Capstone Project Name

Authors: Caroline Crooks, Tahaseen Shaik, Sumayya Syeda
Affiliation: Electrical and Computer Engineering, Carnegie Mellon University

***Abstract*—TableCast is an interactive table projector designed to assist users in the kitchen via computer vision, voice commands, and gestures. Users will choose a recipe and follow instructional content projected on their kitchen countertop, including video demonstrations and user responsive feedback, calibrated to their specific countertop. We accomplish this by using high quality off-the-shelf peripheral devices, extensive computer vision algorithms, open source speech recognition engines, and web application development software.**

***Index Terms*—Computer Vision, Cooking, Gesture Recognition & Tracking, Homography, MediaPipe, Object Detection & Tracking, Projections, SIFT, Speech Recognition**

1 INTRODUCTION

TableCast is an interactive table projector designed to assist users in the kitchen via computer vision and voice commands. Users will be guided through recipes of their choosing via text and video steps without having to touch any sensitive devices.

Currently, most people watch cooking videos or read recipe articles online while they are cooking. There are a few problems with this. In many instances, fingers become messy from cooking. In order to look at the recipe again, you must constantly wash your hands or risk your device also becoming messy. This process is frustrating and unsanitary. Secondly, some people have trouble following the steps due to the differences between the instructor's tools/setup and one's own. There is no guide on how to cook something in one's own kitchen.

In order to create a intuitive and user-friendly interface for TableCast, we will be utilizing computer vision extensively and incorporating user interactions via voice commands and projected buttons. TableCast will initially calibrate to the user's kitchen space before beginning the cooking process. The recipes will be broken into steps and then displayed on the table in text and video format for users to follow. Users will have the option of using voice commands or projected buttons in order to go back a step, pause and play. Furthermore, TableCast will use object re-identification to track the ingredients as they are prepared for the cooking process in order to best direct the user. This design is meant to make cooking new dishes easier and avoid the issues of contamination of devices and overall inefficiency in the kitchen.

2 USE-CASE REQUIREMENTS

User-Interaction:

UR1. Physical interaction with the device or web-app must be limited to 4 unique interactions

UR1a. Start, stop, select recipe, change recipe

Motivation: We wanted a simple application where the user is not overwhelmed by a large quantity of controls. We decided that these four commands were the minimum requirement to complete a recipe.

UR2. User should be able to interact with device using projected buttons.

Motivation: We strive to achieve a system that minimizes physical interaction with any device. This is one method to do so.

UR3. User must have a seamless experience with the system with low latency.

Motivation: This is to ensure that the user can efficiently create recipes w.r.t. time.

Projection:

PR1. Perimeter of projection must be within 2 cm of the counter

Motivation: Ideally the projection should be flush to the countertop boundaries, but added a 2cm buffer as not all setups are always ideal. The 2cm is just enough to be aesthetically pleasing while getting maximum counter top area.

PR2. Section projections on counter must be within 0.75 meter of the users starting position

Motivation: We designed TableCast for a standard 3'x5' counter. Three fourths of a meter is about 2.5 feet which will mean most sections are within arms reach from the center of the table.

PR3. Sections should not overlap and should have at least 1 cm spacing between.

Motivation: This is to ensure a clean user friendly projection UI.

Device:

DR1. Voice command recognition must have an accuracy rate of at least 90%

Motivation: The voice commands should be reliable to use. The Whisper models have a high accuracy of around 95%, so a 90% accuracy considering the environment should be reasonable.

DR2. Survey of user experience must have at least 85% satisfaction rate

Motivation: This satisfaction rate is reasonable because it means that most users are overall satisfied, but it also accounts for variations in people's individual preferences in cooking.

3 ARCHITECTURE AND/OR PRINCIPLE OF OPERATION

3.1 Architecture Overview

The system components comprise of digital hardware components, software components, and mounting mechanisms. Our processing unit is the Legion Pro 7i Gen 8 PC, which will send and receive information to the peripheral hardware devices. A camera and microphone provide input data to the computer. A projector and stereo speaker are output devices. An external PC is both an input and output device that will communicate with a web server. Within the main PC, primary software tasks comprise of computer vision processing, voice recognition, the web server, user interface control, and overall system command processing. Mounting devices are tripods to hold the camera and projector, along with supportive casings to provide additional protection. One tripod will support the camera, while another tripod will support the projector. These components are further described in the system architecture (Figure 6). We are keeping version control using Atlassian Suite's BitBucket. This enables us to consistently track and merge our progress to make system integration easier.

3.1.1 Design Changes

Figure 1: Design Change: Projector Placement

Since the design report, we made significant changes to the hardware structure of the system. In the original design, both the camera tripod and the project tripod were placed across from the user ((Figure 1), left). This orientation cast shadow from the user's hands, ingredients and any other object on the table over much of the UI. To mitigate this issue, we moved the projector tripod to the right side of the user ((Figure 1), right). This way the shadow is cast to the left of the user and never over the area the user is currently working with.

Another significant change we made was the use of a PC instead of the AGX to host the entire system. We had difficulties finding compatible libraries for the voice commands and purchasing the correct connector for the camera module. Specifically, there were compatibility issues with the library version of Whisper and the version of the CUDA on the AGX. The compatibility issues were between the CUDA version 11.4, which came pre-installed with the JetPack), and several versions of the Whisper voice module. Functional voice command code that had been originally developed on a laptop needed to be transferred to the AGX, but it repeatedly failed to run successfully on the AGX. After trying multiple methods to resolve this issue, we decided that it was more time efficient to use the PC.

3.1.2 Hardware Components

The camera is be placed above the center of the table. It must have a full view of the entire table. Real time data will be sent from the camera to the main PC's computer vision processing unit. The camera will be turned on throughout the duration of the recipe, from the calibration step to the completion of the recipe.

The microphone is a wireless, lavalier microphone that a user will attach to their clothing. The user will turn it on after they pick the recipe and may turn it off after the recipe is complete. The microphone will send audio data to the voice command software module to be processed.

The projector displays the user interface onto the table. It will receive image data from the projection module.

The stereo speaker will sound an alarm when a timer counts down to zero.

The user PC communicates with the web server on the main PC. This PC will be a user supplied device that will enable the user to browse recipes and start to cook.

3.1.3 Software Components

The vision module receives the video feed from the camera. It will use MediaPipe and OpenCV as computer vision libraries to do object detection, tracking, and gesture recognition. Ingredients on the table will be monitored in real time, and user gestures will be interpreted as commands to the program.

The voice command module receives data from the microphone. Speech is interpreted with speech-to-text tools within the module to interpret commands.

The projection module receives data from other modules to create a responsive user interface. The interface includes recipe video controls, timer tools, buttons, and ingredient placement guidelines. After an image is created, the image is then warped according to a computed

homography. This warped image is sent to the projector.

The command processing module is the central control system for the overall software system and coordinate the modules. For example, when a user selects a recipe, the web server will pass information to the command processing module, which will initiate the other modules.

The web server starts upon powering on the main PC. It hosts the web application that communicates with the user's PC. The user will browse and select recipes on the application.

3.1.4 Mounting Components

There are two tripods. One tripod supports the projector and holds it in place so that it is secure at an angle. Another tripod has an extending arm that protrudes over the table. This will support the camera that is facing down at the table. Both of these tripods are height adjustable, as required by the design requirements.

3.2 Principle of Operation

3.2.1 Installation

The user will place the camera tripod in front of their kitchen countertop, across the side of the table where they will be cooking. They will place the projector tripod next to the kitchen countertop, to the right of the side of the table they will be cooking. The camera should be angled down, facing the countertop. The projector should be angled down towards the countertop.

3.2.2 Solution Flow

Set Up. The user will turn on the main PC. Upon boot, a script will automatically set up a network interface and a web server. The user will connect their own personal laptop to the network, named TableCast_Device, which will be broadcast from the main PC once it is. This will be a Peer-to-Peer (P2P) connection between the laptop and the main PC. The user will then navigate to the TableCast web application running on the web server. On the application, users can scroll through recipe options and select one to cook with TableCast. The user will then attach the microphone to their clothing.

Recipe Task. A flow diagram is displayed in (Figure 7). With the recipe selected, TableCast will guide users through a calibration step. The computer vision and projection modules will be used to complete this step. Once calibrated, the device will project the user interface on the table. Users will place each ingredient in a box with the corresponding label. With all the ingredients detected, the user can start the recipe. The user will follow a video projected on the table, using gestures or voice commands to aid them while cooking. This will repeat for each step until the recipe is complete.

Shut Down. The user can stop the program at any time. If the user has paused the program, they will then be given the option to exit the recipe or shut down the device.

3.2.3 Principle of Engineering

TableCast was built using a bottom up approach. Specifically, we implemented and tested the functionality of each feature such as the computer vision module and the voice command module before integrating the final system.

The computer vision module incorporates MediaPipe and SIFT for gesture and object detection respectively. MediaPipe is a lightweight ML model created by Meta that has models for a variety of application. In TableCast, we specifically use it for gesture recognition and hand tracking. We use the "Open Palm" gesture trained by the MP Hand Gesture Classifier for button presses. MediaPipe also has a model that can recognize Hand Landmarks. We use the Hand Landmarker model to track hand position and detect a swipe left/right gesture.

3.2.4 Principle of Science

Science was a very important factor in the development of TableCast and for when we were determining suitable trade-offs. One of our tradeoffs was a shift in TableCast's projection methodology, steering us from the initial plan of projecting from behind the table to projecting from its side. By projecting from behind, the dispersion of light and the short throw of the projector risked compromising image quality and legibility. Recognizing this challenge, we opted to project from the side, thereby minimizing the distance traveled through the countertop and reducing the potential for dispersion-induced distortions. This adjustment not only mitigates the adverse effects of light dispersion but also enhances the clarity and fidelity of projected content onto the countertop surface.

3.2.5 Principle of Mathematics

TableCast relies extensively on mathematics in order to successfully display the user interface. During the calibration stage of the solution flow, the table coordinates are extracted from the camera space by tracking red dots and mapped to coordinates from the camera space related to a calibration shape. Using the `cv2` library's function `findHomography()` we are able to make a source to destination mapping. The resulting homography can be inverted and normalized before being applied to any other points - such as table coordinates.

4 DESIGN REQUIREMENTS

IR1. Projector mount must be adjustable.
- This relates to UR2 as the projector must be mounted properly to have projected buttons.

Motivation: The projector should be adjustable so that TableCast can be used on kitchen countertops with varying heights.

IR2. Warped projection on table looks flattened.
- Ensures PR1, PR2 and PR3.

Motivation: The projection should look flattened so that the user can easily interact with the UI.

IR3. Voice commands must recognize wake word to complete any command with an accuracy of 90%.
- Ensures DR1.

Motivation: Similar to the motivation in DR1, an accuracy of 90% is reasonable with the Whisper library.

IR4. Camera must be mounted such that it points down and is 3-5 feet above the counter.
- Ensures PR1, PR2, PR3.

Motivation: To meet all the proper projection requirements, the camera must be mounted directly above the table.

IR5. Gesture recognition must have 90% accuracy.
- Ensures DR3.

Motivation: Gestures should be reliable to use, while still accounting for variations in accuracy due to different environment factors like lighting.

IR6. The latency of the video instruction must be within 3-5 seconds.
- Ensures UR3.

Motivation: The UI should be responsive to the user in a reasonable time frame.

IR7. The gesture and voice commands must be recognized within 3 seconds.
- Ensures UR3.

Motivation: The system should process commands in a reasonable time frame to ensure that the result will be output to the user within a few seconds.

5 DESIGN TRADE STUDIES

5.1 Main Processor

The two options for main processors were the Jetson AGX Xavier and the Legion Pro 7i Gen 8. In the initial design, the AGX was selected because the goal of the project was to develop an embedded solution that would more closely resemble a product that a consumer would receive. However, due to version compatibility issues and time constraints, the main processor changed to the Legion laptop. This issue has been explained in further detail in the Design Changes section. After experiencing compatibility issues with the AGX, both computers were evaluated to ensure that switching over to the Legion would not mean we would lack the necessary processing power. With the comparisons presented in the table, it was concluded that the Legion would also be sufficient for this project. Figure 2 details the factors considered when switching to the Legion.

Processors

	Jetson AGX Xavier (32 GB)	Legion Pro 7i Gen 8
CPU	8-core NVIDIA Carmel Arm®v8.2 64-bit CPU 8MB L2 + 4MB L3 (2.2 GHz)	13th Generation Intel® Core™ i9-13900HX Processor (E-Core Max 3.90 GHz, P-Core Max 5.40 GHz with Turbo Boost, 24 Cores, 32 Threads, 36 MB Cache)
GPU	512-core NVIDIA Volta architecture GPU with 64 Tensor Cores (max 1377 MHz)	NVIDIA® GeForce RTX™ 4090 16GB GDDR6 (1335 MHz)
Ease of Use	1	5
Cost	$0 (borrowed)	$0 (owned)

Figure 2: Main Processor Trade Study

5.2 Gesture Recognition Libraries

Gestures are used as inputs to execute system commands such as start, stop, next and previous. The categories that mattered the most were computing power, latency, and precision. These categories ensure that the system is efficient, fast and precise which allows for a more satisfactory user experience. To score these categories, we used a study[1] that compared the two models with metrics. It was clear that although OpenPose was more robust, MediaPipe would have the specs required for our system while remaining light weight and reliable.

Gesture Recognition

	MediaPipe	OpenPose
Computing Power	5	3
Latency	5	3.5
Precision	4	5
Total	14	11.5

Figure 3: Gesture Recognition Trade Study

5.3 Projector

The projector is one of the cornerstones of our project. Without a high quality projector, it is impossible to meet our project requirements. With our limited budget, it was very difficult to find a satisfactory projector, but we were able to narrow down the decision to be between 2 projectors: the VANKYO Performance V700W and the ViewSonic 4500 Lumens SVGA Projector. In order to choose the most compatible projector, we considered 4 different factors: ANSI Lumens, projection size, projector size and cost. Between the two options, we foudn that the ViewSonic projector had the highest amount of ANSI Lumens (4500) which would be the minimal amount needed to work well in relatively bright environments. The promised projection size was higher by a little bit for the VANYKO, but ultimately it's large form and low ANSI Lumens pushed us to selct the ViewSonic projector instead.

Projector

	ViewSonic	VANYKO
ANSI Lumens	5	3
Projection Size	3	4
Projector Size	4	2
Cost	4	4
Total	16	13

Figure 4: Projector Recognition Trade Study

5.4 Speech Recognition Engines

The software is primarily being programmed in Python, and the SpeechRecognition Python library is well documented. The goal was also to use a free, open source engine. The library simplifies accessing and using several speech recognition models. There are 13 engine options for speech recognition. Out of those 13 options, only 4 options function offline. The system will not be connected to the Internet, so this is a necessary feature. These four options are CMU Sphinx, Snowboy Hotword Detection, Vosk API, and OpenAPI Whisper. Snowboy is a hotword detector, for detecting phrases like "OK Google" or "Alexa", so this engine will not be considered, out of a preference for longer phrase recognition.

To compare accuracy rates, we found a research paper that compared Sphinx, Vosk, Whisper, and two other engines [2]. A graph from this paper is featured in Figure 5. The Whisper engine had the lowest error rate, followed by Vosk, then Sphinx. Another part of the paper researched the execution time of each engine. Vosk was the fastest, followed by Whisper, then Sphinx. For our system, the accuracy of the voice commands is a slightly higher priority than the speed, which was put into consideration when selecting the engine.

Speech Recognition Engines

	OpenAI Whisper	Vosk API	CMU Sphinx
Accuracy	5	3	2
Speed	4	5	2
Ease of Use	5	5	5
Total	14	13	9

Figure 5: Speech Recognition Engine Comparison Table

6 SYSTEM IMPLEMENTATION

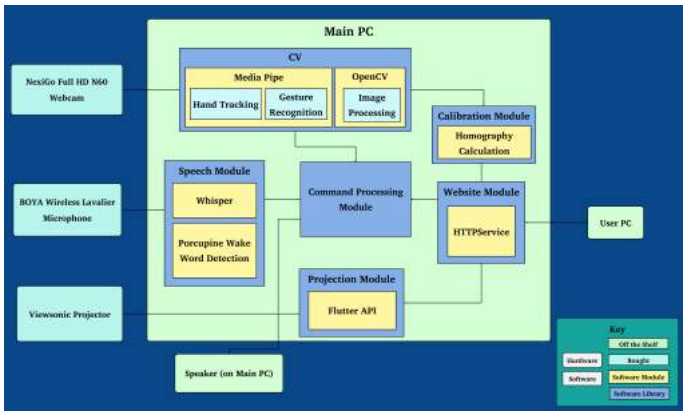

Figure 6: System Block Diagram

6.1 Detailed Components

6.1.1 Software Components

- OpenCV
- MediaPipe
- Porcupine Wake Word
- SpeechRecognition Python library
- OpenAI Whisper
- Python SimpleHTTPServer
- Flutter
- Flask
- ZeroMQ
- websockets

6.1.2 Hardware Components

- NexiGo Full HD N60 Webcam
- Viewsonic Projector
- Legion Pro 7i Gen 8 PC
- Electrical connections
- Laptop
- BOYA Wireless Lavalier Microphone

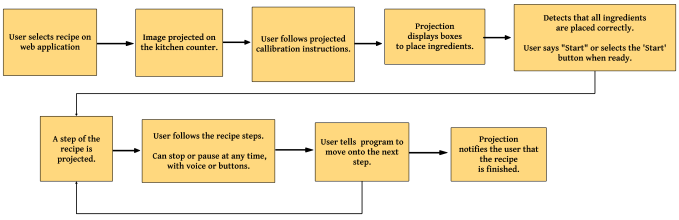

Figure 7: Solution Flow

6.2 Website Module

6.2.1 Network Interface

The PC will broadcast a peer-to-peer connection. Any user can connect to the network by selecting the SSID 'TableCast' with the password 'TableCast'. They will be server web content on port 8000 of the PC.

6.2.2 Web Application

The web application will be developed using Flutter because it is simple to use and lightweight. User actions will include:

- Launching calibration mode.
- Scroll through a list of recipes.
- Click a recipe to read more details about the recipe.
- Click the 'start' button to select the recipe to cook.
- Click the 'pause' button to pause the currently running recipe.
- Click the exit button to exit the recipe.

When a user opens the website, they will be given the option to either use an 'old calibration' or a 'new calibration'. If the user already completed calibration for their current set up, they can use 'old calibration', which will load the homography saved in a file on the system. If it is a new setup, the user should select 'new calibration' to begin calibration mode, which instructs users on how to calibrate their environment step by step.

After the calibration step, users can scroll through recipes, view pictures, preview instructions, and read information about the ingredients and cooking supplies required. Once the user is ready, they will press 'Start' to begin the recipe. Users will then be instructed to follow instructions projected onto the table.

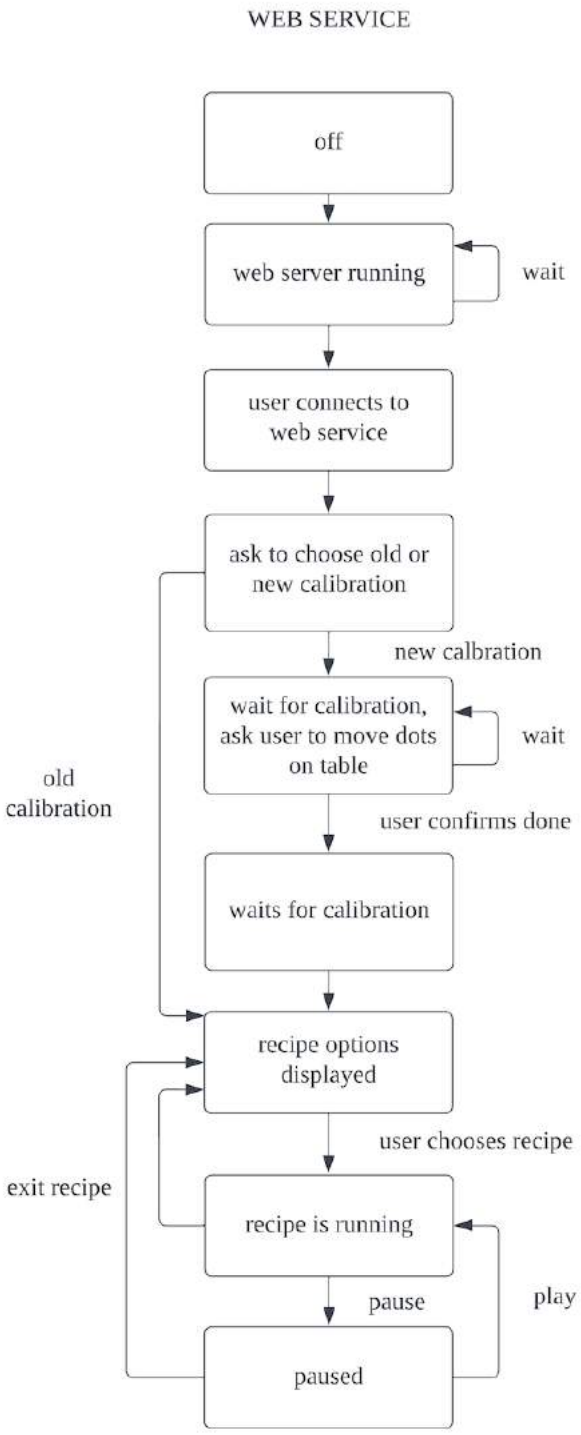

Figure 8: Web Application State Diagram

6.2.3 Debugging/Challenges

One particular challenge was deciding where content for the recipes would be stored and what manner they would be stored in. During the first iteration of the application, it was a hard coded a sample where all the recipes were the same brownie recipe. Later on, the goal to make loading data dynamic with different recipes. One design decision was considering using a database to store the files. Considering the factors of the team not having database experience, low amount of data, and the fact that the web application is not a major component of the project, it was decided that storing json files locally and loading them within the web application was the best decision for this project.

6.3 Voice Commands

6.3.1 Hardware

The microphone (BOYA Wireless Lavalier Microphone) must be powered on and clipped onto the user. The receiving device has a USB C plug in that will be connected to the PC.

6.3.2 Software

There will be two libraries used to process speech recognition: the Picovoice Porcupine Wake Word and the Python library SpeechRecognition with OpenAI Whisper. A free developer Picovoice account is used to create the recognition model for the wake word. The tiny.en model of Whisper is employed in this program. This way, when the user says "TableCast [command]", the program will quickly detect an instance of a command, avoiding accidental command recognition during a normal conversation. Through testing, this method has been found to be more reliable than just transcribing speech with Whisper while searching for the wake word. Once the Wake Word is detected, Whisper's transcription of the phrase after the wake word is searched for a valid command. In this case, the Central Processing module is notified, which will direct the other modules to carry out the corresponding tasks.

6.3.3 Voice Commands

- "TableCast pause video" - pauses the recipe video. Do nothing if already paused.
- "TableCast play video" - plays video the recipe video. Do nothing if already playing.
- "TableCast stop timer" - pauses the timer. Do nothing if already paused.
- "TableCast start timer" - resumes the timer. Do nothing if already running.
- "TableCast next step" - switch to the next step of the recipe.
- "TableCast previous step" - switch to the previous step of the recipe.

6.3.4 Debugging/Challenges

One major change that happened after testing the integrated system was changing the voice commands so that they utilized CUDA. There were latency issues when testing the project during interim demo week, so utilizing

the GPU more was a potential solution. The Whisper library had a built-in option to use CUDA instead of the default CPU setting, so this was tested. It was qualitatively much more noticable that there was no more lag, so this new implementation was used for the rest of the project.

6.4 Back-end

The system's overall central command processing unit is a Python script, titled central.py. This controls the sequence of launching the web service, showing the projected interface, waiting for calibration, starting gesture and voice commands, and coordinating between modules during the recipe runtime. Different modules are spawned as either a separate threads or processes.

6.4.1 System Organization

The system is organized on the PC such that all files are contained within the Capstone folder, and each module has its own folder. The files within the Scripts folder handle system communication, such as websockets for the web application and the projected UI. The file used to start TableCast is central.py. This file sets up the pub-/sub system, executes modules using multiprocessing, and manages the overall user flow from beginning to end.

```
Capstone
└── Root
    ├── Calibration
    ├── CV
    ├── Projection
    ├── Data
    ├── Scripts
    │   ├── central.py
    │   ├── ui_socket.py
    │   ├── web_app_server.py
    │   └── web_app_socket.py
    ├── Voice
    └── Web
```

6.4.2 Inter-Process Communication

To manage the states between different modules, the library ZeroMQ is utilized to implement a pub/sub model. Each topic is the name of the destination module, while the message is a command for that module. For example, if the voice command module detects 'next step', the voice command will publish to the topic=projector_ui, with the message command: next_step. Meanwhile, the projector interface that is subscribed to that topic will receive this data and call its own functions to transition to the next step.

Topics

- web_app
- projector_ui
- calibration
- central
- gesture

6.4.3 Debugging/Challenges

When developing communication between the modules, queues were the original method to do so. When debugging, the limitations of simple queues were made apparent. It was hard to synchronize them between modules that were very interconnected. Instead, a plan B was using a different method for communication, using a pub/sub model. ZeroMQ was simple to use as a library, but much debugging was needed to use in this project. For example, the default pub/sub pattern (PUB, SUB) was using only one publisher to a single endpoint. Reading documentation to learn about alternative methods (XPUB, XSUB) to allow for multiple publishers to connect.

6.5 Computer Vision

The Computer Vision subsystem consisted of gesture recognition and object identification.

6.5.1 Gesture Recognition

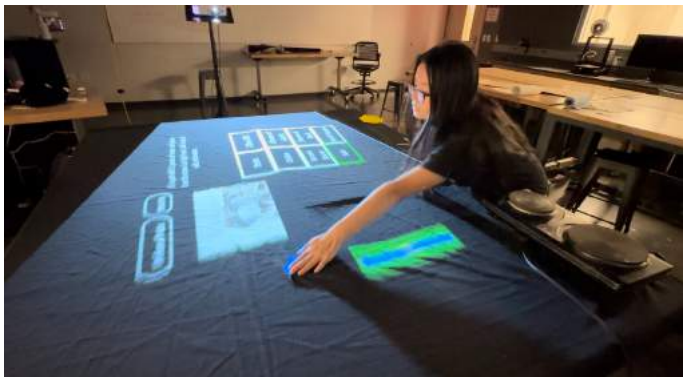

Figure 9: User Executing Button Press

We use Google's ML platform called MediaPipe to recognize hand gestures and track hand movement. The gestures are recognized based on the MP Hand Gesture Classifier that consists of 10 unique hand gestures we could choose from. For TableCast, we only use the "Open Palm" gesture to recognize a button press. We track hand movement to recognize a swipe gesture. Specifically, MediaPipe has a HandLandmark detection model called Hand landmarker that can identify 21 unique points on a hand and collect data on each points x,y,z location. We gather each points location over a select number of frames to determine the direction in which the user's hand moved. This mimics a "swipe" gesture, useful to indicate to go to the previous or next step in the recipe.

Something that didn't work with gesture recognition was the height of the camera. Our camera is set to be 5 feet from the table. This was too far for gesture

recognition and hand landmark model as the hand was too small in the camera frame. We could not lower the camera mount to resolve the issue.

6.5.2 Object Identification

The purpose of object identification in this application is to alert the user if they have picked up the wrong ingredient or to remind them to grab an ingredient if they haven't already. To do this, it was important to know where the grid was location in the image frame.

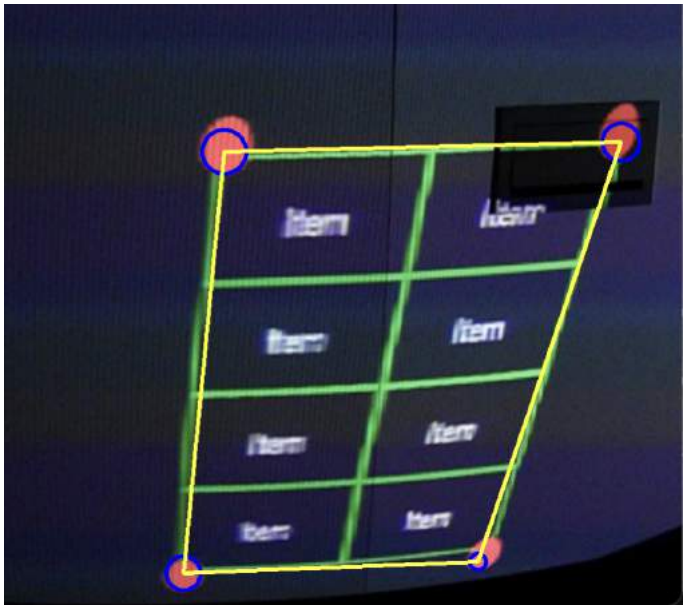

Figure 10: Grid Detection based on grid corner detection

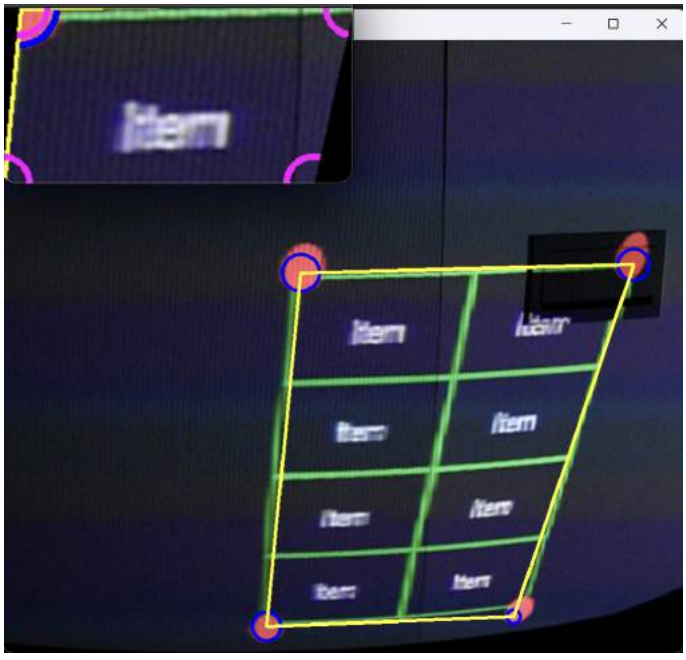

Figure 11: Grid cropped to specified cell (1, 1)

This seemed simple at first as the grid had well defined edges and the same number of cells throughout the recipe making process. But due to the warping of the projector, the grid was slightly skewed and had more of a trapezoidal shape. It was difficult to use to my original method where we looked for straight lines and 90 degree corners.

When testing with the projection image, we found that it was even harder to detect the lines of the grid due to poor lighting conditions.As a result, we changed our method for grid identification by adding red calibration dots to each corner of the grid (Figure 9). We first filtered the image to only see the red dots. Then it was much easier to find the circles in the frame using OpenCV's Hough Transform function. This method also solved our skewed grid problem as we could determine the slope of grid's boundary lines using the location of the four dots. Given the slope and that each cell height was 1/4 the height of the entire grid, and each cell width was 1/4 the width of the grid, we were able to compute the location of each cell's four corners and use this information to crop the cell from the grid (Figure 10). Now we can check just the cell and see if an object is present. If the wrong object is missing from the grid, an alert message is displayed. If the object is present in its cell when it should not be, a reminder message is displayed to inform the user they should grab that item.

6.6 Calibration

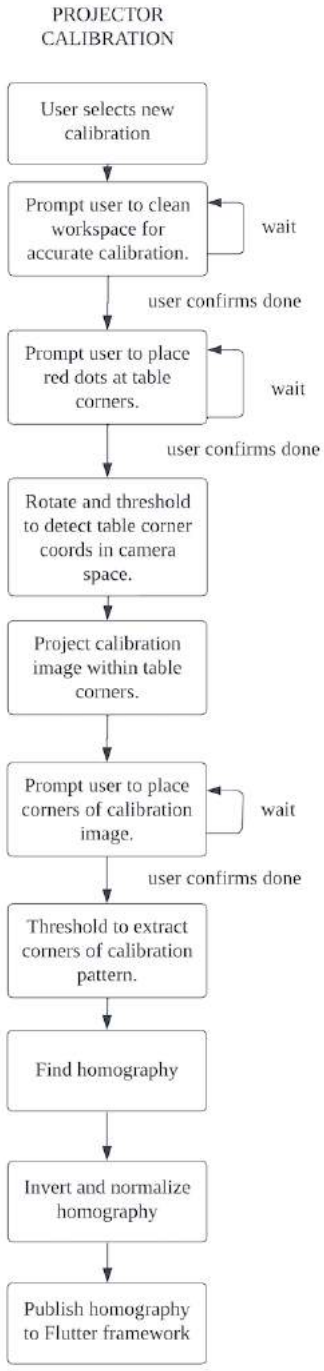

Figure 12: Calibration State Diagram

This pipeline for calibration ensures that the system can identify the appropriate homography to scale and transform the dynamic UI for projection. Additionally, we reduced the amount of user interaction needed to complete the calibration by choosing simple calibration images, as depicted in Figure 13. OpenCV and numpy were used to compute all of the homography calculations and transformations. When implementing the system, we also stored an image to identify the table outline for debugging purposes. If needed, the user can display the projected table outline image onto the table to clarify where the system perceived the table space to be. To clarify, the stove and the area behind the stove is not to be included in the table outline since they are not meant to be monitored by the camera system.

Figure 13: (left) Square Calibration image. (right) Rectangle Calibration image.

The rectangular image was ultimately chosen after running several trials of it on different tables because it resulted in the lowest norm of difference in the images which was the deciding factor. As Figure 18, in the verification section, shows the difference was not statistically significant, but we opted for the rectangular image anyway for the marginal improvement.

6.7 Projection Module

Flutter is used for the projected user interface that the user will interact with on the table. The dynamic content is warped before being displayed on the table

The interface is divided into three sections corresponding to the layout of the table.

- **Burner Area**. There will be no content displayed above the burner due to user safety concerns. Users will not be directed to reach above or across the burner.
- **Workspace Area**. The user is allocated space on the table to cook. Above that, video content, buttons for video control, and a timer.
- **Ingredient Area**. Users place their ingredients in the grid section.

6.7.1 Interactions and Backend

The Flutter application will be launched when it received a command from the Command Processing module to begin the application. Flask will be used to handle data to send to the Flutter application. Components controlled by user input are:

- **Video Controls**. The video can be paused or replayed, either by gesture (tapping on the button) or voice command.

- **Timer Controls**. The user will be suggested to use timers at certain steps. This timer will be displayed in as a widget. The user can pause the timer and resume it with voice commands.
- **Ingredient Guidance**. After the calibration stage, each square in the grid will be labeled with the name of the ingredient. During each step, grids that contain ingredients required at that step will be outlined in green. When the item is correctly chosen, the outline of the corresponding box will change from green to white to indicate that the the user has completed that task.

6.7.2 Navigation

The projected interface has three stages: displaying the logo, the calibration screen, and the recipe screen.

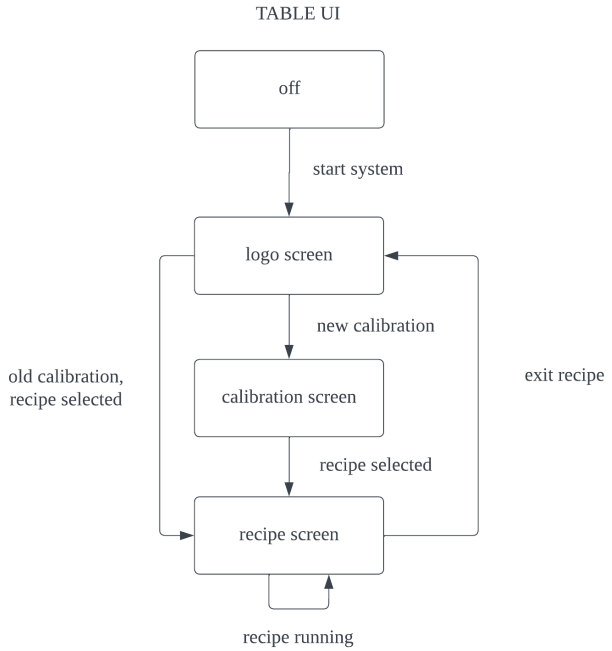

Figure 14: Table UI State Diagram

6.7.3 Debugging/Challenges

Using different widgets from Flutter presented a challenge in the early stages of development. There was difficulty playing videos and embedding timers due to poor documentation, but trying different types of widgets, such as different video player widgets, led to success in these areas. Another part of the debugging process was iteratively making slight changes to the design in order to help integrate with the CV module and calibration module. Such changes include making bigger item boxes, moving the buttons, and adding a swipe bar. Figure 16 shows the UI design planned in the very early stages of the project, while 15 is the final projected result.

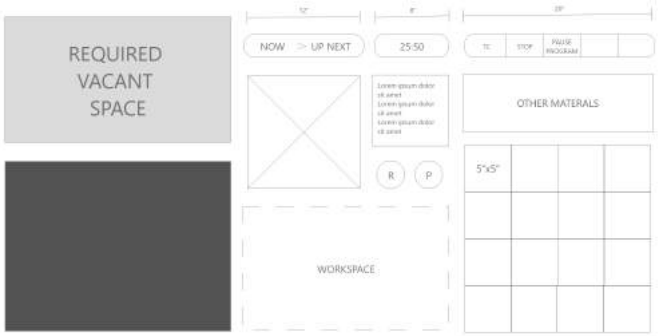

Figure 15: Original UI Wireframe

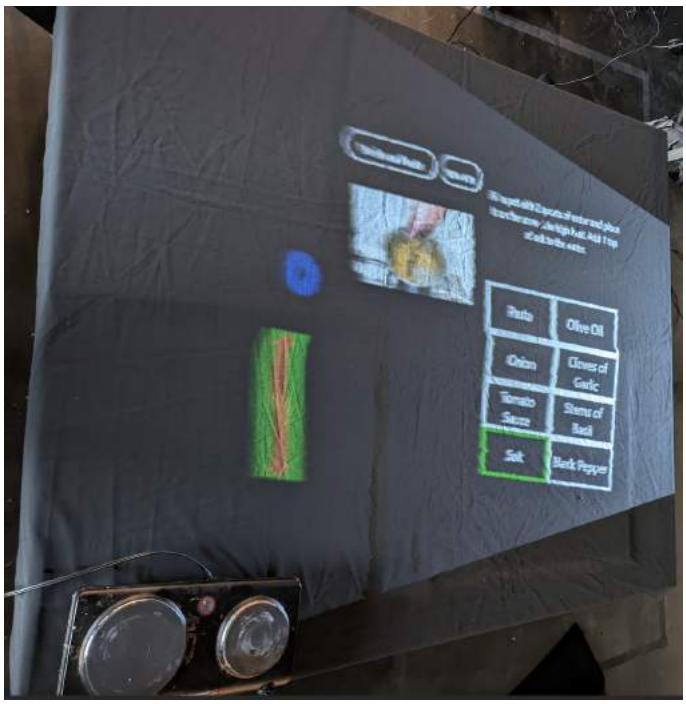

Figure 16: Final layout

6.7.4 Data Storage

Data that is preinstalled on the device are recipe videos and their corresponding instructional text. This data will be stored in an assets folder that has a subfolder for each recipe. A json file will contain the written instructions, information about ingredient locations, and ingredients used at each step. The Flutter application will obtain recipe information from this file.

7 TEST, VERIFICATION, & VALIDATION

7.1 Results for Design Specification IR1

IR1: Projector mount must be adjustable

The projector mount was height and angle adjustable as required. This enabled the user to adjust the height of the projector between 36 inches to 55 inches and the angle from 10° to 90° with the projector still secure. The sure can now roughly fir the projection onto the table and TableCast calibration can handle the specifics of the warp.

7.2 Results for Design Specification IR2

IR2: Warped projection on table looks flattened

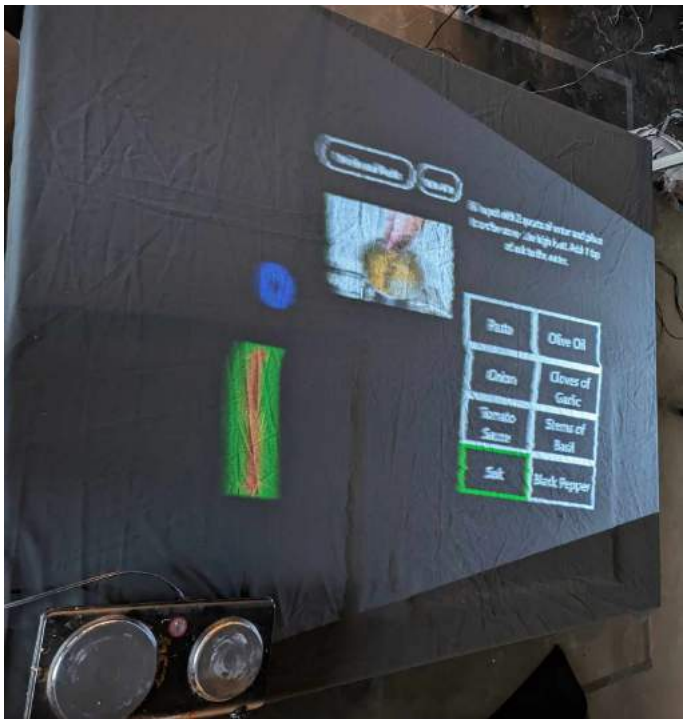

Figure 17: Flattened Projection UI

This requirement was validated by running the norm of the difference between the 2 images after filtering out noise in the projector image, as shown in Figure 18. These results indicate that the rectangular calibration image was marginally superior in terms of appearing flattened, so it was the preferred image. The difference between the 2 images came down to their ability to fill the table space required while having all components maintain a rectangular appearance. It is worthy of note that the gesture buttons on the far left of the image in Figure 17 were the most difficult to straighten because of the effect of light dispersion at such a far range from the projector.

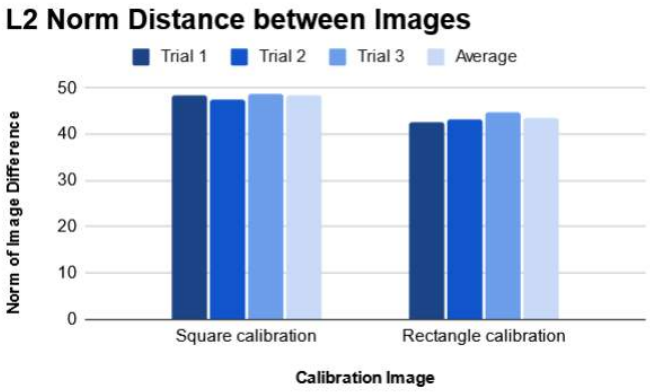

Figure 18: Quantitative comparison of calibration images.

7.3 Results for Design Specification IR3

IR3: Voice commands must recognize wake word to complete any command with an accuracy of 90%

Most commands were able to meet the 90% accuracy requirement, except for the ̀start timeràndstop timercòmmands.

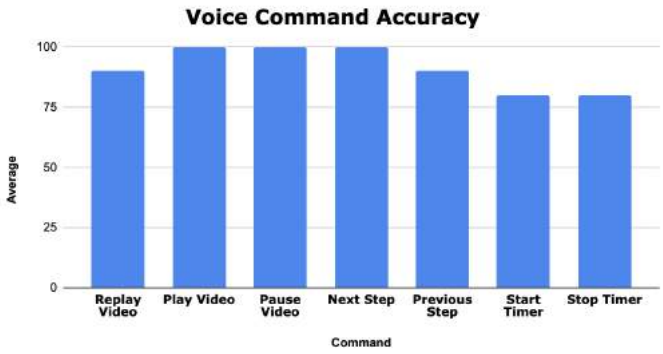

Figure 19: Voice Command Accuracy

Each voice command was said a total of 10 times in a random order that was determined before the test. This order is random, but commands like T́ableCast, play videoẃere not said twice in a row because there would be no clear difference in output. These commands were tested with the entire system running, rather than isolated for authenticity. These results do make sense. The voice commands were often correct, but differences in a users speaking pattern or volume can cause inaccuracies.

7.4 Results for Design Specification IR4

IR4: Camera must be mounted such that it points down and is 3-5 feet above the counter.

The camera mount meets this specification at 4'10" above the table.

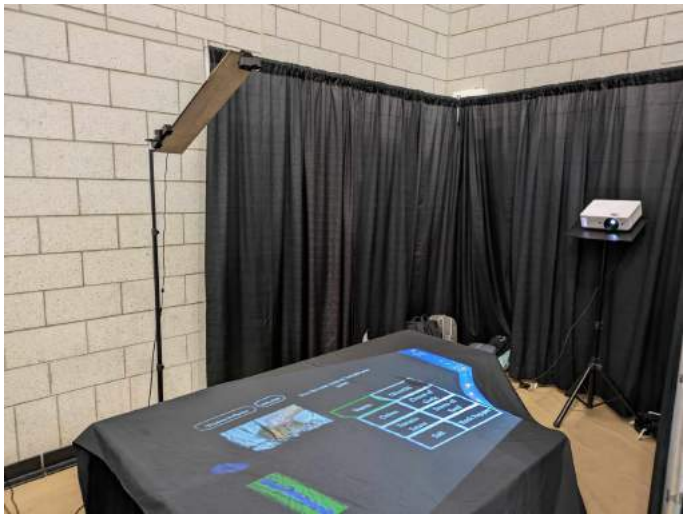

Figure 20: Camera and Projector Setup

This result makes sense. It makes sense that the field of view for a webcam to have a full view of the table at that height. This height was determined by adjusting the height of the tripod until the entire frame was in view.

7.5 Results for Design Specification IR5

IR5: Gesture recognition must have 90% accuracy

To test for gesture accuracy, we executed each gesture command 10 times and confirmed the UI responded as expected. The button press gesture indicates a play/pause video action. The swipe up/down indicates a prev/next video action.

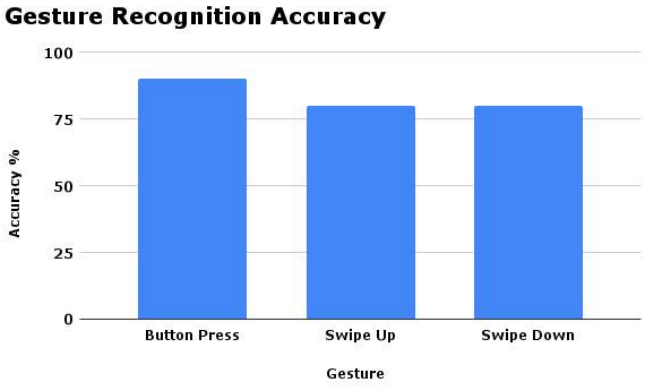

Figure 21: Gesture Command Accuracy

These results show that the button press gesture meets the design requirement with 100% accuracy while the swipe gesture does not meet the design requirement with 80% accuracy. When analyzing this issue, we found that the recognition region for the swipe button was too small. We added a padding of about 50 pixels (found by trial and error) around each each edge of the cropped region which resulted in higher accuracy rates during demo day.

7.6 Results for Design Specification IR6

IR6: The latency of the video instruction must be within 3-5 seconds.

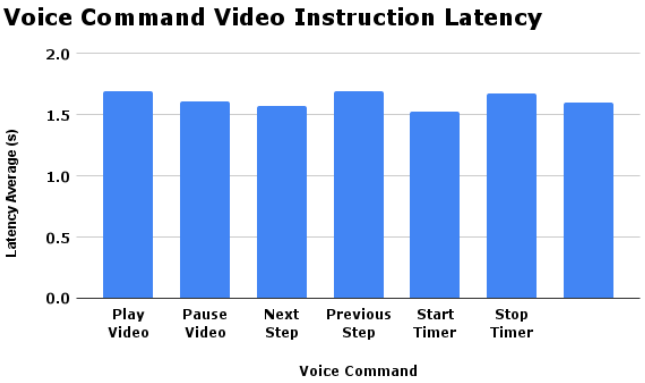

Figure 22: Voice Command Video Instruction Latency

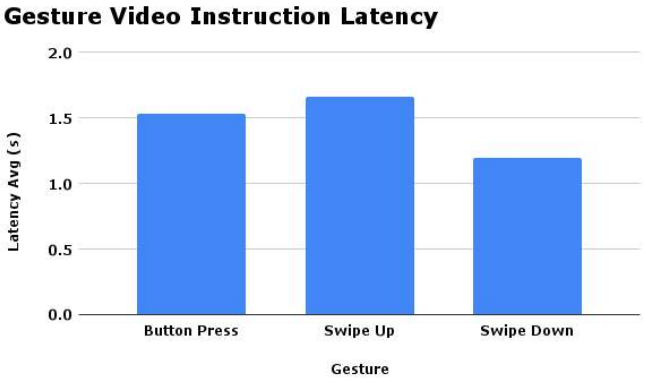

Figure 23: Gesture Command Video Instruction Latency

This test measures how well the video responds to voice/gesture commands based on time. We executed each command and once the command was recognized, we measured how long it took for the video to change. As seen by the data in Figure 22 and Figure 23, no command resulted in more than 1.75 seconds response time. Our requirement was that the response time must be within 3-5 seconds, and these results show that we far exceed this requirement.

7.7 Results for Design Specification IR7

IR7: The gesture and voice commands must be recognized within 3 seconds.

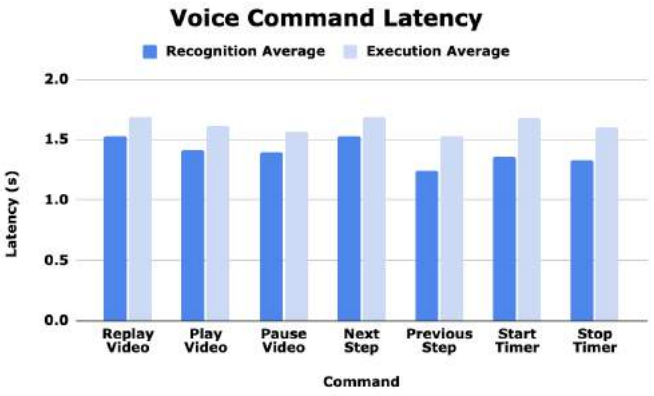

Figure 24: Voice Command Latency

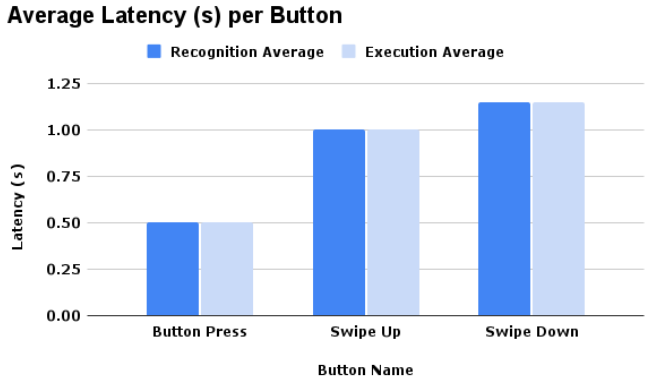

Figure 25: Gesture Command Recognition Latency

The results in Figure 24 and Figure 25 show how long it takes to recognize each gesture and voice command. We compare this data to how long it takes to execute the command, ie how long it takes for the step to change based on the command. It is clear from the tests that both modules meet the desired requirement of 3 seconds.

Throughout the testing process, we made multiple algorithmic changes to improve the latency. For gesture command, we collected data over less continuous frames so less time is spent processing gestures. For voice commands, we ran the module on the GPU rather than the CPU for fast processing and this greatly improved recognition latency.

8 PROJECT MANAGEMENT

8.1 Schedule

The schedule is shown in Fig. 27. The primary changes made to our schedule were to extend our development into our existing slack time so that we could handle the extensive delays working with the AGX Xavier caused us. After deciding to pivot towards using a laptop as our main PC, we had to make up for all the time spent on the AGX. The backend integration saw the most push back in this regard.

8.2 Team Member Responsibilities

- Sumayya - Computer Vision Tasks
  - Gesture Recognition
  - Object Re-Identification
  - Object Tracking
- Tahaseen - Projection Hardware Tasks
  - Projection Warping
  - Projection Calibration
  - Hardware Mounts
- Caroline - Voice System Integration Tasks
  - Voice Recognition
  - Projector Web UI Development
  - Module Communications

8.3 Bill of Materials and Budget

See Table 1 for the Bill of Materials. Most of our budget is allocated toward the projector because the quality of the projection is vital to the user experience. Other major components, such as the AGX and camera, are borrowed from the department.

*Note: These items are borrowed from the course inventory or other campus resources and therefore cost $0.

Table 1: Bill of materials

Description	Model #	Manufacturer	Quantity	Cost @	Total
*Jetson NVIDIA AGX XAVIER	-	Nvidia	1	$0.00	$0.00
*Arducam IMX219	SKU:B0183	Arducam	1	$0.00	$0.00
NETELY Wireless-AC 8265NGW	SKU:B0183	NETELY	1	$19.99	$19.99
*PlatinumPlus 5858D 58” Tripod	620-585BB	Sunpak	2	$0.00	$0.00
BOYA Wireless Lavalier Microphone	BY V20	BOYA	1	$39.95	$39.95
4,500 ANSI Lumens SVGA Business/Education Projector	PA700S	ViewSonic	1	$350.99	$350.99
					$410.93

Table 2: BOM After Design Report

Description	Used	New Item
Jetson NVIDIA AGX XAVIER	No	No
*Arducam IMX219	No	No
*PlatinumPlus 5858D 58” Tripod	Yes	No
BOYA Wireless Lavalier Microphone	Yes	No
4,500 ANSI Lumens SVGA Business/Education Projector	Yes	No
NexiGo Full HD N60 Webcam	Yes	Yes
NETELY Wireless-AC 8265NGW	No	Yes

8.4 Risk Management

The following risks were identified as primary risks and placed in a severity/consequence matrix in order to set priorities. The matrix can be seen in Figure 26 below.

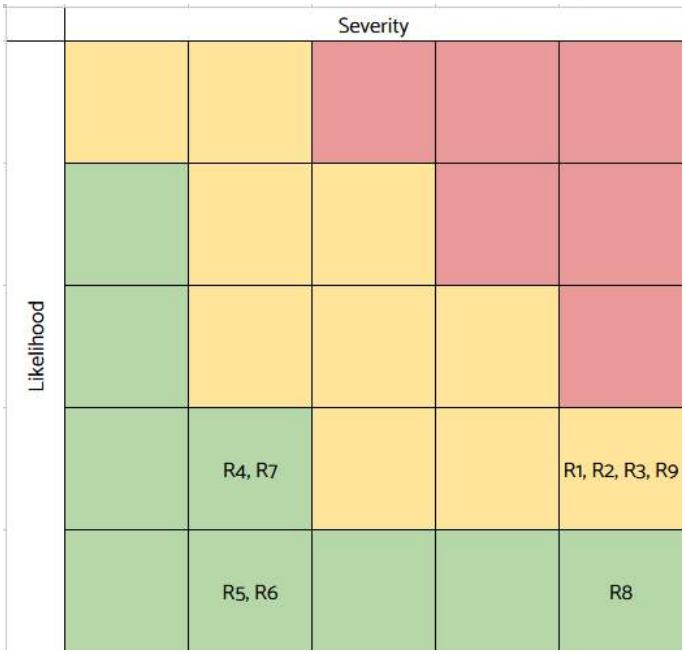

Figure 26: Risk Matrix for Identified Risks

R1. Camera falls due to poorly designed mounting. We mitigated this by having an adjustable camera mount that was redundantly secured.

R2. Projector falls due to poorly designed mounting. This was mitigated because the projector was to the side so that the user could not be harmed. Additionally, we bought a specialized stand so that we could properly prop up the setup.

R3. Camera is damaged by food splatter or any additional spillage. This was mitigated by placing the camera high up and off centered from the stovetop area.

R4. User’s gesture is ignored by chosen algorithm. To mitigate this, we cropped down to the button regions and detected the hand landmarks as noted by MediaPipe.

R5. User’s gesture is recognized incorrectly by chosen algorithm. We mitigated this by tuning how gesture recognition is tested and cropping the search area down to the buttons.

R6. User’s vocal wake word is ignored by vocal command algorithm. This risk could be entirely mitigated, because Porcupine Wake Word recognition was highly sensitive to our wake word ‘TableCast.’

R7. User’s vocal wake word is recognized incorrectly by vocal command algorithm. To mitigate this, we tuned the vocal recognition parameters until they were accepting of all vocal ranges.

R8. Server hosting UI crashes in the middle of a recipe. To mitigate this, we setup safe shutdown states to exit the UI without compromising too much information.

R9. Projector resolution compromising user experience. This risk was mitigated by making sure all of the elements were sized larger, such that they

could be read even with poor resolution.

9 ETHICAL ISSUES

TableCast heavily depends on computer vision to observe user action. As a consequence, there is a high chance of impact from bias on skin color. This can especially be a concern when the counter top color matches closely with that of the user's skin color. MediaPipe depends on image features rather than color and so is less likely to discriminate against one's skin tone for gesture recognition. We used bright and unique colors like red and blue for calibration to avoid any confusion in our region detection pipeline. With these methods we were able to mitigate the issue of a racial bias in our system.

The system's voice command feature may be something that causes concern for users. The voice command module is always listening for the wake up word throughout the cooking process. Although no data from the voice module is saved to a file, there are still potential ethical concerns about how this always-listening voice module is used. There are no security features for this product, so a bad actor could potentially access voice data, and the users' trust could be violated.

Given that the voice commands are more reliable than the gesture recognition, it can cause issues for users who solely rely on gesture commands to use the interface. The user would have to continue attempting to use the gesture recognition instead of moving onto the voice commands.

10 RELATED WORK

There are few similar products to TableCast which make our application unique. The most common method of creating a large digital tabletop is by replacing the table with a large screen. However, we aim to make sure that TableCast is functional for most kitchen counters without requiring expensive replacement. On the other hand, there do exist a few table projectors in prototype phases in the 2010s, but these were never pushed to the market and/or do not cover a large table surface like we intend to do. The primary example for this is Sony's Experia Touch [3]. However, our product is specific to kitchen environments and guiding users through cooking on a large and chaotic environment. This is in contrast to Sony's small and simple environments.

11 SUMMARY

Our final design was mostly able to meet our design specifications but we were severaly limited by the abilities of this projector. The light dispersion, throw and resolution of the projector all came out to be a problem when designing the final product. Even then, there was a certain point after which it was simply not possible to update the output display due to these limiting factors in the projection. These factors also made it difficult for us to crop down the button regions and take into account the faded light setup in that area. Should we have more time, it would be invested in helping the ingredients and a large enough table to include the stove top workspace.

11.1 Lessons Learned

Crooks - The main lessons I learned were from working on the integration step. I did not read about the specifications of the AGX Xavier that we had borrowed from the department, which caused many issues later on. I did not know that there was no wifi card and that there was only 32 GB of storage provided. I had to wait and order parts for this. Later on, when trying to transfer the code to the AGX, there were various compatibility issues with versions. I should have done this research beforehand before trying to use the device. Another lesson that I learned was to be more organized when doing module communications. The connections between devices quickly became confusing, so I learned that next time I should make more detailed diagrams.

Shaik - The main takeaway for me in this project was mostly centered around the table and I would encourage future teams using projector setups to get a larger resolution ad the band can be tuned such that it is more acceptin g of figures when I show with this type of projector. I would make sure that the students understand the brightness and resolution ratings when ordering a huan color.

Syeda - The main lesson I learned was that it is extremely important to do a feasibility analysis of your project before starting. Truly think about what the project will require and what you want to achieve by the deadline **in detail**. With TableCast, I realized that selected a more challenging project with the low reliability on light projectors and image filtering. I also learned to find ways to test software even if the hardware or other components aren't ready. I was unable to test much of my CV pipeline as I was waiting on the projection module to be completed. Once it was complete, I was left with very little time to test CV. If I had found ways to test that mostly mimicked a projection but not did not rely on projectors, I could have had more time to improve the CV pipeline.

Glossary of Acronyms

Include an alphabetized list of acronyms if you have lots of these included in your document. Otherwise define the acronyms inline.

- MQTT – Message Queuing Telemetry Transport
- OBD – On-Board Diagnostics
- RPi – Raspberry Pi

References

[1] Kien Nguyen Phan et al. "Assessing Bicep Curl Exercises by Human Pose Application: A Preliminary Study". In: Mar. 2023, pp. 581–589. ISBN: 978-3-031-27523-4. DOI: 10.1007/978-3-031-27524-1_55.

[2] Akshara Pande et al. "A Comparative Analysis of Real Time Open-Source Speech Recognition Tools for Social Robots". In: *Design, User Experience, and Usability*. Ed. by Aaron Marcus, Elizabeth Rosenzweig, and Marcelo M. Soares. Cham: Springer Nature Switzerland, 2023, pp. 355–365. ISBN: 978-3-031-35708-4.

[3] Nick Statt. *Sony's prototype projector turns any tabletop into a touch-sensitive display*. The Verge, Mar. 2016. URL: https://www.theverge.com/2016/3/13/11215454/sony-interactive-projector-future-lab-sxsw-2016 (visited on 03/01/2024).

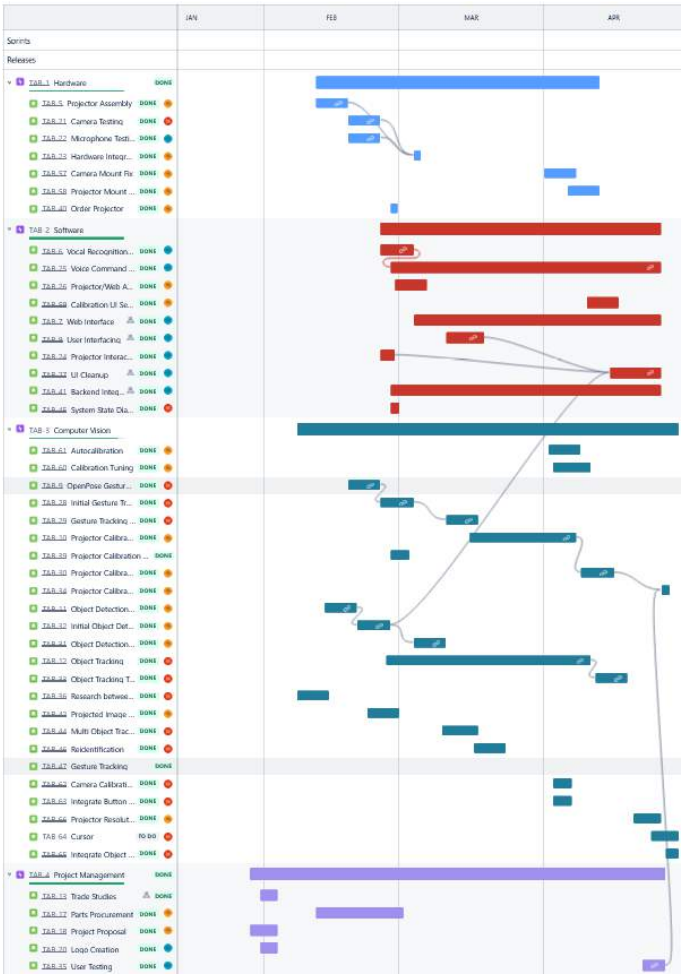

Figure 27: Schedule for Spring 2024