Flavor Fusion

Milestone 6

Team Members:

Ethan Butterfield, Alexandra Figueroa, Ryan Latterell,

Samuel Pabon, Thomas Spurlock, Allison Turner

Due Date:

November 18, 2024

EML4502 – Senior Design 2

Advisor: Luke Ponte

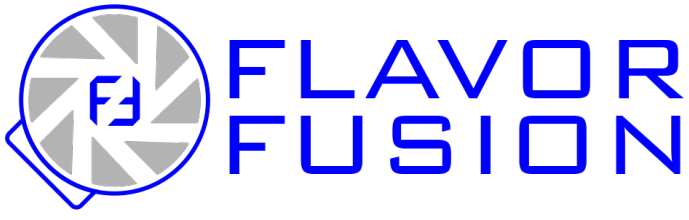


Table of Contents

List of Figures

List of Tables

1. Executive Summary

The team at Flavor Fusion set out to develop and create a spice dispenser for at home and commercial use cases. This device aims to tackle different problems that many foodies, especially budding foodies, run into. One of them is food waste: food waste along with any kind of waste is a huge problem the modern world faces every day, and our product is designed to reduce the overuse and waste of these ingredients by allowing the user to control and adapt recipes or blends however they seem fit. This device is inspired by cereal dispensers and 3D printers where the device will allow refillable spice cartridges to dispense a specified amount of spice that is controlled by the user. The device features an LCD display that gives real-time updates on the dispensing process, and allows users to create one-off blends. Flavor Fusion's mobile app wireless connects to the device via Bluetooth to upload custom blends, determine spice levels, store recipes, and order specific quantities and blends of spices. The physical device has been designed to be a reliable companion that doesn't disrupt the kitchen environment, and allows users to be the best home chef they can be.

2. Project Background and Objectives

2.1 Project Background

Flavor Fusion is a spice dispenser that stores all of your favorite spices and even more of your favorite blends, making cooking more convenient and user-friendly. The device utilizes refillable, dishwasher-safe containers to help declutter your spice cabinet, while our user-friendly app ensures you can select, blend, and measure spices from across the kitchen, all while preventing cross-contamination.

Flavor Fusion sets itself apart from other kitchen appliances by taking the math out of cooking. No longer do you need to dig through your kitchen to find the right measuring cup, or try to scale your recipes, Flavor Fusion does all of that for you. Whether you're an experienced at-home chef or just starting out, Flavor Fusion allows you to focus on other prep work tasks and gives you more time to entertain guests.

Remember the last time you were at the store and couldn't remember if you had enough paprika? You bought another bottle, but then came home to find three other bottles in the cabinet. With Flavor Fusion, you will never forget how much spice you have left, as the mobile app displays the latest spice levels, and reminds you to purchase more when the device is low.

2.2 Long Term Objectives and Payoff

The Flavor Fusion team's future goals for the physical device are to reduce production costs through mass production using injection molding. The team's future goals for the mobile app are to officially publish the app on the App Store and create an Android version. Cost reduction and app publication will improve affordability and accessibility for customers, who serve as the stakeholders for the product. Since the device is a stand-alone product, the team's work would not need to be combined with other future developments, but it would have to be marketed for retailing before long-term benefits are realized.

2.3 Current Semester Objectives

This semester, Flavor Fusion intends to deliver a final developmental prototype that is fully functioning and interfaces with a developer-built iOS application. The device will be sized to target dimensions of 15” x 15”, dispense the correct blend of spices via app-based commands, and will measure the spices being dispensed volumetrically within ±1/8 US teaspoons. The goal of our end product is to demonstrate how a future mass-produced device may appear and function via showcase interactions with stakeholders.

2.4 Technical Approach

The Flavor Fusion team’s technical approach involves the delegation of subsystems to leverage each member’s technical skills. Sprint-style task forces will facilitate the merging of subsystems during final assembly. Early stages of the project will rely on rapid prototyping with 3D printing methods such as FDM and SLS print styles, so the dispensing mechanism, spice containers, and entire housing will be CAD modeled. These early rapid prototypes will create a smoother transition from ideas to real life products. Ethan will work on design and part procurement to ensure accurate sourcing of material. Alexandra will oversee development and design to ensure technical requirements are met. Ryan will lead the project and coordination of subsystems. Samuel will integrate mechanical and electrical components. Thomas will manage manufacturing and design to refine the final product. Allison will code the integration of the device and app while overseeing user interaction and control.

2.5 Project Plan with Deliverables and Dates

Table 2.1 – Project Plan with Deliverables and Dates

Deliverable	**Date Range**	**Deliverable (contd.)**	**Date Range**
Hand Calculations Power, torque, kinematics	Aug. 19 – Sept. 1	Test Dispenser	Sept. 24 - Oct. 11
Research and Order Parts	Aug. 19 – Sept. 3	Test Housing	Sept. 25 - Oct. 11
App Development Layout, features, UX	Aug. 19 - Sept. 17	Test Electronics Motors, sensors, UI	Sept. 30 - Oct. 8
Simulation/ Analysis Circuit/Wiring, FEA/CFD, Motion Study	Sept. 1 – Sept. 14	Program Electronics Arduino, motors, sensors, UI	Sept. 30 - Oct. 8
Dispensing Mechanism Proof of Concept	Sept. 3 – Sept. 8	Integrated Subsystems Testing Dispensing, fit hardware	Oct. 4 - Oct. 11
Test Containers Test and redesign feedback loop	Sept. 13 - Sept. 28	Showcase Presentation Video, PowerPoints, models	Oct. 11 - Nov. 18
Make Housing Base, Container Support	Sept. 15 - Sept. 25	Final Assembly	Oct. 15 - Oct. 25
App Testing Bluetooth, user tests, bug tests	Sept. 17 - Oct. 25	Final Testing	Oct. 25 - Nov. 1
Make Dispenser CAD, 3D-print, redesign	Sept. 18 - Sept. 25	M6 Writing	Nov. 1 - Nov. 15
Outline Arduino Code Pseudocode	Sept. 22 – Sept. 29		

3. Technical Memos

3.1 Industry Examples

When envisioning what a spice dispenser and blender should look like, there are a multitude of factors to consider. Not only does the machine need to dispense the correct amount of spices for the user, but it also needs to do so using food-safe materials and design, intuitive user interactions, and a package that users want in their homes. As such, looking at industry examples across multiple use cases can provide the perspective needed to shape what product is being created for Flavor Fusion. Of these, this section dives into three food and beverage options, TasteTro, Coca-Cola Freestyle, and Keurig, one automotive with FordPass, and one manufacturing with Prusa 3D printers. Each brings their perspective on User Interface and User Experience Design, dispensing qualities, and mobile device interactions.

3.1.1 TasteTro

Starting with the most directly related to Flavor Fusion, TasteTro was at best a trade show prop that never made it off the ground and quietly disappeared in December 2021. It is no wonder the machine was never picked up by retailers or caught on with consumers with its hefty $269 starting price [1] and proprietary spice pods that cost anywhere between $7-12 [2]. The proprietary pods that the machine required were not intended to be user-accessible [2], meaning that consumers had to depend on TasteTro to stock and ship their spices before the machine ran out, but also to not go out of business. Monetarily, it was a tough sell, and as seen in Figure 3.1.1, it took up space on a counter that spices otherwise did not occupy. However, elsewhere, the TasteTro offers much to learn for Flavor Fusion. Starting with its core vision: “To inspire culinary confidence, clean cooking and flavor adventure, while easing the stress of daily cooking” [3], TasteTro closely aligns with Flavor

Figure 3.1.1 - TasteTro at the Chicago Housewares Show, March 2019 [101]

Fusion's goal. The team at TasteTro researched what spices and blends are most popular and had a solid understanding of who may be using their product. They are not targeting a gourmet chef so much as the everyday home cook who may not be as flavorfully ambitious, or who may get overwhelmed in the spice aisle. It removes the guesswork for the user with a simple display and control interface that allows users to dispense the exact amount of spices they need for any of the over 50 pre-programmed blends that came with the machine. A Bluetooth-connected mobile application would allow users to update the machine with new blends, notify the user when the RFID-enabled spice cartridges were low, and order new cartridges [4]. However, despite the simplicity, the device was still flawed from a feature-based standpoint. Users were unable to add their own blends and had to rely on what the machine came pre-programmed with. The spice pods are in theory a great idea, especially knowing that spices lose their flavor as they age. The machine is provided with a start value for how much is in each pod and the RFID technology allows the machine to recognize the spice from any slot, not just a specific one, it is almost a "two birds, one stone" situation. However, the user can then no longer access their spices without the machine and is reliant on the spices that TasteTro offers. Based on what Keurig was able to accomplish, it could be assumed that TasteTro was hoping to make more money on the spice pods than the individual machines, but this razor/razor blade sales model can only work so well when another more convenient option already exists in stores today: the spice aisle.

3.1.2 Keurig

Going down a different aisle of the grocery store, there is a very different situation from TasteTro. Take one look down a coffee aisle and the success of Keurig is abundantly clear. K-Cup pods dominate the shelves and print millions of dollars each year for the company [5], partly due to its brand partnerships [6], but mostly due to America's dependence on coffee. According to a 2011 article from boston.com, "roughly one in two US adults consumes coffee, and the average drinker quaffs two cups a day", and "[Keurig] was surprised to see that people kept buying K-Cups, even in the worst of the [2008] recession". That demand fuels Keurig's very convenient money maker, as users can pop the pod into the machine, press a button for how big of a mug they are using, and within minutes they have a piping hot cup of coffee (or hot cocoa, or tea). The cost of convenience is high, as K-Cups can be up to 10 times the cost of the equivalent amount of bagged coffee [5], and they are known for being detrimental to the environment, despite changes made by Keurig in recent years [7]. The pods are also not the freshest coffee available, as they have to be packaged two weeks after being roasted due to the sealed nature of the cups and the gases released by the coffee grounds [7]. Keurig has also tried the proprietary game, introducing BrewID as a feature that would customize the blending settings by the coffee inside the pod, but it also blocked the machine from brewing if it could not detect the QR code printed on the lid of the pod [8]. After consumer pushback, Keurig removed the proprietary system. BrewID is still functional through their mobile app with features that allow users to be their own baristas, and, like TasteTro, allows users to order pods directly to their door [9]. Keurig also benefits from an easy cleaning process with their own cleaning products [10] or with vinegar and water [11]. This helps the machine stay clean over time and provides a better customer experience overall with no worries about contamination or having to disassemble the machine to clean it.

Figure 3.1.2 - BrewID allows users to customize how their coffee is blended or brewed. [9]

3.1.3 Coca-Cola Freestyle

An additional beverage dispenser with incredible feats of engineering is the Coca-Cola Freestyle machine. The team behind it started from scratch for how a soda fountain should look and operate, leading to an incredibly space-efficient and cost-saving design, a unique single nozzle design, and a machine that made soda fountains cool and slick for a new generation. The machine was designed with the intent of giving users options and messing with the ratios that sodas are blended for a more space-efficient design [12]. As a result, the Freestyle uses micro-dosing technologies found in the medical space for its highly concentrated cartridges, as seen in Figure 3.1.3. The cartridges save a huge amount of space for restaurants and help Coca-Cola increase its sustainability with a lower carbon footprint, as they can pack more cartridges on a truck than the previous bags used in the old dispensers [13]. To combine those specific blends without cross-contamination of flavors, the Freestyle uses an innovative air mix nozzle to allow the ingredients to mix in the air around the nozzle, rather than in hard-to-clean pipes within the machine [12]. This allows one user to choose their own custom Sprite that does not affect the flavors of any of the over 165 other combinations that the Freestyle is capable of dispensing [12]. The nozzle does have to be rigorously cleaned at the end of each day, as does most commercial food equipment [14], but it is still an impeccable feat on Coca-Cola's part. Not only did Coca-Cola give consumers the power to choose from more than just the basic choices, they made the machine look sleek. Coca-Cola recruited Pininfarina, the same company that designs Ferraris, to craft the exterior of its machine, and it worked. The design of its exterior catches the eye of consumers, and the sleek display makes it easy to select a user's desired choice. With a strong design backing it up, Coca-Cola made soda cool again. At its launch, the Freestyle caused an average of 6-8% increase in beverage sales with a meal according to USA Today [15]. As the machine has become more common, Coca-Cola has rolled out app connectivity, allowing users to connect to the machine in stores and dispense their favorite blends from their phones [16], in addition to decreasing

Figure 3.1.3 - An opened Coca-Cola Freestyle in a Wendy's [107]

contamination in the era of COVID-19. The company has also been able to use the data from the Freestyle to influence products they bring to market, like Sprite Cherry in 2017 [16]. They can know what users want, when they want it, and how they want it. The company also makes this data available to restaurant owners on a service screen, so they know what users will want in advance and can stock up accordingly [14]. The Freestyle is also priced very competitively, each machine is $3600 per year [17] in addition to $40-50 per cartridge [18], meaning that a restaurant would need to sell approximately 5 drinks per day for a year to break even. In terms of drawbacks, the biggest hardware drawback is the lack of a built-in ice maker for an extra expense, while on the software side, the machine's UI is not the most intuitive for older, not as tech-savvy users. Additionally, the design of the concentrated cartridges and slots in the Freestyle means that restaurant employees now have to replace them on the floor, rather than in the back of the store. This means if the machine runs out of syrup during a rush, customers have to wait for an employee to come out and replace it. The predictive data analytics of this machine should be able to compensate for that, but during a rush, it may be all hands on deck. One other complaint from users is that the flavors are prone to mixing between users, something that could be attributed to the air nozzle design or lack of proper cleaning and maintenance [19]. Ultimately, the Coca-Cola Freestyle is an exceptional example of what a user-controlled dispenser can be.

3.1.4 FordPass

In terms of how users can interact with Flavor Fusion, looking outside of the food and beverage industry has introduced some creative thinking and unique approaches to UI/UX design. Starting with the FordPass app, users can check the status of their vehicle from anywhere, so long as they are connected to the Internet and their vehicle has a cellular signal [20]. Of course, the app is capable of remotely locking, unlocking, and starting the vehicle, or locating the vehicle on a map, but the key feature similarity for Flavor Fusion is the ability to remotely view the various fluid levels of the vehicle [21]. As seen in Figure 3.1.5, the app presents the remaining fuel in terms of miles left, something that a user can immediately understand, as opposed to offering it in gallons remaining or something that cannot be easily contextualized. Furthermore, the remaining oil is presented with a percentage of oil life remaining, in addition to the distance remaining and when the car may need to be taken into the dealership for service, as seen in Figure 3.1.4. This approach helps the consumer understand what the requirements of their vehicle are, and how to best take care of it. In terms of Flavor Fusion, this could be implemented as a way for customers to check the amount of spices remaining when away from home, like in the grocery store. The UI is also very simple and provides relevant information to the user without being overwhelming.

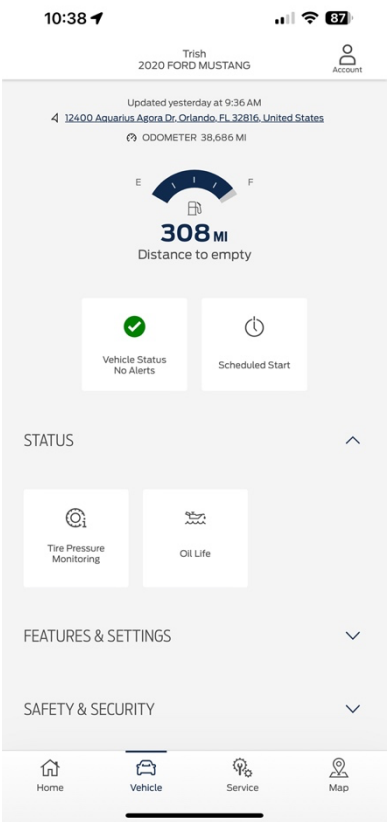


Figure 3.1.5 - FordPass showing the Vehicle status [21]

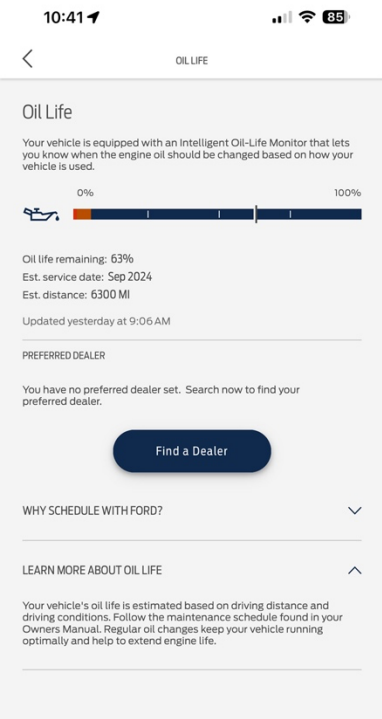


Figure 3.1.4 - FordPass Oil Life Page [21]

3.1.5 Prusa 3D Printers

Figure 3.1.6 - Original Prusa i3 LCD Menu [108]

When it comes to interacting with Flavor Fusion itself, Prusa 3D printers offer a simple menu system and controlling device that presents information to users on an LCD display. The interface is controlled by a simple knob that can twist and press, and a reset button underneath that. As the knob twists, it can be used to scroll through menus or scrub through values, such as temperatures, quantities, or heights. When pressed, it can confirm selections and navigate through menus. This simplistic control is quite intuitive and immediately allows the user to interact with and understand the machine. Furthermore, having a physical display and control module is beneficial to users who may not want to always rely on a mobile device to complete basic tasks. The twisting knob is innate to other systems to control quantity, like faucets, volume knobs, or burners on a stove.

Through all of the options presented here, there are a multitude of industry examples that can be used to shape Flavor Fusion's outcome. It should be important to look outside the food and beverage industry, such as the automotive and manufacturing industries. By doing so, Flavor Fusion can proceed with better outcomes in designing a user-friendly and optimal product.

3.2 Integration

3.2.1 Power Supply

No energy source is 100% efficient. Energy can be lost in multiple forms such as heat, light, or sound. Our goal is to have the most efficient energy source, while still making powering the device affordable and easy.

Gas-powered engines can lose as much as 80% of their energy through engine heat, evaporation, oil extraction, refinement, and transport [22]. This, coupled with the large average size, and the fact that this is the least sustainable form of power being discussed, all but rule it out.

Batteries lose significantly less energy than gas engines. Batteries can reuse between 80–90% of the chemical energy stored [22]. A typical AA battery contains about 3.9 watt-hours, or 0.0039 kilowatt-hours [23]. The average cost of a single AA battery is about 80 cents. The average residential electricity rate in Florida is 16 cents/kWh [24]. This means that for every 80 cents it would cost to buy a battery, the same amount of electricity can be used in a home for only 0.0625 cents. Based on this fact, it makes sense to conclude that the most efficient way to power our device is electricity. This is the power source of most similar appliances, such as blenders, toasters, and air fryers, so it is logical to assume consumers would be prepared for this.

With all of this considered, the group has decided to use an outlet to run our device. As the most efficient, the most cost-effective, and the most consumer-friendly option, it made for a simple choice.

The CAD model in Figure 3.2.1 shows an electrical outlet designed on the back of Flavor Fusion.

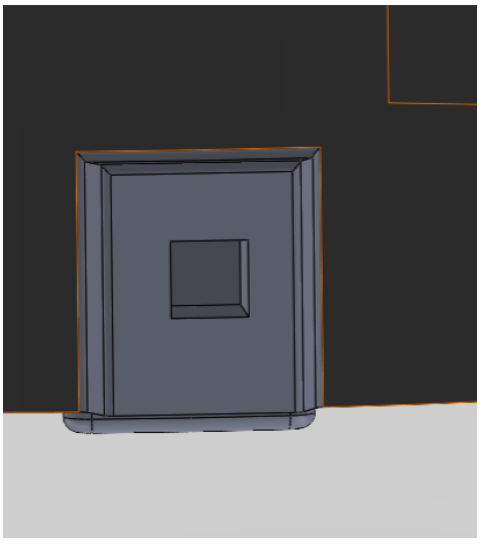


Figure 3.2.1 - Electrical Outlet in CAD Model

3.2.2 Heat Management

The heat management of our device would resemble that of a personal computer or a blender. They use fans to cool their motors. Both desktops and laptops may have more than one fan to keep them cool, while blenders house their fans in their base. These fans work by drawing cooler air from outside the device and expelling warm air from inside Flavor Fusion [25]. These computers often also use a heat sink. Heat sinks are a piece of metal that sits on top of a computer chip and draws power away from components by letting it rise through a series of fins. This could apply to our machine since we are using a processor to automate it.

A second possibility would be to mimic the toaster's heat management. Toasters are made of materials like plastic, metal, or other materials that are chosen for their heat-resistant properties [26]. This material choice works to keep the outside of the toaster cool enough to touch and would be enough to keep our device from overheating.

Nichrome Wire has extreme heat-resistant properties, and it is cheap. It typically costs around ten dollars for twenty-five grams of wiring [26]. These two facts are why it is used in toasters, hairdryers, storage heaters, and even industrial furnaces. These are also reasons why we could use it in our device.

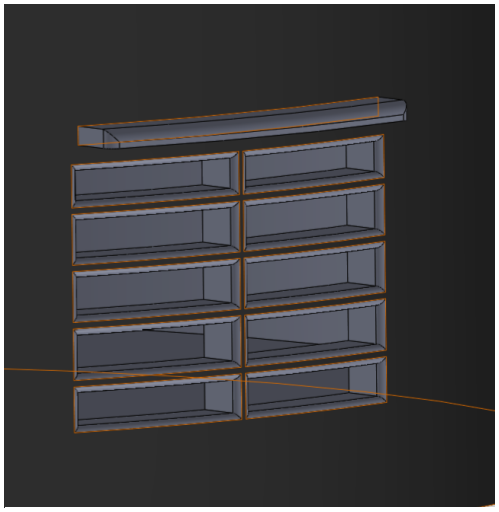


Figure 3.2.2 - CAD Design of Vent Cooling System

With all these options in mind, the group has created a CAD model with an open-air vent to cool our device. As seen in Figure 3.2.2, the CAD shows this air vent, and it also shows a "lip" above the vent that will allow for the device to be put up against a wall and still be able to allow airflow into the device.

3.2.3 Preservation of Spices

If spices harden or dry out, they are most likely not being properly stored. First, spices exposed to the air will begin to dry out and form a hard crust on the surface. Additionally, spices stored in an area with temperature fluctuations can also become hard. The temperature changes cause the moisture in the spices to expand and contract, which can lead to them becoming hard and brittle. Finally, spices that are exposed to light can also become hard because light can cause the moisture to evaporate, making them hard and crusty [27]. All of this means that we should have a cap on our spice capsules. They could be airproof, but this would possibly be overkill and could potentially ruin our ability to grind and pack spices inside of the capsule.

Another product that must be protected from the environment is film. When film is manufactured, it is put into a film can and sealed with tape. A film can is typically a circular box pressed from thin sheet metal, but plastic examples are also used [28]. Food-grade sheet metal fabrications can be turned into baking trays and restaurant equipment, so this proposes another potential material for our cartridges.

3.2.4 Cartridges

The proposed cartridges would have to be removable, machine washable, and storable. Ideally, the device would have extra cartridges that could hold extra spices not currently in the machine. The cartridges could potentially look something like this.

Figure 3.2.3 - Example of Cartridges that fit all requirements [47]

The problem of how to know when a cartridge is inserted within the app has a couple of potential solutions. The first is to use the example of a refrigerator. When the door of a refrigerator is opened, a light is turned on. This is done by a proximity sensor which detects if the appliance door is open or closed [29]. A second solution is to have no sensor. Allowing the machine to do the dispensing process while there is no cartridge inserted would not likely have any drawbacks.

Originally, we were considering putting a sensor into the cartridge. After further research, the group has decided that it is likely better to move forward with a feather-weight switch under the drop zone. This switch will measure the amount of time that spice is dropping to measure the amount of spice that has been loaded, and then stop when it reaches the required amount.

3.2.5 Packing Spices

The process of grinding and packing spices seems exceedingly difficult for this machine. First, there would be some qualifications and some questions. To effectively pack a small amount of spice would you simply crush it? This could lead to a piston-like design. Pistons work by inputting heat to the gas inside the cylinder, letting the gas expand, and then removing heat from the cylinder, and allowing the piston to push into the chamber [30]. This will not be effective because the heating and cooling would dry out the spices. The second potential way to pack and grind spices would be like how a herb grinder works. Herb grinders are cylindrical with two halves. These halves have sharp pegs aligned in a way so that when both halves are turned, the material inside is shredded [31]. This seems much more practical, and the only thing to figure out is how to automate the turning process. Due to this fact, this has been labeled as a "stretch goal."

3.2.6 Liners

For the problem of cleaning, a potential solution would be liners inside cartridges that can be easily removed and, potentially, disposed of. One good example of this would be air fryer liners. Air-fryer liners are typically made of parchment or silicone [32]. The parchment liners are disposable, and the silicone liners are reusable. Either of these materials would be usable for our project. The only question is how to dispense the spices from within the liner, and the simplest solution is to leave a hole in the bottom, like the way a funnel works. Production of these liners could be done by the team or outsourced. The parchment paper option would require only the cutting and shaping of parchment paper and would allow us to create enough liners for our prototype.

3.2.7 Max Amount of Spice

There is no set standard for the size of a spice jar. The most common size is two inches in diameter and five inches tall. For ease of dumping spices from their original containers to our cartridges, we should make the diameter of our cartridges much larger than two inches. The simplest other way to combat the issue of spilling would be to include a funnel. The device can have any number of slots to place cartridges, and outside of the large diameter requirement, there are not many more restrictions that can be placed on the cartridge dimensions. The current CAD model shows a cartridge about five inches in length, with a dispensing range of an additional two inches and diameter of 2 inches.

3.2.8 Creating a Rotating System with Wire

Running wiring through a rotating shaft can be troublesome. Normal wiring methods, like using a cable carrier, are not suitable for a rotating shaft because they can restrict the movement and cause damage to the wires. The easiest way to run wires through a rotating shaft is to use a slip ring. A slip ring is a device that allows the wires to rotate within the shaft without getting tangled or twisted. A slip ring is usually made of a stationary graphite brush which rubs on the outside of a rotating metal ring. As the metal ring turns, the electric current is conducted through the stationary brush to the metal ring, making the connection [33].

3.3 Dispensing Methods

3.3.1 Driving Hardware

The following section details various designs for dispensing powders. These designs are characterized by their driving hardware, but variations in spice transportation and functional concerns are also explored in sections “Delivery Method” and “Logistics,” respectively.

The first powder dispensing design will be referred to as the “dispensing disc.” It consists of a rotating disc with holes to collect and drop granular materials, such as in gumball machines [34] and laboratory pharmaceutical dispensers [35]. Opening sizes may be adjusted [34] or a range of interchangeable discs with different measurements may be used to allow different amounts of spice to enter the cavity before dispensing [35]. Figure 3.3.1 and Figure 3.3.2 display aerial and profile views of dispensing discs, respectively.

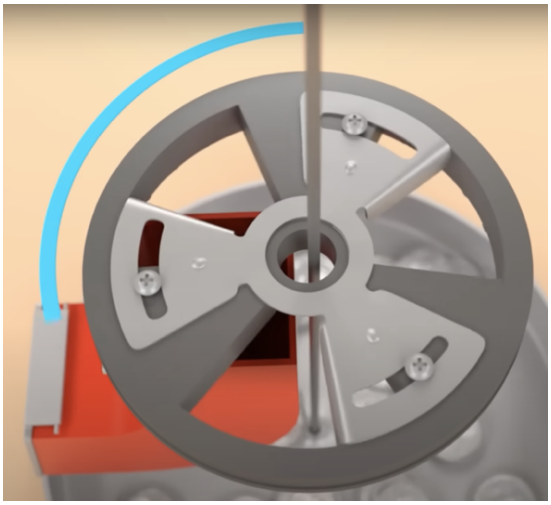


Figure 3.3.1 - Gumball Machine [56]

Discs are advantageous because of their simplicity and ability to access multiple cartridges for dispensing, while only needing one driving component, such as a motor. They can consistently deliver a preset amount of powder. Conversely, dispensing disc mechanisms are only capable of outputting one amount of powder at a time, so multiple discs would be necessary to vary output volume, with each calibrated to a certain hole size. The groove of each disc would be prone to spice contamination when dispensing multiple spices successively. For implementation in a spice mixer, a dispensing disc would likely be driven by a stepper motor with gears to transmit power.

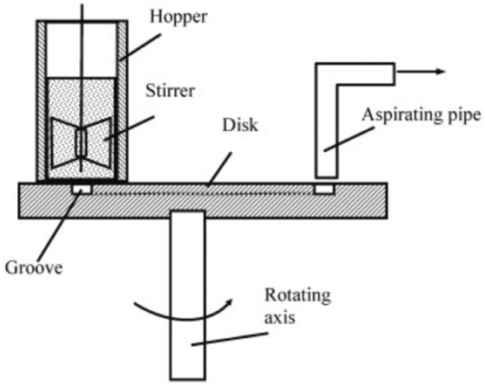


Figure 3.3.2 - Rotating Groove Dispenser [65]

The next mechanism is called an auger, which "uses a rotating helical screw blade, …usually within a tube, to move liquid or granular materials" [36]. As illustrated in Figure 3.3.3, the screw rotates, using frictional forces to displace powder, also seen in Spee-Dee's spice packaging machine [37].

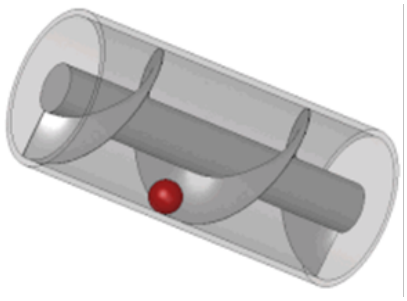


Figure 3.3.3 - Archimedes' Screw (Auger) [58]

Similar principles power watermills, which use rotating buckets to displace water rather than friction. An auger system is also utilized in a spice dispenser presented by Richard Gurberg, which uses a stepper motor along a rail to interface with spur gears and power the screw [38]. Augers allow for a continuous dispensing of powder rather than discrete volumetric amounts. They do not rely on gravity to move substances, but gravity can enhance performance. With respect to spices, the helical threads would assist in breaking apart clumping. In terms of disadvantages, augers can allow powders to fall even when stationary, so care should be taken to close dispensing exits. Rotation speed and screw capacity should be carefully calibrated to consistently deliver the desired amount of spice.

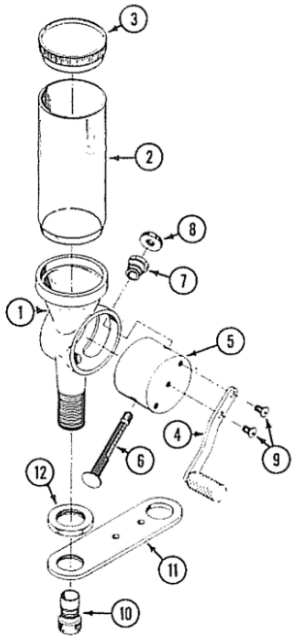


Figure 3.3.4 - Powder Measure Schematic [99]

The "chamber" method of delivering spices consists of displacing a cavity of specified volume to be filled with powder, then dispensing. There are various implementations of this principle, such as a "click lever" that can be pulled to dispense a teaspoon of spice [39]; powder measures, which "use gravity feed from a hopper of powder above an adjustable cavity in a rotating cylinder" [40]; or a spring-loaded covering that can be displaced to drop powder, similar to a Pez dispenser [41]. The rotating cylinder powder measure is most applicable for spice dispensing because its simple rotational motion can be powered by gears and a stepper or servo motor and its chamber size is adjustable, as seen in a video demonstration by YouTube user JunkfoodZombie [42]. An expanded parts view of this powder measure is shown in Figure 3.3.4. The powder measure model uses a simple, adjustable, consistent mechanism and is widely used, proving its

effectiveness. Contrarily, this approach would require separate rotating chambers for each spice container, and it would be difficult to adjust chamber size without the user dismantling much of the dispensing subsystem.

Table 3.1 - Pros and Cons for Dispensing

Mechanism	**Pros**	**Cons**
Dispensing Disc	Simple measurements with only one motor.	Contamination in groove, multiple discs necessary.
Auger	Continuous volumetric dispensing with only one motor. Breaks apart clumping.	Exit should be closed off. Screw and rotation speed should be tested and calibrated.
Chamber	Widely used for consistent measurements. Easy to power with servo or stepper motor.	Difficult to adjust delivery amount. Separate measures for each spice costs space.

3.3.2 Delivery Method

After the spice has been displaced or extracted with the dispensing mechanism, the next step in the dispensing process is to deliver it to the drop zone. The only feasible conveying methods demonstrated in all referenced machines are gravity, such as using chutes to carry spice from containers to the drop zone, and vibration, although the latter necessitates special considerations. Vibratory flow may be inconsistent for powders of different natures and may require operation in hundreds of Hz for proper spice delivery. Thus, vibrational mechanisms must demonstrate the ability to convey a range of spice granules without causing acoustic irritation or interference in a household setting. Other methods include electrostatic forces, Van der Waals forces, or fluid-assisted flow, but lie out of the scope of this project [43]. Aspiration would require the ability to create and manipulate partial vacuum and compressed air conditions and electrostatic conveying involves electrodes of high voltage and often depends on gas-assisted flow; for these reasons, such methods are typically implemented for manufacturing and pharmaceutical uses, where budgeting and resources are of less concern than efficiency and precision [43]. It should be noted that, while gravity and vibration will remain the principal

conveying methods, auger-assisted flow may be implemented to combat spice clumping and energize flow before delivery.

3.3.3 Logistics

While all presented dispensing techniques have strengths and weaknesses, the nature of applying this research to spices and cooking means certain considerations will favor some methods over others. Firstly, food contamination is an important concern for users with allergies as well as to ensure quality dispensing of the product. The only industry precedent identified to stop mixing during delivery was in the Coca-Cola Freestyle machine, which uses an air nozzle to separate constituents [44]. According to the Coca-Cola Company, technological development of the freestyle machine took “about two years,” of which the nozzle was a significant challenge [44]. Such a device is out of scope for this project, so additional solutions will have to be designed, such as a rotating funnel system that will ensure each spice will always be delivered via its respective chute. Grain size will affect the ability of mechanisms to accurately measure spice; dispensing discs in gumball machines fail to dispense smaller candies due to the size of their orifices. Similarly, dispensing designs intended for powders may fail when used to deliver large granules, such as whole peppercorns. For this reason, a grain size limit should be set for the machine and flow testing should be conducted during prototyping. Finally, clumping is a concern for machine performance. Common spices should be investigated for clumping properties to ensure they will not inhibit dispensing performance. For transparency with clients, these spices should be specified to guarantee machine performance under intended operating conditions.

3.3.4 Amount Dispensed

Various challenges exist when selecting a system to dispense desired amounts of spice. If users choose to mix a spice blend, the machine must use the total desired spice output as well as the ratio of constituents to calculate the individual mass or volume of each spice. This means that continuous dispensing is preferred over discrete because it can deliver any desired amount. Discussion continues in the section titled “Sensing.” While the United States uses volumetric measurement, most other countries use mass measurements for cooking, but may still use

volumetric measurements for small amounts of fluids and spices [45]. It is important to note that Australian, American, Canadian, and British volumetric measures (cups, teaspoons, etc.) can vary [46]. For these reasons, it would be most effective to implement a measuring system that can measure volumes of spice and communicate with a scale to sense dispensed mass. The following section, “Measurements” goes into more detail.

3.3.5 Sensing

Continuous dispensing delivers a constant stream of spice until the target goal is met. Based on the dispensing mechanisms introduced, it would have to be implemented with an auger. Examples of continuously dispensed powders may be seen in the TasteTro spice system [47], Spee-Dee spice packager [37], and FA Intellidropper automated gunpowder measure [48]. Discrete dispensing delivers spice in fixed-size amounts, such as dumping ¼ teaspoon at a time. Discrete methods include the chamber and disc methods of dispensing- simpler than the auger, but only capable of delivering one amount of spice at a time.

For users who wish to "freehand" spices without a set input amount, it would be desirable to have an option to dispense spices repeatedly until that goal is achieved. This could be accomplished with a setting that runs a motor indefinitely to continuously dispense spices until the user stops the process or by repeatedly delivering discrete amounts of spice.

3.3.6 Measurements

Volumetric measurements may be accomplished with an auger whose thread size and rotation rate correspond with volumetric flux or with discrete methods that deliver known amounts of material. For a user’s input, such as one tablespoon of spice, these methods would dispense the amount based on the physical properties of the dispensing mechanism. In contrast, mass measurements would necessitate a scale to acquire and send data to the machine to determine when to stop. This is easy to accomplish with continuous dispensing; for discrete methods, it is optimal to include multiple dispensing sizes. Larger amounts quickly deliver most of the desired spice, and a smaller measure will be used to accurately finish the job.

3.3.7 Drop Zone

Based on research in dispensing methods and amounts, initial options for machine features and hardware housing are presented. These options are intended solely to inform future decisions regarding Flavor Fusion's development and serve as recommendations rather than final solutions. The current section, "Drop Zone," discusses features of the machine's dispensing container, while the section "Spice Containers" is about spice cartridges for loading the machine, and "Maintenance Access" covers machine access for serviceability.

A bin to collect dropped spices will be necessary for the user. It should be removable with disposable liners or have a reusable material or coating to ensure repeated use without contamination. The drop zone should also be large enough to accommodate user-specific needs. If the user wants to fill their own vessel or measuring cup, they should have the room to do so.

3.3.8 Spice Containers

Spice containers should feature an inclined hopper-style design to prevent sticking and encourage the dispensing of spices due to gravity. These containers may be arranged radially for easy access with a rotating motor or linearly for access via a rail system. The number and capacity of each container should be considered to fit typical spice bottle contents without leading to a machine that takes up excessive space. Containers should be monitored via RFID or pressure sensing to alert users when a spice is not loaded. Additional monitoring may be investigated to measure spice levels, such as laser sensing used in ink printers [49]. Alternatively, the machine may be capable of tracking the amount of spice dispensed to alert users when spice levels are low.

3.3.9 Maintenance Access

The product should allow access to any internals that contact spices, such as chambers or chutes, for cleaning. Removable containers should allow access to spice for refilling, manual scooping, or other tasks. It may be necessary for users to remove spice containers to undo

clumping. Container locking mechanisms similar to ink cartridges in printers would allow for secure, removable containers.

3.4 Computer Science

3.4.1 Microcontrollers and Microprocessors

A microcontroller or microprocessor serves as the brain of the entire spice-making operation. A microcontroller is a compact integrated circuit on a single chip. A microprocessor is essentially a fully functional computer on a compact board. The main components of both a microcontroller and a microprocessor include a central processing unit (CPU) that acts as the brain, processing instructions and making decisions. There is also random-access memory (RAM) which stores data and instructions for the CPU to access quickly. Input/output pins allow the microcontroller to interact with other components of a circuit. In addition to pins, microprocessors may have Micro-USB ports, HDMI ports, and Micro-SD card slots. Arduino microcontroller boards do not usually have built-in video output capabilities. There are important differences between a microcontroller and a microprocessor. A microcontroller "integrates a CPU with peripherals and memory on one chip suited for embedded applications. A microprocessor contains just the computer engine and must be linked with other chips to interact with electronics" [50].

3.4.2 Arduino

Arduino is an open-source hardware and software platform designed for prototyping and creating interactive electronic projects. It consists of a microcontroller board, a development environment, and a community. Arduino has many types of microcontroller boards. The Arduino Uno is one of the most popular and widely used boards. It uses the ATmega328P microcontroller. The Arduino Uno has a range of different digital and analog pins. The Arduino Nano is a compact board based on the ATmega328 microcontroller. The Arduino Nano functions similarly to the Arduino Uno but has a smaller form factor. The Arduino Mega is designed for projects requiring more processing power and more I/O pins. The Arduino Mega uses the ATmega2560 microcontroller, which offers more memory and pins compared to the Arduino Uno or Arduino Nano [51]. There is also Arduino MKR, including the MKR Wi-Fi 1010, family of boards. Each board is equipped with a "radio module (except MKR Zero), that enables Wi-Fi, Bluetooth®, LoRa®, Sigfox, and NB-IoT communication. All boards in the family are based on

the Cortex-M0 32-bit SAMD21 low-power processor and are equipped with a crypto chip for secure communication" [51].

Table 3.2 - Arduino MKR Wi-Fi 1010 [52]

Pros	**Cons**
Compact form factor: Suitable for projects with space constraints.	Fewer GPIO Pins: Fewer GPIO pins compared to some other Arduino boards.
32-bit Microcontroller: The SAMD21 microcontroller provides more processing power compared to 8-bit microcontrollers.	Not a full microprocessor like Raspberry Pi – less power.
Low Power Consumption: Designed for power efficiency, making it suitable for battery-powered or low-power applications.	Slightly more expensive than non-Arduino controllers.
Operates at 3.3V	

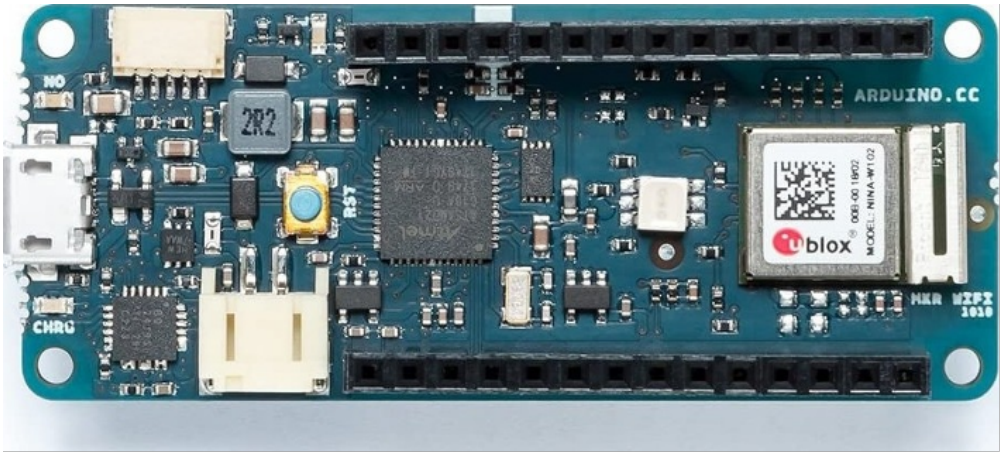


Figure 3.4.1 - Arduino MKR Wi-Fi 1010 [52]

3.4.3 Raspberry Pi

Raspberry Pi is a single-board computer developed by the Raspberry Pi Foundation. Raspberry Pi is defined as a "minicomputer the size of a credit card that is interoperable with any input and output hardware device like a monitor, a television, a mouse, or a keyboard –

effectively converting the set-up into a full-fledged PC at a low cost" [53]. The Raspberry Pi runs on a variety of Linux-based operating systems and is equipped with GPIO (General Purpose Input/Output) pins, enabling it to interact with peripherals. The Raspberry Pi Zero is the smallest and most affordable Raspberry Pi. The Raspberry Pi Zero features a single-core CPU, 512MB of RAM, and GPIO pins. The Raspberry Pi Model A is a budget-friendly option that offers similar features to the Model B but with reduced RAM and fewer ports. It usually includes a single USB port but lacks an Ethernet port. The Raspberry Pi Model B is one of the most popular and widely used Raspberry Pi models. It features a multi-core CPU, 1GB of RAM, multiple USB ports, Ethernet connectivity, HDMI output, and GPIO pins. This model provides ample processing power and connectivity options. The Raspberry Pi Compute Module is designed for industrial applications and embedded systems, offering a more compact form factor and enhanced flexibility compared to other models. It features a system-on-module design that integrates the core components of a Raspberry Pi into a smaller package. The Raspberry Pi 5 Model B is the most recent iteration of the Raspberry Pi. It uses a quad-core CPU, up to 8GB of RAM, multiple USB 3.0 and USB 2.0 ports, dual-band Wi-Fi, Gigabit Ethernet, dual 4K HDMI outputs, and GPIO pins [54].

Table 3.3 - Raspberry Pi Zero [55]

Pros	**Cons**
Single-core ARM11 processor, 512MB RAM, and built-in Wi-Fi and Bluetooth.	Harder to learn for beginners
Compact form factor: Suitable for projects with space constraints.	Could be more expensive when adding on modules, heat sync, etc.
Cheaper for a Raspberry Pi	
Operates at 3.3V but needs to be supplied with 5V for some peripherals.	

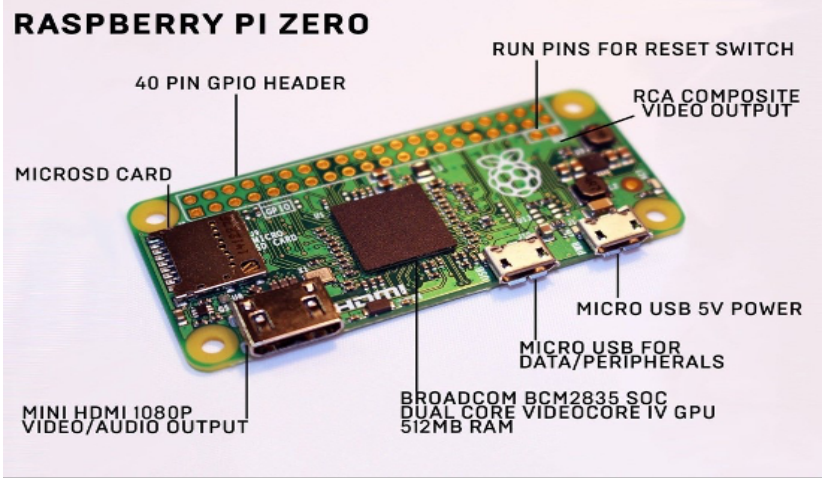


Figure 3.4.2 - Raspberry Pi Zero [77]

3.4.4 Motors

The spice maker will utilize motors to mix and distribute the spices. Integrating a motor driver in the device helps simplify the process of interfacing the microcontroller/microprocessor with the motor. The microcontroller/microprocessor communicates with the motor driver through pins. The motor driver manages the electrical currents and sequencing required to drive the motor. Motor drivers can be used to "control small motors like toys, remote-controlled cars, or robots. Also, you can use it to control larger motors in industrial applications" [56]. A motor driver protects the microcontroller from the high currents needed to run a motor. The main type of motor for this project is the stepper motor. Stepper motors operate by dividing a full rotation into a series of steps, allowing for accurate control of position and speed. Stepper motors usually have wires that connect to the corresponding pins on the board. Selecting a microcontroller or microprocessor will depend on how many pins are on the board (which will depend on how many motors are needed). In the case of having a lot of input and output,

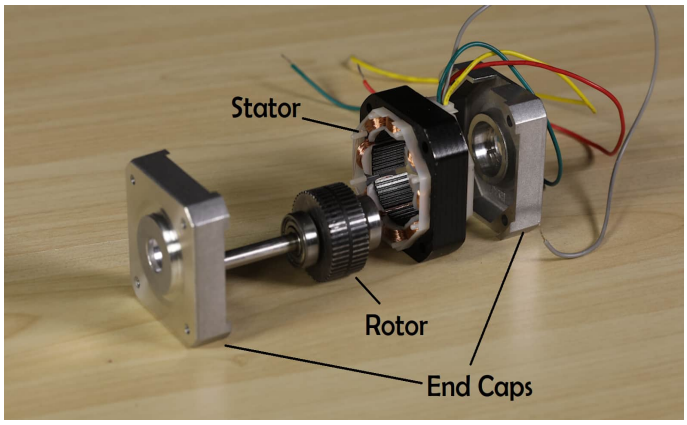


Figure 3.4.3 - Stepper Motor [92]

expanders could be added to the board. There are separate driver modules to handle the current and sequencing of the motor coils. For Arduino, the Arduino Stepper Library can be used for coding these motors. Commands like stepper.setSpeed() and stepper.step() can control the motor's speed and move it a specific number of steps [57].

3.4.5 Arduino Programming

The Arduino programming language is used to program Arduino boards to interact with sensors, actuators, and other peripherals/components. The Arduino programming language is based on C++ and is designed to be easy to learn. The Arduino Desktop IDE and the Arduino Web Editor can be used for development.

3.4.6 Mobile App Development

There are many different options for creating a mobile application. These include creating an iOS application or creating a multiplatform application. Creating an iOS application will allow for easier development and allow more time for focusing on features and wireless connectivity. Cross-platform will be a more complicated development. Cross-platform development is important for apps to reach the largest possible user base. Developers can create applications that operate on multiple operating systems, including the most popular operating systems, Android, and iOS. Cross-platform development allows for cost-effectiveness in a real corporate setting as developers can maintain one codebase, reducing the time and resources needed when maintaining two separate codebases.

3.4.7 Cross Platform Application

The Kotlin Multiplatform simplifies the development of cross-platform projects. With Kotlin Multiplatform, developers can build “cross-platform mobile applications that share code between Android and iOS projects to implement networking, data storage and data validation, analytics, computations, and other application logic” [58]. Kotlin/Native supports macOS, iOS, tvOS, watchOS, Linux, Windows (MinGW), Android NDK, and more. The compiler of

Kotlin/Native offers executables for diverse platforms, static or dynamic libraries with C headers tailored for C/C++ projects, and Apple frameworks tailored for Swift and Objective-C projects [58]. Kotlin/Native supports interoperability, including the use of existing libraries, encompassing static or dynamic C libraries, and C, Swift, and Objective-C frameworks. Another cross-platform option is Flutter. Another cross-platform option is Flutter, which is a UI toolkit developed by Google. Flutter allows developers to build natively compiled applications for mobile, web, and desktop from a single codebase. One of the key advantages of Flutter is its hot reload feature, which allows developers to quickly see the effects of code changes without having to restart the application [59]. This feature speeds up the development process. React Native is a cross-platform mobile development framework created by Facebook that allows developers to build native mobile applications using JavaScript and React. With React Native, developers can write code once and deploy it across both iOS and Android platforms. Kotlin, Flutter, and React Native will all allow the app to work cross-platform on iOS and Android.

3.4.8 iOS App

The other option for creating a mobile app would be to use Swift in Xcode. Swift was first introduced at Apple's Worldwide Developers Conference (WWDC) in 2014 as a modern alternative to Objective-C for iOS, macOS, watchOS, and tvOS app development [60]. Objective-C is a superset of the C programming language and was the primary programming language used by Apple for developing macOS and iOS applications before the introduction of Swift. Swift is supported by Apple's Xcode IDE, which provides a comprehensive set of development tools, including code editing, debugging, testing, and performance analysis features. The built-in debugger also enables real-time inspection of variables, breakpoints, and stack traces. Xcode also includes a built-in simulator that allows developers to test their apps on different devices and iOS versions without the need for physical hardware. We will be creating an iOS app because all the team members use iPhones.

3.4.9 Xcode Testing

Xcode includes built-in simulators to debug an app on a variety of different devices that a developer might not have immediate access to. Prior to building and running the application, a developer should choose a build scheme containing the target app. A scheme encompasses project particulars and configurations, guiding Xcode on how to execute the build and run processes for a project's product. The developer can manage real and simulated devices in the Devices and Simulators window in Xcode. To view this window, choose Window > Devices and Simulators. View and configure simulated devices from the Simulators tab. To add a new simulated device, click the plus (+) button at the bottom of the list of simulators and specify the configuration you want. It is possible to introduce new simulators to indicate a distinct device type or operating system version from the default options. To remove a simulator from the list, select it and then press Delete.

3.4.10 Wireless Connectivity

The mobile app can communicate with the microcontroller/microprocessor through Bluetooth. Bluetooth is a wireless technology standard used for exchanging data over short distances. For this project, Bluetooth will be employed to establish a communication link between the mobile app and the Arduino or Raspberry Pi device. Bluetooth modules like HC-05 or HC-06 can be used with Arduino and Raspberry Pi if the board does not have built-in Bluetooth. These modules are connected via the pins. For programming and communication, one can use programming languages like Python with libraries such as pybluez or pyserial. The ArduinoBLE Library can be used with Arduino to communicate with Bluetooth devices [61]. The users can enable Bluetooth on their phones and can connect to the device from their settings app. If Bluetooth is enabled on the phone, the phone will automatically connect to the spice maker. Another option for connecting the mobile app with the spice maker is Wi-Fi. Wi-Fi works similarly to Bluetooth but could face problems if the user is temporarily without network access. Users can connect to the spice maker through the app by selecting the device from a list of available Wi-Fi networks within their settings. A final method of wireless communication is UART, or Universal Asynchronous Receiver/Transmitter. UART enables bidirectional data transmission over two wires, typically labeled as TX (transmit) and RX (receive) [62]. UART

communication can be implemented using hardware UART ports available on microcontrollers or by emulating UART functionality using software libraries.

3.4.11 Wireless Connection

The mobile app can communicate with the microcontroller/microprocessor through Bluetooth. Bluetooth is a wireless technology standard used for exchanging data over short distances. For this project, Bluetooth will be employed to establish a communication link between the mobile app and the Arduino or Raspberry Pi device. Bluetooth modules like HC-05 or HC-06 can be used with Arduino and Raspberry Pi if the board does not have built-in Bluetooth. These modules are connected via the pins. For programming and communication, one can use programming languages like Python with libraries such as pybluez or pyserial. The ArduinoBLE Library can be used with Arduino to communicate with Bluetooth devices [61]. The users can enable Bluetooth on their phones and can connect to the device from their settings app. If Bluetooth is enabled on the phone, the phone will automatically connect to the spice maker. Another option for connecting the mobile app with the spice maker is Wi-Fi. Wi-Fi works similarly to Bluetooth but could face problems if the user is temporarily without network access. Users can connect to the spice maker through the app by selecting the device from a list of available Wi-Fi networks within their settings. A final method of wireless communication is UART, or Universal Asynchronous Receiver/Transmitter. UART enables bidirectional data transmission over two wires, typically labeled as TX (transmit) and RX (receive) [62]. UART communication can be implemented using hardware UART ports available on microcontrollers or by emulating UART functionality using software libraries.

3.4.12 Bluetooth

Bluetooth will most likely be the primary method of wireless communication between the Flavor Fusion device and the mobile application. There are two main types of Bluetooth, Bluetooth Classic and Bluetooth Low Energy (BLE). Bluetooth Classic or Bluetooth Basic Rate/Enhances Data rate is a “low power radio that streams data over 79 channels in the 2.4GHz unlicensed industrial, scientific, and medical (ISM) frequency band” [63]. Bluetooth Classic is

mostly used for wireless audio and is used most often in headphones and speakers. Bluetooth Low Energy (BLE) is used mostly for low-power operations. Bluetooth Low Energy transmits data over 40 channels in the 2.4GHz ISM frequency band. BLE allows developers the flexibility to create products with unique connectivity requirements. BLE also supports the creation of reliable, large-scale device networks. Bluetooth Low Energy is now also "widely used as a device positioning technology to address the increasing demand for high accuracy indoor location services. Bluetooth LE now includes features that enable one device to determine the presence, distance, and direction of another device" [63]. There are many benefits to using Bluetooth Low Energy over Classic Bluetooth. Bluetooth Low Energy is usually simpler and more cost-effective to implement compared to Classic Bluetooth. In addition, BLE has much quicker connection setup times compared to Classic Bluetooth. Since the device will only need to send data intermittently, BLE's intermittent data transmission capabilities will work with the mobile application. The microcontroller can quickly establish a connection, transmit the necessary data, and then go back to a low-power state. Finally, BLE is compatible with iPhone and Android which are the two main mobile devices. Although Bluetooth Classic was introduced with the very first iPhone back in 2007, BLE was introduced to the iPhone with the release of the iPhone 4S. Some advantages to using Bluetooth Classic include a longer range which could be helpful when connecting peripherals or linking devices across large rooms or a house. Bluetooth Classic will also work for this project, however, the user will have to "pair" their phone with the spice maker the same way they would "pair" their phone with a new set of headphones or speakers.

Some Arduino boards, such as the Arduino Nano 33 BLE series or the Arduino MKR Wi-Fi 1010, come with integrated BLE modules, allowing developers to incorporate Bluetooth communication into a project without the need for an external module. The HC-05 and HC-06 modules are Bluetooth Classic modules. They operate on Bluetooth 2.0 and use a Serial Port Profile (SPP) for communication.

HC-06 vs. HC-05: Specifications

HC-05	HC-06
Low operating voltage: 4V to 6V (Typically +5V)	Input voltage: 3.3~6V (Typically +5V)
Frequency: 2.45 GHz	Built-in antenna
Range: <100m	Operating Current: 30mA
Follows IEEE 802.15.1 standardized protocol	Uses Frequency-Hopping Spread Spectrum (FHSS)
Supported baud rates are 9600,19200,38400,57600,115200,230400 and 460800.	Works with Serial communication (USART) and TTL compatible
Can operate in Master, Slave, or Master/Slave mode	2.4GHz ISM band frequency
	Default baud rate: 9600

Figure 3.4.4 - Bluetooth Modules [64]

3.4.13 Data Storage

There are multiple options for storing data in this project. In the case of Flavor Fusion, there will be a recipe book that will store all the recipes for the spices. Storing data on an SD card in the microcontroller provides a convenient and portable means of storage with ample capacity for storing numerous recipes and associated data. The microcontroller can access the SD card through its built-in SD card interface, typically using SPI (Serial Peripheral Interface) or SDIO (Secure Digital Input Output) communication protocols. This allows the microcontroller to read and write data to the SD card, providing quick access to the recipe book whenever needed. The microcontroller could act as a server, serving the recipe data to the mobile app over Bluetooth. The mobile app can then send requests to the microcontroller, which retrieves the relevant recipe data from the SD card and sends it back to the app for display. The app itself could store the recipes in the app's internal file storage. Recipes could be stored as files within the app's file system, using readable formats such as JSON or XML. Utilizing an external database for storing recipe data offers a more scalable solution for storing recipe data. One common option for an external database is a cloud-based solution like Firebase Realtime Database or Google Cloud Firestore. These databases offer real-time synchronization across devices, enabling seamless access to recipe data from the mobile app. Additionally, they provide security features and scalability, ensuring that the app can handle many recipes and users without sacrificing performance. Implementing an external database for recipe storage also simplifies data management and updates.

3.4.14 Data Transfer

Once a user selects a recipe or specific spices to mix within the mobile application, the data needs to be transferred to the physical spice maker for implementation. In this scenario, where recipes are stored within the app's internal storage, the next step involves transferring this data to the spice maker. Through this Bluetooth connection, the selected recipe or spice combinations can be transmitted, enabling the Arduino program to interpret and execute the mixing instructions. This data transmission involves packaging the recipe information into a specific format that can be understood by both the application and the Arduino program. A commonly used format that can be understood by both ends is JSON (JavaScript Object Notation). JSON is a lightweight data interchange format that is easy for humans to read and write, and it is also easy for machines to parse and generate.

3.4.15 File Sharing

GitHub will be used for file sharing during the development process. GitHub is a widely used web-based platform for hosting and sharing code repositories. GitHub allows developers to store their code, track code changes, manage projects, and collaborate with other developers. GitHub uses Git, a version control system, to keep track of changes to code files over time. Developers can create repositories to store their code. Developers can also create branches to work on different features or fixes concurrently and can merge changes from their branch into the main codebase when ready. This project will most likely have two repositories. The first repository will store all the code used with the microcontroller. The second repository will store all the code related to the mobile application.

```
{
  "recipeName": "Taco Seasoning",
  "ingredients": [
    {
      "name": "Paprika",
      "quantity": "2 tsp"
    },
    {
      "name": "Chile Powder",
      "quantity": "2 tsp"
    },
    {
      "name": "Cayenne Pepper",
      "quantity": "1 tsp"
    },
    {
      "name": "Cumin",
      "quantity": "1 tsp"
    }
  ],
  "instructions": [
    "Add seasoning blend to taco meat."
  ]
}
```

Figure 3.4.5 - JSON Object

3.4.16 Instant Pot Pro Plus

The Instant Pot Pro Plus is a multi-cooker that allows the user to wirelessly operate the cooker through the Instant Brands Connect App. The user can select from one of the 800+ Smart Recipes and the app will wirelessly program the Instant Pot Pro Plus. This cooker has controls on the physical device and app controls [65]. As seen in Figure 3.4.6, the Instant Pot Pro Plus has a screen interface on the device that allows the user to utilize the device without the need for the mobile app. This feature is important in the case of having a dead phone or wireless connectivity issues. The Instant Pot has pre-set smart cooking programs. The programs are executed by a microprocessor chip inside the Instant Pot control panel.

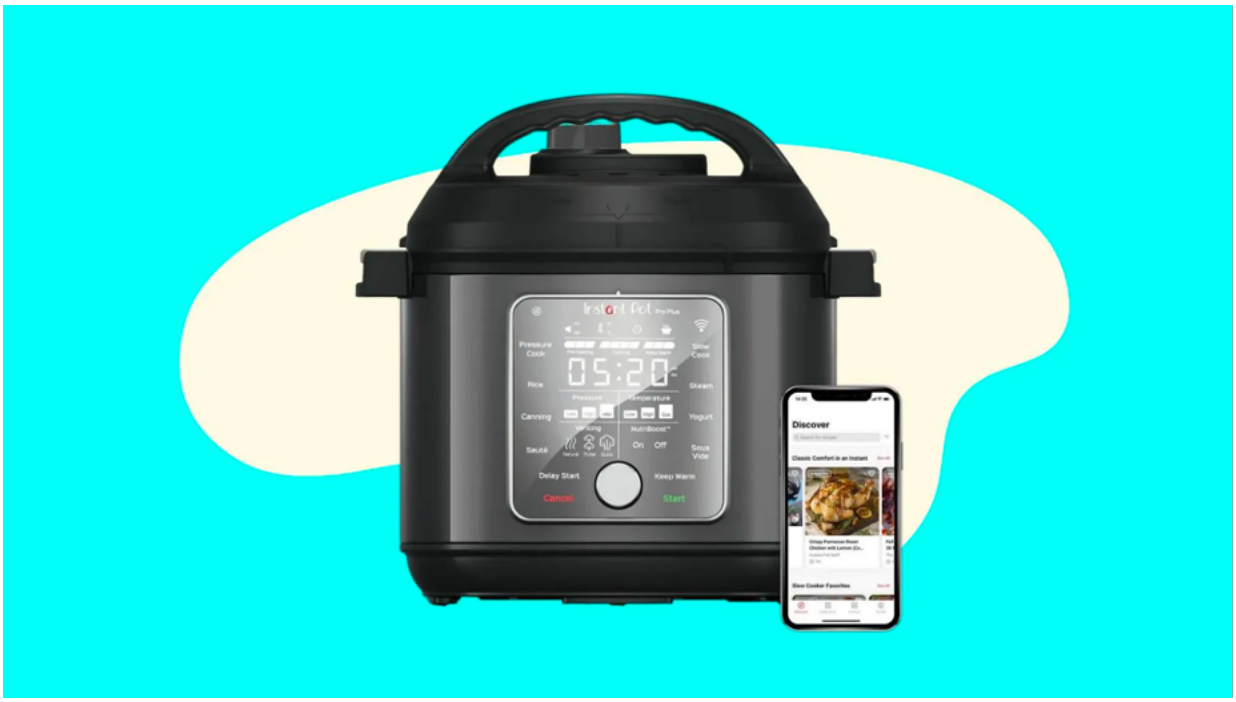


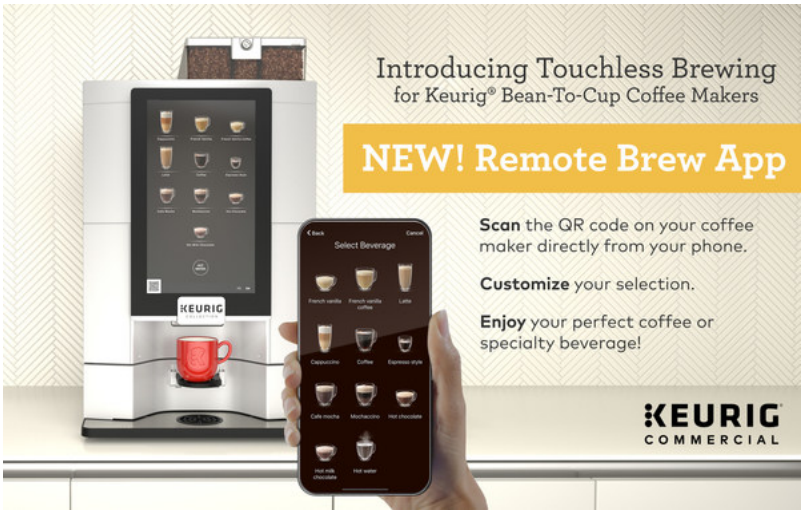
Figure 3.4.6 - Instant Pot Pro Plus [66]

3.4.17 Keurig App

Users can connect to their Keurig via an app. This goes over how to connect a Keurig to your home Wi-Fi: “The Keurig mobile app lets you place and manage your Keurig.com orders, including Auto-Delivery and SMART Auto-Delivery. It also offers innovative features to help enhance the Keurig K-Supreme Plus SMART brewing experience when connected to a home Wi-

Fi network. Access expert brewing recommendations using BrewID, customize your brew settings to your taste, save your favorite brew style, and more. You can download the Keurig® app and connect your Keurig K-Supreme Plus SMART coffee maker by scanning the QR code on the back of the brewer. The app is also available on the iTunes App Store and Google Play Store" [67]. Like the Instant Pot Pro Plus, the Keurig has an on-device interface that allows the user to use the device without the need for the mobile app.

Figure 3.4.7 - Keurig App [68]



4. Professional and Societal Considerations

4.1 At Home Impacts

The at-home benefits of Flavor Fusion are that it makes cooking fun and more enjoyable for beginners. Many people can’t figure out what they really want or don’t know what they have because of the cluttered spice cabinets. With Flavor Fusion, the at-home cooking experience becomes much easier. Flavor Fusion also allows you to do other tasks while cooking. The negative to the user experience is the overall price. Something like this is very expensive and would easily cost $400 or more so it does not fit within a lot of family's budgets. Users also may be hesitant to buy because Flavor Fusion is another appliance that they need to care for and store.

4.2 Economic Impacts

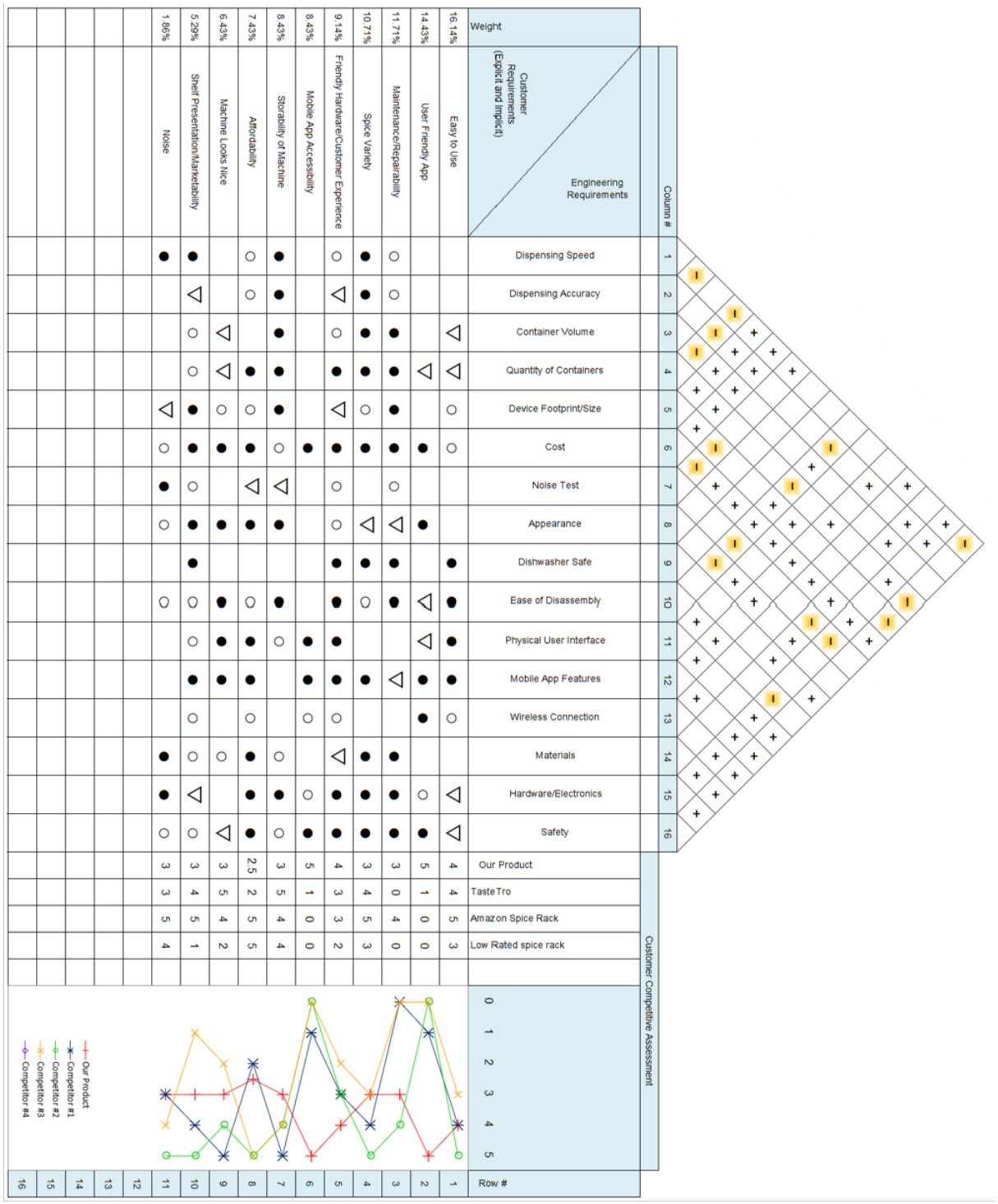
There are multiple economic impacts of Flavor Fusion. On the manufacturing side, a device like this would generate substantial work for manufacturers, as FDM printing isn't ideal for mass production; instead, injection molding would be the primary manufacturing method. For consumers, Flavor Fusion could help reduce food waste, putting more money in users' hands by optimizing spice usage. The device also has potential applications beyond the kitchen, such as shell loading for hobbyists or precise dosage measurements in the medical field for pills or powders. However, Flavor Fusion may face challenges in attracting investors and buyers. Many of its ideas originated from a Kickstarter project called TasteTro. While information on TasteTro is limited, it’s unclear if it will ever become a viable product. TasteTro required users to subscribe monthly and limited them to using only its proprietary spices, unlike Flavor Fusion, which allows users to incorporate their own spices without a subscription model. Despite these advantages, the retail market may not support a device priced over $400 as a spice rack replacement.

4.3 Environmental Impacts

Flavor Fusion has some negative environmental impacts. The device is another large producer of plastic waste which can be controlled by the company and the user but in the end, will most likely not. The prototype produced a lot of plastic waste from FDM 3D printing due to rapid prototyping from support materials and non-conforming parts. Flavor Fusion on a mass production scale could see less of this waste from a production standpoint but it still comes down to the user on how they dispose of the product.

5. System Requirements

5.1 Relationship Matrix



5.2 Engineering Requirements

Table 5.1 - Engineering Requirements

Req. #	Description	Target Value	Validation Method	Confidence
1	Machine can be stored in cabinet or cupboard	Target: 15” x 15” Range: 12” x 12” to 15” x 15”	Visual inspection and measurement	90% confident
2	Device should contain an adequate number of containers	At least 8 containers	Visual observation	90% confident
3	Each slot should be identifiable	Each slot should be labeled numerically and visible from outside device	Visual Observation	90% confident
4	Containers need to be adequately sized	Each container needs to hold at least 5 oz. of spice	Measurement via scale	90% confident
5	Containers need to be easy to load	Containers need to have a lid of 1.5” aperture to reduce need for funnel	Measurement via radius gauge or ruler	90% confident

6	Device should be aesthetically customizable	Offer at least 2 model colors	Visual validation of successful manufacturing in multiple colors	90% confident
7	Specialized screw for non-user accessible components	Use an uncommon screw type	Visual observation of disassembly attempt	90% confidence
8	Device features no sharp edges or corners	Fillet or chamfer of at least 2mm	Measurement via radius gauge, caliper, chalk transfer (to measure arc length), or stencil with 2mm internal fillet	90% confidence
9	Adequate Drop Zone volume	Must hold 8oz or 1 US Cup	Measurement via volume in CAD/Physical model.	90% confidence
10	Drop Zone must inform user of overfilling cup	Must warn user of overfill and split the job	Testing via code and hands on trials	90% confidence

11	User can navigate device and dispense spices without app	True	Physical testing of system to navigate UI and dispense spice using only display buttons	80% confident
12	Accurate Dispensing	Spices should be dispensed at ±1g or ±1/8 US teaspoon	Measurement of dispensed spice	80% confident
13	Easy to clean spice container	Withstand up to 100 washes	Inspect for smell or color leeching into material	80% confidence
14	Emergency stop button	Stop machine functions within 10 seconds	Time duration of button press to machine halt	80% confidence
15	Dispensing is appropriately quiet for kitchen	Maximum noise level of 70 dB	Measure via decibel meter (e.g. phone app)	80% confidence
16	Device can locally store user-specified recipes	At least 5 recipes are stored	Multiple trials: input recipe, verify correct LCD and spice output	80% confidence
17	Stop dispenser if Drop Zone	Stop machine functions within 10 seconds	Remove drop zone container to check stoppage	80% confidence

	container is removed			
18	Optimized air flow for electronics	Max temperature of individual internal components of $220^{\circ F}$	Temp sensor/ Thermal camera inside machine while in use	75% confidence
19	Ability to dispense a range of spices	Need to dispense a range of particle sizes from finely ground to very coarse / flaky	Multiple trials with different particle sizes	70% confident
20	User can continuously “freehand” dispense spice	At least 2 Tbsp of spice dispensed continuously	Physical testing and measurement	70% confident
21	Device should be distinguishable as a kitchen appliance	At least 75% of customers can identify device as a kitchen appliance	Survey a range of subjects on design recognition	70% confident
22	Prevents Cross Contamination	Spices should not mix until reaching the drop zone	Visual testing	70% confident

23	Simple disassembly for user access	Disassemble necessary parts under 15 mins	Visual observation of disassembly attempt	70% confidence
24	Dispense spices in reasonable amount of time	Up to 3 minutes per order	Time machine from Order Submission to Order Finish	70% confidence
25	Users are capable of navigating device interface (knob & button)	Users can perform desired task with 90% reliability, after instruction	Multiple trials with a range of human subjects acting as users	70% confidence
26	Device should be affordable	Production cost under $200	Budget tracking of prototyping expenses	65% confident
27	Device displays order queue and informs when in use	Information is displayed within 20 seconds of order reception	Visual confirmation and stopwatch	60% confidence
28	Device structurally withstands general use and interaction	Physical Testing: 100 cycles with no change in performance	Physical testing: loading/unloading containers, dispensing spice, pressing buttons	60% confidence

5.3 Computer Science Requirements

Table 5.2 - Computer Science Requirements

Req. #	**Description**	**Target Value**	**Validation Method**
CS.C.1.1	User can change text size in settings and see changes reflected in the app	True	Functional Testing
CS.C.1.2	User can use voice to text in the app	True	Functional Testing
CS.C.1.3	User can toggle dark and light mode and see changes reflected in the app	True	Functional Testing
CS.C.1.4	User can view how much spice is left in each container	True	Functional Testing
CS.C.1.5	User can receive a notification when a container is close to empty	True	Functional Testing

CS.C.1.6	User can input any name of a spice for each container	True	Functional Testing
CS.C.1.7	User can view/search through recipe book, add recipes, delete recipes, and edit recipes	True	Functional Testing
CS.C.1.8	User can modify number of servings per recipe	True	Functional Testing
CS.C.1.9	User can receive a notification when their order is complete	True	Functional Testing
CS.F.1.1	Mobile app must support text size adjustment	True	Functional Testing
CS.F.1.2	Mobile app must support an alternative input method such as voice commands	True	Functional Testing

CS.F.1.3	Mobile app must support dark mode and light mode	True	Functional Testing
CS.F.1.4	Mobile app must include spice inventory indicator for each container	True	Functional Testing
CS.F.1.5	Mobile app must send a notification when the container is almost empty	True	Functional Testing
CS.F.1.6	Mobile app must send a notification when the order is done dispensing	True	Functional Testing
CS.F.1.7	Mobile app must have input for custom spice names for each container	True	Functional Testing

CS.F.1.8	Mobile app must include a recipe book for internal storage with add, edit, delete, and search functionality	True	Functional Testing
CS.F.10	Mobile app must include a security function/user authentication system	True	Functional Testing
CS.F.11	Mobile app must have the ability to modify recipes for multiple servings	4 servings	Functional Testing
CS.F.12	Mobile app must have wireless connectivity	True	Functional Testing

5.4 Stretch Goals

Table 5.3 - Stretch Goals

Req. #	Description	Target Value	Validation Method	Confidence
S.1	Device features LEDs to indicate dispensing in progress and container in use	True	Visual verification of consistent performance	40% confident
S.2	Device features a mass sensor for feedback control of dispensing	Successful control with a margin of error of ± 1 gram	Testing and measurement of dispensed amount	40% confident
S.3	Removable containers can be used as spice shakers	True	Physical testing and verification	40% confident
S.4	Spice containers are air-sealed	True	Physical testing for water leakage	30% confident

S.5	Device features an automated grinding mechanism for dispensing	True for range of granules: star anise, fennel, peppercorns, cloves, and herbs	Physical testing. Grinding is performed for multiple spices with no loss of function	15% confident
S.6	Spice quantity is visible within containers	True	Visual verification	15% confident
S.7	Device plays audio upon order completion	True	Audial verification of consistent performance	5% confident
S.8	Ice Spice Promo Deal	Device is seen, approved, and endorsed by Ice Spice	Screenshot of Instagram DM	2.5% confident
S.CS.C.1	App features Digital Twin perspective	True	Functional Testing	50% confident
S.CS.C.2	Mobile App is cross-platform to support iOS and Android	True	Functional Testing	5% confident

6. Failure Modes and Effects Analysis

6.1 Scales Used

The scales used for sections 6.2 to 6.9 are in Table 6.1.A, while sections 6.10 through 6.12 use Table 6.1.B.

Table 6.1.A - FMEA Scale (Engineering)

Rating	Likelihood	Consequence	Detectability
1	100,000 uses	Negligible	Obvious
3	10,000 uses	Low $ loss	Light disassembly
5	1,000 uses	Med $, minor injury	Med disassembly, formal training
7	100 uses	Large $, med injury	Rigorous training
9	10 uses	Serious injury	Total disassembly
10	Every use	Death	Impossible

Table 6.1.B - FMEA Scale (Computer Science)

Rating	Likelihood	Consequence	Detectability
1	1% chance	Negligible	Obvious
3	10% chance	Slight Annoyance	Grandma
5	25% chance	Won't Take Long To Fix	Average User
7	50% chance	Fixable but Long Time	Your Average CS Major
9	75% chance	A Bad Day	Gotta Work in Tech
10	It is gonna happen for sure	Personal Loss	Impossible

6.2 Dispensing Mechanism

Primary Failure Mode	Secondary Failure Mode	Effect	Likelihood (1-10)	Consequence (1-10)	Hazard Score (L * C)	Detectability (1-10)	Risk Priority Number (L * C * D)	Mitigation Method
Acute Overload	Torquing	Part damage, explosion, cracks, cross-contamination, microplastics in spices	3	6	18	5	90	Test for defects, ensure safe speed, have high quality parts
Fatigue/Wear	Abrasion from Spices	Part damage, cross-contamination, microplastics in spices, inaccurate spice dispense	2	3	6	7	42	Test repeatedly before launch, ensure safe speed, have high quality parts
Fatigue/Wear	Crack Propogation	Part damage, increased stress, cross-contamination, microplastics in spices	3	7	21	2	42	Test for defects, ensure safe speed, have high quality parts
Manufacturing Errors	Material Defects	Part damage, explosion, cracks, microplastics in spices	3	6	18	7	126	Test for manufacturing defects, reorder parts if necessary
Manufacturing Errors	Layer Lines	Part damage, explosion, cracks, microplastics in spices	7	3	21	6	126	Test for manufacturing defects, reprint if necessary
Manufacturing Errors	Post Machining	Part damage, explosion, cracks, microplastics in spices	3	4	12	5	60	Test for manufacturing defects, reorder parts if necessary
Manufacturing Errors	Fillets not Correct	Part damage, cracks, cross-contamination, microplastics in spices	4	3	12	9	108	Test for manufacturing defects, reorder parts if necessary
Manufacturing Errors	Contamination (Food Safety)	User get sick	6	7	42	7	294	Clean thoroughly in between uses, make easy to clean
Incorrect Interfacing	User Error	Part damage, explosion, cracks, loss of function, potential injury	7	5	35	2	70	Provide instruction manual and warnings
Incorrect Interfacing	Teeth Wearing Out	Improper spice dispersion, potential rusting leading to user sickness, inability to grind if stretch goal is reached	3	3	9	5	45	Test repeatedly before launch, ensure safe speed, have high quality parts
Incorrect Interfacing	Electronic Failure	Improper spice dispersion, potential electronic damage	5	3	15	1	15	Test before launch, apply unusual loads to endure stability
Incorrect Interfacing	Gets Stuck	Inability to dispense spice, require maintenance	6	3	18	1	18	Use non rust-able parts, ensure lubrication
Wrong Spice	User Error/Placement	Wrong spice dispensed, potential contamination issues	6	3	18	2	36	Provide instruction manual and warnings
Wrong Spice	Signal Error	Inability to dispense spice	2	2	4	2	8	Test signal reception before release, allow a way for user to re-establish connection in the way that bluetooth in a car searches for a cell phone

6.3 Drop Zone

Primary Failure Mode	Secondary Failure Mode	Effect	Likelihood (1-10)	Consequence (1-10)	Hazard Score (L * C)	Detectability (1-10)	Risk Priority Number (L * C * D)	Mitigation Method
Contamination/Not Cleaned Properly	Incorrect Calibration	Improper spice dispension, loss of spices	5	2	10	2	20	Potentially allow for a way to recalibrate machine, or test calibration before release and then do not allow changes to it
Contamination/Not Cleaned Properly	Clogging	Inability to dispense spice, require maintenance	6	4	24	4	96	Store in a temperature controlled room without moisture to prevent spice degradation
Contamination/Not Cleaned Properly	Incorrect Sensing of Drop Zone	Inability to dispense spice, require maintenance	4	5	20	4	80	Test sensor, order new part if necessary
Locking Mechanism Failure	Stuck (if in when fails)	Inability to get to spices or clean device, disassembly required	3	3	9	4	36	Electrical lock powered with the machine, or a manual door like a microwave that pops shut, as opposed to locking
Locking Mechanism Failure	Can't Put In (if out when fails) (abstinence)	Inability to get to spices or clean device, disassembly required	3	3	9	4	36	Electrical lock powered with the machine, or a manual door like a microwave that pops shut, as opposed to locking

6.4 Spice Container

Primary Failure Mode	Secondary Failure Mode	Effect	Likelihood (1-10)	Consequence (1-10)	Hazard Score (L * C)	Detectability (1-10)	Risk Priority Number (L * C * D)	Mitigation Method
Acute Overload	Torquing by Motor	Part damage, explosion, cracks, cross-contamination, microplastics in spices	2	5	10	7	70	Test for defects, ensure safe speed, have high quality parts
Fatigue/Wear	Abrasion from Spices	Part damage, cross-contamination, microplastics in spices, inaccurate spice dispense	5	2	10	8	80	Test repeatedly before launch, ensure that spices inside of containers are not moving much, have non-scratch containers
Fatigue/Wear	Overtightening	Part damage, cross-contamination, microplastics in spices, inaccurate spice dispense	3	2	6	5	30	Have locking mechanism for containers that doesnt require screw-in
Fatigue/Wear	Crack Propogation	Part damage, increased stress, cross-contamination, microplastics in spices	4	6	24	2	48	Test for defects, provide manual for user on how to handle them, have crack resistant material for containers
Manufacturing Errors	Material Defects	Part damage, explosion, cracks, microplastics in spices	3	4	12	6	72	Test for manufacturing defects, reorder parts if necessary
Manufacturing Errors	Layer Lines	Part damage, explosion, cracks, microplastics in spices	7	3	21	4	84	Test for manufacturing defects, reorder parts if necessary
Manufacturing Errors	Post Machining	Part damage, explosion, cracks, microplastics in spices	3	3	9	3	27	Test for manufacturing defects, reorder parts if necessary
Manufacturing Errors	Fillets not Correct	Part damage, cracks, cross-contamination, microplastics in spices	4	4	16	6	96	Test for manufacturing defects, reorder parts if necessary
Manufacturing Errors	Contamination (Food Safety)	User get sick	4	7	28	7	196	Clean thoroughly inbetween uses, make easy to clean
Incorrect Interfacing	User Error	Part damage, loss of function, potential injury	5	5	25	2	50	Provide instruction manual and warnings
Incorrect Interfacing	Gets Stuck	Improper spice dispension, potential electronic damage	5	2	10	2	20	Test interface, allow for a reset mode
Wrong Spice	User Error/Placement	Improper spice dispension	6	5	30	4	120	UI design and matching numbers and or shapes for each slot and container
Wrong Spice	Signal Error	Does not function as intended	3	5	15	5	75	Better connections, double check all soldered componets
Cleaned Property	Incorrect Calibration	Improper spice dispension, Improper amount dispensed	5	8	40	7	280	Failsafes in code
Contamination/Not Cleaned Property	Clogging	Does not function as intended, part breakage	5	5	25	7	175	Failsafes in code, as a possible declogging feature
Environmental	Dishwasher/Cleaning	Container loses tolerance, Container breaks	4	8	32	7	224	Clear instructions on cleaning
Environmental	Spice Leeching	Contaimination, Smell,	5	7	35	5	175	Proper storage material

6.5 Housing

Primary Failure Mode	Secondary Failure Mode	Effect	Likelihood (1-10)	Consequence (1-10)	Hazard Score (L * C)	Detectability (1-10)	Risk Priority Number (L * C * D)	Mitigation Method
Environmental	Cat Knocks It Off Counter	Part Breakage	3	9	27	10	270	Get a new cat
Environmental	Smell	Stinky	3	4	12	6	72	Clean thoroughly inbetween uses, make easy to clean
Environmental	Proximity to Warm Environment	Increased part fatigue, Part warpage	6	6	36	2	72	Clear instructions on cleaning and placement
Environmental	UV Light	Increased part fatigue, Part warpage	5	6	30	2	60	Clear instruction on placement
Environmental	Transportation	Part Breakage	7	9	63	7	441	Handholds, give a lip for where to hold from
Manufacturing Errors	Material Defects	Incorrect tolrance, Premature Part breakage	5	5	25	5	125	Better manfacturing methods
Manufacturing Errors	Layer Lines	Incorrect tolrance, Premature Part breakage	5	5	25	5	125	Better manfacturing methods
Manufacturing Errors	Post Machining	Premature part breakage	5	5	25	5	125	Better manfacturing methods
Manufacturing Errors	Fillets not Correct	Premature part breakage	5	5	25	5	125	Better manfacturing methods
Fatigue/Wear	Weight of Spices	Premature part breakage	6	5	30	6	180	Clear instructions for use
Fatigue/Wear	Loosening of Lid/Components	Part Breakage	4	4	16	7	112	Better manfacturing methods
Fatigue/Wear	Vibrations	Loosen bearings	1	5	5	2	10	Stop the resonace

6.6 Physical Interface

Primary Failure Mode	Secondary Failure Mode	Effect	Likelihood (1-10)	Consequence (1-10)	Hazard Score (L * C)	Detectability (1-10)	Risk Priority Number (L * C * D)	Mitigation Method
Fatigue/Wear	Button is Stuck	Can't control physical interface	4	4	16	6	96	High life cycle buttons
Fatigue/Wear	Display is faded	Can't control physical interface	5	7	35	6	210	Clear instructions on placement, sun shade
Environmental	Cleaning chemicals	Increased part fatigue, Part warpage	7	9	63	7	441	Clear instructions on cleaning
Environmental	Food contaminations	Health Concern	6	7	42	6	252	Clear instructions, Clear loading instructions
Environmental	UV light exposure to display	Can't control physical interface	7	7	49	5	245	Sunshade
Environmental	Heat fluctuations	Burnt Screen	7	5	35	5	175	Clear placement instructions
Environmental	Damage to screen	Can't control physical interface	6	7	42	5	210	Easy to replace screen
User Error	Overload button	Can't control physical interface	2	2	4	5	20	Stop pressing the button
Manufacturing Errors	Soldering	Shorting, Does not work as intended	5	5	25	8	200	Better manfacturing methods
Manufacturing Errors	Clipped wires	Shorting, Does not work as intended	5	5	25	8	200	Better manfacturing methods
Signal failure	Disconnected from board	Inability to dispense spice	2	3	6	3	18	Inspection before assembly

6.7 Electronics

Primary Failure Mode	Secondary Failure Mode	Effect	Likelihood (1-10)	Consequence (1-10)	Hazard Score (L * C)	Detectability (1-10)	Risk Priority Number (L * C * D)	Mitigation Method
Acute Overload	Torquing	Part damage, explosion, cracks, cross-contamination, microplastics in spices	3	6	18	9	162	Test for defects, ensure safe speed, have high quality parts
Fatigue/Wear	Abrasion from Spices	Part damage, cross-contamination, microplastics in spices, inaccurate spice dispense	3	6	18	8	144	Test repeatedly before launch, ensure safe speed, have high quality parts
Fatigue/Wear	Crack Propogation	Part damage, increased stress, cross-contamination, microplastics in spices	4	7	28	8	224	Test for defects, ensure safe speed, have high quality parts
Incorrect Interfacing	Wires disconnect	Inability to dispense spice, loss of function	3	2	6	2	12	Inspection before assembly
Incorrect Interfacing	Motor Wearing Out	Part Damage, loss of function	4	3	12	3	36	Purchase motors from reputable vendor and provide easy replacement guide
Incorrect Interfacing	Electronic Failure	Part damage, loss of function, potential injury, electrocution	2	7	14	3	42	Rigorous testing before installation/assembly, would purchase from reputable retailer
Incorrect Interfacing	Tangled Wires	Loss of function, shorting	1	2	2	3	6	Create guides in the structure of the system and use just enough wire to not have any loose ends
Incorrect Interfacing	Incorrect Grounding	Part damage, loss of function, potential injury	1	6	6	5	30	Rigorous testing before installation/assembly, would purchase from reputable retailer
Incorrect Interfacing	Gets Stuck	Loss of function	5	3	15	2	30	Easy to reset the system without disassembly
Power Surge	Fire	Part damage, loss of function, potential injury	1	9	9	1	9	Use heat resistant materials
Environmental	Humidity	Part damage (rust, short circuiting), loss of function	5	3	15	5	75	Use moisture resistant materials and protections
User Error	User fucks with stuff	Part damage, electrocution	3	5	15	3	45	Restrict access to important electronics via different screw types, or glue components into place

6.8 Arduino

Primary Failure Mode	Secondary Failure Mode	Effect	Likelihood (1-10)	Consequence (1-10)	Hazard Score (L * C)	Detectability (1-10)	Risk Priority Number (L * C * D)	Mitigation Method
Manufacturing Errors	Bad Board	Shorting. Does not work as intended	2	2	4	3	12	Rigorous testing before installation/assembly, would purchase from reputable retailer
Manufacturing Errors	Bad Servo	Shorting. Does not work as intended	2	2	4	3	12	Rigorous testing before installation/assembly, would purchase from reputable retailer
Environmental	Overheat	Part damage/ replacement, fire hazard	3	4	12	7	84	Add fan(s)
Environmental	Humidity	Part damage (rust, short circuiting), loss of function	2	3	6	6	36	Use indoors with AC, keep away from stove for prolonged time. Insulate/ coat electronics
Environmental	Shock due to Shaking	Part damage, loss of function	5	3	15	5	75	Cushion internal hardware, reduce open space for parts to fly around, use fasteners
Environmental	Power Surge	Part damage/ replacement, fire hazard	2	4	8	8	64	Surge protectors, GFCI outlets
Internal Memory	Storage Failure	Temporary loss of function/connection, user data loss	6	2	12	4	48	Ensure best possible connection with secure/ soldered wires, reliable BLE signal. Can't really control hardware hiccups on the MCU
Internal Memory	RAM	Device crashes (could affect dispensing or bluetooth)	3	1	3	2	6	Monitor memory and shut down or clear space when overloaded (flush sensor data, cap request queue, etc.)
Internal Memory	SD Card	Recipes are gone :(	2	2	4	5	20	Test for manufacturing defects. ensure secure connection and physical support, restrict access to reduce physical wear
Programming Bug	Incorrect spice dispensed (wrong function called)	$ loss, contamination	4	3	12	9	108	Bug testing, perhaps add redundant features for robustness?

6.9 Power System

Primary Failure Mode	Secondary Failure Mode	Effect	Likelihood (1-10)	Consequence (1-10)	Hazard Score (L * C)	Detectability (1-10)	Risk Priority Number (L * C * D)	Mitigation Method
Batteries explode	Chemical contamination (or warfare, if you're Alex)	Fire hazard, part damage/ replacement, small chemical burns	1	5	5	2	10	Use common, accessible batteries and warn user when power is low
Environmental	Overheat	Part damage/replacement, fire hazard	1	4	4	3	12	Use common cord and plug types, avoid tangled excess wire or offer someplace to secure it. Give batteries heat sinks or ventilation.
Environmental	Humidity	Part damage (rust, short circuiting), loss of function	2	3	6	4	24	Use insulated cords, seal battery compartments
Environmental	Power Surge	Part damage/replacement, fire hazard	3	4	12	5	60	Surge protectors, GFCI outlets
Manufacturing Errors	Incorrect Grounding	Part damage/replacement, fire hazard	1	4	4	9	36	Test hardware before use
Manufacturing Errors	Soldering	Part damage/replacement, fire hazard	2	3	6	9	54	Test hardware before use

6.10 Security

Primary Failure Mode	Secondary Failure Mode	Effect	Likelihood (1-10)	Consequence (1-10)	Hazard Score (L * C)	Detectability (1-10)	Risk Priority Number (L * C * D)	Mitigation Method
Storage	Remote access to internal storage	User data loss, lost connection	2	2	4	9	36	PIN authentication required for BLE connection. iPhone app very secure
User Error	Access to internals	Part damage, potential minor injury (moving parts)	7	4	28	5	140	Secure internals in separate compartments with specialized fasteners. Include warnings to user about unintended access.
Leaked Passcode	Somebody finds out your passcode	data leak	7	7	49	3	147	Change your passcode
Phone Hacker	Hacking through WiFi	data leak	3	7	21	9	189	Uninstall and reinstall app

6.11 Bluetooth Connections

Primary Failure Mode	Secondary Failure Mode	Effect	Likelihood (1-10)	Consequence (1-10)	Hazard Score (L * C)	Detectability (1-10)	Risk Priority Number (L * C * D)	Mitigation Method
Bluetooth Disconnection	Physically too far from device	Can't mix spices from phone	10	3	30	3	90	Walk closer to device
Phone No Power	Phone dies	Can't mix spices from phone	10	3	30	1	30	Charge your phone

6.12 Queue Overloading

Primary Failure Mode	Secondary Failure Mode	Effect	Likelihood (1-10)	Consequence (1-10)	Hazard Score (L * C)	Detectability (1-10)	Risk Priority Number (L * C * D)	Mitigation Method
Too many items from phone	Not enough storage in Arduino	unable to mix spices, app crashes	6	5	30	5	150	Clear queue on phone
Too many items from physical device	Not enough storage in Arduino	unable to mix spices, device crashes	6	7	42	5	210	Manually prevent overflow in the codebase
Performance Issues	Loading delays	Increased time waiting for spice mixture	6	5	30	6	180	Message informs user what number in the queue their mixture is

7. Concept Development

7.1 Computer Science Concept Visualization

7.1.1 Mobile App User Experience

When developing the user experience of Flavor Fusion, the team prioritized a visual-heavy interface that was user-friendly and intuitive. As such, the home screen displays a very simple UI that emphasizes the spices in the machine and the capability to blend right away. The goal was to go from opening the app to dispensing in as few clicks as possible. The team used existing Swift UI elements to make for an easy coding and debugging experience, as well as a consistent and efficient user experience. Examples include the iOS Contact sheet layout for displaying the individual spices and blends, or the native layers that give the interface depth. Stretch goals were also implemented, such as the individual spice progress bars on the

“Blending” screen. The user journey is simple and straightforward, something we desire for to make Flavor Fusion a device anyone can pick up and immediately understand.

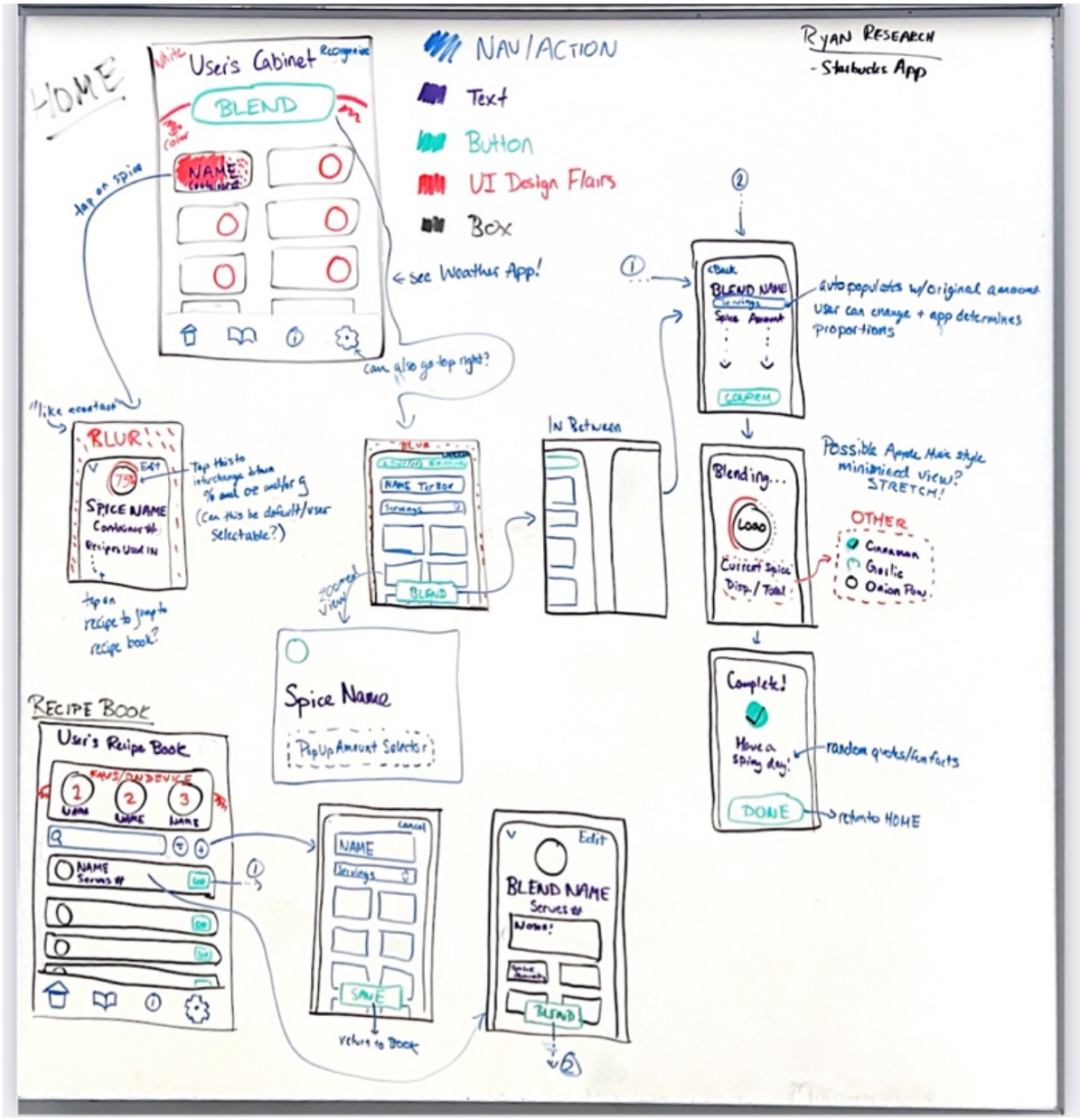


Figure 7.1.1 - App User Experience

7.1.2 App Landing Page

The Flavor Fusion mobile app will be created for iOS devices. The UI will consist of multiple views that contain different features. The main page of the app will have a list or ghost view of all the spices. A ghost view in the Flavor Fusion mobile app would replicate the appearance of the physical spice maker, providing users with a familiar interface that mimics the layout of their spices. This visual representation allows users to easily locate and interact with spices in the app, enhancing their navigation and overall user experience. The ghost view is a stretch feature and will be created time-permitting. Without implementing a ghost view, the app would feature a basic list interface displaying all the spices. The main landing page view will have a virtual "cabinet" of spices organized into two columns. Each spice box overlays a rectangle with variable opacity based on the amount of spice in the container. Each element of the list displays information about a spice including name and container number. Tapping on a spice triggers a popup view. The popup displays detailed information about the spice including name and container number. It also includes a visual indicator of the spice. Additionally, the user can view all the recipes in their recipe book that contain that spice.

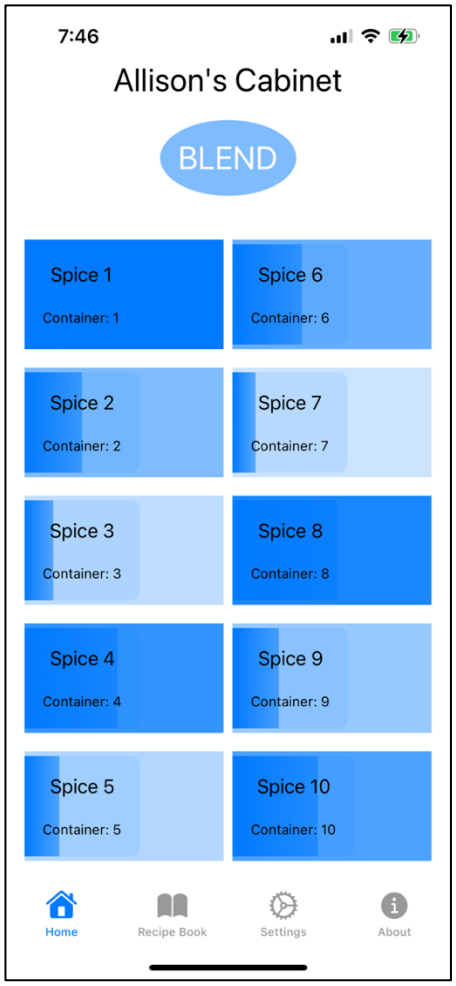


Figure 7.1.2 - App Landing Page

7.1.3 App Navigation

The Flavor Fusion mobile app will have a tab-style navigation interface. Each tab corresponds to a different view: the "Home" tab contains the list of spices (NavigationView with a List), the "Recipe Book" tab displays the recipe book with its corresponding features, the “Settings” tab allows the user to customize some settings in the app, and the "About" tab provides information such as privacy information and a user manual. Each tab is associated with an icon and a label displayed at the bottom of the screen, which makes it easy for users to identify and switch between sections of the app.

Figure 7.1.3 - App Navigation.

7.1.4 Passcode Creation

When users first download and launch the Flavor Fusion app they are prompted to create a passcode. This passcode will protect the user’s spice and recipe information if someone else was using their mobile device. Initially, the user is presented with a message with the prompt "Create a Passcode." This screen includes a grid of buttons with numbers 1 through 9. There is also a "Delete" button which allows the user to erase the last entered digit if needed. Below this grid, there is a text field where the passcode is displayed as the user inputs it. The user interacts with the UI by tapping on the numbered buttons to input their desired passcode. Each time a button is tapped, the corresponding number is appended to the passcode string. If the passcode entered by the user is less than four characters, an error message is displayed indicating that the passcode must be at least four characters long. Once the user enters a passcode that meets the minimum length requirement, they can tap the "Create Passcode" button. This triggers a function that saves the password to UserDefaults, Apple’s interface to the user's defaults database (Appendix **Error! Reference source not found.**). Additionally, a boolean flag

Figure 7.1.4 - Create Passcode

passcodeCreated is set to true, indicating that the passcode has been successfully created. Upon successful creation of the passcode, the user is navigated to main landing page of the app.

7.1.5 Login

Like creating a passcode, the login view has a screen that prompts the user "Enter Passcode." This screen includes a grid of buttons representing numbers 1 through 9, along with a "Delete" button for erasing the last entered digit. Below this grid, there's a secure text field that shows the user typing in their passcode. There is a "Login" button to initiate the login process. Like the passcode creation screen, the user interacts with the UI by tapping on the numbered buttons to input their passcode. Upon tapping the "Login" button, the login() function is triggered. In this function, the stored passcode is retrieved from UserDefaults. If the passcode is incorrect, an alert appears on the screening informing the user that their passcode is incorrect. The user can then type in a new passcode.

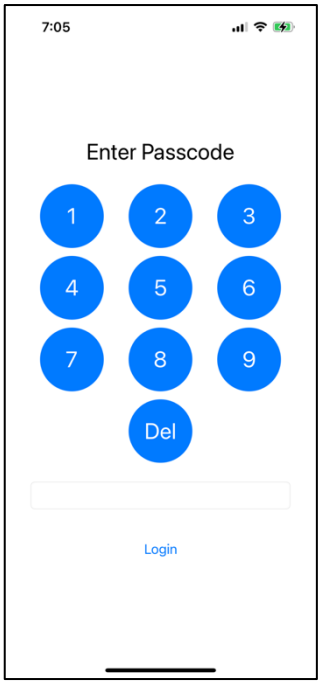


Figure 7.1.5 - Login

7.1.6 Recipe Book

The Recipe Book view in the Flavor Fusion app is as a collection of spice mixes. Within the app, users can browse through their list of recipes. By tapping on a recipe, users can view its details via a navigation link, presenting a dedicated view showcasing the recipe's ingredients and instructions. The app also enables users to add new recipes through a modal sheet, where they can input the recipe details and save them to the book. The AddRecipeView in the Flavor Fusion app facilitates the process of adding a new recipe to the recipe book. There is a form interface where they can input the details of the recipe, including the recipe name and ingredients. Upon completion, users can tap the "Add Recipe" button to add the newly created recipe to the recipes array and update the recipe book. Also, users can cancel the addition process by tapping a "Cancel" button, which also dismisses the modal sheet without saving any changes.

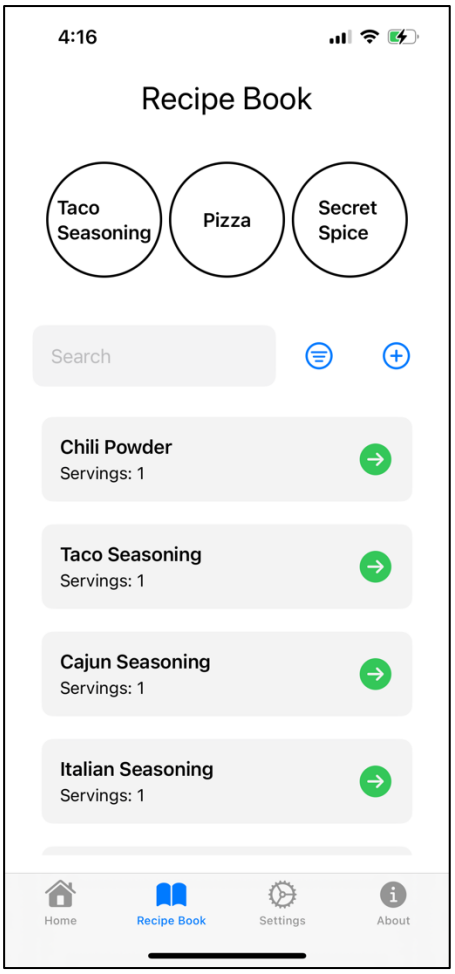


Figure 7.1.6 - Recipe Book

7.1.7 Settings

A settings page on a mobile app usually contains various options for users to customize their experience and manage their account. One of the features includes changing the app passcode for security purposes. Users could also find options to personalize app notifications. Notifications can be used for users to know when their spice containers are running low and when their spices are done mixing. Additionally, settings pages often include options to toggle between light and dark modes for user interface customization.

7.1.8 About

The About view in the Flavor Fusion app serves as an information hub for the user. In the about page, users will see a list of sections including "Project Overview," "User Manual," "Privacy Information," and "Meet the Team". Each section takes users to more detailed information when they tap on it. The Flavor Fusion user manual will contain explanations and diagrams to help the user setup their machine. The project overview gives an overview of the senior design project. The Privacy Information page informs users about the app's handling of Bluetooth data and privacy measures. A privacy page serves to communicate to users how their personal information is collected, used, and protected within the app. It includes details about data collection practices, data usage purposes, security measures implemented to safeguard user data, and any relevant privacy policies or terms of service. Finally, the Team page introduces the Flavor Fusion team and their roles.

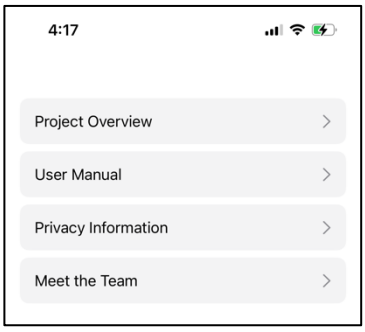


Figure 7.1.7 - About

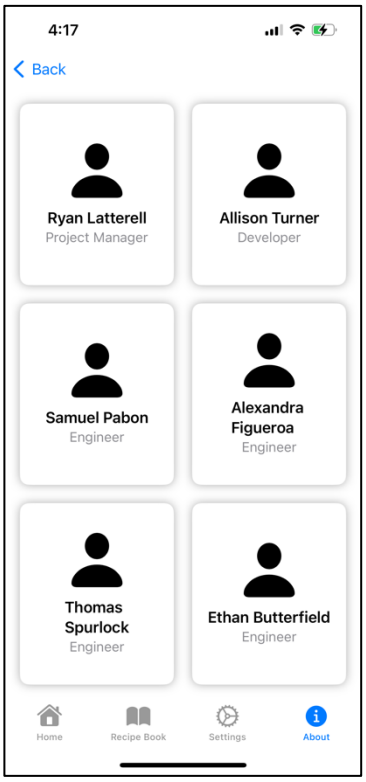


Figure 7.1.8 - Meet the Team

7.1.9 Establishing Bluetooth Connection

The Arduino code establishes a Bluetooth Low Energy (BLE) connection using the ArduinoBLE library. The Arduino advertises its services using a specific Universally Unique Identifier (UUID). When a central device, such as an iPhone, scans and finds the advertised service, it can establish a connection with the Arduino. The Arduino waits for incoming connections in a continuous loop. It then enters a loop to handle communication. If data is received, it processes the command and performs corresponding actions. The connection remains active until the central device disconnects. The BluetoothManager class facilitates Bluetooth Low Energy (BLE) communication within a Swift application. The class manages the central

role in BLE communication, utilizing instances of CBCentralManager and CBPeripheral for scanning, connecting to, and interacting with peripheral devices.

7.1.10 LED Circuit Example

The code sets up a Bluetooth Low Energy (BLE) peripheral device using the ArduinoBLE library. It initializes BLE communication, defines a service and a characteristic with specific UUIDs, and advertises the device's presence for connections. Once data is received from a mobile app, the Arduino code interprets the command (either "ON" or "OFF") and controls an LED connected to pin 6 on the board. The code loops to continuously check for connections and process incoming data, effectively allowing remote control of the LED via Bluetooth. The code defines a SwiftUI app that communicates with a Bluetooth Low Energy (BLE) device. The BLEManager class utilizes CoreBluetooth to handle central role tasks such as scanning for peripherals, connecting to them, and exchanging data. It includes functions to manage the discovery of services and characteristics on the peripheral device, as well as sending commands to the Arduino board. The UI displays the connection status, the name of the Arduino device, and provides a button to toggle an LED connected to the Arduino. A stepper motor can be controlled similarly through Bluetooth.

7.2 Dispensing Mechanism Visualization

Since the overall idea of implementing a singular device; one to contain and dispense species was no longer an idea due to difficulty in manufacturing and cost, our team implemented a new two device theory. The first part being the Spice container and second part being the main Dispenser. As seen in Figure 7.2.1with a simple screw on dispense attached to the spice container we theorized to apply an internal auger. This auger would be measuring out a pre-determined amount of spice with each rotation from the stepper motor. Not only did we ensure to prevent cross contamination till reaching the Drop Zone but the spices where conveniently placed in a normal spice bottle perfect for easily adding custom spice mixes. When comparing against TasteTro, a refill would cost users "$12 dollars" [2] for a replacement and the old cartridge would be tossed into disposal. But with Flavor Fusions attached dispensers' users could easily refill the spice bottles freely, which helps cut down on waste. All in all, we hope to create the Keurig equivalent of a spice dispenser. With hundreds of different spice combinations users could drop, twist, and dispense; having the ability to switch up spice combinations when needed.

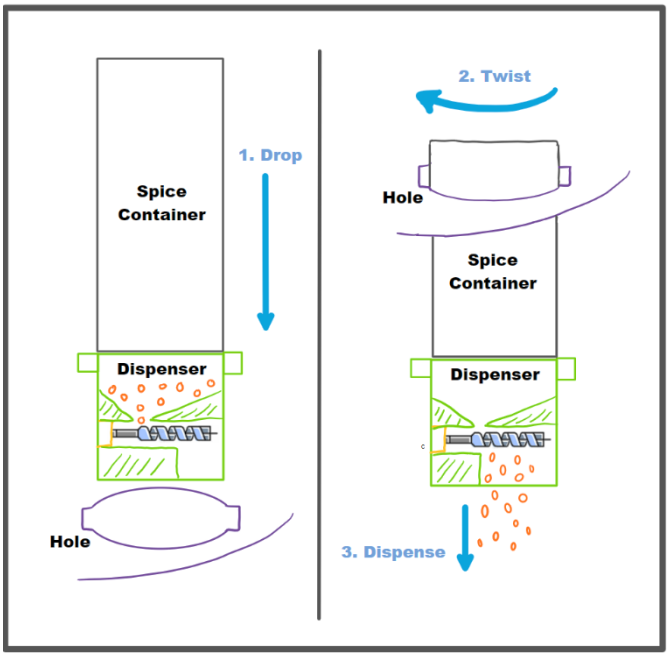


Figure 7.2.1 - Dispenser Mechanism

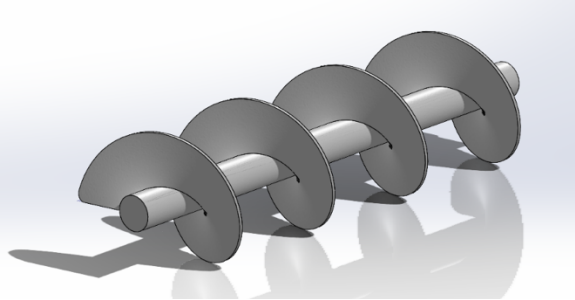


Figure 7.2.2 - CAD Model of Auger

Presented in Figure 7.2.2 is a 3D modeled auger, which was selected as the mechanism for dispensing spice. The auger works by pushing spice as it rotates, taking advantage of frictional forces to counteract gravity and break apart spice clumps. According to factors such as thread diameter, thread pitch, or direction of rotation,

dispensing performance can be altered and tested. By driving the auger with a stepper motor, rotational position and velocity can be controlled, resulting in an amount and speed of spice delivery that can be carefully tuned.

7.3 Circuit and Electronics Visualization

TinkerCAD was used to make basic models of electric circuits, components, and wiring schemes. The following figures serve as visualizations of circuits for stepper motors, analog sensors, LEDs, and LCD screens, respectively.

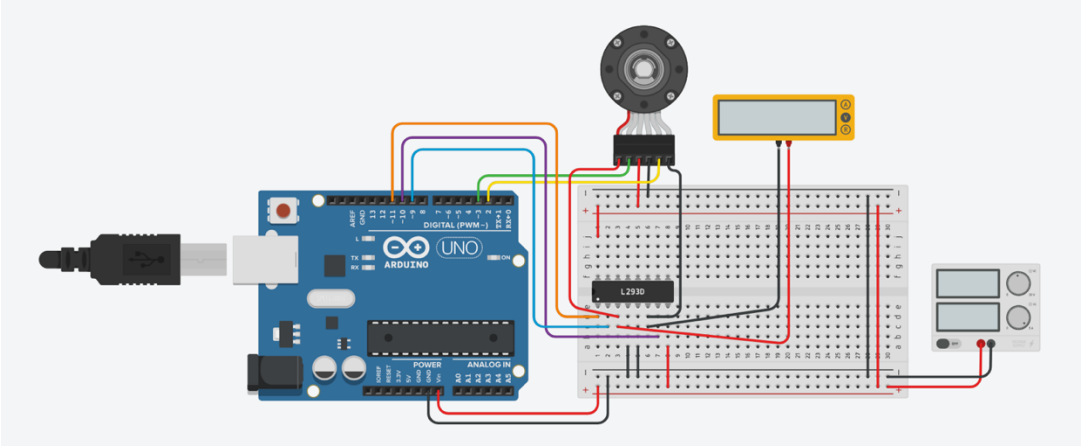


Figure 7.3.1 - Stepper Motor Circuit

Due to limitations in TinkerCAD's component library, a DC motor with an encoder was substituted for a stepper motor in Figure 7.3.1. With a stepper motor, the circuit would differ slightly- 5 wires connect the stepper motor to a driver, and the driver takes 6 input wires from the Arduino: power, ground, and 4 digital pins. With the use of a RAMPS board, stepper motor drivers and power would be centralized. This means the RAMPS shield plugs directly into an Arduino Mega and receives 12 V power with at least 5 amps, then stepper motors plug directly into the drivers on the RAMPS shield [69]. The RAMPS shield would also necessitate firmware such as Marlin for connection to the microcontroller [70].

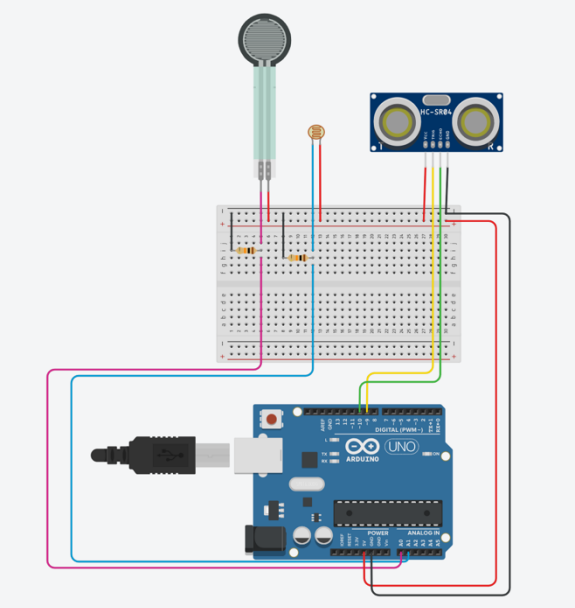


Figure 7.3.2 - Analog Sensors Circuit

Example sensor configurations are shown in Figure 7.3.2. Although TinkerCAD doesn't feature limit switches, which were selected according to the sensor decision matrix, they are functionally similar to the force sensor in the image. Just as the force sensor, a limit switch uses 2 wires and a voltage divider may be necessary to ensure sensor output falls within the 5 V range for the Arduino. A photoresistor is depicted, which can be used for light detection on the Flavor Fusion device. In addition, Hall Effect sensors can be used to detect magnetic field proximity. This offers an alternative for container detection.

To incorporate and control multiple LEDs in the Flavor Fusion model, a multiplexer, or MUX, is useful. Figure 7.3.4 depicts its setup, allowing 8 LEDs to be controlled using only 3 digital pins. This is extremely advantageous in saving space, simplifying wiring, and maximizing pin usage of the Arduino.

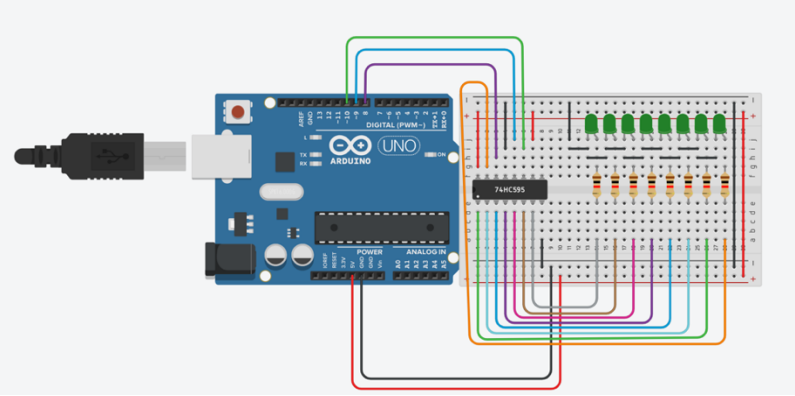


Figure 7.3.4 - LED MUX Circuit

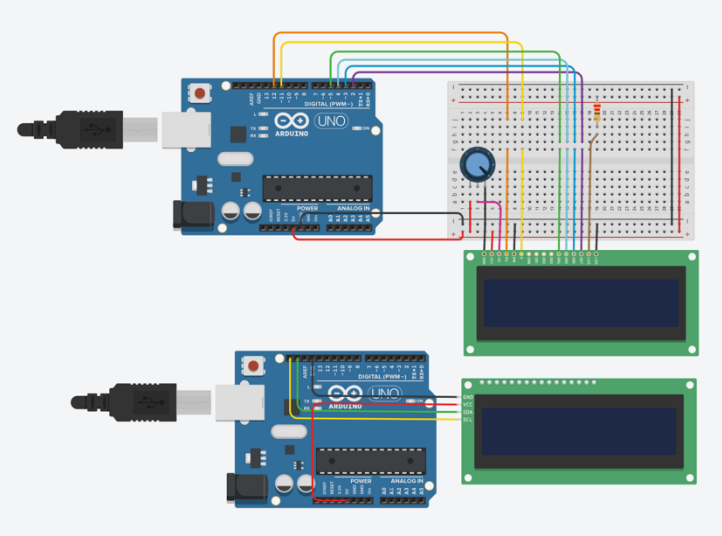


Figure 7.3.3 - LCD Display Circuit

LCD screens can communicate with microcontrollers in multiple manners. Shown in Figure 7.3.3 are two methods: SPI (above) and I^2C (below). I^2C requires only 4 pins, whereas SPI can take as many as 12, although this number depends on the LCD screen used. Also depending on the model, SD cards may have compatibility for data storage, which would allow for local recipe storage in the Flavor Fusion device. If a RAMPS board is connected to the Arduino, there are dedicated pins on the shield that are used for LCD

connection, most often in SPI protocol. As with the stepper motor drivers on RAMPS, a firmware such as Marlin is required to communicate between the Arduino and shield [70].

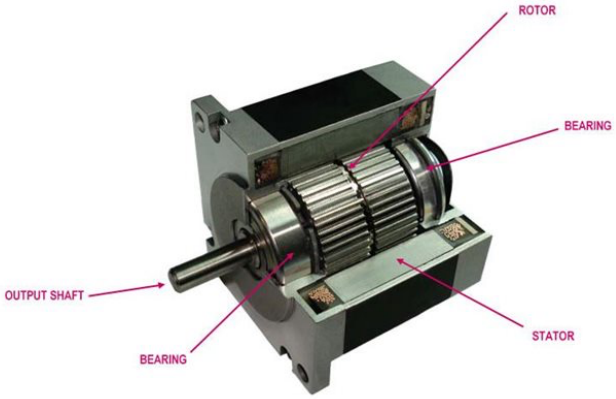


Figure 7.3.5 - Stepper Motor [71]

Stepper motors are unique as compared to other DC motors because their range of motion is divided into "steps", or small angular displacements. By altering the number and speed of motor steps, the motor's position and velocity can be controlled with high accuracy. For these reasons, stepper motors are ideal in the Flavor Fusion device. Using stepper motors allows the dispensing mechanism to return to the same position after the order is complete or allows for spice containers to be rotated exactly into position. Figure 7.3.5 shows the internals of a stepper motor, with many coils arranged in phases that allow the motor to move in steps [71].

7.4 System Design Approach

Flavor Fusion is taking a system design approach in order to efficiently tackle the behemoth task of getting the device finalized for showcase. There are so many moving parts, and by dividing up ownership, each individual on the team is a specialist in one subject and able to assist the other members in developing their components. This also allows each member of the team to play into their strengths and focus on creating the best version of their subsystem they can.

There are some downsides to this approach, especially if the process is not conducted properly. Increased fragmentation is one of the biggest hurdles, but that has been addressed by making the 3D Printing Subsystem an oversight subsystem that focuses on integrating components together and providing direction to the other subsystems. Weekly check-ins and stand-ups are used to check the progress of each team member, and efficiently allocate resources as necessary.

7.5 Dispensing Mechanism Subsystem

7.5.1 Design Objectives

The dispensing mechanism has many design objectives to meet, a lot of which are very similar to the spice housing, and carriage which include dishwasher and food-safe materials, no sharp edges, etc. For the dispensing mechanism on the biggest design goals and challenges specific to the dispensing mechanism, we are going to go into detail on easy to clean, easy to load, and spice leeching. Ease of cleaning multiple design iterations were done and considered from screws to compliant mechanisms to allow a snap fit.

7.5.2 Total Components

The dispensing mechanism consists of the housing device, a funnel, an auger, a drive shaft, a removable bearing cage, and a bearing. These six components work together to form the dispensing mechanism and work in tandem with the carriage device and spice bottles to allow ease of use for these components to work.

7.5.3 COTS versus Custom

Much of this system will be custom. For our application, 5 out of the 6 parts will need to be 3D printed since most of this is a one-off prototype. These custom parts include the dispensing housing, auger, drive shaft, bearing cage, and funnel are all custom parts. We have investigated COTS augers but for our size and applications, none were available to us. Another part was the drive shaft itself. We thought of incorporating a metal piece to connect the auger to the driving wheel but decided against it to save money. The only part that is COTS is the bearing itself. This is due to the high tolerances of these bearings that an FDM printer will struggle with to make them custom. Not to mention the amount of time to accurately tolerate these bearings to work 100% of the time.

7.5.4 Lead Times

The biggest lead times the system faces are the custom parts that need to be printed out. The dispenser takes about 4 hours to print one set, and our system has 10 dispensers so 40 hours to print one full set if the set doesn't have a print failure. Another thing we also must factor in is the post-manufacturing of these items. The amount of support material in these prints can be a lot in parts such as the auger and dispensing housing itself. This postproduction time can take anywhere from 45 to an hour to remove all support material and ensure fitments meet our standards of quality for Flavor Fusion.

7.5.5 Design Path

The biggest design path was finding the spice bottle in which we will use. We went through multiple sizes and threads and diameters to figure out what one we wanted to use in our final design. Another was figuring out the drop zone location and height. This would be a critical design feature for this system in limiting the amount of space the dispenser can hang from the carriage. But the biggest critical design issues this system had were during the design and prototyping phases in figuring out. 1, How can we design this system to allow for ease of assembly not only for the consumer but also for the team assembling the final product. 2, Were how certain items can be placed into a dishwasher and what may need to be hand cleaned. This was a big part in design and will need further improvements if Flavor Fusion wants to become a mass-produced item. For example, the auger in this system is so small that placing it into a dishwasher unassembled will be lost in the dishwashing processes. 3, The tolerancing of everything. This product needs to be able to be dissembled entirely and it should be quick and easy to so. This was very tricky since. Another design path choice we had to make was to prevent spice leeching either into the container itself or having the spice leech into other components. Many ideas were taken from Powder manufacturing machines in how they can move this material without having end up in bearings, gears etc. This is a crucial design consideration because spice that is unwanted can end up in places where it can damage components, go into the drop zone itself and cross contamination food, end up in the bearing cage and cause unnecessary wear on items which and lead to premature failures.

7.6 Mobile Application Subsystem

Allison is the subsystem lead for the mobile application. The design objectives for the mobile app focus on creating a user-friendly experience that integrates wireless connectivity via Bluetooth. The mobile app design objectives include feature objectives/requirements and accessibility objectives/requirements. The app should include a home list of spices, a recipe book, a settings page, and information about the app. The user should be able to view how much spice is in each container and should receive a notification when the container is close to empty. The user should be able to input a name and spice amount for each container. The user should be able to view/search through the recipe book, add recipes, delete recipes, and edit recipes. The user should be able to create an order, modify the number of servings, and receive a notification when their order is complete. The mobile app should have a login passcode. Most importantly, the app should support wireless connectivity over Bluetooth. Accessibility requirements include text size adjustment, light/dark mode, and proper color contrast. The components of the mobile app subsystem include the app itself, App Store Connect, Test Flight, the app itself, and the iPhone. The only part of the app that was purchased was the Apple Developer Program membership for $99. The Apple Developer Program allows developers to deploy an application to Test Flight via App Store Connect. Test Flight allows users to download the app to their iPhones without needing to have their phones in developer mode. The development for the UI is complete with most of the time having been spent on the recipe book and spice list. The plan for completion is to finish development on UI features and final Bluetooth integration. Although all UI features are currently complete, as more progress is made on the physical device, changes may need to be made to the UI. As these changes arise, development will be completed. Bluetooth connection from an Arduino to the iPhone is complete but final integration will be completed alongside the Arduino coding. The main issues that may arise stem from Bluetooth integration. Any Bluetooth issues will be resolved with collaboration between the mobile app team and the electronics team.

7.7 Electronics Subsystem

Samuel is the team member in charge of the electronics system. The main objectives for electronics are to distribute power, enable actuation, sense, and store data, and interact with the user with a display. In addition to these objectives, requirements like electrical safety, heat management, operational noise, and dispensing time serve as important constraints. The entire electronics system consists of around 30 active components such as sensors, motors, or processors. Passive components like wires or resistors are not included in this count. Due to the precise nature of electronic materials and manufacturing, all components were bought or requested from inventory rather than fabricated. Based on these parts and the tasks they accomplish, the following subsystems span the electronic specifications for the device: power, actuation, sensing, display, microcontroller, and aesthetics. The process of working with electronics is rarely as straightforward as prototyping and testing mechanical parts. The combination of materials, quality, and configuration of each electronic part- combined with the intangible nature of electricity and code- often places the design engineer at the mercy of manufacturers. As such, any parts that require lots of “tweaking” or “fiddling” during the design process tend to be the parts that take the most time to configure. In contrast, simple components like LEDs can be wired and tested in a matter of minutes. It is expected that LCD programming, stepper motor and driver tuning, Bluetooth communication logistics (e.g. what data to send, how to serialize, and identifying where Booleans are required to trigger events), and wiring (specifically wire management and soldering) will necessitate the most time to prototype, test, and integrate. As construction of the Flavor Fusion device begins, the overarching tasks for the electronics system will be to prototype and test (i.e. wire, program, and debug) all components individually before advancing to subsystem testing and eventual integrated testing. The idea is to reduce uncertainty by starting at the individual part level, then progressing to higher-level systems. Each of these steps are iterative processes, so it is unknown exactly how long each stage will take. Despite this fact, plenty of flexibility is maintained to react to unexpected complications like hardware issues, software or firmware issues, and integration issues. In each of these cases, the best course of action is to isolate the problem- whether debugging a certain line of code, identifying a faulty wire, or removing parts from the system and resuming testing- before searching for solutions. In particular, hardware testing can be performed with multimeters to measure voltage and current when a component is behaving unexpectedly. As aforementioned,

electronics are complex and highly interdependent, so diagnosing the issue is key. From there, online research or consultation with electronics and programming experts can be indispensable. Many others have experience with similar issues and can quickly resolve what might have been an otherwise hours-long solution process. Finally, if no remedy has been found, it may be time to simplify the system and try a more rudimentary approach, perhaps with more basic parts. Achieving success on a smaller scale may be sufficient for the project, or it may bring enough understanding to be able to resolve the original issue in a more complex configuration.

7.8 Housing Subsystem

Ethan is the lead for the housing subsystem. The housing subsystem features many prominent engineering design objectives revolving around the size and safety of the device. The most important of these objectives are that the machine must fit within a cabinet or cupboard, and that the device must feature no sharp edges or corners. To comply with these objectives, we have designed the device on SolidWorks with parts that fit the size requirements, and we have applied a fillet or chamfer of at least 2mm to each edge.

There are nine major components that make up this subsystem. These include the carriage, the handle that goes on carriage, the motor mounts, the housing front drop zone area, the handle that goes on housing, the lazy Susan, the top ring, the top lid, the housing shell, and the LCD Cover. Each one of these parts will be fabricated besides the Lazy Susan, which we are purchasing.

The only parts that are going to require a long lead time are the carriage and the dispenser. This is due to the fact that the carriage is based on the dispenser dimensions, so the carriage has to be designed around these dimensions. Additionally, the carriage is going to have a longer print time due to its size.

The biggest issue that has arisen so far is the fact that the motor housing and the spice house top were too thin to comfortably print the part with the included magnet housing space. Possible ways to address this situation include making these parts slightly thicker, buying magnets that are smaller in diameter, or moving the parts that connect these two housing pieces

to the inside of the device. The solution that we decided upon was ordering smaller magnets that would fit within the housing space.

7.9 3D Printing Development

Ryan and Alexandra are taking the lead on 3D Printing and Integration development. This section requires oversight of all facets of the device and the ability to adapt quickly to unforeseen design constraints from all subsystems. The main components that make up this system are the housing, carriage, dispensing zone, and motor mounts. These fabricated components are the necessary building blocks to combine our purchased components into our final system.

Due to the size of the device, the 3D printed components will require long lead times to print. Therefore, quick fixes are not as easily done and require planning and refinement in CAD software before continuing to print. As a result, prototyping and testing are critical for addressing issues with spacing and fitment, assuring correct spice dosing, and efficiently interacting with the app and dispenser. These processes provide crucial insights to ensure the machine's design and manufacturing processes are on track. The prototyping will allow the team to assemble the device and have a strong understanding of how the device will look and learn how to optimize the design for future manufacturing builds. This anticipated optimization is expected to reduce the time and complexity for assembly further than just CAD visualizations and simulations alone.

The end goal of our device is to be as food-safe as possible, which makes the material and printer used especially important for the final device. With this in mind, all prototype versions of the device will not be considered food-safe and utilize PLA materials for inexpensive and quick printing. The final device presented at showcase, however, will be considered food-safe and use food-safe PETG printed in a brand-new printer, ensuring a clean path for the material.

The biggest issue facing the 3D printing and integration of Flavor Fusion is its size. At 14 inches in diameter, the 3D printers readily available to the team are not large enough to fit the major components in one piece. As a result, the prints will need to be spliced and combined together using plastic welding. This challenge will require the team to carefully design the

components to be integrated separately and still remain structurally sound. This initially will manifest as cuts in the slicer, and eventually into design features in SolidWorks. The end goal is to have the device housing and carriage printed on a large-format 3D printer at Lockheed Martin, but will entirely depend on the company being able to accommodate our request. By planning and designing the device in case we are not able to print at Lockheed; the team saves time and frustration down the road, while still producing top quality for showcase.

8. Design Analysis

8.1 Physical Hardware

8.1.1 System as a Whole

Flavor Fusion is a complete and fully conceptualized model due to the design considerations from the app development to the end output meets every requirement set by the team behind Flavor Fusion. Flavor Fusion's main goal is to reduce the amount of space your spice cabinet holds as well as reduce the amount of wasted food. The team has put careful consideration into the user and how customers can use Flavor Fusion. Many such areas were explored to their fullest such as Cleaning, Assembly, User Accessibility, and Maintenance of the product.

8.1.2 Early Concepts of Flavor Fusion

Flavor Fusion as a whole has been a very difficult product to design around. Not many production-level models exist for our application. One product that the team took a lot of inspiration from was a failed Kickstarter called TasteTro. TasteTro was a home or commercial-use product that dispensed prepacked spice that was delivered to the users via monthly subscriptions. We took a lot of patent drawings and examined those on how something like this should or could be designed. Most of the design inspiration came from homebrewers making different versions of what Flavor Fusion could look like or how it should behave. The team for Flavor Fusion did many napkin sketches on what certain parts could or should look like in the very early design phases and then shifted to a computer-aided model once concrete requirements were set. Many rapid 3D prototypes were made to figure out sizing and tolerances for parts as well as a cardboard model was created to gauge internal sizing.

8.1.3 Engineering Principles

Flavor Fusion incorporates many engineering principles such as a static load case for our product. This simple load case incorporates a very simple static analysis question and incorporates solid mechanics as well. Very easy simplified hand calculations can be done to give us rough estimates on what loads and max deformation we might have when something is placed on top of our machine. Another engineering principle used is machine design and this is also in tandem with solid mechanics. We have a couple of shafts and will have applied torques to them again. Simple solid mechanic torque hand calculations can be done early on to give us ideas about torque estimations before FEA is done on them. For machine design, we must find what bearings to use for our use cases. As well as in the early prototypes we talked about a gear-driven carriage for our design to make the carriage spin versus a shaft-driven design. The biggest fundamental principle used in the project is mechatronics. The flow of how mechatronics works is the biggest thing that allows this project to function. Having everyone tackle and develop CAD, FEA, Arduino code, and our mobile app development had been crucial in making sure all our parts fit and integrate correctly. Flavor Fusion on the outside seems very simple but once we began breaking it down to its core components, it became a very complex electro-mechanical system.

8.1.4 Design and Analysis Software

Table 8.1 - Comparing Design and Analysis Software

	SolidWorks	**Ansys**
What are the advantages of this approach?	SolidWorks gives us a very intuitive CAD software to go and build and assemble our model compared to Ansys	Ansys is an extremely powerful analysis software that can perform multibody structural analysis which is crucial for our project
What are the disadvantages of this approach?	SolidWorks Simulation package is not the greatest. It lacks the accuracy that ANSYS has and gives a lot of errors in analysis since it's meshing isn't as fine	Ansys is mainly used as an analysis software. It has the ability to perform CAD, but the UI and tools are not very fleshed out compared to SolidWorks. Ansys is also very tedious to use
Why do we think this model is accurate of the physical object?	This is the most accurate model of our system from a pure design and assembly point of view since all parts fit and mesh with each seamlessly. This gives us a guide on how our product will be built	Ansys is extremely accurate when it comes to its Structural analysis packages. The mesh sizing and placements simulate what a real-life object would experience. These simulations can also be compared to rough hand calculations to compare results

8.1.5 Ansys Simulation Results

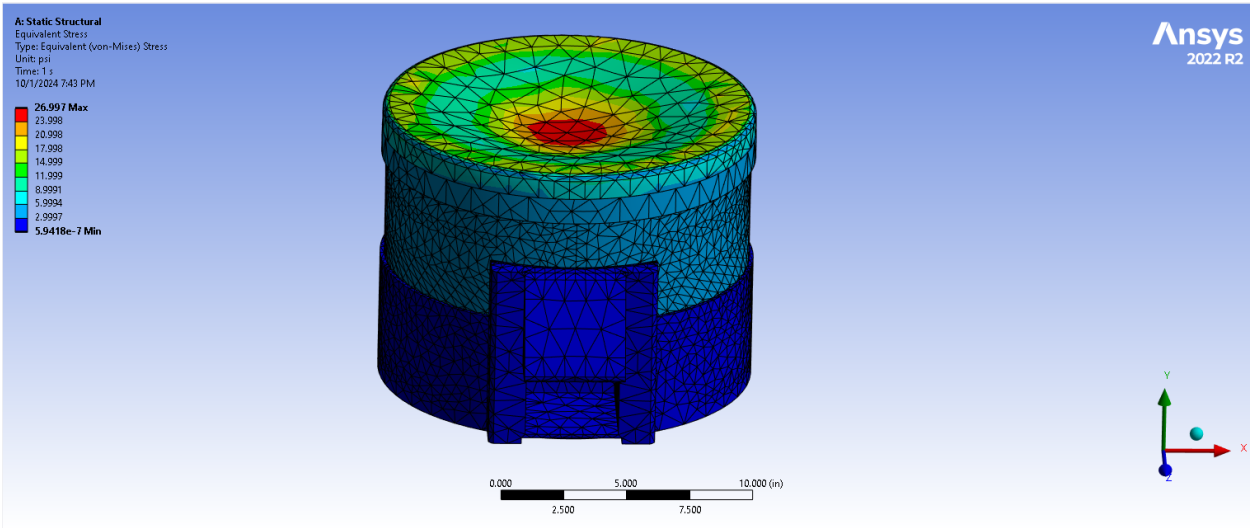


Figure 8.1.1 - Ansys Mechanical Von Mises Stress Profile

To run a Finite Element Analysis in Ansys Workbench, the SolidWorks CAD assembly was converted to a geometric file and imported for meshing. Materials were defined, with PET plastic for the 3D printed components and aluminum for the Lazy Susan. To analyze static loading on the top of the machine, such as a user laying a kitchen appliance atop the device, a fixed support constraint was applied to the bottom face of the housing and a 10-pound load was distributed across the top face of the lid. Relevant results are presented below.

Von Mises stress, or equivalent stress, is an important parameter used for yielding analysis. It incorporates stress contributions from all principal directions to quantify the state of stress at a specific point, which can be compared to the material's yield stress. For the load case specified above, maximum Von Mises stress occurs at the center of the lid, with a value of nearly 27 psi. This corresponds to a safety factor of 15, according to Ansys's material specifications.

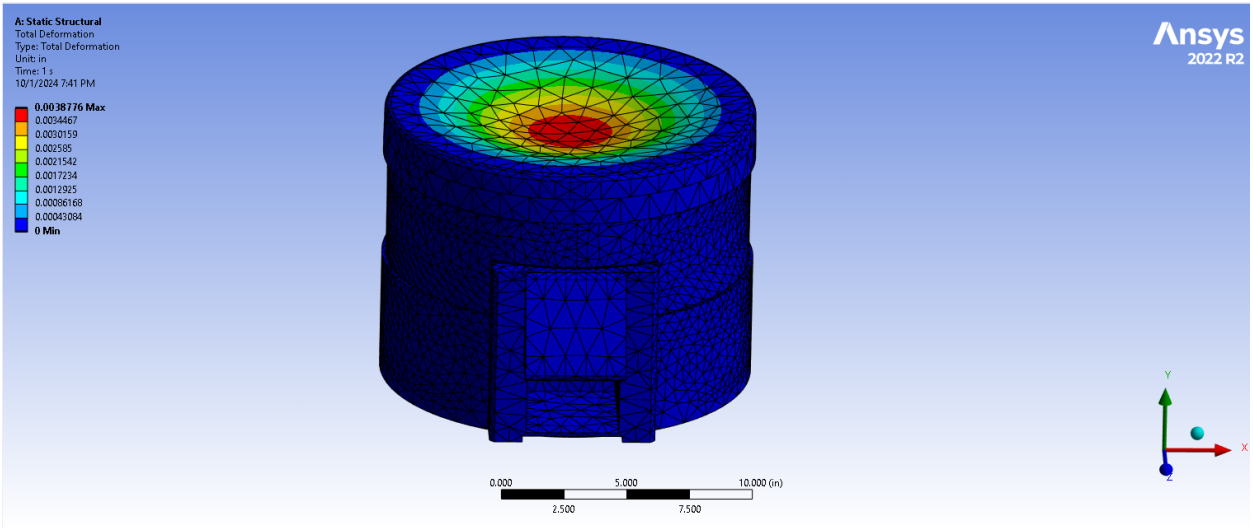


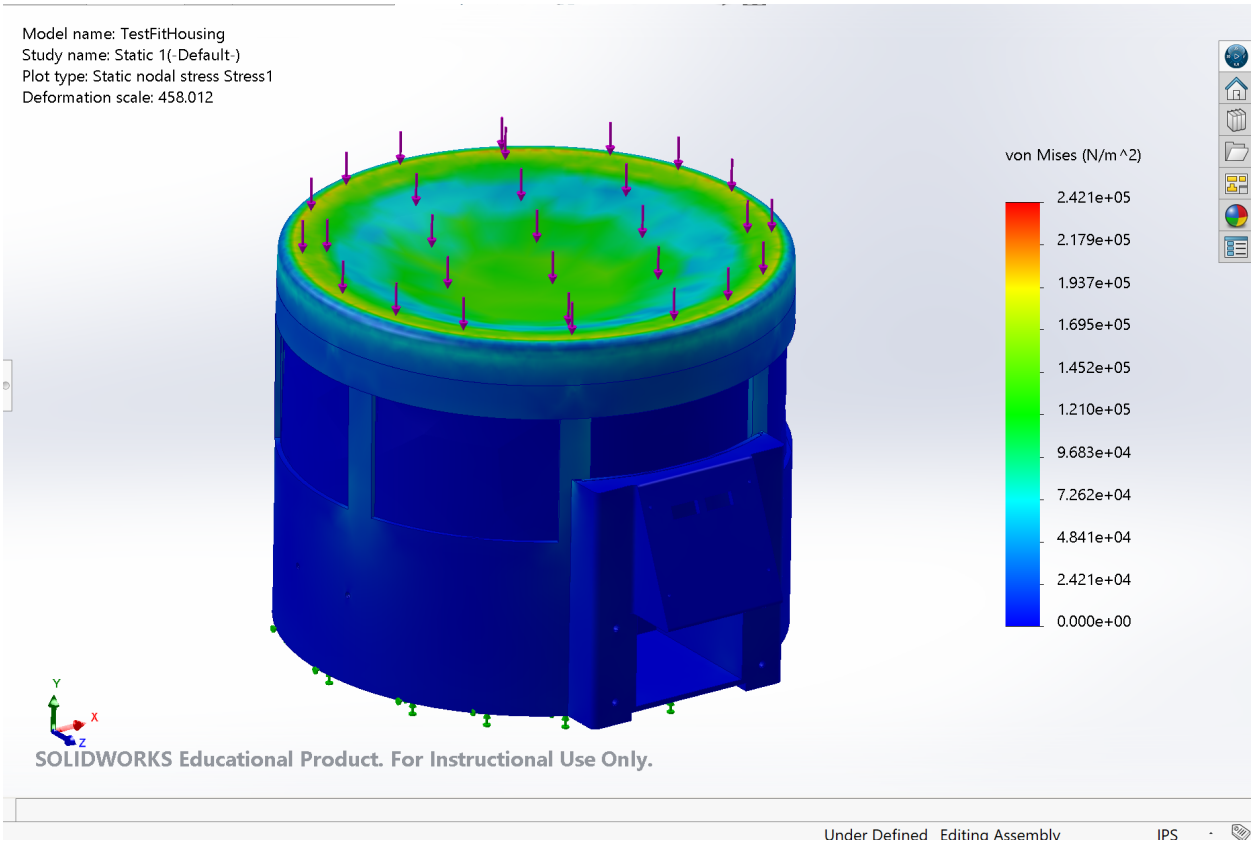
Figure 8.1.2 - Ansys Mechanical Deformation Profile

Deformation, also referred to as static deflection or displacement, is a measure of the change in distance that a material experiences under loading. In addition to stress and strain, deflection is a common measure of failure because materials are often designed to operate within constrained spaces. Deformation is related to stress according to geometry and material properties, such as Young's Modulus. For the load case considered, the maximum deflection occurs in the center of the lid, with a value of 0.00388 inches.

8.1.6 SolidWorks Simulation Results

To verify Ansys's FEA results, the same loading conditions were supplied to the assembly in SolidWorks and results for Von Mises stress and deformation were displayed. SolidWorks, as a primarily CAD-oriented software, is generally considered to have less reliable meshing capabilities and FEA results, but the ultimate purpose of this simulation was a ballpark verification of the more reliable Ansys results.

Figure 8.1.3 - SolidWorks Von Mises Stress Profile



The Von Mises stress profile in SolidWorks also shows significant stress in the center of the lid but has a notable difference in stress concentration along the outside of the top ring. The peak stress is 242.1 kPa, which converts to 35.1 psi, although this occurs along the outside of the lid. The center, according to the color map, experiences a peak equivalent stress of around 150 kPa, or 21.75 psi. While the stress profile does not match that of Ansys exactly, the range of 21.75 psi to 35.1 psi encompasses the peak Ansys stress of 27 psi, demonstrating a general agreement between the models.

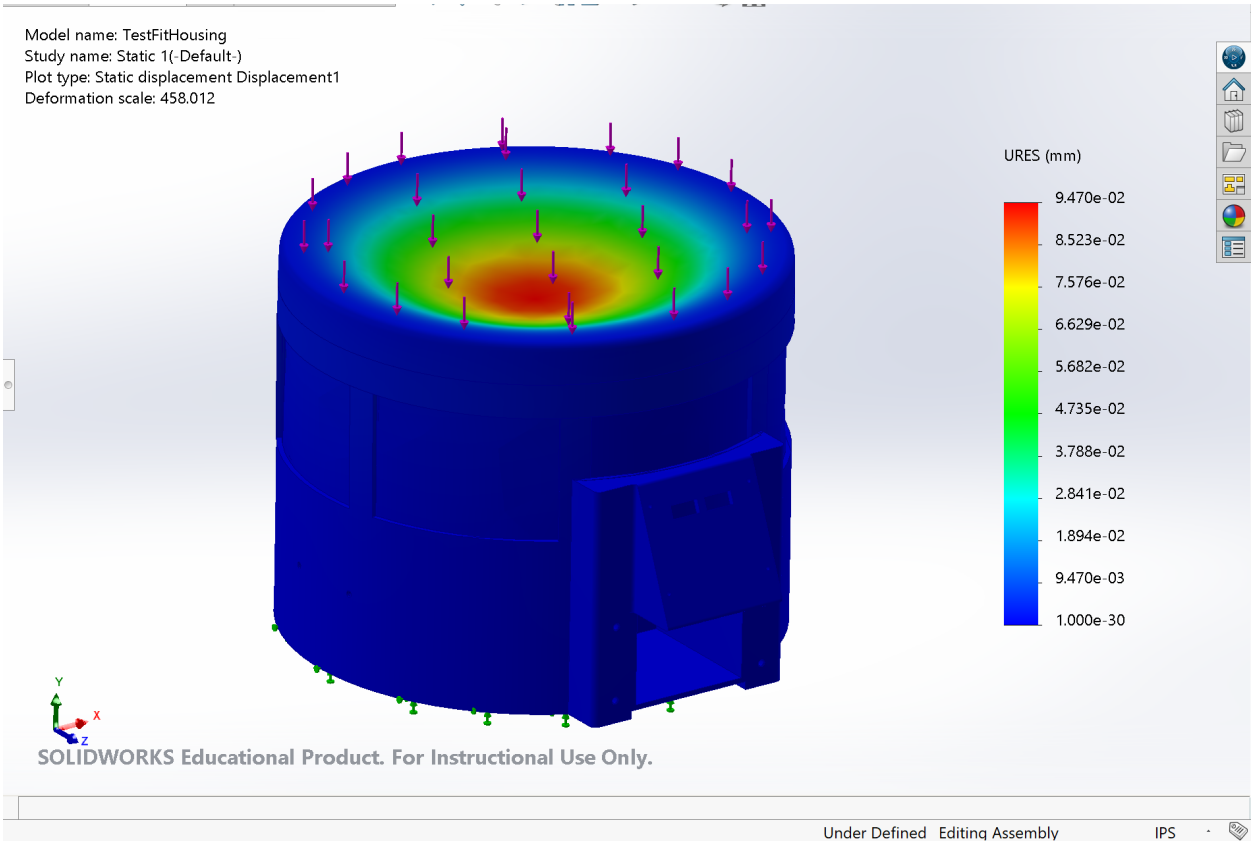


Figure 8.1.4 - SolidWorks Deformation Profile

The static displacement of the SolidWorks model concurs with Ansys that maximum deformation occurs at the center of the lid. In this case, the peak value is 0.0947 mm, or around 0.00373 inches. Comparing to Ansys's maximum deformation, there is only around 3.85% relative error. In other words, there is a very reliable agreement between the Ansys and SolidWorks deflection load cases.

8.1.7 Hand Calculations

For a uniform circular plate with clamped boundary conditions and a uniform distributed load, q_o, Solecki and Conant [72] show that:

Flexural Rigidity	$D = \frac{Eh^3}{12(1-\nu^2)}$	$[lbf \cdot in]$
Maximum Deflection	$w_{max} = \frac{q_o R^4}{64D}$	$[in]$
In-Plane Stresses	$\sigma_{rr}\|_{r=0} = \sigma_{\theta\theta}\|_{r=0} = \frac{3(1+\nu)qR^2}{8h^2}$	$[psi]$
Maximum Normal Stress	$\sigma_{zz}^{max} = \frac{M(\frac{h}{2})}{I} + \frac{N}{A}$	$[psi]$
2nd Moment of Inertia	$I = \frac{1}{12}(2R)h^3$	$[in^4]$
Bending Moments	$M_r = M_\theta = \frac{(1+\nu)q_o R^2}{16}$	$[lbf \cdot in]$
Net Bending Moment	$M_{net} = \sqrt{M_r^2 + M_\theta^2} = \sqrt{2} \cdot M_r$	$[lbf \cdot in]$
Von Mises Stress	$\bar{\sigma} = \sqrt{(\sigma_1-\sigma_2)^2 + (\sigma_1-\sigma_3)^2 + (\sigma_2-\sigma_3)^2}$	$[psi]$

Due to uniaxial loading, it can be assumed that cylindrical coordinates are the principal directions. For the following material properties of PET plastic and loading conditions, results can be obtained:

$$E = 500\ kpsi \quad \nu = 0.33$$

$$q_o = \frac{10\ lbf}{\pi(7\ in)^2} = 64.961\ mpsi \quad R = 7\ in \quad h = 0.25\ in$$

$$\therefore w_{max} = 0.003336\ in$$

$$\therefore \sigma_{rr} = \sigma_{\theta\theta} = 25.4011\ psi$$

$$\therefore \bar{\sigma} = 32.1949\ psi$$

The results of these hand calculations support the findings from the Ansys and SolidWorks load cases, with a relative error of approximately 14% between maximum deformation calculations and a range of stresses (25.4 psi in-plane to 32.2 psi Von Mises) that encompasses the Von Mises stresses from Ansys and SolidWorks. Although the hand calculations were performed under constraints and assumptions that do not perfectly apply to the CAD geometry, the general agreement of calculated results ensures that Ansys's simulation is realistic.

8.2 Electronics

8.2.1 The System as a Whole

From an electronics perspective, Flavor Fusion presents a complete model because every aspect of user interaction and machine operation has been identified and decomposed into necessary parts, with power requirements and safety considered for each circuit.

8.2.2 Early Concepts of Flavor Fusion

References such as 3D printer configurations were crucial to identify other machines' approaches to synthesizing tasks like power delivery, actuation, user control, sensing, and programming. The electronic configuration of the RepRap project for 3D printers using the Ramps 1.4 shield and Arduino Mega served as a starting point for the team's electronics diagram, since it incorporates all the major components of Flavor Fusion [73]. As electronic components were identified, individual circuits were combined into an overall TinkerCAD model for visualization and analysis.

8.2.3 Engineering Principles

The electronics aspects of Flavor Fusion are highly reliant on concepts in Electrical Engineering, Mechatronics, and programming in C. For instance, numerous calculations in power requirements are built on fundamentals like Ohm's Law and Kirchhoff's Laws. Since the microcontroller programming is performed in Arduino IDE, digital logic and sensing is all performed in a C++ based language. Other peripherals such as motor drivers and the LCD screen are also reliant on Serial Communication Protocols like UART or I2C. Additional disciplines such as vibrations and controls are involved to a lesser extent for the dispensing process. Motors will be tuned to vibrate and have slight eccentricity in order to disrupt spice clumping, while controls will be necessary to sense and calibrate motor actuation.

8.2.4 Modeling and Analysis Platforms

Table 8.2 - Comparing Electronics Modeling and Analysis Platforms

	TinkerCAD	**Hand Calculations**
What are the advantages of this approach?	TinkerCAD provides a user-friendly platform to visualize components and wire connections. It also warns users when components are at risk of failure and allows Arduinos to be programmed before simulation.	By calculating electrical interactions by hand, users have complete control over parameters and can tune them to data sheet specifications of actual components.
What are the disadvantages of this approach?	Users are forced to choose components from a preset library, so not all components are included. In addition, built-in components may have slightly differing properties than the models featured in Flavor Fusion.	Hand calculations are cumbersome and error-prone, although this can be reduced with analysis software such as MATLAB.
Why do we think this model is accurate of the physical object?	TinkerCAD can represent the entire electronics subsystem (with some substituted parts) in an easily viewable format. This helps with wire management and pin connections while also testing Arduino code via simulation. Overall, it is a quick and simple way to approximately design and simulate the entire subsystem.	Exact specifications for each component can be retrieved from data sheets and used to calculate with higher certainty. This offers a sanity check for any simulated results, especially if TinkerCAD predicts failure.

8.2.5 Preventing Errors in Modeling

Preventing errors in electronics modeling is a crucial step, as an improper connection could result in part failure and potential safety hazards. From the start, all component connections are compared to the requirements published by RepRap for the Ramps 1.4 shield [74]. This is a reliable source that already encompasses many of the electronics present in the Flavor Fusion subsystem, so it is a helpful starting point to eliminate unnecessary design. In other words, there is no need to reinvent the wheel. From there, individual circuits have been identified and modeled in TinkerCAD to ensure wire connections are defined and parts are not fried. An overall model of all electronics also exists in TinkerCAD. Finally, hand-calculation analysis is completed for each general circuit as well as any components of concern, allowing the team to verify TinkerCAD's results with higher fidelity.

8.2.6 Electronics Modeling

Unlike FEA testing, electronic specifications (e.g. rated voltage and current, number of pins, and connection type) are already known for each component, so a circuit just needs to be designed to achieve the required constraints. The only unknowns are the wires and components needed to bridge these gaps. The main constraint of concern is rated voltage: a power supply unit (PSU) has already been obtained to step 110V AC wall power to 12V DC for compatibility with the Ramps shield and its peripherals. For any additional voltage ratings, a voltage splitter or transistor will need to be utilized. Other constraints to consider include current draw, pins and connection types, and serial communication compatibility for digital components.

It is always possible that some aspects of physical prototyping have been unaccounted for during the modeling and simulation process- for instance, connecting the LCD screen may require more pins than originally accounted for, or may require firmware with specific Arduino libraries that were unknown to the team during planning. For this reason, each component will be prototyped individually to ensure power conditions are controlled, code is concise, and connections are easily managed. After the component has been fully configured under these controlled conditions, additional components can be combined in the circuit. This gradual integration will help to fine-tune power requirements and identify coding errors before tackling the entire subsystem at once. Of course, data sheets will be consulted for each part before supplying power.

Flavor Fusion's electronics subsystem has been carefully designed to fulfill numerous project requirements. Electronic safety is ensured by operating components within their rated conditions. Heat safety is maintained with heat sinks, air vents, a temperature sensor, and a fan. The device will be powered by plugging into the wall, like most kitchen appliances. A shut-off switch is featured for emergencies. An LCD screen with a scroll wheel allows users to navigate the machine and view dispensing progress. Sensors will avoid user error by detecting dispensing cup presence; machine error is also avoided via sensor calibration. Accurate dispensing is ensured by stepper motors and drivers that support 1.8º steps with up to 1/256 microstepping, while the TMC2209 stepper driver also reduces noise to comply with the required operation conditions.

8.2.7 Images, Equations, and Diagrams

A TinkerCAD model was created to visualize the electronics system for Flavor Fusion and plan wiring configurations. This ensures that components will be connection-compatible and provides a convenient visual representation of all parts before the team begins wiring. Due to TinkerCAD's limited component library, some parts have been substituted as a representation of actual Flavor Fusion components. For instance, while there are no stepper motors in TinkerCAD, DC motors with encoders were placed in their stead.

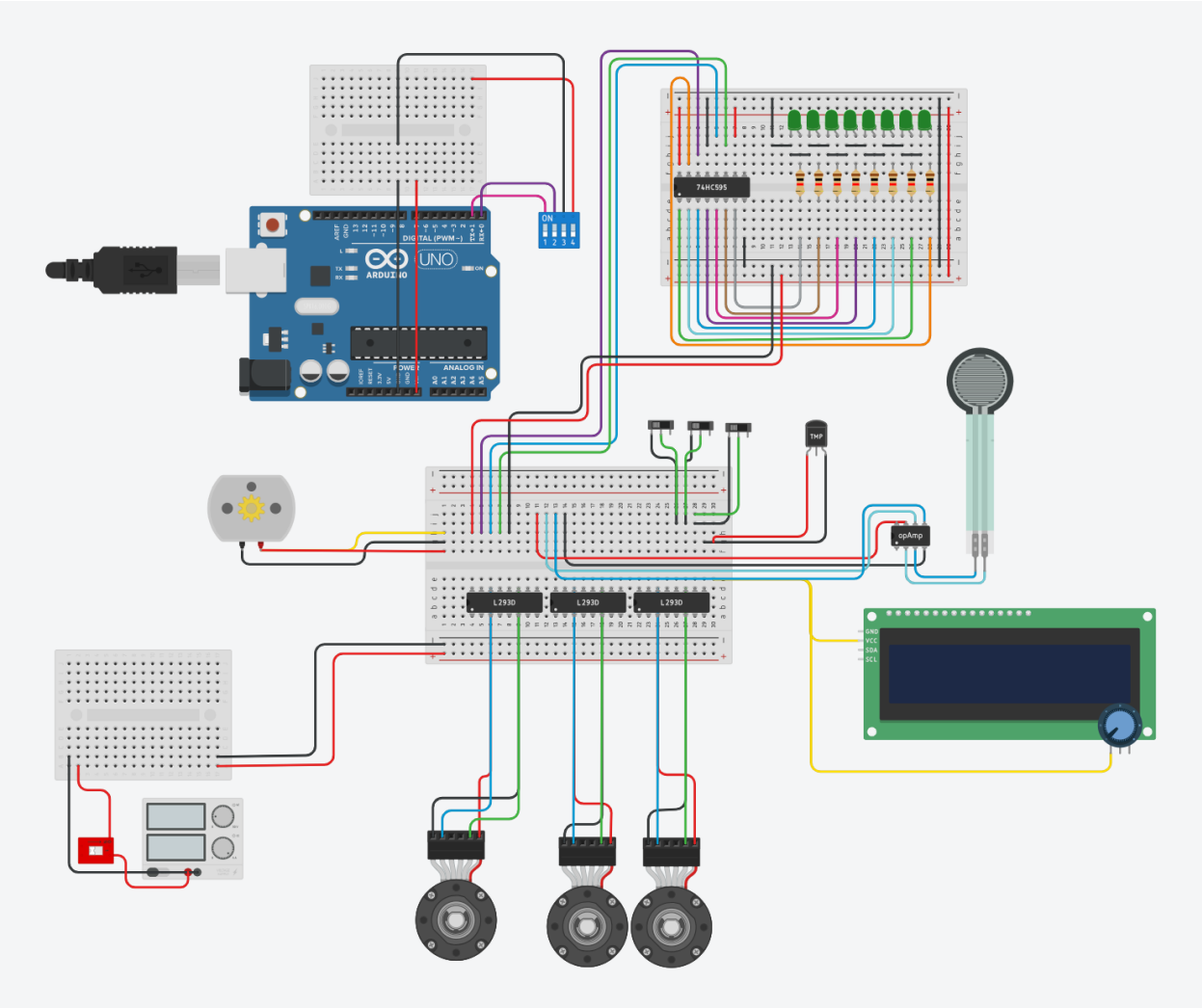


Figure 8.2.1 - TinkerCAD Wiring Diagram

To ensure that electronic safety is maintained and parts are not damaged, it is critical to analyze the voltages and currents across each circuit. Beyond Kirchhoff's Junction and Loop Rules, Ohm's Law is a critical tool for evaluating voltage, current, and resistance. In addition, voltage dividers can be designed to step voltage down from the power supply unit (12 V) to the rated voltages of given components (often 3.3 V or 5 V). An example voltage divider is depicted below.

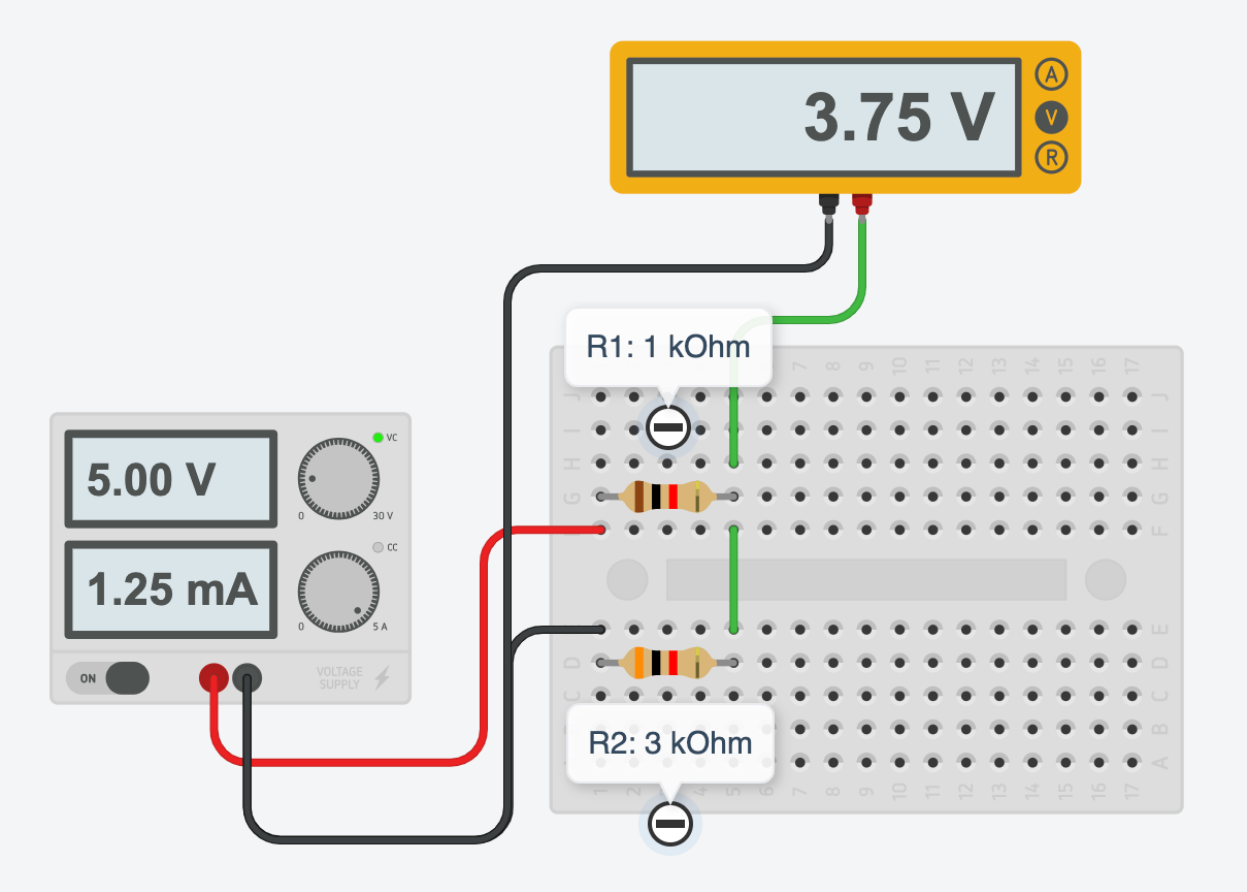


Figure 8.2.2 - Voltage Divider

Ohm's Law $V = IR$

Voltage Divider $V_{out} = V_{in} \cdot \frac{R_2}{R_1 + R_2}$

The table below shows current-consuming parts with their peak current draw for any arbitrary moment during the operation of the device. The Ramps shield is rated to carry 5 amps to all its connections- Arduino Mega, motors, sensors, etc. The current draw of each component was summed to estimate the peak total current draw, which should not exceed the 5 amp rating.

Table 8.3 - Electronic Components and Current Consumption

Component	**Current Draw**	**Notes**
NEMA 8 + A4988	0.5 A	A4988 supports sleep mode, reducing current to 33%
NEMA 17 + TMC2209	1.5 A	Peak operational current of a NEMA 17 is 1.5 A
NEMA 11 + TMC2209	0.5 A	TMC2209 supports current regulation, which can reduce current to as low as near-zero
Arduino MCU	50 mA	Logic
Load Cell	1.6 mA	
Temperature Sensor	2.5 mA	
LCD Screen	0.18 mA	
Limit Switches	1 mA each (x3)	
Fan	0.18 A	
Bluetooth Module*	40 mA	Peak current is during pairing
Generic LEDs*	20 mA each (x8)	
RGB LEDs*	50 mA each (x8)	Rated at 5V operation
Shift Register*	80 µA	
Total	**3.337 A**	

* : Noted components can be powered separately from the PSU to reduce current flowing through the Ramps board.

9. Final Design and Engineering Specifications

9.1 Physical Device

9.1.1 Overall Design

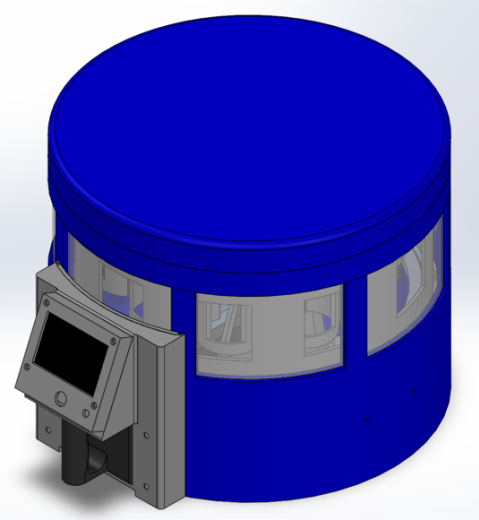


Figure 9.1.1 - Overall Design of Flavor Fusion

9.1.2 Main Housing

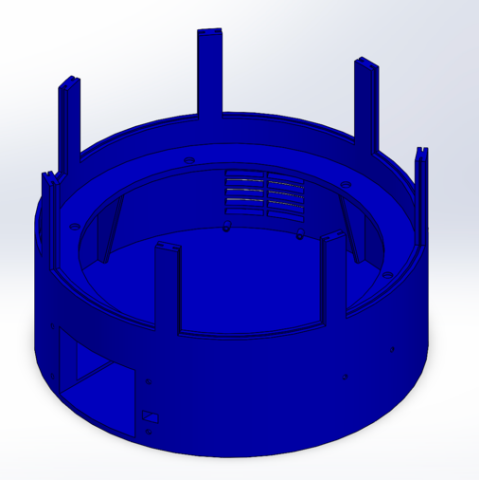


Figure 9.1.2 - Main Housing of Flavor Fusion

9.1.3 Carriage

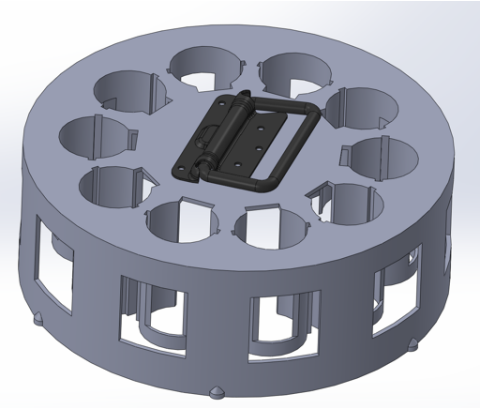


Figure 9.1.3 - Flavor Fusion Carriage

9.1.4 Drop Zone Housing

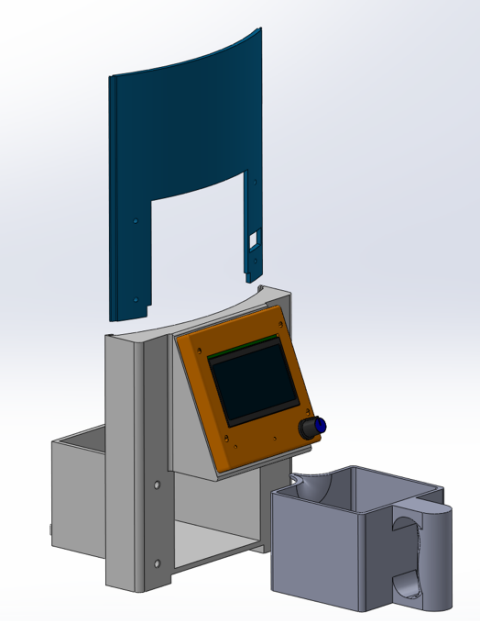


Figure 9.1.4 - Drop Zone Assembly. Contains Drop Zone Housing, LCD Display Cover, Drop Zone Container, and LCD

9.1.5 Spice Dispensers

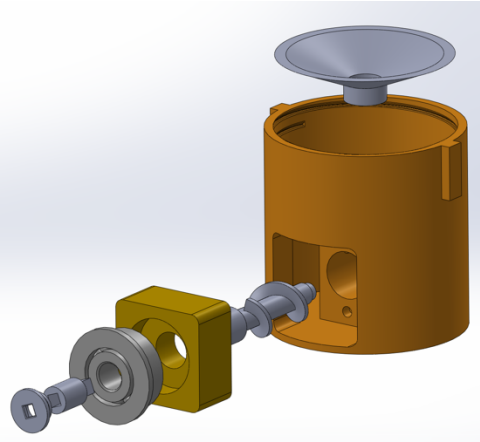


Figure 9.1.5 - Spice Dispenser Assembly. Contains Auger Attachment, Driveshaft, Bearing, Bearing Cage, Auger, Dispensing Container, and Funnel

9.1.6 Motor Mounts

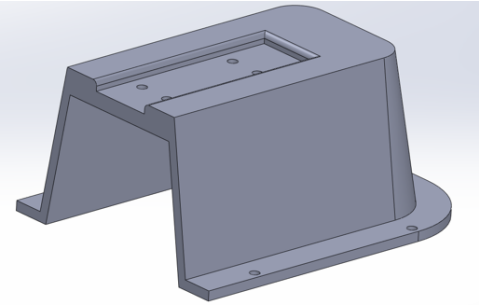


Figure 9.1.6 - Lower Motor Mount, supports Linear Actuator

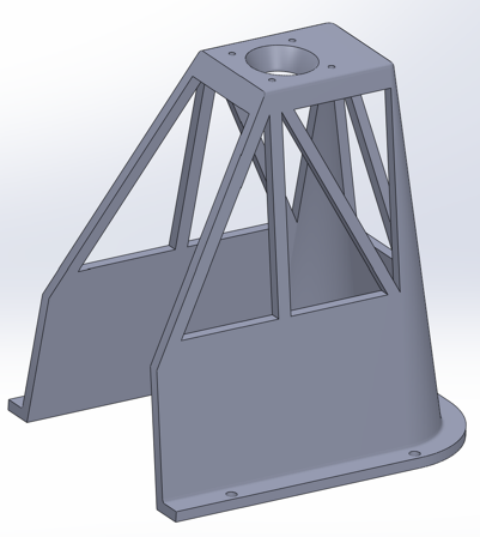


Figure 9.1.7 - Upper Motor Mount, supports NEMA 17

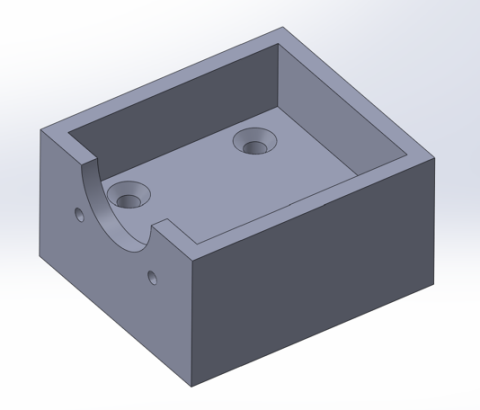


Figure 9.1.8 - NEMA 8 Cage, goes on top of Linear Actuator to spin Auger

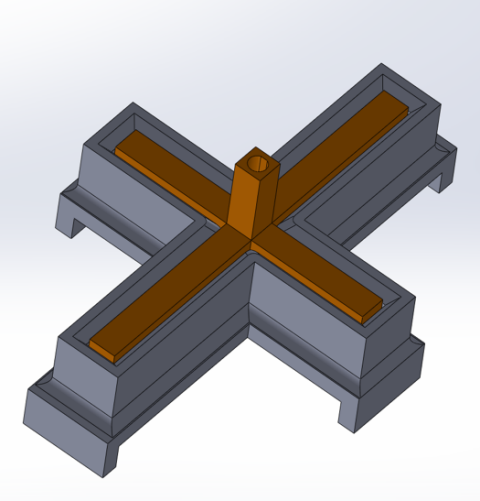


Figure 9.1.9 - NEMA 17 Attachment, used to connect NEMA 17 to carriage. Orange piece is attached to motor shaft and spins grey attachment.

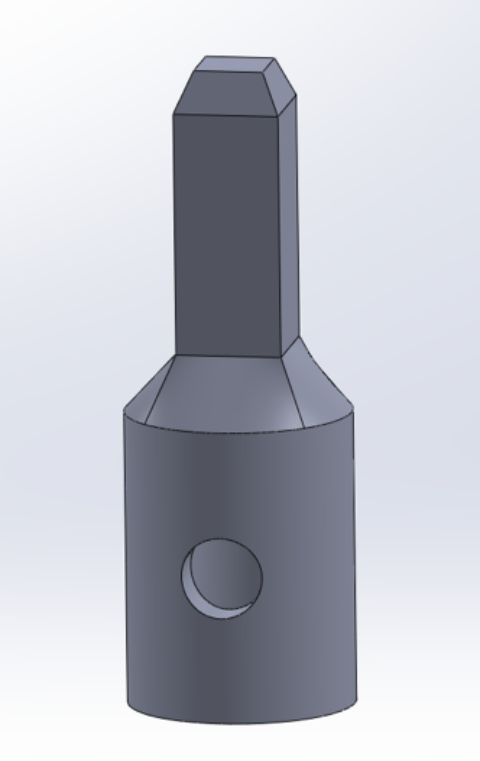


Figure 9.1.10 - NEMA 8 Dispenser Connector, used to spin Auger inside of Dispensing Mechanism assembly.

9.2 Electronics

9.2.1 Overall Circuit

Figure 9.2.1 - Overall Internal Circuit of Flavor Fusion

9.2.2 Arduino

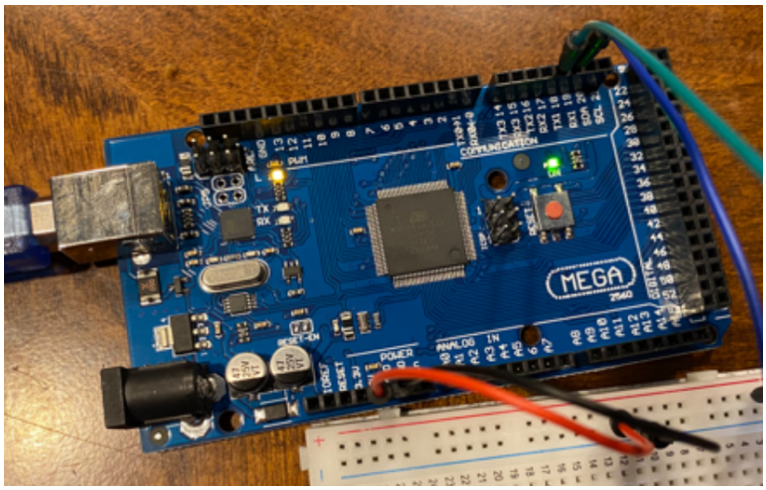


Figure 9.2.2 - Arduino MEGA 2560

9.2.3 LCD Display

Figure 9.2.3 - LCD Display

9.2.4 Linear Actuator

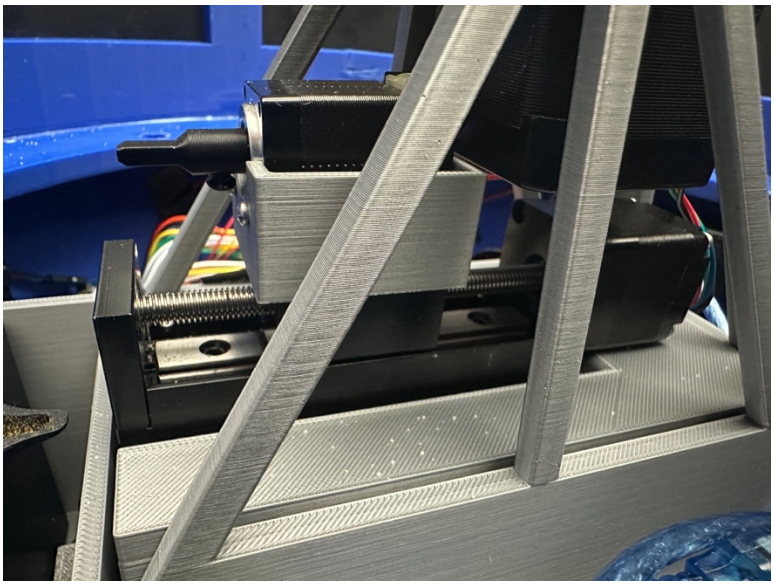


Figure 9.2.4 - Linear Actuator

9.2.5 NEMA 17

Figure 9.2.5 - NEMA 17 (Top Left)

9.2.6 NEMA 8

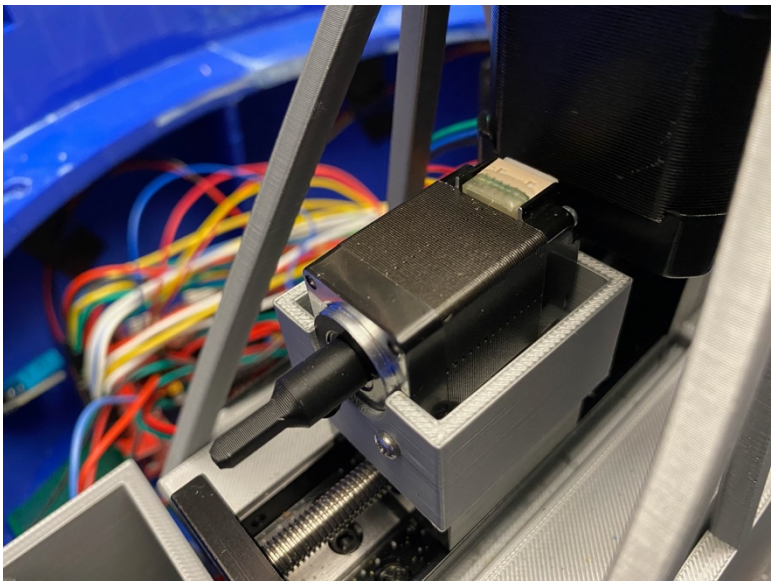


Figure 9.2.6 - NEMA 8 (Baby NEMA)

9.2.7 Limit Switches

Figure 9.2.7 - Carriage Calibration Limit Switch

Figure 9.2.8 - Drop Zone Container Limit Switch

9.2.8 Bluetooth Module

Figure 9.2.9 - Bluetooth Module (Lower)

9.3 Application

9.3.1 Blend Page

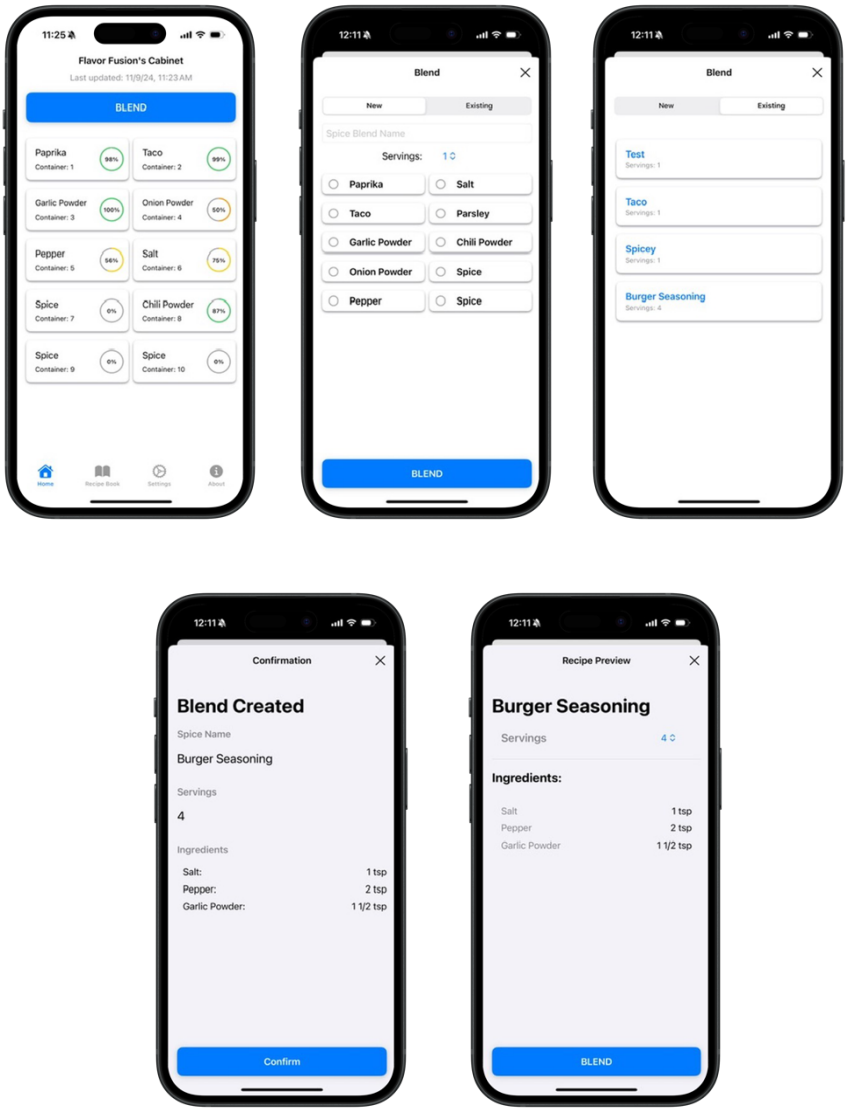


Figure 9.3.1 - Blending. From Left: Home List, Blend New, Blend Existing, Confirmation, Recipe Preview

9.3.2 Recipe Book

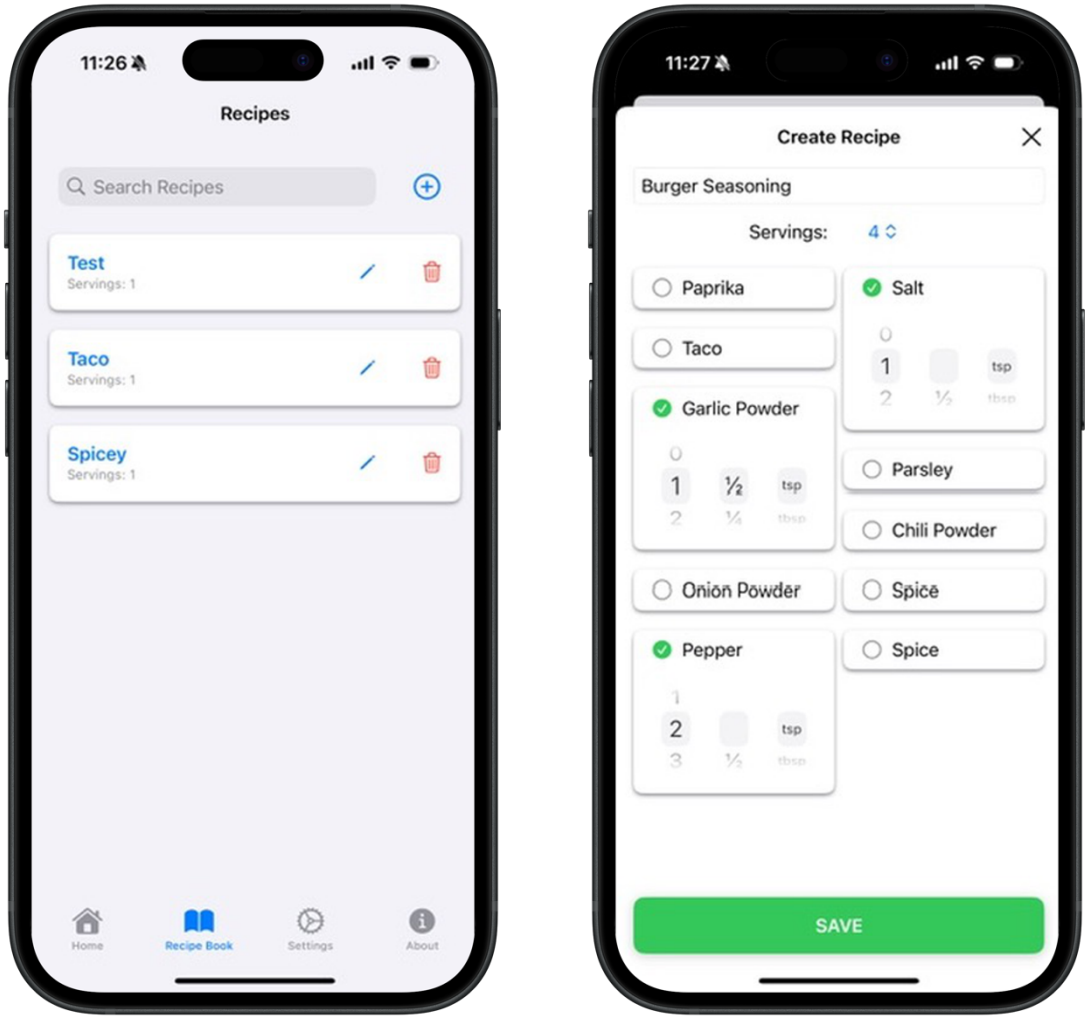


Figure 9.3.2 - Recipe Book. From Left: Recipe List, Create Recipe

9.3.3 Spice Percentages

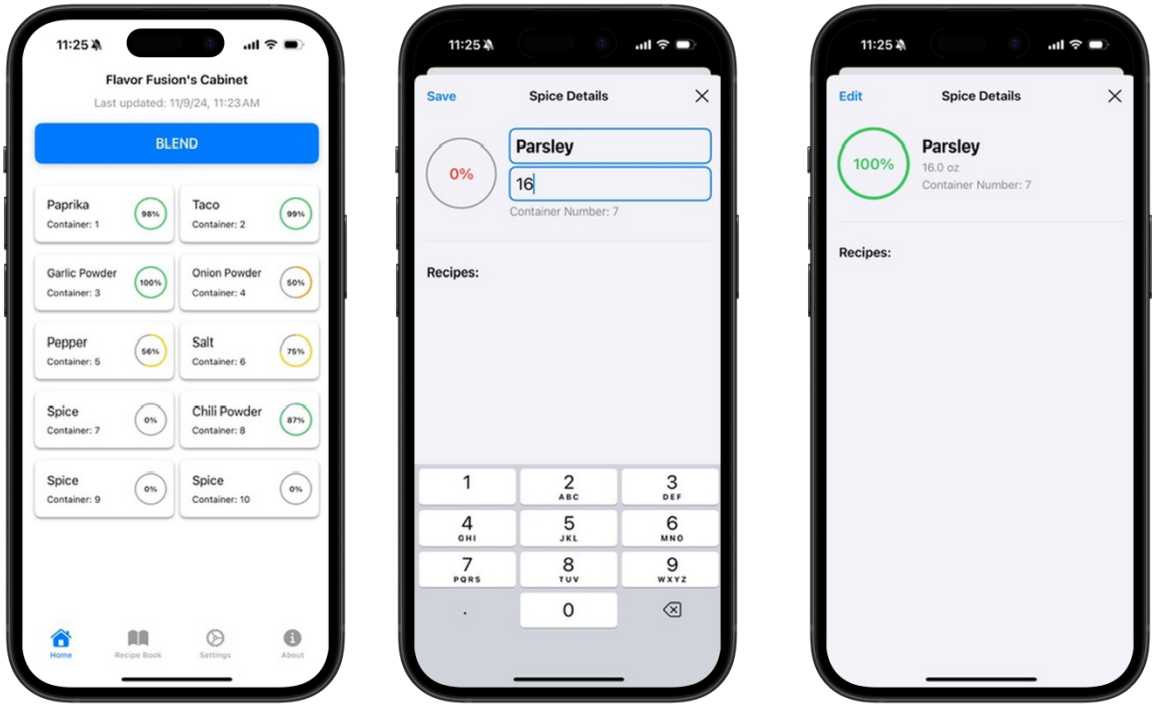


Figure 9.3.3 - Spice Amounts. From Left: Home List, Add Spice to Container

9.3.4 Accessibility

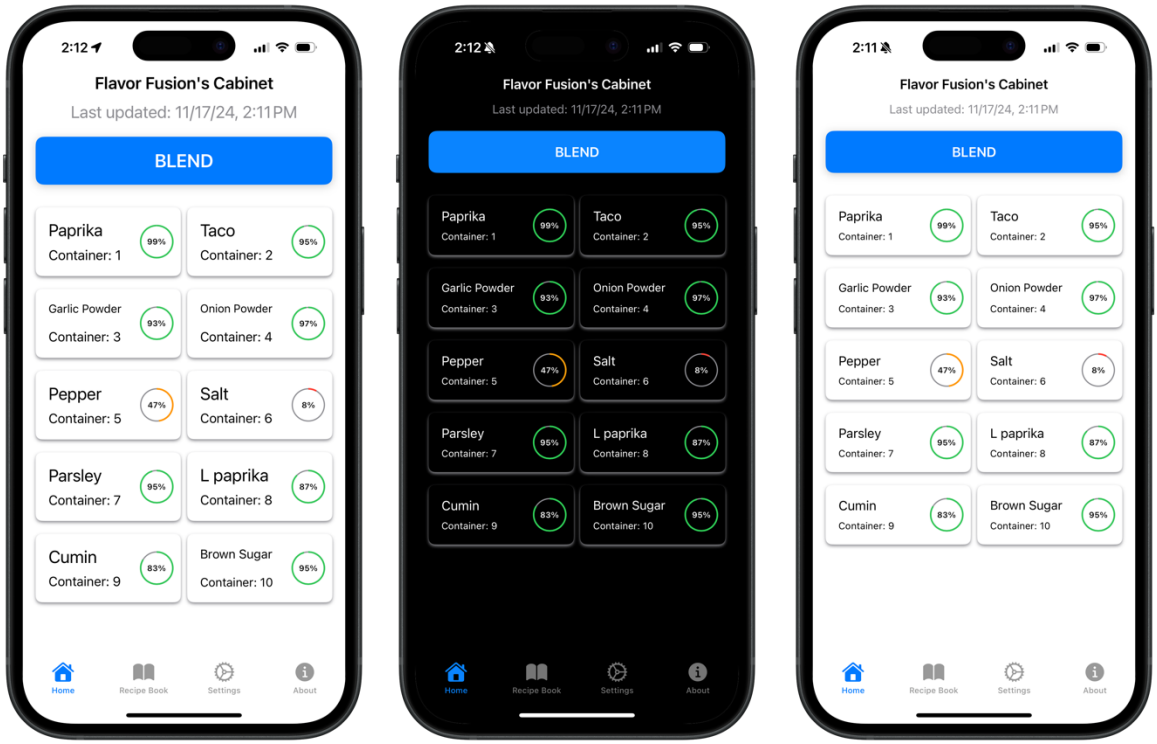


Figure 9.3.4 - Accessible App Options. From Left: Increase text size, Dark Mode, Light Mode

10. System Evaluation

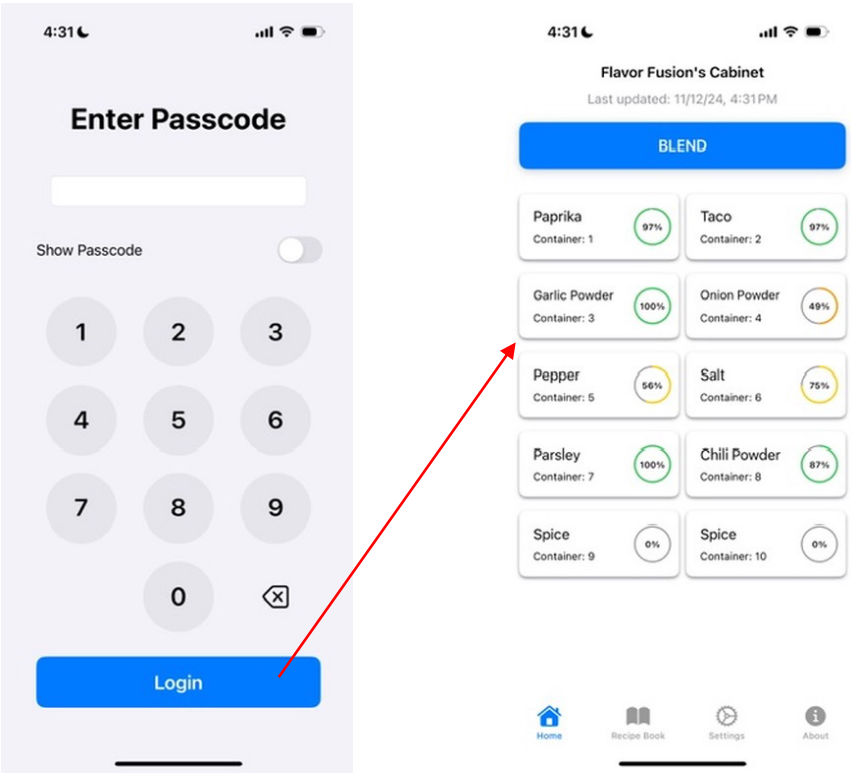
10.1 Mobile App Testing Results

10.1.1 UI Testing Results

Login and Create Passcode Functional Tests

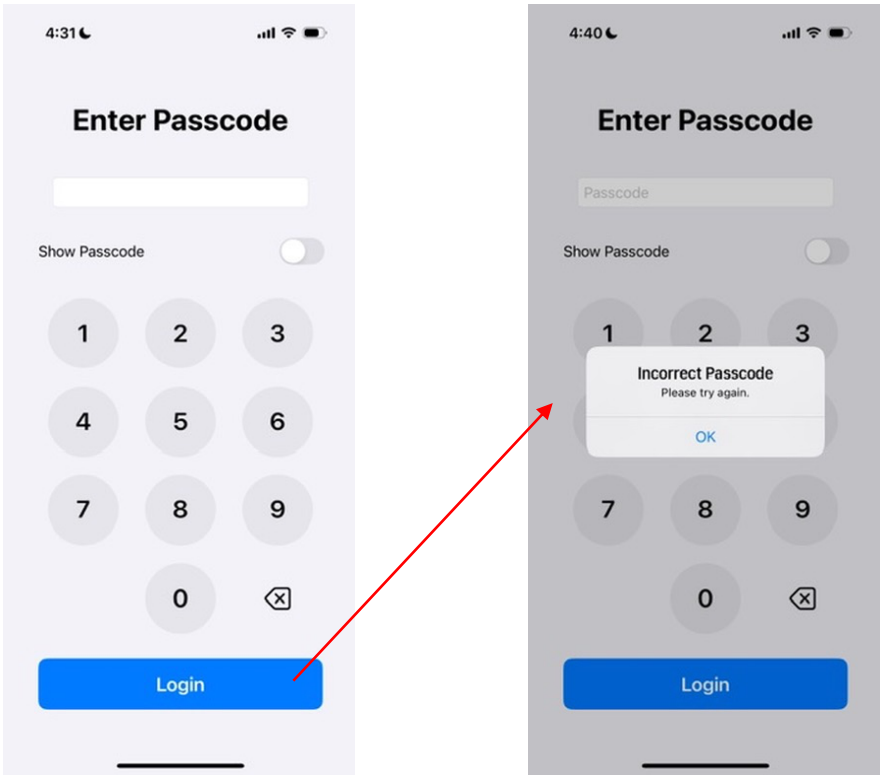
Test Case 1: Successful Login with Correct Passcode

- **Objective:** Verify that users can successfully log in using the correct passcode.
- **Steps:**
    1. Launch the app and navigate to the login screen.
    2. Enter the correct passcode.
    3. Verify that the user is successfully logged in and navigated to the home list screen.
- **Expected Result:** The app allows users to log in with the correct passcode and redirects them to the home screen without errors.
- **Result:** Passed



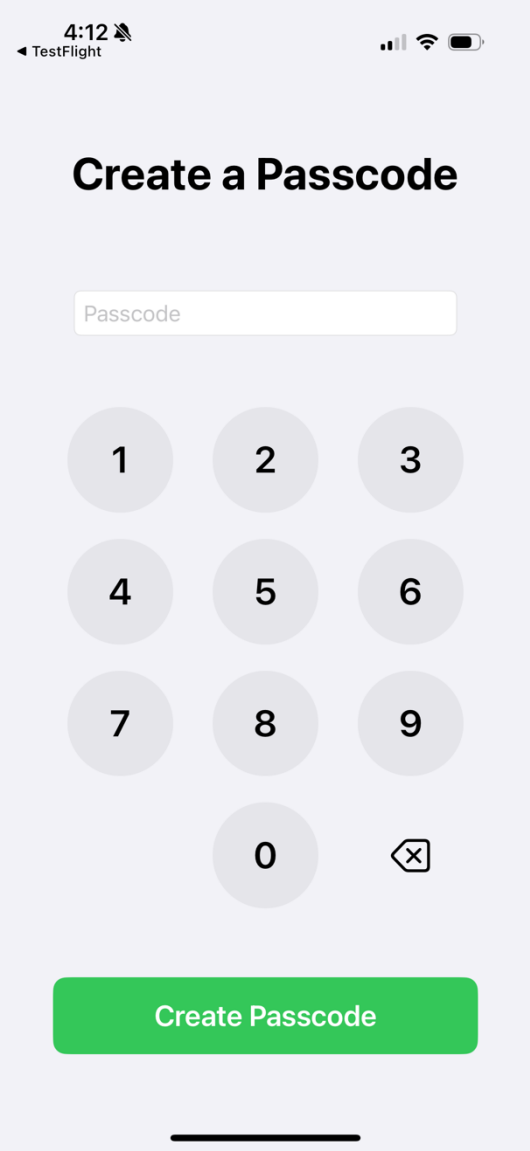
Test Case 2: Failed Login with Incorrect Passcode

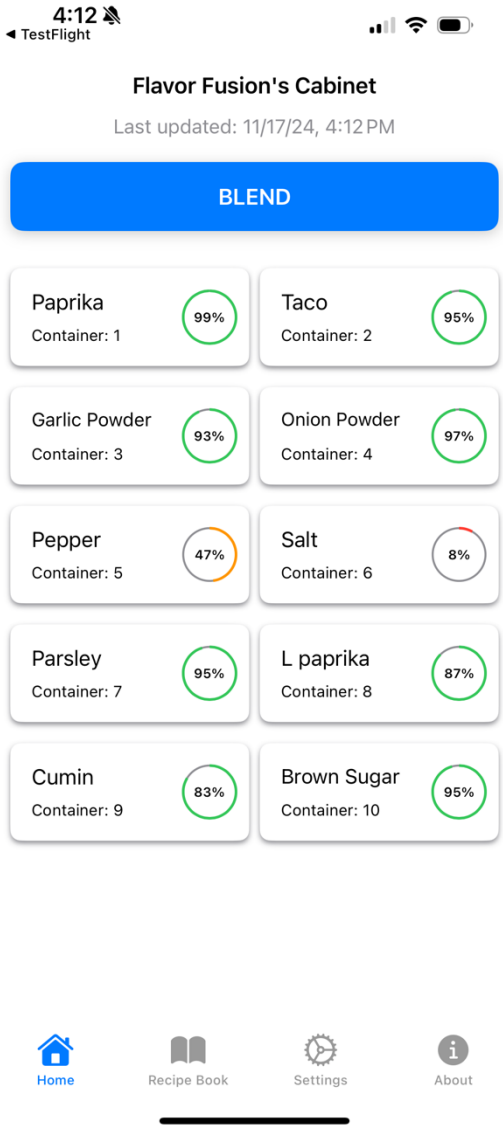
- **Objective:** Verify that users cannot log in with an incorrect passcode.
- **Steps:**
    1. Launch the app and navigate to the login screen.
    2. Enter an incorrect passcode.
    3. Verify that the user cannot log in.
- **Expected Result:** The app does not allow the user to login and displays an appropriate error message.
- **Result:** Passed



Test Case 3: Create Passcode

- **Objective:** Verify that users can successfully create a new passcode during the first app launch.
- **Steps:**
    1. Launch the app after downloading.
    2. Create a new passcode.
    3. Verify that the passcode is successfully created, and the user is navigated to the home list.
- **Expected Result:** The app allows users to create a new passcode and redirects them to the home screen or dashboard upon successful passcode creation.
- **Result:** Passed

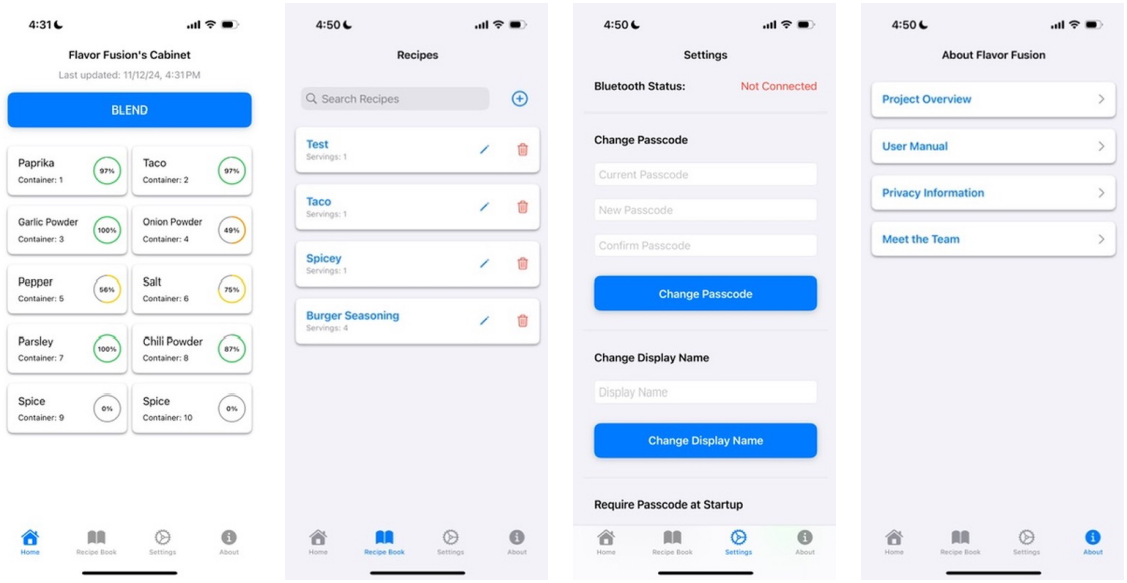




Main Features Functional Tests

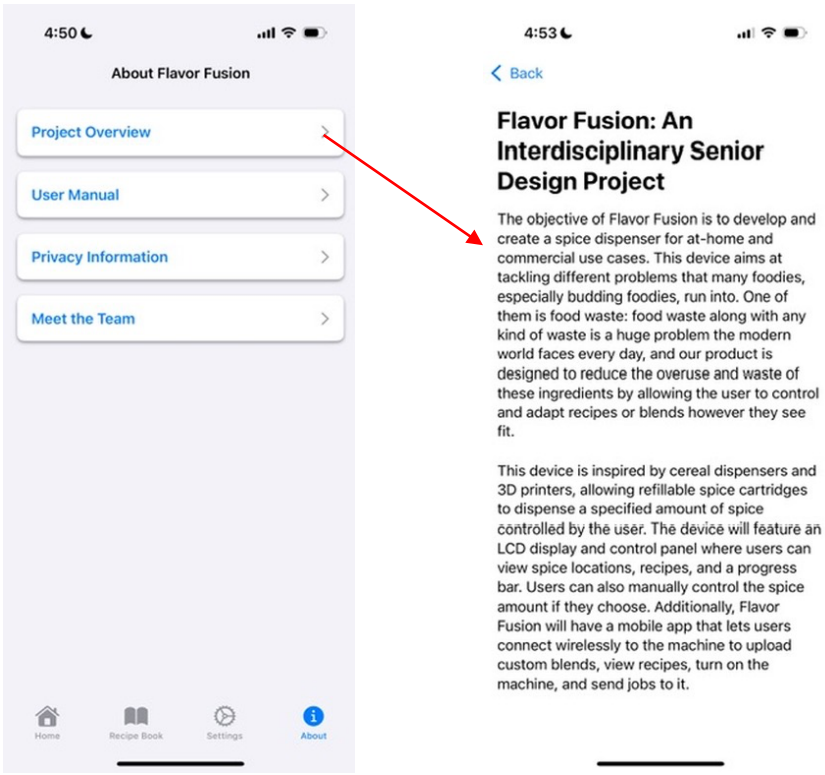
Test Case 1: Tabs

- **Objective:** Ensure that app includes all 4 tabs – Home List, Recipe Book, Settings, and About.
- **Steps:**
    1. Launch the app and navigate to each tab.
    2. Verify that the correct view renders when tapping on each tab.
- **Expected Result:** Each tab should render without errors.
- **Result:** Passed



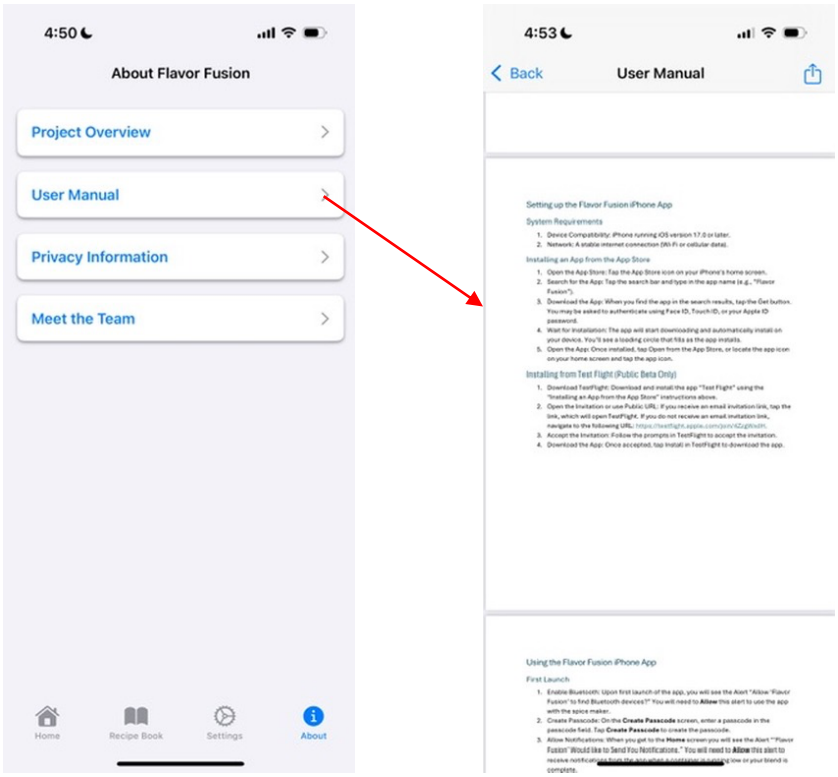
Test Case 2: About View – Project Overview

- **Objective:** Ensure that About View includes a project overview.
- **Steps:**
    1. Launch the app and navigate to the About tab.
    2. Verify that the project overview exists and contains text.
- **Expected Result:** The text should render on the screen.
- **Result:** Passed



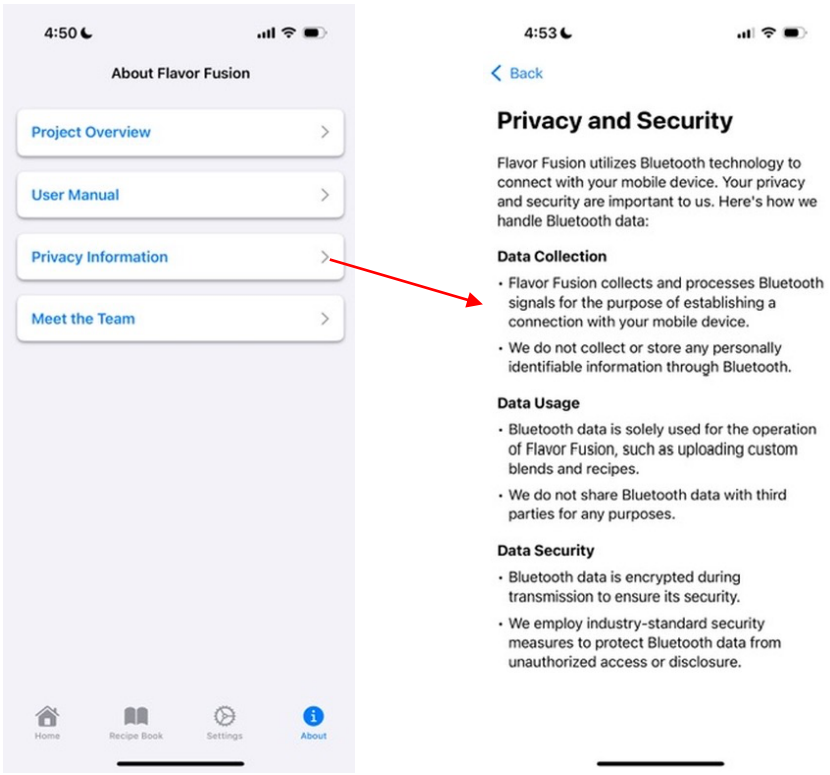
Test Case 3: About View – User Manual

- **Objective:** Ensure that About View includes a user manual.
- **Steps:**
    1. Launch the app and navigate to the About tab.
    2. Verify that the user manual exists and contains text.
- **Expected Result:** The text should render on the screen.
- **Result:** Passed



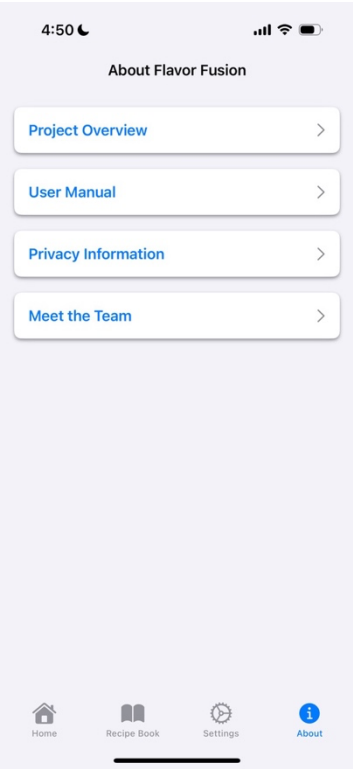
Test Case 4: About View – Privacy Information

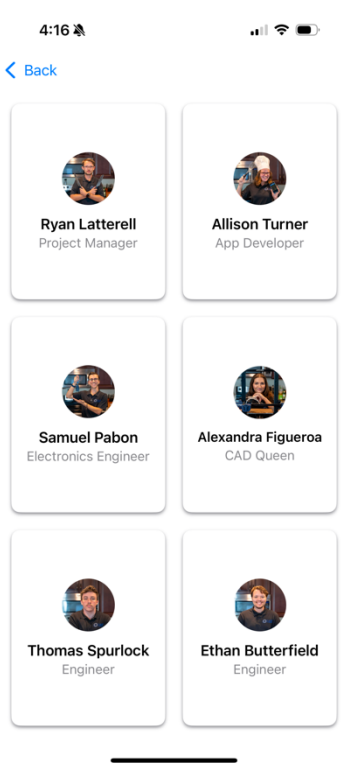
- **Objective:** Ensure that About View includes privacy information.
- **Steps:**
    1. Launch the app and navigate to the About tab.
    2. Verify that the privacy information exists and contains text.
- **Expected Result:** The text should render on the screen.
- **Result:** Passed

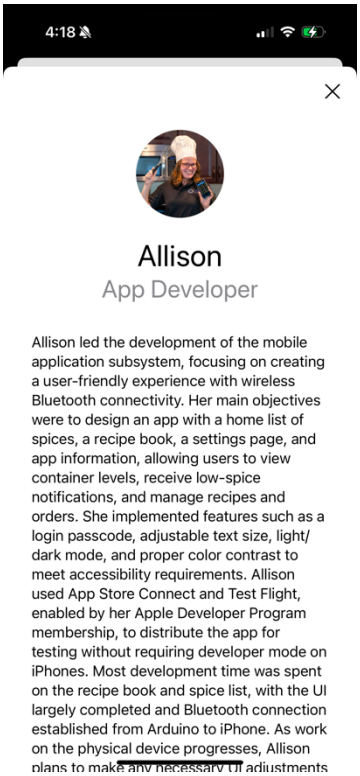


Test Case 5: About View – Meet the Team

- **Objective:** Ensure that About View includes information about the Flavor Fusion team.
- **Steps:**
    1. Launch the app and navigate to the About tab.
    2. Verify that the meet the team section exits.
    3. Tap on each team member and verify that the correct biography exists for each team member.
- **Expected Result:** Each team member should have the correct biography.
- **Result:** Passed

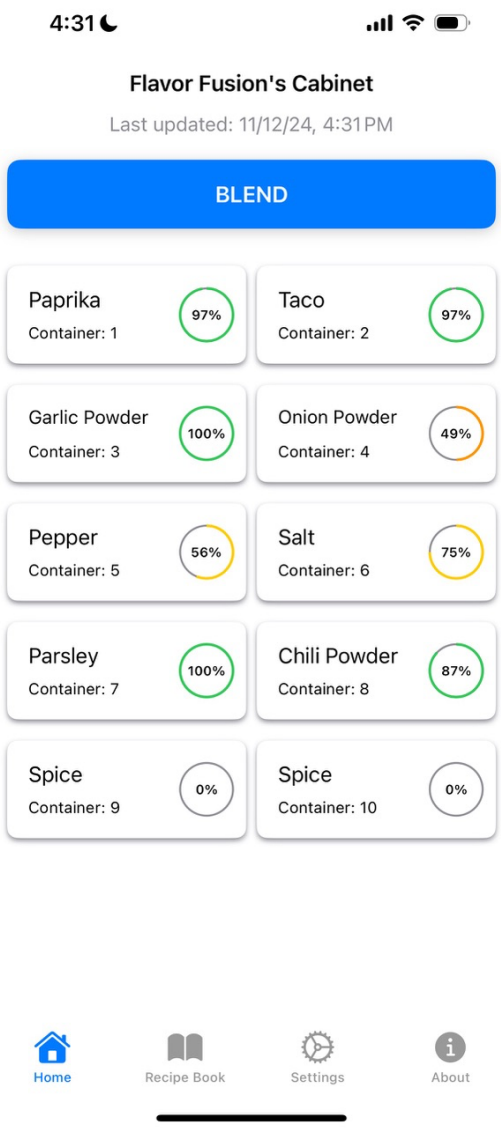






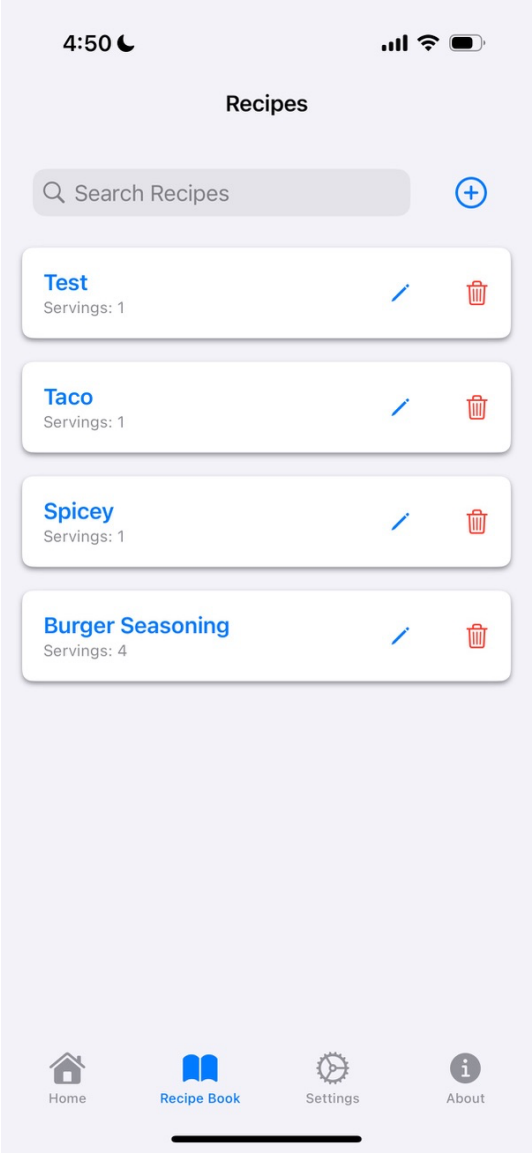
Test Case 6: Home List

- **Objective:** Ensure that the home list tab renders on the screen.
- **Steps:**
    1. Launch the app.
    2. Verify that the home list renders with 10 spice containers.
    3. Tap each spice container and verify that the detailed view renders.
- **Expected Result:** The home list displays 10 spice containers and their details.
- **Result:** Passed



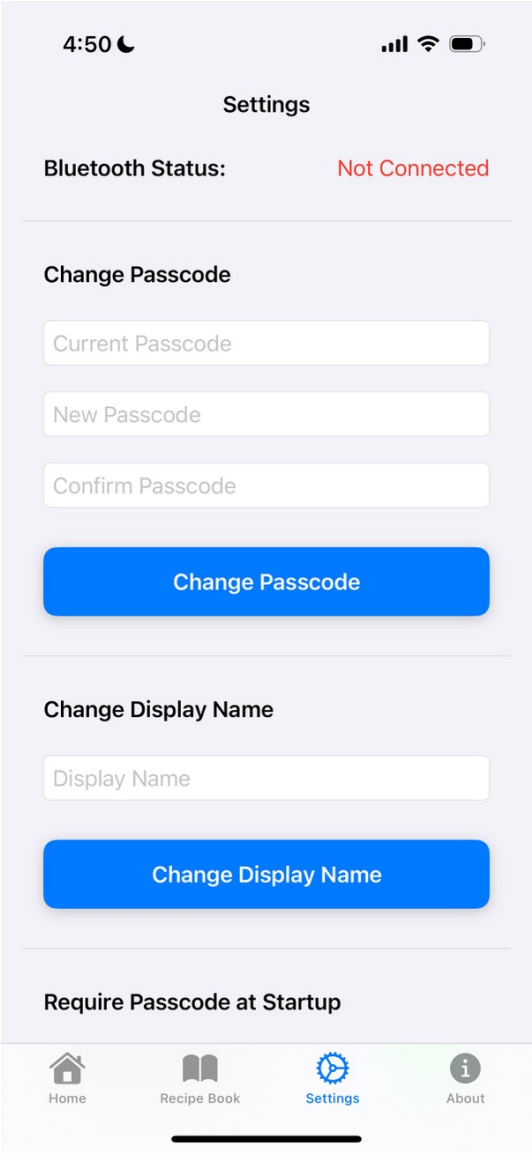
Test Case 7: Recipe Book

- **Objective:** Ensure that the recipe book tab renders on the screen.
- **Steps:**
    1. Launch the app.
    2. Verify that the recipe book renders with recipes.
- **Expected Result:** The recipe book displays recipes.
- **Result:** Passed



Test Case 8: Settings

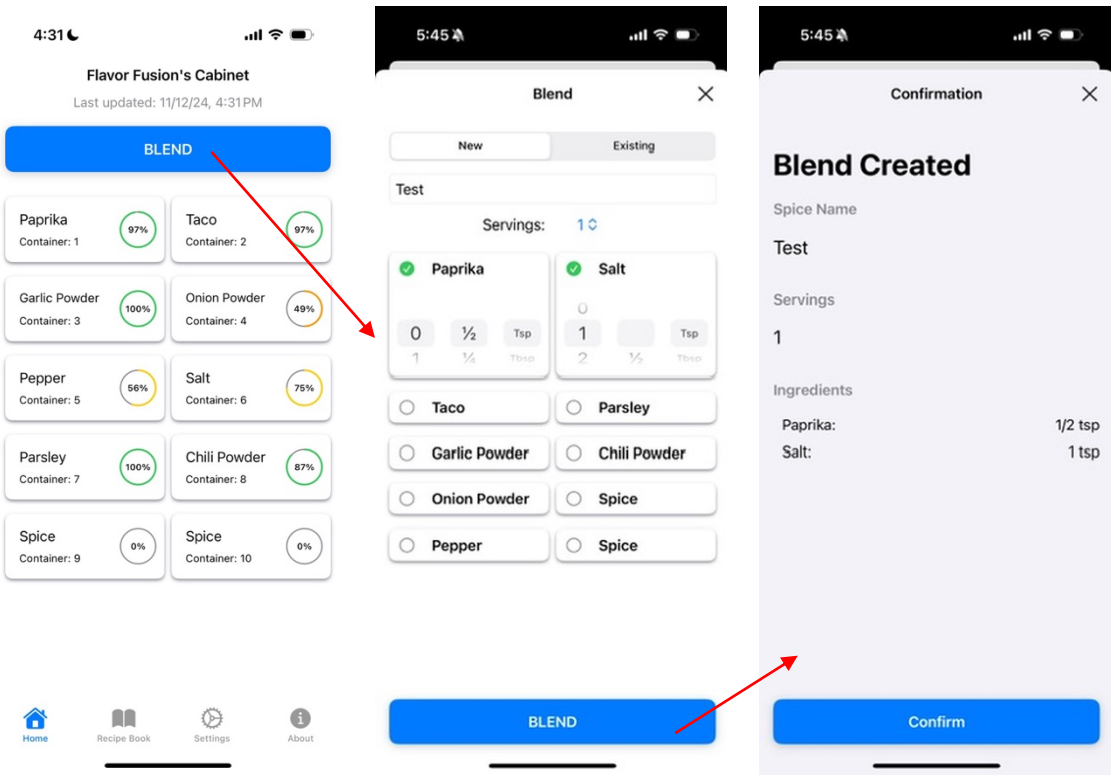
- **Objective:** Ensure that the settings tab renders on the screen.
- **Steps:**
    1. Launch the app.
    2. Verify that the settings renders with recipes.
- **Expected Result:** The settings page shows the ability to change display name, update passcode, and toggle require passcode at startup.
- **Result:** Passed



Blending Functional Tests

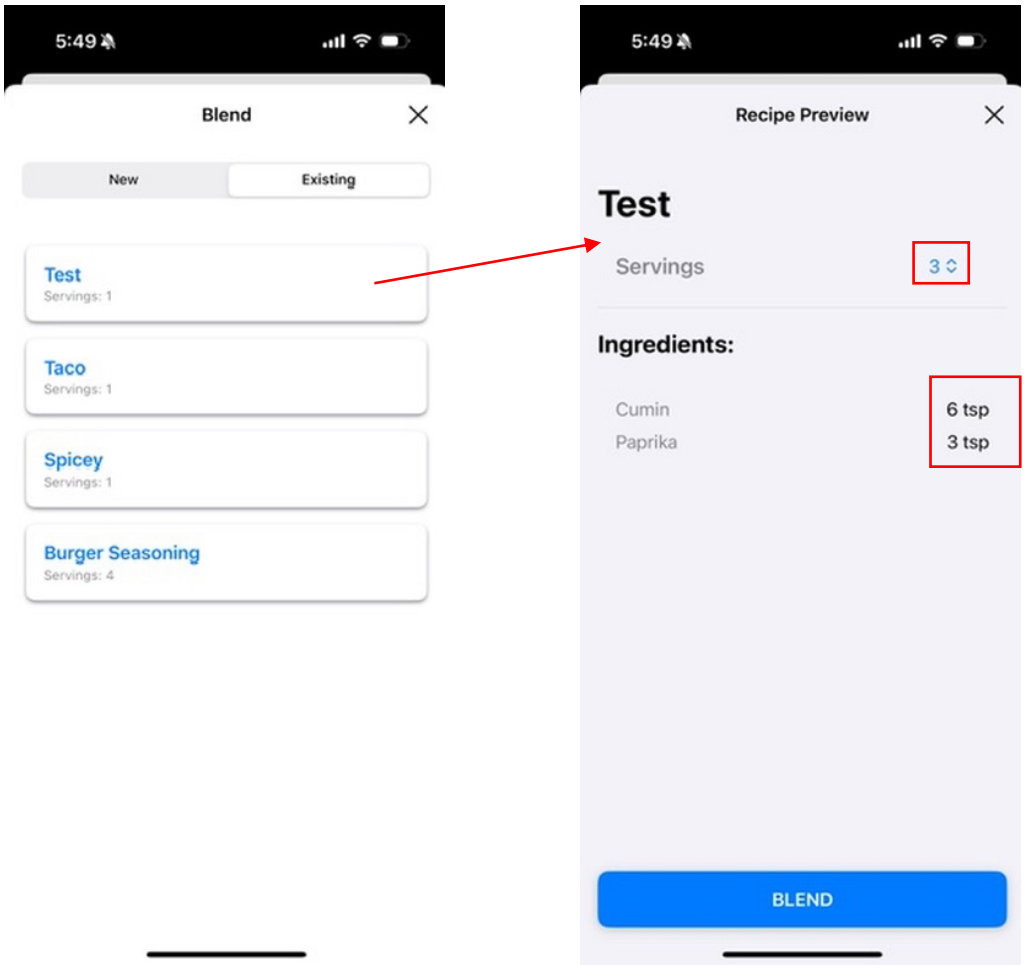
Test Case 1: Blend New Blend

- **Objective:** Verify that users can successfully create a new spice blend that is not saved in the recipe book.
- **Steps:**
    1. Navigate to the “New” section of the blend view.
    2. Select individual spices and specify spice amounts for each spice.
    3. Type in a spice name.
    4. Verify that all the spices are in the blend confirmation.
- **Expected Result:** All selected spices and amounts, spice name, and correct number of servings should be visible in the blend confirmation view.
- **Result:** Passed



Test Case 2: Blend Existing Blend

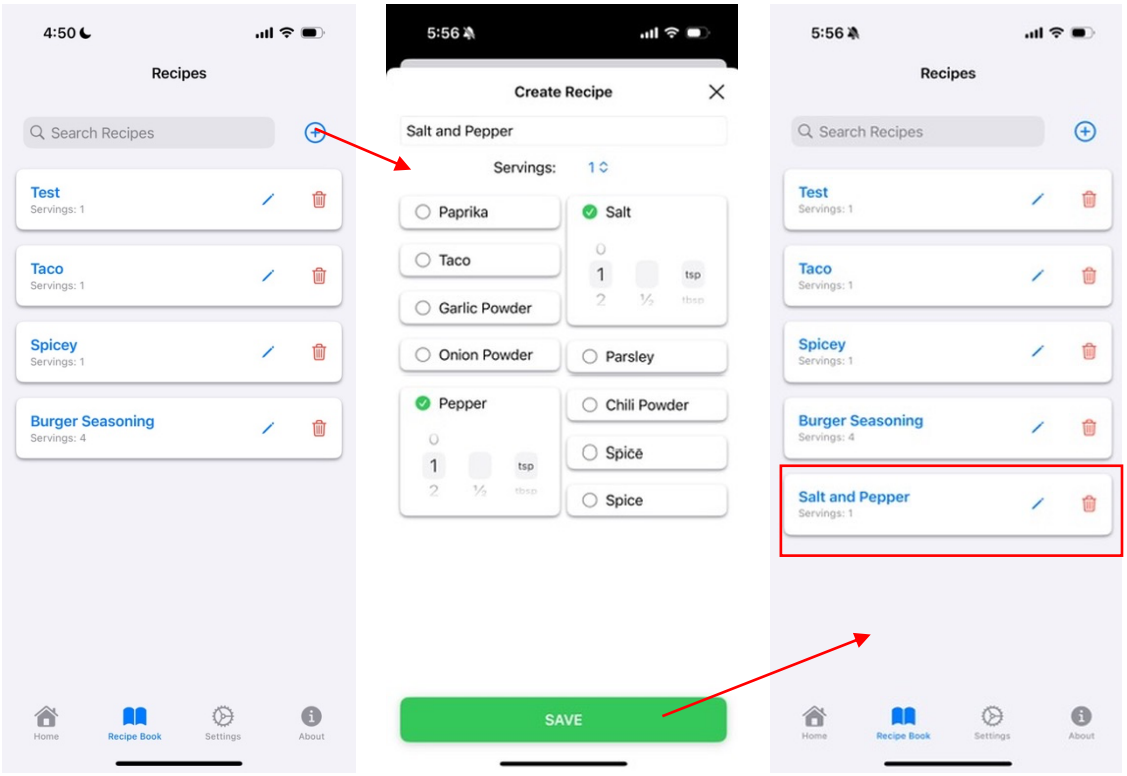
- **Objective:** Verify that users can select a spice blend from the recipe book for blending.
- **Steps:**
    1. Navigate to the “Existing” section of the blend view.
    2. Select a recipe.
    3. Modify the number of servings for the recipe.
- **Expected Result:** The ingredient amounts should match the number of servings.
- **Result:** Passed



Recipe Book Functional Tests

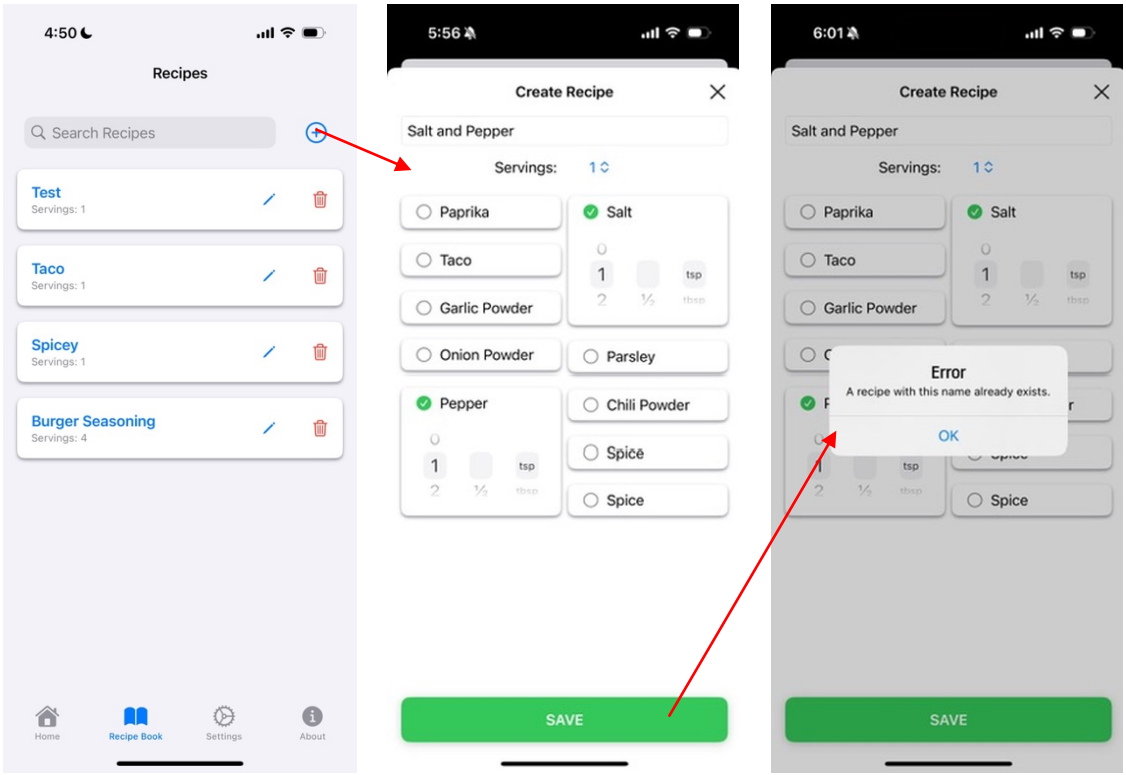
Test Case 1: Add a New Recipe

- **Objective:** Verify that users can successfully add a new recipe to the recipe book.
- **Steps:**
    1. Navigate to the "Add Recipe" screen from the recipe book.
    2. Enter valid details for the recipe.
    3. Navigate back to the recipe book list and check if the newly added recipe appears in the list.
- **Expected Result:** The new recipe is successfully added to the recipe book and is visible in the list with the correct details.
- **Result:** Passed



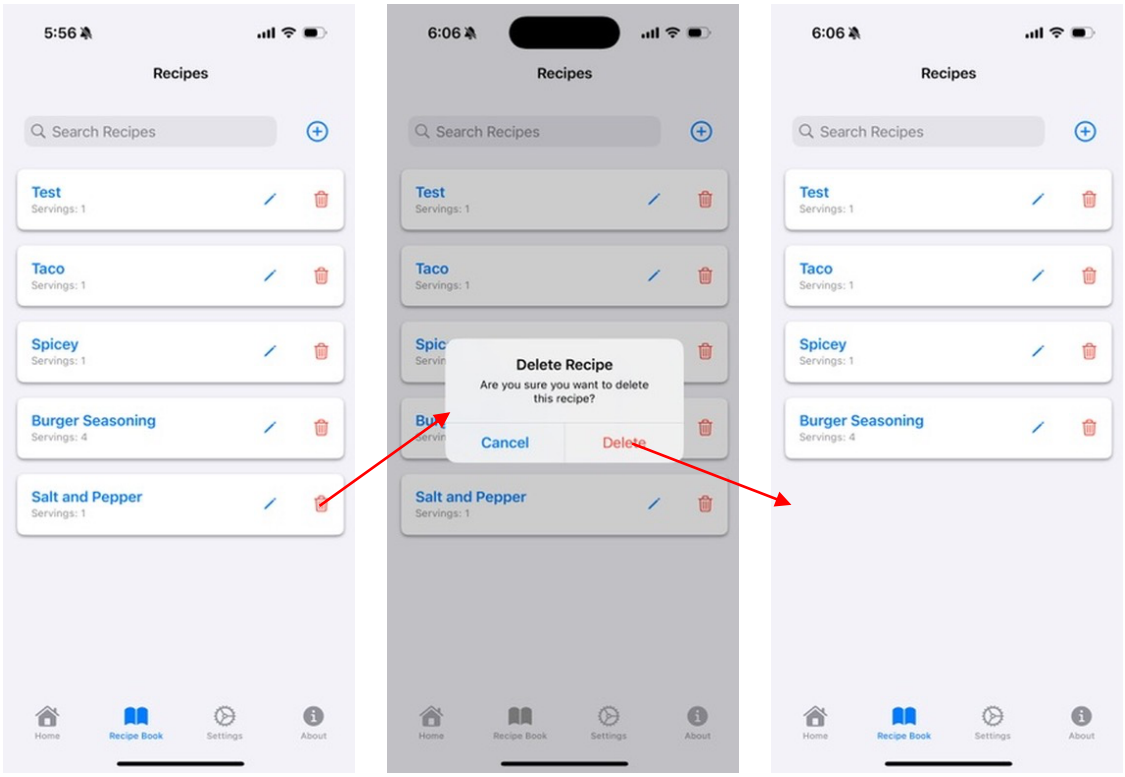
Test Case 2: Add Duplicate Recipe

- **Objective:** Verify that the app handles attempt to add a recipe with the same name as an existing recipe.
- **Steps:**
    1. Add a new recipe with a unique name.
    2. Attempt to add another recipe with the same name as the one just added.
    3. Check if the app allows the duplicate or provides a warning/error message.
- **Expected Result:** The app should prevent the duplicate recipe from being added and display a message.
- **Result:** Passed



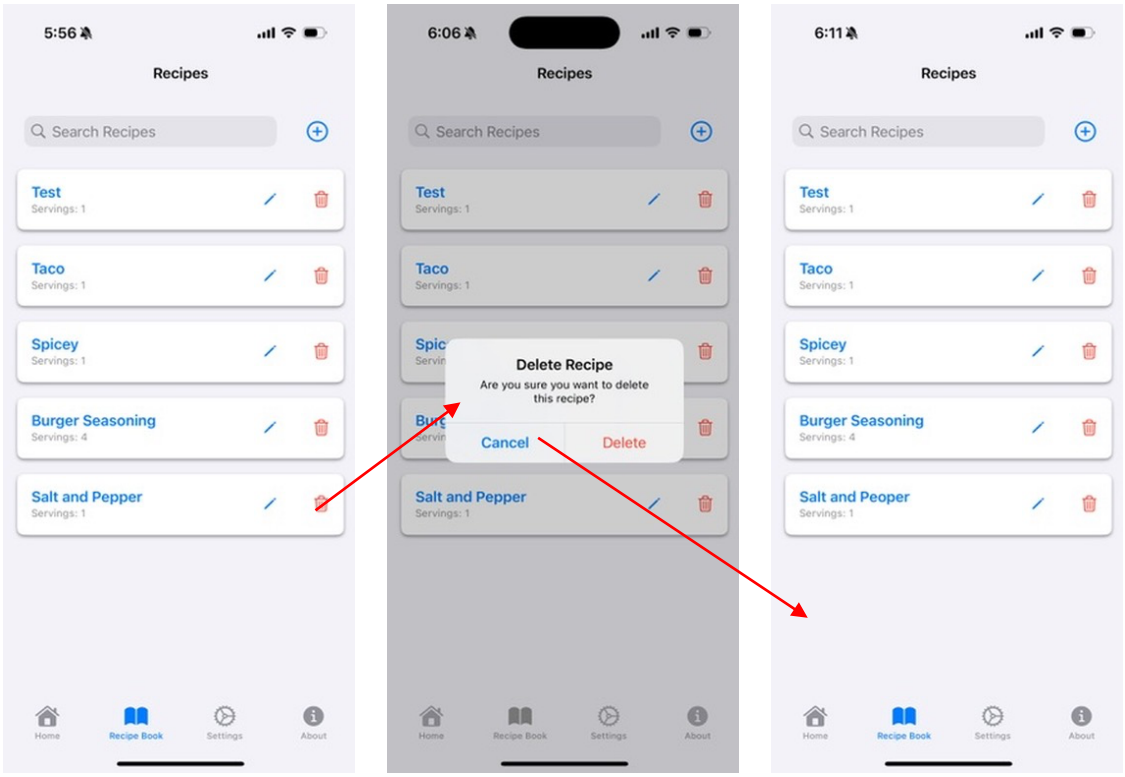
Test Case 3: Delete a Recipe

- **Objective:** Verify that users can delete a recipe from the recipe book.
- **Steps:**
    1. Navigate to the recipe book and select an existing recipe.
    2. Tap the "Delete" button associated with the recipe.
    3. Confirm the deletion when prompted.
    4. Return to the recipe book list and check if the recipe has been successfully removed.
- **Expected Result:** The selected recipe is successfully deleted from the recipe book and no longer appears in the list.
- **Result:** Passed



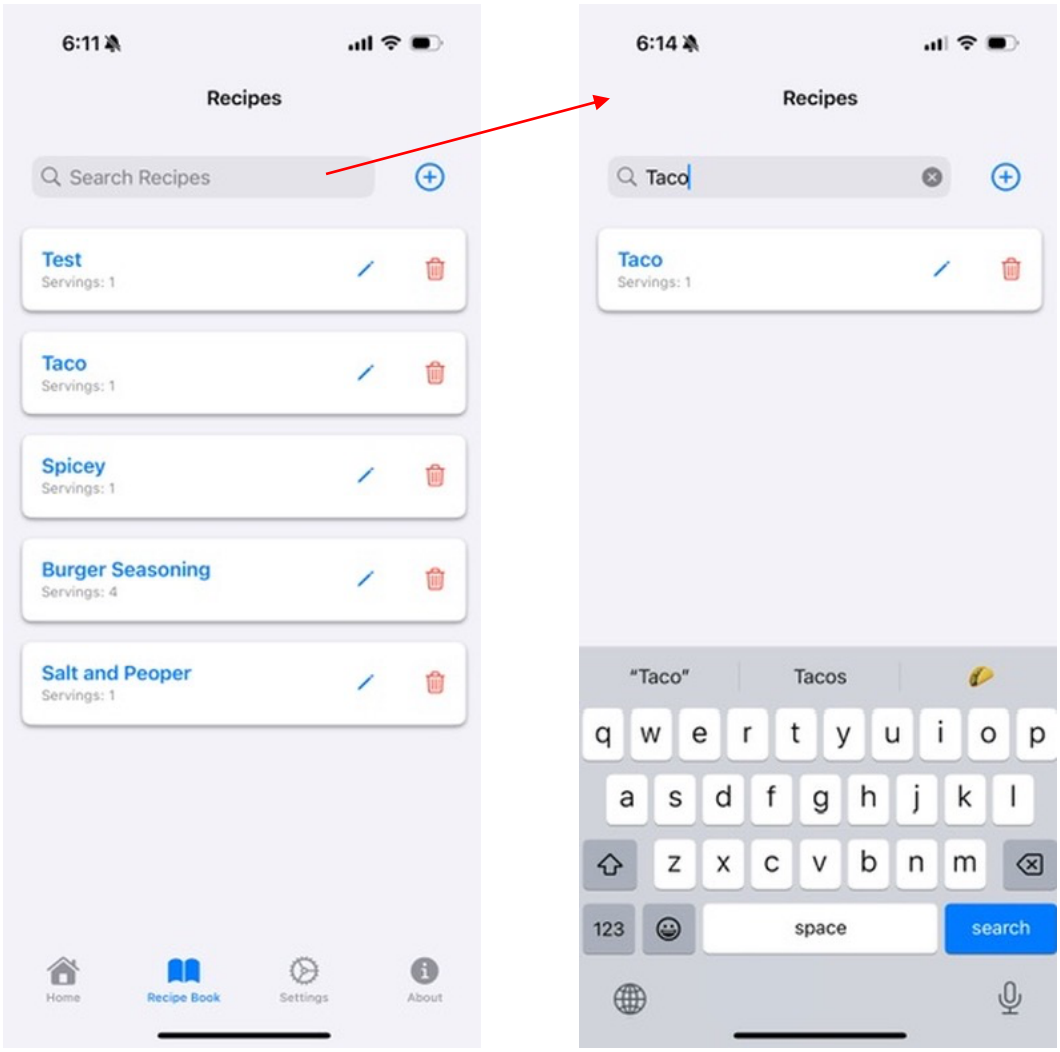
Test Case 4: Cancel Recipe Deletion

- **Objective:** Ensure that users can cancel the recipe deletion process.
- **Steps:**
    1. Navigate to the recipe book and select a recipe to delete.
    2. Tap the "Delete" button and when prompted to confirm, choose "Cancel."
    3. Check if the recipe remains in the recipe book after cancellation.
- **Expected Result:** The deletion process is canceled, and the recipe remains in the recipe book without being removed.
- **Result:** Passed



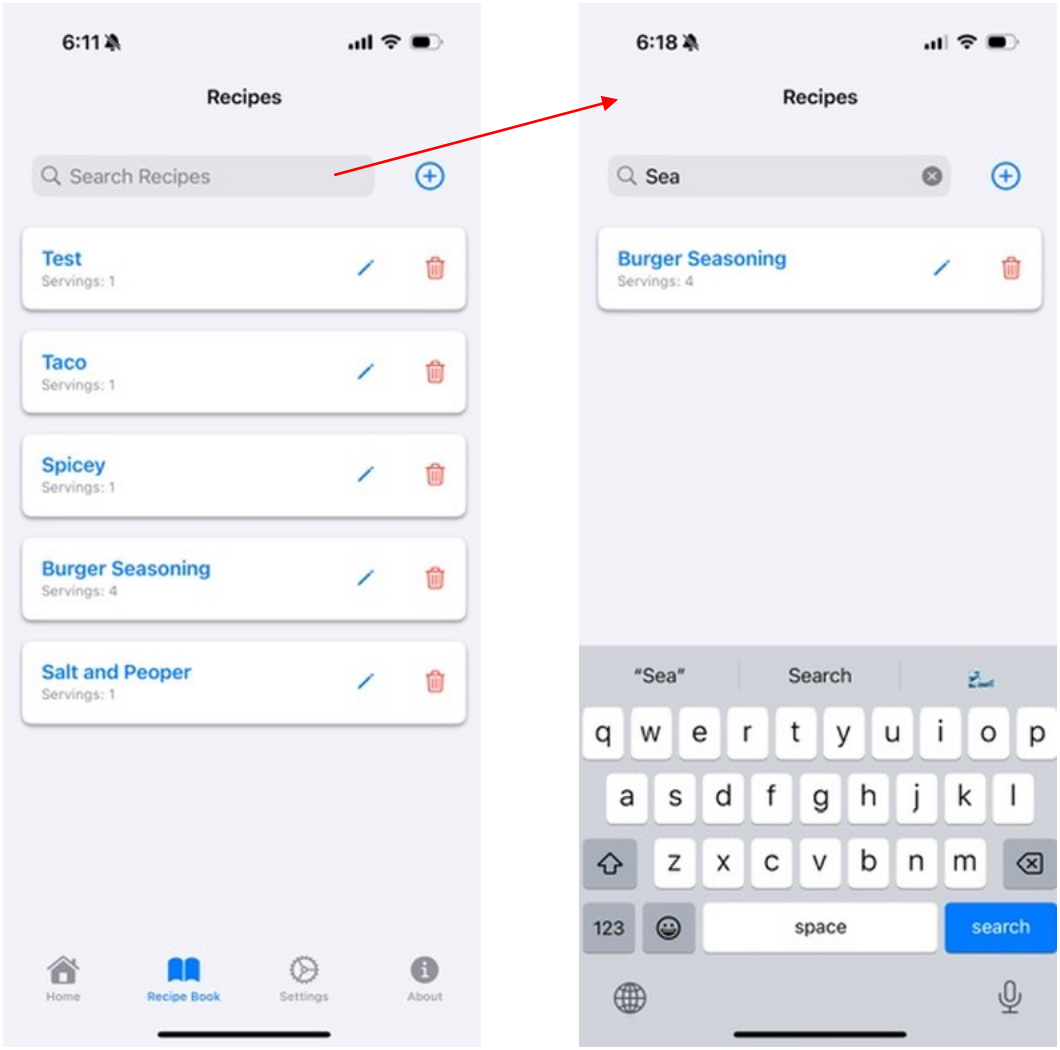
Test Case 5: Search for a Recipe by Name

- **Objective:** Ensure that users can search for recipes by their name and retrieve relevant results.
- **Steps:**
    1. Navigate to the recipe book and enter a valid recipe name into the search bar.
    2. Execute the search.
    3. Review the search results to verify if the relevant recipe(s) is/are displayed.
- **Expected Result:** The app returns the correct recipe(s) that match the entered search term, and no irrelevant results are shown.
- **Result: Passed**



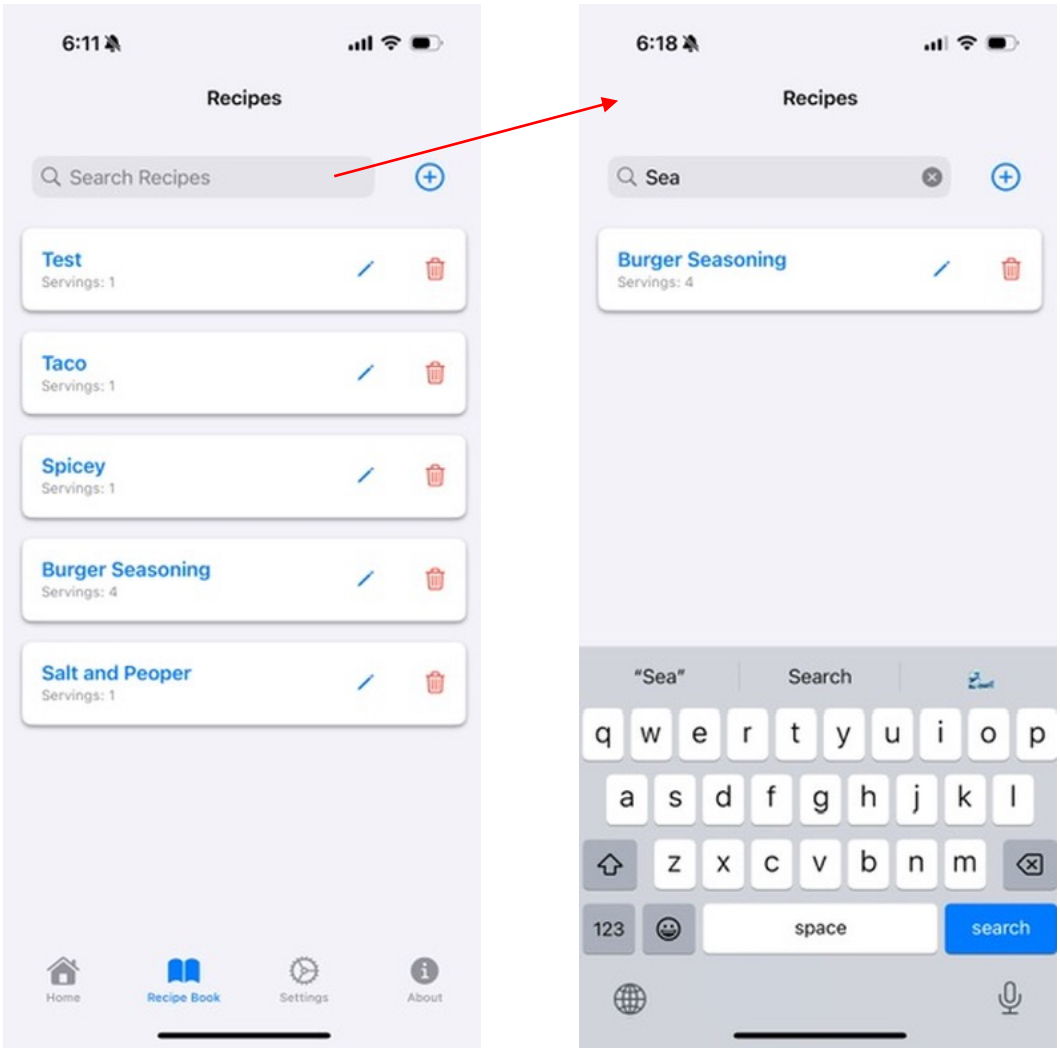
Test Case 6: Search with Partial Recipe Name

- **Objective:** Verify that users can perform a search using partial names and still retrieve relevant results.
- **Steps:**
    1. Navigate to the recipe book and enter a partial name of an existing recipe.
    2. Execute the search.
    3. Check if the results include recipes with names that contain the partial search term.
- **Expected Result:** The app returns all recipes that contain the partial name entered, making the search flexible and user-friendly.
- **Result:** Passed



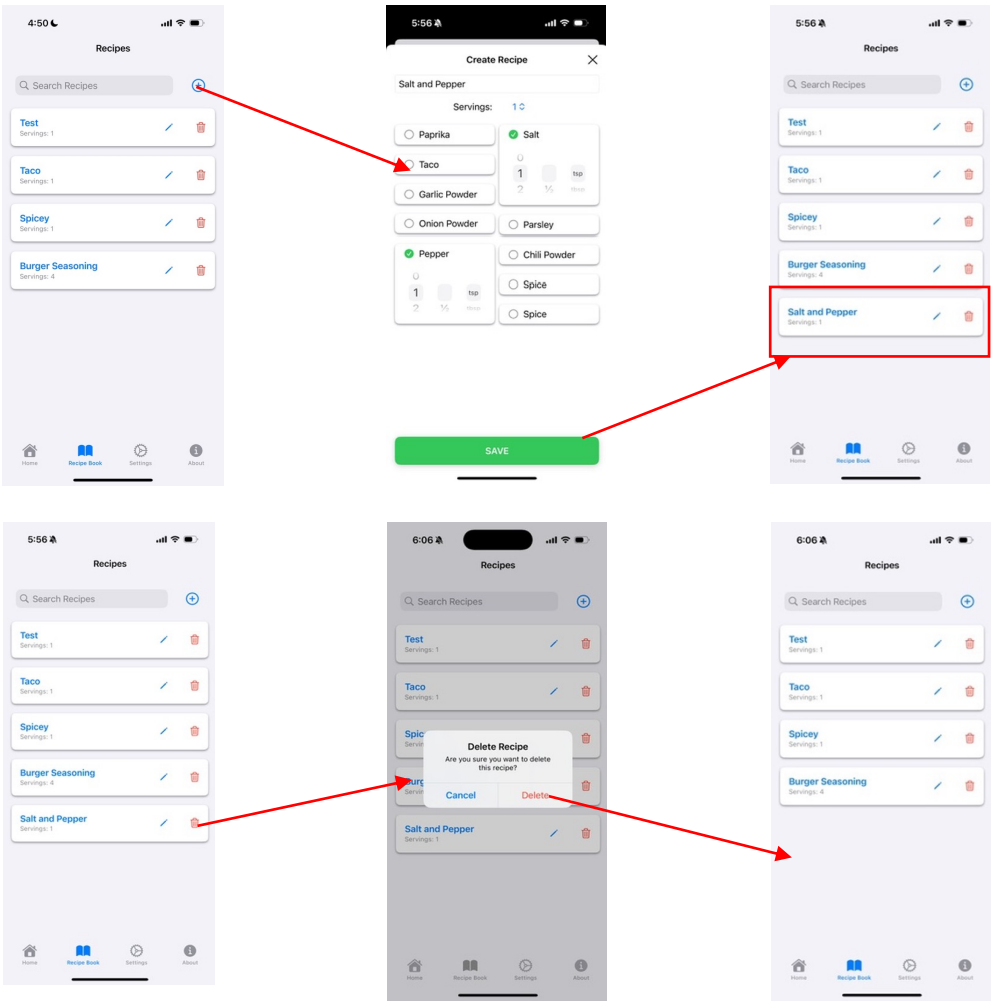
Test Case 7: Case Sensitivity in Search

- **Objective:** Verify that the search function is not case-sensitive and returns results regardless of uppercase or lowercase input.
- **Steps:**
    1. Search for a recipe by entering its name in uppercase.
    2. Repeat the search with the same name in lowercase.
    3. Check if both searches return the same results.
- **Expected Result:** The app's search function should not be case-sensitive, and both uppercase and lowercase searches should return the same results.
- **Result:** Passed



Test Case 8: Add, Delete, and Search Integration Test

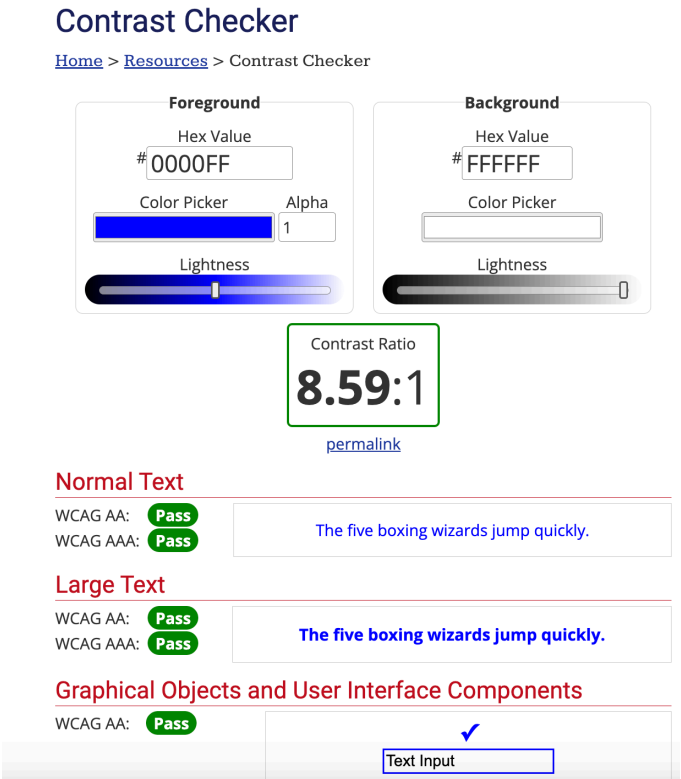
- **Objective:** Ensure that the add, delete, and search functionalities work together seamlessly.
- **Steps:**
    1. Add a new recipe to the recipe book.
    2. Search for the newly added recipe to verify that it appears in the results.
    3. Delete the recipe.
    4. Perform another search to ensure the deleted recipe no longer appears.
- **Expected Result:** The newly added recipe should appear in the search results after being added, and after deletion, it should no longer be found in the recipe book or search results.
- **Result:** Passed



Accessibility Testing

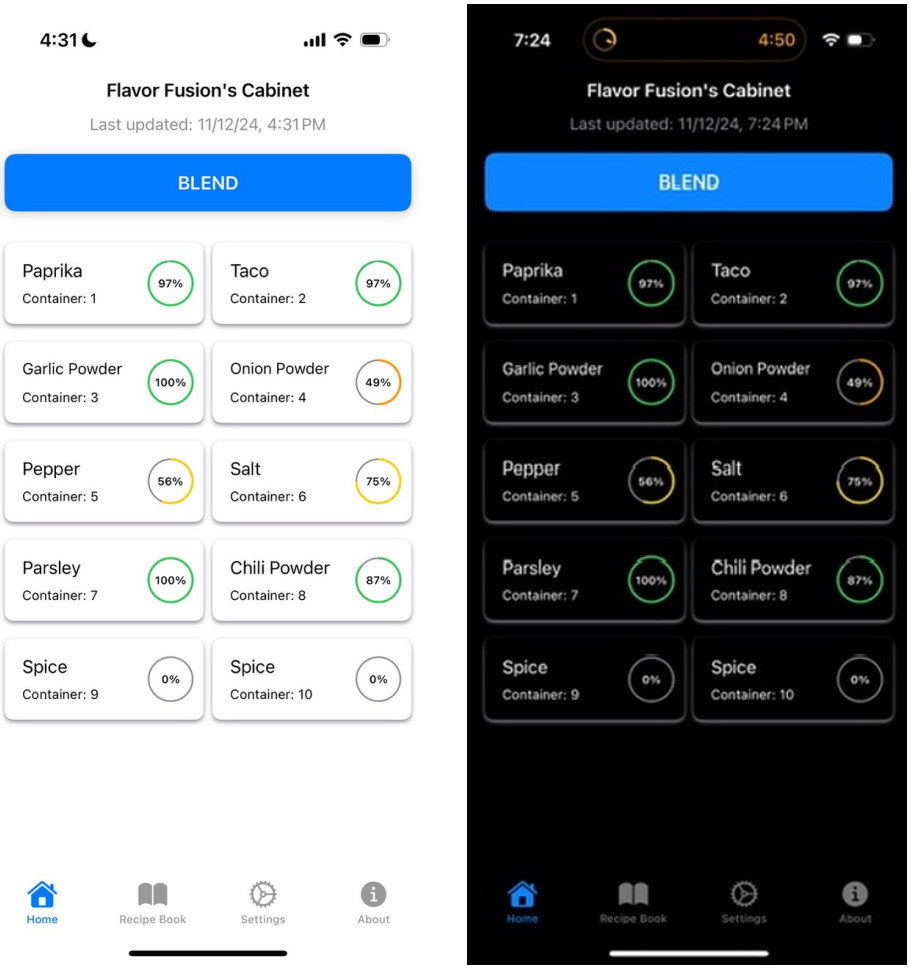
Test Case 1: Color Contrast

- **Objective:** Ensure that text and important UI elements have sufficient color contrast against their background for users with visual impairments.
- **Steps:**
    1. Use a color contrast analyzer tool to check the contrast ratio of text and interactive elements (buttons, links) against the background.
    2. Test in both light and dark mode to ensure contrast remains sufficient in both modes.
- **Expected Result:** The contrast ratio meets WCAG (Web Content Accessibility Guidelines) standards, with a minimum contrast ratio of 4.5:1 for regular text and 3:1 for large text and UI components.
- **Result:** Passed



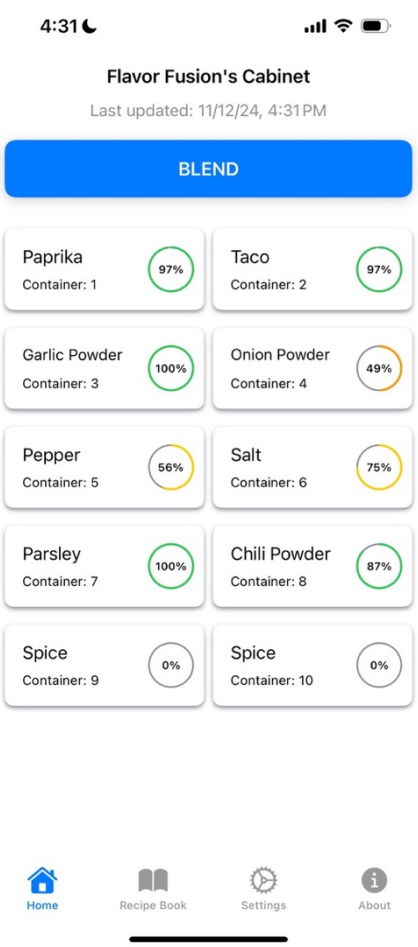
Test Case 2: Light/Dark Mode Compatibility

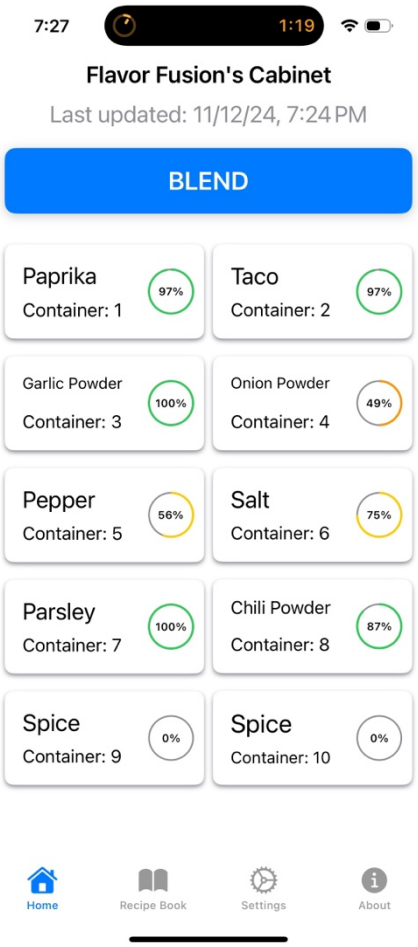
- **Objective:** Verify that the app is accessible and usable in both light and dark modes.
- **Steps:**
    1. Switch between light mode and dark mode in the iPhone's display settings.
    2. Navigate through the app in both modes, paying attention to background colors, text visibility, and element contrast.
- **Expected Result:** The app's UI elements maintain clear visibility and proper contrast in both light and dark modes, with no readability issues or broken design elements.
- **Result:** Passed



Test Case 3: Font Scaling

- **Objective:** Ensure that the app's UI is responsive to dynamic font size changes for users who require larger or smaller text.
- **Steps:**
    1. Go to the iPhone's accessibility settings and adjust the font size to both the smallest and largest available options.
    2. Open the app and navigate through all screens.
    3. Verify that text remains readable, does not overflow and that the layout adjusts properly to accommodate larger or smaller font sizes.
- **Expected Result:** The UI adapts to all font scaling settings without layout issues, and text remains fully readable and accessible.
- **Results:** Passed

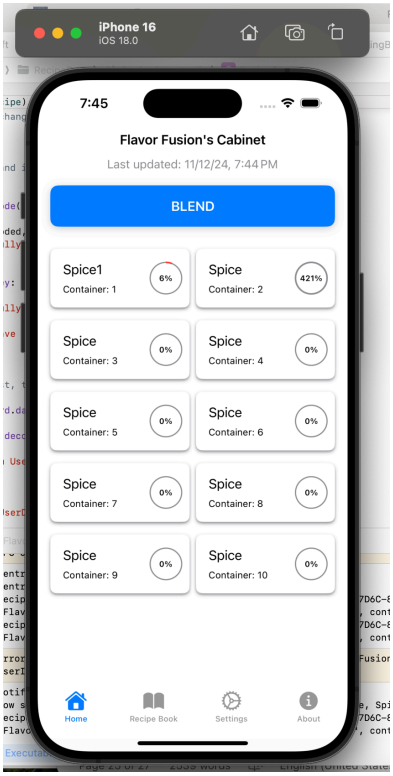


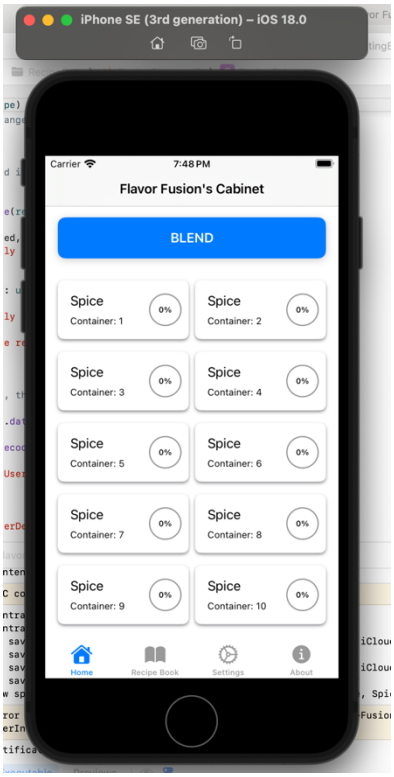


Compatibility Testing

Test Case 1: Layout Consistency Across Different iPhone Models

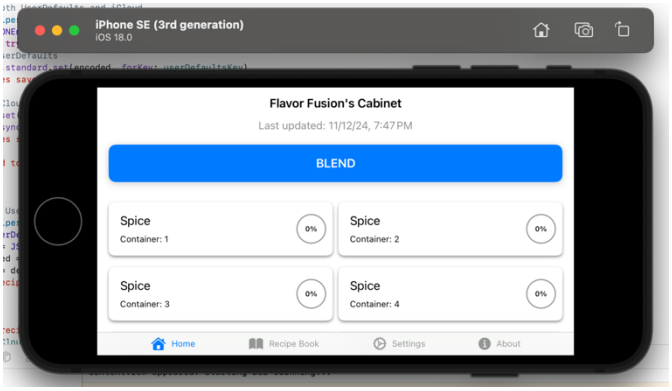
- **Objective:** Ensure that the app's layout and UI elements are consistent and responsive across different iPhone models and screen sizes.
- **Steps:**
    1. Run the app on different iPhone simulators and real iPhones (e.g., iPhone SE, iPhone 15 Pro Max, iPhone 12 Mini, etc.).
    2. Launch the app and navigate through all screens, focusing on layout and UI components like buttons, text fields, images, and menus.
    3. Compare the layout across devices and confirm that no UI elements are distorted, cut off, or misaligned.
- **Expected Result:** The app's layout and UI components adjust properly to different screen sizes, with no elements being distorted or out of place.
- **Result:** Passed

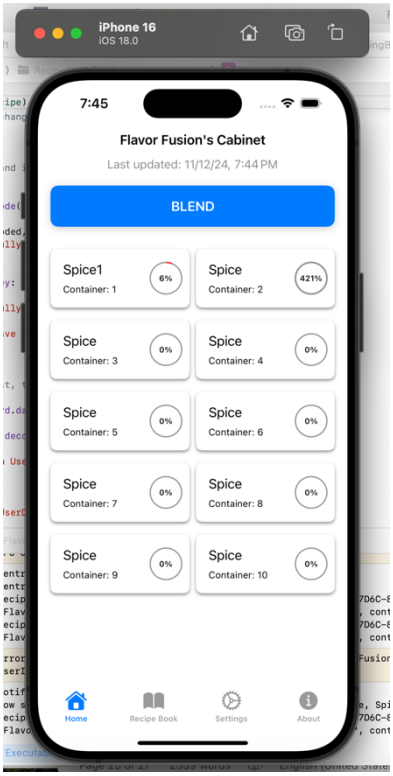




Test Case 2: Landscape and Portrait Mode Compatibility

- **Objective:** Ensure that the app functions correctly in both portrait and landscape orientations on different iPhones.
- **Steps:**
    1. Run the app on different iPhone simulators and real iPhones (e.g., iPhone 15, iPhone 12 Mini, iPhone SE).
    2. Switch between portrait and landscape mode while using the app.
    3. Verify that the UI elements adjust appropriately to the new orientation and no content is distorted or lost.
- **Expected Result:** The app adapts smoothly between portrait and landscape orientations across all iPhones, with no layout issues or distorted content.
- **Result:** Passed

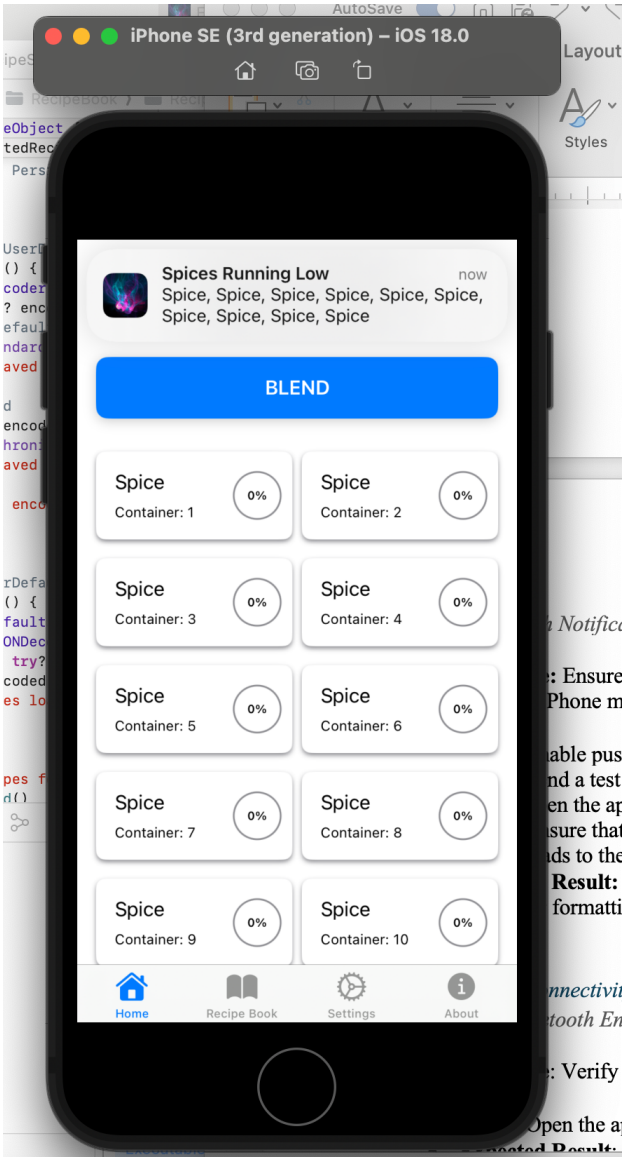




Test Case 3: Push Notification Compatibility Across Models

- **Objective:** Ensure that push notifications are received and displayed correctly across different iPhone models.
- **Steps:**
    1. Enable push notifications for the app on various iPhones.
    2. Send a test notification and check its appearance and functionality (e.g., tap to open the app).
    3. Ensure that the notification appears properly formatted and that it is tappable and leads to the correct screen within the app.
- **Expected Result:** Push notifications are received and displayed properly on all iPhones, with clear formatting and correct navigation behavior when tapped.

- **Result:** Passed



Bluetooth and Connectivity Testing

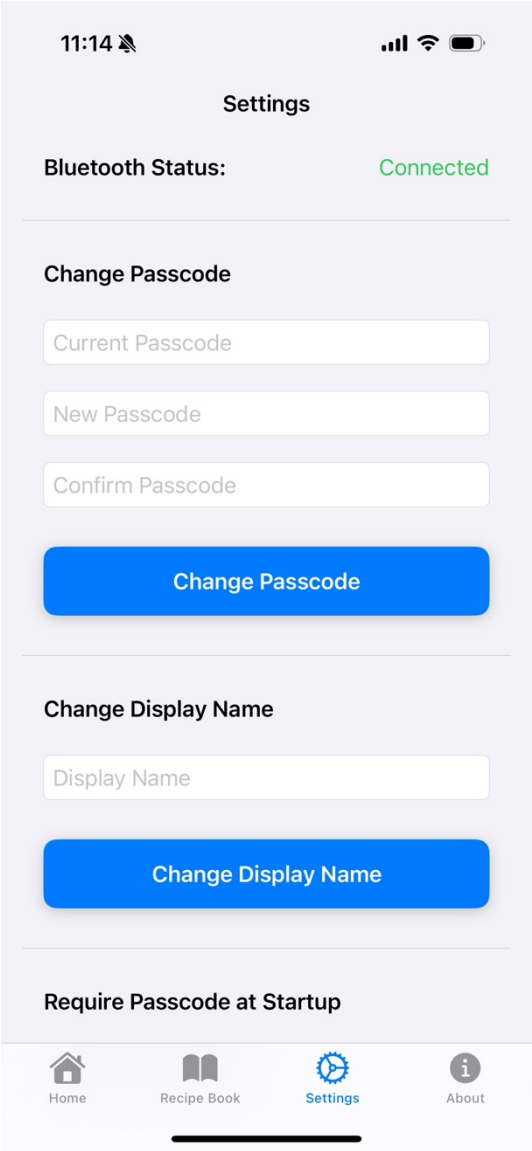
Test Case 1: Bluetooth Enabled on Device

- **Objective**: Verify that the app prompts the user to enable Bluetooth if it is disabled.
- **Steps**:
    1. Open the app.
- **Expected Result**: The app prompts the user to enable Bluetooth.
- **Result**: Passed


11:07
TestFlight
Create a Passcode
Allow “Flavor Fusion” to find Bluetooth devices?
This app uses Bluetooth to connect to and communicate with your spice containers.
[TV] Samsung 8...
Allison's Apple W...
Allison's AirPods...
Tribit MoveBuds...
More than 50 devices found
Information from Bluetooth devices can be used to determine your location and create a profile of you.
Don't Allow
Allow
Create Passcode

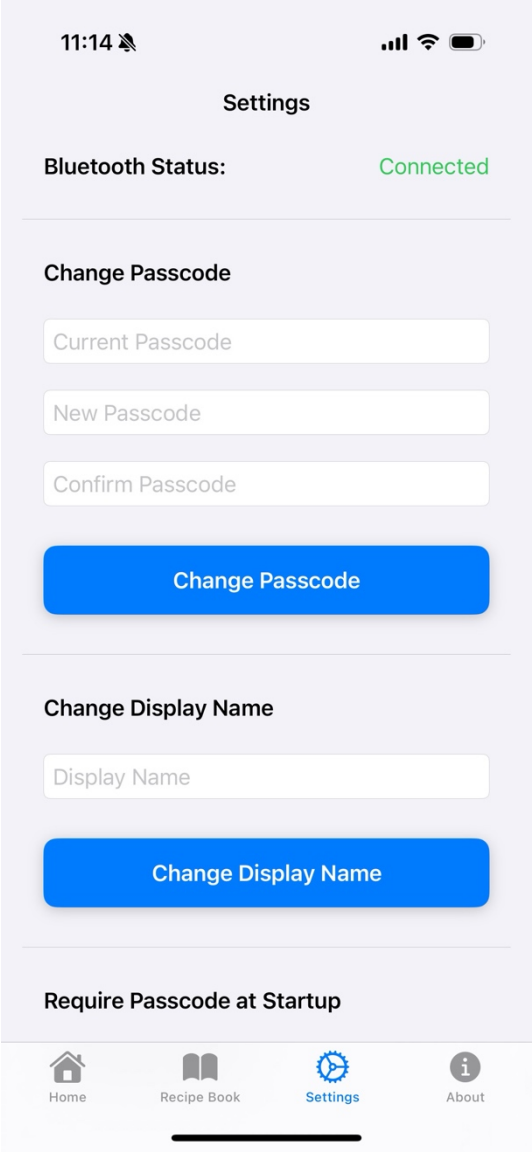
Test Case 2: Device Scanning for Spice Maker

- **Objective**: Ensure that the app successfully scans and finds the Spice Maker (Arduino) device.
- **Precondition**: Spice Maker is powered on and within range, Bluetooth enabled on iPhone.
- **Steps**:
    1. Open the app.
    2. Start the device scanning process.
- **Expected Result**: The app detects the Spice Maker.
- **Result**: Passed



Test Case 3: Successful Connection to Spice Maker

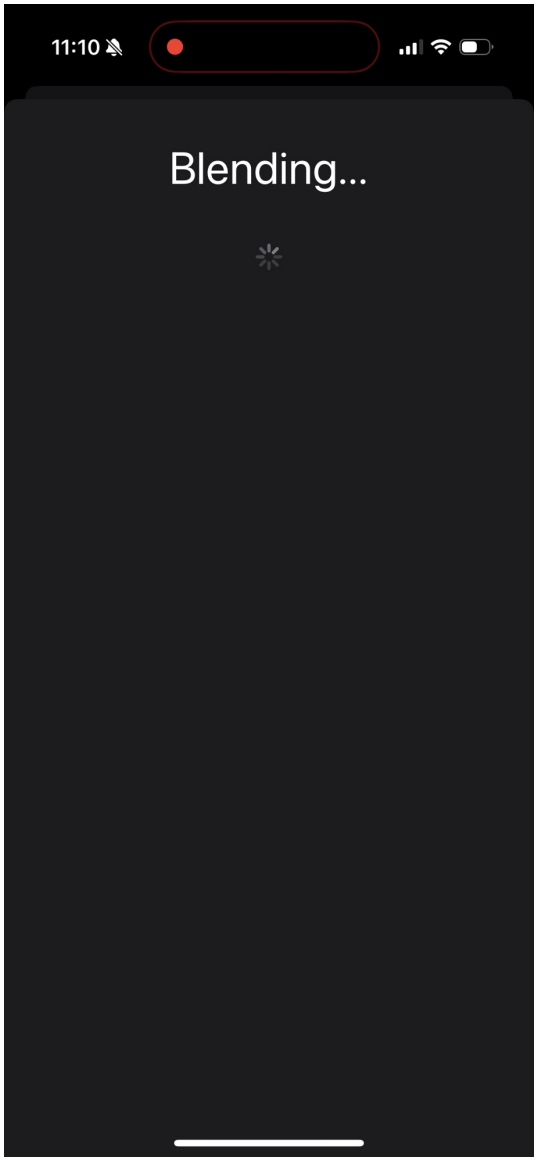
- **Objective**: Verify that the app successfully establishes a connection to the Spice Maker.
- **Steps**:
    1. Open the app.
    2. Scan for the Spice Maker device.
    3. Tap to connect to the device.
- **Expected Result**: The app successfully connects to the Spice Maker, and a "Connected" status is displayed in the console on Xcode.
- **Result**: Passed

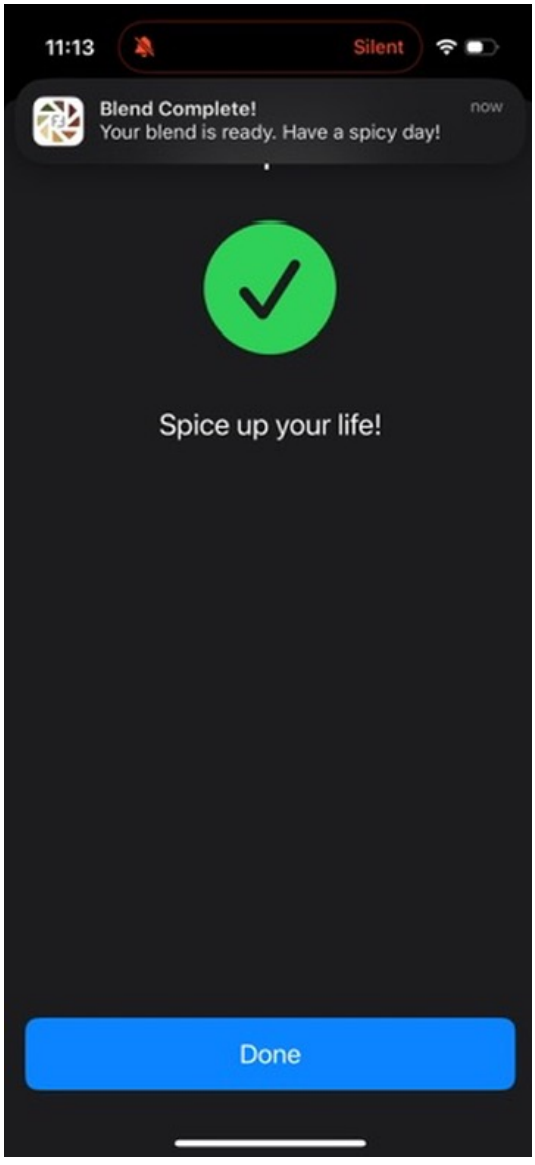


Data Transfer Testing

Test Case 1: Successful Sending of Blend Data

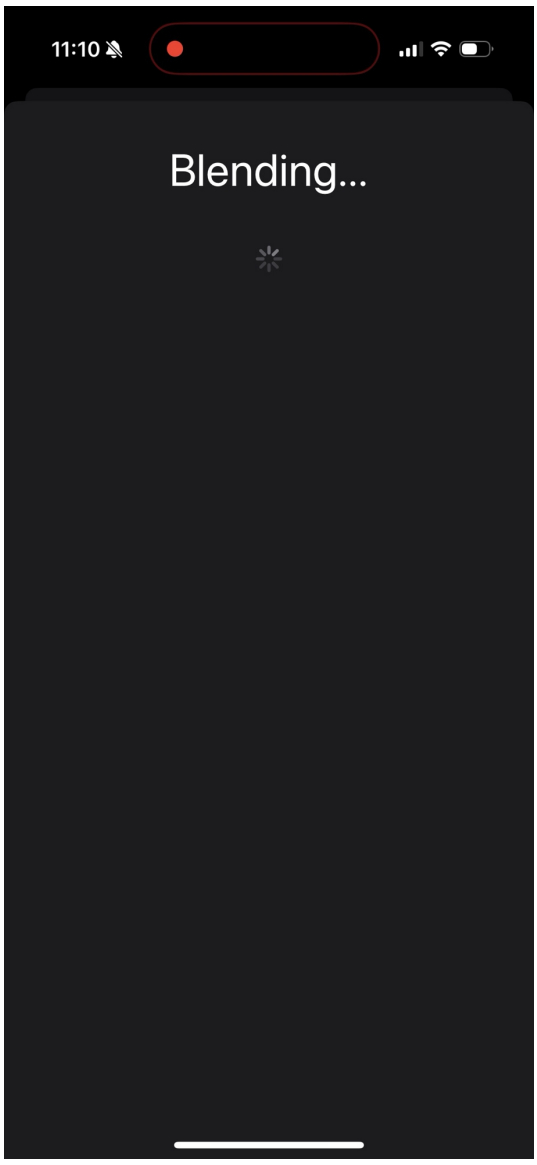
- **Objective**: Verify that the blend data is successfully sent to the Arduino over Bluetooth.
- **Steps**:
    1. Ensure the app is connected to the Arduino via Bluetooth.
    2. Create or select a blend in the app.
    3. Tap the “Blend” Button.
    4. Verify that the blend is shown in the serial monitor on the Arduino.
- **Expected Result**: The app successfully sends the blend data to the Arduino, and the Arduino confirms receipt.
- **Result**: Passed

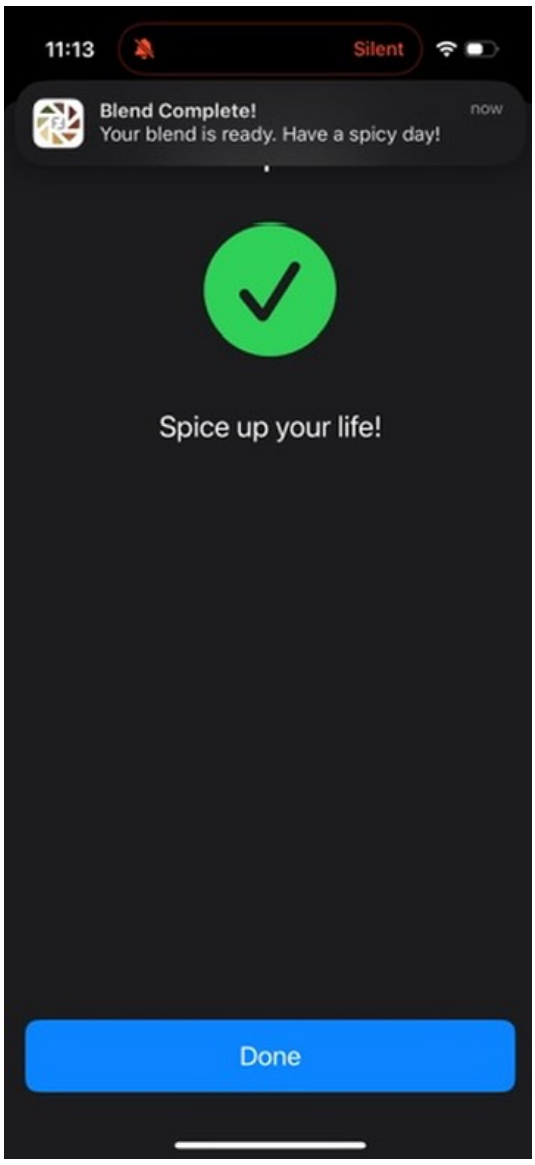




Test Case 2: Successful Sending of Blend Data with large Payload

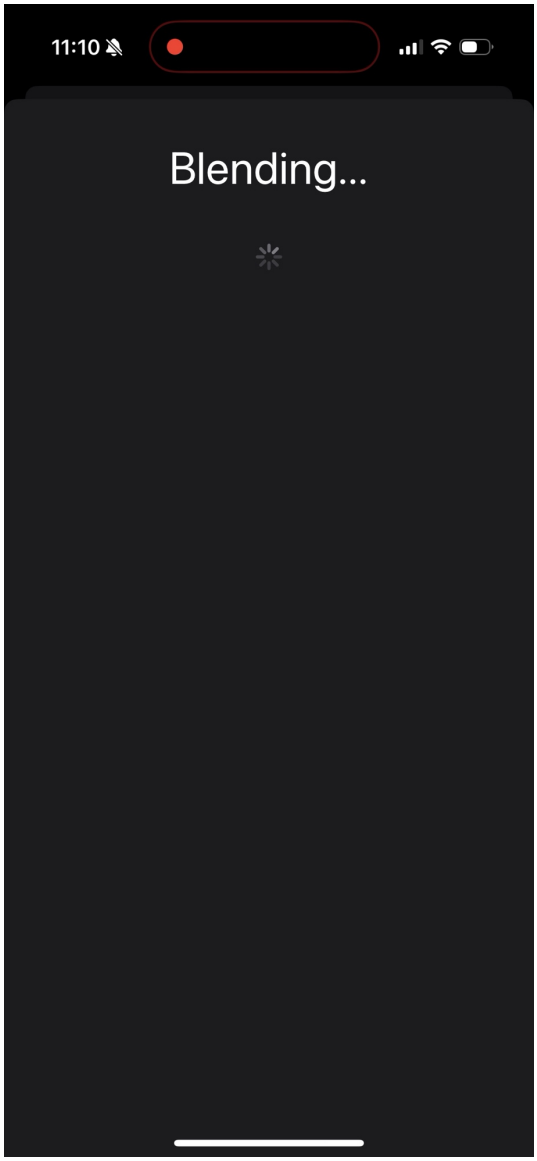
- **Objective**: Ensure that large blend data (multiple spices with complex details) can be sent successfully to the Arduino.
- **Steps**:
    1. Ensure the app is connected to the Arduino via Bluetooth.
    2. Create a blend with many spices and amounts.
    3. Tap the “Blend” Button.
    4. Verify that the blend is shown in the serial monitor on the Arduino.
- **Expected Result**: The blend data is sent without any delay or failure, and the Arduino confirms receipt of the entire payload.
- **Result**: Passed

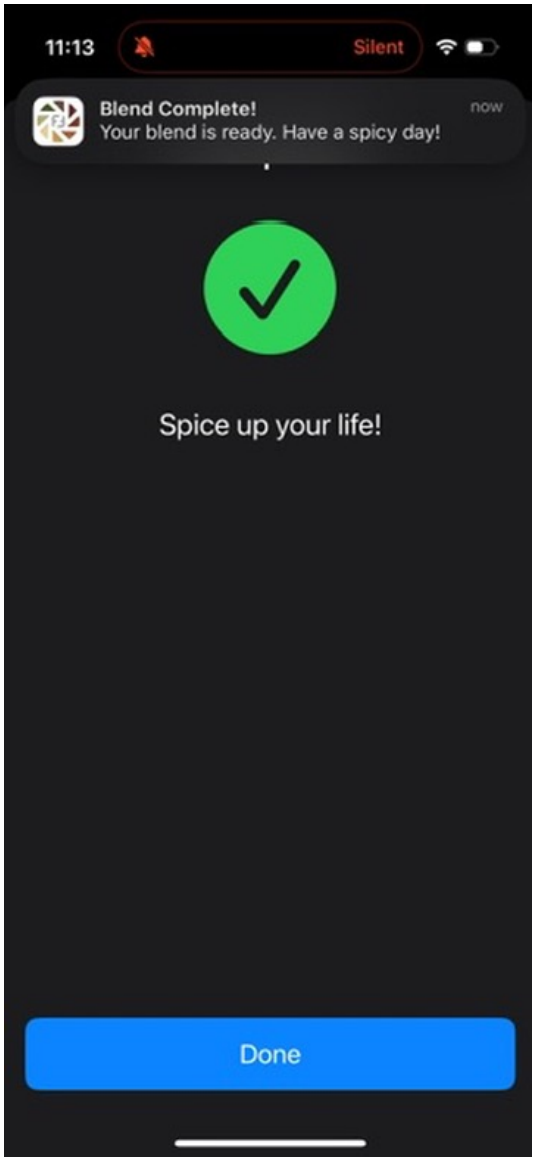




Test Case 3: Successful Switch to Blend Complete View on Data Receipt

- **Objective**: Verify that the app switches to the "Blend Complete" view when the Arduino sends the blend complete message.
- **Steps**:
    1. Ensure the app is connected to the Arduino via Bluetooth.
    2. Start the blending process on the app.
    3. The Arduino completes the blending and sends a "Blend Complete" message.
- **Expected Result**: The app switches to the "Blend Complete" view, displaying the blend completion message.
- **Result**: Passed





Actuation Subsystem Testing

- The actuation subsystem consists of all electronics directly necessary for motion in Flavor Fusion. An A4988 stepper driver and NEMA 11 stepper motor control the linear rail, a TMC2209 and NEMA 8 control the auger, and a TMC2209 and NEMA 17 drive the spice carriage.

Stepper Motors

- ***Test Case 1: Linear Rail NEMA 11 and A4988***

- **Objective:** Control rail to move NEMA 8 bidirectionally.
- **Steps:**
  - Connect motor and driver to power and Arduino. Check pins, connection types, and secure exposed wires.
  - Check power supply to verify input voltage and current draw.
  - Write a function in Arduino IDE for motor actuation.
  - Conduct 5+ trials to move various distances.
- **Expected Result:** NEMA 8 can insert into and extract from auger drive shaft.
- **Result: PASSED**

- ***Test Case 2: Auger Driver NEMA 8***

- **Objective:** Control NEMA 8 to dispense spice.
- **Steps:**
  - Connect motor and driver to power and Arduino. Check pins, connection types, and secure exposed wires.
  - Check power supply to verify input voltage and current draw.
  - Write a function in Arduino IDE for motor actuation.
  - Conduct 5+ trials to move various angles and speeds.
- **Expected Result:** Motor can turn at various speeds for efficiency and accuracy. Exit angle of motor is controlled for shaft interfacing.
- **Result: FAILED – Motor works, but does not dispense spice.**

- ***Test Case 3: Carriage Driver NEMA 17***

- **Objective:** Control NEMA 17 to rotate spice carriage.
- **Steps:**
  - Connect motor and driver to power and Arduino. Check pins, connection types, and secure exposed wires.
  - Check power supply to verify input voltage and current draw.
  - Write a function in Arduino IDE for motor actuation.
  - Conduct 5+ trials to rotate to various positions.

- **Expected Result:** NEMA17 can rotate the carriage until the calibration point is reached, then turn to a desired location with enough precision to insert the auger drive shaft.
- **Result: PASSED**

Stepper Drivers

- ***Test Case 1: TMC2209 UART microstepping***

- **Objective:** Use serial communication to improve microstepping with TMC2209 driver.
- **Steps:**
  - Connect motor and driver to power and Arduino. Check pins, connection types, and secure exposed wires.
  - Check power supply to verify input voltage and current draw.
  - Write a function in Arduino IDE for precise UART control.
  - Conduct 5+ trials to rotate motors at various step sizes.
- **Expected Result: The TMC2209 can drive motors at various rates and step sizes to prioritize speed vs precision during dispensing.**
- **Result: FAILED: A4988 motor driver is used instead of TMC2209**

- ***Test Case 2: A4988 Auger Control***

- **Objective:** Use A4988 microstepping to improve auger precision.
- **Steps:**
  - Connect motor and driver to power and Arduino. Check pins, connection types, and secure exposed wires.
  - Check power supply to verify input voltage and current draw.
  - Write a function in Arduino IDE for precise auger control.
  - Conduct 5+ trials to drive with various step sizes.
- **Expected Result: Precision of dispensing and extraction angle are improved due to microstepping.**
- **Result: PASSED**

- ***Test Case 3: A4988 Carriage Control***

- **Objective:** Use A4988 microstepping to improve carriage rotation precision.
- **Steps:**
  - Connect motor and driver to power and Arduino. Check pins, connection types, and secure exposed wires.
  - Check power supply to verify input voltage and current draw.
  - Write a function in Arduino IDE for precise carriage control.
  - Conduct 5+ trials to rotate with various step sizes.
- **Expected Result: Precision of carriage rotation and linear rail alignment is improved due to microstepping.**
- **Result: PASSED**

- **Display Subsystem Testing**

- The display subsystem consists of an LCD 128x64 screen with a stop button and potentiometer scroll wheel. These parts will allow users to view data from the machine and input job requests.

LCD Screen Testing

- *Test Case 1: Spice Selection Screen*

- **Objective:** Display container numbers and spice names, dispense amounts, and units.
- **Steps:**
  - Connect LCD to power and Arduino. Check pins, connection types, and secure exposed wires.
  - Check power supply to verify input voltage and current draw.
  - Write a function in Arduino IDE for selection screen display.
  - Conduct 5+ trials to display various combinations of spice names.
- **Expected Result: All container names, amounts, and units are represented without symbol errors. Scrolling allows the user to view the entire list.**
- **Result: PASSED**

- *Test Case 2: Dispensing In-Progress Screen*

- **Objective:** Alert user that dispensing is in progress. Stretch goal: display completion status.
- **Steps:**
  - Connect LCD to power and Arduino. Check pins, connection types, and secure exposed wires.
  - Check power supply to verify input voltage and current draw.
  - Write a function in Arduino IDE for dispensing screen display.
  - Conduct 5+ trials to display for various amounts of time, under various job requests.
- **Expected Result: The dispensing screen lasts for the duration of job completion and informs users with any necessary information about job status.**
- **Result: PASSED**
-
-
- *Test Case 3: Completed Job Screen*

- **Objective:** Inform users when a job has been completed.
- **Steps:**
  - Connect LCD to power and Arduino. Check pins, connection types, and secure exposed wires.
  - Check power supply to verify input voltage and current draw.
  - Write a function in Arduino IDE for job completion screen.
  - Conduct 5+ trials to display various job output messages.
- **Expected Result: Display a message that dispensing is complete, with an (optional) spicy saying for fun. Stretch goal: display the blend name dispensed.**
- **Result: PASSED**

Potentiometer Testing

Test Case 1: Scrolling Function

- **Objective:** Users can scroll through list-style screens using the potentiometer wheel.
- **Steps:**
    1. Connect LCD to power and Arduino. Check pins, connection types, and secure exposed wires.
    2. Check power supply to verify input voltage and current draw.
    3. Write a function in Arduino IDE for scrolling.
    4. Conduct 5+ trials to scroll various distances on various lists.
- **Expected Result: The scroll wheel outputs a consistent, user-friendly scroll speed. Stretch goal: add clicks or tick marks that correspond to scroll ticks.**
- **Result: PASSED**

Test Case 2: Selection Function

- **Objective:** Users can press the potentiometer to select options.
- **Steps:**
    1. Connect LCD to power and Arduino. Check pins, connection types, and secure exposed wires.
    2. Check power supply to verify input voltage and current draw.
    3. Write a function in Arduino IDE for selection.
    4. Conduct 5+ trials to select various spices and amounts.
- **Expected Result: The potentiometer is easy to press and selects an option when used, advancing to the next selection screen.**
- **Result: PASSED**

Sensor Subsystem Testing

The sensor subsystem consists of limit switches for cup detection and carriage calibration, a temperature and humidity sensor to monitor thermal conditions, and a load cell with an HX711 amplifier as a stretch goal to detect mass of spice dispensed.

Limit Switch Testing

Test Case 1: Cup Detection

- **Objective:** Limit switch detects the presence of the drop zone cup.
- **Steps:**

1. Connect limit switch to power and Arduino. Check pins, connection types, and secure exposed wires.
2. Check power supply to verify input voltage and current draw.
3. Write a function in Arduino IDE for limit switch detection.
4. Conduct 5+ trials to detect the cup for various amounts of time and loading conditions (inserted quickly, forcefully, gently, etc.).

- **Expected Result: The limit switch consistently signals the Arduino for the duration of the cup's insertion.**
- **Result: PASSED**

Test Case 2: Carriage Calibration

- **Objective:** Limit switch is triggered to reset carriage positioning after a job.
- **Steps:**
    1. Connect limit switch to power and Arduino. Check pins, connection types, and secure exposed wires.
    2. Check power supply to verify input voltage and current draw.
    3. Write a function in Arduino IDE for carriage positioning calibration.
    4. Conduct 5+ trials to calibrate the carriage from various arbitrary positions.
- **Expected Result: The limit switch is consistently triggered during calibration, consistently signals activation to the Arduino, and is activated with sufficient precision and timing for linear rail to be aligned.**
- **Result: PASSED**

Temperature and Humidity Sensor Testing

Test Case 1: Low Temperature

- **Objective:** Temperature sensor is accurate at low temperatures (approx. 32ºF).
- **Steps:**
    1. Connect sensor to power and Arduino. Check pins, connection types, and secure exposed wires.
    2. Check power supply to verify input voltage and current draw.
    3. Write a function in Arduino IDE for temperature sensing.
    4. Conduct 5+ trials to sense cold temperatures various times. This can be accomplished with bags of ice water and a digital thermometer.
- **Expected Result: The sensor detects temperatures within its rated margin of error and environmental uncertainty.**
- **Result: PASSED**

Test Case 2: High Temperature

- **Objective:** Temperature sensor is accurate at high temperatures (above 100ºF).

- **Steps:**
    1. Connect sensor to power and Arduino. Check pins, connection types, and secure exposed wires.
    2. Check power supply to verify input voltage and current draw.
    3. Write a function in Arduino IDE for temperature sensing and emergency stop for dangerous temperatures.
    4. Conduct 5+ trials to sense hot temperatures various times. This can be accomplished with bags of hot water and a digital thermometer.
- **Expected Result: The sensor detects temperatures within its rated margin of error and environmental uncertainty. Dispensing is stopped when temperatures exceed rated safe conditions for materials and components. Stretch goal: fan speed responds to sensed temperature to save power and noise.**
- **Result: PASSED**

Test Case 3: Prolonged Temperature Readings

- **Objective:** Temperature sensor can accurately measure temperatures over long durations.
- **Steps:**
    1. Connect sensor to power and Arduino. Check pins, connection types, and secure exposed wires.
    2. Check power supply to verify input voltage and current draw.
    3. Write a function in Arduino IDE for temperature sensing.
    4. Conduct 5+ trials to test prolonged readings at various temperatures.
- **Expected Result: Temperature sensor does not display drift error over time outside its rated conditions and environmental uncertainty.**
- **Result: PASSED**

Load Cell and HX711 Amplifier Testing

Test Case 1: Static Loading for Various Weights

- **Objective:** Measure set weights using a load cell and signal amplifier.
- **Steps:**
    1. Connect sensor to power and Arduino. Check pins, connection types, and secure exposed wires.
    2. Check power supply to verify input voltage and current draw.
    3. Write a function in Arduino IDE for weight sensing.
    4. Conduct 5+ trials to sense various weights.
- **Expected Result: The sensor can detect weights accurately within its rated margin of error.**
- **Result: FAIL – Not installed**

Test Case 2: Gradual Loading

- **Objective:** Measure spice by weight, during dispensing, using a load cell and signal amplifier.
- **Steps:**
    1. Connect sensor to power and Arduino. Check pins, connection types, and secure exposed wires.
    2. Check power supply to verify input voltage and current draw.
    3. Write a function in Arduino IDE for weight sensing and PID dispensing control.
    4. Conduct 5+ trials to sense and dispense various amounts of spice by weight.
- **Expected Result: The sensor resolution allows detection of spice in grams and PID controller can output spice levels within a gram of the target amount.**
- **Result: FAIL – No load cell installed**

Test Case 3: Recalibration

- **Objective:** User can select a calibration option before dispensing spice by weight.
- **Steps:**
    1. Connect sensor to power and Arduino. Check pins, connection types, and secure exposed wires.
    2. Check power supply to verify input voltage and current draw.
    3. Write a function in Arduino IDE for calibration.
    4. Conduct 5+ trials to calibrate after various conditions.
- **Expected Result: Load cell signal will be zeroed after calibration so that the weight of the drop zone cup is not considered.**
- **Result: FAIL - No Load Cell installed**

Power Subsystem Testing

The power subsystem is designed to distribute power to all other electrical components while regulating voltage, current, and heat. It consists of a power supply unit (PSU), 3-prong power outlet cord, rocker switch, and CPU fan.

PSU Testing

Test Case 1: Voltage Regulation

- **Objective:** The PSU consistently outputs 12VDC.
- **Steps:**
    1. Connect PSU to power and multimeter. Check pins, connection types, and secure exposed wires.
    2. Conduct 5+ trials to measure voltage various times.
- **Expected Result: The PSU outputs power with negligible voltage fluctuations, if any.**
- **Result: PASSED**

Test Case 2: Heat Regulation

- **Objective:** The PSU maintains safe temperatures for materials and components.
- **Steps:**
    1. Connect PSU to power, electronics to draw power, and thermometer. Check pins, connection types, and secure exposed wires.
    2. Conduct 5+ trials to measure temperature under various amounts of current draw.
- **Expected Result: Temperatures are safe for operation (a fan can be added if necessary).**
- **Result: PASSED**

Power Cord Testing

Test Case 1: Secureness Testing

- **Objective:** The power cord is fastened securely and cannot be tugged out of place.
- **Steps:**
    1. Insert cord within device housing. Check connection types and secure exposed wires.
    2. Fasten the power cord to the device housing.
    3. Conduct 5+ trials to ensure the cord withstands various load cases (user or child tugging on it, device being dropped, etc.).
- **Expected Result: The cord stays fastened and live wires are never exposed to the user.**
- **Result: PASSED**

Rocker Switch Testing

Test Case 1: Emergency Power Cutoff

- **Objective:** The rocker switch kills all device power, for emergencies.
- **Steps:**
    1. Connect rocker switch to cords. Check connection types and secure exposed wires.
    2. Build a circuit to indicate open or closed connection, such as an LED indicator of power or a multimeter. Connect electronics to draw a realistic amount of power.
    3. Conduct 5+ trials to test various component configurations.
- **Expected Result: The rocker switch removes power from all components.**
- **Result: PASSED**

Fan Testing

Test Case 1: Constant (RAMPS Aux) Fan

- **Objective:** The fan can be run at a constant rate for cooling.
- **Steps:**
    1. Connect fan to power and Arduino. Check pins, connection types, and secure exposed wires.
    2. Check power supply to verify input voltage and current draw.
    3. Conduct 5+ trials to ensure performance under various stages of the dispensing process.
- **Expected Result: The fan consistently outputs its maximum speed for a constant 12V supply of power. This coincides with the auxiliary fan in the Ramps board configuration.**
- **Result: PASSED**

Test Case 2: Controllable (RAMPS Extruder) Fan

- **Objective:** The fan speed can be controlled.
- **Steps:**
    1. Connect fan to power and Arduino. Check pins, connection types, and secure exposed wires.
    2. Check power supply to verify input voltage and current draw.
    3. Write a function in Arduino IDE for controlling fan speed according to temperature.
    4. Conduct 5+ trials to test speeds at various temperatures.
- **Expected Result: The fan speed can be controlled by altering its input voltage in response to sensed temperatures, allowing for energy efficiency.**
- **Result: Failure - Unnecessary feature**

Microcontroller Subsystem Testing

The microcontroller subsystem refers to the microcontroller unit (MCU) itself, in addition to peripherals that will assist in the communication and processing of data, such as a Bluetooth module.

Bluetooth Testing

Test Case 1: Receive and Display App Order

- **Objective:** Data from the Flavor Fusion app is received by the MCU and displayed for verification.
- **Steps:**

    1. Connect Bluetooth, power, and Arduino. Check pins, connection types, and secure exposed wires.
    2. Check power supply to verify input voltage and current draw.
    3. Write a function in Arduino IDE for Bluetooth communication.
    4. Conduct 5+ trials to receive and display various spice orders.
- **Expected Result:** All spice data is successfully received within seconds, then displayed to prove that data can be interpreted.
- **Result: PASSED**

Test Case 2: Update App Data After App Order

- **Objective:** Spice level data is updated in the app after an order has been placed and dispensed from the app.
- **Steps:**
    1. Connect Bluetooth, power, and Arduino. Check pins, connection types, and secure exposed wires.
    2. Check power supply to verify input voltage and current draw.
    3. Write a function in Arduino IDE for Bluetooth communication.
    4. Conduct 5+ trials to update various amounts and blends of spice.
- **Expected Result:** All spice data is received by the app within seconds, then updated within the app to reflect the dispensed amount.
- **Result: PASSED**

Test Case 3: Update App Data After Local Order

- **Objective:** Spice level data is updated in the app after an order has been placed and dispensed from the local device.
- **Steps:**
    1. Connect Bluetooth, power, and Arduino. Check pins, connection types, and secure exposed wires.
    2. Check power supply to verify input voltage and current draw.
    3. Write a function in Arduino IDE for Bluetooth communication.
    4. Conduct 5+ trials to update various amounts and blends of spice.
- **Expected Result:** All spice data is received by the app within seconds, then updated within the app to reflect the dispensed amount.
- **Result: FAIL**

Aesthetic Subsystem Testing

The aesthetic subsystem relates to all electronics used for decoration or enhancement of the Flavor Fusion device without being critical to performance and safety. While parts like LEDs may play an important role in user interactions and ease of use, they are ultimately nonvital.

LED Testing

Test Case 1: Individual Colored LEDs

- **Objective:** LEDs can be illuminated individually or in patterns to indicate device status or for aesthetic appeal.
- **Steps:**
    1. Connect LEDs to power and Arduino. Check pins, connection types, and secure exposed wires.
    2. Check power supply to verify input voltage and current draw.
    3. Write a function in Arduino IDE for LED control.
    4. Conduct 5+ trials to display various configurations.
- **Expected Result:** LEDs respond in the desired illumination pattern and are bright enough to be clearly visible on the device.
- **Result: FAIL – Not installed**

Test Case 2: RGB LEDs

- **Objective:** RGB LEDs can be illuminated individually or in patterns across a range of colors.
- **Steps:**
    1. Connect LEDs to power and Arduino. Check pins, connection types, and secure exposed wires.
    2. Check power supply to verify input voltage and current draw.
    3. Write a function in Arduino IDE for LED color control.
    4. Conduct 5+ trials to display various colors.
- **Expected Result:** RGB LEDs can display a range of distinct colors to signify various purposes or fulfill aesthetic appeal.
- **Result: FAIL – Not installed**

11. Significant Accomplishments

11.1 Allison's Accomplishments

I created a fully functional iPhone app and deployed it to test flight for beta testing. I successfully implemented a spice inventory system, blending functionalities, a recipe book, settings page, and informational about page. I also successfully implemented accessibility features including adjustable text size, dark/light mode, and voice-to-text input. I created and executed extensive UI tests to ensure that each feature met the requirements initially set by the team. The only mobile app feature that was not completed was queue to allow the user to send multiple blends at once. Adding the queue created performance issues that were unable to be resolved. This feature was also not necessary and would does not affect the usability of the device. I implemented Bluetooth Low Energy to allow the app to send a blend to the Flavor Fusion Device. In the future I would create either a cross-platform application using Flutter, or I would create a separate Android app using Kotlin.

11.2 Thomas' Accomplishments

I have created a dispensing mechanism that allows for fully operational dispensing across multiple grain sizes and shapes. I have also met a majority of my requirements set by the Flavor Fusion Team. Such as dispensing a variety of grain sizes, ease of disassembly and reassembly and ease of cleaning. The only ones I did not make are dishwasher safe, and spice leeching. But these weren't because of design flaws but due to machines that were easy to access. FDM printing augers for the dispenser created a lot of cavities in which spices can leech into and make it hard to clean I would have liked to use SLM style printing which would allow for almost zero-layer lines ensuring no leeching inside my parts, but I did not have this machine available. Also repeated dishwashing hasn't been tested yet, but I do believe that constant dishwashing will drastically fatigue parts over multiple cycles due to the filaments glass transition temperature being very close to where the materials we used are.

11.3 Ethan's Accomplishments

My accomplishments on this project mostly revolve around being the housing subsystem lead. Before this project I had hardly ever used SolidWorks, but throughout this project I was able to use it to design many parts for the housing and carriage including motor housings, magnetic connector housings, spice bottle slots, parts to allow the carriage to spin with the motor and more. Aside from CAD, I was able to help machine and assemble the device, which I had very little experience doing. We created and executed many features that allowed us to satisfy our requirements initially set by the team. On top of the physical accomplishments, I feel accomplished knowing that I was able to step outside of my subsystem in writing for the group, even as recently as writing the script for our eleven-minute-long video submission to showcase.

11.4 Samuel's Accomplishments

Given the limited electronics experience that I possessed at the beginning of Senior Design 2, I am proud not only of my optimism and willingness to increase my skillset, but the success with which I executed Flavor Fusion's electromechanical systems. My prior experience was a single mechatronics class where every circuit was designed and provided, and all components were supplied from a standard Arduino kit. Transitioning to a project where I was in charge of circuit design- both for device performance and the team's safety- and had no preset kit to choose from, I was initially very intimidated by my responsibilities. To make up for my lack of experience, I constantly researched electronics and strove to learn from tutorials, blogs, forums, datasheets, textbooks, and other online resources until I felt confident in my knowledge about each part in the Flavor Fusion device.

Three months later, I can claim with confidence that I have experience in wiring, crimping, and soldering of complex electromechanical systems; familiarity with electronic components such as stepper drivers, sensors, and stepper motors themselves; UI design and implementation; problem-solving and debugging code for physical systems; managing numerous and complex functions while optimizing runtime; and even applying CAD skills for

measurement and 3D printing. As the team's electromechanical engineer, I have touched many components throughout the system.

Ironically, I believe the electromechanical aspect of my responsibilities also serves as a point of improvement in my performance. In the future, I would like to improve my ability to develop electromechanical systems in parallel- considering not only the electronic requirements of each project, but the physical constraints and implications of the design. Sizing motors adaptively based on geometry or placing limit switches *as* CAD models are being finalized are crucial steps that can save money, time, and space by keeping the design simple. For example, if I had considered the locations and lengths of wires in the system much sooner in the design process, cable management would have been a significantly easier task.

Regardless, I can look back on my team's Senior Design project with pride and accomplishment, knowing that I have grown as an engineer while identifying active areas of improvement that I look forward to addressing in the future.

11.5 Alexandra's Accomplishments

Reflecting over the past year of planning, development, and modeling, I would say the biggest accomplishment was being able to take a concept and idea to design a full prototype machine from the ground up. Since having a basic background in CAD when starting out, I would say my skill has improved immensely from figuring out new techniques implementing faster ways to design new parts in less time. Especially when it came to checking the CAD models for fitment test of certain stepper motors, or other components, having the skill to help integrate those parts helped me and the team out. In addition to the CAD design of Flavor Fusion, I researched different materials. Since it had to be dishwasher safe, food safe, and cost effective I took the initiative to ensure that I would reach those goals by viewing many different types of 3D print materials. By reading into the temperature deformations, food safety and toxicity of certain 3D print materials I was able to find a Food Safe PET-G to satisfy the team's safety goals. Not only did I assist the team in CAD and materials, but I also helped in assembly using a heat welding tool to ensure that the machine would be held together correctly.

11.6 Ryan's Accomplishments

Like most of my teammates, I came into this semester without a lot of knowledge of what was to come. However, I feel confident that I have come out of this semester with stronger skills in part integration and electronics testing, as well as mobile app development. With all of these combined, my project management skills have improved, as well as figuring out how to play to each individual's strengths and weaknesses and persevering through challenging work dynamics and different efforts from different team members.

I think our biggest success this semester was overcoming senioritis and creating and showcasing our device in a way that most teams wouldn't. We played to our strengths of being an interdisciplinary team and produced a final report and video that leaned into its commercial roots, quite literally. Recording and editing this video allowed me to embrace my digital media background and help the team stand out against a sea of other engineering projects. I was able to flex my metaphorical Adobe Creative Cloud muscles and get back into my groove.

Despite the many setbacks the team faced, from hurricane delays to career fairs, I feel as though my leadership skills vastly improved and kept the team moving forward. I was able to deliver a role that essentially allowed me to fill in the gaps of wherever I was needed, so I was able to get my hands dirty in CAD, electronics, app, development, and testing. I was able to push the team to deliver a different kind of showcase video, facilitated communication, and connected parts and pieces that fell through the cracks. When our regular Sunday meetings were plagued with issues from not connecting during the week, I introduced midweek standup meetings that started on time and were capped at 30 minutes, making sure everyone was present to hear about the progress of their peers, and offer assistance on any blockers. This cut down our time spent waiting around on Sundays, and kept the team working towards a successful project.

I was also able to come to each subsystem lead with the perspective of multiple subsystems, attacking problems with a multifaceted approach to help eliminate errors and guide the team to a solution, rather than saying what the answer should be or dictating next steps. This collaborative leadership style allowed me to connect with my team and the project at hand.

If given the opportunity to do this again, I would like to further improve my leadership skills and maybe take on a subsystem lead role. This would allow me to further specialize

technically, and understand what it is like to be led as well. I would also reprioritize how we completed our testing and guidelines. I'd refocus efforts on developing an effective digital twin early on so we could use that to address issues with fitment.

12. References

1] "TasteTro Spice System," uncrate, [Online]. Available: https://uncrate.com/tastetro-spice-system/. [Accessed 4 February 2024].

2] "Tastetro Spice System Iot Spice Mixer Forces Users to Buy Proprietary Spice, Identified by RFID, for $7 to $12 Per Pod," Reddit, 2018. [Online]. Available: https://www.reddit.com/r/shittykickstarters/comments/8jlmca/tastetro_spice_system_iot_spice_mixer_forces/. [Accessed 4 February 2024].

3] TasteTro, "About TasteTro," TasteTro, [Online]. Available: https://www.tastetro.com/about/. [Accessed 4 February 2024].

4] TasteTro, "Tastetro Spice System - Indigogo," Indigogo, 2018. [Online]. Available: https://web.archive.org/web/20180515133646/https://www.indiegogo.com/projects/tastetro-spice-system-vegan#/. [Accessed 4 February 2024].

5] D. McGinn, "The Buzz Machine," Boston.com, 7 August 2011. [Online]. Available: https://web.archive.org/web/20210119120311/http://archive.boston.com/business/articles/2011/08/07/the_inside_story_of_keurigs_rise_to_a_billion_dollar_coffee_empire/?page=full. [Accessed 4 February 2024].

6] Keurig Dr Pepper, "Family of Brands," Keurig Dr Pepper, 2024. [Online]. Available: https://www.keurigdrpepper.com/en/our-brands/product-facts-brands. [Accessed 4 February 2024].

7] J. Raub, "How Do Keurig Coffee Makers Work, Anyways?," Serious Eats, 16 March 2023. [Online]. Available: https://www.seriouseats.com/how-do-keurig-coffee-makers-work-

6889331#:~:text=While%20a%20drip%20coffee%20maker,four%20minutes%20to%20mere%20seconds.. [Accessed 2024 February 2024].

8] "Keurig Sensor Blocks Your Brew Unless It's "k-cup Compatible", Aka Has Scannable Foil. Slap on an Old Foil to a 3rd Party Cup and Suddenly No Issue.," Reddit, 2022. [Online]. Available: https://www.reddit.com/r/assholedesign/comments/rvgci4/keurig_sensor_blocks_your_brew_unless_its_kcup/. [Accessed 4 February 2024].

9] Keurig, "How BrewID Powers Our SMART Family of Coffee Makers," Keurig Dr Pepper, 2022. [Online]. Available: https://www.keurig.com/hub/lifestyle/how-brewid-powers-our-smart-family-of-coffee-makers. [Accessed 4 February 2024].

10] Keurig, "Descale & Cleanse Starter Kit," Keurig Dr Pepper, 2024. [Online]. Available: https://www.keurig.com/Descale-%26-Cleanse-Starter-Kit/p/Descale-Cleanse-Starter-Kit. [Accessed 4 February 2024].

11] O. Jones, "How Does a Keurig Coffee Maker Work? Detailed Guide," Coffee Affection, 20 January 2024. [Online]. Available: https://coffeeaffection.com/how-does-a-keurig-work/. [Accessed 4 February 2024].

12] The Coca-Cola Company, "How the Coca-Cola Freestyle Dispenser Came to Be," The Coca-Cola Company, 2017. [Online]. Available: https://www.coca-colacompany.com/media-center/how-the-coca-cola-freestyle-dispenser-came-to-be. [Accessed 4 February 2024].

13] The Coca-Cola Company, "Innovating Beyond the Traditional Package," The Coca-Cola Company, 2018. [Online]. Available: https://www.coca-colacompany.com/media-center/innovating-beyond-the-traditional-package. [Accessed 4 February 2024].

14] The Coca-Cola Company, "Coca-Cola Freestyle 9100 User Guide," The Coca-Cola Company, Atlanta, 2020.

B. Horovitz, "Coke aims for cool with new 146-flavor dispenser," USA Today, 16
15] April 2014. [Online]. Available:
https://www.usatoday.com/story/money/business/2014/04/14/coca-cola-coke-freestyle-soft-drinks-beverages/7478341/. [Accessed 4 February 2024].

C. Doering, "How Coca-Cola turns to its Freestyle machine to create shelf-ready
16] flavors," FoodDive, 15 December 2022. [Online]. Available:
https://www.fooddive.com/news/how-coca-cola-turns-to-its-freestyle-machine-to-create-shelf-ready-flavors/636660/. [Accessed 2024 February 2024].

JES Restaurant Equipment, "How Ice Makers Maintain The Coke Freestyle
17] Illusion," JES Restaurant Equipment, 9 July 2015. [Online]. Available:
https://www.jesrestaurantequipment.com/how-ice-makers-keep-coke-freestyle-illusion.html. [Accessed 4 February 2024].

CostAide, "How Much Does Coca-Cola Freestyle Cost," CostAide, [Online].
18] Available: https://costaide.com/coca-cola-freestyle-cost/. [Accessed 4 February 2024].

"What's Your Thoughts on the Coke Freestyle Machine? What's Your Favorite
19] Flavors/mix?," Reddit, 2023. [Online]. Available:
https://www.reddit.com/r/Soda/comments/16q953i/whats_your_thoughts_on_the_coke_freestyle_machine/. [Accessed 4 February 2024].

Ford Motor Company, "FordPass Support," Ford Motor Company, [Online].
20] Available: https://www.ford.com/support/category/fordpass/. [Accessed 4 February 2024].

Ford Motor Company, "FordPass App (version 4.32.1)," Apple, Inc., 2024.
21] [Online]. Available: https://apps.apple.com/us/app/fordpass/id1095418609. [Accessed 4 February 2024].

"Fuel Cells Vs. Batteries: What's the Difference?," [Online]. Available:
22] https://www.powermag.com/fuel-cells-vs-batteries-whats-the-difference/. . [Accessed 04 February 2024].

"How Many AA Batteries Would It Take to Power a Mercedes?," [Online]. Available: https://sustainable-nano.com/2016/04/29/aa-batteries-mercedes/.. [Accessed 04 February 2024].
23]

"Electricity Cost in Florida: 2024 Electric Rates," [Online]. Available: https://www.energysage.com/local-data/electricity-cost/fl/.. [Accessed 04 February 2024].
24]

"How Blenders Work," [Online]. Available: https://home.howstuffworks.com/blender.htm.. [Accessed 04 February 2024].
25]

"How Toasters Work," [Online]. Available: https://home.howstuffworks.com/toaster.htm.. [Accessed 04 February 2024].
26]

"Why Spices Harden and What You Should Do to Prevent It," [Online]. Available: https://theepicentre.com/why-spices-harden-and-what-you-should-do-to-prevent-it/.. [Accessed 04 February 2024].
27]

"Film Can," [Online]. Available: https://en.wikipedia.org/w/index.php?title=Film_can&oldid=895421023. . [Accessed 04 February 2024].
28]

"Limit Switch Explained | Working Principles - Realpars," [Online]. Available: https://www.realpars.com/blog/limit-switch.. [Accessed 04 February 2024].
29]

"Piston - Energy Education," [Online]. Available: https://energyeducation.ca/encyclopedia/Piston#:~:text=A%20piston%20is%20a%20moving,mechanical%20work%20and%20vice%20versa.. [Accessed 04 February 2024].
30]

"Herb Grinder," [Online]. Available: https://en.wikipedia.org/w/index.php?title=Herb_grinder&oldid=1169227243.. [Accessed 04 February 2024].
31]

"The Best Air-fryer Liners of 2023 | America's Test Kitchen," [Online]. Available:
32] https://www.americastestkitchen.com/equipment_reviews/2498-the-best-air-fryer-liners.. [Accessed 04 February 2024].

"The Easiest Way to Run Wires Through a Rotating Shaft?," [Online]. Available:
33] https://www.physicsforums.com/threads/the-easiest-way-to-run-wires-through-a-rotating-shaft.874264/.. [Accessed 04 February 2024].

J. Owen, "How does a Gumball Machine work?," YouTube, 30 June 2018.
34] [Online]. Available: https://www.youtube.com/watch?v=Q3ZeUNDg4fQ.. [Accessed 6 February 2024].

XQ Instruments, "SDB-1 Benchtop Dosing Machine," [Online]. Available:
35] https://xqinstruments.com/dispensing-solutions/sdb1-dosing-system/.. [Accessed 6 February 2024].

"Screw conveyer," Wikipedia, 7 November 2023. [Online]. Available:
36] https://en.wikipedia.org/w/index.php?title=Screw_conveyor&oldid=1183920977.. [Accessed 6 February 2024].

Spee-Dee Packaging Machinery Inc., "Spice Packing Machine," 2022. [Online].
37] Available: https://www.spee-dee.com/automatic-spice-packing-filling-machine. [Accessed 6 February 2024].

R. Gurberg, "Automated Spice Dispenser DEMO," YouTube, 8 August 2017.
38] [Online]. Available: https://www.youtube.com/watch?v=i3xB1WOZR2Y. [Accessed 6 February 2024].

IDM Ltd., "HPD3-BL Spice Dispenser," [Online]. Available: https://www.idm-
39] dispenser.com/products/hpd3-bl-spice-dispenser. [Accessed 6 February 2024].

"Powder Measure," Wikipedia, 5 December 2023. [Online]. Available:
40] https://en.wikipedia.org/w/index.php?title=Powder_measure&oldid=1188509414. [Accessed 6 February 2024].

Ø. Flatnes, "Jeff Tanner's Powder Dispenser," Svart Krutt, 10 June 2011. [Online]. Available: https://svartkrutt.net/articles/vis.php?id=43. [Accessed 6 February 2024].

41]

JunkfoodZombie, "Powder Measure: How It Works and How to Use It," YouTube, 4 August 2015. [Online]. Available: https://www.youtube.com/watch?v=ekjBPm9zuKg. [Accessed 6 February 2024].

42]

S. Yang and J. Evans, "Metering and dispensing of powder; the quest for new solid freeforming techniques," *Powder Technology,* vol. 178, no. 1, pp. 56-72, 2007.

43]

The Coca-Cola Company, "How the Coca-Cola Freestyle Dispenser Came to Be," 6 December 2017. [Online]. Available: https://www.coca-colacompany.com/media-center/how-the-coca-cola-freestyle-dispenser-came-to-be. [Accessed 6 February 2024].

44]

"Cooking weights and measure," Wikipedia, 19 January 2024. [Online]. Available: https://en.wikipedia.org/w/index.php?title=Cooking_weights_and_measures&oldid=1197047188. [Accessed 6 February 2024].

45]

A. Papapetros, "Measuring Cups and Spoons Worldwide," Salads with Anastasia, [Online]. Available: https://saladswithanastasia.com/measuring-cups-and-spoons-worldwide/. [Accessed 6 February 2024].

46]

TasteTro, "The TasteTro Spice System," [Online]. Available: https://www.tastetro.com/. [Accessed 6 February 2024].

47]

Bolt Action Reloading, "Accurate Powder Charges - Fast and Affordable - FA Intellidropper," YouTube, 7 April 2023. [Online]. Available: https://www.youtube.com/watch?v=N9EwQVD3gKI. [Accessed 6 February 2024].

48]

N. Chandler, "How does your computer know how much ink is left in the cartridge?," HowStuffWorks, 5 June 2014. [Online]. Available:

49]

https://computer.howstuffworks.com/computer-ink-cartridge.htm. [Accessed 6 February 2024].

"What is the Difference Betwen Microcontroller and Microprocessor?," Rayming
50] PCB & Assembly, [Online]. Available: https://www.raypcb.com/microcontroller-vs-microprocessor/. [Accessed 4 February 2024].

"Arduino Hardware," Arduino CC, 11 April 2022. [Online]. Available:
51] https://www.arduino.cc/en/hardware. [Accessed 18 February 2024].

"Arduino MKR WiFi 1010," Arduino CC, [Online]. Available: https://store-
52] usa.arduino.cc/products/arduino-mkr-wifi-1010. [Accessed 4 February 2024].

"What Is Raspberry Pi? Models, Features, and Uses," Spiceworks, 6 August
53] 2022. [Online]. Available: https://www.spiceworks.com/tech/networking/articles/what-is-raspberry-pi/. [Accessed 3 March 2024].

"Raspberry Pi," Raspberry Pi, [Online]. Available:
54] https://www.raspberrypi.com/products/. [Accessed 3 March 2024].

"Raspberry Pi Zero 2 W," Raspberry Pi, [Online]. Available:
55] https://www.raspberrypi.com/products/raspberry-pi-zero-2-w/. [Accessed 4 February 2024].

C. Hu, "What is a Motor Driver and How Do You Choose the Right One?,"
56] WellPCB, [Online]. Available: https://www.wellpcb.com/what-is-motor-driver.html#:~:text=A%20motor%20driver%20is%20an,larger%20motors%20in%20industrial%20applications. [Accessed 4 February 2024].

"Arduino and Stepper Motor Configurations," Arduino CC, [Online]. Available:
57] https://docs.arduino.cc/learn/electronics/stepper-motors/. [Accessed 4 February 2024].

"Kotlin Multiplatform," Kotlin, 2023. [Online]. Available:
58] https://kotlinlang.org/docs/multiplatform.html#get-started. [Accessed 4 February 2024].

"Build For Any Screen," Flutter, [Online]. Available: https://flutter.dev/.
59] [Accessed 3 March 2024].

"Introduction to Swift," Exyte, 20 November 2020. [Online]. Available:
60] https://exyte.com/blog/introduciton-to-swift. [Accessed 3 March 2024].

"Arduino BLE," Arduino CC, [Online]. Available:
61] https://www.arduino.cc/reference/en/libraries/arduinoble/. [Accessed 4 February 2024].

"Universal Asynchronous Receiver-Transmitter (UART)," Docs, [Online].
62] Available: https://docs.arduino.cc/learn/communication/uart/. [Accessed 17 February 2024].

"Bluetooth Technology Overview," Bluetooth, [Online]. Available:
63] https://www.bluetooth.com/learn-about-bluetooth/tech-overview/. [Accessed 1 March 2024].

Tessie, "HC-06 vs. HC-05 Bluetooth Module: What is the difference between
64] HC-06 and HC-05?," Utmel, 24 November 2021. [Online]. Available: https://www.utmel.com/components/hc-06-vs-hc-05-bluetooth-module-what-is-the-difference-between-hc-06-and-hc-05?id=889. [Accessed 1 March 2024].

"Instant Pot® Pro™ Plus 6-quart Smart Multi-Cooker," Instant, [Online].
65] Available: https://www.instanthome.com/product/instant-pot/pro-plus/6-quart-smart-pressure-cooker. [Accessed 4 February 2024].

K. Vaughan, "Instant Pot Pro Plus Review: The Best Model On The Market,"
66] Forbes, 18 November 2022. [Online]. Available: https://www.forbes.com/sites/forbes-personal-shopper/article/instant-pot-pro-plus-review/?sh=784724bc230b. [Accessed 3 March 2024].

Keurig Consumer Care Team, "How to Use the Keurig® App," Keurig, 2022.
67] [Online]. Available: https://www.keurig.com/hub/support/how-to-use-the-keurig-app. [Accessed 4 February 2024].

"Keurig Commercial Unveils Touchless Brewing as Americans Return to New
68] Coffee Routines in the Workplace," Keurig Dr Pepper, 3 February 2021. [Online]. Available: https://news.keurigdrpepper.com/2021-02-03-Keurig-Commercial-Unveils-Touchless-Brewing-as-Americans-Return-to-New-Coffee-Routines-in-the-Workplace. [Accessed 1 March 2024].

R. contributers, "RepRap.org," 11 December 2022. [Online]. Available:
69] https://reprap.org/mediawiki/index.php?title=RAMPS_1.4&oldid=189851. [Accessed 16 April 2024].

D. (. Caditz, "Autodesk Instructables," Autodesk, January 2021. [Online].
70] Available: https://www.instructables.com/Hacking-Arduino-MegaRAMPS-3D-Printer-Shield-to-Con/. [Accessed 16 April 2024].

Heason Technology, "Heason.com," 27 October 2015. [Online]. Available:
71] https://www.heason.com/news-media/technical-blog-archive/how-does-a-stepper-motor-work-#:~:text=A%20stepper%20motor%20works%20when,are%20also%20known%20as%20poles.. [Accessed 16 April 2024].

R. Solecki and R. J. Conant, in *Advanced Mechanics of Materials*, Oxford
72] University Press, 2003, p. 620.

Core3D, "Wiring 3D Printer RAMPS 1.4," AutoDesk Instructables, [Online].
73] Available: https://www.instructables.com/Wiring-3D-Printer-RAMPS-14/. [Accessed 4 October 2024].

"RAMPS 1.4," RepRap, [Online]. Available: https://reprap.org/wiki/RAMPS_1.4.
74] [Accessed 4 October 2024].

"How Keurig made its coffee machines smarter," 14 November 2022. [Online].
75] Available: https://www.bostonlobe.com/2022/11/14/business/how-keurig-made-its-coffee-machines-smarter. [Accessed 5 February 2024].

"Types of Manufacturing You Should Know About," amper, [Online]. Available: https://www.amper.xyz/post/5-types-of-manufacturing-processes-you-should-know-about. [Accessed 5 February 2024].

[76]

"Unit 5.2: Production Methods," SlidePlayer, [Online]. Available: https://slideplayer.com/slide/7862211/. [Accessed 6 February 2024].

[77]

S. Afzal, "Different Types of Manufacturing Processes," LinkedIn, [Online]. Available: https://www.linkedin.com/pulse/different-types-manufacturing-processes-all-one-guide-sayeed-afzal/. [Accessed 6 February 2024].

[78]

"Blow Molding vs. Injection Molding," Team Xometry, 22 March 2022. [Online]. Available: https://www.xometry.com/resources/injection-molding/blow-molding-vs-injection-molding/#:~:text=Injection%20molding%20is%20used%20for,a%20mold%20using%20compressed%20air. [Accessed 6 February 2024].

[79]

"Additive Manufacturing Services," NC State University , [Online]. Available: https://www.ies.ncsu.edu/additive-manufacturing-services/#:~:text=Additive%20Manufacturing%20(AM)%20is%20the,is%20plastic%2C%20metal%20or%20concrete. [Accessed 5 February 2024].

[80]

"What is Additive Manufacturing," LinkedIn: Photocentric Group, 1 August 2022. [Online]. Available: https://www.linkedin.com/pulse/what-additive-manufacturing-photocentric-group/. [Accessed 5 February 2024].

[81]

"Ceramic Coating 3D Printed Parts," Formlabs, [Online]. Available: https://formlabs.com/blog/ceramic-coating-3d-printed-parts/. [Accessed 5 February 2024].

[82]

"How to Use 3D Printing for Injection Molding," Formlabs, [Online]. Available: https://formlabs.com/blog/3d-printing-for-injection-molding. [Accessed 5 February 2024].

[83]

"Automated Dispenser and method for dispensing," Eureka, 5 March 2015. [Online]. Available: https://eureka.patsnap.com/patent-US20150060482A1. [Accessed 5 February 2024].

"FDA Approved Food Grade Plastic," A&Cplasticsinc, [Online]. Available: https://www.acplasticsinc.com/informationcenter/r/food-grade-plastic-faq. [Accessed 5 February 2024].

"Food Safe 3D Printing," 3space, 23 September 2019. [Online]. Available: https://3space.com/food-safe-3d-printing/#:~:text=Food%20Safe%20FDM%20Materials,-The%20best%20materials&text=For%20FDM%203D%20printing%2C%20there,PEEK%2C%20and%20natural%20grade%20Nylon. [Accessed 5 February 2024].

"3-D Printed Parts for Food Contact," okstate, June 2022. [Online]. Available: https://extension.okstate.edu/fact-sheets/3-d-printed-parts-for-food-contact.html. [Accessed 5 February 2024].

"How to make food-grade 3D printed models," Prusa3d, 4 December 2020. [Online]. Available: https://blog.prusa3d.com/how-to-make-food-grade-3d-printed-models_40666/. [Accessed 5 February 2024].

"Is Epoxy Resin Food Safe," industrialclear, [Online]. Available: https://industrialclear.com/blogs/learn/food-safe-epoxy#:~:text=Yes%2C%20epoxy%20is%20considered%20food,either%20liquid%20or%20cured%20form. [Accessed 5 February 2024].

"What are FDA Approved Plastics for Food Contact," acplastics, [Online]. Available: https://www.acplasticsinc.com/informationcenter/r/fda-approved-plastics-for-food-contact#:~:text=%22FDA%20compliant%22%20means%20that%20a,it%20will%20be%20used%20in.. [Accessed 5 February 2024].

"ArtResin Epoxy Resin Passes Food Safety Tests," artresin, 21 October 2020.
91] [Online]. Available: https://www.artresin.com/blogs/artresin/artresin-passes-food-safety-tests. [Accessed 5 February 2024].

FDA Reader, "Equipment and Utensils," FDA Reader, 1 November 2018.
92] [Online]. Available: https://www.fdareader.com/blog/2018/12/17/facility-requirements-8yykg. [Accessed 5 February 2024].

Food Safety Magazine, "Hygienic Design of Equipment in Food Processing,"
93] Food Safety Magazine, 1 February 2003. [Online]. Available: https://www.food-safety.com/articles/4350-hygienic-design-of-equipment-in-food-processing. [Accessed 5 February 2024].

Anonymous, "THIS IS WHY ELECTRONIC DEVICES KEEP EXPLODING,"
94] Class Action, 6 June 2018. [Online]. Available: https://www.classaction.com/news/lithium-ion-battery-fires/. [Accessed 5 February 2024].

WebRoot, "A Review of Bluetooth Attacks and How to Secure Your Mobile
95] Device," WebRoot, [Online]. Available: https://www.webroot.com/us/en/resources/tips-articles/a-review-of-bluetooth-attacks-and-how-to-secure-mobile-workforce-devices. [Accessed 5 February 2024].

A. Hill and N. Olsen, "Do Spices Expire? Shelf Life and When to Toss Them,"
96] Healthline, 21 August 2020. [Online]. Available: https://www.healthline.com/nutrition/do-spices-expire#signs-of-spoilage. [Accessed 5 February 2024].

M. Rundle, "A beginner's guide to the Raspberry Pi Zero," Wired, 2015. [Online].
97] Available: https://www.wired.co.uk/article/raspberry-pi-zero-starter-guide. [Accessed 4 February 2024].

K. Meier, "The Basics of Stepper Motors," Digikey, [Online]. Available:
98] https://www.digikey.no/no/blog/the-basics-of-stepper-motors. [Accessed 21 January 2020].

"Run Apps on the Android Emulator," Android Developers, [Online]. Available:
99] https://developer.android.com/studio/run/emulator. [Accessed 3 March 2024].

M. Lynch, "5 Great Raspberry Pi IDEs for Programmers and Students," 26 June
100] 2023. [Online]. Available: https://www.thetechedvocate.org/5-great-raspberry-pi-ides-for-programmers-and-students/. [Accessed 4 March 2024].

P. Consani, "CLASSIC BLUETOOTH VS. BLUETOOTH LOW ENERGY
101] (BLE)," IoT Lab - Tertium Cloud Blog, [Online]. Available: https://iotlab.tertiumcloud.com/2020/08/19/classic-bluetooth-vs-bluetooth-low-energy-ble/. [Accessed 4 March 2024].

J. Eason, "Android Emulator - AMD Processor & Hyper-V Support," Android
102] Developers Blog, 9 July 2018. [Online]. Available: https://android-developers.googleblog.com/2018/07/android-emulator-amd-processor-hyper-v.html. [Accessed 4 March 2024].

G. Jones, "Cult of Mac," iOS 16’s handy Developer Mode lets you run your own
103] code, 24 August 2022. [Online]. Available: https://www.cultofmac.com/787783/enable-ios-developer-mode/. [Accessed 4 March 2024].

"Boston Globe," 14 November 2022. [Online]. Available:
104] https://www.bostonlobe.com/2022/11/14/business/how-keurig-made-its-coffee-machines-smarter. [Accessed 5 February 2024].

RCBS, "Uniflow Powder Measure Instructions," [Online]. Available:
105] https://www.rcbs.com/on/demandware.static/-/Library-Sites-HuntShootAccessoriesSharedLibrary/default/dw03425e12/productPdfFiles/rcbsPdf/Uniflow_Powder_Measure_Instructions.pdf. [Accessed 5 February 2024].

F. Cells. [Online]. Available: www.test.com.

106]

"Bu-104a: Comparing the Battery with Other Power Sources," [Online].
107] Available: https://batteryuniversity.com/article/bu-104a-comparing-the-battery-with-other-power-sources.. [Accessed 04 February 2024].

"Household Appliances," [Online]. Available: https://cornhusker-
108] power.com/rebates/household-appliances/.. [Accessed 04 February 2024].

"DPD2-BL Spice Dispenser," [Online]. Available: https://www.idm-
109] dispenser.com/products/dpd2-bl-spice-dispenser.. [Accessed 04 February 2024].

"Kitchenart Pro Series Spice Carousel," [Online]. Available:
110] https://www.kohls.com/product/prd-1110245/kitchenart-pro-series-spice-carousel.jsp. . [Accessed 04 February 2024].

"Bu-104a: Comparing the Battery with Other Power Sources," [Online].
111] Available: https://batteryuniversity.com/article/bu-104a-comparing-the-battery-with-other-power-sources.. [Accessed 04 February 2024].

TasteTro, "Tastetro Spice System at Chicago Housewares Show," Twitter, March
112] 2019. [Online]. Available: https://twitter.com/tastetro/status/1101951394389610496?s=61&t=uB4NV-TbnmKtsE54_9jerw. [Accessed 4 February 2024].

"The Inside of a Coke Freestyle Machine," Reddit, 2023. [Online]. Available:
113] https://www.reddit.com/r/mildlyinteresting/comments/13fv9qh/the_inside_of_a_coke_freestyle_machine/. [Accessed 4 February 2024].

Prusa Research, "LCD Menu (original Prusa i3)," Prusa Research, 2023. [Online].
114] Available: https://help.prusa3d.com/article/lcd-menu-original-prusa-i3_142322. [Accessed 4 February 2024].

Appendix A: Customer Requirements

Req. #	Description	Corresponding Eng. Requirement
C.1	Device will dispense in less than 5 minutes	24
C.2	Device will dispense accurately	12
C.3	Device will accommodate a variety of grain sizes	19
C.4	Device will fit into most larger cabinets	1
C.5	Containers should hold 8oz of spice	12
C.6	Fool Proof loading for spice both on bottle and machine	4
C.7	Be pleasing to look at	6,21,8
C.8	Spice drop zone should hold at least 1 US cup of spice	9
C.9	Easy Cleaning of machine	13,23
C.10	App should be easy to use	CS.C1.1-12
C.11	Physical Interface	25,27,11
C.12	Should warn the customer of errors	10
C.13	Should not be loud	15
C.14	Storing favorite recipes	Cs.F.1.8,16
C.15	Easy Disassembly and Reassembly	23
C.16	Feels strong and durable	28,21
C.17	Shows orders in progress	25,27
C.18	Machine should notify when order is done	27,CS.C.1.9

C.19	Shows spices that are in the machine	16,CS.F.1,4
C.20	Prevents spices from going everywhere	22
C.21	Allow for free flow of spice when the user wants	20

Appendix B: System Evaluation Plan

Prototyping Methodology

In building a final full-scale and functioning prototype, Flavor Fusion will be able to demonstrate product performance to key stakeholders and conduct testing opportunities that simulations alone are unable to offer. However, the final prototype will be the culmination of the other developmental prototypes and iterations completed before it. Thus far, the team has created an early cardboard mockup to understand the size and scale of the device, how we can improve initial component packaging, and begin visualizing an assembly process for the device. Additionally, a 1.5x scale print of the dispensing method was used to understand how we can best design the part for 3D printing, and how our end user would disassemble the system for cleaning. An additional learning from this was that print-in-place bearings will not work with the size requirements for the space. These initial prototype builds have allowed the team to have a greater understanding of the appearance and functionality of the device. The next steps are to begin splicing and printing our CAD model to be assembled in parts, as the dimensions we set forth are outside of the capabilities of most 3D printers. If we are able to successfully print using spliced parts, this will allow us to have a backup option in case Lockheed Martin is unable to deliver our final prototype before showcase. The spliced model will also allow the team to understand the fitment of the components and how accurate our GrabCAD models were when developing the device and fittings.

The advantages of the rapid prototyping provided by 3D printing allows the team to quickly test new dispensing designs and implement mounting points based on the actual parts received, not unreliable internet models. We can also use earlier prototype versions to test user functionality with actual users, giving us valuable insights on how to improve our application or physical device interactions. These studies would give us the best real-life testing opportunities for our application. While the physical durability of the device can be proven in simulation studies on common and uncommon grab points, stress testing on the rotational components, and even flow studies on the spices, simulating a diverse set of users is not as easily done. User testing will give us valuable consumer insights and user experience improvements that can be implemented to ultimately deliver a stronger device at showcase.

Physical Device Testing

Dimension and Accuracy Testing

Test Case 1: Purchased Components Dimensional Accuracy Testing

- **Objective:** Verify accuracy of purchased components with provided dimensions (if applicable)
- **Steps:**
    1. Collect purchased components' specification sheets from the manufacturer (if possible).
    2. Collect purchased components' 3D model from the manufacturer or GrabCAD (if possible)
    3. Using calipers, compare the dimensions of the physical component in hand with the dimensions provided in either the spec sheet or 3D models.
    4. If a spec sheet or 3D model is unavailable, gather the dimensions in inches to reconstruct the component in CAD.
    5. Verify the dimensions and ensure the distances add up correctly.
- **Expected Result:** Purchased components will have the correct dimensions and margin of error to be integrated into CAD with few issues on reconstruction.

Test Case 2: CAD Hole and Connection Alignment Testing

- **Objective:** Verify alignment of holes and connections in the CAD and physical models.
- **Steps:**
    1. Using mates in SolidWorks, set the connections to be concentric (if holes) or parallel/colinear, if surfaces.
    2. Verify the components align without collisions prior to printing, and address any errors by modifying the model (if necessary)
    3. If possible, use a 3D model of a purchased component that passed Test Case 1 to verify the alignment of those holes as well.
- **Expected Result:** The assembly of the final model will have fully aligned holes and not require extensive reworking.

Test Case 3: Collison Avoidance Testing

- **Objective:** Verify alignment of parts in the CAD and physical models when in motion.
- **Steps:**
    1. Using mates in SolidWorks, set the connections to be concentric (if holes) or parallel/colinear, if surfaces.
    2. Verify the components align without collisions prior to printing, and address any errors by modifying the model (if necessary)
    3. Once printed, assemble all components and test for collision by:
        - Rotating major components such as the carriage, spice bottles, and auger.

        - Moving components in and out such as the drop zone, drop zone container, and linear actuator.
- **Expected Result:** The assembly of the final model will not have any collisions and be able to move smoothly and without collisions.

Test Case 4: Cabinet/Pantry Storage Space Testing

- **Objective:** Verify device is able to fit in lower cabinet or on pantry shelf, and be safely transported between the storage location and countertop.
- **Steps:**
    1. Using physical device fully assembled, lift device off of countertop and place in lower cabinet to test fit with door shut.
    2. Move device to pantry shelf, note the weight balance of the machine.
    3. Return device to countertop.
- **Expected Result:** The device will easily and safely be able to fit on a countertop, inside a lower cabinet, or on a pantry shelf.

Test Case 5: Asymmetrical Loading Testing

- **Objective:** Ensure device can still reliably rotate with asymmetrical weight load on carriage due to spice filling.
- **Steps:**
    1. Fully fill spice containers 1 and 8 and load them into the carriage, leave the rest of the containers empty
    2. Allow the machine to calibrate itself, and then request a blend of the two spices in containers 1 and 8.
    3. Request the same blend 4 more times to give adequate rotations to the device.
    4. After dispensing, reload containers 1 and 8, and then fill containers 2 and 7. Repeat the process to ensure accurate dispensing with half of the containers filled.
- **Expected Result:** The device will be able to rotate accurately and still line up the containers each time without being affected by asymmetrical loading.

Test Case 6: Asymmetrical Loading Testing

- **Objective:** Ensure device can still reliably rotate with asymmetrical weight load on carriage due to spice filling.
- **Steps:**
    1. Fully fill spice containers 1 and 8 and load them into the carriage, leave the rest of the containers empty
    2. Allow the machine to calibrate itself, and then request a blend of the two spices in containers 1 and 8.
    3. Request the same blend 4 more times to give adequate rotations to the device.
    4. After dispensing, reload containers 1 and 8, and then fill containers 2 and 7. Repeat the process to ensure accurate dispensing with half of the containers filled.

- **Expected Result:** The device will be able to rotate accurately and still line up the containers each time without being affected by asymmetrical loading.

Test Case 7: Spice Dispensing Accuracy Testing

- **Objective:** Determine realistic accuracy for spice dispensing based on grain size.
- **Steps:**
    1. Fill containers with different spices with different grain sizes.
    2. Send spice order for 1 US teaspoon for one of the spices. Measure the output to determine how close the machine is able to reproduce 1 US teaspoon.
    3. Request the same spice 4 more times to give adequate trials and measurements.
    4. Repeat process with the other spices and determine if grain size has impact on dispensing amount.
    5. After dispensing 1 US teaspoon of each spice, reduce the amount in half and repeat steps until each spice is dispensing 1/8 US teaspoon.
- **Expected Result:** The dispensing accuracy will be verified and refined as the test is repeated and code is optimized.

Endurance Testing

Test Case 1: Operational Temperature Testing

- **Objective:** Ensure device maintains operating temperature without overheating under normal conditions.
- **Steps:**
    1. Power up device, ensure the fan is not blocked.
    2. Request a blend of 4 spices (containers 1, 3, 5, and 7) from the app.
    3. Request the same blend 4 more times with a minute between each request to give adequate rotations to the device.
    4. Once dispensed, power off the device and review temperature sensor reading.
- **Expected Result:** The device will maintain a temperature that doesn't cause overheating or melted plastic.

Test Case 2: Structural Loading Testing

- **Objective:** Ensure device is able to support 10lb of weight stacked on its lid.
- **Steps:**
    1. Place device on countertop with spice containers fully loaded.
    2. Gently place a 10lb weight on the lid of the device.
    3. Monitor the housing for any signs of stress or failure.
    4. Remove the weight after doing a visual inspection, and then do a second inspection after the weight is removed.
- **Expected Result:** The device will withstand the 10lb weight and remain structurally sound.

Test Case 3: Cleaning and Maintenance Testing

- **Objective:** Verify device can be disassembled and reassembled for cleaning.
- **Steps:**
    1. Remove container from device and separate it from dispensing mechanism.
    2. Remove auger from dispensing mechanism, and wash by hand.
    3. Test dishwasher safety with top rack and no heated dry for spice container.
    4. Reassemble container and dispensing mechanism.
    5. Disassemble the remainder of the user-serviceable components of the device.
    6. Reassemble entire device, power on and ensure everything works correctly.
- **Expected Result:** The device can be disassembled and reassembled safely and easily by users.

Test Case 4: Grab Point Testing

- **Objective:** Ensure device can withstand grab points besides the handles from users.
- **Steps:**
    1. With the device placed on a flat countertop, grab and pull the device horizontally by the drop zone towards the user.
    2. With the device placed on a flat countertop, lift the front of the device by the drop zone to get the front two feet off of the countertop, then drag the device towards the user.
    3. With the device placed on a flat countertop, reach around to the back of the device and pull it towards the user, like a hug.
    4. With the device placed on a flat countertop, grab the power cable and pull the device towards the user.
    5. Visibly inspect the interior and exterior of the device for damage.
- **Expected Result:** The device will be able to withstand users moving it without using the included handles.

Test Case 5: Drop Testing

- **Objective:** Verify device can withstand a drop onto the floor.
- **Steps:**
    1. Ensure all spice containers are filled properly.
    2. Lift the device using the handles so the base is approximately the same height from the ground as a countertop (approx. 36 in).
    3. Drop the device.
    4. Inspect the device for damage or leaking spices.
- **Expected Result:** The device can withstand a drop from countertop heights.

User Testing

Test Case 1: Display Operation Testing

- **Objective:** Verify user interface on device is intuitive and easy to follow.
- **Steps:**
    1. Ensure test is conducted with user who is not on Flavor Fusion team.
    2. Ask the user to dictate their thought process out loud as they navigate the interface.
    3. Ask user to dispense a pre-saved blend. Do not provide any additional information, and record how they navigate the interface.
    4. Once dispensed, ask the user to manually dispense 2 tablespoons of a specific spice. Again, do not provide any additional information and record how they navigate the interface.
- **Expected Result:** The user will be able to successfully complete the tasks and share feedback as they go.

Test Case 2: Application Operation Testing

- **Objective:** Verify user interface on application is intuitive and easy to follow.
- **Steps:**
    1. Ensure test is conducted with user who is not on Flavor Fusion team.
    2. Ask the user to dictate their thought process out loud as they navigate the mobile app interface.
    3. Ask user to dispense a pre-saved blend. Do not provide any additional information, and record how they navigate the interface.
    4. Once dispensed, ask the user to create a new blend in the app. Again, do not provide any additional information and record how they navigate the interface.
- **Expected Result:** The user will be able to successfully complete the tasks and share feedback as they go.

Test Case 3: Cleaning and Maintenance Testing

- **Objective:** Verify user is able to disassemble and reassemble device on their own.
- **Steps:**
    1. Ensure test is conducted with user who is not on Flavor Fusion team.
    2. Ask the user to dictate their thought process out loud as they complete the following steps.
    3. Ask user to remove the container and dispensing mechanism for cleaning. Do not provide any additional information, and record how they approach the problem.
    4. Once complete, ask user to remove the drop zone assembly and remove the display board. Do not provide any additional information and record how they approach the problem.

- **Expected Result:** The user will be able to successfully complete the tasks and share feedback as they go.

Test Case 4: Grab Point Testing

- **Objective:** Understand how users will physically move the device on the countertop.
- **Steps:**
    1. Ensure test is conducted with user who is not on Flavor Fusion team.
    2. Ask the user to dictate their thought process out loud as they complete the following steps.
    3. Ask user to bring the machine closer to them. Do not provide any additional information, and record how they approach the problem.
    4. Once complete, ask user to put the machine away in a pantry or lower cabinet. Do not provide any additional information and record how they approach the problem.
- **Expected Result:** The user will be able to successfully complete the tasks and share feedback as they go so the team can understand where the user will physically move the device.

Test Case 5: Spice Unloading Testing

- **Objective:** Verify user is able to remove and refill spices on their own.
- **Steps:**
    1. Ensure test is conducted with user who is not on Flavor Fusion team.
    2. Ask the user to dictate their thought process out loud as they complete the following steps.
    3. Ask user to remove one spice container and replace with a new spice in the device. Do not provide any additional information, and record how they approach the problem.
    4. Once complete, ask user to update their new spice in the app. Do not provide any additional information and record how they approach the problem.
- **Expected Result:** The user will be able to successfully complete the tasks and share feedback as they go.

Test Case 6: App and Device Initial Setup Testing

- **Objective:** Verify user is able to set up Flavor Fusion for the first time.
- **Steps:**
    1. Ensure test is conducted with user who is not on Flavor Fusion team. Ensure Flavor Fusion device is at default settings and ready to be set up for the first time.
    2. Ask the user to dictate their thought process out loud as they complete the following steps.

3. Ask user to download the app after being told the name and shown the icon. Do not provide any additional information, and record how they approach the problem.
4. Once complete, ask user to proceed through the initial set up process. Do not provide any additional information and record how they approach the problem.
5. Once complete, ask user to load spices into the machine. Do not provide any additional information and record how they approach the problem.
6. Once complete, ask user to save a blend in the app and send it to the machine for blending. Do not provide any additional information and record how they approach the problem.

- **Expected Result:** The team will understand how initial set up needs to work in order to best benefit the user.

Mobile Application Testing

User Interface Testing

The purpose of user interface testing is to ensure that the app's user interface (UI) is functional, user-friendly, and consistent across different iPhones. This plan will focus on verifying that all UI components behave as expected, that the layout is consistent, and that interactions are smooth and intuitive for the user. The three types of testing used for UI testing are functional, accessibility, and compatibility testing.

We need to ensure that all UI elements, such as buttons, menus, dropdowns, text fields, etc., are operating as intended. Each UI element should trigger the correct action when interacted with, such as navigating to the appropriate screen, or sending data. This ensures that every component responds appropriately to user input across various scenarios and workflows.

Login and Create Passcode Functional Tests

Test Case 1: Successful Login with Correct Passcode

- **Objective:** Verify that users can successfully log in using the correct passcode.
- **Steps:**
    1. Launch the app and navigate to the login screen.
    2. Enter the correct passcode.
    3. Verify that the user is successfully logged in and navigated to the home list screen.
- **Expected Result:** The app allows users to log in with the correct passcode and redirects them to the home screen without errors.

Test Case 2: Failed Login with Incorrect Passcode

- **Objective:** Verify that users cannot log in with an incorrect passcode.
- **Steps:**
    1. Launch the app and navigate to the login screen.
    2. Enter an incorrect passcode.
    3. Verify that the user cannot log in.
- **Expected Result:** The app does not allow the user to login and displays an appropriate error message.

Test Case 3: Create Passcode

- **Objective:** Verify that users can successfully create a new passcode during the first app launch.
- **Steps:**
    1. Launch the app after downloading.
    2. Create a new passcode.
    3. Verify that the passcode is successfully created, and the user is navigated to the home list.
- **Expected Result:** The app allows users to create a new passcode and redirects them to the home screen or dashboard upon successful passcode creation.

Main View Functional Tests

Test Case 1: Tabs

- **Objective:** Ensure that app includes all 4 tabs – Home List, Recipe Book, Settings, and About.
- **Steps:**
    1. Launch the app and navigate to each tab.
    2. Verify that the correct view renders when tapping on each tab.
- **Expected Result:** Each tab should render without errors.

Test Case 2: About View – Project Overview

- **Objective:** Ensure that About View includes a project overview.
- **Steps:**
    1. Launch the app and navigate to the About tab.
    2. Verify that the project overview exists and contains text.
- **Expected Result:** The text should render on the screen.

Test Case 3: About View – User Manual

- **Objective:** Ensure that About View includes a user manual.
- **Steps:**
    1. Launch the app and navigate to the About tab.
    2. Verify that the user manual exists and contains text.
- **Expected Result:** The text should render on the screen.

Test Case 4: About View – Privacy Information

- **Objective:** Ensure that About View includes privacy information.
- **Steps:**
    1. Launch the app and navigate to the About tab.
    2. Verify that the privacy information exists and contains text.
- **Expected Result:** The text should render on the screen.

Test Case 5: About View – Meet the Team

- **Objective:** Ensure that About View includes information about the Flavor Fusion team.
- **Steps:**
    1. Launch the app and navigate to the About tab.
    2. Verify that the meet the team section exits.
    3. Tap on each team member and verify that the correct biography exists for each team member.
- **Expected Result:** Each team member should have the correct biography.

Test Case 6: Home List

- **Objective:** Ensure that the home list tab renders on the screen.
- **Steps:**
    1. Launch the app.
    2. Verify that the home list renders with 10 spice containers.
    3. Tap each spice container and verify that the detailed view renders.
- **Expected Result:** The home list displays 10 spice containers and their details.

Test Case 7: Recipe Book

- **Objective:** Ensure that the recipe book tab renders on the screen.
- **Steps:**
    1. Launch the app.
    2. Verify that the recipe book renders with recipes.
- **Expected Result:** The recipe book displays recipes.

Test Case 8: Settings

- **Objective:** Ensure that the settings tab renders on the screen.
- **Steps:**
    1. Launch the app.
    2. Verify that the settings renders with recipes.
- **Expected Result:** The settings page shows the ability to change display name, update passcode, and toggle require passcode at startup.

Blending Functional Tests

Test Case 1: Blend New Blend

- **Objective:** Verify that users can successfully create a new spice blend that is not saved in the recipe book.
- **Steps:**
    1. Navigate to the “New” section of the blend view.
    2. Select individual spices and specify spice amounts for each spice.
    3. Type in a spice name.
    4. Verify that all the spices are in the blend confirmation.
- **Expected Result:** All selected spices and amounts, spice name, and correct number of servings should be visible in the blend confirmation view.

Test Case 2: Blend Existing Blend

- **Objective:** Verify that users can select a spice blend from the recipe book for blending.
- **Steps:**
    1. Navigate to the “Existing” section of the blend view.
    2. Select a recipe.
    3. Modify the number of servings for the recipe.
- **Expected Result:** The ingredient amounts should match the number of servings.

Test Case 3: Blend New Blend Without Blend Name

- **Objective:** Verify that users cannot create a blend without a name.
- **Steps:**
    1. Navigate to the “New” section of the blend view.
    2. Select individual spices and specify spice amounts for each spice.
    3. Do not type in a spice name.
- **Expected Result:** The app should display an error message stating that this is an incomplete blend.

Recipe Book Functional Tests

Test Case 1: Add a New Recipe

- **Objective:** Verify that users can successfully add a new recipe to the recipe book.
- **Steps:**
    1. Navigate to the "Add Recipe" screen from the recipe book.
    2. Enter valid details for the recipe.
    3. Navigate back to the recipe book list and check if the newly added recipe appears in the list.
- **Expected Result:** The new recipe is successfully added to the recipe book and is visible in the list with the correct details.

Test Case 2: Add Duplicate Recipe

- **Objective:** Verify that the app handles attempt to add a recipe with the same name as an existing recipe.
- **Steps:**
    1. Add a new recipe with a unique name.
    2. Attempt to add another recipe with the same name as the one just added.
    3. Check if the app allows the duplicate or provides a warning/error message.
- **Expected Result:** The app should prevent the duplicate recipe from being added and display a message.

Test Case 3: Delete a Recipe

- **Objective:** Verify that users can delete a recipe from the recipe book.
- **Steps:**
    1. Navigate to the recipe book and select an existing recipe.
    2. Tap the "Delete" button associated with the recipe.
    3. Confirm the deletion when prompted.
    4. Return to the recipe book list and check if the recipe has been successfully removed.
- **Expected Result:** The selected recipe is successfully deleted from the recipe book and no longer appears in the list.

Test Case 4: Cancel Recipe Deletion

- **Objective:** Ensure that users can cancel the recipe deletion process.
- **Steps:**
    1. Navigate to the recipe book and select a recipe to delete.
    2. Tap the "Delete" button and when prompted to confirm, choose "Cancel."
    3. Check if the recipe remains in the recipe book after cancellation.
- **Expected Result:** The deletion process is canceled, and the recipe remains in the recipe book without being removed.

Test Case 5: Search for a Recipe by Name

- **Objective:** Ensure that users can search for recipes by their name and retrieve relevant results.
- **Steps:**
    1. Navigate to the recipe book and enter a valid recipe name into the search bar.
    2. Execute the search.
    3. Review the search results to verify if the relevant recipe(s) is/are displayed.
- **Expected Result:** The app returns the correct recipe(s) that match the entered search term, and no irrelevant results are shown.

Test Case 6: Search with Partial Recipe Name

- **Objective:** Verify that users can perform a search using partial names and still retrieve relevant results.
- **Steps:**
    1. Navigate to the recipe book and enter a partial name of an existing recipe.
    2. Execute the search.
    3. Check if the results include recipes with names that contain the partial search term.
- **Expected Result:** The app returns all recipes that contain the partial name entered, making the search flexible and user-friendly.

Test Case 7: Case Sensitivity in Search

- **Objective:** Verify that the search function is not case-sensitive and returns results regardless of uppercase or lowercase input.
- **Steps:**
    1. Search for a recipe by entering its name in uppercase.
    2. Repeat the search with the same name in lowercase.
    3. Check if both searches return the same results.
- **Expected Result:** The app's search function should not be case-sensitive, and both uppercase and lowercase searches should return the same results.

Test Case 8: Add, Delete, and Search Integration Test

- **Objective:** Ensure that the add, delete, and search functionalities work together seamlessly.
- **Steps:**
    1. Add a new recipe to the recipe book.
    2. Search for the newly added recipe to verify that it appears in the results.
    3. Delete the recipe.
    4. Perform another search to ensure the deleted recipe no longer appears.
- **Expected Result:** The newly added recipe should appear in the search results after being added, and after deletion, it should no longer be found in the recipe book or search results.

Accessibility Testing

Accessibility testing ensures that the UI is accessible for users with disabilities by testing to ensure that text size adjustments don't break the layout, and make sure that all interactive elements, such as buttons and links, have appropriately sized touch targets for ease of use.

Test Case 1: Color Contrast

- **Objective:** Ensure that text and important UI elements have sufficient color contrast against their background for users with visual impairments.
- **Steps:**
    1. Use a color contrast analyzer tool to check the contrast ratio of text and interactive elements (buttons, links) against the background.
    2. Test in both light and dark mode to ensure contrast remains sufficient in both modes.
- **Expected Result:** The contrast ratio meets WCAG (Web Content Accessibility Guidelines) standards, with a minimum contrast ratio of 4.5:1 for regular text and 3:1 for large text and UI components.

Test Case 2: Light/Dark Mode Compatibility

- **Objective:** Verify that the app is accessible and usable in both light and dark modes.
- **Steps:**
    1. Switch between light mode and dark mode in the iPhone's display settings.
    2. Navigate through the app in both modes, paying attention to background colors, text visibility, and element contrast.
- **Expected Result:** The app's UI elements maintain clear visibility and proper contrast in both light and dark modes, with no readability issues or broken design elements.

Test Case 3: Font Scaling

- **Objective:** Ensure that the app's UI is responsive to dynamic font size changes for users who require larger or smaller text.
- **Steps:**
    1. Go to the iPhone's accessibility settings and adjust the font size to both the smallest and largest available options.
    2. Open the app and navigate through all screens.
    3. Verify that text remains readable, does not overflow and that the layout adjusts properly to accommodate larger or smaller font sizes.
- **Expected Result:** The UI adapts to all font scaling settings without layout issues, and text remains fully readable and accessible.

Compatibility Testing

Compatibility testing ensures consistency and adaptability of the UI on different iPhones. It confirms that features work seamlessly on both older and newer iPhone models, ensuring a reliable experience for all users regardless of their device specifications. This testing will focus on iPhones that support iOS 18 which is the latest version of iOS. These include the iPhone SE (2nd generation), iPhone SE (3rd generation), iPhone XR, iPhone Xs family, iPhone 11 and 11 Pro family, iPhone 12 and 12 Pro family, iPhone 13 and 13 Pro family, iPhone 14 and 14 Pro family, iPhone 15 and 15 Pro family, and iPhone 16 and 16 Pro family. Most of these iPhones will be tested using their simulators that are built into Xcode. However, we will also test using the iPhones that we own within the team.

Test Case 1: Layout Consistency Across Different iPhone Models

- **Objective:** Ensure that the app's layout and UI elements are consistent and responsive across different iPhone models and screen sizes.
- **Steps:**
    1. Run the app on different iPhone simulators and real iPhones (e.g., iPhone SE, iPhone 15 Pro Max, iPhone 12 Mini, etc.).
    2. Launch the app and navigate through all screens, focusing on layout and UI components like buttons, text fields, images, and menus.
    3. Compare the layout across devices and confirm that no UI elements are distorted, cut off, or misaligned.
- **Expected Result:** The app's layout and UI components adjust properly to different screen sizes, with no elements being distorted or out of place.

Test Case 2: Landscape and Portrait Mode Compatibility

- **Objective:** Ensure that the app functions correctly in both portrait and landscape orientations on different iPhones.
- **Steps:**
    1. Run the app on different iPhone simulators and real iPhones (e.g., iPhone 15, iPhone 12 Mini, iPhone SE).
    2. Switch between portrait and landscape mode while using the app.
    3. Verify that the UI elements adjust appropriately to the new orientation and no content is distorted or lost.
- **Expected Result:** The app adapts smoothly between portrait and landscape orientations across all iPhones, with no layout issues or distorted content.

Test Case 3: Push Notification Compatibility Across Models

- **Objective:** Ensure that push notifications are received and displayed correctly across different iPhone models.
- **Steps:**
    1. Enable push notifications for the app on various iPhones.
    2. Send a test notification and check its appearance and functionality (e.g., tap to open the app).
    3. Ensure that the notification appears properly formatted and that it is tappable and leads to the correct screen within the app.
- **Expected Result:** Push notifications are received and displayed properly on all iPhones, with clear formatting and correct navigation behavior when tapped.

Bluetooth Testing

The purpose of Bluetooth testing is to ensure that the app's Bluetooth functionality reliably connects the iPhone to the spice maker. This testing verifies that the app can successfully discover, connect, and exchange data with the spice maker, ensuring a smooth user experience and accurate blending operations. Connectivity and data transfer will be tested during Bluetooth testing.

Connectivity Testing

Ensure that the Spice Maker app can discover and establish stable connections with the spice maker device via Bluetooth. This includes testing initial connections, reconnections after disconnections, and maintaining connections over extended periods during blending operations. In addition, connectivity testing will determine the maximum range at which the iPhone can maintain a stable Bluetooth connection with the spice maker. This ensures that users can control the device from various locations in their kitchen without losing connectivity.

Test Case 1: Bluetooth Enabled on Device

- **Objective**: Verify that the app prompts the user to enable Bluetooth if it is disabled.
- **Steps**:
    1. Open the app.
- **Expected Result**: The app prompts the user to enable Bluetooth.

Test Case 2: Device Scanning for Spice Maker

- **Objective**: Ensure that the app successfully scans and finds the Spice Maker (Arduino) device.
- **Precondition**: Spice Maker is powered on and within range, Bluetooth enabled on iPhone.
- **Steps**:
    1. Open the app.
    2. Start the device scanning process.
- **Expected Result**: The app detects the Spice Maker.

Test Case 3: Successful Connection to Spice Maker

- **Objective**: Verify that the app successfully establishes a connection to the Spice Maker.
- **Steps**:
    1. Open the app.
    2. Scan for the Spice Maker device.
    3. Tap to connect to the device.
- **Expected Result**: The app successfully connects to the Spice Maker, and a "Connected" status is displayed in the console on Xcode.

Data Transfer Testing

Ensures that spice data (container numbers, amounts, blending complete indicator) is accurately transmitted from the spice maker to the iPhone. This ensures users receive feedback on blend status and users can view the accurate spice amounts on the app. Similarly, verify that blend data (container numbers and amounts) are accurately transmitted from the iPhone to the spice maker. This ensures that the spice maker receives precise spice amounts from the app.

Test Case 1: Successful Sending of Blend Data

- **Objective**: Verify that the blend data is successfully sent to the Arduino over Bluetooth.
- **Steps**:
    1. Ensure the app is connected to the Arduino via Bluetooth.
    2. Create or select a blend in the app.
    3. Tap the “Blend” Button.
    4. Verify that the blend is shown in the serial monitor on the Arduino.
- **Expected Result**: The app successfully sends the blend data to the Arduino, and the Arduino confirms receipt.

Test Case 2: Successful Sending of Blend Data with large Payload

- **Objective**: Ensure that large blend data (multiple spices with complex details) can be sent successfully to the Arduino.
- **Steps**:
    1. Ensure the app is connected to the Arduino via Bluetooth.
    2. Create a blend with many spices and amounts.
    3. Tap the “Blend” Button.
    4. Verify that the blend is shown in the serial monitor on the Arduino.
- **Expected Result**: The blend data is sent without any delay or failure, and the Arduino confirms receipt of the entire payload.

Test Case 3: Successful Switch to Blend Complete View on Data Receipt

- **Objective**: Verify that the app switches to the "Blend Complete" view when the Arduino sends the blend complete message.
- **Steps**:
    1. Ensure the app is connected to the Arduino via Bluetooth.
    2. Start the blending process on the app.
    3. The Arduino completes the blending and sends a "Blend Complete" message.
- **Expected Result**: The app switches to the "Blend Complete" view, displaying the blend completion message.

End to End Testing

End-to-end (E2E) testing is crucial for ensuring the complete functionality of an application by testing the entire workflow from start to finish. Its purpose is to validate that all integrated components work together as expected. E2E testing helps identify potential issues that could arise in the actual user environment, ensuring that the app functions reliably and efficiently under real-world conditions, ultimately improving the user experience and reducing the likelihood of post-release defects. The below test case is an example of one full end to end test.

Objective:
To verify that the mobile app can successfully connect to the Bluetooth-enabled spice maker, send a customized spice blend, and ensure proper communication, data transmission, and system responses between the app and the spice maker.

Preconditions:

- The mobile app is installed.
- The mobile device's Bluetooth is turned on and permissions are granted.
- The spice maker device is powered on and within Bluetooth range.
- The user has successfully logged in to the app and already has a passcode.
- A list of spices and quantities is available within the app.

Test Steps:

1. **Launch the App**
    - Open the mobile app.
    - Verify the app opens to the main home spice list screen.
2. **Navigate to Spice Blending Section**
    - Tap on the "Blend" button on the home screen.
    - Verify the app navigates to the spice blend creation interface.
    - Select spices and quantities from the list, or input a new custom blend (e.g., 1 tsp of Cinnamon, 1/2 tsp of Nutmeg, 2 tbsp of Paprika).
3. **Send Spice Blend to Spice Maker**
    - Tap the blend button.
    - Verify that the blend confirmation is correct, and tap confirm.
    - Verify that a progress indicator appears on the app (e.g., "Sending blend..."), showing the transmission of the blend.
    - Ensure the spice maker responds.

4. **Confirm Spice Maker Received Blend**
    - Verify that the spice maker acknowledges receiving the spice blend.
5. **Monitor Spice Dispensing Process**
    - Observe the spice maker as it processes the blend and dispenses the correct amounts of each selected spice.
    - Ensure the spices are dispensed accurately based on the user's blend input (e.g., 1 tsp of Cinnamon, 1/2 tsp of Nutmeg, 2 tbsp of Paprika).
    - Check that the spice maker completes the blend within the expected time frame.
6. **Receive Completion Notification**
    - Once the spice maker has completed dispensing the blend, verify that the app displays a "Blending complete!" notification.
    - Verify that the spice maker's display also indicates that the blend is complete.
7. **Return to Home Screen**
    - Tap on the "Home" button or icon.
    - Verify that the app navigates back to the main screen without any errors.
8. **Disconnect from Spice Maker**
    - Verify that the app successfully disconnects from the spice maker when the user closes the app.

Expected Results:

- The app successfully connects to the spice maker over Bluetooth.
- The app sends the correct spice blend data, and the spice maker acknowledges receiving it.
- The spice maker dispenses the correct quantities of each spice.
- The app displays appropriate messages and progress indicators throughout the process.
- The app properly disconnects from the spice maker after the user closes the app.
- The entire process is smooth, with no errors, crashes, or Bluetooth disconnections.

Postconditions:

- The app and the spice maker are successfully disconnected.
- The user is returned to the home screen of the app.
- The spice maker has dispensed the correct blend.

Electronics Testing

Electronics testing will be conducted with individual components under controlled conditions before proceeding to integration. By wiring, powering, and programming parts on their own, mismatches and bugs can be isolated and identified. In contrast, searching through an entire circuit or Arduino sketch for over 30 components simultaneously can pose a challenge. In this way, the objective is for each component to be fully understood under controlled conditions, for iterative prototyping to occur *before* interactions are introduced, and for proofs-of-concept to be generated as evidence of success.

Whereas 3D printing can be prototyped according to scale models, electronic systems can be generalized by breaking the entire system into subsystems and breadboarding for rapid, temporary circuits. During the subsystem breadboarding process, it is useful for components to be as similar as possible to the intended parts for the final system; it is inefficient to design circuits and program components that will inevitably be changed. One never knows how a part will behave until the part itself is tested.

Actuation Subsystem Testing

The actuation subsystem consists of all electronics directly necessary for motion in Flavor Fusion. An A4988 stepper driver and NEMA 11 stepper motor control the linear rail, a TMC2209 and NEMA 8 control the auger, and a TMC2209 and NEMA 17 drive the spice carriage.

Stepper Motors

Test Case 1: Linear Rail NEMA 11 and A4988

- **Objective:** Control rail to move NEMA 8 bidirectionally.
- **Steps:**
    1. Connect motor and driver to power and Arduino. Check pins, connection types, and secure exposed wires.
    2. Check power supply to verify input voltage and current draw.
    3. Write a function in Arduino IDE for motor actuation.
    4. Conduct 5+ trials to move various distances.
- **Expected Result:** NEMA 8 can insert into and extract from auger drive shaft.

Test Case 2: Auger Driver NEMA 8

- **Objective:** Control NEMA 8 to dispense spice.
- **Steps:**
    1. Connect motor and driver to power and Arduino. Check pins, connection types, and secure exposed wires.
    2. Check power supply to verify input voltage and current draw.
    3. Write a function in Arduino IDE for motor actuation.
    4. Conduct 5+ trials to move various angles and speeds.
- **Expected Result:** Motor can turn at various speeds for efficiency and accuracy. Exit angle of motor is controlled for shaft interfacing.

Test Case 3: Carriage Driver NEMA 17

- **Objective:** Control NEMA 17 to rotate spice carriage.
- **Steps:**
    1. Connect motor and driver to power and Arduino. Check pins, connection types, and secure exposed wires.
    2. Check power supply to verify input voltage and current draw.
    3. Write a function in Arduino IDE for motor actuation.
    4. Conduct 5+ trials to rotate to various positions.
- **Expected Result:** NEMA17 can rotate the carriage until the calibration point is reached, then turn to a desired location with enough precision to insert the auger drive shaft.

Stepper Drivers

Test Case 1: TMC2209 UART microstepping

- **Objective:** Use serial communication to improve microstepping with TMC2209 driver.
- **Steps:**
    1. Connect motor and driver to power and Arduino. Check pins, connection types, and secure exposed wires.
    2. Check power supply to verify input voltage and current draw.
    3. Write a function in Arduino IDE for precise UART control.
    4. Conduct 5+ trials to rotate motors at various step sizes.
- **Expected Result: The TMC2209 can drive motors at various rates and step sizes to prioritize speed vs precision during dispensing.**

Test Case 2: TMC2209 Auger Control

- **Objective:** Use TMC2209 microstepping to improve auger precision.
- **Steps:**
    1. Connect motor and driver to power and Arduino. Check pins, connection types, and secure exposed wires.

2. Check power supply to verify input voltage and current draw.
3. Write a function in Arduino IDE for precise auger control.
4. Conduct 5+ trials to drive with various step sizes.

- **Expected Result: Precision of dispensing and extraction angle are improved due to microstepping.**

Test Case 3: TMC2209 Carriage Control

- **Objective:** Use TMC2209 microstepping to improve carriage rotation precision.
- **Steps:**
    1. Connect motor and driver to power and Arduino. Check pins, connection types, and secure exposed wires.
    2. Check power supply to verify input voltage and current draw.
    3. Write a function in Arduino IDE for precise carriage control.
    4. Conduct 5+ trials to rotate with various step sizes.
- **Expected Result: Precision of carriage rotation and linear rail alignment is improved due to microstepping.**

Display Subsystem Testing

The display subsystem consists of an LCD 128x64 screen with a stop button and potentiometer scroll wheel. These parts will allow users to view data from the machine and input job requests.

LCD Screen Testing

Test Case 1: Spice Selection Screen

- **Objective:** Display container numbers and spice names, dispense amounts, and units.
- **Steps:**
    1. Connect LCD to power and Arduino. Check pins, connection types, and secure exposed wires.
    2. Check power supply to verify input voltage and current draw.
    3. Write a function in Arduino IDE for selection screen display.
    4. Conduct 5+ trials to display various combinations of spice names.
- **Expected Result: All container names, amounts, and units are represented without symbol errors. Scrolling allows the user to view the entire list.**

Test Case 2: Dispensing In-Progress Screen

- **Objective:** Alert user that dispensing is in progress. Stretch goal: display completion status.
- **Steps:**
    1. Connect LCD to power and Arduino. Check pins, connection types, and secure exposed wires.
    2. Check power supply to verify input voltage and current draw.
    3. Write a function in Arduino IDE for dispensing screen display.
    4. Conduct 5+ trials to display for various amounts of time, under various job requests.
- **Expected Result: The dispensing screen lasts for the duration of job completion and informs users with any necessary information about job status.**

Test Case 3: Completed Job Screen

- **Objective:** Inform users when a job has been completed.
- **Steps:**
    1. Connect LCD to power and Arduino. Check pins, connection types, and secure exposed wires.
    2. Check power supply to verify input voltage and current draw.
    3. Write a function in Arduino IDE for job completion screen.
    4. Conduct 5+ trials to display various job output messages.
- **Expected Result: Display a message that dispensing is complete, with an (optional) spicy saying for fun. Stretch goal: display the blend name dispensed.**

Test Case 4: Stop Button

- **Objective:** LCD button stops the dispensing process, serving as a soft-stop option.
- **Steps:**
    1. Connect LCD to power and Arduino. Check pins, connection types, and secure exposed wires.
    2. Check power supply to verify input voltage and current draw.
    3. Write a function in Arduino IDE to stop dispensing.
    4. Conduct 5+ trials to stop dispensing at various times for various blends.
- **Expected Result: LCD reads that job has been cancelled, spice amounts are updated to the app, and motors are extracted and have their positions reset.**

Potentiometer Testing

Test Case 1: Scrolling Function

- **Objective:** Users can scroll through list-style screens using the potentiometer wheel.
- **Steps:**
    1. Connect LCD to power and Arduino. Check pins, connection types, and secure exposed wires.
    2. Check power supply to verify input voltage and current draw.
    3. Write a function in Arduino IDE for scrolling.
    4. Conduct 5+ trials to scroll various distances on various lists.
- **Expected Result: The scroll wheel outputs a consistent, user-friendly scroll speed. Stretch goal: add clicks or tick marks that correspond to scroll ticks.**

Test Case 2: Selection Function

- **Objective:** Users can press the potentiometer to select options.
- **Steps:**
    1. Connect LCD to power and Arduino. Check pins, connection types, and secure exposed wires.
    2. Check power supply to verify input voltage and current draw.
    3. Write a function in Arduino IDE for selection.
    4. Conduct 5+ trials to select various spices and amounts.
- **Expected Result: The potentiometer is easy to press and selects an option when used, advancing to the next selection screen.**

Sensor Subsystem Testing

The sensor subsystem consists of limit switches for cup detection and carriage calibration, a temperature and humidity sensor to monitor thermal conditions, and a load cell with an HX711 amplifier as a stretch goal to detect mass of spice dispensed.

Limit Switch Testing

Test Case 1: Cup Detection

- **Objective:** Limit switch detects the presence of the drop zone cup.
- **Steps:**
    1. Connect limit switch to power and Arduino. Check pins, connection types, and secure exposed wires.
    2. Check power supply to verify input voltage and current draw.
    3. Write a function in Arduino IDE for limit switch detection.
    4. Conduct 5+ trials to detect the cup for various amounts of time and loading conditions (inserted quickly, forcefully, gently, etc.).
- **Expected Result: The limit switch consistently signals the Arduino for the duration of the cup's insertion.**

Test Case 2: Carriage Calibration

- **Objective:** Limit switch is triggered to reset carriage positioning after a job.
- **Steps:**
    1. Connect limit switch to power and Arduino. Check pins, connection types, and secure exposed wires.
    2. Check power supply to verify input voltage and current draw.
    3. Write a function in Arduino IDE for carriage positioning calibration.
    4. Conduct 5+ trials to calibrate the carriage from various arbitrary positions.
- **Expected Result: The limit switch is consistently triggered during calibration, consistently signals activation to the Arduino, and is activated with sufficient precision and timing for linear rail to be aligned.**

Temperature and Humidity Sensor Testing

Test Case 1: Low Temperature

- **Objective:** Temperature sensor is accurate at low temperatures (approx. 32ºF).
- **Steps:**
    1. Connect sensor to power and Arduino. Check pins, connection types, and secure exposed wires.
    2. Check power supply to verify input voltage and current draw.
    3. Write a function in Arduino IDE for temperature sensing.
    4. Conduct 5+ trials to sense cold temperatures various times. This can be accomplished with bags of ice water and a digital thermometer.
- **Expected Result: The sensor detects temperatures within its rated margin of error and environmental uncertainty.**

Test Case 2: High Temperature

- **Objective:** Temperature sensor is accurate at high temperatures (above 100ºF).
- **Steps:**
    1. Connect sensor to power and Arduino. Check pins, connection types, and secure exposed wires.
    2. Check power supply to verify input voltage and current draw.
    3. Write a function in Arduino IDE for temperature sensing and emergency stop for dangerous temperatures.
    4. Conduct 5+ trials to sense hot temperatures various times. This can be accomplished with bags of hot water and a digital thermometer.
- **Expected Result: The sensor detects temperatures within its rated margin of error and environmental uncertainty. Dispensing is stopped when temperatures exceed rated safe conditions for materials and components. Stretch goal: fan speed responds to sensed temperature to save power and noise.**

Test Case 3: Prolonged Temperature Readings

- **Objective:** Temperature sensor can accurately measure temperatures over long durations.
- **Steps:**
    1. Connect sensor to power and Arduino. Check pins, connection types, and secure exposed wires.
    2. Check power supply to verify input voltage and current draw.
    3. Write a function in Arduino IDE for temperature sensing.
    4. Conduct 5+ trials to test prolonged readings at various temperatures.
- **Expected Result: Temperature sensor does not display drift error over time outside its rated conditions and environmental uncertainty.**

Load Cell and HX711 Amplifier Testing

Test Case 1: Static Loading for Various Weights

- **Objective:** Measure set weights using a load cell and signal amplifier.
- **Steps:**
    1. Connect sensor to power and Arduino. Check pins, connection types, and secure exposed wires.
    2. Check power supply to verify input voltage and current draw.
    3. Write a function in Arduino IDE for weight sensing.
    4. Conduct 5+ trials to sense various weights.
- **Expected Result: The sensor can detect weights accurately within its rated margin of error.**

Test Case 2: Gradual Loading

- **Objective:** Measure spice by weight, during dispensing, using a load cell and signal amplifier.
- **Steps:**
    1. Connect sensor to power and Arduino. Check pins, connection types, and secure exposed wires.
    2. Check power supply to verify input voltage and current draw.
    3. Write a function in Arduino IDE for weight sensing and PID dispensing control.
    4. Conduct 5+ trials to sense and dispense various amounts of spice by weight.
- **Expected Result: The sensor resolution allows detection of spice in grams and PID controller can output spice levels within a gram of the target amount.**

Test Case 3: Recalibration

- **Objective:** User can select a calibration option before dispensing spice by weight.
- **Steps:**
    1. Connect sensor to power and Arduino. Check pins, connection types, and secure exposed wires.
    2. Check power supply to verify input voltage and current draw.
    3. Write a function in Arduino IDE for calibration.
    4. Conduct 5+ trials to calibrate after various conditions.
- **Expected Result: Load cell signal will be zeroed after calibration so that the weight of the drop zone cup is not considered.**

Power Subsystem Testing

The power subsystem is designed to distribute power to all other electrical components while regulating voltage, current, and heat. It consists of a power supply unit (PSU), 3-prong power outlet cord, rocker switch, and CPU fan.

PSU Testing

Test Case 1: Voltage Regulation

- **Objective:** The PSU consistently outputs 12VDC.
- **Steps:**
    1. Connect PSU to power and multimeter. Check pins, connection types, and secure exposed wires.
    2. Conduct 5+ trials to measure voltage various times.
- **Expected Result: The PSU outputs power with negligible voltage fluctuations, if any.**

Test Case 2: Heat Regulation

- **Objective:** The PSU maintains safe temperatures for materials and components.
- **Steps:**
    1. Connect PSU to power, electronics to draw power, and thermometer. Check pins, connection types, and secure exposed wires.
    2. Conduct 5+ trials to measure temperature under various amounts of current draw.
- **Expected Result:** Temperatures are safe for operation (a fan can be added if necessary).

Power Cord Testing

Test Case 1: Secureness Testing

- **Objective:** The power cord is fastened securely and cannot be tugged out of place.
- **Steps:**
    1. Insert cord within device housing. Check connection types and secure exposed wires.
    2. Fasten the power cord to the device housing.
    3. Conduct 5+ trials to ensure the cord withstands various load cases (user or child tugging on it, device being dropped, etc.).
- **Expected Result: The cord stays fastened and live wires are never exposed to the user.**

Rocker Switch Testing

Test Case 1: Emergency Power Cutoff

- **Objective:** The rocker switch kills all device power, for emergencies.
- **Steps:**
    1. Connect rocker switch to cords. Check connection types and secure exposed wires.
    2. Build a circuit to indicate open or closed connection, such as an LED indicator of power or a multimeter. Connect electronics to draw a realistic amount of power.
    3. Conduct 5+ trials to test various component configurations.
- **Expected Result: The rocker switch removes power from all components.**

Fan Testing

Test Case 1: Constant (RAMPS Aux) Fan

- **Objective:** The fan can be run at a constant rate for cooling.
- **Steps:**
    1. Connect fan to power and Arduino. Check pins, connection types, and secure exposed wires.
    2. Check power supply to verify input voltage and current draw.
    3. Conduct 5+ trials to ensure performance under various stages of the dispensing process.
- **Expected Result: The fan consistently outputs its maximum speed for a constant 12V supply of power. This coincides with the auxiliary fan in the Ramps board configuration.**

Test Case 2: Controllable (RAMPS Extruder) Fan

- **Objective:** The fan speed can be controlled.
- **Steps:**
    1. Connect fan to power and Arduino. Check pins, connection types, and secure exposed wires.
    2. Check power supply to verify input voltage and current draw.
    3. Write a function in Arduino IDE for controlling fan speed according to temperature.
    4. Conduct 5+ trials to test speeds at various temperatures.
- **Expected Result: The fan speed can be controlled by altering its input voltage in response to sensed temperatures, allowing for energy efficiency.**

Microcontroller Subsystem Testing

The microcontroller subsystem refers to the microcontroller unit (MCU), in addition to peripherals that will assist in communication and processing data, such as a Bluetooth module.

Bluetooth Testing

Test Case 1: Receive and Display App Order

- **Objective:** Data from the Flavor Fusion app is received by the MCU and displayed for verification.
- **Steps:**
    1. Connect Bluetooth, power, and Arduino. Check pins, connection types, and secure exposed wires.
    2. Check power supply to verify input voltage and current draw.
    3. Write a function in Arduino IDE for Bluetooth communication.
    4. Conduct 5+ trials to receive and display various spice orders.
- **Expected Result:** All spice data is successfully received within seconds, then displayed to prove that data can be interpreted.

Test Case 2: Update App Data After App Order

- **Objective:** Spice level data is updated in the app after an order has been placed and dispensed from the app.
- **Steps:**
    1. Connect Bluetooth, power, and Arduino. Check pins, connection types, and secure exposed wires.
    2. Check power supply to verify input voltage and current draw.
    3. Write a function in Arduino IDE for Bluetooth communication.
    4. Conduct 5+ trials to update various amounts and blends of spice.
- **Expected Result:** All spice data is received by the app within seconds, then updated within the app to reflect the dispensed amount.

Test Case 3: Update App Data After Local Order

- **Objective:** Spice level data is updated in the app after an order has been placed and dispensed from the local device.
- **Steps:**
    1. Connect Bluetooth, power, and Arduino. Check pins, connection types, and secure exposed wires.
    2. Check power supply to verify input voltage and current draw.
    3. Write a function in Arduino IDE for Bluetooth communication.
    4. Conduct 5+ trials to update various amounts and blends of spice.
- **Expected Result:** All spice data is received by the app within seconds, then updated within the app to reflect the dispensed amount.

Aesthetic Subsystem Testing

The aesthetic subsystem relates to all electronics used for decoration or enhancement of the Flavor Fusion device without being critical to performance and safety. While parts like LEDs may play an important role in user interactions and ease of use, they are ultimately nonvital.

LED Testing

Test Case 1: Individual Colored LEDs

- **Objective:** LEDs can be illuminated individually or in patterns to indicate device status or for aesthetic appeal.
- **Steps:**
    1. Connect LEDs to power and Arduino. Check pins, connection types, and secure exposed wires.
    2. Check power supply to verify input voltage and current draw.
    3. Write a function in Arduino IDE for LED control.
    4. Conduct 5+ trials to display various configurations.
- **Expected Result: LEDs respond in the desired illumination pattern and are bright enough to be clearly visible on the device.**

Test Case 2: RGB LEDs

- **Objective:** RGB LEDs can be illuminated individually or in patterns across a range of colors.
- **Steps:**
    1. Connect LEDs to power and Arduino. Check pins, connection types, and secure exposed wires.
    2. Check power supply to verify input voltage and current draw.
    3. Write a function in Arduino IDE for LED color control.
    4. Conduct 5+ trials to display various colors.
- **Expected Result: RGB LEDs can display a range of distinct colors to signify various purposes or fulfill aesthetic appeal.**

Appendix C: User Manual

Mobile App User Manual

Setting up the Flavor Fusion iPhone App

System Requirements

1. Device Compatibility: iPhone running iOS version 17.0 or later.
2. Network: A stable internet connection (Wi-Fi or cellular data).

Installing an App from the App Store

1. Open the App Store: Tap the App Store icon on your iPhone's home screen.

2. Search for the App: Tap the search bar and type in the app name (e.g., “Flavor Fusion”).
3. Download the App: When you find the app in the search results, tap the Get button. You may be asked to authenticate using Face ID, Touch ID, or your Apple ID password.
4. Wait for Installation: The app will start downloading and automatically install on your device. You’ll see a loading circle that fills as the app installs.
5. Open the App: Once installed, tap Open from the App Store, or locate the app icon on your home screen and tap the app icon.

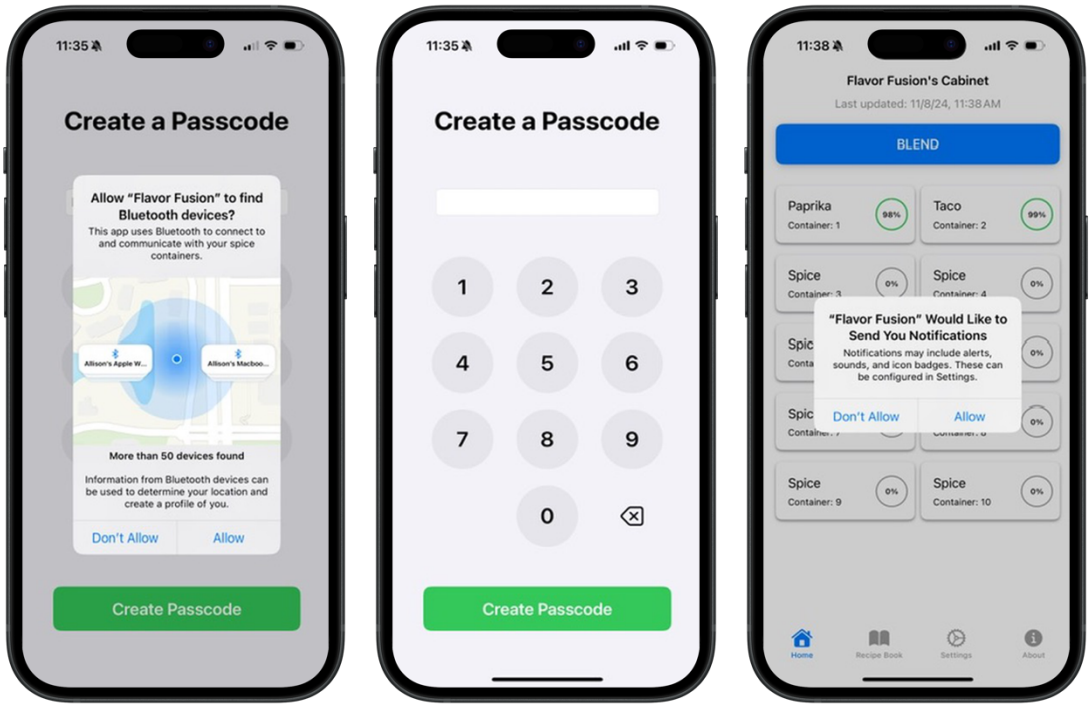
Installing from Test Flight (Public Beta Only)

1. Download TestFlight: Download and install the app “Test Flight” using the “Installing an App from the App Store” instructions above.
2. Open the Invitation or use Public URL: If you receive an email invitation link, tap the link, which will open TestFlight. If you do not receive an email invitation link, navigate to the following URL: https://testflight.apple.com/join/4ZzgWxdH.
3. Accept the Invitation: Follow the prompts in TestFlight to accept the invitation.
4. Download the App: Once accepted, tap Install in TestFlight to download the app.

Using the Flavor Fusion iPhone App

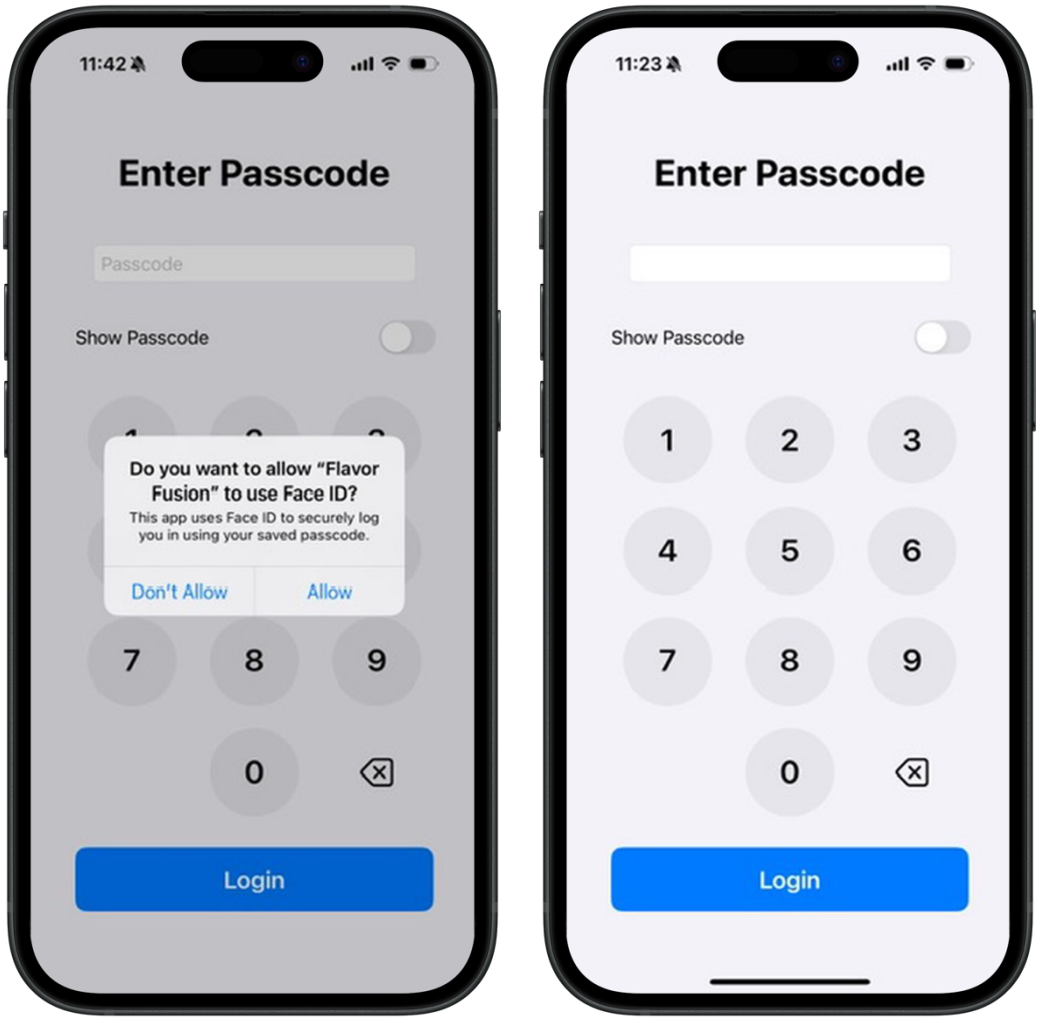
First Launch

1. Enable Bluetooth: Upon first launch of the app, you will see the Alert “Allow ‘Flavor Fusion’ to find Bluetooth devices?” You will need to **Allow** this alert to use the app with the spice maker.
2. Create Passcode: On the **Create Passcode** screen, enter a passcode in the passcode field. Tap **Create Passcode** to create the passcode.
3. Allow Notifications: When you get to the **Home** screen you will see the Alert “‘Flavor Fusion’ Would like to Send You Notifications.” You will need to **Allow** this alert to receive notifications from the app when a container is running low or your blend is complete.



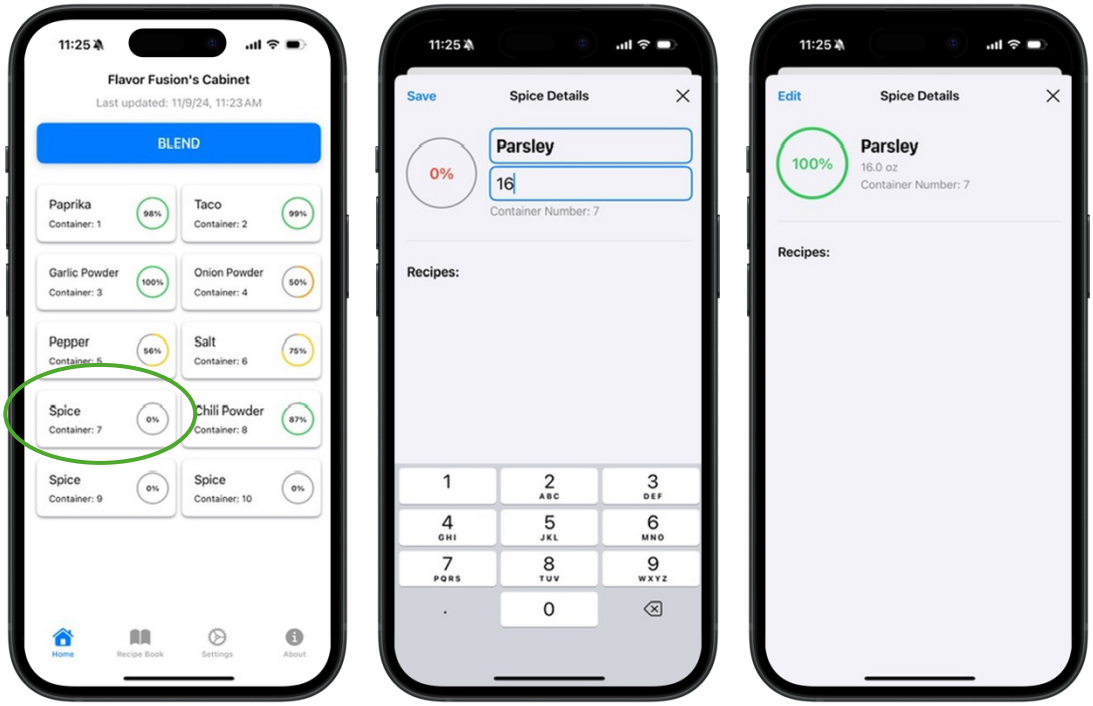
Login with Passcode/Face ID

1. Login with Face ID: Upon launching the app, you will see the Alert “Do you want to allow ‘Flavor Fusion’ to use Face ID?’ You will need to **Allow** this alert to use Face ID to login to the app.
2. Login with Passcode: On the Create Passcode screen, enter your passcode in the passcode field. Tap **Login** to login to the app.



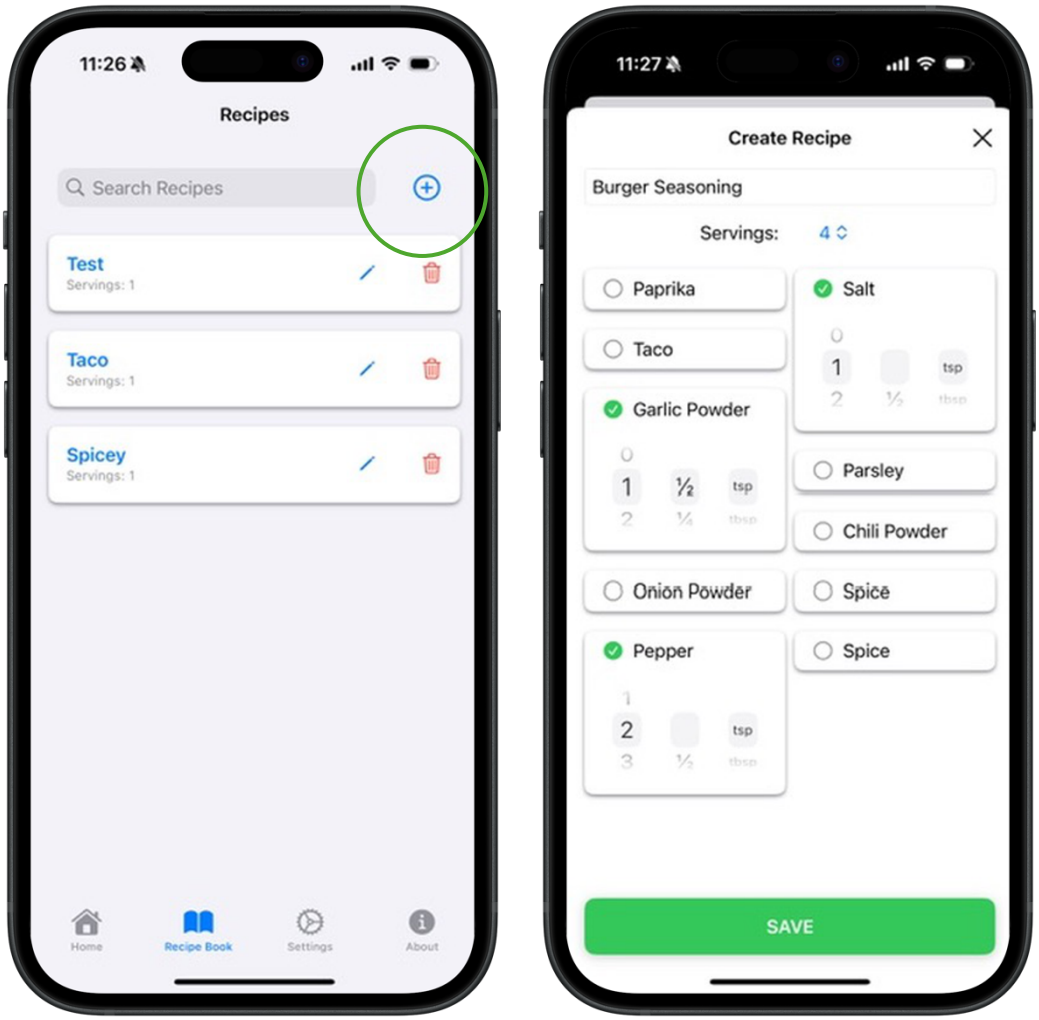
Adding a New Spice

1. Navigate to Home: Open the app and go to the **Home** tab. Locate and tap on an empty container. This will open the **Spice Details** screen.
2. Adding Spice Information: On the **Spice Details** screen, tap the **Edit** button. In the text fields, type in the name of the spice and specify the amount currently in the container.
3. Save Changes: Once you've entered the spice details, tap the **Save** button to store the information. To close the **Spice Details** screen, either tap the **'X'** or swipe down on the screen. This will return you to the Home tab.



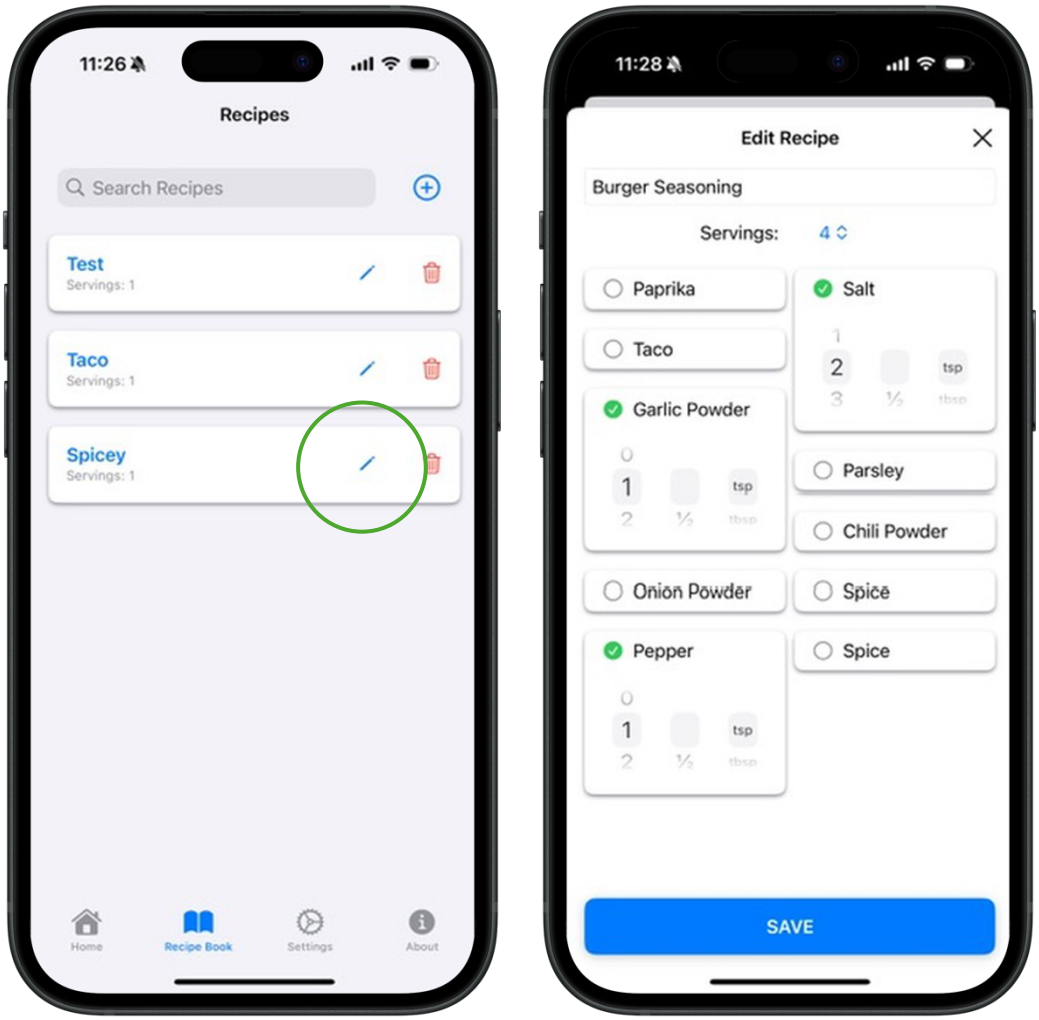
Creating a Recipe

1. Navigate to Recipe Book: Open the app and go to the **Recipe Book** tab.
2. Create Recipe: Tap on the + button to bring up the **Create Recipe** screen. On the **Create Recipe** screen, choose the spices you want to include in the recipe and specify their amounts. Use the **Servings Picker** to set the desired number of servings for your recipe. Enter the recipe name in the **Recipe Name** text field.
3. Save Changes: Once you've entered the recipe details, tap the **Save** button to store the information. If you do not want to save your recipe, either tap the **'X'** or swipe down on the screen. This will return you to the **Recipe Book** tab.



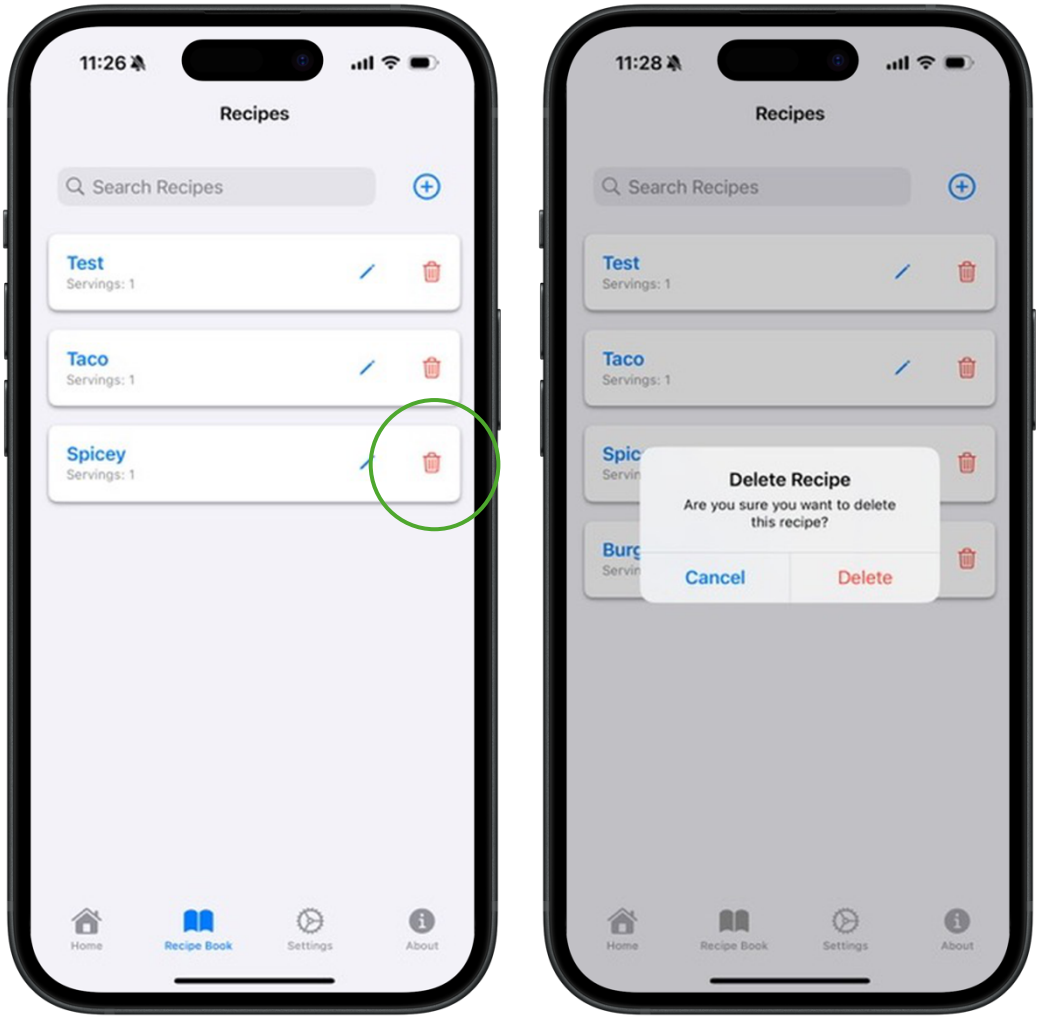
Editing a Recipe

1. Navigate to Recipe Book: Open the app and go to the **Recipe Book** tab.
2. Edit Recipe: Tap on the **pencil** to bring up the **Edit Recipe** screen. On the **Edit Recipe** screen, choose the spices you want to edit in the recipe and specify their amounts. Use the **Servings Picker** to edit the desired number of servings for your recipe. Edit the recipe name in the text field.
3. Save Changes: Once you've entered the recipe details, tap the **Save** button to store the information. If you do not want to save your recipe, either tap the **'X'** or swipe down on the screen. This will return you to the **Recipe Book** tab.



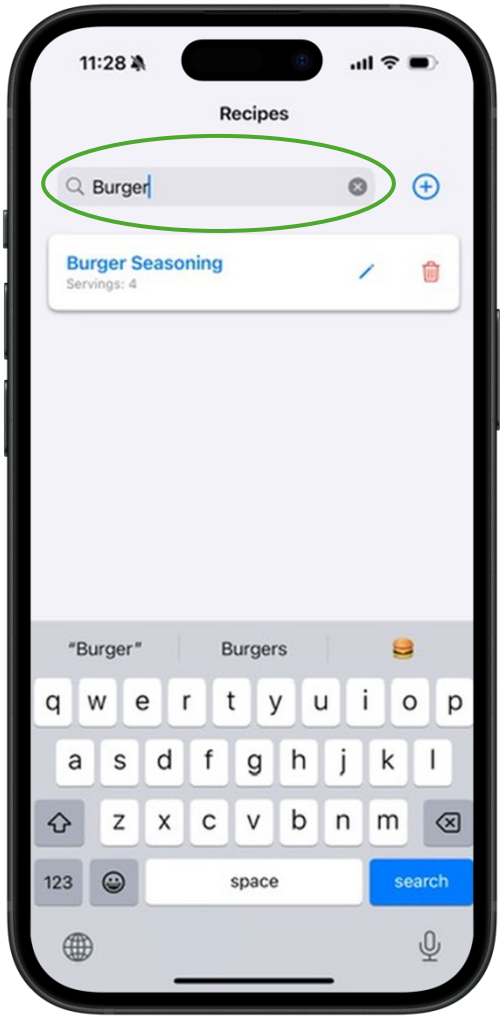
Deleting a Recipe

1. Navigate to Recipe Book: Open the app and go to the **Recipe Book** tab.
2. Delete Recipe: Tap on the **trash can** to bring up **Delete Recipe** Alert. Tap **Cancel** to cancel the deletion or tap **Delete** to confirm the deletion.



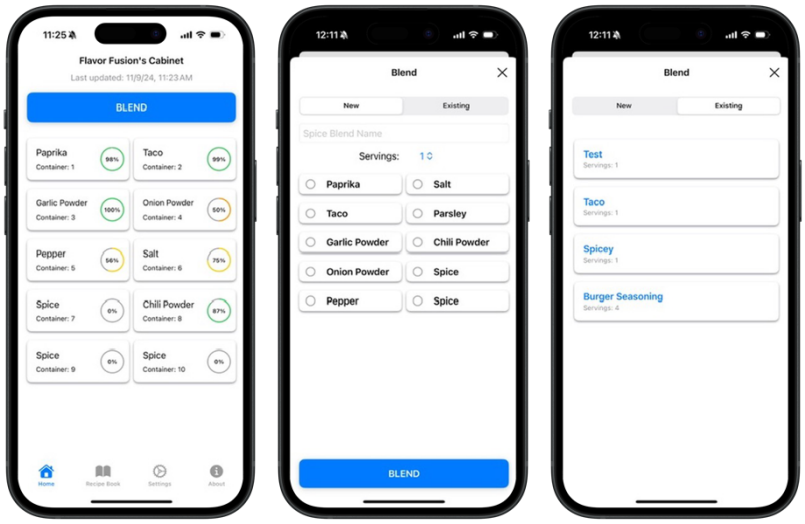
Searching for a Recipe

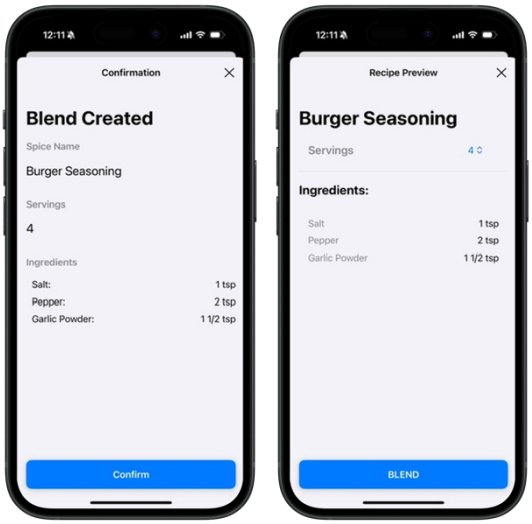
1. Navigate to Recipe Book: Open the app and go to the **Recipe Book** tab.
2. Search Recipe: At the top of the screen, tap the **Search Recipes** bar. Type in your search terms. You can search using exact names or partial keywords to find matching recipes.



Blending a Recipe

1. Navigate to Home: Open the app and go to the **Home** tab, at the top of the screen, tap the **Blend** button.
2. Make Selection: Select either the **New** or **Existing** tab on the **Blend** Screen. The **New** tab lets you to blend a recipe from scratch and the **Existing** tab lets you blend a recipe from the **Recipe Book**.
3. Blending a New Blend: Choose the spices you want to include in the blend and specify their amounts. Use the **Servings Picker** to set the desired number of servings for your blend. If you wish to save the blend to your **Recipe Book** enter the name in the **Blend Name** text field. Tap on the **Blend** button and then confirm.
4. Blending an Existing Blend: Tap the recipe that you would like to blend and then confirm.



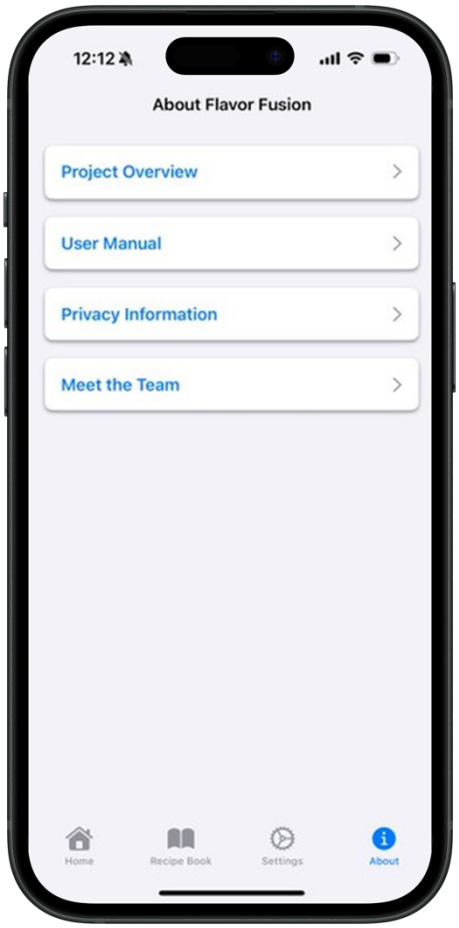


Settings

1. Navigate to Settings: Open the app and go to the **Settings**.
2. Bluetooth Status: Displays **Connected** when your iPhone is connected to the spice maker. Displays **Not Connected** when your iPhone is not connected to the spice maker.
3. Change Passcode: Type in your current passcode, your new passcode, and your new passcode again in the text fields. Tap the **Change Passcode** button to save the new passcode.
4. Change Display Name: Type in your desired display name in the text field. This name will be displayed on the **Home** screen. Tap **Change Display Name** to save the new name.
5. Require Passcode at Startup: Toggle **Require Passcode** if you would like to enter your passcode each time you open the app.

About

1. Navigate to About: Open the app and go to the **About**.
2. Project Overview: Tap **Project Overview** to view an overview of the app and spice maker. To navigate back to **About** tap the Back button in the top left corner of the screen.
3. User Manual: Tap **User Manual** to view the complete **User Manual** for both the app and the spice maker. To navigate back to **About** tap the Back button in the top left corner of the screen.
4. Privacy Information: Tap **Privacy Information** to view important privacy information for how the app utilizes Bluetooth to connect to your iPhone. To navigate back to **About** tap the Back button in the top left corner of the screen.
5. Meet the Team: Tap **Meet the Team** to view all the Flavor Fusion team members. Tap on each member's card to view a pop up with their contributions to the project. To close the pop up either tap the **'X'** or swipe down on the screen. To navigate back to **About** tap the Ba button in the top left corner of the screen.



Troubleshooting

1. Restart the App: Close the app fully by swiping up from the app switcher and reopening it. This often resolves temporary glitches.
2. Check for Updates: Ensure both the app and iOS are up-to-date. Updates often fix bugs and improve performance. Go to **Settings > General > Software Update** for iOS and **App Store > Updates** for the app.
3. Restart the iPhone: Power down your iPhone completely by holding the **Power** button (or **Power + Volume Down** on newer models) and turning it off. Restart and try using the app again.
4. Check Bluetooth Status: Go to **Settings > Bluetooth** and ensure Bluetooth is turned on.
5. Delete and Reinstall the App: Long-press the app icon and select **Delete App**, then reinstall it from the **App Store** (or Test Flight if using the Beta). This can resolve issues with corrupted files in the app.
6. Check App Permissions: Go to **Settings > Privacy** and ensure the app has the necessary permissions (e.g., access to the camera, microphone, location, or Bluetooth) if it relies on those features.
7. Contact App Support: If the above steps don’t work, reach out to the app’s support team.

Device User Manual

Tutorial for General Use

1. Carry the device using its side handles for safe transportation. Place on a secure, level surface.

2. Ensuring the power switch is turned off (“o” icon is depressed), plug the power cable into a standard American 3-prong power outlet (110-120 Volts, 60 Hz).

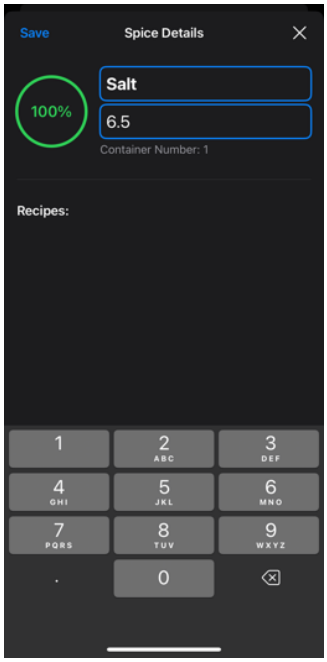
3. To fill or refill spice containers:

    a. begin by selecting the desired container according to its numbered label. Grab the top of the container and slide it out vertically from its compartment.

    b. Flip the container so its numbered label faces down before unscrewing the cap (counterclockwise rotation) and placing it aside.

4. Pour the desired spice into the container and screw the cap back on (clockwise rotation). Replace the container in the corresponding numbered compartment by sliding it down cap-first. The cap is designed to only fit in its compartment when it is aligned correctly, with the bearing facing the interior of the device.

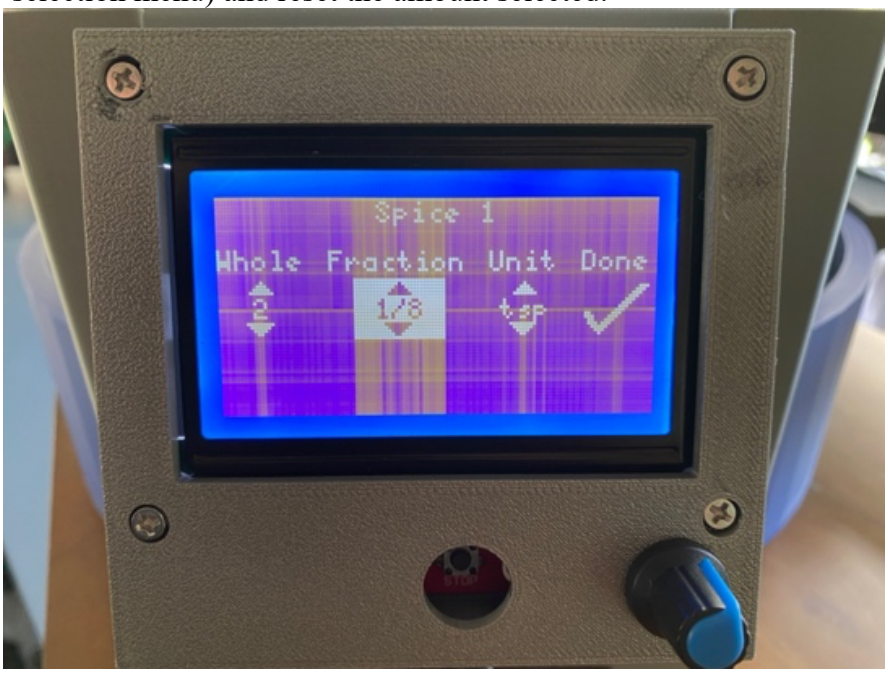
5. After refilling, note the level of spice (measured in ounces) and update its amount in the mobile app by selecting the corresponding container number. Spice names can also be customized according to container, so please ensure that each spice is properly named according to the number it is stored in.



6. Place an order using the mobile app or by navigating the LCD screen on the device. The LCD screens can be navigated by clicking down on the blue and black wheel to proceed, turning it to scroll, and clicking the small black button within the hole to the left of the wheel to go back.
    a. The setup screen will remain visible while the device waits for an order to be input. Press the wheel to proceed to the spice menu.

    b. The spice menu lists container numbers. Scroll to the desired spice by turning the wheel, then select it with a click. The LCD will continue to the amount selection menu.

c. The amount selection menu features different selection parameters. Begin by turning the wheel to select the desired whole number of ounces, teaspoons, or tablespoons. Click to select, then repeat for fractions of the same unit. For example, to input 3 ¼ tablespoons, first select “3” from the whole number amount, then “¼” as the fraction. After clicking on the desired fraction, a unit can be selected (oz for ounces, tsp for teaspoons, and Tbsp for tablespoons). Once the unit is selected, click the check mark to confirm. At any point in this process, the back button can be pressed to retreat to the previous selection (or to the spice selection menu) and reset the amount selected.



d. After selecting a spice amount, the screen transitions back to the spice selection menu. A check mark will indicate any previously selected spices. Continue selecting desired spice amounts until the order is input, then scroll to the top of the spice selection menu and choose “Confirm order”.
e. To send an order from the mobile app after making selections on the device screen, please press the back button repeatedly until the setup screen is reached and the LCD reads “Waiting for order from Flavor Fusion mobile app”.

7. After an order has been placed, the device will request for the cup to be placed into the drop zone until it is fully inserted and a click is audible.

8. The device will dispense the requested spices, displaying its current task as it proceeds. Each time a task begins, it also shows the current interior temperature of the device. If the

temperature becomes dangerously high (above 220°F), please turn off power to the device by flipping the power switch to off, or “o”.

9. After the order is complete, the LCD will provide a summary of the spices and amounts dispensed. The cursor can be scrolled by turning the wheel to navigate among the spices. The screen will return to the setup menu after 15 seconds of inactivity or by pressing either the wheel or the back button.

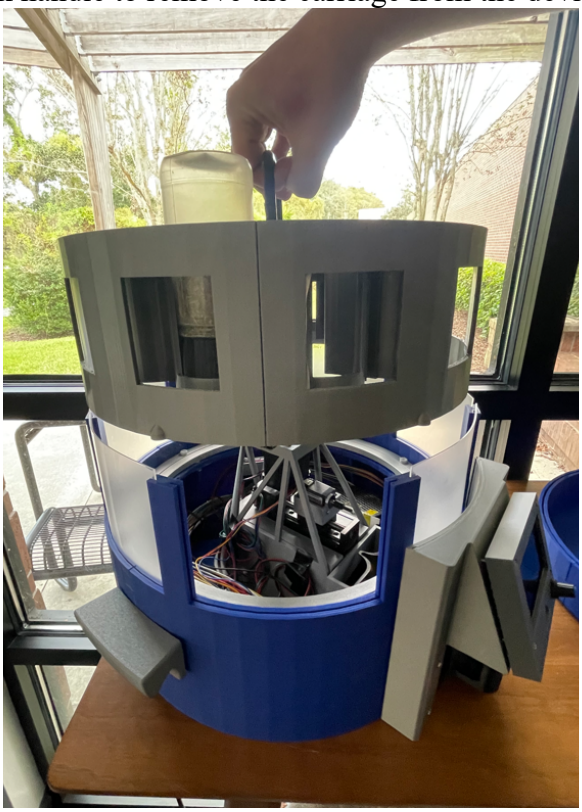
10. Remove the cup from the drop zone and pour the completed spice blend as desired. Until the cup is replaced and a click is audible, the mobile app will warn the user that the cup is still full.
11. Turn off the device by switching the power switch to off, or “o”.

Housing Troubleshooting

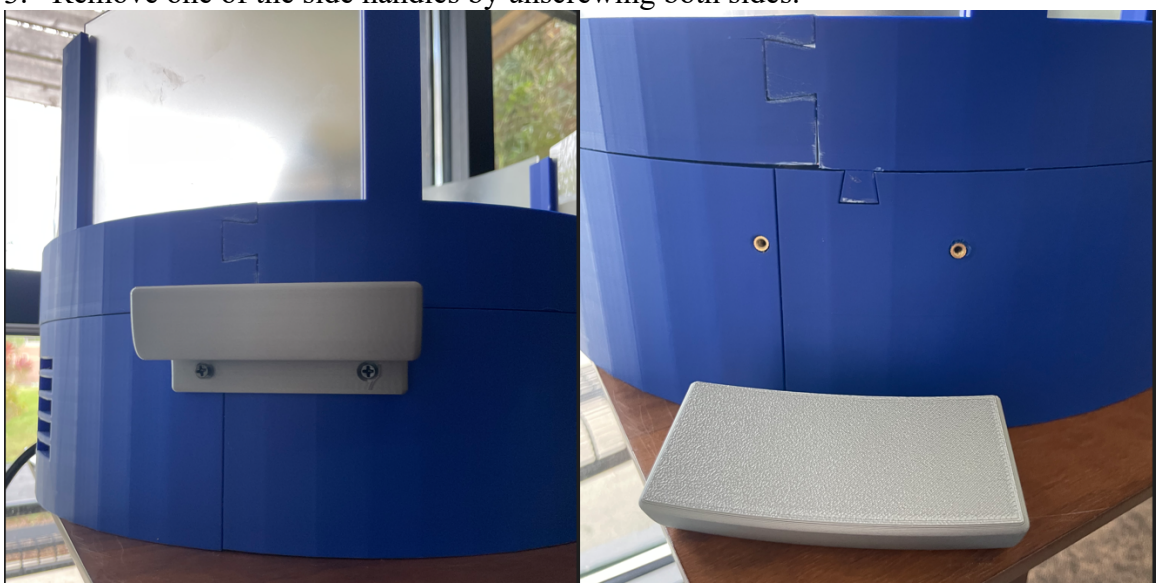
Housing Disassembly

1. Remove the top cover.

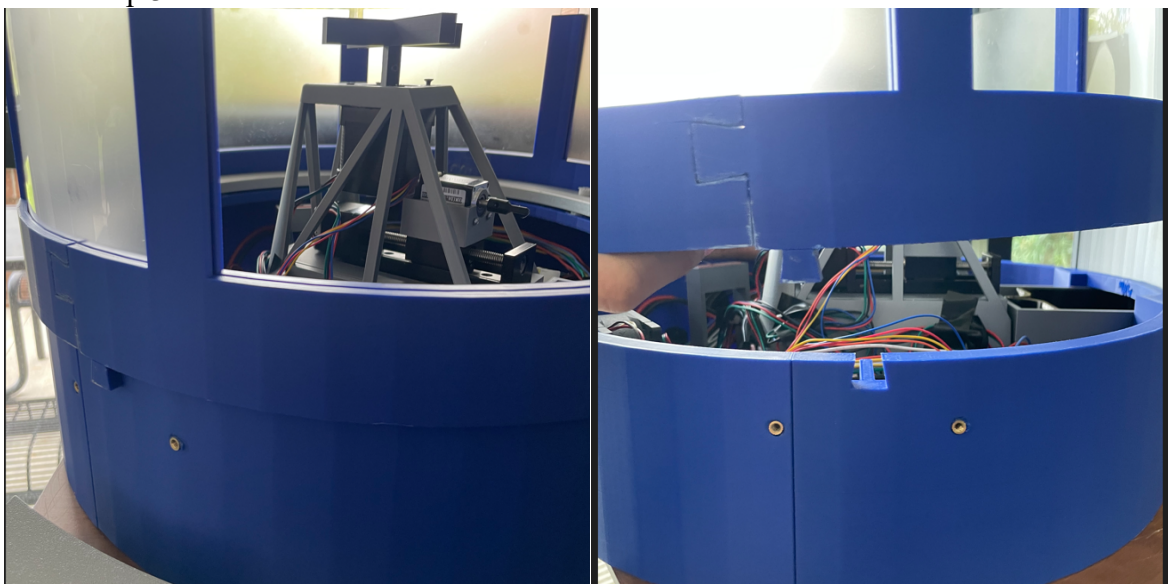
2. Lift using the black handle to remove the carriage from the device.



3. Remove one of the side handles by unscrewing both sides.



4. Remove the crown piece by sliding in the direction of the side handle that was removed in step 3.



To Remove Drop Zone Area

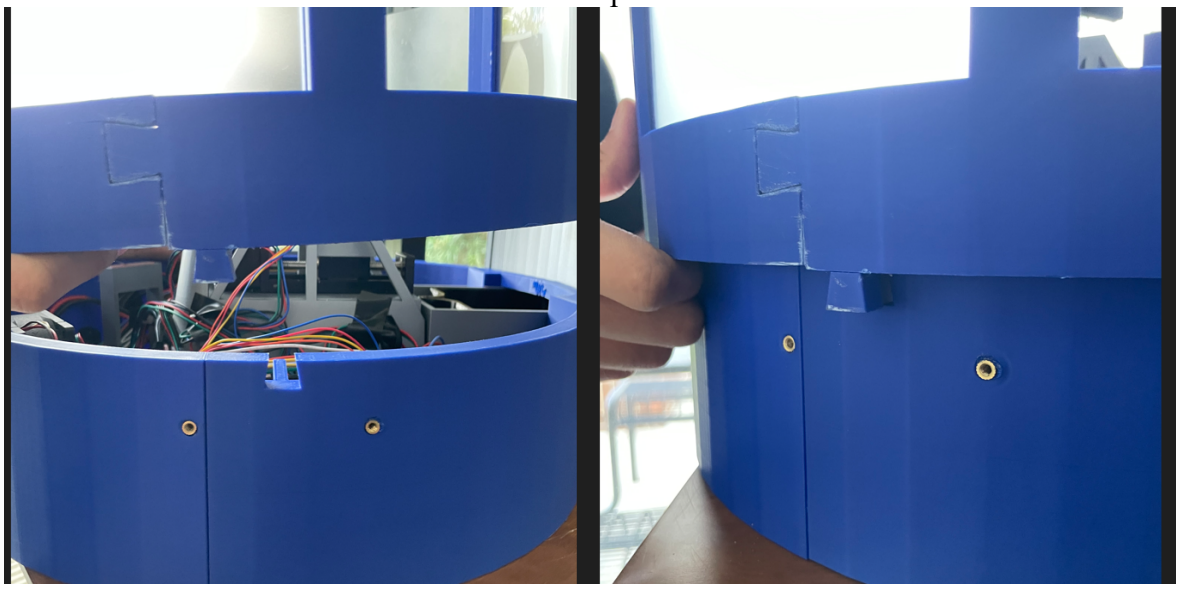
1. Remove the spice cup from the drop zone.

2. Remove the drop zone from the device by unscrewing the four screws along the front face.

3. Gently slide the drop zone from the device housing.

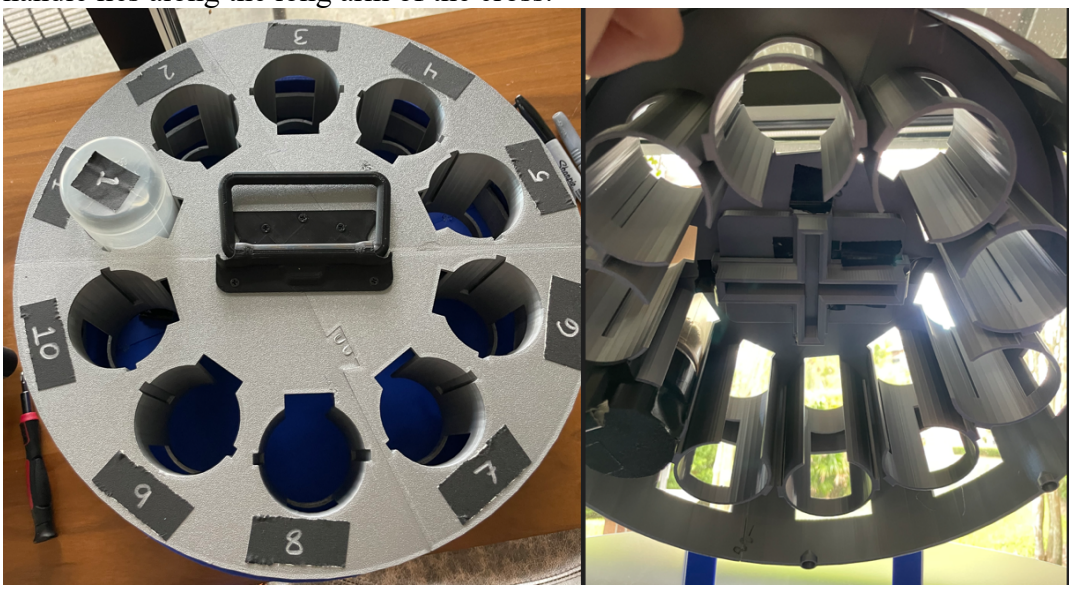
Housing Reassembly

1. Place the crown piece on the device housing and align the dovetails to their respective slots in the device. Slide the dovetails into place.



2. Screw in the side handle(s) to lock the crown piece in place.

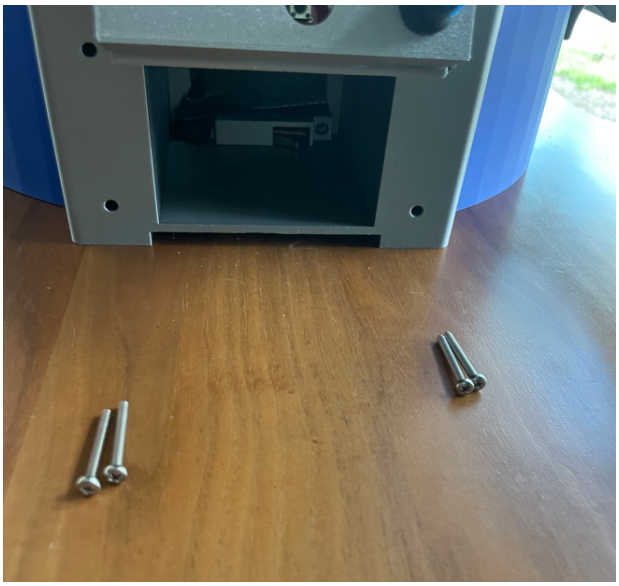
3. Lower the carriage into the device by grabbing it from the top handle. It must be aligned on the cross-shaped mount in the center of the device. To facilitate this alignment, the handle lies along the long arm of the cross.



4. Align Top lid to the sheet of acrylic with the cut outs on the top piece

To Reassemble Drop Zone

1. Carefully slide the drop zone area into the device. Make sure all four screw holes are aligned.
2. Tighten all four screws. Alternate between screws while tightening to ensure alignment is maintained.



3. Replace the cup in the drop zone holding area.

Electronics Troubleshooting

Processing and Motor Errors

In the case of processing errors- which may result in an unresponsive LCD screen, failed Bluetooth connection, or repeated data in the spice order:

1. The microcontroller should be reset by using the switch to turn off power.
2. If the linear rail motor was moving when power was cut, it should be reset by removing the lid and crown piece and manually turning the rail until it reaches its minimum position.
3. If the microcontroller continues to malfunction, the Arduino sketch needs to be reuploaded via USB connection.

Wiring

Disconnected or faulty wiring may result in parts or entire systems that malfunction or fail to even turn on. To double check wiring, consult the appendices for the TinkerCAD diagram, documentation of the RAMPS Shield for Arduino Mega, the HM-10 datasheet, and information on 3D printer electronics from RepRap. Note that alterations to the HM-10 will require perf board soldering. Cords from the power supply unit should be crimped and heat shrunk for protection.

LCD

1. If the rotary encoder wheel is unresponsive or feels stuck, gently pull up on the wheel or press on the plastic cover beneath the encoder to free it.
2. Although a button debouncing system is in place, pressing LCD buttons in a prolonged or inconsistent manner may result in multiple registrations of a single button press. Try to press buttons quickly and consistently to avoid this issue.
3. When scrolling, turn the rotary encoder wheel consistently for the best results. Sudden acceleration, deceleration, or jolty motions may cause jittery scrolling.

Calibration

1. The device carriage is calibration using a limit switch suspended in a tower behind the motors and by the power switch. Ensure the limit switch is properly inserted in this tower, with the switch arm facing up.
2. The limit switch is part of a shield board that contains an LED indicator. To verify the limit switch is properly functioning, it can be triggered while the device has power, and the LED may be observed to measure sensor status.

Bluetooth

Bluetooth errors occur when the mobile application and device microcontroller do not properly communicate. This will often result in the device failing to receive an app-requested order or order dispensing status not updating within the app.

1. Reset the mobile application by closing it, reopening, and reattempting an order.
2. Remove the lid and carriage from the device then inspect the HM-10 Bluetooth module, located in the front of the device between the microcontroller and the drop zone, to confirm a red light on the HM-10's surface. A blinking light indicates the module is disconnected while a solid light means it is paired. This pairing status should coincide with the Bluetooth status indicator in the Flavor Fusion app.

Fan

The fan should remain on, blowing air out of the device, whenever power is turned on. If this is not the case, the fan should be inspected.

1. The fan should be connected to a red wire and a separate black wire, leading to the corresponding colors in the fan’s female connector.
2. The black and red wire should connect to the proper 5V and ground connections on the Ramps board.

Temperature

Temperature readings are reported in Fahrenheit on the LCD during dispensing, since this is when motors contribute heat to the interior. If temperatures are not displayed, errors are displayed, or unlikely temperatures are displayed, then inspection is necessary.

1. Confirm the DHT11 sensor module is secured within the device interior. It is a black board with a blue cage-like attachment and 3 pins.
2. Verify that the pins are connected to the Ramps board in the proper order: signal, 5V, and ground.

Appendix D: Cost Analysis and Manufacturability Analysis

Manufacturing

Production Methods

To achieve a great prototype product for Flavor Fusion, the manufacturing methods and processes of different industries are investigated, providing insight into what to expect with existing products. Companies such as McCormick, KitchenArt, and similar spice dispensing products can help provide answers to many different questions, such as “What is food-safe manufacturing?" “How does the assembly process work?” and "How are materials transformed into spice containers?”. Even present-day innovations in food appliances have shaped the current lives of many people. For example, Keurig takes the tedious action of making coffee into a simple automated process for convenient personal use [75]. This is where manufacturing an automated spice device capable of performing necessary functions becomes key.

For large-scale industries with established demand and volume, it can be easy to identify which type of manufacturing process to use. For example, Keurig uses repetitive manufacturing to produce the same product repeatedly in an assembly line fashion, and there are “only a few changeovers needed as slight variations of the product are made over time.” [76]. This method would prove helpful for Keurig, as it produces coffee-making appliances in large quantities and the machines are assembled to FDA standards. For areas of food, McCormick and other food companies tend to rely on batch manufacturing [76]. A key difference is that “batch manufacturing might be left alone for periods of time and is only used when a new batch of products is required.” [77] Compared to repetitive manufacturing, it can still produce a product quickly but at a higher cost. Moving towards a narrow field of manufacturing, the job shop method is set up at a higher cost and lower output [77]. The benefit of Job Shop is how useful it is for startup companies to use, such as when developing a prototype and running it through tests [76]. With insights and implications for project decisions clearly defined, skilled workers can focus on the quality of the product and push to test their product without the risk of wasting lots of materials.

By addressing the type of manufacturing method to use, the manufacturing process also plays a significant role. The type of process for manufacturing a part can pose questions such as

“accessibility to machines," “object complexity," “number of parts," “time to manufacture," “material choice," and “cost with levels of automation." These unique processes of manufacturing include additive and subtractive manufacturing, molding, casting, forming, and joining. As each form of manufacturing has pros and cons, a few are shown in Figure 0.1 to stand out among the other processes [78].

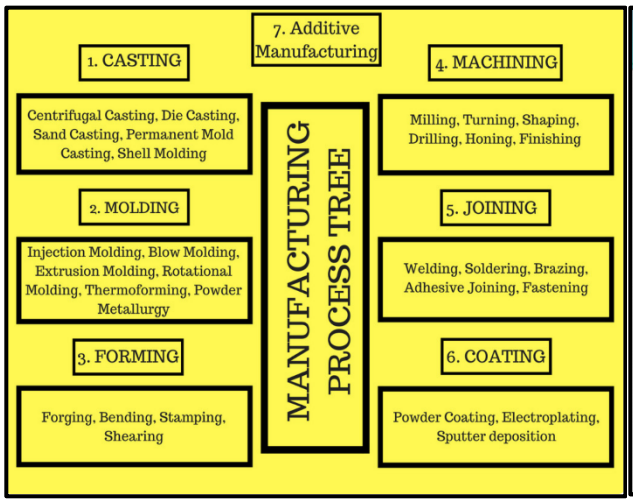
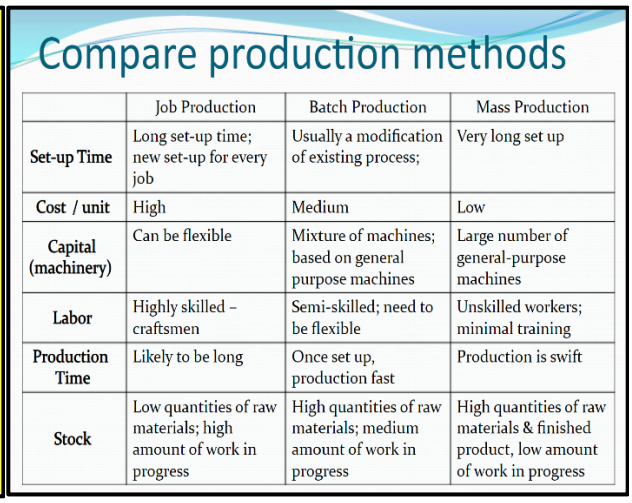


Figure 0.1 - Manufacturing Processes



	Job Production	Batch Production	Mass Production
Set-up Time	Long set-up time; new set-up for every job	Usually a modification of existing process;	Very long set up
Cost / unit	High	Medium	Low
Capital (machinery)	Can be flexible	Mixture of machines; based on general purpose machines	Large number of general-purpose machines
Labor	Highly skilled – craftsmen	Semi-skilled; need to be flexible	Unskilled workers; minimal training
Production Time	Likely to be long	Once set up, production fast	Production is swift
Stock	Low quantities of raw materials; high amount of work in progress	High quantities of raw materials; medium amount of work in progress	High quantities of raw materials & finished product, low amount of work in progress

Figure 0.2 - Production Methods

Manufacturing Processes

A common type of manufacturing process McCormick has for packaging their spices is blow molding. By using FDA-safe plastic, molten material is placed in a mold where pressurized air forces the material to conform to the bottle mold shape [78]. As this is useful for simple containers, Flavor Fusion's unique cartridge design will most likely have moving parts and components that will undergo revision as it progresses, thus making it costly to produce professional molds every time. However, for the product housing and simple nonmoving parts, the prototype could use injection molding, just like other cereal and dry goods dispensers. Comparing the two kinds of molding shows that, “Injection molding is used for complex solid components whereas Blow molding is used for thin-walled hollow parts” [79]. After an injection mold is completed, the part will also need to undergo a machining process. The machining process would entail proper-sized holes to be drilled and any subtractive manufacturing such as trimming edges if need be [79]. For each process the part goes through, a certain amount of time

and cost will also play a big role in the overall production, depending on what resources are available. Injection molding tends to be the quickest option in terms of producing mass quantities at a low cost; however, the initial set-up to produce the mold is costly [79].

Another popular method for prototypes is additive manufacturing. With additive manufacturing, a complex geometric part can be produced without the need for a mold, allowing better accessibility to machines. One downside is time consumption, as it is “adding layer-upon-layer of material, whether the material is plastic, metal or concrete” [80]. Understanding that time consumption is a risk of manufacturing, the part can still be of high quality at a low cost using todays technology [81]. Furthermore, being user-friendly for machinability, a team member can simply upload an STL file to the machine which makes it a simple task for team members with minimum experience with large complex machines. However, this method will also require a secondary process where the surface will have an additional protective coating to seal any pores. This surface-coat finish would help with durability and aesthetics while ensuring compliance with FDA guidelines [82]. Another idea to bridge the gap between manufacturing is to use additive manufacturing to create molds for injection molding [83].

Materials

Since there is nothing worse than clumped or stale-tasting spices, the need for a food-safe material capable of being cost-efficient, dishwasher-safe, and non-stick all while keeping an air-freshness seal are a couple of key points looked at when researching material. In addition, material selection will also become narrow, as the processes of manufacturing are limited when it comes to assembling a prototype product. Some consequences of not being able to achieve these goals could include the risk of accidental particles being consumed by humans, fire, higher manufacturing costs, and even damaging other appliances such as dishwashers when washing at high temperatures.

Food Safe vs. Food Grade Material

To get a basic idea of how some food appliances utilize unique manufacturing methods, the materials chosen by one spice dispenser mechanism, TasteTro [84], use food-safe and food-grade injection molding for their product. When comparing food-grade vs. food-safe materials as defined by the FDA, it states that “food grade materials are generally better for long-term

Table 12.1 - Additive Manufacturing Materials

Filament	Brand	FDA Status	Smoothable	Dishwasher Safe	Hot Liquids
ABS	Adwire PRO	Approved	Yes, acetone	Yes	Yes
ABS	Innofil3D	Approved except red, orange, and pink	Yes, acetone	Yes	Yes
Co-Polyester	Colofabb XT	Approved	No	Yes	Yes
HIPS	Easyfil	Compliant	Yes, d-limonene	Yes	No
HIPS	InnoFil3D	Approved	Yes, d-limonene	Yes	No
Nylon	Taulman Nylon 680	Compliant		No	No
PEI	ULTEM 1000	Compliant		Yes	Yes

storage. So, while food-safe plastic would make a good cutting board, it would not necessarily be the best choice as a container to store and reheat your leftovers in" [85]. Thus, further reading into the patent shows how they applied the material for the outside housing of the dispenser [85] and not just the main components that encounter food. As far as materials go for injection molds, similar plastic materials can also be utilized in additive manufacturing.

FDM 3D Printing

Moving into the contents of Additive manufacturing, it can be a major benefit to use them for their cost as well as adaptability for machine use. Breaking down each one, "For FDM 3D printing, there are several materials that are either FDA-certified or passable for food contact. These include PLA, PET, ULTEM, Polycarbonate, Polypropylene, PEEK, and natural grade Nylon" [86]. The significance of using FDM 3D printing would be to use as small batch prototypes that allow for easy modifications of design models for testing [81]. One material that stood out was based on information gathered from "okstate.edu" website (Table 12.1).

That material being ABS [87] in terms of strength, cost, and food grade approval helps in visualizing models and testing. One downside to using FDM printing is that there are slight limitations on printing pigmentation as well as the materials. As stated by prusa3D.com, "it is good to ask a manufacturer about additives used, or use natural filament with no added coloring" [88]. Thus, reducing the risk of causing harm to humans when using a product that touches food.

SLS and SLA Printing

However, there are other methods in additive manufacturing, such as SLS and SLA printing, which utilize resin in an open vat with a selective laser to solidify the resin [81]. As this could prove useful in designing injection molds, there would be a higher cost, and unfortunately, as stated by 3space.com, "SLS and SLA, will always produce parts that need extra post processing due to crevices, porosity, or powders affecting the safety of the printed part" [86]. This remains true for FMD printing as well, since there would be porous areas susceptible to bacterial growth from layered prints.

Besides resin requiring additional processing after initial manufacturing, another material examined, epoxy, can also provide food-grade results; however, due to its lack of temperature resistance, it would fail to qualify as dishwasher-safe [89]. Thus, if epoxy does not encounter high temperatures, and is kept in a stable environment below “120°F” [89]; it can reliably help cover porous surfaces and holes of the spice dispenser housing or non-dishwashing parts. In addition, as a surface coating, epoxy is also useful as an adhesive, combining parts together [90] that again do not require entering a dishwasher. Research has also shown that ceramics can be used in coating 3D printing materials; however, if the coating were placed on a food contact surface or even the housing, it would require a higher cost and manufacturability. This is due to specialized spray machines that would add to the difficulty of manufacturing an affordable prototype [82].

Overall, the selection of a material proves to be a key step, as without carefully examining different materials, there could not be an amazing product produced. By ensuring that basic requirements are covered, there is a chance to minimize future hazards and even ensure a fresh taste. Flavor Fusion will not only have a material selection based on its food grade, durability, cost, and temperature safety such as ABS or HIPS; consideration for FDA policy regarding food safety [91] also plays an important role in ensuring that rules and regulations are to be followed as well.

Safety

Food Safe Materials

Building any food-related device brings a new set of challenges to designers and the engineering behind these products. One of the major obstacles is making sure you follow food safety requirements set by your government. The biggest safety risks or concerns for food-related technology are choosing materials that will not harbor bacteria or promote the rapid growth of harmful bacteria. This will be tied into the material selection for a lot of different aspects of our project. We are not necessarily dealing with raw ingredients such as meats, eggs, etc. where a very clean environment is needed, but dealing with raw materials such as spices or powders still has risks with food handling and safety. Our dispensing process should allow the consumer to either hand wash or dish wash these components to remove leftover debris and or to sanitize after not using our product for a while. This is also a standard set by the FDA [92] saying that machinery designed for the food industry needs to be able to be clean and maintained, not allow foreign substances or objects in food, as well as parts of the machine that do not touch food to be cleaned. Another aspect of our project to keep secure is dealing with the electronics on board and managing their heat. This device will house motors, motor controllers, a microcontroller, etc., all generate heat. Keeping these considerations in mind narrows our scope of materials down to a few, for metals we can use Stainless steel, Titanium, Cast Iron, and Aluminum. As for plastics, we can use HDPE, the same plastic that is used in milk jugs, or PET/PETG which is a very common 3D printable plastic that is food-safe. These food-grade materials meet a lot of the rules and regulations set by many standards one of them is their ability to take strong detergents and not corrode or lose their material properties.

Table 12.2 - Additive Material Properties

Additive Materials	Deformation Temperature (C	Vapor Smoothir	FDA-Compliar	Food Safe	Dishwasher Safe (50-90°C)
Polylactic acid (PLA)	50-60	Depends	Yes	Yes	No
PLA+	60-70	Depends	Yes	Yes	No
Nylon 12 (PA 12)	60-70	No, Toxic Chemicals	Yes	Yes	Yes
Nylon 11 (PA 11)	60-75	No, Toxic Chemicals	Yes	Yes	Yes
PLA-CF	60-75	Depends	No	No	No
Co-Polyester	65-80	No	Yes	Yes	Yes
PET	70-80	Yes, Ethyl Acetate	Yes (certain colors)	Yes	No
PETG-CF	70-85	Yes, Ethyl Acetate	No	No	No
PLA-HT	70-90	Depends	Yes	Yes	Yes
High-Impact Polystyrene (HIPS)	70-95	Yes, Acetone/D-Limonene	Yes	Yes	Yes
(ABS)	80-100	Yes, Acetone	No	No	Yes
Polypropylene (PP)	80-110	No, Toxic Chemicals	Yes	Yes	Yes
Acrylonitrile Styrene Acrylate (ASA)	90-105	Yes, Acetone	Depends	Yes (only white colored)	Yes
Polycarbonate/acrylonitrile butadinene styrene (PC-ABS)	90-105	Yes, Acetone	Yes	Yes	Yes
Polyethylene Terephthalate glycol (PETG)	90-110	Yes, Ethyl Acetate	Yes	Yes	Yes
Polycarbonate	130-150	Yes, Acetone	Yes	Yes	Yes ("lasts 100
Polyetheretherketone (PEEK)	150-200	No	Yes	Yes	Yes
PEI	200-220	No	Yes	Yes	Yes

Food Safe Manufacturing

The primary source of manufacturing for Flavor Fusion will utilize Fused Deposition Modeling (FDM). Many of our primary sources of materials are food-graded plastics which will be 3D printed using this technique. To make sure that the parts do not get contaminated or to prevent long-term contamination during this process we will have to make sure that our manufacturing methods are clean and do not contaminate our plastic when being extruded. One way will be able to stop this from happening is changing the nozzle. Many if not all printers use a Brass nozzle, these nozzles contain lead within them. Lead is a material that is extremely toxic to humans and can leech into our plastic parts.

Food Safe Design

Food safety also is not always the material used in manufacturing the items as well it incorporates the manufacturing process and design of these products. In Food Safety Magazine

[93] they produced an article talking about the Hygienic Design of parts and machines so that they can be serviceable and clean as well as prevent gunk and buildup of food waste and matter in hard-to-reach places. These such places are crevices, and even the tops of screws that are exposed to these areas need to be evaluated in design decisions. Instead of having two pieces of metal welded together where it creates sharp angles where matter can accumulate, use bends to create a surface that is more accessible to be cleaned properly. However, it is not the end of the world if you create a design that cannot be cleaned effectively. The article also explained [93] “In some cases, crevices are unavoidable. This may be the case if slide bearings must be in contact with the product; for example, as bottom bearings of top-driven stirrers or as bearings in scraped-surface heat exchangers”. If you have a situation where this might be the case, you must come up with a detailed cleaning list that also might include partial or complete dismantling of the machine to clean everything effectively. Another design feature to look for is the roughness of the materials in use. This roughness level is measured by an average Ra value determined by an instrument called a profilometer measuring the peaks and valleys of the materials. One of the best materials that is used is stainless steel due to its low roughness level and high corrosive resistance. If you want to use other materials that might have a high roughness value, you can coat these materials in a special anti-microbial coating to decrease this value.

Heat Management

As for heat management, many consumer electronics have suffered explosions from not dissipating heat away from critical components, the major player being lithium-ion batteries. An article from Class Action [94] goes over many of the problems these technologies face and how these devices can explode. One of the major points is poorly made batteries; these cheaper batteries tend to short-circuit and cause a run-away event. During these events these batteries become a bomb and produce a violent fire, these events can happen quickly and during any state the battery is in, either being on or off. Another area to focus on with electronics is an automated shut-off device to improve the safety of our users as well as the safety of the product. Many products have these devices installed such as washers, dryers, blenders, etc. The way most of these devices shut off is through a switch built into the device. For example, dryers are equipped with a thermal overload switch that will trigger if the dryer gets too hot to prevent a fire. This can

be caused by not cleaning your lint trap clogging this vent. Another safety feature is in the door, all dryers have a sensor to detect if the door is closed. So, if the user opens the door, the dryer will turn off to prevent the user from harming themselves due to the rotating drum in the machine.

Device Security

Since Flavor Fusion has a mobile app development side to this project, we need to be able to secure this machine from unwanted inputs that are not the users. There are many ways to secure a physical electronic device such as a two-factor authentication like DUO Push or a custom QR code as to what Apple does with their Apple TVs when setting them up. One such example is a new emerging smart appliance designed by Samsung: their Smart Washer and Dryer. They allow the user connected to their home Wi-Fi to be able to turn on and off their washer and dryer as well as control different settings such as power usage through their app. Besides using Wi-Fi to talk to the device we can use Bluetooth, but Bluetooth also has some security faults with this system. Leaving the Bluetooth connection on can lead to an attack, some of these attacks are listed in an article written by WebRoot [95] and one of the biggest attacks our device could see is Denial of Service which is stated in the name by not allowing authorized users to use their product. The way to avoid these attacks is to turn off Bluetooth when not in use.

Long-Term Spice Storage

Knowing the proper way to store spices and how long certain spices and powders can improve the safety of the people eating and the taste of the food. According to Healthline, the average shelf life for spices is anywhere from 1-4 years depending on the type of spice/powder, moisture, temperature, and the material of the container [96]. Keeping the material in an airtight cool space is the most ideal environment for it. They state the pores in plastic will allow certain spices' flavor and color to leech into the container and cause issues in cleaning and reuse of the container. Stainless steel, glass, or ceramic jars do the best to keep these out since they do not contain these porous material traits. This can also tie into our heat management system by

controlling the amount of heat getting into the container area to prevent the temperature from rising near the spices in order to prevent spoiling and clumping of spices.

Appendix E: Expense Report

Item	Vendor	Quantity	Unit Price	Total
Lazy Susan	Amazon	1	$25.99	$25.99
TMC2209 Stepper Motor Driver	Amazon	1	$13.99	$13.99
HM-10 BLE Module	Allison	1	$10.99	$10.99
OSOYOO 3D Printer Controller Kit Ramps + Smart Display Controller + Mega2560 + A4988 Stepper Motor Driver	Amazon	1	$47.99	$47.99
Inc PS613163 Power Tool Cord	Amazon	1	$8.97	$8.97
Load cell + HX711	Sam	1	$0.00	$0.00
Shift registers (8-bit)	Sam	4	$0.00	$0.00
SOMELINE® Crimp Tool Connector Kit + 21 Types of Connectors	Amazon	1	$28.99	$28.99
Spice Bottles	Aaron Packaging	1	$12.99	$12.99
SHKI 608 2RS Ball Bearings	Amazon	2	$4.99	$9.98

uxcell PTFE Flat Washers	Amazon	1	$8.19	$8.19
uxcell PTFE Flat Washer	Amazon	1	$7.99	$7.99
Restrictive fasteners			$0.00	$0.00
Square shaft for carriage and NEMA 17			$0.00	$0.00
O-rings (sized for funnels)			$0.00	$0.00
Certified Food Grade PETG 3D Printer Filament	Amazon	2	$34.99	$69.98
Linear Stage Actuator + NEMA 11 Stepper Motor	Amazon	1	$62	$62.00
M3 Flat Head Machine Screws and Hex Nuts Kit	Amazon	1	$7.58	$7.58
Sorbothane Vibration Isolation Circular Disc	Amazon	1	$15.95	$15.95
Apple Developer License	Allison	1	$99.00	$99.00
Total				**$427.58**

Appendix F: List of Manuals and Other Documents

Swift Code

Bluetooth Manager Class

```
        import Foundation
        import CoreBluetooth

        // BLEManager class responsible for managing Bluetooth Low Energy (BLE) connections
        class BLEManager: NSObject, ObservableObject, CBCentralManagerDelegate,
CBPeripheralDelegate {
            @Published var isBluetoothConnected: Bool = false // Track Bluetooth connection
status
            @Published var isDataRetrievedViaBluetooth: Bool = false // Track if data is
retrieved via Bluetooth
            @Published var isOrderMixed: Bool = false   // Track whether the spice blend is
completed
            @Published var isTrayEmpty: Bool = true    // Track whether the tray is empty

            var centralManager: CBCentralManager! // Bluetooth central manager to handle BLE
tasks
            var connectedPeripheral: CBPeripheral? // Currently connected peripheral
            var dataCharacteristic: CBCharacteristic? // Characteristic to communicate with

            var spiceDataViewModel: SpiceDataViewModel

            let spiceServiceUUID = CBUUID(string: "FFE0") // UUID for the service to interact
with
            let dataCharacteristicUUID = CBUUID(string: "FFE1") // UUID for the data
characteristic
            let targetPeripheralName = "HMSoft"  // Peripheral name

            // Initialize BLEManager with a spice data ViewModel
            init(spiceDataViewModel: SpiceDataViewModel) {
                self.spiceDataViewModel = spiceDataViewModel
                super.init()
                centralManager = CBCentralManager(delegate: self, queue: nil)
                print("Central Manager initialized.")
            }

            // MARK: - CBCentralManagerDelegate Methods

            // Called when the central manager's state changes (e.g., Bluetooth power status)
            func centralManagerDidUpdateState(_ central: CBCentralManager) {
                print("Central Manager did update state: \(central.state.rawValue)")

                if central.state == .poweredOn {
                    print("Bluetooth is powered on.")
                    // Start scanning for peripherals if Bluetooth is on
                    centralManager.scanForPeripherals(withServices: nil, options: nil)
                } else {
                    isBluetoothConnected = false // Reset connection status if Bluetooth is
unavailable
                }
            }

            // Called when a peripheral is discovered during scanning
            func centralManager(_ central: CBCentralManager, didDiscover peripheral:
CBPeripheral, advertisementData: [String: Any], rssi RSSI: NSNumber) {
                // Check if discovered peripheral matches the target name
                if let peripheralName = peripheral.name, peripheralName == targetPeripheralName {
                    connectedPeripheral = peripheral
                    centralManager.stopScan()  // Stop scanning once the peripheral is found
```

```
                centralManager.connect(peripheral, options: nil)  // Attempt to connect to
peripheral
            } else {
                print("Discovered peripheral: \(peripheral.name ?? "Unknown") does not match
the target.")
            }
        }

        // Called upon successful connection to the peripheral
        func centralManager(_ central: CBCentralManager, didConnect peripheral: CBPeripheral)
{
            // Update connection status only for the target peripheral
            if peripheral.name == targetPeripheralName {
                print("Connected to target peripheral: \(peripheral.name ?? "Unknown")")
                isBluetoothConnected = true // Set as connected
                peripheral.delegate = self
                peripheral.discoverServices([spiceServiceUUID]) // Discover services
            } else {
                isBluetoothConnected = false // Mark as disconnected for other devices
            }
        }

        // Called when disconnected from the peripheral
        func centralManager(_ central: CBCentralManager, didDisconnectPeripheral peripheral:
CBPeripheral, error: Error?) {
            print("Disconnected from peripheral: \(peripheral.name ?? "Unknown").")
            isBluetoothConnected = false // Reset connection status

            // Attempt to reconnect by resuming scanning after a delay
            DispatchQueue.main.asyncAfter(deadline: .now() + 1.0) {
                print("Reconnecting... Restarting scan for peripherals.")
                self.centralManager.scanForPeripherals(withServices: nil, options: nil)
            }
        }

        // MARK: - CBPeripheralDelegate Methods

        // Called when services are discovered on the peripheral
        func peripheral(_ peripheral: CBPeripheral, didDiscoverServices error: Error?) {
            if let error = error {
                print("Error discovering services: \(error.localizedDescription)")
                return
            }

            print("Discovered services for peripheral: \(peripheral.name ?? "Unknown").")
            if let services = peripheral.services {
                for service in services {
                    print("Service UUID: \(service.uuid)")
                    if service.uuid == spiceServiceUUID {
                        print("Found spice service. Discovering characteristics...")
                        peripheral.discoverCharacteristics([dataCharacteristicUUID], for:
service)
                    }
                }
            }
        }

        // Called when characteristics are discovered for a service on the peripheral
        func peripheral(_ peripheral: CBPeripheral, didDiscoverCharacteristicsFor service:
CBService, error: Error?) {
            if let error = error {
                print("Error discovering characteristics: \(error.localizedDescription)")
                return
            }

            print("Discovered characteristics for service: \(service.uuid).")
            if let characteristics = service.characteristics {
                for characteristic in characteristics {
                    print("Characteristic UUID: \(characteristic.uuid)")
```

```
                    if characteristic.uuid == dataCharacteristicUUID {
                        print("Enabling notifications for characteristic:
\(characteristic.uuid)")
                        peripheral.setNotifyValue(true, for: characteristic) // Enable
notifications
                        self.dataCharacteristic = characteristic // Store reference to
characteristic
                    }
                }
            }
        }

        // Called when data is received from the peripheral
        func peripheral(_ peripheral: CBPeripheral, didUpdateValueFor characteristic:
CBCharacteristic, error: Error?) {
            if let error = error {
                print("Error updating value for characteristic:
\(error.localizedDescription)")
                return
            }

            if let data = characteristic.value {
                isDataRetrievedViaBluetooth = true
                // Process data on a background thread to avoid UI lag
                DispatchQueue.global(qos: .userInitiated).async { [weak self] in
                    self?.processData(characteristic: characteristic, data: data)
                }
            }
        }

        // MARK: - Data Processing Methods

        // Process the received data based on characteristic UUID
        private func processData(characteristic: CBCharacteristic, data: Data) {
            if characteristic.uuid == dataCharacteristicUUID {
                if let message = String(data: data, encoding: .utf8) {
                    print("Received message: \(message)")
                    // Parse and handle the message received
                    parseMessage(message)
                } else {
                    print("Failed to decode data as UTF-8 string.")
                }
            }
        }

        // Parse the received message and update relevant state variables
        private func parseMessage(_ message: String) {
            // Remove trailing characters after "#" if present
            let cleanMessage = message.components(separatedBy: "#").first ?? message

            // Example messages: "ORDER_MIXED:1", "TRAY_EMPTY:0"
            if cleanMessage.hasPrefix("ORDER_MIXED:") {
                let value = cleanMessage.replacingOccurrences(of: "ORDER_MIXED:", with:
"").trimmingCharacters(in: .whitespacesAndNewlines)
                DispatchQueue.main.async { [weak self] in
                    self?.isOrderMixed = (value == "1") // Update blend status based on value
                    print("Order mixed status updated: \((value == "1"))")
                }
            } else if cleanMessage.hasPrefix("TRAY_EMPTY:") {
                let value = cleanMessage.replacingOccurrences(of: "TRAY_EMPTY:", with:
"").trimmingCharacters(in: .whitespacesAndNewlines)
                DispatchQueue.main.async { [weak self] in
                    self?.isTrayEmpty = (value == "1") // Update tray empty status based on
value
                    print("Tray empty status updated: \((value == "1"))")
                }
            } else {
                print("Unknown message received: \(cleanMessage)")
            }
```

```
    }

    // MARK: - Data Sending Method

    // Send spice data to the peripheral in chunks
    func sendSpiceDataToPeripheral(data: Data) {
        isOrderMixed = false // Reset order mixed status before sending data
        guard let peripheral = connectedPeripheral else {
            print("No connected peripheral to send data to.")
            return
        }

        guard let dataCharacteristic = self.dataCharacteristic else {
            print("Data characteristic not found.")
            return
        }

        let chunkSize = 20  // BLE characteristic size limit (20 bytes)
        var offset = 0

        while offset < data.count {
            let chunkLength = min(chunkSize, data.count - offset)
            let chunk = data.subdata(in: offset..<offset + chunkLength)

            // Send this chunk over Bluetooth without expecting a response
            peripheral.writeValue(chunk, for: dataCharacteristic, type: .withoutResponse)
            print("Sending chunk: \(String(data: chunk, encoding: .utf8) ?? "Error")")

            offset += chunkLength
            // Sleep briefly to prevent overwhelming the peripheral
            usleep(10000) // Sleep for 10ms
        }

        print("All data has been sent.")
    }
}
```

Spice Data

```
import Foundation
import Combine
import CloudKit

// MARK: - Spice Data Model

/// Represents a spice in the Flavor Fusion app with details such as name, amount, unit, and container number.
public struct Spice: Identifiable, Equatable, Hashable, Encodable, Decodable {
    public var id: UUID
    public var name: String
    public var spiceAmount: Double
    public var isSelected: Bool
    public var containerNumber: Int
    public var selectedAmount: Double
    public var unit: String

    /// Initializes a new spice with default values
    public init(name: String = "Spice", spiceAmount: Double = 0.0, unit: String = "oz",
containerNumber: Int) {
        self.id = UUID()
        self.name = name
        self.spiceAmount = (spiceAmount * 100).rounded() / 100
        self.isSelected = false
        self.containerNumber = containerNumber
        self.selectedAmount = 0.0
        self.unit = unit
    }
```

```
        /// Updates the spice amount and unit
        public mutating func updateSpiceAmount(amount: Double, unit: String) {
            self.spiceAmount = (amount * 100).rounded() / 100
            self.unit = unit
        }
    }

    // MARK: - Spice Data ViewModel

    /// ViewModel class for managing spice data in Flavor Fusion app, with support for iCloud and UserDefaults storage.
    public class SpiceDataViewModel: ObservableObject {
        @Published public var spices: [Spice] = []
        private let iCloudStore = NSUbiquitousKeyValueStore.default
        private let userDefaultsKey = "savedSpices"

        /// Initializes the ViewModel, loads saved spices, and observes iCloud changes
        public init() {
            self.spices = []
            self.loadSpices()
            NotificationCenter.default.addObserver(self, selector:
#selector(icloudStoreDidChange), name: NSUbiquitousKeyValueStore.didChangeExternallyNotification,
object: iCloudStore)
        }

        deinit {
            NotificationCenter.default.removeObserver(self)
        }

        /// Handles changes from iCloud by reloading spices
        @objc private func icloudStoreDidChange(notification: NSNotification) {
            self.loadSpicesFromiCloud()
        }

        /// Saves the spices to both UserDefaults and iCloud
        public func saveSpices() {
            if let encoded = try? JSONEncoder().encode(spices) {
                UserDefaults.standard.set(encoded, forKey: userDefaultsKey)
                print("Spices saved successfully to UserDefaults.")

                iCloudStore.set(encoded, forKey: userDefaultsKey)
                iCloudStore.synchronize()
                print("Spices saved successfully to iCloud.")
            } else {
                print("Failed to encode and save spices.")
            }
        }

        /// Loads spices from UserDefaults or iCloud, with UserDefaults as the primary source
        public func loadSpices() {
            if let savedSpices = UserDefaults.standard.data(forKey: userDefaultsKey),
               let decodedSpices = try? JSONDecoder().decode([Spice].self, from: savedSpices)
{
                self.spices = decodedSpices
                spiceData = decodedSpices
            } else {
                print("No saved spices found in UserDefaults. Attempting to load from
iCloud.")
                loadSpicesFromiCloud()
            }
        }

        /// Loads spices from iCloud if they are not found in UserDefaults
        public func loadSpicesFromiCloud() {
            if let savedSpicesData = iCloudStore.data(forKey: userDefaultsKey),
               let decodedSpices = try? JSONDecoder().decode([Spice].self, from:
savedSpicesData) {
                self.spices = decodedSpices
                spiceData = decodedSpices
```

```
                    print("Spices loaded from iCloud: \(spices)")

                    // Save to UserDefaults for local persistence
                    UserDefaults.standard.set(savedSpicesData, forKey: userDefaultsKey)
                } else {
                    print("No saved spices found in iCloud.")
                }
            }

            /// Updates the spice name by container number and saves the change
            public func updateSpiceName(containerNumber: Int, newName: String) {
                if let index = spices.firstIndex(where: { $0.containerNumber == containerNumber })
{
                    spices[index].name = newName
                    spiceData[index].name = newName
                    saveSpices()
                    print("Updated spice name to \(newName) for containerNumber
\(containerNumber).")
                }
            }

            /// Updates spice amount and unit by container number, or adds a new spice if none exists
            public func updateSpice(containerNumber: Int, newAmount: Double, newUnit: String) {
                if let index = spices.firstIndex(where: { $0.containerNumber == containerNumber })
{
                    spices[index].updateSpiceAmount(amount: newAmount, unit: newUnit)
                    spiceData[index] = spices[index]
                } else {
                    let newSpice = Spice(containerNumber: containerNumber)
                    spices.append(newSpice)
                    spiceData.append(newSpice)
                    print("Added new spice for container number \(containerNumber) with amount
\(newSpice.spiceAmount) \(newSpice.unit)")
                }

                // Sort spices and save changes
                spices.sort { $0.containerNumber < $1.containerNumber }
                spiceData.sort { $0.containerNumber < $1.containerNumber }
                saveSpices()
            }

            /// Updates spice amount
            public func updateSpiceAmountInOunces(containerNumber: Int, newAmountInOunces:
Double) {
                updateSpice(containerNumber: containerNumber, newAmount: newAmountInOunces,
newUnit: "oz")
            }

            /// Updates multiple spices
            public func updateAllSpices(newSpiceData: [Spice]) {
                for newSpice in newSpiceData {
                    updateSpice(containerNumber: newSpice.containerNumber, newAmount:
newSpice.spiceAmount, newUnit: newSpice.unit)
                }
            }
        }

        // Global array for spices, used throughout the app
        public var spiceData: [Spice] = []
```

Recipe Data

```
        import Foundation

        // MARK: - Ingredient Model
```

```
/// Represents an individual ingredient in a recipe with details such as name, amount, unit, and container number.
struct Ingredient: Codable {
    var name: String         // The name of the ingredient
    var amount: Double       // Quantity of the ingredient
    var unit: String         // Teaspoon or Tablespoon
    var containerNumber: Int // Identifies the container used for this ingredient
}

// MARK: - Recipe Model

/// Represents a recipe, including a unique identifier, name, list of ingredients, and serving size.
struct Recipe: Identifiable, Codable {
    var id: UUID                  // Unique identifier for the recipe
    var name: String              // Name of the recipe
    var ingredients: [Ingredient] // List of ingredients required for the recipe
    var servings: Int             // Number of servings the recipe yields

    /// Initializes a new recipe with a given name, ingredients, and servings
    init(name: String, ingredients: [Ingredient], servings: Int) {
        self.id = UUID()
        self.name = name
        self.ingredients = ingredients
        self.servings = servings
    }
}
```

Recipe Store

```
import Foundation
import CloudKit

/// A class that manages a collection of recipes.
class RecipeStore: ObservableObject {
    @Published var recipes: [Recipe] = []
    private let iCloudStore = NSUbiquitousKeyValueStore.default
    private let userDefaultsKey = "recipes"

    init() {
        self.loadRecipes()
        NotificationCenter.default.addObserver(self, selector:
#selector(icloudStoreDidChange), name: NSUbiquitousKeyValueStore.didChangeExternallyNotification,
object: iCloudStore)
    }

    deinit {
        NotificationCenter.default.removeObserver(self)
    }

    @objc private func icloudStoreDidChange(notification: NSNotification) {
        self.loadRecipesFromiCloud()
    }

    func addRecipe(_ recipe: Recipe) {
        recipes.append(recipe)
        saveRecipes()
    }

    func removeRecipe(_ recipe: Recipe) {
        if let index = recipes.firstIndex(where: { $0.id == recipe.id }) {
            recipes.remove(at: index)
            saveRecipes()
        }
    }

    /// Updates an existing recipe in the store
    func updateRecipe(_ updatedRecipe: Recipe) {
        if let index = recipes.firstIndex(where: { $0.id == updatedRecipe.id }) {
            recipes[index] = updatedRecipe
```

```
                saveRecipes() // Persist the changes
            }
        }

        // Save recipes to both UserDefaults and iCloud
        private func saveRecipes() {
            let encoder = JSONEncoder()
            if let encoded = try? encoder.encode(recipes) {
                // Save to UserDefaults
                UserDefaults.standard.set(encoded, forKey: userDefaultsKey)
                print("Recipes saved successfully to UserDefaults.")

                // Save to iCloud
                iCloudStore.set(encoded, forKey: userDefaultsKey)
                iCloudStore.synchronize()
                print("Recipes saved successfully to iCloud.")
            } else {
                print("Failed to encode and save recipes.")
            }
        }

        // Load recipes from UserDefaults first, then iCloud if not available
        private func loadRecipes() {
            if let data = UserDefaults.standard.data(forKey: userDefaultsKey) {
                let decoder = JSONDecoder()
                if let decoded = try? decoder.decode([Recipe].self, from: data) {
                    recipes = decoded
                    print("Recipes loaded from UserDefaults: \(recipes)")
                    return
                }
            }
            print("No saved recipes found in UserDefaults. Attempting to load from iCloud.")
            loadRecipesFromiCloud()
        }

        // Load recipes from iCloud
        private func loadRecipesFromiCloud() {
            if let data = iCloudStore.data(forKey: userDefaultsKey),
               let decoded = try? JSONDecoder().decode([Recipe].self, from: data) {
                recipes = decoded
                print("Recipes loaded from iCloud: \(recipes)")
                // Save to UserDefaults for local persistence
                UserDefaults.standard.set(data, forKey: userDefaultsKey)
            } else {
                print("No saved recipes found in iCloud.")
            }
        }
    }
```

Arduino Code

Main Loop

```
    void loop() {
      // Check if data is available from HM-10 (BLE central device)
      if (currentMenu == 0 && Serial1.available()) {
        while (Serial1.available()) {
          char c = Serial1.read();
          incomingString += c;

          // Check if data contains the end marker "#END"
          if (incomingString.indexOf("#END") != -1) {
            // Remove the end marker and trim the data
            incomingString.replace("#END", "");
```

```
      incomingString.trim();

      processReceivedIngredients(incomingString); // Process the complete command
      incomingString = ""; // Clear the buffer
    }
  }
}

// Update current LCD menu
if(menu_redraw_required){
  u8g.firstPage();
  do{
    if(currentMenu == 0){
      // setup menu
      drawSetup();
    }else if(currentMenu == 1){
      // spice selection menu
      drawSpiceMenu();
    }else if(currentMenu == 2){
      // amount menu
      drawAmountMenu();
    }else{
      // summary menu
      drawSummary();
    }
  } while(u8g.nextPage());

  menu_redraw_required = 0;
}

// Check for user inputs
updateLCD();

// Reset to setup menu if summary has been up
if(lastSummaryTime != 0){
  currentTime = millis();

  // If 15 seconds have passed:
  if(currentTime - lastSummaryTime >= interval){
    currentMenu = 0;
    menu_redraw_required = 1;
    lastSummaryTime = 0;
  }
}

// check for drop zone cup presence before moving motors
int reading = senseDropZone();

// Process spice data if the order has been received and not mixed
if (!isOrderMixed) {

  // ensure limit switch is LOW
  if(!reading){

    // calibrate carriage position
    calibrate();

    // Loop through all requested spices
    for (int j = 0; j < numSpicesOrdered; j++) {
      // Check if the spice amount is valid before proceeding
      if (spiceArray[j][1].toFloat() > 0) {
        // Move susan to the requested spice
        moveSusan(j);

        // Move rail forward
        moveRailForward();

        // Move auger for requested amount
        moveAuger(j);
```

```
            // Move rail back
            moveRailBackward();
          } else {

            u8g.firstPage();
            do{
              u8g.drawStr(0, 0, "Invalid spice amount");
              int h = u8g.getFontAscent() - u8g.getFontDescent();
              u8g.drawStr(0, h+1, "Skipping this spice");
            } while(u8g.nextPage());

          }
        }

        isOrderMixed = true; // Mark the order as complete

        // Send "ORDER_MIXED:1" to Bluetooth app to indicate the order is mixed
        sendOrderMixedStatus();
        sendTrayStatus(false);
      }else if(reading){
        // warn user to reload cup
        u8g.firstPage();
            do{
              u8g.drawStr(0, 0, "Please insert cup");
            } while(u8g.nextPage());
      }
    }

    // Small delay to prevent overwhelming the BLE connection
    if(currentMenu == 0){
      delay(100);
    }
  }
```

Arduino Process Ingredients

```
void processReceivedIngredients(String data) {
  int start = 0;
  int separatorIndex = data.indexOf(';', start);
  numSpicesOrdered = 0;

  while (separatorIndex != -1) {
    String ingredientPair = data.substring(start, separatorIndex);
    if (ingredientPair.length() > 0) {
      int colonIndex = ingredientPair.indexOf(':');
      if (colonIndex != -1) {
        String ingredientID = ingredientPair.substring(0, colonIndex);
        String ingredientAmount = ingredientPair.substring(colonIndex + 1);

        // Ensure that the amount is properly parsed and valid
        float amount = ingredientAmount.toFloat();
        if (amount > 0) {
          spiceArray[numSpicesOrdered][0] = ingredientID;
          spiceArray[numSpicesOrdered][1] = ingredientAmount;
          numSpicesOrdered++;

        } else {
          u8g.firstPage();
          do{
            u8g.drawStr(0, 0, "Invalid spice amount");
            int h = u8g.getFontAscent() - u8g.getFontDescent();
            u8g.drawStr(0, h+1, "Skipping this spice");
          } while(u8g.nextPage());
        }
      }
```

```
      }
      start = separatorIndex + 1;
      separatorIndex = data.indexOf(';', start);
    }

    // Bubble sort spice array to ascending container number
    for (uint8_t c = 0; c < numSpicesOrdered - 1; c++) { // loop thru container numbers
      for (uint8_t d = 0; d < numSpicesOrdered - c - 1; d++) { // loop thru successive
containers
        if (spiceArray[d][0].toInt() > spiceArray[d + 1][0].toInt()) { // compare
          for (uint8_t e = 0; e < 2; e++) {  // swap both container numbers and amounts
            String tempval = spiceArray[d][e];
            spiceArray[d][e] = spiceArray[d + 1][e];
            spiceArray[d + 1][e] = tempval;
          }
        }
      }
    }

    // After processing all ingredients, mark the order as ready to be mixed
    if (numSpicesOrdered > 0) {
      isOrderMixed = false;
    } else {
      u8g.firstPage();
          do{
            u8g.drawStr(0, 0, "No valid ingredients");
          } while(u8g.nextPage());
    }
  }
```

LCD Draw Spice Summary

```
  void drawSpiceMenu() {
    uint8_t i, h;
    u8g_uint_t w, d;

    u8g.setFont(u8g_font_5x8);
    u8g.setFontRefHeightText(); // line height based on text chars
    u8g.setFontPosTop(); // line position references top left of text

    h = u8g.getFontAscent()-u8g.getFontDescent(); // text height
    w = u8g.getWidth(); // screen width

    uint8_t checkWidth = 8;
    uint8_t checkHeight = 8;
    const uint8_t checkmark_bits[] = {
      0x00,  // Row 0
      0x01,  // Row 1
      0x02,  // Row 2
      0x04,  // Row 3
      0x88,  // Row 4
      0x50,  // Row 5
      0x20,  // Row 6
      0x00   // Row 7
      };

    // if cursor is in range, display normal list
    if(encoderValue <= spiceListMax){
      for( i = 0; i < totalSpices; i++ ) {
        d = (w-u8g.getStrWidth(menu_strings[i]))/2;
        u8g.setDefaultForegroundColor();
        if ( i == encoderValue ) {
          u8g.drawBox(0, i*h+1, w, h);
          u8g.setDefaultBackgroundColor();
        }
        u8g.drawStr(d, i*h, menu_strings[i]);
```

```
            if(isSpiceSelected[i-1]){
              u8g.drawBitmap(d-10, (i)*h, checkWidth/8, checkHeight, checkmark_bits);
            }
          }
        }else{
          // if cursor is out of range, shift menu up
          for( i = 0; i < totalSpices; i++ ) {
            d = (w-u8g.getStrWidth(menu_strings[i+(encoderValue-spiceListMax)]))/2;
            u8g.setDefaultForegroundColor();
            if ( i == spiceListMax ) {
              u8g.drawBox(0, i*h+1, w, h);
              u8g.setDefaultBackgroundColor();
            }
            u8g.drawStr(d, i*h, menu_strings[i+(encoderValue-spiceListMax)]);

            if(isSpiceSelected[i-1+(encoderValue-spiceListMax)]){
              u8g.drawBitmap(d-10, (i)*h, checkWidth/8, checkHeight, checkmark_bits);
            }
          }
        }
      }
```

Motor Functions

```
      void moveSusan(int j) {
        char *temp = senseTemp();

        // Susan picture loop
        u8g.firstPage();
        do {
          int h = u8g.getFontAscent() - u8g.getFontDescent();

          u8g.drawStr(0, 0, "Moving to container #");
          int w = u8g.getStrPixelWidth("Moving to container #");
          u8g.drawStr(w+1, 0, spiceArray[j][0].c_str()); // convert String object to const
char*

          u8g.drawStr(0, 6*h, "Temp: ");
          u8g.drawStr(31, 6*h, temp);
        } while( u8g.nextPage());

        // Exit sleep
        digitalWrite(susanEn, LOW);
        delay(50);

        // Find previous spice container
        int prevSpice;
        if (j == 0) {
          prevSpice = 1; // Assume fully calibrated for first spice
        } else {
          prevSpice = spiceArray[j-1][0].toInt();
        }

        // Calculate number of steps
        int spiceDiff = abs(spiceArray[j][0].toInt() - prevSpice);
        int numSteps = spiceDiff * totalMicrosteps/totalSpices;

        // Actuate
        digitalWrite(susanDir, LOW);
        for (int s = 0; s < numSteps; s++) {
          digitalWrite(susanStep, HIGH);
          delayMicroseconds(2500);
          digitalWrite(susanStep, LOW);
          delayMicroseconds(2500);
        }

        // Finished susan picture loop
```

```
  u8g.firstPage();
  do {
    int h = u8g.getFontAscent() - u8g.getFontDescent();

    u8g.drawStr(0, 0, "Carriage Motion Complete");
    u8g.drawStr(0, 6*h, "Temp: ");
    u8g.drawStr(31, 6*h, temp);
  } while( u8g.nextPage());

  delay(3000);

  // Enter sleep
  digitalWrite(susanEn, HIGH);
}


void moveRailForward() {
  char *temp = senseTemp();

  // Rail fwd picture loop
  u8g.firstPage();
  do {
    int h = u8g.getFontAscent() - u8g.getFontDescent();

    u8g.drawStr(0, 0, "Moving rail forward");
    u8g.drawStr(0, 6*h, "Temp: ");
    u8g.drawStr(31, 6*h, temp);
  } while( u8g.nextPage());

  // Exit sleep
  digitalWrite(railEn, LOW);
  delay(50);

  // 20 revolutions for approx 0.75 in
  int numSteps = 10 * totalSteps;
  digitalWrite(railDir, LOW);
  delay(50);

  // Actuate
  for (int s = 0; s < numSteps; s++) {
    digitalWrite(railStep, HIGH);
    delayMicroseconds(750);
    digitalWrite(railStep, LOW);
    delayMicroseconds(750);
  }

  // Finished rail picture loop
  u8g.firstPage();
  do {
    u8g.drawStr(0, 0, "Rail forward motion");
    int h = u8g.getFontAscent() - u8g.getFontDescent();
    u8g.drawStr(0, h+1, "complete");

    u8g.drawStr(0, 6*h, "Temp: ");
    u8g.drawStr(31, 6*h, temp);
  } while( u8g.nextPage());
  // delay(1000);

  // Enter sleep
  digitalWrite(railEn, HIGH);
}


void moveRailBackward() {
  char *temp = senseTemp();

  // Rail backward picture loop
  u8g.firstPage();
  do {
```

```
    int h = u8g.getFontAscent() - u8g.getFontDescent();

    u8g.drawStr(0, 0, "Moving rail backward");
    u8g.drawStr(0, 6*h, "Temp: ");
    u8g.drawStr(31, 6*h, temp);
  } while( u8g.nextPage());

  // Exit sleep
  digitalWrite(railEn, LOW);
  delay(50);

  // 20 revolutions for approx 0.75 in
  int numSteps = 10 * totalSteps;
  digitalWrite(railDir, HIGH);
  delay(50);

  for (int s = 0; s < numSteps; s++) {
    digitalWrite(railStep, HIGH);
    delayMicroseconds(750);
    digitalWrite(railStep, LOW);
    delayMicroseconds(750);
  }

  // Finished rail picture loop
  u8g.firstPage();
  do {
    u8g.drawStr(0, 0, "Rail backward motion");
    int h = u8g.getFontAscent() - u8g.getFontDescent();
    u8g.drawStr(0, h+1, "complete");

    u8g.drawStr(0, 6*h, "Temp: ");
    u8g.drawStr(31, 6*h, temp);
  } while( u8g.nextPage());
  // delay(1000);

  // Enter sleep
  digitalWrite(railEn, HIGH);
}


void moveAuger(int j) {
  char *temp = senseTemp();

  // Auger picture loop
  u8g.firstPage();
  do {
    int h = u8g.getFontAscent() - u8g.getFontDescent();

    u8g.drawStr(0, 0, "Moving auger");
    u8g.drawStr(0, 6*h, "Temp: ");
    u8g.drawStr(31, 6*h, temp);
  } while( u8g.nextPage());

  // Exit sleep
  digitalWrite(augerEn, LOW);
  delay(50);

  // calculate number of steps
  float spiceAmount = spiceArray[j][1].toFloat();
  int revPerOz = 10;
  int numSteps = ceil(spiceAmount * revPerOz) * totalMicrosteps;

  // Actuate
  digitalWrite(augerDir, LOW);
  for (int s = 0; s < numSteps; s++) {
    digitalWrite(augerStep, HIGH);
    delayMicroseconds(325);
    digitalWrite(augerStep, LOW);
    delayMicroseconds(325);
```

```
  }

  // Finished auger picture loop
  u8g.firstPage();
  do {
    u8g.drawStr(0, 0, "Dispensed ");
    int w = u8g.getStrPixelWidth("Dispensed ");

    // Find decimal point index
    int decimalIndex = spiceArray[j][1].indexOf('.');
    // Truncate string after 5 decimal places
    String truncatedString = spiceArray[j][1].substring(0, decimalIndex+6);
    // convert String object to const char*
    const char *truncatedAmount = truncatedString.c_str();

    u8g.drawStr(w, 0, truncatedAmount);
    int h = u8g.getFontAscent() - u8g.getFontDescent();
    u8g.drawStr(0, h+1, "oz of spice");

    u8g.drawStr(0, 6*h, "Temp: ");
    u8g.drawStr(31, 6*h, temp);
  } while( u8g.nextPage());

  // Enter sleep
  digitalWrite(augerEn, HIGH);
}


void calibrate() {

  // Calibration picture loop
  u8g.firstPage();
  do {
    int h = u8g.getFontAscent()-u8g.getFontDescent();
    u8g.drawStr(0, 0, "Calibrating...");
    u8g.drawStr(0, 6*h, "Temp: ");
    u8g.drawStr(31, 6*h, senseTemp());
  } while( u8g.nextPage());

  // Remove from sleep
  digitalWrite(susanEn, LOW);
  delay(50);

  // Spin carriage quickly until switch triggers for the first time
  digitalWrite(susanDir, LOW);
  for (int s = 0; s < totalMicrosteps; s++) {
    digitalWrite(susanStep, HIGH);
    delayMicroseconds(500);
    digitalWrite(susanStep, LOW);
    delayMicroseconds(500);
    if(!digitalRead(calibrationLimit)){
      break;
    }
  }

  bool isNewTrigger = 0; // track switch state for sensing trigger #2

  // Spin carriage slowly until switch triggers for the second time
  digitalWrite(susanDir, LOW);
  for (int s = 0; s < totalMicrosteps; s++) {
    // sense switch state
    int reading = digitalRead(calibrationLimit);

    // ensure switch has been reset
    if(reading == HIGH){
      isNewTrigger = 1;
    }

    digitalWrite(susanStep, HIGH);
```

```
    delayMicroseconds(2500);
    digitalWrite(susanStep, LOW);
    delayMicroseconds(2500);
    if(reading == LOW && isNewTrigger){
      break;
    }
  }

  // Spin carriage very slowly until switch releases
  digitalWrite(susanDir, LOW);
  for (int s = 0; s < totalMicrosteps; s++) {
    digitalWrite(susanStep, HIGH);
    delayMicroseconds(7500);
    digitalWrite(susanStep, LOW);
    delayMicroseconds(7500);
    if(digitalRead(calibrationLimit)){
      break;
    }
  }

  // Enter sleep
  digitalWrite(susanEn, HIGH);
}
```

Arduino Libraries

U8glib	https://github.com/olikraus/U8glib_Arduino
DHT11	https://github.com/dhrubasaha08/DHT11

Swift Libraries

Foundation	https://developer.apple.com/documentation/foundation/
SwiftUI	https://developer.apple.com/documentation/swiftui/
CryptoKit	https://developer.apple.com/documentation/cryptokit/
LocalAuthentication	https://developer.apple.com/documentation/localauthentication/
CloudKit	https://developer.apple.com/documentation/cloudkit/
UserNotifications	https://developer.apple.com/documentation/usernotifications/
PDFKit	https://developer.apple.com/documentation/pdfkit/

RAMPS Setup Guides

The following sections contain useful information on the setup and use of the RAMPS 1.4 shield for the Arduino Mega.

RAMPS 1.4 Wiring Diagrams

Wiring diagrams are obtainable at the following link, providing the connections within the board itself as well as connections between RAMPS and the corresponding Arduino Mega pins for pin setup in the Arduino sketch.

https://osoyoo.com/2016/07/03/reprap-3d-printer-circuit-connection-graph/

The following tutorial details how to set up the board in terms of wiring and safety, detailing the necessary tools and supplies as well as useful tips, such as the order in which to install components.

https://www.instructables.com/Wiring-3D-Printer-RAMPS-14/

RepRap Documentation

RepRap (short for replicating rapid prototyper) is a project dedicated to assembling and configuring 3D printers at home. Many of the electronic components used in the Flavor Fusion device are commonly found in RepRap systems and thus are well documented online.

The RepRap Discount Smart Controller is an LCD model that is compatible with the Ramps 1.4 shield. It is documented at the following link, with information about pinout, connection types, and setup.
https://reprap.org/wiki/RepRapDiscount_Smart_Controller#Pictures_of_a_Smart_Controller_made_by_RepRapDiscount.com

Power supply setup for RepRap systems is detailed in the link below, with crucial information about safety, useful tools, and other aspects of the setup process.

https://reprap.org/wiki/RAMPS_1.4#Power_Supply

Marlin

Marlin is a firmware designed for RepRap projects using shields such as the RAMPS 1.4. Given its compatibility with RAMPS and Arduino Mega, it provided a precedent for programming the complex and interrelated functions of a 3D-printer style device.

https://github.com/MarlinFirmware/Marlin

A4988 Motor Driver Guides

A4988 Tutorial

A useful tutorial for controlling stepper motors with the A4988 stepper motor driver is provided at the following link. It discusses connections between the driver, motor, and microcontroller as well as how to program basic motion in Arduino IDE.

https://howtomechatronics.com/tutorials/arduino/how-to-control-stepper-motor-with-a4988-driver-and-arduino/

A4988 Datasheet

The datasheet for the A4988 driver is provided below for official information from the board's manufacturer.

https://www.pololu.com/file/0J450/A4988.pdf

Auger Dispensing Tutorial

An example of a similar project that uses a stepper motor and auger to dispense powders is provided, with information like clumping management, manufacturing and materials, and design prototyping.

https://www.youtube.com/watch?v=3coosD16M7U

Appendix G: ABET Accreditation and Topic Criticality Matrix

Project Title:

Flavor Fusion

Please rate the relative importance of the given topic in this design project

ME Design Areas	**Critical/Main contributor**	**Strong contributor**	**Necessary but not a primary contributor**	**Necessary but only a minor contributor**	**Only a passing reference**	**Not Included in this Design Project**
Thermal-Fluid Energy Systems				X		
Machines & Mechanical Systems		X				
Controls & Mechatronics	X					
Materials Selection	X					

Modeling & Measurement Systems	X					
Manufacturing	X					