Team B3 – SceneScribe

Aditi Narasimhan, Nithya Sampath, Jaspreet Singh

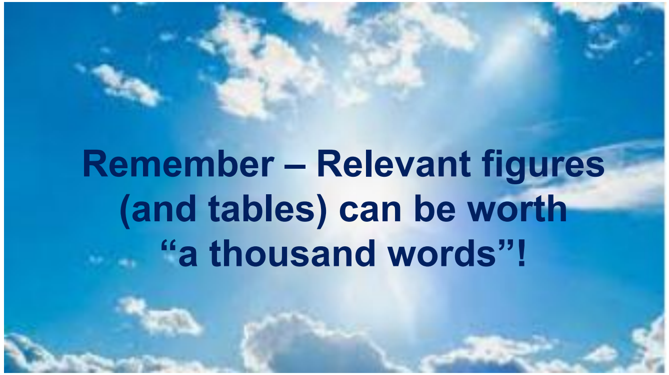
Add your 12 slides after this slide… [remember, 12 min talk + 3 min Q/A]

For more information about formatting or importing slides see:
https://gsuite.google.com/learning-center/products/slides/get-started/

Make sure to cover
(refer to the Proposal Presentation Guidance):

- Use Case
- Use-Case Requirements, especially Quantitative
- Technical Challenges
- Solution Approach
- Testing, Verification and Metrics
- Tasks and Division of Labor
- Schedule

Consider that this slide already works as a introduction slide so use your first slide wisely

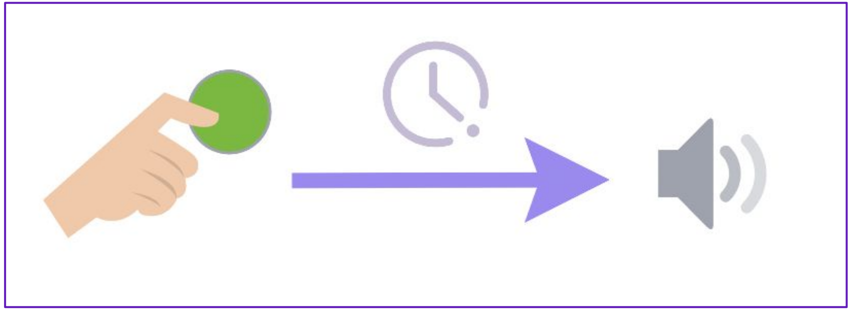


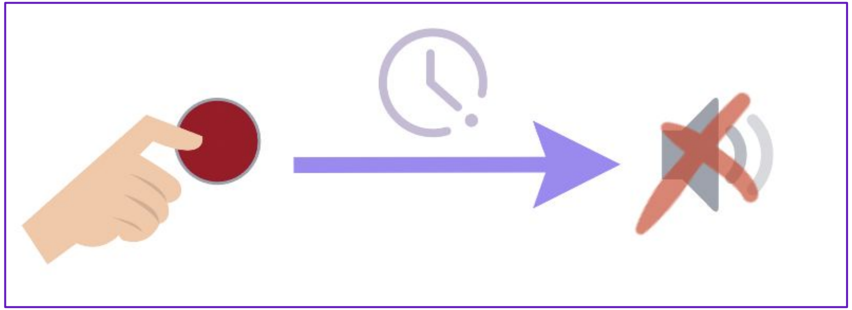
Remember – Relevant figures (and tables) can be worth "a thousand words"!

Use Case

- **Problem:** visually impaired people cannot easily read text on whiteboards and slides in the classroom, as a professor is presenting.
- **Scope:** our solution addresses reading text **during a lecture/presentation**.
  - The device will be a universal **camera attachment** which clips onto glasses, uses an ML model to **extract text**, and reads the text **aloud** to the user through an **iOS app** upon a **button press**.
- **ECE Areas:**
  - **Signals:** pre-processing images, CNN as ML model
  - **Hardware:** camera, start/stop buttons, microcontroller (RPi/Arduino)
  - **Software:** Swift for the app, AWS, integration

Use Case Requirements





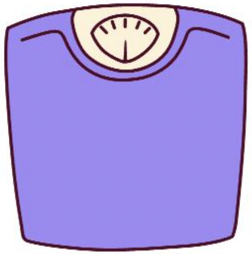
Requirement 1:

Latency from 'start' button press to the beginning of the audio should be ≤ 8 seconds (avg time to get out a phone and take a picture).

Requirement 2:

Latency from 'stop' button press to no audio playing should be ≤ 10 ms.

Use Case Requirements



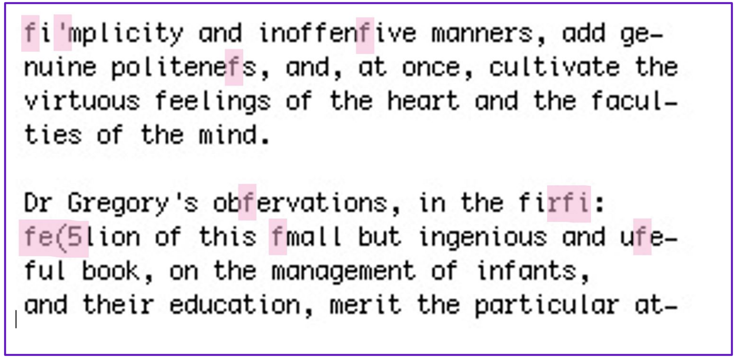
Requirement 3:

Weight of attachment on glasses should be ≤ 30 grams (average glasses are 40g, heaviest glasses are 70g).

Requirement 4:

The battery life of the device should be ≥ 6 hours (length of the average amount of teaching hours in a day).

Use Case Requirements



fi'mplicity and inoffenfive manners, add genuine politenefs, and, at once, cultivate the virtuous feelings of the heart and the faculties of the mind.

Dr Gregory's obfervations, in the firfi: fe(5lion of this fmall but ingenious and ufeful book, on the management of infants, and their education, merit the particular at-

Requirement 5:

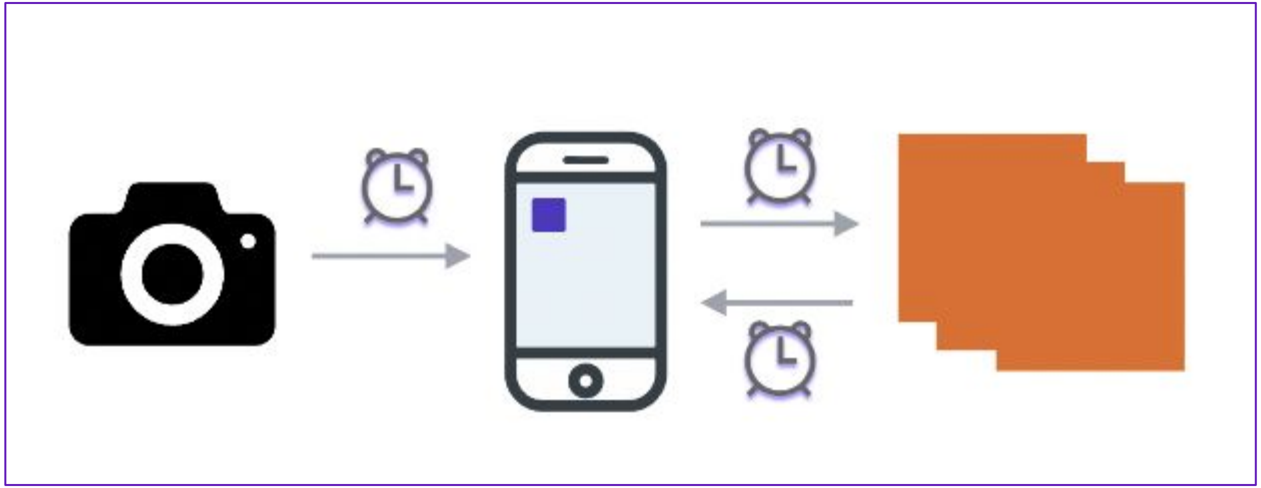
The app should consume an appropriate amount of energy on the mobile device: ≤ 25% of the device battery when used for 6 hours.

Requirement 6:

~100% of well-formatted, standard font words that are spelled correctly must be accurately identified.

Technical Challenge #1

- **Challenge:** optimizing speed of ML model and data transfer.
- **Relevant Requirements:** latency between pressing ‘read image’ button and speech output
- **Risk Mitigation:** using a smaller-size NN to decrease computation time.



Technical Challenge #2

- **Challenge:** accurately transcribing and reading mathematical symbols.
- **Relevant Requirements:** extracted text accuracy.
- **Risk Mitigation:** include training data which includes special characters and equations, increase the complexity of the model to recognize mathematical symbols.

$\int f(x)dx = 0$ → **“The integral of ‘eff’ of ‘ex,’ ‘dee ex’ equals zero”**

Technical Challenge #3

- **Challenge:** performing extraction on poor quality images (blurry, low-contrast, etc.).
- **Relevant Requirements:** extracted text accuracy.
- **Risk Mitigation:** image pre-processing, use a high quality camera, include training images with low contrast and a variety of fonts.

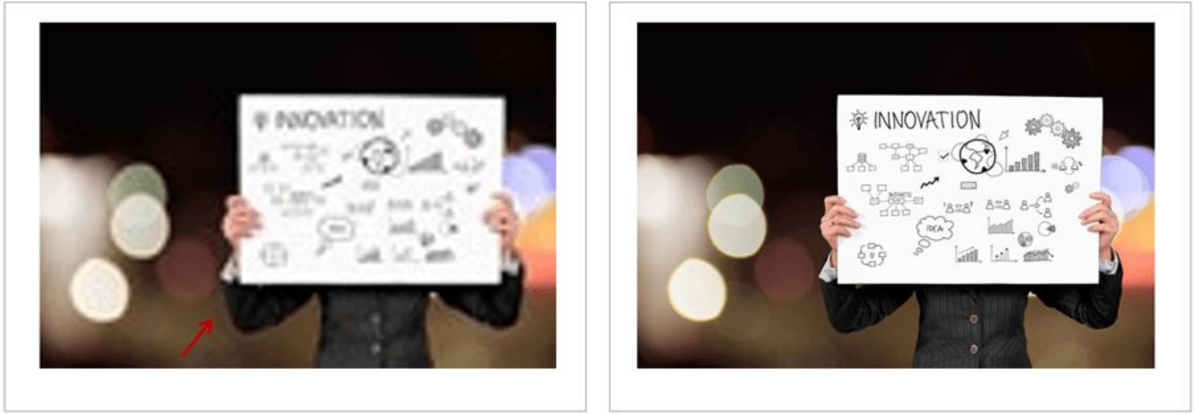
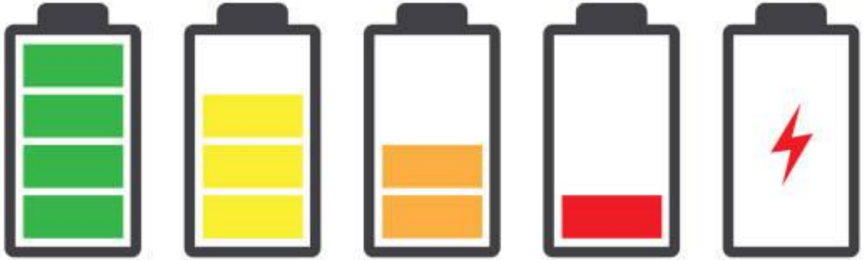


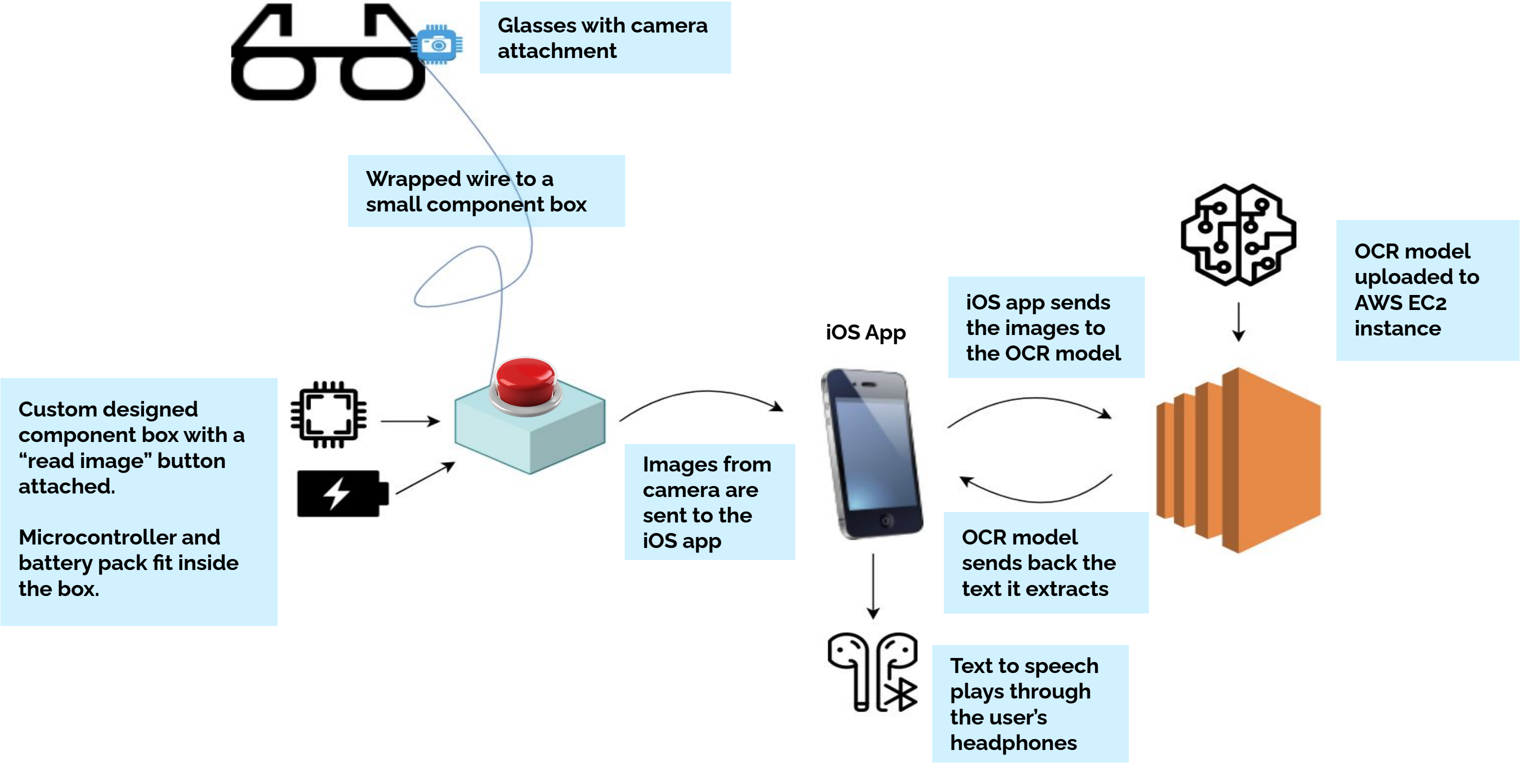
Image Source: https://www.chillibreeze.com/dos-donts-when-using-images-powerpoint-slides-ebook/

Technical Challenge #4

- **Challenge:** minimizing energy consumption of device.
- **Relevant Requirements:** device battery life
- **Risk Mitigation:** compare and switch out components (such as microcontroller) based on power consumption if necessary, use a different method of transmission that consumes less power



Solution Approach



Glasses with camera attachment

Wrapped wire to a small component box

Custom designed component box with a “read image” button attached.

Microcontroller and battery pack fit inside the box.

Images from camera are sent to the iOS app

iOS App

iOS app sends the images to the OCR model

OCR model uploaded to AWS EC2 instance

OCR model sends back the text it extracts

Text to speech plays through the user's headphones

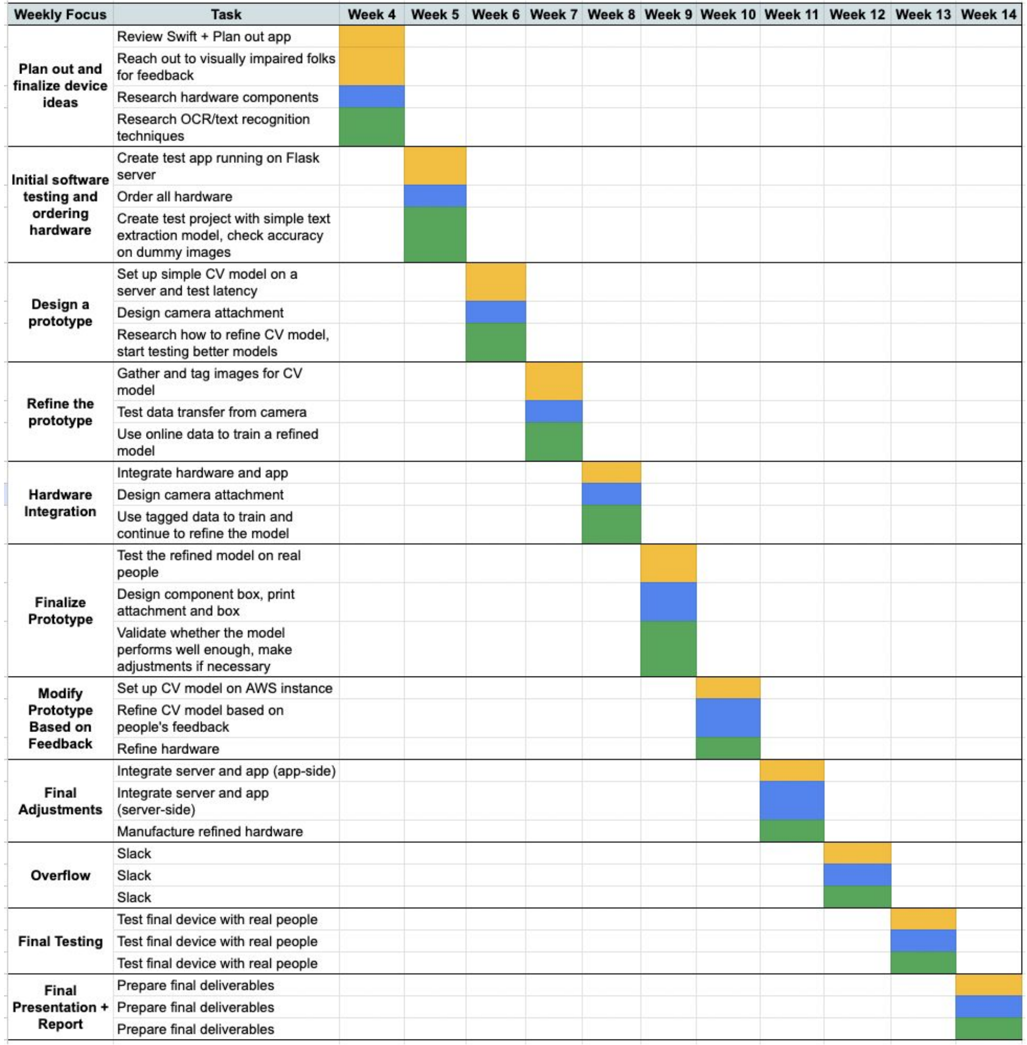
Testing, Verification, and Metrics

1. **Measuring test set accuracy** from ML model: character error rate (CER).
2. **User testing**:
   a. **Visually-impaired volunteers**: Helpful, Somewhat Helpful, Mostly Helpful, or Unhelpful?
   b. **Students from different fields**: Correct, Mostly Correct, Considerable Errors, or Incorrect?
3. **Latency tests**: measure time from button press to start/stop audio.
4. **Battery tests**: measure the amount of time the device and app can run consistently before needing a recharge.
5. **Weight**: measure the weight of attachments on the glasses themselves, as well as separate accessories.

Tasks & Division of Labor

Hardware (Jaspreet)	App & Integration (Aditi)	ML Model (Nithya)	Everyone
Research components and consider tradeoffs: camera, MCU w/WiFi, buttons, battery	Review Swift, create/implement initial design/wireframe for app	Labelling image data (~500 manual images, ≥ 1000 images online)	Gathering image data
Test data transfer from camera	Integrate hardware with app	Comparing advantages of different models (Google Cloud Vision, pytesseract.)	Test device with real people
Design & refine component box	Set up CV model on local test server, AWS server, then integrate	Initial training of model	Refine prototype based on user feedback
Design and refine camera attachment piece	Test functionality of app with accessibility settings on phone	Making tweaks (hyperparameter tuning, switching model, etc.) based on test accuracy	Work on portfolio, presentations, final project

Schedule



Weekly Focus	Task	Week 4	Week 5	Week 6	Week 7	Week 8	Week 9	Week 10	Week 11	Week 12	Week 13	Week 14
Plan out and finalize device ideas	Review Swift + Plan out app	🟨										
	Reach out to visually impaired folks for feedback	🟨										
	Research hardware components	🟦										
	Research OCR/text recognition techniques	🟩										
Initial software testing and ordering hardware	Create test app running on Flask server		🟨									
	Order all hardware		🟦									
	Create test project with simple text extraction model, check accuracy on dummy images		🟩									
Design a prototype	Set up simple CV model on a server and test latency			🟨								
	Design camera attachment			🟦								
	Research how to refine CV model, start testing better models			🟩								
Refine the prototype	Gather and tag images for CV model				🟨							
	Test data transfer from camera				🟦							
	Use online data to train a refined model				🟩							
Hardware Integration	Integrate hardware and app					🟨						
	Design camera attachment					🟦						
	Use tagged data to train and continue to refine the model					🟩						
Finalize Prototype	Test the refined model on real people						🟨					
	Design component box, print attachment and box						🟦					
	Validate whether the model performs well enough, make adjustments if necessary						🟩					
Modify Prototype Based on Feedback	Set up CV model on AWS instance							🟨				
	Refine CV model based on people's feedback							🟦				
	Refine hardware							🟩				
Final Adjustments	Integrate server and app (app-side)								🟨			
	Integrate server and app (server-side)								🟦			
	Manufacture refined hardware								🟩			
Overflow	Slack									🟨		
	Slack									🟦		
	Slack									🟩		
Final Testing	Test final device with real people										🟨	
	Test final device with real people										🟦	
	Test final device with real people										🟩	
Final Presentation + Report	Prepare final deliverables											🟨
	Prepare final deliverables											🟦
	Prepare final deliverables											🟩

🟨 Aditi

🟦 Jaspreet

🟩 Nithya