Funds Management

Criterion B: Design

Table of Contents

Class Functionality

The following highlights the primary functions of each of the 4 classes in the system

1. *Funds:* This is the funds request object class. Each object contains all the details that are required for a funds request to be complete
2. *MainScreen:* This is one of my GUI classes. This summarises all the funds from the 3 different divisions in the organisation. It also has buttons leading to 2 different classes
    1. InputForFundsRequests
    2. ManageFundsRequest
    3. Report1
    4. Report2
3. *InputForFundsRequests:* This is one of the sub-GUI classes in the program which can be accessed from the main screen. This allows the user to input all the fund requests sent by the different divisions. The user also has the option to edit their fund request
4. *ManageFundsRequest:* This is the other GUI class. This allows the user to
    - i. Either pay in instalments or in one go – putting in the date arranged
    - ii. Delete a fund request
    - iii. Search for a fund request
5. *Report1:* This shows a general and detailed report on the fund requests that are pending for the business to pay for
6. *Report2:* This shows a general and detailed report on the payments the business has made for each fund requests

Class Relationships

The following is a diagram illustrating the relationship between the program's 6 classes and was generated using the interactive Java development environment "Blue J".

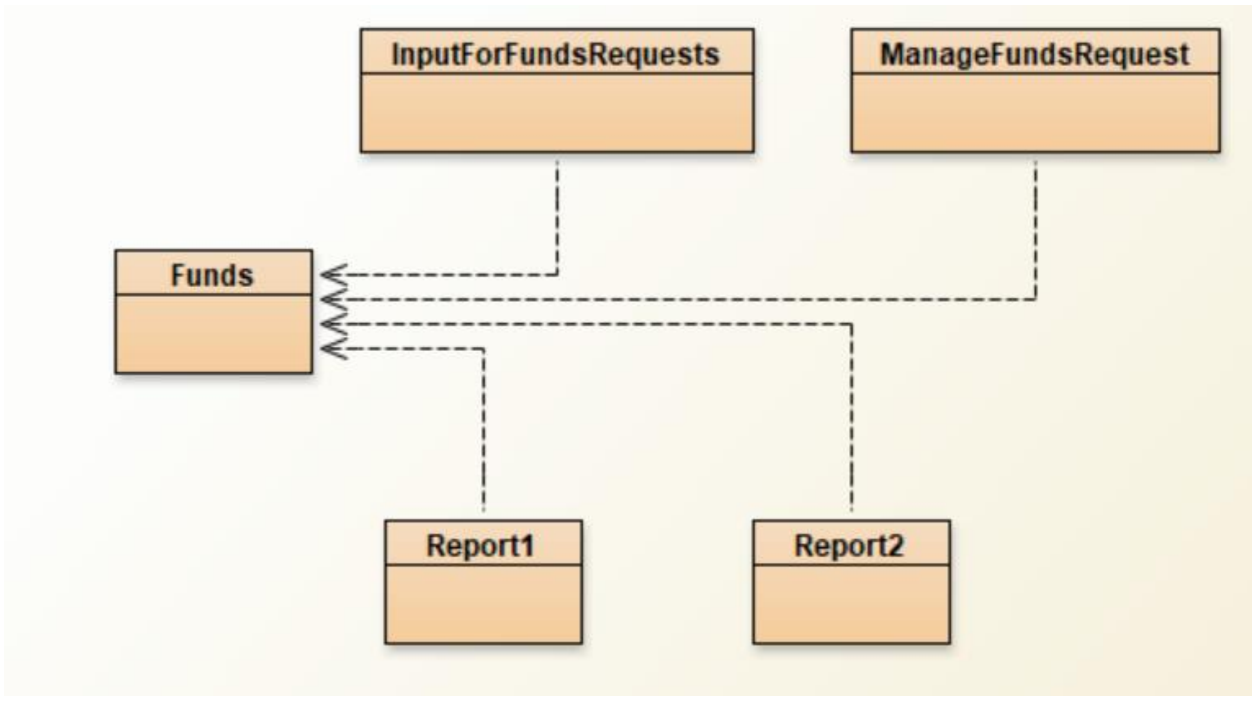
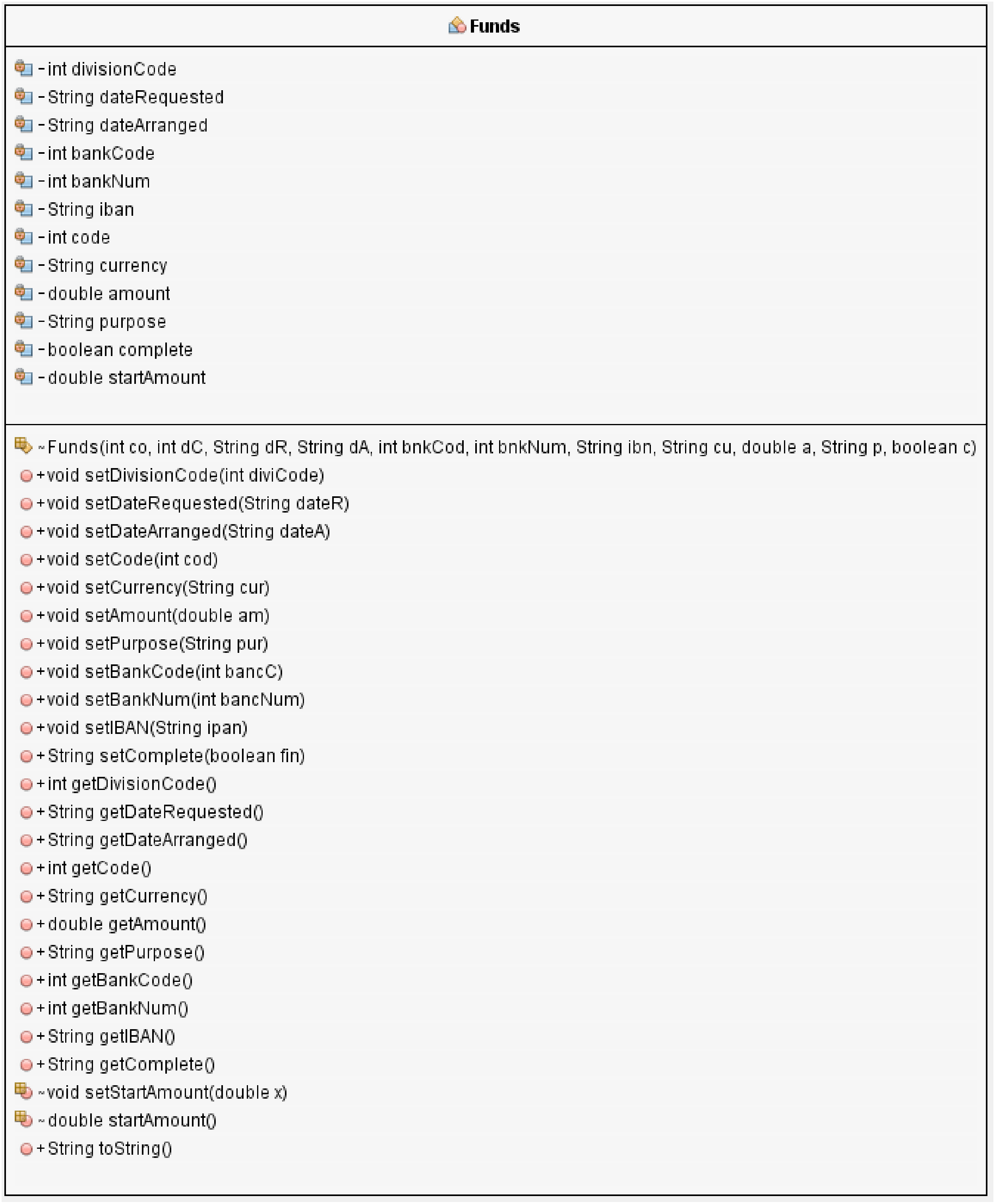


Figure 1: Class Relationships Diagram

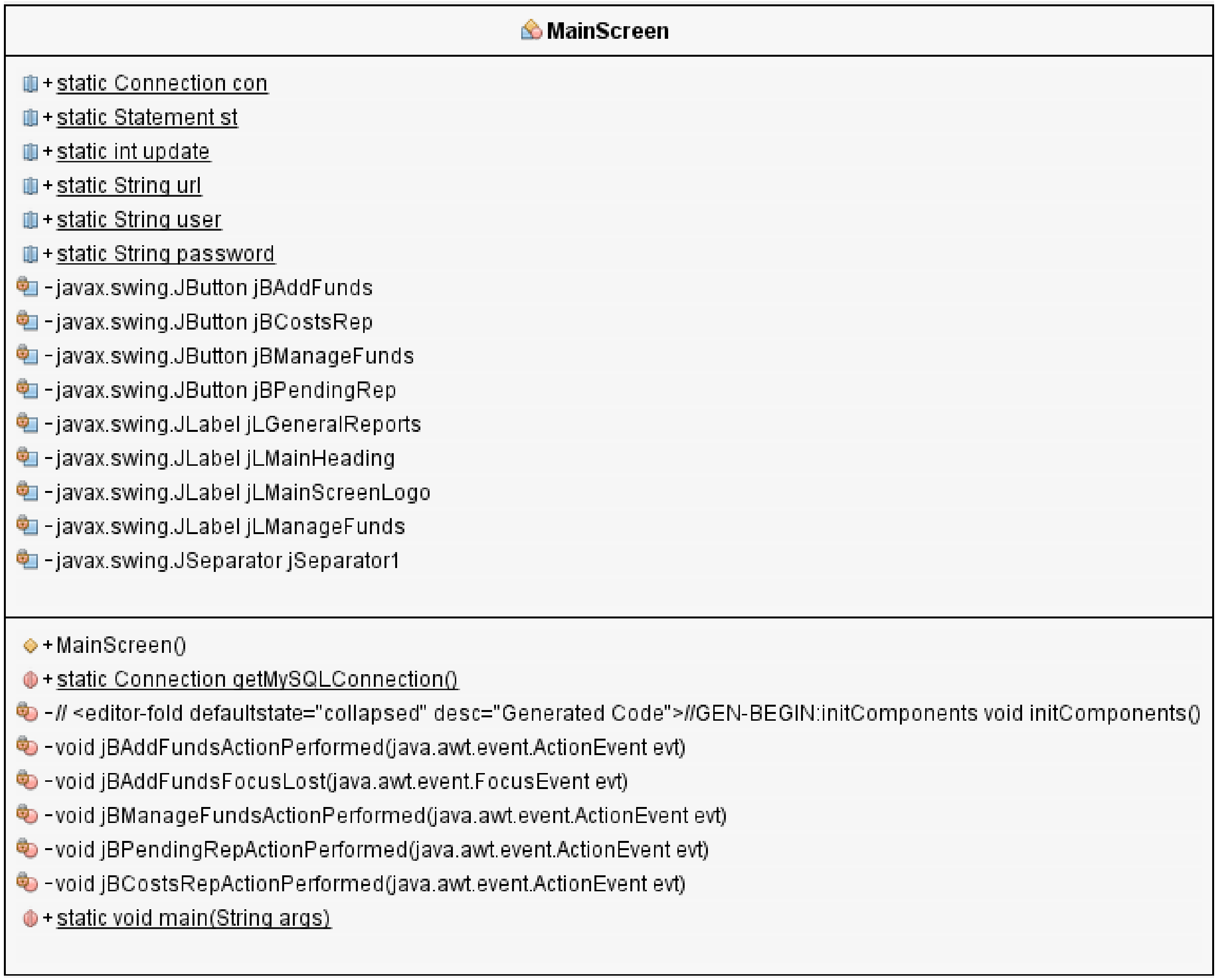
Class (Unified Model Language) Diagrams



```
Funds
-int divisionCode
-String dateRequested
-String dateArranged
-int bankCode
-int bankNum
-String iban
-int code
-String currency
-double amount
-String purpose
-boolean complete
-double startAmount

~Funds(int co, int dC, String dR, String dA, int bnkCod, int bnkNum, String ibn, String cu, double a, String p, boolean c)
+void setDivisionCode(int diviCode)
+void setDateRequested(String dateR)
+void setDateArranged(String dateA)
+void setCode(int cod)
+void setCurrency(String cur)
+void setAmount(double am)
+void setPurpose(String pur)
+void setBankCode(int bancC)
+void setBankNum(int bancNum)
+void setIBAN(String ipan)
+String setComplete(boolean fin)
+int getDivisionCode()
+String getDateRequested()
+String getDateArranged()
+int getCode()
+String getCurrency()
+double getAmount()
+String getPurpose()
+int getBankCode()
+int getBankNum()
+String getIBAN()
+String getComplete()
~void setStartAmount(double x)
~double startAmount()
+String toString()
```

Figure 2: UML Diagram for the Funds object class. It contains all the necessary variables to create a fund request object

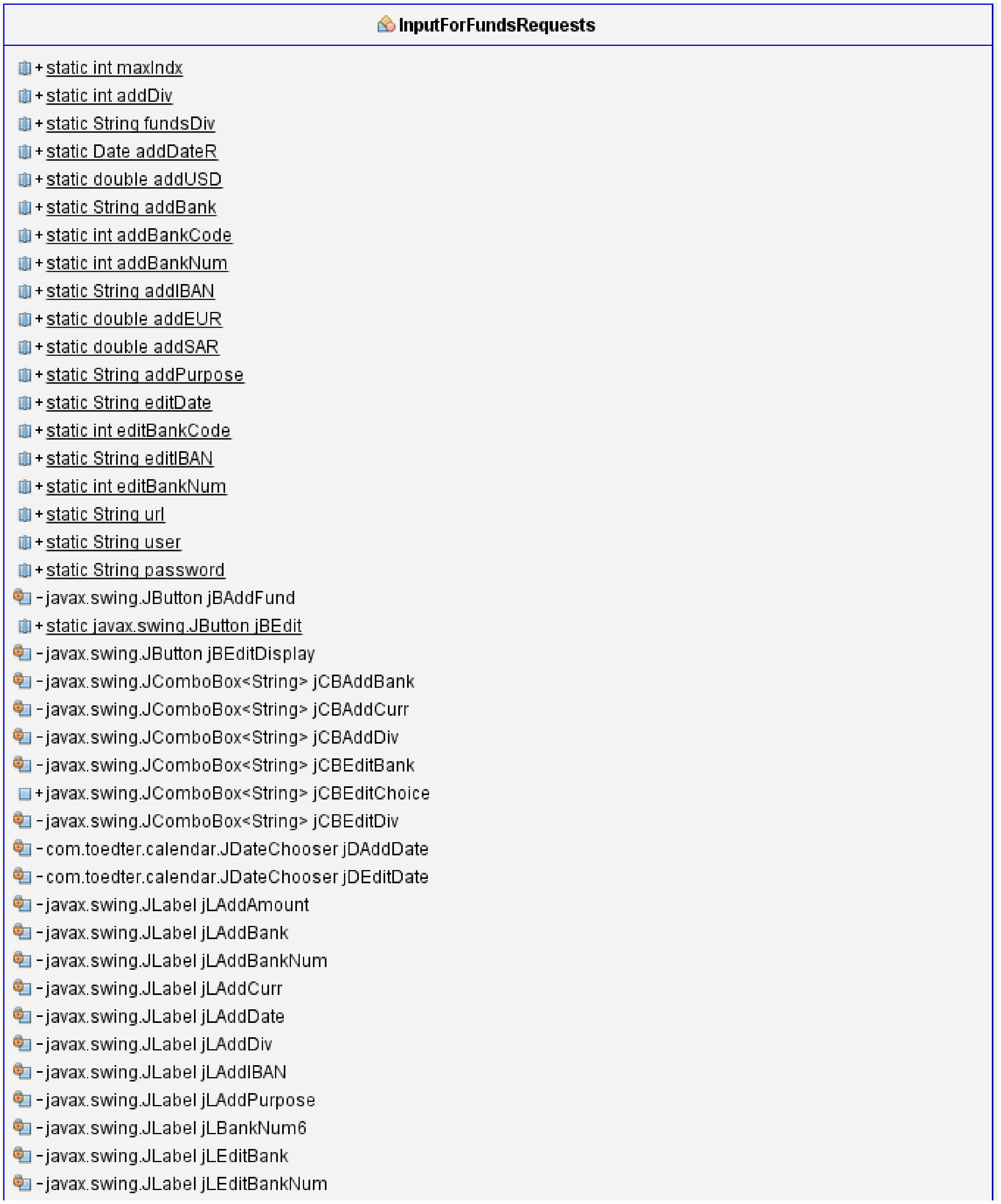


MainScreen

```
+static Connection con
+static Statement st
+static int update
+static String url
+static String user
+static String password
-javax.swing.JButton jBAddFunds
-javax.swing.JButton jBCostsRep
-javax.swing.JButton jBManageFunds
-javax.swing.JButton jBPendingRep
-javax.swing.JLabel jLGeneralReports
-javax.swing.JLabel jLMainHeading
-javax.swing.JLabel jLMainScreenLogo
-javax.swing.JLabel jLManageFunds
-javax.swing.JSeparator jSeparator1
```

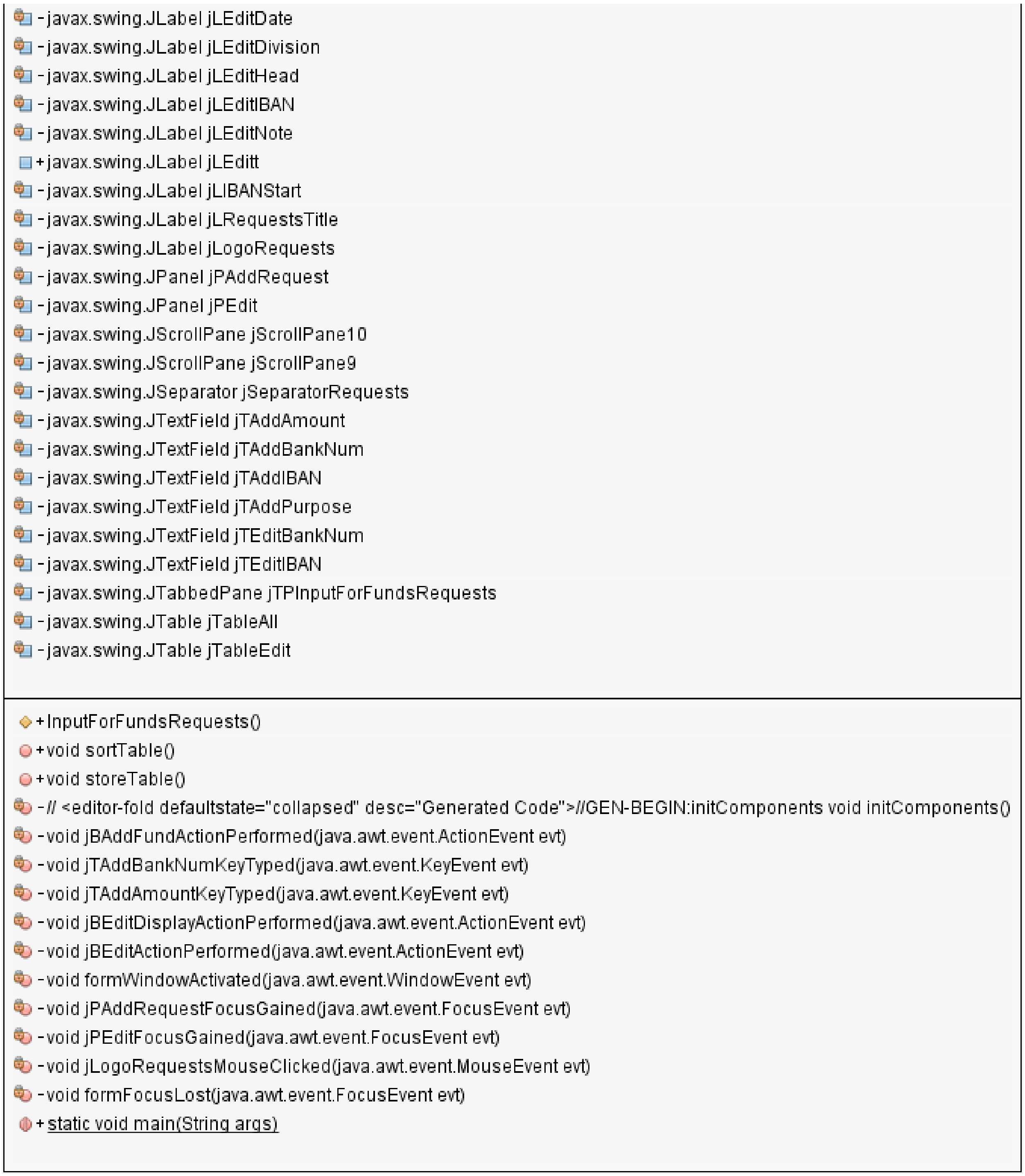
```
+MainScreen()
+static Connection getMySQLConnection()
-// <editor-fold defaultstate="collapsed" desc="Generated Code">//GEN-BEGIN:initComponents void initComponents()
-void jBAddFundsActionPerformed(java.awt.event.ActionEvent evt)
-void jBAddFundsFocusLost(java.awt.event.FocusEvent evt)
-void jBManageFundsActionPerformed(java.awt.event.ActionEvent evt)
-void jBPendingRepActionPerformed(java.awt.event.ActionEvent evt)
-void jBCostsRepActionPerformed(java.awt.event.ActionEvent evt)
+static void main(String args)
```

Figure 3: UML Diagram for the Main Screen class. This is one of the GUIs in this program and it will be the first frame to show up when the user opens the JAR file



InputForFundsRequests

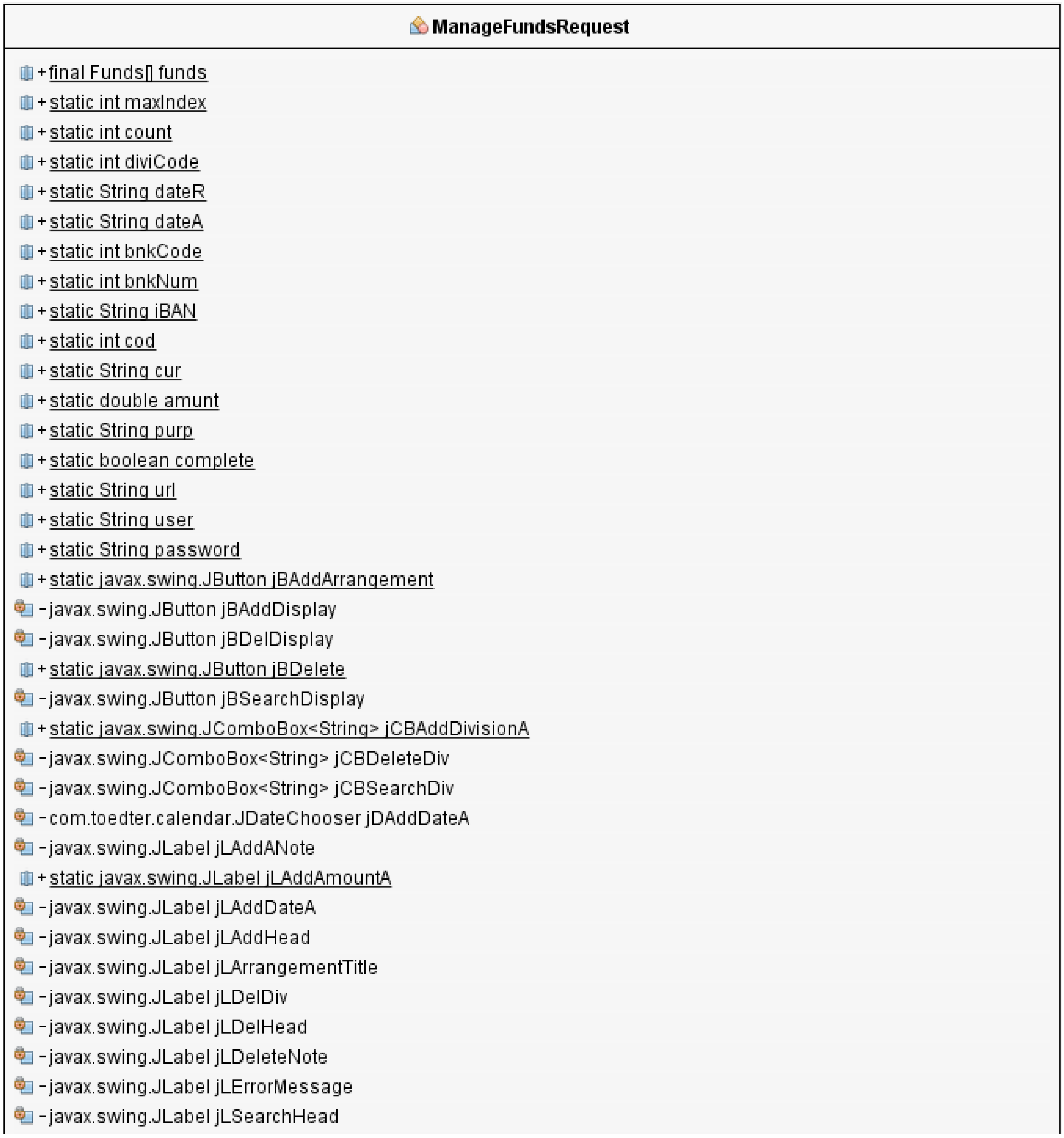
```
+static int maxIndx
+static int addDiv
+static String fundsDiv
+static Date addDateR
+static double addUSD
+static String addBank
+static int addBankCode
+static int addBankNum
+static String addIBAN
+static double addEUR
+static double addSAR
+static String addPurpose
+static String editDate
+static int editBankCode
+static String editIBAN
+static int editBankNum
+static String url
+static String user
+static String password
-javax.swing.JButton jBAddFund
+static javax.swing.JButton jBEdit
-javax.swing.JButton jBEditDisplay
-javax.swing.JComboBox<String> jCBAddBank
-javax.swing.JComboBox<String> jCBAddCurr
-javax.swing.JComboBox<String> jCBAddDiv
-javax.swing.JComboBox<String> jCBEditBank
+javax.swing.JComboBox<String> jCBEditChoice
-javax.swing.JComboBox<String> jCBEditDiv
-com.toedter.calendar.JDateChooser jDAddDate
-com.toedter.calendar.JDateChooser jDEditDate
-javax.swing.JLabel jLAddAmount
-javax.swing.JLabel jLAddBank
-javax.swing.JLabel jLAddBankNum
-javax.swing.JLabel jLAddCurr
-javax.swing.JLabel jLAddDate
-javax.swing.JLabel jLAddDiv
-javax.swing.JLabel jLAddIBAN
-javax.swing.JLabel jLAddPurpose
-javax.swing.JLabel jLBankNum6
-javax.swing.JLabel jLEditBank
-javax.swing.JLabel jLEditBankNum
```



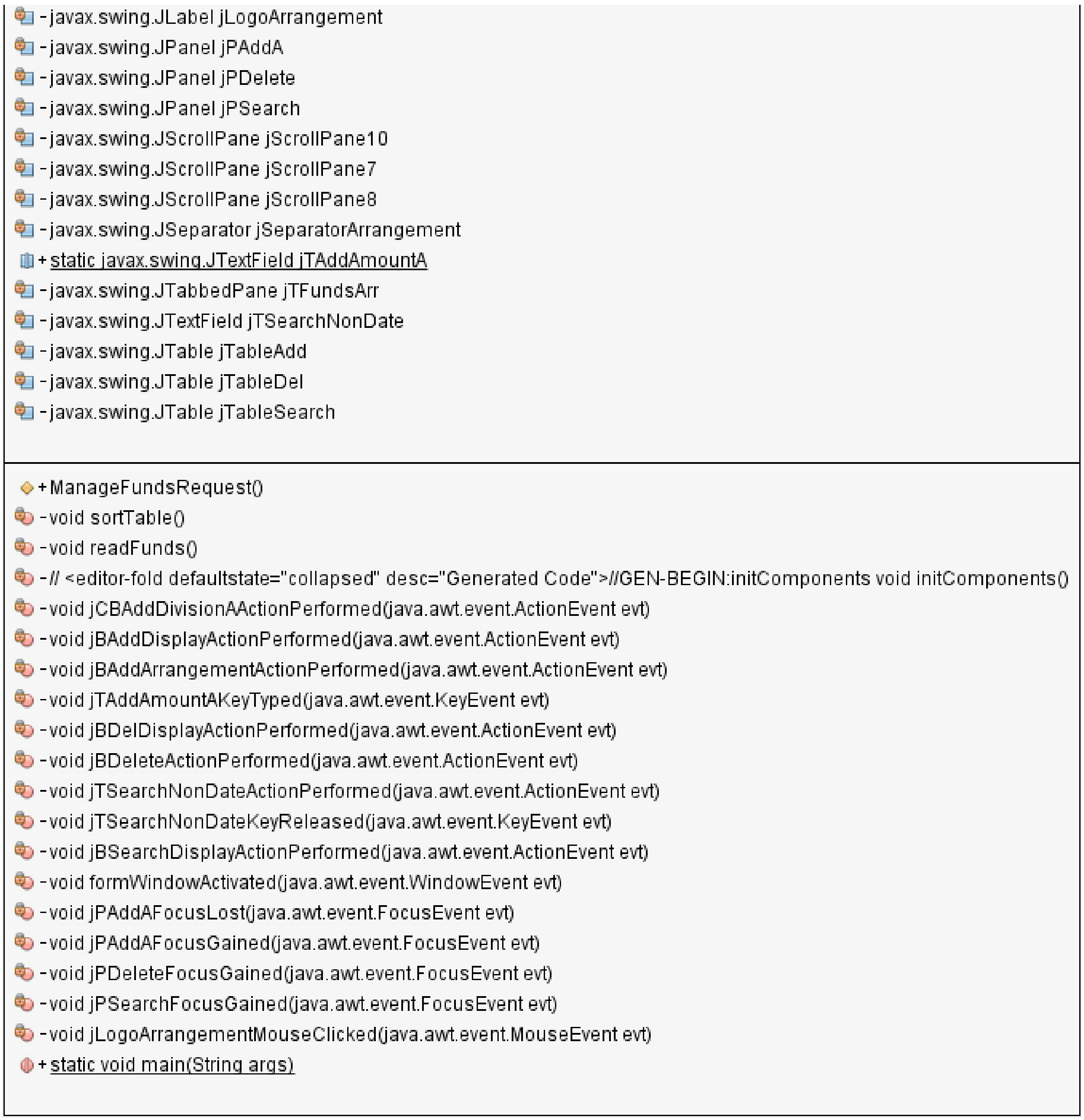
```
-javax.swing.JLabel jLEditDate
-javax.swing.JLabel jLEditDivision
-javax.swing.JLabel jLEditHead
-javax.swing.JLabel jLEditIBAN
-javax.swing.JLabel jLEditNote
+javax.swing.JLabel jLEditt
-javax.swing.JLabel jLIBANStart
-javax.swing.JLabel jLRequestsTitle
-javax.swing.JLabel jLogoRequests
-javax.swing.JPanel jPAddRequest
-javax.swing.JPanel jPEdit
-javax.swing.JScrollPane jScrollPane10
-javax.swing.JScrollPane jScrollPane9
-javax.swing.JSeparator jSeparatorRequests
-javax.swing.JTextField jTAddAmount
-javax.swing.JTextField jTAddBankNum
-javax.swing.JTextField jTAddIBAN
-javax.swing.JTextField jTAddPurpose
-javax.swing.JTextField jTEditBankNum
-javax.swing.JTextField jTEditIBAN
-javax.swing.JTabbedPane jTPInputForFundsRequests
-javax.swing.JTable jTableAll
-javax.swing.JTable jTableEdit

+InputForFundsRequests()
+void sortTable()
+void storeTable()
-// <editor-fold defaultstate="collapsed" desc="Generated Code">//GEN-BEGIN:initComponents void initComponents()
-void jBAddFundActionPerformed(java.awt.event.ActionEvent evt)
-void jTAddBankNumKeyTyped(java.awt.event.KeyEvent evt)
-void jTAddAmountKeyTyped(java.awt.event.KeyEvent evt)
-void jBEditDisplayActionPerformed(java.awt.event.ActionEvent evt)
-void jBEditActionPerformed(java.awt.event.ActionEvent evt)
-void formWindowActivated(java.awt.event.WindowEvent evt)
-void jPAddRequestFocusGained(java.awt.event.FocusEvent evt)
-void jPEditFocusGained(java.awt.event.FocusEvent evt)
-void jLogoRequestsMouseClicked(java.awt.event.MouseEvent evt)
-void formFocusLost(java.awt.event.FocusEvent evt)
+static void main(String args)
```

Figure 4: UML Diagram for the Input for funds requests class. This is also a GUI class which allows the user to either input a new fund request or edit a fund request in the categories shown



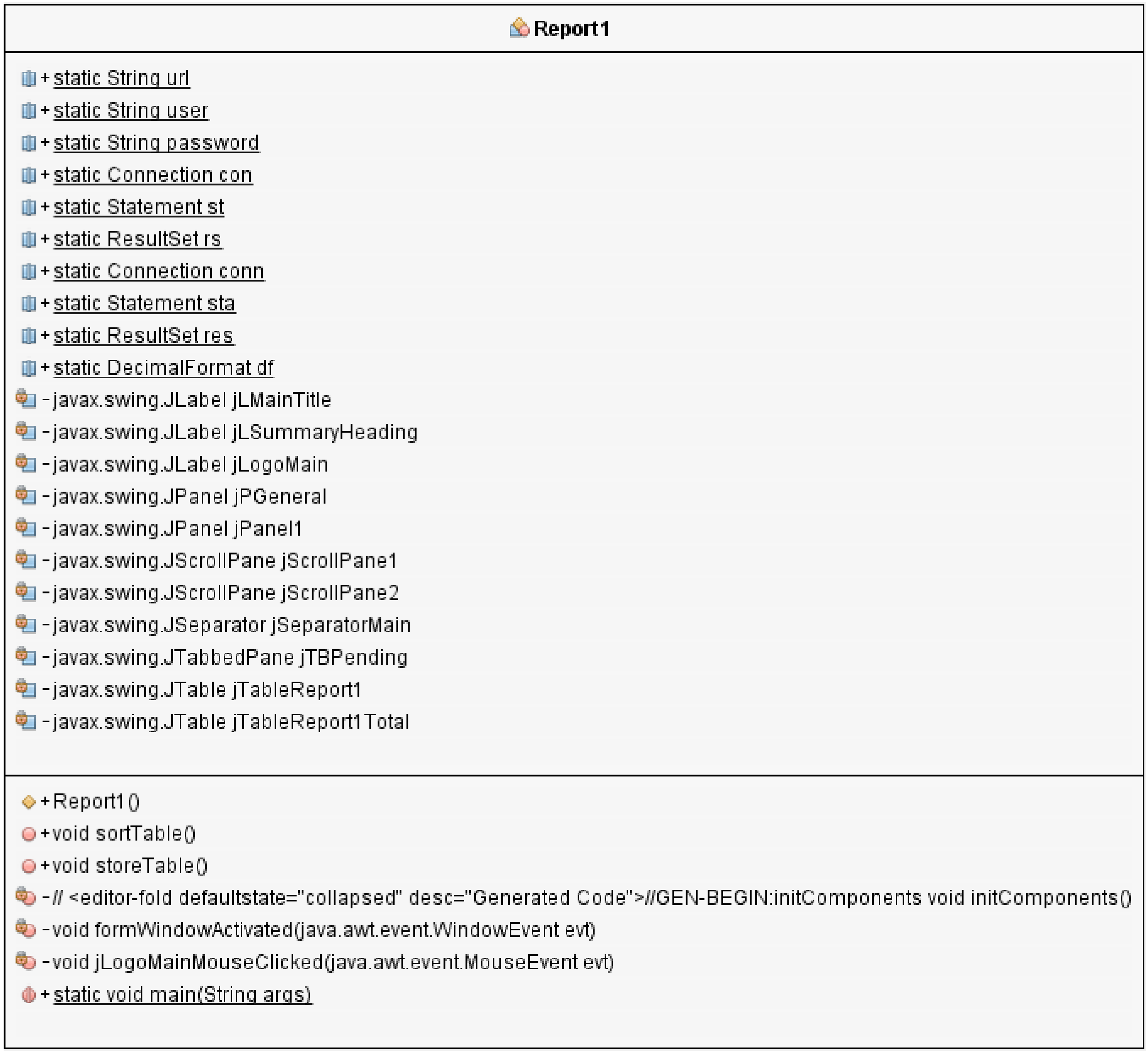
```
ManageFundsRequest
+final Funds[] funds
+static int maxIndex
+static int count
+static int diviCode
+static String dateR
+static String dateA
+static int bnkCode
+static int bnkNum
+static String iBAN
+static int cod
+static String cur
+static double amunt
+static String purp
+static boolean complete
+static String url
+static String user
+static String password
+static javax.swing.JButton jBAddArrangement
-javax.swing.JButton jBAddDisplay
-javax.swing.JButton jBDelDisplay
+static javax.swing.JButton jBDelete
-javax.swing.JButton jBSearchDisplay
+static javax.swing.JComboBox<String> jCBAddDivisionA
-javax.swing.JComboBox<String> jCBDeleteDiv
-javax.swing.JComboBox<String> jCBSearchDiv
-com.toedter.calendar.JDateChooser jDAddDateA
-javax.swing.JLabel jLAddANote
+static javax.swing.JLabel jLAddAmountA
-javax.swing.JLabel jLAddDateA
-javax.swing.JLabel jLAddHead
-javax.swing.JLabel jLArrangementTitle
-javax.swing.JLabel jLDelDiv
-javax.swing.JLabel jLDelHead
-javax.swing.JLabel jLDeleteNote
-javax.swing.JLabel jLErrorMessage
-javax.swing.JLabel jLSearchHead
```



```
-javax.swing.JLabel jLogoArrangement
-javax.swing.JPanel jPAddA
-javax.swing.JPanel jPDelete
-javax.swing.JPanel jPSearch
-javax.swing.JScrollPane jScrollPane10
-javax.swing.JScrollPane jScrollPane7
-javax.swing.JScrollPane jScrollPane8
-javax.swing.JSeparator jSeparatorArrangement
+static javax.swing.JTextField jTAddAmountA
-javax.swing.JTabbedPane jTFundsArr
-javax.swing.JTextField jTSearchNonDate
-javax.swing.JTable jTableAdd
-javax.swing.JTable jTableDel
-javax.swing.JTable jTableSearch

+ManageFundsRequest()
-void sortTable()
-void readFunds()
-// <editor-fold defaultstate="collapsed" desc="Generated Code">//GEN-BEGIN:initComponents void initComponents()
-void jCBAddDivisionAActionPerformed(java.awt.event.ActionEvent evt)
-void jBAddDisplayActionPerformed(java.awt.event.ActionEvent evt)
-void jBAddArrangementActionPerformed(java.awt.event.ActionEvent evt)
-void jTAddAmountAKeyTyped(java.awt.event.KeyEvent evt)
-void jBDelDisplayActionPerformed(java.awt.event.ActionEvent evt)
-void jBDeleteActionPerformed(java.awt.event.ActionEvent evt)
-void jTSearchNonDateActionPerformed(java.awt.event.ActionEvent evt)
-void jTSearchNonDateKeyReleased(java.awt.event.KeyEvent evt)
-void jBSearchDisplayActionPerformed(java.awt.event.ActionEvent evt)
-void formWindowActivated(java.awt.event.WindowEvent evt)
-void jPAddAFocusLost(java.awt.event.FocusEvent evt)
-void jPAddAFocusGained(java.awt.event.FocusEvent evt)
-void jPDeleteFocusGained(java.awt.event.FocusEvent evt)
-void jPSearchFocusGained(java.awt.event.FocusEvent evt)
-void jLogoArrangementMouseClicked(java.awt.event.MouseEvent evt)
+static void main(String args)
```

Figure 5: UML Diagram for the manage funds request class. This is a GUI which allows the user to either add an arrangement, search for a fund request, or delete a fund request

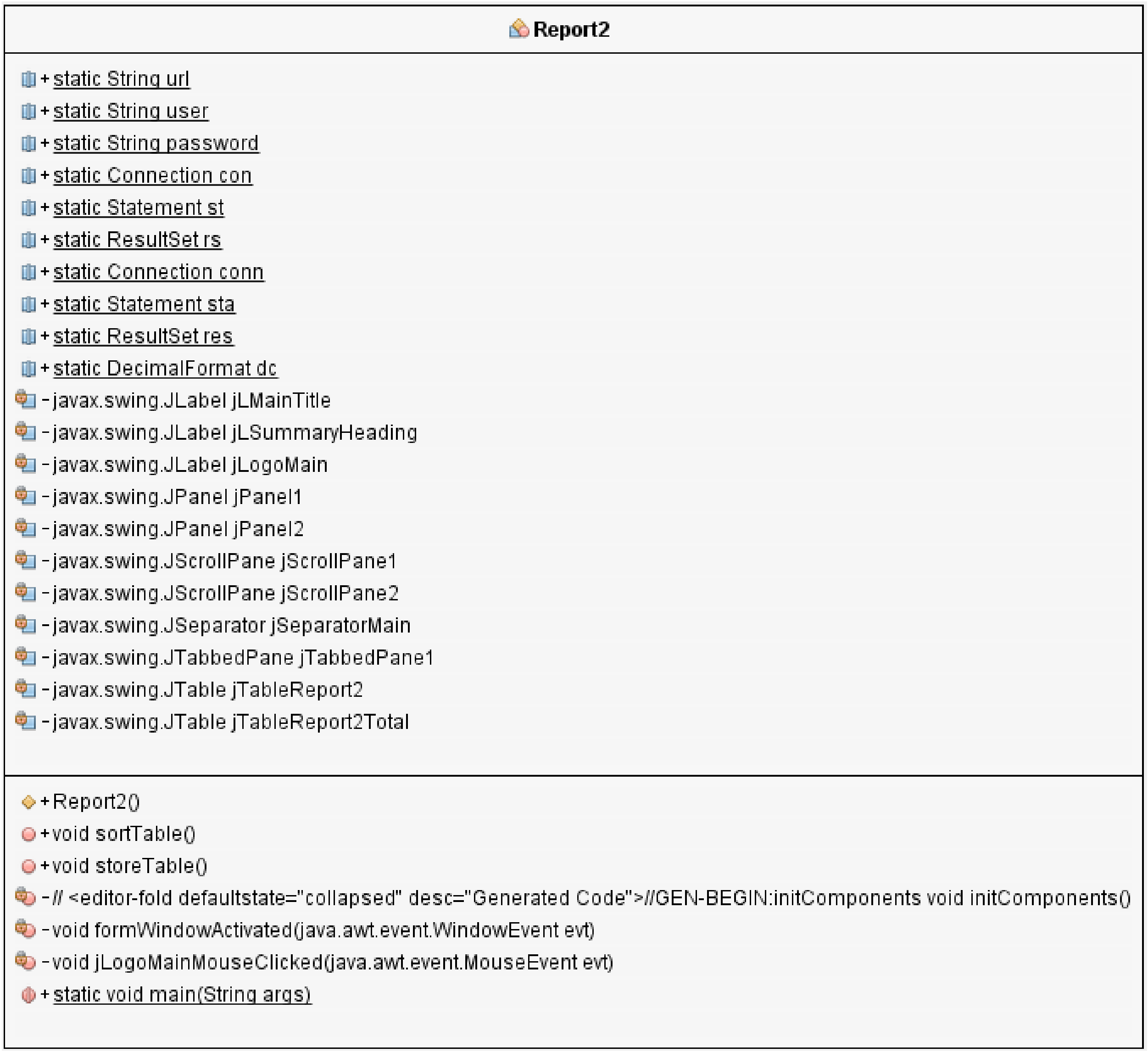


```
Report1

+static String url
+static String user
+static String password
+static Connection con
+static Statement st
+static ResultSet rs
+static Connection conn
+static Statement sta
+static ResultSet res
+static DecimalFormat df
-javax.swing.JLabel jLMainTitle
-javax.swing.JLabel jLSummaryHeading
-javax.swing.JLabel jLogoMain
-javax.swing.JPanel jPGeneral
-javax.swing.JPanel jPanel1
-javax.swing.JScrollPane jScrollPane1
-javax.swing.JScrollPane jScrollPane2
-javax.swing.JSeparator jSeparatorMain
-javax.swing.JTabbedPane jTBPending
-javax.swing.JTable jTableReport1
-javax.swing.JTable jTableReport1Total

+Report1()
+void sortTable()
+void storeTable()
-// <editor-fold defaultstate="collapsed" desc="Generated Code">//GEN-BEGIN:initComponents void initComponents()
-void formWindowActivated(java.awt.event.WindowEvent evt)
-void jLogoMainMouseClicked(java.awt.event.MouseEvent evt)
+static void main(String args)
```

Figure 6: UML Diagram for the reports of funds pending. Here, the user can view a summary of the amount pending to pay for the incomplete fund requests, and they can also see a detailed report for each individual fund request



```
Report2
+static String url
+static String user
+static String password
+static Connection con
+static Statement st
+static ResultSet rs
+static Connection conn
+static Statement sta
+static ResultSet res
+static DecimalFormat dc
-javax.swing.JLabel jLMainTitle
-javax.swing.JLabel jLSummaryHeading
-javax.swing.JLabel jLogoMain
-javax.swing.JPanel jPanel1
-javax.swing.JPanel jPanel2
-javax.swing.JScrollPane jScrollPane1
-javax.swing.JScrollPane jScrollPane2
-javax.swing.JSeparator jSeparatorMain
-javax.swing.JTabbedPane jTabbedPane1
-javax.swing.JTable jTableReport2
-javax.swing.JTable jTableReport2Total

+Report2()
+void sortTable()
+void storeTable()
-// <editor-fold defaultstate="collapsed" desc="Generated Code">//GEN-BEGIN:initComponents void initComponents()
-void formWindowActivated(java.awt.event.WindowEvent evt)
-void jLogoMainMouseClicked(java.awt.event.MouseEvent evt)
+static void main(String args)
```

Figure 7: UML Diagram for the report of funds paid for. The format is similar to the previous report

Database Tables and Columns

The program's database will consist of 3 tables

1. Divisions – will store all the divisions and their code of reference
2. Banks – will store all the banks and their code of reference
3. Funds Details – will store all the details which are saved in a FundsRequest object
4. Costs Made – will store all the payments made respective to their fund request

When data will be entered, and the user wishes to save, all changes will be saved to the funds details table. It could either add a new record, or edit an existing record

The tables and their columns are outlined in Figure 2

Table	Column	Type	Nullable	Default Value	Extra	Sample Data
Divisions	divisionCode	int(11)	NO	NULL		1
Divisions	divisionName	text	NO	NULL		BMW
Banks	bankCode	int(11)	NO	NULL		2
Banks	bankName	text	NO	NULL		ANB
Funds Details	fundCode	int(11)	NO	NULL	auto_increment	1
Funds Details	divCode	int(11)	NO	NULL		1
Funds Details	dateRequested	text	NO	NULL		21/08/2000
Funds Details	dateArranged	text	YES	NULL		06/12/2010
Funds Details	bankCode	int(11)	NO	NULL		1
Funds Details	bankNum	char(6)	NO	NULL		189277

Funds Details	Iban	char(22)	NO	NULL		A0NM12NBJKIFVB234MKHU
Funds Details	usd	Double	NO	NULL		5000
Funds Details	usdchange	Double	YES	NULL		5000
Funds Details	eur	Double	YES	NULL		5000
Funds Details	eurchange	Double	YES	NULL		1000
Funds Details	sar	Double	YES	NULL		5000
Funds Details	sarchange	Double	YES	NULL		5000
Funds Details	Purpose	Text	YES	NULL		To aid with shipments
Funds Details	Complete	Tinyint(1)	NO	NULL		1
Costs Made	fundCodeP	Int(11)	NO	NULL		1
Costs Made	divCodeP	Int(11)	NO	NULL		1
Costs Made	dateRequested P	Text	NO	NULL		21/08/2017
Costs Made	dateArrangedP	Text	YES	NULL		12/01/2018
Costs Made	USDP	Double	YES	NULL		5000
Costs Made	EURP	Double	YES	NULL		5000

Costs Made	SARP	Double	YES	NULL		5000
Costs Made	completeP	Tinyint(1)	NO	NULL		0

Figure 8: Tables and programs created by the program using SQL, in order to facilitate the storage of data in a database

Process Flowcharts

The following flowcharts represent the processes that will be carried out by the proposed program

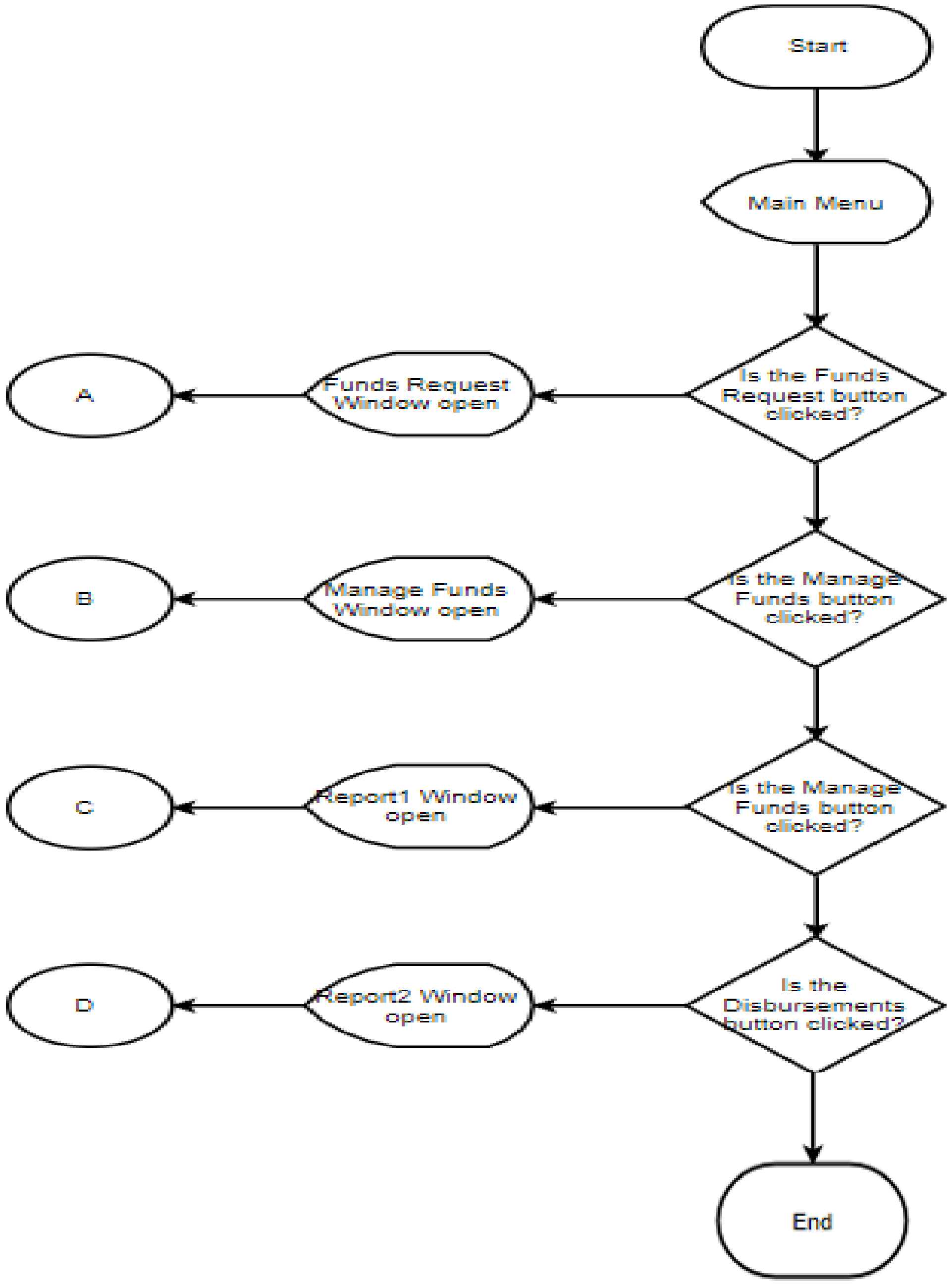

Start
Main Menu
Is the Funds Request button clicked?
Funds Request Window open
A
Is the Manage Funds button clicked?
Manage Funds Window open
B
Is the Manage Funds button clicked?
Report1 Window open
C
Is the Disbursements button clicked?
Report2 Window open
D
End

Figure 9: Main Menu process. This flowchart shows the first thing that happens when the program has started. The user gets to see a summary of all the fund requests and has an option to either input for a fund request or an arrangement

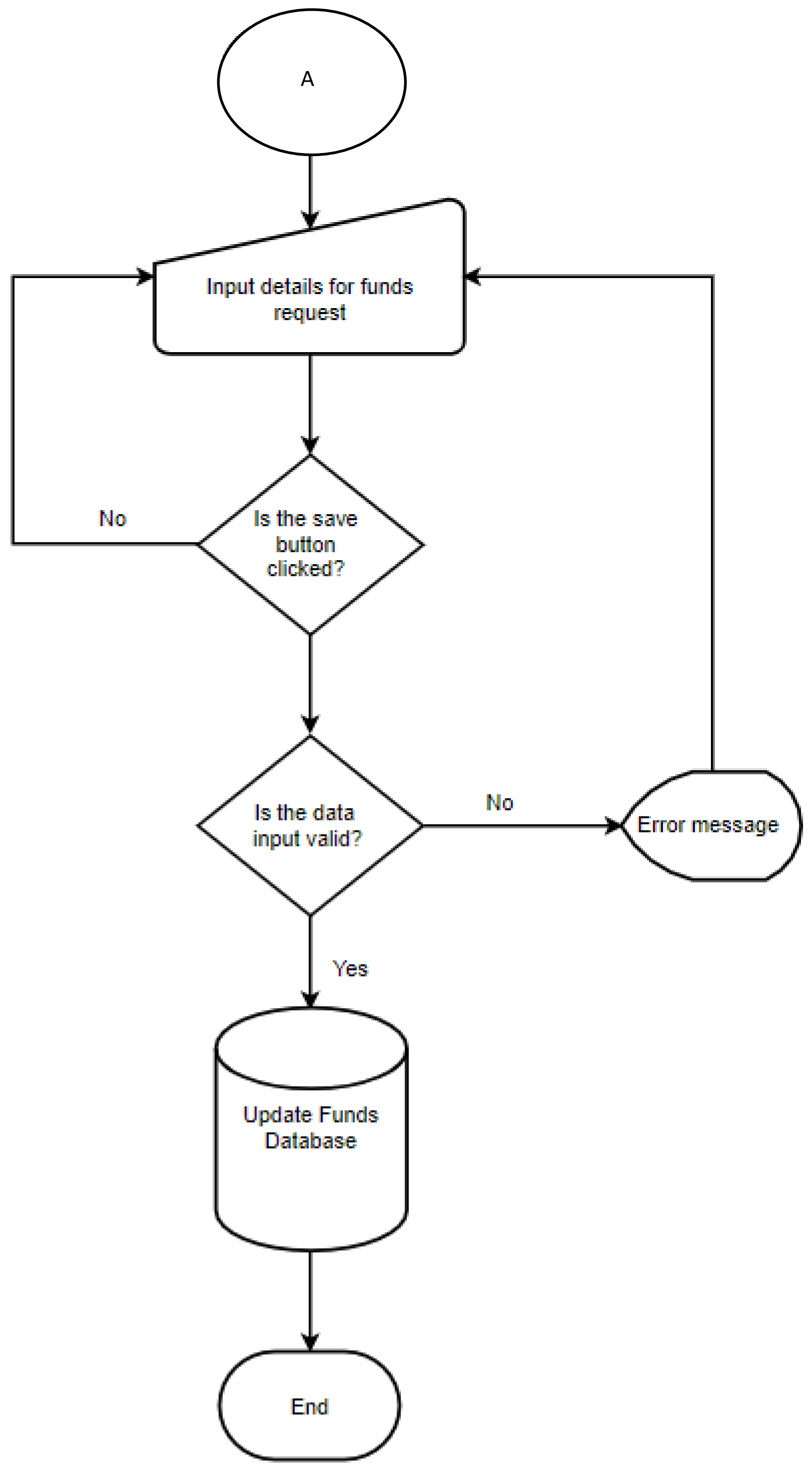


Figure 10: Input for fund request process. The user all the necessary details required to create a valid fund request. This request is then validated and eventually, it is stored in the funds database

The following will be showing one flowchart for each of the features in the InputForFundsArrangement class.

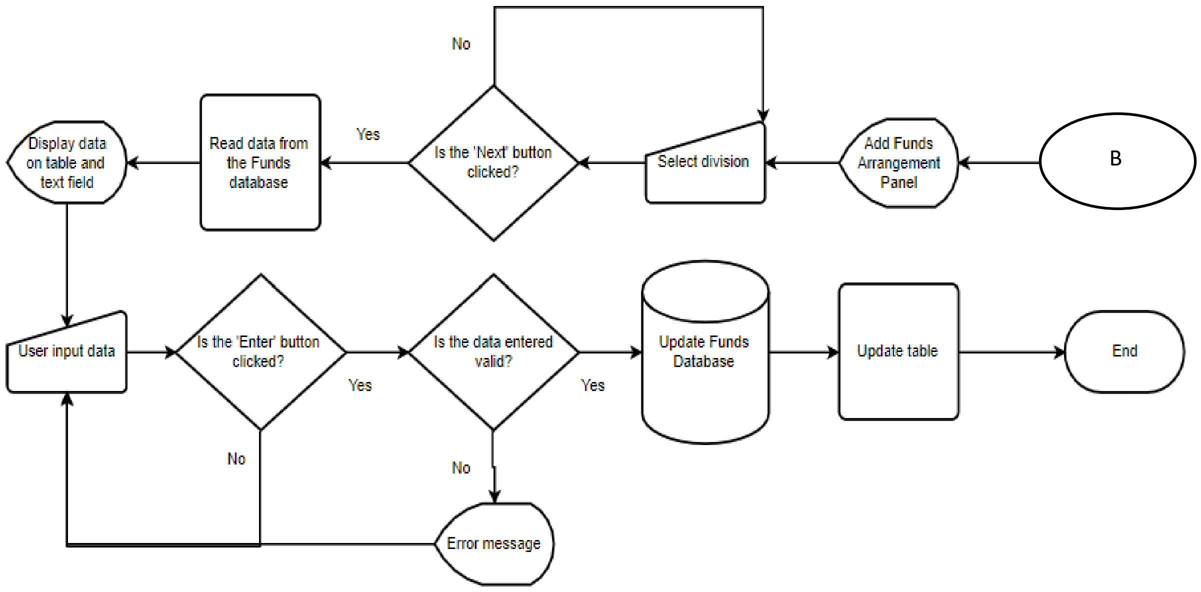

B
Add Funds Arrangement Panel
Select division
Is the 'Next' button clicked?
No
Yes
Read data from the Funds database
Display data on table and text field
User input data
Is the 'Enter' button clicked?
Yes
No
Is the data entered valid?
Yes
No
Error message
Update Funds Database
Update table
End

Figure 11: Input for funds arrangement process. The user decides which division he wished to add the arrangement for. They must then input an arrangement date and the amount they wish to install into the request. The data in the table and the database is then updated

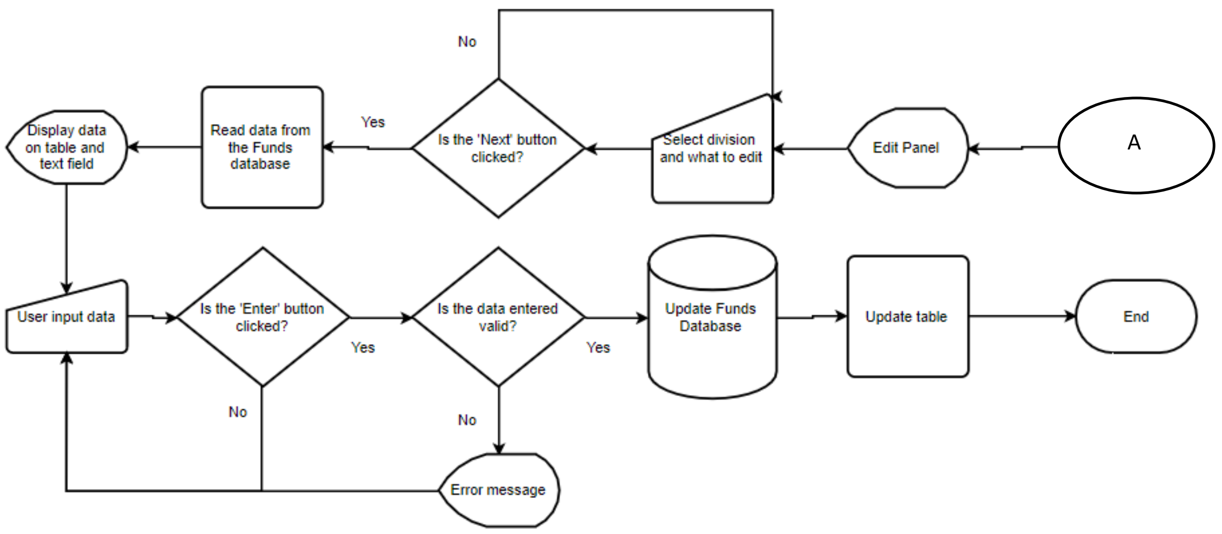

A
Edit Panel
Select division and what to edit
Is the 'Next' button clicked?
No
Yes
Read data from the Funds database
Display data on table and text field
User input data
Is the 'Enter' button clicked?
Yes
No
Is the data entered valid?
Yes
No
Error message
Update Funds Database
Update table
End

Figure 12: Editing process. The user decides which division's request they will edit and what they wish to edit. They then select the row they wish to edit and then enter the necessary details they wish to edit. The table and the database are then edited

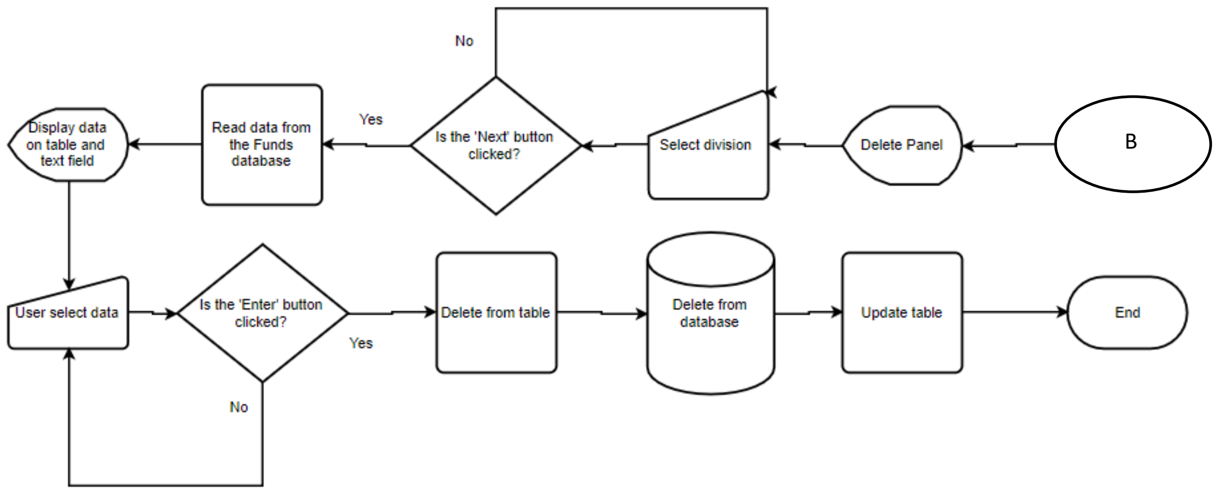

B
Delete Panel
Select division
Is the 'Next' button clicked?
No
Yes
Read data from the Funds database
Display data on table and text field
User select data
Is the 'Enter' button clicked?
Yes
No
Delete from table
Delete from database
Update table
End

Figure 13: Deleting process. The user selects which division's fund request they wish to delete. They then select the row which they wish deleted or type in the code, and then the row is removed from the table and the database

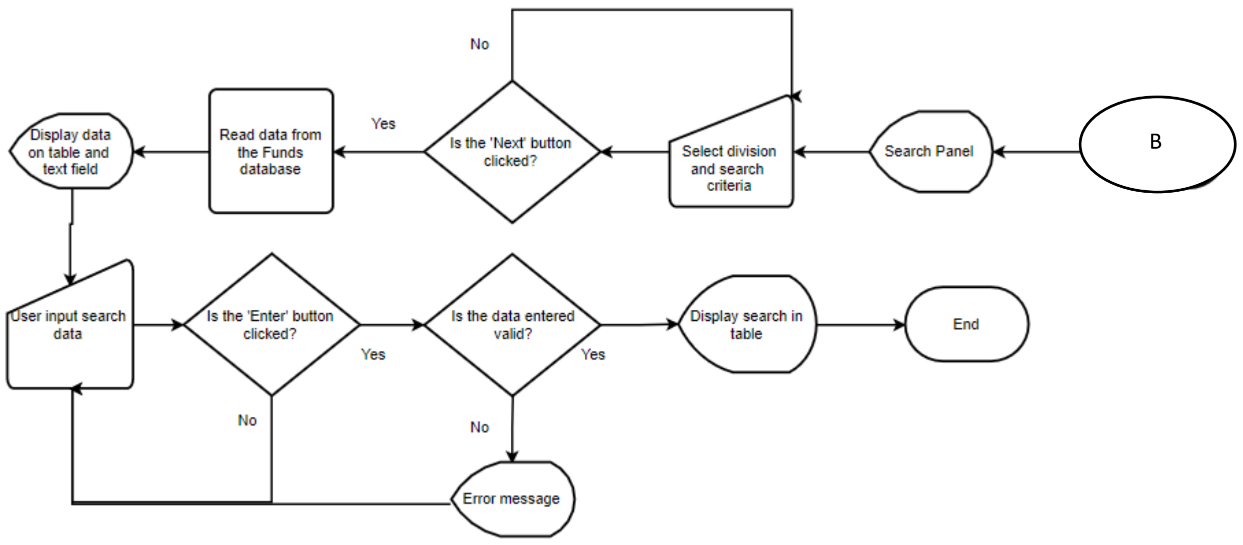

B
Search Panel
Select division and search criteria
Is the 'Next' button clicked?
No
Yes
Read data from the Funds database
Display data on table and text field
User input search data
Is the 'Enter' button clicked?
Yes
No
Is the data entered valid?
Yes
No
Display search in table
End
Error message

Figure 14: searching process. The user selects which division's request they wish to search and which category they want to choose by which they wish to search. They then enter the word/date/number which they use to search, and the row will then appear on the table. Database is not affected

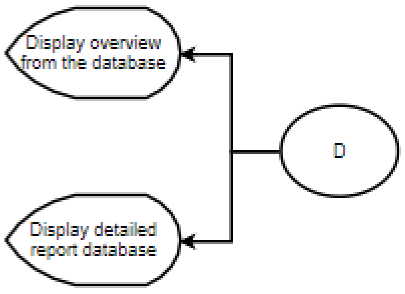

D
Display overview from the database
Display detailed report database

Figure 16: Payments Report

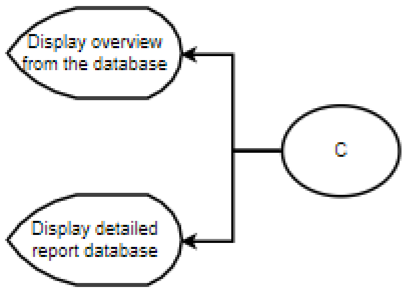

C
Display overview from the database
Display detailed report database

Figure 15: Pending Requests Report

Panel Design and Top-Down Design

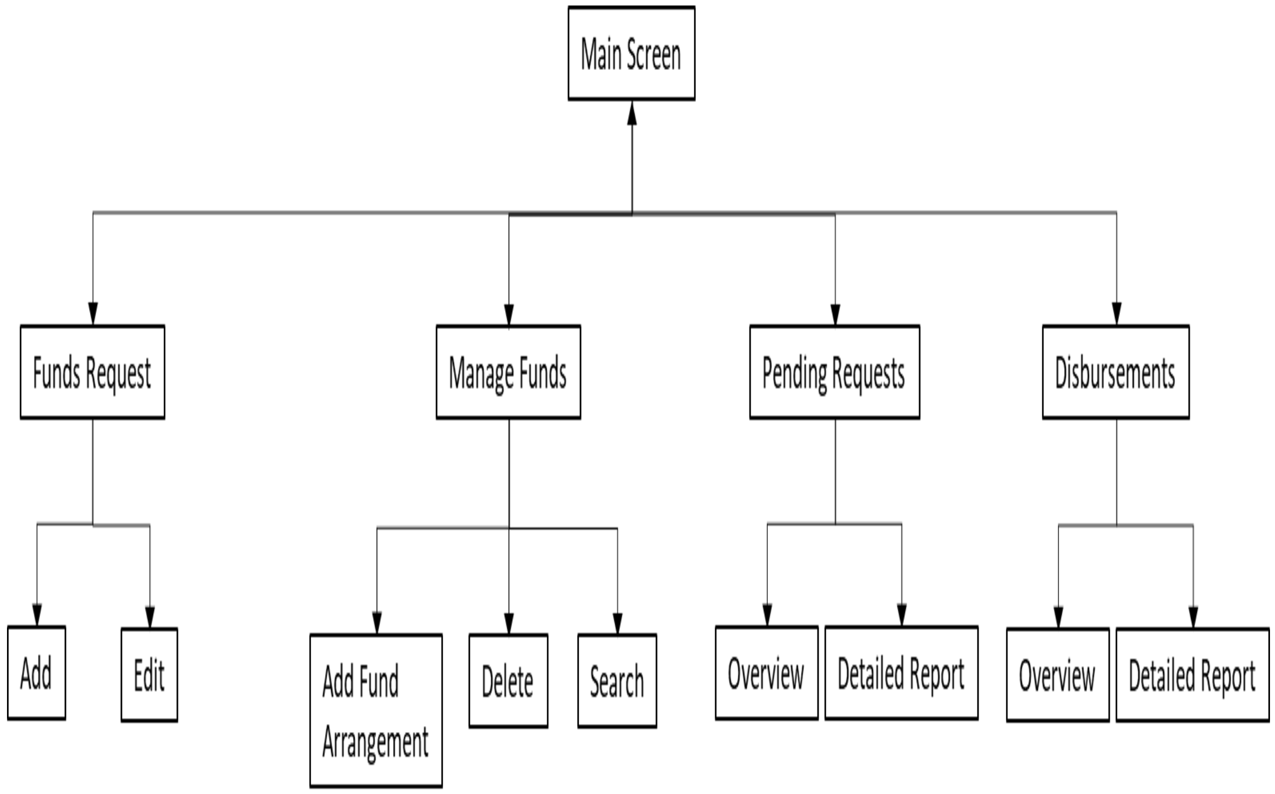


Figure 17: GUI Top-Down Design. The flowchart above illustrates the navigation path between all the panels that are to be used in the program

The following are prototypes for the program's 10 panels. Each design was developed through consultation with the client and to best suit their needs. Initially, my end-user had a completely separate idea of a prototype to what he thought of later on, therefore, this is the most recent and approved prototype. It will clearly outline the purpose and function of the panels.

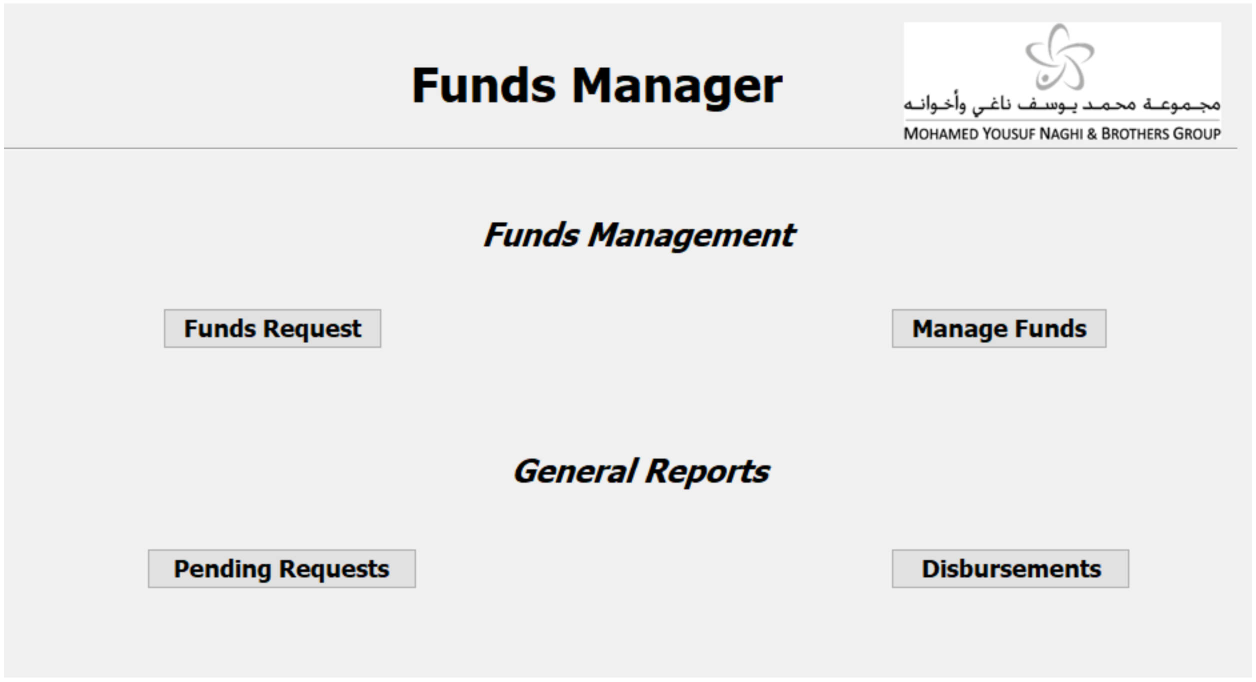
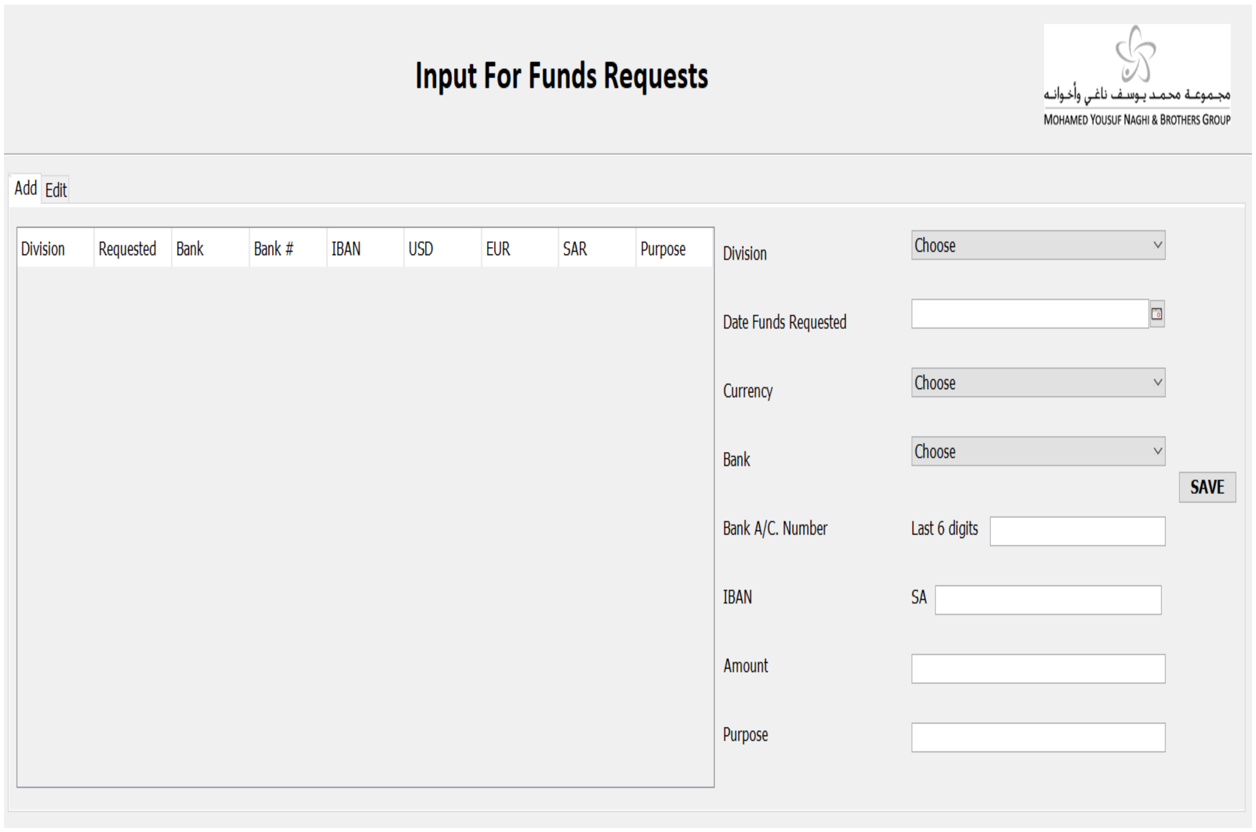


Figure 18: Main Menu. This is the first thing that the end-user will see when opening the program.



Input For Funds Requests

مجموعة محمد يوسف ناغي وأخوانه
MOHAMED YOUSUF NAGHI & BROTHERS GROUP

Add | Edit

Division	Requested	Bank	Bank #	IBAN	USD	EUR	SAR	Purpose

Division	Choose	
Date Funds Requested		
Currency	Choose	
Bank	Choose	SAVE
Bank A/C. Number	Last 6 digits	
IBAN	SA	
Amount		
Purpose		

Figure 19: The first panel in the Input for Fund Requests Frame. The user will be given a set of fields to fill in, and next to it is a table which shows the user all the fund requests for which complete payments are still pending

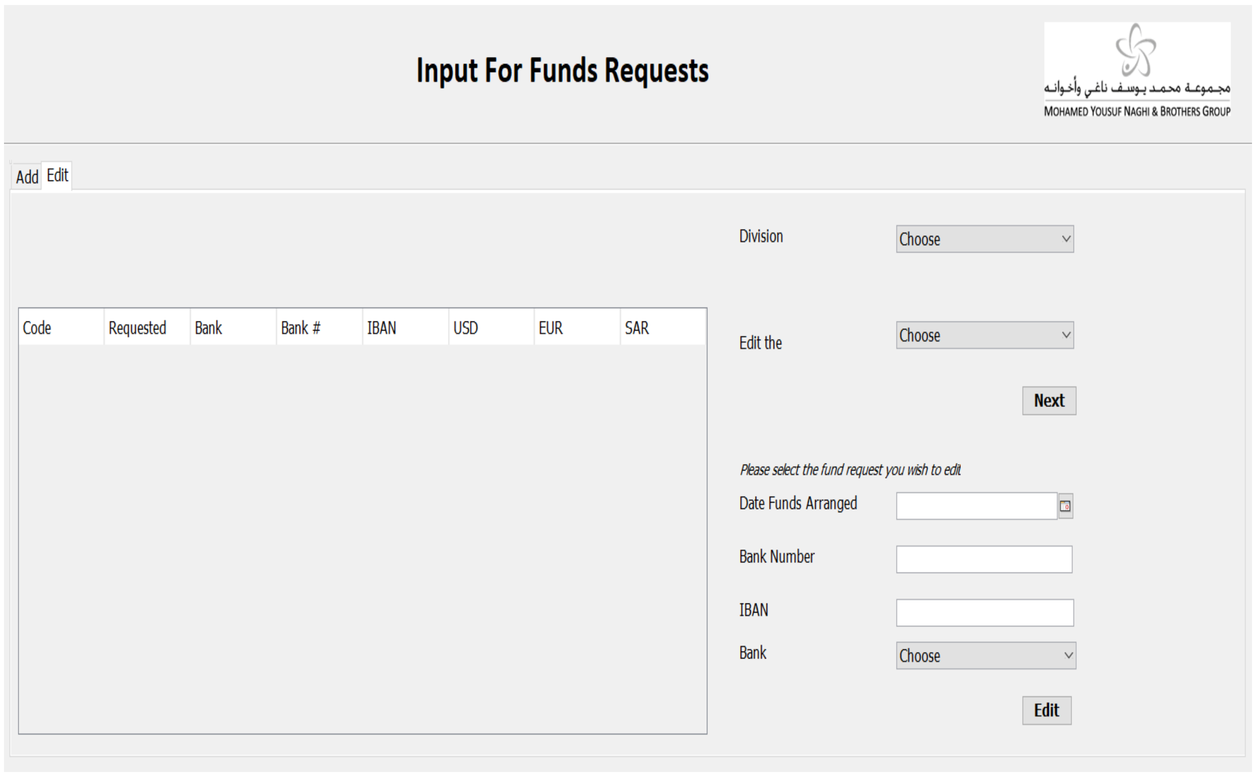
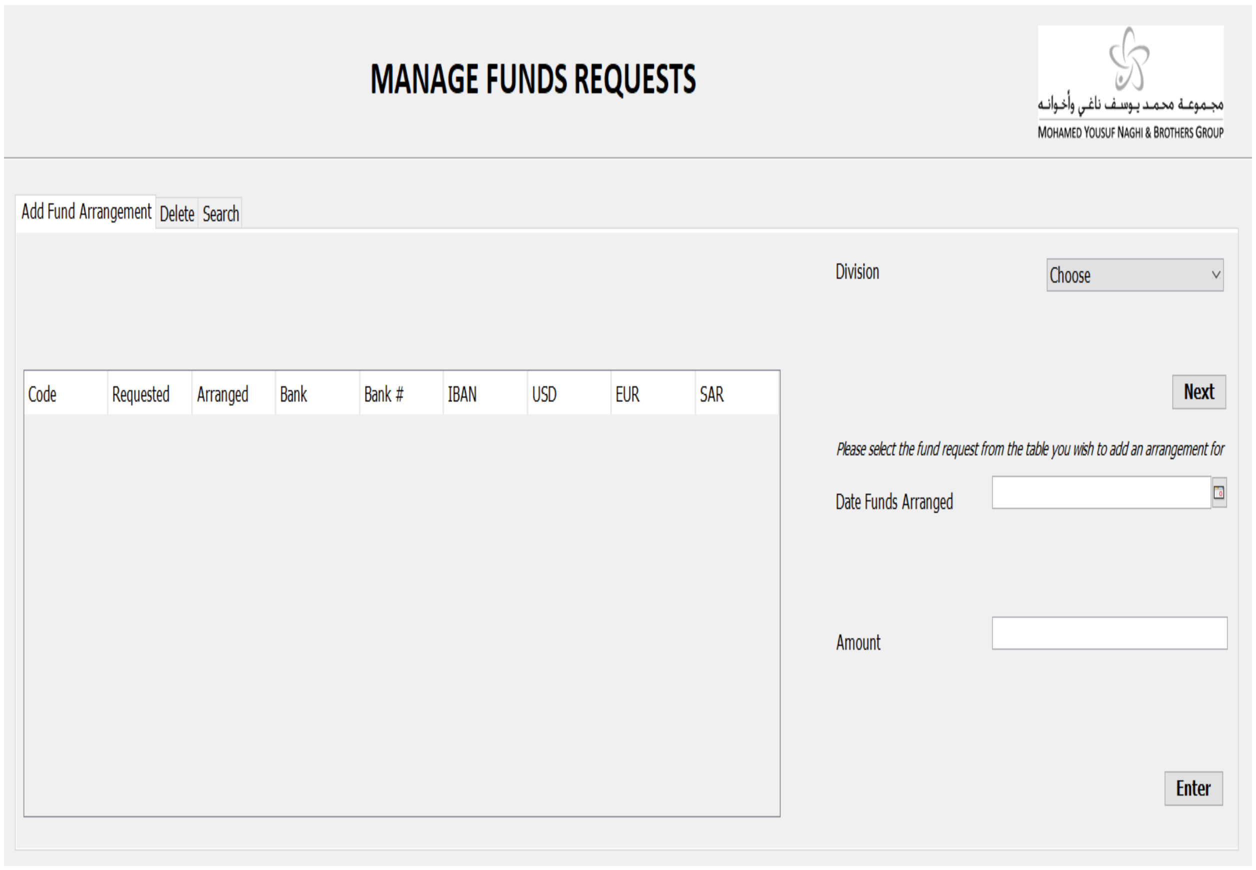


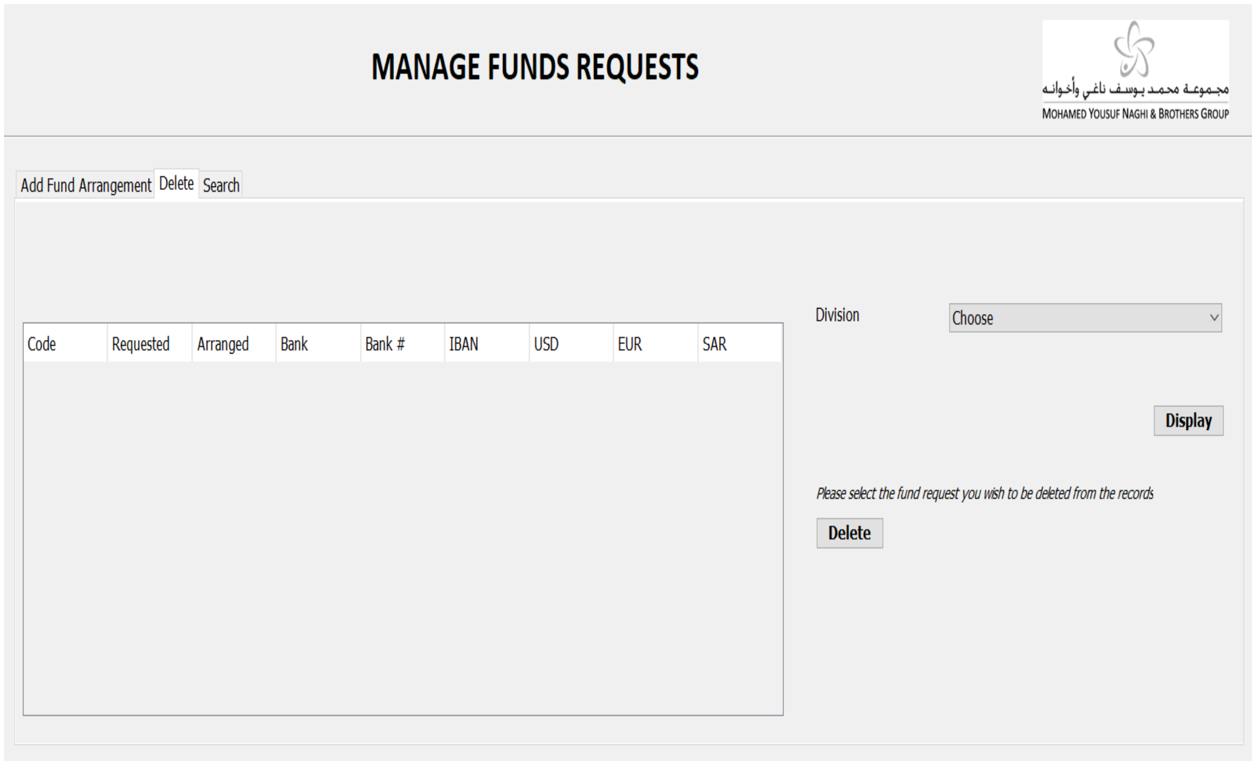
Figure 20: Edit panel. Depending on which category you choose, the corresponding field will be the only one visible to the user



Code	Requested	Arranged	Bank	Bank #	IBAN	USD	EUR	SAR

Please select the fund request from the table you wish to add an arrangement for

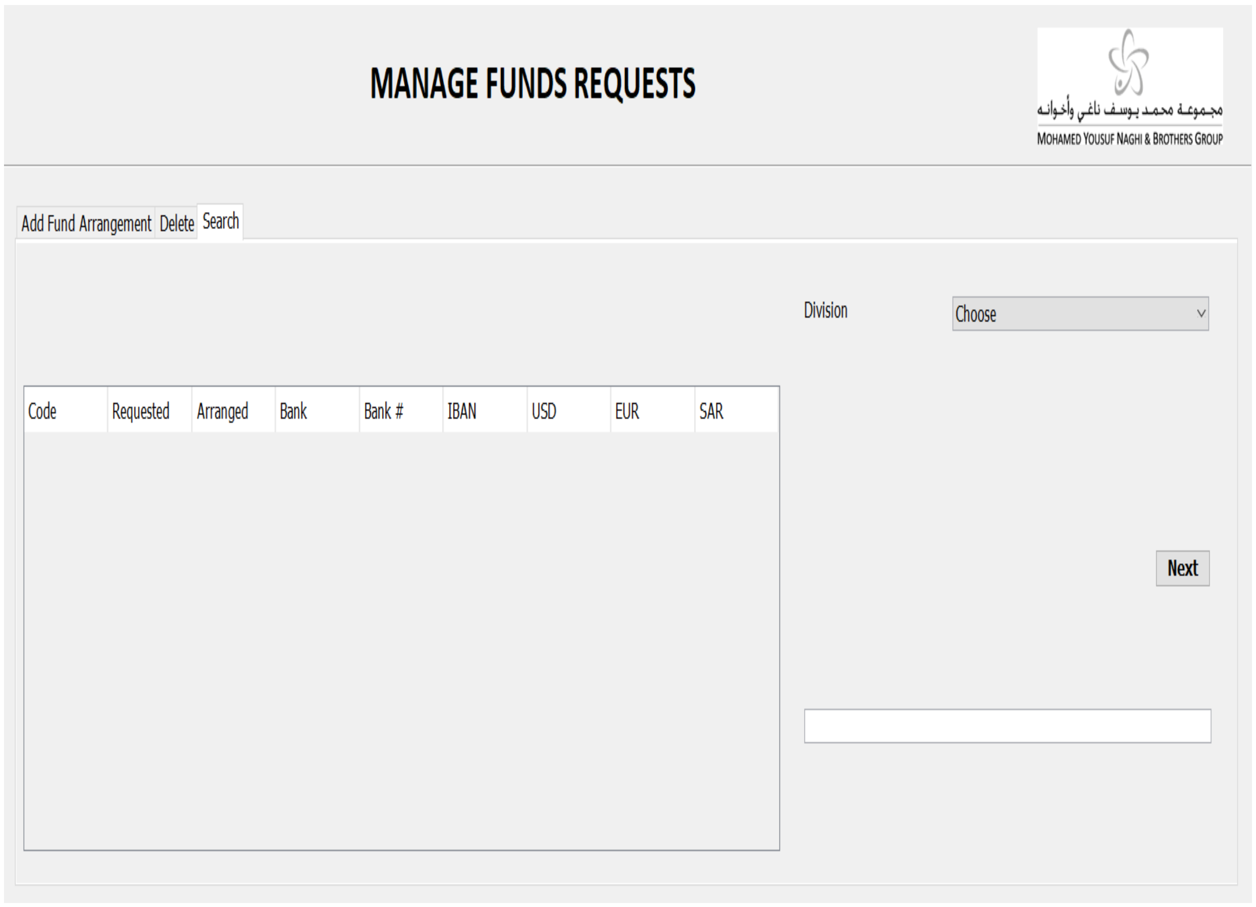
Figure 21: Add fund arrangement panel. In this panel, the user can choose to pay the entire amount for any fund request, or the user can choose to pay in instalments as well



Code	Requested	Arranged	Bank	Bank #	IBAN	USD	EUR	SAR

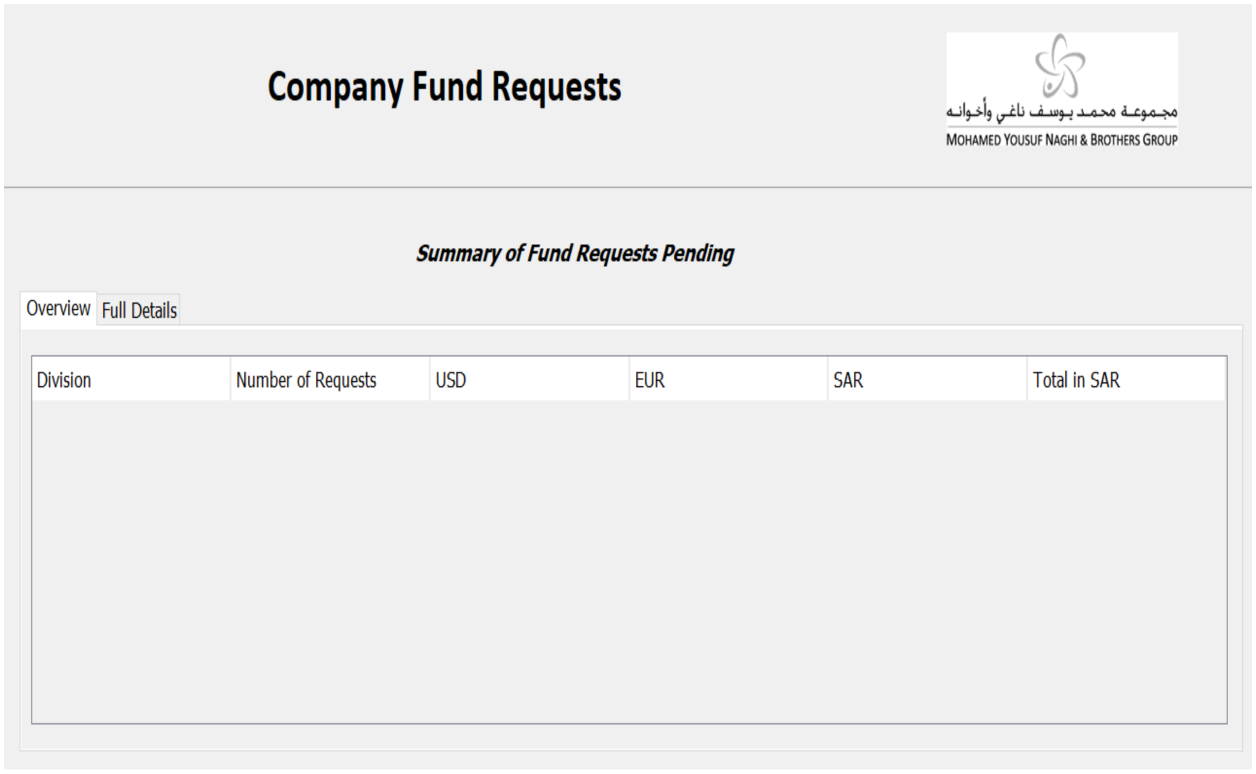
Please select the fund request you wish to be deleted from the records

Figure 22: Delete panel. This is the simplest panel to use as all the user has to do is select a fund request from the table and then select the delete button. The user will be asked to confirm their choice, and once that is completed, the fund request will have successfully deleted



Code	Requested	Arranged	Bank	Bank #	IBAN	USD	EUR	SAR

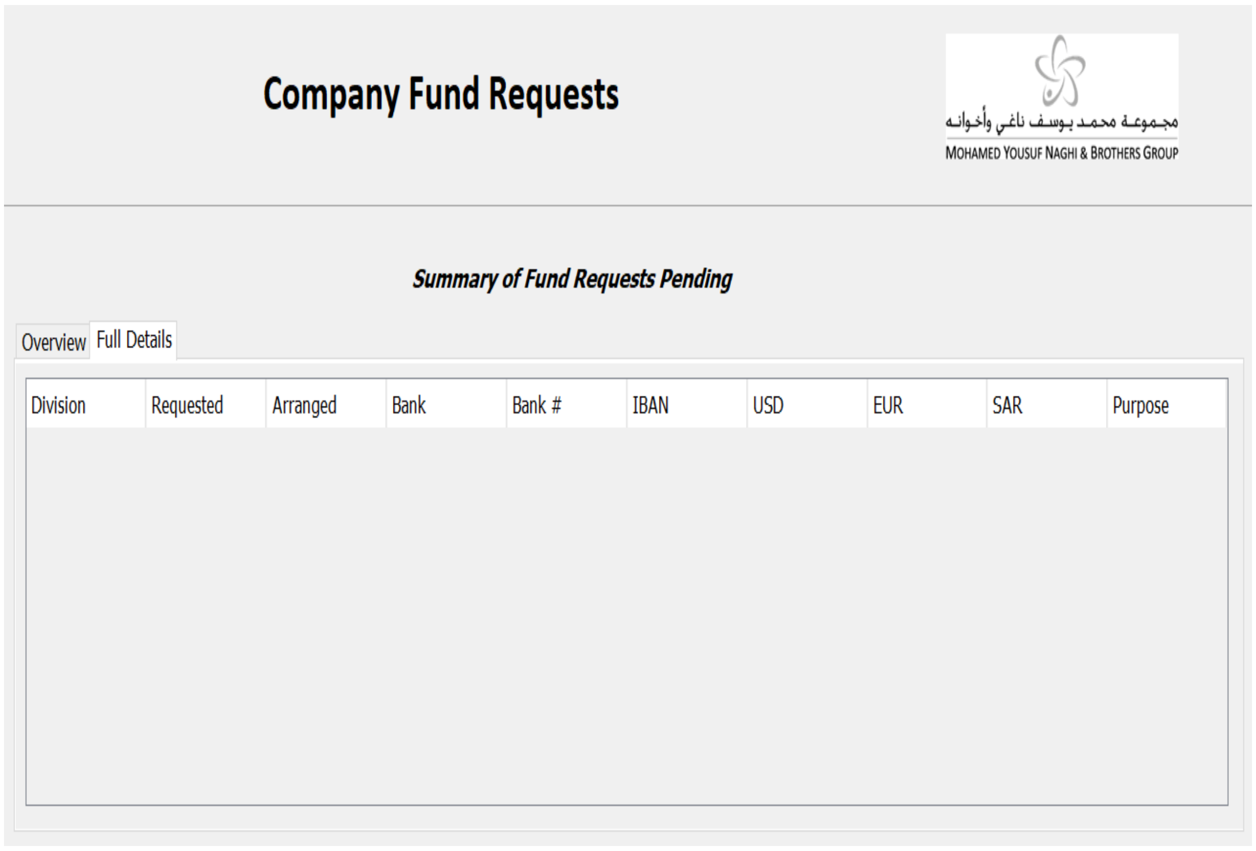
Figure 23: Search panel. The user simply needs to type in what he wishes to find, regardless of what field it is, and the table will then shortlist the results based on the division chosen


Company Fund Requests

Summary of Fund Requests Pending

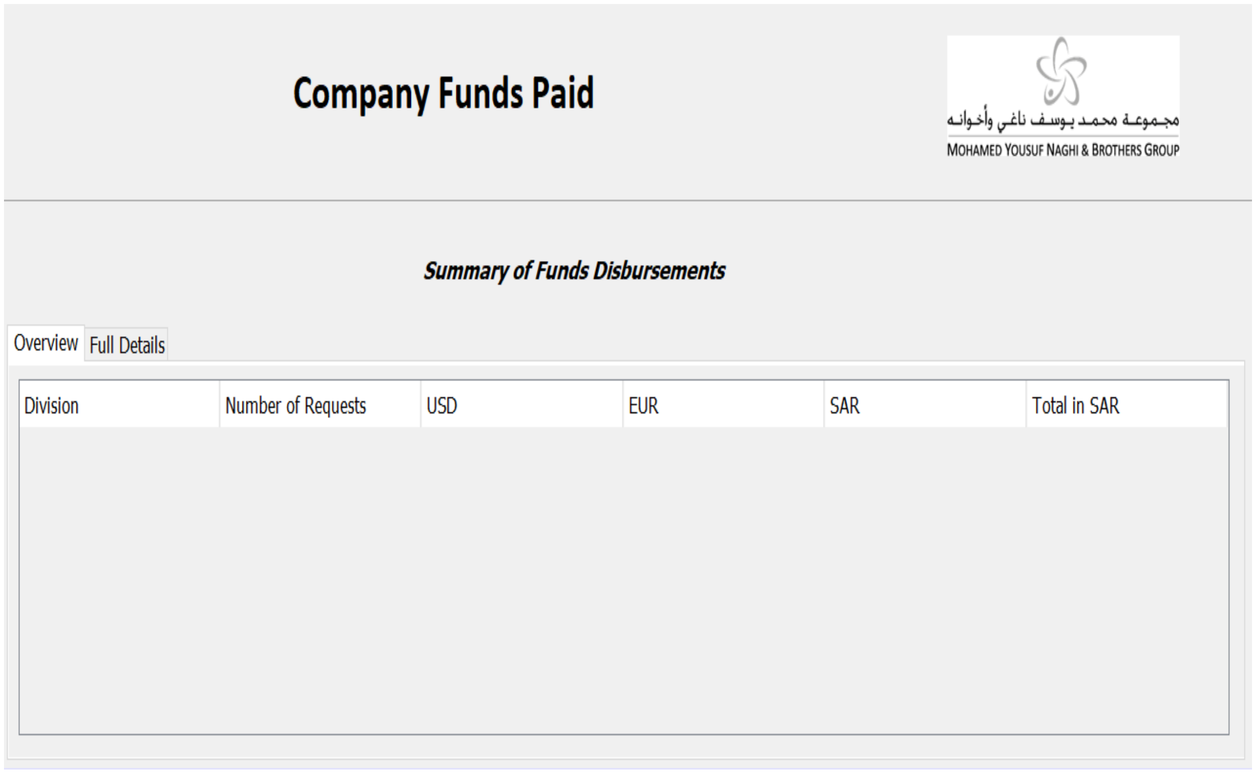
Division	Number of Requests	USD	EUR	SAR	Total in SAR

Figure 24: Requests Pending overview. This summarises the total amount of requests pending for each division. The most accurate currency conversion rate is used in order to find the total in SAR


Company Fund Requests
مجموعة محمد يوسف ناغي وأخوانه
MOHAMED YOUSUF NAGHI & BROTHERS GROUP
Summary of Fund Requests Pending
Overview
Full Details

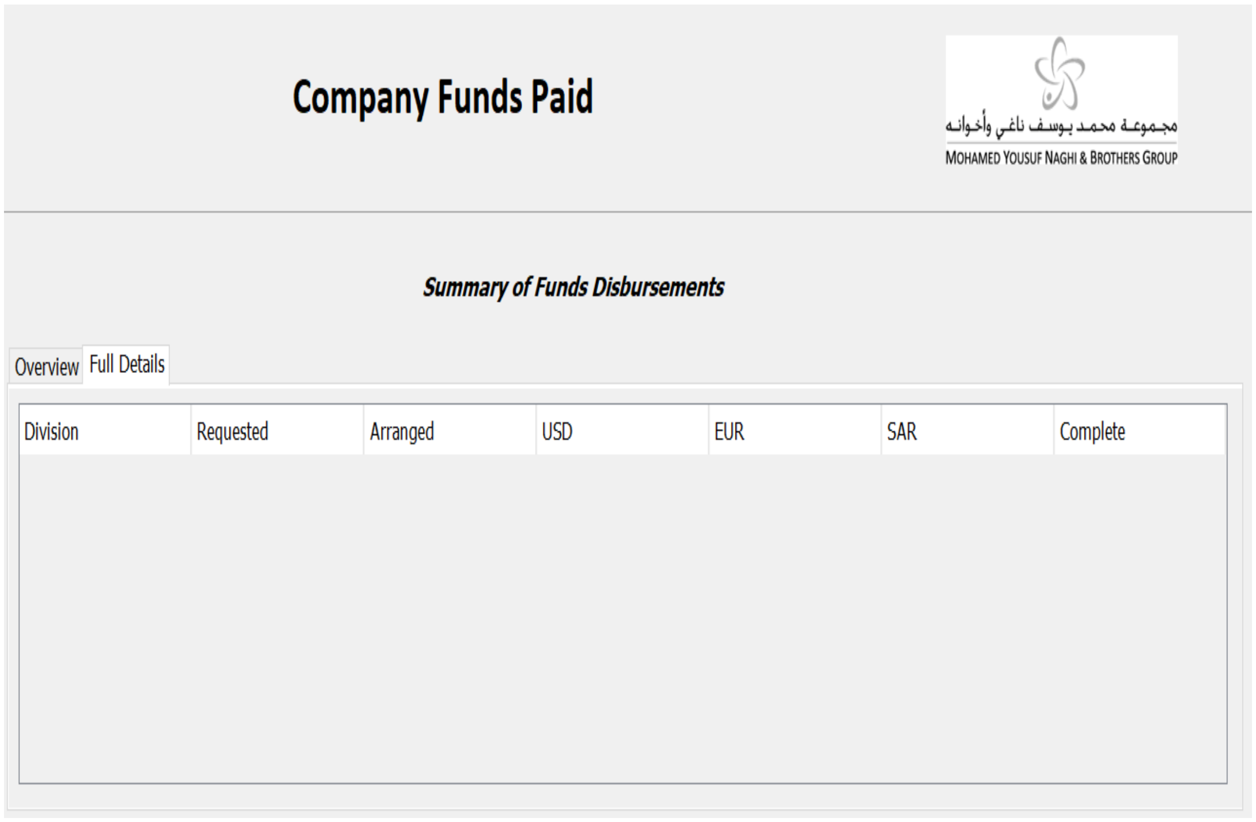
Division	Requested	Arranged	Bank	Bank #	IBAN	USD	EUR	SAR	Purpose

Figure 25: Requests Pending detailed report. This shows all the details for the fund requests for which the payments have not yet been completely paid for


Company Funds Paid
مجموعة محمد يوسف ناغي وأخوانه
MOHAMED YOUSUF NAGHI & BROTHERS GROUP
Summary of Funds Disbursements
Overview
Full Details

Division	Number of Requests	USD	EUR	SAR	Total in SAR

Figure 26: Company payments overview. This is very similar to the requests pending report however in this case, this calculates the amount the business has paid for each division


Company Funds Paid

Summary of Funds Disbursements

مجموعة محمد يوسف ناغي وأخوانه
MOHAMED YOUSUF NAGHI & BROTHERS GROUP
Overview
Full Details

Division	Requested	Arranged	USD	EUR	SAR	Complete

Figure 27: Company payments detailed report. This is also similar to the requests pending however here, the report also shows the fund requests that have been completely paid for

Test Plan

Action to be tested	***Test method***
The program interface should be user friendly	Carry out an interview with my end-user (Appendix 2) and also with my advisor, taking their feedback on how they felt when using the program. If they liked it, then it would mean that the success criteria had been met
The user should only be allowed to add fund request for 3 divisions	When opening either the fund requests, or manage funds frames, you will notice a division field, when selecting the drop-down menu, there should only be 3 divisions for the user to select from
The user should be able to access reports on fund requests	In the main menu, the values shown will have information which will show details on fund requests from each of the 3 divisions and what their pending amount is.
There should be successful and accurate currency conversions	I will make manual calculations to see whether the amounts which had been added to the report were accurate as per the actual currency conversion, taking into account both USD, and EUR
The user should easily be able to edit fund requests	I will enter the necessary details to edit a record. If it appears on the database, then it is successful. I will also input intentional false data to see whether or not the program gives a message to the user saying they need to re-enter their data
The user should be notified if they are taking too long to pay the funds (1 week)	This is tested by entering a false date in the manage funds panel. This date will be before the date requested of that fund. If this works, then the program should send a message to the user,

	telling him that the date entered is invalid and that they need to re-enter an appropriate date
The company's logo should be present at the top corner of each panel	Upon opening each panel of my program, I should clearly see the company logo at the top corner of each of these panels
All sections should be clearly separated for the user to easily navigate	In the input for funds arrangement panel, each module will be separated using tabs. Each tab will be clearly defined and when clicking on the tab, I will make sure that it leads to the correct module to which it was labelled to go to
All data shown in the program should update whenever a change has been made	I will make any valid change when using the entire program altogether, and I will then note down whatever changes have been made. Once I am done, I will then check the database to see whether the correct changes which I had noted down had been made. The report section and the fund requests frame will also be checked to see whether the changes to the funds have been updated in the tables shown. I will also take the help of my end-user to see whether or not he gets what he needs