1. **Environment**

The requirements for our application are not too high, it can be run both on Windows (at least 10) and Linux(version Mint 21.2 or higher). Our application will take about 500MB of disk space and at least 4GB of memory. Components of our application will use external framework such as Intellij, Apache Derby and Java SE Development Kit (JDK) 17.08 or higher. There will not be any special user privileges.

1. **Technical stack**

List of software, components and frameworks we will use:

- Intellij IDEA for layout of our application and writing code

- Apache Derby for database management

-Java JDK (17 or higher)

-Java Swing for GUI

-AWT- Abstract Window Toolkit for GUI

Out of listed elements Intellij and Apache Derby have been proposed by our client, but they are not necessary. We can use different framework for database management as long as there are not too big servers (like Microsoft SQL Server).

Possible options for GUI (other than those mentioned earlier):

- Java FX

- Apache Pivot;

- Grails;

- Google Web Toolkit;

Possible database systems:

1. Approved

- Apache Derby

2. Plan B:

- XAMPP

3. Declined

-Microsoft SQL Server

1. **Required criteria**

add()- This function will create a new pop-up window that will contain four text fields to add data about an employees name, surname, phone number and email. All of this data will be stored in four different Strings that will be marked: “inName”, “inSurname”, “in Phone”, “inEmail”. There will also be a button with a plus symbol that sends data from String to database in Apache Derby. The table to store the data will contain this columns: “ID”, “name”, “surname”, “phone\_nr” and “email”, where ID is an auto-generated, numerical primary key and the other columns are of type varchar (limited successively by 20, 50, 13, 50).

edit()- This function will transform labels with info about selected employee to text fields by removing labels and crating new text fields in their place. After putting there new data program will find edited employee and update his data in database by previously taking informations about him and putting them into query “UPDATE employees (it is a name of a table in our database) SET name = new\_name, surname = new\_surname, phone\_nr = new\_phone\_nr, email = new\_email” where variables with prefix “new” are new data taken from text fields by getText() function.

delete()- This function will engage a query that will look like this: “DELETE FROM employees WHERE name=g\_name, surname = g\_surname, phone\_nr = g\_phone\_nr, email = g\_email” where variables with prefix “g” are data from selected employee taken by getText() function.

find()- This function will firstly open a new pop-up window witch will contain four text fields (name, surname, phone nr, email) where where user puts data by witch he wants to look for an employee and a search button. After pushing search button data from text fields (if there were inputted data to look for an employee) is being put to the Stings with same names as text fields titles. Then program uses given data in a query that selects employees that confirm given parameters thanks to “LIKE %…%” where in “…” is data given by user.

sort()- This function will engage a query that will look like this “SELECT \* FROM employees ORDER BY xxx” where “xxx” stands for option of sorting selected by user, like sorting by name.

changeLang()- This function will change all of the applications language to one chosen by user (English or German) by swapping all of the labels content to ones in that language

show()- This function will select and show in table in our application all of the needed content from database and use a query with SELECT followed by specifications based on where its gonna be used (we plan to use it in other functions like find())

Validation- we will validate all of text field, that: user will not be able to type numbers in name and surname fields; will not be able to type letters in phone number fields; user will have to use “@” and “.” in email field. To do that we will use function contains().

1. **Optional criteria**

1. We want to add a bonus feature to find() function. If data from user has been found correctly in 100% (for example user typed in “Tom” and there is a Tom in database) function show() shows only answers to that accurate search (it will ignore people that name only has part “Tom”). We can do that by using conditional statements (“if”) that will check if there is a direct answer in database by sending a query that will select employees with name “Tom” (like SELECT \* FROM employees WHERE name LIKE “Tom”). If a query returns nothing, function works as usually, otherwise it will only show “Tom’s”.

2. We want to replace “add new” button and add it to a new made task bar that will also have a sort option for our table and option to close the application. This shouldn’t be technically complicated, we would just need to add a task bar from for example Java Swing and assign functions described earlier to his elements.

3. We will use transfer to uppercase to data given from user by text fields using toUppercase() function, so we will prevent problems with comparing Strings with elements in database (that we will also transfer that way).

1. **Technical test cases**

1. Add button – after pushing “add new” button there pops up a window with four text fields (name, surname, phone nr, email) and a button with a plus. After filling text fields and clicking the button, new employee is added to database and is shown in a table in our application.

2. Sort – after selecting option of sorting the table in application refreshes and shows correctly sorted database.

3. Language – after selecting language, all of the text elements visible in our application are translated to chosen language.

4. Find – after clicking the find button there will pop up a new window with four text fields (name, surname, phone nr, email) and a search button. After filling at least one field and clicking “search”, a table in our program will update and show searching results accurate to data given by user.

5. Edit – after clicking the edit button, labels with data about selected user will be transformed into text fields allowing user to give new data. After clicking edit button again, data of employee will be changed on our screen and in our database.

6. Delete – after clicking delete button on an employee, he will be deleted from database and he will disappear from table in our application.

7. Validation – after: typing a number to name/surname field; typing email wrong (no “@” or “.” included); typing a letter to a phone number field, user will not be allowed to preform final action (add/edit/find an employee) by disabling a button to continue as long as will not correct his mistakes.

1. **Installation**

We will deliver our application in a zip file that, after unzipping and entering it, user will be able to enter by double clicking an EXE file.

To turn on the app there will also be needed JDK 17 and Apache Derby installed on users PC.