Connor Cabrera

Project #1

Deliverable #1 / Version 1

Concept of Operations

Contents:

The Current System

The current systems my system will function similarly to are existing restaurant services such as the Chipotle or Papa John's apps. These are apps that allow customer users to remotely create orders for the respective restaurant, pay, and check the status of their order. They allow restaurant users to receive remote orders, update their status and receive payment. All of these tasks are applicable to my system (minus the intricacies of payment, of course).

The Proposed System – Motivation

My system is able to reduce the costs of both the customers and the restaurant. By resorting to pick up only and not relying on a third party such as Uber Eats or Doordash, the customer saves money by not having to pay a service / delivery fee (often much higher than say, the cost of gas to pick up the order), and the restaurant saves money by not having to pay the substantial fees third parties demand for being shown on their platform. I am creating this system to bring restaurants out of the antiquated method of calling in orders and into the market of mobile ordering without depending on the mainstream corporations.

The Proposed System – Users and Modes of Operation

My system will have two classes of Users:

- Customer User
  - Can search for food items and add them to a cart
  - Can "pay" for the order
  - Can check the status of their order
- Restaurant User
  - Can update the status of customer's orders

My system will have three Modes of Operation:

- Customer (not logged in)
  - Default mode
  - Can search for food items and add them to a cart
  - Upon pressing the pay button, they will be prompted to login
- Customer (logged in)

    - Full access to features of the Customer User
- Restaurant (logged in)
    - Full access to features of the Restaurant User

The Proposed System – Operational Scenarios

Typical Scenarios:

- Account Creation – Upon pressing the “Create an Account” button, users will be prompted to answer whether they are creating a Customer or Restaurant account. Customer accounts will be able to be created using an email address and password. However the creation of Restaurant accounts will acquire inputting a 9-digit pin known only to restaurant employees.
- Customer Scenario – The customer scenario will include the execution off all Customer User features in order. A hungry customer will assemble their order and send it off to the restaurant. They will then check their order until it is ready for pickup, then do so. The order will disappear after being marked as “Picked Up”.
- Restaurant Scenario – The Restaurant User will be notified of a new order, and as they prepare said order, they will update its status as needed. The order will disappear after being marked as “Picked Up”.

Atypical Scenarios:

- Loss of Internet – Orders will be stored locally and only use the internet to be sent to Restaurant Users. Restaurant Users can then confirm the order by marking it as “Received Order”. Until this has happened, the Customer User will see their order as “pending”. If the order is still pending after 2 minutes, the system will assume a fault has occurred and notify the Customer User to retry their order. In addition, for all status updates, Customer Users will have to confirm the updated status by pressing “OK”. Should the Restaurant not receive this confirmation after 10 minutes, the system will assume a fault has occurred and automatically retry the status update. The Restaurant User will be able to perform sequential status updates even if they have not received a confirmation.

- Incorrect Input – During the Customer User’s searching of food items, relevant searches will display their respective items, while irrelevant searches will display “No Items Found”. During account creation, users will be asked to confirm their email addresses (which must have an @ in them) and passwords by typing them in twice and having them match. During login, correct user information will result in a “successful” login while incorrect user information will result in an “unsuccessful” login. The system will handle password resets, but not full account resets. The two states for searching and logging in will operate as input sanitization by providing users a destination no matter what they input.

The Proposed System – Operational Features

Must Have:

- Account creation / user separation
- Open line of communication (via status updates) between Customer Users and Restaurant Users, preferably the internet.
- A menu that can be searched and whose items can be added to a cart
- A “pay” button that sends the order to Restaurant Users

- A sequence of updates Restaurant Users can send to Customer Users
- A screen where Customer Users can see and confirm status updates
- Deletion of completed orders

Would Like to Have:

- Push notifications
- Options to customize food item (Ex. No Ketchup, No Lettuce)

The Proposed System – Analysis

My system will be coded in C++ and produced using the Unreal Engine. I know neither, but seeing as how C++ and Unreal are the industry standard for games and applications, I figured it's as good a time as any to learn. This will be a mobile app and the target OS will be Android, as approx. 75% of the population uses it. If Unreal supports IOS, I will try to extend support, however this is pending due to the current legal battle between Apple and Epic Games (the creators of the Unreal Engine). For the sake of this project, I will not be purchasing server space. Optimally I would like to see if it is possible to upload data to a Google Drive Folder that was owned by the restaurant. However if this is not possible data will be stored locally with security measures in place to prevent users from accessing other users' information.

Almost all aspects of this development will be learned from scratch; however the skills will be undeniably helpful in my future so it will be worth the work. The system will be on users' mobile devices, but will require an internet connection between Customer Users and Restaurant Users. Therefore it would not work properly in locations with unreliable internet connections.

Should any failures of the mobile device occur, for example it runs out of battery, by extension the application will be unusable as well.

Both C++ and the Unreal Engine are reputable development tools that have had decades of innovation and implementation. The next alternative would be C# and the Unity Engine. While I have experience with Unity in the past, their discontinuation of Java Script support would basically mean I'd have to learn everything from scratch to use it as well. Since both platforms require the same learning period and Unreal is more widely used, the choice is simple. However the aforementioned legal battle does remain in the back of my head as something that could change the playing field in the years to come.

Though they will try to be learned / ironed out, potential weaknesses in my system would be security should data be stored locally as well as communication between the two types of accounts as networking will be the hardest obstacle to surpass. This system is superior to its rivals as the restaurant won't have to pay ridiculous fees to join a third party's library. As for other restaurants that already have their own app like this, the restaurant that uses my system will have to beat them in the kitchen!

Project Management Plan

Contents:

Applicable Standards

Coding Standard:

- Capitalize the first letter of all class names. Use lowercase for the first letter of all method names
- Any time you open a curly brace, that curly brace shouldn’t start on a new line*
- Any time you open a new code block, indent all the code within that code block one level deeper than you were already indenting
- Be consistent with the amount of indentation you’re using, and be consistent in using either spaces or tabs for indentation throughout your source file
- Avoid block-style comments. Instead, use inline-style comments. Always include a space after the “//” in your comments
- Avoid excessive consecutive blank lines
- Leave a space on both sides of any binary operators you use in your code
- Do not leave a space before the opening parenthesis in an if statement or a loop*
- Use meaningful variable names that convey the purpose of your variables
- Source for all of the above: Professor Sean Szumlanski, UCF. Items marked with an “*” are altered from the professor’s original standards.

Artifact Size Metric Standard:

- Predicted # of Classes: 4 (Account, Menu, Food, Order)
- # of Lines of Code: 400 – 1200 (100 – 300 per class)
- # of Input Sources: 2 (Tapping on screen, Keyboard)
- APIs: Possibly 1 (Google Drive Storage)

Software Life Cycle Process

I will try to follow a hybrid of the Agile Method and Incremental Development Models when creating my software. The Agile Method Model focuses on getting right to work on the product in coordinated sprints, building upon each iteration, and solving problems as they arise. This method does not get caught up in following a fixed plan and heavy documentation and instead focuses on getting a working product as soon as possible. The aspect I will take from the Incremental Development Model is dividing the final product into manageable subsystems and getting them working one at a time before finally getting them to all work in unison. I look forward to this hybrid of approaches to allow me to work quickly while also not getting overwhelmed.

Tools and Computing Environment

I will be designing and building my software on my Windows 10 desktop using C++ in the Unreal Engine. The C++ libraries will help with raw code construction and the libraries and tools in the Unreal Engine will help with design. I will also use the Google Drive API to integrate cloud storage for the software’s data, which will provide reputable security.

Configuration Management

Seeing as how this semester I will be working on this project by myself, I will have access to all versions, changes, and documents since I will be the one making them. I will use GitHub to keep a backlog of all changes made in between uploads.

Quality Assurance

I will be responsible for all quality control. This will include intentionally trying to break the software and looking for weaknesses. During the later stages I will also give the application to friends to see if they can break it as well. As per the Agile Method Model, bugs should be fixed on the spot rather than spending time documenting them, however if this is a requirement for the project, bugs will be documented to the extent necessary. In either case, fixed bugs will be noted with each version upload to GitHub.

Risk Management

- Generic Risks:
    - Procrastination – I purchased a large whiteboard calendar and hung it up next to my desk to better manage my time
    - Illness – Following recommended safety measures during these times
    - Misunderstanding Requirements – Asking the TAs / Professor for clarification
- Project Specific Risks:
    - Database Crash – Using a powerful and reputable database (in my case, Google)
    - GitHub Crash – Maintaining a backup of all versions locally

  - Input File Problem – If the input can’t be changed, focus on acceptance of what is available
  - Library Gets Deprecated – Using reputable resources
  - IDE Crashes – Ctrl + S
- Business Risks:
  - A Competitor is Developing a Similar Product – Focusing on getting my product out as quickly as possible while maintaining quality control. The first to the market usually succeeds
  - User Dislikes Product – Listen to all feedback thoroughly, and focus on satisfying them. The customer is always right.

Table of Work Packages, Time Estimates, and Assignments

Package	Who	Time	Description
Class Design	Me	1.5 Weeks	Building up the classes and their respective components
Class Programming	Me	2.5 Weeks	Programming the interactions between the classes
UI Design	Me	1.5 Weeks	Designing Graphics
Networking	Me	1.5 Weeks	Bringing the Software Online
Testing	Me	1 Week	Majority of Quality Control
Polishing	Me	1 Week	Preparing Demo, Working out Final Bugs
Procrastination	Me	1 Week	Safety Buffer

Plan for Tracking, Control, and Reporting of Progress

My weekly report will include the time I spent working that week, completed tasks, tasks in-progress, tasks planned for the following week, and noticeable bugs or places I got stuck. I will try to follow my Table of Work above, fixing problems as they arise and following the Agile Method Model, focusing of development time rather than documentation time. Should I veer off my timing it may be necessary to take a second look and establish a stricter work plan.

Software Requirements Specification

Contents:

Introduction

Software to be Produced:

The software being developed is a mobile ordering system for a restaurant. It will allow customers to order food from their mobile devices and allow restaurants to manage them. Please refer to the Concept of Operations > The Proposed System for a detailed description of users, scenarios, and features.

Applicable Standards

As per the project description, the system must:

- Allow anyone to create a customer account using an email address and password
- Their passwords must be stored and encrypted in a database
- Customers can search for items and create orders
- Customers press on “Pay” to complete their order
- Customers can see the status of their order
- Restaurant workers must have accounts

- They can update the status of the order

Definitions, Acronyms, and Abbreviations:

None

Product Overview

Assumptions:

- The User will be able to type on their phone and have a stable internet connection
- The User will have adequate memory (will require very little)
- The Developer has adequate time to learn C++, Unreal Engine, App Development, & Networking

Stakeholders:

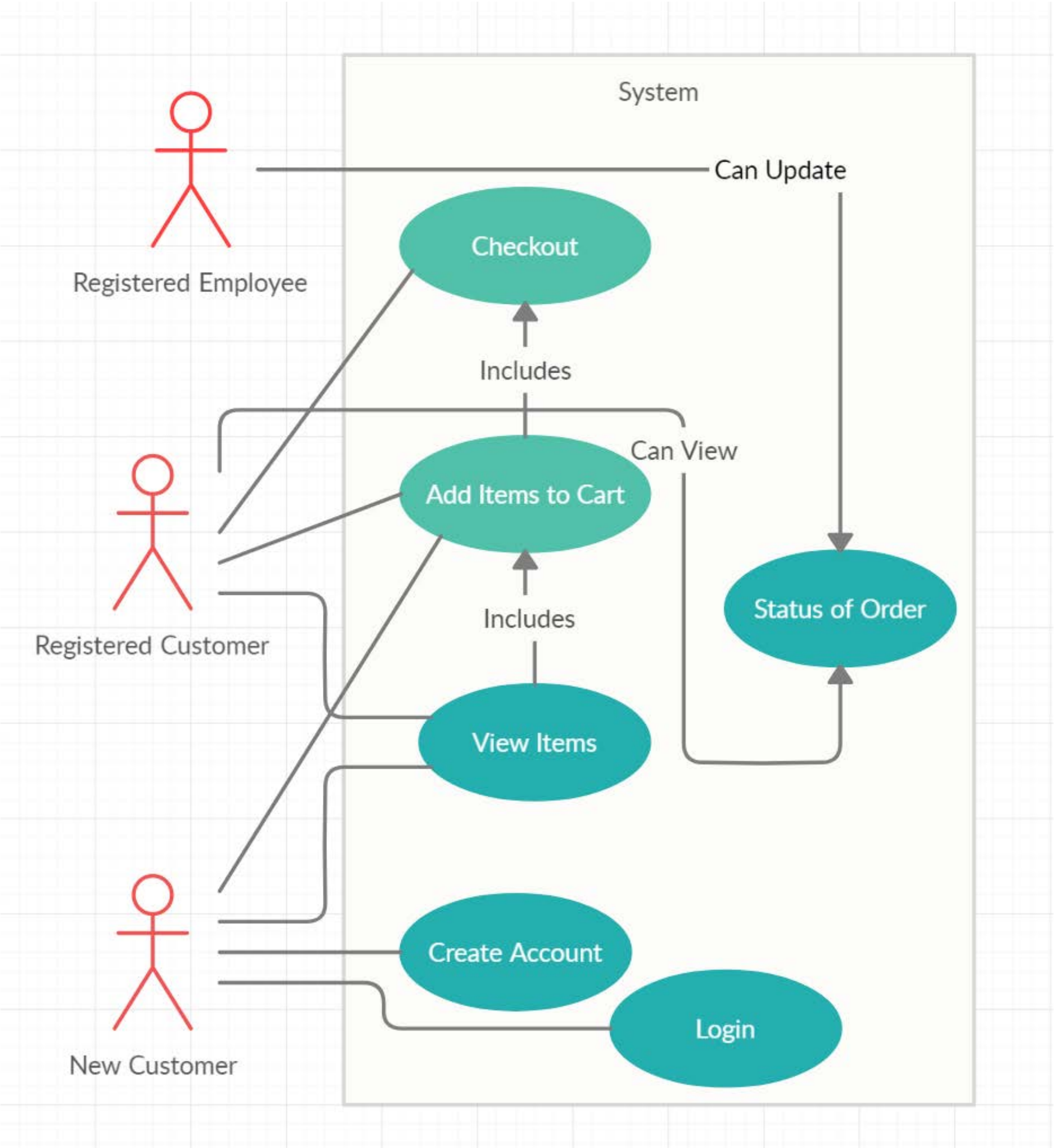
- Nafisa (Restaurant) **–** The restaurant hopes to increase their sales by giving customers the convenience of mobile ordering.
- Myself (Developer) **–** My performance on completing this project will affect my grade for this class.

Event Table:

Note I am not familiar enough with the internal workings of C++, Unreal, or Mobile Apps yet to be able to provide an adequate description of Internal Data & State.

Event Name	External Stimuli	External Response
Select Item	Press on desired food item	Pulls up a page with description of item and option to add to cart
Adjust Quantity of Item in Cart	Press on '+' or '-' while in item page	Corresponding number in item page changes; Cart updates
Go To Login	Press on "Login / Create Account" Button	Opens Login Page with an option at the bottom to Create an Account
Log In	Enter credentials and press "Login" button	(On Success) Taken back to menu with account name visible and option to log out (On Failure) Credentials Deleted and told they were incorrect.
Log Out	Press "Logout" button	Taken back to default (guest) menu page
Create an Account	Press "Create an Account" Prompt	Asked to choose between Customer or Restaurant Employee Accounts
Create Customer Account	Press "Customer" and enter and confirm credentials	Notified account was successfully created and taken back to Login Page
Create Restaurant Account	Press "Restaurant Employee" and enter and confirm credential (including PIN)	Notified account was successfully created and taken back to Login Page
Complete Order	Customer presses on "Pay" Button	Taken to Order Status Page
Receiving Order	Restaurant Employee Confirms Receipt of Order and Updates status accordingly	Each status update will show a sequential button for the next one
Confirming Status Updates	Customer presses "OK" button after receiving each status update	"OK" prompt disappears showing Order Status Page
Order Picked Up	Restaurant changes status of order to "Picked Up" and Customer Confirms it	Order is deleted. App returns to Default State

Use Case Diagram:



Use Case Descriptions:

- New Customers can view the menu items and add them to their cart, but cannot checkout and will be prompted to login or create an account.
- New Customers can login or create an account.
- Registered Customers don't need to login or create an account because they are already logged in.
- Registered Customers may checkout after creating an order.
- Registered Employees can update the status of orders they receive.
- Registered Customers can view the status of their order.

Specific Requirements

Functional Requirements:

No: FR-01
Statement: Shall allow anyone to create a customer / restaurant account using an email address and password
Source: Client
Dependency: None
Conflict: None
Supporting Materials: None
Evaluation Method: After an account is created, credentials will be stored in the database.
Revision History: Myself Created: (9/17/20) Last Edited: (9/19/20)
No: FR-02
Statement: Shall allow anyone to login
Source: Client
Dependency: None
Conflict: None
Supporting Materials: None
Evaluation Method: On success, user will be notified and be able to see their profile.
Revision History: Myself Created: (9/17/20) Last Edited: (9/19/20)
No: FR-03
Statement: Shall allow customers to search for items and create orders

Source: Client
Dependency: None
Conflict: None
Supporting Materials: None
Evaluation Method: Successful searches will show corresponding items and added items will show in the cart
Revision History: Myself Created: (9/17/20) Last Edited: (9/19/20)
No: FR-04
Statement: Shall allow customers to checkout their order
Source: Client
Dependency: FR-03
Conflict: None
Supporting Materials: None
Evaluation Method: Customer will be taken to the order status page
Revision History: Myself Created: (9/17/20) Last Edited: (9/19/20)
No: FR-05
Statement: Shall allow restaurant workers to update the status of the order
Source: Client
Dependency: FR-04
Conflict: None
Supporting Materials: None
Evaluation Method: Sequence of statuses will show after each respective update
Revision History: Myself Created: (9/17/20) Last Edited: (9/19/20)
No: FR-06
Statement: Shall allow customers to view status
Source: Client
Dependency: FR-05
Conflict: None
Supporting Materials: None
Evaluation Method: Customer will have to confirm each update
Revision History: Myself Created: (9/17/20) Last Edited: (9/19/20)

Interface Requirements:

No: IR-01
Statement: Shall take input from a keyboard
Source: Individual
Dependency: None
Conflict: None
Supporting Materials: None

Evaluation Method: Letters pressed will show as appropriate
Revision History: Myself Created: (9/17/20) Last Edited: (9/19/20)
No: IR-02
Statement: Shall take input from touch screen
Source: Individual
Dependency: None
Conflict: None
Supporting Materials: None
Evaluation Method: Buttons pressed will perform their respective actions
Revision History: Myself Created: (9/17/20) Last Edited: (9/19/20)
No: IR-03
Statement: Shall add items to cart
Source: Individual
Dependency: IR-02
Conflict: None
Supporting Materials: None
Evaluation Method: Pressing the ‘+’ or ‘-’ while in the item page will add it to the cart
Revision History: Myself Created: (9/17/20) Last Edited: (9/19/20)

Physical Environment Requirements:

No: PE-01
Statement: Shall run on a mobile phone
Source: Individual
Dependency: None
Conflict: None
Supporting Materials: None
Evaluation Method: The application opens on a mobile phone
Revision History: Myself Created: (9/17/20) Last Edited: (9/19/20)
No: PE-02
Statement: Shall have a stable internet connection
Source: Individual
Dependency: None
Conflict: None
Supporting Materials: None
Evaluation Method: Order statuses will be updated successfully / Logins will be successful
Revision History: Myself Created: (9/17/20) Last Edited: (9/19/20)

User and Human Factors Requirements:

No: UHF-01
Statement: Shall have Customer Users
Source: Client
Dependency: None
Conflict: None
Supporting Materials: None
Evaluation Method: A Customer User can successfully perform all Customer User Features depending on their login status (see The Proposed System – Users and Modes of Operation, pg. 2-3)
Revision History: Myself Created: (9/17/20) Last Edited: (9/19/20)
No: UHF-02
Statement: Shall have Restaurant Users
Source: Client
Dependency: None
Conflict: None
Supporting Materials: None
Evaluation Method: A Restaurant User can successfully perform all Restaurant User Features given they are logged in (see The Proposed System – Users and Modes of Operation, pg. 2-3)
Revision History: Myself Created: (9/17/20) Last Edited: (9/19/20)
No: UHF-03
Statement: Shall be a very low skill application
Source: Individual
Dependency: None
Conflict: None
Supporting Materials: None
Evaluation Method: Anyone can use the application
Revision History: Myself Created: (9/17/20) Last Edited: (9/19/20)
No: UHF-04
Statement: Shall provide protection against misuse / incorrect input
Source: Individual
Dependency: None
Conflict: None
Supporting Materials: None
Evaluation Method: Invalid searches will show as invalid & Invalid logins will be unsuccessful
Revision History: Myself Created: (9/17/20) Last Edited: (9/19/20)

Documentation Requirements:

No: DocR-01
Statement: Developer shall provide all required documentation to the professor. This assumes an expert's skill for reading the documentation. It will be typed in Microsoft Word 2010 with a version number on top.
Source: Individual
Dependency:
Conflict: None
Supporting Materials: None
Evaluation Method: Submission by the required means (Webcourses)
Revision History: Myself Created: (9/17/20) Last Edited: (9/19/20)

Data Requirements:

No: DatR-01
Statement: Account information shall be encrypted
Source: Client
Dependency: None
Conflict: None
Supporting Materials: None
Evaluation Method: Even if access to the information is obtained, it will be unusable due to the encryption
Revision History: Myself Created: (9/17/20) Last Edited: (9/19/20)
No: DatR-02
Statement: Item prices shall be totaled (to 2 decimal places)
Source: Individual
Dependency: None
Conflict: None
Supporting Materials: None
Evaluation Method: Cart total shall reflect total cost of all food items
Revision History: Myself Created: (9/17/20) Last Edited: (9/19/20)
No: DatR-03
Statement: Tax shall be added to the cart's total (to 2 decimal places)
Source: Client
Dependency: None
Conflict: None
Supporting Materials: None
Evaluation Method: Cart grand total reflects total cost of all food items plus a 6.5% tax
Revision History: Myself Created: (9/17/20) Last Edited: (9/19/20)

No: DatR-04
Statement: Data shall be stored on a Google Drive
Source: Individual
Dependency: None
Conflict: None
Supporting Materials: None
Evaluation Method: Successful uploads to the remote folder
Revision History: Myself Created: (9/17/20) Last Edited: (9/19/20)

Resource Requirements:

No: RR-01
Statement: Developer shall learn C++, Unreal Engine, Mobile Development, and Networking
Source: Individual
Dependency: None
Conflict: None
Supporting Materials: None
Evaluation Method: I can’t make the app unless I learn how to
Revision History: Myself Created: (9/17/20) Last Edited: (9/19/20)
No: RR-02
Statement: The system shall run on Android 10 and should run on IOS 13
Source: Individual
Dependency: None
Conflict: None
Supporting Materials: None
Evaluation Method: The application will be tested on my mobile device (Android 10) as well as a friend’s (IOS 13)
Revision History: Myself Created: (9/17/20) Last Edited: (9/19/20)

Security Requirements:

No: SR-01
Statement: Sensitive user information shall be stored on a Google Drive folder (that the restaurant will have access too). This provides all the security and reliability of Google. In addition since it is a project requirement, data will be encrypted too.
Source: Individual / Client
Dependency: None
Conflict: None
Supporting Materials: None
Evaluation Method: Successful uploads to the remote folder as well as data being encrypted

Revision History: Myself Created: (9/17/20) Last Edited: (9/19/20)

Quality Assurance Requirements:

No: QAR-01
Statement: System shall be available 24 hours a day
Source: Individual / Client
Dependency: None
Conflict: None
Supporting Materials: None
Evaluation Method: Appropriate testing at different times
Revision History: Myself Created: (9/17/20) Last Edited: (9/19/20)
No: QAR-02
Statement: System shall handle faults as described in The Proposed System – Operational Scenarios > Atypical Scenarios, pg. 3-4
Source: Individual / Client
Dependency: None
Conflict: None
Supporting Materials: None
Evaluation Method: Appropriate testing by intentionally causing faults
Revision History: Myself Created: (9/17/20) Last Edited: (9/19/20)

Supporting Material

- Deliverable 1 – ConOps.pptx (Located on class Webcourses)
- Deliverable 1 – Project Management Plan.pptx (Located on class Webcourses)
- Deliverable 1 – SRS.pptx (Located on class Webcourses)
- Project 1: Restaurant System (Project Description) (Located on class Webcourses)
- PA-chapter03.pdf (Planning and Managing the Project) (Located on class Webcourses)
- PA-chapter04.pdf (Capturing the Requirements) (Located on class Webcourses)
- UML Use Case Diagram Example (https://www.uml-diagrams.org/examples/online-shopping-use-case-diagram-example.html)

- Configuration Management: What Is It and Why Is It Important? (https://www.plutora.com/blog/configuration-management)
- Google Drive API (https://developers.google.com/drive/api/v2/about-sdk)

High Level Design

Connor Cabrera

Project 1: Restaurant System

COP4331C, Fall, 2020

Contents of this Document

<u>High-level Architecture</u>

- Major Component AND Interface Diagram provided on next page **<u>(I combined them into 1 Diagram)</u>**
- Questions:
- Do any of them (architectural styles) apply?
    - Client-Server
        - My project reflects a Client-Server Architectural Style because the Restaurant (Server) offers the food and order preparation (Services) to the Customer (Client)
        - In addition there is another Client-Server relationship present in this project. Since the plan is to use Google Drive as a database for user's login credentials, the application itself is a Client to Google Drive (Server)
- Can they be combined to form a unique architecture diagram for your system?
    - N/A

- How will the major components interact with each other?
    - Interactions between major components are denoted by arrows in the diagram.

<u>Design Issues</u>

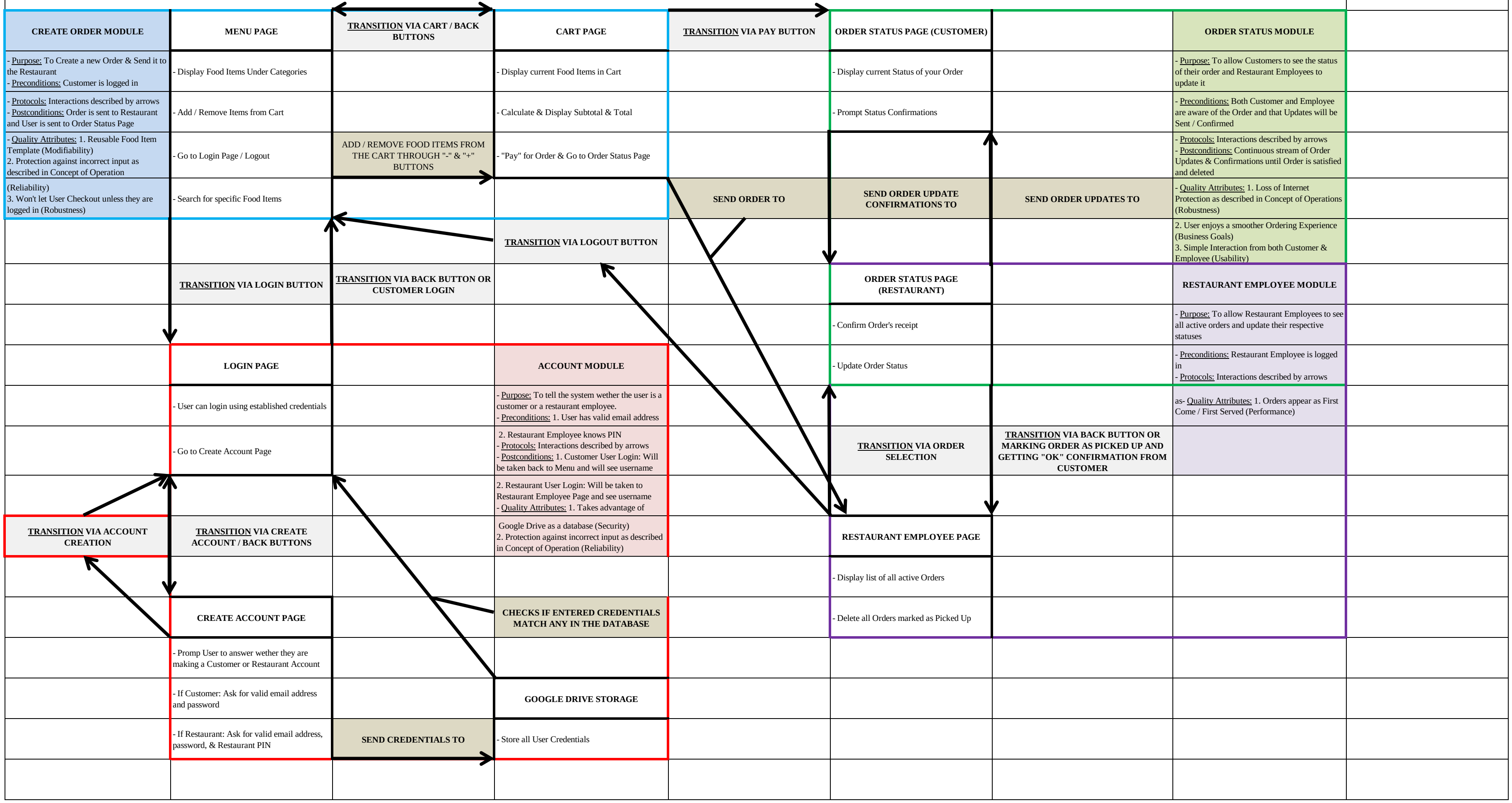
1. Reliability
    - Project has functions in place to defend against faults such as improper inputs and loss of internet connection.
2. Reusability
    - In theory, if this were being made for an actual restaurant, other restaurants could reuse the software by simply changing the menu, database folder, Restaurant PIN, & "payment" destination.

3. Maintainability
    - An easy food item template should make it very easy for the Restaurant to do things like add menu items or edit item prices.
4. Testability
    - Numerous tests will be conducted as will be described in the Test Plan.
5. Performance
    - In terms of speed I believe the speed of the “Restaurant” will be the dominating factor relative to the speed of the software. Orders will be first come / first served and it is up to the Restaurant to satisfy them quickly.
6. Portability
    - The software will be developed for IOS & Android, which make up 99.42% of the global market share for mobile device Operating Systems. Portability should not be an issue.
7. Security
    - The biggest security target (user credentials) is planned to be stored in a Google Drive and protected by their security protocols.
8. Safety
    - Ordering food is not a Safety-Critical system, as its failure cannot result in a loss of life.

- Which issues are relevant to your project?
    - I would say Reliability, Maintainability, & Testability are the issues most relevant to my project as they rely on me the developer to deliver on the promises made in their discussions above.
- What prototypes (if any) will you need to do to evaluate alternate design strategies?
    - Mostly due to time constraints I only plan to make alternative prototypes should a catastrophic failure / oversight arises in my current plan. As described below, by biggest fears are failure to implement Google Drive & Internet Communication.
- What technical difficulties do you expect to encounter?
    - Using Google Drive as a database
    - Communication between the Customer & Employee over the Internet
    - The overall learning curve
- How will you solve them?
    - These problems all share a similar solution. All will take hard work and dedication to solve should the problems arise. The task is not impossible, it has been done before, and it is just a matter of learning how. As for the overall learning curve that will mostly come from learning the new language (C++) and application engine (Unreal), but also just takes time and effort to tackle.
- What design trade-offs did you make in your selection of the architecture?
    - None can come to mind, as explained in the answer to the next question.
- What was your rationale for selecting this architecture?
    - I did not really select the architecture per say. Rather than choosing an architecture and letting it dictate how I designed the project, I instead designed the project exactly how I wanted it to be, and chose an architectural style that best suited the needs of that design.
- What technical risks are involved in this solution?
    - Procrastination, first and foremost. I acknowledge this is a huge problem for me and I am personally working to make it better.

- Failure to implement Google Drive. This would cause me to have to design a whole new way to store the account data and secure it, since my current plan relies on the inherent security of Google.
- Failure to implement Internet Communication. If the Customer and Employee can’t communicate, then the application is useless simple as that.

MAJOR COMPONENTS & INTERFACES



CREATE ORDER MODULE

- Purpose: To Create a new Order & Send it to the Restaurant
- Preconditions: Customer is logged in
- Protocols: Interactions described by arrows
- Postconditions: Order is sent to Restaurant and User is sent to Order Status Page
- Quality Attributes: 1. Reusable Food Item Template (Modifiability)
2. Protection against incorrect input as described in Concept of Operation (Reliability)
3. Won't let User Checkout unless they are logged in (Robustness)

MENU PAGE

- Display Food Items Under Categories
- Add / Remove Items from Cart
- Go to Login Page / Logout
- Search for specific Food Items

TRANSITION VIA CART / BACK BUTTONS

ADD / REMOVE FOOD ITEMS FROM THE CART THROUGH "-" & "+" BUTTONS

CART PAGE

- Display current Food Items in Cart
- Calculate & Display Subtotal & Total
- "Pay" for Order & Go to Order Status Page

TRANSITION VIA PAY BUTTON

ORDER STATUS PAGE (CUSTOMER)

- Display current Status of your Order
- Prompt Status Confirmations

ORDER STATUS MODULE

- Purpose: To allow Customers to see the status of their order and Restaurant Employees to update it
- Preconditions: Both Customer and Employee are aware of the Order and that Updates will be Sent / Confirmed
- Protocols: Interactions described by arrows
- Postconditions: Continuous stream of Order Updates & Confirmations until Order is satisfied and deleted
- Quality Attributes: 1. Loss of Internet Protection as described in Concept of Operations (Robustness)
2. User enjoys a smoother Ordering Experience (Business Goals)
3. Simple Interaction from both Customer & Employee (Usability)

SEND ORDER TO

SEND ORDER UPDATE CONFIRMATIONS TO

SEND ORDER UPDATES TO

TRANSITION VIA LOGOUT BUTTON

TRANSITION VIA LOGIN BUTTON

TRANSITION VIA BACK BUTTON OR CUSTOMER LOGIN

ORDER STATUS PAGE (RESTAURANT)

- Confirm Order's receipt
- Update Order Status

RESTAURANT EMPLOYEE MODULE

- Purpose: To allow Restaurant Employees to see all active orders and update their respective statuses
- Preconditions: Restaurant Employee is logged in
- Protocols: Interactions described by arrows

as- Quality Attributes: 1. Orders appear as First Come / First Served (Performance)

LOGIN PAGE

- User can login using established credentials
- Go to Create Account Page

ACCOUNT MODULE

- Purpose: To tell the system wether the user is a customer or a restaurant employee.
- Preconditions: 1. User has valid email address
2. Restaurant Employee knows PIN
- Protocols: Interactions described by arrows
- Postconditions: 1. Customer User Login: Will be taken back to Menu and will see username
2. Restaurant User Login: Will be taken to Restaurant Employee Page and see username
- Quality Attributes: 1. Takes advantage of Google Drive as a database (Security)
2. Protection against incorrect input as described in Concept of Operation (Reliability)

TRANSITION VIA ORDER SELECTION

TRANSITION VIA BACK BUTTON OR MARKING ORDER AS PICKED UP AND GETTING "OK" CONFIRMATION FROM CUSTOMER

TRANSITION VIA ACCOUNT CREATION

TRANSITION VIA CREATE ACCOUNT / BACK BUTTONS

RESTAURANT EMPLOYEE PAGE

- Display list of all active Orders
- Delete all Orders marked as Picked Up

CHECKS IF ENTERED CREDENTIALS MATCH ANY IN THE DATABASE

CREATE ACCOUNT PAGE

- Promp User to answer wether they are making a Customer or Restaurant Account
- If Customer: Ask for valid email address and password
- If Restaurant: Ask for valid email address, password, & Restaurant PIN

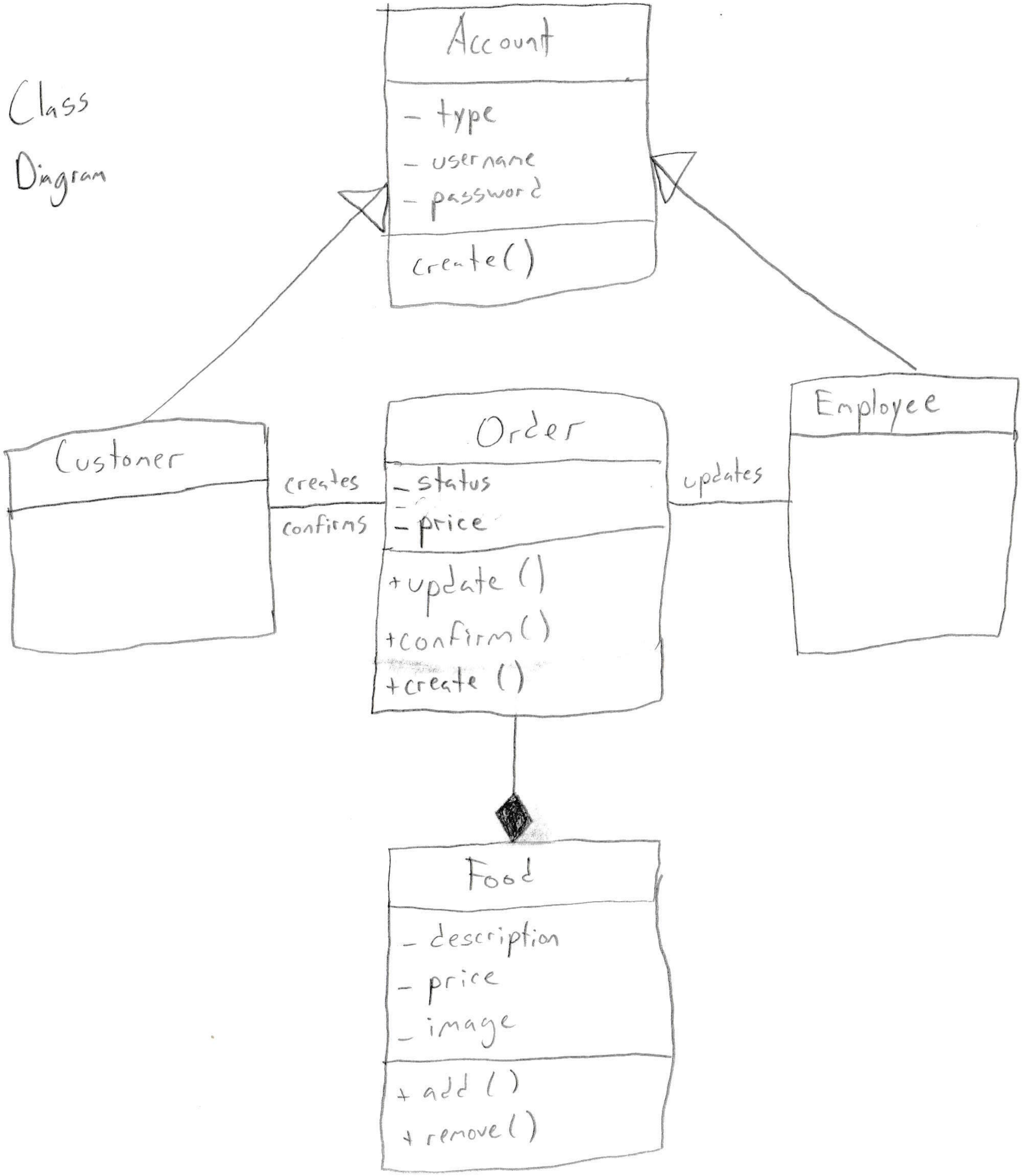
SEND CREDENTIALS TO

GOOGLE DRIVE STORAGE

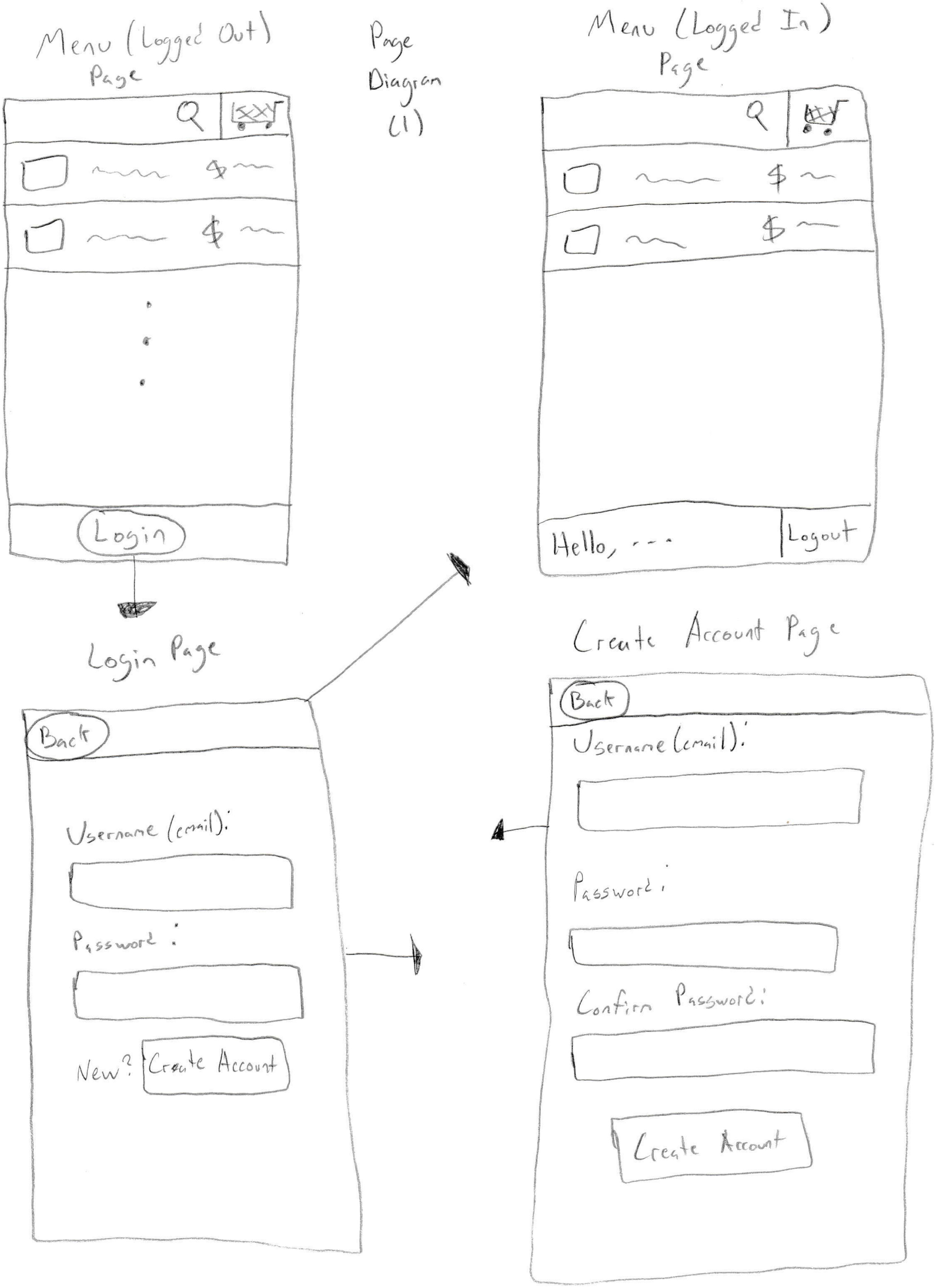
- Store all User Credentials

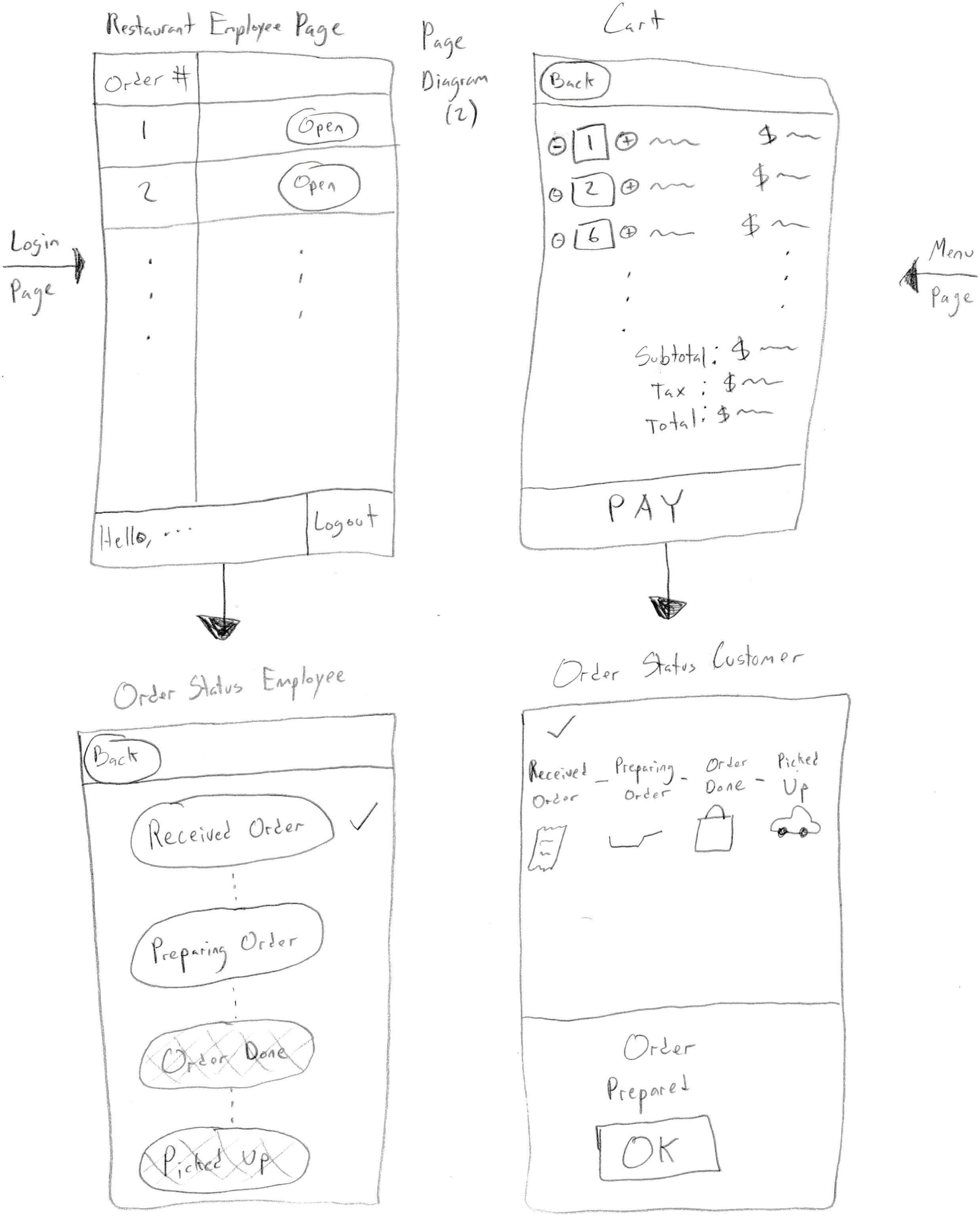
Class

Diagram



Page Diagram (1)




Restaurant Employee Page
Order #
1
Open
2
Open
Hello, ...
Logout
Login Page
Page Diagram (2)
Cart
Back
1
2
6
$
Subtotal: $
Tax : $
Total: $
PAY
Menu Page
Order Status Employee
Back
Received Order
Preparing Order
Order Done
Picked Up
Order Status Customer
Received Order
Preparing Order
Order Done
Picked Up
Order Prepared
OK

Detailed Design

Connor Cabrera

Project 1: Restaurant System

COP4331C, Fall, 2020

Contents of this Document

<u>Detailed Design Issues</u>

<In this section, provide more details to the design issues discussed in the high-level design. Document the results of each design prototype. Document design decisions and associated risks. Provide sufficient detail so the maintainer won't have to say "why did they do it this way?">

- In the High – Level Design I identified 3 Design Issues I fear most:
    - Reliability: Right now I have planned for 2 safeguards against faults: Improper Input & Loss of Internet. Others I can think of, that may be solved in the Development Engine are: Tapping out of Bounds, Trying to tap too fast, intentionally trying to create a restaurant user without knowing the pin. If those exploits are realized, I will have to plan safeguards for them as well.
    - Maintainability: This is probably the easiest issue of the three. All I must do is create a simple food item template that can be copy / pasted / edited as many times as needed to easily make edits to the menu in the future.
    - Testability: This is an issue for me because I can only hope that I find all the errors in my software before the deadline. I must try to get as creative as I can when trying to break the app, but there is always something you don't think of.
- I have no prototypes at this moment to document.
- Most of my design decisions were inspired by already existing apps such as Uber Eats or Doordash. Some risks associated with this decision in the real world could be potential copyright infringement if I borrow too much of my design from their apps. Additional decision risks can be found in my fault safeguards. The inactivity timer that is supposed to detect a loss of internet may be triggered if the restaurant is overwhelmed with orders. Although to combat this risk it is assumed the restaurant already knows the workload it is capable of and has the proper number of employees. Another risk is in the improper input safeguard in that it could maybe declare something invalid that is actually valid. Another decision I made was to overlap the Order Status Module and the Restaurant Employee Module in the Order Status Page (Employee). As risk this could present would be problems with data accessibility and dependency between the modules.

However this problem is not likely to arise as although in the diagram they appear like they might get in each other's ways this design will take place in the User Interface and Unreal should make for easy transitions.

Detailed Design Information

<In this section, include diagrams, listings, etc. to provide the complete details of the design of your product. A detailed class diagram is required. >

< The design documentation must be well organized, modular, and of sufficient detail such that the programmers will be able to start the implementation phase.>

- All diagrams are in between High-Level Design Section and Detailed Design Section. They are the: Major Components& Interfaces Diagram, Class Diagram, & Page Diagrams 1 & 2

Trace of Requirements to Design

<In this section, provide a trace of each requirement in your SRS to the design: for each requirement, indicate which module(s) will fulfill that requirement?>

Requirements that are not associated with the program itself, but more so creation within Unreal Engine will be labeled as: Engine.

Requirement ID	Requirement Description	Module (or Other Design Identifier)
FR-01	Shall allow anyone to create a customer / restaurant account using an email address and password	Account Module
FR-02	Shall allow anyone to login	Account Module
FR-03	Shall allow customers to search for items and create orders	Create Order Module
FR-04	Shall allow customers to checkout their order	Create Order Module
FR-05	Shall allow restaurant workers to update the status of the order	Order Status Module / Restaurant Employee Module
FR-06	Shall allow customers to view status	Order Status Module
IR-01	Shall take input from a keyboard	Engine
IR-02	Shall take input from touch screen	Engine
IR-03	Shall add items to cart	Create Order Module
PE-01	Shall run on a mobile phone	Engine
PE-02	Shall have a stable internet connection	Engine

UHF-01	Shall have Customer Users	Account Module
UHF-02	Shall have Restaurant Users	Account Module
UHF-03	Shall be a very low skill application	Page Diagrams 1 & 2
UHF-04	Shall provide protection against misuse / incorrect input	Test Plan & Concept of Operation Documents
DocR-01	Developer shall provide all required documentation to the professor. This assumes an expert’s skill for reading the documentation. It will be typed in Microsoft Word 2010 with a version number on top.	N/A
DatR-01	Account information shall be encrypted	Account Module
DatR-02	Item prices shall be totaled (to 2 decimal places)	Create Order Module
DatR-03	Tax shall be added to the cart’s total (to 2 decimal places)	Create Order Module
DatR-04	Data shall be stored on a Google Drive	Account Module
RR-01	Developer shall learn C++, Unreal Engine, Mobile Development, and Networking	N/A
RR-02	The system shall run on Android 10 and should run on IOS 13	Engine
SR-01	Sensitive user information shall be stored on a Google Drive folder (that the restaurant will have access too). This provides all the security and reliability of Google. In addition since it is a project requirement, data will be encrypted too.	Account Module
QAR-01	System shall be available 24 hours a day	Engine
QAR-02	System shall handle faults as described in The Proposed System – Operational Scenarios > Atypical Scenarios, pg. 3-4	Test Plan & Concept of Operation Documents

Test Plan

Connor Cabrera

Project 1: Restaurant System

COP4331C, Fall, 2020

Contents of this Document

Overall Objective for Software Test Activity

- A few sentences stating what you expect the software test effort to accomplish
  - Obviously I will test the software to make sure everything in the base product works as it should. For example: Upon account creation, credentials are stored in the Google Drive Folder, User can login, User can make order, etc (the required functions). However I will also test a lot of things I know the software is not meant for (improper input, misuse). Overall I will try my best to break my software.

Description of Test Environment

< What is the hardware and software in the environment in which you will run the test? Who will be the testers (actual users? developers? ...) Will this test environment be the same environment in which the software will operate? If not, how does it differ?>

- Until a final application is rendered, most testing will be performed in the Development Engine (Unreal Engine 4) on my desktop computer. This will be mostly for core functions.
- Once the application looks good on the computer, I will render it for the mobile operating systems. Myself (Android) and a friend (IOS) will take the roles of Customer and Employee (and swap occasionally) and simulate the application’s intended purpose.

Stopping Criteria

<How will you determine when to stop testing the software and either deliver it or send it "back" to development? Things to consider:

- If you find errors during testing: Will you stop testing each time you find a problem and immediately fix that problem? Will you continue testing and recording errors until you find a fatal error that won't allow you to continue? Will you test for 2 hours and then fix whatever errors have been found? Will you test for 2 hours and then hold a group meeting to decide whether to continue to test? ...
- If you find no errors during testing: Does that mean that your software is error-free? How many test cases will you run before you declare the software to be "good enough to deliver"?
- How do you define "good enough to deliver"? Does it require that there are no known errors? Or no known errors other than cosmetic errors? Or no known errors other than cosmetic errors and errors for which there is a well-defined workaround? .... >

  Please clearly state your stopping criteria.

- In terms of strategy, I will likely follow the first recommendation: Once I find an error, I will halt all testing until it is fixed. As for criteria to stop testing: I will never assume my software is 100% free of errors. My criterion for stopping is the project deadline when I physically can't test any longer and must deliver. This does not mean I will be testing my application 24/7 but it does mean it will be regularly tested, say 5 hours / week, until it must be handed in.

Description of Individual Test Cases

Describe EACH individual test to be run. For instance, if you plan to run 30 test cases, you would answer the following questions for each of them. A table format or a bulleted format is acceptable.

- Test Objective: <exactly what does this specific test demonstrate?>
- Test Description: <exactly what will you test? What test data will be used (specifically -- what data values, what data files? This data must be determined in advance. So if you plan to use a test file, include that file in an Appendix of this document. If you plan to key in specific data, include the data here. Make sure the data you choose will allow you to achieve your stated objective for this test.>
- Test Conditions: <Under what conditions will you run this test? This is relevant for software for which there are multiple "modes". For some systems, the test conditions are totally described in the above test environment section -- in this case, the response to this would be "See Test Environment".
- Expected Results: <If the test executes correctly, what will be the result -- i.e., exactly what will the output look like; what will be the resulting data in the database, etc...>

1. Test Objective: Application Runs
    - Test Description: Will the app run when I open it on my mobile device?
        - Test Conditions: App is installed on my phone and I tap it
            - Expected Results: I am taken to the Menu Page
2. Test Objective: Login Button Works
    - Test Description: Will I be taken to the Login Page when I tap the login button?
        - Test Conditions: User is not already logged in & on the Menu Page
            - Expected Results: I am taken to the Login Page
3. Test Objective: Create Account Button Works
    - Test Description: Will I be taken to the Create Account Page when I tap the Create Account Button?
        - Test Conditions: User is on the Login Page
            - Expected Results: I am taken to the Create Account Page
4. Test Objective: Account is Successfully Created
    - Test Description: Will a new account be created?
        - Test Conditions: User is on the Create Account Page
            - Expected Results: New account credentials are stored in the Google Drive Folder
5. Test Objective: Can I login? (Customer)
    - Test Description: Can a User login using credentials they have created
        - Test Conditions: User is on the Login Page
            - Expected Results: Application pulls and validates credentials from Google Drive and User is successfully logged in
6. Test Objective: Improper Input Test
    - Test Description: Will measures in place to reject improper input lead to a valid email address and password being submitted for account creation?
        - Test Conditions: User is on Create Account Page
            - Expected Results: Valid Credentials are stored in the Google Drive Folder
7. Test Objective: Login Transition Test
    - Test Description: Does logging in take me to the Menu Page
        - Test Conditions: User inputs valid login credentials
            - Expected Results: I am taken to the Menu Page and Username is displayed on the bottom.
8. Test Objective: Can I add / remove items from the Cart?
    - Test Description: Will items be added / removed from the cart when pressing the
        - Test Conditions: User is on the Menu Page
            - Expected Results: Items are added / removed from the cart as intended
9. Test Objective: Can I search for Items?
    - Test Description: When I type in the search bar will the item I am looking for be filtered in?
        - Test Conditions: User is on the Menu Page
            - Expected Results: All items that satisfy the search are the only ones that appear on the menu
10. Test Objective: Cart Button Works
    - Test Description: Will I be taken to the Cart Page when I tap the Cart Button?
        - Test Conditions: User is on Menu Page
            - Expected Results: I am taken to the Cart Page

11. Test Objective: Does the Cart display what it should?
    - Test Description: Does the cart accurately display the items I wanted to put in it, their prices, the subtotal, and total?
        - Test Conditions: User is on the Cart Page
            - Expected Results: All information the Cart should display is correct
12. Test Objective: Does the Pay Button Work?
    - Test Description: Will I be taken to the Order Status Page (Customer) when I tap the Pay Button
        - Test Conditions: User is on the Cart Page
            - Expected Results: User is taken to the Order Status Page (Customer)
13. Test Objective: Can I confirm status updates of my order?
    - Test Description: Will the restaurant be able to continue updating my order when I send the OK confirmation
        - Test Conditions: User is on the Order Status Page (Customer) and Employee Sends Order Update
            - Expected Results: Employee can continue on Order
14. Test Objective: Can I logout (Customer)
    - Test Description: Will I be logged out when I tap logout?
        - Test Conditions: User is on Menu Page
            - Expected Results: Menu Page returns to non-logged in User mode
15. Test Objective: Can I login (Employee)
    - Test Description: Will I be sent to the Restaurant Employee Page when I login with Employee Credentials?
        - Test Conditions: User is on the Login screen
            - Expected Results: User is taken to the Restaurant Employee Page
16. Test Objective: Do new orders appear?
    - Test Description: When a customer taps “pay” will the order appear on the Employee’s Restaurant Employee Page?
        - Test Conditions: User is on Restaurant Employee Page
            - Expected Results: New Orders appear on the Restaurant Employee Page
17. Test Objective: Can I select new Orders?
    - Test Description: Will I be taken to the Order Status Page (Employee) when I tap on a new Order?
        - Test Conditions: User is on the Restaurant Employee Page
            - Expected Results: User is taken to the Order Status Page (Employee)
18. Test Objective: Can I send Order Updates
    - Test Description: Will the Customer Get the OK confirmation each time I send a status update
        - Test Conditions: User is on Order Status Page (Employee)
            - Expected Results: Customer gets OK confirmation on Order Status Page (Customer)
19. Test Objective: Will the Order Cancel if Updates are not sent / Confirmations are not received?
    - Test Description: Will measures in place to protect against loss of communication (internet) activate?
        - Test Conditions: Application is inactive (described in Concept of Operations)
            - Expected Results: Order is Cancelled
20. Test Objective: Can I logout (Employee)
    - Test Description: Will I be logged out when I tap logout?

- Test Conditions: User is on Restaurant Employee Page
  - Expected Results: User is taken to non-logged in User Menu Page

21. Test Objective: Can someone new use my application
    - Test Description: Can someone who has never used the app figure it out without any help?
      - Test Conditions: New User is given phone
        - Expected Results: They understand what the application is and how to use it
22. Test Objective: Can I use the application at all hours of the day
    - Test Description: Does the application function as it should at all times?
      - Test Conditions: Using the application at various times
        - Expected Results: It works as expected

<u>Trace of Individual Test Cases to Requirements</u>

< In this section, provide a trace of individual test cases to requirement(s): for each test case, indicate which requirement(s) are validated by that test case? Please make sure each requirement appears at least in one test case and some requirements may appear in multiple test cases >

Please fill the cells of this table for Requirements Traceability Matrix

Requirement ID	Requirement Description	Test Case Reference	Status
FR-01	Shall allow anyone to create a customer / restaurant account using an email address and password	4	In Progress
FR-02	Shall allow anyone to login	5, 15	In Progress
FR-03	Shall allow customers to search for items and create orders	9	In Progress
FR-04	Shall allow customers to checkout their order	12, 16	In Progress
FR-05	Shall allow restaurant workers to update the status of the order	18	In Progress
FR-06	Shall allow customers to view status	12, 13	In Progress
IR-01	Shall take input from a keyboard	9	In Progress
IR-02	Shall take input from	2, 3, 8, 10, 12, 20	In Progress

	touch screen		
IR-03	Shall add items to cart	8	In Progress
PE-01	Shall run on a mobile phone	1	In Progress
PE-02	Shall have a stable internet connection	19	In Progress
UHF-01	Shall have Customer Users	5	In Progress
UHF-02	Shall have Restaurant Users	15	In Progress
UHF-03	Shall be a very low skill application	21	In Progress
UHF-04	Shall provide protection against misuse / incorrect input	6, 9	In Progress
DocR-01	Developer shall provide all required documentation to the professor. This assumes an expert's skill for reading the documentation. It will be typed in Microsoft Word 2010 with a version number on top.	N/A	N/A
DatR-01	Account information shall be encrypted	4	In Progress
DatR-02	Item prices shall be totaled (to 2 decimal places)	11	In Progress
DatR-03	Tax shall be added to the cart's total (to 2 decimal places)	11	In Progress
DatR-04	Data shall be stored on a Google Drive	4	In Progress
RR-01	Developer shall learn C++, Unreal Engine, Mobile Development, and Networking	N/A	N/A
RR-02	The system shall run on Android 10 and should run on IOS 13	1	In Progress

SR-01	Sensitive user information shall be stored on a Google Drive folder (that the restaurant will have access too). This provides all the security and reliability of Google. In addition since it is a project requirement, data will be encrypted too.	4	In Progress
QAR-01	System shall be available 24 hours a day	22	In Progress
QAR-02	System shall handle faults as described in The Proposed System – Operational Scenarios > Atypical Scenarios, pg. 3-4	6, 9, 19	In Progress

Appendices:

Include any test files used in your test cases. If you do not use any test files, leave this section blank.