Mom & Pops Pizzeria
Requirements

Requirements Documents

Requirements Definition **3**
Functional Requirements 3-5
Non-Functional Requirements 5

Use Case Diagram **6**

Use Case Flow of Events **7-8**

UML Class Diagrams **9-12**

Class Documentation **13-14**

ER Diagram **15-17**

Decision Tables **18**
Customer Login Page 18
Account Creation Page 18
Employee Login Page 18
Order Page 19
Pizza Selection Page 19
Drink Selection Page 20
Sides Selection Page 20
Customer Payment Page 21
Employee Payment Page 21

Requirements Documents

Functional Requirements

1. Login- User should be able to login into our system. There is guest ordering access that allows any user to order. There must be account access that allows our system to store the customers information/data for easier ordering on subsequent orders.
2. Account- Creation can be done in store visits. Phone numbers, addresses and passwords are needed to establish an account.
3. Ordering- Users should be able to order from the entire menu. Once they have selected the items that are applicable, customization options should be available. Customers and employees should be able to select an option for pickup or delivery.
4. Online Ordering- Running total should be held for all items being ordered to let the customer know what their current total is.
5. Address- Only required if creating an account to have an address. Reasonable validation required.
6. Employee- If an employee is taking orders, use the employee login on the app to create orders and place orders.
7. Employee Computer- Employee Computer will be able to use the full menu and have a running total. The computer will also be able to print receipts or produce electronic receipts.
8. Pay- User should be able to go to checkout and enter in their information for payments. Returning users have access to saved payment methods for faster checkout.
9. Credit Card- Validation required for use of credit card via zip code.
10. Receipts- Receipts generated for orders. Price justified and CC line.
11. Payment -Methods for cash, credit, cc, and check
12. Customer order entry needs CC
13. Employee order entry needs all forms of payment.
14. CC needs to have CVV. validation of: 3-4 numerics.
15. CC must be 16 numeric characters long
16. CC expiration date must be in the future
17. Only Visa and MC are accepted
18. The following menu items and pricing must be used.

Crusts: thin, regular, pan
Sizes: small, medium, large, extra large
Sauce: only regular tomato based marinara sauce

Toppings: cheese, pepperoni, sausage, ham, green pepper, onion, tomato, mushroom, pineapple

Pricing $4-6-8-10. Cheese and 1 topping (first topping is free) No pre made combos

Extra toppings .50, .75, 1.00, 1.25 based on the size of the pizza

Drinks: Pepsi drinks (pepsi, diet pepsi, orange and diet orange, root beer and diet root beer, sierra mist and lemonade. Small, medium and large all priced at $1

Sides: bread sticks ($4), bread stick bites($2), big chocolate chip cookie($4)

19. Orders cannot exceed 10 pizzas
20. Receipts must be labeled “Customer Copy” and “Merchant Copy” with a signature line for merchant receipts.
21. Receipts should have the business name and address at the top.

    Business Address: 680 Arntson Rd, suite 161 Marietta, ga 30060

22. Orders must have a kitchen ticket that prints with the necessary order information for fulfillment.

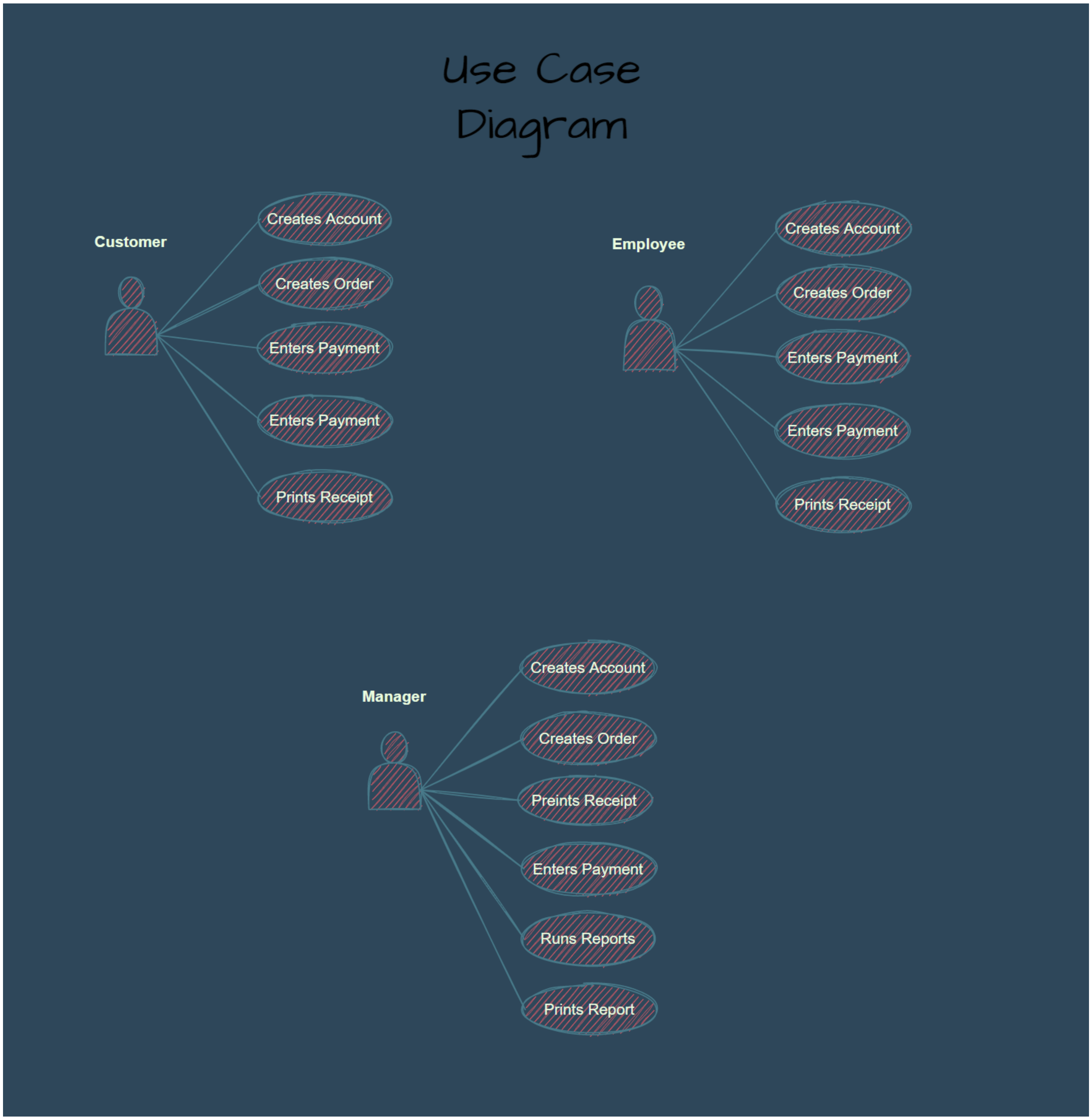
Non-Functional Requirements

23. Store app should be up and running/accepting orders from our scheduled hours of operation. Sunday-Thursday 11am-9pm, Friday and Saturday 11am- 12am
24. Store App should be nice to look at with not too much going on onscreen to dissuade customer use.
25. Security must be a high priority since customers are inputting credit card information and potentially addresses as well.
26. Manager should have training for employees that gets them comfortable with using the help and helping customers with the app as well.
27. System should have backed up files in case of a crash to protect from data loss.
28. The program should run efficiently and there should be a minimalistic amount of failure within the app.
29. The system should be designed in a way that will make future cross platform support a possibility. Future implementations are planned for the internet and mobile devices.
30. The app must be designed in such a way that any user's level of technical knowledge can be able to navigate the app.
31. Troubleshooting the app should not be a complex process.
32. Home screen should display 3 options: guest order, order with customer account, create customer account.
33. Input should be shown for returning customers (Phone Number and Password).
34. Account creation information spaces should be shown for user creating accounts.
35. The system should show the full menu items and their prices when selected.

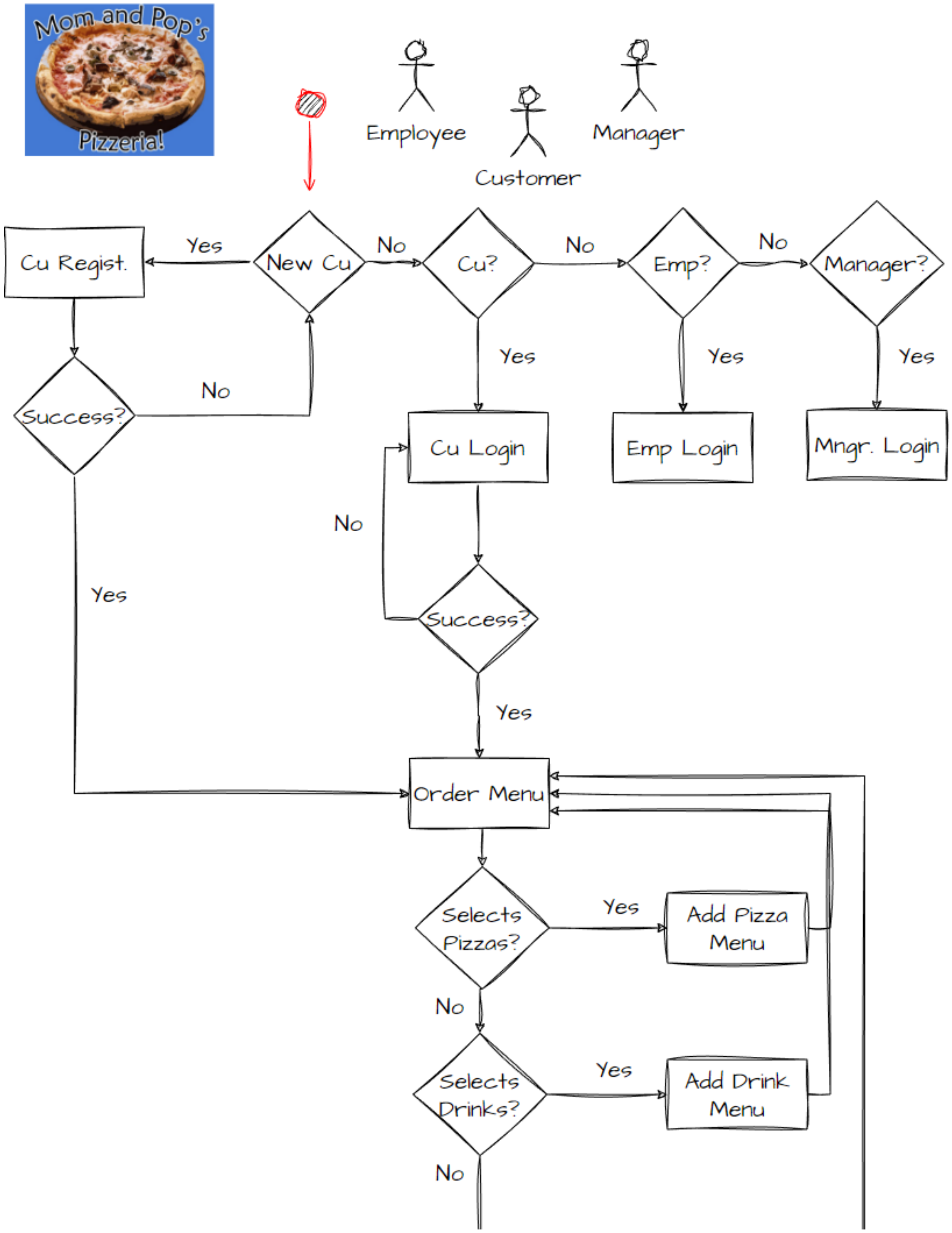
Requirements Documents

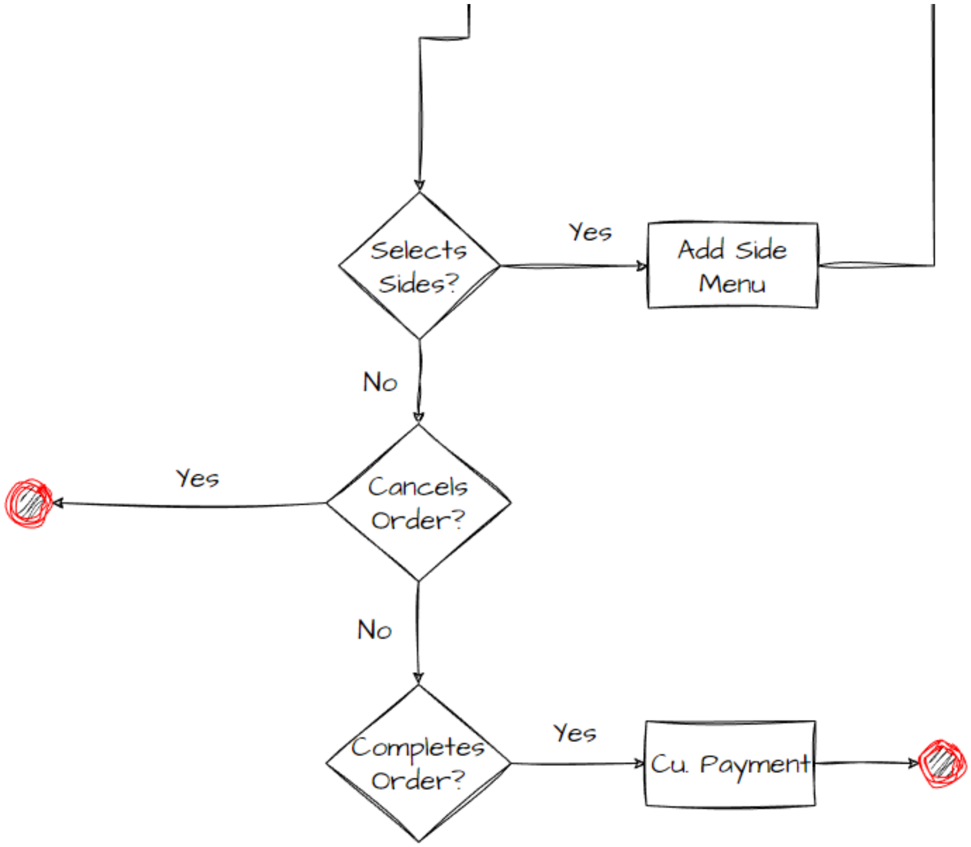
36. The different customization options should be shown by the system.
37. The system should show the order total.
38. The system should show where the order is taking place. (Delivery, Pickup, or Instore)
39. The system should show the input payment information.
40. The systems should show the receipt confirmation.
41. The system should show the customized order.
42. The system should show the customer information neatly and organized.
43. The system should allow the customer to edit their order.
44. The system should show a home button to return to the initial screen.

Requirements Documents



Use Case Flow of Events

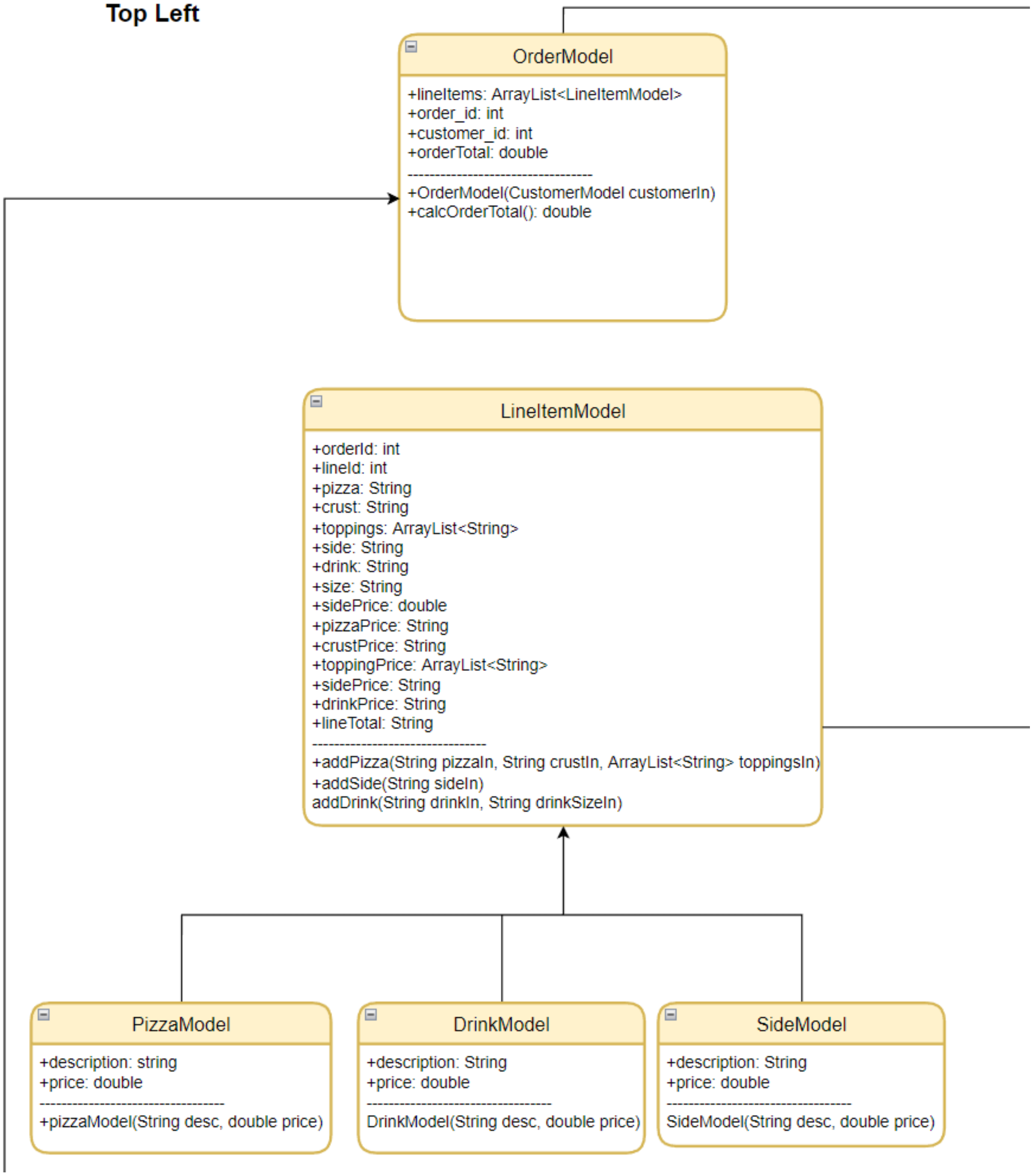



Selects Sides?
Yes
Add Side Menu
No
Yes
Cancels Order?
No
Completes Order?
Yes
Cu. Payment

Requirements Documents

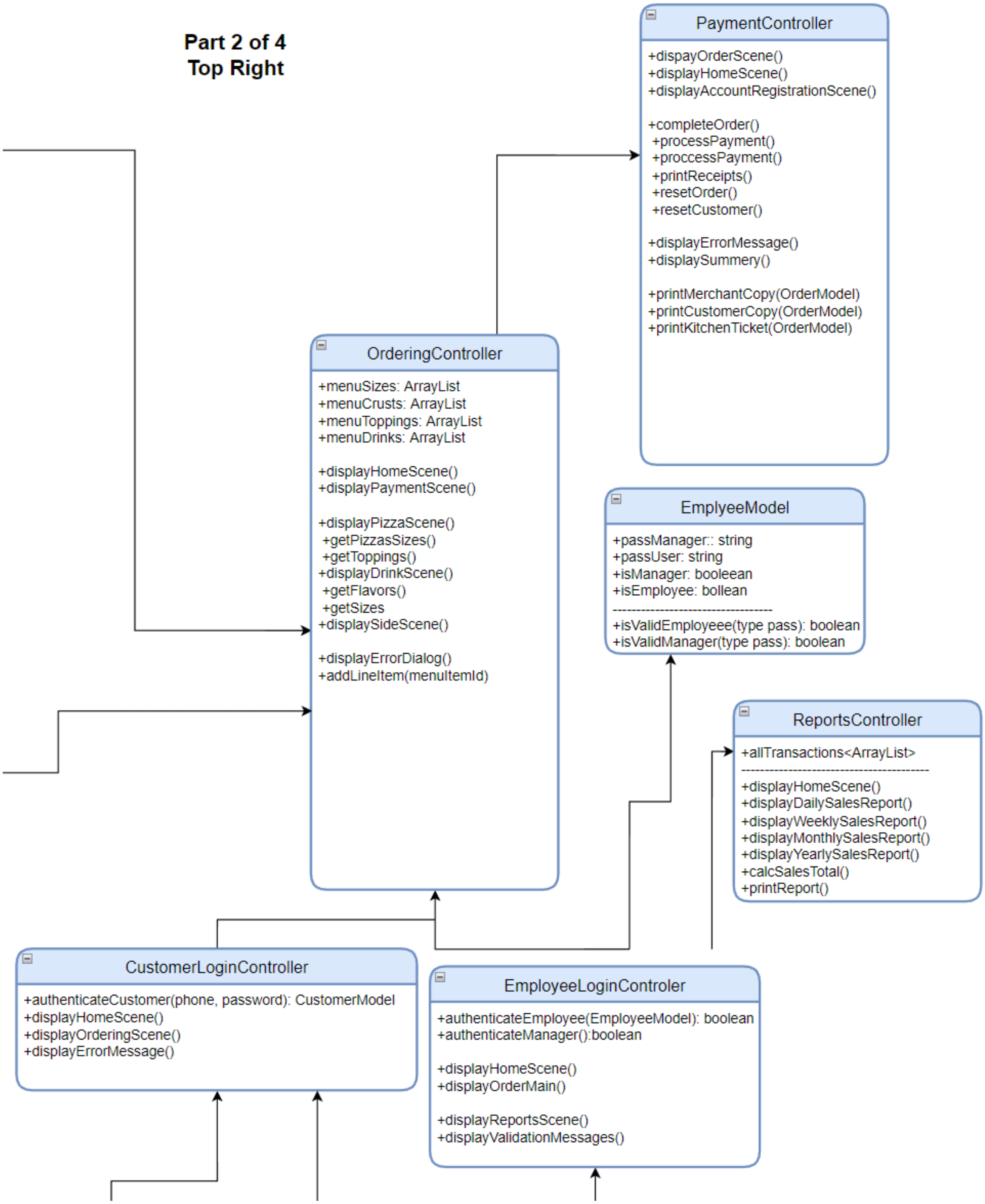
UML Class Diagrams

Part 1 of 4
Top Left



Requirements Documents

Part 2 of 4
Top Right



PaymentController

+dispayOrderScene()
+displayHomeScene()
+displayAccountRegistrationScene()

+completeOrder()
+processPayment()
+proccessPayment()
+printReceipts()
+resetOrder()
+resetCustomer()

+displayErrorMessage()
+displaySummery()

+printMerchantCopy(OrderModel)
+printCustomerCopy(OrderModel)
+printKitchenTicket(OrderModel)

OrderingController

+menuSizes: ArrayList
+menuCrusts: ArrayList
+menuToppings: ArrayList
+menuDrinks: ArrayList

+displayHomeScene()
+displayPaymentScene()

+displayPizzaScene()
+getPizzasSizes()
+getToppings()
+displayDrinkScene()
+getFlavors()
+getSizes
+displaySideScene()

+displayErrorDialog()
+addLineItem(menuItemId)

EmplyeeModel

+passManager:: string
+passUser: string
+isManager: booleean
+isEmployee: bollean

+isValidEmployeee(type pass): boolean
+isValidManager(type pass): boolean

ReportsController

+allTransactions<ArrayList>

+displayHomeScene()
+displayDailySalesReport()
+displayWeeklySalesReport()
+displayMonthlySalesReport()
+displayYearlySalesReport()
+calcSalesTotal()
+printReport()

CustomerLoginController

+authenticateCustomer(phone, password): CustomerModel
+displayHomeScene()
+displayOrderingScene()
+displayErrorMessage()

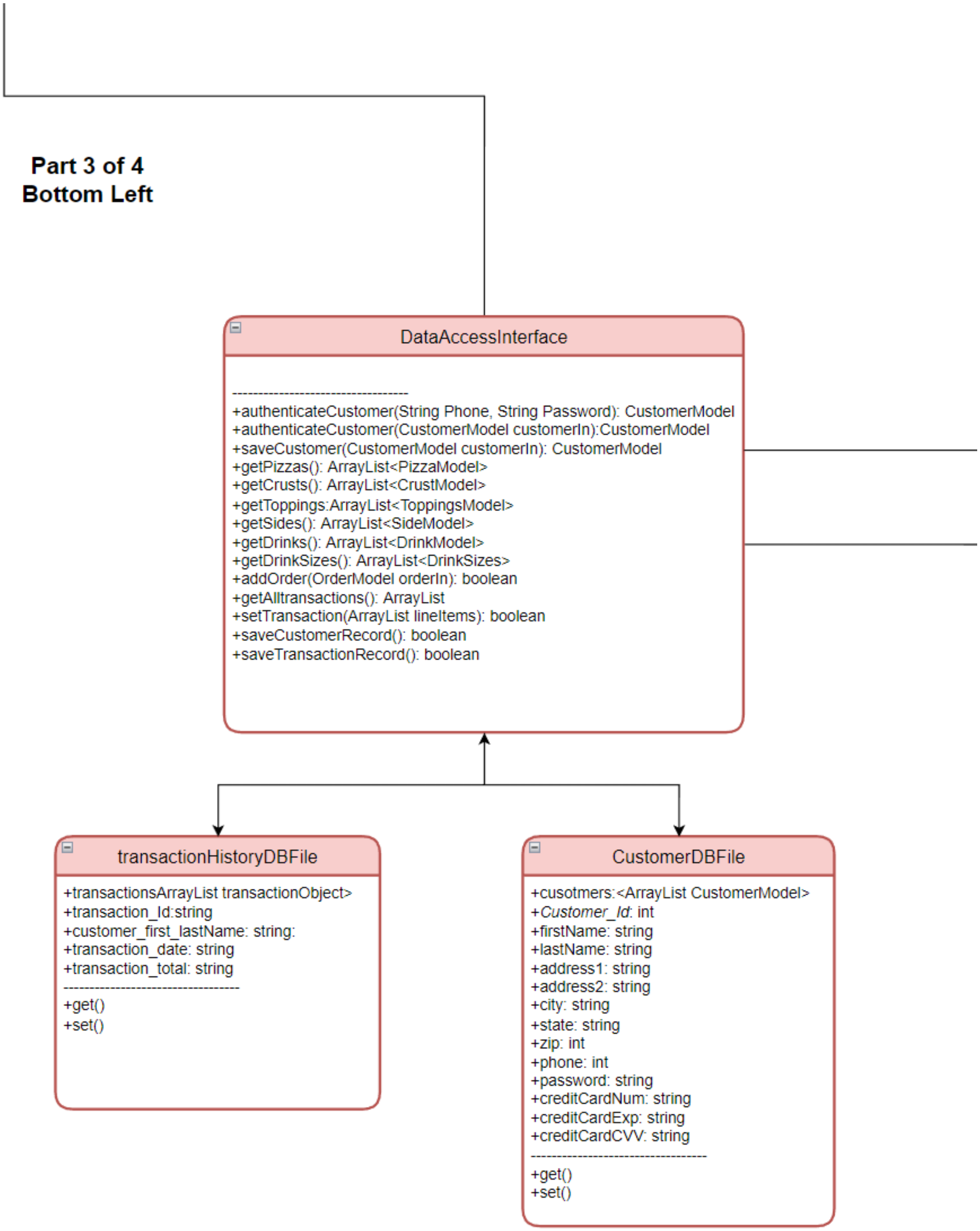
EmployeeLoginControler

+authenticateEmployee(EmployeeModel): boolean
+authenticateManager():boolean

+displayHomeScene()
+displayOrderMain()

+displayReportsScene()
+displayValidationMessages()

Part 3 of 4
Bottom Left



DataAccessInterface

```
-----------------------------------
+authenticateCustomer(String Phone, String Password): CustomerModel
+authenticateCustomer(CustomerModel customerIn):CustomerModel
+saveCustomer(CustomerModel customerIn): CustomerModel
+getPizzas(): ArrayList<PizzaModel>
+getCrusts(): ArrayList<CrustModel>
+getToppings:ArrayList<ToppingsModel>
+getSides(): ArrayList<SideModel>
+getDrinks(): ArrayList<DrinkModel>
+getDrinkSizes(): ArrayList<DrinkSizes>
+addOrder(OrderModel orderIn): boolean
+getAlltransactions(): ArrayList
+setTransaction(ArrayList lineItems): boolean
+saveCustomerRecord(): boolean
+saveTransactionRecord(): boolean
```

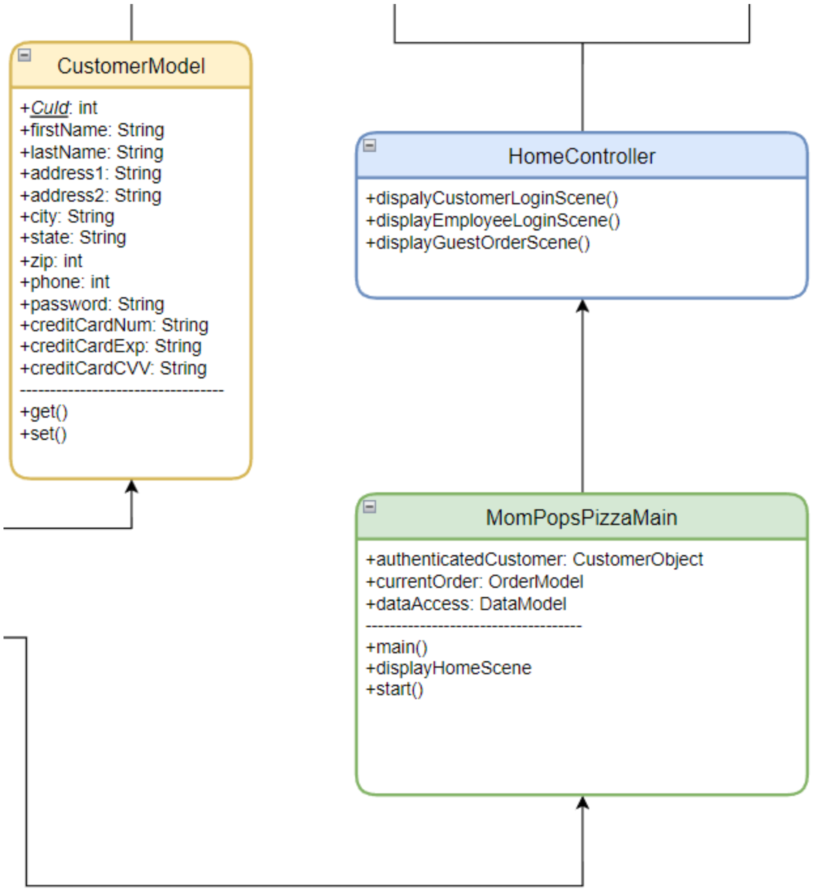
transactionHistoryDBFile

```
+transactionsArrayList transactionObject>
+transaction_Id:string
+customer_first_lastName: string:
+transaction_date: string
+transaction_total: string
-----------------------------------
+get()
+set()
```

CustomerDBFile

```
+cusotmers:<ArrayList CustomerModel>
+Customer_Id: int
+firstName: string
+lastName: string
+address1: string
+address2: string
+city: string
+state: string
+zip: int
+phone: int
+password: string
+creditCardNum: string
+creditCardExp: string
+creditCardCVV: string
-----------------------------------
+get()
+set()
```

Requirements Documents



Part 4 of 4
Bottom Right

Class Documentation

MomPopsPizzaMain

This class starts the application and contains the main method. It is a parent class that shares its global variables with the controllers. This includes a dataAccess object and an order object that are accessed throughout the ordering process. Lastly, it launches the home-View.fxml which relies on the HomeController. This is the first view that loads when the application is started.

HomeController

This controller class contains the methods for launching three views. The first is the CustomerLoginScene. The second is the EmployeeLoginScene. The third is the OrderScene that is directly accessible when the guest order option is selected.

CustomerLoginController

This controller class processes input and output for the customerLogin-view.fxml. It interacts with the dataAccess objects methods to validate customer credentials. It also is responsible for initializing the authenticatedCustomer global variable in MomPopsPizzaMain parent class.

EmployeeLoginController

This controller class is used by the empLogin-view.fxml to authenticate employees and managers. It instantiates an EmployeeModel class. Its methods take in user inputs for password and employee type. If either password is valid the authenticatedEmployee global variable from the MomPopsPizzaMain parent class is updated to reflect the authenticated state.

EmployeeModel

This object class contains static passwords for the manager and employee. The passwords are validated and will return true or false is authenticated. If either an employee or manager are validated the model is updated and remains available in memory so that the user type and authentication can be checked before the payment screen is displayed.

ReportsController

This class is used when a manger is logged in to retrieve historical transaction details from the dataAccess interface. It also contains methods for calculating and displaying reports. When the manager is done viewing reports, they may cancel the activity which will return the application to the home-view.fxml.

OrderingController

This class lies at the heart of the application and contains the methods that work in conjunction with the order-view.fxml to build orders. There are methods for launching the payment-view.fxml which relies on the payment controller. This class is dependent on the OrderModel class.

OrderModel

This class maintains an array list of LineItemModel class objects. As line items are added to it the order total is calculated and held for the OrderingController class. This class also retains the CustomerModel instance associated with the order. The order instance is instantiated by the MomPopsPizzaMain class when the program is started and reset at the end of each order cycle.

DataAccessInterface – Data

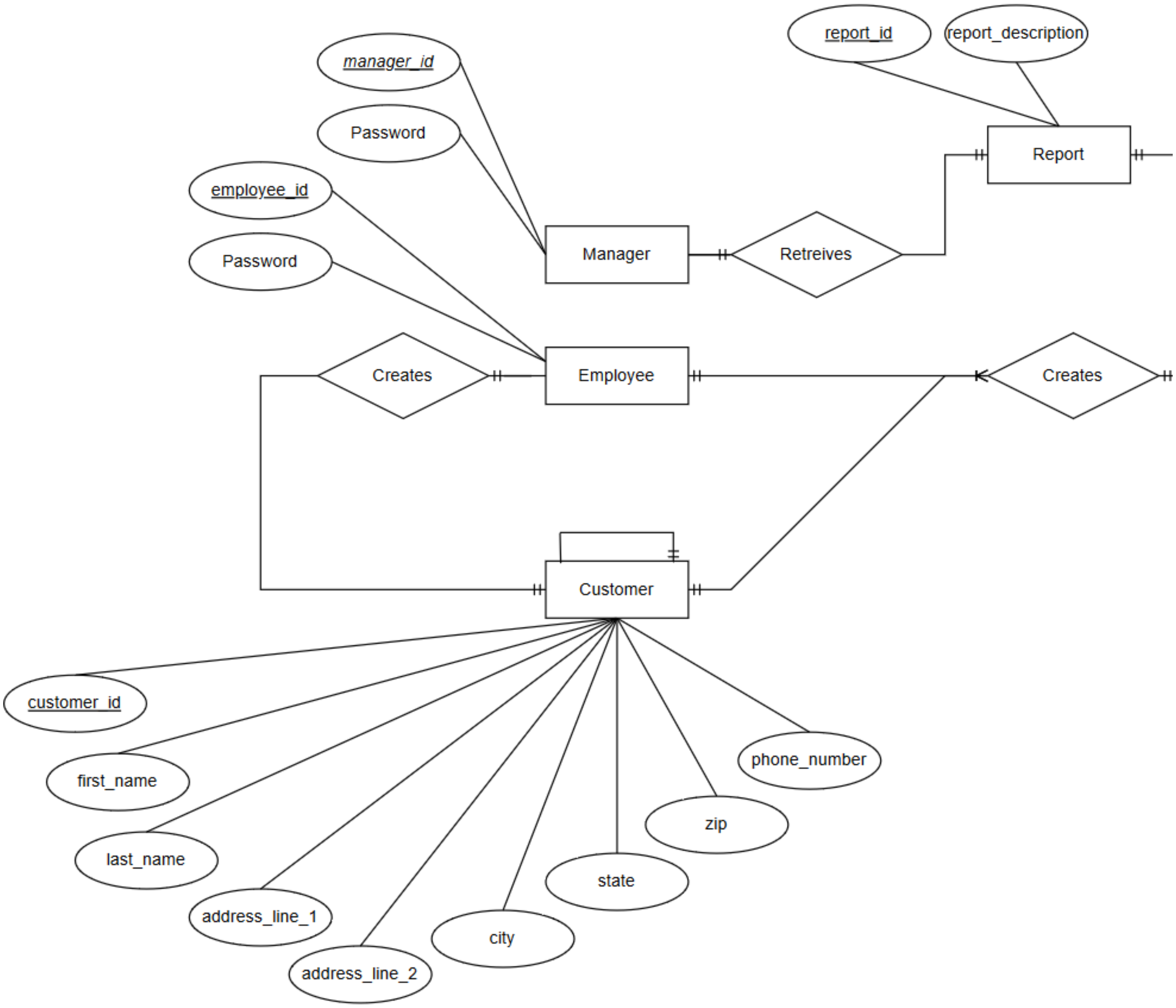
The Data class implements the DataAccessInterface class. This class is responsible for loading and maintaining the Customer.txt and transactions.txt data files. It is instantiated by the MomPopsPizzaMain class at application startup. There are methods for all necessary CRUD operations that are used by most all the controllers.

Requirements Documents

+The diagram below was created with an AWS cloud MySQL relational database in mind. The initial development phase will utilize text files with a simplified format for data retention.

ER Diagram

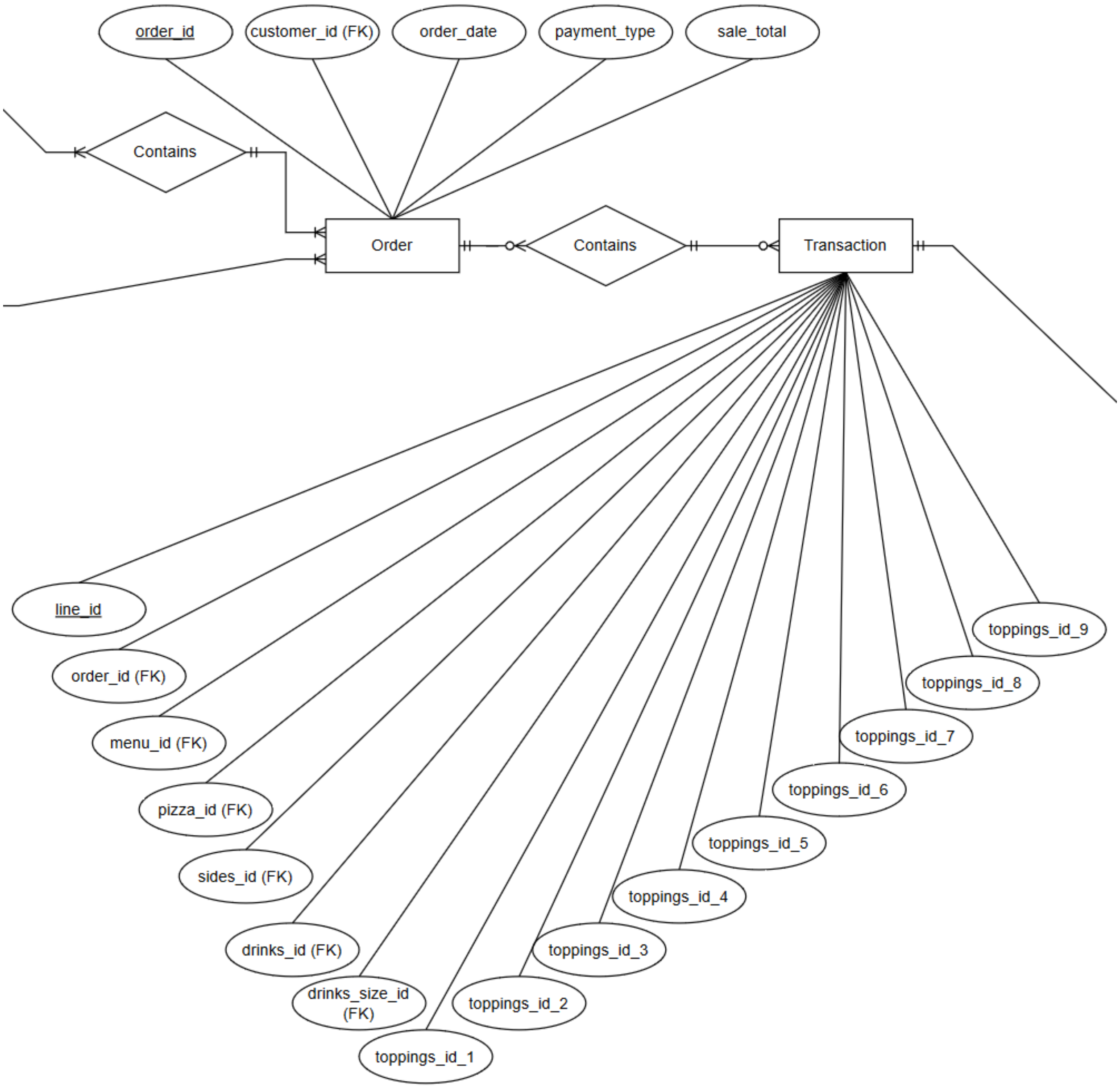
Page 1.



Requirements Documents

ER Diagram

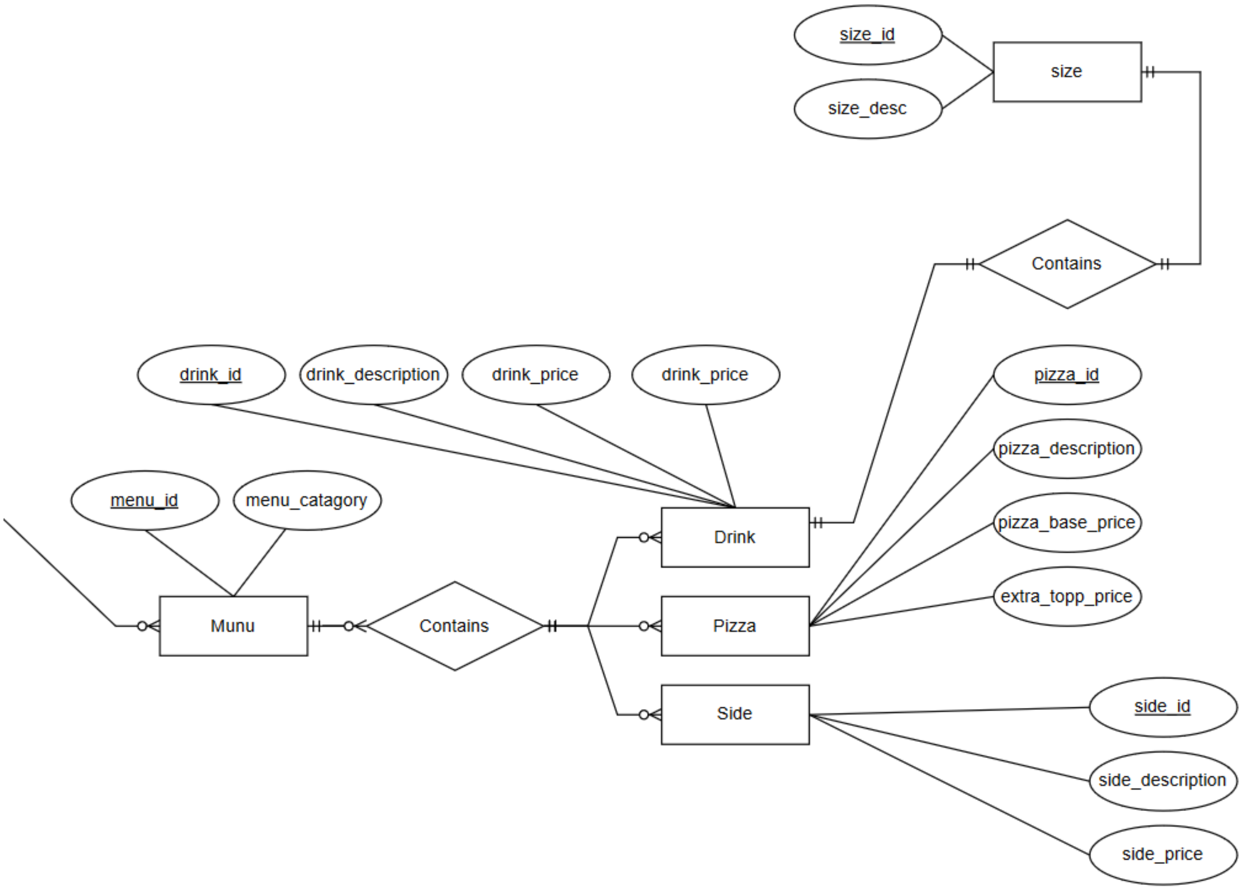
Page 2.



Requirements Documents

ER Diagram

Page 3.



Decision Tables

Customer Login Page

Conditions	**Case 1**	**Case 2**	**Case 3**	**Case 4**	**Case 5**	**Case 6**
Button Pressed	Login	Login	Login	Login	Login	Cancel
Phone Number	Valid	Valid	Valid	Invalid	Invalid	n/a
Password	Valid	Valid	Invalid	Valid	Invalid	n/a
Credentials	In database	Not in database	Not in database	Not in database	Not in database	n/a
Output	Menu	Error message: user not found	Error message: invalid password	Error message: invalid phone number	Error message: invalid username and password	Return home

Account Creation Page

Conditions	**Case 1**	**Case 2**	**Case 3**	**Case 4**	**Case 5**	**Case 6**
Button Pressed	Create Account	Create Account	Create Account	Back	Back	Home
User input	Valid	Valid	Invalid	n/a	n/a	n/a
Has already ordered	No	Yes	n/a	Yes	No	n/a
Output	Menu	Complete order, Return home	Error message: invalid input	Return to payment processing	Return home	Return home

Employee Login Page

Conditions	**Case 1**	**Case 2**	**Case 3**	**Case 4**	**Case 5**
Button Pressed	OK	OK	OK	OK	Cancel
Employee Type	Employee	Employee	Manager	Manager	n/a
Password	Valid	Invalid	Valid	Invalid	n/a
Output	Menu	Error message: invalid	Reports	Error message: invalid	Return home

		password		password	

Order Page

Conditions	Case 1	Case 2	Case 3	Case 4	Case 5	Case 6	Case 7
Button Pressed	Check Out	Check Out	Check Out	Check Out	Back	Back	Home
Has selected Item(s)	Yes	Yes	No	No	n/a	n/a	n/a
Is Employee?	Yes	No	Yes	No	Yes	No	n/a
Output	Employee Payment Page	Customer Payment Page	Error message: No items selected	Error message: No items selected	Return to Employee Login	Return to Customer Login	Return Home

Pizza Selection Page

Conditions	Case 1	Case 2	Case 3	Case 4	Case 5	Case 6	Case 7	Case 8	Case 9	Case 10
Button Pressed	Add Pizza	Add Pizza	Add Pizza	Add Pizza	Add Pizza	Add Pizza	Add Pizza	Add Pizza	Back	Home
Size Selected?	Yes	Yes	Yes	Yes	No	No	No	No	n/a	n/a
Crust Selected?	Yes	Yes	No	No	Yes	Yes	No	No	n/a	n/a
Toppings Selected?	Yes	No	Yes	No	Yes	No	Yes	No	n/a	n/a
Output	Add to order, Return to Menu	Add to order, Return to Menu	Error message: select a crust	Error message: Select a crust	Error message: Select a size	Error message: Select a size	Error message: Select a size and a crust	Error message: Select a size and a crust	Return to Menu	Return Home

Requirements Documents

Drink Selection Page

Conditions	Case 1	Case 2	Case 3	Case 4	Case 5	Case 6	Case 7	Case 8	Case 9	Case 10
Button Pressed	Add to Order	Add to Order	Add to Order	Add to Order	Add to Order	Add to Order	Add to Order	Add to Order	Back	Home
Flavor Selected?	Yes	Yes	Yes	Yes	No	No	No	No	n/a	n/a
Size Selected?	Yes	Yes	No	No	Yes	Yes	No	No	n/a	n/a
Quantity Selected?	Yes	No	Yes	No	Yes	No	Yes	No	n/a	n/a
Output	Add to order, Return to Menu	Error message: select quantity	Error message: select size	Error message: select size and quantity	Error message: select flavor	Error message: select flavor and quantity	Error message: select flavor and size	Error message: select flavor, size, and quantity	Return to Menu	Return Home

Sides Selection Page

Conditions	Case 1	Case 2	Case 3	Case 4
Button Pressed	Add All	Add All	Back	Home
At least one item selected?	Yes	No	n/a	n/a
Output	Add selections to order, Return to Menu	Error message: select at least one item	Return to Menu	Return Home

Requirements Documents

Customer Payment Page

Conditions	**Case 1**	**Case 2**	**Case 3**	**Case 4**	**Case 5**	**Case 6**
Button pressed	Pay Now	Pay Now	Pay Now	Pay Now	Back	Home
Credentials	Valid	Valid	Invalid	Invalid	n/a	n/a
Is Logged In?	Yes	No	Yes	No	n/a	n/a
Output	Complete Order, Print receipt, Return Home	Create Account? Page	Error message: invalid credentials	Error message: invalid credentials	Return to Menu	Return Home

Employee Payment Page

Conditions	**Case 1**			
Button Pressed	Complete	Complete	Back	Home
Amount Due	>$0.00	$0.00	n/a	n/a
Output	Error message: Additional Payment Required	Complete order, Print receipt, Return Home	Return to Menu	Return Home