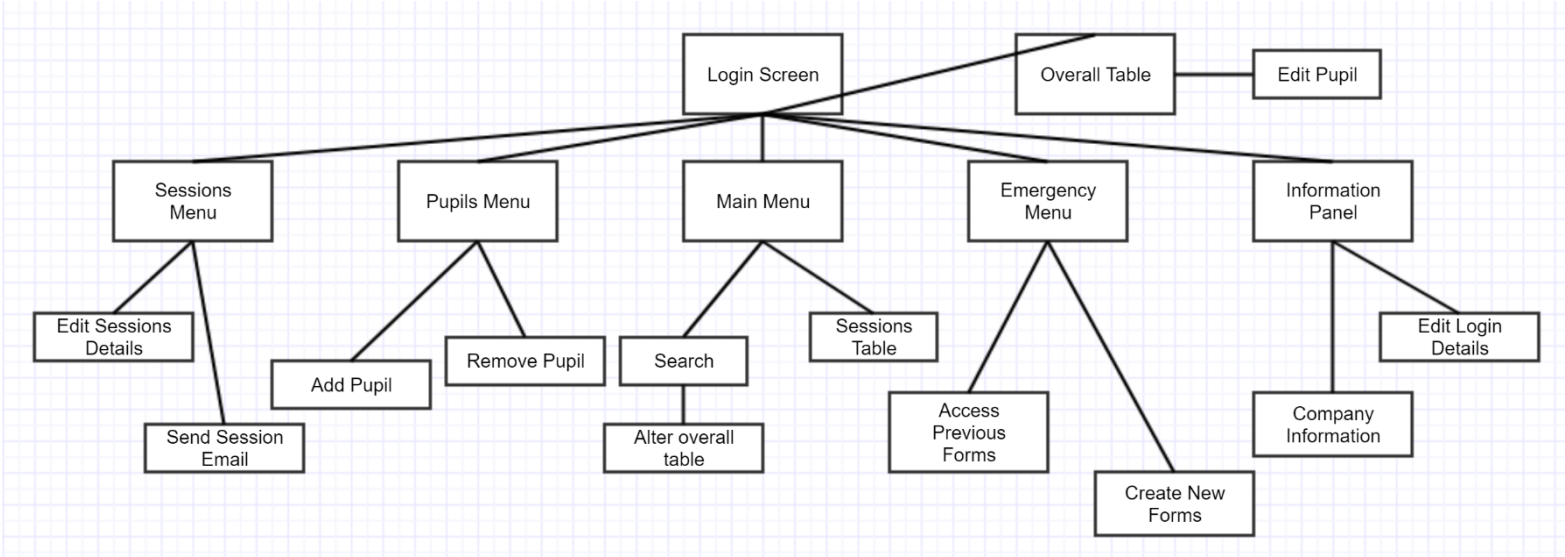
Criterion B: Design

Contents

Program Top Down Design



This diagram outlines how my program works and briefly shows all the possible functions.

SQL Tables

Column	Type	Default Value	Nullable	Char...	Collation	Privileges	Extra
ID	int(11)		NO			select,insert,update,references	auto_increment
Name	text		YES	utf8	utf8_general_ci	select,insert,update,references	
Age	int(11)		YES			select,insert,update,references	
DOB	date		YES			select,insert,update,references	
Gender	text		YES	utf8	utf8_general_ci	select,insert,update,references	
Email	text		YES	utf8	utf8_general_ci	select,insert,update,references	
Phone	text		YES	utf8	utf8_general_ci	select,insert,update,references	
Emergency	text		YES	utf8	utf8_general_ci	select,insert,update,references	
Hand	text		YES	utf8	utf8_general_ci	select,insert,update,references	
SessDate	date		YES			select,insert,update,references	
SessTime	int(11)		YES			select,insert,update,references	
SessInfo	text		YES	utf8	utf8_general_ci	select,insert,update,references	
Exp	text		YES	utf8	utf8_general_ci	select,insert,update,references	
Medical	text		YES	utf8	utf8_general_ci	select,insert,update,references	
Pay	text		YES	utf8	utf8_general_ci	select,insert,update,references	
PaySess	int(11)		YES			select,insert,update,references	

My “Pupil” table from which I create my objects and store majority of my data in.

ID	Name	Age	DOB	Gender	Email	Phone	Emergency	Hand	SessDate	SessTime
1	Disha	18	1999-10-21	Female	disha@mail.com	0044837394		Left	2018-02-23	19:00
2	Abdullah	17	2000-03-29	Male	abdullah@mail.com	0044123987		Left	2018-02-23	05:45
3	Saved	17	2000-03-10	Male	saved@mail.com	0044239341		Left	2018-03-01	13:45
4	Salman	18	1999-11-23	Male	salman@mail.com	0044239837		Left	2018-02-17	23:15
5	Irmak	18	1999-12-15	Female	irmak@mail.com	0044298135		Left	2018-02-03	06:30
6	Salma	18	1999-12-31	Female	salma@mail.com	0044981771		Left	2018-02-09	09:00
7	Ali	14	2003-09-21	Male	ali@mail.com	0044819276		Left	2018-04-23	10:45
8	Zarah	16	2001-08-12	Female	zarah@mail.com	0044092317		Left	2018-03-12	06:30
9	Sarah	16	2001-04-19	Female	sarah@mail.com	0044213852		Left	2018-02-21	15:30
10	Adam	17	2000-06-21	Male	adam@mail.com	0044912743		Left	2018-02-23	09:00

SessInfo	Exp	Medical	Pay	PaySess
At the park	Intermediate player. 4 years expereince. ...	She broke her arm a few years ago.	She will pay weekly	300
				0
		well. hello there		0
At the childrens park		One time he fell and shouted "help"		0
	She's ok nothing special			0
		A very large forehead		0
				0
At the park		stupid		100
	none			0
at the park	Experienced. He's been playing for 4 years.	Broke his arm 3 years ago.		0

Sample data in the “Pupil” table

Column	Type	Default Value	Nullable	Character Set	Collation	Privileges	Extra
Username	text		YES	utf8	utf8_general_ci	select,insert,update,references	
Passcode	text		YES	utf8	utf8_general_ci	select,insert,update,references	

The “Login” table used for my client to save his username and password

	Username	Passcode
	Arif	Cricket

Sample data for the “Login” table

Column	Type	Default Value	Nullable	Character Set	Collation	Privileges	Extra
Name	text		YES	utf8	utf8_general_ci	select,insert,update,references	
Form	text		YES	utf8	utf8_general_ci	select,insert,update,references	

The “Forms” table which will contain all the information regarding emergency forms.

	Name	Form
	Disha	Date: 21/03/2018 Disha recieved an injury at S...
	Adam	Date: 29/03/2018 Adam recieved an injury at ...

Sample data for the “Forms” table

Column	Type	Default Value	Nullable	Character Set	Collation	Privileges
State	text		YES	utf8	utf8_general_ci	select,insert,update,references
Question	text		YES	utf8	utf8_general_ci	select,insert,update,references
Answer	text		YES	utf8	utf8_general_ci	select,insert,update,references

The “Forgot” table which allows my client to set up a security question in case he forgets his login credentials

	State	Question	Answer
	true	Where was your first ever vacation?	Disneyland

Sample data for the “Forgot” table

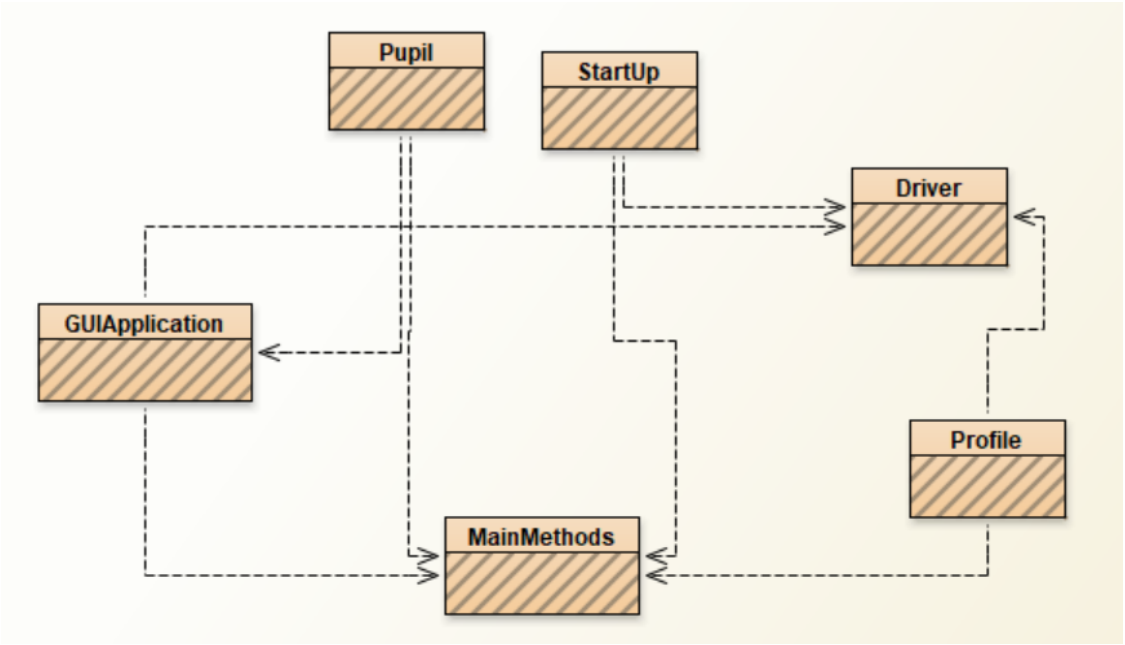
Column	Type	Default Value	Nullable	Character Set	Collation	Privileges
Name	text		YES	utf8	utf8_general_ci	select,insert,update,references
FileName	text		YES	utf8	utf8_general_ci	select,insert,update,references
FilePath	text		YES	utf8	utf8_general_ci	select,insert,update,references

The “Documents” table which allows my client to upload any documents for any pupil in the database

	Name	FileName	FilePath
	Adam	AdamCricket	C:##Users##abdul##Pictures##Saved Pictur…

Sample data for the “Documents” table

Class Relationships



BlueJ is being used to show how the classes are interconnected and which classes plan to be used for what purpose.

Pupil Class – This is the one and only Object Class I use to create the objects for all my clients pupils.

StartUp – The GUI class used in the login phase. It goes to open the GUIApplication class

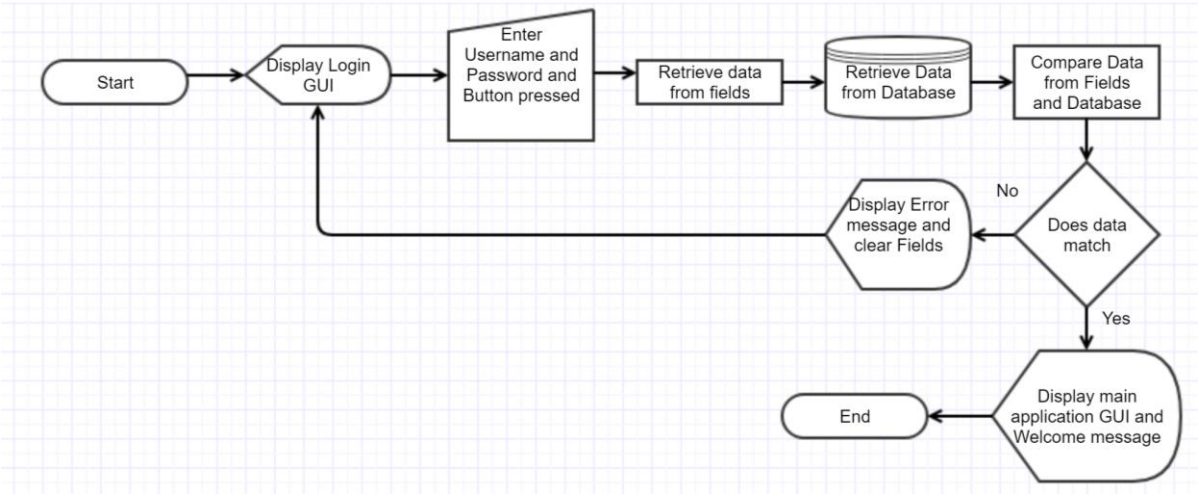
Profile – The GUI class being used to edit and display all the attributes of a specific pupil.

GUIApplication – The main GUI class where I can use and display majority of the data.

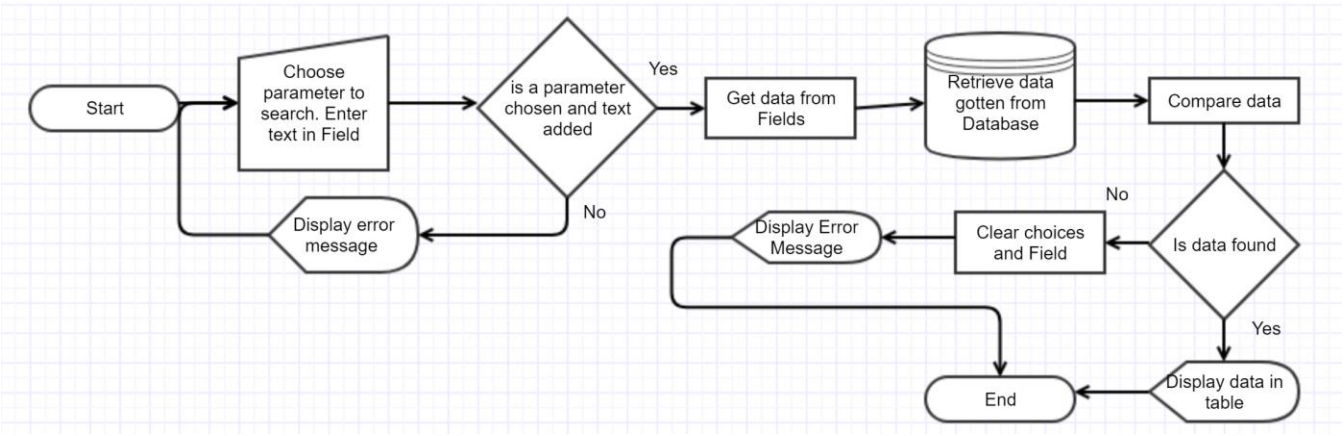
Driver – The layer of abstraction between the DBMS and Netbeans, here I hold all SQL related code. This also contains the code to create the database initially.

mailMerge – The class concerned with sending emails. It is connected to the Application and StartUp classes

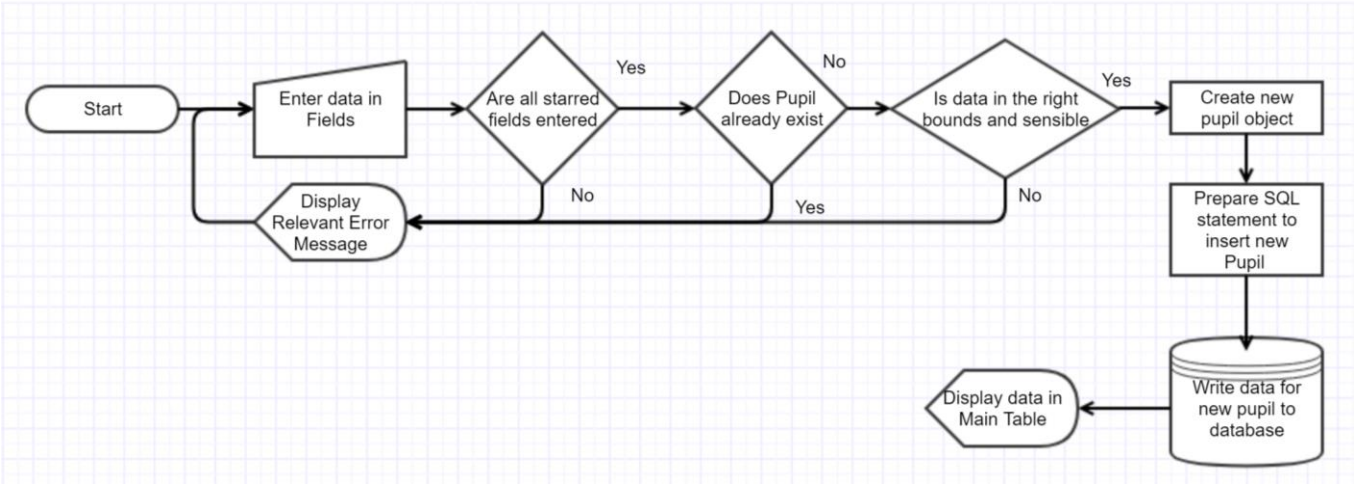
Functions Flowcharts



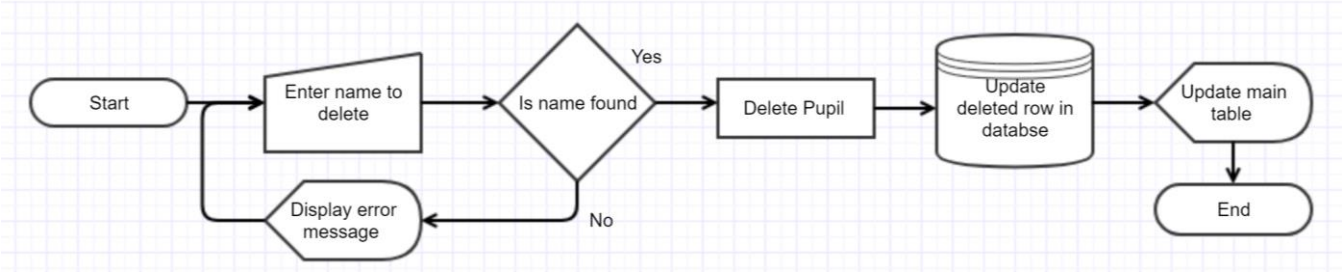
The flowchart illustrating the login method



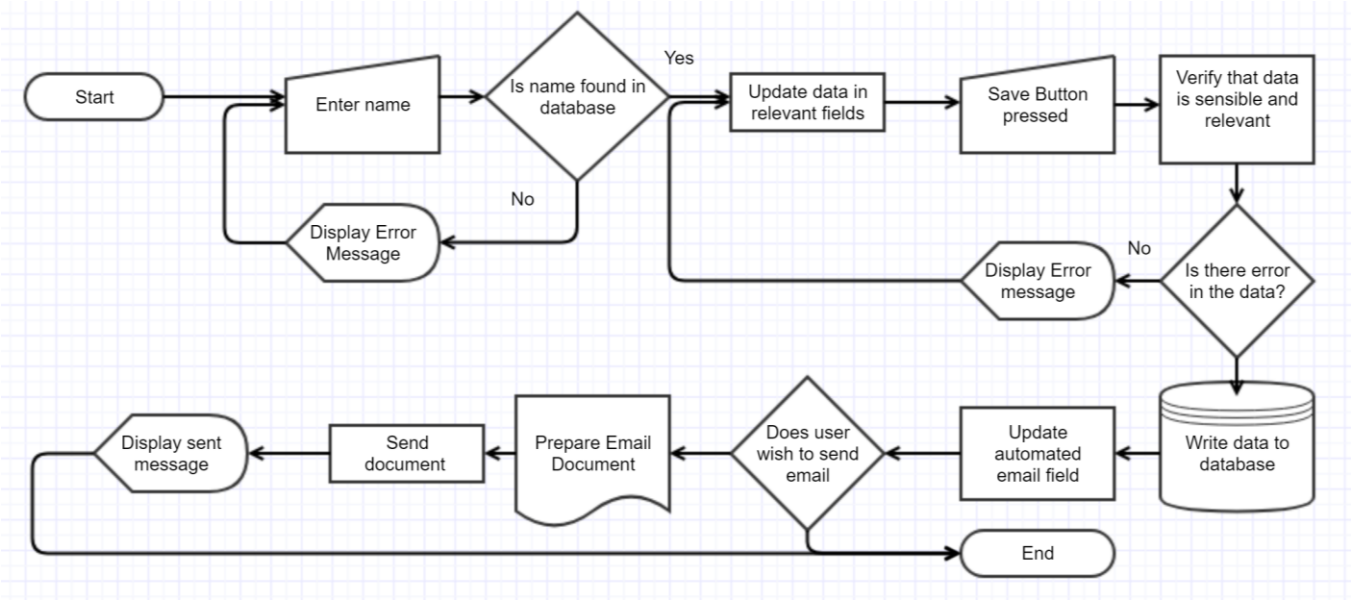
The flowchart for the search feature in the program



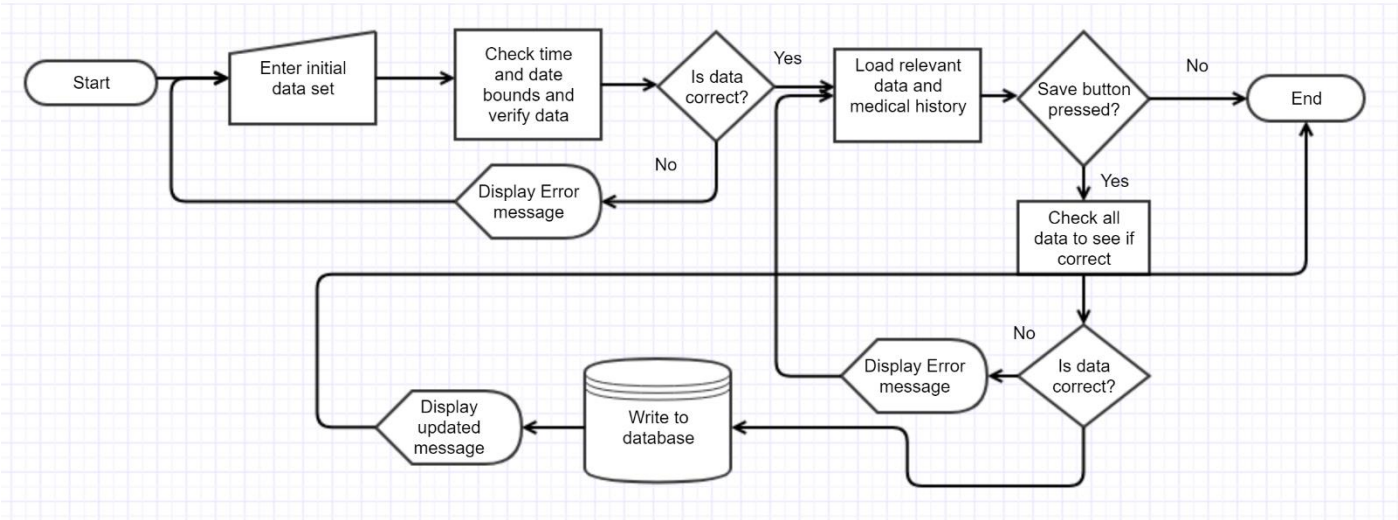
The flowchart outlining the method to add a pupil to the database



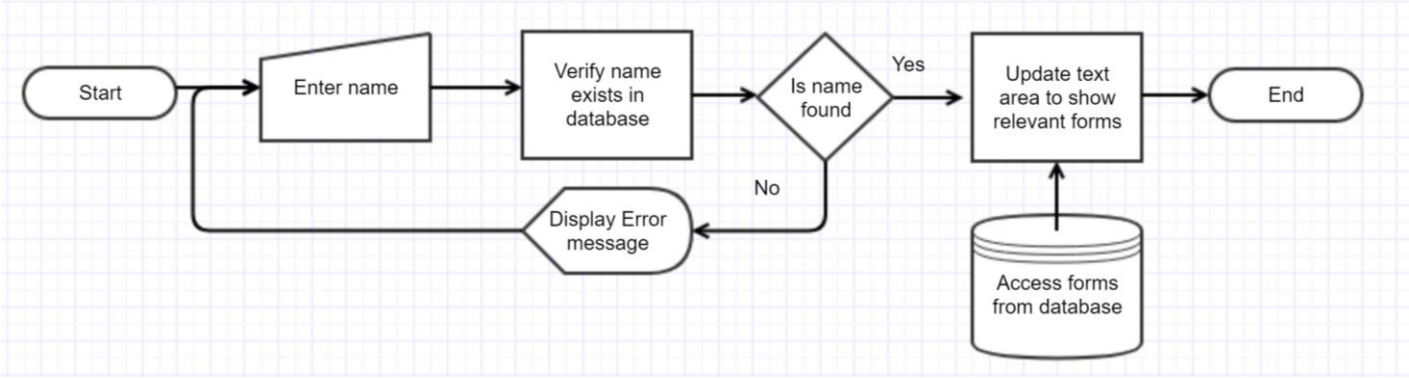
The flowchart illustrating the delete function of the program.



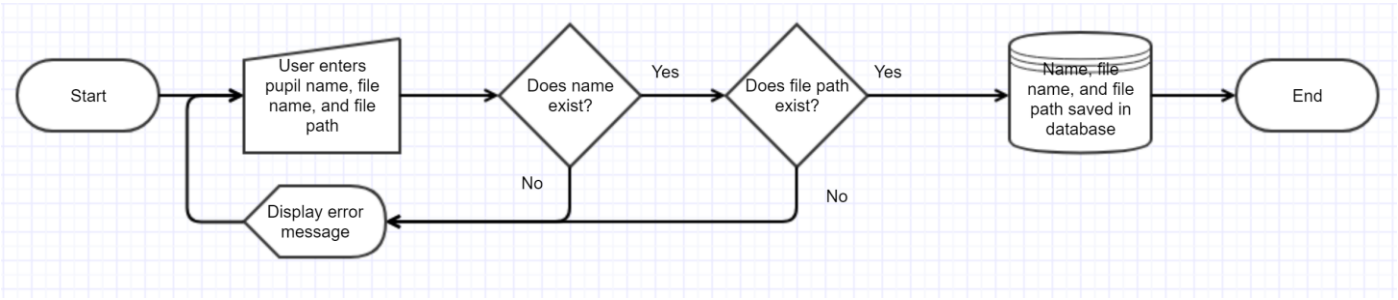
The flowchart showing the method to update the session in the program.



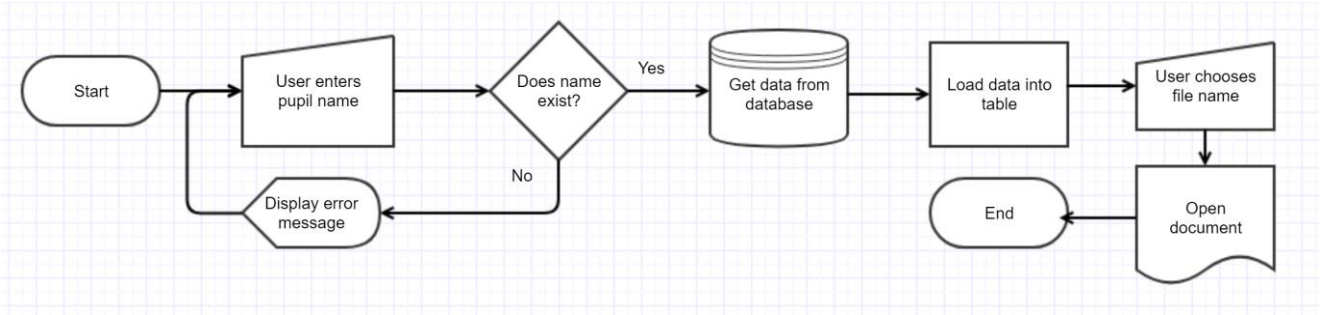
The flowchart to create a new emergency form.



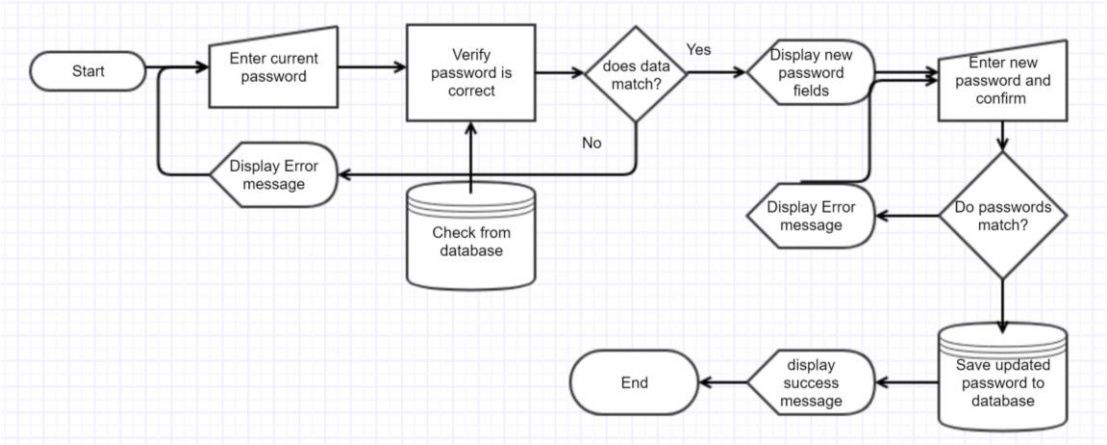
The flowchart to access a form



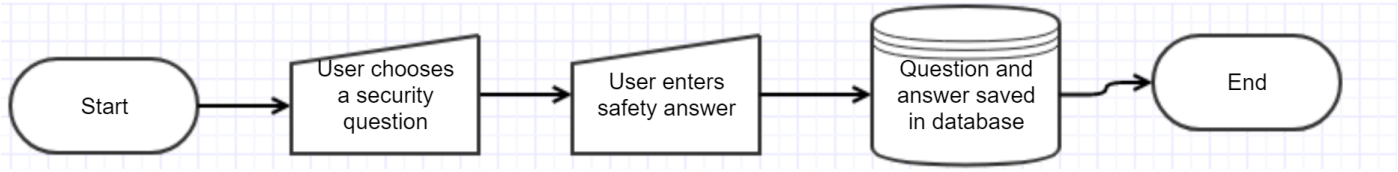
The flowchart to upload a document to the database



The flowchart used to access a document for a pupil from the database



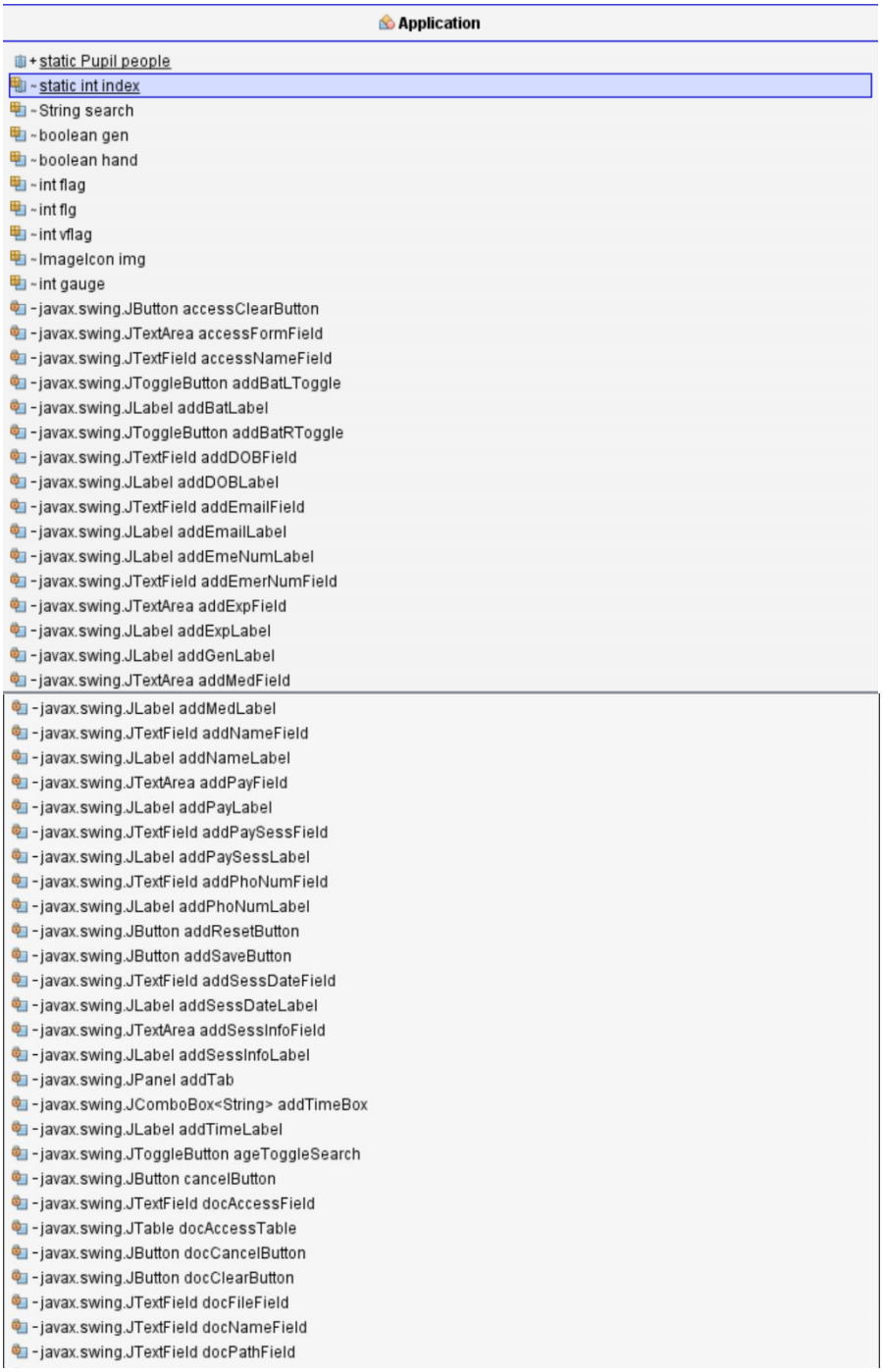
The flowchart for the user to update their password



The flowchart for the user to update their security question

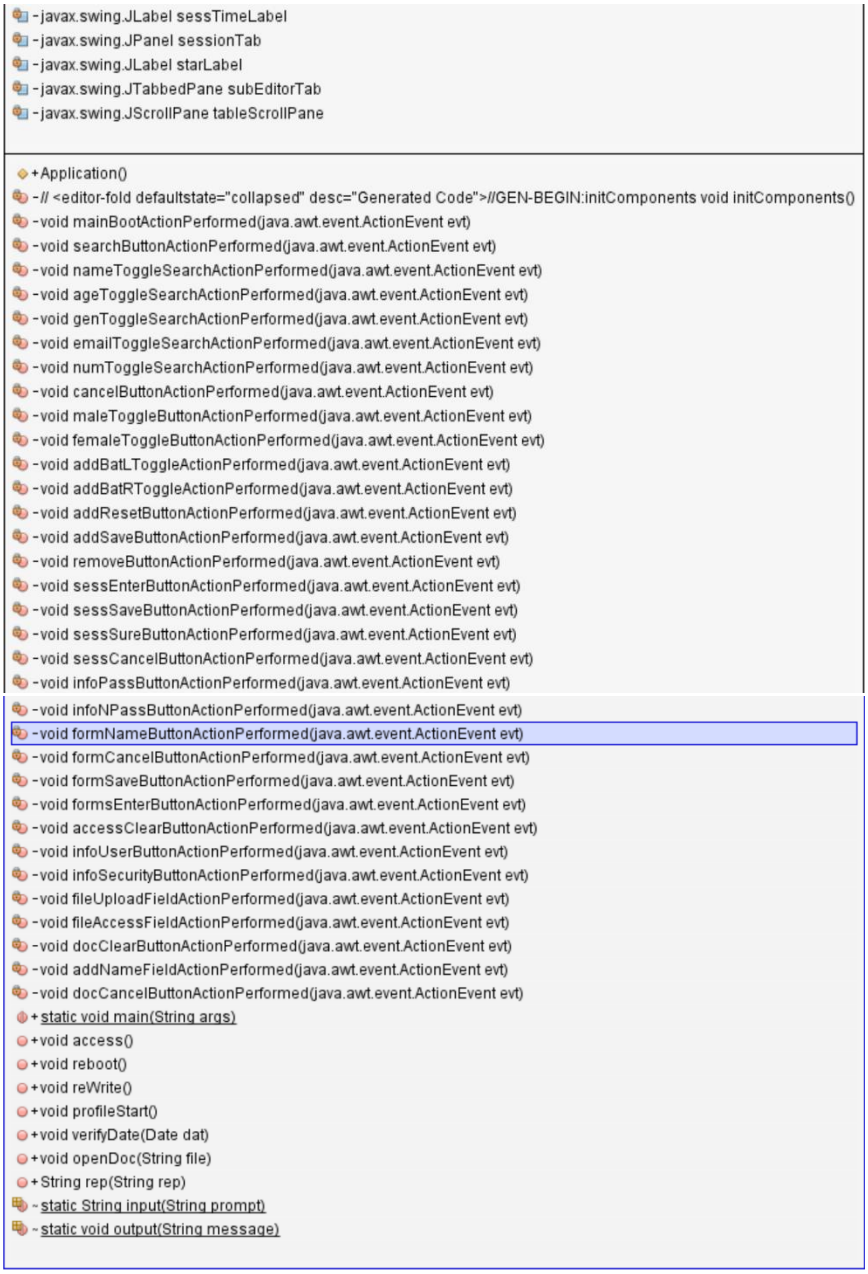
UML Diagrams

On the following page we can see the UML for the classes I plan on having, these were made using the plugin easyUML on NetBeans.

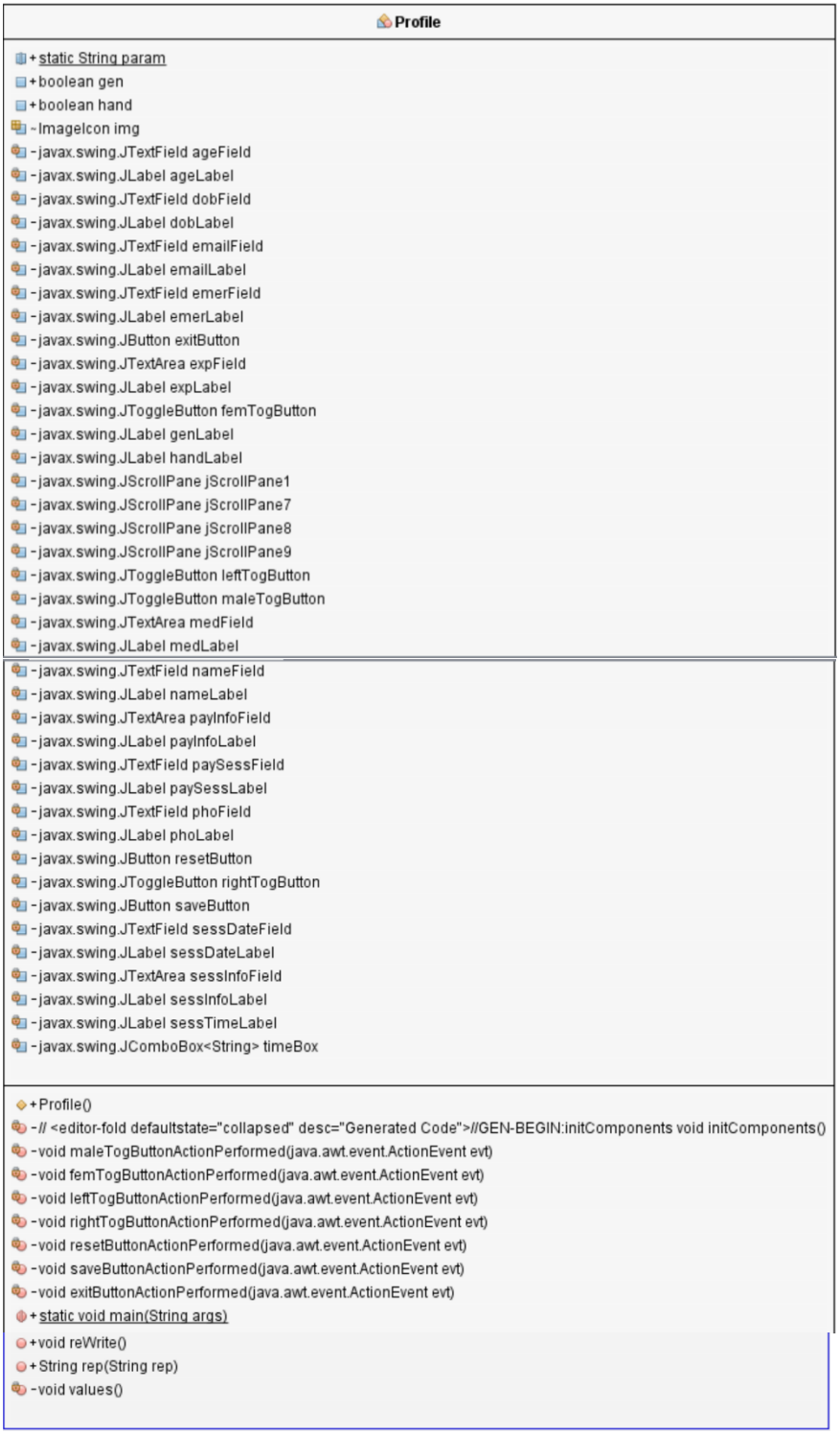


- javax.swing.JTabbedPane docsPane
- javax.swing.JPanel editorTab
- javax.swing.JToggleButton emailToggleSearch
- javax.swing.JToggleButton femaleToggleButton
- javax.swing.JButton fileAccessField
- javax.swing.JButton fileUploadField
- javax.swing.JButton formCancelButton
- javax.swing.JTextField formDateField
- javax.swing.JTextField formEmailField
- javax.swing.JTextField formEmerField
- javax.swing.JTextArea formInjuryField
- javax.swing.JTextArea formMedField
- javax.swing.JButton formNameButton
- javax.swing.JTextField formNameField
- javax.swing.JTextField formPhoField
- javax.swing.JButton formSaveButton
- javax.swing.JPanel formTab
- javax.swing.JComboBox<String> formTimeBox
- javax.swing.JButton formsEnterButton
- javax.swing.JToggleButton genToggleSearch
- javax.swing.JLabel infoCEMailLabel
- javax.swing.JPasswordField infoCPassField
- javax.swing.JLabel infoCPassLabel
- javax.swing.JLabel infoDateLabel
- javax.swing.JLabel infoIncomeLabel
- javax.swing.JLabel infoMEmailLabel
- javax.swing.JButton infoNPassButton
- javax.swing.JPasswordField infoNPassField
- javax.swing.JLabel jLabel7
- javax.swing.JLabel jLabel8
- javax.swing.JLabel jLabel9
- javax.swing.JPanel jPanel1
- javax.swing.JPanel jPanel2
- javax.swing.JPanel jPanel3
- javax.swing.JPanel jPanel4
- javax.swing.JPanel jPanel5
- javax.swing.JScrollPane jScrollPane1
- javax.swing.JScrollPane jScrollPane10
- javax.swing.JScrollPane jScrollPane12
- javax.swing.JScrollPane jScrollPane13
- javax.swing.JScrollPane jScrollPane2
- javax.swing.JScrollPane jScrollPane3
- javax.swing.JScrollPane jScrollPane4
- javax.swing.JScrollPane jScrollPane5
- javax.swing.JScrollPane jScrollPane6
- javax.swing.JScrollPane jScrollPane7
- javax.swing.JScrollPane jScrollPane8
- javax.swing.JScrollPane jScrollPane9
- javax.swing.JTabbedPane jTabbedPane1
- javax.swing.JTabbedPane jTabbedPane2
- javax.swing.JTextArea jTextArea4
- javax.swing.JTextField jTextField10
- javax.swing.JTextField jTextField8
- javax.swing.JButton mainBoot
- javax.swing.JPanel mainTab
- javax.swing.JTable mainTable

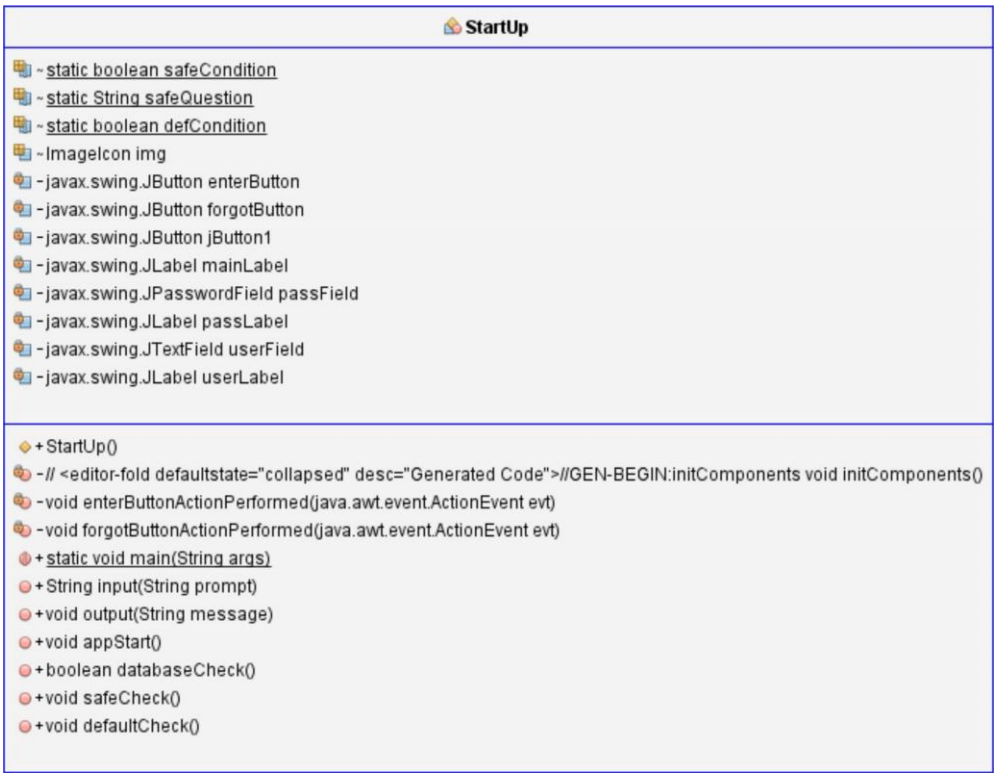
- javax.swing.JLabel infoNPassLabel
- javax.swing.JLabel infoNameLabel
- javax.swing.JLabel infoNumLabel
- javax.swing.JButton infoPassButton
- javax.swing.JPasswordField infoPassField
- javax.swing.JLabel infoPassLabel
- javax.swing.JButton infoSecurityButton
- javax.swing.JComboBox<String> infoSecurityCombo
- javax.swing.JTextField infoSecurityField
- javax.swing.JLabel infoSecurityLabel
- javax.swing.JPanel infoTab
- javax.swing.JButton infoUserButton
- javax.swing.JTextField infoUserField
- javax.swing.JLabel infoUserLabel
- javax.swing.JButton jButton5
- javax.swing.JLabel jLabel1
- javax.swing.JLabel jLabel10
- javax.swing.JLabel jLabel11
- javax.swing.JLabel jLabel12
- javax.swing.JLabel jLabel13
- javax.swing.JLabel jLabel14
- javax.swing.JLabel jLabel15
- javax.swing.JLabel jLabel16
- javax.swing.JLabel jLabel2
- javax.swing.JLabel jLabel3
- javax.swing.JLabel jLabel4
- javax.swing.JLabel jLabel5
- javax.swing.JLabel jLabel6
- javax.swing.JToggleButton maleToggleButton
- javax.swing.JToggleButton nameToggleSearch
- javax.swing.JToggleButton numToggleSearch
- javax.swing.JButton removeButton
- javax.swing.JTextField removeField
- javax.swing.JLabel removeLabel
- javax.swing.JPanel removeTab
- javax.swing.JButton searchButton
- javax.swing.JLabel searchLabel
- javax.swing.JTextField searchTextField
- javax.swing.JButton sessCancelButton
- javax.swing.JTextField sessDateField
- javax.swing.JLabel sessDateLabel
- javax.swing.JTextField sessEmailField
- javax.swing.JLabel sessEmailLabel
- javax.swing.JButton sessEnterButton
- javax.swing.JTextArea sessInfoField
- javax.swing.JLabel sessInfoLabel
- javax.swing.JTextField sessNameField
- javax.swing.JLabel sessNameLabel
- javax.swing.JTextField sessPayField
- javax.swing.JLabel sessPayLabel
- javax.swing.JButton sessSaveButton
- javax.swing.JTextArea sessSendField
- javax.swing.JLabel sessSendLabel
- javax.swing.JButton sessSureButton
- javax.swing.JTable sessTable
- javax.swing.JComboBox<String> sessTimeBox



The UML Diagram for the Application GUI class

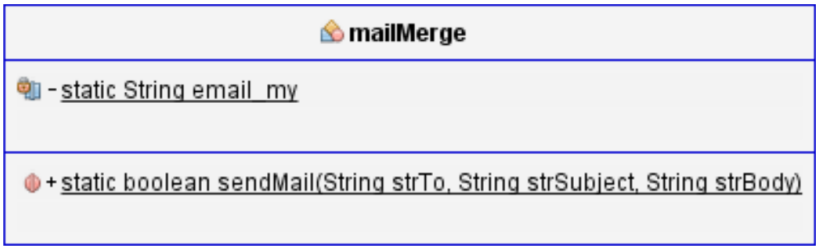


The UML diagram for the Profile GUI class

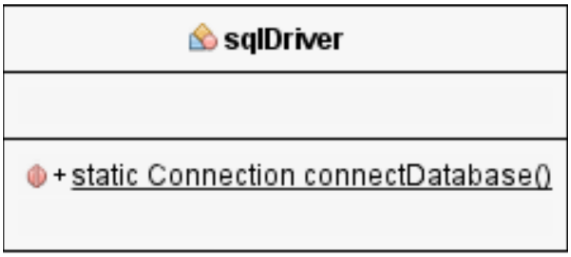


The UML diagram for the StartUp GUI class

The UML diagram for the Pupil object class



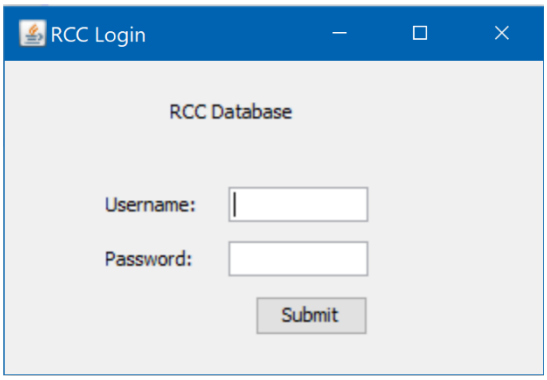
The UML diagram for the mailMerge class



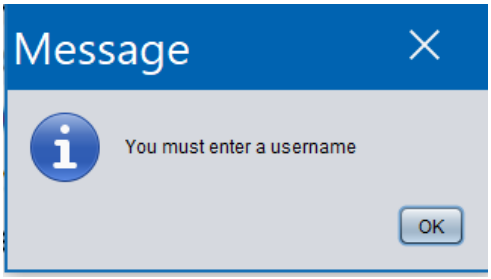
The UML diagram or the sqlDriver

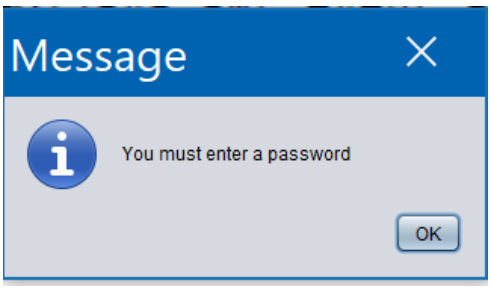
Program GUI

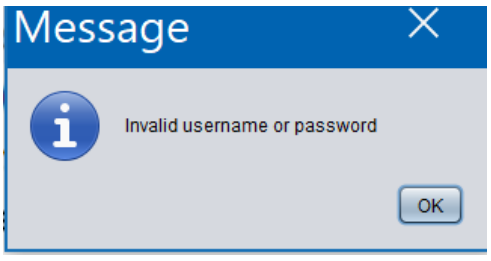
Login GUI



The outline for the login screen

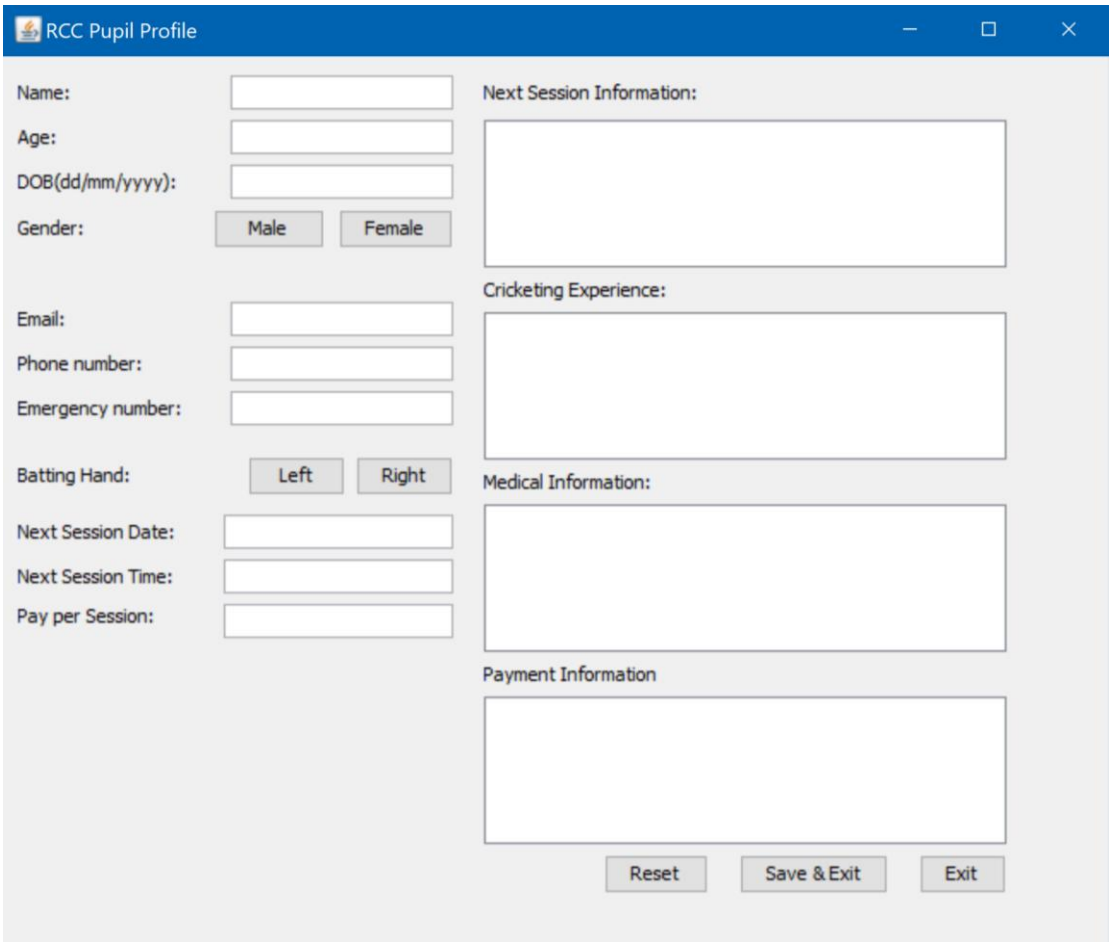






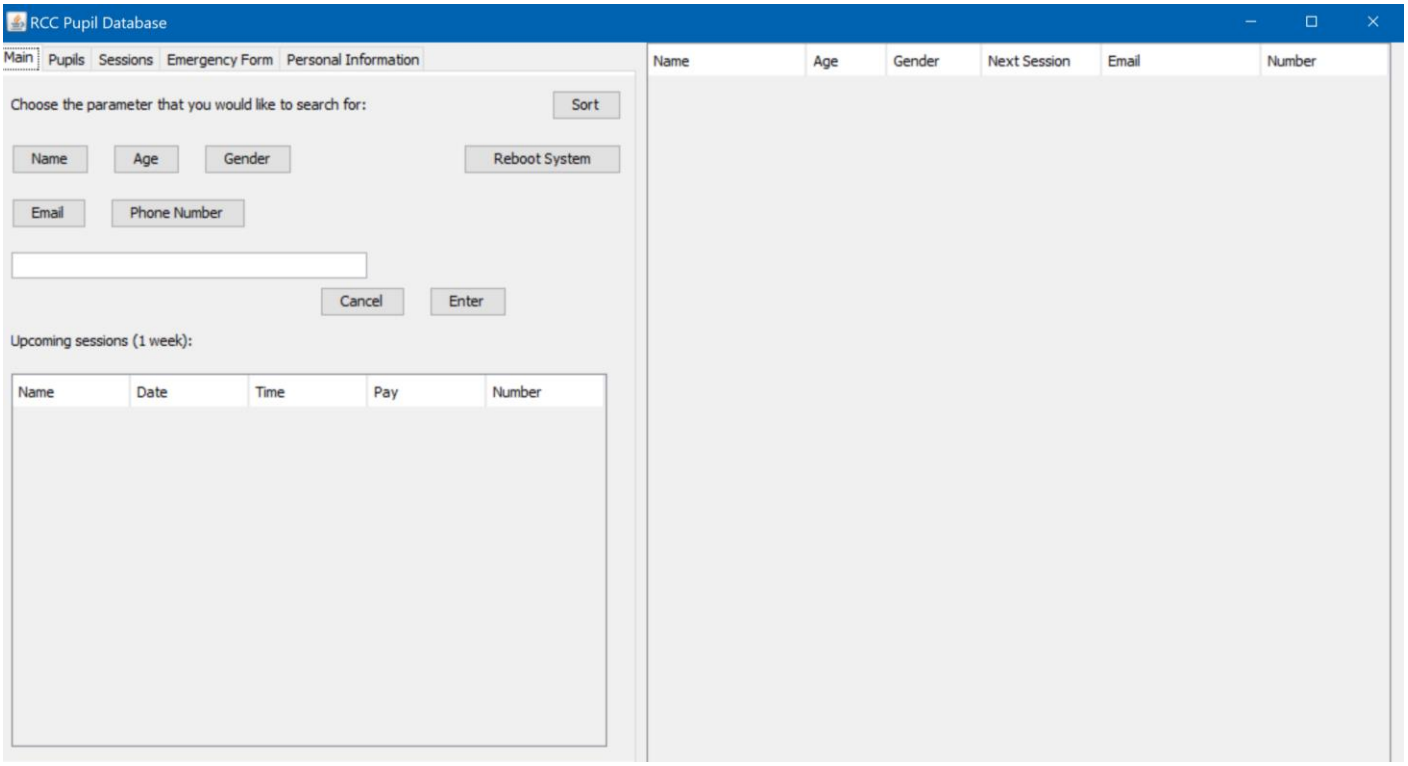
A series of prompts available to display to the user from the login screen depending on the situation.

Profile GUI

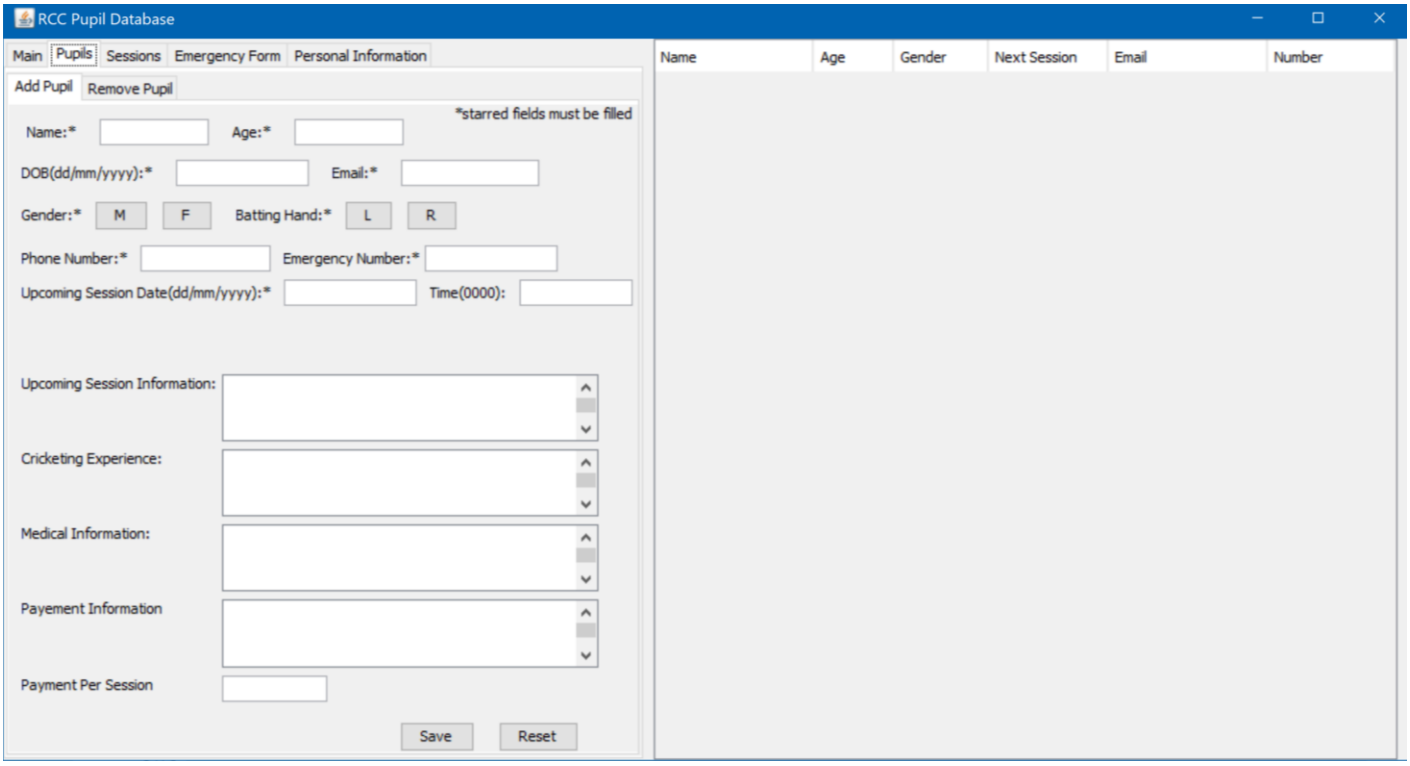


The screen used to edit the pupils and show all the information on them at once. These edits can also be saved in the database. I had originally planned for this to a tab in the main GUI, however after consultation and contact with my client and my advisor, I decided to make it a separate frame as it enabled ease of use.

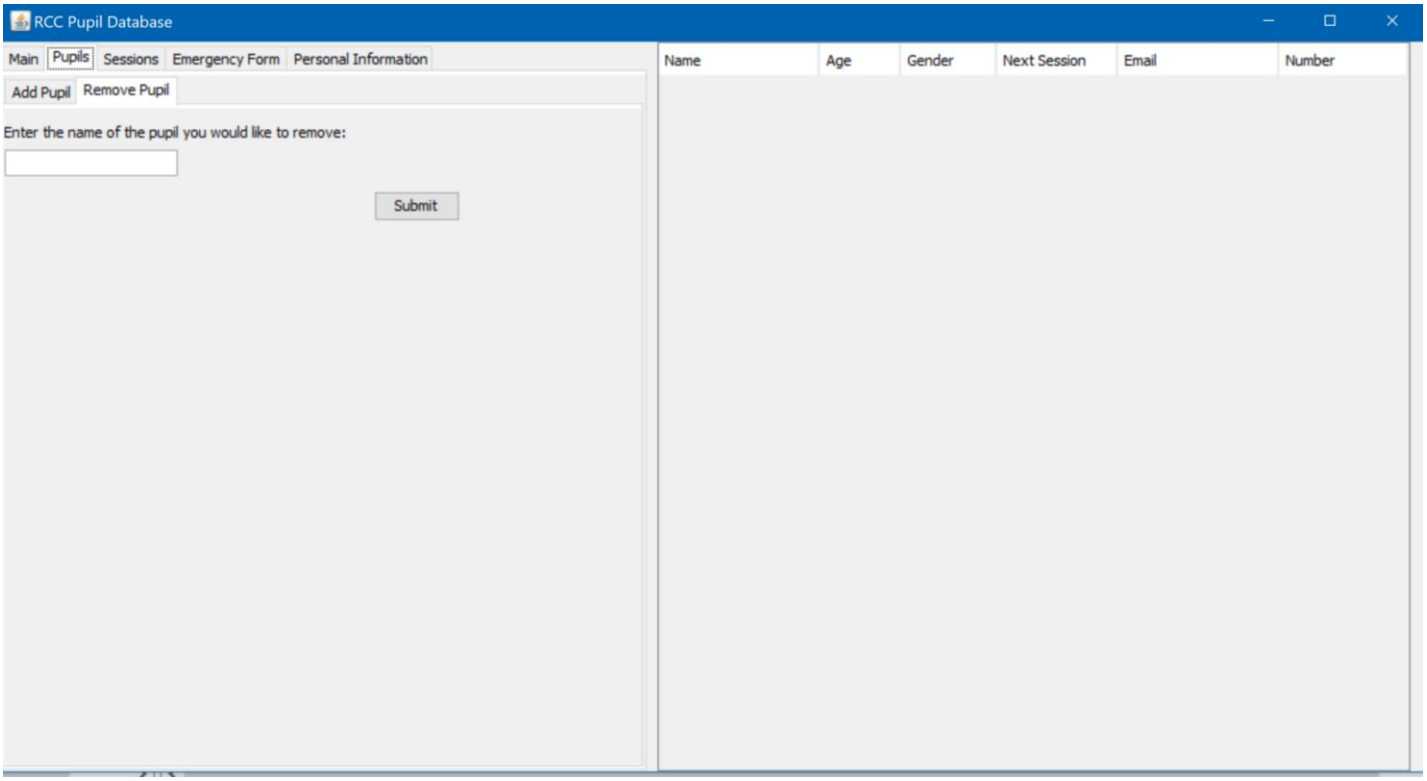
Application GUI



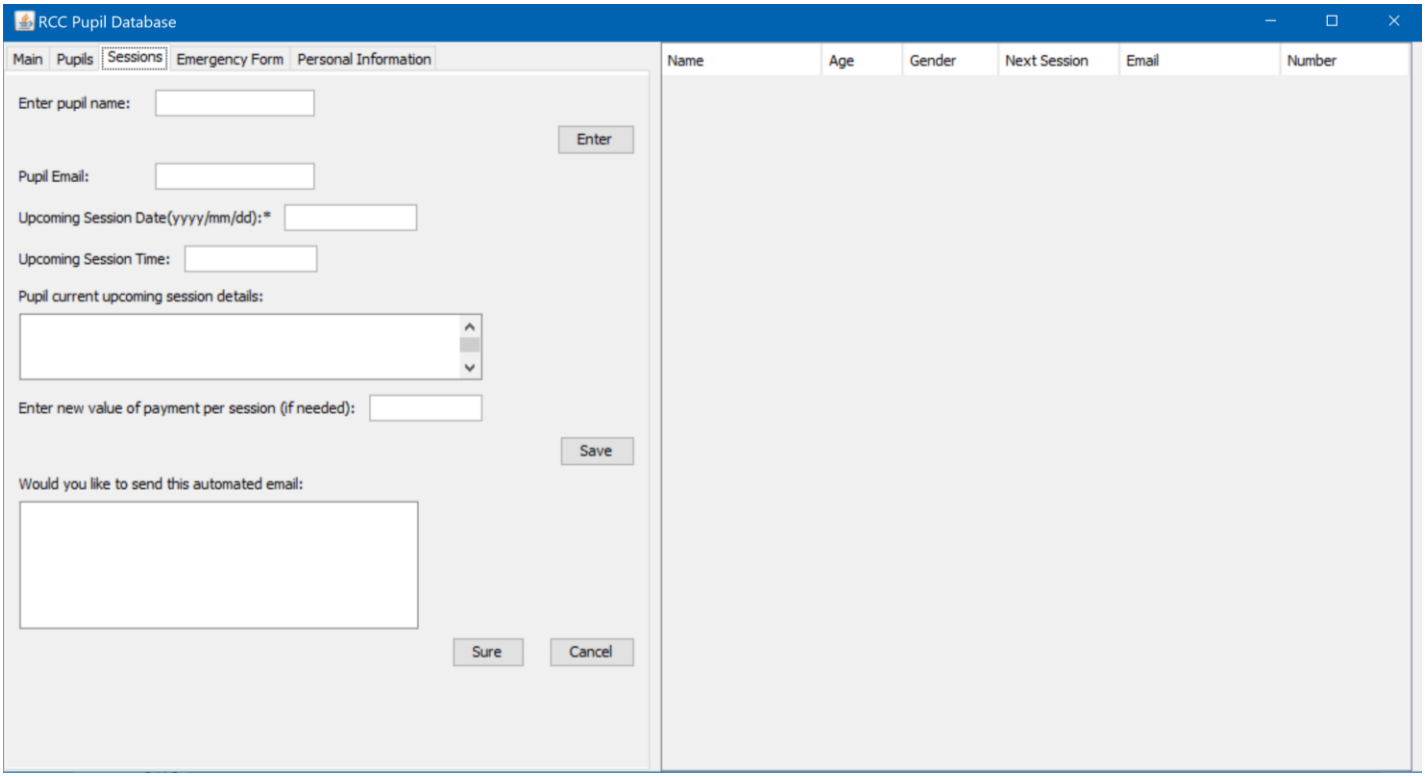
The main screen in the GUI, this is the first thing the user sees after logging in. The main table is showed alongside at all times. Here we see the search features and the sort. We can also use the table to show upcoming sessions.



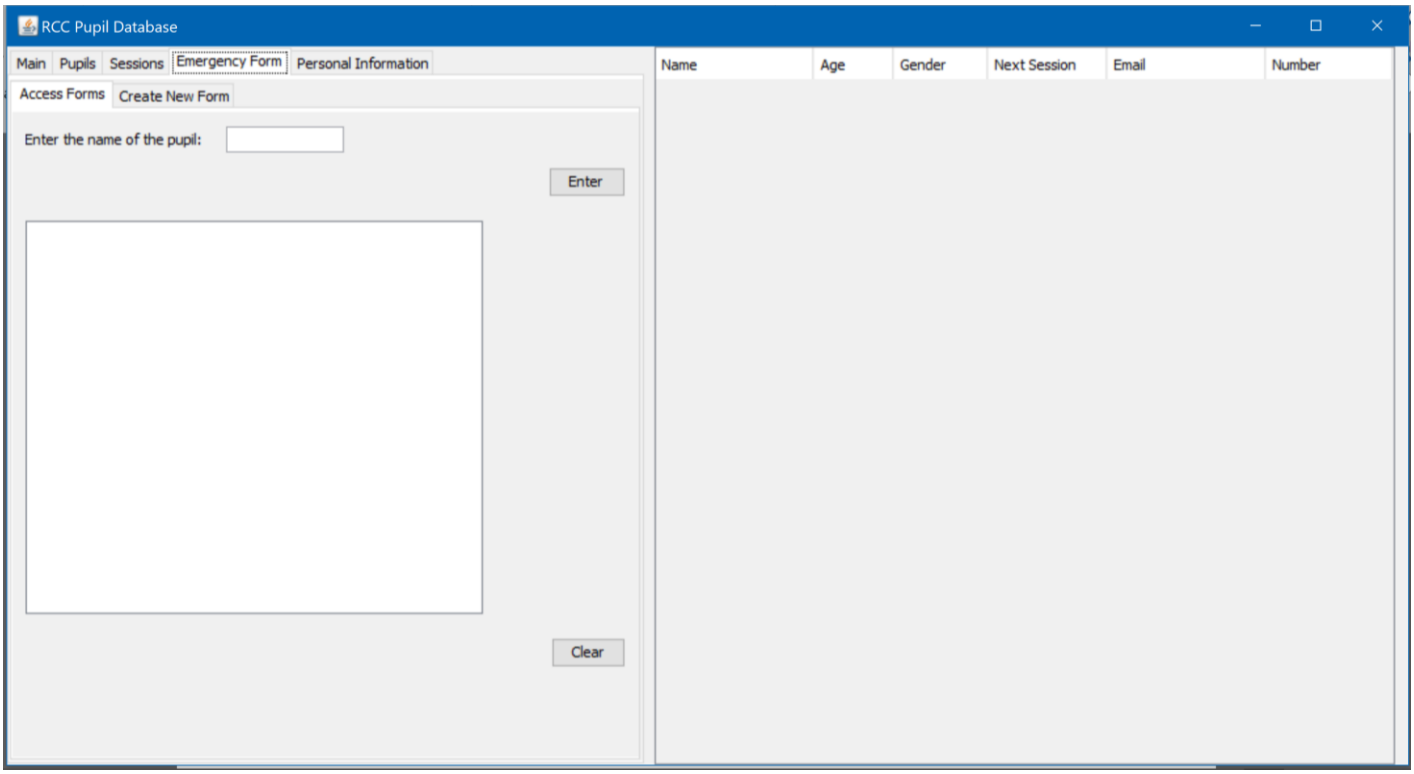
The screen to add pupils into the database.



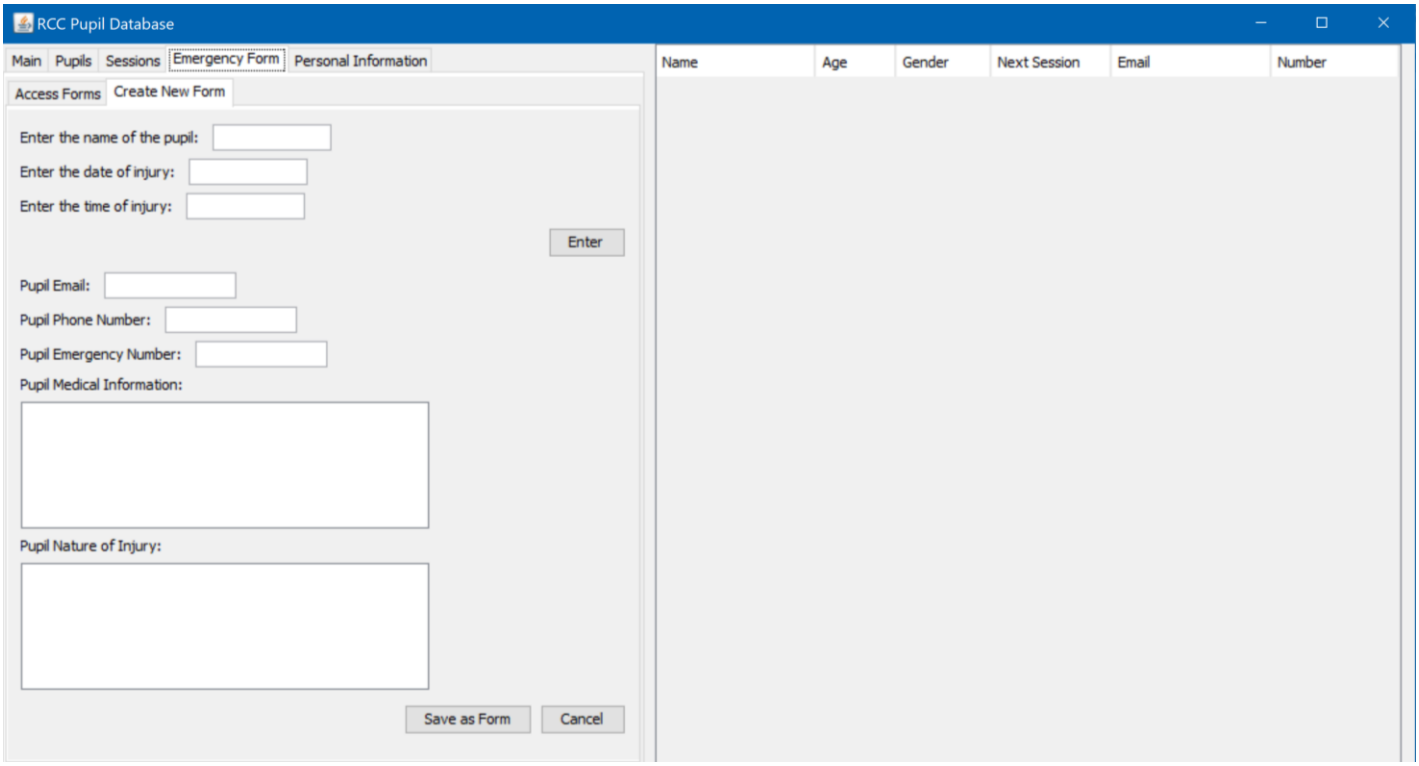
The screen for the user to delete pupils from the database.



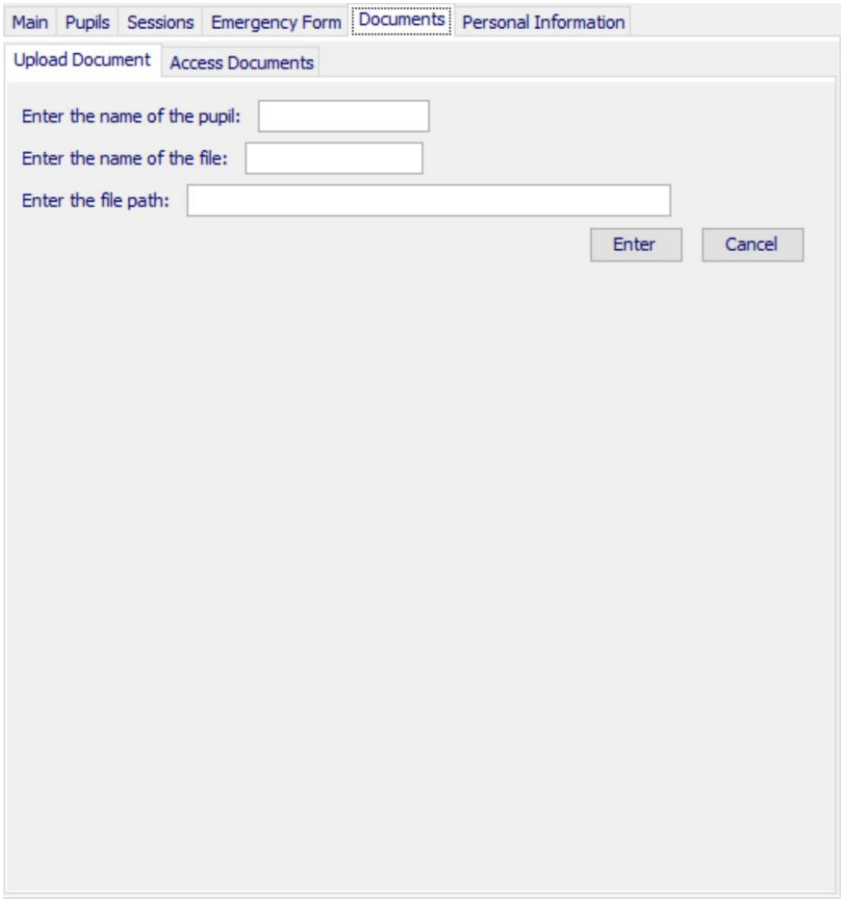
The screen which allows the user to manage pupil sessions and also send automated emails to his pupils reminding them of the session. I contacted my client and he suggested that the option to send the email to be directly after saving the session to make it efficeint.



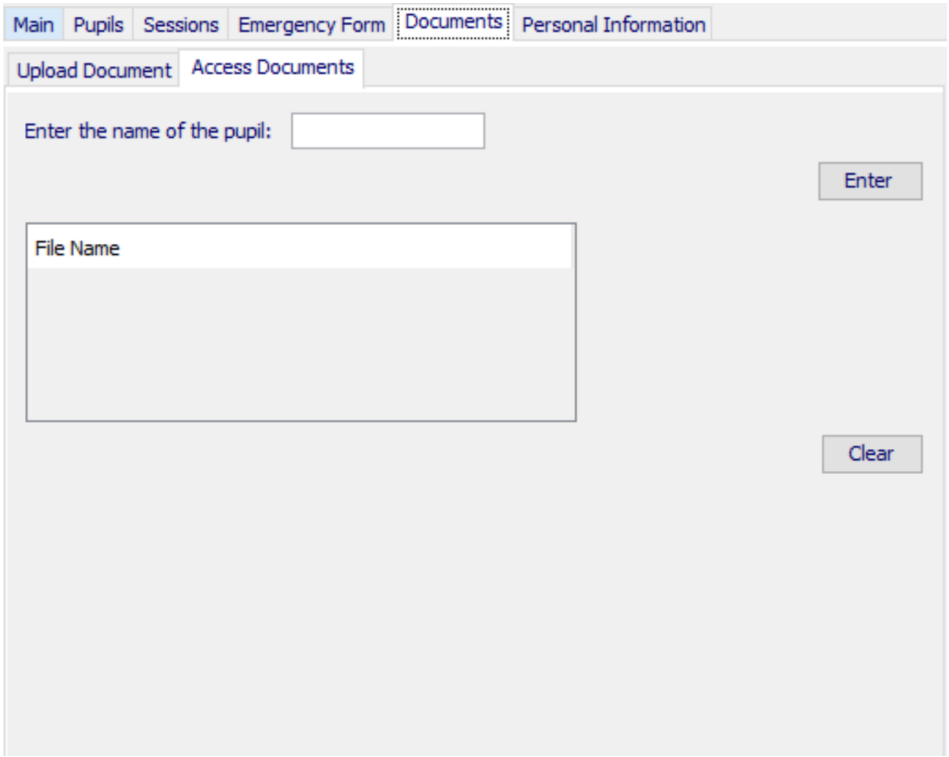
The screen shown for the user to access previous emergency forms



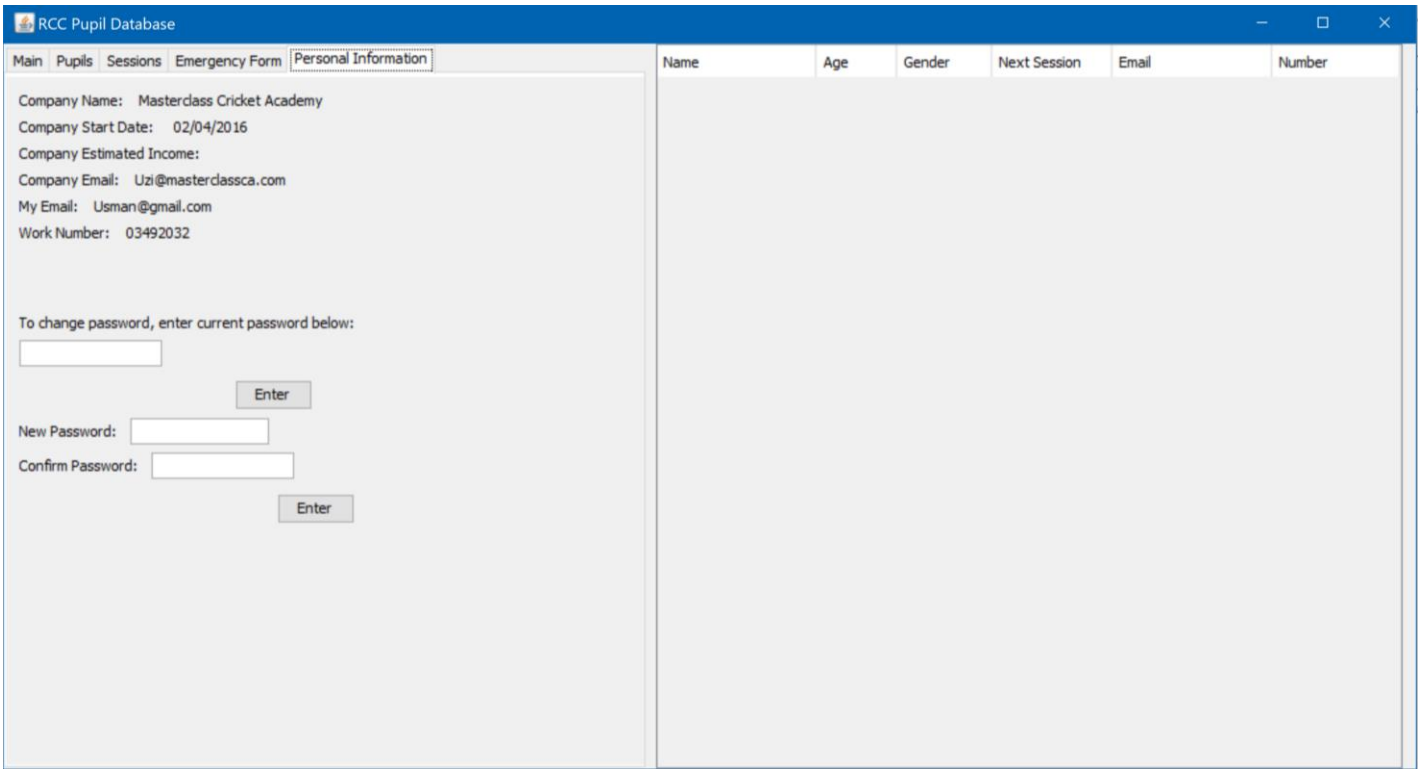
The screen for the user to create new forms and save them to the database.



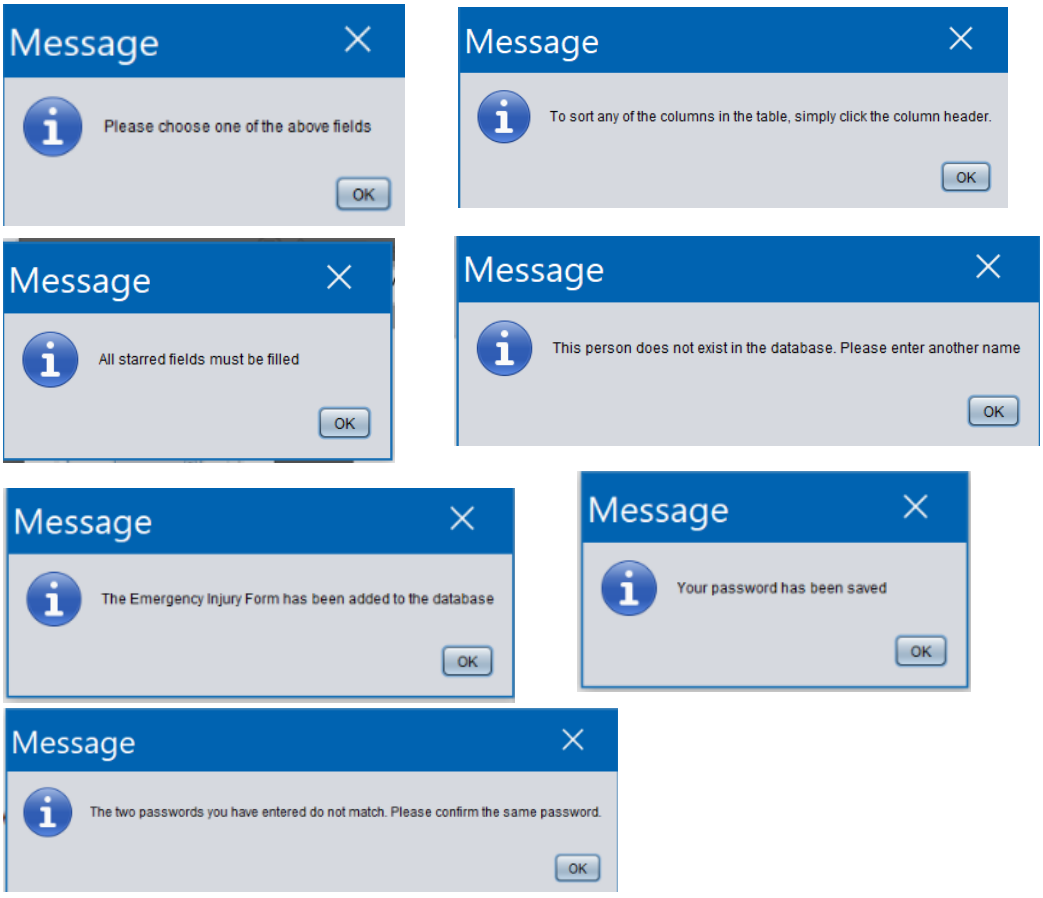
The screen to upload documents to the database



The screen to access documents from the database



The screen showing the company information and allowing the user to change his password.



This shows many of the possible UI messages that can show up throughout the program

Test Plan

Test Plan:

Actions to Test	Testing Method
Does program connect to database?	Initially I can make sure if this works by seeing whether or not the Login class effectively calls the correct details from the right SQL table.
Does Login check for wrong or out of bound values?	For this I will test for different sets of data and implement checks to make sure that the program effectively reduces the chance of error.
Does Search feature work?	Can be tested by making sure relevant error messages are shown when input errors are made and by manually checking the database, I can see if all the data relevant to the search is updated in the table.
Do main Table and sessions table show correct and relevant data?	This can also be checked by manually going through the SQL database to see if there are errors.
When edited through Profile class, is the database and are the other classes updated?	This can be checked by making sure that when the save button is pressed an output message confirms that the editing was successful. Checks can be placed for the field to see if the data was or wasn't relevant and correct. Then we can go and check the main table in the other class to see if the updates were made.
Is the Pupil added to the database?	Once again after making sure that the data is within a specified data bound and is relevant, through practical applications like SQL Workbench we can easily see if the database itself was also updated.
Is pupil deleted from database?	After making sure that the Pupil exists, the object can be deleted and the SQL table can be updated. Changes can be made sure through the use of the SQL Workbench or Command Client.
Is next session data correctly added?	We can know that the database is sending the correct data as this will be seen in whether or not the name is verified and whether or not the correct data is auto filled in the relevant fields. Subsequently when the user saves the changes, we should be able to see the changes on the sessions table on the main menu and also clearly be able to see them in the SQL database.
Is automated Email properly reloaded?	We can check this if the data in the email corresponds to the data previously entered and this can also be checked again by checking the Database.
Is the email sent?	We can check this by using our own email in the mailMerge system and seeing if we receive the automated email.

Are the emergency forms loaded properly for the corresponding name?	If the data is relevant to the Pupil and if the form can also be checked from the database manually then we can ensure that the form is loaded properly.
Is the new emergency form created?	This can be checked by the access forms feature as the new form will show up in the text area. We can also go back in the database and check.
Can files be uploaded into the database using the file path?	This can be checked using the programs access Documents tab which checks to see if the document is added. We can also go in the database and check.
Are the correct files loaded into the program for the corresponding name?	This can be checked by the user name in the database to see if the files do exist and that they were all loaded.
Can the password be successfully changed?	This can be checked by logging out and logging back in again. If the new password works then we can know that the password was updated. Once again, manually checking the database also works.
Is the program user friendly?	I will check this by asking the other students in my year to try it out and seeing if they are able to use the program well.
Multiple records can be stored?	This can be checked by adding multiple records and seeing if they all show up on the table and if the database is updated.
There are data checks?	This can be checked by entering slightly incorrect data and seeing whether or not if its rejected.