eCuisine

Meal Prep and Delivery System

<u>System Specifications</u>

Prepared For: Elaine Weltz

Prepared By:
Michael Reese, Software Developer
Reese Solutions

Table of Contents

Executive Summary

Cuisine by Car, an organization operated by Ms. Weltz, has consulted Reese Solutions to construct an ordering system that will allow communication between customers and employees/ management.

Reese Solutions has created this document to assist in the design and creation of eCuisine. This document will outline the software and hardware necessary to implement this system, and will be used by the software developers during the development of eCuisine. It is the firm opinion of Reese Solutions that eCuisine will be relatively easy to implement, and will show a great return.

Introduction

Problem Statement and Project Vision

Ms. Weltz has contacted us about building eCuisine because she is looking for a solution to allow seamless communication throughout every division of her company. Ms. Weltz (and Reese Solutions) believes that implementing eCuisine as the core of the company's operations, will allow for consistent and efficient day-to-date operations, as Cuisine by Car will take care of the organizing customer orders, meal recipes, and delivery routes.

Ultimately, building eCuisine will prove beneficial for both Ms. Weltz and Reese Solutions, we believe we are able to build an affordable solution for a system that will be relied upon by every division of her company, including her clients. We will create a user-friendly system that will allow Ms. Weltz to implement and customize its implementation to serve her organization's needs in the present, and they grow into the future.

System Services

The following is a summary of the services that eCuisine will provide. More detail can be found in the Functional Model section of the attached SRS. eCuisine will:

-Allows the creation of new user accounts(Use case 0, page 16)
-Allows a customer to create a new order(Use case 2, page 18)
-Allows staff to edit recipes(Use case 3, page 19)
-Allows kitchen staff to receive orders(Use case 4, page 20)
-Allows a user order to be prepared/delivered(Use case 5, page 21)

Nonfunctional Requirements and Design Constraints

eCuisine will need to meet the following requirements:

-Allow access for users/family members
-Will need to be able to deploy user and driver applications to pre-existing devices, which will not be difficult to ensure all devices(to a certain requirement) are supported.
-eCuisine will need to be straight-forward and intuitive to navigate. As we may be dealing with a large number of customers who are senior/disabled, eCuisine will be easy to navigate for those who are unfamiliar with technology.
-Startup investment will be within a tight budget.
-System will need to be supported by Reese Solutions to ensure that changes can be made in accordance with growth of Ms. Weltz's operations.

System Evolution

In Version 1 of eCuisine, it will contain all of the functionality described in the previous sections. In future releases, we hope to move the database to the cloud to alleviate server maintenance by Cuisine by Car/Reese Solutions. We also hope to allow for automatic ingredient re-ordering to streamline operation when stock is low. These features are not expected to be included in the first release, though, as they have been determined to be low priority.

Document Outline

This section contains a general outline for the rest of this document. This document will contain:

Structural Model

A class diagram describing component necessary for eCuisine, as well as detailed data about the functions within the classes.

Architecture Design

A diagram of the physical system architecture necessary for eCuisine, as well as hardware and software requirements for the system.

User Interface

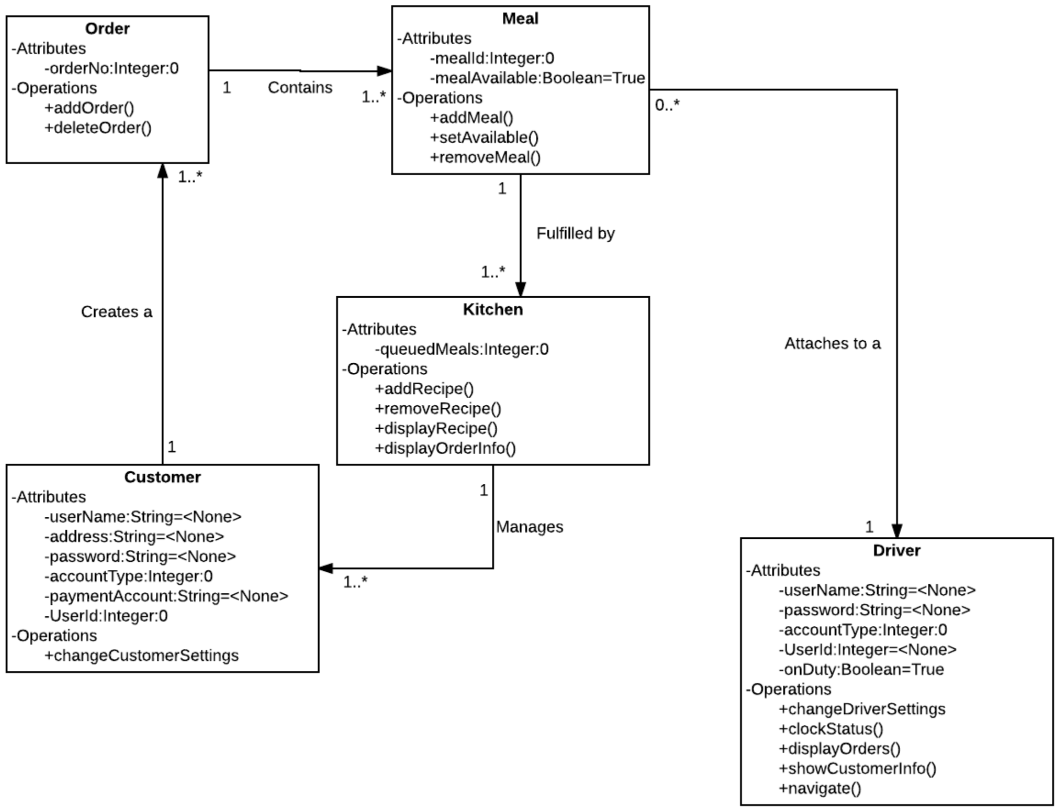
Requirements, constraints, and designed for the user interface.

Structural Model

Introduction

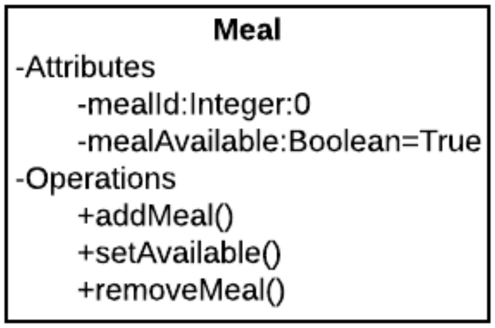
This section contains two subsections. The First contains a UBML class diagram showing the data classes and their attributes and operations. The second contains detailed metadata about the classes shown in the previous subsection

Class Diagram



Metadata

Meal



Description: Represents a meal

Visibility: Public

Is Abstract: No

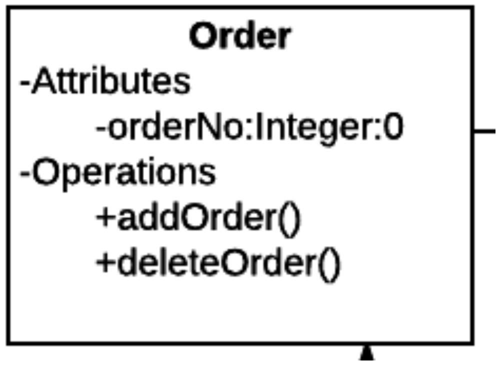
Attributes:

Name	Description	Data Type	Is Derived	Is Read Only	Visibility	Multiplicity	Default Value
mealId	Identification number of meal	Integer	No	No	Private	1	0
mealAvailable	If a specific meal is available(inventory)	Boolean	No	No	Private	1	True

Operations:

Name	Description	Return type	Parameter	Visibility	Scope	Is Query	Is Polymorphic
addMeal()	Add a meal to the database	None	name Direction: in String Default: none	Public	Instance	No	No
setAvailable()	Sets value of mealAvailable	None	mealAvailable: Direction: in Boolean Default: None	Public	Instance	No	No
removeMeal()	Remove meal from database	None	name Direction: in String Default: none	Public	Instance	No	No

Order



Description: Represents a order

Visibility: Public

Is Abstract: No

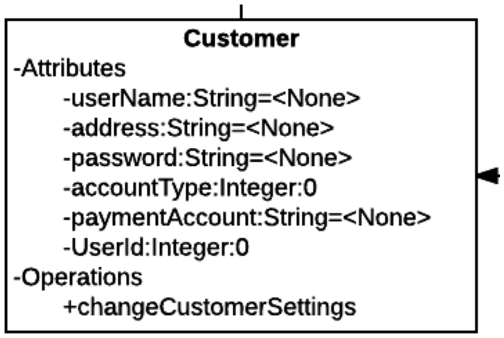
Attributes:

Name	Description	Data Type	Is Derived	Is Read Only	Visibility	Multiplicity	Default Value
orderNo	Identification number of order	Integer	No	No	Private	1	0

Operations

Name	Description	Return type	Parameters	Visibility	Scope	Is Query	Is Polymorhpic
addOrder()	Add order	None	orderId: Direction: in Integer Default: none	Public	Instance	No	No
deleteOrder()	Delete order	None	orderId: Direction: in Integer Default: None	Public	Instance	No	No

Customer



Description: Represents a customer

Visibility: Public

Is Abstract: No

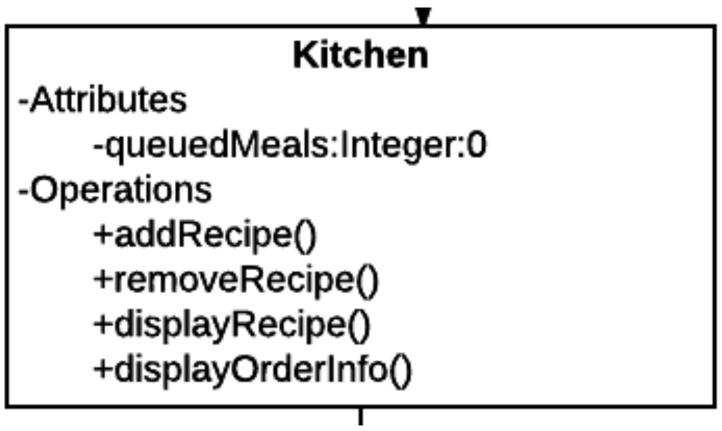
Attributes:

Name	Description	Data Type	Is Derived	Is Read Only	Visibility	Multiplicity	Default Value
userName	The username used by the customer to login	String	No	No	Private	1	<None>
address	Address of customer	String	No	No	Private	1	<None>
password	Password used to login(securely hashed)	String	No	No	Private	1	<None>
accountType	Represents the user’s privileges	Integer	No	No	Private	1	0
paymentAccount	Data used to link to payment	String	No	No	Private	1	String
userId	Id number of the user	Integer	No	No	Private	1	0

Operations:

changeCustomerSettings()	Allows user to change their account settings	None	orderId: Direction: in Integer Default: none	Public	Instance	No	No

Kitchen



Description: Represents kitchen

Visibility: Public

Is Abstract: No

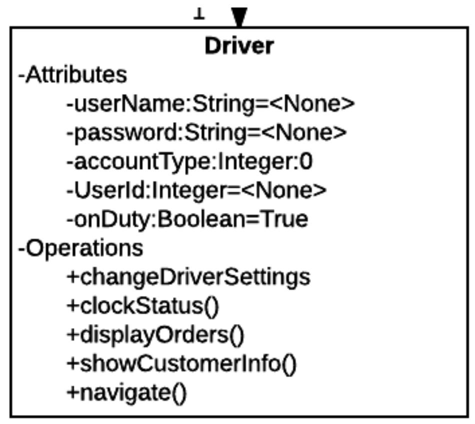
Attributes

Name	Description	Data Type	Is Derived	Is Read Only	Visibility	Multiplicity	Default Value
queuedMeals	Number of meals in queue to be prepared	Integer	No	No	Private	1	0

Operations:

Name	Description	Return type	Parameter	Visibility	Scope	Is Query	Is Polymorphic
addRecipe()	Add a recipe to the database	None	name Direction: in String Default: none	Public	Instance	No	No
removeRecipe()	Remove a recipe to the database	None	name: Direction: in Boolean Default: None	Public	Instance	No	No
displayRecipe()	Displays recipe	None	name: Direction: in String Default: none	Public	Instance	Yes	No
displayOrderInfo()	Display order information	None	orderId: Direction: in integer Default: none	Public	Instance	Yes	No

Driver



Description: Represents a driver

Visibility: Public

Is Abstract: No

Attributes:

Name	Description	Data Type	Is Derived	Is Read Only	Visibility	Multiplicity	Default Value
userName	The username used by the customer to login	String	No	No	Private	1	<None>
password	Password used to login(securely hashed)	String	No	No	Private	1	<None>
accountType	Represents the user’s privileges	Integer	No	No	Private	1	0
userId	Id number of the user	Integer	No	No	Private	1	0
onDuty	Represents if driver is current working	Boolean	No	No	Private	1	True

Operations:

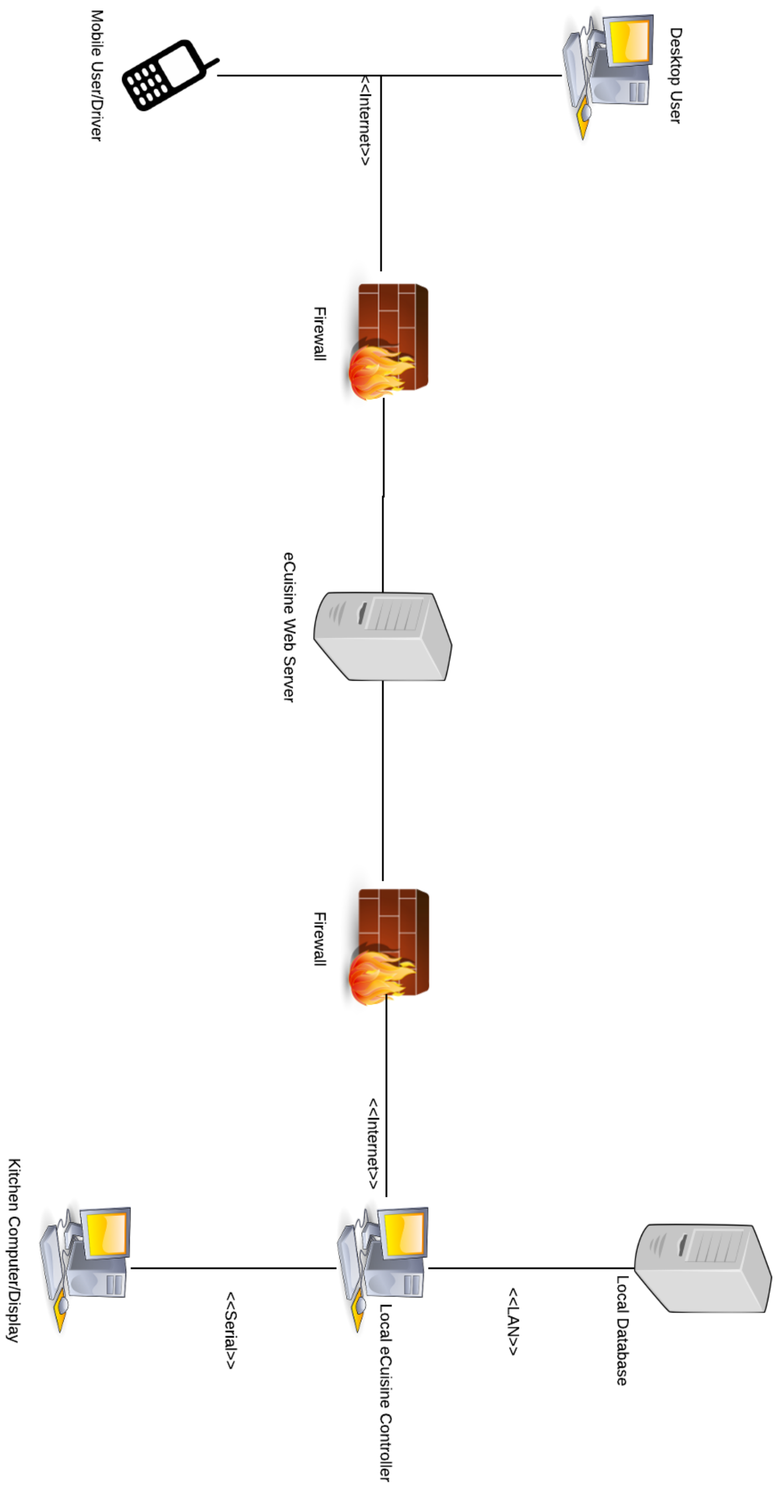
Name	Description	Return type	Parameter	Visibility	Scope	Is Query	Is Polymorphic
changeCustomerSettings()	Allows user to change their account settings	None	orderId: Direction: in Integer Default: none	Public	Instance	No	No
clockStatus()	Allows user to change their clock status	None	onDuty: Direction: in Boolean Default: None	Public	Instance	No	No
displayOrders()	Displays orders	None	order: Direction: in Integer Default: none	Public	Instance	Yes	No
showCustomerInfo()	Display customer information	None	userId: Direction: in integer Default: none	Public	Instance	Yes	No
Navigate()	Allows driver to navigate to order location(GPS)	None	orderId: Direction: in Integer Default: none	Public	Instance	No	No

Architecture Design

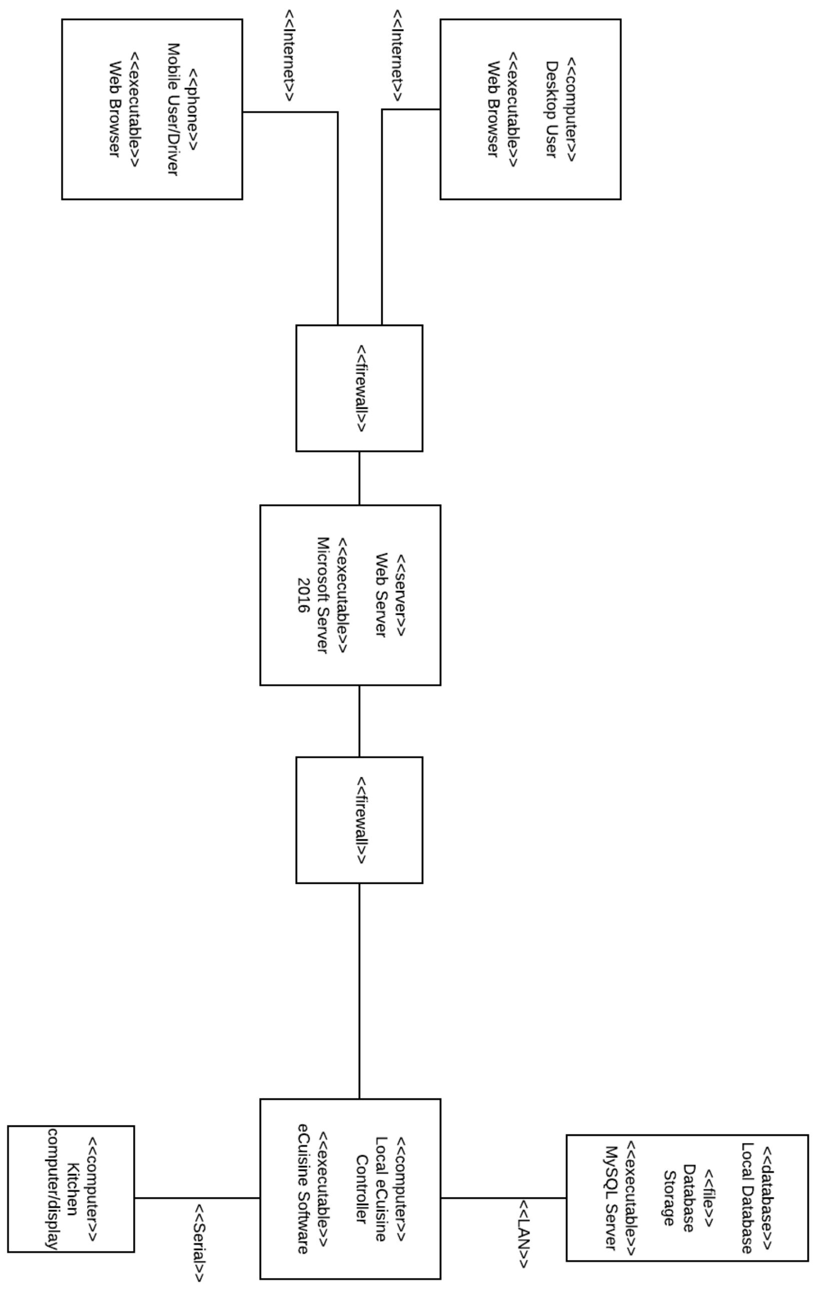
Introduction

This section contains graphics describing the architectural layout of the eCuisine system. The first diagram will be an overview of the system architecture, and the second will be a UML Deployment with Component diagram showing the location of system nodes. eCuisine will be a 3-tier client-server system, where the application logic mainly resides within a web server, the data access logic and storage lie within another server located in the kitchen, and the user interface logic is handled by the client/driver’s computer or phone. eCuisine will require the purchase of two servers to run.

Architecture Overview



Nodes and Artifacts



Hardware & Software Requirements

Hardware Requirements

Web Server

A machine capable of running Windows Server 2016. This server will host the web interface for eCuisine that users connect to. This machine may either be purchased new or paid for through a web hosting service.

Local eCuisine Computer

A machine running in the kitchen. This computer will run the local eCuisine application that communicates with the local database. A computer capable of running at least Windows 7 is recommended. Buying a brand new computer for this task is not necessary if a sufficient powerful machine is already available for use.

Database

A database running locally in the kitchen. Must be able to run a MySQL Server. 2 TB of storage space is recommended to ensure that the system will easily be able to store all user accounts and parking history for the foreseeable future.

Kitchen computer/display

A computer that can display orders and recipes for employees preparing ordered meals.

Software Requirements

Windows Server 2016

Will run on the Web Server to handle hosting the eCuisine web interface. If it is decided to host the website through a third party service, purchasing this software would become unnecessary.

MySQL Server software

Will run on the local database at the garage. The most up-to-date version of the software is recommended in other to ensure that eCuisine runs at optimal efficiency.

Linux OS

A version of the Linux operating system to run on the local database that host the MySQL software. A newer version of Ubuntu or Debian is recommended, as they are popular and stable builds.

Windows 7(or newer)

A version of the Windows operating system to run on the local eCuisine controller. The OS must be Windows-based in order to run eCuisine, with Windows 7 as the recommended earliest version to use. Any newer version of Windows will also be sufficient.

Security Plan

This section described various potential threats to the different aspects of the eCuisine architecture and what must be considered in preparation for their possibility of occurrence.

	Disruption, Destruction, Disaster					Unauthorized access		
Components	Fire	Flood	Power loss	Circuit Failure	Virus	External Intruder	Internal Intruder	Eavesdrop
Server	1,3,13	3,13	3,5	13	10,11	6,7,8	6,7	9,18
Company computers	3,13,14	13,14	15	13,14	11,14	6,7,8	6,7	18
Driver cell phones	13,14	13,14	15	13,14	12,14	6,7,8,12	6,7,12	12

1. Halon fire system in room that servers are hosted in
2. Sprinkler system throughout containing building
3. Backup server that is hosted in a remote hot site form main server which clients know how to connect to
4. Temporary emergency power for all kitchen equipment
5. Interruptible Power Supply (UPS) for all servers
6. Strong password software
7. Strong, randomly generated passwords on a rotating schedule
8. Application layer firewall
9. Network firewall
10. Virus protection software on network
11. Virus protection software as application on individual computers
12. Non-modified operating system(not “rooted” or “jailbroken”
13. Disaster recovery plan/business continuity plan
14. Temporary backup device available during recovery
15. Readily available battery-charging equipment
16. Physically locked compartment storing computer
17. Intentional, recorded distribution of computer compartment keys
18. Spyware and Malware protection software as application on individual computers
19. Tamper detection software

User Interface

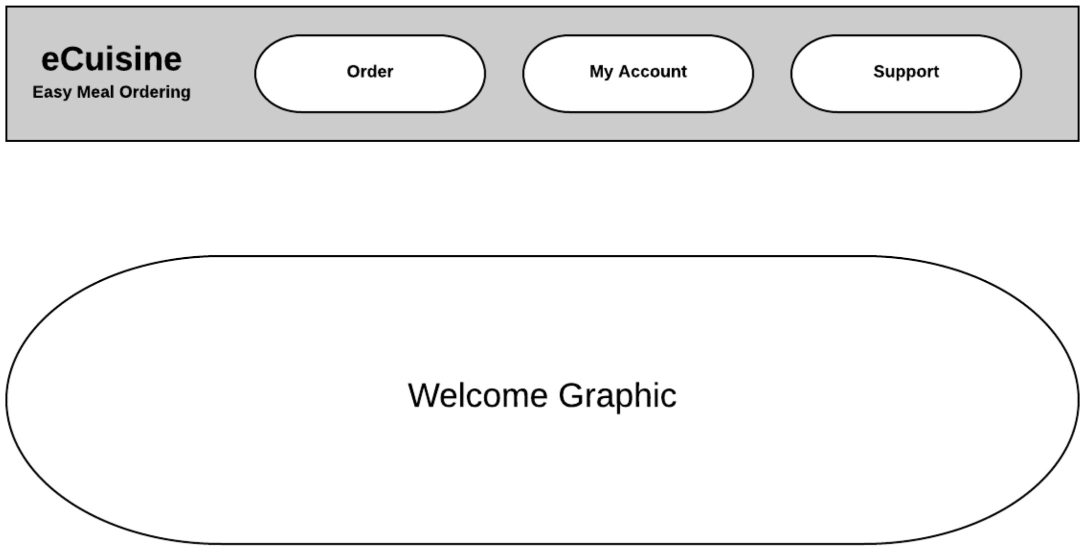
Requirements and Constraints

This section provides a guideline for how the user interface should be designed. The drawings were not designed with color schemes, font choice, or graphics in mind. They serve to show how the functionality that eCuisine provides for the users should be spread across the web interface. Since we expect many users to not be tech savvy, the interface should be designed with ease of use in mind. Buttons should be labelled, and functionality should be spread among multiple pages, grouped by the type of functionality. The following section contains many picture samples what the interface might look like, with a brief description of what functionality the view will provide for the user.

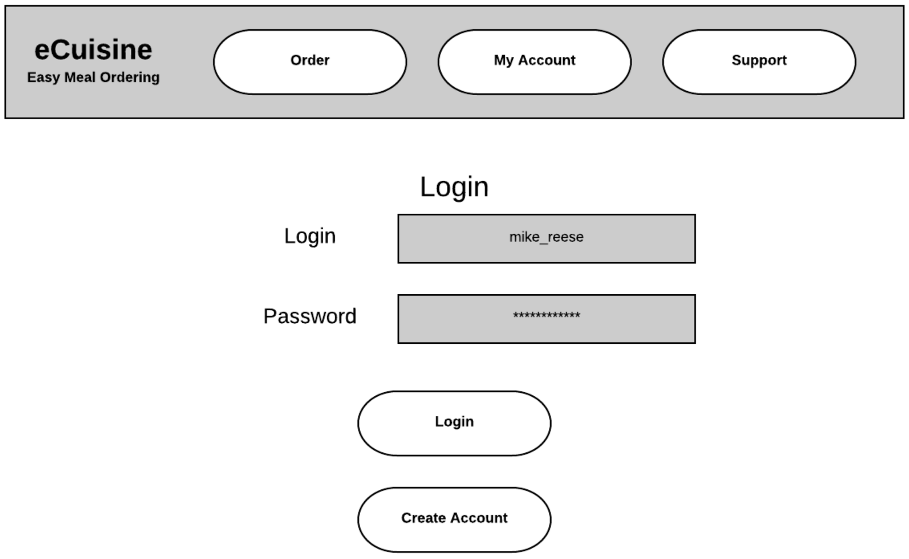
User Interaction Design: Laptop / PC

These screens represent a typical laptop or PC user will see when they navigate to the eCuisine web interface.

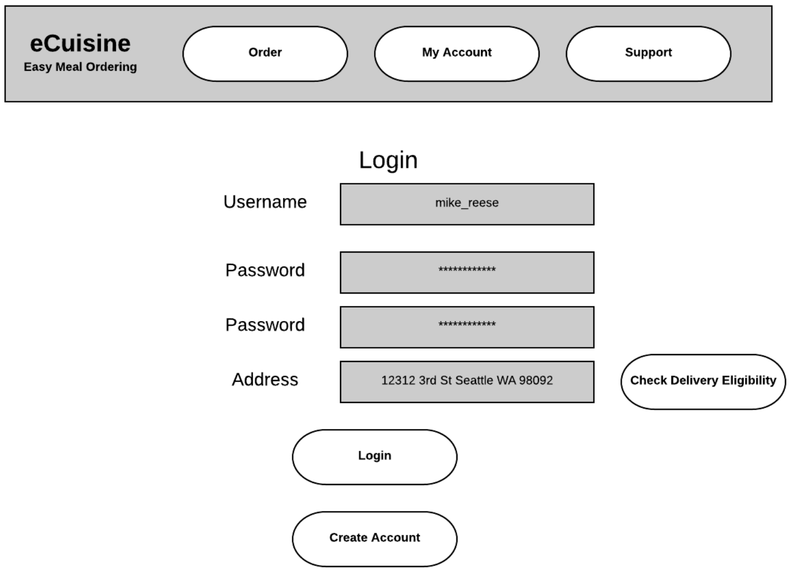
Welcome Screen: The home page for the site. Offers basic navigation to the other main areas of the site. The navigation bar at the top will remain consistent across the whole site.



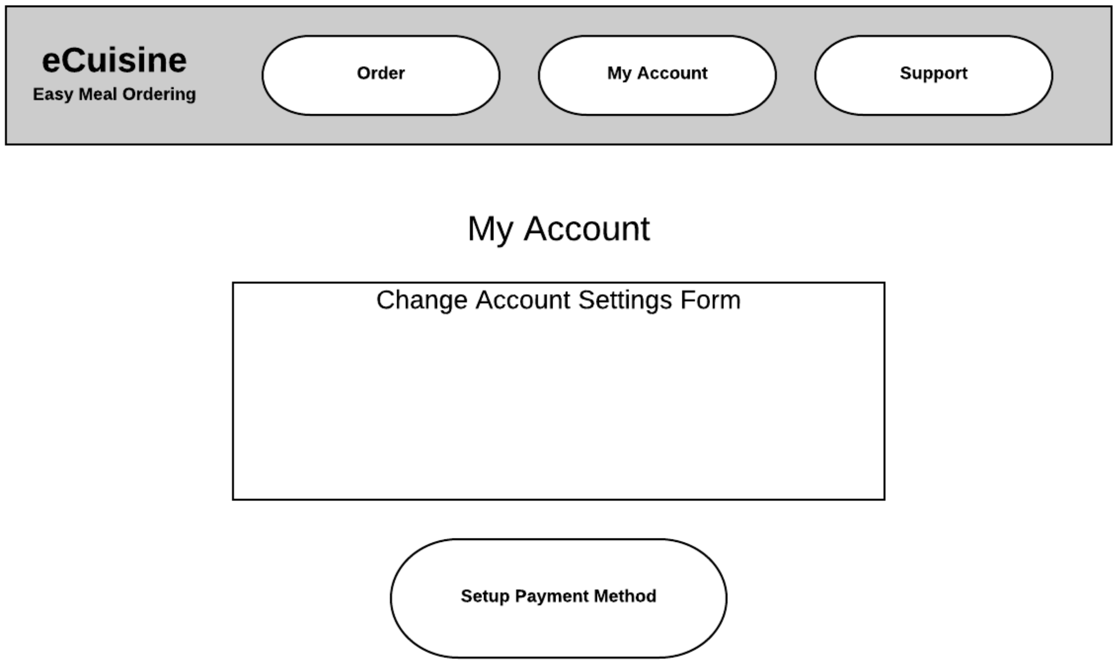
Login Screen: The view users will see before logging in to their account. Allows for users to either login or navigate to the Create Account screen.



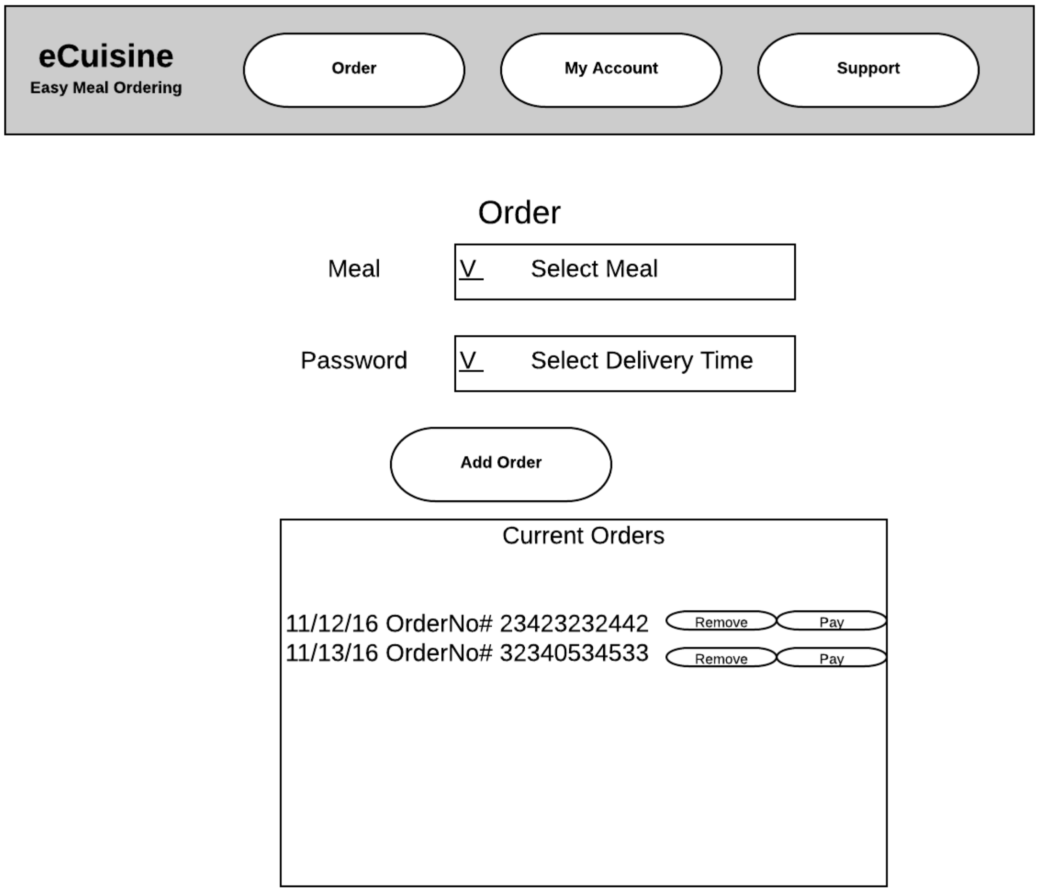
Create Account: Allows user to enter information for creating a new account



Account Page: Shows the user's account settings and allows setup for payment method.

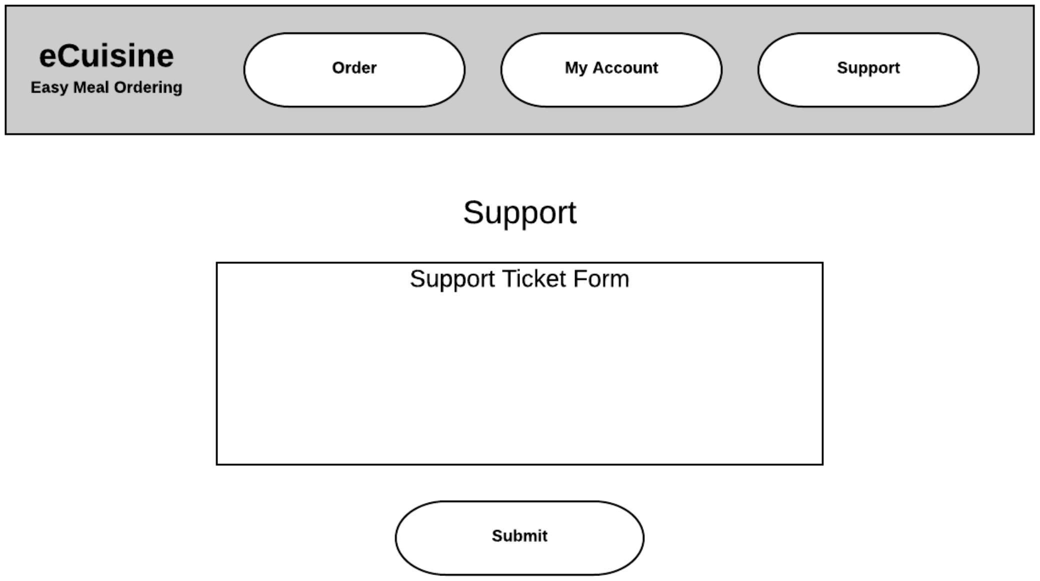


Order Screen: Allows user to add/remove orders as well as submit payment.



11/12/16 OrderNo# 23423232442
11/13/16 OrderNo# 32340534533

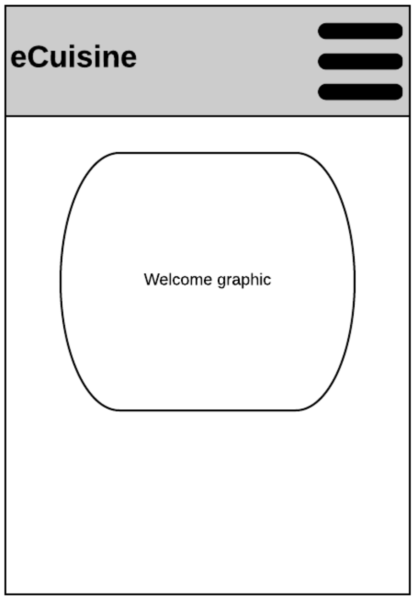
Support Screen: Allows user to submit a support ticket. May also provide contact information for Cuisine by Car.



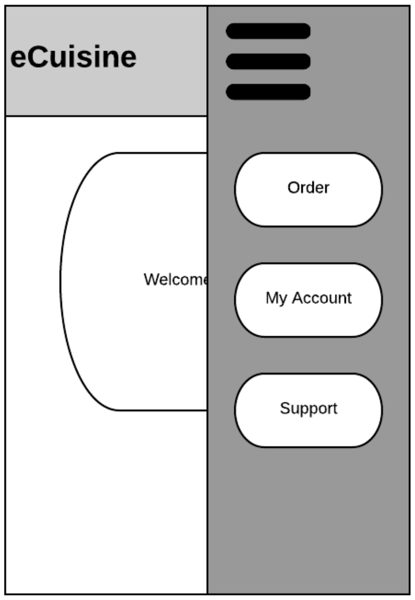
User Interaction Design: Smartphone

No new functionality is accessible to users through a smartphone, and many of the screens will look extremely similar. The next section shows the differences that a mobile user will see when they navigate to the eCuisine web interface.

Navigation bar: The main difference between the mobile and desktop views will be the navigation bar. Instead of the navigation bar always being present at the top, the user will click on the menu icon in the top right to expand out the navigation menu. The page content in each view will be the same as with the laptop view, but it will be condensed to fit on the smaller screen.



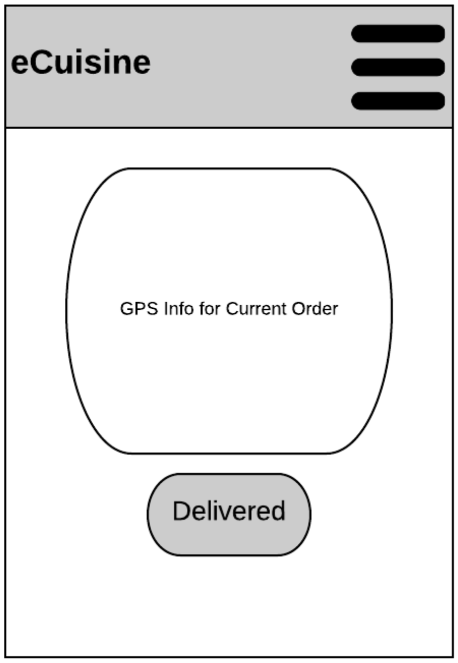
Before menu click



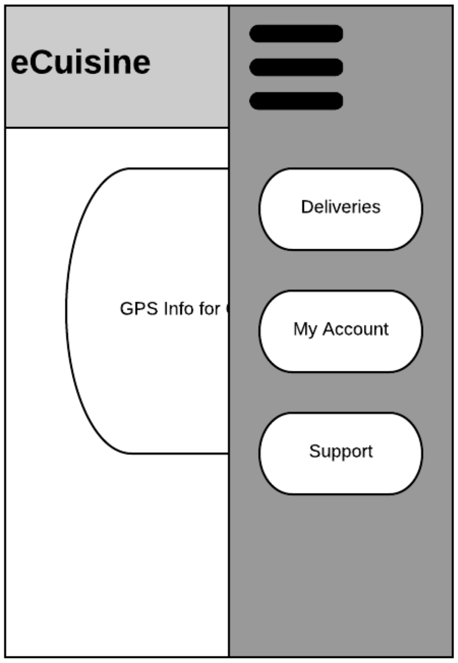
After menu click

Driver Interaction Design: Smartphone

Shows the menu and added functionality for Cuisine by Car drivers. Will allow driver to view deliveries, displaying GPS info. Driver is to click “Delivered” once order has been completed.



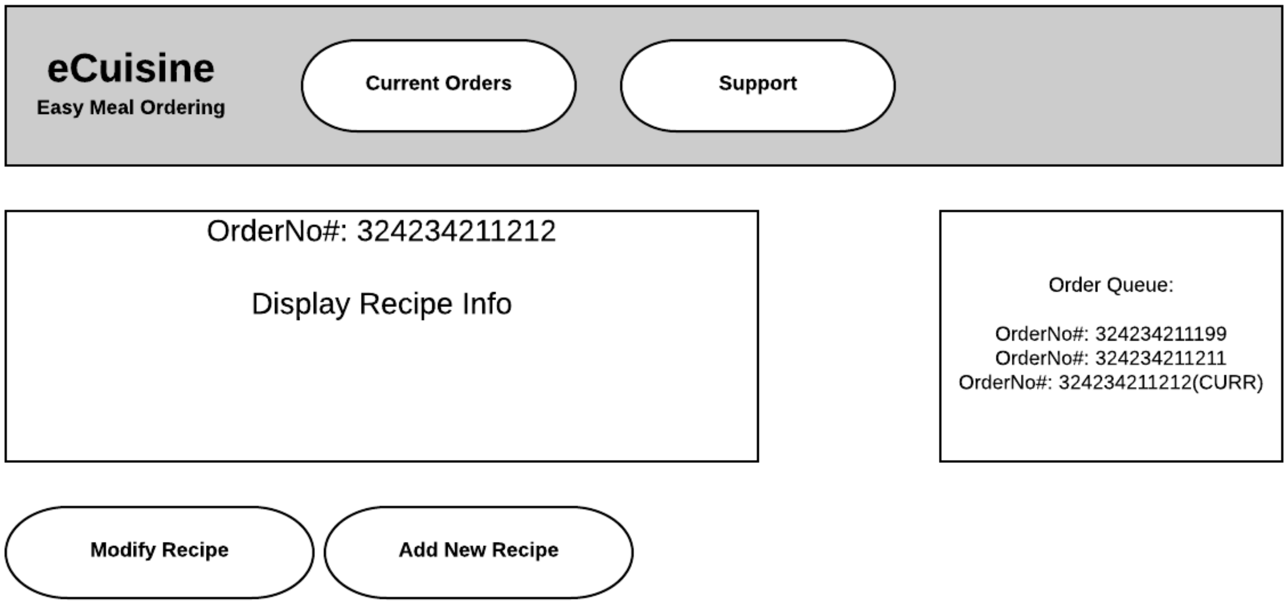
Before menu click



After menu click

Reports

Shows order information & recipes for kitchen employees. Also displays some of their limited functionality/control over system. Allows employees to modify and add recipes(assumed that they are submitted for approval to appropriate person prior).



Appendices

Bibliography

CSC 3150 Sample Book, CSC 3150. Seattle Pacific University, Seattle, Washington. 2016.

Dennis, Alan, Barbara Wixom, and David Paul Tegarden. System Analysis Design, UML Version 2.0: An Object Oriented Approach. Hoboken, NJ: John Wiley & Sons, 2012. Print