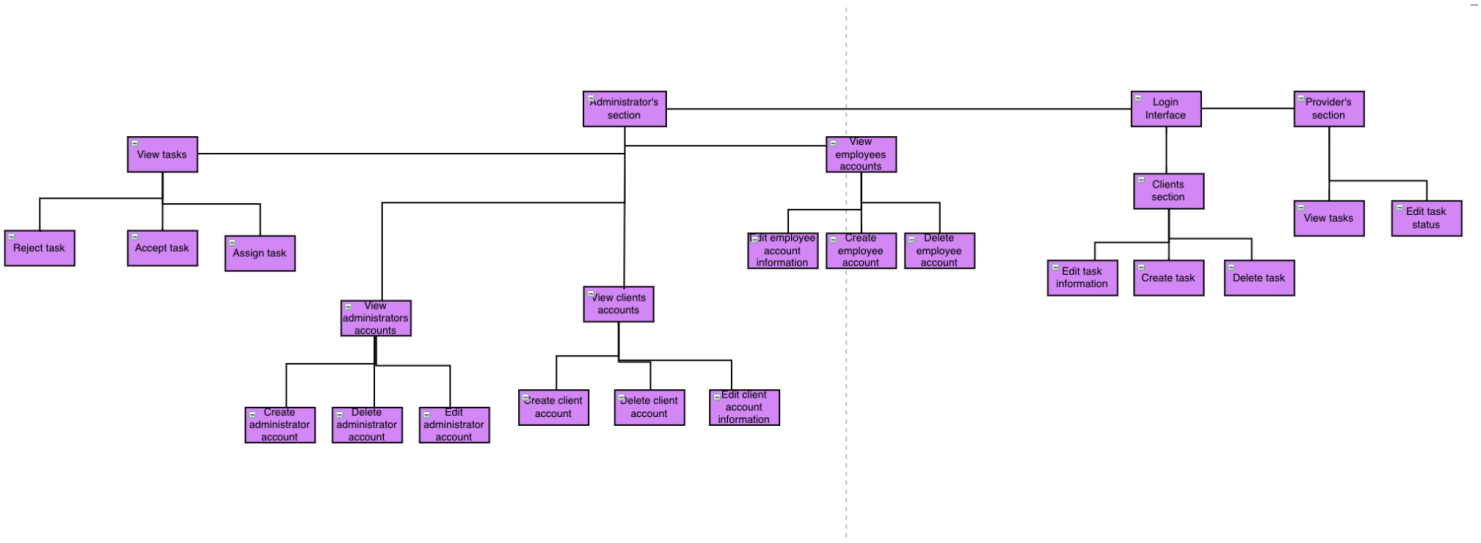
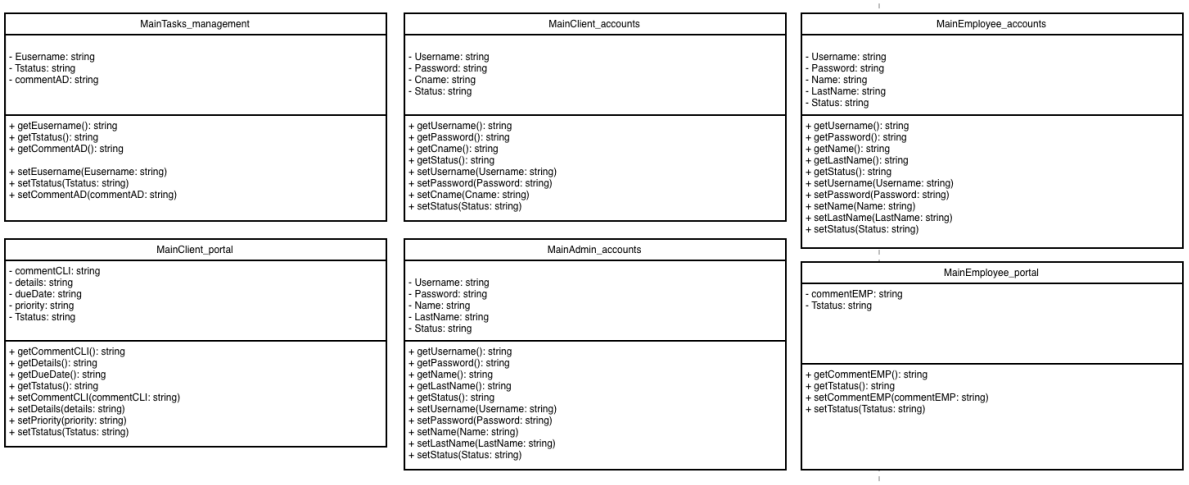
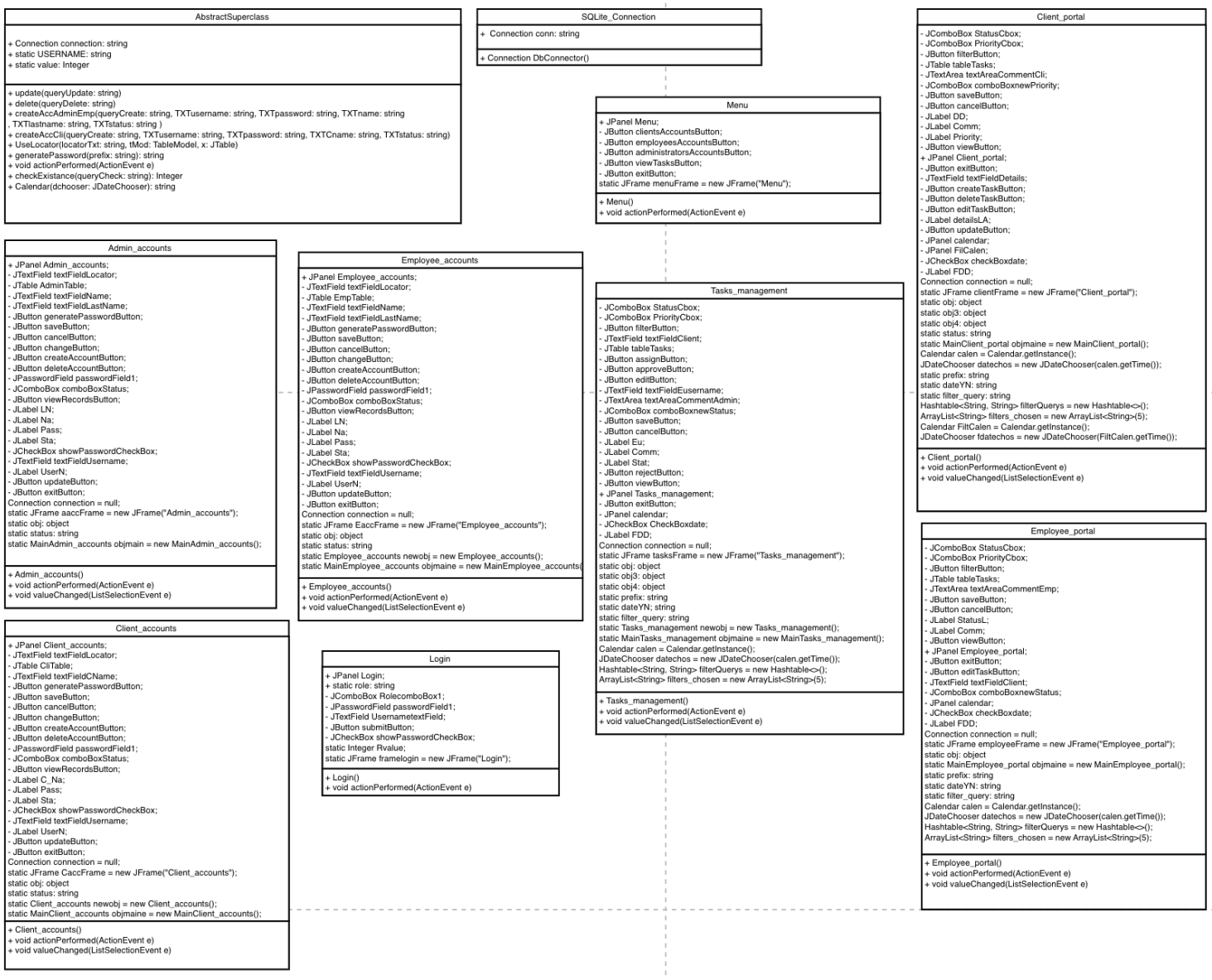
Criterion B: Design Overview

Organisation chart:

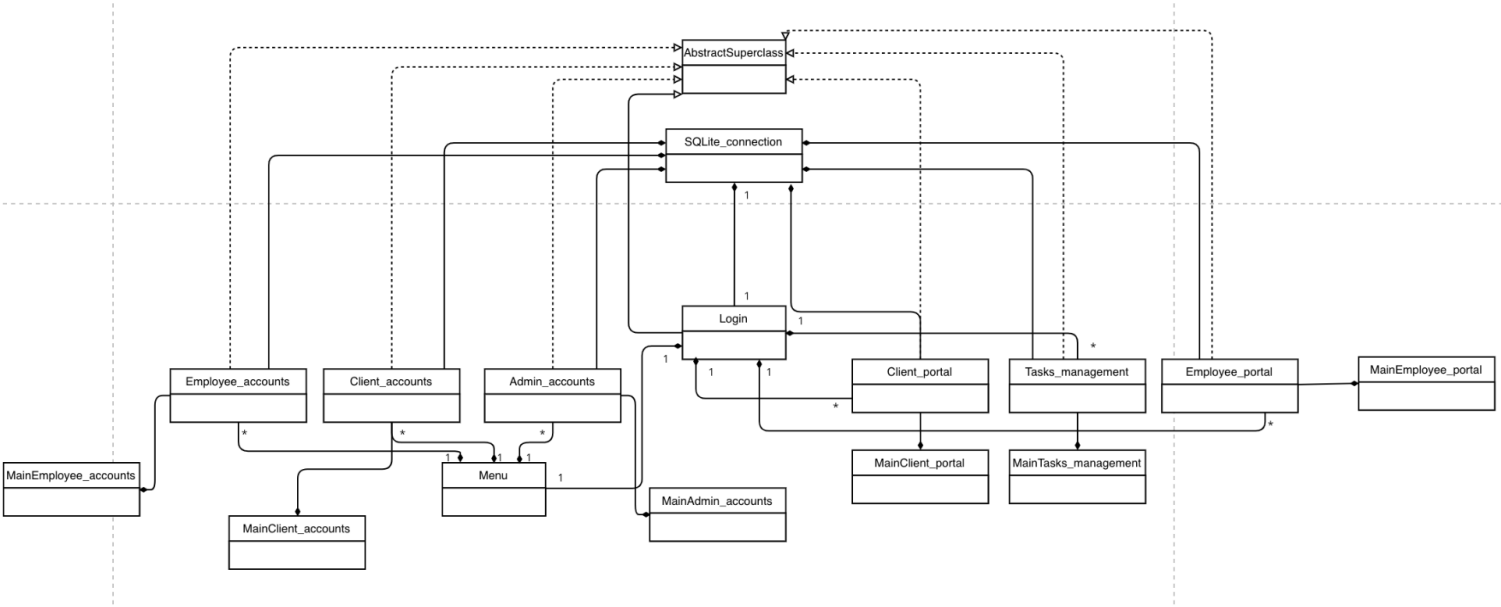


UML charts:




AbstractSuperclass
+ Connection connection: string
+ static USERNAME: string
+ static value: Integer
+ update(queryUpdate: string)
+ delete(queryDelete: string)
+ createAccAdminEmp(queryCreate: string, TXTusername: string, TXTpassword: string, TXTname: string
, TXTlastname: string, TXTstatus: string)
+ createAccCli(queryCreate: string, TXTusername: string, TXTpassword: string, TXTCname: string, TXTstatus: string)
+ UseLocator(locatorTxt: string, tMod: TableModel, x: JTable)
+ generatePassword(prefix: string): string
+ void actionPerformed(ActionEvent e)
+ checkExistance(queryCheck: string): Integer
+ Calendar(dchooser: JDateChooser): string
SQLite_Connection
+ Connection conn: string
+ Connection DbConnector()
Client_portal
- JComboBox StatusCbox;
- JComboBox PriorityCbox;
- JButton filterButton;
- JTable tableTasks;
- JTextArea textAreaCommentCli;
- JComboBox comboBoxnewPriority;
- JButton saveButton;
- JButton cancelButton;
- JLabel DD;
- JLabel Comm;
- JLabel Priority;
- JButton viewButton;
+ JPanel Client_portal;
- JButton exitButton;
- JTextField textFieldDetails;
- JButton createTaskButton;
- JButton deleteTaskButton;
- JButton editTaskButton;
- JLabel detailsLA;
- JButton updateButton;
- JPanel calendar;
- JPanel FilCalen;
- JCheckBox checkBoxdate;
- JLabel FDD;
Connection connection = null;
static JFrame clientFrame = new JFrame("Client_portal");
static obj: object
static obj3: object
static obj4: object
static status: string
static MainClient_portal objmaine = new MainClient_portal();
Calendar calen = Calendar.getInstance();
JDateChooser datechos = new JDateChooser(calen.getTime());
static prefix: string
static dateYN: string
static filter_query: string
Hashtable<String, String> filterQuerys = new Hashtable<>();
ArrayList<String> filters_chosen = new ArrayList<String>(5);
Calendar FiltCalen = Calendar.getInstance();
JDateChooser fdatechos = new JDateChooser(FiltCalen.getTime());
+ Client_portal()
+ void actionPerformed(ActionEvent e)
+ void valueChanged(ListSelectionEvent e)
Menu
+ JPanel Menu;
- JButton clientsAccountsButton;
- JButton employeesAccountsButton;
- JButton administratorsAccountsButton;
- JButton viewTasksButton;
- JButton exitButton;
static JFrame menuFrame = new JFrame("Menu");
+ Menu()
+ void actionPerformed(ActionEvent e)
Admin_accounts
+ JPanel Admin_accounts;
- JTextField textFieldLocator;
- JTable AdminTable;
- JTextField textFieldName;
- JTextField textFieldLastName;
- JButton generatePasswordButton;
- JButton saveButton;
- JButton cancelButton;
- JButton changeButton;
- JButton createAccountButton;
- JButton deleteAccountButton;
- JPasswordField passwordField1;
- JComboBox comboBoxStatus;
- JButton viewRecordsButton;
- JLabel LN;
- JLabel Na;
- JLabel Pass;
- JLabel Sta;
- JCheckBox showPasswordCheckBox;
- JTextField textFieldUsername;
- JLabel UserN;
- JButton updateButton;
- JButton exitButton;
Connection connection = null;
static JFrame aaccFrame = new JFrame("Admin_accounts");
static obj: object
static status: string
static MainAdmin_accounts objmain = new MainAdmin_accounts();
+ Admin_accounts()
+ void actionPerformed(ActionEvent e)
+ void valueChanged(ListSelectionEvent e)
Employee_accounts
+ JPanel Employee_accounts;
- JTextField textFieldLocator;
- JTable EmpTable;
- JTextField textFieldName;
- JTextField textFieldLastName;
- JButton generatePasswordButton;
- JButton saveButton;
- JButton cancelButton;
- JButton changeButton;
- JButton createAccountButton;
- JButton deleteAccountButton;
- JPasswordField passwordField1;
- JComboBox comboBoxStatus;
- JButton viewRecordsButton;
- JLabel LN;
- JLabel Na;
- JLabel Pass;
- JLabel Sta;
- JCheckBox showPasswordCheckBox;
- JTextField textFieldUsername;
- JLabel UserN;
- JButton updateButton;
- JButton exitButton;
Connection connection = null;
static JFrame EaccFrame = new JFrame("Employee_accounts");
static obj: object
static status: string
static Employee_accounts newobj = new Employee_accounts();
static MainEmployee_accounts objmaine = new MainEmployee_accounts(
+ Employee_accounts()
+ void actionPerformed(ActionEvent e)
+ void valueChanged(ListSelectionEvent e)
Tasks_management
- JComboBox StatusCbox;
- JComboBox PriorityCbox;
- JButton filterButton;
- JTextField textFieldClient;
- JTable tableTasks;
- JButton assignButton;
- JButton approveButton;
- JButton editButton;
- JTextField textFieldEusername;
- JTextArea textAreaCommentAdmin;
- JComboBox comboBoxnewStatus;
- JButton saveButton;
- JButton cancelButton;
- JLabel Eu;
- JLabel Comm;
- JLabel Stat;
- JButton rejectButton;
- JButton viewButton;
+ JPanel Tasks_management;
- JButton exitButton;
- JPanel calendar;
- JCheckBox CheckBoxdate;
- JLabel FDD;
Connection connection = null;
static JFrame tasksFrame = new JFrame("Tasks_management");
static obj: object
static obj3: object
static obj4: object
static prefix: string
static dateYN: string
static filter_query: string
static Tasks_management newobj = new Tasks_management();
static MainTasks_management objmaine = new MainTasks_management();
Calendar calen = Calendar.getInstance();
JDateChooser datechos = new JDateChooser(calen.getTime());
Hashtable<String, String> filterQuerys = new Hashtable<>();
ArrayList<String> filters_chosen = new ArrayList<String>(5);
+ Tasks_management()
+ void actionPerformed(ActionEvent e)
+ void valueChanged(ListSelectionEvent e)
Employee_portal
- JComboBox StatusCbox;
- JComboBox PriorityCbox;
- JButton filterButton;
- JTable tableTasks;
- JTextArea textAreaCommentEmp;
- JButton saveButton;
- JButton cancelButton;
- JLabel StatusL;
- JLabel Comm;
- JButton viewButton;
+ JPanel Employee_portal;
- JButton exitButton;
- JButton editTaskButton;
- JTextField textFieldClient;
- JComboBox comboBoxnewStatus;
- JPanel calendar;
- JCheckBox checkBoxdate;
- JLabel FDD;
Connection connection = null;
static JFrame employeeFrame = new JFrame("Employee_portal");
static obj: object
static MainEmployee_portal objmaine = new MainEmployee_portal();
static prefix: string
static dateYN: string
static filter_query: string
Calendar calen = Calendar.getInstance();
JDateChooser datechos = new JDateChooser(calen.getTime());
Hashtable<String, String> filterQuerys = new Hashtable<>();
ArrayList<String> filters_chosen = new ArrayList<String>(5);
+ Employee_portal()
+ void actionPerformed(ActionEvent e)
+ void valueChanged(ListSelectionEvent e)
Client_accounts
+ JPanel Client_accounts;
- JTextField textFieldLocator;
- JTable ClTable;
- JTextField textFieldCName;
- JButton generatePasswordButton;
- JButton saveButton;
- JButton cancelButton;
- JButton changeButton;
- JButton createAccountButton;
- JButton deleteAccountButton;
- JPasswordField passwordField1;
- JComboBox comboBoxStatus;
- JButton viewRecordsButton;
- JLabel C_Na;
- JLabel Pass;
- JLabel Sta;
- JCheckBox showPasswordCheckBox;
- JTextField textFieldUsername;
- JLabel UserN;
- JButton updateButton;
- JButton exitButton;
Connection connection = null;
static JFrame CaccFrame = new JFrame("Client_accounts");
static obj: object
static status: string
static Client_accounts newobj = new Client_accounts();
static MainClient_accounts objmaine = new MainClient_accounts();
+ Client_accounts()
+ void actionPerformed(ActionEvent e)
+ void valueChanged(ListSelectionEvent e)
Login
+ JPanel Login;
+ static role: string
- JComboBox RolecomboBox1;
- JPasswordField passwordField1;
- JTextField UsernametextField;
- JButton submitButton;
- JCheckBox showPasswordCheckBox;
static Integer Rvalue;
static JFrame framelogin = new JFrame("Login");
+ Login()
+ void actionPerformed(ActionEvent e)

Relationship between the classes:



Class	Functionality
AbstractSuperclass	Contains most of the functions for the common actions between every type of user of the program. Defines most of the functionalities of the system and is the superclass for many other classes that inherit methods from this class.
Login	Validates account fields and redirects users to their corresponding portals if fields match and exist in the database.
Menu	Displays a menu only for administrators which connects them to other interfaces.
Admin_accounts	Allows administrators to manage the other administrators accounts: edit, create and delete.

MainAdmin_accounts	Encapsulates the fields of the database table Administrators. It gets and sets the values for every variable that will be stored in the database table.
Employee_accounts	Allows administrators to manage employee accounts: create, delete or edit their account information.
MainEmployee_accounts	Encapsulates the fields of the database table Employees. It gets and sets the values for every variable that will be stored in the database table.
Client_accounts	Allows administrators to manage clients accounts: create, delete or edit their account information.
MainClient_accounts	Encapsulates the fields of the database table Clients. It gets and sets the values for every variable that will be stored in the database table.
SQLite_connection	Connects all the classes(front-end) with the database(back-end).
ClientPortal	Provides an interface for the clients to create, edit and delete a task.
MainClient_portal	Encapsulates the fields required by the client to fill of the database table Tasks. It gets and sets the values for every variable that will be stored in the database table.
Tasks_management	Allows administrators to accept or reject a task, assign them and comment on it.
Employee_portal	Provides a panel for employees to comment and change the status of the task that was assigned to them.
MainEmployee_portal	Encapsulates the fields required by the employee to fill of the database table Tasks. It gets and sets the values for every variable that will be stored in the database table.
MainTasks_management	Encapsulates the fields required by the administrator to fill of the database table

	Tasks. It gets and sets the values for every variable that will be stored in the database table.

The task-organiser input and output tables:

Input tables:

Note: Due date field does not need to be validated because of the implementation of calendars, which restrict users from entering abnormal information.

1. Clients interface (task creation)	
Ex. Normal data (accepted data) Due date: 17/6/2022 Details: I would like you to add a new algorithm that allows us to find the most efficient route with/without tolls. Comment: This task can be done through any method; there isn't any regulation on how to perform the requirement. Priority: 3 **Accepted client entry:** Length check (length of details and comment fields >50 characters). Presence check (all fields were completed).	**Ex. Abnormal data (rejected data)** Due date: 15/8/2022 Details: Comment: hello@ Priority: 1 **Rejected client entry:** Presence check (details field is empty). Length check (comment field entry length <50--- detail is lacking for employee comprehension).
Limitations: Tasks may only be created one at a time per client account.	

2. Administrators interface (assignation of task)	
Ex. Normal data (accepted data) Employee's username: @javiguerrero Comment: Please try to complete the task by the due date. If you have any concerns or need aid please contact me in the office. **Accepted administrator entry:** Written username exists in database Presence check (all fields were completed).	**Ex. Abnormal data (rejected data)** Employees username: @12hihi Comment: **Rejected administrator entry:** Username doesn't exist in the database. Presence check (comment field is empty).
Limitations: Tasks can only be assigned to one employee since the relationship between the "Tasks" database table and the "Employees" table is many to 1 respectively.	

3. Administrators interface (Creation of a client account)

Ex. Normal data (accepted data) Company name: Walmart Username: @walmartcom Password: CLI457 **Accepted administrator entry:** Username is unique in the database. Presence check (all fields were completed). Format check (username contains the compulsory character “@” and the password contains the adequate prefix, “CLI”, for the role of the account).	**Ex. Abnormal data (rejected data)** Company name: Username: walmartcom Password: 123 **Rejected administrator entry:** Presence check (fields are empty). Format check (username doesn’t contain the compulsory character “@” and password doesn’t have the corresponding prefix, “CLI”).
Limitations: Usernames need to be unique due to the fact that this field is the primary key of every record in the database tables.	

4. Administrators interface (Creation of an employee account)	
Ex. Normal data (accepted data) Name: Jorge Last Name: Gine Username: @jorgegine Password: EMP457 **Accepted administrator entry:** Username is unique in the database. Presence check (all fields were completed). Format check (username contains the compulsory character “@” and the password contains the adequate prefix, “EMP”, for the role of the account).	**Ex. Abnormal data (rejected data)** Name: Last Name: Username: jorgegine Password: 123 **Rejected administrator entry:** Presence check (fields are empty). Format check (username doesn’t contain the compulsory character “@” and password doesn’t have the corresponding prefix, “EMP”).
Limitations: Usernames need to be unique due to the fact that this field is the primary key of every record in the database tables.	

5. Administrators interface (Creation of an administrator account)	
Ex. Normal data (accepted data) Name: Paul Last Name: Lerra Username: @paulerra Password: ADMIN457 **Accepted administrator entry:** Username is unique in the database. Presence check (all fields were completed). Format check (username contains the compulsory character “@” and the password contains the adequate prefix,	**Ex. Abnormal data (rejected data)** Name: Last Name: Username: paulerra Password: 34 **Rejected administrator entry:** Presence check (fields are empty). Format check (username doesn’t contain the compulsory character “@” and password doesn’t have the corresponding prefix, “ADMIN”).

“ADMIN”, for the role of the account).	
Limitations: Usernames need to be unique due to the fact that this field is the primary key of every record in the database tables.	

Output tables:

1. **Successful creation and assignation of a task**
The program will notify the client by displaying a message indicating that the task has been created successfully and that it will be shortly approved or rejected. Additionally, the task is saved in the database table tasks in a record with the company name, due date, priority, comments from every entity, assignation username, and default status of “pending”. Furthermore, the task is given a number according to the respective order of generation in the entire system.

2. **Successful assignation of a task**
The program will notify the administrator by displaying a message indicating that the task has been assigned successfully. Furthermore, the record in the tasks database table will be updated since the assignation field will be filled with the username of an employee.

3. **Successful account creation (administrator, client or employee account)**
The program will notify the administrator by displaying a message indicating that the account has been created successfully. Furthermore, the information will be saved in the respective database table according to the role selected to the account.

Flowcharts:

Figure 1: Admin portal - Clients’ accounts administration flowchart (create, edit, delete and filter accounts)
Note: This flowchart is the same for the clients, employees, and administrators' accounts. Furthermore, the program performs a logical removal meaning that when an account is deleted it is not removed from the database, but instead have a status setted as “deleted”.

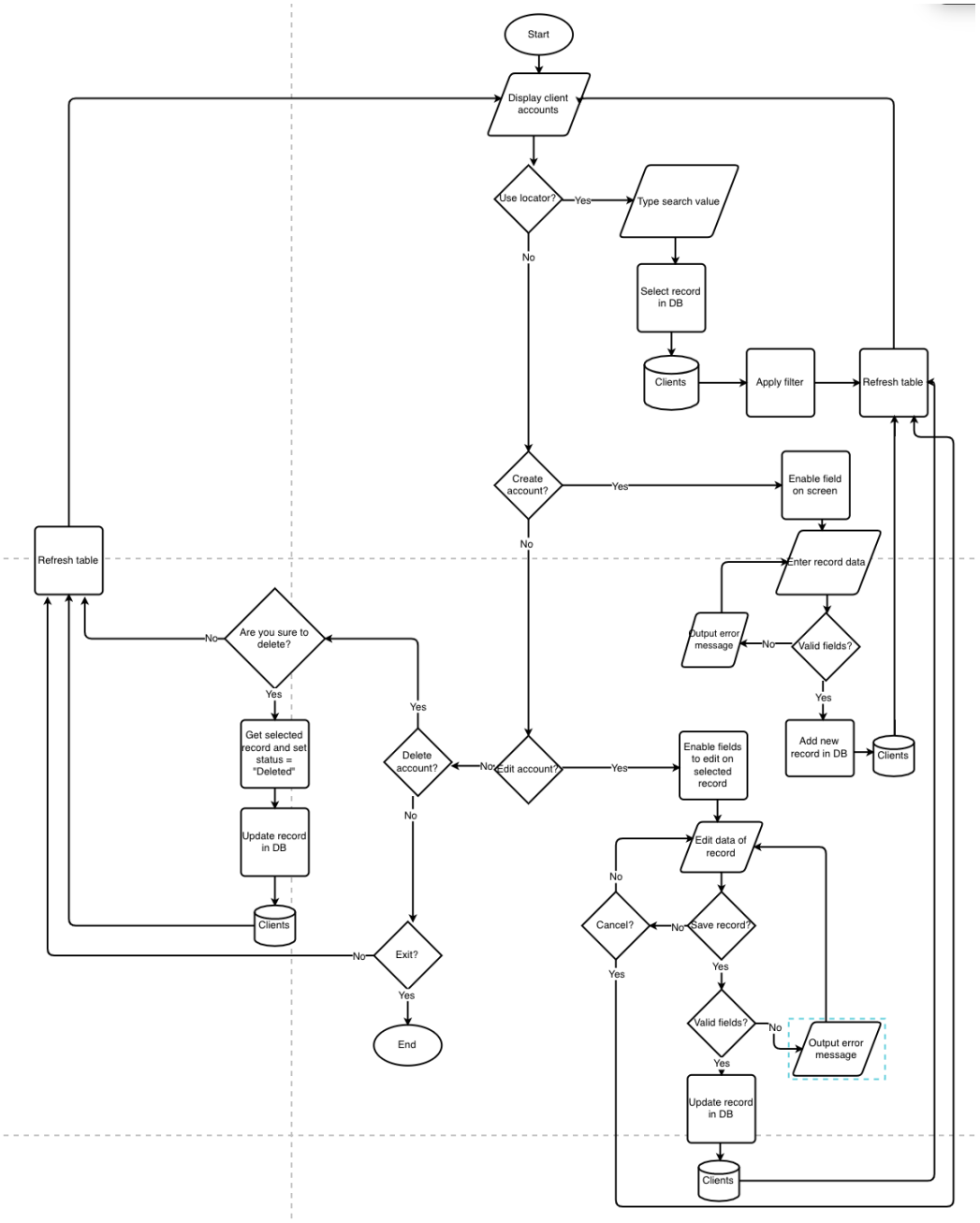


Figure 2: Admin portal - Administrator's accounts administration flowchart (create, edit, delete and filter accounts)

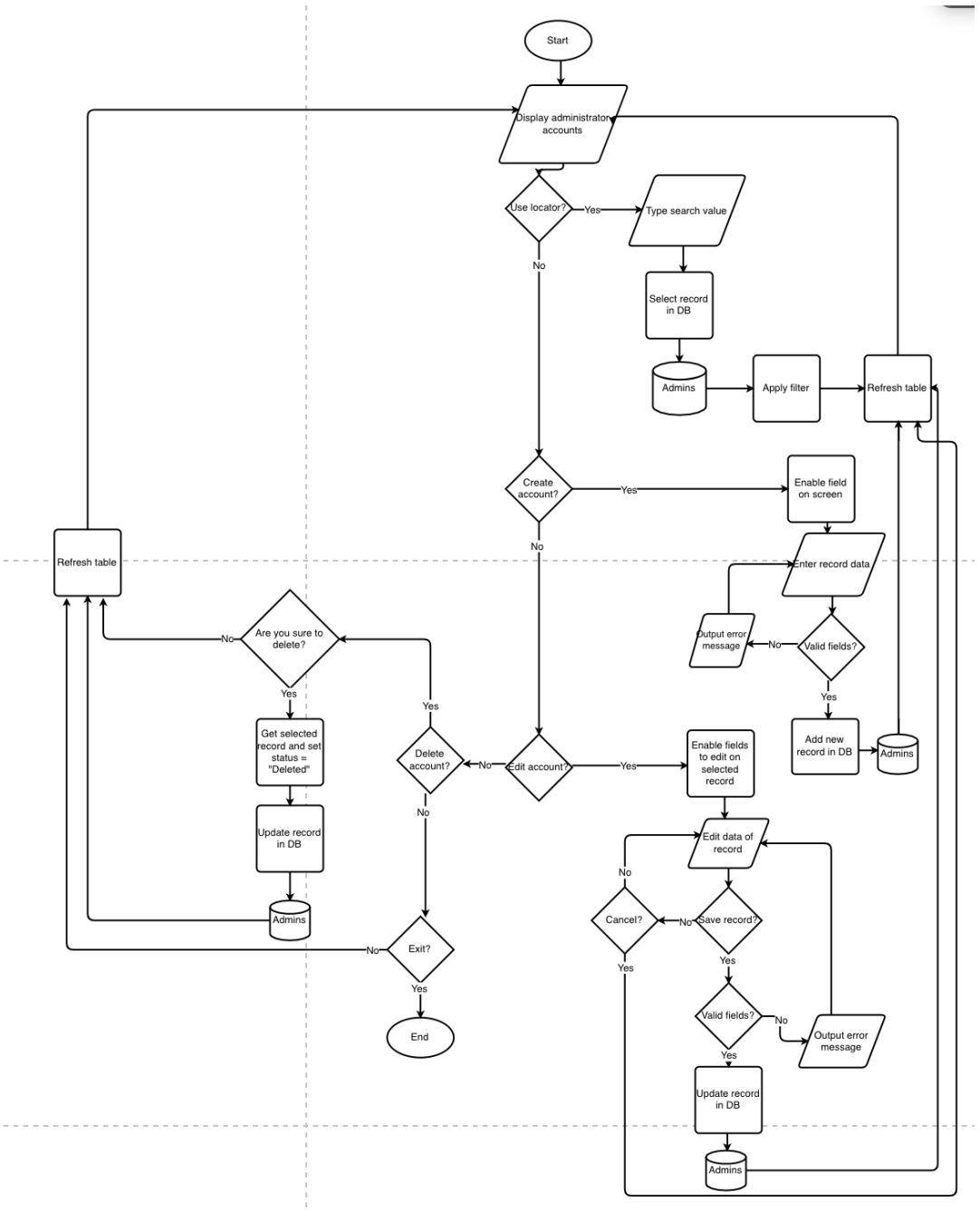

Start
Display administrator accounts
Use locator?
Yes
Type search value
No
Select record in DB
Admins
Apply filter
Refresh table
Create account?
Yes
Enable field on screen
No
Enter record data
Refresh table
Are you sure to delete?
No
Output error message
No
Valid fields?
Yes
Yes
Get selected record and set status = "Deleted"
Yes
Delete account?
No
Edit account?
Yes
Enable fields to edit on selected record
Add new record in DB
Admins
Update record in DB
No
Edit data of record
Admins
Exit?
No
No
Cancel?
No
Save record?
Yes
Yes
Valid fields?
No
Output error message
Yes
End
Update record in DB
Admins

Figure 3: Admin portal - Employees' accounts administration flowchart (create, edit, delete and filter accounts)

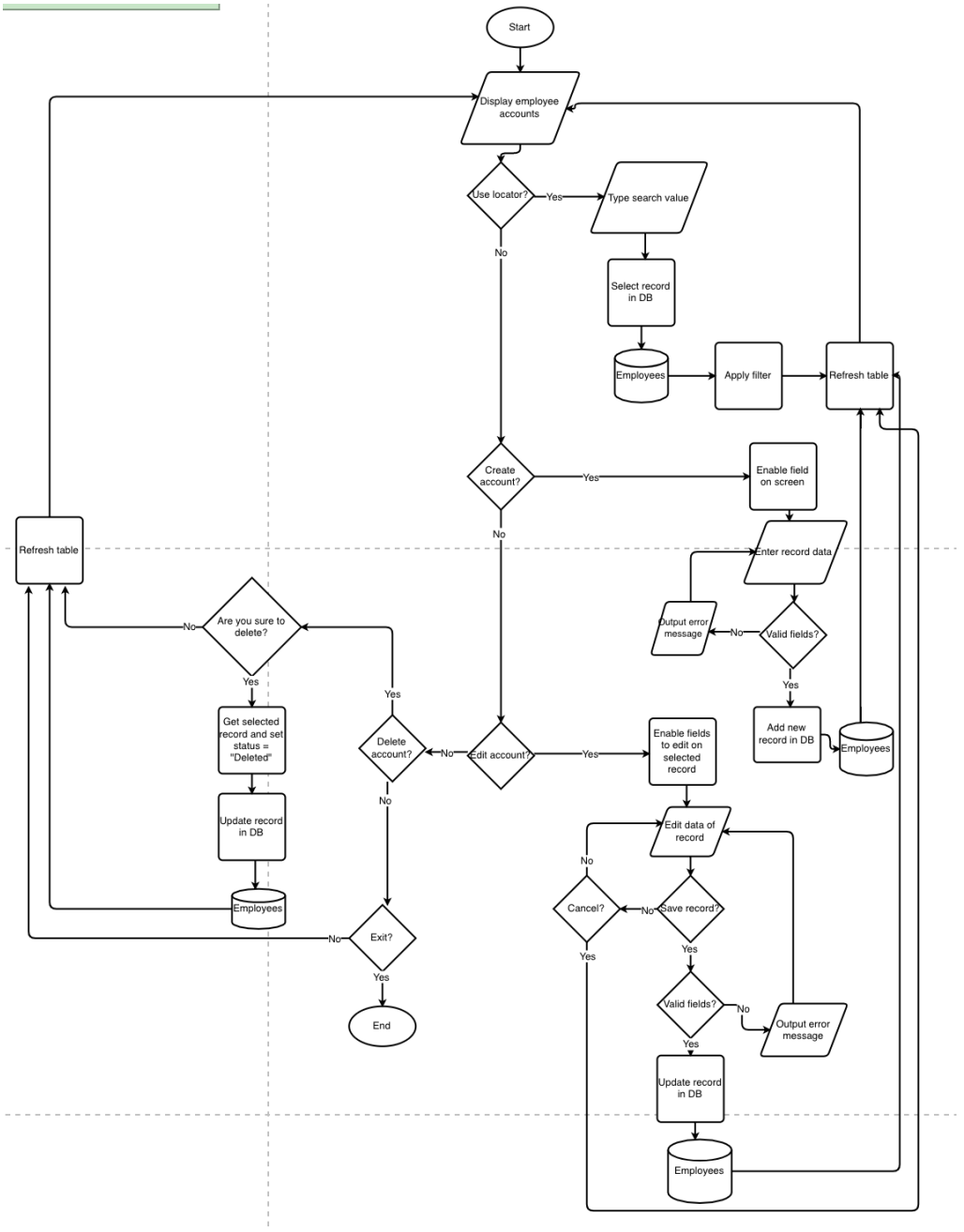


Figure 4: Admin portal - task administration (accept, reject, assign and filter tasks)

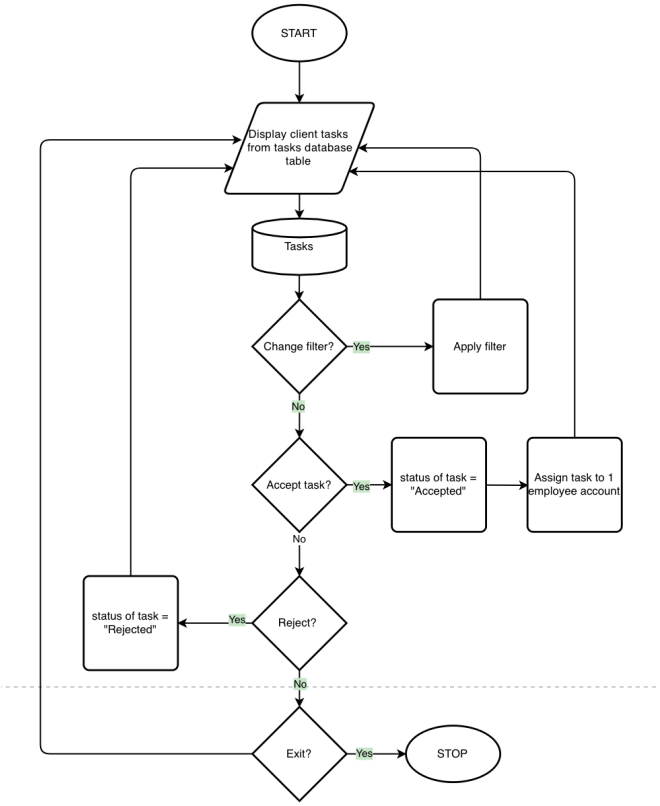


Figure 5: Admin portal - flexible filtering algorithm flowchart

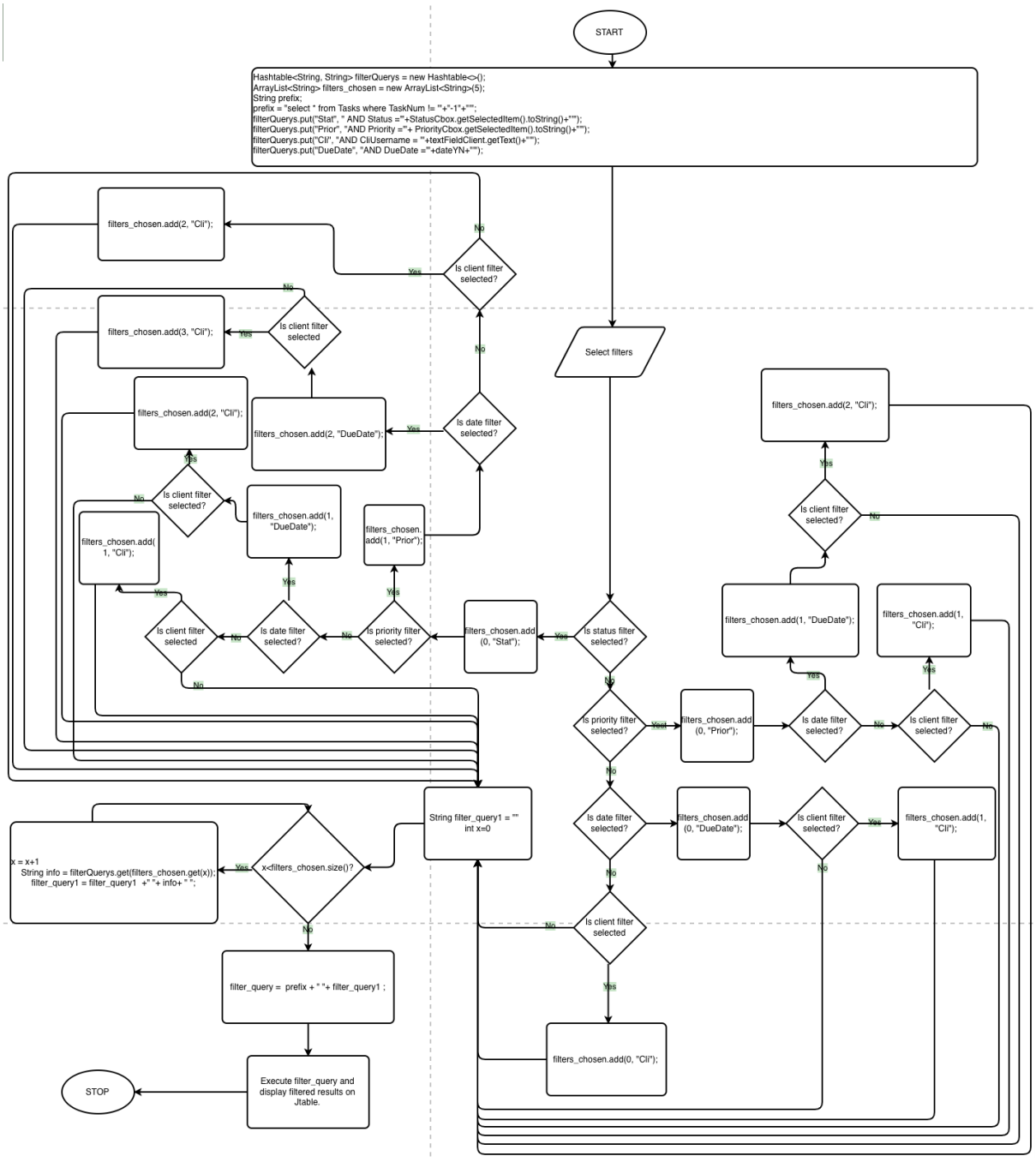


Figure 6: Login flowchart

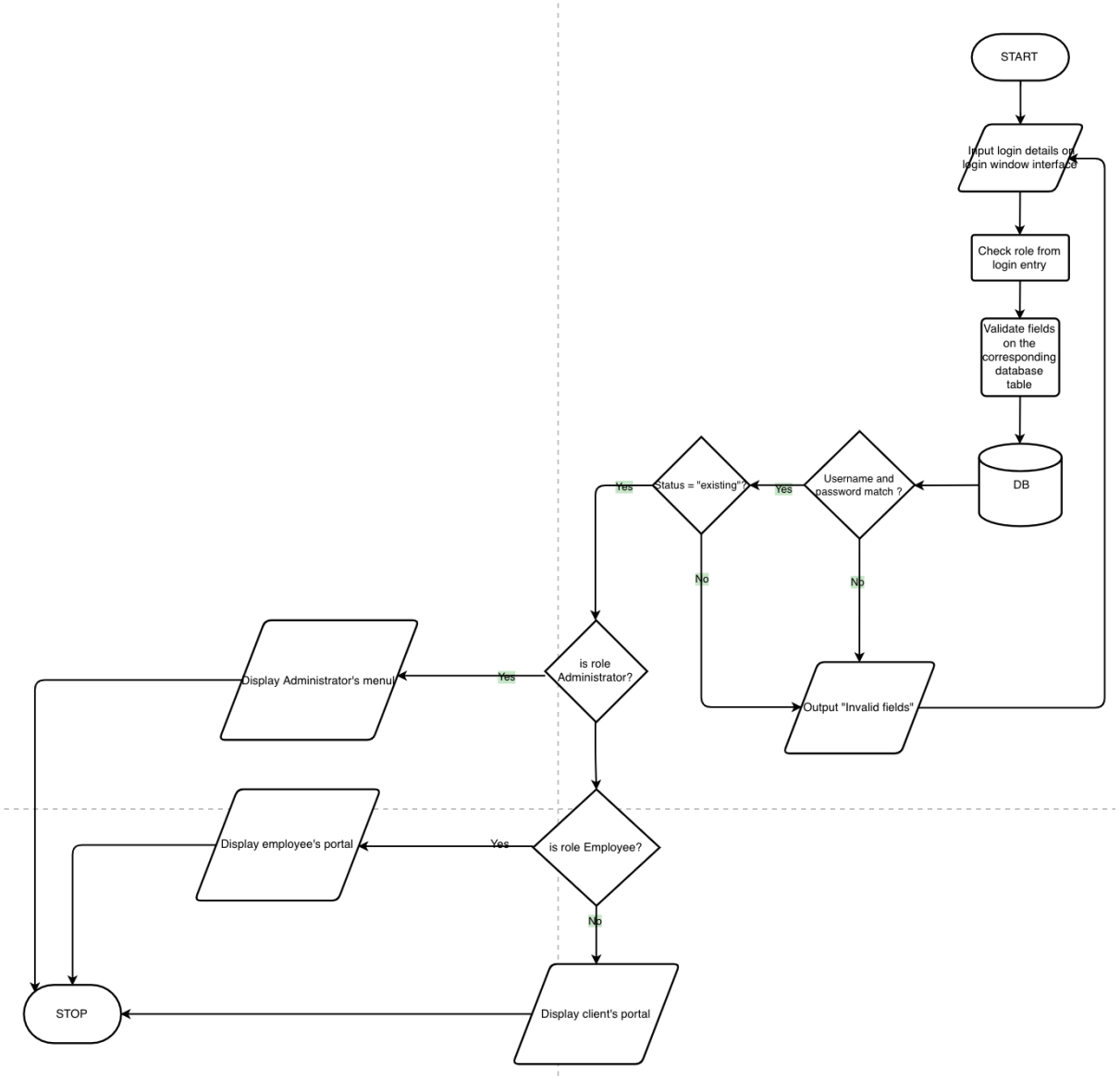


Figure 7:
Employees portal - edit status of the assigned task and add comments to it.

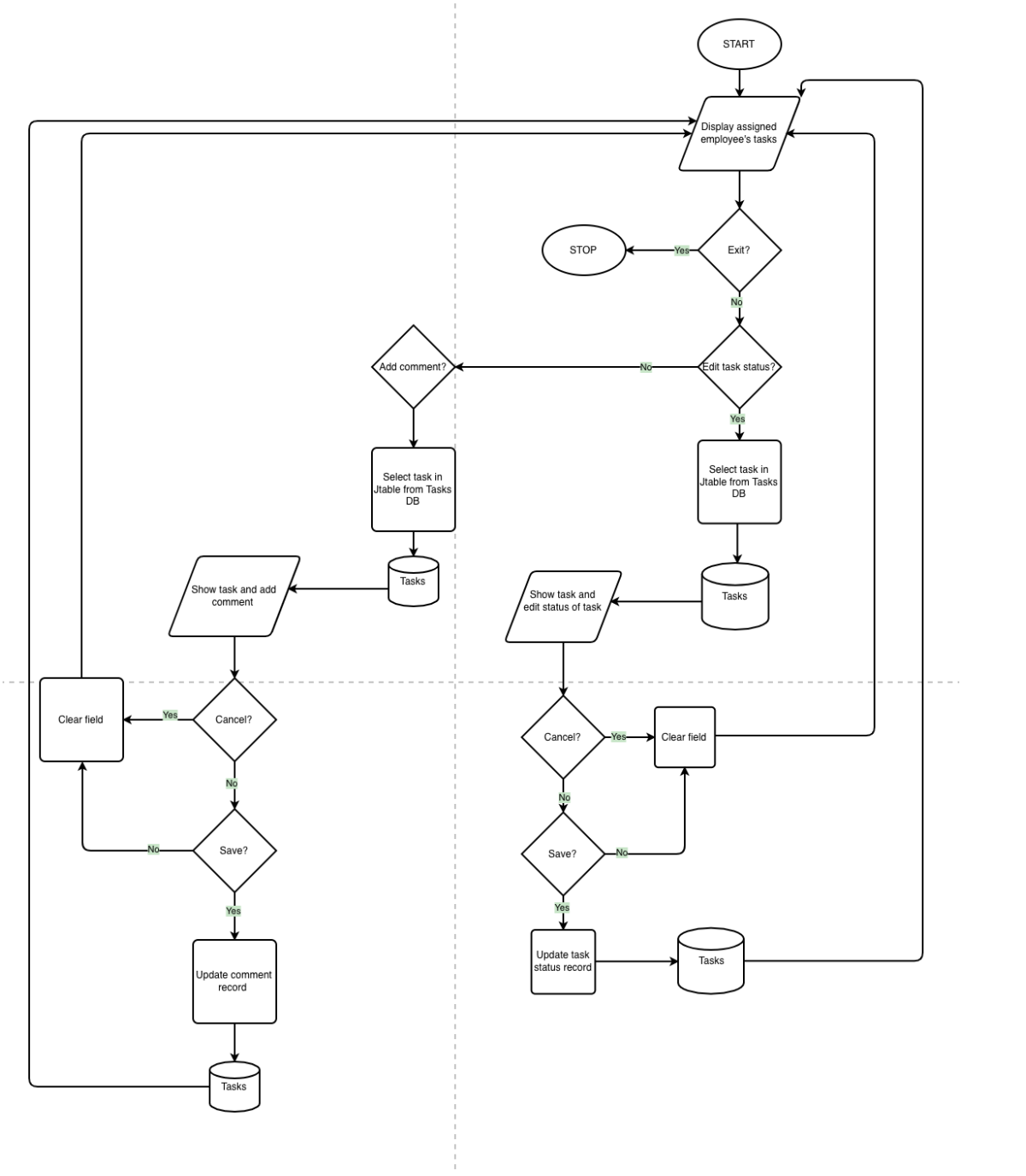
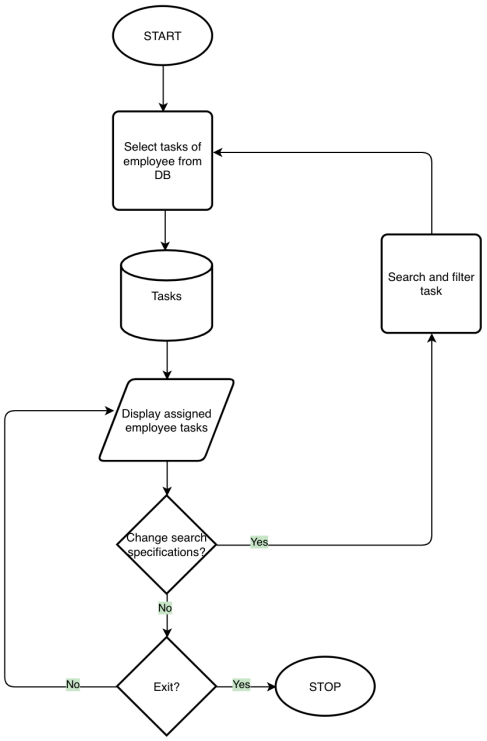

START
Display assigned employee's tasks
Exit?
Yes
STOP
No
Edit task status?
No
Add comment?
Yes
Select task in Jtable from Tasks DB
Tasks
Show task and add comment
Cancel?
Yes
Clear field
No
Save?
No
Yes
Update comment record
Tasks
Select task in Jtable from Tasks DB
Tasks
Show task and edit status of task
Cancel?
Yes
Clear field
No
Save?
No
Yes
Update task status record
Tasks

Figure 8: Employees portal - View and filter assigned tasks



The assigned task is displayed to each employee as the employee ID is the foreign key of the table “tasks” . Each task will have only one assigned employee username for simplicity reasons. Therefore, with this variable the different tasks created with the same assigned employee username will be displayed.

Figure 9: Employee portal - flexible filtering algorithm flowchart

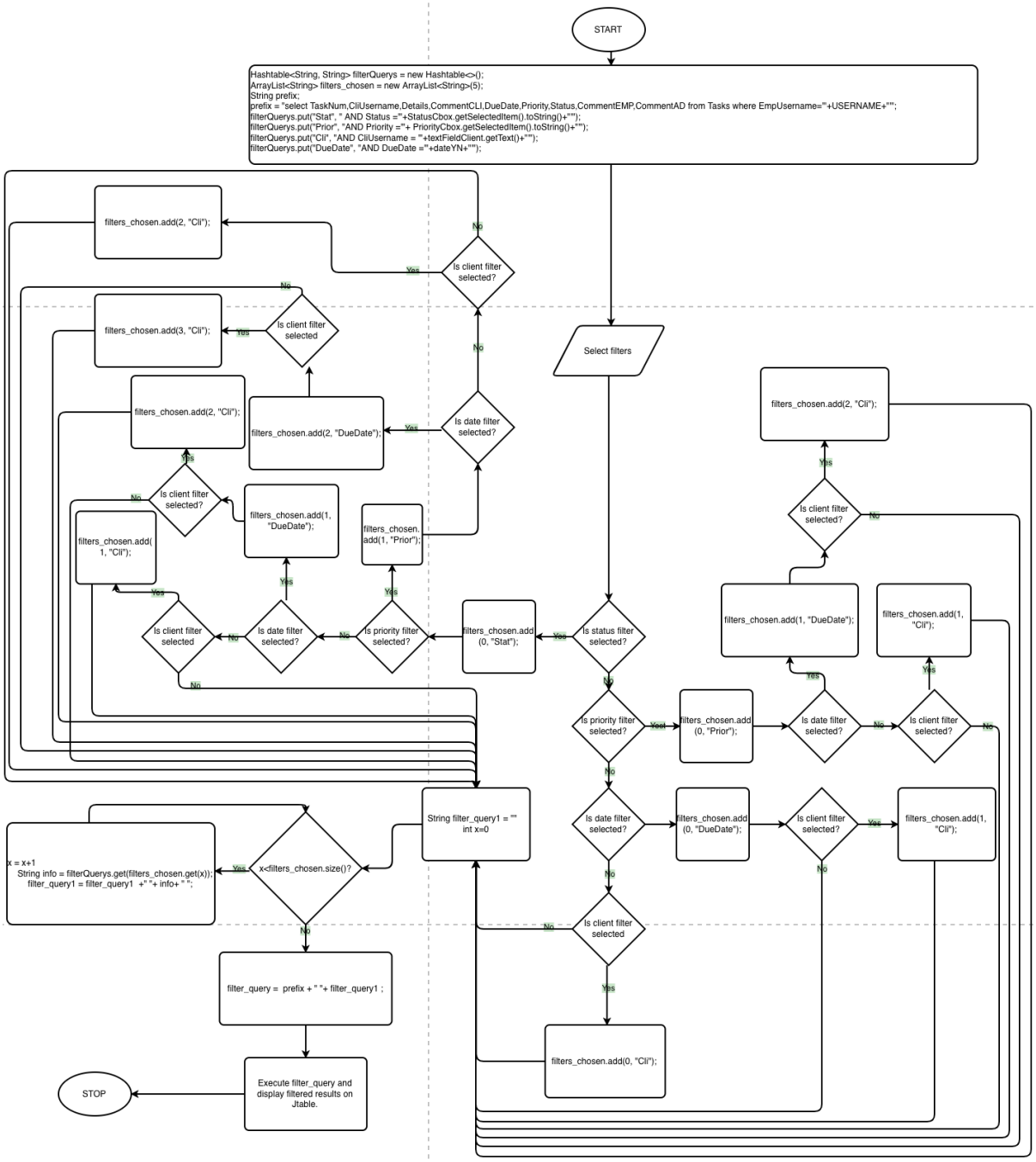

START
Hashtable<String, String> filterQuerys = new Hashtable<>();
ArrayList<String> filters_chosen = new ArrayList<String>(5);
String prefix;
prefix = "select TaskNum,CliUsername,Details,CommentCLI,DueDate,Priority,Status,CommentEMP,CommentAD from Tasks where EmpUsername='"+USERNAME+"'";
filterQuerys.put("Stat", " AND Status ='"+StatusCbox.getSelectedItem().toString()+"'");
filterQuerys.put("Prior", "AND Priority ='"+ PriorityCbox.getSelectedItem().toString()+"'");
filterQuerys.put("Cli", "AND CliUsername = '"+textFieldClient.getText()+"'");
filterQuerys.put("DueDate", "AND DueDate ='"+dateYN+"'");
Select filters
Is status filter selected?
Yes
filters_chosen.add (0, "Stat");
Is priority filter selected?
Yes
filters_chosen. add(1, "Prior");
Is date filter selected?
Yes
filters_chosen.add(1, "DueDate");
Is client filter selected?
Yes
filters_chosen.add(1, "Cli");
Is client filter selected
No
Is date filter selected?
No
Is priority filter selected?
No
Is client filter selected?
No
filters_chosen.add(2, "Cli");
Is client filter selected?
Yes
filters_chosen.add(2, "Cli");
Is date filter selected?
Yes
filters_chosen.add(2, "DueDate");
Is client filter selected?
Yes
filters_chosen.add(3, "Cli");
Is client filter selected
No
String filter_query1 = ""
int x=0
x<filters_chosen.size()?
Yes
x = x+1
String info = filterQuerys.get(filters_chosen.get(x));
filter_query1 = filter_query1 +" "+ info+ " ";
No
filter_query = prefix + " "+ filter_query1 ;
Execute filter_query and display filtered results on Jtable.
STOP
Is priority filter selected?
Yes
filters_chosen.add (0, "Prior");
Is date filter selected?
No
Is client filter selected?
No
filters_chosen.add(2, "Cli");
Is client filter selected?
Yes
filters_chosen.add(1, "DueDate");
filters_chosen.add(1, "Cli");
Is date filter selected?
Yes
filters_chosen.add (0, "DueDate");
Is client filter selected?
Yes
filters_chosen.add(1, "Cli");
Is client filter selected
No
Yes
filters_chosen.add(0, "Cli");

Figure 10: Client portal - Task administration flowchart (create, edit, delete and filter tasks)

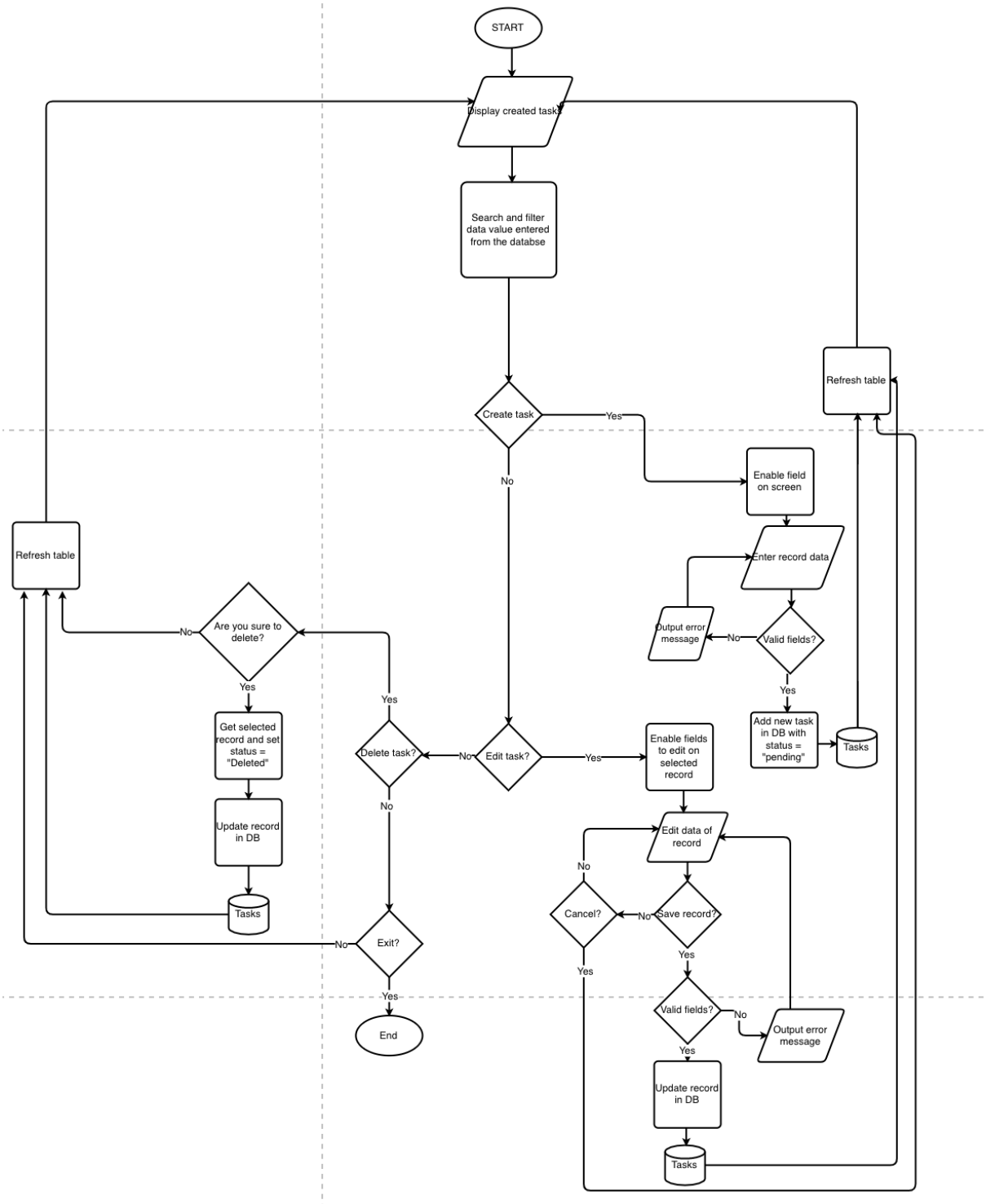


Figure 11: Client portal - Flexible filtering algorithm flowchart

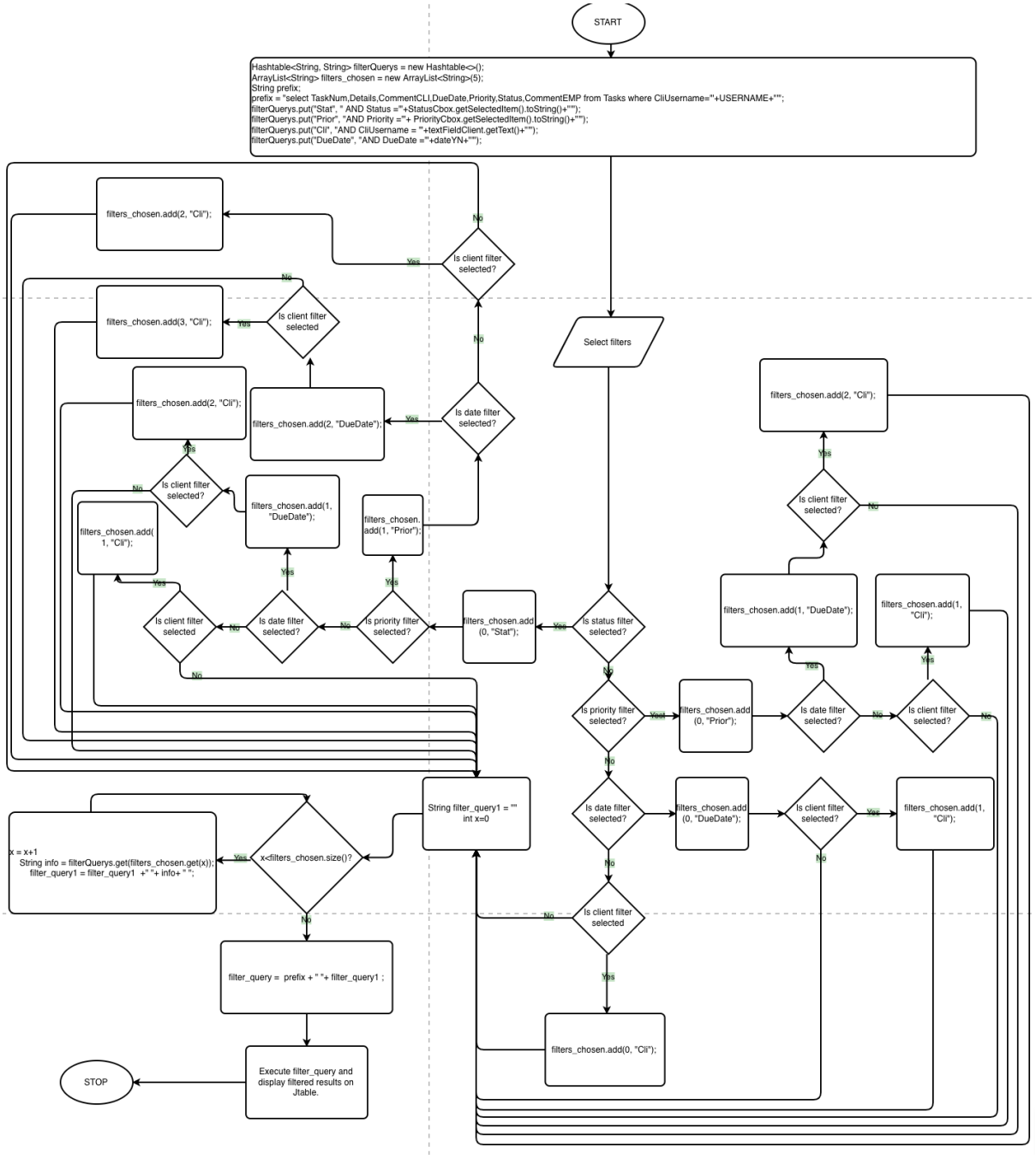


Figure 12: Random password generator

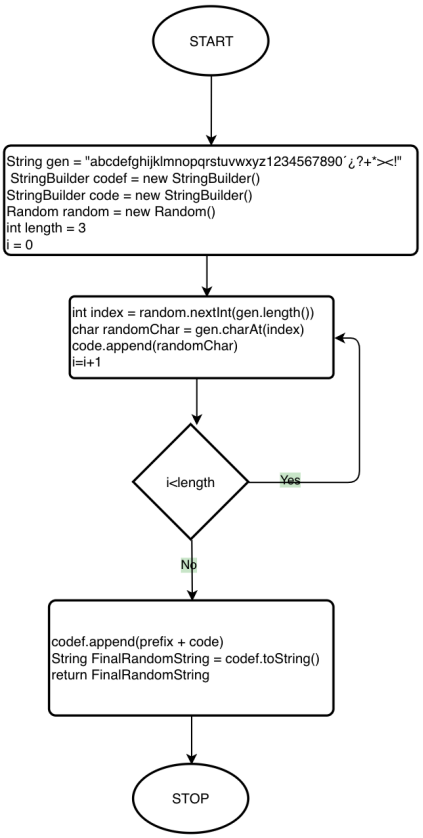
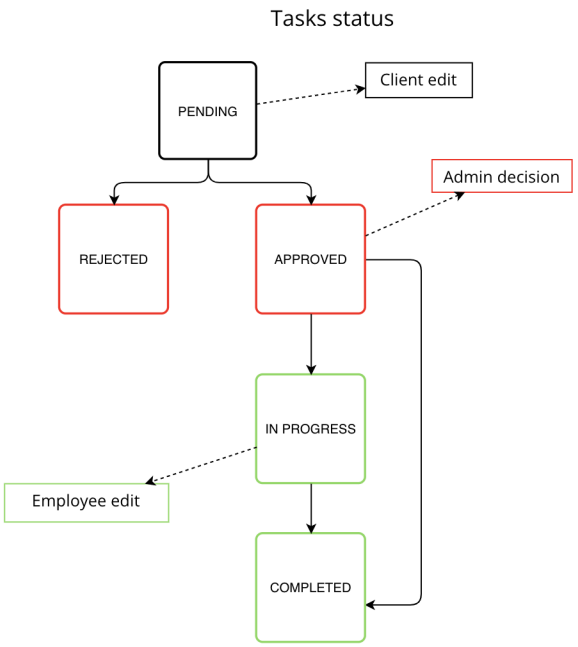
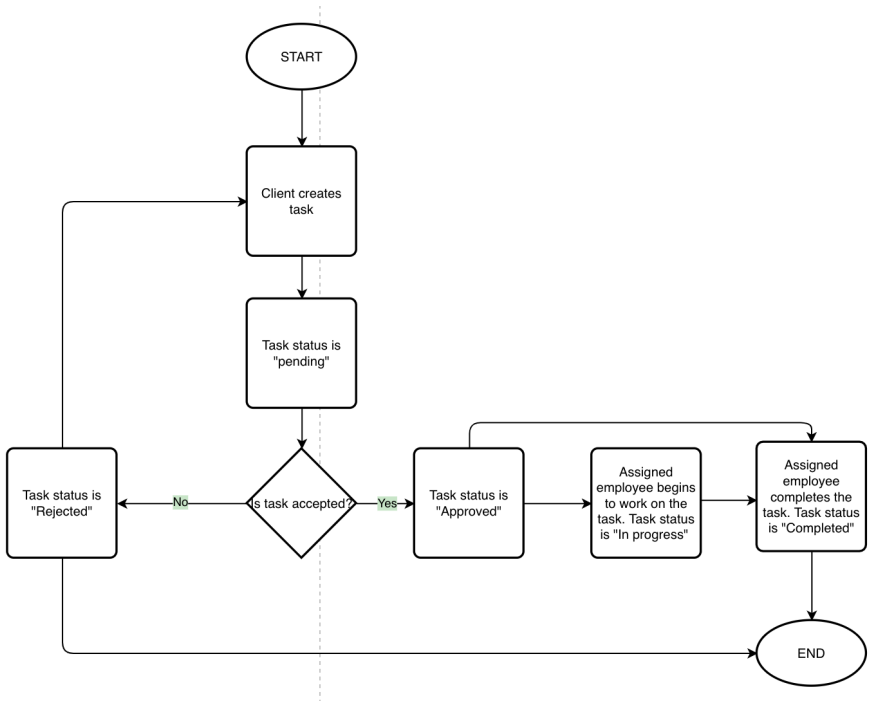


Figure 13: Entity task status transition diagram





Design of panels (graphical user interfaces)

Figure 11: Login interface

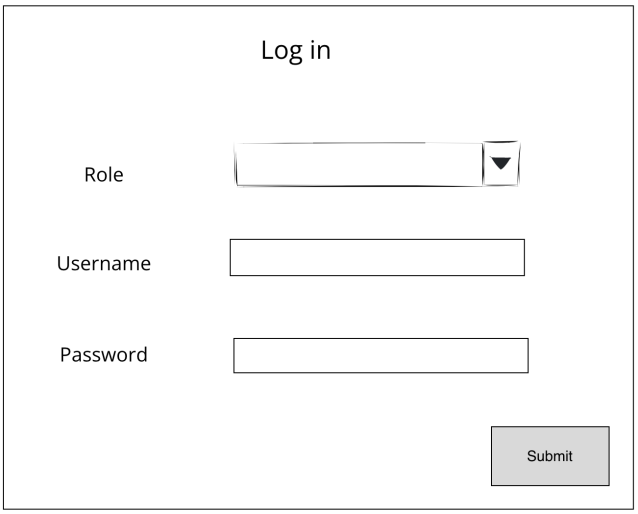


Figure 12: Administrator menu

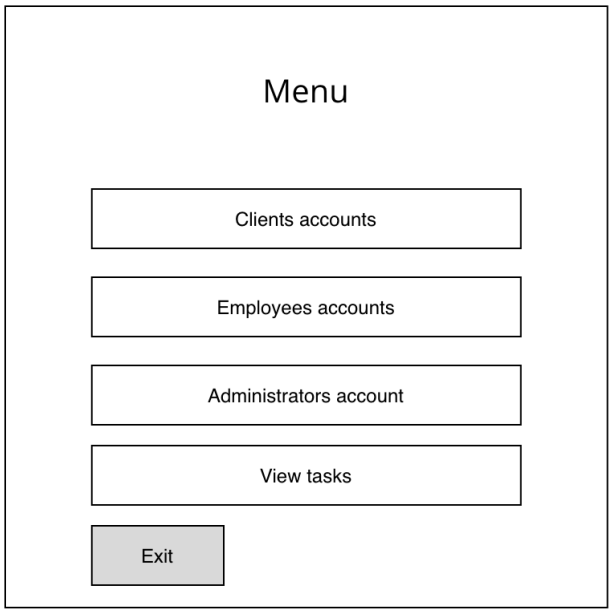


Figure 13: Clients accounts administration GUI

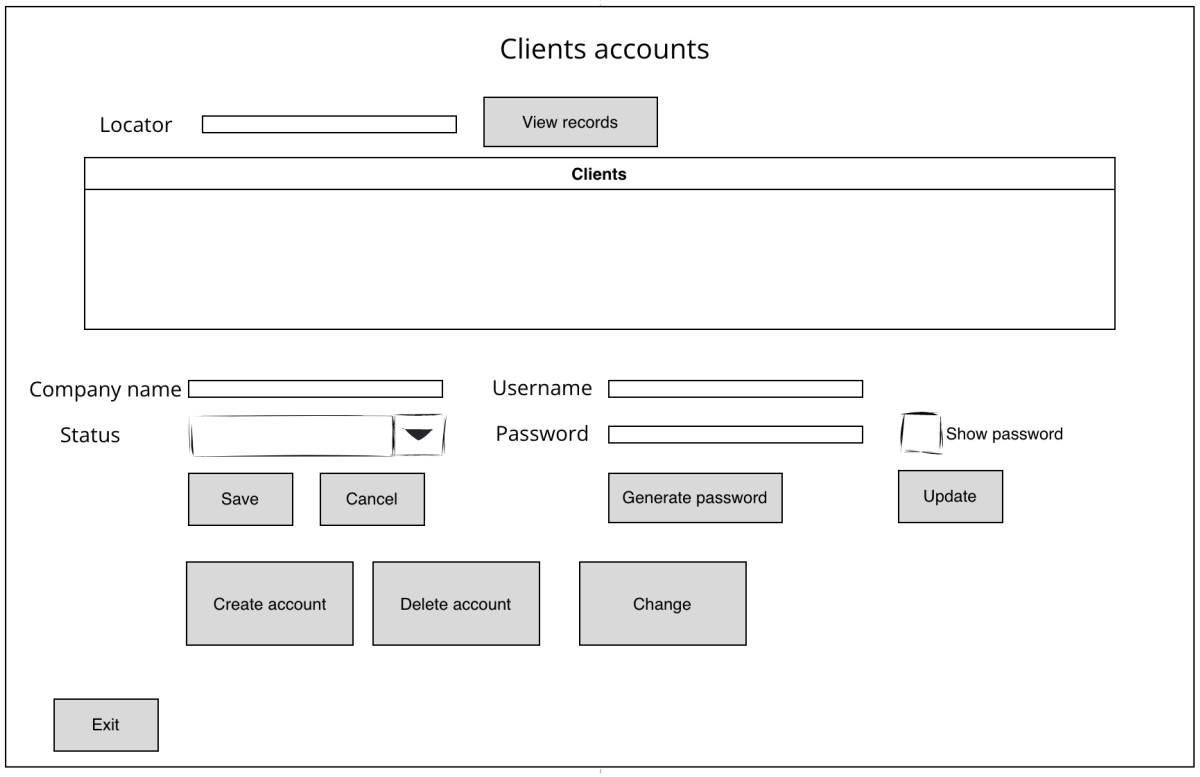


Figure 14: Employees accounts administration GUI

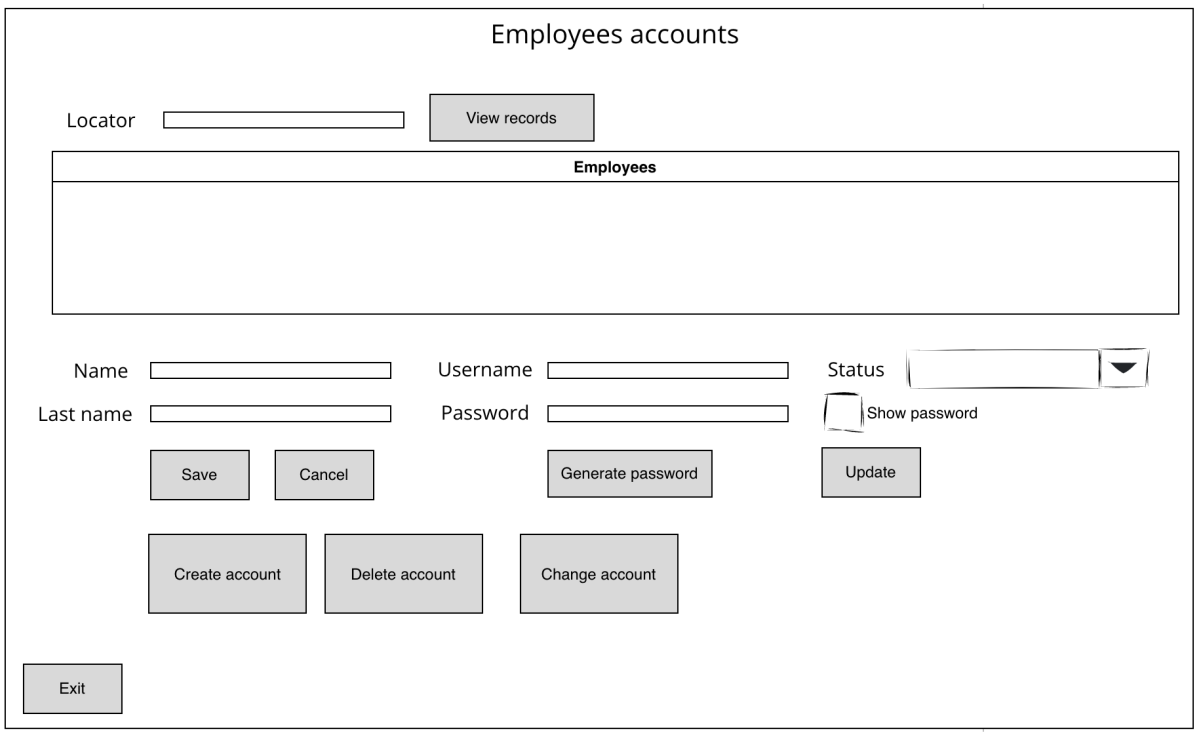


Figure 15: Administrators accounts management GUI

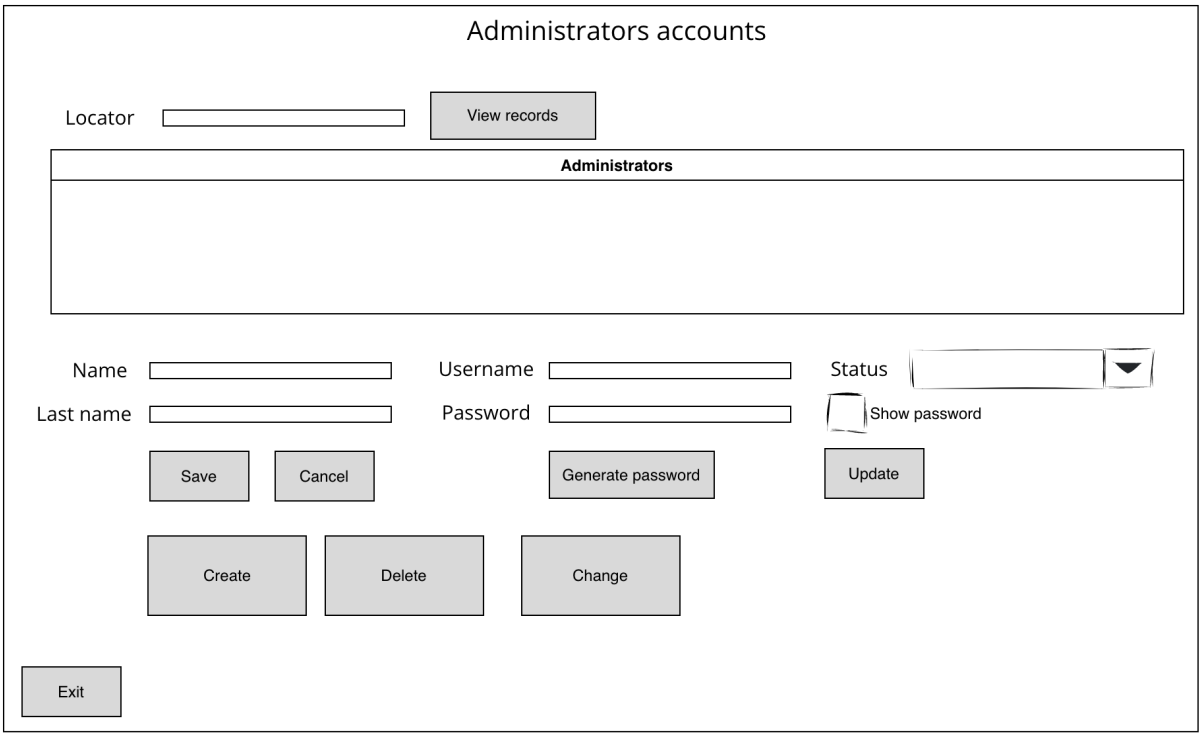


Figure 16: Tasks management GUI

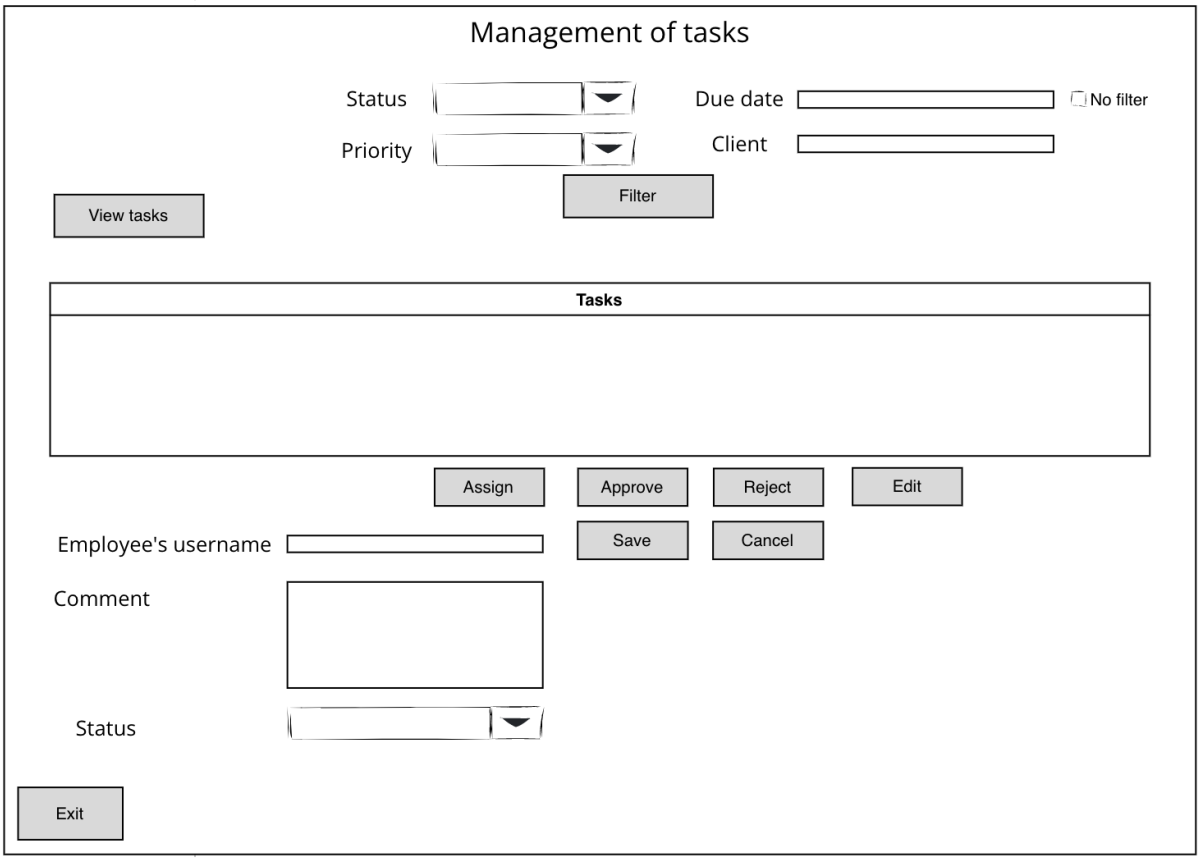


Figure 17: Employee portal

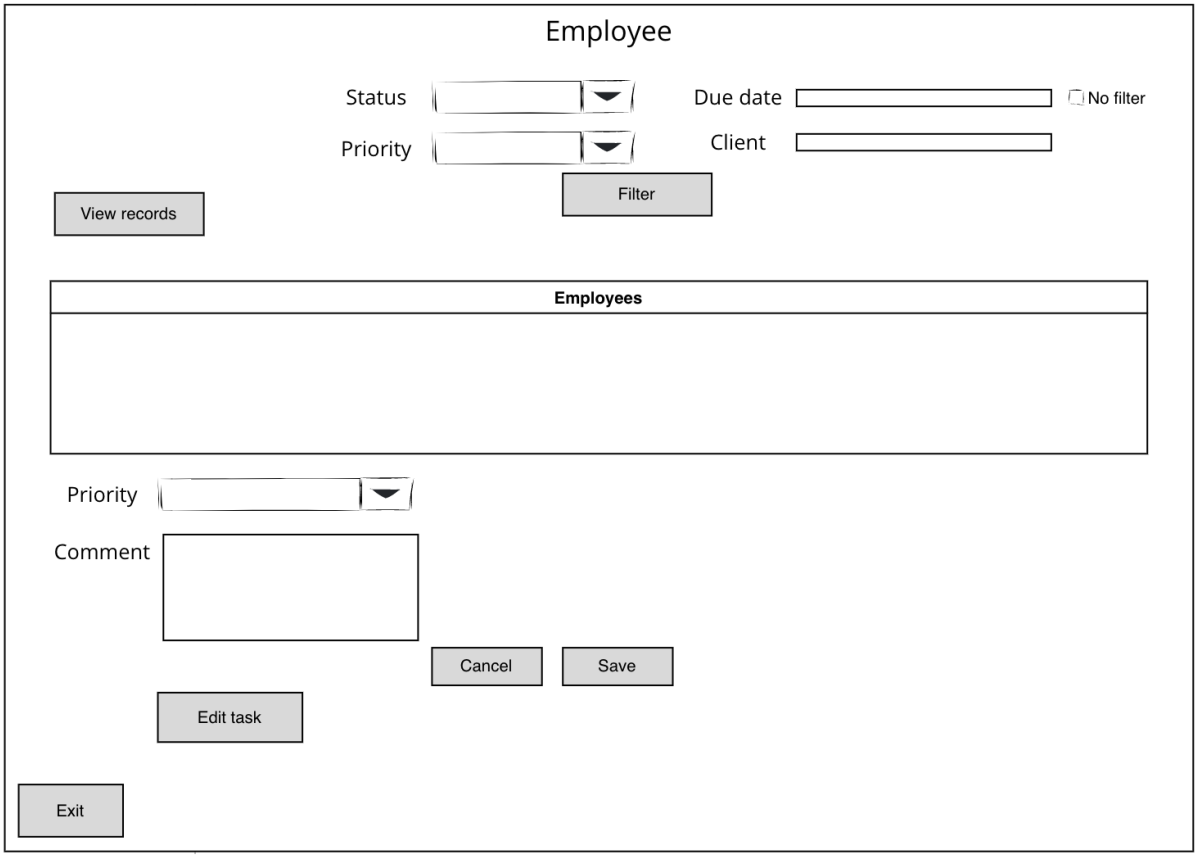
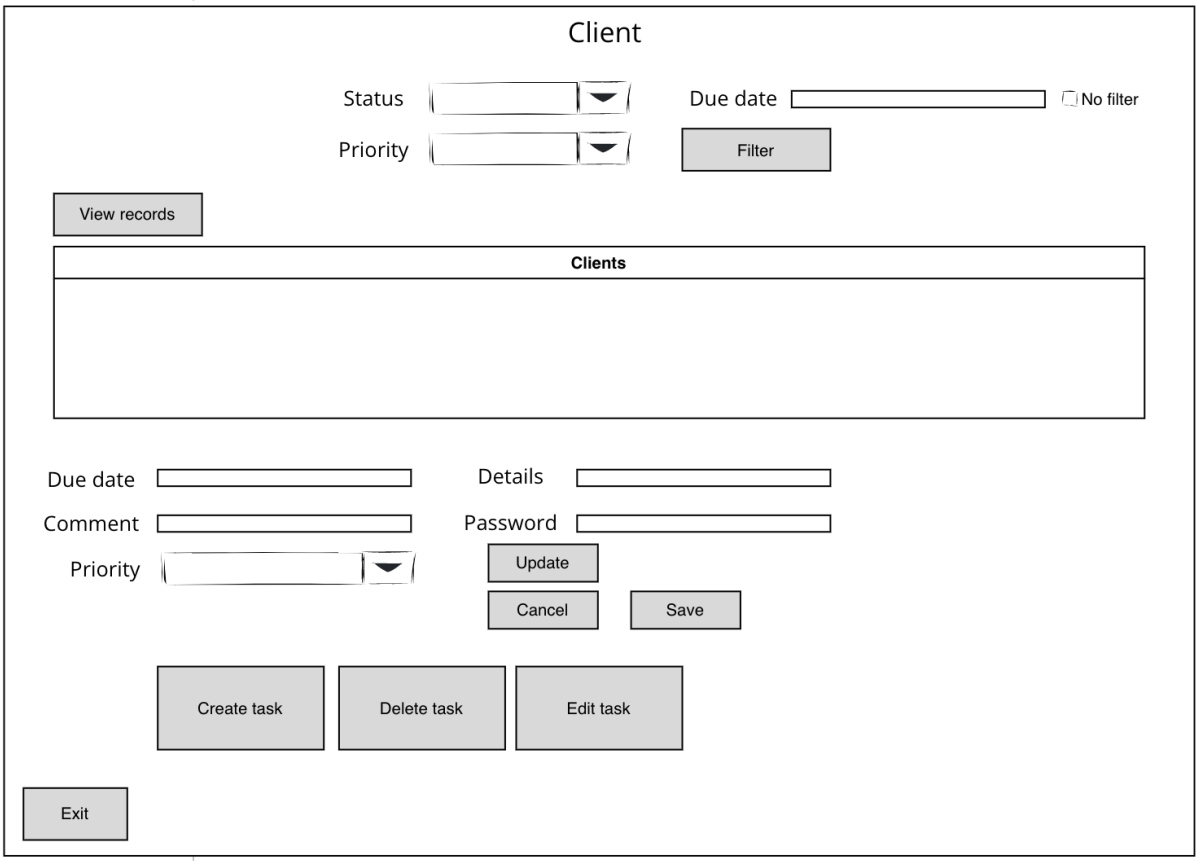
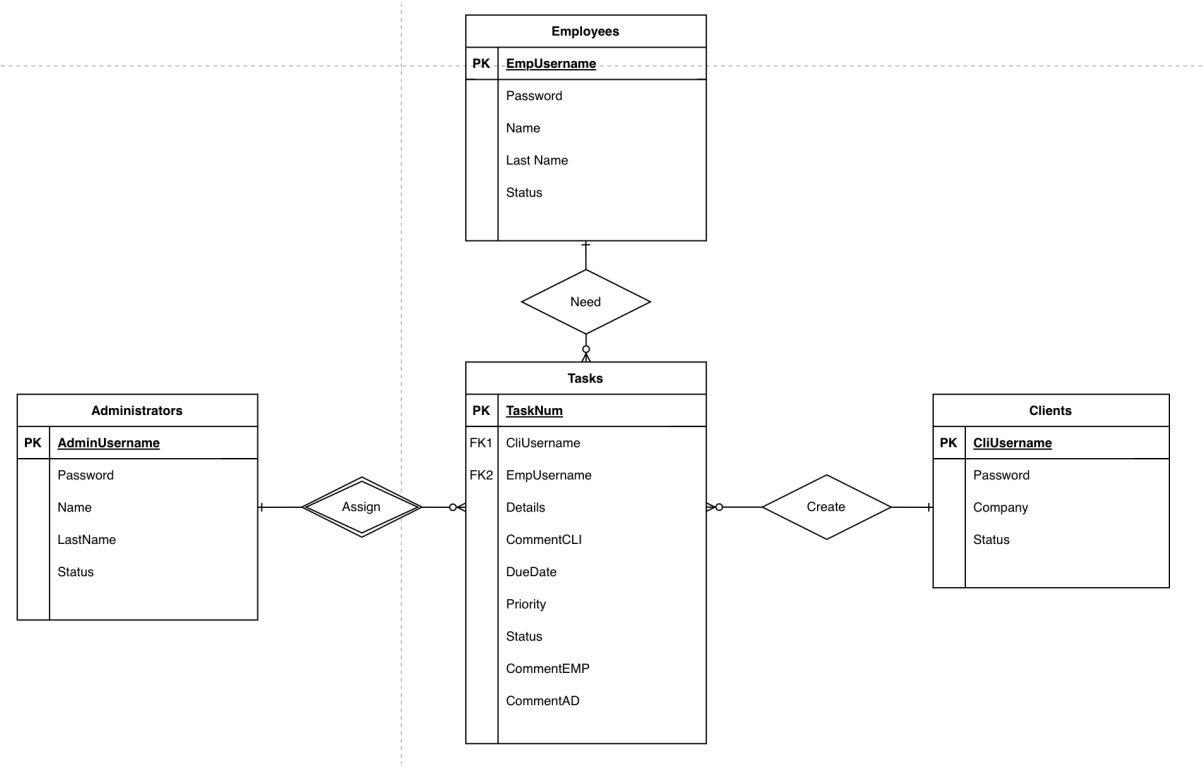


Figure 18: Client portal



Entity relationship diagram (ERD):



Test plan:

Action to be tested	Test method	Test outcome
Check if the program opens the administrator interface to start the flow of the system.	The user creates an admin account from the database. The user uses this account's information to register in the login panel. Afterwards, the user clicks the "submit" button to be redirected to the menu.	The administrator menu is successfully displayed.
Check if each user gets redirected to their respective e-portal windows successfully	Each user selects their role(administrator, employee, user) from a dropdown menu on the login window and then clicks the submit button to be successfully redirected to their e-portal.	Each window portal is successfully opened based on user role selection.
Validation of fields in the login interface	The user writes incorrect account details and then clicks the submit button.	The program displays an error message for invalid data entries from the user.
Check that each section button on the menu window works properly ("employees accounts", "administrators accounts", "clients accounts", "view tasks", and "exit")	The user clicks on any button on the menu portal.	The appropriate window e-portals are displayed when a button in the menu section is clicked.
Creation of an account	User creates the first administrator account from the database (to be able to run the program) and then logs in to the system. User clicks on the create account button to enable and complete the corresponding fields for the account being tested. The save button is clicked to store new data in the database.	The database table updates and the system displays the new records of the accounts in the corresponding Jtable.
Removal of an account	User creates the first administrator account from the database (to be able to run the program) and then logs in to the system. The user selects the record	The database table updates and the system displays the record of the deleted account with a status of "removed".

	in the database (on the Jtable displayed) of the account wanting to be deleted. The user clicks on the delete account button.	
Editing of details of any account	User logs in with an administrator account to have the rights to manage the accounts. The user clicks on the desired portal in the menu to be edited. Once the user is on the corresponding window, he/she clicks “change” in order to enable the fields that can be edited from the account in the database. User fills the fields and clicks “save”. If “cancel” is clicked, then the accounts’ information isn’t stored.	If saved, database records are updated and displayed over the Jtable. Afterwards, fields are cleared and disabled. If the canceled button is clicked, the database isn’t updated, and the fields are cleared and disabled.
A random password is generated successfully.	The administrator clicks on the generate password button.	A random password is created and displayed over the text field.
Management of tasks (accept, reject, assign or edit)	User logs in with an administrator account into the “View tasks” interface. User selects a task being displayed on the Jtable and clicks on “Accept/reject” to enable the corresponding fields. The user completes them up accordingly and stores them by clicking on “save”. The administrator assigns the task by clicking on “assign task” and filling the corresponding fields. To edit, the user clicks on “edit” and completes the fields. The user updates the record in the database by clicking on the save button.	Jtable is displayed with updated table records or new data.
Filter by the use of a locator (function that can only be used by administrators)	Over the locator text field the administrator types in the username of the client or employee depending on which portal of the system the user is viewing.	The program displays filtered results in the Jtable according to the data in the locator’s text field.

Flexible filtering(employees, clients, and administrators can use this function on their portals)	User logs in with an employee, client or administrator account. The user selects and completes the fields enabled desiring to search. Then, the user clicks on “filter”. Administrators can filter through selection on the management of tasks window. Both admins and employees can filter tasks through status, priority, client, and due date. Clients can filter their own tasks by status, priority, and due date.	The program displays filtered results over the Jtable in the corresponding portal.
Creation of a task and validation of data entries.	User logs in with a client account and clicks on “create task” to enable the fields necessary to make a requirement. The client will have to assign, with a combo box, a level of priority to the task ranging from 1-5 and also write a short description and select from a calendar the due date. The user clicks on “save” to store a new task in the database.	The program displays an error message if the fields are invalid. However, the database table “Tasks” is updated if the fields are valid.
Removal of task (client)	User logs in with a client account and clicks on “View tasks” to see al the requirements proposed. The client selects a task and clicks on “Delete task” to remove the selected requirement. If the task’s status is anything other than “pending”, the button “Delete task” will be disabled.	The program displays “record has been deleted successfully” and the database table Tasks updates the status of the selected record to “Deleted”
Edit task (client)	User logs in with a client account and clicks on “Edit task” to enable the fields necessary to update the requirement. The client can change the due date, the comments, and the details. The user clicks on “Update”	If the task is successfully updated, fulfilling the validations established, database table Tasks will change the fields CommentCLI, Due date, and Details of the selected record.

	and a validation message will pop up to confirm the changes being made to the record. The user can also click “Cancel” and the fields to update the task will disappear.	
Auto Incrementation of task number in database table	User creates a task from the clients' section after a successful login. The user completes the corresponding fields within the validation ranges and clicks on the save button. then check in the database if the task number is correct and follows an order of creation.	The database table “Tasks” is updated with each record following an ascending order of auto-incremented numbered tasks created in the task database table.
View tasks (employee portal)	User logs in with an employee account and clicks “View tasks”.	The Jtable in the employee’s portal displays all the tasks that were assigned to the employee account that logged in.
Edit Task (employee portal)	User logs in with a client account and clicks “edit task” to enable the fields to make an update. Employee can change the status of the task to “In progress” or “Completed” only and can write a comment. User clicks “Save” and a validation message will pop up to confirm the changes being made to the record. The user can also click “Cancel” and the fields to update the task will disappear.	If the task is successfully updated, fulfilling the validations established, the database table Tasks will change the fields Status and CommentEMP according to the information inputted.