Odin Health

Business Manager

Technical Documentation

Systems Maintenance Documentation

Identification of the Problem

Currently, Odin Health uses a huge number of spreadsheets for tracking its business expenses, tracking expenses, wages, etc. This creates a very inefficient system in which a lot of time is wasted formatting and creating the spreadsheets as well as locating and collating them.

The current spreadsheet solution is very unorganised and requires the manual input of data across forms. Applications for reimbursements and such must also be manually sent via email or approved through phone calls. For a business, time is money, and it is vitally important that a system is as efficient as possible.

Thus as shown, the current system is inefficient and unorganised and a new system is required to keep all the data together and allow for automatic calculation and tasks.

Identification of Prospective Users

Almost all employees in Odin Health would be using this system, as most employees will at one point require reimbursement for travel or business expenses. The employees will also most probably apply for travel or business expenses. Therefore, all the employees would need to be able to access this system.

The jobs range from Accountants to Developers, and thus all the different users have varying levels of computer experience. Due to this, the system will have to be user-friendly and very precise so that little or no training is required for the employees on top of the information provided in the User Manual.

Requirement Specifications

The following specifications are based upon the investigation and analysis of the previous system and multiple interviews with the client. The specifications describe all the information, logic and requirements of the new system.

The new system must be able to fulfil the needs and requirements of the clients. It is important that the client and I agree on the same requirements otherwise the new system would be inefficient and perhaps a waste of time. The requirements specification is based upon the decision to use Visual Basic .NET and Microsoft Access 2013.

General Requirements

1. The system must be efficient and quick to use.
2. Data should be stored and organised quickly and practically.
3. The user interface must be unambiguous and easy to navigate and use.
4. Reports must be able to be printed.
5. Back-ups of the database must be easy to perform.

Hardware and Software Requirements

The new system must be able to be used on any computer running Windows 7, 8 or 8.1. A computer with a mouse, keyboard and printer would be necessary to utilise the system to its full potential.

The main database must be stored on the network so all employees can access it. Furthermore, an option to store the database locally (i.e. save it to a different directory), so that data may be accessed when the network cannot be accessed, is essential.

The new system must also be able to print reports to any printer that the client wishes. The required information must be able to be easily viewed and sorted for this printing.

The new system will be built on Microsoft Visual Basic .NET while using Microsoft Office Access 2013 to manage the database. This would require all Users computers to have the .NET Framework installed and relevant Microsoft Access Drivers or Microsoft Access 2013 to enable OLE DB (Object Linking and Embedding, Database) database connections.

Database Requirements

The new system must be to store, organise and sort all of the following fields. Additionally, the user must be able to easily edit fields and add new records. The system must be able to prepare reports and sort them and allow the user to print these. Furthermore, the system must allow for the searching of records by different field parameters and display results.

The record structures and a data dictionary may be found further on in this documentation.

The Objectives of the Solution

<u>The solution must be able to:</u>

1. Produce printable reports with the relevant data.
2. Add, delete and update Usernames, Passwords and Full Names (Accounts).
3. Provide a security barrier to prevent unauthorised access to the system.
4. Limit access to some sections of the system to selected Users.
5. Change the location of the database and back-up the database.
6. Allow users to send direct emails to the developer for Technical Support.
7. Allow users to apply for Travel or Business Expenses by filling in the relevant fields and sending a direct email to the CEO asking for approval.
8. Allow the CEO to approve user applications through the new system and send relevant emails if the application is approved,
9. Add, delete and search these applications for expenses and automatically assign tracking numbers.
10. Allow users to add, modify and delete travel and business expenses.
11. Add, edit and delete revenues and wages and all their relevant fields.
12. Display all this data in a table-based view.
13. Allow the entry, addition and deletion of data through a form-based interface.
14. Sort data based on specific fields.
15. Query data based on specific fields.

Installation Procedure

Prerequisites

The files required for installation are located on several USB drives which have been provided to Odin Health. They have also been securely stored on a cloud storage facility which also allows for any new updates to be pushed.

Firstly, it is important that the .NET Framework 4.5 and Access Database Engine are installed or else the program will not work. Normally, the .NET Framework is already installed on Windows devices but the Access Database Engine is not.

Follow the following link to download the .NET Framework Setup:
http://www.microsoft.com/en-nz/download/details.aspx?id=30653

Follow the following link to download the Access Database Engine (also included on the USB drives):
http://www.microsoft.com/en-us/download/details.aspx?id=23734

Odin Business Manager Installation

Follow the following steps below after the prerequisite steps have been satisfied.

1. Insert the USB drive into the computer
2. Navigate to the 'Install' Folder on the USB and click on setup.exe
3. Follow the instructions on the InstallShield Wizard. Select a different directory to install in if needed.
4. Click Install. If a prompt appears asking for your approval then select 'Yes'.
5. The program has been installed and a shortcut is available on the desktop.

<u>Then perform the following steps to begin using the program:</u>

1. Open the Odin Business Manager
2. If the program is blurry and pixelated then right click on it, go to 'Compatibility' and check 'Disable display scaling on high DPI settings" and finally click 'OK'.
3. Change the database location from the Settings so that your database can work. (Click File then Settings on the login form)
4. Change the email for applications in the Settings form.
5. Use the Account Manager (Click File then Account Manager) to add an account to the database so that you can login to the system.

Please note that unhandled errors may appear if the database location is not set correctly so it is beneficial to do so before using the program.

Database Design

Microsoft Access was only used to create the tables and fields. As mentioned in the design stage of the project, relational databases have not been used in the project in order to allow for the complete deletion and inserting of data. All ‘validation’ occurs on the forms.

Data Structures and Dictionary

The data types shown on these record structures are the types displayed on the Access table design view. The field size has been measured in bytes.

Accounts Table

This is the accounts table which will store the data required for the user to login. Furthermore, it contains information on the user’s full name and whether or not they are an ‘Admin’. The user’s full name is used to populate a combination box for selecting names when submitting applications, adding expenses, etc. while the ‘Admin’ Boolean is used to enable some controls on the forms to edit sensitive data.

The primary key for this table is the ‘Username’.

Field Name	Data Type	Field Size	Description	Example
Username	Short Text	40	Username for user to login	william.shen
Password	Short Text	50	Password for username	Password
FullName	Short Text	80	First and last name	William Shen
Admin	Yes/No	1	Used to lock some users out	Yes

BEA Table

The BEA table is short for Business Expense Applications. It contains the data which is required when processing this sort of application such as Name, Email, Item Description, etc. The primary key of this table is the Tracking Number.

Field Name	Data Type	Field Size	Definition	Example
TrackingNumber	Short Text	10	Unique tracking number	BS1106151
FullName	Short Text	80	First and last name	William Shen
Email	Short Text	255	Email address of user	me@kings.net.nz
ItemDescription	Short Text	100	Description of item	Surface Pro 3
Quantity	Number	2	How many of this item	12
UnitPrice	Currency	8	Cost per item	$23.50
TotalPrice	Currency	8	Total cost	$282.00
Vendor	Short Text	50	The vendor of the items	Microsoft NZ
Reason	Short Text	255	Reason for this purchase	I need a new computer.
Approved	Yes/No	1	Used to approve applications	Yes

Business Expenses Table

This table is used to store all the data on business expenses once they have incurred. It contains important data required for record-keeping purposes and for overall analysis. Many of the fields in this table have been 'brought over' from the BEA table. The primary key in this table is the Tracking Number.

Field Name	Data Type	Field Size	Definition	Example
TrackingNumber	Short Text	10	Unique tracking number	BS1106151
FullName	Short Text	80	First and last name	William Shen
DateIncurred	Date/Time	8	Date expense incurred	15/06/2015
Description	Short Text	100	Description of item	Surface Pro 3
Quantity	Number	2	How many of this item	12
UnitPrice	Currency	8	Cost per item	$23.50
TotalPrice	Currency	8	Total cost	$282.00
Vendor	Short Text	50	The vendor of the items	Microsoft NZ
PaymentMethod	Short Text	30	How the expense was paid for	Personal
ReimbursementRequired	Yes/No	1	If reimbursement is required	Yes
Reimbursed	Yes/No	1	Expense has been reimbursed or not	No

Investment Table

This table will be used to store the information about investments, the inflow of money. The fields contained in this table are very brief and concise as investments do not need to be recorded precisely in this situation. The primary key of this field is the ID which is automatically assigned when the record is inserted into the database.

Field Name	Data Type	Field Size	Definition	Example
ID	AutoNumber	4	Unique ID to identify each investment	5
Investor	Short Text	50	The organisation who invested	William Shen
DateInvested	Date/Time	8	Date money was invested	15/06/2015
InjectionAmount	Currency	8	How much was invested	$25600.00

Revenues Table

The revenues table is used to store brief information about the revenues made by the company. It has been changed from the design stage to give a more of an ‘overview’ rather than a detailed view. The primary key of this table is the Purchase Order Number (PONumber).

Field Name	Data Type	Field Size	Definition	Example
PONumber	Short Text	15	Unique purchase order number	123456789
InvoiceNumber	Short Text	15	The invoice number	123456789
InvoiceDate	Date/Time	8	Date invoice was issued	19/10/2015
Customer	Short Text	50	Who invoice is addressed to	China Software
Subtotal	Currency	8	Total before GST added	$6520.00
GST	Currency	8	GST	$978.00
Total	Currency	8	Total with GST added	$7498.00

TEA Table

The TEA table is short for Travel Expense Applications. It contains the data which is required when processing this sort of application such as Name, Email, Reason, Estimated costs, etc. The primary key of this table is the Tracking Number.

Field Name	Data Type	Field Size	Definition	Example
TrackingNumber	Short Text	10	Unique tracking number	TV1106151
FullName	Short Text	80	First and last name	William Shen
Email	Short Text	255	Email address of applicant	me@kings.net.nz
Destinations	Short Text	80	Destinations planned to go to	Shanghai, China
Duration	Number	2	How many days the trip is planned	12
Reason	Short Text	255	Reason for this trip	Make a new business deal.
AccomodationCosts	Currency	8	Estimated accommodation costs	$1200.00
DomesticTravelCosts	Currency	8	Estimated domestic travel costs	$500.00
InternationalTravelCosts	Currency	8	Estimated international travel costs	$2000.00
TotalCost	Currency	8	The estimated total cost	$3700.00
Approved	Yes/No	1	Used to check whether application has been approved or not	No

Travel Expenses Table

This travel expenses table is used to store the data about the travel expenses. It contains data about each travel expense in detail such as Date Incurred, Expense Type, Payment Method, etc. The difference between 'Reimburse' and 'Reimbursed' is that 'Reimburse' is equivalent to if a reimbursement is required while 'Reimbursed' is whether the expense has already been reimbursed or not. The primary key of this table is an automatically generated ID.

Field Name	Data Type	Field Size	Definition	Example
ID	AutoNumber	4	Unique ID to identify each record	18
TrackingNumber	Short Text	10	Tracking number previously assigned	TV1106151
FullName	Short Text	80	First and last name	William Shen
DateIncurred	Date/Time	8	Date expense was incurred	12/07/15
ExpenseType	Short Text	20	Type of expense	Accommodation
Description	Short Text	100	Short description of this expense	Buffet Dinner
BillClient	Yes/No	1	Whether client will be billed or not	Yes
PaymentMethod	Short Text	30	How the expense was paid for	Personal
ForeignAmount	Currency	8	The foreign currency amount	$1200.00
ForeignCurrency	Short Text	3	The foreign currency type	RMB
ExchangeRate	Currency	8	The exchange rate from Foreign Currency to NZD	$0.2456
NZDAmount	Currency	8	The converted NZD amount	$294.72
Reimburse	Yes/No	1	Whether expense needs to be reimbursed	No
Reimbursed	Yes/No	1	Checks if expense has been reimbursed or not	Yes

Wages Table

The wages table stores all information about the wage payments. It includes fields such as Start Date, End Date, Net Pay, etc. The primary key of this table is an automatically generated ID field.

Field Name	Data Type	Field Size	Definition	Example
ID	AutoNumber	4	Unique ID to identify each wage payment	12
StartDate	Date/Time	8	Beginning date for payment	11/04/2015
EndDate	Date/Time	8	End date for payment	15/04/2015
FullName	Short Text	80	First and last name of employee	William Shen
DatePaid	Date/Time	8	Date wage was paid	12/04/2015
GrossPay	Currency	8	The gross pay	$5000.00
PAYE	Currency	8	PAYE amount	$500.00
KiwiSaver	Currency	8	KiwiSaver amount	$400.00
NetPay	Currency	8	The final net pay	$4100.00

Validation Rules

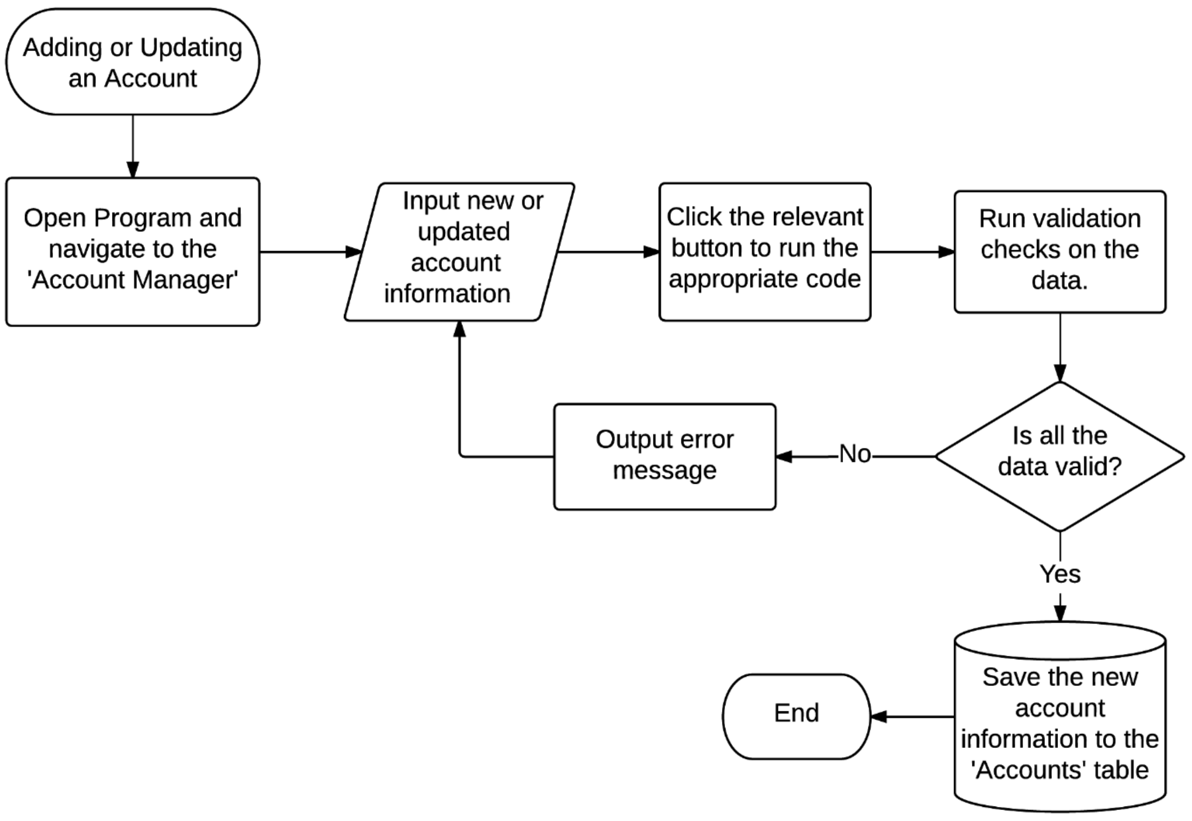
The following are the rules for validation on each relevant form. A large majority of 'validation' has already been carried out in the program by restricting user input and thus have not been included here (e.g. no letters in currency textboxes).

Form	Rule
Business Expense – New Business Expense – View and Edit	All controls contain text.
Business Expense Applications – New Business Expense Applications – View and Edit	All controls contain text.
Contact Support	All controls contain text, checkbox checked and email address is valid.
Investment	All controls contain text.
Revenues –New Revenues – View and Edit	All controls contain text.
Travel Applications – New Travel Applications – View and Edit	All controls contain text.
Travel Expense	All controls contain text and length of currency type is exactly 3 characters (e.g. RMB)
Wages – New Wages – View and Edit	All controls contain text, the net pay is greater than 0, and gross pay is bigger than PAYE and KiwiSaver.

System Flowcharts and Data Flow

Adding or Updating an Account

Adding or updating a login account is an important task to perform when either a new employee enters a company, or someone's login details becomes vulnerable. The deletion of an account is relatively similar except the user chooses the Username from a drop-down list and then presses the button 'Delete'.

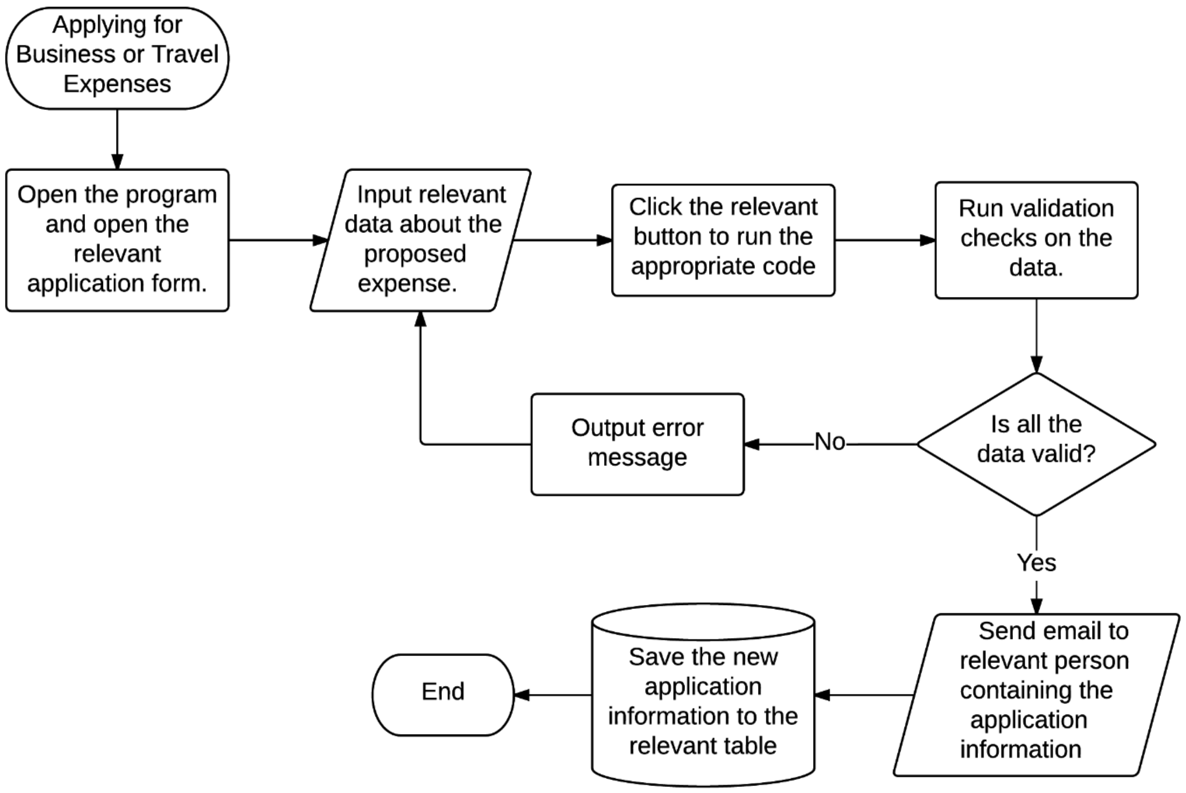


Adding or Updating an Account
Open Program and navigate to the 'Account Manager'
Input new or updated account information
Click the relevant button to run the appropriate code
Run validation checks on the data.
Is all the data valid?
No
Output error message
Yes
Save the new account information to the 'Accounts' table
End

Applying for Business or Travel Expenses.

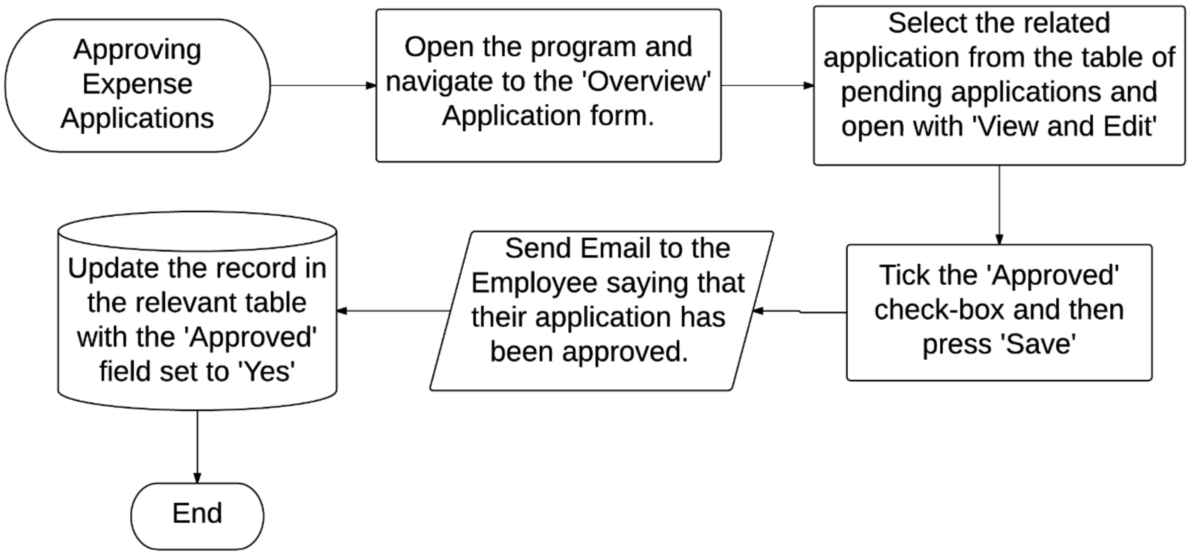
It is important that employees are able to directly and easily apply for either travel or business expenses. The proposed system would increase efficiency as an application would be directly emailed to the relevant person (e.g. CEO) when it is submitted. The relevant forms are the 'Business Expense Application – New' and 'Travel Application – New' forms which may be accessed through their overview forms.

The system flowchart may be found on the next page.


Applying for Business or Travel Expenses
Open the program and open the relevant application form.
Input relevant data about the proposed expense.
Click the relevant button to run the appropriate code
Run validation checks on the data.
Is all the data valid?
No
Output error message
Yes
Send email to relevant person containing the application information
Save the new application information to the relevant table
End

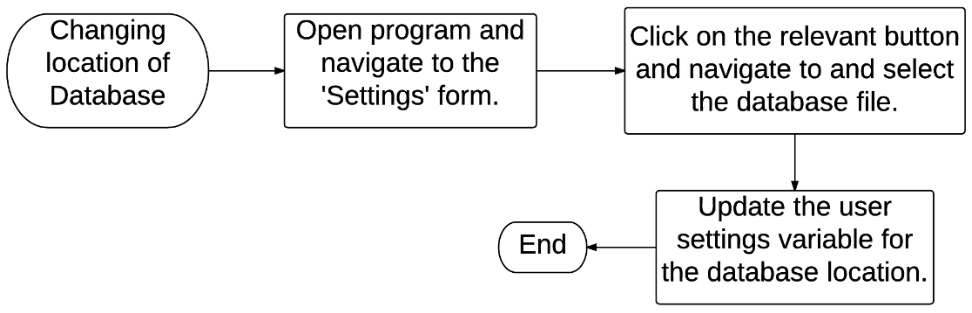
Approving Applications for Expenses

The new system must provide a way for the CEO or relevant person to approve the expense applications of the employees. As shown by the system flowchart, this process is very intuitive and helpful as it allows the sending of an email to the employee indicating that their application has been approved. It must be noted that only an Admin will be able to approve an application.


Approving Expense Applications
Open the program and navigate to the 'Overview' Application form.
Select the related application from the table of pending applications and open with 'View and Edit'
Tick the 'Approved' check-box and then press 'Save'
Send Email to the Employee saying that their application has been approved.
Update the record in the relevant table with the 'Approved' field set to 'Yes'
End

Updating the Database Location

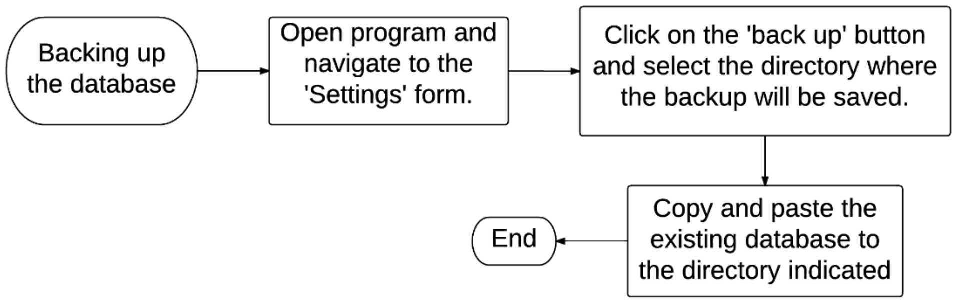
This is the system flowchart for updating or changing the location of the database. The program will utilise an 'Open File Dialog' which would be easy and efficient to use for any user.


Changing location of Database
Open program and navigate to the 'Settings' form.
Click on the relevant button and navigate to and select the database file.
Update the user settings variable for the database location.
End

Backing up the Database

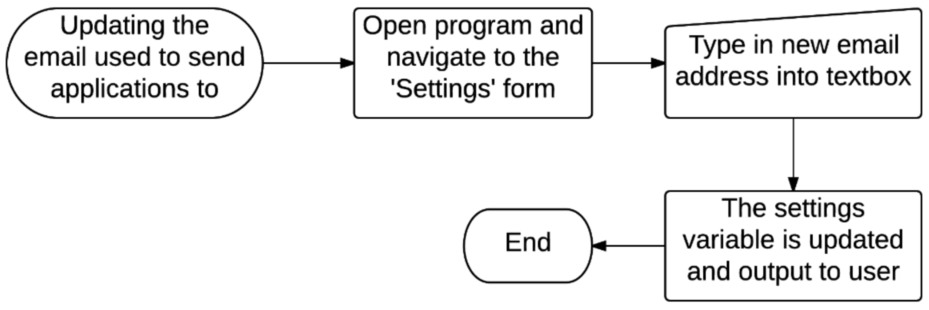
Backing up the database is important for cases where the main database file becomes corrupted and thus unusable. It is advised that the client makes a backup at least every week.

The way the proposed system will back up the database is by copying the current database and pasting it to a location which the user has indicated.


Backing up the database
Open program and navigate to the 'Settings' form.
Click on the 'back up' button and select the directory where the backup will be saved.
Copy and paste the existing database to the directory indicated
End

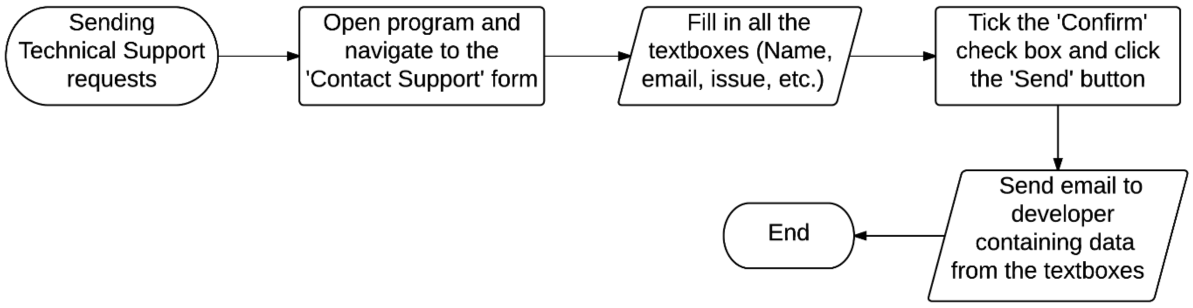
Changing the Email Address which Applications are sent to

It may be required to change the email address for which applications are sent to if there is a change in manager or an introduction of a resources manager. Thus, it is important to be able to change this email.


Updating the email used to send applications to
Open program and navigate to the 'Settings' form
Type in new email address into textbox
The settings variable is updated and output to user
End

Contacting Technical Support

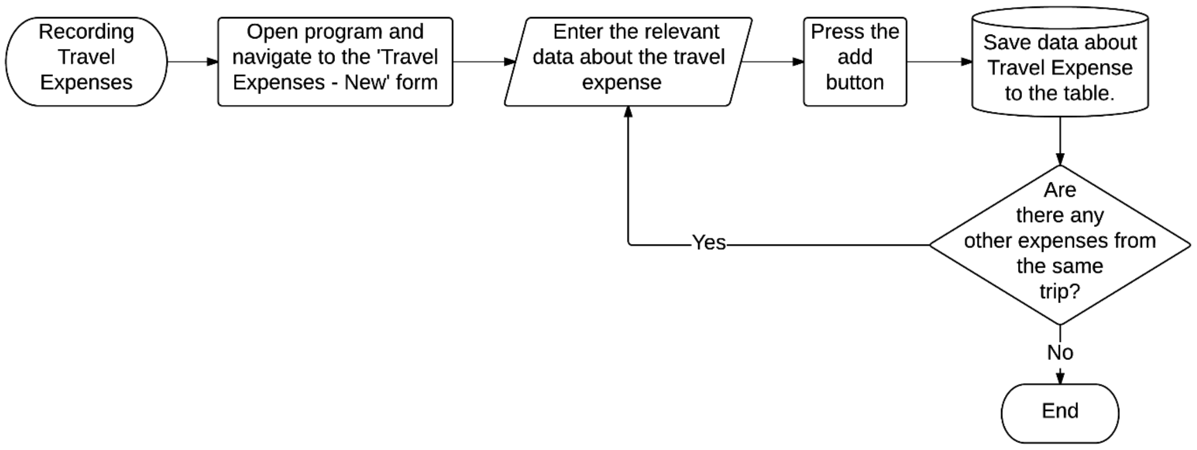
It is important for the client to relay any problems they are experience to the developer who can then go on to fix these glitches or bugs and then 'push' a new version of the solution to the client. Furthermore, general enquiries about the program can be sent to the developer


Sending Technical Support requests
Open program and navigate to the 'Contact Support' form
Fill in all the textboxes (Name, email, issue, etc.)
Tick the 'Confirm' check box and click the 'Send' button
Send email to developer containing data from the textboxes
End

Adding Travel Expenses

All travel expenses are costs incurred to Odin Health and must be carefully recorded by all employees. Furthermore, these travel expenses contain information that is required in order to reimburse the employees for their purchases.

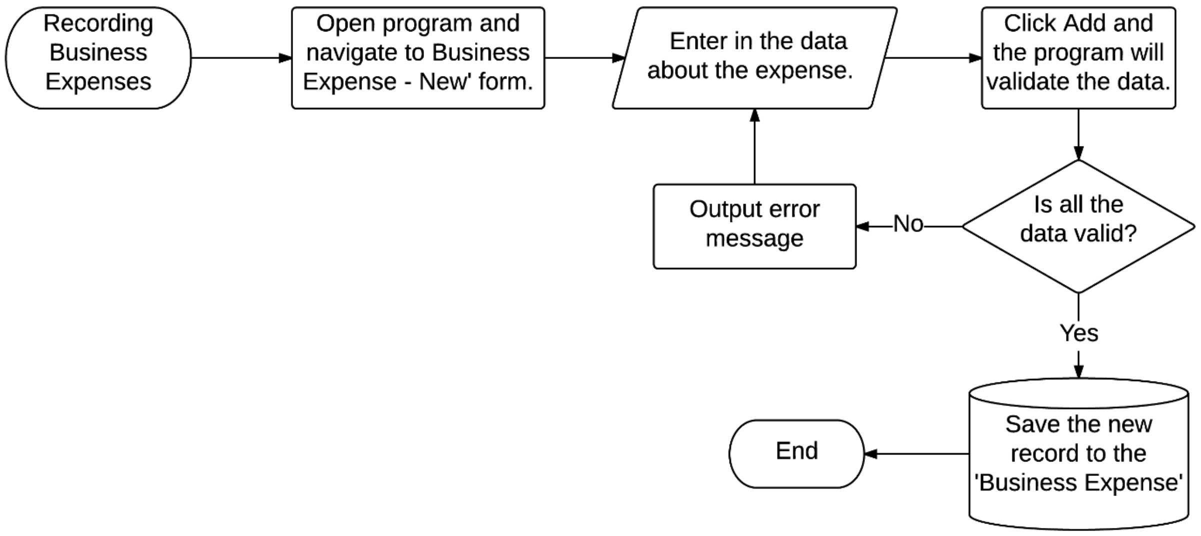
The program will allow the user to add multiple entries via the same tracking number by using a unique ID for the primary key instead of the tracking number itself. The program will additionally allow the user to print out a copy of their expenses once they have completed adding them for a single tracking number.


Recording Travel Expenses
Open program and navigate to the 'Travel Expenses - New' form
Enter the relevant data about the travel expense
Press the add button
Save data about Travel Expense to the table.
Are there any other expenses from the same trip?
Yes
No
End

Adding Business Expenses

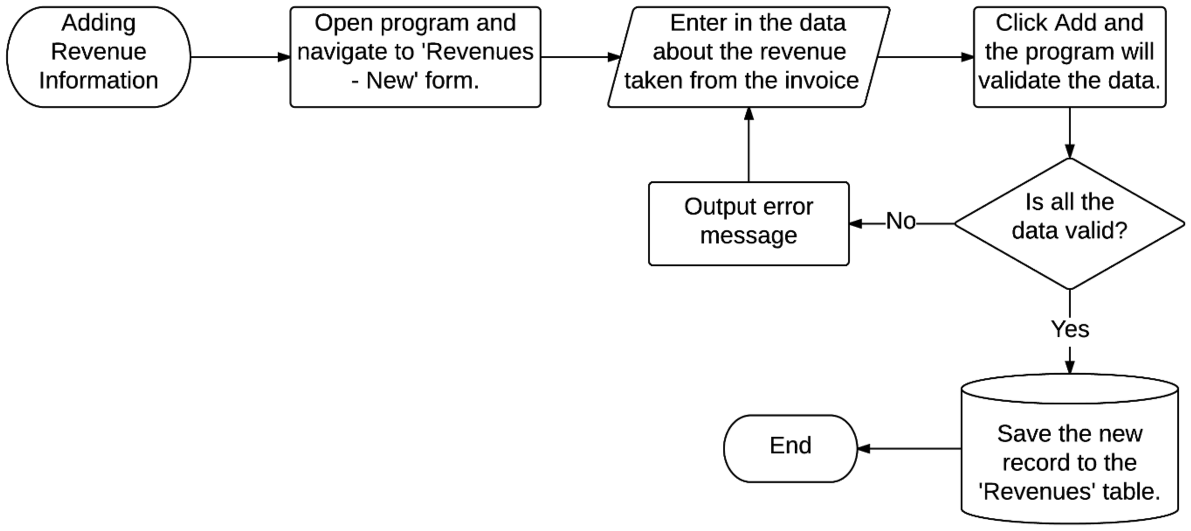
The business expenses incurred by the company must be recorded for legal purposes and so that employees may be reimbursed. The unique identifier for these business expenses is the tracking number.

This proposed system would greatly increase the efficiency of recording business expenses as new spreadsheets do not need to be created.


Recording Business Expenses
Open program and navigate to Business Expense - New' form.
Enter in the data about the expense.
Click Add and the program will validate the data.
Is all the data valid?
No
Output error message
Yes
Save the new record to the 'Business Expense'
End

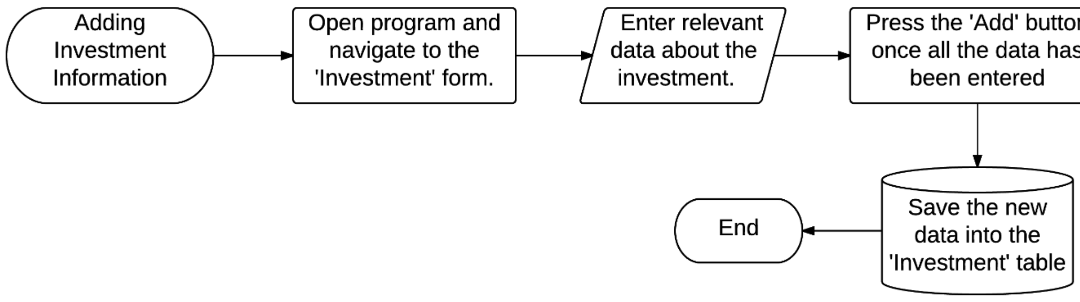
Adding Revenues

Revenues are an integral part of a business which allows them to continue with their operations, expand and research on new technologies. These must be carefully recorded to allow indication of overspending in a company. The data on this form would be directly taken from the 'Sales Invoice'.


Adding Revenue Information
Open program and navigate to 'Revenues - New' form.
Enter in the data about the revenue taken from the invoice
Click Add and the program will validate the data.
Is all the data valid?
No
Output error message
Yes
Save the new record to the 'Revenues' table.
End

Adding Investments

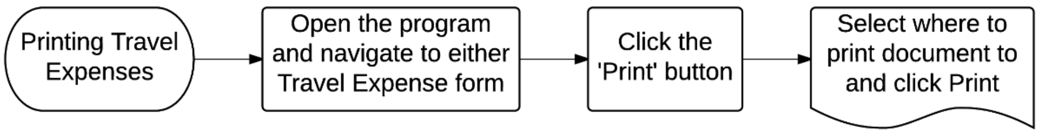
Investments indicate that the general consensus of a company is good and that it is in a good business health and position. It is vitally important that these are recorded as with revenues so that Odin can keep track of their costs and profits and check if they are running a surplus or a deficit.


Adding Investment Information
Open program and navigate to the 'Investment' form.
Enter relevant data about the investment.
Press the 'Add' button once all the data has been entered
Save the new data into the 'Investment' table
End

Printing Travel Expenses

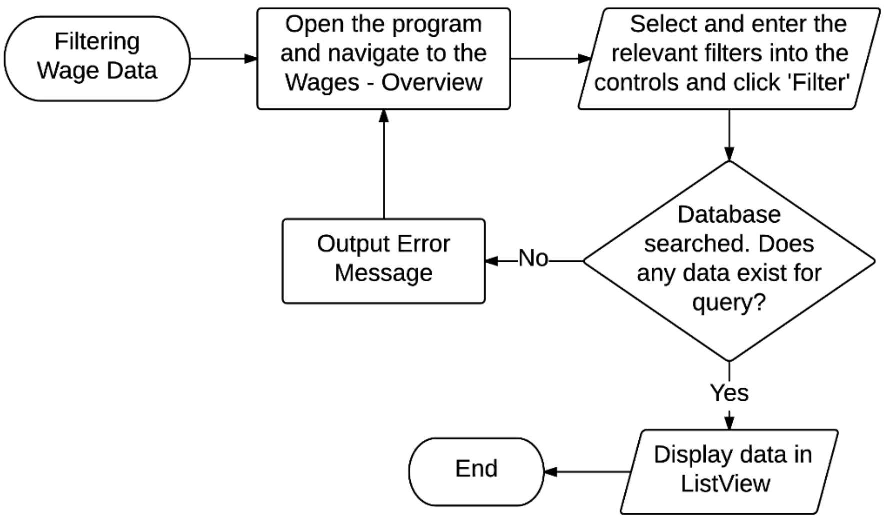
It is important to be able to print travel expenses so that they can be analysed and then relevant expenses reimbursed for the employees. Travel Expenses must also be recorded carefully for record-keeping purposes.

All other report printing follows a similar structure and thus will not be detailed on system flowcharts.


Printing Travel Expenses
Open the program and navigate to either Travel Expense form
Click the 'Print' button
Select where to print document to and click Print

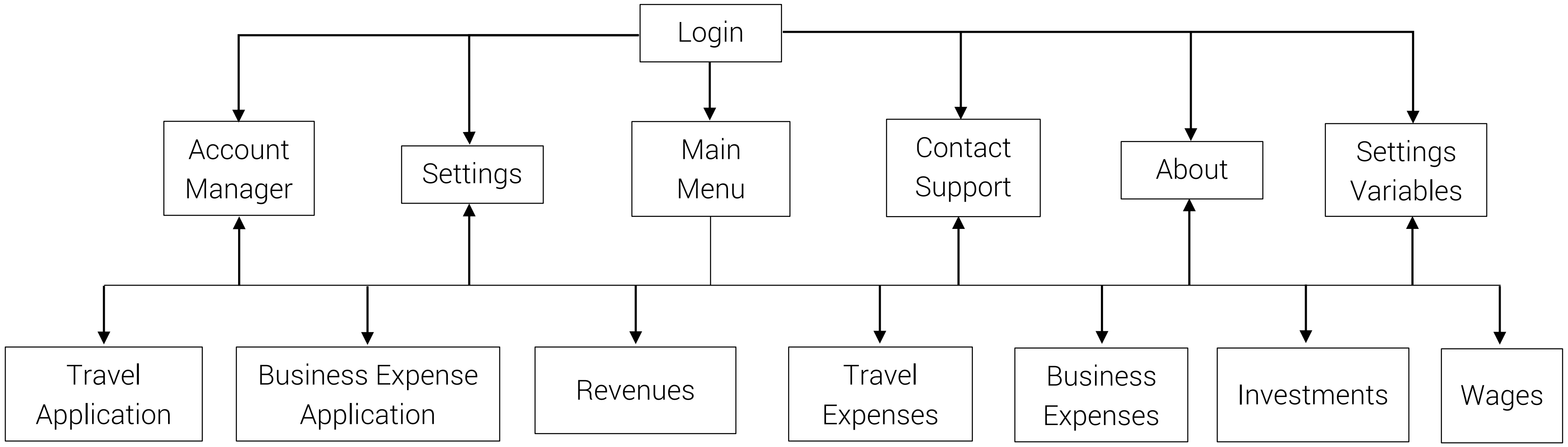
Filtering Wage Data (Basic Query)

In some situations, data may need to be filtered to show data for one employee, between one time period, etc. As a result, relevant forms contain filtering functions which allow data to be successfully filtered and viewed. Below is an example.


Filtering Wage Data
Open the program and navigate to the Wages - Overview
Select and enter the relevant filters into the controls and click 'Filter'
Database searched. Does any data exist for query?
No
Output Error Message
Yes
Display data in ListView
End

Navigation Paths

The following is a map containing the different pathways to navigating around the system. The different forms for each main function (i.e. overview, new and view and edit) have not been listed in this navigation path map as they entail the same use.



Program Listings

Please note that full program listings are located in a separate document due to their large size. The code has been produced with good procedural programming techniques including annotations to code, whitespace, meaningful identifier names and advanced formatting.

Algorithm Design

Updating the clock on the Main Menu form

This algorithm makes use of a timer set to tick every 1000 milliseconds to update the labels on the Main Menu form to show the date and the time. It is only used on the Main Menu form and will keep running through every second if the form is open.

```
Private Sub DateTimer_Tick(sender As Object, e As EventArgs) Handles DateTimer.Tick
      'Code to update the date and time shown on labels
      lblTime.Text = DateTime.Now.ToString("dddd dd MMMM yyyy")
      lblDate.Text = DateTime.Now.ToString("hh:mm:ss tt")
End Sub
```

Limiting Input for Quantities

This algorithm is used to limit a control to accept numbers only. This is used for controls such as quantity on Business Expenses. However, the algorithm allows the backspace key to be pressed.

```
Shared Sub numbersOnly(ByVal sender As Object, ByVal e As System.Windows.Forms.KeyPressEventArgs)
    'Checks if keypress is not a backspace
    If e.KeyChar <> Chr(&H8) Then
        'Checks if keypress is not a digit
        If Not Char.IsDigit(e.KeyChar) Then
            e.Handled = True
        End If
    End If
End Sub
```

Limiting Input for Currencies

The algorithm listed below will be executed when a key is pressed in a relevant control. The system will only accept either digits or a decimal point. The backspace key may also be pressed. However, there may only be one decimal point in the control and only two numbers after that decimal point. This algorithm has been implemented to controls such as Estimated Costs on the Travel Expense Application form.

```
Shared Sub twoDecimalPoints(ByVal sender As Object, ByVal e As System.Windows.Forms.KeyPressEventArgs)
    'Handles the event if the key pressed is not a digit or a decimal point
    If Not Char.IsDigit(e.KeyChar) And Not e.KeyChar = "." Then
        e.Handled = True
    Else
        'Handles the keypress if it is a decimal and it currently does exist (does not allow two decimal points)
        If e.KeyChar = "." And sender.Text.IndexOf(".") <> -1 Then
            e.Handled = True
        ElseIf e.KeyChar = "." Then 'If the keypress is a decimal point and none exists then allows input
            e.Handled = False
        ElseIf Char.IsDigit(e.KeyChar) Then  'If the keypress is a digit then checks if there is a decimal point
            If sender.Text.IndexOf(".") <> -1 Then
                'Handles if there are 2 numbers after the decimal point
                If sender.Text.Length >= sender.Text.IndexOf(".") + 3 Then
                    e.Handled = True
                End If
            End If
        End If
    End If
    'Allow backspace in textbox
    If e.KeyChar = Chr(&H8) Then
        e.Handled = False
    End If
End Sub
```

Different Combinations for Business Expense Filters

Here is the algorithm used to account for all the different possibilities of filtering values presented in the Business Expense - Overview form. There are totally 16 combinations. The algorithm works on the logic that each checkbox checked will add a value to x to form a final value which will be checked against with a Select Case statement and then the appropriate code assigned to search the database.

Please note that the database queries have been removed from this program listing for ease of viewing.

```
'Used for the different combinations
If chkFilterDate.Checked = True Then x = 1
If chkFilterName.Checked = True Then x = x + 2
If chkCompleted.Checked = True Then x = x + 4
If chkPending.Checked = True Then x = x + 8
If chkAll.Checked = True Then x = 16
'Selects the different cases. 15 possibilities + 1 possibility for chkAll
Select Case x
      'Declares relevant OleDbCommands and gives them the parameters
      Case 1 'Date only
      Case 2 'Name only
      Case 3 'Date and Name
      Case 4 'Completed only
      Case 5 'Date and Completed
      Case 6 'Name and Completed
      Case 7 'Date, Name and Completed
      Case 8 'Pending only
      Case 9 'Date and Pending
      Case 11 'Date, Name and Pending
      Case 12 'Completed and Pending (same as everything)
      Case 13 'Date, Completed and Pending (same as Date only)
      Case 14 'Name, Completed and Pending (same as Name only)
      Case 15 'Date, Name, Completed, Pending (same as Date and Name only)
      Case 16 'All checkbox is checked
 End Select
```

Sending Emails

The algorithm listed below may be found on multiple forms such as “Contact Support” or “Travel Applications” but with different contents. The code is used to set up an email for sending and then sends it. The format of the email has been predefined and the controls will be used to fill in the relevant details required.

```
Private Sub SendEmail()
    'Creates new message
    Dim Email As New MailMessage()
    'Sets the email preferences
    Email.From = New MailAddress(txtEmail.Text)
    Email.[To].Add("w.shen+odin@kings.net.nz")
    'Sets the content of the email
    Email.Subject = "Odin Support Request from " & txtName.Text
    Email.Body = "<H2>Support Request for 'Odin Health Business Manager'</H2>" & "Time Submitted: " & _
       DateTime.Now.ToString("dd/MM/yyyy HH:mm:ss") & "<BR>Name: " & txtName.Text & "<BR>Email Address: " & _
       txtEmail.Text & "<BR><BR><u>Problem Description:</u><BR>" & rtbMessage.Text
    Email.IsBodyHtml = True
    'Sets the server to send the email
    Dim smtp As New SmtpClient("smtp.gmail.com")
    smtp.UseDefaultCredentials = False
    smtp.Credentials = New Net.NetworkCredential("shen.odinhealth@gmail.com", "c9dMy6RKPr")
    smtp.Port = 587
    smtp.EnableSsl = True
    smtp.Host = "smtp.gmail.com"
    'Sends the email and shows feedback
    disableControls()
    smtp.Send(Email)
    lblStatus.ForeColor = Color.Green
    lblStatus.Text = "Your message has been successfully sent. Thank You."
End Sub
```

Passing Function values to the Search Box form

The algorithm shown below is used to pass the relevant function value to the Search Box form so that it can execute the appropriate code once the 'Search' button is pressed

```
Private Sub btnSearch_Click(sender As Object, e As EventArgs) Handles btnSearch.Click
    'Opens search box form and passes relevant function name
    SearchBox.lblFunction.Text = "BusinessExpense"
    SearchBox.ShowDialog()
End Sub
```

Updating the Email which applications are sent to

This algorithm is located on the Settings form and is used to update the email to which the Business Expense and Travel Applications are sent to. Firstly, the code checks if the email address is valid and will then update the settings accordingly. On the other hand if the email address is not valid, the program will output an error to the user.

```
Private Sub btnApplicationEmail_Click(sender As Object, e As EventArgs) Handles btnApplicationEmail.Click
    'Checks if the email address in the textbox is in the correct format or not
    Try
        Dim Email As New MailAddress(txtApplicationEmail.Text)
        'Sets the settins variable to the email in the textbox
        My.Settings.applicationEmail = txtApplicationEmail.Text
        'Output to show email was updated
        lblApplicationEmail.ForeColor = Color.Green
        lblApplicationEmail.Text = "Email successfully updated."
    Catch
        'Output to show error in validating email address
        lblApplicationEmail.ForeColor = Color.Red
        lblApplicationEmail.Text = "Email address in incorrect format."
    End Try
End Sub
```

Changing the database location

As outlined in the objectives, the user must be able to change the location of the database and this is done through the settings form. An open file dialog will be opened where the user can navigate to the database file and select it. The algorithm will then update the setting for database location with the location of the user selected file.

```
Private Sub btnLocation_Click(sender As Object, e As EventArgs) Handles btnLocation.Click
    'Declares an OpenFileDialog and assigns it a title
    Dim OpenFD As OpenFileDialog = New OpenFileDialog
    OpenFD.Title = "Select your Database File"
    'Opens file dialog and checks if OK button was pressed or not
    If OpenFD.ShowDialog = Windows.Forms.DialogResult.OK Then
        'Sets the settings variable for the database location to the file the user selected
        My.Settings.dbLocation = OpenFD.FileName
        'Output to user showing success
        lblLocationDirectory.Text = My.Settings.dbLocation
        lblLocationStatus.Visible = True
        lblLocationStatus.ForeColor = Color.ForestGreen
        lblLocationStatus.Text = "Database Location successfully updated."
        'Displays a MsgBox asking user whether to restart the program
        Dim Response = MsgBox("The program must restart for changes to take full effect. Restart now?",
                              MsgBoxStyle.YesNo, "Restart Odin")
        If Response = MsgBoxResult.Yes Then
            'Restarts application if user selects yes
            Application.Restart()
        Else : MsgBox("Please ensure you restart the program.")
        End If
    Else
        'Output to user showing the database was not updated
        lblLocationStatus.Visible = True
        lblLocationStatus.ForeColor = Color.Red
        lblLocationStatus.Text = "Database Location was not updated."
    End If
End Sub
```

Backing up the database

Listed below is the code used to make a backup of the database through the Settings form. As shown by the code, a folder browser will be opened and the user will navigate and select the location he or she wishes to save the database to. Then the algorithm will find the location of the current database and copy and paste it to the user selected location and provide appropriate output.

```
Private Sub btnBackup_Click(sender As Object, e As EventArgs) Handles btnBackup.Click
    'Declares FolderBrowserDialog and gives it a description.
    Dim FolderBD As FolderBrowserDialog = New FolderBrowserDialog
    FolderBD.Description = "Select the directory where you want the database backed up to."
    'Opens the folder browser dialog and checks if OK button was pressed or not
    If FolderBD.ShowDialog = Windows.Forms.DialogResult.OK Then
        'Declares the directory and file name for the new file
        Dim Directory As String = FolderBD.SelectedPath & "\OdinHealth Database Backup (" &
                                  DateTime.Now.ToString("dd-MM-yyyy") & ").accdb"
        'Copies the file from current database location to user selected directory
        My.Computer.FileSystem.CopyFile(My.Settings.dbLocation, Directory, _
                                        Microsoft.VisualBasic.FileIO.UIOption.AllDialogs, _
                                        Microsoft.VisualBasic.FileIO.UICancelOption.DoNothing)
        'Output to show success
        lblBackup.ForeColor = Color.ForestGreen
        lblBackup.Text = "Database backed up successfully."
        My.Settings.backupDate = DateTime.Today
    Else
        'Output to show database was not backed up
        lblBackup.Visible = True
        lblBackup.ForeColor = Color.Red
        lblBackup.Text = "Database was not backed up."
    End If
End Sub
```

Database Query Design

Inserting Data

This is an example of the code used to add a new record to the database. As shown below, firstly the code will run a validation function and if it returns true, it will then do the following:

1. Declare an OleDbCommand, initialise it and then give it its parameters.
2. Open the connection, execute the query and close the connection.
3. Give output to user and resets the controls

```
Private Sub btnAdd_Click(sender As Object, e As EventArgs) Handles btnAdd.Click
    'Runs validation function and checks if it returns true
    If Validation() Then
        'Declares the new OleDbCommand
        Dim cmd As New OleDbCommand("INSERT INTO Investments (Investor, DateInvested, InjectionAmount) " & _
                                    "VALUES (@Investor, @DateInvested, @InjectionAmount)", connection)
        'Gives the parameters its values
        cmd.Parameters.AddWithValue("Investor", txtInvestor.Text)
        cmd.Parameters.AddWithValue("DateInvested", dtpDate.Value.Date)
        cmd.Parameters.AddWithValue("InjectionAmount", txtAmount.Text)
        'Opens database connection, executes query and closes the database
        connection.Open()
        cmd.ExecuteNonQuery()
        connection.Close()
        'Output to show record has been added and repopulates listview
        lblAddStatus.ForeColor = Color.Green
        lblAddStatus.Text = "Investment information added into database."
        populateListView()
        'Resets controls
        txtInvestor.Text = ""
        dtpDate.Value = DateTime.Today
        txtAmount.Text = ""
    End If
End Sub
```

Updating Data

Listed below is an example of the code used to update a record in the database. This may be found on many "View and Edit" forms. Firstly the subroutine runs a Validation function and if it returns true then does the following:

1. Declare an OleDbCommand, initialise it and give it its parameters.
2. Opens the connection, executes the query and then closes the connection.
3. Gives output to the user and will close the form in some cases.

```
Private Sub btnUpdate_Click(sender As Object, e As EventArgs) Handles btnUpdate.Click
    If Validation() Then
        'Declares the new OleDbCommand and gives it parameters
        Dim cmd As New OleDbCommand("UPDATE [Business Expenses] SET FullName = ?, DateIncurred = ?, Description = ?, " &
                                    Quantity = ?, UnitPrice = ?, TotalPrice = ?, ReimbursementRequired = ?, " &
                                    PaymentMethod = ?, Vendor = ?, Reimbursed = ? WHERE TrackingNumber = '" &
                                    txtTrackingNumber.Text & "'", connection)
        cmd.Parameters.AddWithValue("@p1", txtName.Text)
        cmd.Parameters.AddWithValue("@p2", dtpDate.Value.Date)
        cmd.Parameters.AddWithValue("@p3", txtDescription.Text)
        cmd.Parameters.AddWithValue("@p4", txtQuantity.Text)
        cmd.Parameters.AddWithValue("@p5", txtUnit.Text)
        cmd.Parameters.AddWithValue("@p6", txtTotal.Text)
        cmd.Parameters.AddWithValue("@p7", chkReimbursementRequired.Checked)
        cmd.Parameters.AddWithValue("@p8", cboMethod.Text)
        cmd.Parameters.AddWithValue("@p9", txtVendor.Text)
        cmd.Parameters.AddWithValue("@p10", chkReimbursed.Checked)
        'Executes the command and closes form
        connection.Open()
        cmd.ExecuteNonQuery()
        connection.Close()
        'Output to show successful updation on Overview form and closes form
        Business_Expense___Overview.lblDataStatus.Text = "Record successfully updated."
        Business_Expense___Overview.lblDataStatus.ForeColor = Color.Green
        Business_Expense___Overview.lblDataStatus.Visible = True
    End If
End Sub
```

Deleting Data

The following code is an example of the code used to delete a record from the database. Similar code can be found on many forms which support a delete function. Firstly, a Yes/No message box is presented to user. If the user selects yes, then the following code is run:

1. Declare an OleDbCommand and gives it its value.
2. Open the connection, execute the query and close the connection.
3. Give output to the user and close the form in some cases.

```
Private Sub btnDelete_Click(sender As Object, e As EventArgs) Handles btnDelete.Click
    'Opens a Yes/No messagebox asking user to confirm to delete the record
    Dim Response = MsgBox("Are you sure you want to delete this Business Expense Application?", MsgBoxStyle.YesNo,
                        "Delete Application")
    'Code if user selects yes
    If Response = MsgBoxResult.Yes Then
        'Declares a new OleDbCommand which deletes the record
        Dim TrackingNumber As String = txtTrackingNumber.Text
        Dim cmd As New OleDbCommand("DELETE FROM [Business Expenses] WHERE TrackingNumber = '" & TrackingNumber & "'",
                                 connection)
        'Opens connection, executes the command and closes the connection
        connection.Open()
        cmd.ExecuteNonQuery()
        connection.Close()
        'Output to show successful updation on Overview form
        Business_Expense___Overview.lblDataStatus.Text = "Record successfully deleted."
        Business_Expense___Overview.lblDataStatus.ForeColor = Color.Green
        Business_Expense___Overview.lblDataStatus.Visible = True
        Me.Close()
    End If
End Sub
```

Viewing Data in ListViews

Similar code listed below is found on almost all forms which contain a ListView. This code is used to populate the ListViews by selecting all relevant data from the database and then adding the records one by one into a ListView. Furthermore, this code is run after new data has been added, updated or deleted.

1. Clears any existing data in the ListView and enables the ListView.
2. Declares an OleDbConnection and initialises it.
3. Opens the connection, executes the query and loops through the relevant data.
4. Adds data into the ListView record by record then closes the query.

```
Public Sub populateListViewPending()
    'Clears and enables listview
    lstViewPending.Items.Clear()
    lstViewPending.Enabled = True
    'Opens the connection and declares a new OleDbCommand and Reader
    connection.Open()
    Dim cmd As New OleDbCommand("SELECT * From [Business Expenses] WHERE Reimbursed = False ORDER BY TrackingNumber",
                                connection)
    Dim dr As OleDbDataReader = cmd.ExecuteReader
    'Loops the Business Expenses table of the database and adds each record to a new ListViewItem
    Do While dr.Read()
        Dim newitem As New ListViewItem(dr.Item("TrackingNumber").ToString)
        newitem.SubItems.Add(dr.Item("FullName").ToString)
        newitem.SubItems.Add(dr.Item("DateIncurred"))
        newitem.SubItems.Add(dr.Item("Description").ToString)
        newitem.SubItems.Add(dr.Item("TotalPrice"))
        lstViewPending.Items.Add(newitem)
    Loop
    'Closes the connection
    connection.Close()
End Sub
```

Adaptive Maintenance

Existing Functions

In order to 'ease' previous potential maintenance problems, the 'Settings' form of the program contains controls to change the settings variables for many different functions including: changing the database and changing the email which applications are sent to.

This means that changing the code is not required to change these variables but instead the program itself can be used. These functions may be seen as a form of future-proofing as some changes in the future have been catered for.

Change of Currency

There are many situations which the base currency could change. Perhaps the company relocates to a different country or opens another branch in another country.

If the base currency were to change from NZD to something else then the following steps should be taken:

1. Change the labels and relevant controls in the program that contain 'NZD' to the new base currency.
2. Edit the database fields from 'NZDAmount' to 'ABCAmount' and change the database queries accordingly.
3. Search through the whole program for any matches to 'NZD' and change them to the new base currency (e.g. Exchange Rate Text Changed in TravelExpenses.vb)

Changing Fields to be Printed in Reports

Due to changes in regulation or external influences, it may be required for some different fields to be printed in either the business expense, travel expenses or wage payment reports. To do this, one must change the code in the printing functions on the relevant forms.

The following example is taken from the Travel Expense forms where first a data table is filled with the relevant data, then the source of a DataGridView is set to this datatable, then the column names are changed. Finally, total values are calculated then a function is run to print the report.

Please see the next page. The highlighted text portrays code that may potentially need to be edited to reflect the changes in the database query. Changes required have also been explained in detail.

```
'//PRINTING
Private Sub btnPrint_Click(sender As Object, e As EventArgs) Handles btnPrint.Click
    'Checks if an item has been selected or not
    If lstView.SelectedItems.Count > 0 Then
        'Declares variables to be used in query
        Dim TrackingNumber As String = lstView.SelectedItems(0).Text
        Dim Name As String = lstView.SelectedItems(0).SubItems(1).Text
        'Declares OleDbCommand, Data Adapter and Dataset
        Dim cmd As New OleDbCommand("SELECT Format(DateIncurred, 'dd/mm/yy'), Description, BillClient, ExpenseType, ForeignCurrency " & _
                                    "& ' $' & ForeignAmount, ExchangeRate, '$' & NZDAmount, Reimbursed From [Travel Expenses] WHERE " & _
                                    "TrackingNumber = '" & TrackingNumber & "' ORDER BY DateIncurred", connection)
        Dim da As New OleDbDataAdapter(cmd)
        Dim dt As New DataTable
        'Fills dataset with data from table and outputs it to datagridview
        da.FillSchema(dt, SchemaType.Source)
        dt.Columns(2).DataType = GetType(String)
        dt.Columns(5).DataType = GetType(String)
        dt.Columns(7).DataType = GetType(String)
        da.Fill(dt)
        'Displays data on datagridview and sets the column names
        DataGridView.DataSource = dt
        With DataGridView
            .Columns(0).HeaderCell.Value = "Date"
            .Columns(2).HeaderCell.Value = "BC"
            .Columns(3).HeaderCell.Value = "Type"
            .Columns(4).HeaderCell.Value = "Amount"
            .Columns(5).HeaderCell.Value = "XRT"
            .Columns(6).HeaderCell.Value = "NZD"
            .Columns(7).HeaderCell.Value = "Repaid"
        End With
        'Loops through datagridview and replaces True with Yes and False with Nothing for BillClient and
        'False with No for Repaid. Calculates total for the NZD and Reimbursed column (if it is false) of the datagridview
        Dim totalDec As Decimal = 0
        Dim repaidDec As Decimal = 0
        For i = 0 To DataGridView.Rows.Count - 1
            'Checks if cell for Bill Client contains True
            If DataGridView.Rows(i).Cells(2).Value = "True" Then
                DataGridView.Rows(i).Cells(2).Value = "Yes"
            Else : DataGridView.Rows(i).Cells(2).Value = ""
            End If
            'Checks if cell for Repaid contains True
            If DataGridView.Rows(i).Cells(7).Value = "True" Then
                DataGridView.Rows(i).Cells(7).Value = "Yes"
```

Change query to include or delete fields which are required.

Change the numbers to reflect their column number in the DataGridView (starting from 0)

Change the text to reflect the different columns. Abbreviations may be required.

Change the numbers to reflect a change in the ‘Bill Client’ column number.

Change the numbers to reflect a change in the ‘Repaid’ column number.

```
        Else
            DataGridView.Rows(i).Cells(7).Value = "No"
            repaidDec += DataGridView.Rows(i).Cells(6).Value
        End If
        'Sums the values in the NZD column
        totalDec = totalDec + DataGridView.Rows(i).Cells(6).Value
    Next i
    'Rounds the total variables to 2 decimal points and converts them to string with $ symbol
    totalDec = Decimal.Round(totalDec, 2, MidpointRounding.AwayFromZero)
    repaidDec = Decimal.Round(repaidDec, 2, MidpointRounding.AwayFromZero)
    Dim totalStr As String = "$" & CStr(totalDec)
    Dim repaidStr As String = "$" & CStr(repaidDec)
    'Adds new row to DataTable to update the datagridview. Sets values in rows then adds
    Dim newRow As DataRow = dt.NewRow
    For i = 0 To 5
        newRow.Item(i) = ""
    Next
    newRow.Item(6) = totalStr
    newRow.Item(7) = repaidStr
    dt.Rows.Add(newRow)
    'Runs function to set up printing
    If SetupThePrinting(TrackingNumber & " - " & Name) Then
        'Declares and opens print preview dialog
        Dim ppd As PrintPreviewDialog = New PrintPreviewDialog()
        ppd.Document = pntDocument
        ppd.Height = Screen.PrimaryScreen.Bounds.Height / 1.5
        ppd.Width = Screen.PrimaryScreen.Bounds.Width / 2.5
        ppd.FindForm().StartPosition = FormStartPosition.CenterScreen
        ppd.Text = "Travel Expenses"
        ppd.ShowDialog()
    End If
Else : MsgBox("Please select a record from the table.") 'If user did not select a listview item
End If
End Sub
```

Change the numbers to reflect a change in the 'Repaid' column number.

Change the numbers to reflect a change in the 'NZD' column number.

Change the numbers to reflect the number of columns that are to be blank

Change the numbers to reflect any changes in 'NZD' or 'Reimbursed' columns

Additional Information

Additional information such as test logs and test results have been included in the overall project file which has been provided to Odin Health.

This project file also contains information about form design, report formatting, etc. This technical guide acts as a document outlining important aspects of the system for technical maintenance in depth.

If there are additional problems then please contact the developer at:
william.w.y.shen@gmail.com