IMPLEMENTATION REPORT

Liu liu Megatech

TEAM 6

Contents

Compilation

Package description

- src/bapers

The main package is “bapers”, and this only contains one class, “BAPERS.java”, which holds the main method for the application, and other code that is performed before the application initialises.

- src/bapers/data

The “data” package holds all of the files that were generated by netbeans, and are the only files that will directly access the database using the eclipselink jpa.

- src/bapers/data/dataAccess

The “dataAccess” package holds jpa controller classes that will provide abilities, such as being able to create new entities in the database, or to execute queries on the dataset.

- src/bapers/data/domain

The “domain” package holds pojo mapping of all the entities from the database, to java objects.

- src/bapers/service

The implementation/interface pairs utilise the jpa controllers in the “dataAccess” package to provide services to the guis to allow them to interact with the database.

- src/bapers/userInterface

This package holds the controller classes for the guis to allow them to interact with the system, and to handle user input.

- src/bapers/userInterface/fxml
- src/bapers/userInterface/styles

The “fxml” package holds all the markup files that describe the guis, and the “styles” holds all the styling information for each of the guis.

- src/bapers/utility

The “utility” package holds all the “utility” classes that do not depend on any other part of the system, and can be accessed statically.

Description of Compilation

The BAPERS program was written in java, using jdk 1.8. The code was written using the netbeans IDE, targeting the ant build system; in the root directory of the codebase, the nbproject directory and build.xml were both autogenerated by netbeans, and hold the build targets as well as other properties for the project. The build.xml file was modified so that, when the project is built, the source code, and all the dependencies are all bundled into a single .jar file in the "dist/" directory.

As the current BAPERS system is only a prototype, no installer has been provided. The BAPERS system relies on MySQL server as the database backend, where all the data will be stored, and from which the queries will be run. In order to get the system installed, install the latest version of MySQL Community Server 5.7, MySQL workbench and MySQL Utilities 1.6. Also, ensure that the binary files included with MySQL Utilities are added to the system path variable, as the system relies upon these utilities to implement system backup and restore. Once MySQL server is installed and running, open the provided IN2018-Database/BAPERS.mwb with MySQL workbench, then forward engineer the schema with root user, and the root password being "haddockexecellipsis". Please note that we intend to create a dedicated user for the final distribution of the software.

In order to build the project, either, open the project with the netbeans ide and click clean & build, or type "ant jar" in the root directory. Please note, that before building, setting the value of the “TESTING” constant to false in src/bapers/BAPERS.java will produce the functionality required for end use. Then run the project by executing the "IN2018-BARERS.jar" file in the "dist/" directory. When running the jar for the first time, a file called "intervals.dat", and a folder called "backups/" will be generated in the location the file was called from. The "intervals.dat" file contains the description of the times between automated system events, such as database backup and report generation, while the "backups" directory holds all the backups that have been taken of the system. If the "intervals.dat" exists when the jar is run, the system will detect this, and attempt to read the relevant data from these.

In the "IN2018-BAPERS/src/" directory lies two sub directories, "bapers/" and "META-INF/", the latter is the directory that stores the persistence information relevant to our MySQL integration, while the former contains our java source code.

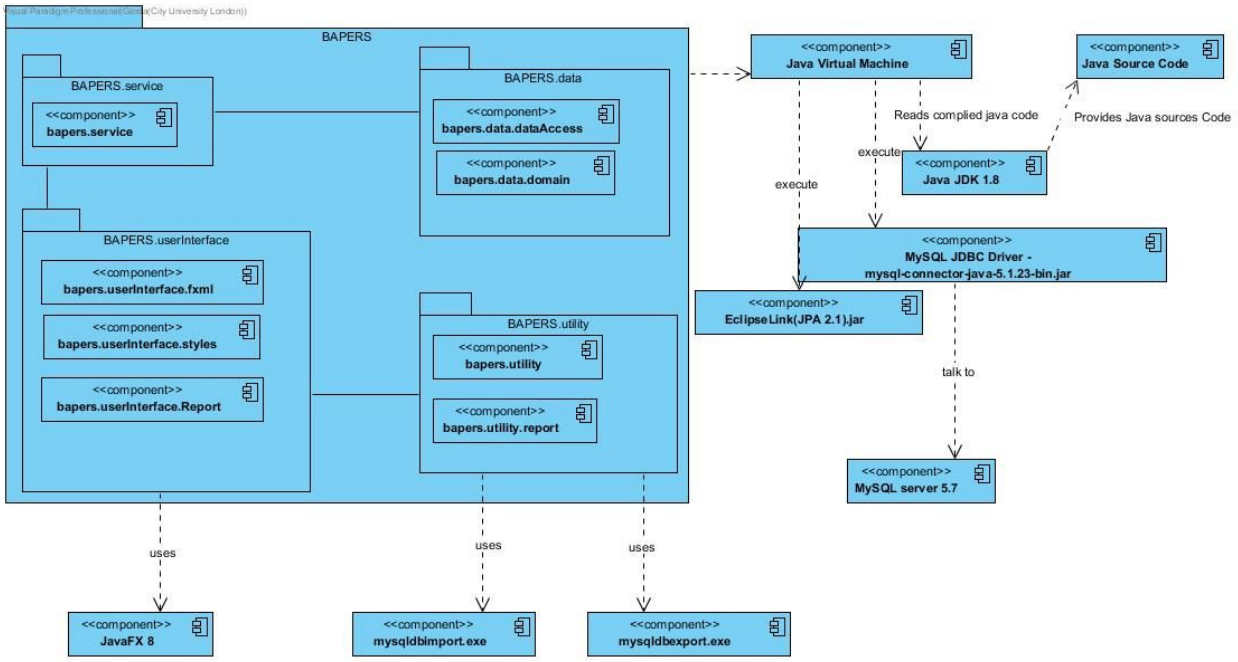
All the files inside the "bapers.data" package were autogenerated by the IDE, and are used to access the database utilising the eclipselink implementation of jpa.

The front end of the BAPERS system was developed with JavaFX to create an appealing and efficient Graphical User Interface which implements the systems visual aspects and interacts with the backend of the system.

Run Time

Since the prototype is a stand-alone application this may not be necessary, provided everything is on the same stand-alone PC therefore all run-time components will be deployed on the same computer.

Component Diagram developed for the BAPERs system



Testing

Unit Testing

We have made the following Test Cases based on our use case specs to test our BAPERS system

Upgrade Customer Account

Main Flow:

Use case ID: 13	**Use case name:** Upgrade Customer Account
Test number: 1	
Objective: Test the main flow	
Set up: 1. Create a customer account called “City, University of London (City)” with Account number: ACC0001 ensuring that it is not a valued account, thus no discount plan linked to it however, should be applicable for the different discounts. 2. The office manager must be logged in to the system (Username: Manager and Password: Get_it_done). He/she has access to the customer accounts therefore, he/she will have access to City account details and can edit them at his/her discretion.	
Expected results: 1. The new “Valued” status will be assigned to City. 2. The database will be updated with the new value for City. 3. The City account will now have a discount plan (fixed discount plan).	
Test: 1. Office Manager accesses the “Customer Accounts”, then searches for City, University of London account and selects it. 2. All customer account details are shown with available discount plans. 3. Office Manager fills in a Fixed discount plan with a discount rate of 1% in available/respective fields. 4. The Office Manager then clicks on the “Set Fixed Plan” GUI button.	
Test record: The database is updated with the new “valued” status and fixed discount plan for City, University of London account.	
Date: 17/04/2018	**Tester:** Mugheeth
Result: Passed	
Date: 18/04/2018	**Tester:** Gerda
Result: Passed	

Alternative Flow:

Use case ID: 13	**Use case name:** Upgrade Customer Account : Existing Valued Customer
Test number: 2	
Objective: Test the alternative flow where the process of making a customer “valued” cannot happen.	

Set up: 1. Create a valued customer account called “City, University of London (City)” with Account number: ACC0001 and should already have a fixed discount plan with a discount rate of 1%. 2. The office manager must be logged in to the system (Username: Manager and Password: Get_it_done). He/she has access to the customer accounts therefore, he/she will have access to City account details and can edit them at his/her discretion.
Expected results: 1. The Office Manager will be prompted that the City, University of London account has already been set a discount plan (meaning it already a valued customer account) and cannot be further upgraded.
Test: 1. Office Manager accesses the “Customer Accounts”, then searches for City, University of London account and selects it. 2. All customer account details are shown with available discount plans/discount plan set. 3. Office Manager fills in a Fixed discount plan with a discount rate of 1% in available/respective fields. 4. The Office Manager then clicks on the “Set Fixed Plan” GUI button.
Test record: The Office manager gets prompted Invalid Account Upgraded as City, University of London is an existing valued customer account.

Date: 17/04/2018	**Tester:** Mugheeth
Result: Passed	
Date: 18/04/2018	**Tester:** Gerda
Result: Passed	

Login

Main Flow:

Use case ID: 5	**Use case name:** Login
Test number: 1	
Objective: To test the main flow	
Set up: 1. An account must be made for the Office Manager (username: Manager and password: Get_it_done), copy room Technician (username: Copy and password: Too_dark), Shift Manager (username: Clerk and password: Paperwork) and Receptionist (Username: Hello and password: Hello_there).	
Expected results: 1. The user logs in to the system.	
Test: 1. All the users mentioned above, enters their username and password in the respective present fields, correctly. 2. Then click the login GUI button.	
Test record: The respective users logs into the system.	
Date: 17/04/2018	**Tester:** Mugheeth

Result: Passed	
Date: 18/04/2018	**Tester:** Gerda
Result: Passed	

Alternative Flow:

Use case ID: 5	**Use case name:** Login : Invalid Details
Test number: 2	
Objective: To test the alternative flow where the user is unable to login due to invalid/incorrect details.	
Set up: 1. An account must be made for the Office Manager (username: Manager and password: Get_it_done), copy room Technician (username: Copy and password: Too_dark), Shift Manager (username: Clerk and password: Paperwork) and Receptionist (Username: Hello and password: Hello_there).	
Expected results: 1. The user is unable to log in to the system.	
Test: 1. All the users mentioned above, enters their username and/or password incorrectly purposely in the respective present fields. 2. They then click the login GUI.	
Test record: Login Failed prompt comes up saying “Invalid id or password” and shows “login attempts: x”.	
Date: 17/04/2018	**Tester:** Mugheeth
Result: Passed	
Date: 18/04/2018	**Tester:** Gerda
Result: Passed	

Generate Reports

Main Flow:

Use case ID: 12	**Use case name:** Generate Reports
Test number: 1	
Objective: To test the main flow	
Set up: 1. Create an account for Ms Eva Bauyers with account number ACC0004 and additional information as required. 2. Set a job for Ms Eva Bauyers. 3. Projected time frame for the job is 5 days and nights. 4. The office manager must be logged in to the system (Username: Manager and Password: Get_it_done). 5. Date start: 12/04/2018. end: 20/04/2018	
Expected results: 1. Generate the Individual job report. 2. Generate the Individual performance report for each team/member. 3. Generate the Summary Performance report for each shift (day/night).	

Test: 1. Office manager access the generate report functionality by clicking the report tab. 2. To generate the “individual performance report” he/she selects that radio button, selects the start and end dates(start: 12/04/2018 and end: 20/04/2018) and enters the first and surname of the employee that he/she wants the report for (i.e. Lee Hong). 3. To generate the “Summary performance report” he/she selects that radio button, selects the start and end dates (start: 12/04/2018 and end: 20/04/2018). 4. To generate the “individual report” he/she selects that radio button, selects the start and end dates(start: 12/04/2018 and end: 20/04/2018) and enters the Account Number (ACC0004). 5. Then presses the “Confirm” GUI button after filling in the fields for the chosen report to be generated. 6. The corresponding report will be generated with further options to print.
Test record: The corresponding desired reports are being generated.

Date: 17/04/2018	**Tester:** Mugheeth
Result: Failed- MySQl utilities not installed and have binaries added to the path	
Date: 18/04/2018	**Tester:** Gerda
Result: Passed	

Update Job Status

Main Flow:

Use case ID: 6	**Use case name:** Update Job Status
Test number: 1	
Objective: Test the main flow	
Set up: 1. Create an account for Ms Eva Bauyers with account number ACC0004 and additional information as required. 2. Process a job that is incomplete for Ms Eva Bauyers that involves the Mount Transparencies Task in the finishing room as the last task. 3. The finishing room technician must be logged in to the system (Username: Finish and Password: Fine_touch). 4. All tasks must be completed.	
Expected results: 1. The status of the Job is updated and marked as completed.	
Test: 1. The Finishing room technician logs into the system. 2. Access the Job Processing tab. 3. Select Ms Eva Bauyer from the select customer options to show just that individual’s jobs/tasks. 4. Accesses the last task (Mount Transparencies Task) and fills in the fields (i.e. Job Task ID: x, Task ID: x etc.). 5. Clicks on the “Update Task ID” GUI button. 6. As it is the last task for the job, the Job Status automatically then changes to a “Completed” status.	

7. The status of the job can be altered to fit the progress of the job.	
Test record: The Job status changes to a completed status.	
Date: 17/04/2018	**Tester:** Mugheeth
Result: Passed	
Date: 17/04/2018	**Tester:** Gerda
Result: Passed	

Record Payment

Main flow:

Use case ID: 1	**Use case name:** Record Payment
Test number: 1	
Objective: Test the main flow	
Set up: 1. The office manager must be logged in to the system (Username: Manager and Password: Get_it_done). 2. Customer name "Hello Magazine" must be set up and have outstanding balance of £187.18 of unpaid jobs. Date: 15/04/2018	
Expected results: 1. Payment is stored in the Database. 2. Payment amount deducted from the customer's bill.	
Test: 1. Select the Payment functionality, select "Hello Magazine" 2. Enter the amount paid: £187.18. 3. Select date 2018/04/15. 4. Click on payment button.	
Test record: Payment recorded successfully	
Date: 17/04/2018	**Tester:** Patrick G
Result: Failed- Could not view Customer List first time with the unpaid jobs	
Date: 18/04/2018	**Tester:** Mugheeth
Result: Passed	

Alternative flow:

Use case ID: 2	**Use case name:** Record Payment: Card Payment
Test number: 1	
Objective: Test the alternative flow	

Set up: 1. The office manager must be logged in to the system (Username: Manager and Password: Get_it_done). 2. Customer name “Hello Magazine” must be set up and have outstanding balance of £187.18 of unpaid jobs. 3. Customer card last 4 digits 9090. Expiry 01/20. (MasterCard). Date: 15/04/2018	
Expected results: 1. Payment is stored in the Database. 2. Payment card details are stored in the Database. 3. Payment amount deducted from the customer’s bill.	
Test: 1. Select the Payment functionality, select “Hello Magazine” 2. Enter the amount paid: £187.18. 3. Select the card payment option. 4. Select date (2018/04/19), 5. Enter card type (MasterCard), number (9090), expiry date (01/20). 6. Click on payment button.	
Test record: Card payment recorded successfully	
Date: 17/04/2018	**Tester:** Patrick G
Result: Passed	
Date: 18/04/2018	**Tester:** Mugheeth
Result: Passed	

Place Order

Main flow:

Use case ID: 1	**Use case name**: Place Order: Urgent Job
Test number: 1	
Objective: Test the main flow	
Set up: 1. Create a customer account called “City, University of London (City)” with Account number: ACC0001. 2. The office manager must be logged in to the system (Username: Manager and Password: Get_it_done). He/she has access to the customer accounts therefore, he/she will have access to City account details and can edit them at his/her discretion.	
Expected results:	

1. An order will be placed for “City, University of London(City)”
Test: 1. Office Manager logs in to the system, accesses “Place Order”, then searches for City, University of London account and selects it 2. Selects radio button “Yes” for urgent job 3. Records a deadline for the job “200 mins” and enters in extra percentage charge “2%” 4. Enter the item amount “£49” 5. 5.Component description are added in “Add gold colouring” 6. 6.Click “add” to be able to add the component description 7. Select the component and the tasks that will be added and click add 8. The Office Manager then clicks on the “Place Order” GUI button.
Test record: The database is updated with a new Job added

Date:17/04/2018	**Tester:** Mugheeth
Result: Passed	
Date:18/04/2018	**Tester:** Gerda
Result: Passed	

Alternative Flow:

Use case ID: 1	**Use case name:** Place Order: Standard Job
Test number: 2	
Objective: Test the alternative flow	
Set up: 1. Create a customer account called “City, University of London (City)” with Account number: ACC0001. 2. The office manager must be logged in to the system (Username: Manager and Password: Get_it_done). He/she has access to the customer accounts therefore, he/she will have access to City account details and can edit them at his/her discretion.	
Expected results: 1. An order will be placed for “City, University of London(City)”	

Test: 1. Office Manager logs in to the system, accesses “Place Order”, then searches for City, University of London account and selects it. 2. Selects radio button “No” for urgent job 3. Records a deadline for the job “200 mins” 4. Enter the item amount “£49” 5. Component description are added in “Add gold colouring” 6. Click “add” to be able to add the component description 7. Select the component and the tasks that will be added and clicks add 8. The Office Manager then clicks on the “Place Order” GUI button.	
Test record: The database is updated with a new Job added	
Date:17/04/2018	**Tester:** Mugheeth
Result: Passed	
Date:18/04/2018	**Tester:** Gerda
Result: Passed	

Add User

Main Flow:

Use case ID: 3	**Use case name:** Add User
Test number: 1	
Objective: Test the main flow	
Set up: The office manager must be logged in to the system (Username: Manager and Password: Get_it_done).	
Expected results: 1. Database creates a new user. 2. Newly created user account has correct privileges. 3. The System informs that user account has been successfully created.	
Test: 1. Select User functionality from the BAPERS homepage 2. Enter details for the user account (Username: John, Password: John1, User type: Office Manager, User Location: Finishing room). 3. The System informs that user has been successfully created.	
Test record: User created successfully	
Date: 17/04/2018	**Tester:** Patrick G

Result: Passed	
Date: 18/04/2018	**Tester:** Mugheeth
Result: Passed	

Alternative Flow:

Use case ID: 4	**Use case name:** Add User: Already Exists
Test number: 2	
Objective: Test the alternative flow when the System's response to adding user of the same username	
Set up: The office manager must be logged in to the system (Username: Manager and Password: Get_it_done).	
Expected results: 1. The System informs the user that user account with the same username already exists	
Test: 1. Select "User" functionality, 2. Enter details for the user account (Username: John, Password: John2, User type: Office Manager, User Location: Copy room). 3. The System informs the user that user account with the same username already exists	
Test record: Alert pops up, user account has not been created	
Date: 17/04/2018	**Tester:** Patrick G
Result: Passed	
Date: 18/04/2018	**Tester:** Mugheeth
Result: Passed	

Automatic Backup

Main Flow:

Use case ID: 5	**Use case name:** Automatic backup
Test number: 1	
Objective: Test the main flow	
Set up: Automatic backup period (Day: 1, Hours:0, Minutes:0 (every day) is specified.	
Expected results: 1. The database server is backed up every set time period(every day)	

Test: 1. Select Intervals form from homepage 2. Select the Radio button “Database Backup” 3. Select the (Day: 1, Hours:0, Minutes:0 (every day) for the backup from the drop down menus 4. Select the “Set Time” button 5. Specified time occurs (Each day), the System automatically creates new database backup.	
Test record: Backup occurs each day.	
Date: 17/04/2018	**Tester:** Patrick G
Result: Passed	
Date: 18/04/2018	**Tester:** Mugheeth
Result: Passed	

Update Existing Task

Main Flow:

Use case ID: 6	**Use case name:** Update existing task
Test number: 1	
Objective: Test the main flow	
Set up: 1. The office manager must be logged in to the system (Username: Manager and Password: Get_it_done). 2. Task (ACT108) for “InfoPharma Ltd” is created.	
Expected results: 1. Task has been updated. (User: Manager) 2. The updated task user (Manager) corresponds to the one in the database records.	
Test: 1. Select Job Processing->Select “InfoPharma Ltd”-> Click on the task ACT108. 2. Change User from “Development” to “Manager”. 3. Check whether task User value (Manager) corresponds to the one in the database records.	
Test record: Task has been successfully updated	
Date: 17/04/2018	**Tester:** Patrick
Result: Passed	
Date: 18/04/2018	**Tester:** Mugheeth
Result: Passed	

Alternative flow:

Use case ID: 7	**Use case name:** Update existing task: NoUser
Test number: 2	
Objective: Test the alternative flow when the System's response to updating task with nonexistent user.	
Set up: 1. The office manager must be logged in to the system (Username: Manager and Password: Get_it_done). 2. Add Non Existing user with username: Fake.	
Expected results: 1. The System doesn't change the task value for User.	
Test: 1. Select Job Processing->Select "InfoPharma Ltd"-> Click on the task ACT108. 2. Change User from "Development" to "Fake". 3. Check whether task User value (Development) corresponds to the one in the database records.	
Test record: Task has been unchanged	
Date: 17/04/2018	**Tester:** Patrick
Result: Passed	
Date: 18/04/2018	**Tester:** Mugheeth
Result: Passed	

Create Customer Account

Main Flow:

Use case ID: 3	**Use case name:** Create Customer Account
Test number: 1	
Objective: To test the main flow, where we create a new customer account.	
Set up: 1. An account must be made for the Office Manager (username: Manager and password: Get_it_done) and he/she must be logged in to the system.	
Expected results:	

1. New Customer account is created.
Test: 1. The Office manager access the “Create Account” tab within the system. 2. An account is made for City, University of London (City), filling in all the fields and with the following information: 3. **Account holder name:** City, University of London, **Email:** David.Rhind@city.ac.uk, **Landline:** 0207 040 8000, **Address:** Northampton Square, London EC1V 0BH (in all its corresponding fields). 4. The office manager then clicks the “Create Customer Account” GUI Button. 5. In the background the system then uses the “Lookup Customer Account” use case to check the database to see if the account already exists. 6. Account does not exist.
Test record: A new customer account is made with the inputted details and a GUI popup is shown saying “Customer Account has been created!”

Date: 17/04/2018	**Tester:** Mugheeth
Result: Fail - the mySQL database server was not functioning properly thus the new customer accounts could not be created/saved. To fix this the database had to be revisited and checked/tested.	
Date: 18/04/2018	**Tester:** Patrick G
Result: Pass	

Alternative flow:

Use case ID: 3	**Use case name:** Create Customer Account: Account Already Exists
Test number: 2	
Objective: To test the alternative flow, where the account that is being created already exists thus cannot be created.	
Set up: 1. An account must be made for the Office Manager (username: Manager and password: Get_it_done) and he/she must be logged in to the system.	
Expected results: 1. Notified that the customer account cannot be created as it already exists.	

<table>
<tr><td colspan="2">Test:
2. The Office manager access the “Create Account” tab within the system.
3. An account is made for City, University of London (City), filling in all the fields and with the following information:
4. Account holder name: City, University of London, Email: David.Rhind@city.ac.uk, Landline: 0207 040 8000, Address: Northampton Square, London EC1V 0BH (in all its corresponding fields).
5. The office manager then clicks the “Create Customer Account” GUI Button.
6. In the background the system then uses the “Lookup Customer Account” use case to check the database to see if the account already exists.
7. Account exists.</td></tr>
<tr><td colspan="2">Test record: The office manager is notified with a GUI pop up stating “Customer Account already exists!”</td></tr>
<tr><td>Date: 17/04/2018</td><td>Tester: Mugheeth</td></tr>
<tr><td colspan="2">Result: Passed</td></tr>
<tr><td>Date: 18/04/2018</td><td>Tester: Patrick G</td></tr>
<tr><td colspan="2">Result: Passed</td></tr>
</table>

Print Late Payment Reminder

Main Flow:

The print late payment reminder functionality was not implemented within the BAPERS system therefor we were unable to test this case/testing was not required however, a test case was created for it as if it was to be tested.

Use case ID: 7	**Use case name:** Print Late Payment Reminder
Test number: 1	
Objective: To test the main flow	
Set up: 1. Create a customer account for Ms Eva Bauyer, account number: ACC0004. 2. Place an order for Ms Eva Bauyer using the Place Order tab with a set deadline. 3. The office manager must be logged in to the system (Username: Manager and Password: Get_it_done) and accesses the customer accounts and sees that the payment deadline has passed at deadline day + 1 day and no payment has been made by Ms Eva Bauyer.	
Expected results: 1. A GUI pops up asking to print a late payment letter for Ms Eva Bauyer. 2. Once the GUI is clicked a late payment letter is printed.	
Test: 1. Office Manager logs in to the system (Username: Manager and Password: Get_it_done) 2. Views completed jobs in the job process tab by selecting Ms Eva Bauyer in the select customer field. 3. Access the payment tab and filters the search to Ms Eva Bauyers to see what payment is due/overdue. 4. The office manager clicks on the overdue payment in Ms Eva Bauyers jobs list. 5. A GUI appears informing the office manager of the payment not having been made for that job. 6. The office manager clicks the GUI button to print a late payment reminder.	
Test record: N/A	
Date: xx/xx/xxxx	**Tester:** xxxxxxx
Result: Unable to Test	
Date: xx/xx/xxxx	**Tester:** xxxxxxx
Result: Unable to Test	

Alternative Flow:

Use case ID: 7	**Use case name:** Print Late Payment Reminder : Printer Error
Test number: 2	
Objective: To test the alternative flow where the Office Manager is unable to print the letters required due to a print error.	
Set up: 1. Create a customer account for Ms Eva Bauyer, account number: ACC0004. 2. Place an order for Ms Eva Bauyer using the Place Order tab with a set deadline. 3. The office manager must be logged in to the system (Username: Manager and Password: Get_it_done) and accesses the customer accounts and sees that the payment deadline has passed at deadline day + 1 day and no payment has been made by Ms Eva Bauyer. 4. An error with the printer not allowing any prints to be made.	
Expected results: 1. Late payment letter is not printed.	
Test: 1. Office Manager logs in to the system (Username: Manager and Password: Get_it_done) 2. Views completed jobs in the job process tab by selecting Ms Eva Bauyer in the select customer field. 3. Access the payment tab and filters the search to Ms Eva Bauyers to see what payment is due/overdue. 4. The office manager clicks on the overdue payment in Ms Eva Bauyers jobs list. 5. A GUI appears informing the office manager of the payment not having been made for that job. 6. The office manager clicks the GUI button to print a late payment reminder. 7. Another GUI pop up appears informing the office manager that it was “unable to print late payment reminder due to a printer error”.	
Test record: N/A	
Date: xx/xx/xxxx	**Tester:** xxxxxxx
Result: Unable to Test	
Date: xx/xx/xxxx	**Tester:** xxxxxxx
Result: Unable to Test	

Generate 2nd Letter

Main Flow:

Print late payment reminder was not implemented - therefore this case was not implemented. A test was designed however to show how it would have been tested.

Use case ID: 10	**Use case name:** Generate 2nd letter
Test number: 1	
Objective: Test the main flow	
Set up: 1. The office manager must be logged in to the system (Username: Manager and Password: Get_it_done). 2. Customer name “Hello Magazine” must be set up. 3. First letter has been generated and sent to the customer. 4. One month passes after first letter is sent and the outstanding payment £187.18 is not covered.	
Expected results: 1. The System suspends customer account name “Hello Magazine” 2. The System alerts User with user type Office Manager and generates second letter to print. 3. The System connects to a printer. 4. The System informs Office Manager the print has been completed. 5. Letter has correct values of customer number (*) and its outstanding payment (*)	
Test: 1. Receive alert and notification of 2nd letter generated. 2. User confirms the print. The system connects to the printer and prints the letter with correct customer name “Hello Magazine” and outstanding payment £187.18	
Test record: N/A	
Date: xx/xx/xxxx	**Tester:** xxxxxxx
Result: Unable to carry out tests	
Date: xx/xx/xxxx	**Tester:** xxxxxxx
Result: Unable to carry out tests	

Alternative Flow:

Use case ID: N/A	**Use case name:** Create 2nd Letter: No Printer Connection
Test number:	
Objective: Test the System's response to lack of printer connection.	
Set up: 1. Customer number (*) must be set up. First letter has been generated and sent to the customer. One month passes after first letter is sent and the outstanding payment (*) is not covered. There is no communication channel between terminal and printer.	
Expected results : 1. The System suspends customer account number (*) 2. The System alerts User with user type Office Manager and generates second letter to print. 3. The System informs User that there is no communication channel to the printer.	
Test: 1. Check whether customer account number (69420) is marked as suspended. Log in to user type Office Manager, receive alert and notification of 2nd letter generated. User confirms the print. The System informs user that there is no communication to the printer.	
Test record: N/A	
Date: xx/xx/xxxx	**Tester:** xxxxxxx
Result: Unable to carry out tests	
Date: xx/xx/xxxx	**Tester:** xxxxxxx
Result: Unable to carry out tests	

Non-Functional Testing

We have chosen to test Security and Reliability as the two non-functional requirements for our system. We have included 2 volere templates to display testing related to the BAPERS system.

Security

Application security testing is a process that verifies that the information system protects the data and maintains its intended functionality. It is one of the most important tasks as cyber attacks are on the rise.

Requirement #: NF1	**Requirement Type:** NFR	**Event/use case #:** 1
Description: The System shall be resistant to Brute-Force attacks.		
Rationale: Brute-Force attacks can result in overloading the system, thus crashing it (DoS), or even granting access to unauthorized users.		
Originator: Liu Liu Megatech		
Fit Criterion: For each account 1. BAPERS will shutdown after 3 consecutive attempts of entering incorrect passwords		
Customer Satisfaction: N/A	**Customer Dissatisfaction:** N/A	
Priority: High	**Conflicts:** N/A	
Supporting Materials: Non-functional testing.docx		
History: V.1		

Reliability

Reliability testing is a type of testing to verify that software is capable of performing without failures for a specified period of time, which is crucial for any system.

<table>
<tr><td>Requirement #: NF2</td><td>Requirement Type: NFR</td><td>Event/use case #: 1</td></tr>
<tr><td colspan="3">Description: Reliability of the BAPERS system using Feature testing</td></tr>
<tr><td colspan="3">Rationale: Feature testing must be performed to ensure all operations work successfully.</td></tr>
<tr><td colspan="3">Originator: Liu Liu Megatech</td></tr>
<tr><td colspan="3">Fit Criterion:
1. Each operation in the software is executed at least once without failures
2. Each operation has to be checked for its proper execution.
3. Operations that are dependent on each other should execute without failure

For example Logging in with a new user account(username: Test, Password: Test1) that has been created by the office manager.

4. Ensure different user accounts can access the system making changes to the same data at the same time

For example Office Manager and Copy room Technician want to make changes to an existing job with a set of tasks. Changes that will be made shall allow both users to see this concurrently</td></tr>
<tr><td colspan="2">Customer Satisfaction: N/A</td><td>Customer Dissatisfaction: N/A</td></tr>
<tr><td colspan="2">Priority: High</td><td>Conflicts: N/A</td></tr>
<tr><td colspan="3">Supporting Materials: Non-functional testing.docx</td></tr>
<tr><td colspan="3">History: V.1</td></tr>
</table>