# **BAPERS: Bloomsbury’s Automated Process Execution Recording System**

04 March 2018

## **Summary**

This is a Requirements Specification and System Design, a proposal to the owner of the Bloomsbury’s Image Processing Laboratory (BIPL), Mr Glynne Lancaster, from the Software Development Team, CODEX.

## **Authors**

Henal Desai, Kishan Patel, Sai Bharath, Sameer Ali, Warsame Ali and Khesrou Zemaray

## **Distribution**

The owner of BIPL, and the Development Team at CODEX.



**Table of Contents**

[1 Preface……………………………………………………………………………………… 3](#_2et92p0)

[1.1 Purpose and scope of the document…………………………………………………….. 3](#_tyjcwt)

[1.2 History of the document………………………………………………………………... 3](#_3dy6vkm)

[2 Introduction…………………………………………………………………………………. 3](#_1t3h5sf)

2.1 Current BIPL system…………………………………………………………………… 4

2.2 How BAPERS will be different to the existing system………………………………… 4

3 User Requirements………………..........................................................................................5

3.1 UML Use case models………………………………………………………………..... 5

3.1.1 Use case diagram……………………………………………………………………... 5

3.1.2 Use case specification………………………………………………………………… 6

3.1.3 Use case priority table………………………………………………………………. 13

4 Design class diagram………………………………………………………………………. 19

4.1 Overview……………………………………………………………………………… 19

4.1.1 Packages……………………………………………………………………………..19

5 ER Diagram………………………………………………………………………………... 25

5.1 DDL Statements……………………............…………………………………….……26

5.2 Select, Insert, Update, Delete Statements………………………………………….…..34

5.3 Non-Trivial Reports………………………………………………………..…………..35

6 GUI Designs……………………………………………………………………………......38

6.1 Visual representation and navigation between screens…………………………..……38

6.2 GUI Sitemap ……………………………………………………………………….….52

6.3 GUI Mapping…………………………………………………………………..………53

## **1 Preface**

## Purpose and scope of the document

This document is the Requirements Specification and System Design of the software being developed by the Software Development Team CODEX, for their, Mr Glynne Lancaster, owner of Bloomsbury’s Image Processing Laboratory (BIPL). The customer’s requirements have been given for the development of a software for the use in their laboratory.

## History of the document

|  |  |  |  |
| --- | --- | --- | --- |
| **Version** | **Changes** | **Responsible for changes** | **Date** |
| 1.0 | Added use class diagram to the documentation and the use case specifications | Henal, Kishan, Sameer, Sai, Warsame and Khesrou | 20/02/2018 |
| 1.1 | Made changes to the use case priority table | Henal, Kishan, Sameer, Sai, Warsame and Khesrou | 28/02/2018 |
| 1.2 | Made changes to the document by going over the scope of the current system. | Henal, Kishan, Sameer, Sai, Warsame and Khesrou | 02/03/2018 |

## **2 Introduction**

BIPL is a company which handles work of professional photographers, who offer a range of jobs to customers, including many tasks which are undertaken by the staff at the laboratory and respond to any job’s special instructions that are given by their customers.

The laboratory carries jobs on behalf of customers, with their unique ID and special instructions for specific job. Urgent jobs must be dealt within 6 hours and normal jobs must be completed within 24 hours. However, customer can also ask for completion to be done within 3 hours by paying at a higher rate.

2.1 Current BIPL system

The technician can accomplish 30 standard tasks, which take place at specific processing stations/locations within the laboratory. Once it is half or completed, it is placed in a shelf where it is either taken for further processing or to be sent to the customer. Having a meeting with Mr Lancaster regarding the needs of the new system requirements, there have been multiple issues with their current system being outdated and unreliable. BIPL’s current system is paper-based, where staff have to individually record all customers payments, the process of the jobs and the communication between the employees. This is an inefficient way of transporting document which can become problematic for the company.

Paperwork would take up a large amount of space and would make storage become unorganised. This means that their documents can be misplaced or left unaccountable, leading to further damage/loss for both customers and company.

BIPL would like to put their employees on a reception desk where they can enter the jobs on a computer system. This is something that will be considered and implemented when developing the BAPERS software system.

2.1 How BAPERS will be different from the current system

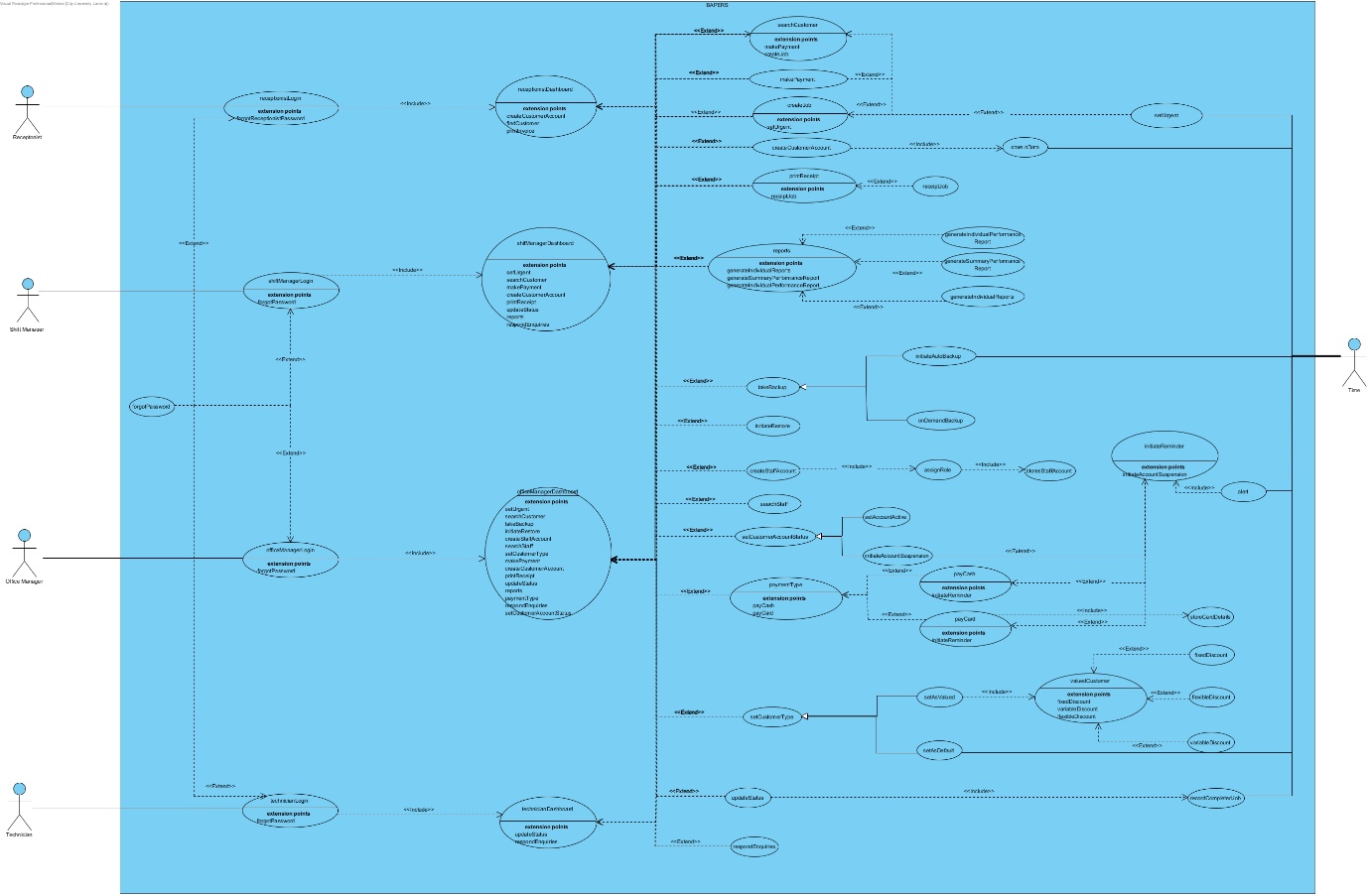
The laboratory will have many jobs going through with numerous tasks, therefore it is crucial to make sure that there are no confusions between them. This includes ensuring that is no mistreatment or loss of customer’s materials. The new system being developed, BAPERS, will have flexibility to schedule jobs according to how urgent they are, given the deadline, giving priority to jobs that need to be completed.

Implementing he electronic software system will allow entering information and editing customer’s accounts become easier to access and use. The concurrent use of this system will allow multiple members of staff, The Office Manager, Shift Manager, Receptionist and Technician, to gain easy access to customers’ accounts to carry a range of tasks with their different accesses to the software system. It will make it easier for them to view the types of jobs that are on-going or completed. The payment method for customers will become enhanced, as the payment records will be stored in the system giving both options of cash and credit/debit.

**3. User Requirements**

3.1 UML use-case models

3.1.1 Use case diagram

**

3.1.2 Use case specification

|  |  |
| --- | --- |
| ID: 1 | **Use case:** createCustomerAccount |
| Brief description:  The actor is able to create a customer account if necessary in order for their jobs to be processed by the system and for information to be stored. | |
| Primary actors: Receptionist, Shift Manager, Office Manager | |
| Secondary actors: None | |
| Preconditions:   1. A new customer requests for a job to be completed. | |
| Flow of events:   1. Actor inputs the customer’s details.   Extension point: createCustomerAccount   1. The created account is then stored by the system. | |
| Postconditions:   1. The system has recorded a new customer account. | |

|  |  |
| --- | --- |
| ID: 2 | **Use case:** setUrgent |
| Brief description:  Flexible scheduling is required to allow priority to be given to urgent jobs. The system must allow this to happen for jobs to be completed in time in order of importance. | |
| Primary actors: Receptionist, Shift Manager, Office Manager | |
| Secondary actors: Time | |
| Preconditions:   1. The job must be completed within 6 hours or less. | |
| Flow of events:   1. Actor accepts new job through the system dashboard.   Extension point: createJob   1. The system allows job priority to be set, which the actor sets to urgent.   Extension point: setUrgent | |
| Postconditions:   1. The system has recorded a new job with the priority set to urgent. | |

|  |  |
| --- | --- |
| ID: 3 | **Use case:** createJob |
| Brief description:  The system should provide functionality for creating a job for a customer. | |
| Primary actors: Technician, Shift Manager, Office Manager | |
| Secondary actors: None | |
| Preconditions:   1. An existing customer has requested for a job to be completed. | |
| Flow of events:   1. The actor selects the appropriate job from the list of jobs. 2. The actor checks whether a discount is applicable for the particular customer. 3. The actor confirms the job with the price that has been set with the customer. | |
| Postconditions:   1. The job is then processed and assigned to a technician. | |
| Alternative flow:  No Available Job | |

|  |
| --- |
| Alternative flow: No Available Job |
| ID: 3.1 |
| Brief description:  There are no jobs available to be processed. |
| Primary actors: Receptionist, Shift Manager, Office Manager |
| Secondary actors: None |
| Preconditions:   1. The actor attempts to select the appropriate job from the list. |
| Alternative flow:   1. The job cannot be set as it is not available. 2. The system displays a pop-up window stating that the job is not available. |
| Postconditions: The actor returns to the dashboard. |

|  |  |
| --- | --- |
| ID: 4 | **Use case:** recordCompletedJobs |
| Brief description:  The technician updates the job status as completed, which is then recorded in the system. | |
| Primary actors:  Technician, Shift Manager, Office Manager | |
| Secondary actors: None | |
| Preconditions:   1. The technician undertakes a job to be completed. | |
| Flow of events:   1. The technician completes the job and then sets the job as completed. 2. The system records the job as completed. | |
| Postconditions: The record will be updated with a completed job. | |

|  |  |
| --- | --- |
| ID: 5 | **Use case:** paymentType |
| Brief description:  Payment records are stored by the system. Payment information is input once the customer’s job is completed. | |
| Primary actors: Receptionist, Shift Manager, Office Manager | |
| Secondary actors: None | |
| Preconditions:   1. The customer’s job should be completed. 2. The customer decides to pay by card or cash. | |
| Flow of events:   1. Receptionist inputs customer’s card details/cash and amount into the system.   Extension point: payCard  Extension point: payCash | |
| Postconditions:   1. The system records the payment in the customer’s account and keeps it stored for access. | |
| Alternative flow: Payment Unsuccessful | |

|  |
| --- |
| Alternative flow: Payment Unsuccessful |
| ID: 5.1 |
| Brief description:  The payment is unsuccessful. |
| Primary actors: Receptionist |
| Secondary actors: None |
| Preconditions:   1. The customer’s card details are invalid/unacceptable or insufficient balance. |
| Alternative flow:   1. The payment is unsuccessful and not stored in the system. 2. The system displays a pop-up window to the receptionist stating that the payment is unsuccessful. |
| Postconditions:   1. The system doesn’t record the payment. |

|  |  |
| --- | --- |
| ID: 6 | **Use case:** alert |
| Brief description:  Late payments are automatically detected by the system and pop-up windows are displayed as alerts to the office manager at regular intervals. | |
| Primary actors: Office Manager | |
| Secondary actors: Time | |
| Preconditions:   1. The valued customer has not paid for a completed job within deadline | |
| Flow of events:   1. The system checks the customer’s account for payment after their job has been completed. 2. The system detects that there hasn’t been a payment made. 3. When the office manager logs in, the system will notify them by displaying a warning pop-up window every 15 minutes until manually closed. | |
| Postconditions: The office manager is notified of a late payment. | |

|  |  |
| --- | --- |
| ID: 7 | **Use case:** initiateReminder |
| Brief description:  Late payment reminder letters are automatically generated by the system for valued customers by the 20th of the next month. | |
| Primary actors: Office Manager | |
| Secondary actors: Time | |
| Preconditions:   1. The customer has not paid for a completed job within the deadline. | |
| Flow of events:   1. The system generates a reminder letter using the customer’s account information. 2. When the office manager logs in, the system will notify them to print the letter with a pop-up window. | |
| Postconditions:   1. The office manager acknowledges the pop-up window warning. | |

|  |  |
| --- | --- |
| ID: 8 | **Use case:** initiateAccountSuspension |
| Brief description:  The customer’s account will be placed in suspension. | |
| Primary actors: Office Manager | |
| Secondary actors: Time | |
| Preconditions:   1. The customer has not paid for a completed job a month after the first reminder letter was sent, or the office manager explicitly decides to change the customer’s privilege. | |
| Flow of events:   1. The system will suspend the customer’s account. | |
| Postconditions:   1. Any further jobs will not be processed for that account. | |

|  |  |
| --- | --- |
| ID: 9 | **Use case:** takeBackup |
| Brief description:  The system is either automatically or manually backed up to prevent loss of data. | |
| Primary actors: Office Manager | |
| Secondary actors: Time | |
| Preconditions:   1. There must be data stored in order for a backup to occur, or the office manager explicitly decides to take a backup by using the on demand backup feature. | |
| Flow of events:   1. The system calculates the total size to backup.   2. The system initiates the backup sequence. | |
| Postconditions: The system is backed up. | |
| Alternative flow: System Termination, Insufficient Space | |

|  |
| --- |
| Alternative flow: System Termination |
| ID: 9.1 |
| Brief description:  The system terminates while undertaking the process of backing up. |
| Primary actors: None |
| Secondary actors: None |
| Preconditions:   1. The system attempts to backup however, the system terminates. The system is then automatically restarted. |
| Alternative flow:   1. The system deletes the attempted backup file. 2. The system calculates the total size to backup. 3. The system initiates the backup sequence. |
| Postconditions:   1. The system saves the backup as a filename followed by that particular time. |

|  |
| --- |
| Alternative flow: Insufficient Space |
| ID: 9.2 |
| Brief description: There is not a sufficient amount of space available for a backup to be made. |
| Primary actors: Office Manager |
| Secondary actors: None |
| Preconditions:   1. The size of the backup file is more than the free space available. |
| Alternative flow:   1. The backup sequence is terminated. 2. The system alerts the office manager regarding insufficient space through a pop-up window. |
| Postconditions:   1. The backup halts while alerting the office manager. |

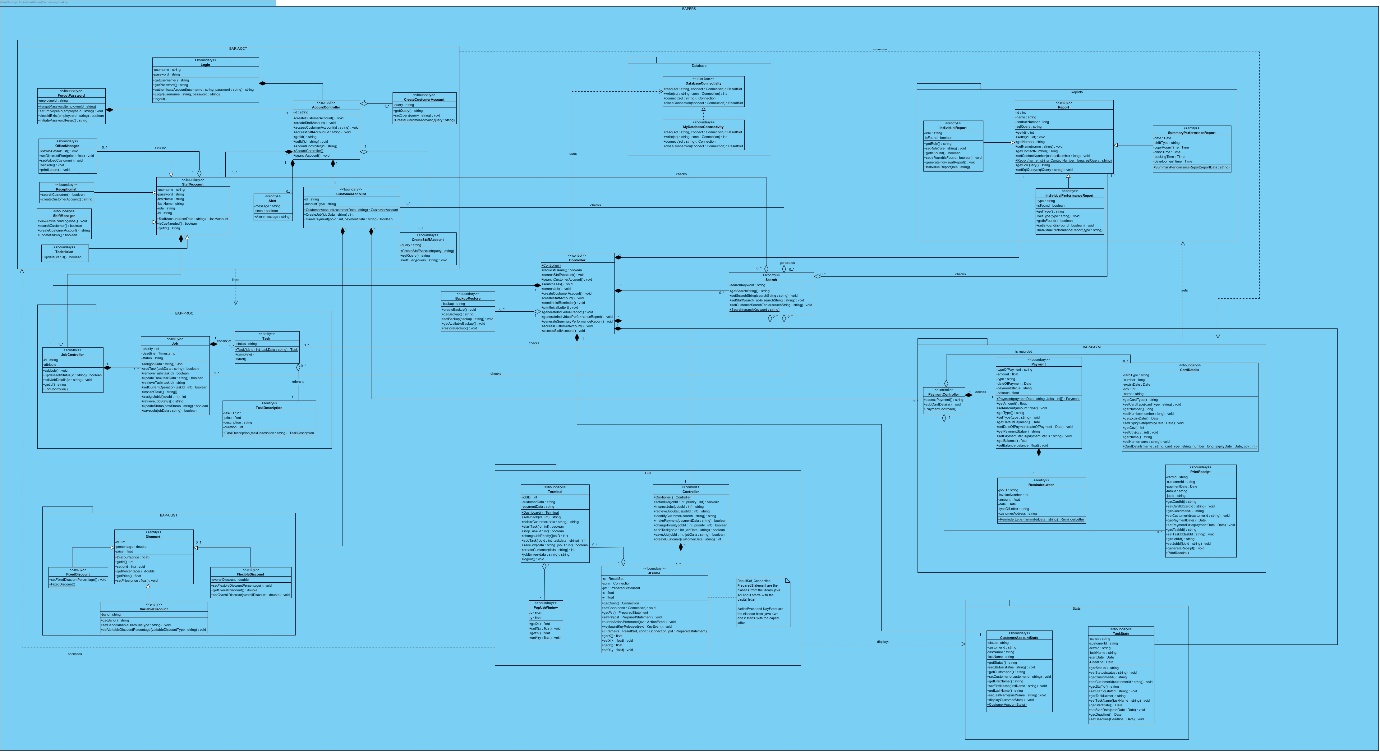
|  |  |
| --- | --- |
| ID: 10 | **Use case:** setAsDefault |
| Brief description:  One month after suspension, the customer’s account is set to default if they have not yet paid the balance amount. The office manager can also set the customer’s account to default. | |
| Primary actors: Office Manager | |
| Secondary actors: Time | |
| Preconditions:   1. The customer’s account has been suspended for a month as a result or the office manager has explicitly decided to set the account status as default. | |
| Flow of events:   1. The customer account is set to default either by the system or the office manager.   Extension Point: setAsDefault | |
| Postconditions:   1. The customer’s account loses its valued privileges. | |

3.1.3 Use case priority table

|  |  |  |  |
| --- | --- | --- | --- |
| **Use Case ID** | **Use Case Name** | **Priority** | **Description** |
| 1 | receptionistLogin | High | Receptionist is required to login to use the system therefore if the system fails to login, the staff would not be able to use the system and complete any tasks, which would be costly and would be time consuming. |
| 2 | receptionistDashboard | High | Dashboard is required in order to access features in the system. Risk associated with this is not being able to complete task for the receptionist. |
| 3 | searchCustomer | Medium | The system gives access for actor to search for a customer to view jobs. Cost is associated with this task as if it would fail then there would be some serious storage implications and inconvenience for the company as a whole. |
| 4 | makePayment | High | System records the payments of the customers and it impacts the company as if the system fails the payments would fail to go through and this could incur huge losses. |
| 5 | setUrgent | Medium | As jobs are being filtered into the system, jobs with a higher urgency to be completed can be set to urgent. Time is an important factor in order for customer satisfaction, and if this feature fails this could result in losses. |
| 6 | createCustomerAccount | High | The system allows the receptionist to create new customer account, consequently this is of high importance due to the high risk associated with this failing and causing jobs to not be processed which may lead to losses. |
| 7 | storeInData | Medium | After a new account is created the details are stored in data in the system, if this should fail this would lead to it being a really costly as well a as lengthy issue to be fixed. This would also lead to actor not able to view the customer account and the jobs associated with it. |
| 8 | printReceipt | Low | This feature lets the actor print out receipt for the jobs, and so this is important as this is needed for the customer to make a payment. The risk involved is mainly an inconvenience for the customer. |
| 9 | receiptJob | Medium | The feature is available for future reference for the jobs undertaken by the company. This is a potential issue which would cause problems for the company as they would not have evidence of jobs that have been processed. |
| 10 | shiftManagerLogin | High | Shift manager is required to login to use the system therefore if the system fails to login, this would be time consuming. |
| 11 | shifManagerDashboard | High | Dashboard is required in order to access features in the system. Risk associated with are not being able to complete task for the actor. |
| 12 | reports | Medium | The system generates reports for different circumstances. |
| 13 | generateIndividualPerformanceReport | Medium | System generates a report consisting of the work undertaken by the individual technician, the risk to consider for the following feature is to keep track of the time and resources being used. |
| 14 | generateSummaryPerformanceReport | Medium | Report generated for the sum of all the time undertaken by all the technicians. Projected risk for this feature is failure to keep a record in the system. |
| 15 | generateIndividualReports | Medium | The system generates the reports brought in by a particular customer over a set amount of time.Projected risk for this feature is failure to keep a record in the system. |
| 16 | officeManagerLogin | High | Actor required to login to use the system therefore if the system fails to login, this would be time consuming. |
| 17 | officeManagerDashboard | High | Dashboard is required in order to access features in the system. Risk associated with are not being able to complete task for the actor. |
| 18 | takeBackup | High | Takes a backup of the entire system for safe keeping, risk involved with not having this feature would be a potential loss of large amount of data, which would be costly. |
| 19 | initiateAutoBackup | Medium | The system allows for a backup to be make over a configurable period of time risk involved with not having this feature would be a potential loss of large amount of data, which would be costly. |
| 20 | onDemandBackup | Low | The system allows the office manager to initiate a backup on demand. This is less of a risk compared to the autobackup as there would still be a certain amount of data backed up onto the system prior to initialization of the on-demand backup. Potential loss of data. |
| 21 | initiateRestore | Low | Restores the data from a set period of time to prevent an issue from occurring. Risk to the backup as potential loss of data from going back in time of the data. |
| 22 | createStaffAccount | High | The system allows for the creation of new staff account, consequently this is of high importance due to the high risk associated with this failing would have the actors not able to perform task at hand and would interfere with the productivity of the jobs needed to be completed. |
| 23 | assignRole | High | The system allows the office manager to assign roles to member of staff depending their role within the company which decide their access levels in their system. The risk involved in this is that if it fails, certain member of the staff may not be able to gain access to what they require for their job. |
| 24 | storesStaffAccount | Medium | After a new account is created the details are stored in data in the system, if this should fail this would lead to it being a really costly and a lengthy issue to be fixed. This would also lead to the actor not being able to view the jobs assigned for completion. |
| 25 | searchStaff | Medium | Being able to search for staff will allow the office manager to see which job is assigned to which particular staff and if this fails it will become difficult to keep track of the progress of the jobs. |
| 26 | setCustomerAccountStatus | High | The office manager will be able to decide which accounts are valued or default. Without this, this may cause financial loss if discounts applied to customers are not tracked. |
| 27 | paymentType | Medium | This use case describes how an actor selects a payment method. The payment method indicates how the company will be paid for the job completed. The customer may choose to either: have it paid by cash or by card. Risk associated if this should fail would cause delay in payments, which would result in losses. |
| 28 | payCash | Medium | The system will record payments made by cash through the amount given. If this is not implemented then this could result in financial issues occuring as a result of a lack of evidence. |
| 29 | payCard | Medium | The system will record payments made by card through the details given. If this is not implemented then this could result in financial issues occuring as a result of a lack of evidence. |
| 30 | initiateReminder | High | The system sends out reminder for any payments that hasn’t been paid to the office manager which would be a first reminder and if still not covered a second reminder. If this should fail it would affect how the system would behave with the other functionality of suspending account when the reminders are sent. This could negatively impact the finances for the parties involved in the task of handling the jobs. |
| 31 | setAccountDetails | High | Storing staff members account details is necessary in order to assign jobs to staff members. Without having this access, you cannot create new staff accounts, therefore any jobs that are assigned to them will not be recorded in the system. |
| 32 | initiateAccountSuspension | High | The system automatically suspends any account that hasn’t cleared its payments after the second reminder is sent. If this should malfunction customers would still be able to be permitted for new jobs to be completed and go about as everything is. This would incur a loss in its finances and productivity as a whole. |
| 33 | storeCardDetails | Low | The card details need to be stored in the database for easy access to make quick payments when necessary for future jobs. However, if this is unsuccessful, the main issue would be an inconvenience rather than any harm to the overall performance of the system. This could be easily rectified if this functionality should fail. |
| 34 | setCustomerType | Medium | The system allows for a customer’s account to be updated depending on their transactions with the company. This could have serious implications for the system as it could fail to notify the office manager of the status of customer’s regarding payments. |
| 35 | setAsValued | Medium | The system gives the office manager functionality to upgrade a customer to a valued customer if and when they so require it. Issues with this feature of the system may result in financial losses regarding customer payments in future investments. |
| 36 | setAsDefault(change name as this allows the office manager to change the state of account back to old one) | High | The office manager is allowed by the system to change the status of a customer from the in default state to whatever its prior state. This would have severe implications if the actor is unable to change the customer account state, as it would stop the customer from requesting new jobs, which would in turn cause losses for the company. |
| 37 | valuedCustomer | Medium | Valued customers are allowed by the system to be given specific discounts depending on their transactions. This provides a strong incentive and would result in potential financial and reputational issues if implemented incorrectly. |
| 38 | fixedDiscount | Medium | System provides a fixed percentage of discount for valued customers for each job. This provides a strong incentive and would result in potential financial and reputational issues if implemented incorrectly. |
| 39 | flexibleDiscount | Medium | The system sets the percentage of the discount depends on the values of the jobs by the same customer accumulated within a calendar month. This provides a strong incentive and would result in potential financial and reputational issues if implemented incorrectly. |
| 40 | variableDiscount | Medium | The system sets the percentage of the discount for each task and may vary between the tasks. This provides a strong incentive and would result in potential financial and reputational issues if implemented incorrectly. |
| 41 | technicianLogin | High | Actor required to login to use the system therefore if the system fails to login, this would be time consuming. |
| 42 | technicianDashboard | High | Dashboard is required in order to access features in the system. Risk associated with are not being able to complete task for the actor. |
| 43 | updateStatus | Medium | The technician is allowed to update the system that a job has been completed once processed. If this is not able to be triggered by the system then the customer will not be able to make payments for their jobs, resulting in losses. |
| 44 | recordCompletedJob | High | The system records which of the jobs have been completed and saves it to the database. The risk that are associated when this is not possible would be the loss of payments as jobs would show up as not completed and would be able to pay for. |
| 45 | respondEnquiries | Medium | The technician is able to respond to enquiries from any computer terminal about status of a job, or of all jobs that are “in progress”, or of all jobs (including the completed ones). Should this fail there would be a backlog of jobs that are not being progressed which would result in delaying the time for the finished product. |
| 46 | forgotPassword | Medium | The system gives the functionality to the actors to reset their password if necessary. If the feature is not available the actor would not have access to the system, causing halt in the progress of jobs which would be costly. |

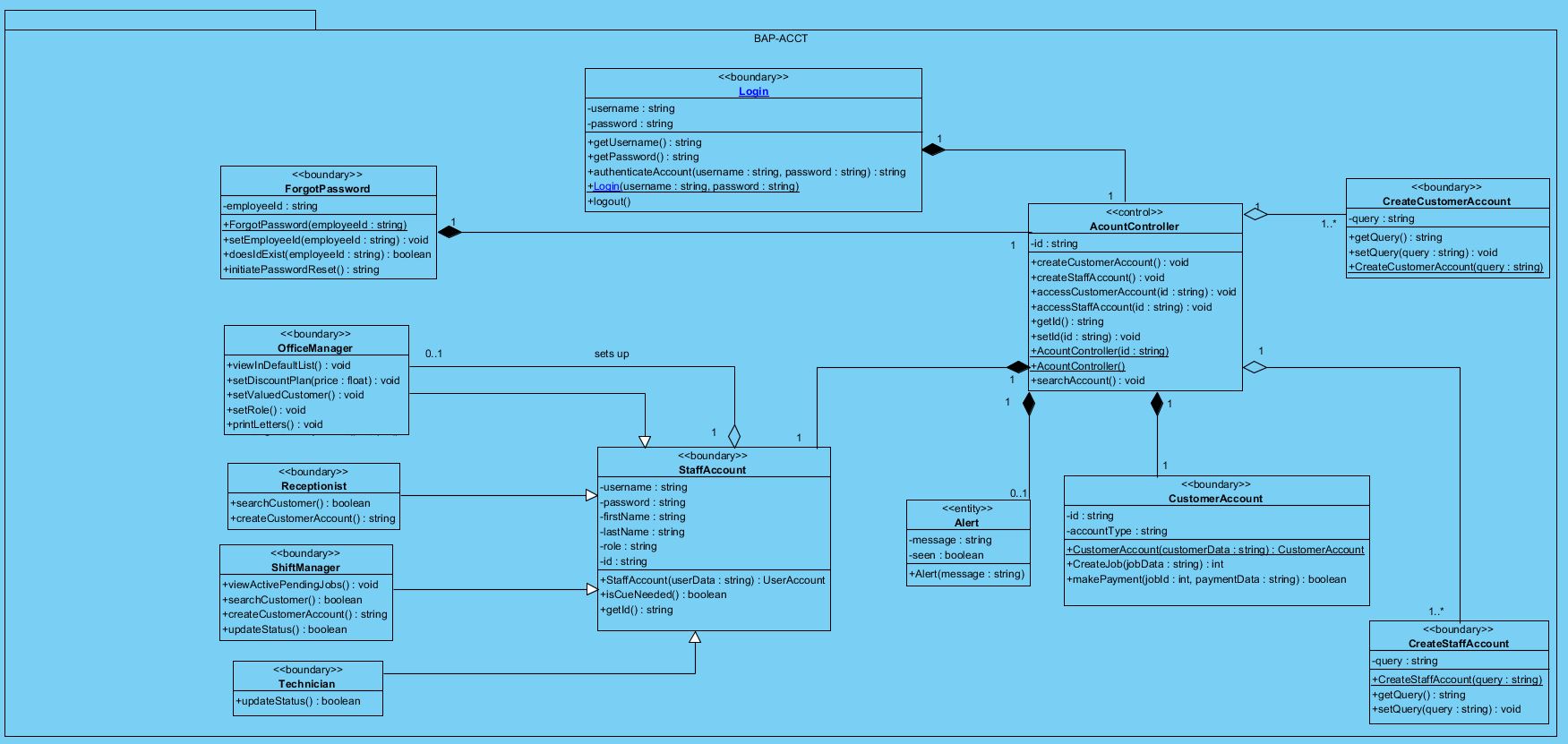
**4 Design class diagrams**

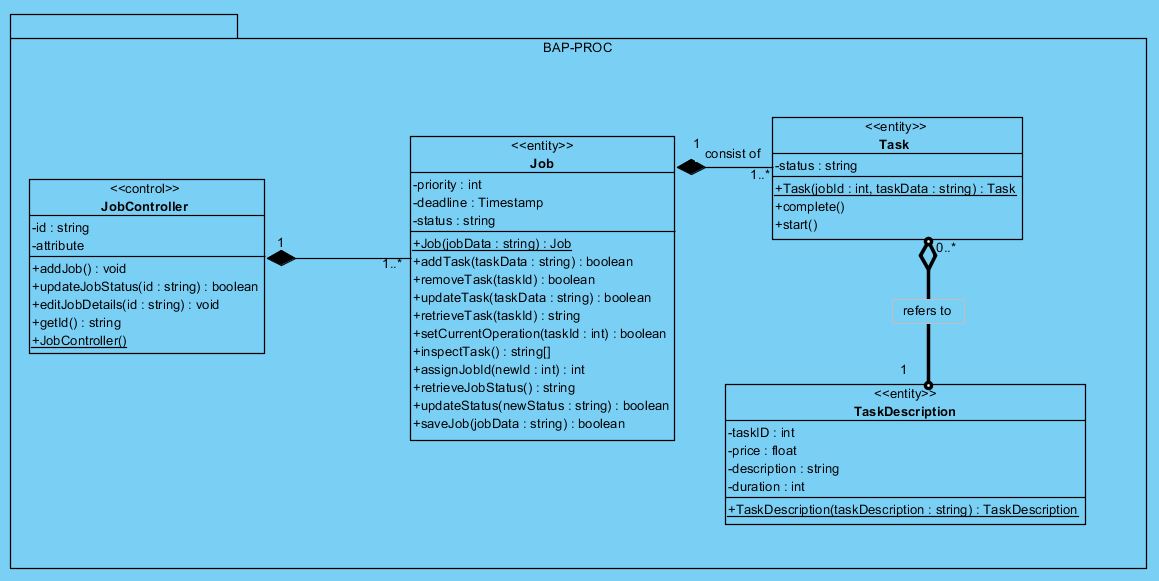
4.1. Overview



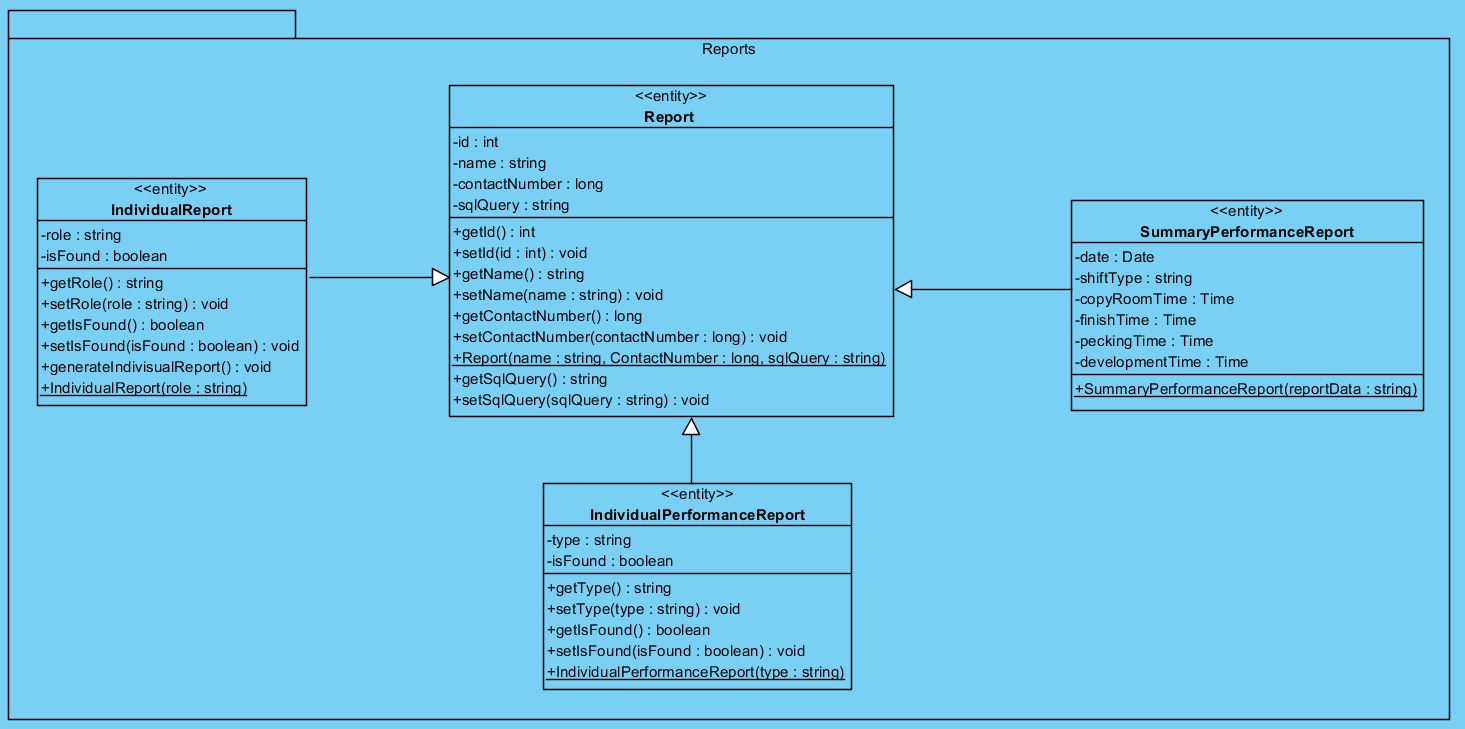
4.1.1 Packages

**BAP-ACCT**

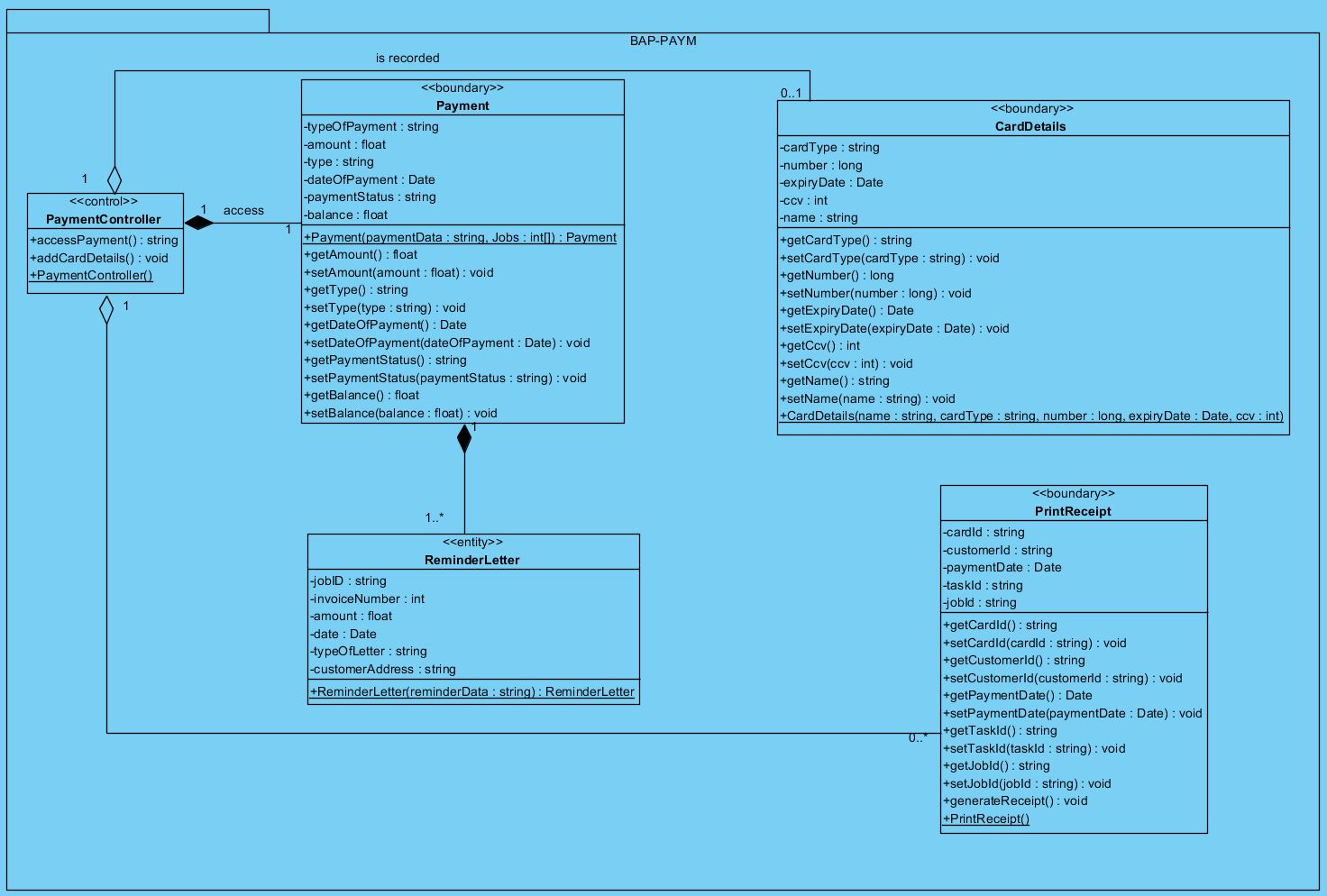


**BAP- PROC**

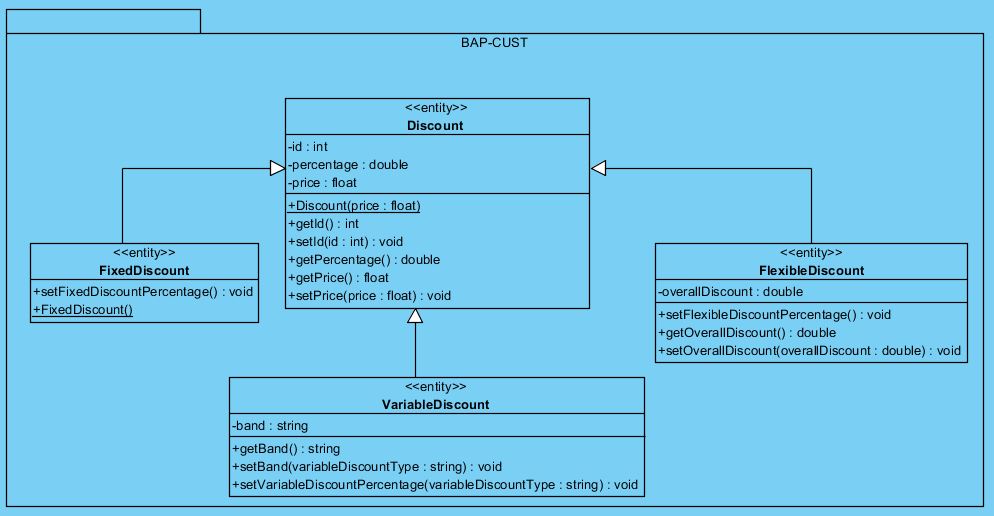
**BAP-REPT**



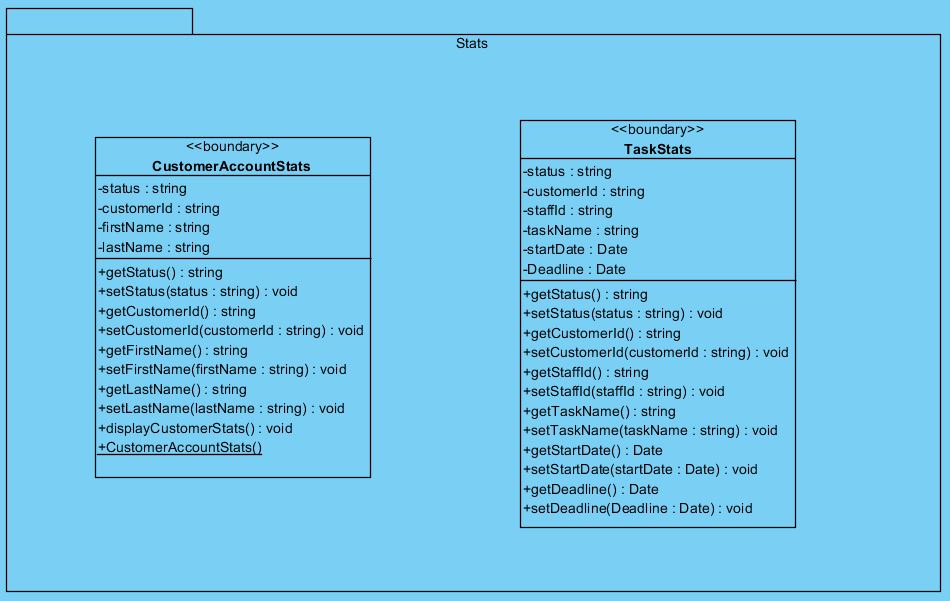
**BAP-PAYM**



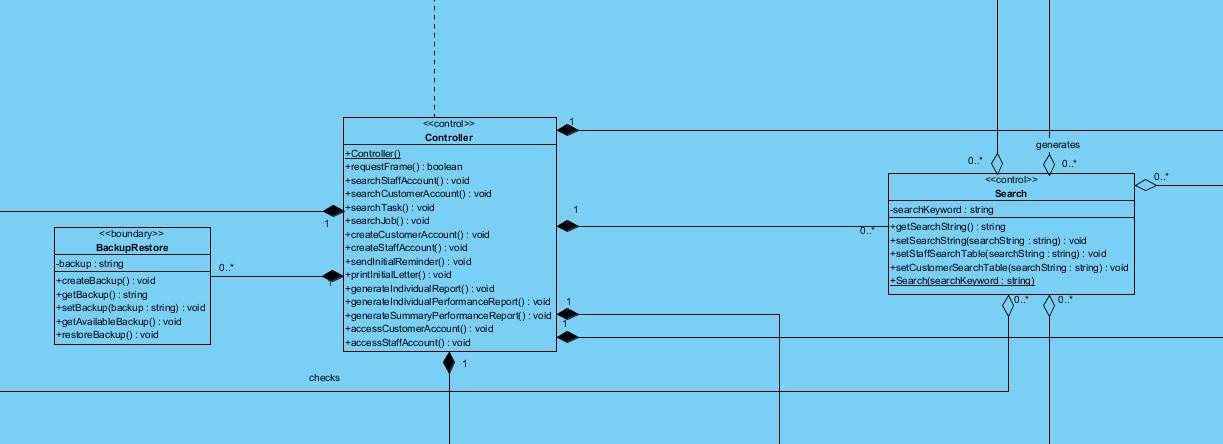
**BAP-CUST**



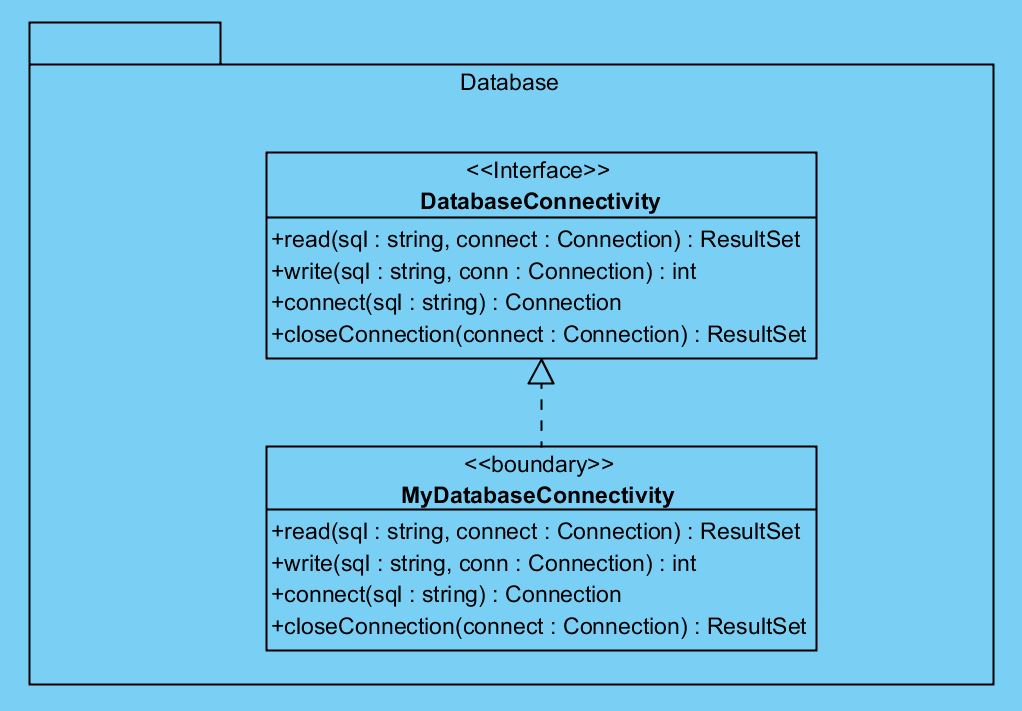
**Stats**



**Controller and Search**

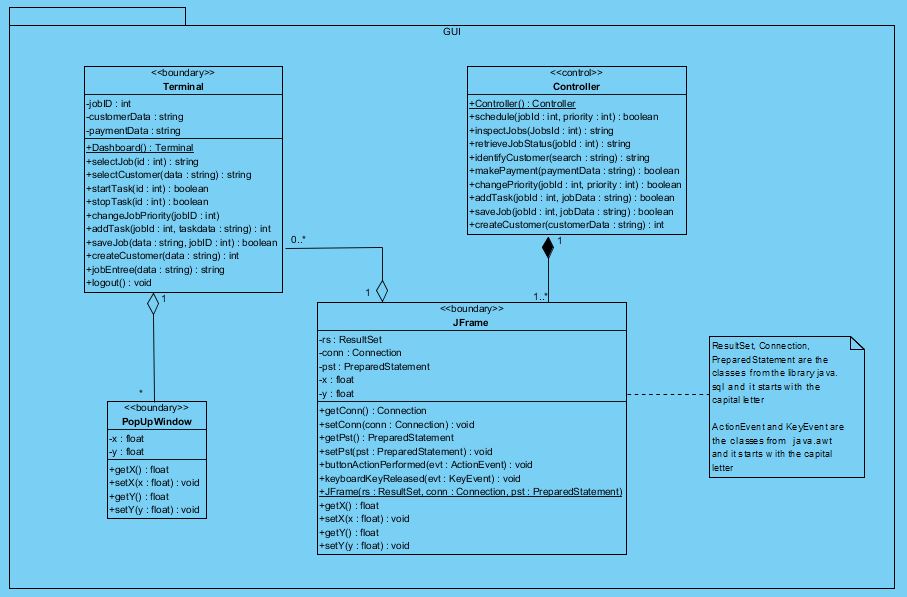


**Database**



The database we are going to use during implementation is Datagrip. There will be transaction support and concurrency of transition.

**GUI**



**5 ER Diagram**

*A screenshot of a cell phone

Description generated with high confidence*

5.1 DDL Statements

SET @OLD\_UNIQUE\_CHECKS=@@UNIQUE\_CHECKS, UNIQUE\_CHECKS=0;

SET @OLD\_FOREIGN\_KEY\_CHECKS=@@FOREIGN\_KEY\_CHECKS, FOREIGN\_KEY\_CHECKS=0;

SET @OLD\_SQL\_MODE=@@SQL\_MODE, SQL\_MODE='TRADITIONAL,ALLOW\_INVALID\_DATES';

-- -----------------------------------------------------

-- Schema bapers

-- -----------------------------------------------------

-- -----------------------------------------------------

-- Schema bapers

-- -----------------------------------------------------

CREATE SCHEMA IF NOT EXISTS `bapers` DEFAULT CHARACTER SET utf8 ;

USE `bapers` ;

-- -----------------------------------------------------

-- Table `bapers`.`band`

-- -----------------------------------------------------

DROP TABLE IF EXISTS `bapers`.`band` ;

CREATE TABLE IF NOT EXISTS `bapers`.`band` (

`volume` VARCHAR(20) NOT NULL,

`discount\_rate` FLOAT NULL DEFAULT NULL,

PRIMARY KEY (`volume`))

ENGINE = InnoDB

DEFAULT CHARACTER SET = utf8;

-- -----------------------------------------------------

-- Table `bapers`.`customer\_account`

-- -----------------------------------------------------

DROP TABLE IF EXISTS `bapers`.`customer\_account` ;

CREATE TABLE IF NOT EXISTS `bapers`.`customer\_account` (

`id` INT(11) NOT NULL AUTO\_INCREMENT,

`first\_name` VARCHAR(20) NOT NULL,

`last\_name` VARCHAR(20) NOT NULL,

`address\_1` VARCHAR(30) NOT NULL,

`address\_2` VARCHAR(30) NULL DEFAULT NULL,

`town\_city` VARCHAR(20) NOT NULL,

`county` VARCHAR(20) NULL DEFAULT NULL,

`postcode` CHAR(7) NOT NULL,

`country` VARCHAR(20) NOT NULL DEFAULT 'United Kingdom',

`type` ENUM('Default', 'Valued') NULL DEFAULT 'Default',

`contact\_no` BIGINT(11) UNSIGNED ZEROFILL NOT NULL,

`email` VARCHAR(30) NOT NULL,

`Suspended` ENUM('True', 'False') NOT NULL DEFAULT 'False',

PRIMARY KEY (`id`),

UNIQUE INDEX `customer\_account\_id\_uindex` (`id` ASC),

UNIQUE INDEX `customer\_account\_email\_uindex` (`email` ASC),

UNIQUE INDEX `customer\_account\_contact\_no\_uindex` (`contact\_no` ASC))

ENGINE = InnoDB

AUTO\_INCREMENT = 5

DEFAULT CHARACTER SET = utf8;

-- -----------------------------------------------------

-- Table `bapers`.`card`

-- -----------------------------------------------------

DROP TABLE IF EXISTS `bapers`.`card` ;

CREATE TABLE IF NOT EXISTS `bapers`.`card` (

`id` INT(11) NOT NULL AUTO\_INCREMENT,

`number` INT(16) NOT NULL,

`name` VARCHAR(20) NOT NULL,

`address` LINESTRING NOT NULL,

`postcode` VARCHAR(8) NOT NULL,

`expire\_date` DATE NOT NULL,

`ccv` INT(3) NOT NULL,

`customer\_id` INT(11) NOT NULL,

PRIMARY KEY (`id`),

UNIQUE INDEX `card\_id\_uindex` (`id` ASC),

INDEX `card\_customer\_account\_id\_fk` (`customer\_id` ASC),

CONSTRAINT `card\_customer\_account\_id\_fk`

FOREIGN KEY (`customer\_id`)

REFERENCES `bapers`.`customer\_account` (`id`))

ENGINE = InnoDB

DEFAULT CHARACTER SET = utf8;

-- -----------------------------------------------------

-- Table `bapers`.`discount`

-- -----------------------------------------------------

DROP TABLE IF EXISTS `bapers`.`discount` ;

CREATE TABLE IF NOT EXISTS `bapers`.`discount` (

`id` INT(11) NOT NULL AUTO\_INCREMENT,

`type` ENUM('Fixed', 'Flexible', 'Variable') NOT NULL,

`percentage` INT(11) NULL DEFAULT NULL,

`variable\_band` VARCHAR(20) NULL DEFAULT NULL,

PRIMARY KEY (`id`),

UNIQUE INDEX `Discount\_id\_uindex` (`id` ASC),

INDEX `volume\_idx` (`variable\_band` ASC),

CONSTRAINT `volume`

FOREIGN KEY (`variable\_band`)

REFERENCES `bapers`.`band` (`volume`)

ON DELETE NO ACTION

ON UPDATE NO ACTION)

ENGINE = InnoDB

DEFAULT CHARACTER SET = utf8;

-- -----------------------------------------------------

-- Table `bapers`.`job`

-- -----------------------------------------------------

DROP TABLE IF EXISTS `bapers`.`job` ;

CREATE TABLE IF NOT EXISTS `bapers`.`job` (

`id` INT(11) NOT NULL AUTO\_INCREMENT,

`customer\_id` INT(11) NOT NULL,

`time\_taken` TIME NULL DEFAULT NULL,

`start\_time` DATE NULL DEFAULT NULL,

`shelf\_on\_complete` VARCHAR(6) NULL DEFAULT NULL,

`priority` ENUM('Standard', 'Urgent') NOT NULL DEFAULT 'Standard',

`discount\_rate` INT(11) NULL DEFAULT NULL,

`price` FLOAT NOT NULL,

PRIMARY KEY (`id`),

UNIQUE INDEX `job\_id\_uindex` (`id` ASC),

INDEX `job\_customer\_account\_id\_fk` (`customer\_id` ASC),

INDEX `id\_idx` (`discount\_rate` ASC),

CONSTRAINT `id`

FOREIGN KEY (`discount\_rate`)

REFERENCES `bapers`.`discount` (`id`)

ON DELETE NO ACTION

ON UPDATE NO ACTION,

CONSTRAINT `job\_customer\_account\_id\_fk`

FOREIGN KEY (`customer\_id`)

REFERENCES `bapers`.`customer\_account` (`id`))

ENGINE = InnoDB

DEFAULT CHARACTER SET = utf8;

-- -----------------------------------------------------

-- Table `bapers`.`staff\_account`

-- -----------------------------------------------------

DROP TABLE IF EXISTS `bapers`.`staff\_account` ;

CREATE TABLE IF NOT EXISTS `bapers`.`staff\_account` (

`id` INT(11) NOT NULL AUTO\_INCREMENT,

`first\_name` VARCHAR(20) NOT NULL,

`last\_name` VARCHAR(20) NOT NULL,

`address\_1` VARCHAR(30) NOT NULL,

`address\_2` VARCHAR(30) NULL DEFAULT NULL,

`town\_city` VARCHAR(20) NOT NULL,

`county` VARCHAR(20) NULL DEFAULT NULL,

`postcode` CHAR(8) NOT NULL,

`country` VARCHAR(20) NOT NULL,

`role` VARCHAR(20) NOT NULL,

`email` VARCHAR(30) NOT NULL,

`password` VARCHAR(255) NOT NULL,

`contact\_no` BIGINT(11) UNSIGNED ZEROFILL NOT NULL,

PRIMARY KEY (`id`),

UNIQUE INDEX `staff\_account\_id\_uindex` (`id` ASC),

UNIQUE INDEX `staff\_account\_email\_uindex` (`email` ASC),

UNIQUE INDEX `staff\_account\_contact\_no\_uindex` (`contact\_no` ASC))

ENGINE = InnoDB

AUTO\_INCREMENT = 5

DEFAULT CHARACTER SET = utf8;

-- -----------------------------------------------------

-- Table `bapers`.`task`

-- -----------------------------------------------------

DROP TABLE IF EXISTS `bapers`.`task` ;

CREATE TABLE IF NOT EXISTS `bapers`.`task` (

`id` INT(11) NOT NULL AUTO\_INCREMENT,

`location` VARCHAR(255) NOT NULL,

`shelf\_slot` VARCHAR(6) NOT NULL,

`duration` TIME NOT NULL,

`description` LINESTRING NOT NULL,

`staff\_id` INT(11) NOT NULL,

PRIMARY KEY (`id`),

UNIQUE INDEX `task\_int\_uindex` (`id` ASC),

INDEX `task\_staff\_account\_id\_fk` (`staff\_id` ASC),

CONSTRAINT `task\_staff\_account\_id\_fk`

FOREIGN KEY (`staff\_id`)

REFERENCES `bapers`.`staff\_account` (`id`))

ENGINE = InnoDB

DEFAULT CHARACTER SET = utf8;

-- -----------------------------------------------------

-- Table `bapers`.`job\_task`

-- -----------------------------------------------------

DROP TABLE IF EXISTS `bapers`.`job\_task` ;

CREATE TABLE IF NOT EXISTS `bapers`.`job\_task` (

`date` DATE NOT NULL,

`id` INT(11) NULL DEFAULT NULL,

`job\_id` INT(11) NOT NULL,

`task\_id` INT(11) NOT NULL,

PRIMARY KEY (`date`),

UNIQUE INDEX `Job\_Task\_id\_uindex` (`id` ASC),

INDEX `job\_task\_job\_id\_fk` (`job\_id` ASC),

INDEX `job\_task\_task\_id\_fk` (`task\_id` ASC),

CONSTRAINT `job\_task\_job\_id\_fk`

FOREIGN KEY (`job\_id`)

REFERENCES `bapers`.`job` (`id`),

CONSTRAINT `job\_task\_task\_id\_fk`

FOREIGN KEY (`task\_id`)

REFERENCES `bapers`.`task` (`id`))

ENGINE = InnoDB

DEFAULT CHARACTER SET = utf8;

-- -----------------------------------------------------

-- Table `bapers`.`receipt`

-- -----------------------------------------------------

DROP TABLE IF EXISTS `bapers`.`receipt` ;

CREATE TABLE IF NOT EXISTS `bapers`.`receipt` (

`id` INT(11) NOT NULL AUTO\_INCREMENT,

`type` ENUM('Card', 'Cash') NOT NULL,

`card\_id` INT(11) NULL DEFAULT NULL,

`paid` FLOAT NOT NULL DEFAULT '0',

`balance` FLOAT NOT NULL DEFAULT '0',

`date` DATE NOT NULL,

PRIMARY KEY (`id`),

UNIQUE INDEX `reciept\_id\_uindex` (`id` ASC),

INDEX `reciept\_card\_id\_fk` (`card\_id` ASC),

CONSTRAINT `reciept\_card\_id\_fk`

FOREIGN KEY (`card\_id`)

REFERENCES `bapers`.`card` (`id`))

ENGINE = InnoDB

DEFAULT CHARACTER SET = utf8;

SET SQL\_MODE=@OLD\_SQL\_MODE;

SET FOREIGN\_KEY\_CHECKS=@OLD\_FOREIGN\_KEY\_CHECKS;

SET UNIQUE\_CHECKS=@OLD\_UNIQUE\_CHECKS;

5.2 Select, Insert, Update, Delete Statements

-- Insert

INSERT INTO customer\_account (first\_name, last\_name, customer\_id, address\_1, address\_2, town\_city, county, postcode, country, type, contact\_no, email, suspended)

VALUES (‘John’,’Smith', '0001', '60 fleet street', 'Valued Customer’);

INSERT INTO Task (task\_id, location, shelf\_slot, duration, description)

VALUES (‘0112’,’Copy room’);

-- Delete

DELETE FROM customer\_account

WHERE suspended = true;

DELETE FROM staff\_account

WHERE email ='dave.allen@bapers.com';

-- Update

UPDATE customer\_account

SET type = ‘Valued’

WHERE customer\_id = 0004;

UPDATE Card

SET name = ‘Chris Peterson’;

-- Select

SELECT first\_name, last\_name FROM staff\_account WHERE role = ‘Receptionist’

SELECT first\_name, last\_name FROM Customer\_account WHERE customer\_id lIKE %1222%

5.3 Non-Trivial Reports

-- -----------------------------------------------------

-- Individual Performance report

-- -----------------------------------------------------

SELECT first\_name,last\_name, job.id, task.id, location, duration, start\_time, time\_taken, sum(time\_taken) AS Total

FROM staff\_account INNER JOIN job INNER JOIN task

ON staff\_account.id = task.staff\_id AND job.start\_time = 2018-01-13 -- type the required date here

GROUP BY first\_name,last\_name, job.id, task.id, location, duration, start\_time, time\_taken;

-- -----------------------------------------------------

-- Summary Performance Report

-- -----------------------------------------------------

SELECT location,(SELECT day from job) AS Date , SUM(job.time\_taken) = (SELECT time\_taken FROM job WHERE job.start\_time BETWEEN 0500 AND 1430 -- Filter according to the need

GROUP BY location) AS 'Total'

FROM job

INNER JOIN (

SELECT job\_id,task\_id FROM job\_task) x ON x.task\_id = job.job\_id

INNER JOIN(

SELECT task\_id, location, start\_time, time\_taken

FROM task,job) y ON y.task\_id = x.task\_id

WHERE job.start\_time BETWEEN 0500 AND 0230 -- Filter the date according to the need

GROUP BY Date;

SELECT location,(SELECT day from job) AS Date , SUM(job.time\_taken) = (SELECT time\_taken FROM job WHERE job.start\_time BETWEEN 1430 AND 1000 -- Filter according to the need

GROUP BY location) AS 'Total'

FROM job

INNER JOIN (

SELECT job\_id,task\_id FROM job\_task) x ON x.task\_id = job.job\_id

INNER JOIN(

SELECT task\_id, location, start\_time, time\_taken

FROM task,job) y ON y.task\_id = x.task\_id

WHERE job.start\_time BETWEEN 0230 AND 1000 -- Filter the date according to the need

GROUP BY Date;

SELECT location,(SELECT day from job) AS Date , SUM(job.time\_taken) = (SELECT time\_taken FROM job WHERE job.start\_time BETWEEN 1000 AND 0500 -- Filter according to the need

GROUP BY location) AS 'Total'

FROM job

INNER JOIN (

SELECT job\_id,task\_id FROM job\_task) x ON x.task\_id = job.job\_id

INNER JOIN(

SELECT task\_id, location, start\_time, time\_taken

FROM task,job) y ON y.task\_id = x.task\_id

WHERE job.start\_time BETWEEN 1000 AND 0500 -- Filter the date according to the need

GROUP BY Date;

-- -----------------------------------------------------

-- For Period

-- -----------------------------------------------------

SELECT SUM(time\_taken) = (SELECT time\_taken FROM task WHERE time\_taken BETWEEN 0500 AND 1430 -- Filter according to the need

GROUP BY location)AS 'Shift1', SUM(time\_taken) = (SELECT time\_taken FROM task WHERE start\_time BETWEEN 1430 AND 2200 -- Filter according to the need

GROUP BY task.location) AS 'Shift 2' ,SUM(time\_taken) = (SELECT time\_taken FROM task WHERE start\_time BETWEEN 2200 AND 0500 -- Filter according to the need

GROUP BY y.location) AS 'Shift 3', location, SUM(time\_taken) = (SELECT time\_taken FROM task WHERE start\_time BETWEEN 0500 AND 0459 -- Filter according to the need

GROUP BY location) AS 'Total'

FROM job

INNER JOIN (

SELECT job\_id,task\_id

FROM job\_task

) x ON x.job\_id = job.job\_id

INNER JOIN(

SELECT task\_id, location, duration

FROM task

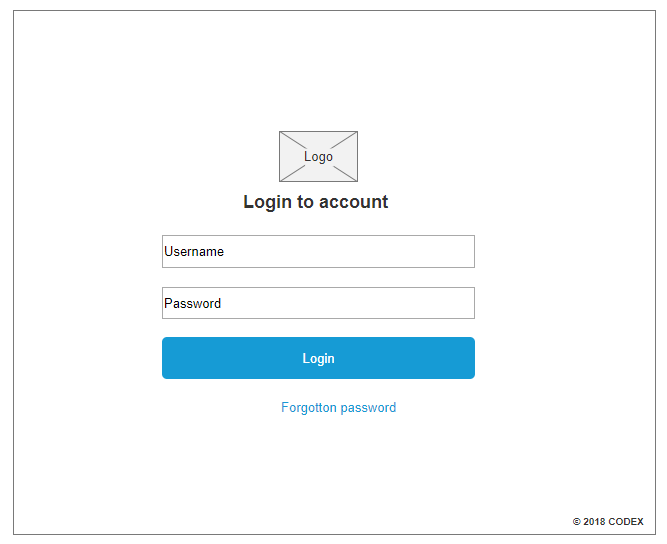
) y ON y.task\_id = x.task\_id

WHERE start\_time BETWEEN 2018-01-13 AND 2018-01-20 -- Filter according to the need

GROUP BY location

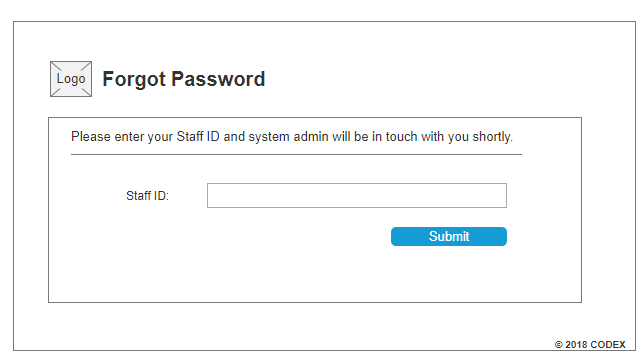
## **6 GUI Designs**

## 6.1 Visual representation and navigation between screens

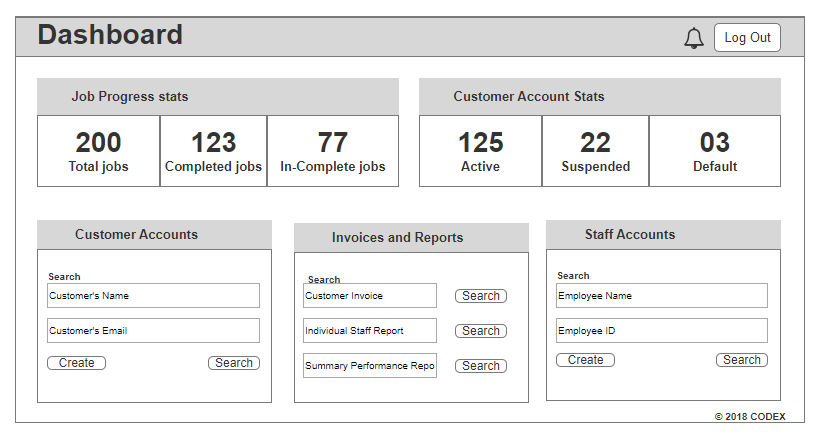
****

**Login page**

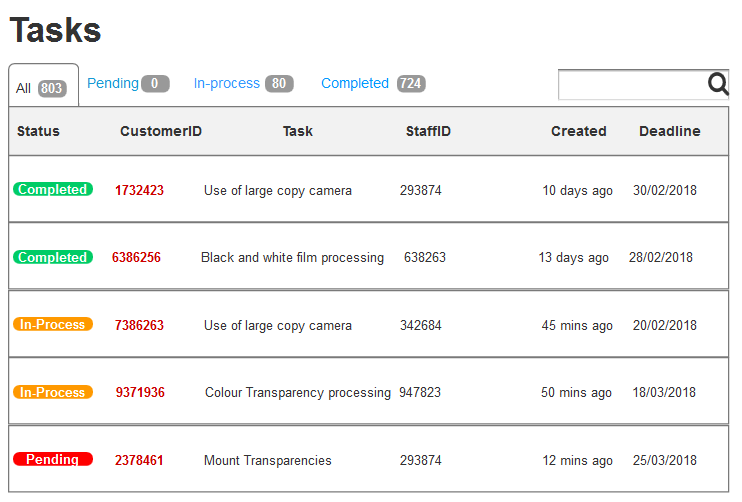
|  |  |  |
| --- | --- | --- |
| No. | Button | Interaction/Transition |
| 1 | Login | Once the user put’s in their username and respective password, they will have to click on the Login button. This will take them onto the dashboard. |
| 2 | Forgot Password | If the user has forgotten their password, they can click onto the Forgot Password button which will bring up a pop-up where they can recover their password. |

**Forgotten Password**

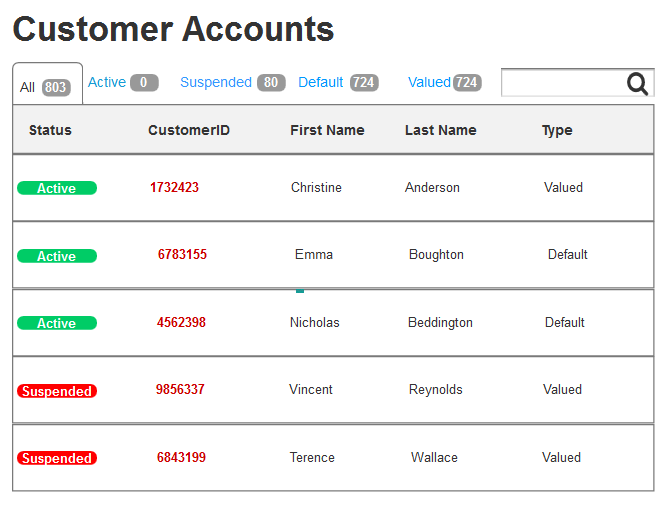
|  |  |  |
| --- | --- | --- |
| No. | Button | Interaction/Transition |
| 1 | Submit | The user needs to put their staff ID and click on the Submit button to recover their password. The system admin will be notified and will get in touch with the user. |

**Dashboard**

|  |  |  |
| --- | --- | --- |
| No. | Button | Interaction/Transition |
| 1 | Job Progress Stats | On the dashboard, the Office Manager, Shift Manager, Receptionist and the Technician has access to the Job Progress Stats. Once they click on to it, it will take them to the Job Progress Stats page where they can view the progress of all the jobs in more detail. |
| 2 | Customer Account Stats | This can be accessed by all staff members via the dashboard. By clicking on any of the numbers or the box around it, will take the user onto a sperate customer accounts form where they can view information about all types of accounts. |
| 3 | Customer Accounts – Search/Create | The Office Manager, Shift Manager and Receptionist have access to searching for customer accounts. They can type in a customer’s name or email address and click on search, which will take them onto the customer search results screen. They can also click on the create button to create a new customers account and it will navigate to another screen to create a new account. |
| 4 | Invoices and Reports | Both Office Manager and Shift Manager can search for staff reports by typing in keywords and clicking on the search button. This will direct them onto another screen. |
| 5 | Staff Accounts – Search/Create | Only the Office Manager has the authority to view the staff accounts. It will take them onto the staff search results screen and they can click on the create button if they want to create a new staff account. |
| 6 | Notifications (Bell) | All staff member’s dashboard will have a notifications bell symbol at the top, where they can click on to view updated notifications. |
| 7 | Log Out | There will be a log out button on every dashboard, where staff members can click on the log out to exit the system. |

**Job Progress Stats**

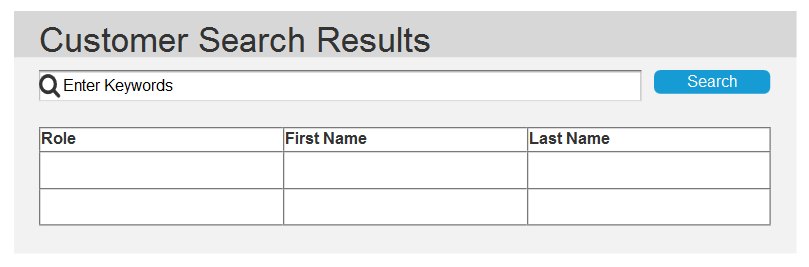
|  |  |  |
| --- | --- | --- |
| No. | Button | Interaction/Transition |
| 1 | 1. All 2. In-progress 3. Completed | By clicking on all, it will show all the jobs. Through the search bar, you can see all the tasks associated by the JobID. There are also buttons to view only Pending jobs, In-process or Completed jobs. You can view the details about the type of task, when it was created and when its deadline is. |
| 2 | CustomerID | The user can click on the CustomerID which will direct them onto the individual customer’s account. |

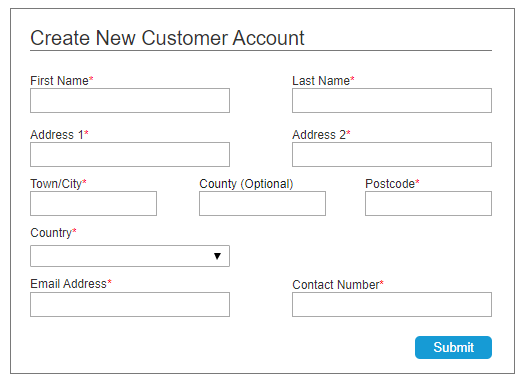
**Customer Accounts**

|  |  |  |
| --- | --- | --- |
| No. | Button | Interaction/Transition |
| 1 | 1. All 2. Active 3. Suspended 4. Default 5. Valued | All staff members can view this on their dashboard. There are many buttons at the top of the screen, to view all types of customer accounts, or they can narrow down their search to view accounts. These can be active or suspended account and default or valued accounts. |
| 2 | CustomerID | The user can click on any CustomerID on this screen and it will navigate them onto the individual customer’s account. |

**Customer Search Results**

|  |  |  |
| --- | --- | --- |
| No. | Button | Interaction/Transition |
| 1 | Search | After clicking on the search button from the dashboard, this screen will appear. The user can enter any keyword and click on the search button order to look for a customer’s account. |



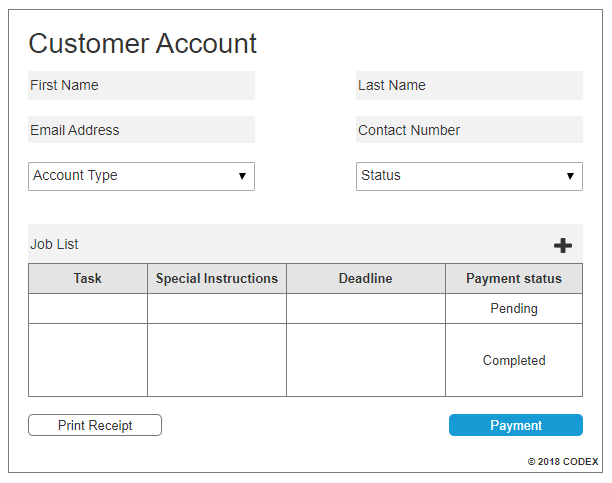
**Create New Customer Account**

|  |  |  |
| --- | --- | --- |
| No. | Button | Interaction/Transition |
| 1 | Submit | Clicking on the create button from the dashboard directs the user to this screen. To create a new customer account, here they must enter all the mandatory information before pressing the Submit button. |



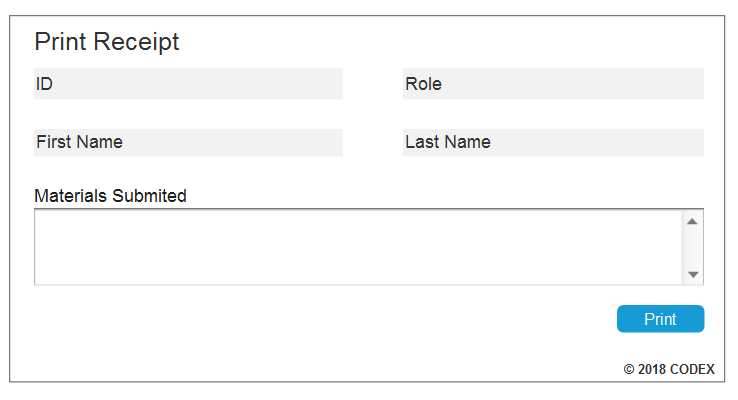
|  |  |  |
| --- | --- | --- |
| No. | Button | Interaction/Transition |
| 1 | Print Receipt | This button will be on every customer’s account where the staff can click on to print a receipt of their jobs. |
| 2 | Add Task | A list of jobs will on every customer’s account. There is an add sign where the user must click on if they want to add a new task onto the customer’s account. This will take the user onto another screen where they can add the task. |
| 3 | Payment | At the bottom of every customer account will be a Payment button underneath the list of tasks. By clicking on this button, it will take the user onto a payment page. |

**Customer Account**



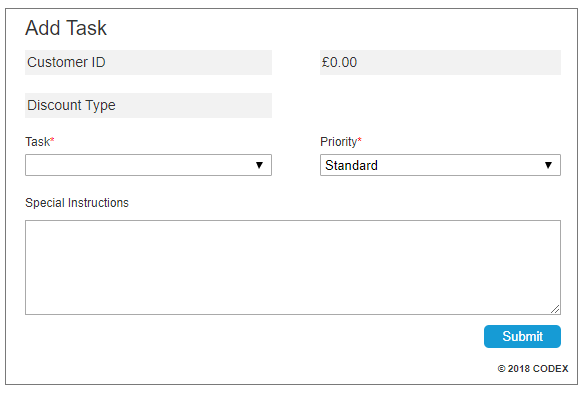
**Print Receipt**

|  |  |  |
| --- | --- | --- |
| No. | Button | Interaction/Transition |
| 1 | Print | After clicking on the print receipt button from the customer’s account, it will take the user onto this screen. Here they can view the details of the customer and see which materials were submitted. The receipt can be printed by clicking on the print button. |

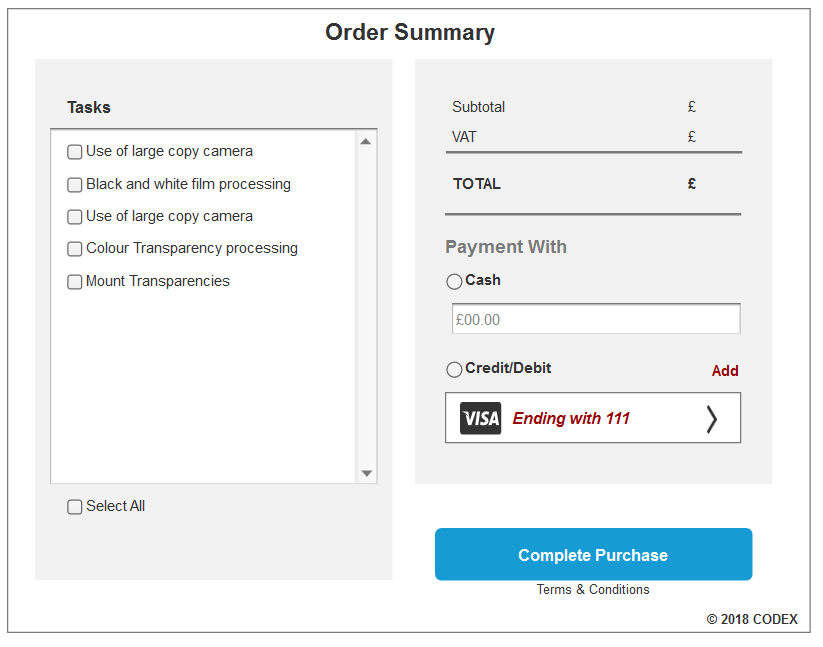


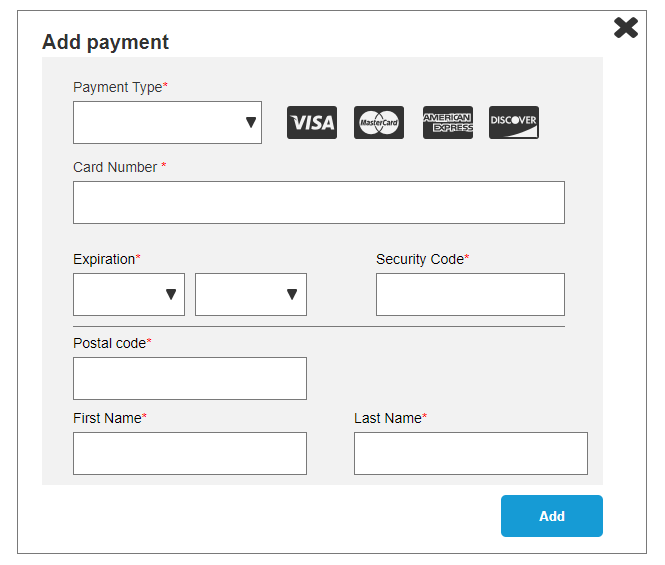
**Add Task**

|  |  |  |
| --- | --- | --- |
| No. | Button | Interaction/Transition |
| 1 | Submit | Once clicking on the add symbol from the customer’s account, the user can create a new task and add it onto the customer account. The user must fill in the type of task, select the priority and can also write any special instructions before clicking on the submit button. |

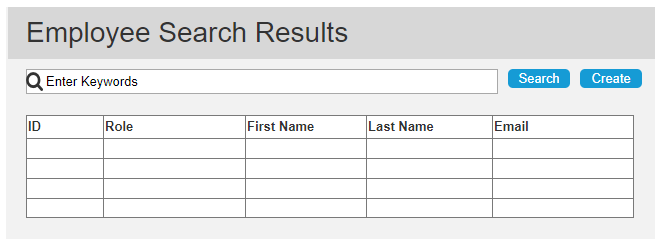


|  |  |  |
| --- | --- | --- |
| No. | Button | Interaction/Transition |
| 1 | Tasks | Clicking on the payment button on the customer’s account will direct the user to this screen. Tasks can be selected and it will automatically calculate the total cost on the right-hand side. |
| 2 | Cash/Card | There are two buttons where the user can select which type of payment method the customer wants. If the user selects the cash button, they can enter the value of payment that the customer is going to pay. |
| 3 | Add | If the Credit/Debit button is selected, it will come up with another add button. This will take the user to another screen where they can add their payment details. |
| 4 | Complete Purchase | Once the tasks and the payment method both have been selected, the user must click on the complete purchase button. |

**Payment**

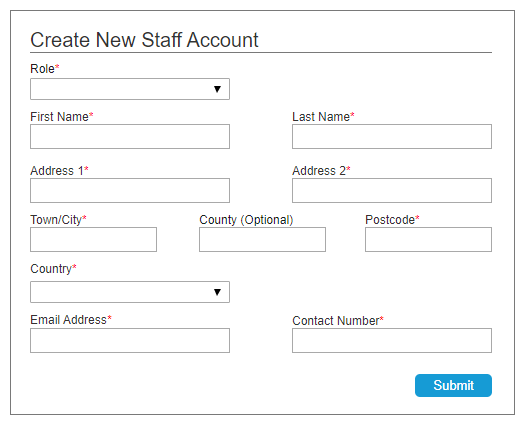
**Add Payment**

|  |  |  |
| --- | --- | --- |
| No. | Button | Interaction/Transition |
| 1 | Add | From the payment page, the add payment page will appear. Here the user must enter the mandatory payment details such as payment type and the customer’s information. Once completing, the add button must be clicked to add the payment. This will direct the screen back to the payment page. |

**Employee Search Results**

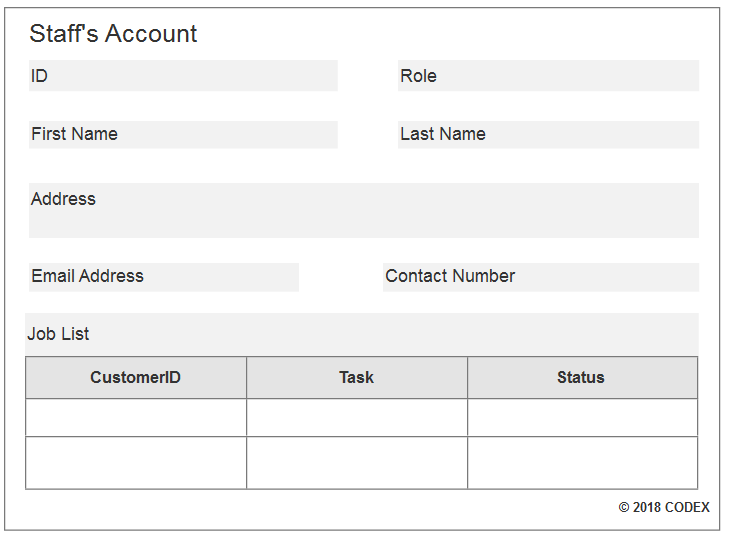
|  |  |  |
| --- | --- | --- |
| No. | Button | Interaction/Transition |
| 1 | Search/Create | Only the Office Manager can click on the search button on their dashboard and it will direct them onto this page. After clicking on the search button, It will come up with information depending on what they have typed. Also, there is a create button on this screen which the Office Manager can click on if they want to create a new staff account. |

**Create New Staff Account**

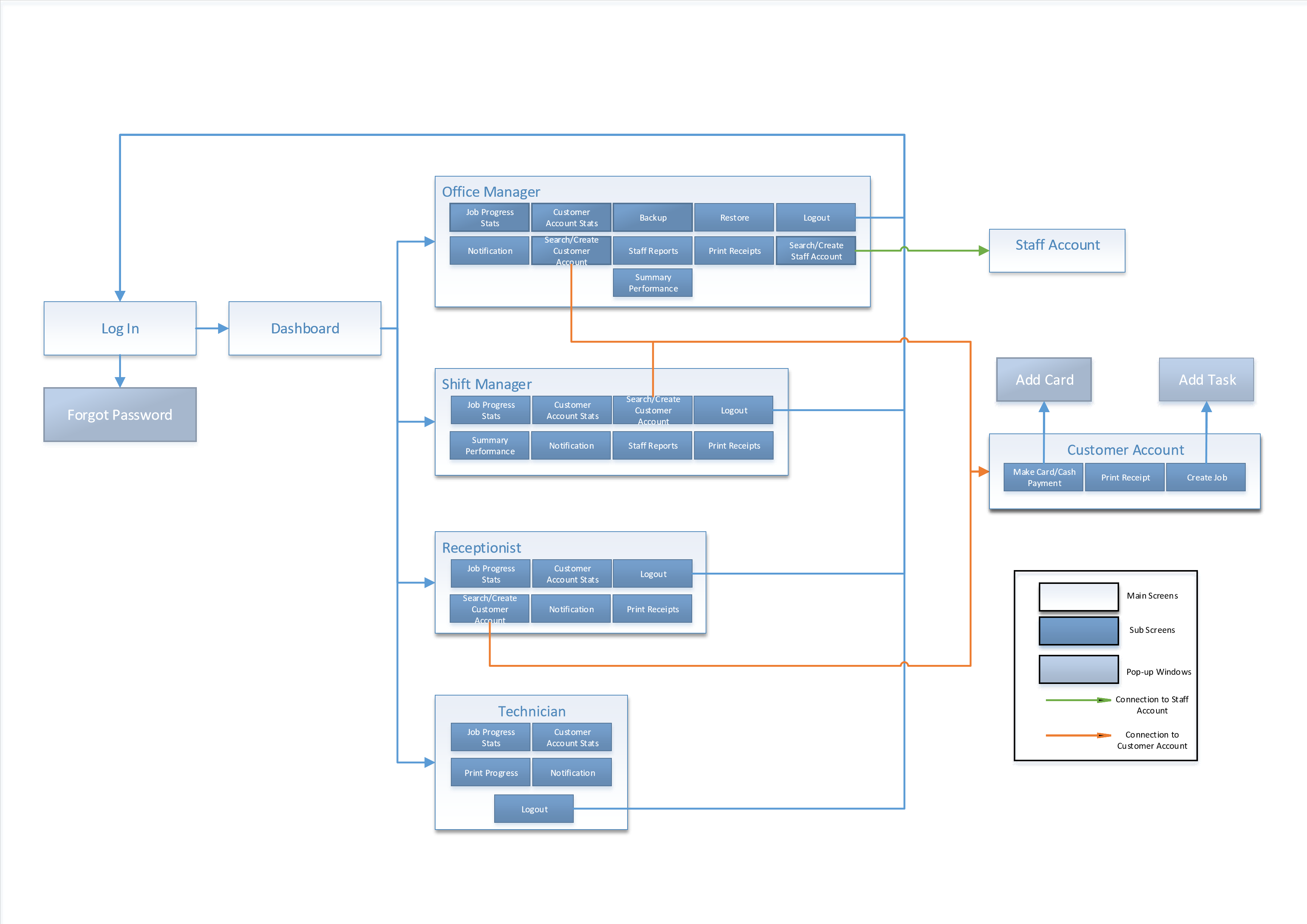




|  |  |  |
| --- | --- | --- |
| No. | Button | Interaction/Transition |
| 1 | Submit | After clicking on the create button, this screen will appear. The mandatory details must be filled out before clicking on the submit button. |

**Staff Account**

|  |
| --- |
| Interaction/Transition |
| Only the Office Manager has access to view staff accounts. This screen is obtained via dashboard by searching or clicking on an individual staff member’s name. The office manager can view their details and the jobs which have been assigned to them. There are no buttons on this screen. |



* 1. GUI Site Map
  2. GUI Mapping

|  |  |
| --- | --- |
| GUI Design Screen | Corresponding boundary classes in the class diagram |
| Login | Login |
| Forgotten Password | ForgotPassword |
| Dashboard | Terminal |
| Office Manager Dashboard | OfficeManager |
| Shift Manager Dashboard | ShiftManager |
| Receptionist Dashboard | Receptionist |
| Technician Dashboard | Technician |
| Customer Account Stats | CustomerAccountStats |
| Customer Search Result | Search |
| Create New Customer Account | CreateCustomerAccount |
| Customer Account | CustomerAccount |
| Print Receipt | PrintReceipt |
| Add Task | Task |
| Task Stats | TaskStats |
| Payment Page | Payment |
| Add Payment | CardDetails |
| Staff Search Results | Search |
| Create New Staff Account | CreateStaffAccount |
| Staff Account | StaffAccount |