

edpAssignment

[Document subtitle]



August 27, 2019

Lewis Mccormac

Dundalk etb

Contents

[Task 1 2](#_Toc18058342)

[Sub- Introduction 2](#_Toc18058343)

[What is event driven programming 2](#_Toc18058344)

[Characteristics 2](#_Toc18058345)

[Features 3](#_Toc18058346)

[Conclusion 3](#_Toc18058347)

[Task 2 3](#_Toc18058348)

[Sub-introduction 3](#_Toc18058349)

[Design 4](#_Toc18058350)

[Main Window 4](#_Toc18058351)

[Login screen 5](#_Toc18058352)

[Account Screen 6](#_Toc18058353)

[Transaction menu 7](#_Toc18058354)

[Task 2.5 7](#_Toc18058355)

[Task 3 7](#_Toc18058356)

[Sub-introduction 7](#_Toc18058357)

[Test 1 8](#_Toc18058358)

[Test 1 results 8](#_Toc18058359)

[Test 2 9](#_Toc18058360)

[Test 2 results 9](#_Toc18058361)

[Test 3 10](#_Toc18058363)

[Test 3 results 10](#_Toc18058364)

[Test 4 11](#_Toc18058365)

[Test 4 results 11](#_Toc18058366)

[Security issues 11](#_Toc18058367)

# Task 1

# Sub- Introduction

In task 1 I will talk about the principles and features of the event driven paradigm. It will talk about what is event driven programing and some of the parts that make it usable and so flexible.

# What is event driven programming

Event driven programming is at its core based on user interaction and works through clicks and other user interactions. The program would have event listeners which wait for events to continue the flow of the program.

Events can be:

* Mouse clicks
* Keyboard inputs
* Mouse hovers
* Button presses
* Time driven

Time driven is a somewhat unique event as it doesn’t directly require user input. It can be started by a user but after a certain length of has elapsed it would run the code. This could be anything from an alarm clock which would be set by a user. Or an inactivity log out in a banking app or video game, which would wait for user input and if no input is detected for a certain length of time the user would be logged out.

At the basic level all operating systems in common use today are examples of event driven programming because the operating system doesn’t wait for a certain piece of information but instead is waiting for an array of different input and reacts based on the object clicked or option ticked.

# Characteristics

Event driven programming can be very closely related with object orientated programming. The event handlers would wait on input from the user and then take that information and pass it on to an object, or based on their inputs it would output information that is in the object or it would change information in the object.

This makes it a very flexible and strong paradigm because it can interact with objects easily and its flow would be based on the events created by the user.

# 

# Features

The main features of event driven programming are:

* Event handlers
* Events
* Time driven events
* Event loops
* Forms
* Trigger function

End of task 1

# Task 2

## Sub-introduction

Task 2 is broken down into two parts, part one is design of the banking login system, while part two is the physical implementation of the design. The design of the banking login system will show a visual representation of how I believe it should look along with a system flow guiding users through the application.

## Design

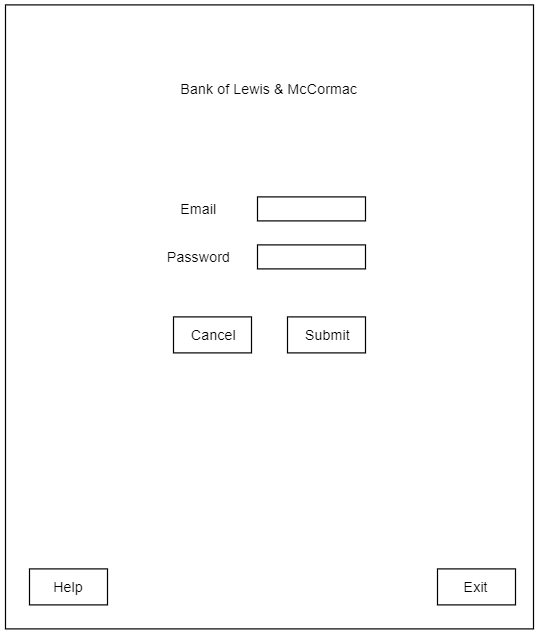
### Main Window

The main window is the foundation of the program. Using a panel, the size of the usable area of the main window I will be able to display the other user controls. On its own the panel would be blank and would not show any information.



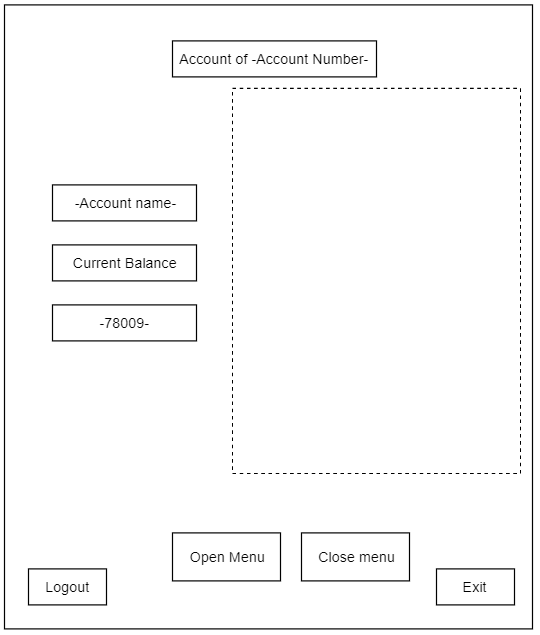
### Login screen

The login screen is the front screen of the program it is the first screen the user will interact with. It has a title naming the program that being “Bank of Lewis & McCormac”. Then there will be an email and password field allowing the user to enter in their user name and password. Below these fields will be a cancel option which will clear the fields and a submit button which will check the validity of the email and password and if valid will bring the user into the next screen. To the bottom left of the screen is the help button which will be elaborated upon later. To the bottom left is the exit button which will close the program.



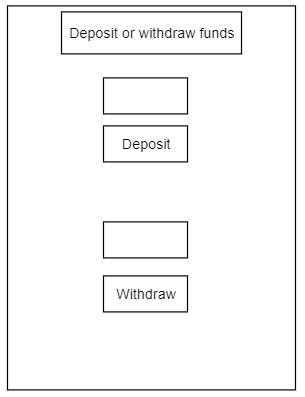
### Account Screen

The account screen will show the title “Account of” followed by the account number. Then to the bottom left will be the account name of the account holder being in First Name, Second Name format. Below this will be the “current balance” header followed by the current balance the account has. To the right will be a panel which will show the transaction menu which will be displayed later. Below the area for the transaction menu is the controls for opening and closing the transaction menu. To the bottom left of the page is a log out button which will save any changes to the user account and will then return the user to the front screen. To the bottom right is an exit button which will save the program and close the application.



### Transaction menu

The transaction menu will be displayed when the user clicked open menu on the account screen. It will give the user the ability to deposit money into their account or they will be able to withdraw money provided they have sufficient funds in their account.



# Task 2.5

Task 2.5 is the implementation of the above design

End of task 2

# Task 3

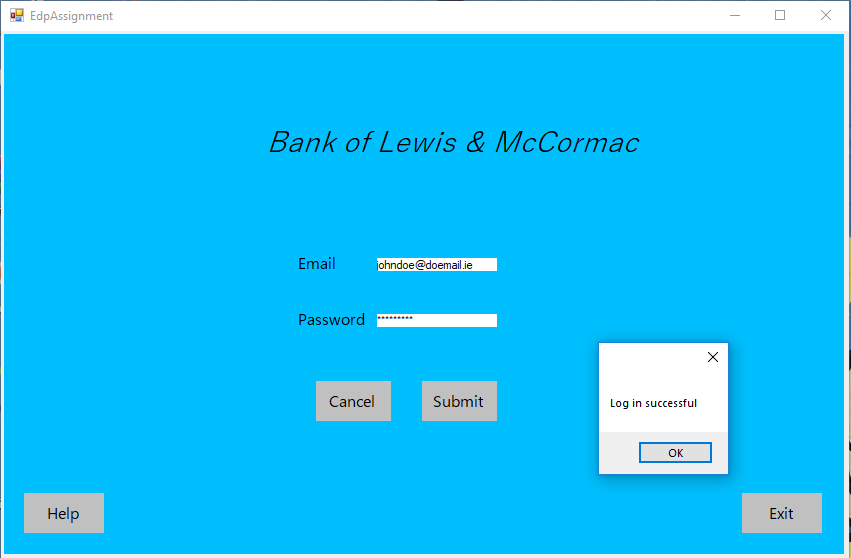
## Sub-introduction

Task 3 is the testing of the banking application. I will use end users testing to develop a report of shortcoming of the application along with any possible implementations to make the program more robust and resistant to attacks and mitigate damage from bugs.

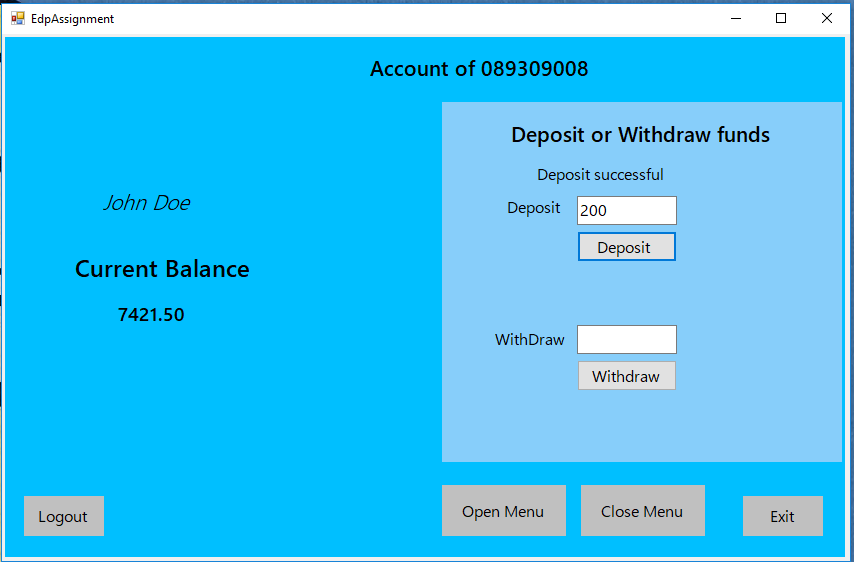
## Test 1

|  |  |
| --- | --- |
| Test Email: [JohnDoe@doemail.ie](mailto:JohnDoe@doemail.ie) | Test Password: Passw0rd1 |
|  |  |
| Test: | Success/Fail: |
| User can login with selected data | Success |
| User can deposit $200 | Success |
| User can withdraw $300 | Success |
| User can log out | Success |
| User can exit program | Success |

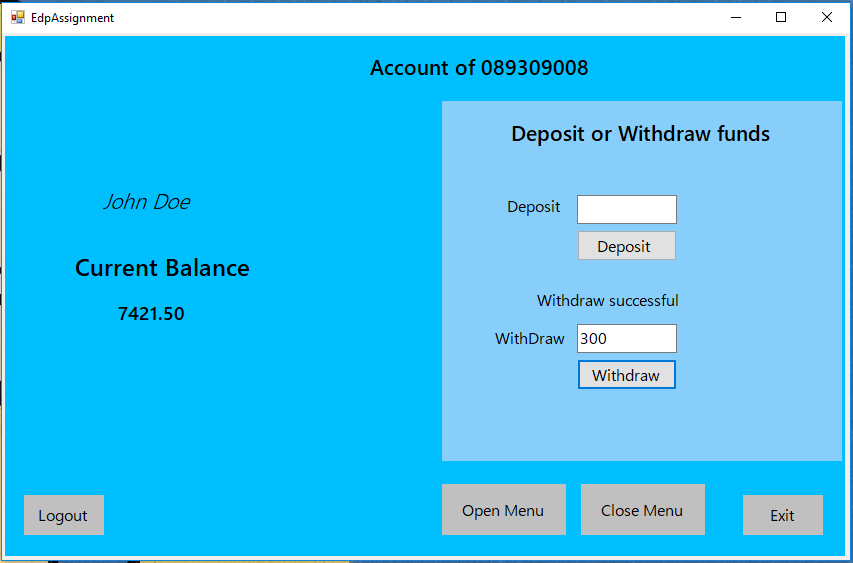
## Test 1 results



When logging in with correct details the user can login successfully.



When the user attempts to deposit 200 they can successfully deposit the amount

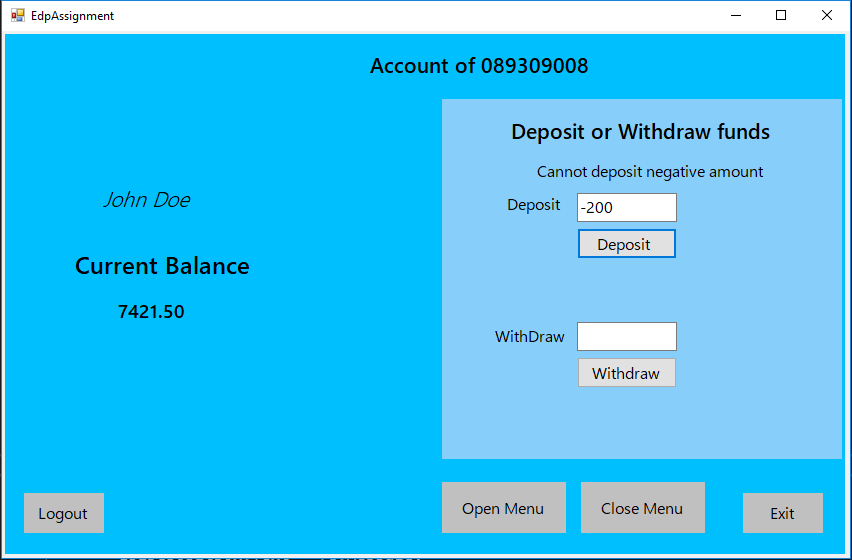


When the user attempts to withdraw 300 the test is successful.

## Test 2

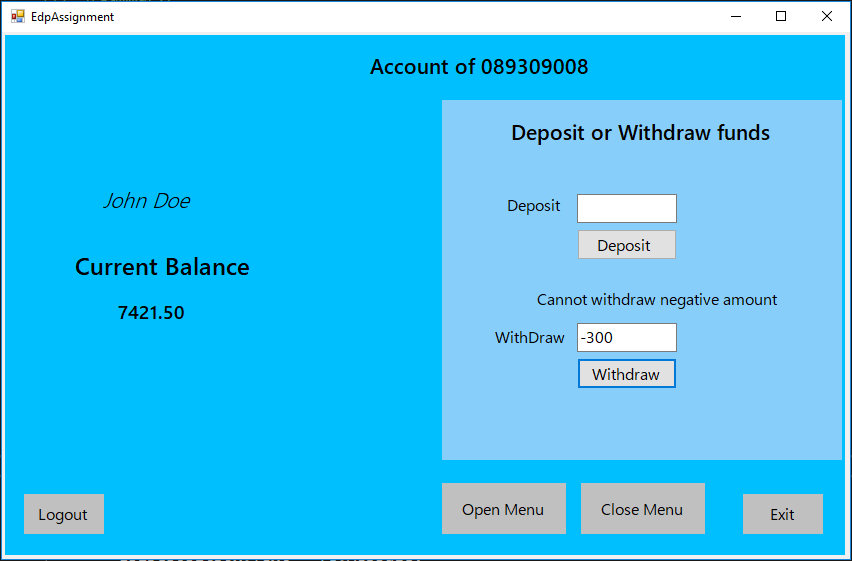
|  |  |
| --- | --- |
| Test Email: [JohnDoe@doemail.ie](mailto:JohnDoe@doemail.ie) | Test Password: Passw0rd1 |
|  |  |
| Test: | Success/Fail: |
| User can login with selected data | Success |
| User can deposit $-200 | Success |
| User can withdraw $-300 | Success |
| User can log out | Success |
| User can exit program | Success |
|  |  |

## Test 2 results



When the user attempts to withdraw -200 they are stopped from doing this and are given a warning correcting them saying that -200 cannot be deposited.

## 

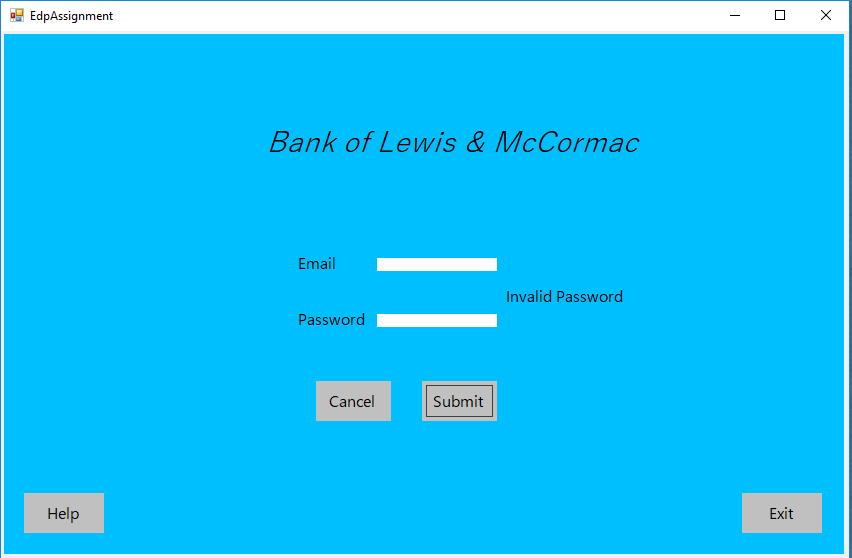


When the user tries to withdraw -300 they are told they cannot withdraw negative amounts and they are stopped from proceeding before doing so.

## Test 3

|  |  |
| --- | --- |
| Test Email: [JohnDoe@doemail.ie](mailto:JohnDoe@doemail.ie) | Test Password: Password1 |
|  |  |
| Test: | Success/Fail: |
| User can login with selected data | Success |

## Test 3 results

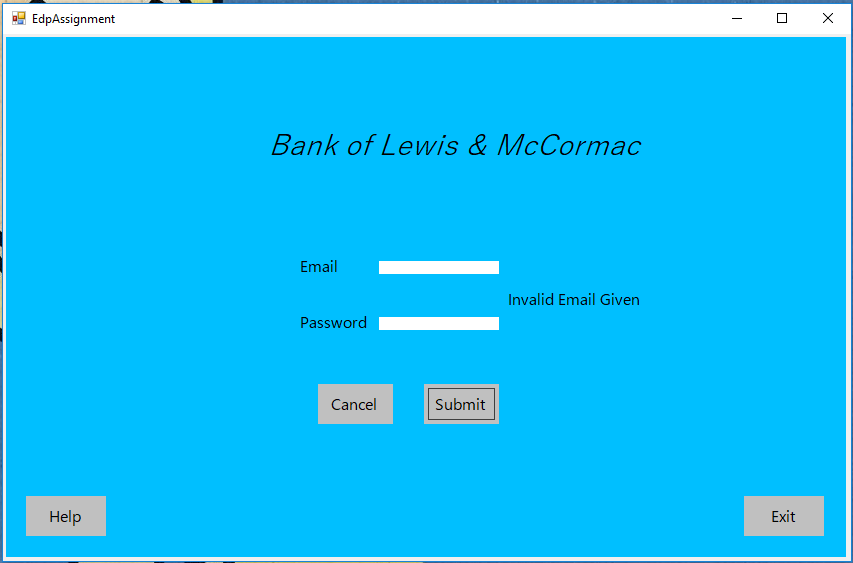


When the user tries to login with a correct email but an incorrect password they are stopped from logging in with incorrect details.

## Test 4

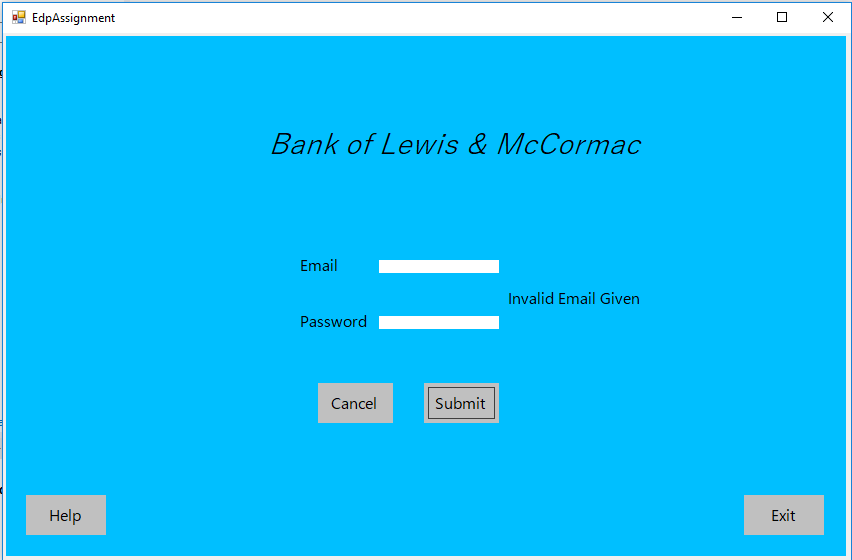
|  |  |
| --- | --- |
| Test Email: [JohnDoe@geemail.ie](mailto:JohnDoe@geemail.ie) | Test Password: Passw0rd1 |
|  |  |
| Test: | Success/Fail: |
| User can login with selected data | Success |

## Test 4 results



When the user tries to login with an incorrect email but correct password they are stopped from logging in until they give the correct details.

## Security issues





The major security risk is that there is no sanitation of queries in the log in phase this means that the app is susceptible to SQL injection attacks this means that a user who is aware of SQL injections can use this for a variety of reasons from modifying accounts to dropping whole tables as can be seen above.