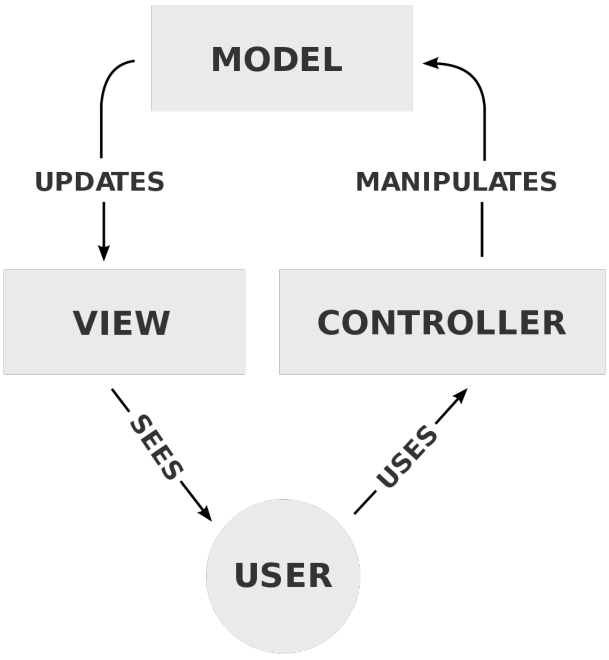


Figure 8.1: Architectural Diagram

Chapter 9

Design Rationale

The following sections will go over why we chose our user interface design and the technologies for our application.

9.1 User Interface

The user interface was designed to allow visually impaired users to navigate all of the application's features with ease. People with visual disabilities depend mainly on their hearing and feeling senses to perceive the world around them. Thus, our application has been designed to incorporate the use of sound and vibration to interact with users. This will allow our intended users to fully use our application without having to depend on seeing their mobile device's screen. The screen will have a simple display to navigate the application for people who are assisting our intended users.

9.2 Technologies Used

- **Swift**: This programming language was chosen for its ability to develop iOS mobile applications. We chose to make an iOS mobile application because we already had iOS mobile devices and because of its popularity and ease of use.
- **Tesseract OCR iOS**: We chose this optical character recognition framework because it is easily integrable with iOS applications through Cocoapods. It also provides a very simple API and well trained model that allowed us to build a pretty accurate prototype of our application. Compared to the other open source options, like SwiftOCR, Tesseract OCR performed significantly better at processing images of full page notes, thus it was a better choice for our specific application.
- **Core Data**: This data management framework from Apple was chosen because of its simplicity and ease of integration with our application. Instead of having to deal with SQL, Core Data allowed us to save our textual data simply through basic Swift commands.

- **Vision**: We chose this computer vision framework from Apple in order to isolate text strings within a photograph of text. It uses the newer iPhone's dedicated neural network cores to quickly complete computer vision tasks. By isolating the strings, we were able to test the OCR component on more isolated text images.
- **Natural Language**: Apple's NLP framework provided us with the functionality to convert user audio into text. We chose this framework because of its accuracy and ease of use within an iOS application.

Chapter 10

Test Plan

The following sections describe how each respective type of testing was implemented for our system.

10.1 Unit Testing

The application's code was written in a modular fashion with the goal of having high cohesion and low coupling for each module/class. Each module/class was tested to ensure its independent functionality. In addition, each function within the modules was unit tested as well. By doing this, we ensured proper functionality of the system and quickly corrected any malfunctions by knowing what module was faulty.

10.2 GitFlow and Integration Testing

We utilized GitFlow methodology and integration testing to ensure that each feature branch we merged into the development branch was fully functional and that the finalized development branch worked as intended. This method of testing ensured continuous functionality of the system during development.

10.3 User Experience Testing

In order to fully measure how useful our application was for those with visual disabilities, we found several volunteers who matched our intended user to test our system and provide feedback. We asked several questions about the user interface's ease of use, how likely they would be to use it, and suggestions on how to improve the overall application. Their feedback was pivotal as a measurement of the success of our application.

Chapter 11

Test Results

The following sections describe the results of testing our preliminary builds of the application.

11.1 Unit Testing

We ensured modular functionality through specific unit testing, considering each view controller as an individual module. The following are the qualitative results from our tests.

11.1.1 Segues

Segues between modules often ended up in the application force quitting. We found that programmatic implementation was needed rather than storyboard actions depending on the return values, return locations, or type of transition from table view to other view controllers. User testing also revealed moments in the application where cancelling the current action would result in an infinite loop due to an incorrect implementation of exit segues. Unit testing per module was extremely helpful in finding segue errors.

11.1.2 Voice UI

The Voice UI module was originally implemented as one view controller accessible from various screens. The functionality of this module came with few errors, except for segue issues. We split this module into different ones catered to serve specific use cases as detailed in the user experience test results below. Unit testing helped us isolate the functionality of commands to ensure each use case worked as intended.

11.2 User Experience Testing

Up until the point of writing this documentation, user experience (UX) testing has been primarily helpful in revealing actions and circumstances for unintended behavior to happen. Beyond bug fixes for modules working incorrectly, we found ourselves revisiting action flows and the design of controls most frequently. These are the most notable findings:

11.2.1 Voice UI

Rather than have one voice assistant for the whole app, we decided to give each use case a voice assistant, each with a smaller and more specific set of instructions pertinent to the current view. We thought this would help reduce the complexity and increase ease of use. Testing a newer version with the split assistants gave more positive reviews, while also simplifying the segues between storyboards.

11.2.2 Error Recovery

Scanning documents often came with minor misspelling or spacing errors. We are still in the process of ideating possible solutions for having a method of correcting potential errors without the need to look at the screen. One possibility is isolating the predicted words that have a low confidence level, reading the sentence around it, presenting a short list of potential matches, and asking the user if based on the context and predicted possibilities, if any of the matches should be confirmed to edit before saving. This will be highly dependent on the state of the OCR model.

Chapter 12

Future Work

12.1 User Interface

12.1.1 Enhanced Voice Assistant

The next iteration of the application should have a voice assistant that is able to handle variations in commands to catch the user's intended goal despite changing the order of words or using different word choices. This should, in theory, allow for more natural interactions with this application.

12.1.2 Further UX Research

Our testing so far has been limited to a narrow group of people around us. We make sure the testers get to practice using the VoiceOver interface and use the app with either a blindfold or strong glasses to simulate no and low vision. In addition to testing with people who have real vision disabilities, the next phase of testing should include more test subjects and with more iteration, we will be able to provide multiple samples of the app for A/B testing.

12.2 Optical Character Recognition Model

12.2.1 Improved Pre-Processing

The captured images by the user may not have the necessary qualities to allow the optical character recognition (OCR) model to accurately detect the text found in the image. The image quality may be poor because of the lighting, which is common with shadows or flash. If the text in the image is not aligned horizontally, the OCR model will not be able to properly recognize the characters. The borders and background of an image can also lead to extra characters being detected.

The current NOVI prototype combats poor lighting with a combination of techniques including an image luminous threshold filter, which is an image pre-processing technique that turns the image's pixels into true black or white depending on their level of luminance. Although our current pre-processing techniques have worked well, we would like to experiment with more complex image processing methods that will hopefully lead to complete clarity between

the text and the surrounding background. We would also like to expand our pre-processing to include text alignment correction and border cropping so that the OCR model will almost always receive an image it can process accurately.

12.2.2 Upgraded Model

Tesseract OCR has worked well enough for the first prototype, but it occasionally outputs incorrect text and adds characters that were not in the image. It's lack of accuracy when dealing with varied quality of images has lead us to look into better options for the OCR model. Apple offers a ML framework, Create ML, that would allow us to develop an iOS compatible text recognition model tailored for our specific use case. Apple also offers a feature in their Core ML framework that allows a pre-trained model file to be converted into a format that is compatible on iOS devices. Using these frameworks, we will develop an OCR model that rivals the best open source ones and ideally surpasses them. Firebase, an ML kit provided by Google, has proven to be the most accurate open source OCR model available at the moment. We will develop a prototype using the Firebase OCR model and then use that as a comparison point for the model we develop on our own.

12.3 Handwriting Recognition

Being able to recognize handwritten text is a fundamental piece to the future development of NOVI. A significant amount of class information is handwritten, which makes recognizing it essential for a student with a visual impairment. Similar to the methodology used for printed text recognition, we will be both developing our own model from scratch and also finding the most advanced open source option to compare our model to. This will ensure that we are using the most accurate model available, either our own or open source.

12.4 Real Time Layout Alignment Assistance

Taking a perfectly aligned picture of a document can be challenging, especially for people with varying levels of visual impairment. Our further research will be focused on developing a camera that can detect the edges of a document and guide a user into taking a perfectly aligned picture of their notes. Voice feedback could inform the user as to which way they can tilt or move the camera to get a clearer image, resulting in a more accurate scan. Once the document is aligned, the picture could be either taken automatically, directed by a voice command, or even by pressing the screen. This functionality is essential to providing visually disabled students with an independent learning experience.

12.5 Remaining Recommended Functional Requirements

There are still several recommended functional requirements that we will continue to research and develop. Those include: in-note searching, note summarization, in-note bookmarking, video as input, and electronic document as

input. These features would set NOVI apart from any other visual impairment assistance application available at the moment.

Chapter 13

Lessons Learned

13.1 Software Development

13.1.1 Adapting to Development Software Updates

Early in the software development stages, we noticed that the tutorials that helped us become familiar with the tools, or to build the basic foundations for our modules, most likely contained a lot of deprecated functions. In some cases we found in app code snippets, built earlier in the year, that were no longer functional due to deprecation. Although it was challenging to find updated versions of certain deprecated commands, it forced us to learn the reasons why Apple made such framework changes and how to effectively use them.

13.1.2 Refine Our Development Model

Throughout our development, we encountered many bugs and incompatibilities due to our development strategy of primarily relying on tutorials for the foundation of some of our modules. The lack of full understanding of some of these base components lead to unexpected actions and eventual errors as noted in our test results. The key lesson here is that while online development guides and resources are helpful in demonstrating frameworks and feature building, we should turn to them in times of confusion rather than relying on their concepts for our initial code-base.

13.1.3 Sandbox Learning

Even though git branches allow for nondestructive editing of the final product, we found it to be smoother to implement a component of the application if we learned it using a separate example from a tutorial or just experimenting with concepts like the Voice UI in new project files, separate from our main application. Creating these demo applications for specific components of the application allowed us to ensure independent functionality and eventually develop unit tests for them.

Bibliography

[1] "ACM Code of Ethics." *ACM Code of Ethics and Professional Conduct*, Association for Computing Machinery, https://www.acm.org/code-of-ethics.

[2] Baker, Justin. "Voice User Interfaces (VUI) - The Ultimate Designers Guide" *Medium*, https://medium.muz.li/voice-user-interfaces-vui-the-ultimate-designers-guide-8756cb2578a1.

[3] Balderrama, Wilson. "How to segue between ViewControllers with different storyboards" *Medium*, https://medium.com/@wilson.balderrama/how-to-segue-between-storyboards-86c582f976f7.

[4] Blair, Ian. "App Accessibility is The New Must in Mobile Development" *Buildfire*, https://buildfire.com/app-accessibility-mobile-development/.

[5] "Blindness and vision impairment." *Fact sheets — WHO*, World Health Organization, https://www.who.int/news-room/fact-sheets/detail/blindness-and-visual-impairment.

[6] "Core Data." *Core Data — Apple Developer Documentation*, Apple, developer.apple.com/documentation/coredata.

[7] "Core ML." *Core ML — Apple Developer Documentation*, Apple, developer.apple.com/documentation/coreml.

[8] "Create ML." *Create ML — Apple Developer Documentation*, Apple, developer.apple.com/documentation/createml.

[9] "Detecting Objects in Still Images." *Detecting Objects in Still Images — Apple Developer Documentation*, Apple, developer.apple.com/documentation/vision/detecting_objects_in_still_images.

[10] Edrisian, Daniel, "Building a Speech-to-Text App Using Speech Framework in iOS 10." *Appcoda*, 9 Aug. 2016, https://www.appcoda.com/siri-speech-framework/.

[11] Galiotto, Daniele, et al. "gali8/Tesseract-OCR-IOS." *Github*, 11 Apr. 2019, github.com/gali8/Tesseract-OCR-iOS.

[12] Goossens, Frederik, ”Designing a VUIŁŁVoice User Interface.” *UX Planet*, 14 May 2018, https://uxplanet.org/designing-a-vui-voice-user-interface-c0b3b9b57ace.

[13] ”ML Kit for Firebase — Firebase.” *Google*, Google, firebase.google.com/docs/ml-kit.

[14] ”Natural Language.” *Natural Language — Apple Developer Documentation*, Apple, developer.apple.com/documentation/naturallanguage.

[15] Pham, Khoa. ”Vision in IOS: Text Detection and Tesseract Recognition.” *Medium*, Flawless App Stories, 22 June 2018, medium.com/flawless-app-stories/vision-in-ios-text-detection-and-tesseract-recognition-26bbcd735d8f.

[16] Rames, Jeff. ”Speech Recognition Tutorial for iOS” *raywenderlich.com*, 13 Jun. 2017, https://www.raywenderlich.com/573-speech-recognition-tutorial-for-ios.

[17] Swift. *Swift — Apple Developer Documentation*, Apple, developer.apple.com/documentation/swift.

[18] Swift. *Swift — Apple Developer Documentation*, Apple, developer.apple.com/documentation/swift.

[19] Vision. *Vision — Apple Developer Documentation*, Apple, developer.apple.com/documentation/vision.

[20] Xiong, Ying. ”Fast and Accurate Document Detection for Scanning.” *Medium*, Flawless App Stories, 9 August 2016, https://blogs.dropbox.com/tech/2016/08/fast-and-accurate-document-detection-for-scanning/.

Appendix A

Deployment Guide

A.1 Install or Update Xcode

Install or upgrade to the newest Xcode:

1. Download the latest Xcode version from the Mac App Store using this link: `https://itunes.apple.com/us/app/xcode/id497799835?mt=12`

A.2 Install Command Line Developer Tools

Install the latest command line tools for Mac OS X through the terminal:

1. `xcode-select --install`

A.3 Git Clone Repository

Git clone the NOVI repository in a directory of your choosing:

1. Through terminal, enter the directory where you would like to download NOVI
2. `git clone https://github.com/clutesen/NOVI.git`

A.4 Run NOVI

Run the NOVI application through Xcode:

1. Open the Xcode application
2. Click on the ”Open another project...” button, as seen in Figure A.1, to find the NOVI directory previously downloaded

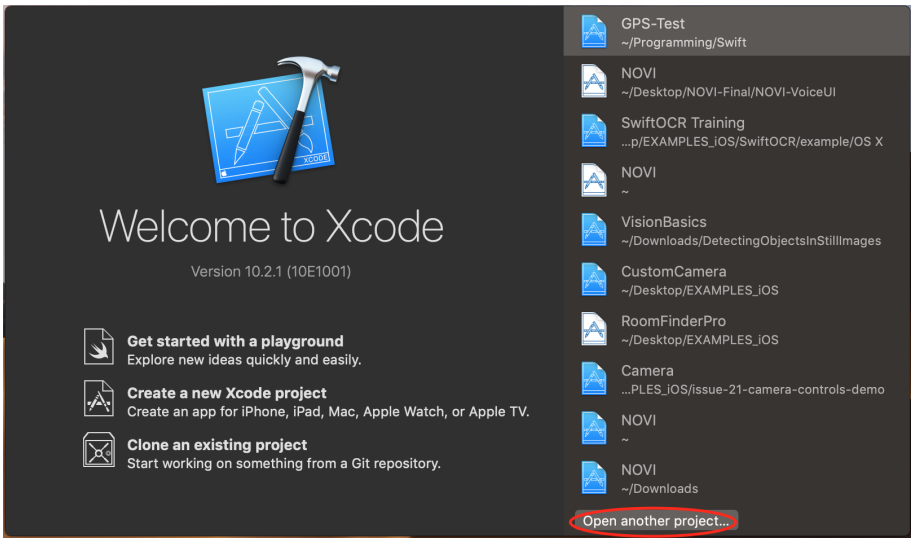


Figure A.1: Open the NOVI application in Xcode

3. Choose the ”NOVI.xcworkspace” file, as seen in Figure A.2, in order to include the necessary Cocoapods from the Podfile

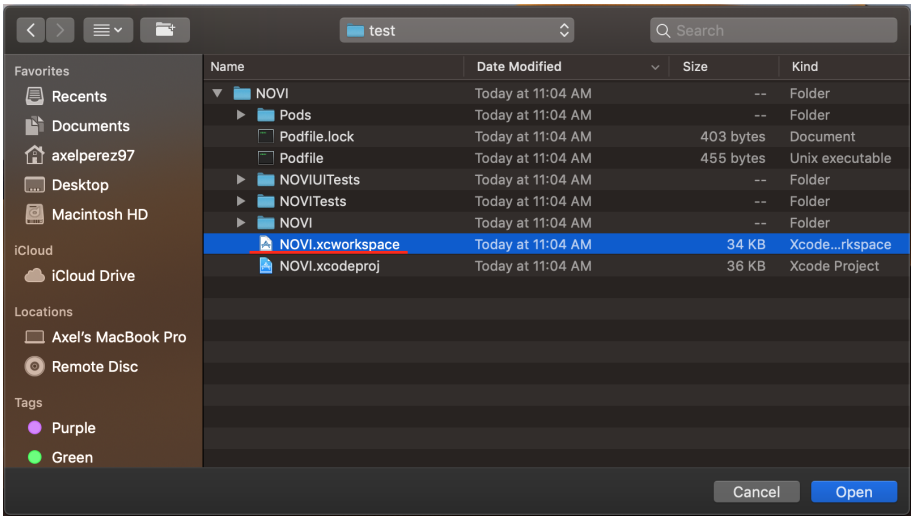


Figure A.2: Open the NOVI workspace file

4. Choose either an iOS simulator (recommend iPhone XR) or connect a physical iOS device to the computer (this option requires an Apple developers account and the appropriate setup), as seen in Figure A.3

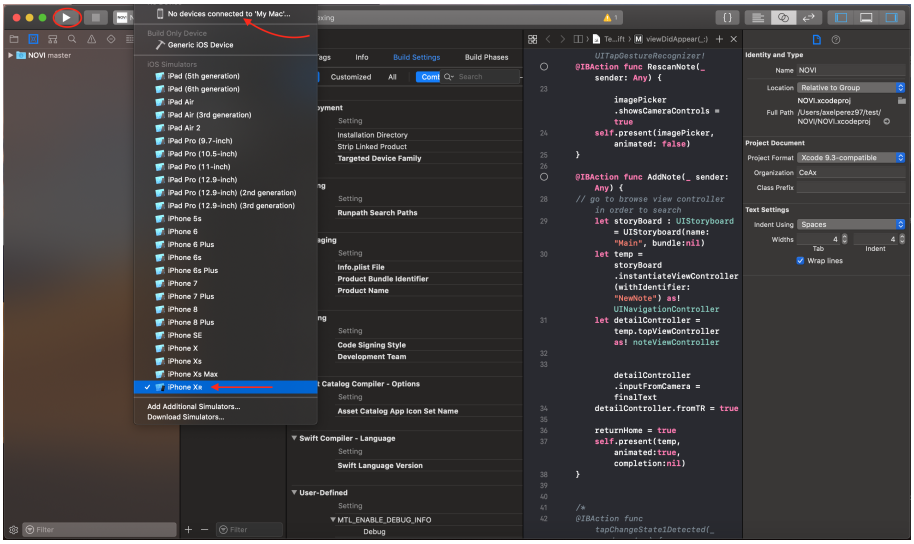


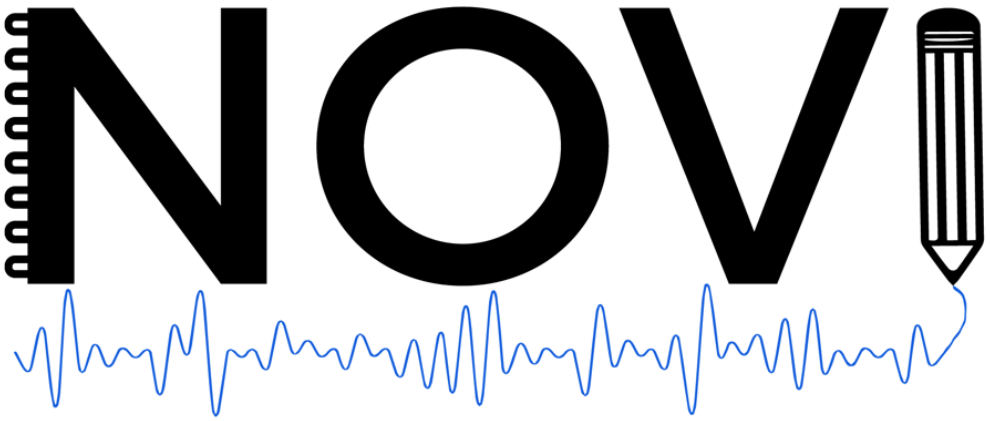
Figure A.3: Choose device to run NOVI

5. Run the NOVI application in Xcode by pressing the triangle button in the upper left corner, as seen in Figure A.3

6. Refer to the User Manual in Appendix B for instructions on the application's use

Appendix B

User Manual



B.1 Main Menu

B.1.1 Main actions

- **Scan:** takes you to the scanning interface where you can take a picture of the notes you wish to transcribe. The ”Scanning” section of this user guide goes more into detail of the process from picture to archived note.
- **Explore:** takes you to the notes explorer interface where you can choose which note you wish to interact with. ”The ”Notes Explorer” section of this user guide goes more into detail of the navigation options available such as filtering, editing, deleting, etc.

B.1.2 Voice assist options

These are the voice prompts you can give the application along with the expected behavior of each prompt:

- By saying **”Scan Notes”**, you will be taken to the notes scanning interface.
- By saying **”Explore Notes”**, you will be taken to the notes explorer view.
- By saying **”Search for note”**, in which ”note” is replaced by the keyword for the note you wish to find, you will be taken to the search results for that query.

B.2 Scanning

Upon selecting scan from the main menu or through a voice prompt, the following steps are required in order to create a new note.

B.2.1 Taking the picture

The standard iOS camera interface appears to take a picture of the notes to process. Future improvements will include changing this interface to include a voice guide to announce whether or not all edges of a page are detected or if text is being clipped from the sides.

B.2.2 Post-process Check

For this early release, the post-process check allows for developers to check that the bounding boxes correctly reflect the desired elements to be extracted and transcribed to plain text. This menu also gives the option to cancel or retake the photo in the case of an unsatisfactory scan.

B.3 Notes Explorer

After selecting explore from the main menu or through a voice prompt, you will have a list of all stored notes. You have the option of selecting a note and interacting with it, scrolling down to see more, or searching a keyword to filter down the notes and find the one you're looking for.

B.3.1 Voice assist options

These are the voice prompts you can give the application along with the expected behavior of each prompt:

- By saying **"Search for note"**, in which "note" is replaced by the keyword for the note you wish to find, you will be taken to the search results for that query.

B.4 Individual Note

This early release lets you playback and pause the stored note, future releases will include the ability to search and select playback points based on which result you wish to choose.

Appendix C

Maintenance Guide

Assuming this application were to be distributed in the app store, we recommend the following maintenance procedures in addition to routine bug fixes:

C.1 Bi-annual app audit

C.1.1 Audit 1: User Feedback

The first audit should focus on implementing user suggestions or modifying features to better serve users. Between official releases, we can release beta versions through TestFlight to invite external testers to give us a better size and variation of feedback. Testflight is a developer tool provided by Apple that allows us to upload a pre-release version and gather feedback data from voluntary testers around the world. By combining this data with real-world UX testing, we can make sure the app stays updated to cater to current users' needs and takes full advantage of emerging APIs and technologies.

C.1.2 Audit 2: Post-WWDC Feature Evaluation

WWDC is Apple's annual conference at which they showcase their newest software and technologies to software developers. New iOS developer tools are presented each year, and these should be taken into consideration to potentially add, replace, or modify features in a new release. By embracing the latest and greatest software, we can make sure that the application will work and adapt to new iterations of hardware or core frameworks. In addition to this, the application remains relevant, becomes more useful, and better helps users. For example, the 2019 conference which took place during the last phases of this project announced a new voice control system which gives developers more effective tools for creating an entirely hands-free experience on iOS.