BAPERS Software

Summary

This is a Software Requirements Document, a proposal to Mr Lancaster of BAPERS and our consultant, from the software development team at Digital Inspirations.

Authors

Anthony Gibson, Sharmistha Dutt, Hamzah Malik, Martin Ivanov, Mariia Onyshchenko Devina Thakker

Distribution

Mr Lancaster and the consultant.

Contents

1. Preface

1.1 Purpose and scope of the document

This is a software requirements document for the BAPERS Software that covers the requirements with the various implementation constraints defined in the initial statement of requirements by Mr Lancaster of BAPERS, the client.

This document is here to explain what our proposed system shall do. It features various models to convey ideas in more detail to the system developers, whilst showing the expected behaviour of the system.

1.2 Intended audience

Mr Lancaster from the BAPERS lab, and our consultant.

1.3 History of the document

Version Number	Date	Revision Authot	Description
V1.0	6th February	Hamzah	This is a new document
V1.0.1	8th February	Hamzah	Updated document to include company logo, a summary page, with authors and distributions made and an update to the table of contents layout.
V1.1	10th February	Hamzah	Added a purpose of scope of the document, describing the purpose of the document. Listed the intended audience and wrote up an introduction, describing the existing system and its inefficiency.
V1.2	11th February	Anthony, Hamzah	A first attempt at a use case priority list made with a complete set of use cases.
V.1.2.1	12th February	Anthony, Hamzah	After detailed group discussion, a **final set** of use cases has been assigned and created, and a priority list has been established.
V.1.3	16th February	Sharmistha, Devina	A rough Design class diagram has been uploaded, with classes and operations decided together.
V.1.3.1	18th February	Hamzah, Anthony	A **FINAL** use case priority list has been made and a use case specification has been uploaded
V.1.4	19th February	Mariia, Martin	A selection of GUI's have been established and made.
V.1.5	22nd February	Sharmistha, Devina	A **FINAL** design class diagram has been made and a package diagram has been made
V.1.5.1	24st February	Hamzah, Anthony	A **FINAL** use case diagram has been made alongside a **FINAL** use case specification
V.1.6	25th February	Sharmistha, Devina	A **final package** diagram has been uploaded and **a final** ER diagram with SQL scripts have been updated

V.1.7	27th February	Mariia	A **final GUI** has been made with appropriate wire frames, mapping, connection to design classes. Completed GUI.

2. Introduction

2.1 Description of the existing system

Bloomsbury's Image Processing Laboratory, or BIPL, is a laboratory that handles the work of photographers who commission work to be processed at the lab. This is mainly attributed with the printing of photographs provided by customers. BAPERS prides itself the hope of producing perfect results every time, whilst working within tight deadlines without sacrificing any quality.

The main issue is the paper based system that already exists for BIPL. Everything completed is done by hand and this is a major issue in several ways.

One of these issues is human error, when moving work from one place to another, there is an inevitable opportunity for work to be lost or misplaced by staff members.

Another is efficiency and time; a lot of time is wasted by doing something by hand when it can be completed much faster by a system. There are also environmental concerns that arise from focusing on a paper-based system too.

A massive flaw within a paper-based system is communication. It is very hard to identify a customer record amongst a collection of other records. This can lead to misplaced records or lost customer records, which could be very devastating in regard to customer satisfaction. Currently, the only way of informing other staff members about customer information is to manually go to them and inform them about updates to the job and task and anything else that is needed to be done.

A recurring theme about the existing system is time, when receiving a job on the current system, the receptionist will forward information about the job to the other employees in the lab by directly going to them and informing them about the task. From here, the staff member will write down the progress of completed tasks by hand and continue to update on this.

Payments are sometimes left unlogged and there is no way to access these payments. Once again, this affects customer satisfaction and can affect the company, due to significant things such as being unable to access previous payments, if a customer would like to review them.

3. Requirements Specification

3.1 *Use Case Priority List*

Before creating our use case diagram, we decided it was important produce a list of all the use cases that cover the functionality of the BAPERS system and highlight their priority.

We devised a marking criterion, where the priority of each use case was placed on scale between one and five, one being the most prioritised use cases, and five being the least prioritised. The “priority” of a use case was determined by the user’s priorities for the system and the impact of projected risks during development. This includes issues such as time restraints and budget issues.

The full set of use cases and their priority are labelled with their reasoning and priority level down below.

Use Case ID	Use Case Name	Priority Level	Reasoning
1	Create Job	1	Creating a job and completing a job must be completed within 24 hours of acceptance. Although they have a high priority in the BAPERS system as all jobs must be completed to keep up a good brand image, they are slightly less important than urgent jobs which must be completed within a time scheme of 6 hours, or in some cases, under three hours. However, it should be clear that creating jobs is a very important process in the BAPERS system as the system essentially revolves around creating and completing jobs in a set amount of time.
2	Print Receipt	2-3	Although some customers may not request a receipt to be printed, receipts are mandatory as evidence of a job being paid for and should be given to the customers. Not printing a receipt for the payment of a job may appear malicious by staff members and provides customers no proof of a job being paid for. Once again this is important for customer satisfaction.
3	Accept Urgent Job	1	Accepting an Urgent Job is very important as they must be completed in a set amount of time (under 6 hours) so their acceptance is very important. If an Urgent Job has been accepted too late, this can be detrimental for the company if an urgent job has not been completed in time in terms of brand image.
4	Alert Shift Manager	2	It is important that the shift manager is alerted to whenever a job has been created, so the job can be logged and stored in the database and each task completion can be updated.
5	Assign Job Number	3	In terms of user's priorities, assigning a job number to each made job is important as BAPERS may receive multiple jobs per day and having a job number can make finding a specific job assigned to a customer account a lot easier. Despite this, its priority isn't as important, as without a job number, a job can still be manually found by scrolling down a list of jobs for the day. As jobs are given a deadline of 24 hours, the amount of jobs received in one day won't be detrimental to manually scroll through and find the specified job requested, although it might take slightly longer than manually entering the job number itself, affecting a time constraint.
6	Add Task	2	Adding tasks are high in priority for users as tasks can be used to measure progress for an overall job over a set of completed or in progress jobs.
7	Identify Customer Account	1	Identifying a customer account is one of the most important use cases in the system. Not being able to identify a customer

			account means a job cannot be assigned to the customer account. If the customer already has an account and the account cannot be identified, this causes even more issues as the staff must figure out why and work out what the issue. This will take time and become detrimental in terms of time problems.
8	Create Customer Account	1	If a Customer Account can't be found, creating a customer account is the next thing an Office Manager must do, and it is vital to the BAPERS system. Creating a customer account allows the customer to have a valid job assigned to their name alongside the Job Number and other such information.
9	Update Task Completion	2-3	This use case is dependent on the way the users work on jobs. If tasks are updated incrementally and in small updates, then the use case is very important for the users, whereas if the users update a tasks completion once it has been completed, then its priority will be lower.
10	Set Complete	2	Setting a tasks completion lets other staff members in BAPERS know which tasks have been completed and lets them work on other tasks for the job. If all tasks for a job have been completed, it lets the users move to another job now the job has been completed.
11	Set Progress	2	Similarly to set complete, having an 'in progress' task allows the users of BAPERS to know which tasks still must be completed within a given time (24 hours for normal jobs, 6 for urgent jobs)
12	Inspect Jobs	2	Being able to inspect jobs is important as it allows the staff member to see a list of completed, in progress and a complete list of all jobs that have existed in the past. This may be useful to satisfy job deadlines, to see a list of active jobs that have not yet been completed.
13	Inspect Tasks	3	Inspecting individual or all tasks under specific jobs is important, but not as important as inspecting an overall job. It is expected that staff member of BAPERS are more concerned about the completion of a job rather than smaller sub tasks in the whole job.
14	Reactivate Customer Account	1	This is done automatically and is important as customer's accounts must be reactivated once a late payment has been made for an account that has been placed in 'default'.
15	Downgrade Customer Account	3	Although unlikely, a customer account may be downgraded for late payments from a valued customer. This would be a rare occurrence however and in terms of user priority, it's not as important to downgrade a customer account in comparison to use cases such as creating jobs.
16	Upgrade Customer Account	3	Similar to above, its rare a customer account is upgraded to a 'valued' customer unless they have a good track record of payments paid on time and this is decided by the Office Manager. This is not as important as other key use cases and is not high on the user's priorities.
17	Set Discount Plan	3	Although setting a discount plan is key for customer satisfaction and loyalty, in terms of user priorities it is not as important as use cases such as accepting urgent jobs or logging

			into the system. The likelihood of having a 'valued' customer to set a discount plan is also less likely than a regular customer. Despite this, the Office Manager having the ability to set discounts for valued customers is important for customer satisfaction and therefore will be included in our use case specification. Furthermore, a Discount Plan should be something only for valued customers that have proved they are consistent in payment. This will end up being only a select list of people and so setting a discount plan will usually be used sparingly, which is why its priority is 3. If everyone was offered a discount plan, it would be detrimental in terms of budget.
18	Set Fixed Discount	3	This use case enables the office manager to decide on a fixed discount percentage for a valued customer. This is one of three discount options that are important, as one must be assigned to a valued customer account.
19	Set Variable Discount	3	This use case enables the office manager to decide on a variable discount percentage for a valued customer. This is one of three discount options that are important, as one must be assigned to a valued customer account.
20	Set Flexible Discount		This use case enables the office manager to decide on a flexible discount percentage, that can change, for a valued customer. This is one of three discount options that are important, as one must be assigned to a valued customer account.
21	Login	1	Being able to log in to the BAPERS system is very important for users. Not being able to log in can be detrimental for users as they cannot access the system features without logging in. Furthermore, different users can access certain sub-systems within BAPERS and without login privileges, they cannot access these subsystems.
22	Logout	1	Logging out of the system is mandatory for users accessing the system which is why it gets such as high priority level. Not logging out of the BAPERS system can lead to data breaches and vulnerable information about customers and users alike left accessible to external threats.
23	Monitor Payment Record	1	Monitoring payment records is vital, as it allows BAPERS users to keep track of what customers have paid for their jobs, which haven't, and those that have paid late (This could affect the choice in deciding if a customer deserves to be upgraded to a Valued Customer).
24	Record Payment Amount	1	Recording payment amounts is also very important as it allows the BAPERS Front Staff to monitor how much of the job has been paid off. This also can be used to decide whether a reminder letter should be sent notifying the customer that the entirety of the payment has not yet been paid off by the given deadline.
25	Record Card Details	2	Recording card details is important for users as it allows the customer to not have to enter their details each time, as some of their information is already stored.

			Despite this, some customers may not want their information to be stored so ethical issues may arise from having card details if the customer don't want it.
26	Create User Account	1	Creating a user account is vital, as Users of the BAPERS system cannot access subsystems within BAPERS without an account and password being created by the Office Manager.
27	Generate Reports	1	Generating reports is vital, as it allows staff members to see the types of reports that have been generated.
28	Generate Summary Report	1	Generating summary reports is important as it allows the office manager to see the summary performance report for work undertaken by BIPL during the day and night shifts.
29	Generate Individual Report	1	Generating individual reports are important as they highlight the jobs brought in by a customer.
30	Generate Individual Staff Report	1	Generating an individual staff report is important and has a high priority as it allows the Office Manager to see the performance of individual staff members for work undertaken.
31	Complete Manual Database Restore	5	Although database restores are important, they are not of a very high priority to the Office Manager as the system automatically restores the database after a set amount of time. The existence of this use case is to allow manual database restores to exist.
32	Complete Manual Database Backup	5	Although database backups are important, they are not of a very high priority to the Office Manager as the system automatically backs up the database after a set amount of time. The existence of this use case is to allow manual database backups exist.
33	Print Reminder	1	Printing reminder letters is essential within the BAPERS system because the office manager may only want to print a specific set of reminders.
34	Specific Reminder	1	This use case is important as it allows the Office Manager to print a specific reminder.
35	Batch Reminder	1	Similarly to the use case above, this use case allows the Office Manager to print reminders; however they can be printed in a batch set of reminder.
36	Individual Reminder	1	This allows the Office Manager to print a specific reminder, and thus is important to the system for individual reminders.
37	Auto Generate First Reminder	1	Auto generating the first reminder is an important use case as it is triggered automatically by time. This reminder is sent to the office manager and is high on user priorities as it allows reminds the Office Manager to print a reminder letter.
38	Auto Generate Second Reminder	1	Auto generating the second reminder is an important use case as it is triggered automatically by time, one month after the initial reminder. This reminder is sent to the office manager and is high on user priorities as it allows reminds the Office Manager to print a second reminder and the customer's account is automatically terminated.
39	Suspend Account	1	Having the use case suspend account is important and high on the priority list as the customer account is automatically terminated once a second reminder letter has been sent. This is important to staff member under BAPERS as they can see

			which customers are now placed in 'default' after being terminated for not paying for completed jobs.
40	Detect Late Payments	1	Detecting late payments for jobs is one of the most important use cases in the system as it allows the Office Manager to be alerted on late payments that have not yet been paid by customers.
41	Alert Office Manager	2	As mentioned above, alerting the office manager about any late payments is very important within the BAPERS system as it allows them to see whenever a customer has gone over due a set deadline for payments.
42	Auto Restore	2	Having the ability to auto restore in any system is very important, and within the BAPERS system with many different active jobs and previous jobs, ensuring that vital information is restored is very important. Other data such as stored customer account details should be restored automatically after a set amount of time and ensuring this happens is vital.
43	Auto Backup	2	Similarly, when dealing with a system with large volumes of data, one must ensure that all data is backed up. Whether that is customer accounts that have been created, tasks and jobs that have been completed or updated or any other data, the system must ensure this is all backed up onto the Database.

3.2 Use Case Specifications

Once finalising our use cases that will be used in our use case diagram, we chosen ten key use-cases that we believed were significant in the functionality of the BAPERS system.

ID: UC21	**Use Case:** *Login*
Brief description: A staff member with a valid BAPERS user account can log into the computer terminal.	
Primary Actors: Receptionist, Shift Manager, Office Manager, Technician (All).	
Secondary Actors: None	
Preconditions: 1) BAPERS is operational 2) The Staff Account is a valid staff account and has already been created.	
Flow Of Events: 1) The user inputs their username and password for access into the BAPERS system 2) The user clicks on the login button to enter.	
Postconditions: 1) The user is logged into the computer terminal and has access to specific subsystems within the BAPERS system, based on their privileges and restrictions.	
Alternative Flow: 1) UnregisteredUser 2) IncorrectLoginDetails	

Alternative flow: *UnregisteredUser*
ID: UC21.1
Brief description: The Staff Member could not log in

Primary actors: Receptionist, Shift Manager, Office Manager, Technician (All).
Secondary actors: None
Preconditions: 1. A staff member is trying to log into the BAPERS subsystem but does not have a valid account set up.
Alternative flow: 1) The alternative flow begins after Step 2 of the main flow. 2) The system will display an error message saying “Account not recognised, please try again or create a new account”
Postconditions: The ‘Create Account’ button will be highlighted.

Alternative flow: *IncorrectLoginDetails*
ID: 21.2
Brief description: A staff member is trying to log into the BAPERS subsystem but can’t.
Primary actors: Receptionist, Shift Manager, Office Manager, Technician (All).
Secondary actors: None
Preconditions: 1. The staff member attempting to log into the BAPERS system has a combination of either an invalid username, or password, or perhaps both. This does not match the username/password stored in the BAPERS database.
Alternative flow: 1. The alternative flow starts after Step 2 of the main flow. 2. The system will display an error message saying, “Invalid login details, please try again”.

ID: UC26	Use Case: *CreateUserAccount*

Brief description:
The Office Manager has the option to create a new user account for the staff members who will be using the BAPERS system.

Primary Actors:
Office Manager

Secondary Actors:
Database

Preconditions:
1) BAPERS is operational
2) User currently does not have an account.

Flow Of Events:
1) The Office Manager searches for the user account and establishes a new user account is required for BAPERS.
2) The Office Manager chooses the type of user account that is to be created, determining their privileges and restrictions based on whether the account is for a receptionist, technician, office manager or shift manager.
3) The Office manager will then input new user details about the staff member and create the account.

Postconditions:
1) The specific user for the BAPERS system now has a valid customer account to log into.

Alternative Flow:
None

ID: UC1	Use Case: *CreateJob*

Brief description:
Front Staff users can create jobs for customers which are made up of sub tasks that must be completed.

Primary Actors:
Front Staff (Receptionist, Shift Manager, Office Manager)

Secondary Actors:
Database

Preconditions:
1) BAPERS is operational and a member of the Front Staff is logged in.
2) The customer has a valid customer account to assign the created job to.

Flow Of Events:
1) A User of the Front Staff navigates and finds the customer account linked to the request job that is to be created.
2) Once found, information is entered to describe what the specific job entails, and the job is created.
3) The job is assigned a job number to help find the specific job easier when navigating and searching through the system.
4) A receipt is printed out describing the details of the created job.
5) The Shift Manager is alerted to the creation of a new job.

Postconditions:
1) The customer now has a job assigned to their account and Front Staff can add new tasks under the job.
2) Information such as the creation of the job and its job number is stored within the database.

Alternative Flow:
None

ID: UC9	Use Case: *UpdateTaskCompletion*

Brief description:
Staff members can update the completion of existing tasks under a specific job. This covers a range of minor tasks or major tasks under an individual job that has been agreed to with the customer.

Primary Actors:
Front Staff (Receptionist, Shift Manager, Office Manager)

Secondary Actors:
Database

Preconditions:
1) BAPERS is operational
2) A job has been accepted, and created by a member of the Front Staff

Flow Of Events:
1) Whenever a task has been completed, a member of the Front Staff can decide to update the tasks completion.
2) The Front Staff identifies the customer account and find the task under the job that is currently under way.
3) The Front Staff highlights the changes made to the task and leaves the task either on 'SetProgress' or 'SetComplete' when a task has been completed for the customer.
4) The Front Staff then updates the task to show the changes made.

Postconditions:
1) The Task is updated, showing the progression of the overall job to the Customer.

Alternative Flow:
None

ID: UC23	Use Case: *MonitorPaymentRecord*

Brief description:
Payments are monitored on the system by creating a payment record and storing details on the type of payment and the specific payment amount.

Primary Actors:
Front Staff (Receptionist, Office Manager, Shift Manager)

Secondary Actors:
Database

Preconditions:

1) BAPERS is operational.
2) A user account from a member of the Front Staff is logged into a valid staff account.

Flow Of Events:

1) Monitoring payment records begins once a payment has been received for a completed job by the customer.
2) The Front Staff then searches for the specified job.
3) **IF** the customer is a valued customer
   3.1) One of the three given discount plans can be applied.
4) **ELSE**
   4.1) A customer continues to pay for their jobs all together or individually.
5) The payment amount and the specific payment type (Card or Cash) is entered.
6) **IF** the payment type is by card
   6.1) The Front Staff record additional information such as the expiry date, the type and the last 4 digits of the card used.
7) A record of the current payment and any future payments will now be stored into the BAPERS database.

Postconditions:

1) A payment record is now stored within the BAPERS database and can be pulled up on request of the customer.
2) Card details of the customer is stored in the payment record, useful for any future payments for services.

Alternative Flow:
PaymentDeclined

Alternative flow: *PaymentDeclined*
ID: UC23.1
Brief description: A customer's payment is declined when a transaction paid by card has been declined.
Primary actors: Office Manager
Secondary actors: None
Preconditions: 1) Insufficient funds on the customers card/PIN entered incorrectly.
Alternative flow: 1) The alternative flow begins after Step 4 of the main flow. 2) The system will display an error message saying, "Transaction Failed, please try again".
Postconditions: 1) An on-screen prompt will appear, signalling to retry the payment.

ID: UC17	Use Case: *SetDiscountPlan*
Brief description: A valued customer can be offered a discount plan which either revolves around a Fixed Discount plan, a Variable Discount plan or a Flexible Discount plan.	
Primary Actors: Office Manager	
Secondary Actors: None	
Preconditions: 1) BAPERS must be operational. 2) An Office Manager has logged into the computer terminal, gaining access to the list of customer accounts.	
Flow Of Events: 1) An Office Manager finds the specific valued customer account amongst the list of other valued customers. 2) Once found, an Office Manager can choose a discount plan for the valued customer. 3) Once decided upon, the Office Manager proceeds to select either a fixed discount plan, a flexible discount plan, or a variable discount plan.	
Postconditions: 1) The system will record which specific discount has been associated to the valued customer account.	
Alternative Flow: IncorrectCustomerType	

Alternative flow: *IncorrectCustomerType*
ID: UC17.1
Brief description: This occurs when the Office Manager attempts to apply a discount on a non-valued customer account.
Primary actors: Office Manager
Secondary actors: None
Preconditions: 1. The customer account is not recognised to be a 'ValuedCustomer' in order to seek the benefits of receiving a discount plan.
Alternative flow: 1) The alternative flow begins at Step 1 of the main flow. 2) The Office Manager will notice that the customer account sent to be assigned a discount type is not on the list of valued customers. 3) The System will display a pop up message for the other staff members within BAPERS to see that, which will say "This Customer Account is not part of the ValuedCustomer list".
Postconditions: 1) The customer account is placed back on the list of regular customers, instead of being seen as a 'valued customer'.

ID: UC41	Use Case: *AlertOfficeManager*
Brief description: The system detects any unpaid late payments after a set amount of time, alerting the Office Manager about the unpaid payment for a specific customer.	
Primary Actors: Time	
Secondary Actors: None	
Preconditions: 1) BAPERS is operational 2) The system has detected a late payment.	
Flow Of Events: 1) The system detects customers that have not paid in time, for their deadline. 2) The system will send out an alert message to the office manager through pop up messages on screen. 3) This alert will repeat itself several times every fifteen minutes until the 'Office Manager' has acknowledged the alert.	
Postconditions: 1) BAPERS has now alerted the Office Manager, giving a list of all the customers who have not paid in time.	
Alternative Flow: None	

ID: UC14	Use Case: *ReactivateCustomerAccount*
Brief description: After having their account terminated, if a customer pays back a late payment before legal action has been taken against them, the office manager will reactivate their account.	
Primary Actors: Office Manager	
Secondary Actors: None	
Preconditions: 1) BAPERS is operational and Office Manager has logged into BAPERS. 2) The customer account has already been terminated after a second reminder has been sent to the customer.	
Flow Of Events: 1) A one-month timeline is set after a second reminder has been sent, alerting the Office Manager informing them of any payments that has been made by a customer who's account has been terminated. 2) The Office Manager noticed the alert automatically generated by the system. 3) The Office Manager chooses to reactivate the customer account that has been terminated and placed in 'default'.	
Postconditions: 1) The customer account has been reactivated and they can request jobs as normal, now that the amount required has been payed off.	
Alternative Flow: None	

ID: UC33	Use Case: *PrintReminder*

Brief description:
If a valued customer fails to clear an outstanding balance for a late payment, a reminder letter is sent out to them This can happen twice before the account is then terminated.

Primary Actors:
Office Manager

Secondary Actors:
None

Preconditions:
1) BAPERS must be operational
2) The job requested by the customer has been completed.
3) The Office Manager is logged into the computer terminal.

Flow Of Events:
1) When the Office Manager has logged into his/her account, the system props an alert informing the Office Manager of the late payments.
2) The Office Manager prints out a reminder letter based on whether a first reminder letter has been sent already, or whether this is a second reminder.
3) The Office Manager can choose the type of reminder letter being printed, from a choice of either batch reminders, individual reminders or specific reminders.
4) The choice in letter is then printed out, ready to be sent to the specific customer.

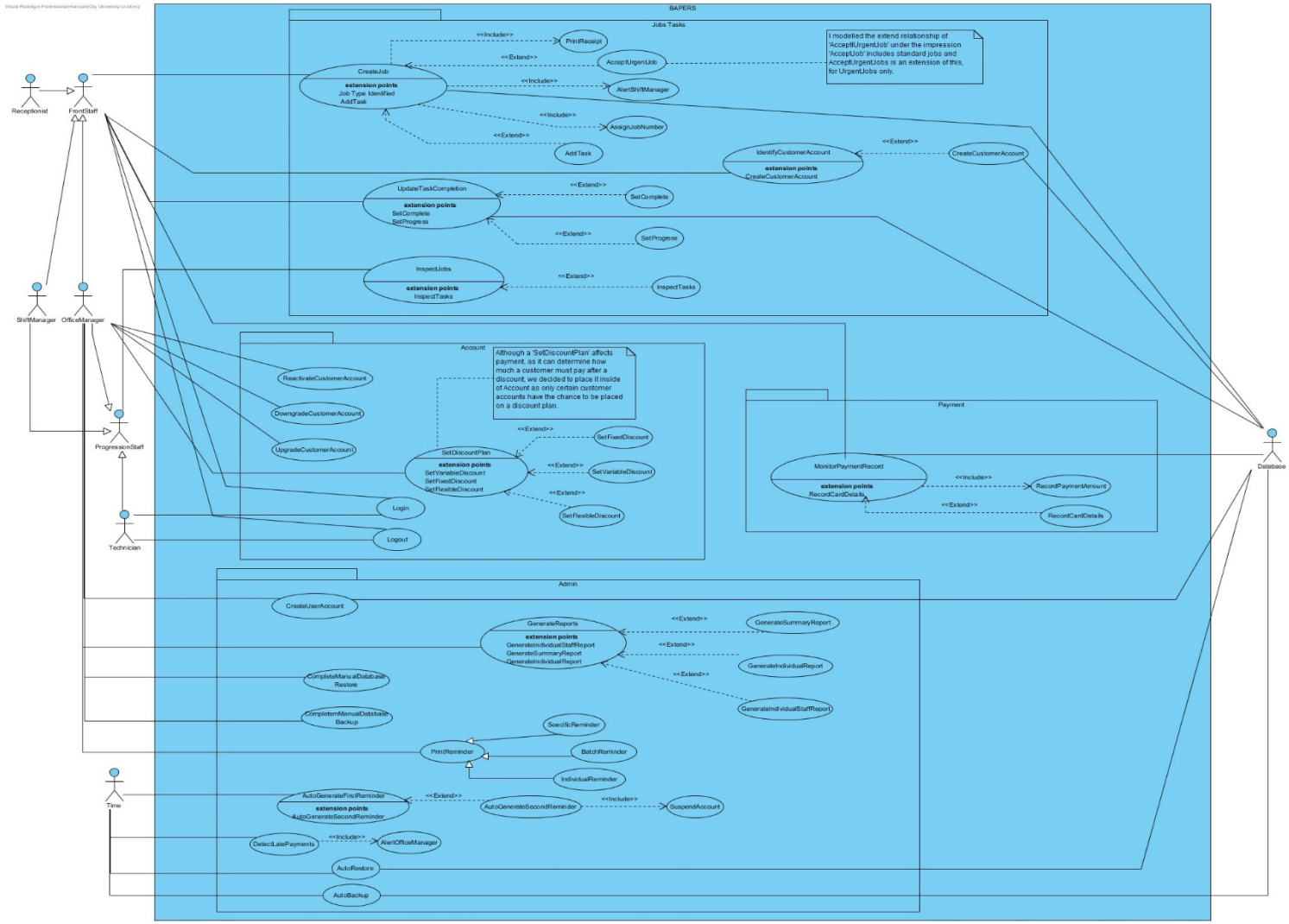
Postconditions:
1.) A reminder letter has been sent out to the customer.
2.) The system shall automatically suspend the valued customer account if a second reminder letter has been sent. The letter itself will inform the customer their account has been suspended from accessing BAPERS until a payment has been processed.

Alternative Flow:
PrintFailure

Alternative flow: *PrintFailure*
ID: UC33.1
Brief description: The printer could not print out the correct reminders either due to a lack of ink or a technical issue.
Primary actors: Office Manager
Secondary actors: None
Preconditions: 1. An Office Manager is attempting to print but the printer fails.
Alternative flow: 1) The alternative flow begins during Step 1 of the main flow. 2) The printer will throw an error and not print the reminder letter correctly.
Postconditions: 1) The Printer will display an error automatically, usually stating what the problem may be.

ID: UC22	Use Case: *Logout*
Brief description: Any staff member that has a created user account can log out of their account.	
Primary Actors: Technician, Receptionist, Shift Manager, Office Manager.	
Secondary Actors: None	
Preconditions: 1) BAPERS is operational 2) A user account has been logged into the system.	
Flow Of Events: 1) This use case becomes available once a staff member has finished using the BAPERS system and chooses to log out. 2) The system logs the user out of their staff account. 3) The login screen will then prompt the user to log in, via a log in page.	
Postconditions: 1) The user will be logged out of their User Account 2) The system will display a login page, prompting the user to log back into their account to continue access to the system.	
Alternative Flow: None	

3.3 *Use Case diagram*



Actors

FRONT STAFF -
The 'Front Staff' is made up of the three actors 'Receptionist', 'Shift Manager' and 'Office Manager', generalized under this one actor. The main responsibility of the Front Staff in our system is to create jobs, customer accounts, and handle the responsibility of assigning tasks to a specified job, assigning each job a job number, printing receipts of created jobs, and the added ability to accept an urgent job if needed. Front Staff also must alert shift managers when creating jobs, and can find customer accounts if needed.

The FrontStaff actor exists as Office Manager, Shift Manager and Receptionist all have access to BAP-ACCT, so a generalized actor has been made to simplify our use case diagram, as all three actors have access to BAP-ACCT subsystem.

PROGRESSION STAFF-
Much like the 'FrontStaff' actor, the 'ProgressionStaff' actor is made up of three different actors generalized under this one, those being: Technician, OfficeManager and ShiftManager.
This Actor exists as the Technician, Office Manager and Shift Manager all have access to the BAP-PROC subsystem, and thus a generalized actor has been made to once again simplify our use case diagram.

OFFICE MANAGER-
The Office Manager has access to **all** subsystems of BAPERS and so completes a ranged set of activities within the system. The OfficeManager can reactivate accounts that have been terminated for not paying fees. They have the authority to deactivate and upgrade accounts to valued customer accounts, whilst being the only actor within BAPERS that can set a discount plan for 'valued customers', that are decided by the Office Manager. They can also create 'User Accounts' for other users within BAPERS and complete manual database backups and restores, something which is also automatically completed over a set amount of time.

RECEPTIONIST-
The Receptionist class usual tasks such as accepting jobs has been already generalized under the 'FrontStaff' actor, which does whatever Receptionist can do.

SHIFT MANAGER-
Similarly to our 'Receptionist' actor, Shift Manager is generalized, but under both Front Staff and Progression Staff.

TECHNICIAN-
The technician is generalized under the Progression staff Actor.

TIME-
Although time is usually a secondary actor within use case diagrams, we established that time in our scenario triggers some use cases such as 'AutoBackup' and 'AutoRestore' and therefore cannot be left as a secondary actor, as a primary actor triggers use cases, which is exactly what time does in our scenario.

DATABASE-
Our database is our only secondary actor within our system, as Mr.Lancaster explicitly states in his requirements brief, that he wants specific jobs to found and accessed and listed. In order to do this, individual pieces of data must be stored. One example of this is how we have modelled 'CreateJob' with an association to Database, as once a job is created, its information will be stored within our database.

Similarly, monitoring payment records such as the payment amount and the chosen payment type should be stored in the system, so staff know how customers will be paying for future record and existing payments made by the customer. In this example, monitoring payment may be useful for Office Managers to determine whether a Customer is worthy of being a 'Valued Customer', based on if customers have payed previous payments on time with the correct amount given.

Packages

Packages have been used to simplify our diagram and show the layout of the BAPERS system in a more readable format.
Jobs/Tasks-
The Jobs/Tasks package has been used to show how the specific jobs and tasks are created within BAPERS. This includes creating a job, inspecting jobs, accepting urgent jobs,and setting the progression or completion of specific Tasks within each job.

Account-
The account package is used to highlight tasks that are completed in regards to an account. Setting a discount plan for valued customer directly affects a customer account, and upgrading/downgrading/reactivating a customer account is included in this. Logging into the BAPERS system requires an account for the staff members too.

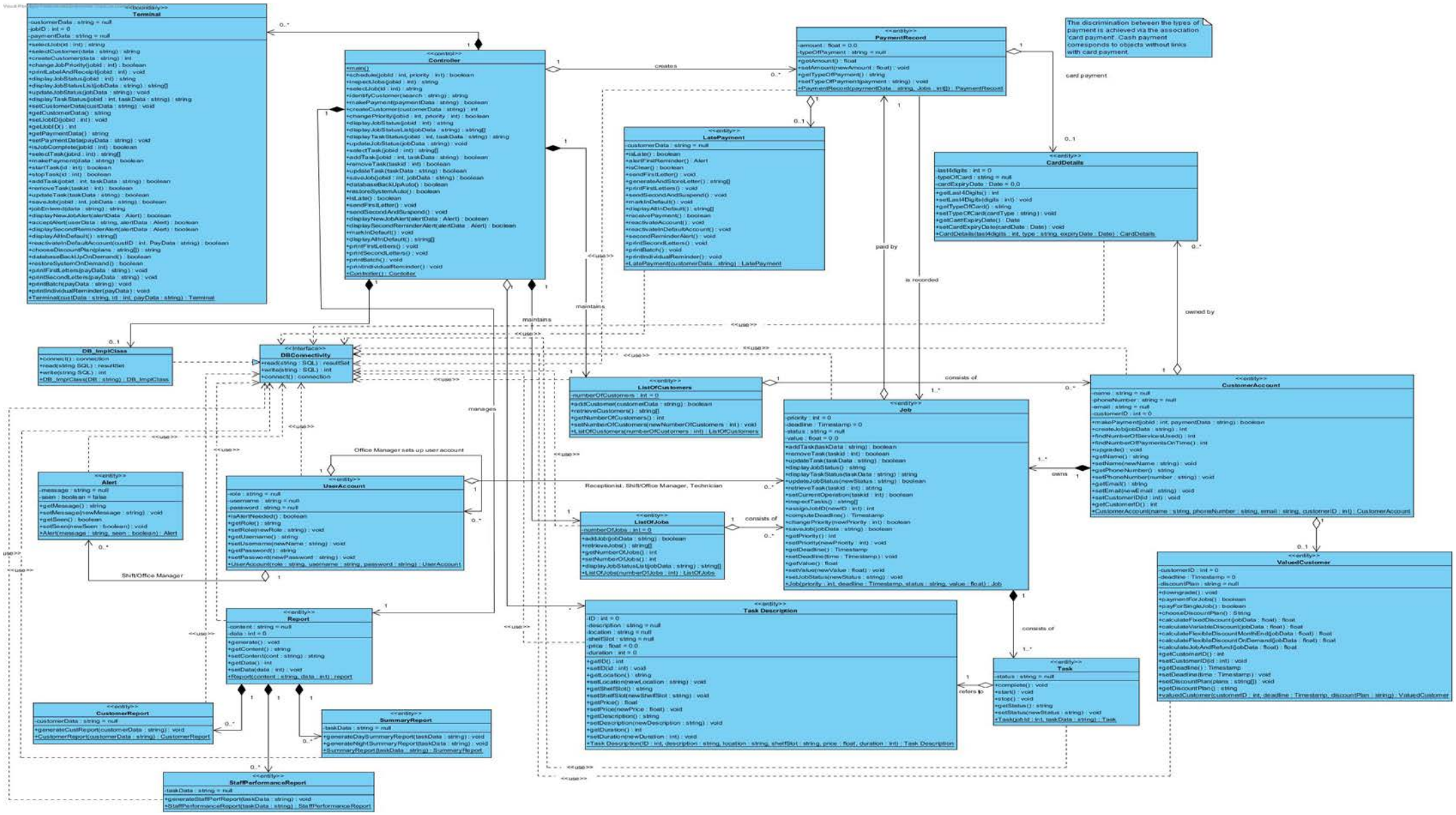
Payment-
The payment package is exclusively used to monitor the payment records within BAPERS by customers. It highlights the amount of money payed for the job and what type of payment has been used(whether it be card payments or cash in hand payments)

Admin-
The admin package is used to handle miscellaneous tasks such as generating reports about staff, or summary reports, or even individual reports. It allows general administrative duties to be accomplished such as backing up the the database or restoring the database. The admin package is also used to monitor alerts and reminders, such as generating first reminders, and eventually terminating an account if a customer has not payed. Although suspending an account may have to do with the 'Account' package, the alerts generated occur in the 'admin' package and so we feel it fits better here.

4. System Design

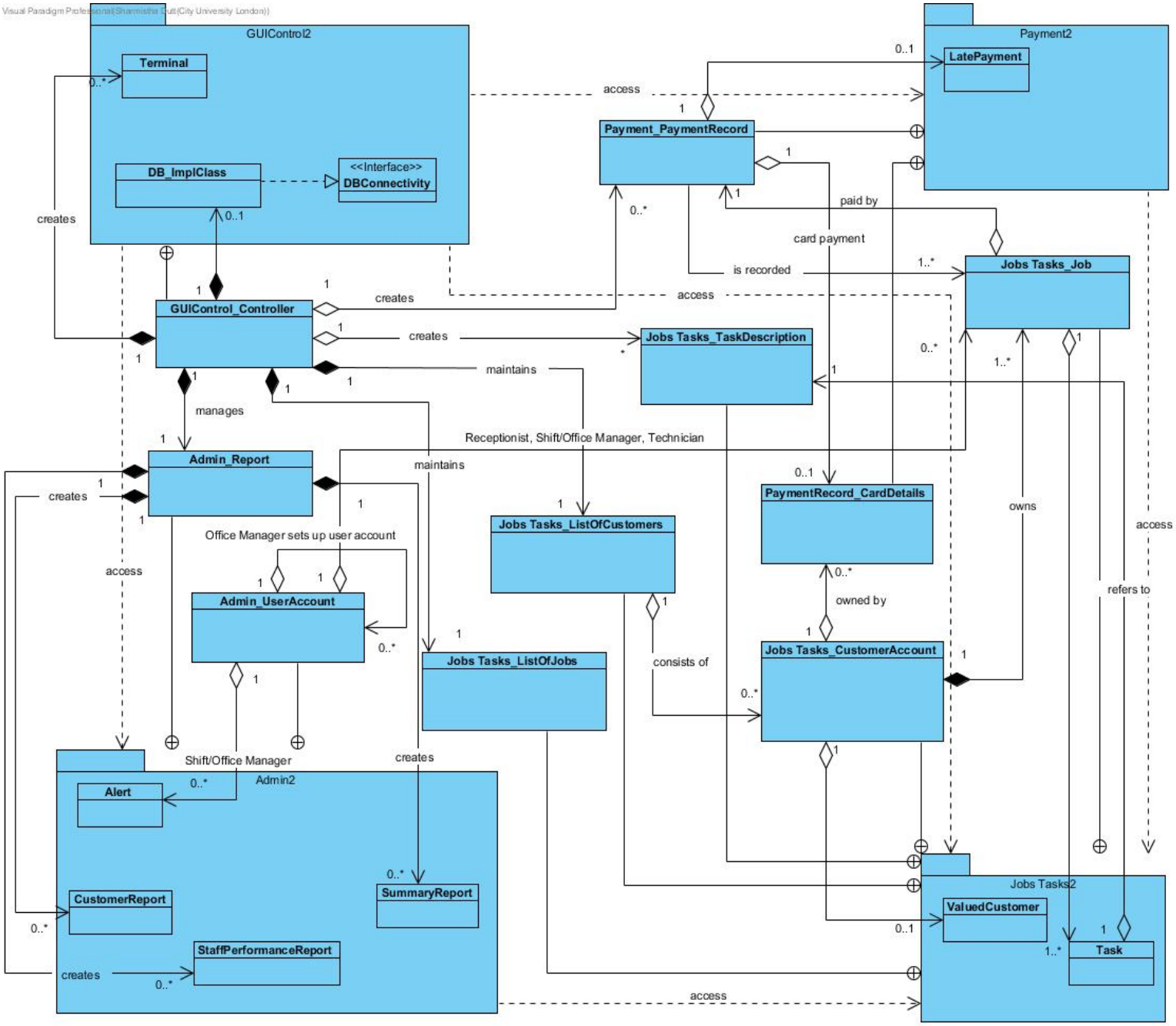
4.1 Design Class Diagram

The diagram is displayed on the page below

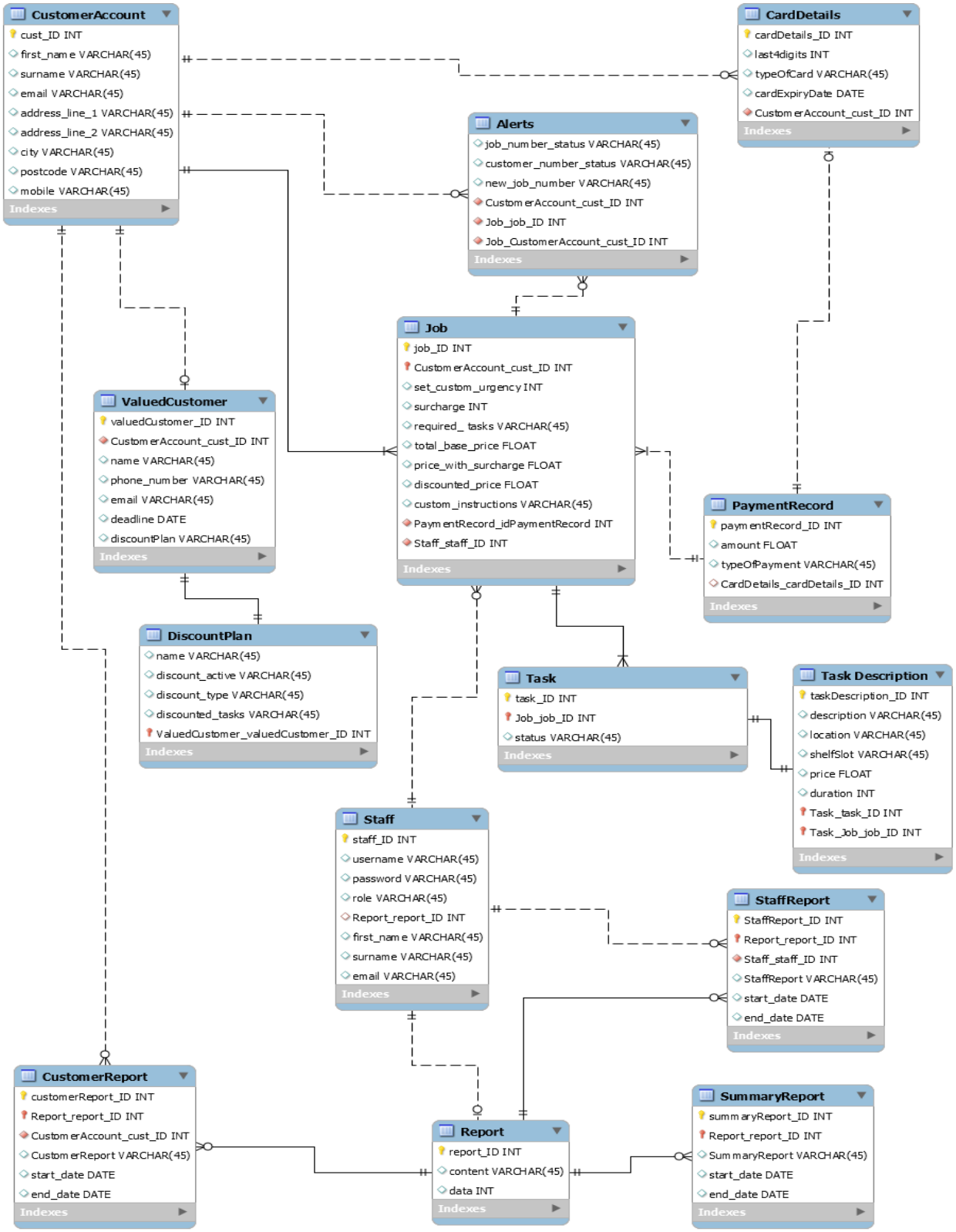

Terminal
Controller
PaymentRecord
LatePayment
CardDetails
DB_ImplClass
DBConnectivity
ListOfCustomers
CustomerAccount
Alert
UserAccount
ListOfJobs
Job
Task Description
Task
Report
CustomerReport
SummaryReport
StaffPerformanceReport
ValuedCustomer
The discrimination between the types of payment is achieved via the association 'card payment'. Cash payment corresponds to objects without links with card payment.
creates
card payment
paid by
is recorded
owned by
maintains
consists of
manages
owns
refers to
Office Manager sets up user account
Reception ist, Shift/Office Manager, Technician
Shift/Office Manager

4.2 Package Class Diagram

A package diagram has been used here to make the Design Class Diagram more readable.



4.3 ER Diagram



4.4 SQL Statements

DDL STATEMENTS AND DML STATEMENTS

CREATE TABLE statements

Discount Plan

```
CREATE TABLE `bapers2`.`discountplan` (
  `Customer_ID` INT NOT NULL,
  `Customer name` VARCHAR(45) NULL,
  `Discount Active` VARCHAR(45) NULL,
  `Discount Type` VARCHAR(45) NULL,
  `Discount Task` VARCHAR(45) NULL,
  PRIMARY KEY (`Customer_ID`));
```

Job

```
CREATE TABLE `bapers2`.`job` (
  `CustomerID` INT NOT NULL,
  `Set Duration` VARCHAR(45) NULL,
  `Time Taken` VARCHAR(45) NULL,
  `Final Price` INT NULL,
  `Custom Instruction` VARCHAR(45) NULL,
  `Material Instruction` VARCHAR(45) NULL,
  `Job Status` VARCHAR(45) NULL,
  `Payment Status` VARCHAR(45) NULL,
  PRIMARY KEY (`CustomerID`));
```

Customer Report

```
CREATE TABLE `bapers2`.`customer_report` (
  `CustomerReport_ID` INT NOT NULL,
  `CustomerReport` VARCHAR(45) NULL,
  `start_date` DATE NULL,
  `end_date` DATE NULL,
  `Report_ID` INT NULL,
  `Customer_ID` VARCHAR(45) NULL,
  PRIMARY KEY (`CustomerReport_ID`),
  UNIQUE INDEX `Report_ID_UNIQUE` (`Report_ID` ASC),
  UNIQUE INDEX `Customer_ID_UNIQUE` (`Customer_ID` ASC));
```

Staff Report

```
CREATE TABLE `bapers2`.`staff_report` (
  `Code` INT NULL,
  `TaskID` VARCHAR(45) NOT NULL,
  `Department` VARCHAR(45) NULL,
  `Date` DATE NULL,
  `Start Time` VARCHAR(45) NULL,
  `Time Taken` VARCHAR(45) NULL,
  `Total` VARCHAR(45) NULL,
```

```
  PRIMARY KEY (`TaskID`));
```

Summary Report

```
CREATE TABLE `bapers2`.`summary_report` (
  `Date` DATE NOT NULL,
  `Copy Room` VARCHAR(45) NULL,
  `Development` VARCHAR(45) NULL,
  `Packaging` VARCHAR(45) NULL,
  PRIMARY KEY (`Date`));
```

Alerts

```
CREATE TABLE `bapers2`.`alerts` (
  `job_number_status` VARCHAR(45) NOT NULL,
  `customer_number_status` VARCHAR(45) NULL,
  `new_job_number` VARCHAR(45) NULL,
  PRIMARY KEY (`job_number_status`));
```

Customer Account

```
CREATE TABLE `bapers2`.`customer_account` (
  `Customer_ID` INT NOT NULL,
  `First Name` VARCHAR(45) NULL,
  `Surname` VARCHAR(45) NULL,
  `Email address` VARCHAR(45) NULL,
  `Address 1` VARCHAR(45) NULL,
  `Address 2` VARCHAR(45) NULL,
  `City` VARCHAR(45) NULL,
  `Post code` VARCHAR(45) NULL,
  `Mobile` VARCHAR(45) NULL,
  `Status` VARCHAR(45) NULL,
  PRIMARY KEY (`Customer_ID`));
```

Card details

```
    CREATE TABLE `bapers2`.`carddetails` (
      `CardDetail_ID` INT NOT NULL,
      `Last4Digits` INT NULL,
      `TypeOfCard` VARCHAR(45) NULL,
      `CardExpiryDate` DATE NULL,
      `Customer_ID` VARCHAR(45) NULL,
      PRIMARY KEY (`CardDetail_ID`));
```

Valued Customer

```
CREATE TABLE `bapers2`.`valuedcustomer` (
  `Name` VARCHAR(45) NULL,
  `PhoneNumber` VARCHAR(45) NULL,
  `Email` VARCHAR(45) NULL,
  `Deadline` DATE NULL,
  `DiscountPlan` VARCHAR(45) NULL,
  `Customer_ID` INT NULL,
  UNIQUE INDEX `Customer_ID_UNIQUE` (`Customer_ID` ASC));
```

Payment Record

```
CREATE TABLE `bapers2`.`paymentrecord` (
  `PaymentRecord_ID` INT NOT NULL,
  `Amount` FLOAT NULL,
  `TypeOfPayment` VARCHAR(45) NULL,
  `CardDetails` INT NULL,
  PRIMARY KEY (`PaymentRecord_ID`),
  UNIQUE INDEX `CardDetails_UNIQUE` (`CardDetails` ASC));
```

Task

```
CREATE TABLE `bapers2`.`task` (
  `Task_ID` INT NOT NULL,
  `Job` INT NULL,
  `Status` VARCHAR(45) NULL,
  PRIMARY KEY (`Task_ID`));
```

Task description

```
CREATE TABLE `bapers2`.`taskdescription` (
  `TaskDescription_ID` INT NOT NULL,
  `Location` VARCHAR(45) NULL,
  `Price` INT NULL,
  `Description` VARCHAR(45) NULL,
  `Duration` INT NULL,
  `Task` INT NULL,
  `Job` INT NULL,
  PRIMARY KEY (`TaskDescription_ID`));
```

Staff

```
CREATE TABLE `bapers2`.`staff` (
  `Staff_ID` INT NOT NULL,
  `Username` VARCHAR(45) NULL,
  `Password` VARCHAR(45) NULL,
  `Role` VARCHAR(45) NULL,
  `Report_ID` VARCHAR(45) NULL,
  `First name` VARCHAR(45) NULL,
  `Surname` VARCHAR(45) NULL,
  `Email` VARCHAR(45) NULL,
  PRIMARY KEY (`Staff_ID`));
```

Report

```
CREATE TABLE `bapers2`.`report` (
  `Report_ID` INT NOT NULL,
  `Content` VARCHAR(45) NULL,
  `Data` INT NULL,
  PRIMARY KEY (`Report_ID`));
```

INSERT statements

Insert Statements 1

```
UPDATE `bapers2`.`task` SET `Task_ID`='1', `Job`='Large Copy Camera' WHERE `Task_ID`='1';
INSERT INTO `bapers2`.`task` (`Task_ID`, `Job`) VALUES ('2', 'Black& White film processing ');
INSERT INTO `bapers2`.`task` (`Task_ID`, `Job`) VALUES ('3', 'Bag Up');
INSERT INTO `bapers2`.`task` (`Task_ID`, `Job`) VALUES ('4', 'Colour film processing ');
INSERT INTO `bapers2`.`task` (`Task_ID`, `Job`) VALUES ('5', 'Colour Transparency processing ');
INSERT INTO `bapers2`.`task` (`Task_ID`, `Job`) VALUES ('6', 'Small copy Camera');
INSERT INTO `bapers2`.`task` (`Task_ID`, `Job`) VALUES ('7', 'Mount Transparencies ');
```

Insert statements 2

```
INSERT INTO `bapers2`.`staff` (`staff_ID`, `Username`, `Password`,  `Role`, `Report_ID`, `First
name`, `Surname`, `Email`) VALUES ('1432', 'Joe', 'Cooper', 'Technician', '1', 'Joe',
'Cooper' ,'Joecooper@ orgrimmar.com');
INSERT INTO `bapers2`.`staff` (`staff_ID`, `Username`, `Password`,  `Role`, `Report_ID`, `First name`,
`Surname`, `Email`) VALUES ('3008', 'Katie ', 'Smith', 'Office Manager', '2','Katie ',
'Smith', Katiesmith@orgrimmar.com');
INSERT INTO `bapers2`.`staff`(`staff_ID`, `Username`, `Password`,  `Role`, `Report_ID`, `First name`,
`Surname`, `Email`) VALUES ('1988', 'Josh ', 'William', 'Shift Manager ', '3' , 'Josh ',
'William','Joshwilliam@orgrimmar.com');
INSERT INTO `bapers2`.`staff` (`staff_ID`, `Username`, `Password`,  `Role`, `Report_ID`, `First name`,
`Surname`, `Email`) VALUES ('3088', 'Tom', 'Ashford', 'Technician ', '4', 'Tom',
'Ashford','TomAshford@orgrimmar.com');
```

SELECT statements

Select Statement 1

```
SELECT Staff_ID,Username,Password,Role, Report_ID,First Name, Surname,Email FROM
Bapers2.Staff
```

Select Statement 2

```
SELECT Task_id,Job,Status from Bapers2.Task;
```

UPDATE statements

Update Statement 1

```
UPDATE Bapers2.Staff set role='Shift Manager' where Staff_ID=1432;
```

Update Statement 2

```
UPDATE Bapers2.Task set Job='Use of large copy camera' where Task_ID=1;
```

DELETE statements

Delete Statement 1

DELETE from Bapers2.Task WHERE Task_ID=7;

Delete Statement 2

DELETE from Bapers2.Staff WHERE Staff_ID=4;

Report Statements

Summary Report

SELECT SummaryReport (SummaryReport_ID,SummaryReport,Report_ID)
FROM Report INNER JOIN SummaryReport ON Report_ID=SummaryReport.SummaryReport_ID

Staff report

SELECT StaffReport (StaffReport_ID,StaffReport,Report_ID,Staff_ID)
FROM Report INNER JOIN StaffReport ON Report_ID = StaffReport.StaffReport_ID
FROM Staff INNER JOIN StaffReport ON Staff_ID =StaffReport.StaffReportID.

5. GUI

5.1 GUI Documentation

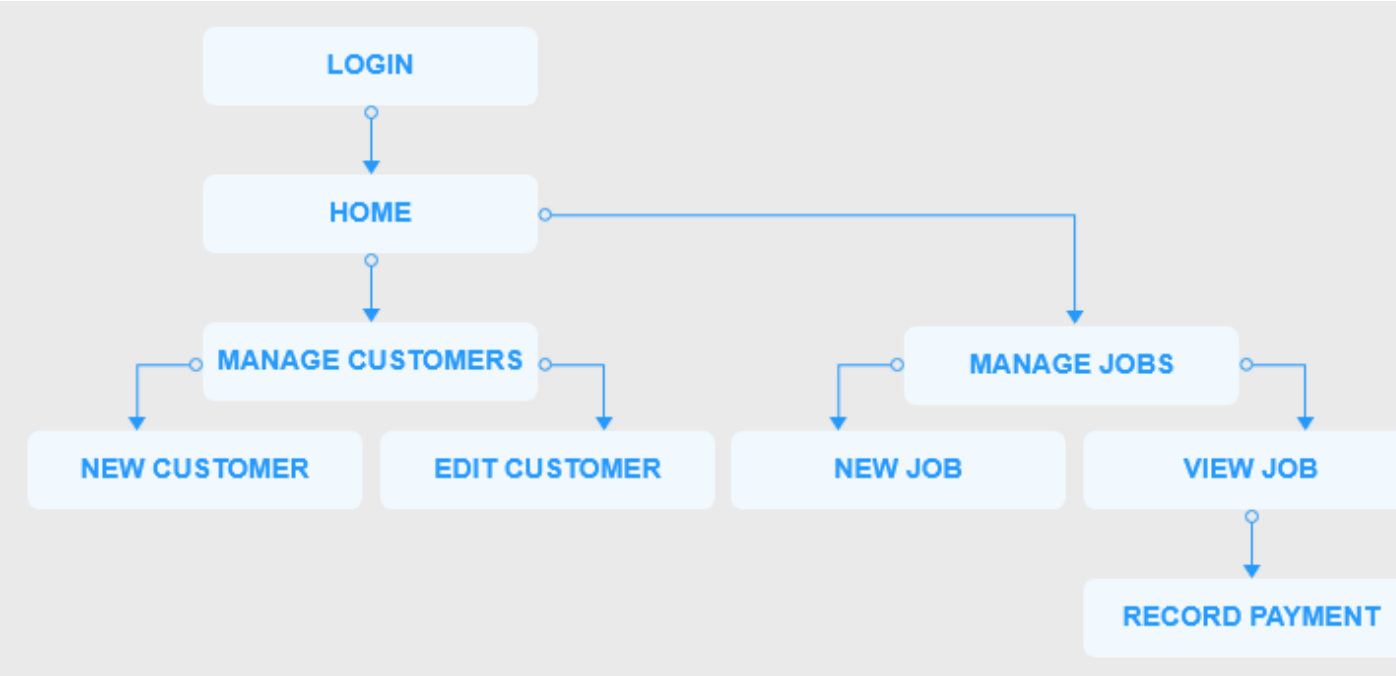
We have settled on a web-based method of implementation for this project; therefore, the GUI designs have been done as webpage wireframes (prototypes), showing the basic structure of the GUI on each web page. The designs were made in Adobe Experience Design (producing source files in .xd format), later being exported as PNG images to be showcased in this document.

The documentation for the GUI consists of three sections: a Sitemaps section which will give an overview of webpage navigation, a Page Navigation section which will showcase all individual page designs and describe in detail the navigation between the pages and what each respective UI element does, and the GUI and Design Classes section which will link the Design Class Diagram to the appropriate UI elements in the design.

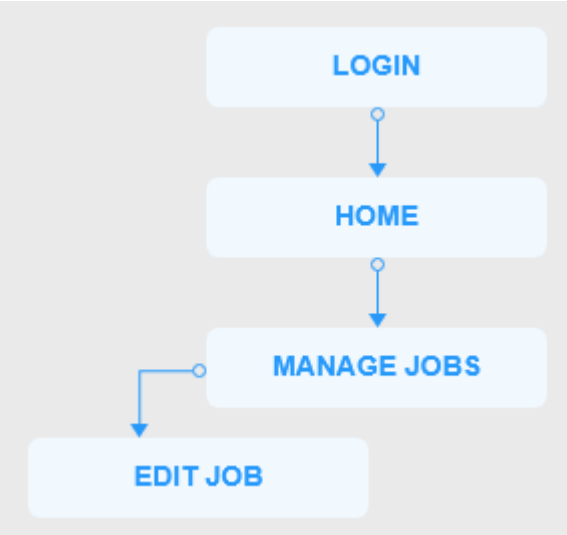
5.2 Sitemaps

This section will provide an overview of navigating the system for users with different access privileges, using sitemaps to show the hierarchy of the accessible GUI elements. There would be little point in re-designing each individual page in the GUI to show minimal differences based on user access levels; so the hierarchy will represent all pages accessible to each type of employee account in the system. Accounts that are not meant to access a particular element of the UI being demonstrated in the designs will simply not see the respective UI element in their version of the web pages.

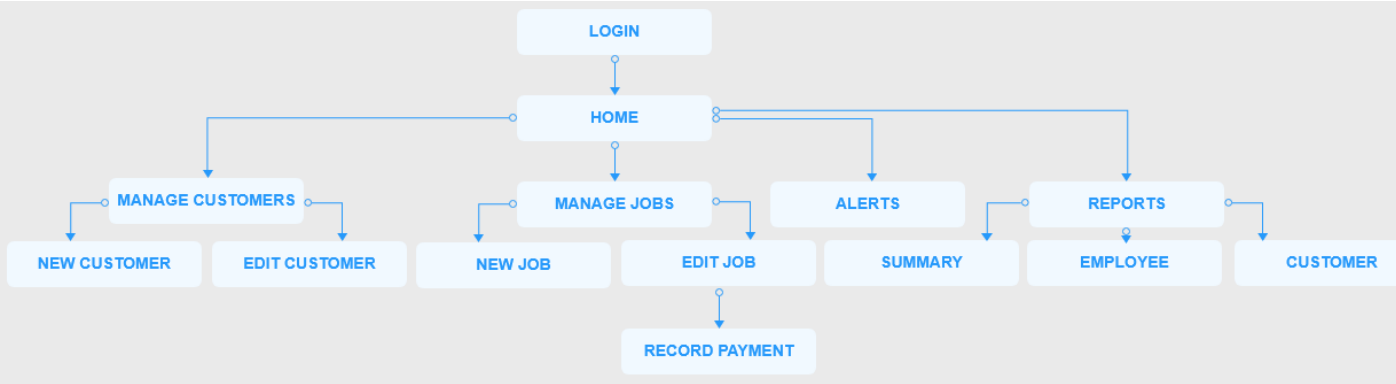
Receptionist sitemap



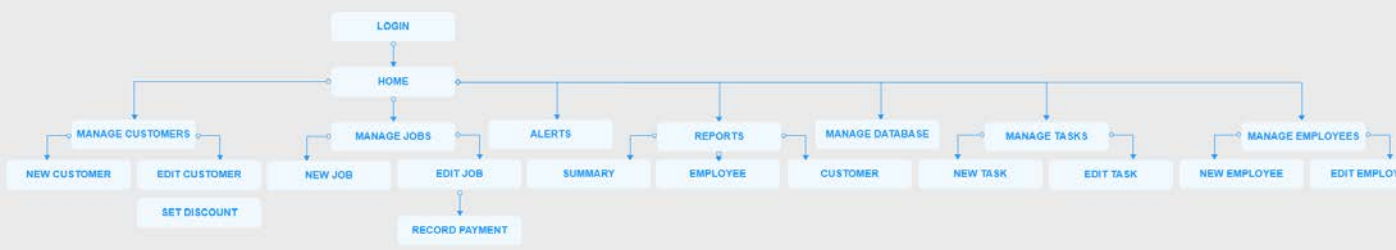
Technician sitemap



Shift manager sitemap



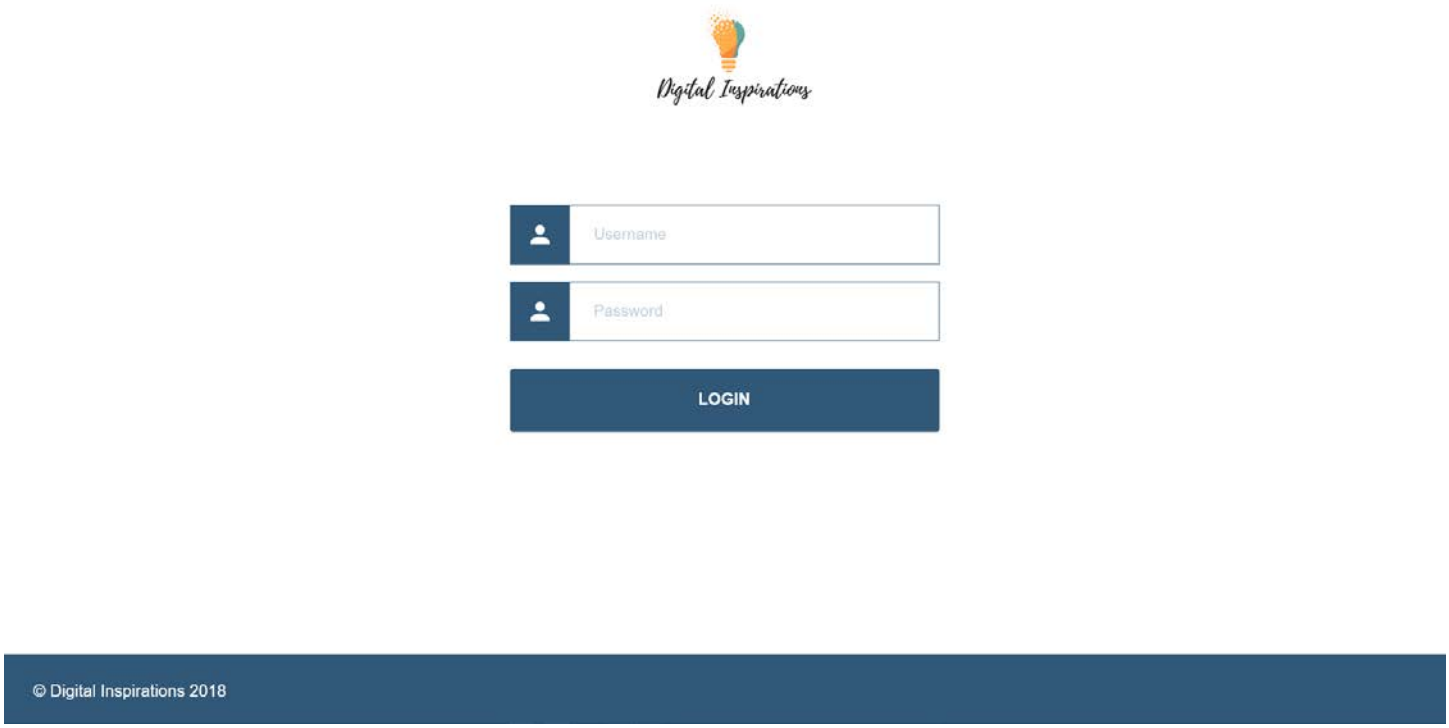
Office manager sitemap



5.3 Page Navigation

In this section I will describe in detail how a BAPERS user will navigate through the web-based GUI. I will refer to each interactive element in the UI and explain its function.
All pages are given **identification numbers**, so when I mention that element X on a page leads to page number 2.2.1, for example, it means that the webpage numbered respectively will open in the GUI window.

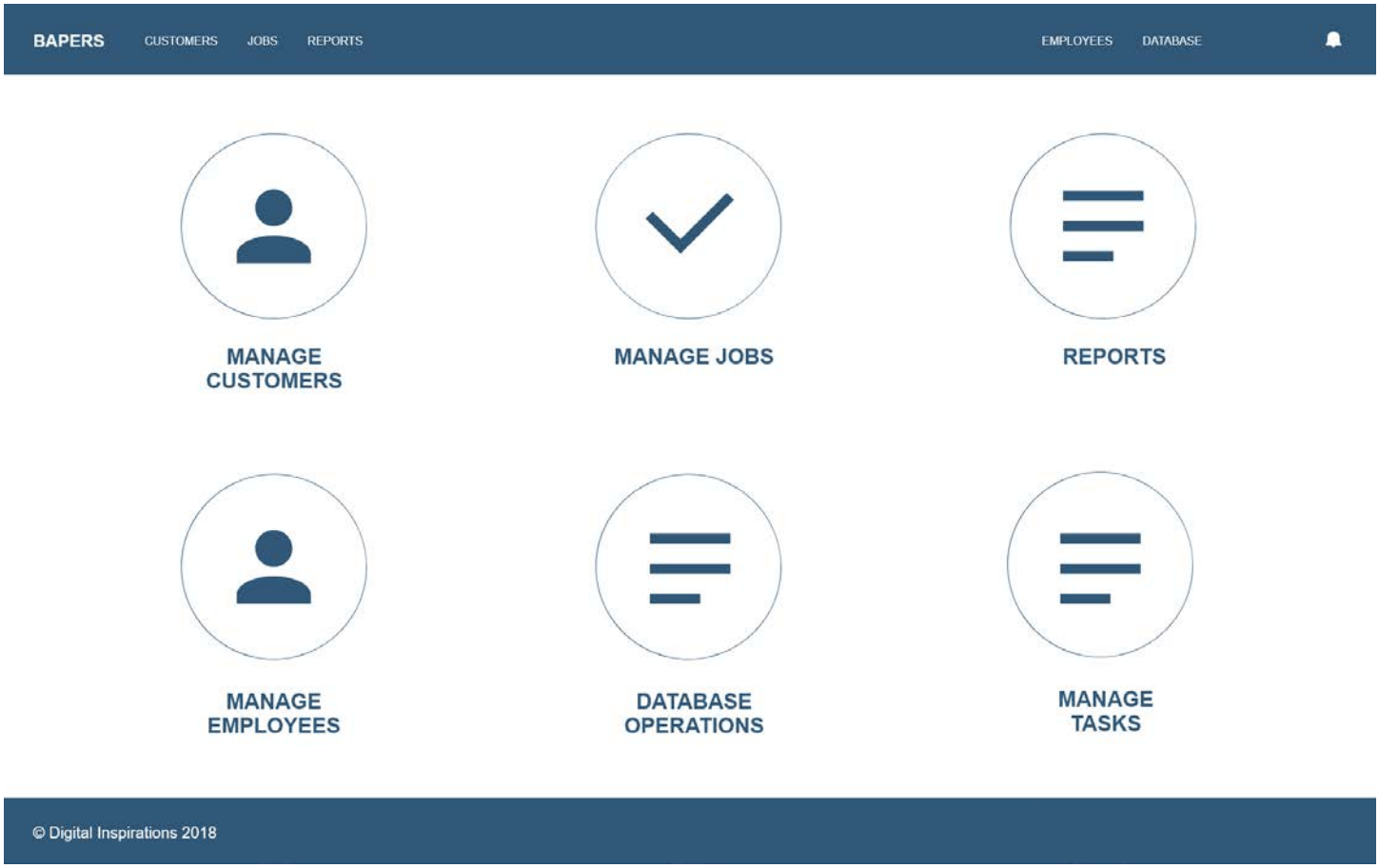
0. Login Page



The login page is what the user will see upon first interacting with a terminal connected to the system. There are only three interactive elements on this page:

- The **username** field, where the employee will enter their employee ID recorded in the system;
- The **password** field, where the employee will enter the password they use to log into the system;
- And the LOGIN button, which upon entering a matching set of credentials, will lead the user onto **page 1** – the **Home Page** for the system.

1. Home Page



From this page, all the primary system navigation and functionality will occur. It is worth noting that the look of the page will be different based on the role of the person logged on. For example, a receptionist will only have the ability to manage customers and manage jobs; a technician will only be able to manage jobs; while an office manager will have full access to all features of the system.

Some of the elements in the UI are also going to be common with the rest of the system. This includes the **navigation bar** at the top and the footer at the bottom. The footer has no functionality, however each of the items in the navigation bar will have the following interactions:

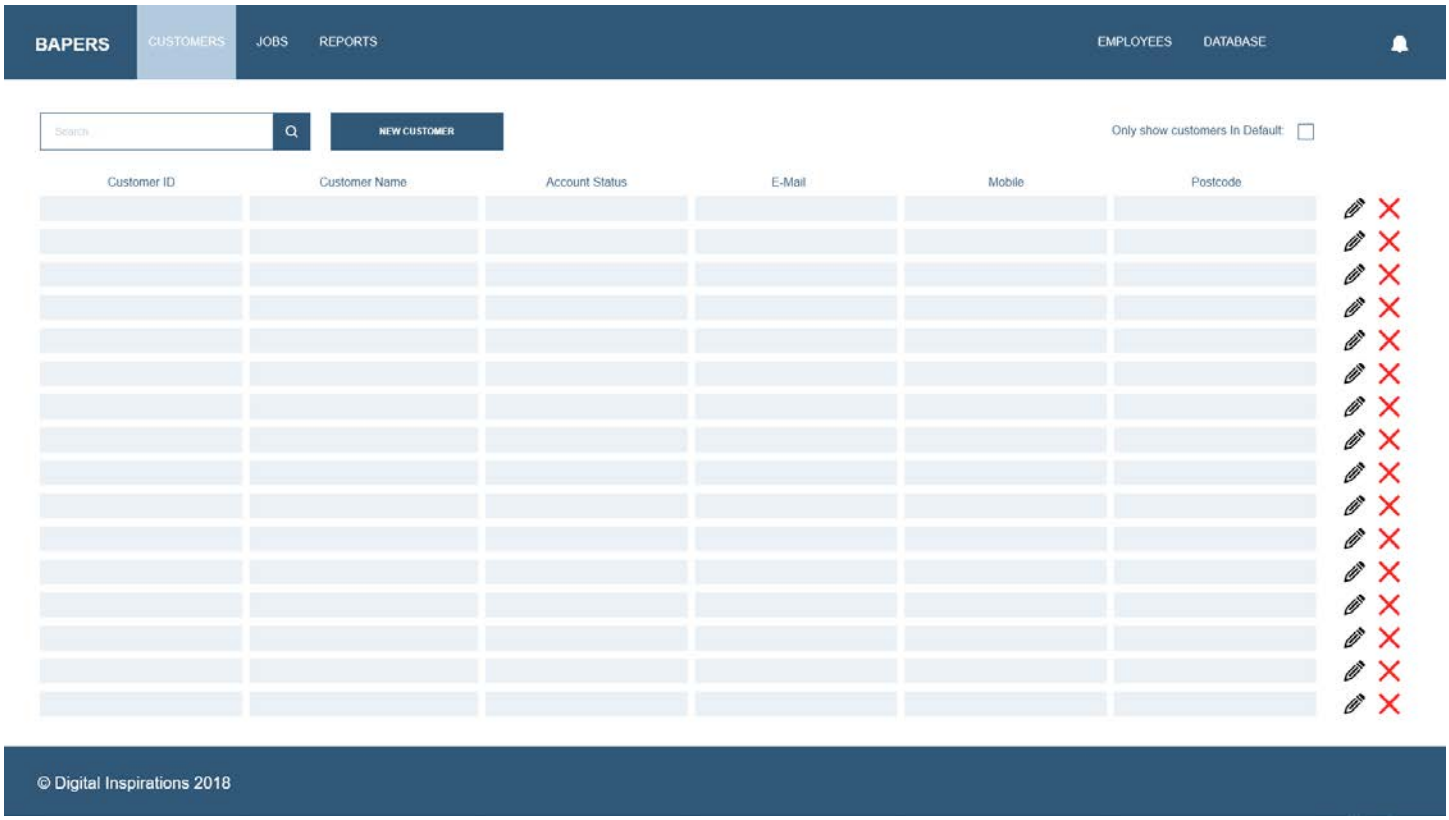
- Clicking on the BAPERS placeholder logo will lead the user back to the Home Page (i.e. the page being described right now) from any other page on the site.
- The CUSTOMERS menu item will lead the user to **page 2**, which allows to search for and manage customers in the system,
- The JOBS menu item will lead the user to **page 3**, allowing for an overview and management of currently active jobs.
- The REPORTS menu item (visible only to shift and office managers) will lead the user to **page 4**, allowing them to select and generate various reports about the system.
- The EMPLOYEES menu item (visible only to shift and office managers) will lead the user to **page 5**, where new employee accounts can be set up and existing ones can be edited.
- The DATABASE menu item (visible only to office managers) will lead to **page 7**, providing the functionality of backing up and restoring the database state.

- The **notification bell** located at the far right of the navigation bar is only visible to shift and office managers, and will lead to **page 8** upon clicking, where they can see important alerts about the state of the system such as the arrival of new jobs and late payment alerts.

The home page, like the navigation bar, provides access to the same respective elements in the system:

- MANAGE CUSTOMERS leads to **page 2**;
- MANAGE JOBS leads to **page 3**;
- REPORTS leads to **page 4**;
- MANAGE EMPLOYEES leads to **page 5**;
- DATABASE OPERATIONS leads to **page 7**;
- And MANAGE TASKS leads to **page 6**, allowing the system user to create new types of tasks to register within jobs, and to edit information about existing ones.

2. Customers – Manage/Search



This page provides the user with all functionality they need to search, edit, and add new customers.

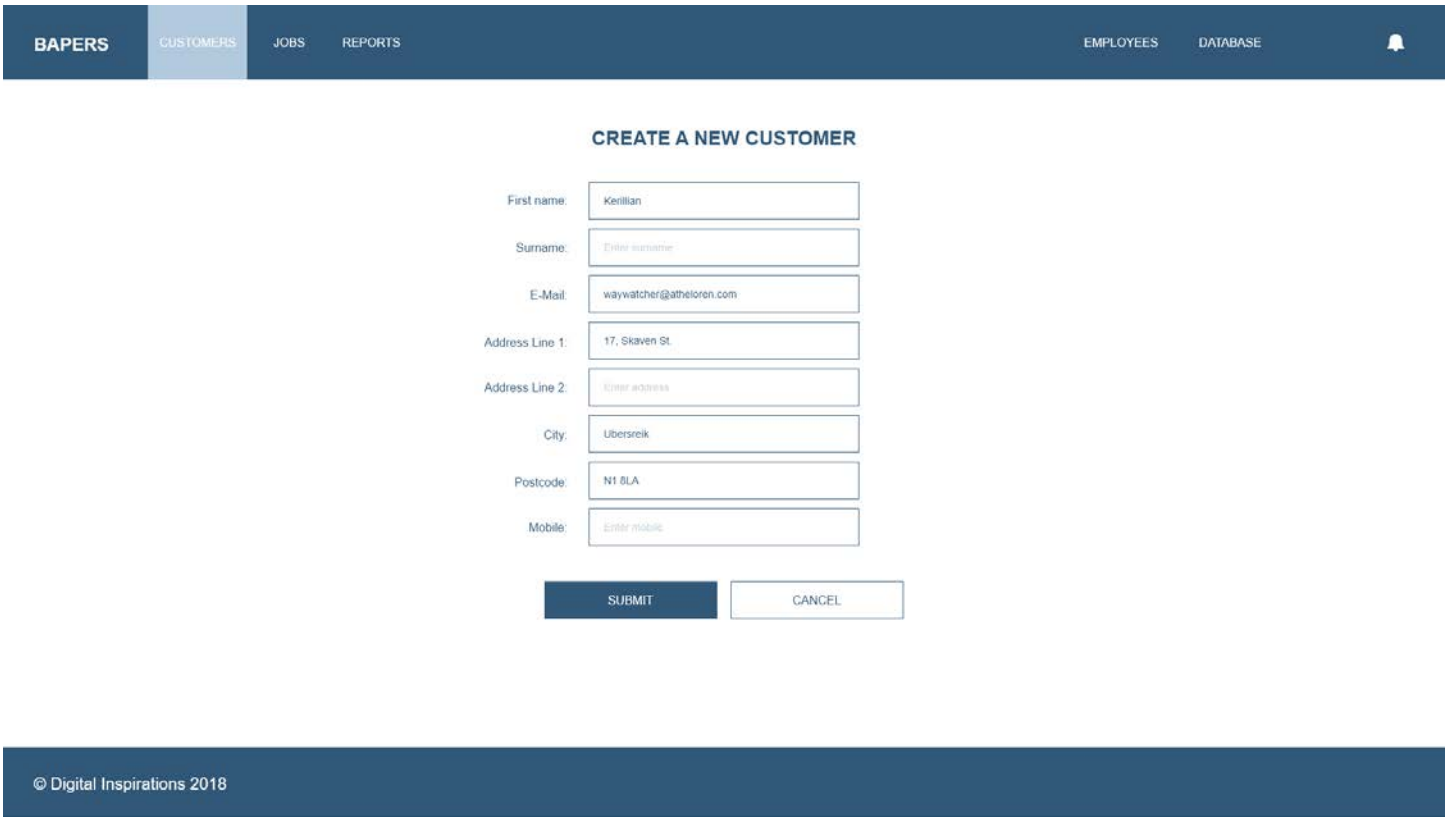
The **search field** at the top left of the page provides the ability to look up customers by using their ID, name, or e-mail.

To the right of the search field, the NEW CUSTOMER button lets the system user set up a new customer account, leading to **page 2.1**.

On the top right of the page, visible only to office managers, is the checkbox which will filter the list of customers to only show those who are **In Default**, so that legal action against them may be taken.

Search results for customers are formatted as a large table displaying all of their essential information. To the right of each table row are two buttons: the **crayon icon** allows an employee to edit the information of that particular customer leading to **page 2.2**, while the **cross icon** allows for deletion of customer accounts (with a default pop-up confirmation window).

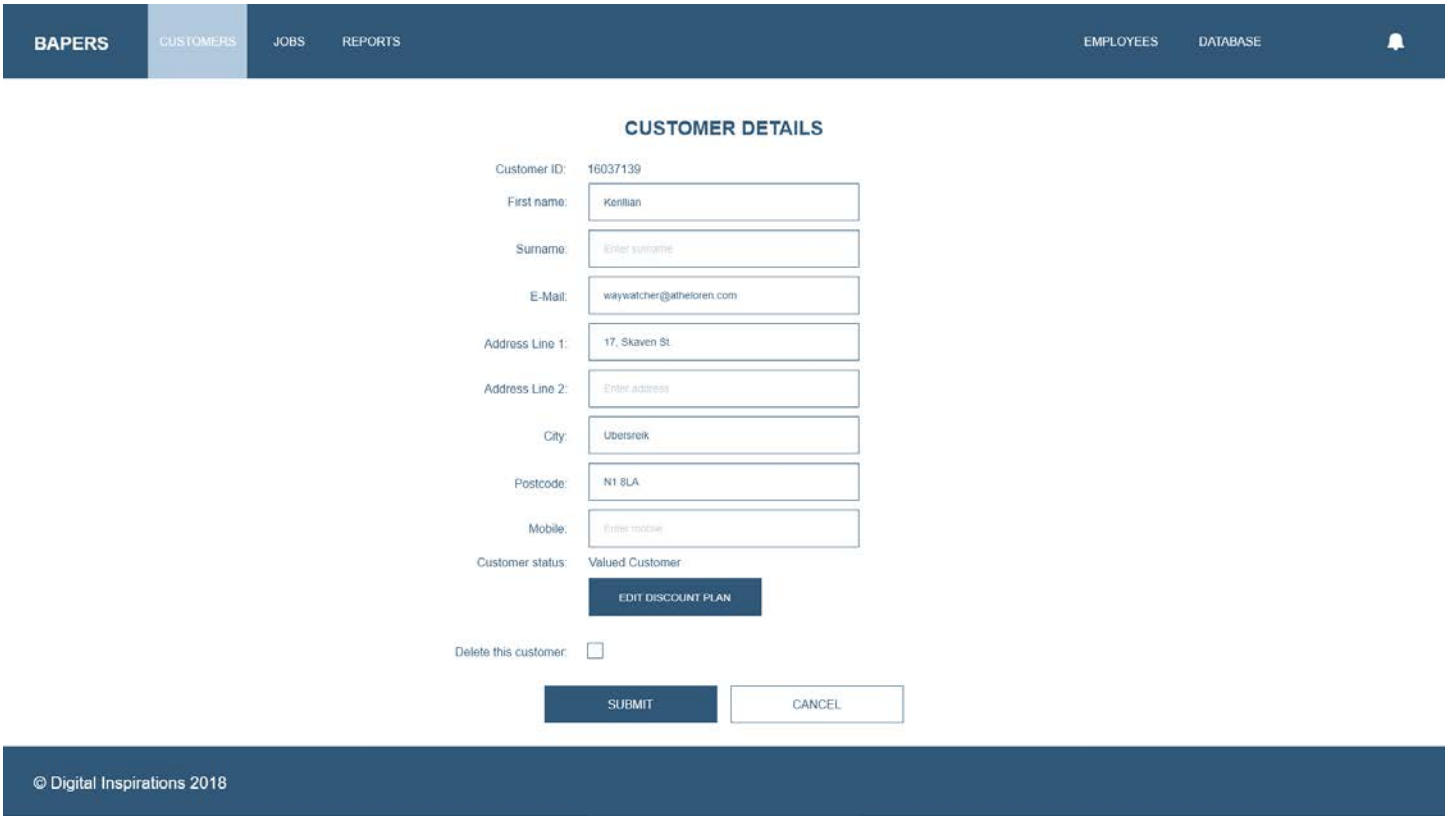
2.1. Customers – Create New



This page allows a user of the system to create a new customer account. The following details are required, entered into their respective data fields: customer's first name, their surname, e-mail, address and postcode; with address line 2 and mobile fields being optional rather than required.

If all required information is entered, when the employee clicks SUBMIT, the system will register the new customer, generating a new customer ID for them and taking the user back to **page 2**. The employee may also click CANCEL at any time to cancel the creation of a new customer account, also returning to **page 2** in the process.

2.2. Customers – Edit/View Customer Details



This page provides the user with the ability to edit an existing customer account, delete it, or, if the user is an office manager, to set up a discount plan for the customer, making them a valued customer in the process.

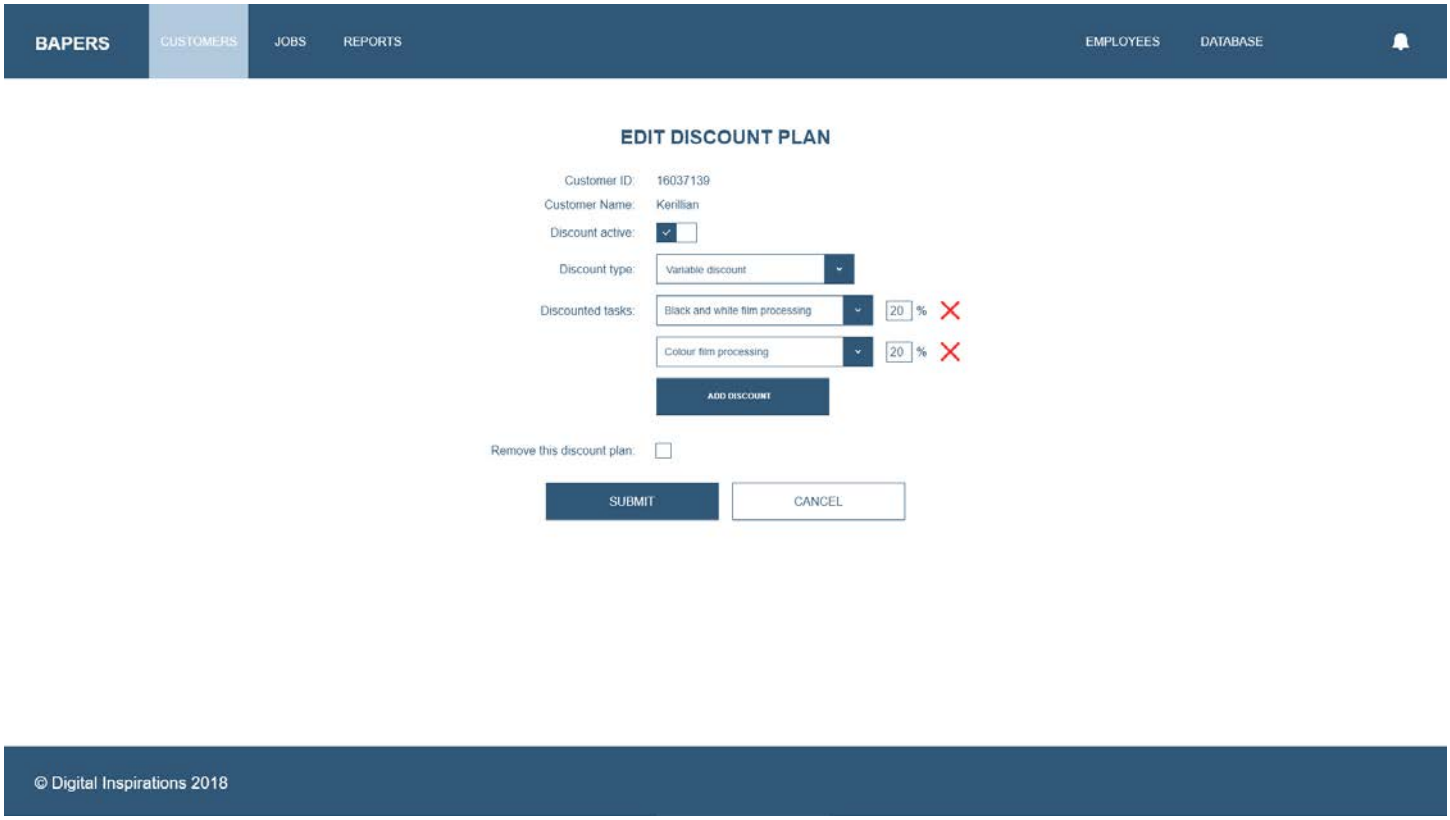
Any of the information about the customer can be edited, except for the ID assigned to them by the system. This way information is easy to change in case the user moves to a different location or starts using a different e-mail.

Only visible to office managers, the button EDIT DISCOUNT PLAN takes the system user to **page 2.2.1**, allowing them to either set up a new discount plan if one has not been made before, or to edit the existing discount plan. The information about the discount plan is saved on its own individual page and therefore to edit the discount plan there is no need to submit any other edits on the customer page.

Once the user clicks SUBMIT, if all form data is correct, the data about the customer is saved and the user returns to **page 2**. If the 'delete customer' checkbox is checked, the customer is removed from the system and the user also returns to **page 2**.

The user can cancel the editing process by clicking CANCEL and returning to **page 2**.

2.2.1. Customers – Discount Plan Setup



On this page, an office manager can set up a discount plan for a particular customer. They have the option to select the **type of the discount** between Fixed, Variable and Flexible discount.

A **fixed discount** will provide the system user with a single input field which takes a percentage value as the discount.

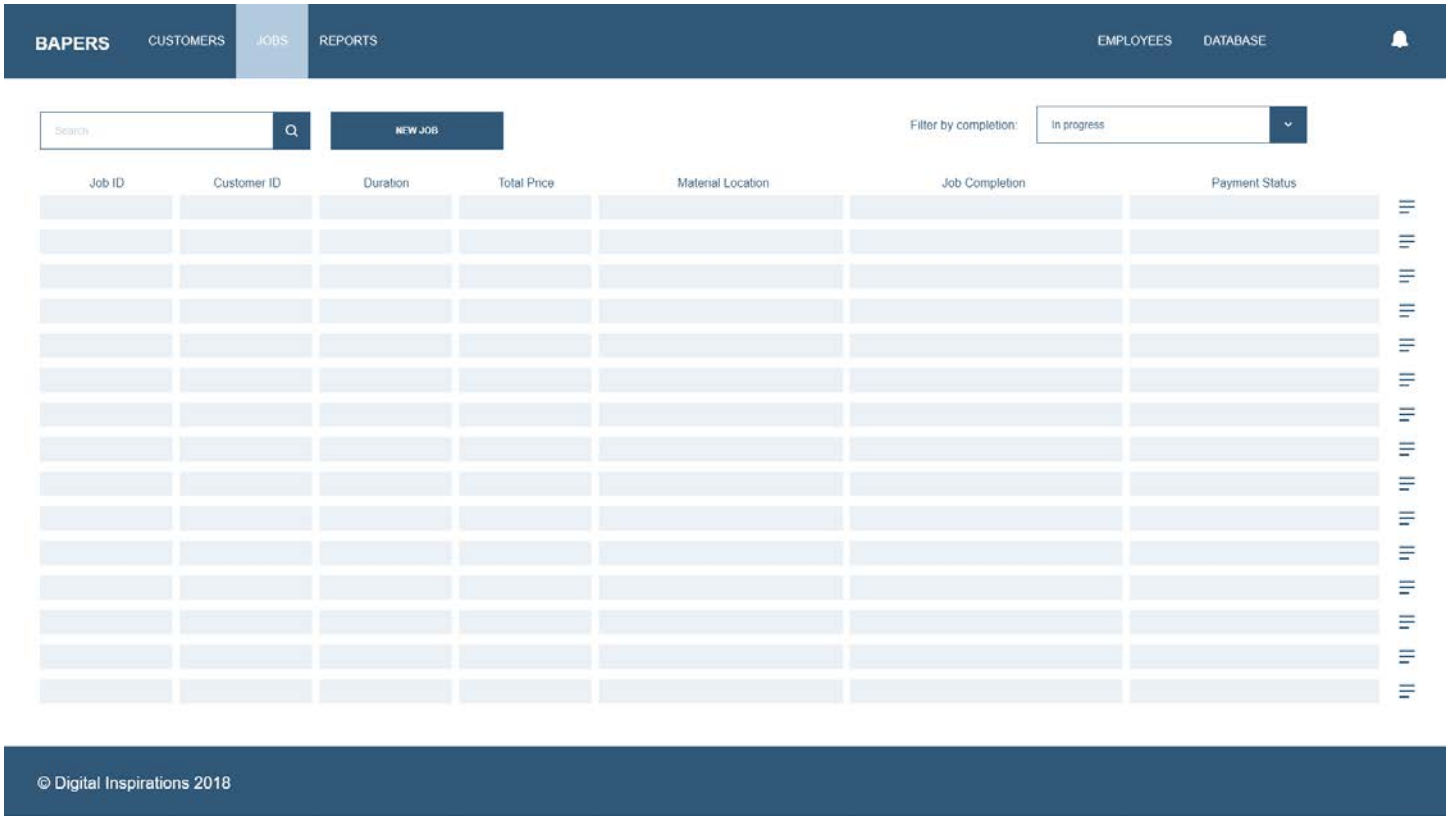
A **variable discount**, as showcased in the design, will provide the user with the ability to set individual discounts for any of the tasks in the system. The user first clocks on ADD DISCOUNT to add a new dropdown field, then in the **dropdown** they select the task they wish to set a discount on, and to the right of the dropdown they enter the percentage value of the discount. Further to the right with the **cross icon** the user can get rid of a discount for a particular task (for example if they made a new one accidentally).

Setting a **flexible discount** will allow the office manager to add a value band (or range) where they will input the floor and ceiling of the band and a respective discount percentage for it.

The office manager can choose to delete a discount plan by using the **checkbox** at any time and clicking SUBMIT, or they can disable it temporarily by using the discount active selector and also submitting the data.

Clicking on SUBMIT will save the discount plan information and take the user back to **page 2.2** with the information about the customer they have just edited. Clicking on CANCEL will take the user back to **page 2**.

3. Jobs – Manage/Overview/Search



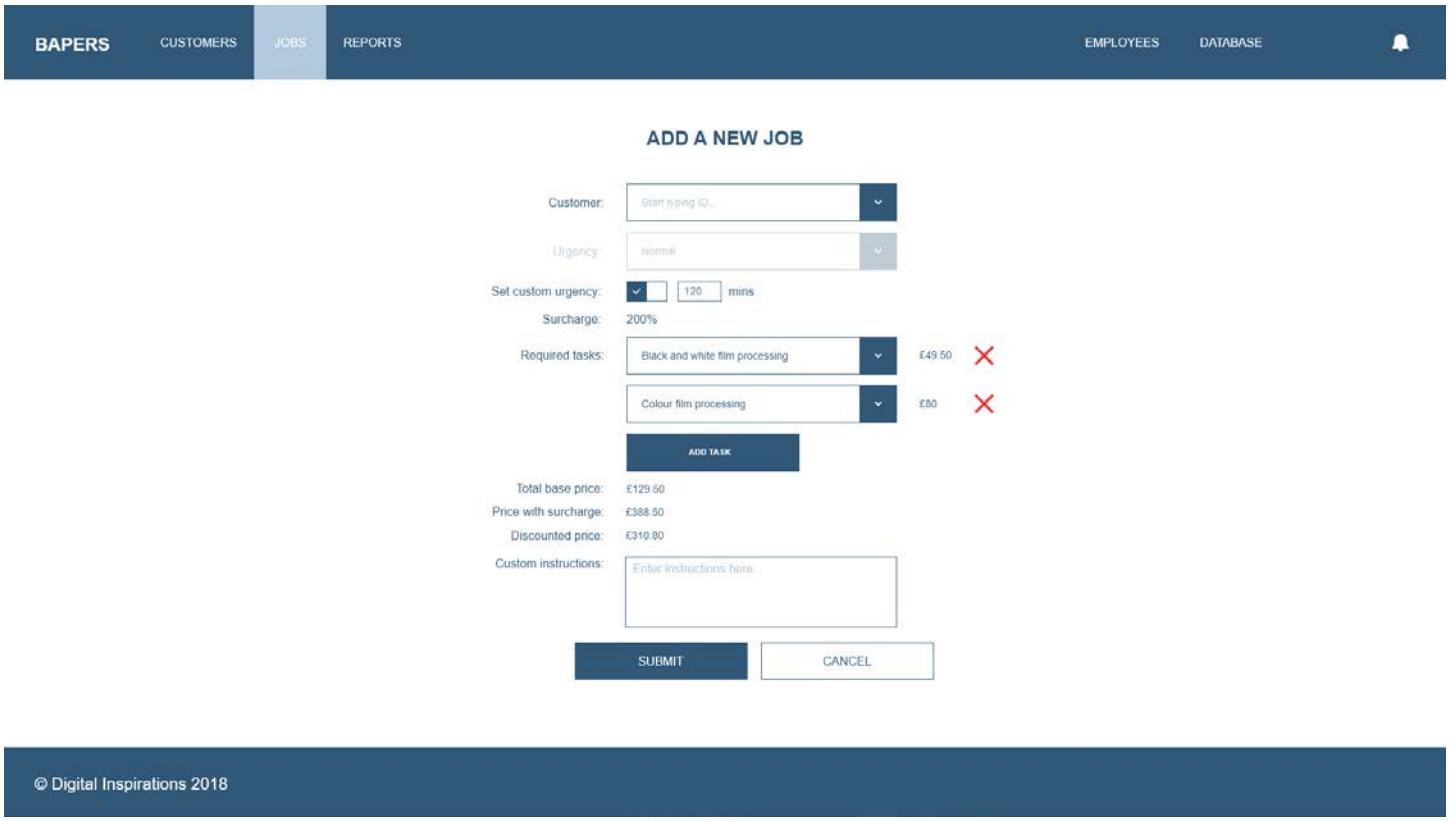
This page provides the user with an overview of jobs in the system.

The **search field** functions similarly to how it does on **page 2** (Customers), allowing to search for jobs by their descriptors (e.g. job and customer IDs, etc.) and the NEW JOB button takes the user to **page 3.1**.

Additionally, to the right is a dropdown box which allows the jobs to be **filtered by status**. The filters available are to show the jobs **In Progress**, to show **Complete** jobs, or to show **All** jobs regardless of status.

The jobs are displayed in a table format with all essential information about them. To the right of each table row is a **details icon** which upon clicking will take the user to the details page of a particular job (**page 3.2**), allowing them to mark tasks as completed or take payment for the job.

3.1. Jobs – Create New



Here, a user can register a new job in the system.

The job has to be tied to a particular customer, so the **first field** doubles as a search and a dropdown. When the user starts typing a customer ID into the field, suggested customers will pop up. The customer ID entered will be the one the job is registered for.

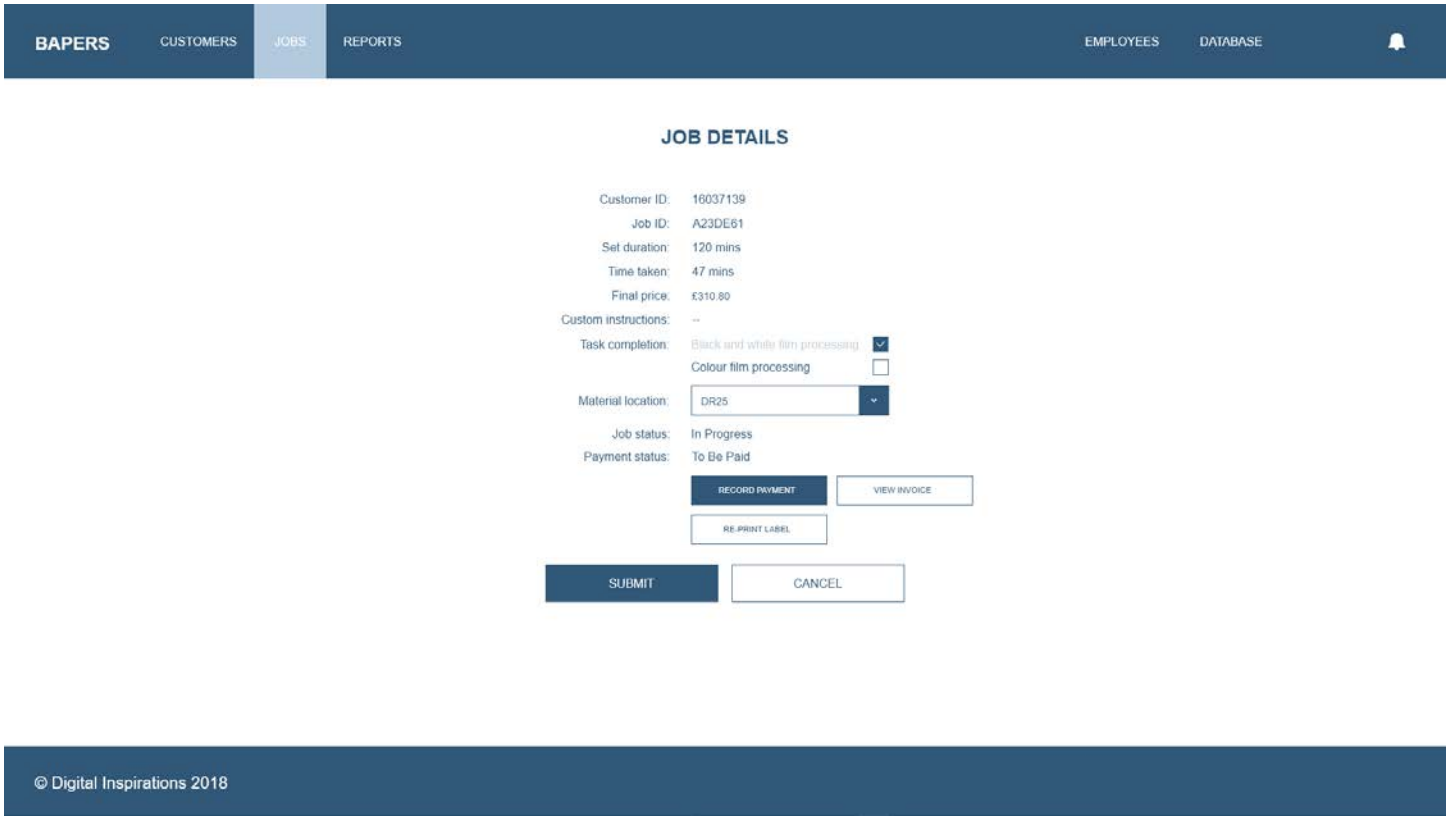
Below is a **dropdown box** (currently greyed out) which allows to select one of the default urgencies for the job. The urgencies available are **normal** (24 hour deadline) or **urgent** (6 hour deadline). Below, a there is a selector for **setting a custom urgency** for a job. The user of the system then specifies the deadline for the job in minutes, and a respective surcharge is calculated.

Below that, the user can click on ADD TASK to attach new tasks to the job. Tasks show up as **dropdown selectors** and their default price will be shown to the right. Further to the right, the user can delete a task they've added by mistake using the **cross icon**.

Below that, the system will show the total base price of the job, then add surcharge on top, and then apply a discount provided the user has a discount plan set up. In the field below, the user can enter **custom instructions** for the job in case they are provided.

Once done, clicking on SUBMIT will add the job to the system and return the user to **page 3**. Clicking on CANCEL will void any entered information and also return the user to **page 3**.

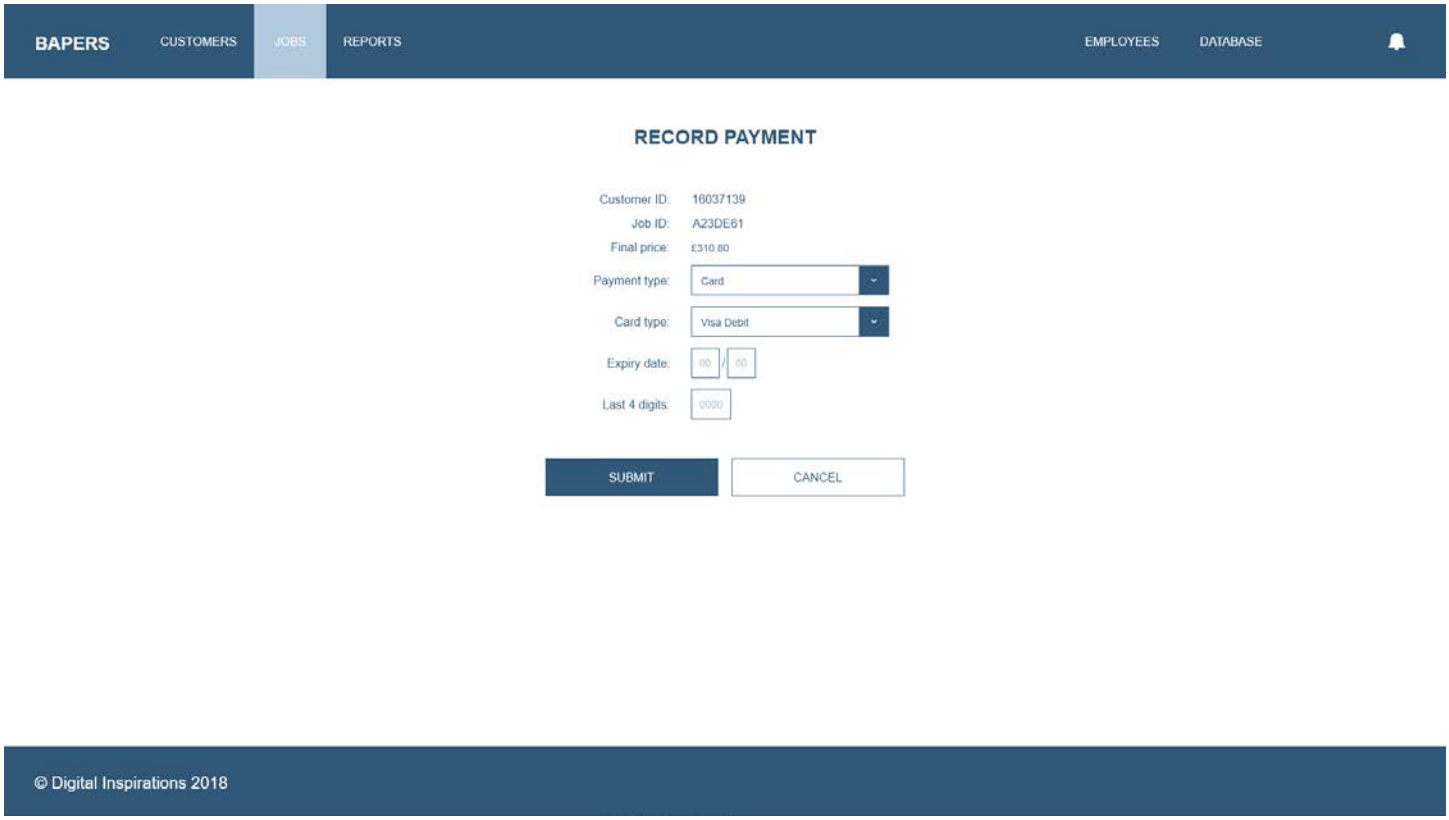
3.2. Jobs – View or Edit Job



This page displays information about a particular job in the system. It also provides employees such as technicians with the ability to mark the tasks from a job as completed using the **checkboxes** next to task completion and clicking SUBMIT; as well as to indicate a change in the location of the material for the job using a **dropdown list**.

Employees such as receptionists, on the other hand, will have the ability to RECORD PAYMENT for the job (**page 3.2.1**), VIEW INVOICE for it and print it (**3.2.2**), or to RE-PRINT LABEL for the material in case it gets lot or printing fails upon the original creation of the job.

3.2.1. Jobs – Record Payment



Here an employee can record a successful payment for a job. Since BAPERS is not connected to an external payment system, and it only receives cash payments or direct bank transfers/card payments, it is assumed that a payment has to be recorded in the system manually.

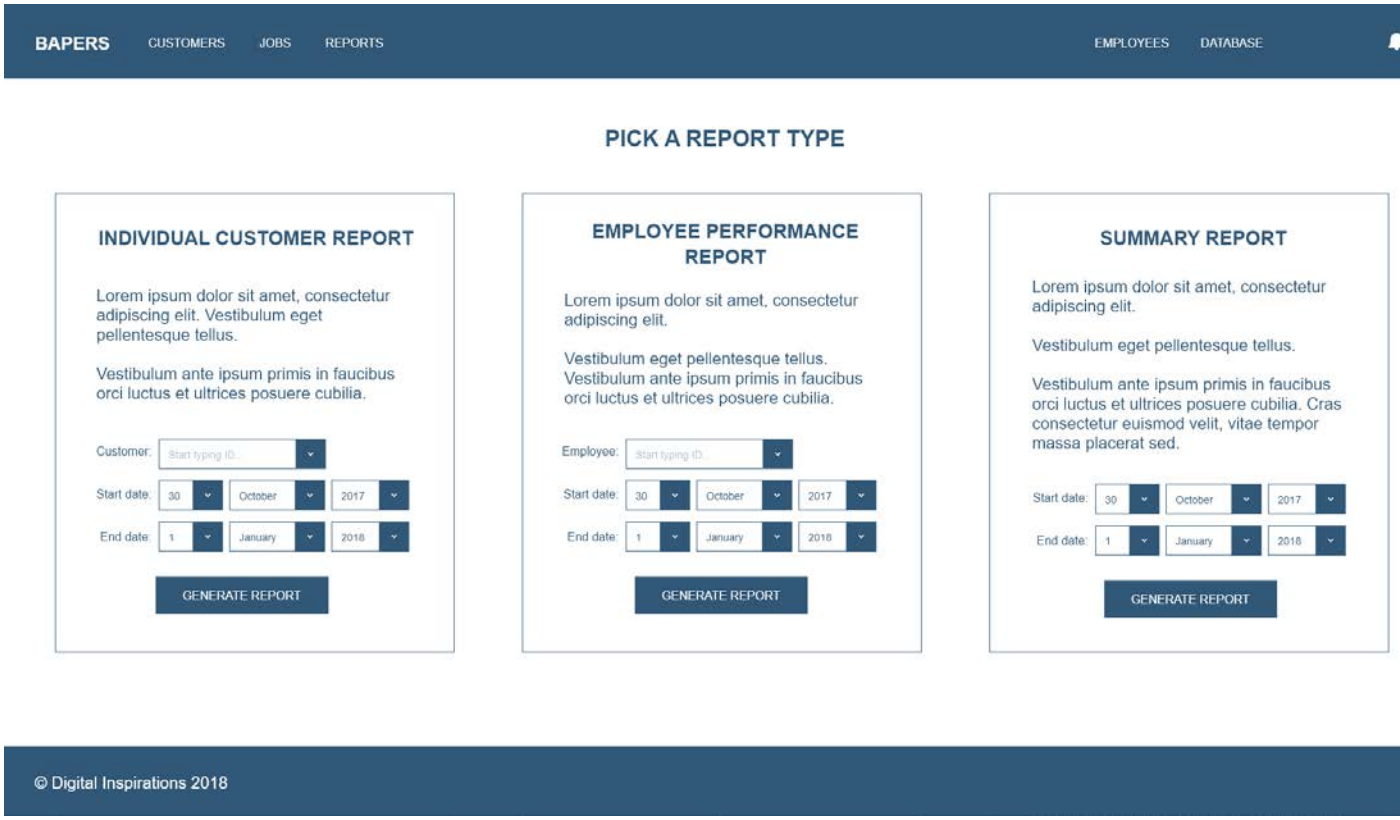
The page automatically displays the customer ID, job ID, and total price for the job, and provides the user with the opportunity to choose the **payment type** (either cash or card). If cash is selected, no further information is needed and the user can click SUBMIT to return to the previous page and register the job as paid for. If card is selected, the user also needs to enter the **type of card**, its **expiry date**, and the **last 4 digits** on it, all of which are later stored in the system.

Clicking CANCEL at any time will void all entered information and return the user to the previous page.

3.2.2. Jobs – Generate Invoice

The only functionality on this page (which mainly serves as formatting for auto-generated invoices) is to PRINT INVOICE as displayed or to return the user BACK to the previous page.

4. Reports – Report Selection and generation page

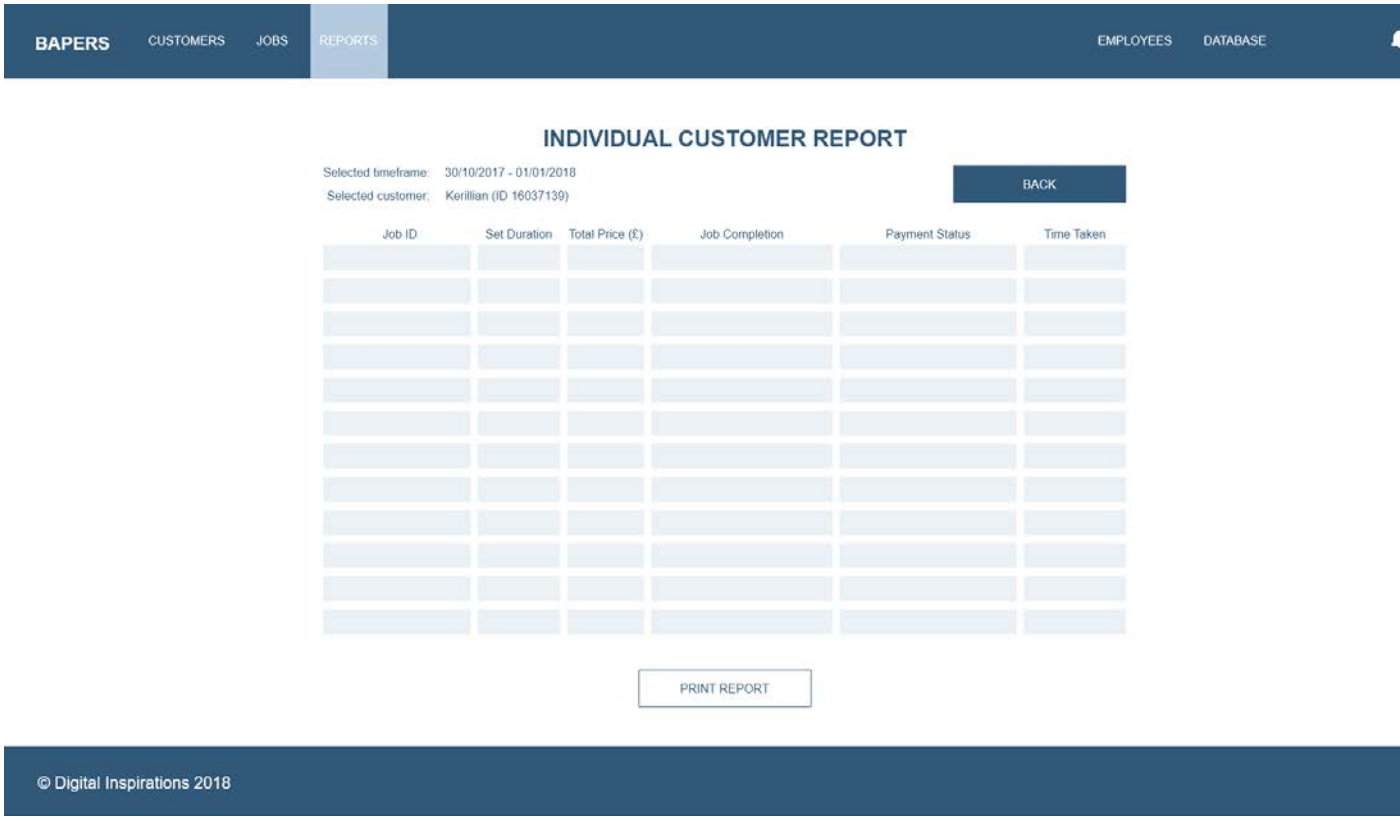


This page allows an office or shift manager to generate reports about the performance within the system.

For each of the reports, the user can set a **timeframe** in the form of a **start and end date** consisting of a **day**, **month** and **year** selectable through **dropdowns**.

For the customer and employee reports, a respective **ID** also has to be typed in, after which the user can click on GENERATE REPORT which, depending on the type of report chosen will take the user to either **page 4.1** for the **customer report**, **page 4.2** for the **employee report**, or **page 4.3** for the **summary report** about the system.

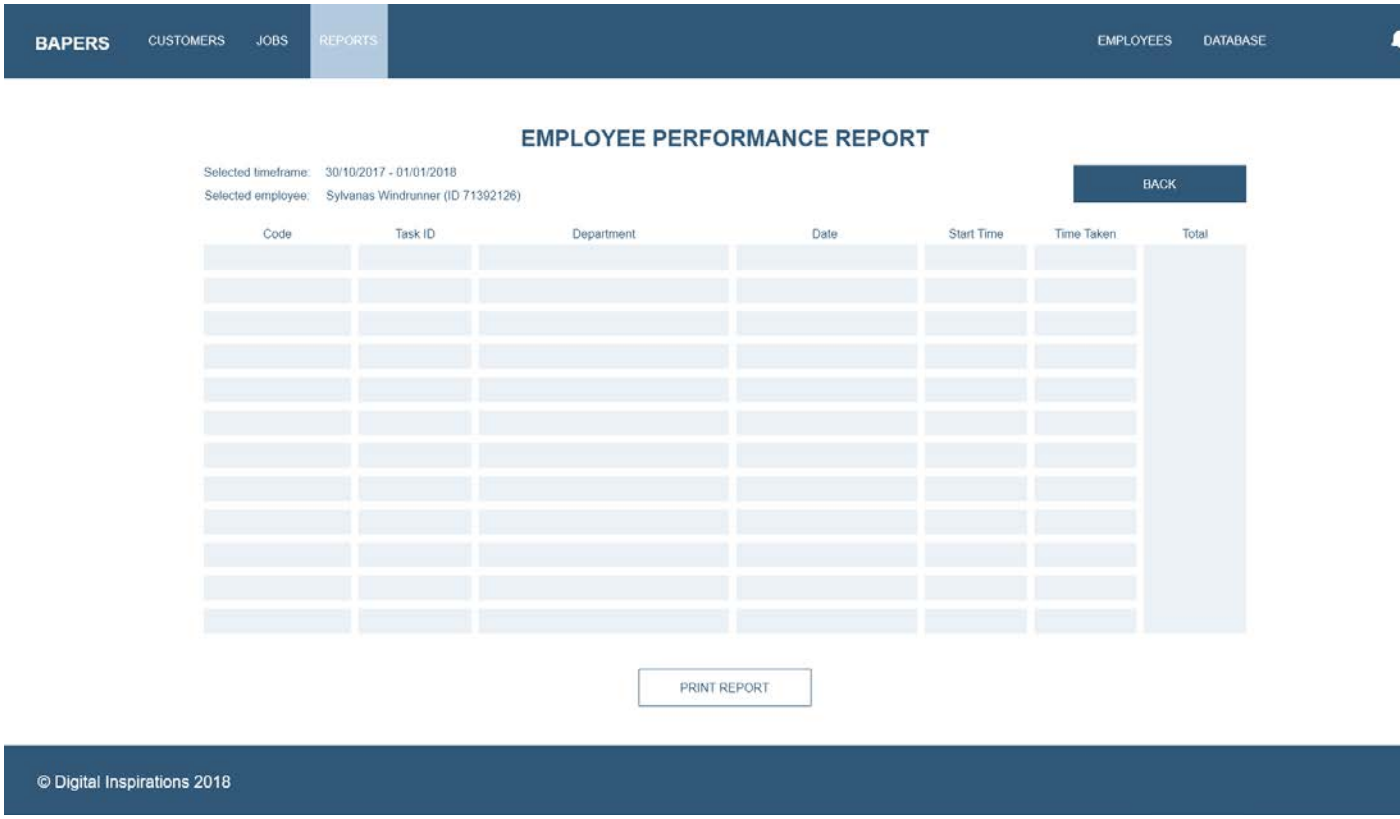
4.1. Reports – Generated Customer Report



This report provides an overview of all jobs brought in by a customer over a period of time selected during the generation of the report.

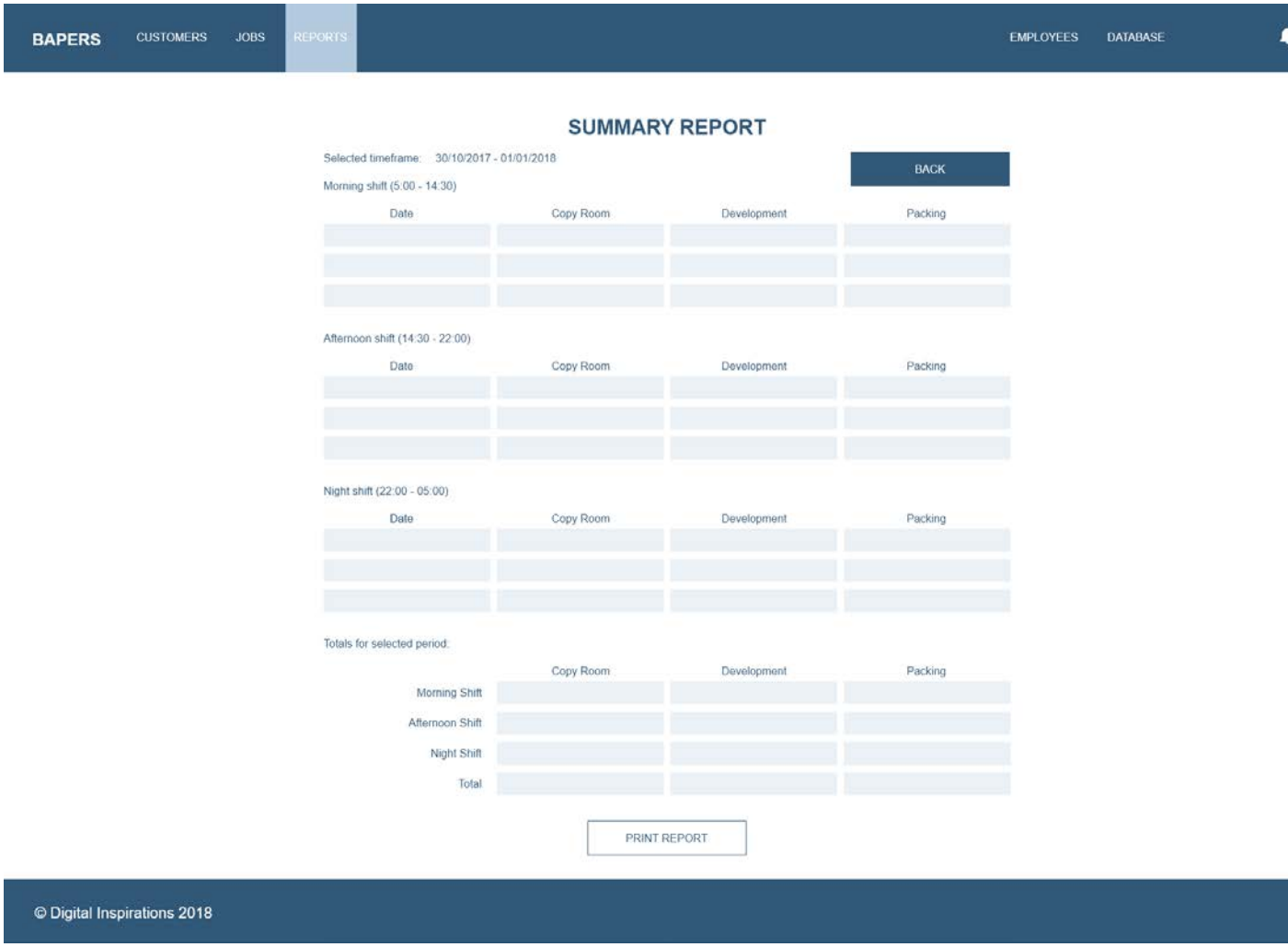
The user can choose to PRINT REPORT or to go BACK to the previous page they have visited.

4.2. Reports – Generated Employee Report



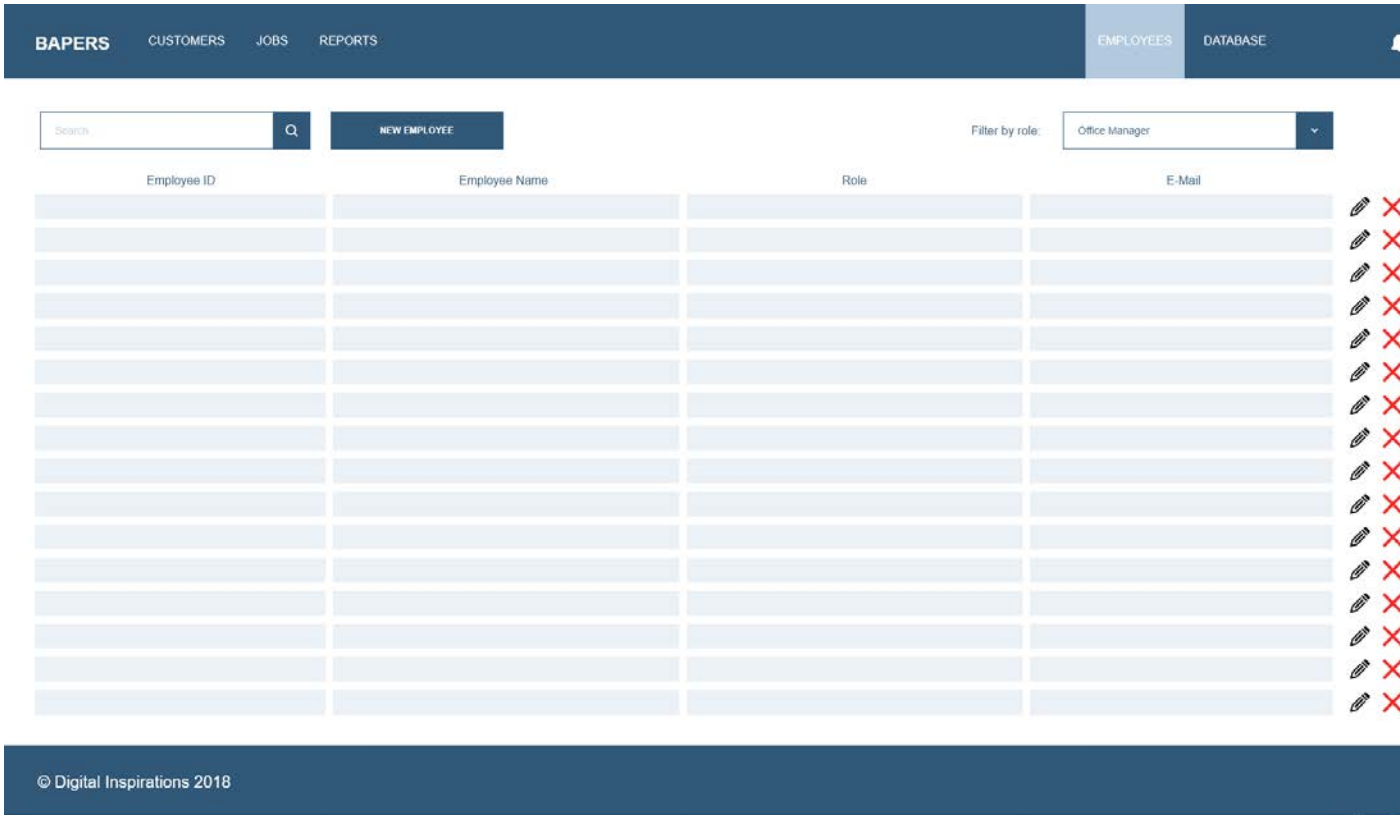
This report provides an overview of the tasks completed and the time taken on each of them, as well as in total, by a particular employee in the system. Once again, the report can be printed or the user can go back to the previous page.

4.3. Reports – Generated Summary Report



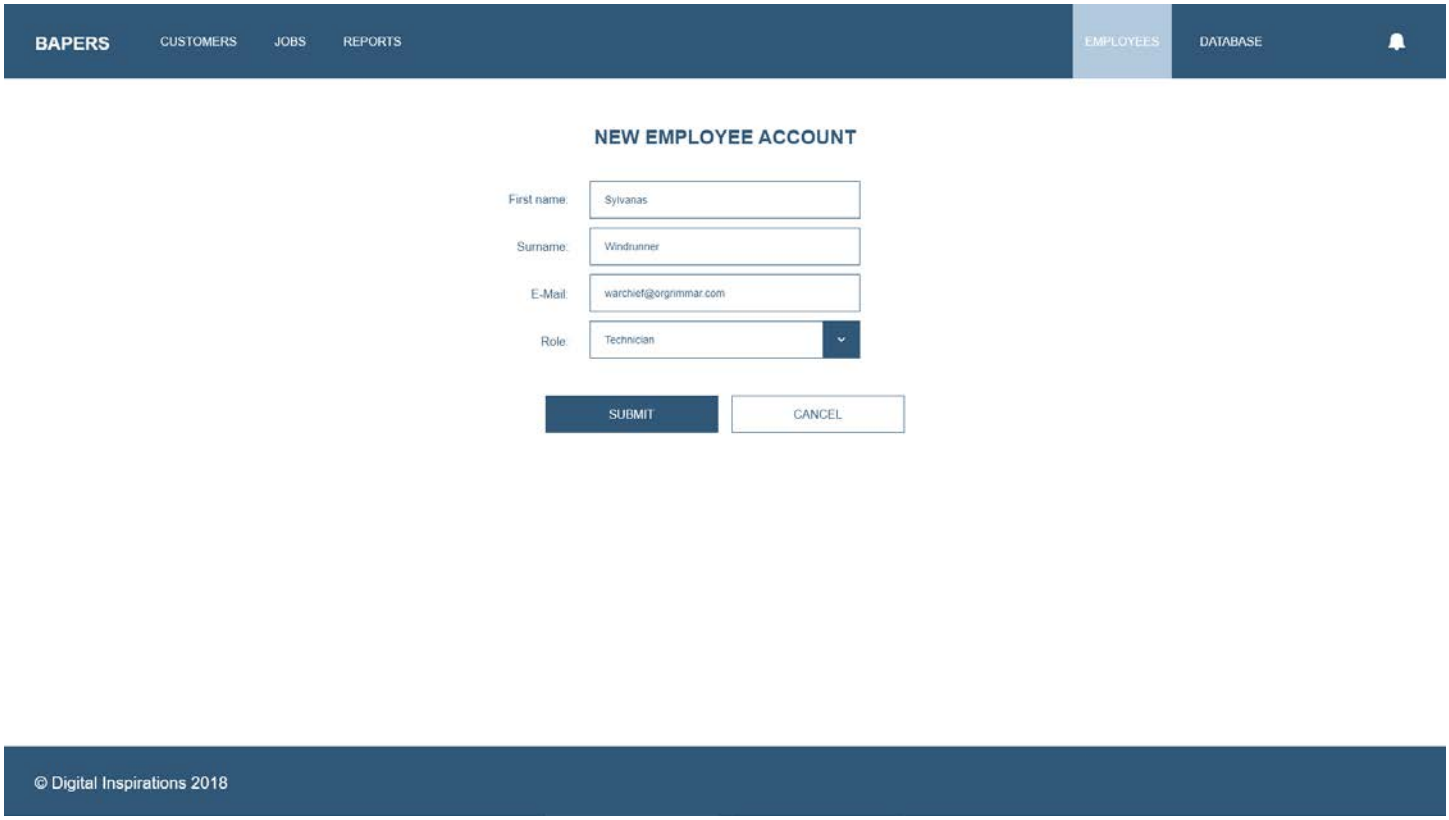
This is a summative report about time spent working in each of the areas of The Lab on different shifts during the day.

5. Employees – Manage/Search



This page, available only to office managers, provides them with the opportunity to **search** for and **filter** employees currently registered in the system, as well as create new ones by clicking on NEW EMPLOYEE (see **page 5.1**) or **edit and delete** existing ones (using the crayon and cross icons). Choosing to **edit** an employee's information takes the user to **page 5.2**.

5.1. Employees – Create New Employee Account

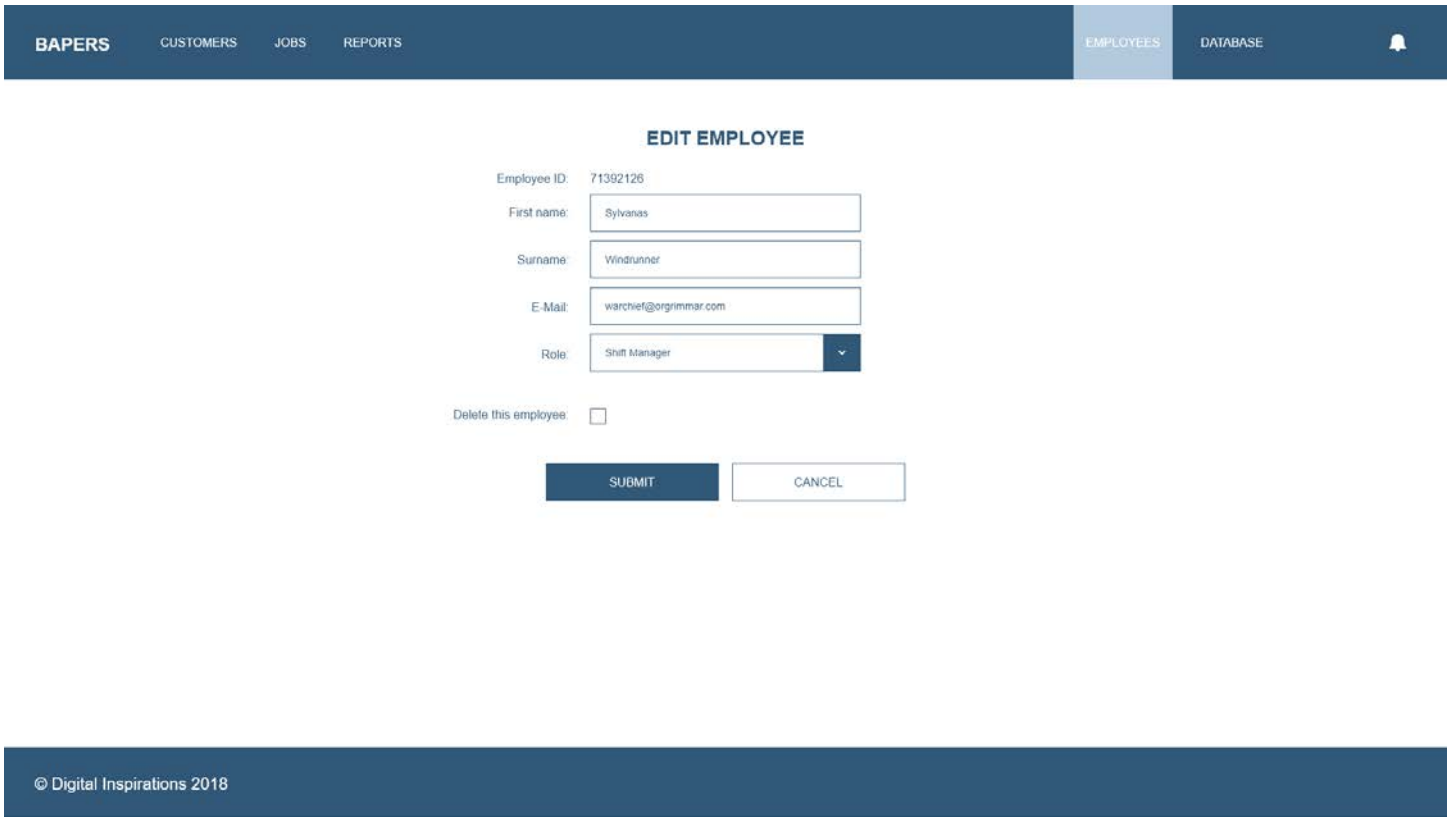


This page allows an office manager to create new employee accounts.

To make a new employee, the user has to specify their name, e-mail, as well as select their **role** using the dropdown selector. Roles available are Receptionist, Technician, Shift Manager, and Office Manager. Only

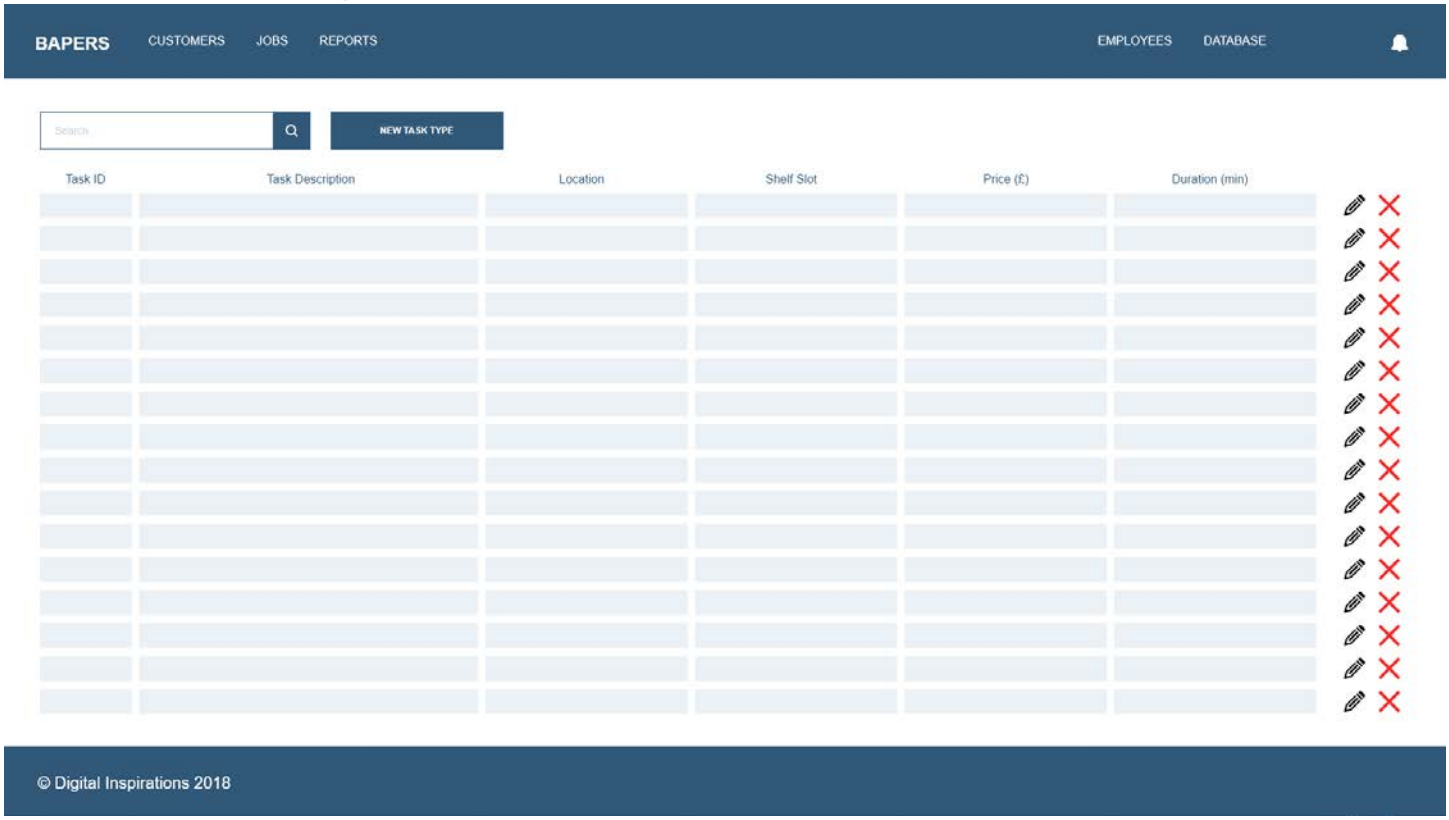
After clicking SUBMIT, if all information is correct, an employee account will be added to the system, with their ID and password used to log into the system being generated automatically.

5.2. Employees – Edit Employee Account



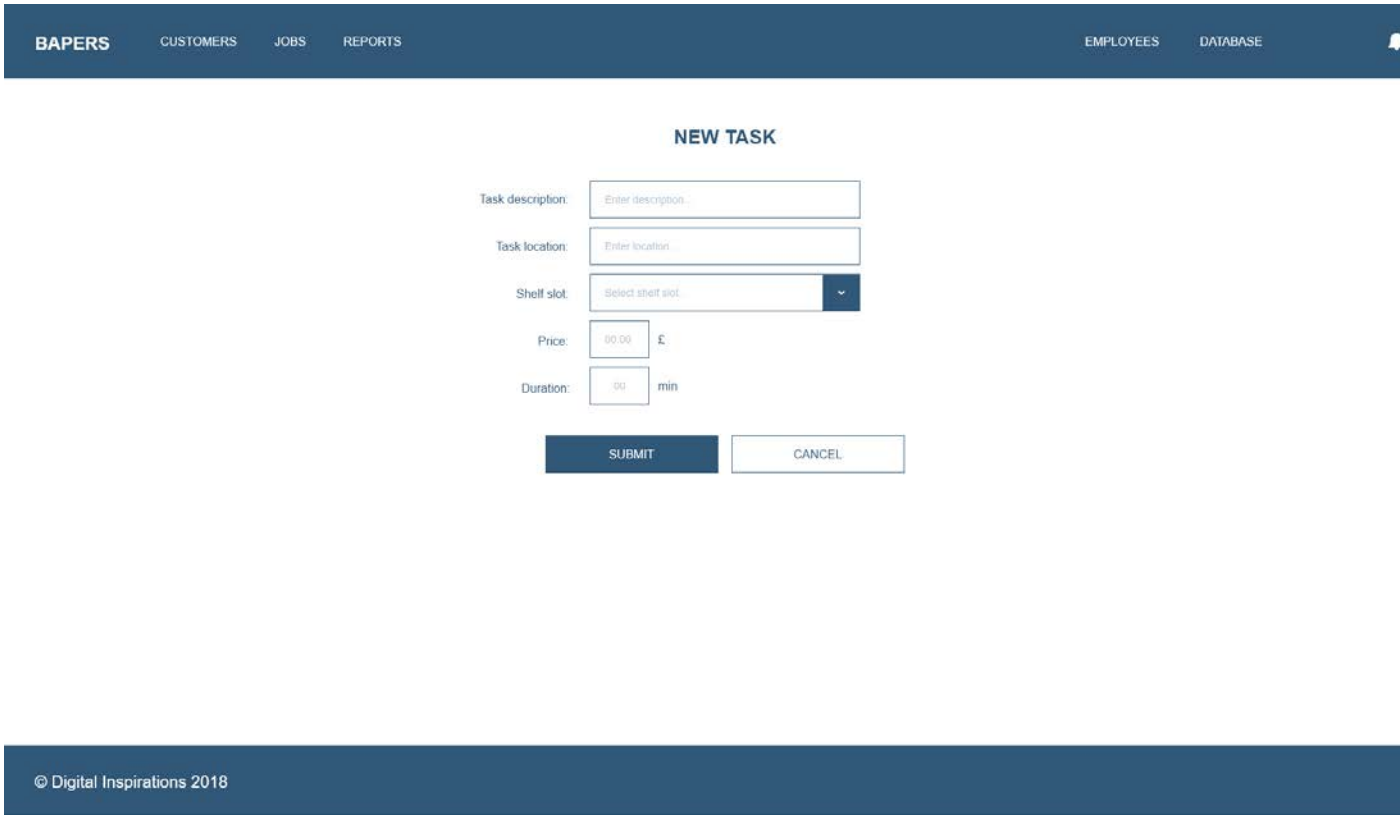
Similarly to creating a new employee, this page allows the office manager to edit all information about an existing employee except for their ID which is auto-generated by the system. Deletion of the employee account is also possible from this page by using the checkbox.

6. Tasks – Manage/Search



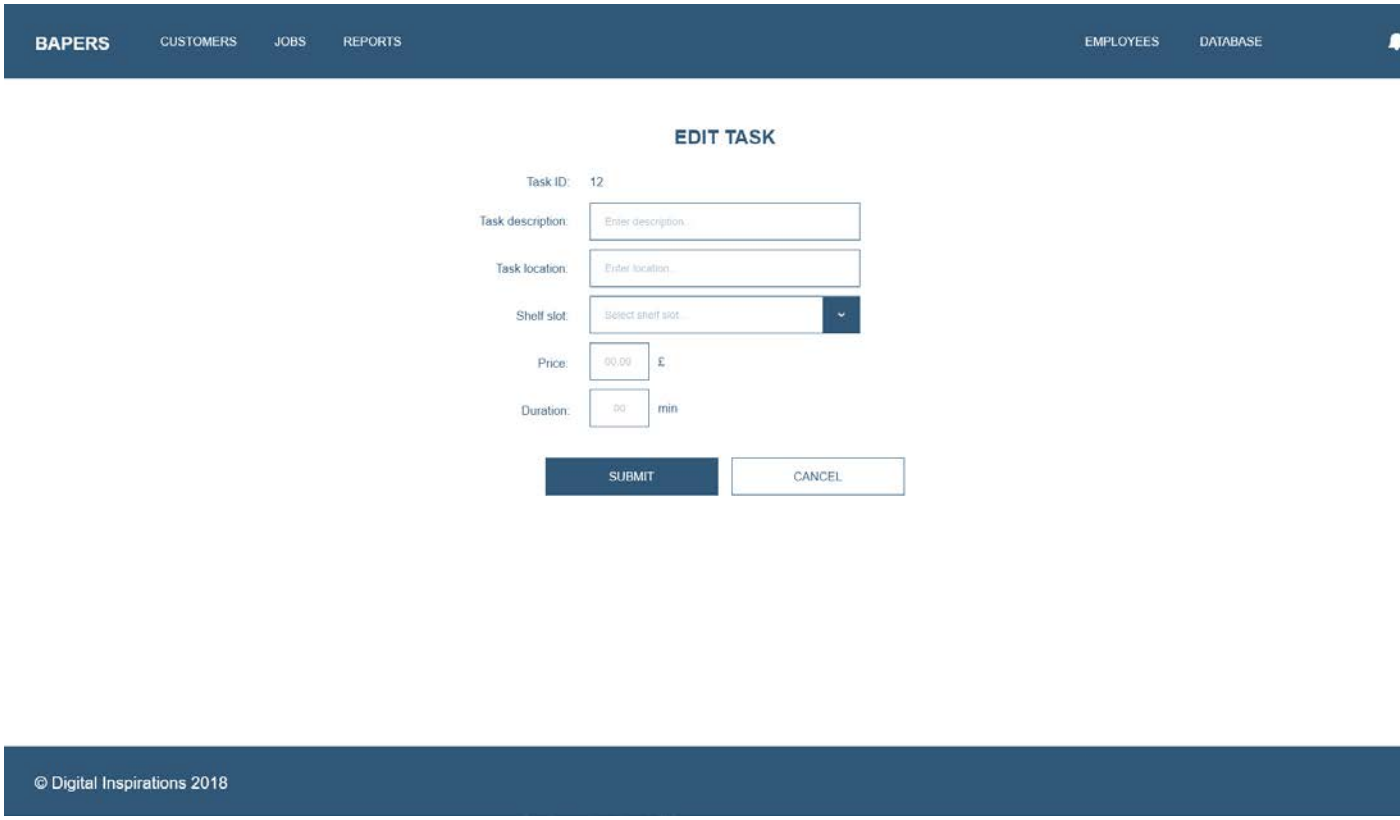
Office managers are capable of adding new types of tasks to the system using the NEW TASK TYPE button (**page 6.1**), as well as **editing or deleting** (**page 6.2**) tasks using the respective crayon and cross icons. Overall the page operates in the same way as customer, job, and employee management pages which are similarly formatted and described in detail.

6.1. Tasks – Create New Task



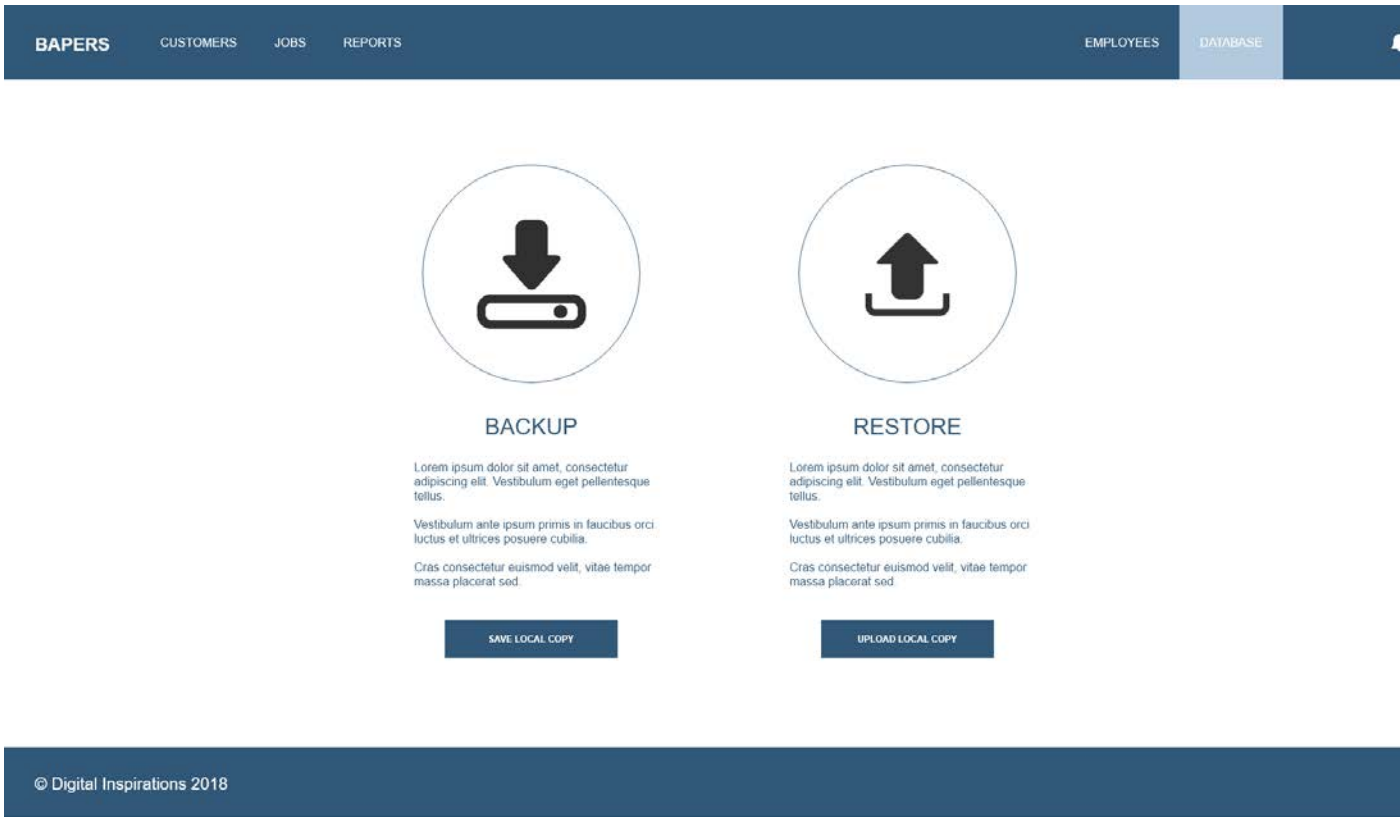
When creating a new task, the office manager has to specify its description, its location in The Lab, and select its designated shelf slot. A base price and duration for the task also have to be set.

6.2. Tasks – Edit Existing Task



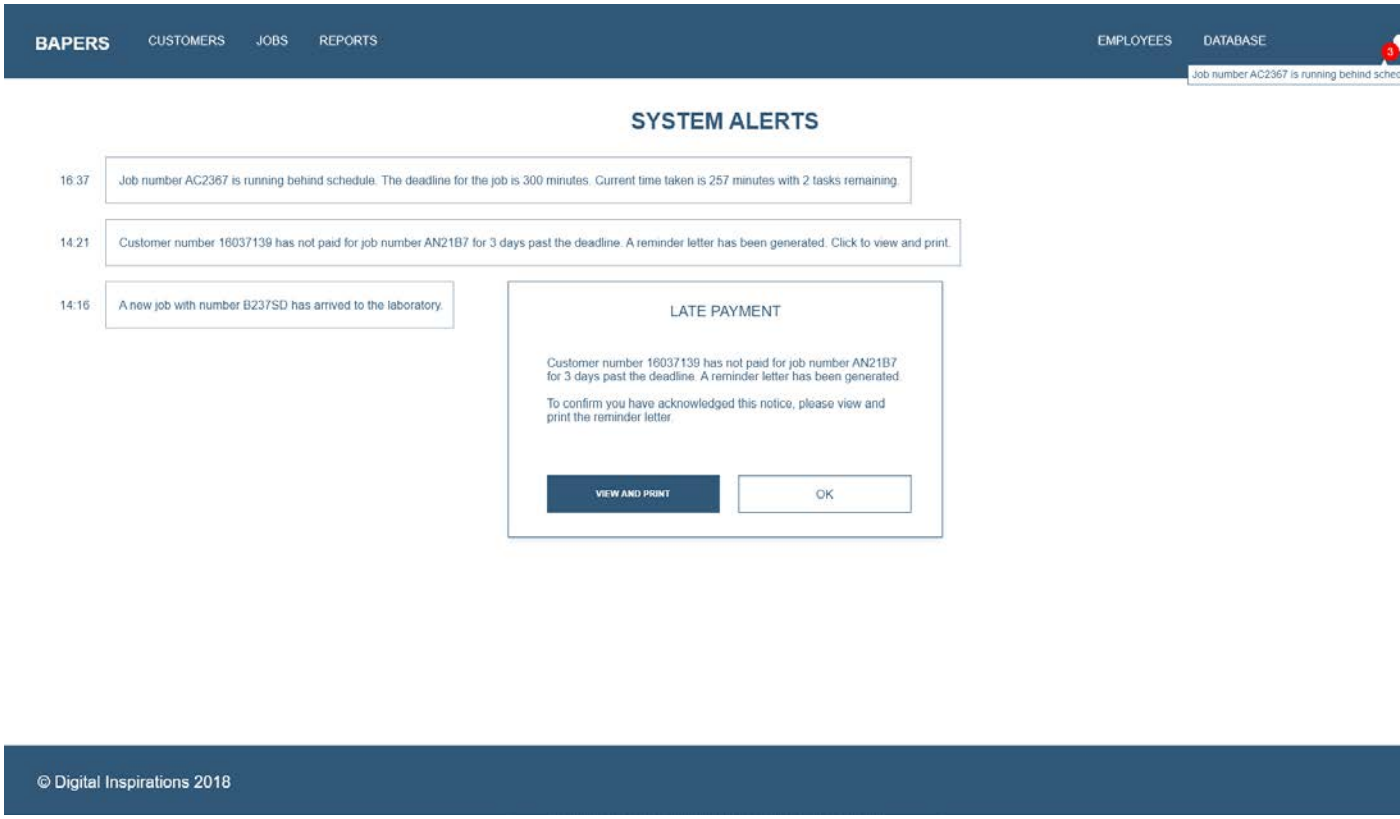
Editing a task provides the user with the ability to change any information about the task apart from its auto-generated ID.

7. Database Management Page



From this page (only accessible to office managers) a user can force the creation of a local backup of the current state of the database for the whole system, using the SAVE LOCAL COPY button. Similarly, they can use the UPLOAD LOCAL COPY button to restore the state of the database from a previously saved file, be it saved automatically or on demand.

8. System Alerts Page



This page, only available to office and shift managers, provides them with an overview of all of the alerts they have received about the changes in the system.

When a user receives a new alert, they can see a red circle with the number of unread alerts appear on the **bell icon** on the navigation bar. When they hover the icon, they can see the most recent alert.

If the alert is urgent (e.g. if it's a late payment alert), the user will also see a **pop-up window** giving them the opportunity to VIEW AND PRINT a reminder letter for the customer who is late with payment. Unless the office manager prints said letter, the pop up will keep showing up every 15 minutes until the letter is reviewed and printed.

5.4 GUI and Design Classes

To describe the link between the design class diagram and the GUI designs, I will describe where each boundary class operation from the design class diagram can be accessed in the GUI. The boundary class for the GUI in the design class diagram is as follows:

- The user can **selectJob()** on **page 3** of the GUI, where they have the opportunity to edit any of the jobs shown in the table;
- **selectCustomer()** is available on **page 2** similarly to selectJob();
- **Page 2.1** in the GUI is responsible for **createCustomer()**;
- **changeJobPriority()** is available through **page 3.1** where job urgency can be set during its creation phase;
- **printLabelAndReceipt()** is available from the job details page (**page 3.2**);
- **displayJobStatus()**, **displayJobStatusList()**, **updateJobStatus()** and **displayTaskStatus()**, **selectTask()**, **startTask()**, **stopTask()** and **updateTask()** are also all accessible from the job details page (**page 3.2**), where you can see the current progress on the sub-tasks within the job and mark them as completed.
- **setCustomerData()** occurs during customer creation on **page 2.1**, and **getCustomerData()** is available on **page 2.2**;
- **setJobID()** occurs automatically upon creation of a new job, and **getJobID()** is available on **page 3.2** where the ID is displayed;
- **getPaymentData()** is available from each individual job details page and refers to its payment status (**page 3.2**);
- **setPaymentData()** and **makePayment()** refers to **page 3.2.1** where a user can record a payment for a job;
- **isJobComplete()** can be found on the job details page (**3.2**) and refers to the status of completion of all of its tasks;
- **addTask()** and **removeTask()** can be done during the creation of a new job (**page 3.1**); **saveJob()** occurs when the user clicks submit on this page;

- **jobEntered()**, **displayNewJobAlert()**, **acceptAlert()**, **displaySecondReminderAlert()** refer to elements of the alerts page (**page 8**) in the system, as well as the respective alert pop up windows described on that page.
- **displayAllInDefault()** is part of the customer search interface on **page 2**, where it is available as a filter checkbox.
- **reactivateInDefailtAccount()** is available to office managers from **page 2.2**.
- **chooseDiscountPlan()** refers to discount type selection on **page 2.2.1** (during the discount set up process).
- **databaseBackUpOnDemand()** and **restoreSystemOnDemand()** are available from **page 7**, the database management page.
- All of the printing functions are available from the respective alert pop-up described on **page 8**.