Blue Minev

22808922

CI401 ASSESMENT REPORT

Contents

[INTRODUCTION 1](#_Toc135299819)

[DESIGN AND DEVELOPMENT 1](#_Toc135299820)

[CRITICAL REVIEW 1](#_Toc135299821)

[CONCLUSION 1](#_Toc135299822)

[ESTIMATED GRADE 1](#_Toc135299823)

[REFRENCES 1](#_Toc135299824)

Title page

• Introduction (~0.5 page): the idea for the improvements.

• Design and Development (~4 pages): highlight any code additions and how you developed them. Include developed code screenshots that are clearly readable and clearly describe the purpose of the additions.

• Critical Review (~1 page): identify three reasons why the code additions are good. Further identify three reasons where the code couldbe improved.

• Conclusion (~0.5 page): what are the main take-away messages – what are the key concepts that you learned during the development.

• Estimated grade: your self-assessed grade based on assessment criteria below. Example: Development: A, Report: B

• References: include references to existing articles (whether books, research papers, online blogs or educational videos). Clearly identifyand reference any 3rd party tutorials / assets / code used and provide their source and license. Failure to do this is likely to lead toserious consequences, including a 0 mark

# INTRODUCTION

For my Programming Project I chose to expand on the ATM code. After reviewing this code in tutorials and fixing the issue. I began to consider what changes I wanted to make.

The first adjustment I chose to do was make changes to the CSS of the ATM. I decided to do this to enhance the user experience and make the UI appear more modern and sleek. This would be done by changing aspects of the buttons and adjusting colours and fonts. I also wanted to ensure that my UI was as accessible to those with disabilities as I could make it.

An addition I made was to add a button to change the pin. Being able to change the pin is a common feature of ATMs so I chose to include this.

A feature I added was a statement button. The goal with this button is to print the last 5 transactions completed. Most ATMs and banking software have this feature as it is good practice to include it. Due to these reasons, I chose to add this change.

The final major change I decided to include was the ability to have an overdraft on an account. This was to add extra functionality to the app and widen the scope of options available to the user.

# DESIGN AND DEVELOPMENT

## design

One major point of the design for this project is the look of the ATM. Before deciding anything I decided to interview some hypothetical end users to get some requirements for my end software. During this interview they stated that they would prefer the software to have the look of a regular ATM. Because of this I chose to keep the basic layout of the ATM. This was a decision was also made due to the fact that the IDE I chose to use did not have an inbuilt scene builder, meaning that making major changes ot the layout would be incredibly difficult.

After this interview I decided on a colour scheme for the ATM. I decided to go for muted grays and a muted blue with large white writing for the button. During one interview with an end user I discovered that the colour scheme I had proposed was good for those with a high sensitivity to colour and contrast

IDE issues

Java fx css issues

User interviews with those who use ATMs to gain valulble information on preferences and features.

## testing

The majority of the testing I did was white box testing, where I tested every function at the end of my development phase, along with as I went along, testing the current function I was working on, including as many edge cases as I could.

During this testing I discovered that during login if the user inputted the wrong account number or password, they were not being sent back to the initial state as they should, they were stuck on the ‘enter your password’ screen and no button input was changing anything, except for the log out button.

I started by adding more debug statements to the parts of the code that I thought were the issue. Through this, I found that the account number was staying when I believed it should have been reset. Due to this, I tried many ways to set this value to 0 but none were working. I think checked the command line and found that there was an error being sent.

This error was a ‘NullPointerException’ after reading te error and trying ot understand what was causing this, I realised that I was looking in the wrong file. Originally I was looking in the Model file at the processEnter() function. In this function, it calls bank.login() and this was where the issue was.

I returned to my starting strategy of adding debug statements to the function to try and understand what was going wrong. I decided to put a debug statement after the for loop to see if this loop was executing correctly. During this I realsied that the for loop was exectuting 3 times when there was only 2 bank accounts. This was due to the fact that it was looping through the value of maxAccounts. However, once it reached the third iteration, the values in b.accNumber and b.accPassword were null, this caused the code to throw the ‘NullPointerException’ error.

I chose to solve this issue through exception handling. Using a try{} catch{} statement I was able to allow the code to run how it is expected to.

After white box testing, I did black box testing with the end user I interviewed. I gave him a list of instructions to complete and monitored his interaction. During this, we discorvred a bug with the new pin button where if you pressed it without inputting a number first, it would set the pin to 0.

My first approach to fixing this was to add extra logic to the if statement in changePin(), located in the bank file. A screenshot of a computer code

Description automatically generated with low confidence

This, however, caused the function to return -1, though the pin is not actually changed to -1 I chose to move the logic to the proccessNewPin() function in the Model file.

A screenshot of a computer

Description automatically generated with medium confidence

I did this by adding an if else statement that checks that the number is not 0. If the number is 0, it changes the display to remind the user to enter a valid pin before pressing the button.

I then conducted white box testing on this specific button again to ensure that there were no more issues.

# CRITICAL REVIEW

When reviewing my code, I chose to review it in the style that would be typical in a peer code review. To do this I gave myself a checklist as to what ‘good code’ would be. That checklist included:

* Is the code formatted correctly
  + Is the code properly aligned
  + Are there any unneeded whitespaces
  + Is it easy to see where functions and loops start and end
* Is the code maintainable?
  + Is the code readable and easily understood
  + If the code is difficult to understand are there comments to help others understand the code
  + Is the code easy to test
* Is the code usable?
  + Is the output of the code easy to understand for the general population?

After this review I also looked at the code in the eyes of the developer and considered what I belived was good about my code.

I believe that I have added a good amount of functionality to the app, such as the new pin button and overdrafts. This added functionality means that in a real world scenario would not have to leave the software to do common things like getting a statement. Thus increasing the time spent on the application.

I also believe that my code and software has a high awareness of users and accessibility. This is due to my interviews with end users and my research into inclusivity in UI design. By making my software more accessible I have increased the scope of users as well as improved the experience of the original user demographic.

Due ot my end user intervews I believe tha I also had a good understanding of what features are common in ATMs widley used. I believe that this increased the quality of the end software as I was able to decide which features would be the most used and organise which features I would work on.

I also considered what I think I could have improved about my code.

This included expanding the overdraft feature, I would have liked to add a button to make changes to the account such as adding an overdraft or editing the limit however, due to my issues in the beginning of the project I had to sacrifice this to make a fully functional application.

Another feature I would have liked to expore more was the CSS and the model view. I would have investigated more changes I could have made with the CSS to make a more enjoyable experience for the user as well as increasing the accessiability, such as adding a large font feature or the ability to change the font to a more dislexyia friendly design.

A feature I would have added if I had time was a receipt function. This function would have printed a log of all transactions to a .txt file for the user to keep.

# CONCLUSION

Talk about issues again and what ive learnt – fmailiarise myself with the ide

How to find information about plugins that are not widely documented

Greatly imporvoved my problem solving skills as well as my general understanding of java and javafx

# ESTIMATED GRADE

Report: a/b depending

Development: b

# REFERENCES