Use-Case

Money Tracker APP with Voice Input

- Customers could use EchoBudget at home and outside with their phones or desktops
- Customers could verbally:
  - Record the spending
  - Change each entry
  - Request report for a given range
- **ECE Areas**:
  - Software
  - Signals

Our Customers

- **Visually impaired people**
- **People with physical disabilities**
    - If typing is painful or difficult
- **Elder people**
    - Easier user interfaces -> easier interaction
- **People who want to record their spendings**
    - Hands-free operation
    - Quick and efficient data entry

Use-Case Requirements

- **Languages**
    - Should at least support English
- **System should be portable**
    - Weight: <= 200g (weight of a mobile phone)
- **Record customer speech**
    - Frequency: 80-260 Hz (general frequency for adult male is 85-155 Hz, and for adult female is 165-255 Hz)
    - Volume: 50-65 dB (normal conversation, people may feel annoyed if the sound >70dB)
    - Should still work in noisy environment
- **Identify amount of money customer spends**
    - Accuracy: expected >= 95%

Use-Case Requirements

- **Categorize each spending**
  - Should be accurate with given categories (currently 5 different categories)
  - Accuracy: expected >= 90%
  - No customized category
- **Read out the information upon request**
  - E.g. generate a report for last month
- **User Interface**
  - Concise and simple
  - Elder people could learn how to use it within one day

Technical Challenge and Solution: Speech Recognition

- **Challenge**
  - Effectively record customer speech under 65-75 dBA environment
  - Achieve 80% average accuracy
- **Solution**
  - Noise Reduction through signal processing
    - Spectral gating using PyAudio, PyPI noisereduce, and SciPy
  - SpeechRecognition library for speech-to-text transformation
- **Risk Mitigation**
  - Recommend users to use the application in quiet environment

Technical Challenge and Solution: **NLP Model**

- **Challenge**
  - Correctly recognize user commands
  - Correctly identify the amount of money customer spend and item categories
- **Solution**
  - Rule-based matching model for user command recognition(e.g. “Generate report”)
    - spaCy library for matching command words to app functionalities
  - Custom Named Entity Recognition(NER) model for item categorization and price number
    - Random forest in Scikit-learning model
    - Pre-trained BERT models
  - Adjust NLP models base on edge cases found during testing
- **Risk Mitigation**
  - Introduce standardized command format instructions to the users
  - Provide corresponding guidance towards more accurately recognized commands

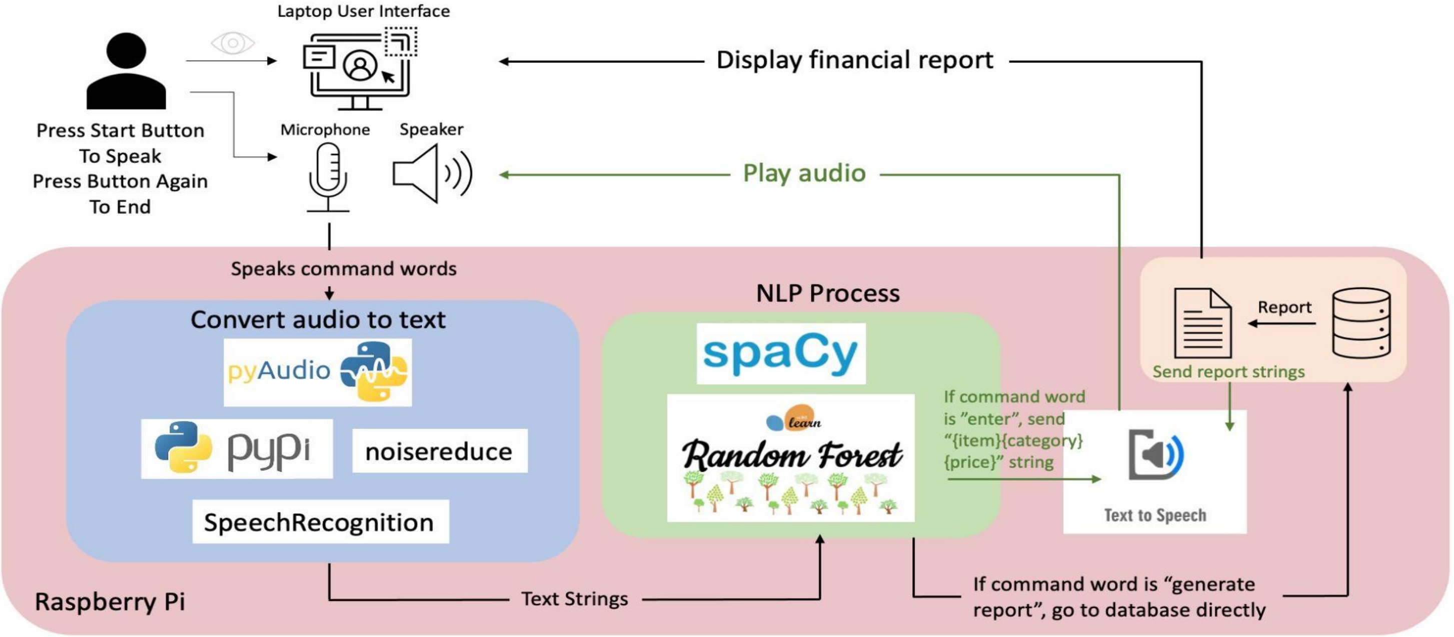
Technical Challenge and Solution: **Usability**

- **Challenge**
  - Accessible for visually-impaired groups
  - Hand-free operations and corresponding trade-offs
- **Solution**
  - A “Start/End Speaking” button is designed to avoid continuous idle listening of commands
    - When user wants to give commands, press the button and speak
    - Press the button again to finish the current speech recording session
  - Audio Assistant implemented using Text-To-Speech(TTS) model gTTS could serve basic functionalities including input entry verification, start/end recording notice, and report reading
- **Risk Mitigation**
  - The “Start/End Speaking” button is designed large enough for visually-impaired users to operate without difficulties

Solution Approach

- **Software**
  - Django framework Web Application
  - Speech Recognition models
    - PyAudio, PyPI noisereduce, SpeechRecognition
  - Natural Language Processing models
    - spaCy, Scikit-learning random forest/BERT model
  - Text-To-Speech model: gTTS
- **Hardware**
  - Raspberry Pi
  - USB microphone and speaker

Solution Approach Block Diagram



Normal Mode
Audio Assistant Mode

Testing, Verification and Metrics

Functional tests:

- Run Django-based web application on Raspberry Pi
- Turn on and off microphone by clicking button or space on keyboard
- Create an entry for parsed expense information on web app
- Generate a spending report based on a time range

User tests:

- Invite 5 volunteers to use our app, rate it out of 10 points and provide feedback

Performance tests:

- Speech recognition test (80% accuracy, or 20% word error rate)
- NLP test (identify amount spent and item names with 95% accuracy, classifying item names with 90% accuracy)
- Spending report should be generated in 100ms
- Spending entry is generated in 3 seconds after receiving user input

Tasks and Division of Labor

Grace Xiao

- Django framework setup on Raspberry Pi
- Web application MVC design
- Web application UI
- Web application - Manual input
- Web application - Report generation

Ella Yang

- NLP model selection
- NLP dataset creation
- NLP model training
- NLP integration

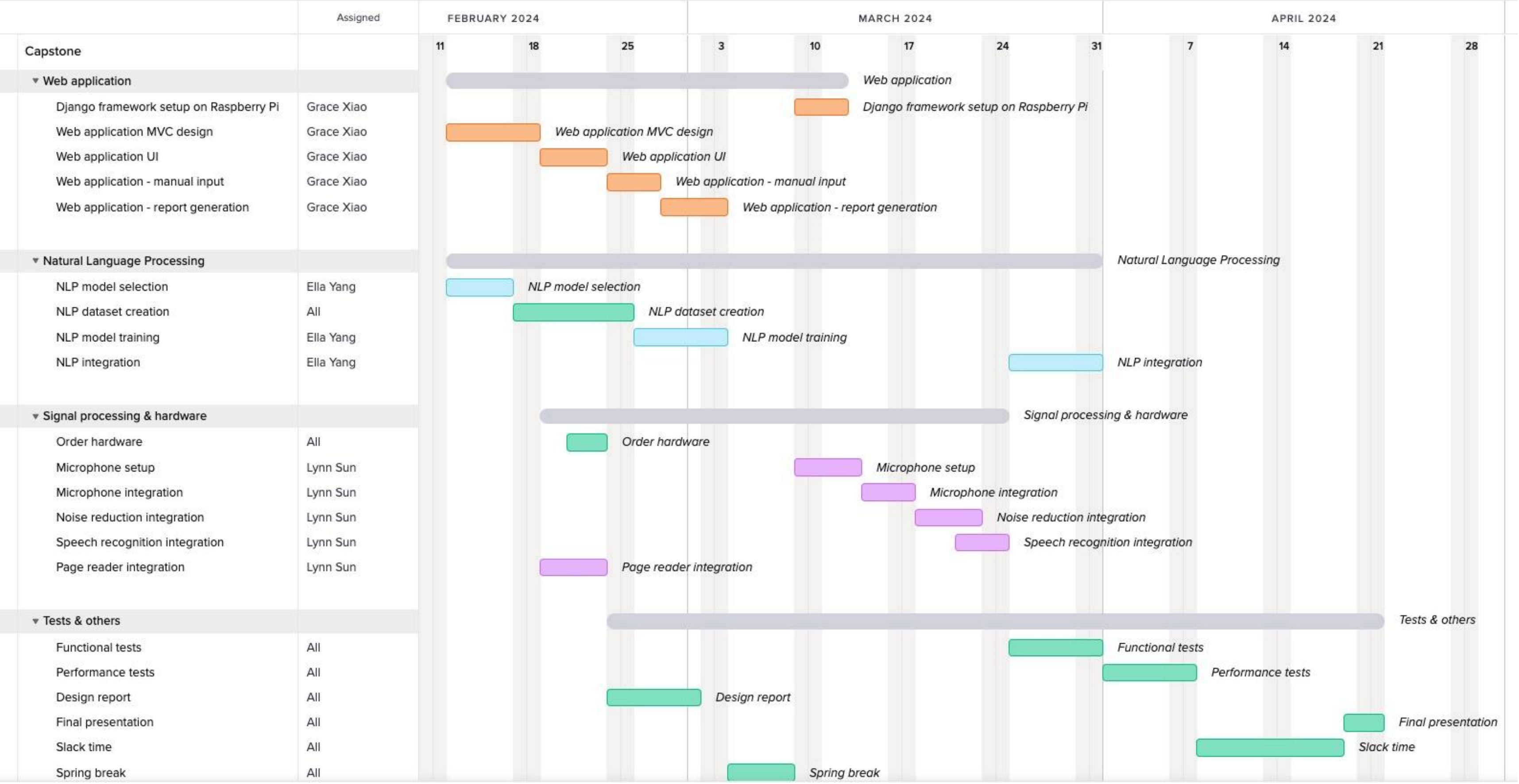
Lynn Sun

- Microphone setup
- Microphone integration
- Noise reduction integration
- Speech recognition integration
- Page reader integration

Everyone

- Order hardware
- Functional tests
- Performance tests

Schedule



	Assigned
Capstone	
▾ Web application	
Django framework setup on Raspberry Pi	Grace Xiao
Web application MVC design	Grace Xiao
Web application UI	Grace Xiao
Web application - manual input	Grace Xiao
Web application - report generation	Grace Xiao
▾ Natural Language Processing	
NLP model selection	Ella Yang
NLP dataset creation	All
NLP model training	Ella Yang
NLP integration	Ella Yang
▾ Signal processing & hardware	
Order hardware	All
Microphone setup	Lynn Sun
Microphone integration	Lynn Sun
Noise reduction integration	Lynn Sun
Speech recognition integration	Lynn Sun
Page reader integration	Lynn Sun
▾ Tests & others	
Functional tests	All
Performance tests	All
Design report	All
Final presentation	All
Slack time	All
Spring break	All