IMPLEMENTATION rEPORT

Liu liu Megatech

Team 6

Contents

[Compilation 3](#_Toc512214691)

[Run Time 4](#_Toc512214692)

[Testing 5](#_Toc512214693)

[Unit Testing 5](#_Toc512214694)

[Upgrade Customer Account 5](#_Toc512214695)

[Print Late Payment Reminder 6](#_Toc512214696)

[Login 7](#_Toc512214697)

[Generate Reports 8](#_Toc512214698)

[Update Job Status 9](#_Toc512214699)

[Record Payment 10](#_Toc512214700)

[Generate 2nd Letter 12](#_Toc512214701)

[Add User 13](#_Toc512214702)

[Automatic Backup 14](#_Toc512214703)

[Update Existing Task 15](#_Toc512214704)

[Non-Functional Testing 17](#_Toc512214705)

[Security 17](#_Toc512214706)

[Reliability 18](#_Toc512214707)

# 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. 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 persitence 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 utilizing the eclipselink implementation of jpa.

# Run Time

Component diagram goes here

# 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: Pass** | |
| **Date:18/04/2018** | **Tester: Gerda** |
| **Result: Pass** | |

**Alternative Flow:**

|  |  |
| --- | --- |
| **Use case ID:** 13 | **Use case name:** Upgrade Customer Account |
| **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: Pass** | |
| **Date: 18/04/2018** | **Tester: Gerda** |
| **Result: Pass** | |

## Print Late Payment Reminder

**Main Flow:**

**This case was not implemented**

|  |  |
| --- | --- |
| **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. 4. GUI appears which the Office Manger clicks. | |
| **Test record:** A late payment letter is printed for Eva Bauyers after the GUI is clicked. | |
| **Date:xx/xxx/xxx** | **Tester:xxxxxxx** |
| **Result:** | |
| **Date:** | **Tester:** |
| **Result:** | |

**Alternative Flow:**

|  |  |
| --- | --- |
| **Use case ID:** 7 | **Use case name:** Print Late Payment Reminder |
| **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, accesses the customer list and sees that John Doe has missed his payment deadline. 2. GUI appears which the Office Manger clicks. | |
| **Test record:** A late payment letter is not printed for Eva Bauyer due to a printer error after the GUI is clicked. | |
| **Date:xx/xxx/xxx** | **Tester:xxxxxxx** |
| **Result:** | |
| **Date:** | **Tester:** |
| **Result:** | |

## 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: Pass** | |
| **Date: 18/04/2018** | **Tester: Gerda** |
| **Result: Pass** | |

**Alternative Flow:**

|  |  |
| --- | --- |
| **Use case ID:** 5 | **Use case name:** Login |
| **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: Pass** | |
| **Date: 18/04/2018** | **Tester: Gerda** |
| **Result: Pass** | |

## 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). | |
| **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, and enters the first and surname of the employee that he/she wants the report **for (i.e. ).** 3. To generate the “Summary performance report” he/she selects that radio button, selects the start and end dates. 4. To generate the “individual report” he/she selects that radio button, selects the start and end dates, 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: Pass** | |
| **Date: 18/04/2018** | **Tester: Gerda** |
| **Result: Pass** | |

## 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: Pass** | |
| **Date: 17/04/2018** | **Tester: Gerda** |
| **Result: Pass** | |

## Record Payment

**Main flow:**

|  |  |
| --- | --- |
| **Use case ID:** 1 | **Use case name:** Record Payment |
| **Test number: 1** | |
| **Objective:** Test the primary path | |
| **Set up:** Customer name “Hello Magazine” must be set up and have outstanding balance of £187.18 of unpaid jobs. Date 2018/04/15 | |
| **Expected results:**   1. Payment is stored in the Database. 2. Payment amount deducts 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: 2018/04/16** | **Tester: Patrick G** |
| **Result: Passed** | |
| **Date:** | **Tester:** |
| **Result:** |  |

**Alternative flow:**

|  |  |
| --- | --- |
| **Use case ID:** 2 | **Use case name:** Record Payment: CardPayment |
| **Test number:** **1** | |
| **Objective:** Test the alternative path | |
| **Set up:** Customer name “Hello Magazine” must be set up and have outstanding balance of £187.18 of unpaid jobs. Customer card last 4 digits 9090. Expiry 01/20. (MasterCard). Date 2018/04/15 | |
| **Expected results:**   1. Payment is stored in the Database. 2. Payment card details are stored in the Database. 3. Payment amount deducts 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: 2018/04/16** | **Tester: Patrick G** |
| **Result: Passed** | |
| **Date:** | **Tester:** |
| **Result:** |  |

## Generate 2nd Letter

**Main Flow:**

**Not sure whether to add**

|  |  |
| --- | --- |
| **Use case ID:** 10 | **Use case name:** Generate 2nd letter |
| **Test number:** | |
| **Objective:** Test the main path. | |
| **Set up:** 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 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 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. Customer account number (\*) 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 connects to the printer and prints the letter with correct customer number (\*) and outstanding payment (\*) | |
| **Test record:** | |
| **Date:** | **Tester:** |
| **Result:** | |
| **Date:** | **Tester:** |
| **Result:** |  |

**Alternative Flow:**

|  |  |
| --- | --- |
| **Use case ID:** N/A | **Use case name:** Create 2nd Letter: NoPrinterConnection |
| **Test number:** | |
| **Objective:** Test the System’s response to lack of printer connection. | |
| **Set up:** 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:** | |
| **Date:** | **Tester:** |
| **Result:** | |
| **Date:** | **Tester:** |
| **Result:** |  |

## Add User

**Main Flow:**

|  |  |
| --- | --- |
| **Use case ID:** 3 | **Use case name:** Add User |
| **Test number: 1** | |
| **Objective:** Test the functionality of adding a user. | |
| **Set up:** User Manager with user type Office Manager is logged in. | |
| **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. 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: 2018/04/16** | **Tester: Patrick G** |
| **Result: Passed** | |
| **Date:** | **Tester:** |
| **Result:** |  |

**Alternative Flow:**

|  |  |
| --- | --- |
| **Use case ID:** **4** | **Use case name:** Add User: AlreadyExists |
| **Test number:** 2 | |
| **Objective:** Test the System’s response to adding user of the same username | |
| **Set up:** User Manager with user type Office Manager is logged in. User with username Manager exists. | |
| **Expected results:**   1. The System informs the user that there already is user with the same username | |
| **Test:**   1. Select “User” functionality, 2. Enter details for the user account (Username: Manager, Password: Random123, User type: Shift Manager, User Location: Copy room). 3. The System informs that there already is user with the same username. | |
| **Test record: Alert comes up, user account has not been created** | |
| **Date: 2018/04/16** | **Tester: Patrick G** |
| **Result: Passed** | |
| **Date:** | **Tester:** |
| **Result:** |  |

## Automatic Backup

**Main Flow:**

|  |  |
| --- | --- |
| **Use case ID:** **5** | **Use case name:** Automatic backup |
| **Test number:** 1 | |
| **Objective:** Test the automatic backup functionality. | |
| **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. Specified time occurs (Each day), the System automatically creates new database backup. | |
| **Test record: Backup occurs each day.** | |
| **Date: 2018/04/16** | **Tester: Patrick G** |
| **Result: Passed** | |
| **Date:** | **Tester:** |
| **Result:** |  |

## Update Existing Task

**Main Flow:**

|  |  |
| --- | --- |
| **Use case ID:** **6** | **Use case name:** Update existing task |
| **Test number: 1** | |
| **Objective:** Test the “Update existing task” functionality. | |
| **Set up:** Office Manager (Username: Manager) is logged in. 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: 2018/04/16** | **Tester: Patrick** |
| **Result: Passed** | |
| **Date:** | **Tester:** |
| **Result:** |  |

**Alternative flow:**

|  |  |
| --- | --- |
| **Use case ID:** **7** | **Use case name:** Update existing task: NoUser |
| **Test number:** 2 | |
| **Objective:** Test the System’s response to updating task with unexisting user. | |
| **Set up:** Office Manager is logged in (Manager). Unexisting 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: 2018/04/16** | **Tester: Patrick** |
| **Result: Passed** | |
| **Date:** | **Tester:** |
| **Result:** |  |

# Non-Functional Testing

We have chosen to test Security and Reliability as the two non-functional requirements for our system.

## Security

|  |  |  |  |
| --- | --- | --- | --- |
| Requirement #: NF1 | Requirement Type: NFR | | Event/use case #: |
| 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: ? | | | |
| Fit Criterion: For each account, if password is entered incorrectly more than 3 times:   1. Account is locked out for 3min. 2. IP from which account was trying to log in is blocked for 3min. | | | |
| Customer Satisfaction: N/A | | Customer Dissatisfaction: N/A | |
| Priority: High | | Conflicts: N/A | |
| Supporting Materials: Non-functional testing.docx | | | |
| History: V.1 | | | |

## Reliability

|  |  |  |  |
| --- | --- | --- | --- |
| Requirement #: NF2 | Requirement Type: NFR | | Event/use case #: |
| Description: Reliability of the BAPERS system using Feature testing | | | |
| Rationale: Feature testing must be performed to ensure all operations work successfully. | | | |
| Originator: ? | | | |
| 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. | | | |
| Customer Satisfaction: N/A | | Customer Dissatisfaction: N/A | |
| Priority: High | | Conflicts: N/A | |
| Supporting Materials: Non-functional testing.docx | | | |
| History: V.1 | | | |