EchoBudget

Lynn Sun, Yuxuan Xiao, and Yixin Yang

Department of Electrical and Computer Engineering, Carnegie Mellon University

***Abstract*—The system is a web-based application that provides money tracking functionalities with audio input. Traditional money tracking apps are not accessible for visually impaired people and the elderly due to lack of visual aids and complex user interfaces. To meet their needs for money tracking, we propose to implement an application that records, shows, and analyzes spending solely with audio input and output.**

***Index Terms*—Design, natural language processing, noise reduction, signal processing, speech recognition, text-to-speech conversion, user interface, web application.**

I. Introduction

With the development of social welfare and humanitarianism, the living standard has been greatly enhanced for visually impaired and elderly people. However, their need for accessible money tracking methods is not satisfied because the traditional apps require the user to know what functionality each component has in a complex UI, which is too sophisticated for visually impaired and elderly people to manipulate. The voice assistants in smart phones are not capable of converting voice input to a specific command interpretable by a traditional money tracking app. In addition, many visually impaired and elderly people do not possess smartphones and thus have no access to the traditional apps.

A portable device with an app supporting audio-command conversion and content reader can address this issue. Our project is to develop a voice-controlled money tracking app on a low-cost device. The system can parse the audio input into parameterized commands and provide audio output of the generated contents to best satisfy our users' need for recording their expenses without visual help.

II. Use-Case Requirements

Available Operations: Our system EchoBudget would support common functionalities of money management apps in the market. Users are able to create, edit, and remove entries to record their daily expenses. In addition, users could ask our system to generate spending reports to acquire more information about their spending habits. While other applications require users to manually input all the information, our system could be controlled fully with voice commands. There would be a button on each page and users could push the button and begin to give their instructions. Our system would then complete the tasks assigned by users.

Voice Input: Users would communicate with our system like sending voice messages through their phones. Therefore, when users are holding our system and talking to it, our system would be able to receive the voice input. According to Gitnux, the average adult female arm length is about 27-31 inches (68.58-78.74 cm) and the average adult male arm length is about 28-34 inches (71.12-86.36 cm). Therefore, our system should be able to receive inputs with distances from 0 to 100cm. According to the Centers for Disease Control and Prevention, the average sound level for normal conversation is about 60 dB, and that for whisper is about 30 dB. Therefore, our system should work well with human voice ranges from 30 to 70 dB. Since we expect our users would use our system in outdoor circumstances including restaurants and supermarkets, our system should also work in a relatively noisy environment.

Latency: According to WebsiteBuilderExpert, 25% of users would leave if the websites did not load within 4 seconds. Therefore, we would expect our websites to give any update within 4 seconds. To be more specific, our system would create, edit, and delete the entry within 4 seconds. Additionally, our system would generate a report within 4 seconds.

Portability: It will be inconvenient for customers to bring a large and heavy device with them. Thus, the size of our system is designed to mimic the size of a mobile phone (iPhone 13 pro) and the weight of our system would be similar to that of the iPad Air (500g).

Battery Life: With the monitor on, our device should be able to operate for at least one hour. Since customers would only use our device for money management, one hour of operation would be enough to record daily spending. When the monitor is off, we would try to minimize the energy consumption to allow customers to charge the device less often and we would expect our system to operate for at least 24 hours with the monitor off.

Accessibility: Our system is designed to be friendly to both people with visual impairment and elder people.

To accommodate the needs of visually impaired people, other than the voice input command, we also provide a content reader. When users ask the system to read out the content (e.g. the report last week), our system will read out what is displayed on the screen. Additionally, the button that needs to be pushed to record the commands is designed to be large and fixed in a position on every page, making it easy for visually impaired people to interact with our system. For elder people, we would make concise UIs and expect elder people could master our system within one day.

III. Architecture and/or Principle of Operation

EchoBudget is a device that executes a web application on Raspberry Pi and allows users to interact with it via a touchscreen and a microphone. The design could be distributed into a hardware section, a signal processing section that could be further split into speech recognition and NLP processes, and a software section. The hardware and signal processing sections transform interactions in the physical world into data that can be interpreted via software. The software section analyzes the data and provides corresponding performances based on user requirements.

A. Hardware

Our system performs all functionalities on a Raspberry Pi 4 with a touchscreen monitor and a USB microphone. Customers could interact with the system via finger tapping and audio inputting. After an initial tutorial that provides users with a basic introduction to the web application through the built-in speaker, the primary user interface would be displayed on the monitor screen. Customers could use the microphone to deliver voice command inputs and the audio signals would be transmitted to the speech recognition subsystem.

B. Speech Recognition

The speech recognition pipeline is designed to convert the voice commands from the customers to text strings that would then be fed to the natural language processing models. The audio input stream goes through a noise reduction algorithm to guarantee that the voice commands can be effectively distinguished under environments with volume from 30 to 70 dB. The treated stream is then delivered to a trained speech recognition model and outputs as text strings.

C. Natural Language Processing

Once the audio is successfully converted to text strings, two natural language processing models could be used to parse the strings, extract key information including command words, item names, price numbers, entry numbers, or dates, and classify the distinguished item names to different categories based on the command inputted.

Command Parsing: A command verb should be parsed first for all received text strings. The corresponding web page would be rendered based on the command given. After loading the page, the model will parse the remaining information from the input string according to the command. If the NLP model fails to recognize the command word or date information, it will send a notification to the customers so that they could repeat their commands. As for parameters including item name and price number, the model would let the customers confirm whether the parsed elements are correct or not.

Classification: There are five standard categories for the entries including food, housing, necessities, entertainment, transportation, and a backup category named “others”. After the customer enters the item name and price, the new entry will automatically be assigned a category based on its name. The customers could modify the classification if needed by providing new commands.

D. Web Application

The Django framework web application is the major interface for the users to interact with the systems. The application allows customers to input new entries purchased to the database, view existing entries and modify if needed, and learn the speeding statistics via financial reports. To benefit the visually impaired groups, a large button is designed on the right-hand side of all pages of the application, and the customers could give their voice commands by tapping the button. The customers could speak up after a “start recording” audio notification, and they should tap the button again to stop the current voice input. An “end recording” notification will be delivered, indicating that the recording process ends and the customers could click the button for new inputs.

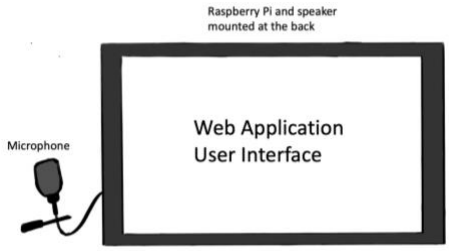
Based on the received commands, different pages of the web application would be rendered. Command verb “Enter” will render the Entry Enter page, and an item name and price number are expected to be parsed as the parameters for a new entry. An audio notification that repeats the entered information will be played via the speaker, and the customers could use the “Confirm” command to save the entry to the database. If any field is incorrect, command word “Change” with the corresponding field name and expected value should be given.

Customers could be able to view their entries via the command “View” or “View entries”. The page that displays entries for the current month will be rendered and each entry will be assigned a serial number. The customers use the “Read” command to ask for an audio version of the entry lists and could use “Modify” with a parameter serial number to adjust a specific entry. The modification page for that entry should then be rendered, and the customers could make changes via the same approach as the Entry Enter page adjustment operations. Command “Remove” with the serial number parameter of a specific entry is used when the customer wants to delete an existing entry.

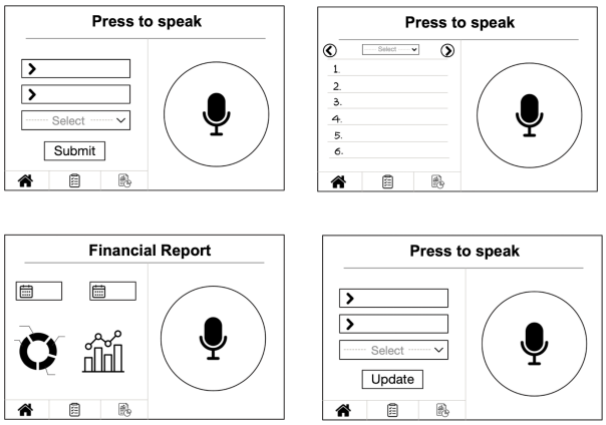
Command “Get”, or “Get report” refers to the Financial Report page, and the customer could use the additional command “Go to” with a Year-Month date information parameter to get the report of that specific month.

Customers could always use the command “Help” to request the application tutorial.

An additional set of ordinary UI is designed for customers to perform all operations manually and is displayed on the left-hand side of the screen.



(a)



(b)

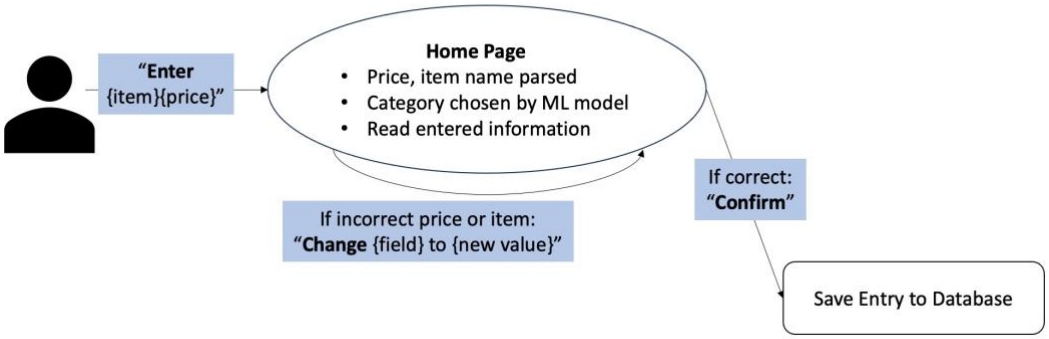
The customers could manually input item names and price numbers via a virtual keyboard in RPi, and select the corresponding category using the dropdown menu. The entry will be saved by tapping the "Submit" button.

Three tabs at the bottom of all pages are used to navigate between pages. The "home" icon indicates the Entry Enter page(Home Page), the list board icon represents the View Entry page, and the graph icon implies the Financial Report page.

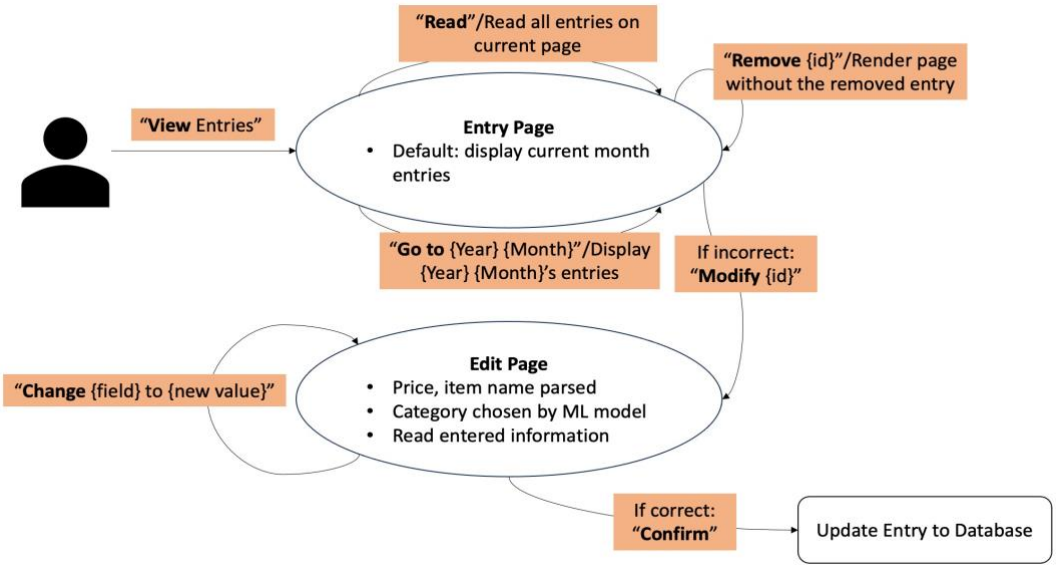
The entry modification page could be reached by tapping on a specific entry in the View Entry page. Similar to the Entry Enter page, the modified entry will be saved by clicking on the "Update" button, and the View Entry page will be rendered and displayed again.

Fig. 1. System drawing. (a) overall system. (b) User Interface Design.

(a)



(b)



(c)

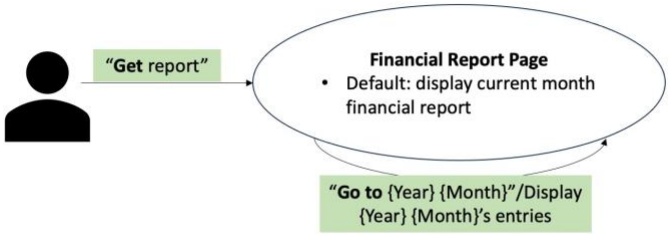


Fig. 2. State Transition Diagram for (a) Home Page. (b) Entry Page and Edit Page. (c) Financial Report Page.

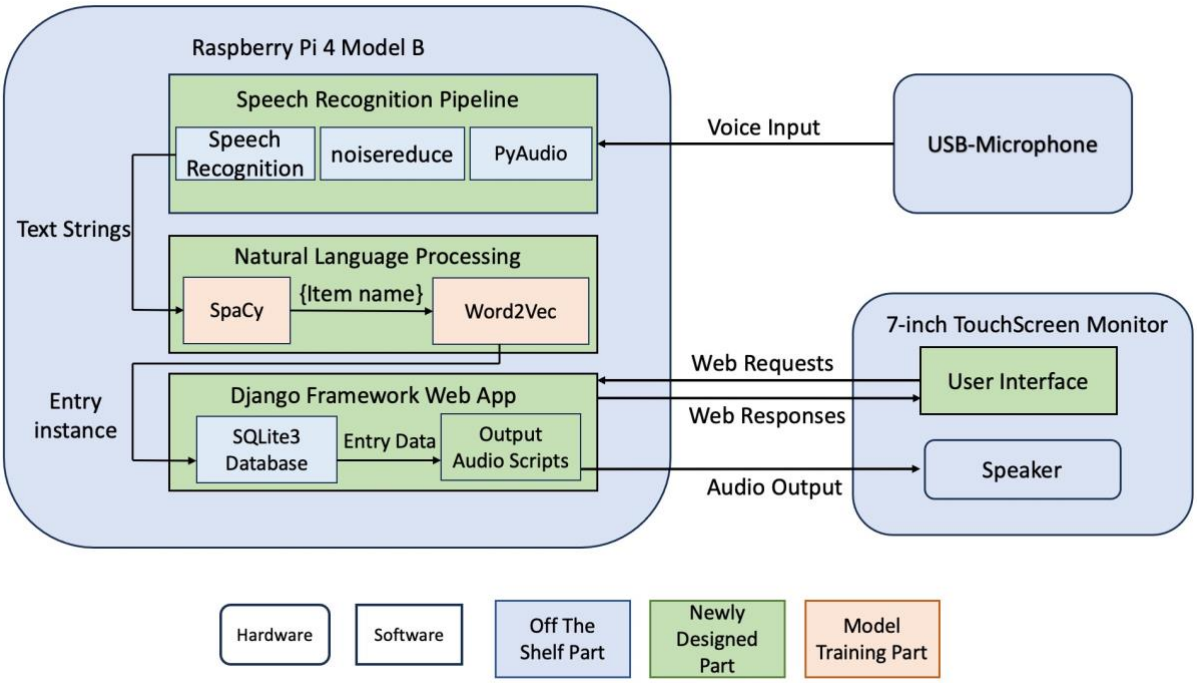


Fig. 3. Block diagram for the system.

IV. Design Requirements

Speech Recognition: According to Statista, the error rate of speech-to-text technology of Amazon, Microsoft, and Google Video is 18.42%, 16.51%, and 15.82%. Therefore, we expect our system to have less than a 20% error rate for the whole sentence. The important information for our system is the action verbs (e.g. generate the report, create, delete, etc.), item names, and prices. We would expect at least 90% accuracy for these important words to ensure our system works smoothly.

Voice Input Parsing: Our task for this part would be to extract the action verbs, item names, and prices from the whole sentence. The action verbs would be use to identify what would be the main task of our system. Item names and prices would be the primary information we need for each of the entry, and the item names would be the inputs for categorization. Since the information we get from this part would directly influence the whole system operation, we would expect the accuracy for this part to be at least 95%.

Item Classification: We would have six basic categories: food, housing, necessities, entertainment, transportation, and others. It would be burdensome if the auto-categorization fails too often, so we would ensure at least 90% accuracy. By MVP, we would not allow customers to add personalized categories. This is because the newly created category would have negligible training datasets, which would substantially decrease the accuracy of classification.

Battery Life: We would use the battery life of mobile phones to estimate the size of our power bank. The battery capacity of iPhone 15 Pro Max would be 4441 mAh. If it is active for 2 hours 30 minutes every day, its battery life would be 2 days and 21 hours. According to our use case requirements, we would expect our system to be active one hour every day and the battery life would be at least 24 hours. Therefore, we would want the battery capacity of our bank to be around 1200 mAh.

Active Interaction: We would want our system gives constant feedback to the users so that they would know the system is actually processing their request. For the voice input part, we would have voice notifications when button is pressed to inform the users that the following commands would be recorded. After the users release the button, our system would also have voice notification that indicates the recording ends.

UI Design: For every page of our system, we would divide the page into two parts. The left part would handle all the functionality of a money tracker app, including editing, displaying, and deleting the entries and showing the reports we generated. The right side would always be the button for voice input, which ensures that users could easily use voice commands on every page.

Financial Reports: We would generate two kinds of reports for a given periods. One would be the pie charts, which show how a user's spending is divided among different categories. The other would be a line graphs, which would track the changes of spending across different time periods. For example, if the user is requesting reports for last week, the line char would showing how much he or she spend every day in last week.

V. Design Trade Studies

Based on our use-case and design requirements, 5 crucial choices are made and the trade-offs are analyzed.

A. Hardware Selection

The hardware that carries the signal processing modules and web application is crucial for the general product. An embedded

system consisting of RPi 4 and a touchscreen monitor is preferred over laptops and smartphones in our design. The major factor that leads to this decision, other than a laptop, is portability. Customers should be able to use our systems in various scenarios, with or without the condition of using laptops. For example, one of the expected use cases would be inputting entries at grocery stores. Therefore, the hardware should be lightweight and portable to satisfy the use case requirements.

On the other hand, although a smartphone could be an appropriate device that guarantees portability, we decided not to choose it as the carrier hardware due to the target customer group's specific requirements. The target user group of our design is visually impaired people, and the majority of this group either do not have a phone or simply use phones with basic functions. As a result, a smartphone application fails to provide the service to a large group of our customers.

Based on the research and discussions, a portable embedded system implemented using Raspberry Pi and a monitor is finalized as the hardware selection for our design. The total weight of the design is about 10.6 ounces (2.29 ounces for RPi 4 + 8.3 ounces for the monitor), and customers can interact with the system via the touchscreen.

B. Speech Recognition

As the primary module of the signal processing system, the speech recognition pipeline should convert the voice input from the customers to text strings for NLP process in environments under 30 to 70 dB volume. Therefore, three Python libraries are used to construct the pipeline script and guarantee a high performance of the speech recognition module. PyAudio is selected to convert audio signals to streaming data because this library simplifies the audio-capturing process, making it ideal for microphone and speaker interfaces. Although there are several libraries that could handle noise reduction such as pydub and PyAudio, the noisereduce library is selected because it is designed specifically for identifying and reducing background noise in audio signals, which is crucial for the speech recognition process under high-volume environments. The library SpeechRecognition is chosen for the actual speech-to-text component in the pipeline because it could support multiple engines and APIs, which provides flexibility and versatility. Therefore, backup scripts implemented by different engines could be utilized when the primary engine fails.

C. Command Matching

For parsing the whole sentences, we are deciding between two models: SpaCy and NLTK. We choose to use SpaCy for the following reasons.

Firstly, spaCy provides out-of-the-box support for Named Entity Recognition (NER) which is more advanced and accurate compared to NLTK. This feature of spaCy can effortlessly identify and classify proper nouns, including product names and monetary values, directly from the text, which is essential for our use case.

Moreover, spaCy's processing pipeline is designed for production use and can handle large volumes of text rapidly, making it highly scalable and efficient for real-time applications. NLTK, while powerful for academic and research purposes, is not optimized for speed and may not perform as well in a production environment where quick processing is critical.

Another advantage of spaCy is its superior part-of-speech tagging and dependency parsing which are crucial for understanding the grammatical structure of sentences. This allows for a more accurate extraction of action verbs to the items and prices mentioned, providing a clearer understanding of the sentence's intent.

D. Item Categorization

In the item categorization process, our main goal is to assign one of the predefined categories to the item name obtained from the first half of the NLP pipeline. There are many text classification algorithms such as Random Forests (multiple Decision Trees), Logistic Regression, and Decision Trees. However, these algorithms are used to handle long texts with many features, while we only aim to parse single words or short phrases. Therefore, models that implement the above algorithms are not under consideration of this task.

Word vector representation is what we can make use of. It is a method of converting words into high-dimensional vectors so that similar words are close to each other in the vector space. The two most popular word vector representation methods are Word2Vec and GloVe, both of which can potentially be customized for our word classification task. Although Word2Vec and GloVe differ in how they train the model, they are similar in the result, so there is little difference for us using one or the other. For convenience, we adopt the pre-trained Google News Word2Vec model, which contains 300-dimensional vectors for 3 million words and phrases and is readily available for download and use through Python’s gensim library. GloVe, on the other hand, can provide an alternative if the performance of Word2Vec is not ideal.

E. Database

Our project is a local app with a single user and does not contain complex user data to be stored in the database. Therefore, we will be using the SQLite that comes with the Django framework. SQLite is easy to set up and stores data in a single file, making it handy to use for a standalone application like ours. Other databases supported by Django provide features, such as authentication, distributed systems, and duplication of databases, which are unnecessary to our project.

VI. System Implementation

The whole system of our project runs on a Raspberry Pi 4. A touchscreen monitor that displays the web application UI and a USB microphone for voice input is connected to the RPi.

A. Hardware Setup

There are three major hardware components in the design: an RPi 4, a 7-inch RPi touchscreen monitor with speaker, and a USB microphone. The PRi could be mounted on the back of the

touchscreen with 4 screws. A 3-pin GPIO cable is connected to RPi pin 2,4(5V power) via the red wires and pin 6(Ground) via the black wire to output power from the RPi to the monitor. Thus a separate power for the monitor is not required and portability is guaranteed. To enable the touch feature of the monitor, a microUSB to USB-A cable between the USB port of the RPi and the 5V-touch port on the monitor is connected. An FPC cable with HDMI connector is also needed between the RPi and the monitor to transmit audio and video signals. The device could be charged either through a receptacle or a portable charger. A 10000mAh 5V/3A power bank is chosen currently for portable charging. The USB microphone is connected to the RPi directly.

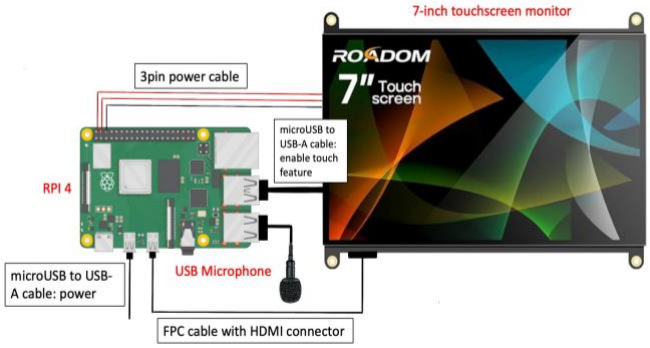


Fig. 4. RPi, touchscreen monitor, and USB-microphone connection

B. Speech Recognition

To convert the voice inputs given by the users to text strings that could be fed to NLP models, a speech recognition pipeline script is implemented using several Python libraries. An audio stream is initialized once a new recording session starts, and corresponding data is continuously written using the methods in PyAudio library. The recorded data is then passed on to the noise reduction module that is implemented via the noisereduce library.

After eliminating possible environmental noise, the modified data would be transferred to the speech recognition module built based on the SpeechRecognition library. This library supports different speech recognition engines and APIs that could work both offline and online. The output will be concatenated as strings that are sent to the NLP models once the conversion process is complete.

C. Text Parsing

The input of the NLP models would be the script from speech recognition. Our initial guide for customers to learn how to interact with our system would require them to use imperative sentences. Therefore, we would expect all the commands given by the customers to be imperative sentences.

Based on our requirements and the structure of imperative sentences, for most of the time, there should be only one verb in the sentence and that verb would be the key action users want our system to proceed. Therefore, we would load the existing spaCy model and use POS (Part-of-speech tagging) figures of spaCy to identify the verb.

It would be harder for us to extract item names and prices. The features we would use here is Named Entity Recognition. For this part, we would first trained our own spaCy model. For example, we would want our model to learn that commodities like “apples”, “keyboards”, and “tickets” would have label items and we would want things like “fifteen dollars” to have label prices.

D. Item Classification

After parsing the required components from the text, we use a pre-trained Word2Vec model, specifically Google News 300, for the item categorization process. The model is downloaded and customized with the gensim library in Python. An initial dataset is created mapping item names into categories. Given a new item name, our algorithm will calculate the similarity of the new item name and each of the item names in the dataset, obtaining a normalized similarity score for each category. The category with the highest score is assigned to the new item name. The user has chances to modify the category of a spending, and once confirmed, the dataset will be updated with the item name and the new category for future comparisons.

E. Web App UI & Text-to-Speech

The web application user interface is split into a GUI section and a VUI section that satisfies user requirements for different customer groups. HTML pages could be rendered via both voice commands and button clicking. Customers could submit new entries through the Entry form and a new instance of the Entry model will be created and saved to the database if confirmed. Similarly, entries could be modified by extracting the model instance data from the database and saving the updated instance back. To display the monthly entry lists and financial reports, the application would retrieve and filter Entry data based on the time field and feed the data to specific command handler functions.

The major component of the VUI section is a large button on all pages that enables and disables the voice input functionality. Customers interact with the web application through voice commands in this section, and the system will respond accordingly. After parsing the key parameters from the audio input and processing the command, the response information strings will fill the corresponding pre-generated message templates that are stored in static text files. The completed response texts are fed to a text-to-speech converter implemented using gTTS and the output audio will be delivered to the customer via the built-in speaker of the touchscreen monitor.

The GUI section is mainly constructed using HTML elements and JavaScript components. Customers could manually create or update entries through HTML forms and could view the entry lists and responsive financial reports that include a pie chart and a histogram.

F. Data Storage & Management

The database used in the design is the SQLite database in the Django framework and the key data that needs to be stored in the database is Entry. The Django model for Entry is defined in models.py and consists of an item name character field, a price

number decimal field, a category foreign key field, and a date field that indicates the creation time of the entry. Item name and price number are parsed from user input, while category and date are determined or assigned by scripts. Once a new entry is created, the instance will be stored in the database table for Entry model. To update specific fields of an existing Entry, a serial number that corresponds to the ID of the entry(which is assigned automatically by Django) should be entered and the entry could be found in the database. Modifiable fields for Entry model are item name, price number, and category.

When a financial report is requested, entries created in the given month are extracted via the date field. These entries are distributed into 6 arrays by categories, and the total price spent for each category is calculated. The statistics are fed to chart components and the complete report page is generated.

VII. Test, Verification and Validation

We will evaluate the functionality and performance of the system through both unit tests and integration tests. These tests will be conducted during the implementation or after the integration of all parts.

A. Tests for Manual Input Functionalities

The app is expected to function correctly as a normal money tracking tool supporting manual inputs. It should serve basic functions discussed in the Architecture and/or Principle of Operation section and behave as described in the flow chart (Figure 3). The tests include but are not limited to the following: the pages for entry input, expense list, and financial report contain all components as shown in the UI design (Figure 1); upon the submission of an entry input, the information is stored into the database and can be viewed in the expense list; users can modify or delete any spending entry in the expense list, and the change will be made correspondingly in the database and reflected in the expense list; the financial report calculates the total amount spent and the amount spent in each category for the selected time period correctly.

These basic functionalities will be tested immediately after their implementation so that later audio input functionalities can be built upon these basic functions.

B. Tests for Accuracy

Since there are several parts in our speech recognition and NLP pipeline, we will test their accuracy separately and as a whole. We will randomly create 20 pieces of speech for this test.

The speech-to-text conversion done by the SpeechRecognition library is expected to reach an accuracy of 90%. We will parse the text for the command verb, and potentially the item name and price if the command is to enter, modify, or delete an entry, with SpaCy library. The element extraction process is expected to reach an accuracy of 95% per element. If the command is to enter a new entry, we will then classify the item name into one of the categories, and this process should reach an accuracy of 90%.

For the audio command processing as an integral part, we are aiming to achieve an overall accuracy of 90%. That is, 9 out of 10 audio commands should be interpreted and executed without mistakes in order for the product to be useful and labor-saving for our users.

C. Tests for Latency

We will test the total time spent waiting for the system to parse the command. From the time the user stops recording to when the parsing result is reflected on the page, the elapsed time should be less than 3 seconds for 90% of the trials. We will run 10 trials for this test.

D. Tests for Noise Tolerance

Since our product should function well in noisy environments such as supermarkets and restaurants, we will repeat the test for accuracy with surrounding noise of 70dB. If we cannot find a place with a steady ambient noise of around 70dB, we will artificially play audio conversation to mimic the noise in real life. The test is expected to meet the same requirements in the test for accuracy.

E. Tests for Battery Life

We will power the battery with the power bank only and keep using the app for 1 hour to test if the battery can support the system for 1 hour with the monitor on. We will also shut down the monitor for sleep mode and see if the battery can last for 24 hours without charging.

F. Tests for User Experience

We will invite 5 volunteers to use our product without our excessive interference. After they thoroughly explore our product, they are invited to provide any feedback regarding the conciseness of the UI, the complexity level of their interaction with the app, and the overall experience. This will guide us to make potential improvements or adjustments before the final demonstration.

VIII. Project Management

A. Schedule

Please see the Gantt Chart (Figure 5) for our detailed schedule.

B. Team Member Responsibilities

The project is roughly divided into three parts and distributed to team members.

Lynn is responsible for UI design, graphic works, speech recognition, and text-to-speech conversion. Yixin researches and trains the SpaCy model for parsing commands and parameters from the text. Yuxuan develops the web application and customizes the Word2Vec model for word classification. Everyone in the team participates in hardware setup, component integrations, and testing process. The division of labor is subject to change if certain tasks require extra effort.

C. Bill of Materials and Budget

See Table I for a complete list of materials and costs.

D. Risk Mitigation Plans

Accuracy: Our NLP models might not be able to handle all variations of commands. If the accuracy test fails, we will

suggest our user to follow templates of commands that are guaranteed to be recognized by our models.

Latency: If it takes too long for our models to convert audio to commands (users wait over an average of 3 seconds for the results), then we will consider truncating the number of dimensions of the vectors in the Word2Vec model or using a GloVe model in place of the Word2Vec model in the classification process.

Accessibility: It might be hard for visually impaired people to press accurately on the button. To assist them in pressing on the recording button, we will notify the user upon the start and end of the recording with voice output of "start recording" or "end recording". We will also respond to a "help" command by telling the user that the recording button is on the right half of the screen. The "help" message will be displayed by audio upon each launch of our application.

IX. Related Work

There is currently no money tracking app with built-in voice control feature in the market. Apple does have comprehensive accessibility support on iPhone, iPad, and iPod. There are voice controls that help users interact with the screen, display accommodations for the visually impaired, and Siri to convert voice into very basic commands. However, the accessibility support cannot describe UI components for the user, and most apps do not support internal interaction with Siri. Hence, Apple's accessibility support does not provide a practical or efficient way to help the visually impaired, especially fully blind, group to interact with a money tracking app.

X. Summary

Our project is a money tracking application integrated on Raspberry Pi with innovative features of audio input and webpage reader. It's a highly interactive, easy-to-use, and inclusive money tracking app that helps visually impaired and elderly people log, review, and visualize their spendings with our powerful speech recognition and NLP models. The main challenge is to correctly parse elements from a text with NLP models, and our team will carefully create datasets and tune the models for satisfactory outputs. Our team is confident that this product provides a promising solution to catering the need for financial management and budget keeping of the visually impaired and elderly people.

Glossary of Acronyms

FPC - Flexible Printed Circuit
GPIO - General Purpose Input/Output
GUI - Graphical User Interface
HDMI - High-Definition Multimedia Interface
NER - Named Entity Recognition
NLP - Natural Language Processing
RPi - Raspberry Pi
UI - User Interface
VUI - Voice User Interface

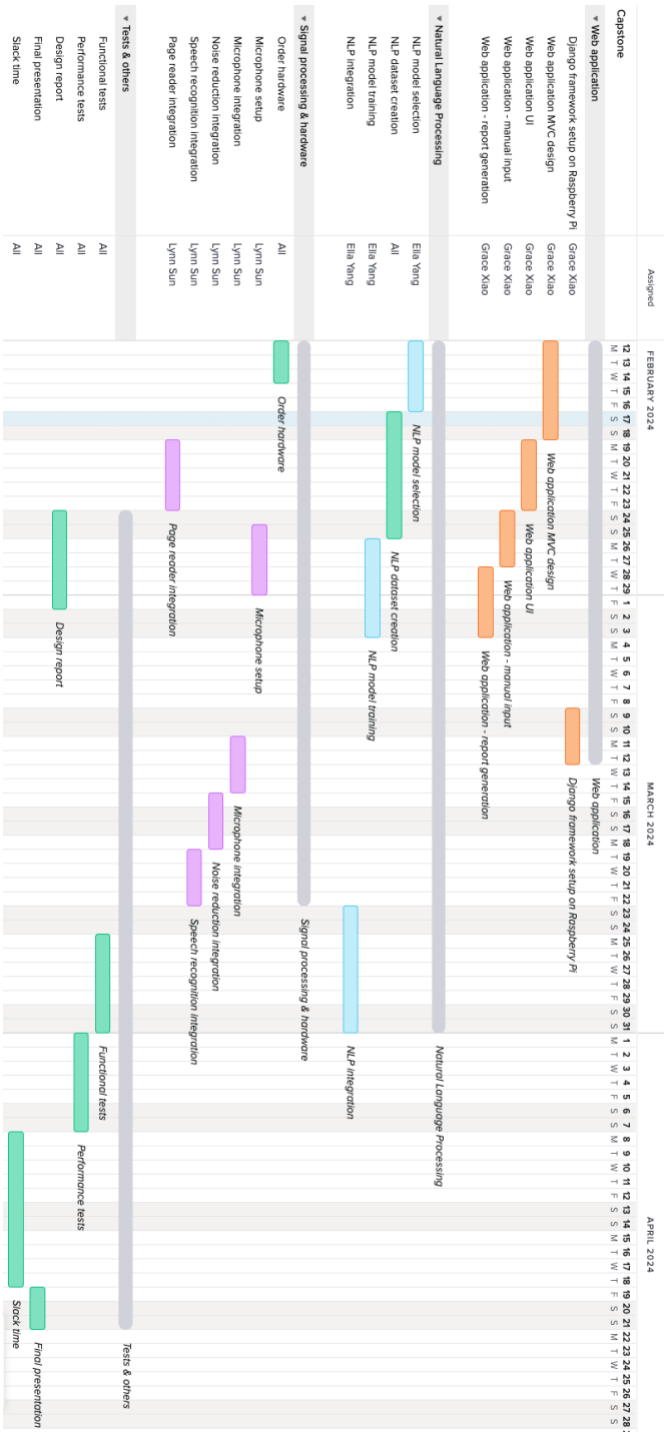


Fig. 5. Schedule Gantt Chart

References

[1] "Gensim: Topic Modelling for Humans." Modelsl.word2vec embeddings – genism, December 21, 2022[Online]. Available: https://radimrehurek.com/gensim/models/word2vec.html.

[2] Great Learning Team. "What Is Word Embedding: Word2vec: Glove." Great Learning Blog, February 23, 2024 [Online]. Available: https://www.mygreatlearning.com/blog/word-embedding/.

[3] S., Edward. "SQLite vs Mysql – What's the Difference." Hostinger Tutorials, December 21, 2022 [Online]. Available: https://www.hostinger.com/tutorials/sqlite-vs-mysql-whats-the-difference/.

[4] "Support." Official Apple Support. Accessed March 1, 2024 [Online]. Available: https://support.apple.com/accessibility.
[5] J., Lindner. "Statistics About The Average Arm Length." Gitnux, February 7, 2024 [Online]. Available: https://gitnux.org/average-arm-length/#:~:text=An%20average%20NBA%20player's%20arm,humans)%20is%20about%2082%20cm..
[6] What Noises Cause Hearing Loss?" Centers for Disease Control and Prevention, November 8, 2022 [Online]. Available: https://www.cdc.gov/nceh/hearing_loss/what_noises_cause_hearing_loss.html#print.
[7] E., Ryan. "Website Load Time Statistics: Why Speed Matters in 2024." WebsiteBuilderExpert, November 27, 2023 [Online]. Available: https://www.websitebuilderexpert.com/building-websites/website-load-time-statistics
[8] "Speech-to-Text transcript accuracy rate among leading companies worldwide in 2021." Statista, May 2021 [Online]. Available: https://www.statista.com/statistics/1133833/speech-to-text-transcript-accuracy-rate-among-leading-companies/.
[9] "Apple iPhone 15 Pro Max Battery Test." DXOMARK, December 8, 2023 [Online]. Available: https://www.dxomark.com/apple-iphone-15-pro-max-battery-test/
[10] Abraham CH, Boadi-Kusi B, Morny EKA, Agyekum P. Smartphone usage among people living with severe visual impairment and blindness. Assist Technol. 2022 Sep 3;34(5):611-618. doi: 10.1080/10400435.2021.1907485. Epub 2021 May 3. PMID: 33760680.

TABLE I. BILL OF MATERIALS

Description	**Model #**	**Manufacturer**	**Quantity**	**Unit Cost**	**Total**
Raspberry Pi 4 4GB	Model B	Raspberry Pi	1	ECE inventory	$0
Raspberry Pi Starter Kit	SD card, cables, etc.	CanaKit	1	ECE inventory	$0
7-inch Touch Screen	IPS HD 1024*600	NORSMIC	1	$69.99	$69.99
Microphone	AU-UL10 USB	MAONO	1	$22.99	$22.99
Portable Power Supply	10000mAh 5V/3A	INIU	1	$29.99	$29.99
				Grand Total	**$122.97**