TAKEOUT – A MOBILE FOOD ORDERING SYSTEM

CMSE 322 FINAL REPORT

Team members: Oyinbobola Shalom Owojori, 128236

Abdulrazak Omeiza Yakubu, 138859

Kamshinen Rotdung, 138893

Zainab Sada, 149190

Computer Engineering Department

Eastern Mediterranean University

Spring 2016-2017

ABSTRACT

This document includes all the requirements and design documents for the app android mobile application TakeOut. TakeOut is a mobile android application that eliminates the need to talk on the phone when you want to make an order for food. It allows users to order food from local restaurants that have delivery services with ease, provided that they have an internet connection. It also allows restaurants receive order notifications and respond to them. Offering a more efficient customer to restaurant relationship.

The Requirements stage involved creating the system requirements and documenting them.

The Design stage involved thinking about the system and coming up with a design fitting the requirements and creating a UI mockup to fit the system design

The Implementation involved coding with android studio and connecting the application to the database (firebase)

We have carried out, the design, implementation and testing of the application and the documentation of the results are included in this document.

**Keywords:** TakeOut, Mobile, Android, Customer, Restaurant, Firebase

Table of Contents

[1. INTRODUCTION 1](#_Toc483554434)

[2. REQUIREMENTS ANALYSIS 1](#_Toc483554435)

[2.1 Functional Requirements 1](#_Toc483554436)

[2.2 Actor - Customer 3](#_Toc483554437)

[2.2.1 Description and Priority 3](#_Toc483554438)

[2.2.2 Detailed User Requirements 3](#_Toc483554439)

[2.3 Actor - Restaurant 7](#_Toc483554440)

[2.3.1 Description and Priority 7](#_Toc483554441)

[2.3.2 Detailed User Requirements 7](#_Toc483554442)

[2.4 Use case Diagram 10](#_Toc483554443)

[2.4.1 Customer 10](#_Toc483554444)

[2.4.2 Restaurant 11](#_Toc483554445)

[2.5 Non-Functional Requirements 11](#_Toc483554446)

[2.5.1 Reliability 11](#_Toc483554447)

[2.5.2 Performance 12](#_Toc483554448)

[2.5.3 Security 12](#_Toc483554449)

[2.5.4 Usability 13](#_Toc483554450)

[2.5.5 Safety 13](#_Toc483554451)

[2.5.6 Maintainability 13](#_Toc483554452)

[2.6 Realistic constraints 14](#_Toc483554453)

[2.7 Ethical issues 15](#_Toc483554454)

[3. DESIGN 16](#_Toc483554455)

[3.1 High level design (architectural) 16](#_Toc483554456)

[3.2 Low level design (components used) 16](#_Toc483554457)

[3.2.1 Use Cases 16](#_Toc483554458)

[4. IMPLEMENTATION 33](#_Toc483554459)

[4.1 Tools, technologies and platforms used 33](#_Toc483554460)

[4.2 Use of Software Engineering Process Steps 33](#_Toc483554461)

[4.2.1 Project planning and management 33](#_Toc483554462)

[4.2.2 PERT Analysis 46](#_Toc483554463)

[4.3 Requirements Analysis And Specification 63](#_Toc483554464)

[4.3.1 Class Diagram 63](#_Toc483554465)

[4.3.2 Entity- Relationship Diagram 63](#_Toc483554466)

[4.3.3 Sequence Diagram 64](#_Toc483554467)

[4.4 Architectural Design 65](#_Toc483554468)

[4.4.1 System Architecture Diagram 65](#_Toc483554469)

[4.5 Software Design 66](#_Toc483554470)

[4.5.1 Modular hierarchy diagram 66](#_Toc483554471)

[4.6 Coding 67](#_Toc483554472)

[4.6.1 Data Flow Diagram 67](#_Toc483554473)

[4.6.2 Food Delivery System Level 0 Data Flow Diagram 67](#_Toc483554474)

[4.6.3 System Level 1 Data Flow Diagram 67](#_Toc483554475)

[4.7 Testing Verification and Validation 68](#_Toc483554476)

[4.8 Algorithm 68](#_Toc483554477)

[4.8.1 Description of the Main Modules in the System 69](#_Toc483554478)

[4.9 Standards 71](#_Toc483554479)

[4.10 Detailed Description of the Implementation 71](#_Toc483554480)

[4.10.1 CUSTOMER USER SIDE 71](#_Toc483554481)

[4.10.2 RESTAURANT USER SIDE 83](#_Toc483554482)

[4.10.3 DETAILED DESIGN 90](#_Toc483554483)

[5. TESTING 93](#_Toc483554484)

[6. USER GUIDE OF THE SYSTEM 99](#_Toc483554485)

[6.1 For Customers: 99](#_Toc483554486)

[6.2 For Restaurants: 101](#_Toc483554487)

[7. DISCUSSION 102](#_Toc483554488)

[8. CONCLUSION 102](#_Toc483554489)

[9. REFERENCES 102](#_Toc483554490)

[10. APPENDICES 104](#_Toc483554491)

[10.1 Instructions for installing the system 104](#_Toc483554492)

[10.2 Code for the system 104](#_Toc483554493)

[10.2.1 MainActivity.java 104](#_Toc483554494)

[10.2.2 OrderFoodAdapter.java 107](#_Toc483554495)

[10.2.3 MakeOrderActivity.java 108](#_Toc483554496)

[10.2.4 ReceiveOrderActivity.java 109](#_Toc483554497)

[10.2.5 ViewOrdersFragment.java 111](#_Toc483554498)

[10.3 Other relevant material 113](#_Toc483554499)

[10.3.1 UML DIAGRAMS 113](#_Toc483554500)

LIST OF FIGURES

Figure 2.1 Customer Use Case Diagram 10

Figure 2.2 Restaurant Use Case Diagram 11

Figure 3.1 Context Diagram 16

Figure 4.1 Organizational Chart 34

Figure 4.2 Network Diagram 58

Figure 4.4 Class Diagram 63

Figure 4.5 Entity-Relationship Diagram 64

Figure 4.6 Sequence Diagram 64

Figure 4.7 System Architecture Diagram 65

Figure 4.8 Modular Hierarchy Diagram 66

Figure 4.9 Level 0 Data Flow Diagram 67

Figure 4.10 Level 1 Data Flow Diagram 67

Figure 4.11 Registration UI Screen 72

Figure 4.12 User Login Screen 73

Figure 4.13 Confirmation Screen 74

Figure 4.14 Main Screen 75

Figure 4.15 Cart Window 76

Figure 4.16 Favorites Window 77

Figure 4.17 Restaurant Screen 78

Figure 4.18 Item Window 79

Figure 4.19 Cart Window 80

Figure 4.20 Order Status Window 81

Figure 4.21 Edit Profile 82

Figure 4.22 Registration Screen 83

Figure 4.23 Build Restaurant Screen 84

Figure 4.24 Contact Information 85

Figure 4.25 Region Selection 86

Figure 4.26 Delivery Time Range 87

Figure 4.27 Upload Logo 88

Figure 4.28 Order Viewing and Status 89

Figure 6.1 Registration Screen 100

Figure 6.2 Build Restaurant Screen 101

Figure 10.1 BPMN Diagram of Overall System 114

Figure 10.2 Activity diagram of Overall System 114

LIST OF TABLES

Table 3.1 Login Detailed Use Case 16

Table 3.2 Register Detailed Use Case 17

Table 3.3 View Restaurant Detailed Use Case 18

Table 3.4 Select/Open Restaurant Detailed Use Case 19

Table 3.5 Select Menu Detailed Use Case 19

Table 3.6 Add to Cart Detailed Use Case 20

Table 3.7 Increase/Reduce Items in Cart Detailed Use Case 21

Table 3.8 View Cart Detailed Use Case 21

Table 3.9 Remove from Cart Detailed Use Case 22

Table 3.10 Customer View Orders Detailed Use Case 23

Table 3.11 View Favorite Detailed Use Case 23

Table 3.12 Add to Favorites Detailed Use Case 24

Table 3.13 Add Order to Favorites Detailed Use Case 24

Table 3.14 Send Order/Checkout Detailed Use Case 25

Table 3.15 Restaurant View Orders 26

Table 3.16 Accept Order Detailed Use Case 27

Table 3.17 View Order Details Detailed Use Case 27

Table 3.18 Delete Order Detailed Use Case 28

Table 3.19 Reject Order Detailed Use Case 28

Table 3.20 Build Restaurant Detailed Use Case 29

Table 3.21 Build Menu/Add Menu Item Detailed Use Case 30

Table 3.22 Edit Restaurant Detailed Use Case 31

Table 3.23 Logout Detailed Use Case 32

Table 4.1 Project Team Details 33

Table 4.2 Lists of Risks 34

Table 4.3 List of Milestones 43

Table 4.4 Network Earliest and Latest Time 58

Table 4.5 Path Duration 59

Table 4.6 Expected Time 59

Table 4.7 Expected Earliest and Latest Time 60

Table 4.8 Expected Earliest and Latest Time 60

Table 4.9 Project Variance 61

Table 4.10 Path Variance 61

Table 4.11 Project Z Values 62

Table 4.12 Project Crashing Time 62

Table 5.1 Customer Login Test Case 93

Table 5.2 Restaurant Login Test Case 94

Table 5.3 Customer Logout Test Case 94

Table 5.4 Restaurant Logout Test Case 95

Table 5.5 Customer Register Test Case 95

Table 5.6 Customer View Order Test Case 95

Table 5.7 Customer Edit Profile Test Case 96

Table 5.8 Restaurant Edit Profile Test Case 96

Table 5.9 Customer View Restaurants Test Case 96

Table 5.10 Customer View Menu Test Case 97

Table 5.11 Restaurant View Menu Test Case 97

Table 5.12 Customer Send Order Test Case 97

Table 5.13 Restaurant Accept Order Test Case 98

Table 5.14 Restaurant Reject Order Test Case 98

Table 5.15 Restaurant View Orders Test Case 99

Table 5.16 Reset Password Test Case 99

# INTRODUCTION

Going out to eat food from our favorite restaurants is a common thing among individuals in the world. However, when we do not want to stress ourselves and eat out because of that big test we are studying for, or some other type of deadline we absolutely cannot spare anytime to cook or go out for, we order food. Delicious food from our favorite restaurants brought right to our doorsteps and all we have to do is call. Just call and you can have all you need to eat delivered right to you.

How about the times when calling just is not an option? Like when we run out of call credit, or we love the food this particular restaurant makes but the person there doesn’t speak English well so it is always frustrating talking to them on the phone. How about the times we wake up hungry and we do not know what restaurants are open? Trial and error would just take too much time.

TakeOut is a mobile android application that eliminates the need to talk on the phone when you want to make an order for food. It allows users to order food from local restaurants that have delivery services with ease, provided that they have an internet connection. It also allows restaurants receive order notifications and respond to them. Offering a more efficient customer to restaurant relationship.

# REQUIREMENTS ANALYSIS

## Functional Requirements

REQ1: The system should enable Customers to register.

REQ2: The system should enable Customers to login.

REQ3: The system should enable Customers to logout.

REQ4: The system should enable Customers to change a forgotten password

REQ5: The system should enable customers to edit their profile.

REQ6: The system should enable customers to view their profile.

REQ7: The system should enable Customers to view Restaurants that deliver to their location.

REQ8: The system should enable Customers to view a restaurant.

REQ9: The system should enable Customers to search for menu item

REQ10: The system should enable Customers to view a menu item

REQ11: The system should enable Customers to view cart

REQ12: The system should enable Customers to add menu items to cart

REQ13: The system should enable Customers to remove menu items from cart

REQ14: The system should enable Customers to checkout / View order summary

REQ15: The system should enable Customers to add Addresses

REQ16: The system should enable Customers to Send Order

REQ17: The system should enable Customers to Receive order confirmation

REQ18: The system should enable Customers to search Restaurant names

REQ19: The system should enable Customers to set confirmation wait time

REQ20: The system should enable Customers to add restaurant and orders to favorites

REQ21: The system should enable Customers to view favorites

REQ22: The system should enable Customers to remove restaurant from favorites

REQ23: The system should enable Customers to view order history

REQ24: The system should enable Restaurants to register.

REQ25: The system should enable Restaurants to login.

REQ26: The system should enable Restaurants to edit Restaurant info

REQ27: The system should enable Restaurants to build menu

REQ28: The system should enable Restaurants to view menu

REQ29: The system should enable Restaurants to View Orders

REQ30: The system should enable restaurants to delete orders

REQ31: The system should enable Restaurants to Select/Open Orders

REQ32: The system should enable Restaurants to Accept/Reject Orders

## Actor - Customer

### Description and Priority

The customer is the user that carries out the ordering process. He/She orders food from the restaurant.

The following are features of the system associated with the customer. They describe how the customer interacts with the system.

### Detailed User Requirements

REQ1: The system should enable Customers to register.

* Collect user information (Name, Email, Password)
* Check if information is valid
* Password not empty
* Password and Password confirm the same
* Email has not been used before and right Email format.
* If information is valid, save and add user to database
* Send confirmation message to user’s email.
* When user clicks confirmation link, authorize user access.

REQ2: The system should enable Customers to login.

* They shall enter their email and password.
* The information given shall be valid.
* Access shall be granted/denied.

REQ3: The system should enable Customers to logout.

* Log user out when user selects the logout button.

REQ4: The system should enable Customers to change a forgotten password

* Collect user email
* Send change password message to user’s email
* Collect user’s new password
* Save and edit user’s password in database.

REQ5: The system should enable customers to edit their profile.

* User shall be allowed to edit old information by entering new information
* This information shall replace the old one in the database

REQ6: The system should enable customers to view their profile.

* The system shall show users their information as saved in the database.

REQ7: The system should enable Customers to view Restaurants that deliver to their location.

* The system shall use the user’s address to display a page containing restaurants that deliver to the user’s address.
* The system shall display both restaurants that are open and closed for delivery, differentiating between closed and open with a color code.

REQ8: The system should enable Customers to view a restaurant.

* The system shall display the restaurant’s menu

REQ9: The system should enable Customers to search for menu item

* The Customer should be able to enter the name of the menu item to be searched for
* The system displays all menu items that fit the Customer’s search criteria.

REQ10: The system should enable Customers to view a menu item

* The system shall display the details of the selected menu item (Name, Picture, Description, Price)
* The system shall display the option to add the displayed menu item to cart

REQ11: The system should enable Customers to view cart

* The system shall display menu items the user has added to cart from various restaurants.
* The system shall display the selected quantity of each item.
* The system shall display individual and total prices of all displayed cart items

REQ12: The system should enable Customers to add menu items to cart

* The system shall allow customers to long press to add a menu item to cart.
* The system shall also allow customers to add menu item to cart by clicking a button when the customer views the menu item’s details.

REQ13: The system should enable Customers to remove menu items from cart

* The system shall display the option of deleting a menu item from the cart.

REQ14: The system should enable Customers to checkout / View order summary

* The system shall display all cart items (Name, Short Description, and Price) and allow the user to remove the item.
* The system shall display the total price of all the selected items
* The system shall give the user the option of selecting whether to pay with cash or card.
* The system shall allow the user to select delivery address from saved address list.
* The system shall allow the user to add a delivery address to the address list.
* The system shall allow the customer to reduce or increase quantity of a menu item.
* The system shall display a button that enables user to confirm and send the order to the restaurant.

REQ15: The system should enable Customers to add Addresses

* When editing profile and during checkout, the system shall give the user the option of adding an address to the users address list.

REQ16: The system should enable Customers to Send Order

* The customer should click the button on the checkout page that enables the user to send order.
* The system shall send the order to the respective restaurants the user has selected items from.

REQ17: The system should enable Customers to Receive order confirmation

* The system shall display an ‘order accepted’ message if the restaurant accepts the user’s order
* The system shall display an ‘order rejected message’ if the restaurant rejects the user’s order. The reason for rejection shall accompany the rejection message.
* If a time of X minutes (minutes saved on the user profile for wait time) runs out since the customer sent the order and the restaurant has not accepted or rejected the order, The system displays an order rejected message with reason – ‘time ran out’. Giving the user the option to resend the order.

REQ18: The system should enable Customers to search Restaurant names

* The Customer should be able to enter the name of the restaurant to be searched for
* The system displays all restaurants in the customer’s general location that fit the Customer’s search criteria.

REQ19: The system should enable Customers to set confirmation wait time

* The customer shall input the desired wait time before order is rejected by default due to time running out.
* The system shall allow restaurants to accept or reject the customer’s order in this period. After which a default reject message is sent to the customer with reason – ‘time ran out’. Giving the user the option to resend the order.

REQ20: The system should enable Customers to add restaurant and orders to favorites

REQ21: The system should enable Customers to view favorites

REQ22: The system should enable Customers to remove restaurant from favorites

REQ23: The system should enable Customers to view order history

## Actor - Restaurant

### Description and Priority

The restaurant is the person who makes and delivers the food. This actor receives orders from the customer.

The following are features of the system associated with the restaurant. They describe how the restaurant interacts with the system.

### Detailed User Requirements

REQ24: The system should enable Restaurants to register.

* The system collects restaurant information (Restaurant name, Email, Password)
* Check if information is valid

Password not empty

* Password and Password confirm the same
* Email has not been used before and right Email format.
* If information is valid, save and add restaurant to database
* Send confirmation message to restaurant’s email.
* Send a special access code to the restaurant’s email.
* When restaurant clicks confirmation link, prompt for the access code.
* If access code is authorized, grant the restaurant access.

REQ25: The system should enable Restaurants to login.

* They shall enter their email, access code and password.
* The information given shall be valid.
* Access shall be granted/denied.

REQ26: The system should enable Restaurants to edit Restaurant info

* On first login, the system displays this page to fill restaurant info.
* The system shall allow the restaurant to fill in contact info (Phone number, Address, State, Region, Country)
* The system shall allow the restaurant to pick delivery regions.
* The system shall allow the restaurants to set delivery time
* The system shall allow the restaurants to upload logo

REQ27: The system should enable Restaurants to build menu

* The system shall allow the restaurant to input menu item (name, description, price, picture)
* The system shall allow the restaurant to add optional menu items (name, picture, price)

REQ28: The system should enable Restaurants to view menu

* The system shall display the saved restaurant menu as it will be displayed to the customers.

REQ29: The system should enable Restaurants to View Orders

* The system shall display all current orders becoming less opaque (disappearing) as the time for the order confirmation runs out.

REQ30: The system should enable restaurants to delete orders

* The system shall display accepted or rejected offers with a delete button
* The system shall delete orders every hour regardless of whether the restaurant deletes it or not

REQ31: The system should enable Restaurants to Select/Open Orders

* The system shall display a page with order details (Items, Optional Ingredients, Order address, Price, Customer telephone number, etc.)

REQ32: The system should enable Restaurants to Accept/Reject Orders

* The system shall display all current orders with an accept or reject button

## Use case Diagram

### Customer

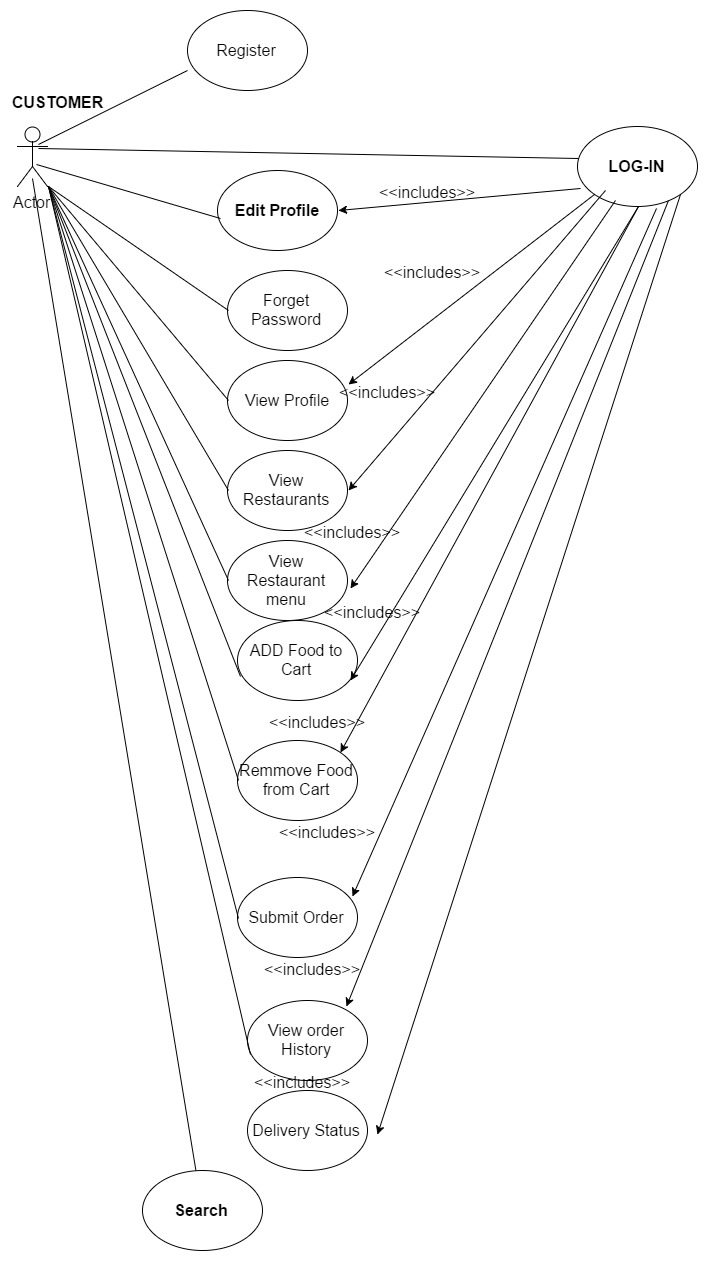


Figure . Customer Use Case Diagram

### Restaurant

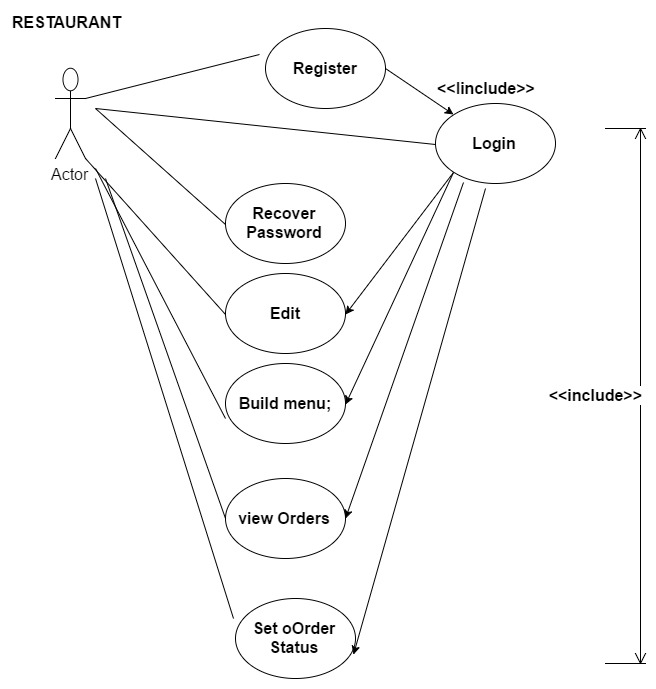


Figure . Restaurant Use Case Diagram

## Non-Functional Requirements

### Reliability

* The system should be available when requested for service by users: The system should work 24/7, it should always be up and running so that whenever the user wants to use it, it is available.
* The system should have a very low failure rate: The failure rate should be kept as minimal as possible, preferably less than 0.01.

### Performance

* The system must have a good response time.
* The load time for the user interface should take less than two seconds.
* The log in information should be verified within five seconds.
* Queries shall return results within five seconds.
* The system should be able to achieve a lot in a specified amount of time.
* The system should be able to withstand a heavy workload.
* It should be able to respond to multiple numbers of people at the same time.
* The system must run error free while operating with a huge set of data.
* The system should be precise and accurate when dealing with data.
* The system’s error rate should be minimal

### Security

* All external communications between the system’s data server and clients must  
  be encrypted
* To ensure that the system is secure access to the various subsystems will be  
  protected by a user log in screen and requires a user name and password.
* The System admin shall provide an access code to restaurants upon registration after successful verification of their restaurant.
* All system data must be backed up every 24 hours and the backup copies stored  
  in a secure location, which is not in the same building as the system: This is  
  done to avoid loss of information in case of system crash. The system data  
  should be stored in storage device e.g. hard drive, CD, Flash drive or it could be  
  stored in files.

### Usability

* The system should include a questions page for users to ask questions or complaints about the application
* The system should have a frequently answered questions page to tackle major frequent issues that user’s might have with using the application.
* The system should have a well formed, easy to use graphical user interface
* The system should be user friendly
* The system must be easy to learn for both novices and users with experience from similar systems
* The system must be efficient for the frequent use
* The system must be easy to remember for the casual user
* The user must understand what the system does
* The user must feel satisfied with the system

### Safety

* The system should maintain a good backup: Maintaining backups ensures that the system’s database is secure, which means that in case of an emergency or accident the system can be easily restored.

### Maintainability

* The system should be easy to maintain.
* In order for the system to be easy to maintain, it is done with an object-oriented language, which is easy to maintain.
* Maintenance of the system should be cost efficient
* Maintenance of the system should be less frequent
* The system should easily adapt to changes made
* The system should be able to detect the location of the user

## Realistic constraints

1. The project is economically Feasible, it was proven to be of ample benefits to the Northern Cyprus economy:

* It reduces ordering time and costs.
* It increases the general revenue of the food sector.

1. The project is environmentally safe:

* Producing or using it doesn`t consume a lot of power. It only requires a

sufficiently charged system.

* Using the system doesn`t lead to pollution of the environment due to the

fact that the system is used in the confinements of people`s houses or

offices.

1. The project is socially Feasible:

* It has made the food ordering procedure better and more convenient.
* It will encourage more customers to order food.
* It can be used by people of all ages provided they are computer literate.

1. There are no political constraints therefore it is politically feasible.
2. There are no ethical constraints: The codes were self-written so there are no copyright infringements.
3. Health and safety: Using this product is very safe and it doesn`t endanger the

health and safety of people or society. It helps reduce the risk of allergic reactions since the customer can be more specific with the order at no risk of mistake.

1. Manufacturability: The system can be manufactured using reasonable amount of available resources.
2. Sustainability: The product be used over the long term provided that it is

properly maintained.

## Ethical issues

There are no major ethical issues with using this system as all restaurants will be officially licensed and undergo a verification process before they are given an access code for system use.

# DESIGN

## High level design (architectural)

A context Diagram showing the system and its main modules, including the relationship between them is shown below. The context diagram shows the most basic functions of the system. The Customer and Restaurant are external entities which interact with the system. The customer places an order, which is sent to the restaurant by the system. The restaurant responds by either accepting the order or declining it. This response is sent to the customer by the system.



Figure . Context Diagram

## Low level design (components used)

### Use Cases

Table . Login Detailed Use Case

|  |  |  |  |
| --- | --- | --- | --- |
| Use Case UC-1: | | | Login |
| Related  Requirements: | | | REQ 2, 25 |
| Initiating Actor: | | | Customer, Restaurant |
| Actor’s Goal: | | | To log into account |
| Participating  Actors: | | | Customer, Restaurant |
| Preconditions: | | | The system displays the login page for customer or restaurant to enter login details – (Email and Password for customer. Restaurant name, Access code and Password for restaurant). |
| Post conditions: | | | Customer: The system should redirect user to the display restaurants page.  Restaurant: On first login, the system should redirect user to the build restaurant page. For subsequent logins, the system should redirect user to the view orders page. |
| Flow of Events for Main Success Scenario: | | | |
|  | 1. | Customer/Restaurant opens application and clicks customer/restaurant. | |
|  | 2. | System shows user a menu to enter login details. | |
|  | 3. | User enters the information and clicks on login button. | |
|  | 4. | System redirects user to default page after login successful. For customers, display restaurants. For Restaurants, view orders. | |
| Flow of Events for Extensions (Alternate Scenarios): User enter invalid username or password | | | |
|  | 1. | System shows a warning message that the login details entered were wrong and asks the user to enter information again. | |

Table . Register Detailed Use Case

|  |  |  |  |
| --- | --- | --- | --- |
| Use Case UC-2: | | | Register |
| Related  Requirements: | | | REQ 1, 24 |
| Initiating Actor: | | | Customer, Restaurant |
| Actor’s Goal: | | | To register for an account |
| Participating  Actors: | | | Customer, Restaurant |
| Preconditions: | | | The system displays the register page for customer or restaurant to enter details. |
| Post conditions: | | | The system should redirect user to the registration confirmed page |
| Flow of Events for Main Success Scenario: | | | |
|  | 1. | Customer/Restaurant opens login page and clicks register | |
|  | 2. | System shows user a menu to enter details. – (Name, Email, Password and Confirm Password). | |
|  | 3. | User enters the information and clicks on register button. | |
|  | 4. | System redirects user to the registration confirmed page. Telling the user to check their mail for further instructions. | |
|  | 5. | Customer: System sends message to user’s email with a link to confirm registration  Restaurant: System sends message to user’s email with a link to confirm registration and a special access code for security purposes | |
|  | 6. | User clicks confirm registration link | |
|  | 7. | System displays registration successful message. Allowing user to login. | |
| Flow of Events for Extensions (Alternate Scenarios): User enter invalid username or password | | | |
|  | 1. | System shows a warning message that username or password entered was wrong and asked for entering information again. | |

Table . View Restaurant Detailed Use Case

|  |  |  |  |
| --- | --- | --- | --- |
| Use Case UC-3: | | | View restaurants |
| Related  Requirements: | | | REQ 7 |
| Initiating Actor: | | | Customer |
| Actor’s Goal: | | | To view restaurants close to user’s location |
| Participating  Actors: | | | Customer |
| Preconditions: | | | Customer must be logged in |
| Post conditions: | | |  |
| Flow of Events for Main Success Scenario: | | | |
|  | 1. | Customer/Restaurant logs in or clicks ‘restaurants’ link. | |
|  | 2. | System shows the user a page containing restaurants that can deliver to the user’s location. The page displays restaurant names, short descriptions, if the restaurants are open or closed. | |
| Flow of Events for Extensions (Alternate Scenarios): None | | | |

Table . Select/Open Restaurant Detailed Use Case

|  |  |  |  |
| --- | --- | --- | --- |
| Use Case UC-4: | | | Select/Open Restaurant |
| Related  Requirements: | | | REQ 8 |
| Initiating Actor: | | | Customer |
| Actor’s Goal: | | | To view a restaurant’s menu |
| Participating  Actors: | | | Customer |
| Preconditions: | | | The customer taps on a restaurant from the restaurants page. |
| Post conditions: | | | None |
| Flow of Events for Main Success Scenario: | | | |
|  | 1. | Customer/Restaurant opens restaurant page and selects a restaurant. | |
|  | 2. | System shows user a page containing the restaurant’s name and menu. | |
| Flow of Events for Extensions (Alternate Scenarios): None | | | |

Table . Select Menu Detailed Use Case

|  |  |  |  |
| --- | --- | --- | --- |
| Use Case UC-5: | | | Select menu item |
| Related  Requirements: | | | REQ 10 |
| Initiating Actor: | | | Customer |
| Actor’s Goal: | | | To view details about an item on a restaurant’s menu. |
| Participating  Actors: | | | Customer |
| Preconditions: | | | The user clicks on a restaurant’s menu item. |
| Post conditions: | | | None |
| Flow of Events for Main Success Scenario: | | | |
|  | 1. | Customer selects a menu item | |
|  | 2. | System shows user a page with the details of the menu item. İncluding the option to add the selected item to cart. | |
| Flow of Events for Extensions (Alternate Scenarios): None | | | |

Table . Add to Cart Detailed Use Case

|  |  |  |  |
| --- | --- | --- | --- |
| Use Case UC-6: | | | Add item to Cart |
| Related  Requirements: | | | REQ 12 |
| Initiating Actor: | | | Customer |
| Actor’s Goal: | | | To add a menu item to cart |
| Participating  Actors: | | | Customer |
| Preconditions: | | | The user selects the ‘add to cart’ option on the view menu item page.  The user long presses a menu item from the view restaurant menu page. |
| Post conditions: | | | None |
| Flow of Events for Main Success Scenario: | | | |
|  | 1. | Customer selects the ‘add to cart’ option on the view menu item page  Customer long presses a menu item from the view restaurant menu page. | |
|  | 2. | System displays an ‘added to cart’ message. | |
| Flow of Events for Extensions (Alternate Scenarios): None | | | |

Table . Increase/Reduce Items in Cart Detailed Use Case

|  |  |  |  |
| --- | --- | --- | --- |
| Use Case UC-7: | | | Increase/Reduce item quantity in cart |
| Related  Requirements: | | | REQ 11 |
| Initiating Actor: | | | Customer |
| Actor’s Goal: | | | To change quantity of items selected for order |
| Participating  Actors: | | | Customer |
| Preconditions: | | | The user opens the cart |
| Post conditions: | | | None |
| Flow of Events for Main Success Scenario: | | | |
|  | 1. | Customer selects the cart on the application | |
|  | 2. | System displays the cart containing all items selected to be ordered, giving the user the option to send order | |
|  | 3. | Customer selects either the + or – buttons to increase or reduce item quantity respectively | |
|  | 4. | System responds to user’s choice: increase quantity by 1 if user selects ‘+’, reduce quantity by 1 if user selects ‘– ‘ | |
| Flow of Events for Extensions (Alternate Scenarios): None | | | |

Table . View Cart Detailed Use Case

|  |  |  |  |
| --- | --- | --- | --- |
| Use Case UC-8: | | | View Cart |
| Related  Requirements: | | | REQ 11 |
| Initiating Actor: | | | Customer |
| Actor’s Goal: | | | To view items selected for order |
| Participating  Actors: | | | Customer |
| Preconditions: | | | The user opens the cart link |
| Post conditions: | | | None |
| Flow of Events for Main Success Scenario: | | | |
|  | 1. | Customer selects the cart link on the application | |
|  | 2. | System displays the cart containing all items selected to be ordered, giving the user the option to send order | |
| Flow of Events for Extensions (Alternate Scenarios): None | | | |

Table . Remove from Cart Detailed Use Case

|  |  |  |  |
| --- | --- | --- | --- |
| Use Case UC-9: | | | Remove item from cart |
| Related  Requirements: | | | REQ 13 |
| Initiating Actor: | | | Customer |
| Actor’s Goal: | | | To remove a menu item from cart |
| Participating  Actors: | | | Customer |
| Preconditions: | | | The user opens the cart menu.  The user selects the ‘x’ beside the menu item wished to be deleted on the cart. |
| Post conditions: | | | None |
| Flow of Events for Main Success Scenario: | | | |
|  | 1. | Customer views cart. | |
|  | 2. | Customer selects the ‘x’ beside the menu item wished to be deleted on the cart. | |
|  | 2. | System deletes item from cart and displays a ‘removed’ message. | |
| Flow of Events for Extensions (Alternate Scenarios): None | | | |

Table . Customer View Orders Detailed Use Case

|  |  |  |  |
| --- | --- | --- | --- |
| Use Case UC-10: | | | Customer View Orders |
| Related  Requirements: | | | REQ 23 |
| Initiating Actor: | | | Customer |
| Actor’s Goal: | | | To view previously sent orders |
| Participating  Actors: | | | Customer |
| Preconditions: | | | The user is logged in |
| Post conditions: | | | None |
| Flow of Events for Main Success Scenario: | | | |
|  | 1. | Customer selects view orders | |
|  | 2. | System displays page showing history of previously sent orders | |
|  | 3. | Customer selects a previously made order | |
|  | 4. | System shows user details of the order and option to reorder items | |
| Flow of Events for Extensions (Alternate Scenarios): None | | | |

Table . View Favorite Detailed Use Case

|  |  |  |  |
| --- | --- | --- | --- |
| Use Case UC-11: | | | View Favorites |
| Related  Requirements: | | | REQ 21 |
| Initiating Actor: | | | Customer |
| Actor’s Goal: | | | To view previously saved favorite restaurants and orders |
| Participating  Actors: | | | Customer |
| Preconditions: | | | The user opens favorites |
| Post conditions: | | | None |
| Flow of Events for Main Success Scenario: | | | |
|  | 1. | Customer selects favorites on the application | |
|  | 2. | System displays the favorites containing the customers previously saved restaurants and orders. | |
| Flow of Events for Extensions (Alternate Scenarios): None | | | |

Table . Add to Favorites Detailed Use Case

|  |  |  |  |
| --- | --- | --- | --- |
| Use Case UC-12: | | | Add restaurant to favorites |
| Related  Requirements: | | | REQ 20 |
| Initiating Actor: | | | Customer |
| Actor’s Goal: | | | To add a liked restaurant to favorites |
| Participating  Actors: | | | Customer |
| Preconditions: | | | The user is viewing the restaurant’s menu |
| Post conditions: | | | None |
| Flow of Events for Main Success Scenario: | | | |
|  | 1. | Customer selects add to favorites on the restaurant’s menu page | |
|  | 2. | System adds restaurant to favorite restaurants list and displays ‘added to favorites’ | |
| Flow of Events for Extensions (Alternate Scenarios): None | | | |

Table . Add Order to Favorites Detailed Use Case

|  |  |  |  |
| --- | --- | --- | --- |
| Use Case UC-13: | | | Add order to favorites |
| Related Requirements: | | | REQ 20 |
| Initiating Actor: | | | Customer |
| Actor’s Goal: | | | To add a specific order to favorites |
| Participating Actors: | | | Customer |
| Preconditions: | | | 1. The user is on the place order page 2. The user is on the view orders page |
| Post conditions: | | | None |
| Flow of Events for Main Success Scenario: | | | |
|  | 1. | Customer selects add order to favorites on the place order page or when viewing a previous order on the view orders page | |
|  | 2. | System adds order to favorite orders list and displays ‘added to favorites’ | |
| Flow of Events for Extensions (Alternate Scenarios): None | | | |

Table . Send Order/Checkout Detailed Use Case

|  |  |  |  |
| --- | --- | --- | --- |
| Use Case UC-14: | | | Send order/Checkout  \*/8/89  3 |
| Related Requirements: | | | REQ 16 |
| Initiating Actor: | | | Customer |
| Actor’s Goal: | | | To send selected items to restaurant as order |
| Participating Actors: | | | Customer, Restaurant |
| Preconditions: | | | Customer clicks the ‘send order’ button on the cart |
| Post conditions: | | | None |
| Flow of Events for Main Success Scenario: | | | |
|  | 1. | Customer views cart and selects the send order button on the cart. | |
|  | 2. | System displays page containing cart items, price and other options. Options include: Pick address, Add new address, pay with cash/card, change quantity, delete item, cancel order, set wait time. | |
|  | 3. | Customer adjusts order to preferences. And sets amount of time to wait for confirmation | |
|  | 4. | Customer selects ‘takeOut’ at the bottom of the page. | |
|  | 5. | System sends order to restaurant. | |
|  | 6. | Restaurant responds within specified wait time | |
|  | 7. | The system sends the customer a message with the restaurant’s response (accepted or rejected) | |
| Flow of Events for Extensions (Alternate Scenarios): User selects cancel order | | | |
|  | 1. | System shows an alert asking the user if their sure they want to cancel the order. | |
|  | 2. | Customer selects yes | |
|  | 3. | System cancels order and returns user to main page | |
| Flow of Events for Extensions (Alternate Scenarios): Restaurant does not accept/reject order within specified wait time | | | |
|  | 1. | System sends a message to customer ‘order rejected’ with reason – ‘time ran out’ and an option to resend order. | |

Table . Restaurant View Orders

|  |  |  |  |
| --- | --- | --- | --- |
| Use Case UC-15: | | | Restaurant View Orders |
| Related Requirements: | | | REQ 28 |
| Initiating Actor: | | | Restaurant user |
| Actor’s Goal: | | | To view orders sent from customers |
| Participating  Actors: | | | Restaurant user |
| Preconditions: | | | The user is logged in |
| Post conditions: | | | None |
| Flow of Events for Main Success Scenario: | | | |
|  | 1. | Restaurant selects view orders | |
|  | 2. | System displays page showing current orders sent from customers with options to accept or reject. With each individual order reducing in opacity according to time left as specified by customer | |
| Flow of Events for Extensions (Alternate Scenarios): None | | | |

Table . Accept Order Detailed Use Case

|  |  |  |  |
| --- | --- | --- | --- |
| Use Case UC-16: | | | Accept order |
| Related Requirements: | | | REQ 31 |
| Initiating Actor: | | | Restaurant |
| Actor’s Goal: | | | To accept order |
| Participating  Actors: | | | Restaurant, Customer |
| Preconditions: | | | The user is logged in |
| Post conditions: | | | None |
| Flow of Events for Main Success Scenario: | | | |
|  | 1. | Restaurant selects accept order | |
|  | 2. | System changes order view options to view order and delete order | |
|  | 3. | System sends acceptance message to customer | |
| Flow of Events for Extensions (Alternate Scenarios): None | | | |

Table . View Order Details Detailed Use Case

|  |  |  |  |
| --- | --- | --- | --- |
| Use Case UC-17: | | | View order details |
| Related Requirements: | | | REQ 30 |
| Initiating Actor: | | | Restaurant |
| Actor’s Goal: | | | To view an order details |
| Participating  Actors: | | | Restaurant |
| Preconditions: | | | The user is logged in |
| Post conditions: | | | None |
| Flow of Events for Main Success Scenario: | | | |
|  | 1. | Restaurant selects an order | |
|  | 2. | System displays page showing details of order and options to accept or reject order | |
| Flow of Events for Extensions (Alternate Scenarios): None | | | |

Table . Delete Order Detailed Use Case

|  |  |  |  |
| --- | --- | --- | --- |
| Use Case UC-18: | | | Delete order |
| Related Requirements: | | | REQ 29 |
| Initiating Actor: | | | Restaurant |
| Actor’s Goal: | | | To delete an order |
| Participating  Actors: | | | Restaurant |
| Preconditions: | | | The user is logged in |
| Post conditions: | | | None |
| Flow of Events for Main Success Scenario: | | | |
|  | 1. | Restaurant selects delete order | |
|  | 2. | System displays page with orders except the deleted order and displays message ‘order deleted’ | |
| Flow of Events for Extensions (Alternate Scenarios): None | | | |

Table . Reject Order Detailed Use Case

|  |  |  |  |
| --- | --- | --- | --- |
| Use Case UC-19: | | | Reject order |
| Related Requirements: | | | REQ 31 |
| Initiating Actor: | | | Restaurant |
| Actor’s Goal: | | | To reject order |
| Participating  Actors: | | | Restaurant, Customer |
| Preconditions: | | | The user is logged in |
| Post conditions: | | | None |
| Flow of Events for Main Success Scenario: | | | |
|  | 1. | Restaurant user selects reject order | |
|  | 2. | System displays page showing reject reasons | |
|  | 3. | Restaurant user selects a reason and selects reject order | |
|  | 4. | System displays ‘order rejected’ and sends reject message with reason to customer | |
| Flow of Events for Extensions (Alternate Scenarios): None | | | |

Table . Build Restaurant Detailed Use Case

|  |  |  |  |
| --- | --- | --- | --- |
| Use Case UC-20: | | | Build Restaurant |
| Related Requirements: | | | REQ 26 |
| Initiating Actor: | | | Restaurant |
| Actor’s Goal: | | | To fill details of restaurant on the application |
| Participating  Actors: | | | Restaurant |
| Preconditions: | | | 1. First time login for Restaurant |
| Post conditions: | | | None |
| Flow of Events for Main Success Scenario: | | | |
|  | 1. | Restaurant user logs in | |
|  | 2. | System displays message ‘Build your restaurant’ and prompts user to tap to continue to next step | |
|  | 3. | Restaurant user taps the page | |
|  | 4. | System displays page to input contact info (Phone number, Address Line 1, Address line 2, Address line 3, State, Region, Country) and prompts user to tap to continue to next step | |
|  | 5. | Restaurant user fills in info and taps to continue | |
|  | 6. | System displays page of selectable regions in the specified state and country of previously filled info asking the user to select regions food can be delivered to | |
|  | 7. | Restaurant user selects regions and taps to continue | |
|  | 8. | System displays page to set delivery time | |
|  | 9. | Restaurant user fills in delivery time and taps to continue | |
|  | 10. | System displays page to upload logo | |
|  | 11. | Restaurant user uploads logo and taps to continue | |
|  | 12. | System displays ‘Build menu’. Prompting the user to continue to the next step of building the restaurant’s digital menu that customers will see. | |
|  | 13. | Restaurant user taps screen to build menu | |
| Flow of Events for Extensions (Alternate Scenarios): None | | | |

Table . Build Menu/Add Menu Item Detailed Use Case

|  |  |  |  |
| --- | --- | --- | --- |
| Use Case UC-21: | | | Build Menu/Add menu item |
| Related Requirements: | | | REQ 27 |
| Initiating Actor: | | | Restaurant |
| Actor’s Goal: | | | To build the restaurant’s menu on the application |
| Participating  Actors: | | | Restaurant |
| Preconditions: | | | Restaurant is logged in |
| Post conditions: | | | None |
| Flow of Events for Main Success Scenario: | | | |
|  | 1. | Restaurant user selects build menu/edit menu | |
|  | 2. | System displays page with options to add a menu item (Name, description, picture, price, add optional menu item, category). | |
|  | 3. | Restaurant user taps the + button to add the item to the menu. | |
|  | 4. | System displays page with options to either view menu or add another item to menu | |
|  | 5. | Restaurant user selects to add another item to menu | |
|  | 6. | Return to 2. | |
| Flow of Events for Extensions (Alternate Scenarios): User selects view menu | | | |
|  | 6. | System displays built menu | |

Table . Edit Restaurant Detailed Use Case

|  |  |  |  |
| --- | --- | --- | --- |
| Use Case UC-22: | | | Edit Restaurant |
| Related Requirements: | | | REQ 26 |
| Initiating Actor: | | | Restaurant |
| Actor’s Goal: | | | To edit details of restaurant on the application |
| Participating  Actors: | | | Restaurant |
| Preconditions: | | | Restaurant has to be logged in |
| Post conditions: | | | None |
| Flow of Events for Main Success Scenario: | | | |
|  | 1. | Restaurant user selects edit restaurant | |
|  | 2. | System displays message ‘Edit your restaurant’ and prompts user to tap to continue to next step | |
|  | 3. | Restaurant user taps the page | |
|  | 4. | System displays page with previously saved contact details and prompts user to tap to continue to next step | |
|  | 5. | Restaurant user edits info and taps to continue | |
|  | 6. | System displays page of selected delivery regions | |
|  | 7. | Restaurant user edits delivery regions and taps to continue | |
|  | 8. | System displays page with saved delivery time | |
|  | 9. | Restaurant user changes delivery time and taps to continue | |
|  | 10. | System displays ‘Details saved’ | |
|  | 11. | Restaurant user taps screen to return to view orders page | |
| Flow of Events for Extensions (Alternate Scenarios): None | | | |

Table . Logout Detailed Use Case

|  |  |  |  |
| --- | --- | --- | --- |
| Use Case UC-23: | | | Logout |
| Related Requirements: | | | REQ 3 |
| Initiating Actor: | | | Customer |
| Actor’s Goal: | | | To logout of account |
| Participating  Actors: | | | Customer |
| Preconditions: | | | The user is logged in |
| Post conditions: | | | None |
| Flow of Events for Main Success Scenario: | | | |
|  | 1. | Customer selects logout | |
|  | 2. | System displays prompt asking user ‘do you really want to log out?’ | |
|  | 3. | Customer selects yes | |
|  | 4. | System logs user out and redirects to login page | |
| Flow of Events for Extensions (Alternate Scenarios): None | | | |

# IMPLEMENTATION

## Tools, technologies and platforms used

Programming Language: Java

Programming environment: Windows 10, Android Studio

Documentation: Visual Paradigm, Microsoft Project, Microsoft Word, Atomic.io

Database: Firebase

## Use of Software Engineering Process Steps

### Project planning and management

#### Project organization activities

Table . Project Team Details

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| Personnel Name | Title | ID No | Education Status | Graduation Date | Date of Starting Work | Idea Owner |
| Oyinbobola Owojori | Leader and Analyzer, User Interface Designer | 128236 | Undergraduate | Spring 2017 | 15/03/2017 | Yes |
| Kamshinen Rotdung | System Developer and tester | 132034 | Undergraduate | February 2018 | 24/03/2017 |  |
| Abdulrazak Omeiza Yakubu | System developer | 138859 | Undergraduate | Spring 2017 | 24/03/2017 |  |
| Zainab Sada | Database Designer | 149190 | Undergraduate | Spring 2017 | 24/03/2017 |  |

#### Roles and responsibilities activities

This is shown by use of an organizational chart given below:

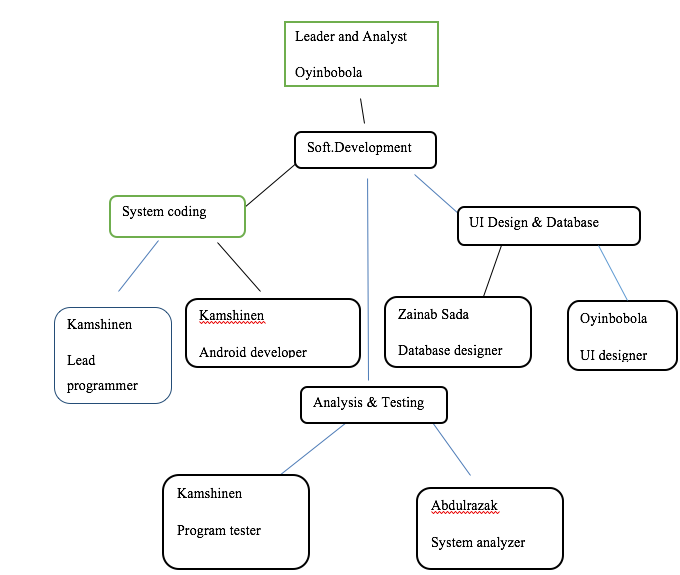


Figure . Organizational Chart

#### List of Risks

Table . Lists of Risks

|  |  |  |  |
| --- | --- | --- | --- |
| Risk | Probability | Effects | Your Strategy |
| Poor Productivity:  Given long project timelines, the sense of urgency to work in earliest is often absent resulting to time lost in early project stages that can never be regained | Low | Serious | Short iterations, right people on team, coaching and team development. |
| Inherent Schedule Flaws:  Software development, given the intangible nature and uniqueness of software, is inherently difficult to estimate and schedule. | High | Tolerable | Get the team more involved in planning and estimating. Get early feedback and address slips directly with stakeholders. |
| Requirements Inflation:  As the project progresses more and more features that were not identified at the beginning of the project emerge that threaten estimates and timelines. | Moderate | Tolerable | Constant involvement of customers and developers. |
| Employee Turnover:  Key personnel leave the project taking critical information with them that significantly delays or derails the project. | Low | Serious | Increased collaboration and information sharing on the team. |
| Specification Breakdown:  When coding and integration begin it becomes apparent that the specification is incomplete or contains conflicting requirements. | Moderate | Tolerable | Use a dedicated Product Manager to make critical trade off decisions. |

#### List of Work Packages

|  |  |
| --- | --- |
| Work Package No | 1 |
| Work Package Name | Project Feasibility and Pre-Research (SRS-Feasibility stage) |
| Start-End Date and Time | March 15th – March24th |
| Related Organizations |  |

|  |
| --- |
| 1- List the activities of work packages. |
| 1.1 Project Process and Economic Feasibility:  Project Initiation  -Market research  -Identification of the requirements and cost analysis of relevant sectors  -Analysis of Workflow  1.2 Technological Feasibility:  -Output technical and technological requirements analysis  -Determine the technological resources will be needed in the project  -Literature and patent research  -Examination of similar national and international projects made by applied technology  -Conceptual design  -Potential research approaches and methods  -Software requirements analysis |
| 2- Describe the methods and parameters that will be used for work package. |
| Research on the internet for similar applications  - Acquiring the list of necessary tools  -determinations of processes and methodology  -Research of recent trends, risks, and best practices |
| 3- List the experiments, tests and analysis in the work package. |
| -Economic feasibility analysis  -Analysis of similar products  - Budget feasibility analysis |
| 4- List the output of work package and its success criteria. |
| Outputs:  -Sufficient guides found from research to help put our program in perspective  -Complete economic and technological feasibility study  Success Criteria:  - project is put in perspective and successful initiation achieved |
| 5- Explain the relation of output with other work packages |
| The output here ere helps in the documentation sages with strategies needed alongside creation and delivering the deliverables |

|  |  |
| --- | --- |
| Work Package No | 2 |
| Work Package Name | Take Out Based System Design Technology (SRS-design stage) |
| Start-End Date and Time | March 24 –April 2nd |
| Related Organizations |  |

|  |
| --- |
| 1- List the activities of work packages. |
| * Determining System Parameters * Design of System * Selection of Device to be used * Evaluation of System Design |
| 2- Describe the methods and parameters that will be used for work package. |
| * Time is analyzed to see if pace is slow * Management tools are used e.g. IBM for diagram drawings * Requirements are rechecked and revised for better understanding * Customer feedback |
| 3- List the experiments, tests and analysis in the work package. |
| * Time analysis * Budget analysis * Effort estimation * Cost of change estimation * Risk management |
| 4- List the output of work package and its success criteria. |
| Outputs:   * System requirement document * Project structure * Dataflow diagrams * Sequence diagrams * Use case diagrams * E-R diagrams   Success Criteria:  With this documents estimates are more accurate like use case estimates using use case point as a unit of measurement |
| 5- Explain the relation of output with other work packages |
| With this output, the development and unit testing stages can commence |

|  |  |
| --- | --- |
| Work Package No | 3 |
| Work Package Name | Take Out Development of System Software (SRS-Development Stage) |
| Start-End Date and Time | April 6th –April 21st |
| Related Organizations |  |

|  |
| --- |
| 1- List the activities of work packages. |
| -Creating Database  -Software Development  -Establishment of Structure and Necessary Software  -Algorithm Modeling  -Creating a system Programming Language for Web Server  -Creating a Database Connection Between Web Services  -User Interface Design and Programming  -Software Integration  -Creating of link between Web Services and Database Module  -User Interface Testing  -System Testing |
| 2- Describe the methods and parameters that will be used for work package. |
| -Android studio for coding  -UI design  -Database implementation  -Functional testing  -Unit testing |
| 3- List the experiments, tests and analysis in the work package. |
| -Risk analysis and monitoring  -Functional testing  -Connection testing  -Algorithmic analysis  -UI testing  -broken link test and fixing |
| 4- List the output of work package and its success criteria. |
| Outputs:  -A working database  -A Forgiving User Interface  -An improved program  -AN integrated system  Success Criteria:  -constant successful connection to server and database  -user interface components correctly carry out its functionality  -dataflow correctness  -robustness acquired  -database can hold large size of information  -no broken link |
| 5- Explain the relation of output with other work packages |
| When all pieces of the system have been completed, it will lead to the implementation and thus the creation of a prototype |

|  |  |
| --- | --- |
| Work Package No | 4 |
| Work Package Name | Prototype Implementation and Test Study and Maintenance (SRS-Test & Maintenance stage) |
| Start-End Date and Time | April 19th- April 30th |
| Related Organizations |  |

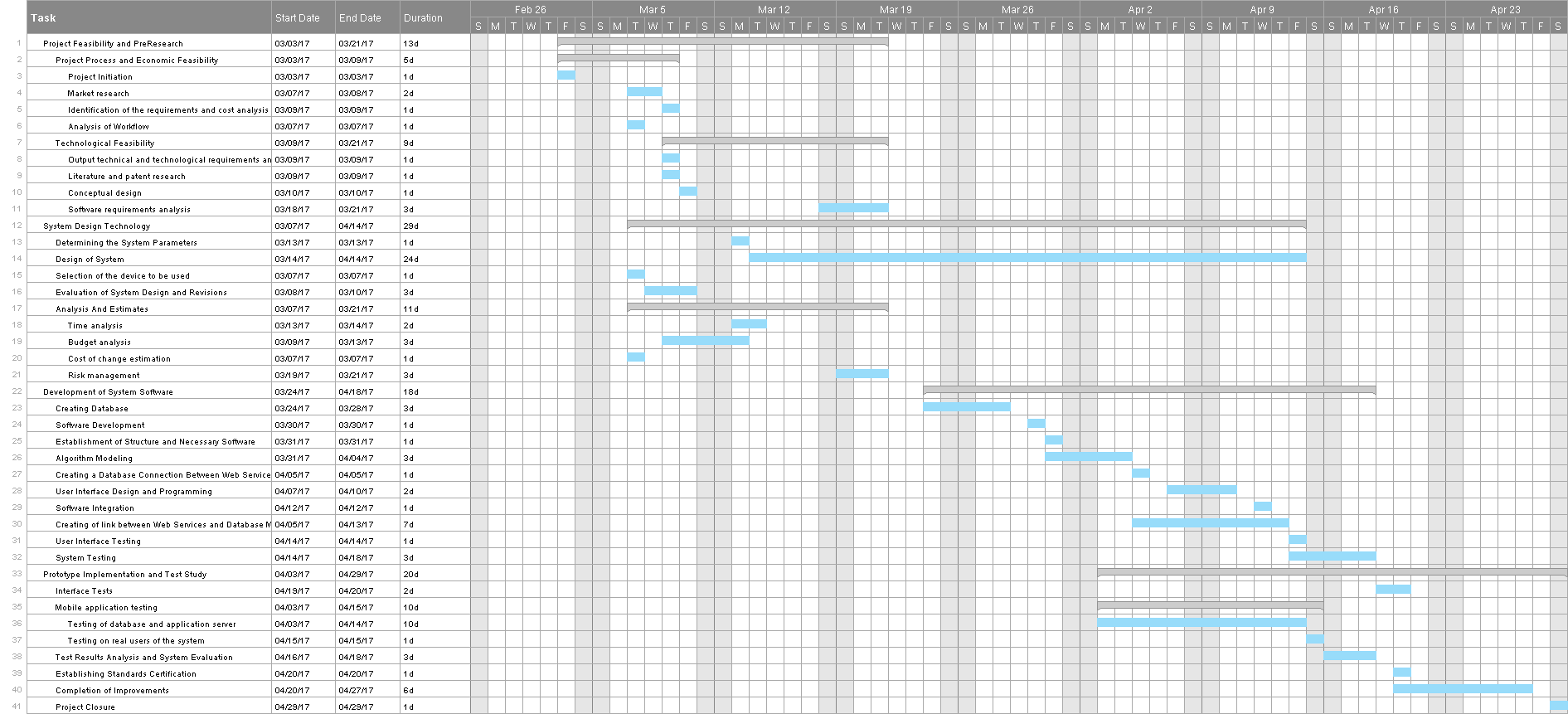
|  |
| --- |
| 1- List the activities of work packages. |
| -Interface Test  -Mobile Application Testing  -Testing of Database and Application Server  -Testing on Real Users of the System  -Test Results Analysis and Software Evaluation |
| 2- Describe the methods and parameters that will be used for work package. |
| * Black-box testing * White-box testing * User experience measurement * Analysis of algorithms |
| 3- List the experiments, tests and analysis in the work package. |
| -Interface Test  -Mobile Application Test  -Testing of Database and Application… |
| 4- List the output of work package and its success criteria. |
| Outputs:  A presentable prototype of our system  Success Criteria:  A working system in accordance to our initial idea and time plan and no budget problems |
| 5- Explain the relation of output with other work packages |
| If output was what was expected is signifies that all other packages were implemented accurately. |

#### List of Milestones

Table . List of Milestones

|  |  |  |
| --- | --- | --- |
|  | Description of Output | Expected Time Interval |
| 1 | Economic and technological feasibility study | March 15th 2017 -March 20 2017 |
| 2 | 1st draft of the system’s source code | March 24th 2017-March 29th 2017 |
| 3 | A working database | April 1st - April 19th 2017 |
| 4 | A User Interface | April 21st - May 2nd |
| 5 | An improved program | May 3rd - May 19th 2017 |
| 6 | A presentable prototype of our system | May 19th - May 23rd 2017 |

#### Gantt chart



### PERT Analysis

#### WORK BREAKDOWN STRUCTURE



#### NETWORK DIAGRAM

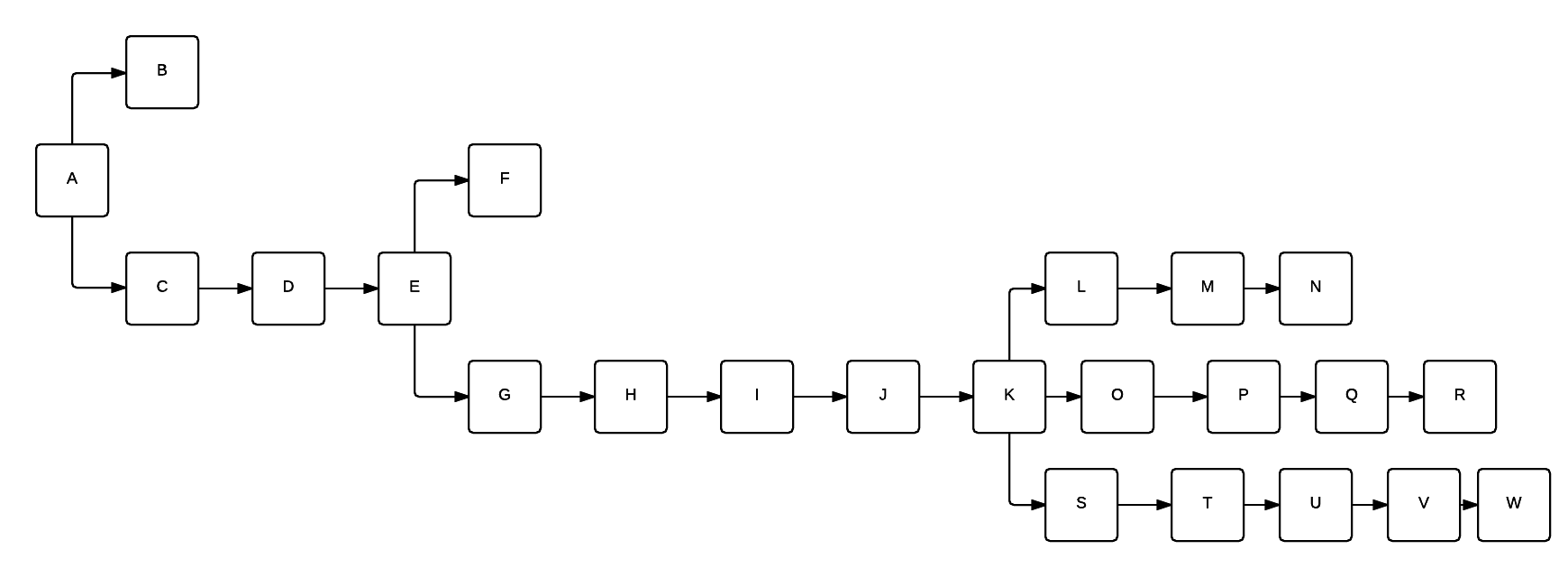


Figure . Network Diagram

#### Calculate the project completion time

Table . Network Earliest and Latest Time



Table . Path Duration



#### Calculate the expected task times



Table . Expected Time



#### Estimated Path Durations

Table . Expected Earliest and Latest Time



Table . Expected Earliest and Latest Time



Variance = ((Pessimistic time - optimistic time)/6)^2

Table . Project Variance



Table . Path Variance



Calculating the probability of completing the project in less than 109 days

Z = (Specified time – path expected time)/Path standard time

Table . Project Z Values



#### PROJECT CRASHING TIME

Table . Project Crashing Time

****

## Requirements Analysis And Specification

### Class Diagram

The class diagram shows the classes of the system and their attributes. It also shows their methods and relationships with each other.

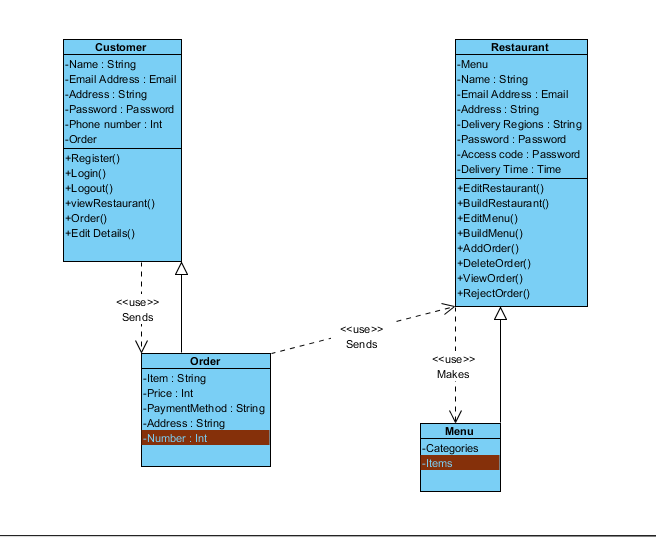


Figure . Class Diagram

### Entity- Relationship Diagram

The ER diagram shows the entities in the database of the system and their attributes and relationships with one another.

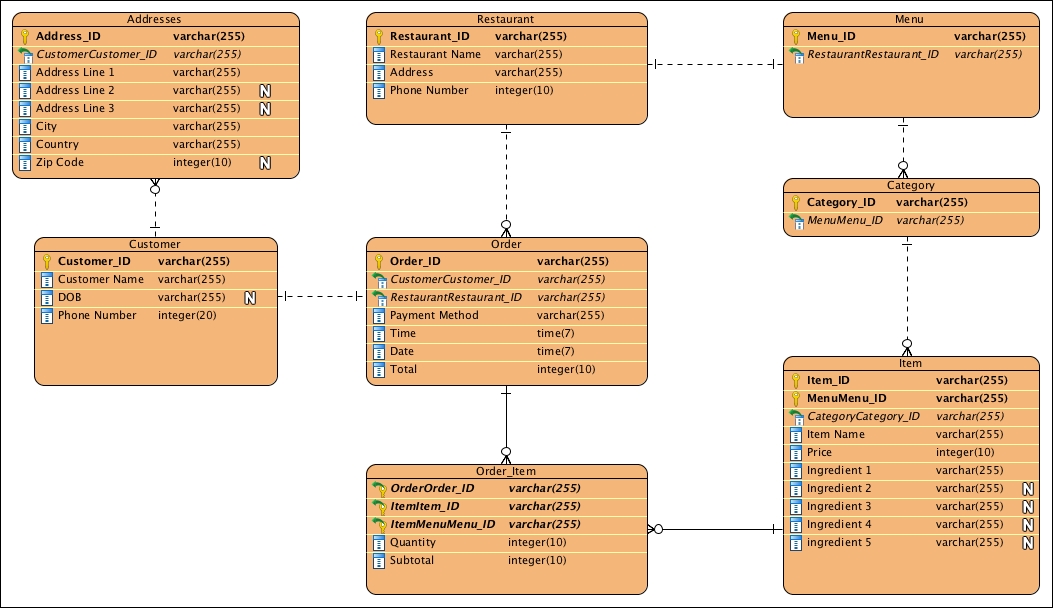


Figure . Entity-Relationship Diagram

### Sequence Diagram

The sequence diagram shows how the customer and restaurant interact with the system in order. For example, to order food, the customer has to initiate the process and then the restaurant can respond.

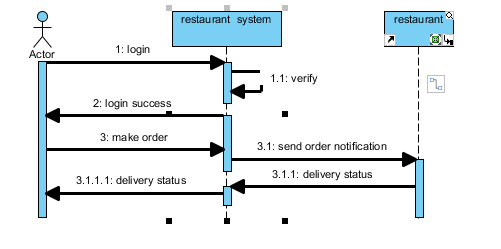


Figure . Sequence Diagram

## Architectural Design

### System Architecture Diagram

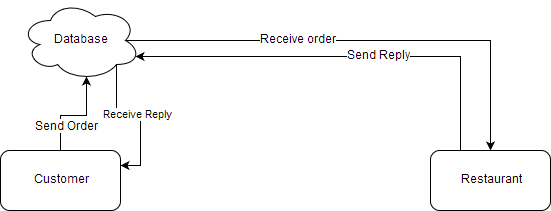


Figure . System Architecture Diagram

## Software Design

### Modular hierarchy diagram

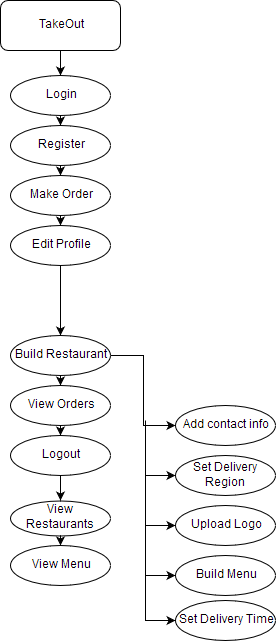


Figure . Modular Hierarchy Diagram

## Coding

### Data Flow Diagram

### Food Delivery System Level 0 Data Flow Diagram

Level 0 is a decomposition of the Food Delivery System process in Figure 4.1. It shows how the Customer and Restaurant can login and edit their account profiles. It also shows how the restaurant can build or edit their menu.

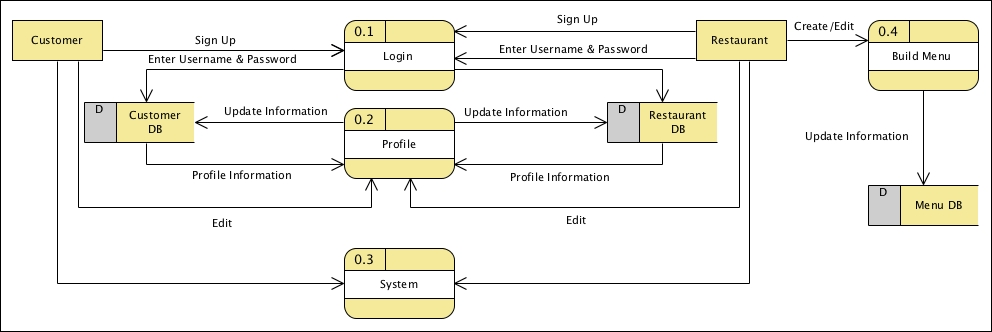


Figure . Level 0 Data Flow Diagram

### System Level 1 Data Flow Diagram

Level 1 is a decomposition of the System process in Level 0. It shows the processes involved when picking items and placing an order.

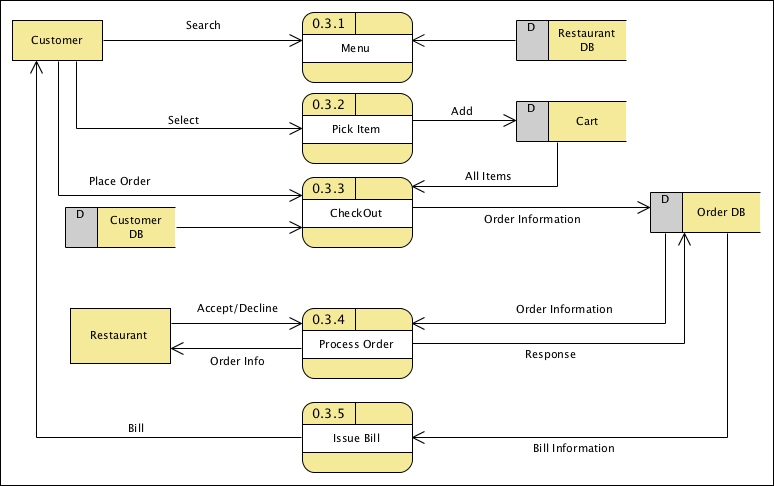


Figure . Level 1 Data Flow Diagram

## Testing Verification and Validation

The testing method we used was Black-Box testing, where the tests are done from a user’s point of view

## Algorithm

For the implementation of this system, we used the object-oriented approach of programming. We constructed classes to represent every object and modeled each Customer and Restaurant as an object in the system. As a requirement for the database – Firebase, we implemented all information meant to be saved in the database as an object before storing it in the database. This step was very important because whenever we want to retrieve information from the firebase database, the information retrieved is a form of data snapshot which is casted to the form of the class that was created to pass the information.

In the database, data are stored as JSON objects. Here there are no records or tables. It uses a flat structure which means, whenever you add data to the JSON tree, it becomes a node to the existing tree with an associated key.

For each of these keys, we decided appropriate keys to store every form of data stored. Using the entity relationship datagram, we generated JSON objects with their appropriate keys to represent every entity that was modeled in the entity relationship diagram. We made sure we avoided nesting and for every relationship between the entities we did the proper appropriate structuring that was needed.

For one to one relationships, the entities remained as they were but with each having the primary key from the other. For one to many relationships, we eliminated the many part and included the attributes into the other entity. Also, for many to many relationships, we included the primary keys of both into each of the tables. Because of this, whenever we want to update data on one of the locations we have to update on the other location as well.

Finally, for security constraints we decided to create a folder that would store the user ID’s of Restaurant and Customers separately. This was done so the users all have different access to the database from which we get a pool of User ID’s to determine which page to view for that particular user, since our application allows you to choose which kind of user you are before your login into the system. This method was carried out because Firebase is not like the traditional SQL database that offers a well predefined method for that.

### Description of the Main Modules in the System

#### Login Module

The login module is designed for both the customer and the restaurant. The login module for each of the users is accessed on two different pages; the customer page and the restaurant page. Whenever the customer wants to login on the restaurant page and the login information is correct, the customer is redirected to the customer login page and vice versa in the other case where the restaurant wants to login on the customer page. Now the login information used to login into the system is the email address of the user and the password. The login field checks the format of the email address to see whether it is correct before it sends the information for authentication. When the information is gotten from the text fields, the information is sent to the firebase authentication server to authenticate this login information and after this is successful a toast message is displayed to show the success of the login and immediately the appropriate page for the user is displayed. When login is not successful a message is displayed and the user is asked to reenter the login information again and again until an unlimited number of trials.

#### Registration Module

The registration module is used to register only customers of the system. The registration page provides four different fields. The fields take in the customer name, customer street address, customer email address and the password for the user. These fields check the appropriate format for the information passed written into them. The email address field checks the format of the email and the phone field checks the correct format of phone number inserted. When the customer violates the rules, the appropriate message is displayed to tell the user that the format is not correct. When all the information is correct, the information is sent to the firebase registration server to register the customer as a user on the firebase. This registration is completed by saving the customer in the category table as “customer” so whenever the customer wants to login, the user can login on the customer login page and it’s appropriate page is displayed after successful login.

#### Reset Password Module

A reset password page is created for this module. The page provides a text field that takes in the email address of the customer and checks the format for the email address of the customer. After this is done the email address is sent to the firebase re-authentication server that handles this information. It sends a password to the email address for the user to get and use for login. Now the firebase server doesn’t change the unique key it registered the customer with, so every of the customer’s information is kept in the same order and unchanged.

#### Make Order Module

This module allows the customer to make an order from the restaurant menu items. Here the user selects the particular food item to order and the pop up menu is shown to allow the customer add other recipes and increase the quantity to order. When the order is made the order is sent to the firebase database and it immediately updates the order table for the particular restaurant the customer is ordering from. After this happens, a notification is sent to the restaurant to view the order that has been made to the restaurant. Our order has an extra field which contains the customer’s unique key so that a reply can be sent to that particular user. This allows the customer to order from different restaurants.

#### Receive and Reply Order Module

This module is created to allow the restaurant to view order for every customer that has ordered an item from the restaurant menu. When the restaurant selects a particular order that has been sent to Order page, the restaurant can give an immediate reply to the order that has been placed by a particular customer. The restaurant might choose to accept the order or reject the order. When the user accepts the order the customer is notified of this action immediately it’s done. Also, when the restaurant rejects the order, the order is rejected with a particular reason. In our implementation, the restaurant has the option of choosing four different reasons: No food, Bad weather, No Delivery and Too many orders. This information is sent back to the customer and the customer is aware of the reason why the order was rejected.

#### Edit Profile Module

This is a very important module in our system. The customer is given the privilege to edit his profile in the system. This is to allow the users that have relocated to other locations to have food delivered to them at those locations or for customers who want to order from different locations can see the feature as an important one. The module also allows the customer is edit the phone number of the profile. These specific set of information are very important for ordering food items. The module provides text fields for the customer to edit the information to change and update. The text field checks the format of information entered into those fields. This information is sent to the firebase server to update all the tables that the user is particularly registered to.

#### Log Out Module

This module allows the user to log out from the system. The module is implemented by displaying a page that has a log out button on it. When the button is clicked, the user is logged out of the system and every information is saved before going to the login page of the restaurant. The log out procedure is carried out by sending a session id to the firebase authentication server to end the session and disable the user from access to the firebase database. This is carried out automatically on the firebase server site. After this is done, a message is displayed to the user that the log out process was successful and the user is taken to the login page of the system.

## Standards

The standards we used in this project are as follows:

* Requirements analysis and development part, we use the IEEE standard
* The ISO 9000 was used to ensure that we meet the needs of customers and  
  other stakeholders.

## Detailed Description of the Implementation

Here every user interface is clearly explained with the help of screenshots. This was designed Using the mock flow tool [5].

### CUSTOMER USER SIDE

#### Register Screen

This screen is the basic register screen, it will be available to all new customers who wish to register on Take Out. This screen will provide authentication procedures just in case there are wrong entries.

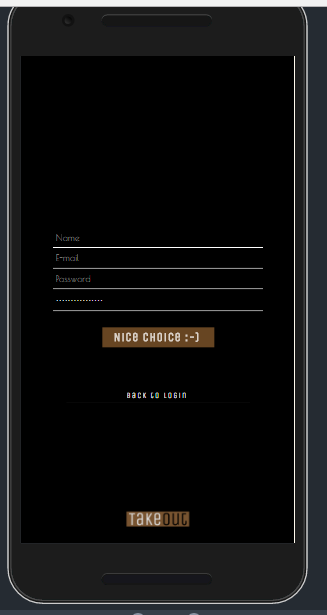
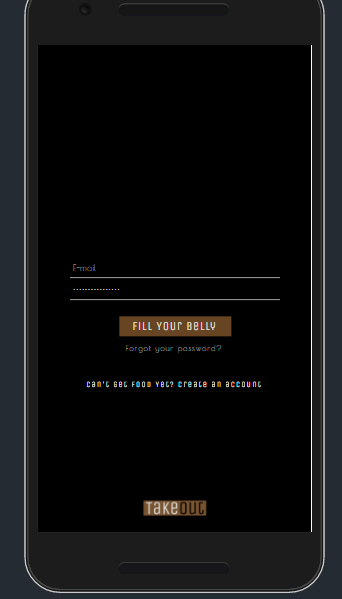


Figure . Registration UI Screen

#### LOGIN SCREEN

This screen here also shall be available to all customers providing authentication to registered users giving them access to Take Out.



Forgiving UI to allow you reset your user password

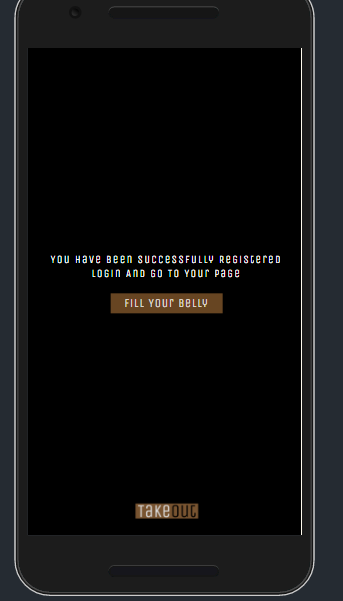
Press to submit and proceed

Redirects user to registration screen

Figure . User Login Screen

#### User Confirmation Page

This shows the success of your registration as seen below in figure 5

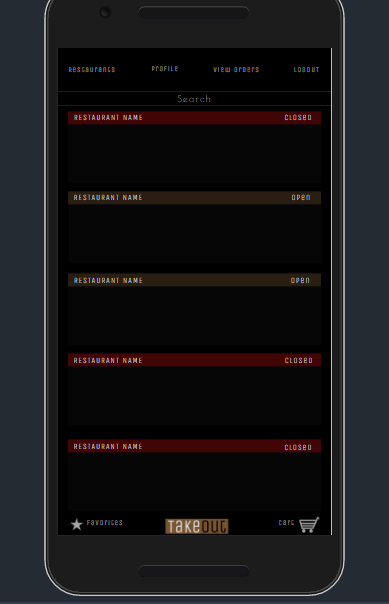


Proceed to Take Out main menu

Figure . Confirmation Screen

#### Main Screen

This displays various predetermined restaurants for user to view based on nearest location, also containing tabs for search functionality, favorites, cart etc. This screen is the heart of the system as it is the major page linking all others.



Log out

View personal profile

Search field

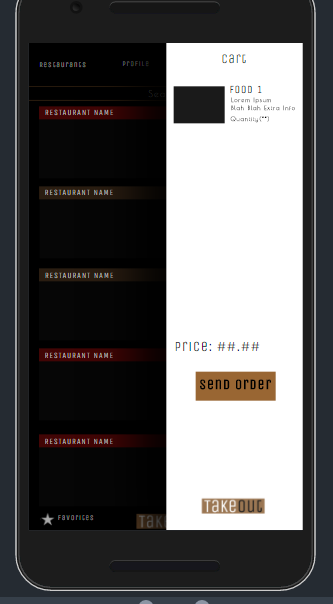
View favorites window

View cart screen

Figure . Main Screen

#### Cart window

This window gives a visual off item already in the cart and a quick access to the order placing functionality. It displays the total price and quantity of selected items to be processed.



Total price to be charged

Selected food

And details

Figure . Cart Window

#### Favorite window

Supplies quick access to ordering often ordered products by customer help in saving time by not having to search for same item or restaurant. The system keeps track of it for you (the customer).

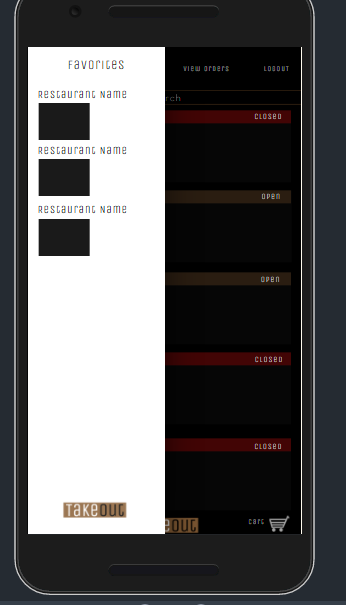


Figure . Favorites Window

#### Restaurant screen

Shows a single restaurant as selected by user to view its details, It displays its variety of products in categories for user to choose from as seen below.

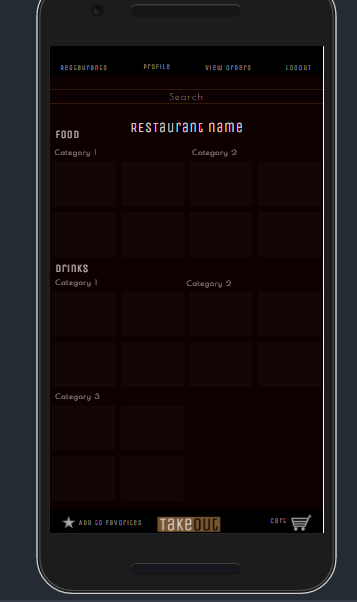


Figure . Restaurant Screen

#### Item Window

After item has been selected these windows allows you to choose how exactly you want that item prepared, you have the option to add or remove certain ingredients before adding to cart, helps with people with food allergies. It also allows you to select item quantity needed.

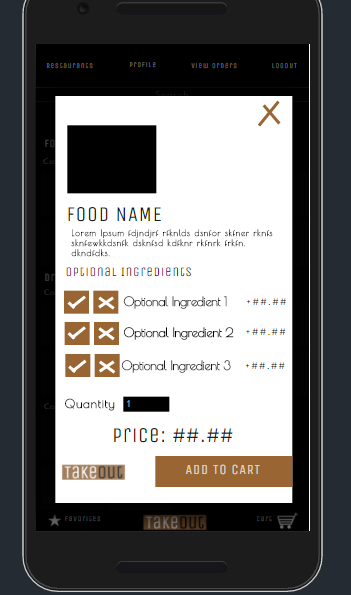


Figure . Item Window

#### Cart window

This window gives you a full view of items you have previously selected, quantity and prices and also allow you to choose your ideal address and payment methods either by cash or credit. Take Out doesn’t handle transactions this information is just avoid misunderstandings during the delivery process.

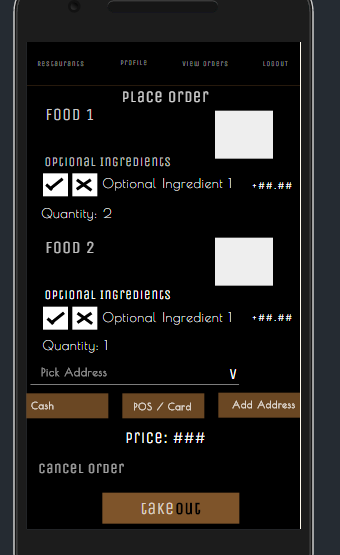


Figure . Cart Window

#### Order status window

This is one of two either your order has been accepted or it has been rejected. If rejected a reason is supplied an example might be insufficient ingredients or maybe underage in situation where an alcoholic item is ordered by a minor

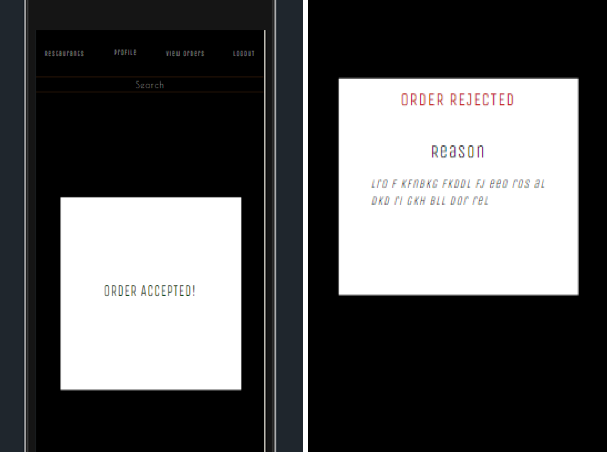


Figure . Order Status Window

#### Edit Profile

This screen gives you ccess to change your profile details which will be immediately updated/deleted in the database .

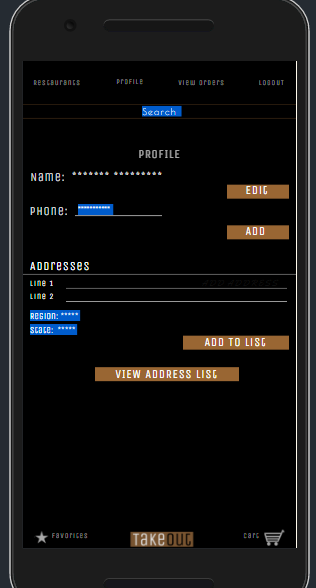


Figure . Edit Profile

### RESTAURANT USER SIDE

#### Registration Screen

This screen is the basic register screen, it will be available to all new restaurants who wish to register on Take Out. This screen will provide authentication procedures just in case there are wrong entries. Also provides success note for successful registration.

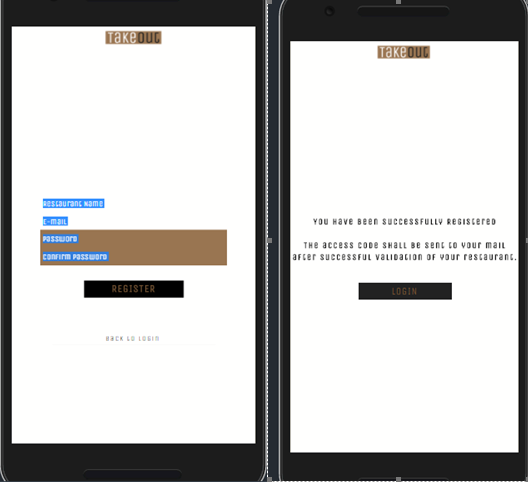


Figure . Registration Screen

#### Restaurant Setup.

The series of images below show the step by step process involved in setting up restaurant with all required details and building digital menus to show you items which will be displayed on system when the specified restaurant comes up.

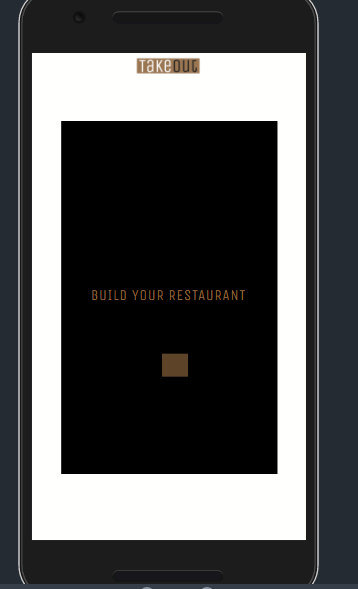


Figure . Build Restaurant Screen

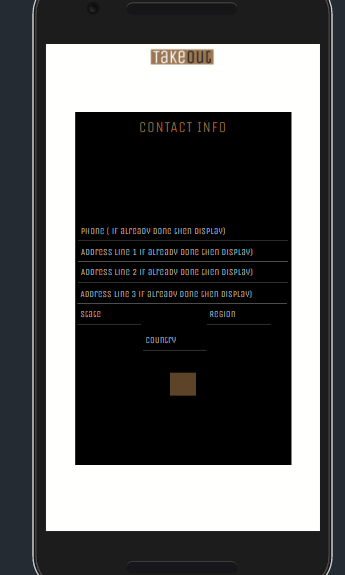


Figure . Contact Information

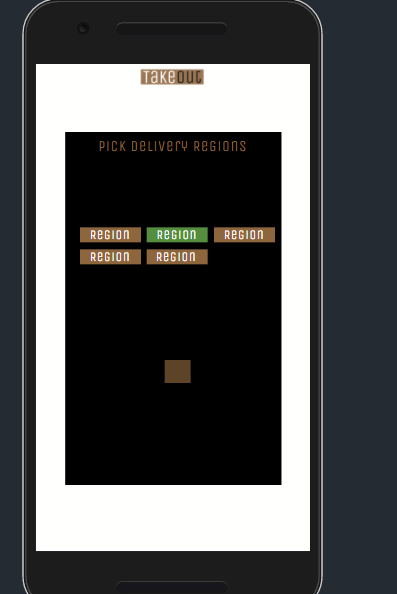


Figure . Region Selection

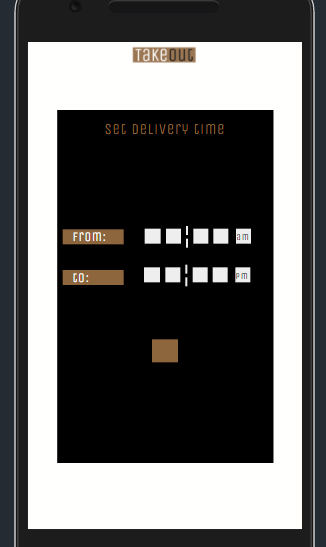


Figure . Delivery Time Range

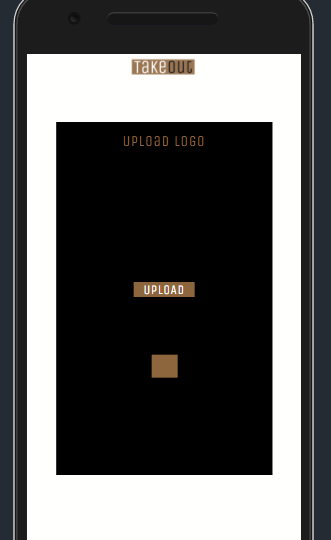


Figure . Upload Logo

#### Order Viewing

A view of received orders in order of arrival time to be processed by restaurant admin if it is to be accepted or rejected.

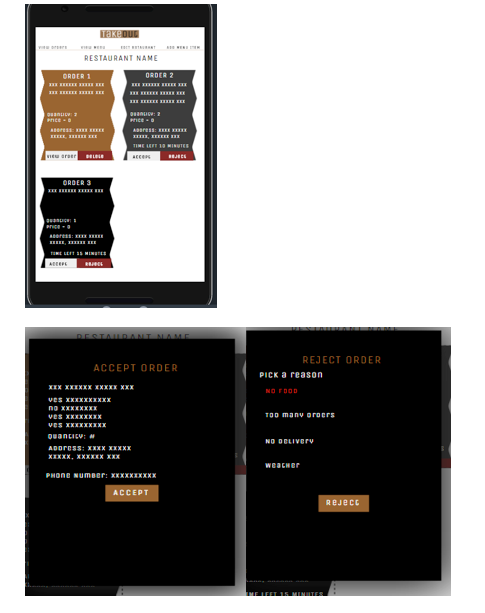


Figure . Order Viewing and Status

### DETAILED DESIGN

Below is a diagram showing the view of entities of Take Out system in a hierarchical fashion, in this Section a brief explanation for each entity shall be given.

In this Section, each entity in Figure 4.5 will be individually discussed clearly in full with respect their attributes and functions they perform or are used as inputs for.

#### Address detailed

This entity is one which hold all addresses of the customer to be stored in the database.

Attributes

* Address id: This is the address unique identifier, the primary key
* CustomerCustomer id: Is a reference from the customer id it’s responsible for matching customer to his address.
* Address Line 1-3: Three distinct address only one is needed the two are optional.
* City: a region where the system can locate in other to display nearest restaurants
* Country: The country the customer is located at the moment not customer’s nationality
* Zip code: series of numerical digits to show a more precise location of the customer

Functions: This entity has no predefined functions but it is used when making delivery

#### Customer detailed

This entity contains just the major details needed by the system to make its delivery functions

Attributes

* Customer id: unique customer identifier, the primary key
* Customer name: the customer’s name
* DOB: customer date of birth this is needed to know if what is served is legal.
* Phone number: customer contact number

Functions

* Place order (item food, restaurant name): selected item to be ordered is being sent to the desired restaurant.
* addtoCart (item food, restaurant name): adding items to cart for further processing by customer not yet sent over.
* Editprofile (): customer address lines and/or removing.
* cancelorder (int order\_id): cancelling an order within the time frame its possible.
* editOrder (): add/remove ingredients from items to order.
* Search (restaurant, item): search Take Out database.

#### Restaurant detailed

This entity contains details of registered restaurants stored in the database

Attributes

* Restaurant id: unique restaurant identifier, the primary key.
* Restaurant name: the restaurant name to be displayed in the system.
* Address: where this restaurant is located.
* Phone number: the restaurant contact info.

Functions

* orderstatus (): a Boolean function which returns false if order is rejected and vice versa.
* Makedelivery (): occurs after a precondition has been satisfied which is if other has been accepted.
* Editmenu (): add/remove items to/from its menu.

#### Order detailed

This entity contains information about the order being made with respect to just the customer and restaurant simply.

Attributes

* Order\_id. unique order identifier given as order is being made, the primary key.
* CustomerCustomer\_id: to identify which customer made the order.
* RestaurantRestaurant\_id: to identify which restaurant was the order made to.
* Payment method: means of payment selected by customer.
* Time: specific time of the order.
* Date: date of the order.
* Total: total number of others received.

#### Order\_item

This is a more detailed version of order entity.it contains unique details of other made including what was ordered.

Attributes

* OrderOrder\_id: a foreign key to identify the order made uniquely.
* ItemItem\_id: also, a foreign key to identify the selected item from the menu.
* ItemMenuMenu: foreign key to identify which menu the item was ordered from
* Quantity: total distinct amount of that item was ordered.
* Subtotal: total price to be billed for that order.

#### Menu detailed

This entity holds details about the digital menu created by each restaurant.

Attributes

* Menu\_id: a unique menu id for every restaurant registered, Primary key.
* RestaurantRestaurant\_id: an identifier to match restaurant to its menu

#### Category detailed

This is details about item categories like drinks, deserts etc.

Attributes

* Category\_id: unique identifier for each item category available.
* MenuMenu\_id: an identifier to match which menu the category was from.

#### Item detailed

This entity hold information used in order processing before making delivery.

Attributes

* Item\_id: unique identifier for which item was ordered.
* MenuMenu\_id: unique identifier for which menu was the order from.
* CategoryCategory\_ID: identifier to match which category from which menu the order was made from.
* Item name: name of ordered item of the menu.
* Price: price of ordered item.
* Ingredient 1-5: ingredient contained in the ordered item.

# TESTING

Table . Customer Login Test Case

|  |  |
| --- | --- |
| Test – Case ID: | TC - 01 |
| Test – Case Name: | Customer Login |
| Pass/Fail Criteria: | System should redirect user to the display restaurants page |
| Input Data: | Email, Password |
| **Test Procedure** | **Test Results** |
| Step1: Enter Email and Password |  |
| Step2: Click ‘Fill Your Belly’ | System redirects to restaurants page |

Table . Restaurant Login Test Case

|  |  |
| --- | --- |
| Test – Case ID: | TC - 02 |
| Test – Case Name: | Restaurant Login |
| Pass/Fail Criteria: | On first login, the system should redirect user to the build restaurant page. For subsequent logins, the system should redirect user to the view orders page. |
| Input Data: | Email, Password |
| **Test Procedure** | **Test Results** |
| Step1: First Login | System redirects to build restaurant page |
| Step2: Subsequent Login | System redirects to view orders page |

Table . Customer Logout Test Case

|  |  |
| --- | --- |
| Test – Case ID: | TC - 03 |
| Test – Case Name: | Customer Logout |
| Pass/Fail Criteria: | System displays prompt asking user ‘do you really want to log out?’ if yes logs user out and redirects to login page, if no prompt disappears |
| Input Data: | Click Logout |
| **Test Procedure** | **Test Results** |
| Step1: Click Logout | System displays prompt |
| Step2: Select No | Prompt disappears |
| Ste3: Select Yes | System redirects to login page |

Table . Restaurant Logout Test Case

|  |  |
| --- | --- |
| Test – Case ID: | TC - 04 |
| Test – Case Name: | Restaurant Logout |
| Pass/Fail Criteria: | System displays prompt asking user ‘do you really want to log out?’ if yes logs user out and redirects to login page, if no prompt disappears |
| Input Data: | Click Logout |
| **Test Procedure** | **Test Results** |
| Step1: Click Logout | System displays prompt |
| Step2: Select No | Prompt disappears |
| Ste3: Select Yes | System redirects to login page |

Table . Customer Register Test Case

|  |  |
| --- | --- |
| Test – Case ID: | TC - 05 |
| Test – Case Name: | Customer Register |
| Pass/Fail Criteria: | System should redirect user to the registration confirmed page |
| Input Data: | Name, Email, Password, Confirm Password |
| **Test Procedure** | **Test Results** |
| Step1: Click Register | System shows ‘Enter Details’ panel |
| Step2: Enter Input Data |  |
| Step3: Click Register | System redirects user to the registration confirmed page |

Table . Customer View Order Test Case

|  |  |
| --- | --- |
| Test – Case ID: | TC - 06 |
| Test – Case Name: | Customer View Order |
| Pass/Fail Criteria: | System should redirect to view orders page |
| Input Data: | View Order |
| **Test Procedure** | **Test Results** |
| Step1: Click View Order | System displays page showing previously sent orders |

Table . Customer Edit Profile Test Case

|  |  |
| --- | --- |
| Test – Case ID: | TC - 07 |
| Test – Case Name: | Customer Edit Profile |
| Pass/Fail Criteria: | System show display profile page and update database with the changes made by the user |
| Input Data: | Name, Phone, Address |
| **Test Procedure** | **Test Results** |
| Step1: Click Edit Profile | System redirects to profile page |
| Step2: Edit Name | System saves and updates database with new name |
| Step3: Edit Phone Number | System saves and updates database with new number |
| Step4: Add Address | System add and updates database with new address |

Table . Restaurant Edit Profile Test Case

|  |  |
| --- | --- |
| Test – Case ID: | TC - 08 |
| Test – Case Name: | Restaurant Edit Profile |
| Pass/Fail Criteria: | System show display profile page and update database with the changes made by the user |
| Input Data: | Name, Phone, Address |
| **Test Procedure** | **Test Results** |
| Step1: Click Edit Profile | System redirects to profile page |
| Step2: Edit Name | System saves and updates database with new name |
| Step3: Edit Phone Number | System saves and updates database with new number |
| Step4: Add Address | System add and updates database with new address |

Table . Customer View Restaurants Test Case

|  |  |
| --- | --- |
| Test – Case ID: | TC - 09 |
| Test – Case Name: | Customer View Restaurants |
| Pass/Fail Criteria: | System show redirect to restaurants page |
| Input Data: | Restaurants |
| **Test Procedure** | **Test Results** |
| Step1: Click Restaurants | Systems redirects to restaurants page to display a list of all restaurants |

Table . Customer View Menu Test Case

|  |  |
| --- | --- |
| Test – Case ID: | TC - 10 |
| Test – Case Name: | Customer View Menu |
| Pass/Fail Criteria: | System should redirect user to the menu page of the selected restaurant |
| Input Data: | Some Restaurant |
| **Test Procedure** | **Test Results** |
| Step1: Click Some Restaurant | Systems redirects to that restaurant’s menu page |

Table . Restaurant View Menu Test Case

|  |  |
| --- | --- |
| Test – Case ID: | TC - 11 |
| Test – Case Name: | Restaurant View Menu |
| Pass/Fail Criteria: | System should redirect to the restaurant’s menu page |
| Input Data: | View Menu |
| **Test Procedure** | **Test Results** |
| Step1: Click View Menu | System redirects to the restaurant’s menu page to display all items on the menu |

Table . Customer Send Order Test Case

|  |  |
| --- | --- |
| Test – Case ID: | TC - 12 |
| Test – Case Name: | Customer Send Order |
| Pass/Fail Criteria: | System should display order status window send order details to restaurant from which the customer ordered food from |
| Input Data: | TakeOut |
| **Test Procedure** | **Test Results** |
| Step1: Click TakeOut | System displays to order status window |

Table . Restaurant Accept Order Test Case

|  |  |
| --- | --- |
| Test – Case ID: | TC - 13 |
| Test – Case Name: | Restaurant Accept Order |
| Pass/Fail Criteria: | System should change order options to View Order and Delete Order and should send acceptance message to customer |
| Input Data: | Accept |
| **Test Procedure** | **Test Results** |
| Step1: Click Accept | System displays View Order and Delete Order options and sends acceptance to customer |

Table . Restaurant Reject Order Test Case

|  |  |
| --- | --- |
| Test – Case ID: | TC - 14 |
| Test – Case Name: | Restaurant Reject Order |
| Pass/Fail Criteria: | System should redirect to Reject Reasons page and shout get reason from user and send an Order Rejected message to the customer with the reason attached |
| Input Data: | Reject |
| **Test Procedure** | **Test Results** |
| Step1: Click Reject | Systems redirects to Reasons page |
| Step2: Select Reason | System sends message to customer |

Table . Restaurant View Orders Test Case

|  |  |
| --- | --- |
| Test – Case ID: | TC - 15 |
| Test – Case Name: | Restaurant View Orders |
| Pass/Fail Criteria: | System should redirect to Orders page and display Accept and Reject options |
| Input Data: | View Orders |
| **Test Procedure** | **Test Results** |
| Step1: Click View Orders | System redirects to Orders page |

Table . Reset Password Test Case

|  |  |
| --- | --- |
| Test – Case ID: | TC - 16 |
| Test – Case Name: | Reset Password |
| Pass/Fail Criteria: | System should display Password Reset Successful message and update database with the user’s new password |
| Input Data: | New Password |
| **Test Procedure** | **Test Results** |
| Step1: Click Reset Password | System displays new password page |
| Step2: Enter New Password |  |
| Step3: Click Reset | System displays Successful message and updates database |

# USER GUIDE OF THE SYSTEM

## For Customers:

* First, you need to install the ‘.apk’ file
* Run the installed app on your phone (TakeOut)
* To Register or login as a customer you have to select the Customers part of the screen at the landing page.

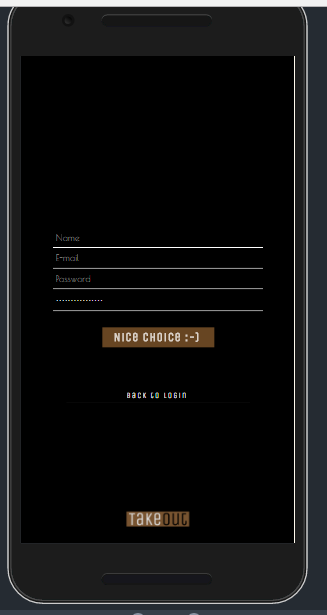


Figure . Registration Screen

* The first screen seen after logging in is a list of all restaurants your location. The Restaurants that are currently delivering are indicated as open. The restaurants that are not currently delivering are indicated as closed.
* To view a restaurant’s menu, tap the restaurant. You can click on a menu item from the restaurant menu to view and add to cart.
* From the Cart you can, remove item, reduce/increase quantity of an order and you can Send order.
* After sending the order the app takes you to an order confirmation page where you can easily review your order and select whether you want to pay with cash or POS. You can also select to add a new address for the food to be delivered to at this page.
* As a Customer, you can edit your details by going to the edit details page.

## For Restaurants:

* First, you need to install the ‘.apk’ file
* Run the installed app on your phone (TakeOut)
* To register/login as a restaurant you have to select the Restaurants part of the screen at the landing page. After registration as a Restaurant, you have to wait for confirmation from the admin for access to the system. You will receive an email confirmation if your restaurant has been successfully verified.
* On your first login, you have to build your restaurant: add contact details, set delivery regions, Set delivery time, Upload logo.

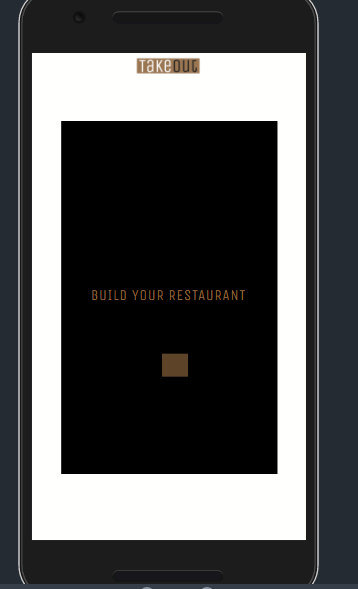


Figure . Build Restaurant Screen

* Incoming orders are shown on the view orders page. This is the hub for most of the Restaurants activities. Accepting, rejecting, viewing and deleting orders all take place from this page.
* As a restaurant, you can edit your restaurant details by going to the edit restaurant page.

# DISCUSSION

Using this application will be like an answer people have been hoping for. Getting food anytime you want is a luxury everybody wants to have. The aim of takeout is to help this happen, getting food anytime you want, the way you want it. Since the customers’ orders are in text that the restaurant can read when the order is being prepared, the chances of making a mistake when preparing a customer’s order are greatly reduced. Which causes customers to be happier with restaurant food services, providing more revenue for the restaurant.

The application will allow people save a lot of time; unnecessary time wasted in the process of calling the restaurant over the phone. The application is going to improve the food section of North Cyprus It helps people get easy access to food at any time they want supported by the favorites- save order function in the application. Making the order process seamless and less stressful

# CONCLUSION

We implemented all the functionalities stated in the requirements, all  
were tested and it showed that they are all working properly. When working on this project, we learned how to use various new software - Visual paradigm, Microsoft project, Android studio, Firebase.

# REFERENCES

[1] IEEE. IEEE STD 830-1998 IEEE Recommended Practice for Software Requirements  
Specifications. IEEE Computer Society, 1998.

[2] Automated Guide-Through Project Management Process. (n.d.). Retrieved May 24, 2017, from

<https://www.visual-paradigm.com/>

[3] (n.d.). Retrieved May 24, 2017, from <https://firebase.google.com/>

[4] Android Studio The Official IDE for Android. (n.d.). Retrieved May 24, 2017, from

<https://developer.android.com/studio/index.html>

[5] Cloud Prototyping for Product, UX and UI Design Teams. (n.d.). Retrieved May 24, 2017, from

<https://atomic.io/>

[6] Deliver winning projects. (n.d.). Retrieved May 24, 2017, from

<https://products.office.com/en-us/project/project-and-portfolio-management-software?tab=tabs-1>

[7] Free flowchart maker and diagrams online. (n.d.). Retrieved May 24, 2017, from <https://www.draw.io/>

# APPENDICES

## Instructions for installing the system

Download the APK file (only on android mobiles)

Install the APK file

## Code for the system

### MainActivity.java

package com.example.casper.userregistration;

import android.app.Activity;

import android.app.ProgressDialog;

import android.content.Intent;

import android.graphics.Typeface;

import android.support.annotation.NonNull;

import android.support.v7.app.AppCompatActivity;

import android.os.Bundle;

import android.text.TextUtils;

import android.util.Log;

import android.view.View;

import android.widget.Button;

import android.widget.EditText;

import android.widget.ProgressBar;

import android.widget.TextView;

import android.widget.Toast;

import com.google.android.gms.tasks.OnCompleteListener;

import com.google.android.gms.tasks.Task;

import com.google.firebase.auth.AuthResult;

import com.google.firebase.auth.FirebaseAuth;

import com.google.firebase.auth.FirebaseAuthException;

import com.google.firebase.auth.FirebaseUser;

import com.google.firebase.database.DataSnapshot;

import com.google.firebase.database.DatabaseError;

import com.google.firebase.database.DatabaseReference;

import com.google.firebase.database.FirebaseDatabase;

import com.google.firebase.database.ValueEventListener;

public class MainActivity extends AppCompatActivity {

FirebaseAuth mAuth;

EditText emailText;

EditText passwordText;

Button registerBtn;

Button loginBtn;

TextView regText;

TextView forgetText;

Button snap;

DatabaseReference mRef;

FirebaseDatabase database;

FirebaseUser user;

@Override

protected void onCreate(Bundle savedInstanceState) {

super.onCreate(savedInstanceState);

setContentView(R.layout.activity\_main);

mAuth=FirebaseAuth.getInstance();

emailText=(EditText)findViewById(R.id.email\_text);

passwordText=(EditText)findViewById(R.id.password\_text);

//registerBtn=(Button)findViewById(R.id.register\_btn);

loginBtn=(Button)findViewById(R.id.loginbtn);

regText=(TextView)findViewById(R.id.registerText);

forgetText=(TextView)findViewById(R.id.forgetText);

snap=(Button)findViewById(R.id.snapbtn);

database= FirebaseDatabase.getInstance();

mRef= database.getReference();

//Adding to fonts to different views

Typeface custom=Typeface.createFromAsset(getAssets(),"fonts/UnicaOne-Regular.ttf");

emailText.setTypeface(custom);

passwordText.setTypeface(custom);

loginBtn.setTypeface(custom);

regText.setTypeface(custom);

forgetText.setTypeface(custom);

//Login User using information

loginBtn.setOnClickListener(new View.OnClickListener() {

@Override

public void onClick(View v) {

String email=emailText.getText().toString().trim();

String password=passwordText.getText().toString().trim();

if(TextUtils.isEmpty(email)){

Toast.makeText(getApplicationContext(),"Please enter email",Toast.LENGTH\_SHORT).show();

return;

}

if(TextUtils.isEmpty(password)){

Toast.makeText(getApplicationContext(),"Please enter email",Toast.LENGTH\_SHORT).show();

return;

}

loginAuthenticate(email,password);

}

});

// Go to Customer registration page

regText.setOnClickListener(new View.OnClickListener() {

@Override

public void onClick(View v) {

startActivity(new Intent(MainActivity.this,RegistrationPage.class));

}

});

}

private void loginAuthenticate(String email, String password) {

mAuth.signInWithEmailAndPassword(email,password).addOnCompleteListener(MainActivity.this, new OnCompleteListener<AuthResult>() {

@Override

public void onComplete(@NonNull Task<AuthResult> task) {

if(task.isSuccessful()){

user=FirebaseAuth.getInstance().getCurrentUser();

checkUser(user);

}

else {

FirebaseAuthException e = (FirebaseAuthException)task.getException();

Log.e("LoginActivity", "Login Failed", e);

Toast.makeText(getApplicationContext(), "Login failed...... "+e.getMessage().toString() , Toast.LENGTH\_SHORT).show();

}

}

});

}

public void checkUser(FirebaseUser user){

mRef.child("registeredUsers").child(user.getUid()).addListenerForSingleValueEvent(new ValueEventListener() {

@Override

public void onDataChange(DataSnapshot dataSnapshot) {

UserGroup info= (UserGroup)dataSnapshot.getValue(UserGroup.class);

if(info.getUsergroup().matches("customer")){

Toast.makeText(MainActivity.this,"Successful login",Toast.LENGTH\_SHORT).show();

startActivity(new Intent(MainActivity.this,CustomersPage.class));

}

else{

Toast.makeText(MainActivity.this,"Login on the other page",Toast.LENGTH\_SHORT).show();

startActivity(new Intent(MainActivity.this,RestaurantLoginActivity.class));

}

}

@Override

public void onCancelled(DatabaseError databaseError) {

}

});

}

}

### OrderFoodAdapter.java

package com.example.casper.userregistration;

import android.content.Context;

import android.support.annotation.LayoutRes;

import android.support.annotation.NonNull;

import android.support.annotation.Nullable;

import android.view.LayoutInflater;

import android.view.View;

import android.view.ViewGroup;

import android.widget.ArrayAdapter;

import android.widget.TextView;

import java.util.ArrayList;

import java.util.List;

import java.util.Objects;

/\*\*

\* Created by CASPER on 5/25/2017.

\*/

public class OrderFoodAdapter extends ArrayAdapter<ViewOrderInfo>

{

Context context;

ArrayList<ViewOrderInfo> objects;

ArrayList<ViewOrderInfo> mobjects;

public OrderFoodAdapter(@NonNull Context context, @LayoutRes int resource, @NonNull ArrayList<ViewOrderInfo> objects) {

super(context, resource, objects);

this.context=context;

this.objects=objects;

this.mobjects=objects;

}

public int getCount() {

return objects.size();

}

@Nullable

@Override

public ViewOrderInfo getItem(int position) {

return objects.get(position);

}

@NonNull

@Override

public View getView(int position, @Nullable View convertView, @NonNull ViewGroup parent) {

if (convertView==null){

convertView= LayoutInflater.from(context).inflate(R.layout.order\_layout\_view,parent,false);

}

ViewOrderInfo viewOrderInfo=(ViewOrderInfo) getItem(position);

TextView textView=(TextView)convertView.findViewById(R.id.OrderNameView);

TextView textView1=(TextView)convertView.findViewById(R.id.OrderAddressView);

TextView textView2=(TextView)convertView.findViewById(R.id.OrderPhoneView);

TextView textView3=(TextView)convertView.findViewById(R.id.OrderFoodView);

TextView textView4=(TextView)convertView.findViewById(R.id.OrderPriceView);

textView.setText(viewOrderInfo.getName());

textView1.setText(viewOrderInfo.getAddress());

textView2.setText(viewOrderInfo.getPhone());

textView3.setText(viewOrderInfo.getFood());

textView4.setText(viewOrderInfo.getPrice());

return convertView;

}

}

### MakeOrderActivity.java

package com.example.casper.userregistration;

import android.content.Intent;

import android.support.v7.app.AppCompatActivity;

import android.os.Bundle;

import android.view.View;

import android.widget.AdapterView;

import android.widget.Button;

import android.widget.ListView;

import android.widget.Toast;

import com.google.firebase.auth.FirebaseAuth;

import com.google.firebase.auth.FirebaseUser;

import com.google.firebase.database.DataSnapshot;

import com.google.firebase.database.DatabaseError;

import com.google.firebase.database.DatabaseReference;

import com.google.firebase.database.FirebaseDatabase;

import com.google.firebase.database.ValueEventListener;

import java.util.ArrayList;

public class MakeOrderActivity extends AppCompatActivity {

ListView listView;

DatabaseReference mRef;

FirebaseDatabase database;

FirebaseUser user;

static ArrayList<RestaurantItems> restaurantFood=new ArrayList<RestaurantItems>();

RestaurantItemsAdapter adapter;

Button food;

Button drink;

Button water;

@Override

protected void onCreate(Bundle savedInstanceState) {

super.onCreate(savedInstanceState);

setContentView(R.layout.activity\_make\_order);

database=FirebaseDatabase.getInstance();

mRef= database.getReference();

user= FirebaseAuth.getInstance().getCurrentUser();

listView=(ListView) findViewById(R.id.menuCustomerlistview);

Button food=(Button) findViewById(R.id.foodbtn);

Button drink=(Button) findViewById(R.id.drinkbtn);

Button water=(Button) findViewById(R.id.waterbtn);

// Setting up the adapter

adapter=new RestaurantItemsAdapter(getApplicationContext(),android.R.layout.simple\_expandable\_list\_item\_1,restaurantFood);

mRef.child("items").addValueEventListener(new ValueEventListener() {

@Override

public void onDataChange(DataSnapshot dataSnapshot) {

restaurantFood.clear();

for(DataSnapshot data:dataSnapshot.getChildren()){

RestaurantItems items=(RestaurantItems) data.getValue(RestaurantItems.class);

Toast.makeText(MakeOrderActivity.this, "Loading...", Toast.LENGTH\_SHORT).show();

restaurantFood.add(items);

}

adapter.notifyDataSetChanged();

}

@Override

public void onCancelled(DatabaseError databaseError) {

}

});

listView.setAdapter(adapter);

//Making an order with the listview

listView.setOnItemClickListener(new AdapterView.OnItemClickListener() {

@Override

public void onItemClick(AdapterView<?> parent, View view, int position, long id) {

RestaurantItems item=((RestaurantItems) parent.getItemAtPosition(position));

ViewOrderInfo info=new ViewOrderInfo("Ahmet","Naviaprks",item.getName(),item.getPrice(),"6790312",user.getUid());

mRef.child("order").child(item.getRestaurantId()).child(user.getUid()).setValue(info);

Toast.makeText(MakeOrderActivity.this, "Order Sent", Toast.LENGTH\_SHORT).show();

startActivity(new Intent(MakeOrderActivity.this,CustomersPage.class));

}

});

}

}

### ReceiveOrderActivity.java

package com.example.casper.userregistration;

import android.content.Intent;

import android.support.v7.app.AppCompatActivity;

import android.os.Bundle;

import android.view.View;

import android.widget.AdapterView;

import android.widget.ArrayAdapter;

import android.widget.Button;

import android.widget.EditText;

import android.widget.ListView;

import android.widget.Spinner;

import android.widget.Toast;

import com.google.firebase.auth.FirebaseAuth;

import com.google.firebase.auth.FirebaseUser;

import com.google.firebase.database.DatabaseReference;

import com.google.firebase.database.FirebaseDatabase;

import java.util.ArrayList;

public class ReceiveOrderActivity extends AppCompatActivity {

EditText text1;

EditText text2;

EditText text3;

EditText text4;

EditText text5;

Button reply;

Spinner spinner;

String result;

DatabaseReference mRef;

FirebaseDatabase database;

FirebaseUser user;

@Override

protected void onCreate(Bundle savedInstanceState) {

super.onCreate(savedInstanceState);

setContentView(R.layout.activity\_receive\_order);

text1=(EditText) findViewById(R.id.Customername\_text);

text2=(EditText) findViewById(R.id.CustomerAddress\_text);

text3=(EditText) findViewById(R.id.CustomerPhone\_text);

text4=(EditText) findViewById(R.id.CustomerFood\_text);

text5=(EditText) findViewById(R.id.CustomerPrice\_text);

reply=(Button) findViewById(R.id.ReplyOrderbtn);

spinner=(Spinner) findViewById(R.id.spinnerView);

database=FirebaseDatabase.getInstance();

mRef= database.getReference();

user= FirebaseAuth.getInstance().getCurrentUser();

// Setting up adapter

ArrayAdapter<CharSequence> dataAdapter = ArrayAdapter.createFromResource(this, R.array.reply\_arrays,

android.R.layout.simple\_spinner\_item);

dataAdapter.setDropDownViewResource(android.R.layout.simple\_spinner\_dropdown\_item);

spinner.setAdapter(dataAdapter);

spinner.setOnItemSelectedListener(new AdapterView.OnItemSelectedListener() {

@Override

public void onItemSelected(AdapterView<?> parent, View view, int position, long id) {

result= (String) parent.getItemAtPosition(position);

}

@Override

public void onNothingSelected(AdapterView<?> parent) {

}

});

final ViewOrderInfo info=(ViewOrderInfo) RestaurantViewOrderFragment.ordered;

text1.setText(info.getName());

text2.setText(info.getAddress());

text3.setText(info.getPhone());

text4.setText(info.getFood());

text5.setText(info.getPrice());

reply.setOnClickListener(new View.OnClickListener() {

@Override

public void onClick(View v) {

text4.setText(result);

String name=text1.getText().toString();

String address=text2.getText().toString();

String phone=text3.getText().toString();

String food= text4.getText().toString();

String price=text5.getText().toString();

mRef.child("order").child(user.getUid()).child(info.customerId).setValue(new ViewOrderInfo(name,address,food,price,phone,info.customerId));

Toast.makeText(ReceiveOrderActivity.this, "Reply sent", Toast.LENGTH\_SHORT).show();

startActivity(new Intent(ReceiveOrderActivity.this,RestaurantPageActivity.class));

}

});

}

}

### ViewOrdersFragment.java

package com.example.casper.userregistration;

import android.graphics.Typeface;

import android.os.Bundle;

import android.support.annotation.Nullable;

import android.support.v4.app.Fragment;

import android.view.LayoutInflater;

import android.view.View;

import android.view.ViewGroup;

import android.widget.TextView;

import android.widget.Toast;

import com.google.firebase.auth.FirebaseAuth;

import com.google.firebase.auth.FirebaseUser;

import com.google.firebase.database.DataSnapshot;

import com.google.firebase.database.DatabaseError;

import com.google.firebase.database.DatabaseReference;

import com.google.firebase.database.FirebaseDatabase;

import com.google.firebase.database.ValueEventListener;

/\*\*

\* A simple {@link Fragment} subclass.

\*/

public class ViewOrdersFragment extends Fragment {

DatabaseReference mRef;

FirebaseDatabase database;

FirebaseUser user;

TextView nameOfCustomer;

TextView addressOfCustomer;

TextView phoneNumber;

TextView food;

TextView price;

static ViewOrderInfo info;

public ViewOrdersFragment() {

// Required empty public constructor

}

@Override

public void onCreate(@Nullable Bundle savedInstanceState) {

super.onCreate(savedInstanceState);

database=FirebaseDatabase.getInstance();

mRef= database.getReference();

user= FirebaseAuth.getInstance().getCurrentUser();

}

@Override

public View onCreateView(LayoutInflater inflater, ViewGroup container,

Bundle savedInstanceState) {

// Inflate the layout for this fragment

View view= inflater.inflate(R.layout.fragment\_view\_orders, container, false);

nameOfCustomer=(TextView) view.findViewById(R.id.nameCustomer);

addressOfCustomer=(TextView) view.findViewById(R.id.addrCustomer);

phoneNumber=(TextView) view.findViewById(R.id.phoneCustomer);

food=(TextView) view.findViewById(R.id.foodCustomer);

price=(TextView) view.findViewById(R.id.priceCustomer);

//Applying custom fonts to list view

Typeface custom=Typeface.createFromAsset(getActivity().getAssets(),"fonts/UnicaOne-Regular.ttf");

nameOfCustomer.setTypeface(custom);

addressOfCustomer.setTypeface(custom);

phoneNumber.setTypeface(custom);

food.setTypeface(custom);

price.setTypeface(custom);

//Put the order in the fragment

mRef.child("order").child("DGMFkEM7JoXsEhSvgvq8NIVqR763").child(user.getUid()).addValueEventListener(new ValueEventListener() {

@Override

public void onDataChange(DataSnapshot dataSnapshot) {

info = dataSnapshot.getValue(ViewOrderInfo.class);

nameOfCustomer.setText(info.getName());

addressOfCustomer.setText(info.getAddress());

phoneNumber.setText(info.getPhone());

food.setText(info.getFood());

price.setText(info.getPrice());

}

@Override

public void onCancelled(DatabaseError databaseError) {

}

});

return view;

}

@Override

public void onActivityCreated(@Nullable Bundle savedInstanceState) {

super.onActivityCreated(savedInstanceState);

// Toast.makeText(getContext(),info.getName(), Toast.LENGTH\_LONG).show();

}

}

## Other relevant material

### UML DIAGRAMS

#### BPMN diagram.

The Business Process Model and Notation shows the flow of activities associated with the business process of the food delivery system. The business process only involves the customer and the restaurant.

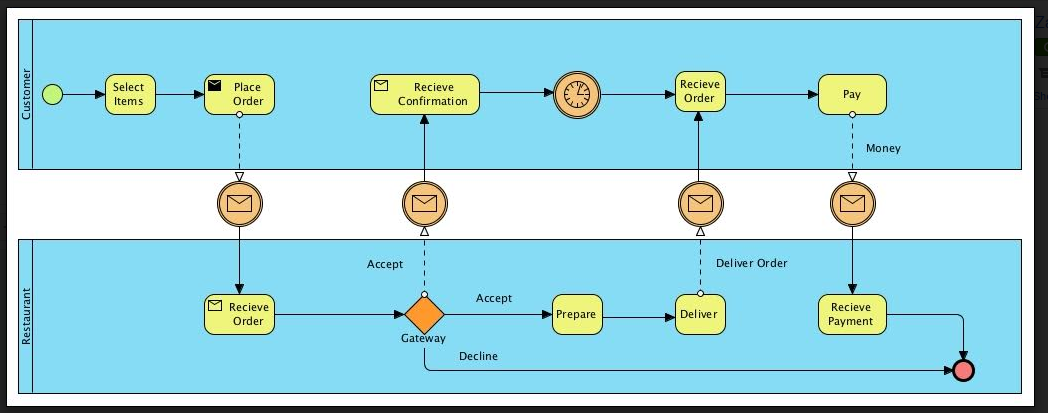


Figure . BPMN Diagram of Overall System

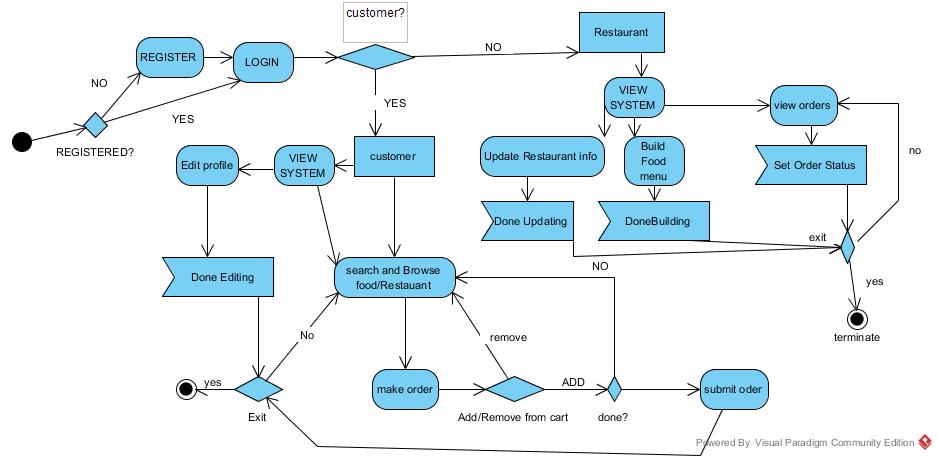


Figure . Activity diagram of Overall System