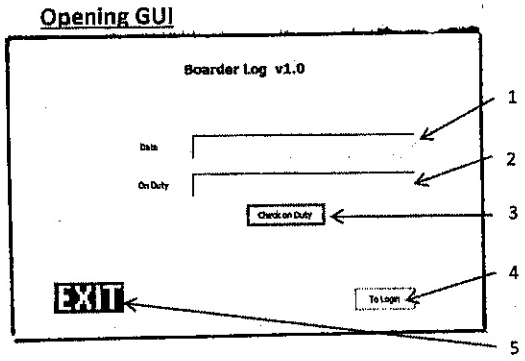
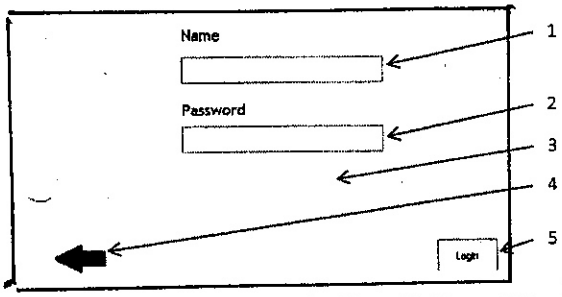
1. **USER INTERFACE DESIGN**

Opening GUI



Description	Displays the date and faculty member on duty.
Security Group	All users
Data	1. Displays the date 2. Displays the Faculty member on duty and their contact (phone) number.
Actions	3. Checks/returns the date and returns the faculty member that is on duty. 4. Moves to the login GUI. 5. Closes the program.

Login GUI

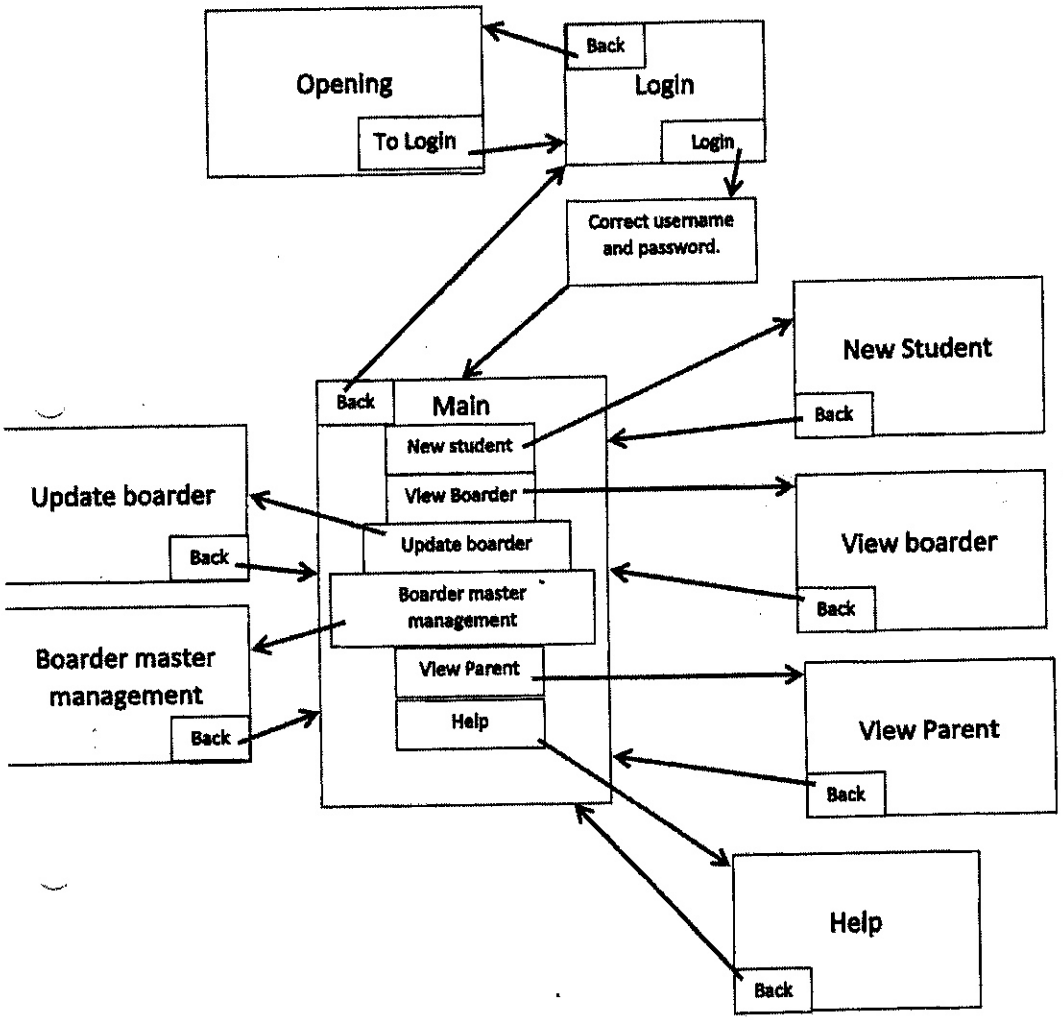


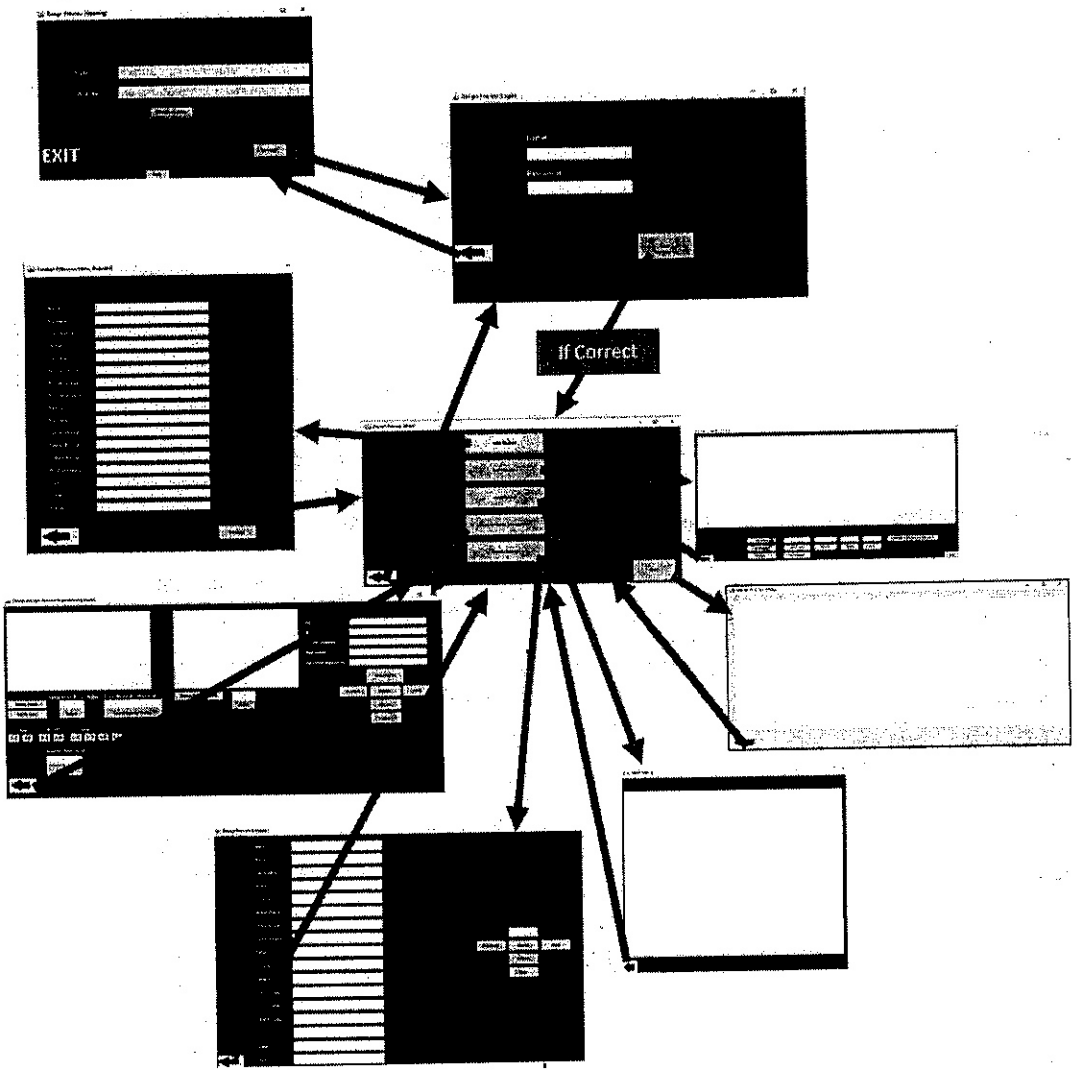
Description	User enters their username and password, program checks if they match any in the database.
Security Group	All Users
Data	1. User enters their username. 2. User enters their password. 3. Displays Error message if incorrect username or password is entered.
Actions	4. Opens the Opening GUI. 5. Checks if the username and password are in the database. If correct, opens the Main GUI. If incorrect, displays error message

2.4) Sequencing

This section describes the flow of events in your program. A flowchart is often the easiest way of representing sequencing in a program; you could also use pseudocode or linked GUI screenshots. Essentially what actions on a screen lead to the opening up of other screen – is the flow logical and useful.? Overleaf are some examples from other projects that you may find useful.

2. SEQUENCING





2.5) Class Design

This section deals with the class design of your NON-INTERFACE CLASSES ie the design of your object classes, array classes and database class (which ever is relevant). Each of the classes you list must have the following information:

- Class name
- Attributes (including types and access modifiers)
- Methods (including parameters, return types and access modifiers)

You must not list your GUI classes in this section. Here only your working code is important. Here you should have a method for the features listed in your System Specification Document. For example if you have a feature called "Add a Client" then you will have a method somewhere in this section called 'addNewClient" which takes the details of the new client as its parameters. This section is important and is worth a significant amount of marks.

Example:

Class - Student

Fields	**Description**
- int ID	Unique number assigned to student
- String firstName	Stores' a student's first name
- String lastName	Stores' a student's last name
- String mother	Stores' a student's mothers name
etc	etc
Methods	**Description**
+ getID( )	Returns student's unique ID
+ String getName( )	Returns student's
+ String getGrade( )	Returns student's grade
+ String getFather( )	Returns student's father's name
+ String getMedical ( )	Returns student's medical issues
+ String toString( )	Returns selected concatenated student fields as per the design of the toString method.
+ public String toStringns( )	Returns student's information in a form that can be easily used anywhere in the program

Array class - Student Manager

Fields	**Description**
- Student[ ] learner = new Student [1000]	Array of objects of all students
- String heading	Stores the heading in the display
- int size	Stores the number of students in the array
- listsize	Stores the number of students in the array that is currently being displayed.
etc	etc
Methods	
+ String [ ] displayFields ( )	Returns an array with all the students and their information.
+ String [ ] getStud(int studid)	Receives an ID, searches for the student with the same ID and returns their information in an array.
+ String [ ] getStudName (String studname)	Receives a name or part of a name and returns all the students that have the same characters as the received name. Returns an array.
+ String [ ] sortGrade(int grade)	Receives a number and returns all the students that are in that grade in an array.
+ void makeRegister( )	Writes all the students names and grades into a text file for printing
+ int getListSize( )	Returns the number of students.

Class Father

Fields	**Description**
- int ID	Unique number assigned to student
- String firstName	Stores' fathers first name
- String lastName	Stores' a fathers last name
- String address1	Stores' part one of the father's address
- String address2	Stores' part two of the father's address
etc	etc
Methods	**Description**
+ getID( )	Returns student's unique ID
+ String getFathersName( )	Returns father's name
+ String getFatherCell( )	Returns father's cell phone number
+ String toString( )	Returns father's information
etc	etc

Array Class – Father manager

Fields	**Description**
- Father[ ] father = new Father [1000]	Array of objects of all Fathers
- String heading	Stores the heading in the display
- int size	Stores the number of fathers in the array
etc	etc
Methods	
+ String [ ] getID (int studid)	Returns the information of a father that matches an ID
+ String displayFields( )	Returns all the information of a father
etc	etc

2.6) Persistent Storage Design

Here you provide a detailed account of your storage design i.e. your database and or text files.

Database - The tables your database is going to consist of . . .
For each table provide

- Field name and types
- Primary key
- Foreign key
- Any relevant formatting of fields in your tables
- Relationship diagram (showing links between tables)
- Sample data for each table

Text files – if you are using text files to store data provide the following

- Names of the text files
- Where they are stored in the folder structure
- Format of the text files including relevant delimiters
- Sample data for each type of text file

Examples:

Database:
Tables:
Student:
Primary key *

Field name	Data Type	Description
StudentID *	AutoNumber	Unique number that identifies a record
FirstName	Short Text	First name of a student
LastName	Short Text	Last name of a student
Grade	Number	The grade of a student
Mother	Short Text	The name of the students mother
Father	Short Text	The name of the students father
House	Short Text	School house of the student
etc	etc	etc

Database:
Tables:
Father:
Foreign Key **

Field name	Data Type	Description
StudentID **	AutoNumber	Student's number
FatherFirstName	Short Text	First name of the student's father
FatherLastName	Short Text	Last name of the student's father
Address1	Number	Address line one of the father
Address2	Short Text	Address line two of the father
CellNumber	Short Text	Father's cell phone number
Email	Short Text	Father's email address
etc	etc	etc

Text Files – some examples and how to record them

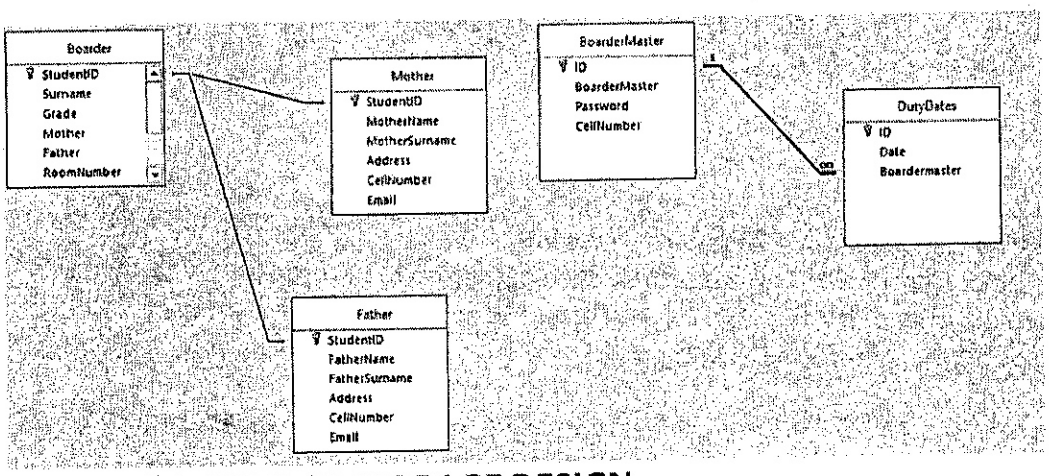
Names:

- Stored in package Pat2018JoeSoap
- Contains students details separted by the # delimiter

Help

- Stored in package Pat2018JoeSoap
- Contains text for the help file

Relationships
Take a screen shot of the relationships in your database
Example:



EXPLANATION OF STORAGE DESIGN

2.7) Explanation of storage design
The reasons behind your storage design. What were the advantages of using a database in your program? If you used a text file, what advantages did it bring?

What type of data do you need to write to the database and what type of data can merely remain in RAM?

Examples: You would write a new student to the database but the results of a name search would merely remain in RAM for temporary viewing.

Document 3: Technical and Testing Document
In this document you have your actual Java code with explanations. Also the results of testing your program when bad data.

1.1 Cover
- Performance Assessment Task
- Technical and Testing Document
- Your full name
- Your grade
- The year.

1.2 Table of contents

1.3) Externally sourced code
You may have used specialty code that you did not write yourself – this is the place to declare it.
Examples
- Connect class. Used to connect to the database. Supplied.