**ASSIGNMENT COVER SHEET**

|  |  |  |
| --- | --- | --- |
| **Student’s name** | Hirushan | Thavaseelan |
| **Module name** | Software Engineering | |
| **Title of assignment** | Customer database management Python Application | |
| **Complete Word Count in my assignment** | 1362 | |
| **Date submitted** | 03/04/2023 | |

All work must be submitted by the due date. If an extension of time to submit work is required, a [Mitigating Circumstances Extension Form](https://canvas.qa.com/courses/1041) must be submitted.

X

**Has an extension been approved? Yes No If yes, please give the new submission date ….…/..…./…….**

|  |
| --- |
| IMPORTANT: THIS STATEMENT MUST BE READ & SIGNED  **Academic Integrity Statement**  Academic integrity and honesty are fundamental to the academic work you produce at the University of Roehampton. You are expected to complete coursework that is your own and which is referenced appropriately. The university has in place measures to detect academic dishonesty in all its forms. If you are found to be cheating or attempting to gain an unfair advantage over other students in any way, this is considered academic misconduct and you will be penalised accordingly.   ​  **I declare that the work I am submitting is my work, is properly referenced and has not been submitted elsewhere.** |
| **Student Signature (Full Name): Hirushan Thavaseelan**  **Date: 03/04/2023** |

**Task 1:** Design Doc

This app will serve as a comprehensive tool to manage customer information within a database, streamlining the workflow for these associates while maintaining a high level of data organisation and accuracy. The application is set to incorporate a variety of features that facilitate the creation, reading, updating, and deleting of records, providing a user-friendly platform that addresses the needs of the associates throughout the entire customer management process.

One of the critical aspects of this app is the incorporation of data validation, which ensures that any information entered into the system is accurate and reliable. This feature is essential in maintaining the integrity of the customer database, minimising the risk of errors or inconsistencies that may arise as a result of manual data input. By implementing data validation, the app will help maintain the overall quality of the customer information, ultimately leading to better decision-making and improved sales performance.

Another key element of the application's design is the implementation of a two-tiered login system, which establishes different access levels for admins and sales associates. This distinction is essential in maintaining the app's security, ensuring that only authorised personnel have access to specific functions. While admins will have full access to all features, including creating, reading, updating, and deleting records, sales associates will have the same capabilities, except for the ability to delete records. This restriction is in place to prevent accidental or unauthorised deletion of valuable customer information, preserving the integrity of the database.

In addition to the features mentioned above, the app will also include an error log functionality that automatically records any errors encountered during its operation. This feature will prove invaluable in identifying and addressing any technical issues that may arise, facilitating a quick resolution and ensuring the smooth functioning of the application. By documenting these errors, developers can gain insights into the performance of the app and make necessary improvements, optimising its efficiency and usability. This structure can be seen in Figure 1 below. The flow of actions can be seen in Figure 2 below.

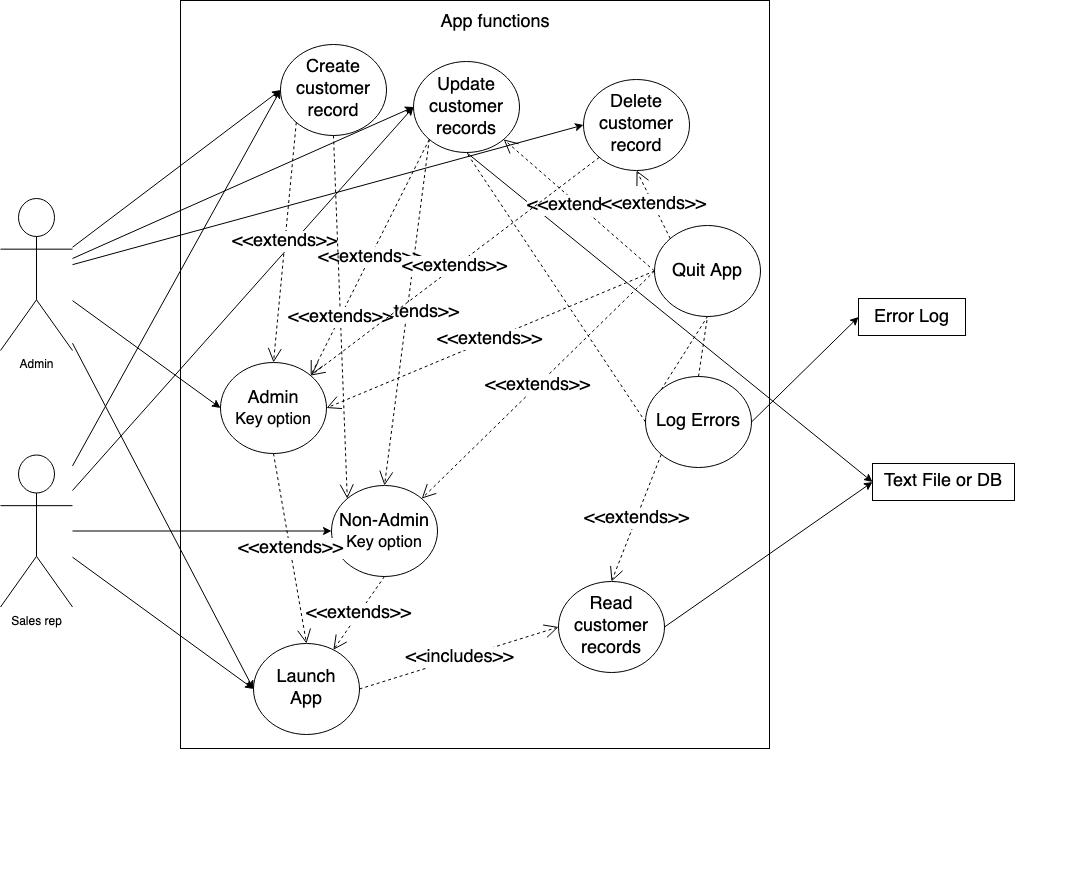


Figure Use Case Diagram

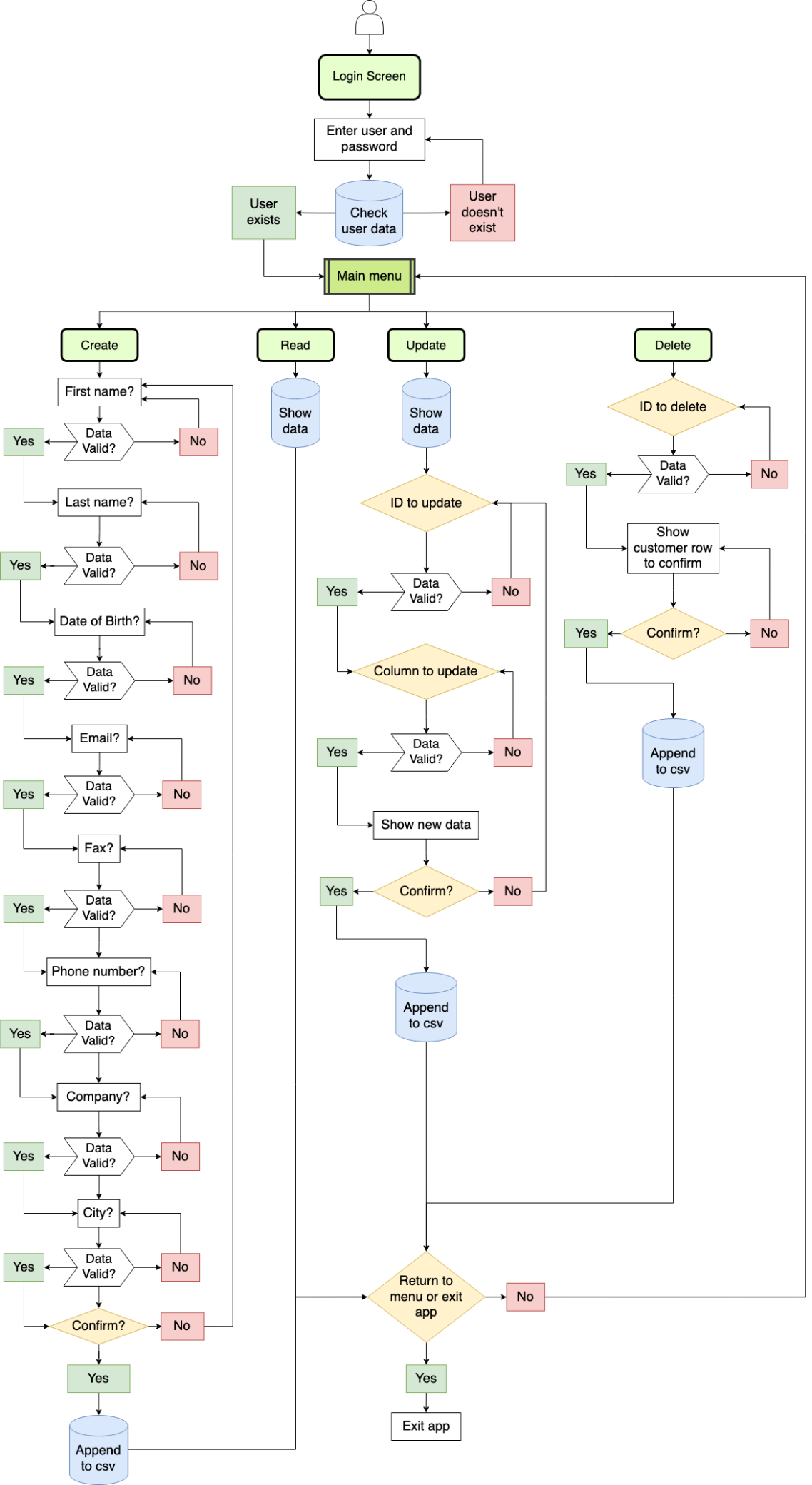


Figure Activity Diagram

The data store designed for the app, displayed below in Table 1, will comprise a comprehensive list of expected data fields, ensuring that all necessary customer information is captured and managed effectively. To maintain the integrity and accuracy of the data, the app will implement validation against whitespaces and empty fields for all these data fields. This decision to limit validation to the basic whitespace and empty field checks stems from the understanding that company names can, in practice, include a wide variety of characters, and imposing more stringent validation rules may inadvertently exclude valid entries.

In addition to this, the app's validation process for first and last names will focus primarily on checking for numbers within the names. This decision was made in recognition of the fact that different cultures have diverse naming conventions, and it is not uncommon for some names to include special characters such as hyphens. By adopting this approach, the app aims to be more inclusive in its data validation process for names, accommodating the unique structures and characters that may be present in various cultural contexts. This inclusive approach ensures that the app caters to a diverse range of users and customers, promoting a more universally accessible platform for managing customer information. As part of validating the data as it is being entered into the database using create/update functions; name/company/city fields will also need to have their first letters automatically capitalised, regardless of the user capitalising.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Field Name** | **Description** | **Data Store Type** | **Python Type** | **Validation Details** |
| Customer ID | ID of Customer | int | int | Must be unique number |
| First Name | First Name of Customer | text | text | Must not contain numbers |
| Last name | Last name of Customer | text | text | Must not contain numbers |
| Date of Birth | Date of Birth of Customer | text | text | Must be dd/mm/yyyy format |
| Email | Email of Customer | text | text | Must contain @ and .xx |
| Fax | Fax of Customer | int | int | Must not contain letters |
| Phone Number | Phone Number of Customer | int | int | Must not contain letters |
| Company | Company of Customer | text | text | Must not be blank or have whitespace |
| City | City of Customer | text | text | Must not contain number |

Table Application Data Store

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| **Test ID** | **Test Purpose** | **Test Steps**  **Task 3:** Test Plan | **Test Data** | **Test Outcome** | **Pass/Fail** | **Test Evidence** | **Required Actions** |
| **T01** | Admin has access to all functions | Run all functions | No data being used | No permission restrictions | Pass | Appendix A, B, C, D | None |
| **T02** | User Bob has access to all functions except the delete function | Run all functions | No data being used | Permission restriction for only delete function | Pass | Appendix E, F, G, H | None |
| **T03** | User Bob trying to access delete function is logged in the CustomerDataErrorLog | Run delete function as Bob | No data being used | Event is logged correctly | Pass | Appendix I | None |
| **T04** | The user should be able to return to the Main Menu no matter where they are in the application | Run through all screens and return to menu | “<” to return to menu | Correctly able to return to menu no matter where in the application | Pass | Not sure how to provide evidence | None |
| **T05** | If the data in the csv is found to be corrupt, an error message should be written to a log file | Read in currupted csv and check log | Corrupted csv data being read in | Not all corrupted data has been logged, the fax number that is just a space should have been logged but hasn’t. | Fail | Appendix J | The read database function and the is\_currupted function need to be looked at, that is potentially where the issue is coming from |
| **T06** | Duplicated customer IDs should be skipped over, only the first instance of that ID should be displayed | Read in currupted csv and run read function | Corrupted csv data being read in | Duplicate Customer correctly skipped over in display and error shown to user | Pass | Appendix K | None |
| **T07** | Confirm with the user before quitting the application | Enter “>” to quit application | “>” to quit app | There is no confirmation screen before app is quit | Fail | Appendix L | A function needs to be put in place to confirm with the user before quitting the app |
| **T08** | Confirm with the user before deleting a record | Attempt to delete a record | Customer ID for deleting | Correctly confirming with user before deleting a record | Pass | Appendix M | None |
| **T09** | Verify create\_row() function handles invalid first name | Run create\_row() function. | Enter a name with numbers | Correctly catching error, displaying error to user, and logging it | Pass | Appendix N, O | None |
| **T10** | Verify create\_row() function handles invalid last name | Run create\_row() function. | Enter a name with numbers | Correctly catching error, displaying error to user, and logging it | Pass | Appendix P, Q | None |
| **T11** | Verify create\_row() function handles invalid Date of birth | Run create\_row() function. | Enter a date of birth with letters | Correctly catching error, displaying error to user, and logging it | Pass | Appendix R, S | None |
| T12 | Verify create\_row() function handles invalid email address | Run create\_row() function. | .Enter an invalid email address (missing @ symbol) | Correctly catching error, displaying error to user, and logging it | Pass | Appendix T, U | Pass |
| T13 | Verify delete\_row() function handles invalid customer ID | Run delete\_row() function. | Enter an invalid customer ID (non-existent in the database). | Correctly handles invalid Customer ID, displays the error and logs to error log. | Pass | Appendix V, W | None |
| T14 | Verify delete\_row() function successfully deletes a customer record | Run delete\_row() function. | Enter the ID of a customer record in the database. | Verify that the function deletes the customer record and updates the database accordingly. | Pass | Appendix X, Y | None |
| T15 | Verify animationscreen() function displays animation | Run animationscreen() function. | No data being used | Animation screen successfully displays animation for a few seconds for both “Loading” and “Saving” | Pass | Appendix S, AA | None |
| T16 | Verify login() function handles incorrect password | Run login() function.  Enter a valid username and an incorrect password. | Valid username and an incorrect password. | Login function correctly prevents access, displays error to user and logs the error | Pass | Appendix BB, CC | None |
| T17 | Verify login() function handles incorrect username | Run login() function.  Enter an invalid username and a valid password. | Valid password and an incorrect username. | Verify that the function prompts the user to enter a valid username. | Pass | Appendix DD, EE | None |
| T18 | Verify login() function successfully logs in user | Run login() function.  Enter a valid username and password. | Valid username and password. | Verify that the function logs the user in and displays the main menu. | Pass | Appendix FF | None |
| T19 | Verify that the update\_customer() function updates a customer record in the database | Run update\_customer() function. Enter the ID of a customer record in the database. Enter the updated customer information. Check that the customer record is updated in the database. | Customer record in the database, updated information | The customer record is updated in the database. | Pass | Appendix GG, HH | None |
| T20 | Verify that the update\_customer() function handles invalid input | Run update\_customer() function. Enter an invalid customer ID. Verify that the function prompts the user to enter a valid customer ID. | Invalid customer ID | Correctly displays error message to user and logs to error log. | Pass | Appendix II, JJ | None |
| T21 | Verify that the update\_customer() function handles invalid email addresses | Run update\_customer() function. Enter the ID of a customer record in the database. Enter an invalid email address. | Customer record in the database, invalid email address, without @ or .com | Correctly displays error message to user and logs to error log. | Pass | Appendix KK, LL | None |
| T22 | Verify that the update\_customer() function handles invalid phone numbers | Run update\_customer() function. Enter the ID of a customer record in the database. Enter an invalid phone number. | Invalid phone number with letters | Invalid number is incorrectly accepted and proceeds to confirmation screen. | Fail | Appendix MM | Data validation for this needs to be checked, the issue is potentially with the regex search that maybe allows letters. |
| T23 | Verify that the update\_customer() function handles empty fields | Run update\_customer() function. Enter the ID of a customer record in the database. Choose all options and check if empty fields are allowed to progress. | Empty fields | Verify that the function prompts the user to enter valid values for all fields. | Pass | Appendix NN, OO, PP, QQ, RR, SS, TT, UU | None |

**Task 4:** Review and Reflection

One significant strength of the application lies in its emphasis on user experience and design. By prioritizing these aspects, the application boasts a visually appealing interface that is both engaging and easy to navigate. Placing the user at the forefront of the development process has allowed for the creation of a seamless and enjoyable experience tailored to their needs.

The combination of an aesthetically pleasing design and intuitive navigation allows users to quickly familiarize themselves with the application and its features. Ease of use is crucial for user retention and satisfaction, as it encourages users to continue using the application and explore its various functionalities. By focusing on user experience, this application succeeds in providing an engaging and accessible platform for its target audience. This user-centric approach is essential in today's competitive market, where applications must not only meet functional requirements but also captivate and retain users through exceptional design and usability.

However, no application is without areas for improvement. Future projects can benefit from addressing certain shortcomings, further enhancing the overall user experience and the application's reliability. For instance, next time I would include more data handling to better manage user inputs and application processes. By implementing more robust data handling mechanisms, the application's efficiency and reliability would be significantly improved. In turn, this would contribute to a more satisfying user experience, as users would encounter fewer issues related to data management.

Another area for improvement is the error logging system. Currently, the error messages may not provide sufficient information to accurately identify the source of the issue. By incorporating more specific error log messages, I can more effectively pinpoint the root cause of any problems, which would lead to faster troubleshooting and resolution. A more efficient error logging system would contribute to the application's overall stability and performance, ensuring a smoother experience for users.

In addition to refining the error logging system, the language used in error messages displayed to users could also be improved. Clearer explanations and more precise wording can provide users with a better understanding of the nature of the issue and where the error has originated. This transparency will empower users to take appropriate action to resolve the problem, or at the very least, equip them with relevant information when seeking assistance. Ultimately, improving the language used in error messages will enhance the user experience by facilitating better communication and promoting a sense of trust in the application.

Moreover, the lessons learned from this project can be applied to future endeavours, ensuring a more streamlined and efficient development process. For example, focusing on establishing a solid foundation in code, being more mindful of time allocation, and striving to maintain a balance between aesthetics and functionality are all valuable insights that can be carried forward. By applying these lessons, I can create better applications that not only meet the needs and expectations of users but also reflect a higher level of quality and professionalism in the development process.

In conclusion, the application's emphasis on user experience and design is a notable strength, but there is always room for improvement. By incorporating more data handling, refining error log messages, and improving the language used in error messages displayed to users, future projects can excel in both aesthetics and functionality. These enhancements will contribute to a more robust and user-friendly application, providing an even better experience for users.

The insights gained from this project will guide my future work in application development. By learning from the challenges faced and applying these lessons to subsequent projects, I can continue to grow as a developer and create well-rounded applications that effectively balance visual appeal with efficient and reliable code. As I continue on this journey, I am confident that I will be able to deliver increasingly sophisticated and satisfying user experiences, ensuring the ongoing success of my applications and the satisfaction of those who use them.

Appendix Admin Read function.

Graphical user interface, text

Description automatically generated

Appendix Admin Edit function.

Graphical user interface, text

Description automatically generated

Appendix Admin Delete function.

Graphical user interface, text

Description automatically generated

Appendix Admin Create function.

Graphical user interface, text, application

Description automatically generated

Appendix Sales Read function.

Graphical user interface, text

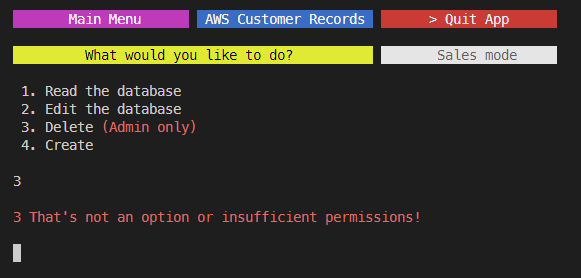
Description automatically generated

Appendix Sales Edit function

Graphical user interface, text

Description automatically generated

Appendix Sales Delete function.



Appendix Sales Create function.

Graphical user interface, text, application

Description automatically generated

Appendix Sales delete function access error log.

Graphical user interface, text, application

Description automatically generated

Appendix Corrupted csv Read in Error log.

Graphical user interface, text, application

Description automatically generated

Appendix Duplicate customer ID error handling

Graphical user interface, text

Description automatically generated

Appendix Quitting app

Graphical user interface, text

Description automatically generated

Appendix Confirmation before deleting a record

A screenshot of a computer

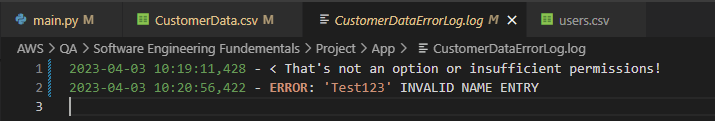
Description automatically generated

Appendix Create function, invalid first name.

Text

Description automatically generated

Appendix Create function, invalid first name log.



Appendix Create function, invalid last name.

Text

Description automatically generated

Appendix Create function, invalid last name log.

Graphical user interface, text

Description automatically generated

Appendix Create function, invalid date of birth.

Graphical user interface, text, website

Description automatically generated

Appendix Create function, invalid date of birth log.

Graphical user interface, text, application

Description automatically generated

Appendix Create function, invalid email.

Graphical user interface, text, website

Description automatically generated

Appendix Create function, invalid email log.

Graphical user interface, text

Description automatically generated

Appendix Delete function, invalid customer number.

Graphical user interface, text

Description automatically generated

Appendix Appendix V Delete function, invalid customer number log.

A screenshot of a computer

Description automatically generated with medium confidence

Appendix Delete function, delete test 1

Graphical user interface

Description automatically generated

Appendix Delete function, delete test 2.

Graphical user interface, text

Description automatically generated

Appendix Animation screen test 1.

Graphical user interface, text

Description automatically generated

Appendix Animation screen test 2.

Text

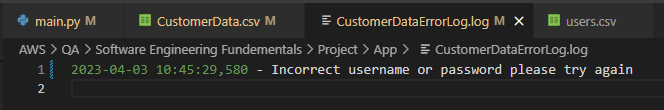
Description automatically generated

Appendix Login function, wrong password test.

Text

Description automatically generated

Appendix Login function, wrong password test log.



Appendix Login function, wrong user test.

A picture containing text

Description automatically generated

Appendix Login function, wrong user test log.

Graphical user interface, text

Description automatically generated

Appendix User log in test.

Text

Description automatically generated

Appendix Update record test 1.

A screenshot of a computer

Description automatically generated with medium confidence

Appendix Update record test 2.

Graphical user interface, text

Description automatically generated

Appendix Update function, invalid Customer ID.

Graphical user interface, text

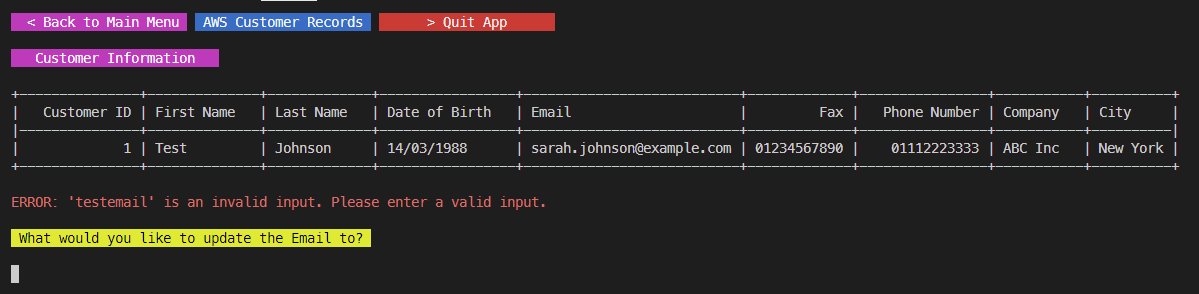
Description automatically generated

Appendix Update function, invalid Customer ID log.

Text

Description automatically generated

Appendix Update function, invalid email.



Appendix Update function, invalid email log.

A screenshot of a computer

Description automatically generated

Appendix Update function, invalid phone number test.

A screenshot of a computer

Description automatically generated

Appendix Update function, empty First Name input.

Text

Description automatically generated

Appendix Update function, empty Last Name input.

Text

Description automatically generated

Appendix Update function, empty Date of Birth input.

A screenshot of a computer

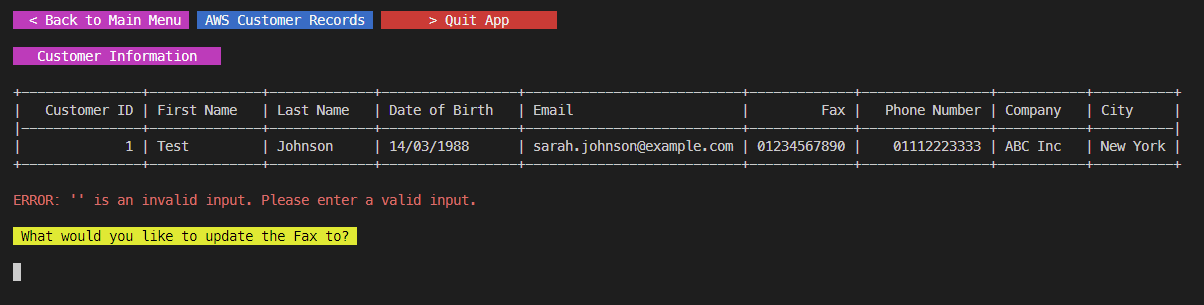
Description automatically generated with medium confidence

Appendix Update function, empty Email input.

A screenshot of a computer

Description automatically generated with medium confidence

Appendix Update function, empty Fax input.



Appendix Update function, empty Phone Number input.

A screenshot of a computer

Description automatically generated with medium confidence

Appendix Update function, empty Company input.

A screenshot of a computer

Description automatically generated with medium confidence

Appendix Update function, empty City input.

Text

Description automatically generated