Design Document: Rocket Chef

1 Executive Summary

Rocket Chef is a fast-paced, immersive VR experience with a simple goal but engaging ways to implement it. The goal behind the game is to create a cooking-like experience - in space. The player is a “cook” in a kitchen, who receives many different orders from robots trying to dig on the nearby moon’s core. The player’s goal is to complete as many orders as possible in the time allotted or risk angering her fickle robot customers. In order to help the player create the different orders, multiple mixing tools are provided in the kitchen, with different motion control actions associated with them. In order to keep track of the many orders that need to be completed, the player has a helmet that shows which orders are left and where to find the materials to complete them. This helmet also allows the player to use rocket boots to move between the different customer locations quickly.

The target audience for this game is someone who is looking for a quick form of entertainment that is fast-paced, but doesn’t require a lot of mental energy to play. The game is appropriate for all ages and sufficiently challenging to be playable by more advanced VR gamers. The VR environment makes for an immersive game experience, and the player is engaged on multiple levels, from time management to multitasking to motor coordination. The nature of the game is competitive, with scorekeeping and a social leaderboard.

In order to implement the key aspects of this game, three critical things we’ll be working on are motion (using our rocket boot assets), mixing actions for different tools using motion controllers, and procedurally generating customers and the orders that they want fulfilled. Besides teleporting, motion in VR games is still relatively unexplored and we’re looking to tackle that challenge by making movement rooted through the realm of a helmet covering most of the player’s head as they travel.

We have a detailed timeline that we believe will allow us to complete this game by May 2nd, the date of the final demo. Within this allotted month, we will have time to do multiple playtest demos as well as keep some buffer time for polishing game logic and asset creation. See Section 4 for more details.

2 GAME Design - CREATIVE

2.1 High Concept

“Diner Dash” in space. Players mix and match fuel components to create the perfect concoction for robots whose mission is to mine rare minerals from the center of the moon. The player moves around on rocket boots to get from robot to robot and mixes within a kitchen located in the fueling station. The player has to be quick, because if the robot runs out of fuel, it will explode, causing the center to destabilize.

2.2 Design Goals

2.2.1 Main Design Features

What is it like to play the game, including:

2.2.1.1 Player goals and objectives.

Main goal: Receive the highest score possible before the game ends.
Secondary goals: Mix each fuel order as perfectly as possible. Serve customers as efficiently as possible.

Type of challenge(s): Fine motor skills (mixing fuel). Time management (taking orders and serving drinks). Navigational (moving around on rocket boots).

Type of conflict(s): Each robot must be served within a certain time limit. If they aren’t refueled within the time limit, then they explode. Three robot deaths means game over.

Winning condition: There is no winning condition, but players can beat friends’ scores and their own previous scores on a leaderboard.

2.2.1.2 Main Rules and Procedures

Operational rules – how do you play the game?

- The flow of the game is that there is a stream of robot customers that appear in your fueling station. Each robot must be served before their timer reaches zero. To satisfy a robot, the player must first determine their fuel order by navigating to the robot and speaking to them. Then they must correctly mix the fuel in the kitchen, and present the drink to the robot. The mixed fuel will have a quality grade depending on how well the player followed the recipe, which along with the robot’s time remaining determines the score the player receives for serving that robot.

Main game mechanic – Describe the main way the user interacts with the system

- The user must follow the steps outlined above. The user constrained by the fact that three failed orders means game over.
- Mixing drinks: The mechanics we want to include are pouring, stirring/mixing, squeezing, and shaking. Each of these actions will also be timed.
- Navigation: Tentatively, we would like to implement a floating-type navigation that is controlled via controller trackpad. To combat motion sickness, we will slow the movement of the player and black out the user's peripheral vision (e.g. Eagle Flight game). We will also have the player wear a helmet to provide them with a fixed point of reference as well as a UI.
- Interacting with robots: When the user approaches a robot close enough, the user can either take their order or serve them the fuel that they are holding. Once the user takes a robot's order, their order will appear on the user's helmet UI.

On a second by second basis: Most of the game will be spent either moving around or mixing fuel. The player should always be preoccupied with some task - this isn't a game requiring pauses to think.

On a minute by minute basis: The player should constantly be switching between the various tasks of the game: mixing fuel, taking orders, or delivering fuel. They must also keep track of the overall game state (e.g. all the robots currently at the fuel station) and individual items (e.g. the current order they're mixing, or how close a robot is to exploding).

What makes this fun or interesting? The archetype of time management and task management games has been proven to be successful. We believe VR can add a whole new level of immersion to this game concept, and thus make it even more engaging to play. We hope that the end product makes the player feel as if they are truly inside the game. Specifically, instead of clicking to complete tasks, users will physically engage with the game using the controllers, their arms, and by moving around, whether that be walking within the kitchen, or using our floating rocket boot system in other parts of the fuel station.

2.2.1.3 Player Resources

Types of resources:

- Player helmet: This is attached to the HUD display. It contains a diegetic UI consisting of a list of orders that have been taken and are still yet to be delivered.

- Fuel recipe book: This is a menu contained in a book that will be located on the kitchen table for easy reference while mixing fuel. Within the menu, the player can select the appropriate fuel order and view the recipe.

- Fuel ingredients: Some of these will be used depending on each individual recipe/order. The ingredients and maybe even the recipes themselves will mirror bartending ingredients.

    - Fuel liquids: These will be color-coded and dispensed from tanks. These form the basis of the recipes, and the most simple recipes will probably only involve this ingredient.
    - Specialty ingredients: These will be dispensed in small quantities from special dispensers, perhaps a salt-shaker or a squeeze dispenser.

- Mixing tools: These will aid the player in completing recipes properly.
    - Shaker
    - Measuring cups
    - Spoon/stirrer
    - Final container (perhaps different dependent on the recipe)

How does the player use resources to achieve goals? This was explained in the above section.

2.2.1.4 Boundaries and Constraints

Describe the game environment and any limitations it imposes on the game play: The environment is separated into two parts.

1. Fuel station: No gravity here. Player uses rocket boots to propel themselves forward as if floating in space. Relatively spacious, allowing for the robots to spread out to provide more of a navigational challenge. However, boundaries are clearly defined by the walls/windows of the station.

2. Kitchen area: Normal gravity applies here. VR here will be room-scale, which means everything is walkable. It is an enclosed area where the player is expected to walk around a bit to collect ingredients and tools.

2.2.2 Appeal

What is the game genre? Fast-paced, time-management action game in a science fiction setting.

Who is the target audience? Gamers looking for a more casual game that isn't too intellectual, but challenges their time management skills and physical coordination.

What is their age? Ages 12 and up.

Why is the game fun to play?
There are four reasons why we think the game will be fun:

1. The optimization aspect. It appeals to the player's desire to perform simple tasks as efficiently as possible. This kind of game has a wide appeal, exhibited by the success of Diner Dash and its imitators.

2. The hand coordination/motor skills component. We are planning on setting the skill cap high, but also making the fuel mixing very intuitive. This way, the player feels challenged to improve themselves and accomplished when they do a good job mixing fuel.
3. Whimsical nature of the game. The story behind our game is an entertaining concept that doesn't take itself too seriously. The fact that robots explode if they aren't refueled is itself an amusing idea, and the animation can be funny as well (the robot head flies off, or nuts and bolts shower the player). We plan on making the fuel ingredients whimsical as well: making the fuel bright colors, and have them mix to produce new colors, for example.
4. The concept of the game mirrors that of a bartender serving drinks, itself a fun concept, but also made more interesting by the fact that our game premise is similar, but also different: patrons are robots, and drinks are concoctions of fuel.

2.2.3 Look and Feel

Style of artwork/color schemes:

We want to go for futuristic, but familiar.

Our game's look and feel will be some combination of Job Simulator and Alien Surgeon Simulator. The alien game has more of a serious vibe to it, and the palette's colors are more muted and suited to what you would expect on the inside of a spaceship. In contrast, Job Simulator has a bright and happy aesthetic. We are thinking of striking a balance between the two: the environment will probably be more on the serious, sci-fi/space side, whereas the robots and the fuel mixing can be brighter and cuter.

What other games is it similar to:

Job Simulator:
https://www.youtube.com/watch?v=FTBkp5im7-Q

- General blocky, busy aesthetic
- Customer interaction

Alien Surgeon Simulator:
https://www.youtube.com/watch?v=j9jDo-NnyaY

- Space aesthetic

Restaurant:
https://www.youtube.com/watch?v=B4Z9QdhdEYo

- "Cooking" mechanics

2.3 Worlds, Characters and Story (if relevant)

2.3.1 Back Story

In a futuristic world, the Earth has run out of precious metals to sustain life. In an effort to continue the species, the space agencies of the world teamed up to create a program Metal Extraction Robot Team (M. E. R. T.) These robots travel to the moon, drill under the surface and extract the metal before traveling back to Earth.

Unfortunately, they only have fuel capacity to take them half way to the moon (trade-off between size and capacity). Thus there must be a refuel station in the middle of space to make sure that the robots can complete the mission. If a robot runs out of fuel, it is designed to explode so that there is no large debris floating around in space.

The fueling stations have thus been designed to be as efficient as possible to make sure that no robot explodes inside. It is your first day on the job and you have to hit the ground running or end up being responsible for the destruction of an entire fueling station.

2.3.2 Spaces/Worlds

Our game is set in a post-modern gas station in space. Each of the robots has their own fueling pod where they get ready to be refueled. The fuel captain prepares the fuel in the “kitchen”.

The kitchen contains different four types of fuel that can be combined to complete the various orders robots have. A player will mix the fuel with a few key actions: dragging and dropping the correct ingredients, inserting them into the containers in a set order, and shaking the contents when necessary. The kitchen will also have different tools that can be used to create the correct mixture of fuels.

2.3.3 Characters

Robots -- These robots travel to the moon, drill under the surface and extracting metal for the Earth. They need to be refueled once they reach the fueling station.

Main Player [[name]] -- It’s your first day on the job and you have to hit the ground running. Your job is to fill the robots up with their fuel to continue the mission.

2.3.4 Levels of Difficulty

We can vary the difficulty level in multiple ways.

1. Decrease the amount of time the robot waits in the station before blowing up.
2. Increase the speed at which robots enter the refueling station.
3. Increase the complexity of the tasks needed to be performed in order to make the fuel.
4. Make it more or less challenging to navigate using the rocket boots.

a. Specifically, we can make the controls regarding changing direction or propelling yourself forward more or less coarse.

2.4 Interaction Models

2.4.1 VR Interface - Navigation and Movement/Control

The player in our world is wearing a space helmet with a heads-up display. The heads-up display has information on it, like the current order, the level number, and the player's score. The space helmet has a glass screen through which the player can view the world.

In order to move, the player points in the direction they want to travel by extending their arms in that direction and then pressing some button to activate the rocket boots. We can attach a clicker to the player in order to have a physical artifact in the game that activates the rocket boots. We believe the player will not experience motion sickness because the helmet will help ground the player by not moving either.

2.4.2 Game Play Sequence and Levels

There is a single map/level. The game will become progressively harder as time passes. The gameplay sequence is simply a stream of robots that enter the refueling station and wish to place fuel orders.

2.4.3 User/Environment – Obstacles and Props

The user mixes fuel in the kitchen using a variety of equipment (whisk, ladle, etc.). Collision detection and pattern recognition will be done using collision boxes around the player or derivative math. For example, if the user is supposed to use a whisk like motion, we would create a class that tracks the location and figures out when the player is changing the direction of his/her hand. User/Environment interactions will also be dictated by the collision meshes around objects in the game.

2.4.4 User/Character

The user then takes the aforementioned mixture to the robot character that needs to be fueled and the robot character takes the fuel mixture. This is the main user/character interaction within the game. Other interaction that may be possible include interaction during the explosion when robots self-destruct. However, at this time we do not foresee any additional interactions between the user and the characters.

2.4.5 Character/Character

For the most part, the characters in the game will not be interacting with each other. The robot will go to the refueling center and wait for its order. The only type of interaction that needs to be simulated is the explosion. On explosion all the physical assets near the character will be blown

away or destroyed in some way. We will need some physics in order to determine this, we may also need some AI to determine how the robots move to their locations in the refueling center (in order to avoid collisions with other robots).

2.4.6 Motion Tracking

We are going to be using motion tracking for two things:

1) The player will have to use the controllers in a specific way to mix ingredients properly. This includes shaking, mixing, and selecting the types of fuel
2) The player will need to be standing in a specific orientation for their rocket boots to activate. Their hands will be pointed in the direction they want to travel.

2.4.7 Multi-Player

Not Applicable

2.4.8 Mobile

Not Applicable

2.4.9 Networked Play

Not Applicable

2.5 Performance and Scoring

2.5.1 State Variables

- Current score -- This variable would store the current score of the player
- Number of Lives -- This variable would store the number of lives the player has remaining
- Number of Robots -- This variable stores the number of robots currently in the refueling center. This variable is required for tiered play. At the beginning of the game the number of robots allowed in the center is lower, while as play goes on the number of robots allowed in the center is higher.
- Number of Robots Fueled -- This variable stores the number of serviced robots. This variable is required to set the level. (eg. every 10 robots fueled increases the max number of robots allowed, etc.)

2.5.2 Feedback

Positive Feedback:

- Haptic feedback when a robot is fueled.
- Score increase when a robot is fueled.

Negative Feedback:

- Haptic feedback when a new robot enters the fueling center.
- Material on robot starts glowing as time goes on to indicate which robots are more important to serve next.
- Haptic feedback and flash when a robot explodes on the screen.

General Feedback:

- Robot positive or negative reaction depending on the quality of fuel that was served to them.
- Sound effects when mixing fuel.

2.5.3 Performance and Progress Metrics

There are no winning conditions for the game. The game ends whenever three robots explode and then a score is generated based on how many robots were fueled, what the average response time is, etc. The player can view their current score and level on the helmet that they are wearing.

3 Game Design - Implementation Details

3.1 Design Assumptions

3.1.1 Hardware

Minimum:
CPU: Intel Pentium G4560
GPU: AMD RX 480
Motherboard: MSI B250M Pro-VD
Storage: Seagate Barracuda 1TB HDD
Power Supply: 430W EVGA W1
RAM: 8GB DDR4
Case: Rosewill Galaxy-01
CPU Cooler: Stock
Operating System: Windows 10

Recommended:
CPU: Intel Core i5-7500
GPU: AMD RX 480 (or GeForce GTX 1060 3GB)
Motherboard: ASRock H270 Pro4
Storage: Seagate Barracuda 1TB HDD
Power Supply: EVGA 500B
RAM: 8GB DDR4-2400
Case: Corsair 200R
CPU Cooler: Stock (or Arctic Freezer 13)
SSD (Optional): Crucial MX300 275GB
Operating System: Windows 10

** See this link for more information: VR Guide

3.1.2 Software

Install the software associated with your device.

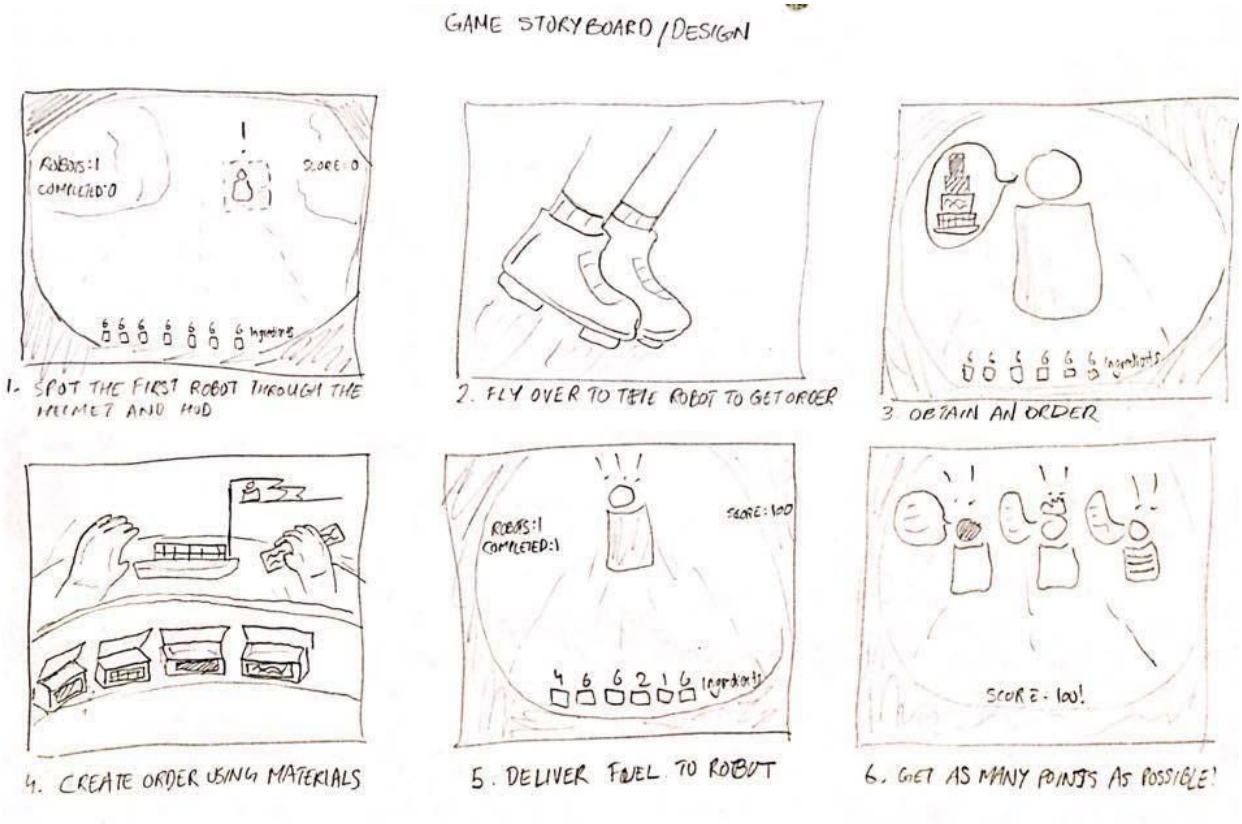
3.1.3 Algorithms and Techniques

The essence of the game relies on the random generation of fuel mixing orders from the robots, and as a result, we will be implementing an algorithm to procedurally generate robots with fuel orders that are sufficiently different to create a challenging and interesting gameplay experience. There are significant challenges involved in implementing the translation of the player character using rockets and we will have to deal with the effects of pitch, yaw and roll to

prevent serious motion sickness. Our development engine of choice, Unreal Engine, uses Perlin noise as a material noise generation and this will be reflected in our asset design and map layout.

3.2 Storyboards

This is a mock of what the helmet will look like. (Source from previous link)



This storyboard describes the flow of our game from the perspective of the player.

3.3 Design Logic

3.3.1 User Solution/Actions

Robots work incessantly and require a lot of energy to keep going. The player's job is to continue to fuel the robots with the right orders for as long as possible, earning points for each successful refuel. Since the player only has the night shift for one day, the objective is to refuel as many robots as possible in the given time frame and score as high as possible. A correct fuel mix delivered in a short amount of time earns more points than one delivered after a significant delay, and incorrect fuel mixes will negatively impact the total score. Each fuel order consists of multiple materials that need to be mixed together. A successfully crafted fuel order will involve a good balance of timing and precision, as too much or too few of a required mix material can upset the final brew. The fuel orders can be crafted using 5-7 original base fuel materials, and these are provided to the player in the kitchen, along with all tools needed to create the fuel order itself (like plates or bottles or containers). The puzzle aspect of the game is using the ingredient fuel materials correctly to create a fuel order as specified by a robot in their procedurally generated recipie.

4 Work Plan

4.1 Tasks

List ALL the tasks and subtasks necessary to build the game application.

Provide separate descriptions of each task and subtask, which members of the group are assigned to it, and the expected task duration.

1. Basic Asset Building for the Kitchen (**Natasha: 12 hours**)
    a. Building multiple materials for objects used in kitchen to mix food: Natasha - 4 hours
    b. Building tools that can be used by user: Natasha - 5 hours
    c. Creating recipe book for possible food options: Natasha - 3 hours

2. Helmet Creation that allows for Rocket Boots Mobility (**Krishna: 14 hours**)
    a. Generating rocket boots asset: Krishna - 2 hours
    b. Adding separate displays within helmet screen to display information: recipes, pending orders, next destination, etc.: Krishna - 6 hours
    c. Adding movement based on rocket boots - 6 hours

3. Basic Asset Building for the Scene between the Kitchen and Moon's Core (**Natasha: 9 hours**)
    a. Multiple planets in the distance: Natasha - 4 hours
    b. Detailed layers of moon's core: Natasha - 5 hours

4. Game Mechanics **(Devesh/Carolina: 24 hours**)
    a. Different ways of combining materials used to mix food: Carolina - 4 hours
    b. Different motion controls for using various tools: Carolina - 4 hours
    c. Testing motion from one spot to the next - preventing motion sickness: Carolina - 4 hours
    d. Setting up a timer and a score-keeping system: Devesh - 2 hours
    e. Creating a sense of urgency: adding sound effects: Devesh - 1 hour
    f. Setting up customers with randomly generated orders: Devesh - 9 hours

5. Asset Generation for the Types of Customers being Served (**Krishna: 5 hours**)
    a. Multiple custom meshes and textures for different customers in scene: Krishna - 5 hours

Total: 64 hours (conservative estimate)

4.2 Milestones

4.2.1 Minor

Describe the functionality you plan to achieve (and will be able to demonstrate) at the end of each week of development

4/6:

- Basic asset-building for the scene of the kitchen, moon's core, planets
- Creating meshes for the food materials and robot fuel pods
- Stretch feature: implementing helmet for user, without additional screens and without rocket boot functionality with it

4/13:

- Implementing motion controls and functionality of tools used to create foods
- Setting up critical game logic: randomly generating orders
- Stretch feature: Adding sound effects, polishing gameplay

4/20:

- Creating rocket boots assets
- Adding additional screens to helmet for user
- Implementing rocket boots movement
- Integrating rocket boots with helmet for user
- Stretch feature: creating meshes for the robots who need to be refueled

5/2:

- Implementing suggestions from play-testing
- Updating rocket boots movement to prevent motion sickness
- Polishing game logic: implementing timer, score-keeping system, end-game condition

4.2.2 Major

Describe the major milestones of the project.

- Generating scene assets: kitchen, planets, fuel pods, etc.
- Generating rocket boots assets and connecting motion control with them
- Implementing the helmet display and preventing motion sickness using them
- Setting up critical game logic for random order generation
- Movable and interactive meshes and characters for the customers asking for orders

Alpha Version

Describe the functionality you plan to achieve and demonstrate in the version.

- Basic asset-building for the scene of the kitchen, moon's core, planets
- Creating meshes for the food materials and robot fuel pods
- Stretch feature: implementing helmet for user, without additional screens and without rocket boot functionality with it
- **Ability to test this with:**
    - **allowing the user to teleport to different aspects of the scene (without rocket boots ability)**
    - **Setting up a customer with one basic order and showing the user the materials in the kitchen needed to be able to generate that order**

Beta Versions

Describe the functionality you plan to achieve and demonstrate in the beta1 and beta2 versions.

Beta1 Version:

- Implementing motion controls and functionality of multiple mixing tools used to create foods
- Setting up critical game logic: randomly generating orders, adding the generated orders to the helmet display for the user,
- Stretch feature: Adding sound effects, polishing gameplay
- **Ability to test this with:**
    - **Randomly generating orders and allowing the user to teleport to the kitchen to generate those orders: end condition will not be completed yet, so the user will not be able to see their score change, but there will be some console validation of correctly/incorrectly mixed orders**
    - **Allowing the user to see their pending orders on their helmet: still unable to teleport using the helmet**

Beta2 Version:

- Creating rocket boots assets
- Adding additional screens to helmet for user
- Implementing rocket boots movement
- Integrating rocket boots with helmet for user
- Stretch feature: creating meshes for the robots who need to be refueled
- **Ability to test this with:**
    - **Allowing the user to teleport using rocket boot assets: this is the critical focus of this demo**
        - **Test to see if user is experiencing motion sickness**
        - **Have the user teleport to multiple scenes in a short period of time**

        - **Get feedback on whether the helmet display is overwhelming - feedback crucial for changes made before final demos**
    - **Test to see how hard the game is with rocket boots movement: are they still able to complete orders in a timely manner/see where the necessary materials are and access them?**

4.3 Development Schedule

Organize your work plan tasks in some kind of readable format and attach it to this document.

Task	Description	Finish By	Hours Needed	Assigned to
Kitchen Scene Asset Building	-Generating assets for the different mixing tools in the kitchen -Generating assets for the different materials used in food -Creating recipe book for different food options	4/6	12 hours	Natasha
Rest of the World Asset Building	-Generating assets for the multiple planets in the distance -Generating assets for the robots' fuel pods -Generating materials and textures for the detailed layers of the moon's core -Generating assets for the space background	4/13	14 hours	Natasha/Krishna
Movement Asset Building	-Creating rocket boots assets -Creating helmet for user with multiple displays for different information -Integrating rocket boots with helmet -Preventing motion sickness	4/20	14 hours	Krishna/Carolina
Motion Controls	-Integrating different	4/20	16 hours	Carolina

	mixing tools with discrete motions on the controllers to create food -Integrating rocket boots with helmet/teleportation capabilities			
Game Logic	-Generating random orders from customers -Displaying the orders on the user's helmet -Creating end-game logic: timer and score-keeping system	5/2	18 hours	Devesh
Customer Asset-Building	-Procedurally generating interactive meshes that can be new customers in the scene	5/2	10 hours	Krishna/Natasha