Project Title: Touchless Kiosks
Team Number: 37
Name: Aaryaman Sharma
Student ID: 21012050

1. Personal contributions and challenges faced

1.1 Initial stages/Research

During the start of the project, I was heavily involved in the research and requirement gathering process. I worked with Aryan and Yidan to conduct client interviews and draft up a list of initial requirements for our project. Based on these, I created numerous sketches for our project prototype based on these requirements, which were used in our HCI report. Following this, I conducted a lot of the research for our literature review, including extensive research into the current accessibility of kiosks and the feasibility of state-of-the-art gesture-based solutions. We received extremely positive feedback from Dean, who praised my discussion of current gesture-based solutions and their drawbacks.

1.2 Start of development

Following this initial research stage, our team was a bit stuck due to being limited to M1 MacBooks. It was impossible to run MotionInput since it was Windows only. I extensively experimented with the software and managed to figure out a boot-flag that would force Parallels to use low-level APIs, finally making MotionInput fully functional in Parallels. I reported my findings to Dean, who praised my efforts and asked that I create a guide for other students in the cohort. I wrote out and distributed a thorough guide for the process and managed to help other teams with M1 Macs run MotionInput without any issues.

After this, we were finally able to dig into the existing MotionInput codebase and spent the next few weeks familiarizing ourselves with the project. I was the most involved in reading the codebase and explaining the layout of the project and key concepts to my teammates. One of our key early requirements was to be able to focus on the main kiosk-controlling hand and ignore any hands detected in the background. To get a better grasp on this problem, I extensively researched the ML libraries used in MotionInput, such as MediaPipe, and experimented with alternatives such as YOLO. After writing some quick tests, I found that MediaPipe was best suited for hand tracking performance-wise but didn't have the crowd rejection capabilities that we desired. MediaPipe would choose a hand effectively at random, and we didn't yet have a way to consistently focus on the best hand. I kept this issue for later and focused on creating our Elevator Pitches for project partners.

1.3 Elevator Pitches

For our elevator pitches, I contributed my understanding of the basic functionality of MotionInput, and the challenges associated with the project, such as the crowd rejection problem. My work on the literature review was also used as a basis for explaining our requirements and project scope. We had a few issues with making our pitches concise and engaging, but we got some much-needed feedback from Dean and made the appropriate improvements.

1.4 Continuing development on MotionInput

During the main development phase, I took on the most responsibility and was the most involved in the coding and testing aspect of the project. I would have frequent discussions with my teammates regarding the direction of the project, which features to add, and would implement most of the features myself. I would make consistent progress each week and display them to the TA. I started out with improving the MotionInput backend for our use-case. Using the knowledge gained from my machine learning research, I developed an accurate heuristic for detecting the best hand in frame using MediaPipe. This would make its way into the MotionInput backend and solved our crowd detection problem, since only the best hand in frame, according to my heuristic, would be considered the controlling hand. I conducted several experiments with my teammates and friends to finetune the detection, and added it to the MotionInput backend, along with an appropriate config flag that other teams could use to turn it on or off.

After this, a lot of work was spent on implemented and perfecting the swipe gesture that was intended to be used with kiosks. From empirical testing, we discovered that a key problem with swipe based touchless gestures is knowing when the gestures start and end. For example, if a user swipes their hand to the left, is it because they want to do the left-swipe gesture or because they are coming back to the middle from a right-swipe gesture? I made some initial protypes involving speed gradients but eventually decided it was too unwieldy. Instead, Aryan and I worked together to devise an acceptable alternative. We designed a threshold-based swipe system, where the user would swipe from the center of the frame to one of the edges depending on the set sensitivity. Swipes would only register if the user first started from the center, making the gestures much more reliable yet still intuitive. The overall result didn't perform very well, so I redid the logic to use only vector-based calculations using fast C-based libraries such as NumPy and remove any native Python calculations or slow constructs such as for loops. The code was a little bit more mathematically involved as a result and required some linear algebra knowledge to understand, but the massive performance drain and removal of lag spikes was worth it.

1.5 Starting development on frontend

After finishing most of the development on the MotionInput backend, we moved on to creating an appropriate frontend. We knew that most kiosks simply ran in sandboxed web browser environments, and so we needed to develop a frontend that could work with those. The natural solution was to develop an extension that could hook into any such kiosk interface, and act as the frontend for our MotionInput backend. However, I had no experience with extension development and decided to first work on a dedicated website with this functionality, and then convert it to an extension later if possible. The first major problem was facilitating inter-program communication between MotionInput and the frontend. This was exacerbated by the fact that they were built with different programming languages, and that the frontend would have no access to the native OS filesystem due to the browser sandbox. Aryan and I first researched technologies such as WebRTC and tried implementing a dummy frontend to test its communication capabilities. However, we discovered that it was too complex for our needs, which was just two-way communication between two programs, and required things such as signaling servers. After some research, I discovered the mature WebSocket protocol that allowed real-time communication with websites over TCP. It is like HTTP requests and responses, but doesn't have the additional overhead of polling, making it extremely efficient for our use-case of a live webcam preview.

Once we figured out what protocol to use, I started development on a quick test page that would receive some information from the MotionInput backend via WebSockets. Python has an extremely mature WebSockets library with good high-level APIs, making the process much more reasonable. However, the library heavily relied on cooperative asynchronous programming concepts in Python, such as the Python asyncio library and event loops. While I was familiar with such concepts in other programming languages such as JavaScript, Python's implementation is still in development, and the APIs use change from version to version. It was quite challenging trying to learn about asynchronous programming in Python, when most documentation about the asyncio library was outdated. It took quite a bit of time to catch up to modern Python's version of cooperative asynchronous programming, but once I got the hang of it, mostly by trial and error, I could appreciate how useful it was for our use-case. I managed to integrate a WebSocket server into MotionInput that would send messages to the WebSocket client dummy page. I created the dummy page in vanilla JavaScript but knew that this wouldn't be feasible for a larger scale project. I already had experience with React, so I started building my frontend in that instead.

1.6 Starting Development on Intermediary Server

Another issue was that Dean wanted the kiosk solution to be resilient and be able to launch it directly from the frontend in case the MotionInput backend crashed. At that point, the WebSocket server was directly integrated into MotionInput and so if it crashed, there would be no communication between the frontend and backend. Also relaunching wouldn't be possible since the browser sandbox doesn't allow launching apps. I investigated some solutions such as registering custom URL schemes that would open MotionInput but ultimately decided against it, since it wasn't very universal. Instead, I thought of creating an intermediary relay server that would maintain bidirectional communication, but still be alive if MotionInput crashed. The frontend could then request a relaunch of MotionInput, and the intermediary server could directly do so due to running on the host OS. I migrated the WebSocket server code to the intermediary server, but faced a new problem, which was communicating between MotionInput and the server. I had some experience with UNIX sockets in Python before but thought that it was too low-level for our uses. I eventually discovered the popular and mature ZeroMQ messaging library which is a cross-platform abstraction of UNIX sockets. It also allowed sending JSON-encoded messages without worrying about specifying message headers, message sizes etc. The only issue I faced was making the ZeroMQ server work with asyncio, but I eventually abandoned that idea, since the feature wasn't fully implemented in the library, and instead pivoted to traditional multithreaded concurrency for the ZeroMQ communication. I then added an appropriate ZeroMQ client to MotionInput and provided a high-level API for other modules. I faced another issue with huge CPU usage due to excessive polling, but once I added appropriate rests to the threads, the intermediary server became extremely performant and lightweight. Now that bidirectional communication was fully functional, I started working on the extension.

1.7 Starting development on WebExtension frontend.

Turning a webpage into an extension was surprisingly straightforward, and I just had to specify a manifest.json file following the Manifestv3 documentation. My frontend only needed to render a page in a sidebar, so I could simply specify the main webpage I needed to render without worrying about content security policies, background scripts etc. I had a lot of experience with React so developing the frontend was relatively straightforward. I faced one issue with the live abstract hand preview, where its movement was very choppy. I realized this was because I was using the CanvasAPI, and was redrawing the canvas at each message received, adding lots of overhead. Instead, I pivoted to using native DOM elements and some CSS and JS trickery to position an abstract preview of the user's hand in the specified preview window of the frontend. I also added some event listeners to the MotionInput backend that could receive config changes such as sensitivity changes from the frontend and act on them. While the extension could hook into any sidebar, I also created a demo hospital kiosk UI webpage to show to our clients at Great Ormond Street Hospital. Aryan and I extensively discussed our solution with the clients, and we demonstrated our solution live to the clients. We received some great feedback about user experience, which I implemented later.

1.8 Finishing up

After development was complete, I was responsible for creating build and compilation scripts that would compile all our source code into an easily packageable format. I also submitted our sidebar extension to Mozilla and got it officially signed and notarized, meaning that users could permanently install it to Firefox without any security warnings. For our portfolio website, I also wrote the entirety of the system design, implementation and testing section since I was primarily involved in these aspects of the project and wrote out the deployment manual. I helped with recording a demo for the 4-minute video presentation and edited the entire video as well.

2. Assessment of team members

Me: I would say that my strengths were technical skills, research skills as well as document writing. I was almost singlehandedly responsible for developing the crowd detection features in MotionInput, creating the WebExtension frontend and demo website, and the intermediary server, as well as compiling and distributing everything. My research and document writing skills were demonstrated during the literature review phase as well as when writing deployment manuals, writing guides for getting MotionInput running on M1 Macs, and writing the entirety of the system design, implementation and testing parts of the portfolio website. My weaknesses were with the website and team communication. I only contributed the writing for my sections, while the entire design, layout and deployment were handled by Aryan. I was also initially reticent in our team, and relied on the others to initiate meetings, communicate with the TA etc. While I did communicate much more during the development phase, I would say that I could do a better job of establishing strong ties at the start of the project. Overall, I would say that my main role was the programmer and tester. I would give myself a score of 9 for my extensive research, huge contributions to the code and documentation, as well as client liaison when it came to showcasing demos etc. I'm docking off a point due to my lack of initiative at the starting point of the project.

Aryan: Aryan was also very involved in the project from the start. His strengths were document writing and communication skills and reliability. He was always the one to initiate conversations, creating all our group chats and hosting meetings when necessary. He kept our project extremely organized and kept on top of all our deadlines and any future work we needed to do. He didn't contribute as much to the code section, but he did extensively discuss the general architecture and functioning of our system and developed the MFC admin frontend, which was a significant contribution to our project. He also did by far the most work on the group portfolio by developing the entire website himself, writing most of the other sections, clearly asking for deliverables from other team members and generally keeping us on track. His weaknesses were technical skills and research skills. He mostly let me and Yidan handle the literature review and research part of the project, and only contributed to the structure of our presentations. As I mentioned before, for the technical skills, he also mostly contributed with the MFC frontend, while I handled every other component of our project. Overall, I would say his main role was the report editor and project partner liaison. I would also give him a score of 9, due to his thorough report writing, the creation and maintenance of the portfolio website, ability to organize the project throughout and frequently communicate with clients and Dean to gather feedback. He loses a point for his relatively low contribution to the code part of the project.

Yidan: Yidan was by far the least involved member of the group. I would say his strengths include research skills and document writing. He was heavily involved near the start of the project and did majority of the research for the HCI presentations and elevator pitches. I really liked how he went out and analyzed kiosks in real-life as case studies for our research. He also helped in discussions about the interface of our project and made good critical evaluations about our user experience and bugs, which helped drive our project in a better direction. He also wrote a lot of reports, personas and documentation such as dev-blogs for our project portfolio website. His weaknesses include communication, reliability and technical skills. While he did write a lot of useful documents, they usually required a lot of proof-reading from Aryan to distill the best portions of his work. He also almost never engaged in conversation during the development phase of our project and would only show up to meetings with the TA. It was very frustrating having an essentially missing team member during such a key phase of our project, and it increased the load on me and Aryan. He never showed up to any meetings with Dean and had no liaison with the clients. For instance, he wasn't present during our important demonstrations at Great Ormond Street Hospital. He contributed nothing to technical parts such as system design and coding, but did do a decent amount of empirical testing, allowing me to fix many bugs that went unnoticed. He did eventually start contributing again during the group portfolio phase and helped Aryan with writing a decent number of documents as I mentioned before. Overall, his main role was the researcher due to his extensive contributions during the initial research phase of the project, and the decent contributions made during the group portfolio writing phase. However, I would only give him 4 points due to him refusing to communicate for most of the project, and not providing any help during the development phase of the project. I still appreciate his help with the research-based deliverables, and his work on the group portfolio documents, but it is minimal compared to the work Aryan and I had to do to cover up for his lack of engagement.

Overall, I would say that the individual contribution table on the report website agrees with my assessment, and I have no qualms with the assigned percentages.