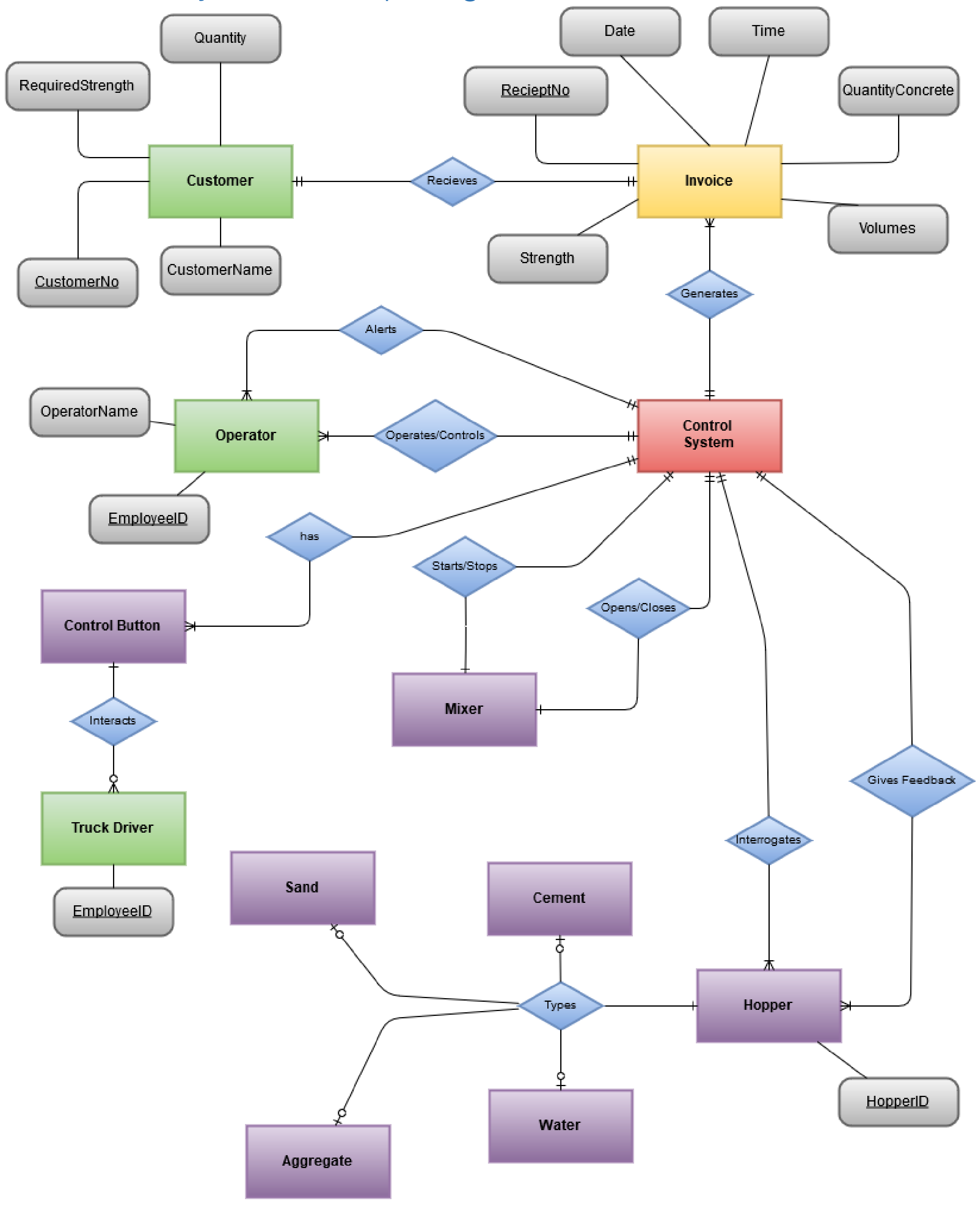
CONCRETE MIXING PLANT

Design Specification

Marc Colin Berman - 23438738
FIT3037 – Software Engineering

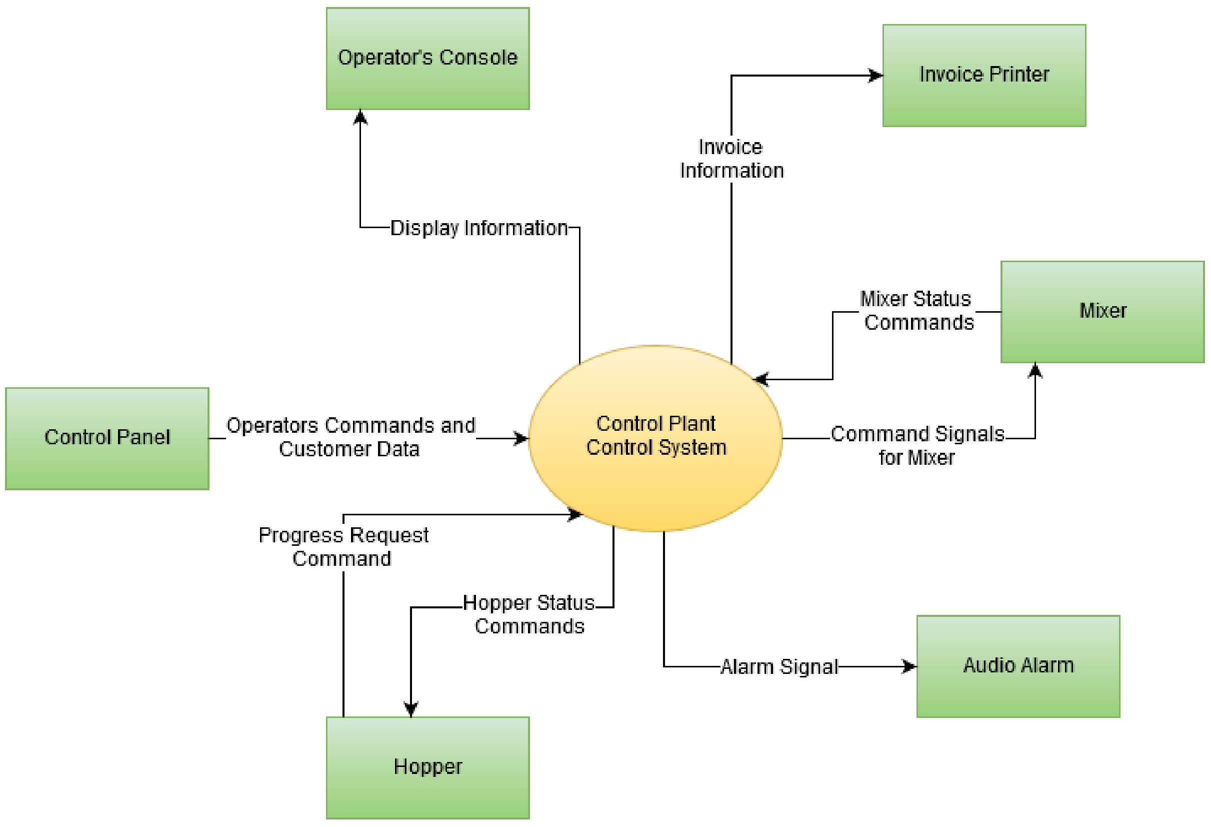
Table of Contents

External Entity Relationship Diagram

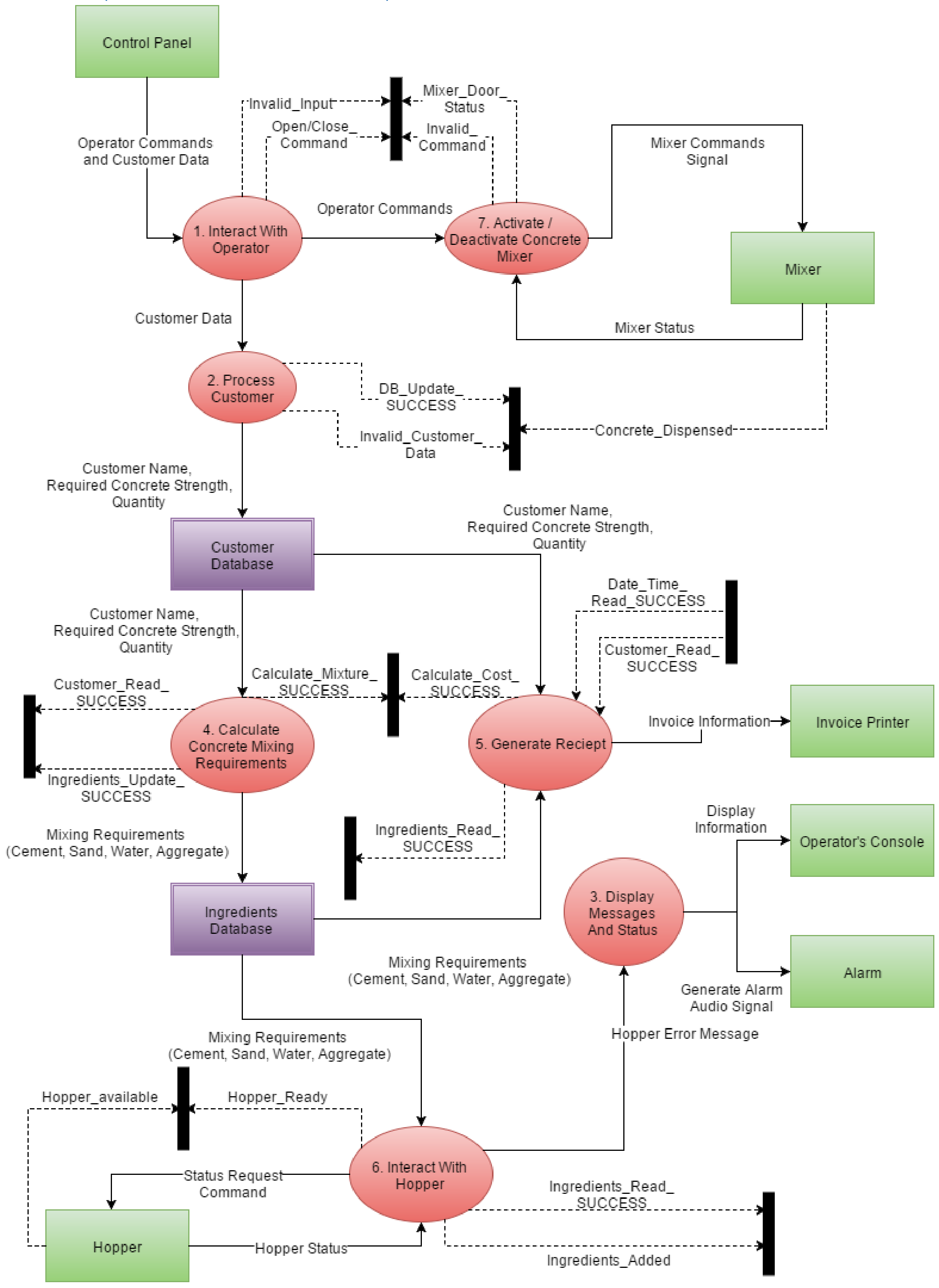


Data Flow Diagrams

Level 0

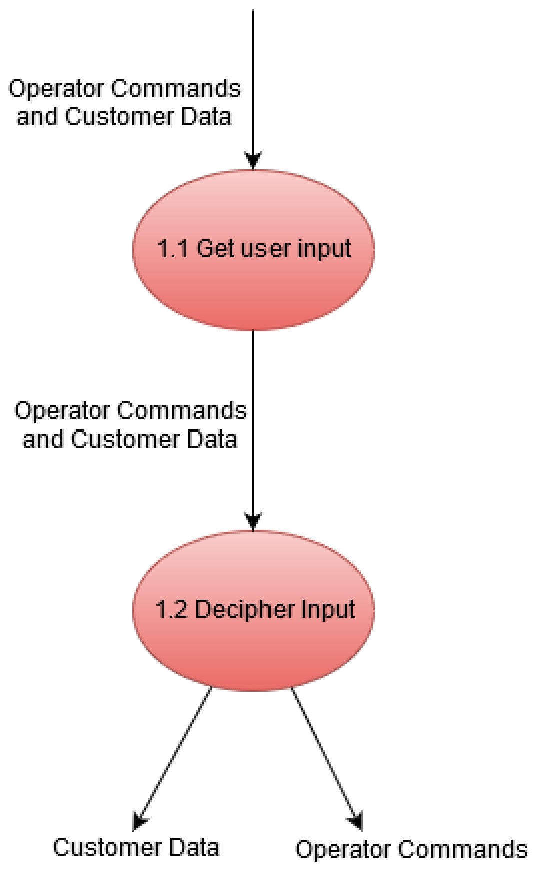


Level 1 (Data and Control Flow)

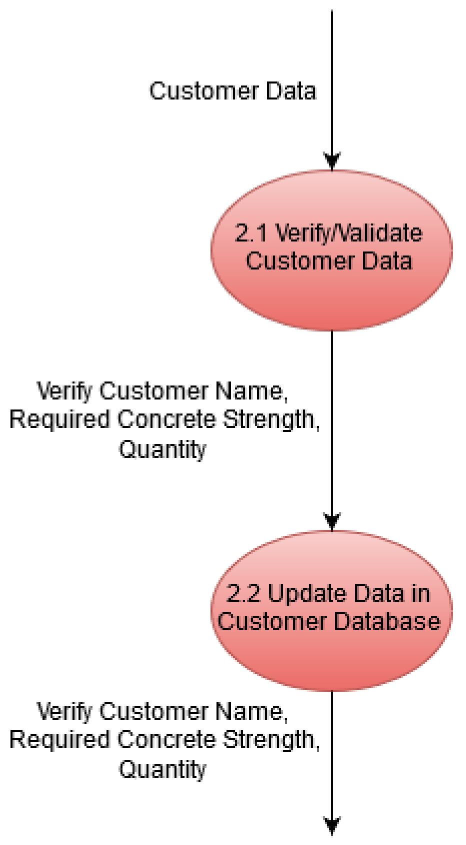


Level 2

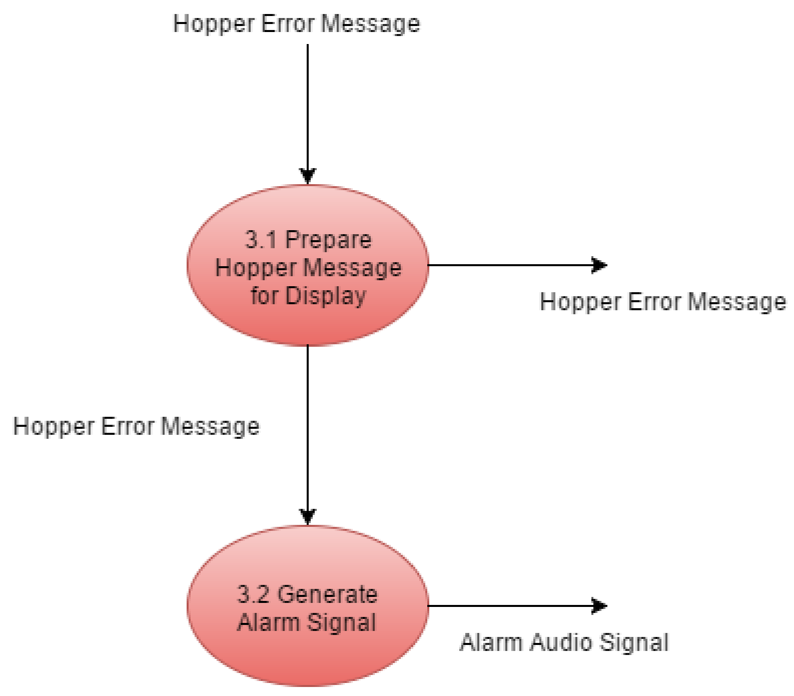
1) User Interaction Process


Operator Commands and Customer Data
1.1 Get user input
Operator Commands and Customer Data
1.2 Decipher Input
Customer Data
Operator Commands

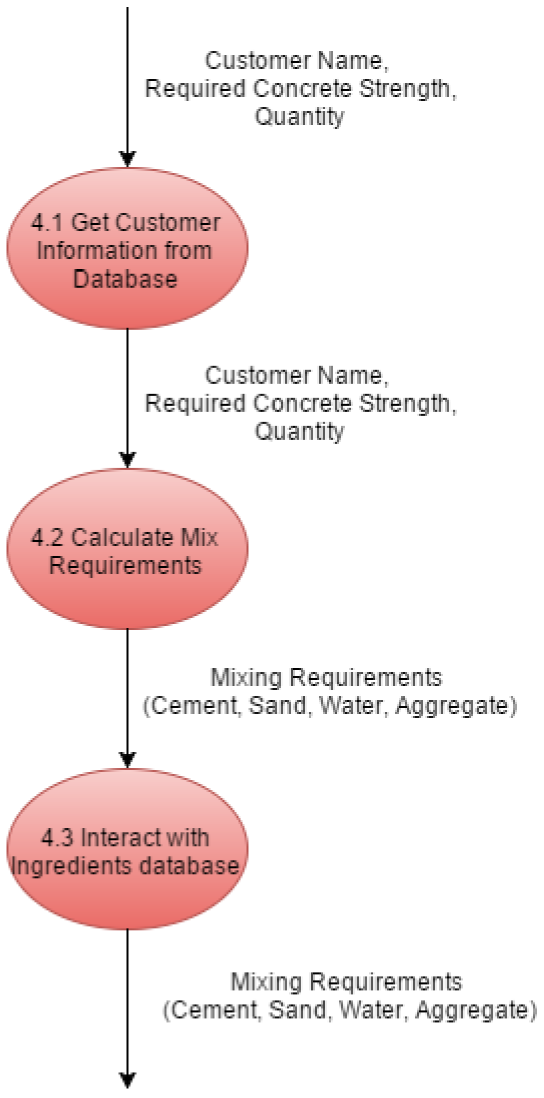
2) Customer Process


Customer Data
2.1 Verify/Validate Customer Data
Verify Customer Name, Required Concrete Strength, Quantity
2.2 Update Data in Customer Database
Verify Customer Name, Required Concrete Strength, Quantity

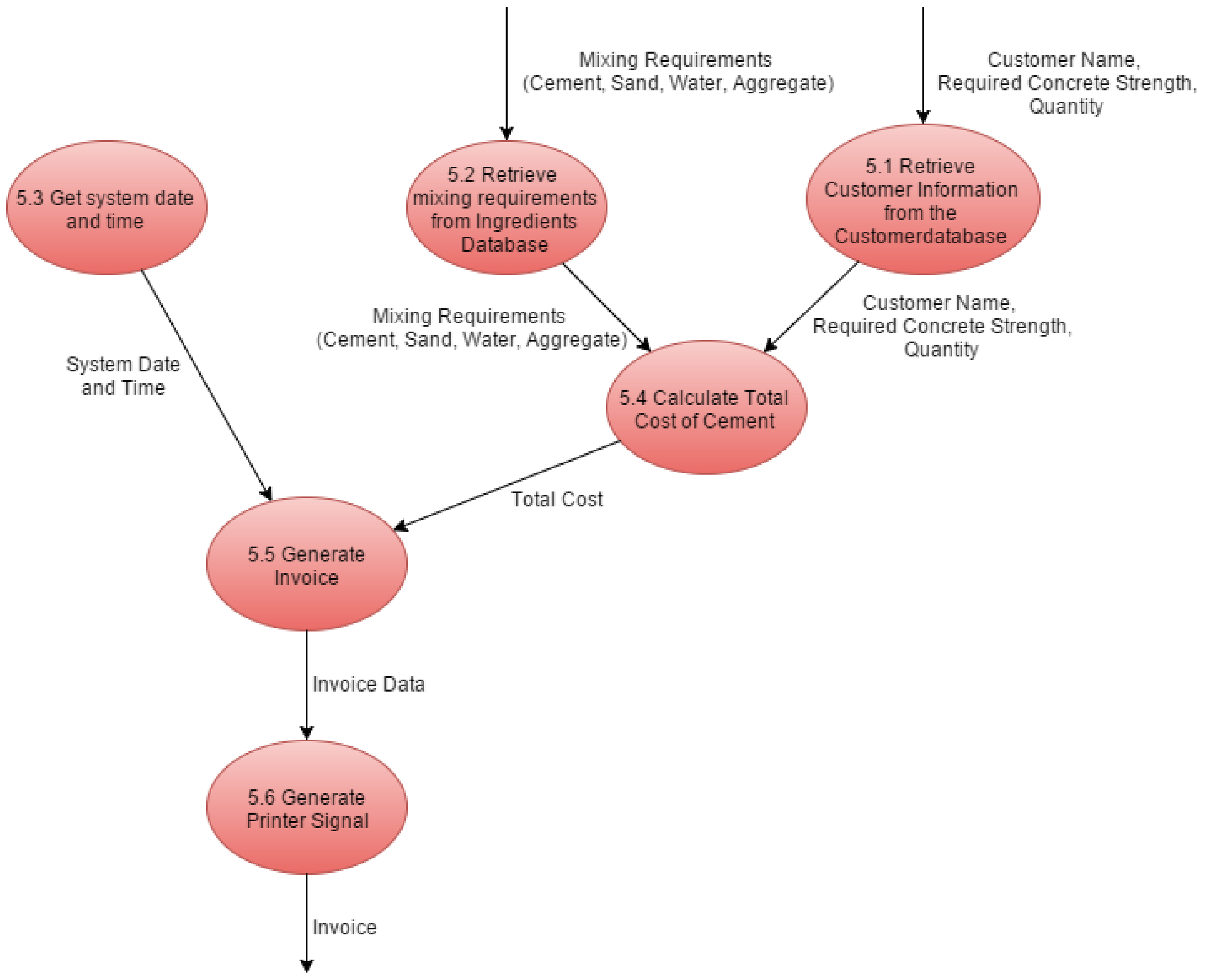
3) Alert Process



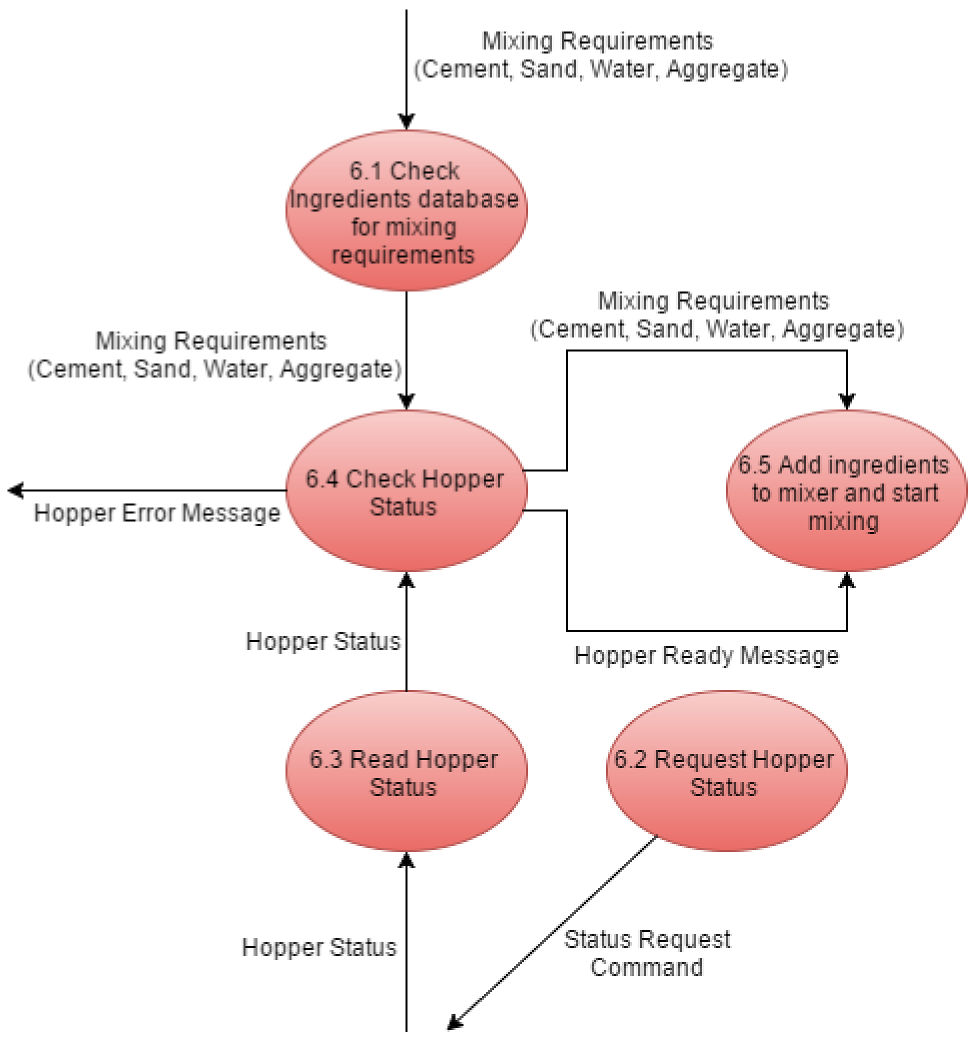
4) Calculate Mix Requirement Process


Customer Name, Required Concrete Strength, Quantity
4.1 Get Customer Information from Database
Customer Name, Required Concrete Strength, Quantity
4.2 Calculate Mix Requirements
Mixing Requirements (Cement, Sand, Water, Aggregate)
4.3 Interact with Ingredients database
Mixing Requirements (Cement, Sand, Water, Aggregate)

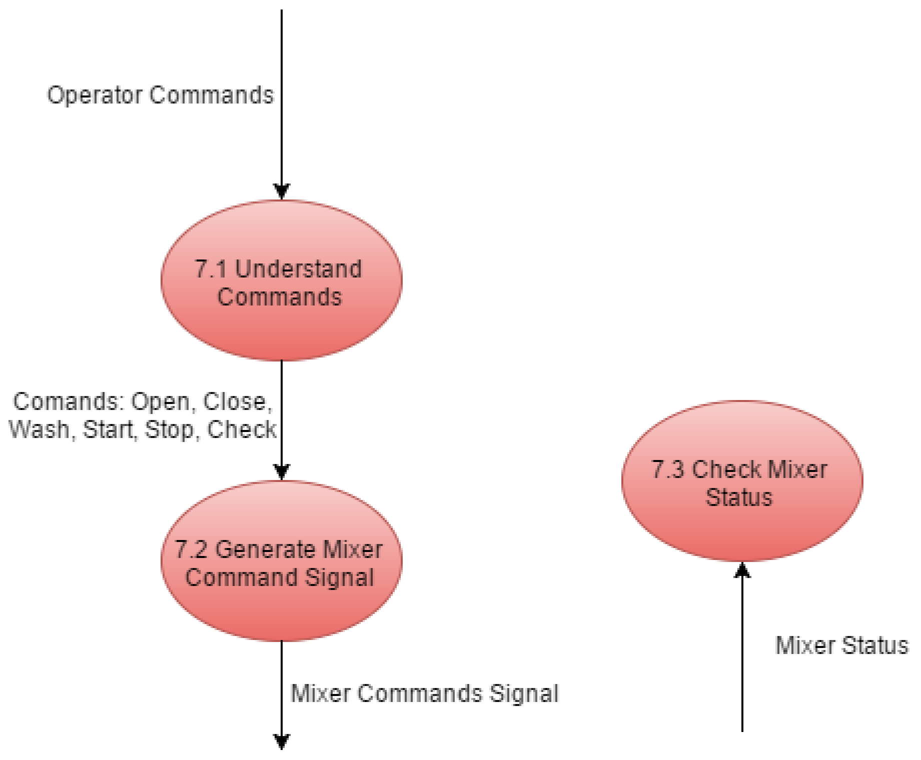
5) Invoice Generation Process


Mixing Requirements (Cement, Sand, Water, Aggregate)
Customer Name, Required Concrete Strength, Quantity
5.3 Get system date and time
5.2 Retrieve mixing requirements from Ingredients Database
5.1 Retrieve Customer Information from the Customerdatabase
Mixing Requirements (Cement, Sand, Water, Aggregate)
Customer Name, Required Concrete Strength, Quantity
System Date and Time
5.4 Calculate Total Cost of Cement
Total Cost
5.5 Generate Invoice
Invoice Data
5.6 Generate Printer Signal
Invoice

6) Hopper Mixing Process


Mixing Requirements (Cement, Sand, Water, Aggregate)
6.1 Check Ingredients database for mixing requirements
Mixing Requirements (Cement, Sand, Water, Aggregate)
Mixing Requirements (Cement, Sand, Water, Aggregate)
6.4 Check Hopper Status
6.5 Add ingredients to mixer and start mixing
Hopper Error Message
Hopper Status
Hopper Ready Message
6.3 Read Hopper Status
6.2 Request Hopper Status
Hopper Status
Status Request Command

7) Mixing Control Process



Process Specification

1) User Interaction Process

Process:	1.1 – Get User Input
Description:	This process gets the input from the user via the control panel. There are two types of input: Customer details with an order or control commands for the mixers. The input is assigned an ID type in order to determine the

Pseudo-Code:	`Variables: USER_INPUT`
	`    READ(USER_INPUT)`

Process:	1.2 – Decipher Input
Description:	This process receives the read user input from process 1.1. The processing determines if the ID is customer data which is passed to process 2.1. If the ID is a command then it is passed to process 7.1.

Pseudo-Code:	`IF  USER_INPUT[ID] = 1 THEN`
	`     VALIDATE(USER_INPUT)`
	`IF  USER_INPUT[ID] = 2 THEN`
	`     DECIPHER_COMMAND(USER_INPUT)`

2) Customer Process

Process:	2.1 – Verify/Validate Customer Data
Description:	This process is designed to verify/validate the customer data that has been passed from Process 1.2. The data consists of 3 major variables (Name, Required strength and Quantity). The Data is then validated.

Pseudo-Code:	`Public validate(USER_INPUT){`
	`Boolean flag = false`
	`IF  length(USER_INPUT.getCustomerName()) > 30  THEN`
	`    flag = true`
	`IF  !contains(USER_INPUT.getCustomerName(),”!@#$%^&*()”) THEN`
	`    flag = true`
	`IF  USER_INPUT.getStrength()  <= 10 AND USER_INPUT.getStrength() >= 35  THEN`
	`    flag = true`
	`IF USER_INPUT.getQuantity()  <= 1 AND USER_INPUT.getQuantity () >= 50  THEN`
	`    flag = true`
	`IF  flag = false THEN {`
	`   updateCustomer(USER_INPUT)`
	`}`
	`}`

Process:	2.2 – Update Data in Customer Database
Description:	This process simply updates the database via the required module with the data passed from Process 2.1.

Pseudo-Code:	`Public updateCustomer(USER_INPUT){`
	`Update(USER_INPUT)`
	`}`

3) Alert Process

Process:	3.1 – Prepare Hopper Message for Display
Description:	This process receives the Hopper Error Message from Process 6.2 and formats it correctly to be displayed.

Pseudo-Code:	`Public toString(Hopper_Status_Message){`
	`Print(Hopper_Status_Message)`
	`}`

Process:	3.2 – Generate Alarm Signal
Description:	This process simply generates the audio signal to notify the operator that input is required.

Pseudo-Code:	`Audio_signal = new AudioSignal()`
	`Stream(audio_signal)`

4) Calculate Mix Requirement Process

Process:	4.1 – Get Customer Information from Database
Description:	This process retrieves the customer information and requirements in the database and passes the information to Process 4.2 for calculating.

Pseudo-Code:	`Public processOrder() {`
	`Customer = retrieveCustomer(CustomerID)`
	`custName = customer.getName()`
	`Strength = customer.getStrength()`
	`Quantity = customer.getQuantity()`
	`calculateRequirements(custName, strength, Quantity)`
	`}`

Process:	4.2 – Calculate Mix Requirements
Description:	This process calculates the cement, sand, aggregate, water requirements based on the input from Process 4.1. The formula to calculate the requirements is still to be provided.

Pseudo-Code:	`Public calculateRequirements(custName, strength, Quantity){`
	`Input(custName, strength, Quantity)`
	`Processing: The formula then outputs an array with the requirements.`
	
	`updateIngredients(Array(cement, sand, aggregate, water))`
	`}`

Process:	4.3 – Interact with Ingredients Database
Description:	This process simply updates the Ingredients database with the details from Process 4.2.

Pseudo-Code:	`Public updateIngredients(Array(cement, sand, aggregate, water)){`
	`Update(Array(cement, sand, aggregate, water)`
	`}`

5) Invoice Generation Process

Process:	5.1 – Retrieve Customer Information from the Customer Database
Description:	This process retrieves the customer details from the database and passes the information to Process 5.4

Pseudo-Code:	`Public retrieveCustomer(CustomerID) {`
	`Customer = retrieveCustomer(CustomerID)`
	`custName = customer.getName()`
	`Strength = customer.getStrength()`
	`Quantity = customer.getQuantity()`
	`Return custName, strength, Quantity`
	`}`

Process:	5.2 – Retrieve mixing requirements from Ingredients Database
Description:	This process retrieves the ingredient details from the database and passes the information to Process 5.4

Pseudo-Code:	`Public retrieveIngredients(CustomerID) {`
	`Ingred = retrieveIngredients(CustomerID)`
	`cement = customer.getCement()`
	`sand = customer.getSand ()`
	`aggregate = customer.getAggregate()`
	`water = customer.getWater ()`
	`return cement, sand, aggregate, water`
	`}`

Process:	5.3 – Get system date and time
Description:	This process simply checks the current date of the system for the mixture and returns the value to Process 5.5
Pseudo-Code:	Return systemDateTime

Process:	5.4 Calculate Total Cost of Cement
Description:	This process calculates the total cost of the mixture before passing the cost to Process 5.5. The cost is calculated by adding all the costs of each ingredient per weight, the cost based on the strength, and a small markup with tax.
Pseudo-Code:	`Public calculeCost(CustomerID) {` `retrieveIngredients(CustomerID)` `retrieveCustomer(CustomerID)` `Total = ((cement x weight) + (sand x weight) + (aggregate x weight) + (water x weight) + strength + quantity) x (markup + 0.14)` `Return total` `}`

Process:	5.5 – Generate Invoice
Description:	This process generates the invoice for the customer. It receives the data and time from Process 5.2 and total cost from Process 5.4. This data will be passed in the form of a printer signal to be printed in Process 5.6
Pseudo-Code:	Invoice = generateInvoice(dataTime, total) Stream = new PrinterStream(Invoice) Return stream

Process:	5.6 – Generate Printer Signal
Description:	This process receives the printer stream and then send the invoice to be printed.
Pseudo-Code:	Print (Stream)

6) Hopper Mixing Process

Process:	6.1 – Check Ingredients Database for Mixing Requirements
Description:	This process retrieves ingredient mixing requirements information from the database and passes it to process 6.4
Pseudo-Code:	`Public retrieveIngredients(CustomerID) {` `Ingred = retrieveIngredients(CustomerID)` `cement = customer.getCement()` `sand = customer.getSand ()` `aggregate = customer.getAggregate()` `water = customer.getWater ()` `return ingredients_requirements(cement, sand, aggregate, water)` `}`

Process:	6.2 – Request Hopper Status
Description:	This process requests the status of the hopper
Pseudo-Code:	`Request  hopper_status`

Process:	6.3 – Read Hopper Status
Description:	This process returns the status of the hopper being requested. This is passed to Process 6.4
Pseudo-Code:	`Return hopper_status`

Process:	6.4 – Check Hopper Status
Description:	This process determines if a hopper s available to be used and if the hopper has the required ingredients or if it needs additional ingredients to be added. If there is an error then this is passed to Process 3.1 and 3.2. If the hopper is ready then Process 6.5 receives the hopper ready status and proceeds.
Pseudo-Code:	`If hopper_status = ingredients_requirements` `    Then return processMixture()` `    Else return toString(hopper_status)`

Process:	6.5 – Add Ingredients to mixer and start mixing
Description:	The hopper receives the status message and proceeds to add ingredients to the hopper in order to begin mixing.
Pseudo-Code:	`Public processMixture() {` `Dispense(cement, hopperID)` `Dispense(sand, hopperID)` `Dispense(aggregate, hopperID)` `Dispense(water, hopperID)`

7) Mixing Control Process

Process:	7.1 – Understand Commands
Description:	This process receives the operator's commands and determines what action needs to be taken. This is then passed to Process 7.2

Pseudo-Code:	`Case USER_INPUT of`
	`  0x010 =  input = "Open Door";`
	`  0x011 =  input = "Close Door";`
	`  0x012 =  input =  "Status";`
	`  0x013 =  input = "Start";`
	`  0x014 =  input = "Stop";`
	`End case`
	`Return input`

Process:	7.2 – Generate Mixer Command Signal
Description:	This process generates the actual signal to complete the required operation.

Pseudo-Code:	`If input = "Open Door"`
	`Then genOpen()`
	`Else If input = "Close Door"`
	`Then genClose()`
	`Else If input = "Status"`
	`Then genStatus()`
	`Else If input = "Start"`
	`Then genStart()`
	`Else If input = "Stop"`
	`Then genStop()`

Process:	7.3 – Check Mixer Status
Description:	This process determines the status of the Mixer
Pseudo-Code:	`Return mixer_status`

Process Activation Table

Action	Process	
	1) Interact With Operator	2) Process Customer
Invalid_Input	True	
Open/Close_Command	True	
DB_Update_SUCCESS		True
Invalid_Customer_Data		True
Customer_Read_ SUCCESS		
Ingredients_Update_ SUCCESS		
Calculate_Mixture_ SUCCESS		
Calculate_Cost_ SUCCESS		
Ingredients_Read_ SUCCESS		
Date_Time_Read_SUCCESS		
Customer_Read_Success		
Hopper_Available		
Hopper_Ready		
Ingredients_Read_ SUCCESS		
Ingredients_Added		
Mixer_Door_Status		
Invalid_Command		
Concrete_Dispensed		

Action	Process	
	3) Display Messages And Status	4) Calculate Concrete Mixing Requirements
Invalid_Input		
Open/Close_Command		
DB_Update_SUCCESS		
Invalid_Customer_Data		
Customer_Read_ SUCCESS		True
Ingredients_Update_ SUCCESS		True
Calculate_Mixture_ SUCCESS		True
Calculate_Cost_ SUCCESS		
Ingredients_Read_ SUCCESS		
Date_Time_Read_SUCCESS		
Customer_Read_Success		
Hopper_Available		
Hopper_Ready		
Ingredients_Read_ SUCCESS		
Ingredients_Added		
Mixer_Door_Status		
Invalid_Command		
Concrete_Dispensed		

Action	Process	
	5) Generate Reciept	6) Interact with Hopper
Invalid_Input		
Open/Close_Command		
DB_Update_SUCCESS		
Invalid_Customer_Data		
Customer_Read_ SUCCESS		
Ingredients_Update_ SUCCESS		
Calculate_Mixture_ SUCCESS		
Calculate_Cost_ SUCCESS	True	
Ingredients_Read_ SUCCESS	True	
Date_Time_Read_SUCCESS	True	
Customer_Read_Success	True	
Hopper_Available		True
Hopper_Ready		True
Ingredients_Read_ SUCCESS		True
Ingredients_Added		True
Mixer_Door_Status		
Invalid_Command		
Concrete_Dispensed		

	7) Activate/Deactivate Concrete Mixer
Invalid_Input	
Open/Close_Command	
DB_Update_SUCCESS	
Invalid_Customer_Data	
Customer_Read_ SUCCESS	
Ingredients_Update_ SUCCESS	
Calculate_Mixture_ SUCCESS	
Calculate_Cost_ SUCCESS	
Ingredients_Read_ SUCCESS	
Date_Time_Read_SUCCESS	
Customer_Read_Success	
Hopper_Available	
Hopper_Ready	
Ingredients_Read_ SUCCESS	
Ingredients_Added	
Mixer_Door_Status	True
Invalid_Command	True
Concrete_Dispensed	True

Data Dictionary

1) Entities

Name:	Control Panel
Description:	The control panel is used by the operator to input commands and customer data and requirements
Name:	Operator’s Console
Description:	This is a visual display for the operator to provide alerts if the system needs any input or an error has occoured.
Name:	Invoice Printer
Description:	This is just a printer that prints out the invoice in the desired format
Name:	Mixer
Description:	A mixer is where the mixing takes place
Name:	Hopper
Description:	This is where the ingredients are stored
Name:	Audio Alarm
Description:	Used to alert an operator that something has happened and the system needs attention

2) Processes

Name:	Interact with Operator
Description:	This process simply gets the input from the operator. The input can be customer data or commands
Name:	Process Customer
Description:	The customer data is validated to ensure it is in the required format. It is then processed and added to the database.
Name:	Display Messages and Status
Description:	This process is directly interfaced with the Operator's Console as well as the alarm system. It is used to notify the Operator if an error has occurred.
Name:	Calculate Concrete Mixing Requirements
Description:	This process retrieves data from both data stores and then completes a calculation to determine the cost and the ingredients needed.
Name:	Generate Invoice
Description:	This process is used to calculate the final costs as well as generate and print the invoice for the customer.
Name:	Interact with Hopper
Description:	This process directly interacts with the Hopper. It checks the status among other things.
Name:	Activate / Deactivate Concrete Mixer
Description:	This is a process deals with the Operator. If the input is a command this process deals with the commands and executes them after validating them.

3) Data Store

Name:	Customer Data store
Description:	Customer Database is a data store that holds all customer related details such as customer name and id, and the order details.
Name:	Ingredients Data store
Description:	Stores the details that pertain to ingredients. This includes quantities for each substance.