

SCHOOL OF COMPUTING, ENGINEERING AND INFORMATION SCIENCES

**CM0721 – Implementation of Object Oriented Designs**

*Quick House Construction (QHC) – Order Management*

Module Tutor: Mark Hurrell

Student: Eshani Eshani (W14038320),

Jonathan Pickering (W14036479),

Gary Storey (J163159)  
Programme: MSc Computer Science  
Submission Date: 4th May 2015

# Introduction

Quick House Construction (QHC) Ltd requires an Order Management and Scheduling system. This document describes the analysis, design and development of a solution based on the requirement.

# Selecting an Appropriate Methodology

An AGILE methodology using object oriented techniques was chosen based on the need to produce a maintainable, quality product with industry standard documentation in C# within a short timeframe.

# Requirements Definition

## Use Case Diagrams

## Use Case Descriptors

## Implementation Order

A MoSCoW technique was used to prioritise the functionality identified by the use cases into an implementation order. The strict time constraint imposed by the project resulted in the time boxing identified by table 1.

|  |  |  |  |
| --- | --- | --- | --- |
| **Item** | **Use Case** | **MoSCoW** | **Release** |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |

*Table 1 – Implementation Order*

# Analysis and Design

## Class Diagrams

## Sequence Diagrams

## Interface Design

## Wire Frames

## Database Design

## Design Patterns (Domain Layer)

## Software Engineering Principals

## Presentation Patterns

# Implementation

## Test Plan

## Source Code

# Conclusion

# Appendix

## Individual Design – Eshani Eshani

## Individual Design – Jonathan Pickering

## Individual Design – Gary Storey

### Use Case Descriptors

**Use Case ID: 1**

**Use Case Name**: Create New Order

**Goal in context**: A new order for a customer is created on the system but not yet submitted.

**Pre-conditions**: N/A

**Successful End Condition**: An initial order is saved within the order management system as un-submitted with the details of the property type and cost.

**Failed End Condition**: Order is deleted and therefore not saved within in the system.

**Primary Actors:** User

**Secondary Actors:** None

**Trigger:** User clicking on create new order button in system.

**Main Flow:**

1. User searches for customer account using the customers ID.

2. User selects the type of house these wish to purchase from the available selection.

3. User selects the options they require for the house.

4. System calculates cost for this order.

5. System calculates estimated date for fabrication for this order.

6. System saves the creation date of the order.

7. include: Submit Order

**Extensions**:

1.1 – Customer does not have an existing customer account.

1.2 – include: Register New Customer

7.1 – include: Delete Order

7.2 – include: Save Order

**Use Case ID: 2**

**Use Case Name**: Register New Customer

**Goal in context**: A new customer is registered on the system.

**Pre-conditions**: Customer does not currently have an account on the system.

**Successful End Condition**: The customer is registered on the system which can be used to create an order.

**Failed End Condition**: N/A

**Primary Actors:** User

**Secondary Actors:** None

**Trigger:** Customer does not have an account when attempting to place an new order and the add new customer account button is pressed.

**Main Flow:**

1. User enters the type of customer account either individual or company.
2. User enters an email address for the account which will be the username.
3. User enters a password.
4. User enters a contact name.
5. User enters a contact phone number.
6. User enters a contract address.
7. User clicks the add user button.
8. System validates the input.
9. System adds customer account and return to the new order form.

**Extensions**:

7.1 Validation fails and the user must re-enter the information.

**Use Case ID: 3**

**Use Case Name**: Edit Un-submitted Order

**Goal in context**: A customer accesses a previous un-submitted order and can alter the available options for this order.

**Pre-conditions**: Customer must have an un-submitted order in the system.

**Successful End Condition**: Customer has accessed the order and made alterations leading to the order being either deleted, saved or submitted.

**Failed End Condition**: N/A

**Primary Actors:** User

**Secondary Actors:** None

**Trigger:** User has clicked upon the edit un-submitted order button in the system.

**Main Flow:**

1. User enter the order ID they wish to edit.
2. System returns details of the order to the screen.
3. User changes the details of the property type and/or options.
4. System calculates cost for this order.
5. System calculates estimated date for fabrication for this order.
6. include: Submit Order.

**Extensions**:

* 1. include: Delete Order.
  2. include: Save Order.

**Use Case ID: 4**

**Use Case Name**: Delete Order

**Goal in context**: An order has its status set to be deleted within the system and saved to the data store.

**Pre-conditions**: Customer may have an un-submitted order in the system or be in the process of creating a new order.

**Successful End Condition**: Order is set to deleted status.

**Failed End Condition**: N/A

**Primary Actors:** User

**Secondary Actors:** None

**Trigger:** User has clicked upon the delete button on the form.

**Main Flow:**

1. System set the status of the order to deleted.
2. System presents a message to user informing them of the deletion.
3. User is taken back to the main system form.

**Use Case ID: 5**

**Use Case Name**: Save Order

**Goal in context**: An order has its status set to un-submitted within the system and saved within the data store.

**Pre-conditions**: Client must have an email account with some provider.

**Successful End Condition**: Client successfully signs up for use of the system and is in possession of a valid username and password to sign into the system.

**Failed End Condition**: N/A

**Primary Actors:** User

**Secondary Actors:** None

**Trigger:** User has clicked upon the save button on the form.

**Main Flow:**

1. System set the status of the order to un-submitted.
2. System stores the information from the order form.
3. System presents a message to user informing them of the save.
4. User is taken back to the main system form.

**Use Case ID: 6**

**Use Case Name**: Submit Order

**Goal in context**: An order is submitted to the system and has the status changed to the planning phase and saved to the data store.

**Pre-conditions**: A customer must have a completed order form.

**Successful End Condition**: Order status is now set to planning and a planning pack, £1000 invoice and contract are generated and sent to the customer.

**Failed End Condition**: N/A

**Primary Actors:** User

**Secondary Actors:** None

**Trigger:** User has clicked upon the submit button on the form.

**Main Flow:**

1. System set the status of the order to planning.
2. System stores the information from the order form.
3. System creates an invoice for the customer for the planning pack.
4. System creates a contract for the order.
5. System presents a message to user informing them of the submission.
6. User is taken back to the main system form.

**Use Case ID: 7**

**Use Case Name**: Cull Orders

**Goal in context**: Order that have not been progressed for a set amount of time will be set to the status of deleted.

**Pre-conditions**: Order must have been in within a status in the systems for the specified amount of time.

**Successful End Condition**: Order is set to deleted status.

**Failed End Condition**: N/A

**Primary Actors:** Timer

**Secondary Actors:** None

**Trigger:** A timed process is run within the system.

**Main Flow:**

1. System check order set to un-submitted with a creation date of over
2. System set each of these orders to deleted status.
3. System check order with planning rejected status that is over 1 year old.
4. System set each of these orders to deleted status.
5. System message generated to inform user of successful cull of orders.

**Use Case ID:** 8

**Use Case Name**: Update Planning Status Order

**Goal in context**: An order which is in the planning status require to be updated which can be either planning permission granted, planning permission rejected or the payment of the planning pack invoice, the updates are then saved and any status update for the order is completed.

**Pre-conditions**: Customer must have order within the planning status in the system.

**Successful End Condition**: The order is updated appropriately within the system.

**Failed End Condition**: N/A

**Primary Actors:** User

**Secondary Actors:** None

**Trigger:** User has clicked upon the update order button in the system.

**Main Flow:**

1. User enter the order ID they wish to update.
2. System returns details of the order to the screen.
3. User checks planning permission granted if true.
4. User checks planning permission rejected if true.
5. User checks planning pack invoice paid if true.
6. User clicks update order button.
7. extend: Pay Invoice.
8. include: Update Order Status.
9. User is taken back to the main system form.

**Use Case ID:** 9

**Use Case Name**: Update Contract Status Order

**Goal in context**: An order which is in the contract status require to be updated which can be either contract signed, foundation date or the payment of the invoice, the updates are then saved and any status update for the order is completed.

**Pre-conditions**: Customer must have order within the contract status in the system.

**Successful End Condition**: The order is updated appropriately within the system.

**Failed End Condition**: N/A

**Primary Actors:** User

**Secondary Actors:** None

**Trigger:** User has clicked upon the update order button in the system.

**Main Flow:**

1. User enter the order ID they wish to update.
2. System returns details of the order to the screen.
3. User checks contract returned if true.
4. User enters foundation ready date if known.
5. User checks planning full invoice paid if true.
6. User clicks update order button.
7. extend: Pay Invoice.
8. include: Update Order Status.
9. System presents a message to user informing them of the update.
10. User is taken back to the main system form.

**Use Case ID:** 10

**Use Case Name**: Pay Invoice

**Goal in context**: Set the invoice to paid and calculates the total balance required for an order.

**Pre-conditions**: User must have checked the paid invoice checkbox.

**Successful End Condition**: The order is no longer stored in the system.

**Failed End Condition**: N/A

**Primary Actors:** User

**Secondary Actors:** None

**Trigger:** User has checked the pay invoice box within the relevant section of the update order form.

**Main Flow:**

1. System sets the invoice status to paid.
2. System stores the date the invoice was paid.
3. System calculates the remaining balance for the order.

**Use Case ID:** 11

**Use Case Name**: Update Order Status

**Goal in context**: When an order is updated the update order status checks to see what was updated and checks to see if the order needs its status changed and any action taken based upon this change.

**Pre-conditions**: Within the Update Planning Status Order use case.

**Successful End Condition**: The order status is updated if required.

**Failed End Condition**: N/A

**Primary Actors:** User

**Secondary Actors:** None

**Trigger:** User has clicked upon the update order button in the system.

**Main Flow:**

1. System check the current status
2. System checks which actions have been updated
3. System updates the status if the relevant actions have been completed.
4. System performs actions based upon rules for each status is updated.
5. System updates the order queue schedule based on the updates.
6. Include: Create Order Queue Report if the schedule changes.

**Use Case ID:** 12

**Use Case Name**: Organise Assembly

**Goal in context**: Once the order has been fabricated the customer updates the order with a assembly date to complete the order.

**Pre-conditions**: The order must be in the fabricated state.

**Successful End Condition**: A date is set for the assembly of the building.

**Primary Actors:** User

**Secondary Actors:** None

**Trigger:** User has clicked upon the update order button in the system.

**Main Flow:**

1. User enters the order ID they wish to update.
2. System returns details of the order to the screen.
3. User enters foundation ready date if known.
4. User clicks update order button.
5. include: Update Order Status.
6. System presents a message to user informing them of the update.
7. User is taken back to the main system form.

**Use Case ID:** 13

**Use Case Name**: Inform Foundation Delay

**Goal in context**:

**Pre-conditions**: The order must be in the contract state.

**Successful End Condition**: A date is set for the assembly of the building.

**Primary Actors:** User

**Secondary Actors:** None

**Trigger:** User has clicked upon the update order button in the system.

**Main Flow:**

1. User enter the order ID they wish to update.
2. System returns details of the order to the screen.
3. User clicks foundation delayed button.
4. User clicks update order button.
5. include: Update Order Status.
6. System presents a message to user informing them of the update.

**Use Case ID:** 14

**Use Case Name**: Create Order Queue Report

**Goal in context**: A report is generated containing a queue of the houses to be built and the options and sent this to the factory system.

**Pre-conditions**: N/A

**Successful End Condition**: Report sent to the factory system.

**Primary Actors:** Administrator, Internal Trigger

**Secondary Actors:** None

**Trigger:** Administrator clicks create report button or system triggers on a change of schedule with the update order status use case.

**Main Flow:**

1. System gets a list of all of the orders within the order queue

2. System sends this report to the factory system.

**Use Case ID:** 15

**Use Case Name**: Update Schedule from Factory Report

**Goal in context**: Factory sends a file of the order being fabricated which takes place on a daily basis.

**Pre-conditions**: N/A

**Successful End Condition**: The system updates the relevant order status based upon the information received from the factory.

**Primary Actors:** Factory

**Secondary Actors:** None

**Trigger:** Factory sends daily file to the system.

**Main Flow:**

1. System checks for the factory

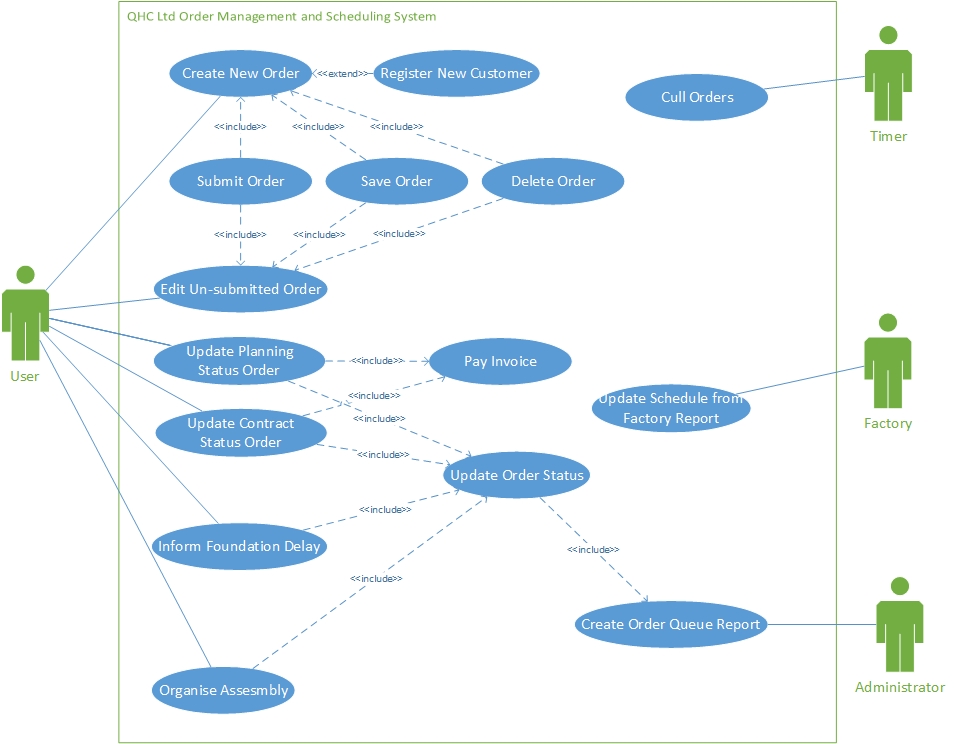
2. System compares the current factory file with the current order queue.

3. include: Update Order Status

**Extensions**:

3.1 – If still fabricating the same order then no need to update order status.

### Use Case Diagram



### Class Diagram

### Sequence Diagram