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 Design Review Guidance):

- Use Case / Application
- Use-Case Requirements, especially quantitative
- Solution Approach (include Design Requirements here)
- System Specification / Block Diagram
- Implementation Plan (include Design Trade Study(ies) here; i.e why choose that implementation)
- Test, Verification and Validation Plans (including quantitative metrics with target values)
- Project Management

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

Use Case & Application

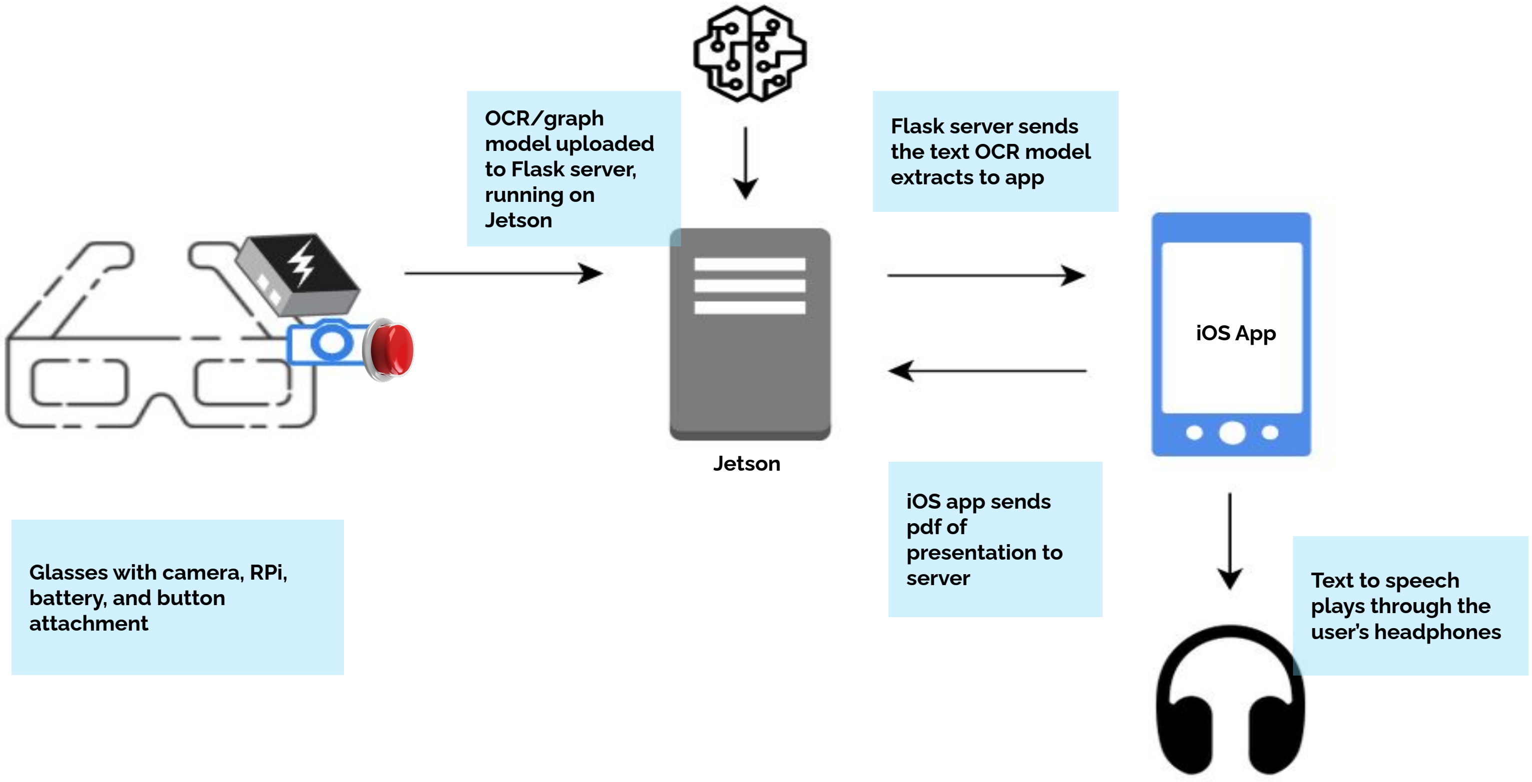
- **Problem:** Professors usually do not explain all content on their lecture slides, causing an **information disparity** between visually impaired and sighted students.
- **Scope:** our solution addresses reading text **during a lecture/presentation**.
  - The device will be a universal **camera attachment** which clips onto glasses, uses ML models to **extract text**, and reads the text **aloud** to the user through an **iOS app** upon a **button press**.
- **Major Changes**: slides will be **pre-uploaded** to app; new ML model now matches up image of slide with actual slide; we will generate **audio descriptions of graphs**.
  - Restricting use case to bar graphs, scatterplots, line graphs, and pie charts.

Source: https://www.dnaindia.com/education/report-ngos-take-caution-finding-writers-for-the-blind-in-gujarat-2591162

Use Case Requirements

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).
2. **Latency** from 'stop' button press **to no audio playing** should be ≤ 10 ms.
3. **Weight of attachment** on glasses should be ≤ 60 grams.
4. The **battery life** of the device should be ≥ 6 hours (length of the average amount of teaching hours in a day).
5. The app should have **appropriate power consumption** on the mobile device: ≤ 25% of the device battery when used for 6 hours.
6. ~100% of well-formatted, standard font words that are spelled correctly must be **accurately identified**.
7. Must be able to **identify existence of graphs** with close to 100% accuracy, and identify type of graph (line, scatterplot, bar, pie) and axis labels in description.

Solution Approach: Diagram



Solution Approach: Considerations

- **Health**: Must be electrically safe, shouldn't overheat, components should not touch skin directly.
- **Safety**: Our product should only be used as a learning aid, *not* a substitute for accessibility or navigation tools.
- **Welfare**: The attachment should be light and comfortable to wear.
- **Global**: To limit complexity, our product will only be able to recognize English text, but to increase global accessibility, an expansion could be to include different languages.
- **Environmental:** Should be energy efficient, shouldn't deplete battery too quickly.
- **Economic**: Should be affordable, so we've chosen cheaper components.

Camera Attachment Design

- Send images wirelessly from camera to server on button press
- **Raspberry Pi Zero WH**
  - GPIO pins for button inputs
  - **WiFi** connectivity
  - **PiSugar 2** Power Module + Rechargeable Battery
- Camera Module: **Arducam 5MP OV5647 for Pi Zero**
- **3D printed case** for components that attaches to glasses

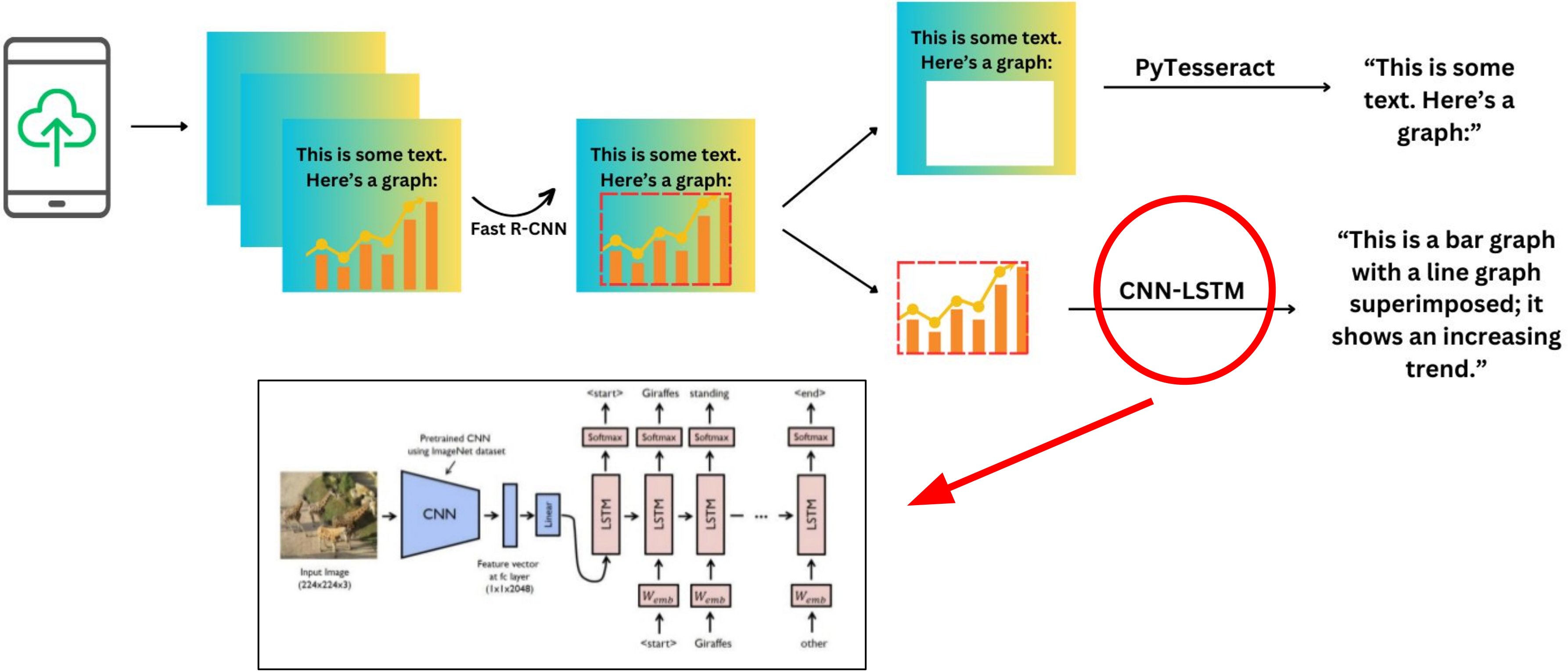
iOS App + Server Design

- Use a **Flask server** running on a **Jetson**.
  - Packets sent to AWS can be sniffed.
  - AWS incurs an extra cost out of pocket.
- Upload PDF to app with a large button on the app with **haptics**.
  - Button will vibrate when clicked, and will speak out loud: “upload a PDF”.
  - With accessibility mode is “on”, each file the user can upload will be spoken out loud, and the user can swipe through the files to upload it.
  - Once uploaded, the user can click on it again to upload another pdf.

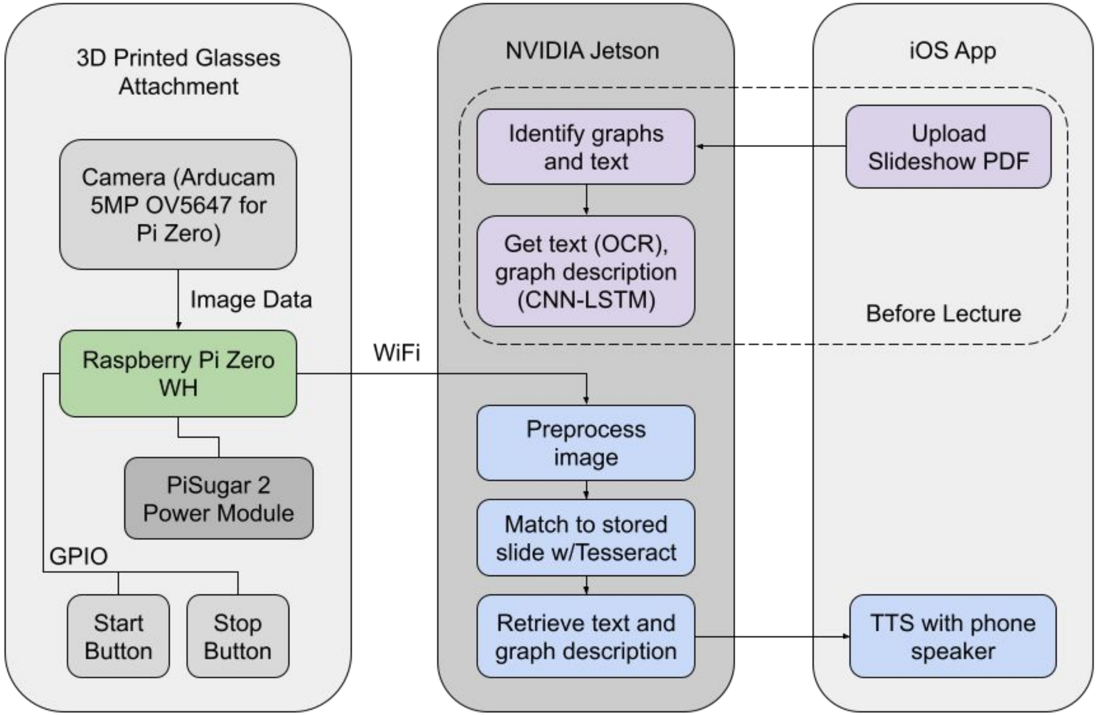
Slide to Text Pipeline

Before Class	During Class
1. User **uploads slides** to app. 2. For each slide: a. Fast R-CNN gets **bounding boxes** around graphs (object detection). b. Slide with **whited-out graph sections** → PyTesseract to extract text. c. CNN-LSTM model gets **graph description**. d. Store slide text & graph description.	1. User **takes picture** of slide. 2. Picture is **preprocessed** (deskewed, warped, gray-scaled). 3. **Extract** and post-process text (PyTesseract, NLTK's autocorrect). 4. **Match slide** based on extracted text. 5. Matched text and graph description are **read aloud** with TTS.

ML Models Design



Block Diagram



Implementation Plan

Hardware:

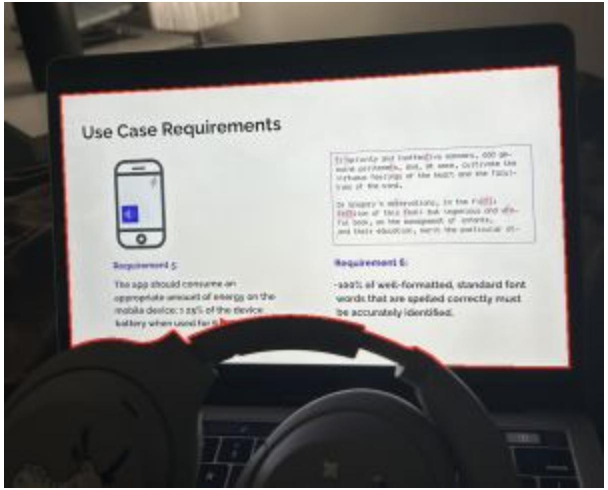
- Raspberry Pi, PiSugar, Arducam module will all be purchased.
- Component case will be custom and 3D printed.

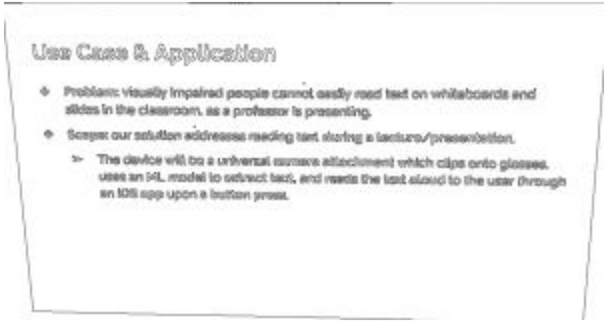
Software:

- iOS app developed with Swift and SwiftUI, scraped.
- Flask server will be set up on Jetson.
- RPi will be integrated with the server.

ML:

- Fast R-CNN for bounding boxes, PyTesseract, and preprocessing code will be custom-tuned to our dataset.
- Code for CNN-LSTM architecture borrowed, but lots of changes will have to be made for our use-case.
- Custom reference descriptions for all graphs in dataset.





Testing, Verification, and Metrics

1. **Measuring test set accuracy** from ML model: character error rate (CER).
    a. Metric for graphs will be similarity score between reference and candidate embeddings.
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.

Schedule

Weekly Focus	Task	Week 5	Week 6	Week 7	Week 8	Week 9	Week 10	Week 11	Week 12	Week 13	Week 14
Initial software testing and ordering hardware	Pre/post-process data with CV techniques										
	Order all hardware										
	Create test project with simple text extraction model, check accuracy on dummy images										
Design a prototype	Set up simple iOS app and Flask server. Also talk to disability resources staff										
	Setup image to server pipeline										
	Research model for bounding boxes around graphs										
Refine the prototype	Gather and tag images for CV model										
	Test data transfer from camera										
	Start gathering and labeling references for graph description. Look into matching algorithm.										
Hardware Integration	Integrate hardware and app										
	Design glasses attachment										
	Continue gathering graph description data, finalize graph description model										
Finalize Prototype	Test the refined model on real people										
	Print and test glasses attachment										
	Validate whether the model performs well enough, make adjustments if necessary										
Modify Prototype Based on Feedback	Set up ML model on server										
	Refine hardware										
	Refine ML model based on people's feedback										
Final Adjustments	Integrate server and app (app-side)										
	Manufacture refined hardware										
	Integrate server and app (server-side)										
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