|  |  |
| --- | --- |
|  | *Kingdom of Saudi Arabia*  *Ministry of Education*  *King Faisal University*  ***College of Computer Sciences & Information Technology*** |

# **[** ATM Management System **]**

*A project submitted as a requirement for the course “Data Structure & Algorithms-CS-211”*

By:

**Group 8 - Section 70**

Salma Nahedh Aljabr – 2224\*\*\*\*\*

Zahra Ali Albarrak – 2224\*\*\*\*\*

Nouf Yousef Aljaffar –2224\*\*\*\*\*

Sara Khalid Alhussein –2224\*\*\*\*\*

**Course Instructors**

Muneerah Alhajri

Submission Date: 13-Dec-23

**Project Marking Scheme**

|  |  |  |
| --- | --- | --- |
| **Criteria** | | **Mark contribution** |
| Project Report | Technical contents (comprehensiveness, clarity and accuracy) | 1/ |
| English language (grammar and spelling) | .5/ |
| Style and formatting | .5/ |
| Presentation | Presentation Skills | 2/ |
| Project Code | Solution Implementation | 4/ |
| Questions and answers | 2/ |
| Total | | 10/ |

**Table of Content**

[Introduction 4](#_Toc152921864)

[Problem statement 4](#_Toc152921865)

[Methodology 5](#_Toc152921866)

Results and discussions 17

[Conclusion 22](#_Toc152921868)

# **Introduction**

In the ever-changing landscape of banking, where innovation takes center stage, this report tackles a common challenge faced by account holders in traditional banking systems. The inherent limitations and inefficiencies of these systems underscore the necessity for an advanced ATM solution. Addressing issues of time-consuming processes and restricted access, our proposed system presents a streamlined approach to banking operations. This sophisticated ATM system facilitates smooth cash deposits, fund withdrawals, balance inquiries, bill management, and payments through an intuitive interface. By integrating modern technologies and user-friendly features, our solution not only optimizes financial transactions but also redefines the user experience, offering account holders a more efficient and convenient way to manage their finances.

# **Problem statement**

The issue we're looking at is about the problems people face with regular banks. These banks usually have slow and complicated processes that make things inconvenient for users. There's no easy way to handle different money tasks, like putting money in, taking money out, and paying bills. The goal is to solve these problems and make things better for users by introducing a new and improved ATM system. This statement focuses on the need for a more efficient and user-friendly solution in the world of banking.

# **Methodology**

* 1. Data structures used for the solution and why?

The backbone of our solution lies in the effective use of data structures. A singly linked list structure is employed to manage and organize user bills efficiently. each node points to the next node in the sequence, forming a unidirectional chain of elements. This choice of data structure allows for efficient bill additions, removals, and sorting based on due dates, contributing to a streamlined and effective solution.

* 1. Executable Program Code

1. **#include <iostream>**
2. **#include <string>**
3. **#include <ctime> // display the current time**
4. **#include <conio.h> // for \_getch() function**
5. **using namespace std;**
6. **int accountNumber = 1532;**
7. **string accountHolders = "Salma Aljabr";**
8. **string accountHoldersAddress = "Al ahssa";**
9. **double accountBalance = 20000.00;**
10. **string branch = "KSA";**
11. **double limit = 9000.00;**
12. **// Node structure for the linked list**
13. **struct BillNode {**
14. **string name;**
15. **int day, month, year;**
16. **double price;**
17. **BillNode\* next;**
18. **};**
19. **BillNode\* head = NULL; // Head of the linked list**
20. **// Function to insert a bill at the beginning of the linked list**
21. **void insertNode(BillNode\*& head, BillNode\* newNode) {**
22. **newNode->next = head;**
23. **head = newNode;**
24. **}**
25. **// Function to remove a bill from a specified position in the linked list**
26. **void removeFromPosition(int position) {**
27. **if (head == NULL) {**
28. **cout << "Linked list is empty! Cannot remove more bills.\n";**
29. **return;**
30. **}**
31. **if (position == 0) {**
32. **BillNode\* temp = head;**
33. **head = head->next;**
34. **delete temp;**
35. **} else {**
36. **BillNode\* current = head;**
37. **BillNode\* previous = NULL;**
38. **for (int i = 0; i < position && current != NULL; ++i) {**
39. **previous = current;**
40. **current = current->next;**
41. **}**
42. **if (current != NULL) {**
43. **previous->next = current->next;**
44. **delete current;**
45. **} else {**
46. **cout << "Invalid position. Cannot remove bill from position " << position << ".\n";**
47. **}**
48. **}**
49. **}**
50. **void details() {**
51. **system("cls");**
52. **cout << "\n\xdb\xdb\xdb\xdb\xdb\xdb\xdb\xdb\xdb\xdb\xdb\xdb\xdb\xdb\xdb\xdb\xdb\xdb\xdb\xdb\xdb\xdb\xdb\xdb ACCOUNT DETAILS SYSTEM \xdb\xdb\xdb\xdb\xdb\xdb\xdb\xdb\xdb\xdb\xdb\xdb\xdb\xdb\xdb\xdb\xdb\xdb\xdb\xdb\xdb\xdb\xdb\xdb\xdb\n";**
53. **cout << "\n\nThe Name of the Account Holder is: " << accountHolders << "\n\n";**
54. **cout << "\tThe Account Holder's address is: " << accountHoldersAddress << "\n\n";**
55. **cout << "\tThe Branch location is: " << branch << "\n\n";**
56. **cout << "\tAccount number: " << accountNumber << "\n\n";**
57. **cout << "\tAccount Balance: $" << accountBalance << endl;**
58. **}**
59. **void detail(){**
60. **details();**
61. **cout << "\n\xdb\xdb\xdb\xdb\xdb\xdb\xdb\xdb\xdb\xdb\xdb\xdb\xdb\xdb\xdb\xdb\xdb\xdb\xdb\xdb\xdb\xdb\xdb\xdb\xdb\xdb\xdb\xdb\xdb\xdb\xdb\xdb\xdb\xdb\xdb\xdb\xdb\xdb\xdb\xdb\xdb\xdb\xdb\xdb\xdb\xdb\xdb\xdb\xdb\xdb\xdb\xdb\xdb\xdb\xdb\xdb\xdb\xdb\xdb\xdb\xdb\xdb\xdb\xdb\xdb\xdb\xdb\xdb\xdb\xdb\xdb\xdb\xdb\xdb\xdb\xdb\xdb\xdb\xdb\n";**
62. **system("PAUSE");**
63. **}**
64. **void deposit() {**
65. **system("cls");**
66. **cout << "\n\xdb\xdb\xdb\xdb\xdb\xdb\xdb\xdb\xdb\xdb\xdb\xdb\xdb\xdb\xdb\xdb\xdb\xdb\xdb\xdb\xdb\xdb\xdb\xdb ATM ACCOUNT DEPOSIT SYSTEM \xdb\xdb\xdb\xdb\xdb\xdb\xdb\xdb\xdb\xdb\xdb\xdb\xdb\xdb\xdb\xdb\xdb\xdb\xdb\xdb\xdb\xdb\xdb\n\n";**
67. **details();**
68. **cout << "\n\xdb\xdb\xdb\xdb\xdb\xdb\xdb\xdb\xdb\xdb\xdb\xdb\xdb\xdb\xdb\xdb\xdb\xdb\xdb\xdb\xdb\xdb\xdb\xdb\xdb\xdb\xdb\xdb\xdb\xdb\xdb\xdb\xdb\xdb\xdb\xdb\xdb\xdb\xdb\xdb\xdb\xdb\xdb\xdb\xdb\xdb\xdb\xdb\xdb\xdb\xdb\xdb\xdb\xdb\xdb\xdb\xdb\xdb\xdb\xdb\xdb\xdb\xdb\xdb\xdb\xdb\xdb\xdb\xdb\xdb\xdb\xdb\xdb\xdb\xdb\xdb\xdb\xdb\xdb\n";**
69. **cout << "\tEnter the Amount to be Deposited $";**
70. **double depositAmount;**
71. **cin >> depositAmount;**
72. **accountBalance += depositAmount;**
73. **cout << "\n\tYour new available Balanced Amount is $" << accountBalance << endl;**
74. **cout << "\n\t\t\tThank You!\n\n"<< endl;**
75. **system("PAUSE");**
76. **}**
77. **void withdraw() {**
78. **system("cls");**
79. **cout << "\n\xdb\xdb\xdb\xdb\xdb\xdb\xdb\xdb\xdb\xdb\xdb\xdb\xdb\xdb\xdb\xdb\xdb\xdb\xdb\xdb\xdb\xdb\xdb\xdb ATM ACCOUNT WITHDRAWAL \xdb\xdb\xdb\xdb\xdb\xdb\xdb\xdb\xdb\xdb\xdb\xdb\xdb\xdb\xdb\xdb\xdb\xdb\xdb\xdb\xdb\xdb\xdb\xdb\n\n";**
80. **details();**
81. **cout << "\n\xdb\xdb\xdb\xdb\xdb\xdb\xdb\xdb\xdb\xdb\xdb\xdb\xdb\xdb\xdb\xdb\xdb\xdb\xdb\xdb\xdb\xdb\xdb\xdb\xdb\xdb\xdb\xdb\xdb\xdb\xdb\xdb\xdb\xdb\xdb\xdb\xdb\xdb\xdb\xdb\xdb\xdb\xdb\xdb\xdb\xdb\xdb\xdb\xdb\xdb\xdb\xdb\xdb\xdb\xdb\xdb\xdb\xdb\xdb\xdb\xdb\xdb\xdb\xdb\xdb\xdb\xdb\xdb\xdb\xdb\xdb\xdb\xdb\xdb\xdb\xdb\xdb\xdb\xdb\xdb\xdb\xdb\xdb\xdb\xdb\xdb\n";**
82. **cout << "\tEnter the Amount to be Withdrawn $";**
83. **double withdrawalAmount;**
84. **cin >> withdrawalAmount;**
85. **if (withdrawalAmount > accountBalance || withdrawalAmount > limit) {**
86. **cout << endl << "\tPlease Try Again, the withdrawal amount exceeds your available balance or limit!\n" << endl;**
87. **system("PAUSE");**
88. **} else {**
89. **accountBalance -= withdrawalAmount;**
90. **cout << "\n\tYou have successfully Withdrawn: $" << withdrawalAmount << "\n";**
91. **cout << "\n\tYour new Balance Amount is $" << accountBalance << endl;**
92. **cout << "\n\t\t\tThank You!\n\n" << endl;**
93. **system("PAUSE");**
94. **}**
95. **}**
96. **// Function to arrange bills based on due date using bubble sort**
97. **void arrangeBillsByDate() {**
98. **// Bubble sort to arrange bills based on due date**
99. **for (BillNode\* i = head; i != NULL; i = i->next) {**
100. **for (BillNode\* j = head; j->next != NULL; j = j->next) {**
101. **// Compare due dates**
102. **if (j->year > j->next->year ||**
     1. **(j->year == j->next->year && j->month > j->next->month) ||**
     2. **(j->year == j->next->year && j->month == j->next->month && j->day > j->next->day)) {**
     3. **// Swap the bills if they are out of order**
     4. **swap(j->name, j->next->name);**
     5. **swap(j->day, j->next->day);**
     6. **swap(j->month, j->next->month);**
     7. **swap(j->year, j->next->year);**
     8. **swap(j->price, j->next->price);**
103. **}**
104. **}**
105. **}**
106. **}**
107. **void AddBills() {**
108. **system("cls");**
109. **int numberOfBills;**
110. **cout << "\xdb\xdb\xdb\xdb\xdb\xdb\xdb\xdb\xdb\xdb\xdb\xdb\xdb\xdb\xdb\xdb\xdb\xdb\xdb\xdb\xdb\xdb\xdb\xdb Enter the number of bills you want to add: \xdb\xdb\xdb\xdb\xdb\xdb\xdb\xdb\xdb\xdb\xdb\xdb\xdb\xdb\xdb\xdb\xdb\xdb\xdb\xdb\xdb\xdb\xdb" << endl;**
111. **cin >> numberOfBills;**
112. **// Iterate through each bill**
113. **for (int i = 0; i < numberOfBills; ++i) {**
114. **BillNode\* newNode = new BillNode;**
115. **cin.ignore(); // Consume the newline character left in the input buffer**
116. **// Get bill details from the user u**
117. **cout << "Enter bill name: ";**
118. **getline(cin, newNode->name);**
119. **cout << "Enter day, month, and year (separated by spaces): ";**
120. **cin >> newNode->day >> newNode->month >> newNode->year;**
121. **cout << "Enter the price for the bill: $";**
122. **cin >> newNode->price;**
123. **// Insert the current bill at the beginning of the linked list**
124. **insertNode(head, newNode);**
125. **}**
126. **// Arrange the bills based on due date**
127. **arrangeBillsByDate();**
128. **system("PAUSE");**
129. **}**
130. **// Function to count the number of bills in the linked list**
131. **int countBills() {**
132. **int count = 0;**
133. **BillNode\* current = head;**
134. **while (current != NULL) {**
135. **count++;**
136. **current = current->next;**
137. **}**
138. **return count;**
139. **}**
140. **void displayArrangedBills() {**
141. **BillNode\* current = head;**
142. **cout << "\xdb\xdb\xdb\xdb\xdb\xdb\xdb\xdb\xdb\xdb\xdb\xdb Arranged bills based on due date: \xdb\xdb\xdb\xdb\xdb\xdb\xdb\xdb\xdb\xdb\xdb\xdb" << endl;**
143. **while (current != NULL) {**
144. **cout << "Bill Name: " << current->name**
145. **<< " (Date: " << current->day << "/" << current->month << "/" << current->year**
146. **<< ", Price: $" << current->price << ")" << endl;**
147. **current = current->next;**
148. **}**
149. **}**
150. **// Function to handle the payment of bills**
151. **void payBills() {**
152. **// Clear the console screen for a clean display**
153. **system("cls");**
154. **// Check if there are bills to pay**
155. **if (head == NULL) {**
156. **cout << "No bills to pay. Add bills first.\n";**
157. **system("PAUSE");**
158. **return;**
159. **}**
160. **// Display the list of arranged bills with due dates and amounts**
161. **displayArrangedBills();**
162. **// Ask the user to select a bill to pay**
163. **cout << "\xdb\xdb\xdb\xdb\xdb\xdb\xdb\xdb\xdb\xdb\xdb\xdb Enter the number of the bill to pay (1-" << countBills() << "): \xdb\xdb\xdb\xdb\xdb\xdb\xdb\xdb\xdb\xdb\xdb\xdb" << endl;**
164. **int billNumber;**
165. **cin >> billNumber;**
166. **// Validate the user input for the selected bill number**
167. **if (billNumber < 1 || billNumber > countBills()) {**
168. **cout << "Invalid bill number. Please try again.\n";**
169. **system("PAUSE");**
170. **return;**
171. **}**
172. **// Get the amount of the selected bill**
173. **double billAmount;**
174. **BillNode\* current = head;**
175. **for (int i = 1; i < billNumber; ++i) {**
176. **current = current->next;**
177. **}**
178. **billAmount = current->price;**
179. **// Check if the user has sufficient funds to pay the bill**
180. **if (billAmount > accountBalance) {**
181. **cout << "Insufficient funds to pay the bill.\n";**
182. **system("PAUSE");**
183. **return;**
184. **}**
185. **// Deduct the bill amount from the account balance**
186. **accountBalance -= billAmount;**
187. **// Remove the paid bill from the linked list**
188. **removeFromPosition(billNumber - 1);**
189. **/// Display a success message with the updated account balance**
190. **cout << "Bill paid successfully. Your new balance is $" << accountBalance << endl;**
191. **// Pause the system to allow the user to read the message before returning to the main menu**
192. **system("PAUSE");**
193. **}**
194. **void accountExit() {**
195. **system("cls");**
196. **cout << "\n\xB2\xB2\xB2\xB2\xB2\xB2\xB2\xB2\xB2\xB2\xB2\xB2\xB2\xB2\xB2\xB2\xB2\xB2\xB2\xB2\xB2\xB2\xB2\xB2\xB2\xB2\xB2\xB 2\xB2\xB2\xB2\xB2\xB2\xB2\xB2\xB2\xB2\xB2\xB2\xB2\xB2\xB2\xB2\xB2\xB2\xB2\xB2\xB2\xB2\xB2\xB2\xB2\xB2\xB2\xB2\xB2\xB2\xB2\xB2\xB2\xB2\xB2\xB2\xB2\xB2\xB2\xB2\xB2\xB2\xB2\xB2\xB2\xB2\xB2\xB2\xB2\xB2\xB2\xB2\xB2\xB2\xB2\xB2\xB2\xB2\xB2\xB2\xB2\xB2\xB2\xB2\xB2\xB2\xB2\xB2\xB2\xB2\xB2\xB2\xB2\xB2\xB2\n";**
197. **cout << "\n\t\tThank you for using the ATM. Have a great day!\n\n";**
198. **system("PAUSE");**
199. **}**
200. **int getHiddenInput() {**
201. **int input = 0;**
202. **char ch;**
203. **// Read characters until Enter is pressed**
204. **while ((ch = \_getch()) != 13) { // 13 is the ASCII code for Enter**
205. **// Check if the input is a digit**
206. **if (isdigit(ch)) {**
207. **cout << '\*'; // Print '\*' instead of the actual character**
208. **input = input \* 10 + (ch - '0'); // Convert char to int**
209. **}**
210. **}**
211. **cout << endl; // Move to the next line after Enter is pressed**
212. **return input;**
213. **}**
214. **string isUserRobot() {**
215. **cout << "\xdb\xdb\xdb\xdb\xdb\xdb\xdb\xdb\xdb\xdb\xdb\xdb Are you a robot? (yes/no): \xdb\xdb\xdb\xdb\xdb\xdb\xdb\xdb\xdb\xdb\xdb\xdb "<< endl;**
216. **string answer;**
217. **cin >> answer;**
218. **// Convert each character in the answer to lowercase for case-insensitive comparison**
219. **for (size\_t i = 0; i < answer.size(); ++i) {**
220. **answer[i] = tolower(answer[i]);**
221. **}**
222. **return answer;**
223. **}**
224. **int main() {**
225. **int choice;**
226. **while (true) {**
227. **system("cls");**
228. **cout<<"\n\xdb\xdb\xdb\xdb\xdb\xdb\xdb\xdb\xdb\xdb\xdb\xdb\xdb\xdb\xdb\xdb\xdb\xdb\xdb\xdb\xdb\xdb\xdb\xdb ATM ACCOUNT DETAILS \xdb\xdb\xdb\xdb\xdb\xdb\xdb\xdb\xdb\xdb\xdb\xdb\xdb\xdb\xdb\xdb\xdb\xdb\xdb\xdb\xdb\xdb\xdb\xdb\n\n";**
229. **cout << "\t\t\t--------------------\n" << endl;**
230. **cout << "\t\tCurrent date : ";**
231. **time\_t now;**
232. **time(&now);**
233. **printf("%s\n", ctime(&now));;**
234. **cout << "\t\t\t--------------------\n" << endl;**
235. **cout << "\xB2\xB2\xB2\xB2\xB2\xB2\xB2\xB2\xB2\xB2\xB2\xB2\xB2\xB2\xB2\xB2\xB2\xB2\xB2\xB2\xB2\xB2\xB2\xB2\xB2\xB2\xB2\xB2\xB2\xB2\xB2\xB2\xB2\xB2\xB2\xB2\xB2\xB2\xB2\xB2\xB2\xB2\xB2\xB2\xB2\xB2\xB2\xB2\xB2\xB2\xB2\xB2\xB2\xB2\xB2\xB2\xB2\xB2\xB2\xB2\xB2\xB2\xB2\xB2\xB2\xB2\xB2\xB2\xB2\xB2\xB2\xB2\xB2\xB2\xB2\n\n";**
236. **cout << "\tPress 1 and Then Press Enter to Access Your Account Via Pin Number\n\n";**
237. **cout << "\t\t\t\t\t or \n\n";**
238. **cout << "\tPress 0 and press Enter to get Help.\n\n";**
239. **int access;**
240. **cin >> access;**
241. **switch (access) {**
242. **case 1:**
243. **system("cls");**
244. **int i, pin;**
245. **if (isUserRobot() == "yes" ) {**
246. **cout << "Sorry, only humans are allowed to access this account." << endl;**
247. **system("PAUSE");**
248. **exit(1);**
249. **}**
250. **system("cls");**
251. **cout << "\n\xB2\xB2\xB2\xB2\xB2\xB2\xB2\xB2\xB2\xB2\xB2\xB2\xB2\xB2\xB2\xB2\xB2\xB2\xB2\xB2\xB2\xB2\xB2\xB2\xB2\xB2\xB2\xB2\xB2\xB2 ATM ACCOUNT ACCESS \xB2\xB2\xB2\xB2\xB2\xB2\xB2\xB2\xB2\xB2\xB2\xB2\xB2\xB2\xB2\xB2\xB2\xB2\xB2\xB2\xB2\xB2\xB2\xB2\xB2\xB2\xB2\xB2\xB2\xB2\xB2\xB2\xB2\xB2\xB2\xB2\xB2\xB2\xB2\xB2\xB2\xB2\xB2\xB2\xB2\xB2\xB2\xB2\xB2\xB2\xB2\xB2\xB2\xB2\xB2\xB2\xB2\xB2\xB2\xB2\xB2\xB2\n\n";**
252. **cout << "\n\nEnter Your Acc Pin Access Number! [Only one attempt is allowed]\n\n" << endl;**
253. **cout << "\xB2\xB2\xB2\xB2\xB2\xB2\xB2\xB2\xB2\xB2\xB2\xB2\xB2\xB2\xB2\xB2\xB2\xB2\xB2\xB2\xB2\xB2\xB2\xB2\xB2\xB2\xB2\xB2\xB2\xB2\xB2\xB2\xB2\xB2\xB2\xB2\xB2\xB2\xB2\xB2\xB2\xB2\xB2\xB2\xB2\xB2\xB2\xB2\xB2\xB2\xB2\xB2\xB2\xB2\xB2\xB2\xB2\xB2\xB2\xB2\xB2\xB2\xB2\xB2\xB2\xB2\xB2\xB2\xB2\xB2\xB2\xB2\xB2\xB2\xB2\xB2\xB2\n\n";**
254. **pin = getHiddenInput();**
255. **system("cls");**
256. **if (pin == 12345) {**
257. **system("cls");**
258. **do {**
259. **system("cls");**
260. **cout << endl << "\n\xdb\xdb\xdb\xdb\xdb\xdb\xdb\xdb\xdb\xdb\xdb\xdb\xdb\xdb\xdb\xdb\xdb\xdb\xdb\xdb\xdb\xdb\xdb ATM Main Menu Screen \xdb\xdb\xdb\xdb\xdb\xdb\xdb\xdb\xdb\xdb\xdb\xdb\xdb\xdb\xdb\xdb\xdb\xdb\xdb\xdb\xdb\xdb\xdb\xdb\n" << endl << endl;**
261. **cout << "\tPLEASE ENTER A SELECTION AND PRESS RETURN KEY: \n\n";**
262. **cout << "\t\tEnter [1] To Deposit Cash" << endl;**
263. **cout << "\t\tEnter [2] To Withdraw Cash" << endl;**
264. **cout << "\t\tEnter [3] To Balance Inquiry" << endl;**
265. **cout << "\t\tEnter [4] Add new bills" << endl;**
266. **cout << "\t\tEnter [5] Pay bills" << endl;**
267. **cout << "\t\tEnter [0] to Exit ATM" << endl << endl;**
268. **cin >> choice;**
269. **switch (choice) {**
270. **case 0:**
271. **system("cls");**
272. **cout << "\n\xdb\xdb\xdb\xdb\xdb\xdb\xdb\xdb\xdb\xdb\xdb\xdb\xdb\xdb\xdb\xdb\xdb\xdb\xdb\xdb\xdb\xdb\xdb\xdb ATM ACCOUNT STATUS \xdb\xdb\xdb\xdb\xdb\xdb\xdb\xdb\xdb\xdb\xdb\xdb\xdb\xdb\xdb\xdb\xdb\xdb\xdb\xdb\xdb\xdb\xdb\xdb\n\n";**
273. **cout << "\t See your bank representative for assistance during bank opening hours\n\n";**
274. **cout << "\t\tThank you for choosing our services today!!\n\n";**
275. **cout << "\xdb\xdb\xdb\xdb\xdb\xdb\xdb\xdb\xdb\xdb\xdb\xdb\xdb\xdb\xdb\xdb\xdb\xdb\xdb\xdb\xdb\xdb\xdb\xdb\xdb\xdb\xdb\xdb\xdb\xdb\xdb\xdb\xdb\xdb\xdb\xdb\xdb\xdb\xdb\xdb\xdb\xdb\xdb\xdb\xdb\xdb\xdb\xdb\xdb\xdb\xdb\xdb\xdb\xdb\xdb\xdb\xdb\xdb\xdb\xdb\xdb\xdb\xdb\xdb\xdb\xdb\xdb\xdb\xdb\xdb\xdb\xdb\xdb\xdb\xdb\xdb\xdb\xdb\xdb\xdb\xdb\xdb\xdb\xdb\xdb\n\n";**
276. **system("PAUSE");**
277. **exit(0);**
278. **break;**
279. **case 1:**
280. **deposit();**
281. **break;**
282. **case 2:**
283. **withdraw();**
284. **break;**
285. **case 3:**
286. **detail();**
287. **break;**
288. **case 4:**
289. **AddBills();**
290. **displayArrangedBills();**
291. **break;**
292. **case 5:**
293. **payBills();**
294. **break;**
295. **default:**
296. **cout << "\nInvalid choice. Please try again.\n";**
297. **break;**
298. **}**
299. **} while (choice != 0);**
300. **} else {**
301. **system("cls");**
302. **cout << "\n\xdb\xdb\xdb\xdb\xdb\xdb\xdb\xdb\xdb\xdb\xdb\xdb\xdb\xdb\xdb\xdb\xdb\xdb\xdb\xdb\xdb\xdb\xdb\xdb THANK YOU \xdb\xdb\xdb\xdb\xdb\xdb\xdb\xdb\xdb\xdb\xdb\xdb\xdb\xdb\xdb\xdb\xdb\xdb\xdb\xdb\xdb\xdb\xdb\xdb\n\n";**
303. **cout << "\nYou had made your attempt which failed!!! No More attempts allowed!! Sorry!!\n\n";**
304. **cout << "\xdb\xdb\xdb\xdb\xdb\xdb\xdb\xdb\xdb\xdb\xdb\xdb\xdb\xdb\xdb\xdb\xdb\xdb\xdb\xdb\xdb\xdb\xdb\xdb\xdb\xdb\xdb\xdb\xdb\xdb\xdb\xdb\xdb\xdb\xdb\xdb\xdb\xdb\xdb\xdb\xdb\xdb\xdb\xdb\xdb\xdb\xdb\xdb\xdb\xdb\xdb\xdb\xdb\xdb\xdb\xdb\xdb\xdb\xdb\xdb\xdb\xdb\xdb\xdb\xdb\xdb\xdb\xdb\xdb\xdb\xdb\xdb\xdb\xdb\xdb\xdb\xdb\xdb\xdb\xdb\xdb\xdb\xdb\xdb\xdb\xdb\xdb\xdb\xdb\n\n";**
305. **system("PAUSE");**
306. **exit(1);**
307. **}**
308. **break;**
309. **case 0:**
310. **system("cls");**
311. **cout << "\n\n\n\t\t\t\t ATM HELP AND CUSTOMER SUPPORT\n\n\n";**
312. **cout << "The ATM is a computerized device which provides various facilities to its account holders, such as:\n\n";**
313. **cout << "\t- Cash withdrawals\n";**
314. **cout << "\t- Balance inquiry\n";**
315. **cout << "\t- Cash deposit\n";**
316. **cout << "\t- Bill payments\n";**
317. **cout << "\t- Account management\n";**
318. **cout << "\t- And more...\n\n";**
319. **cout << "For assistance or to report a problem, please contact our customer\n";**
320. **cout << "support at 1-800-123-4567.\n\n";**
321. **system("PAUSE");**
322. **break;**
323. **default:**
324. **cout << "Invalid choice. Please try again.\n";**
325. **break;**
326. **}**
327. **}**
328. **return 0;**
329. **}**

# صورة تحتوي على نص, لقطة شاشة, برمجيات, برامج الوسائط المتعددة تم إنشاء الوصف تلقائياً**Results and Discussion**

# 

# 

# 

# 

# 

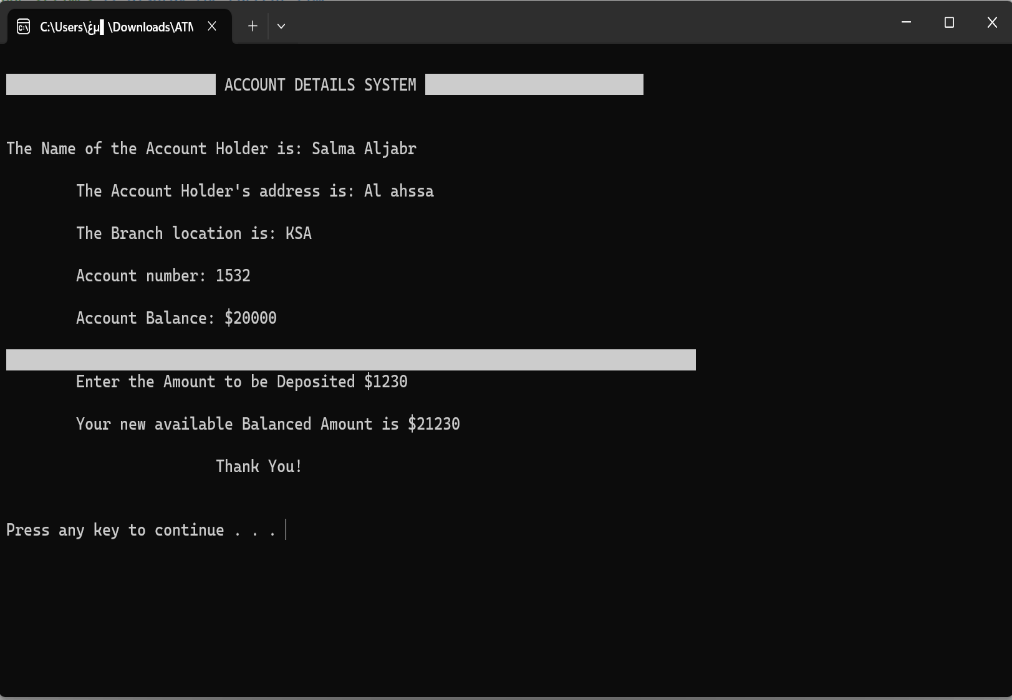
# 

# 

**صورة تحتوي على نص, لقطة شاشة, برمجيات

تم إنشاء الوصف تلقائياً**

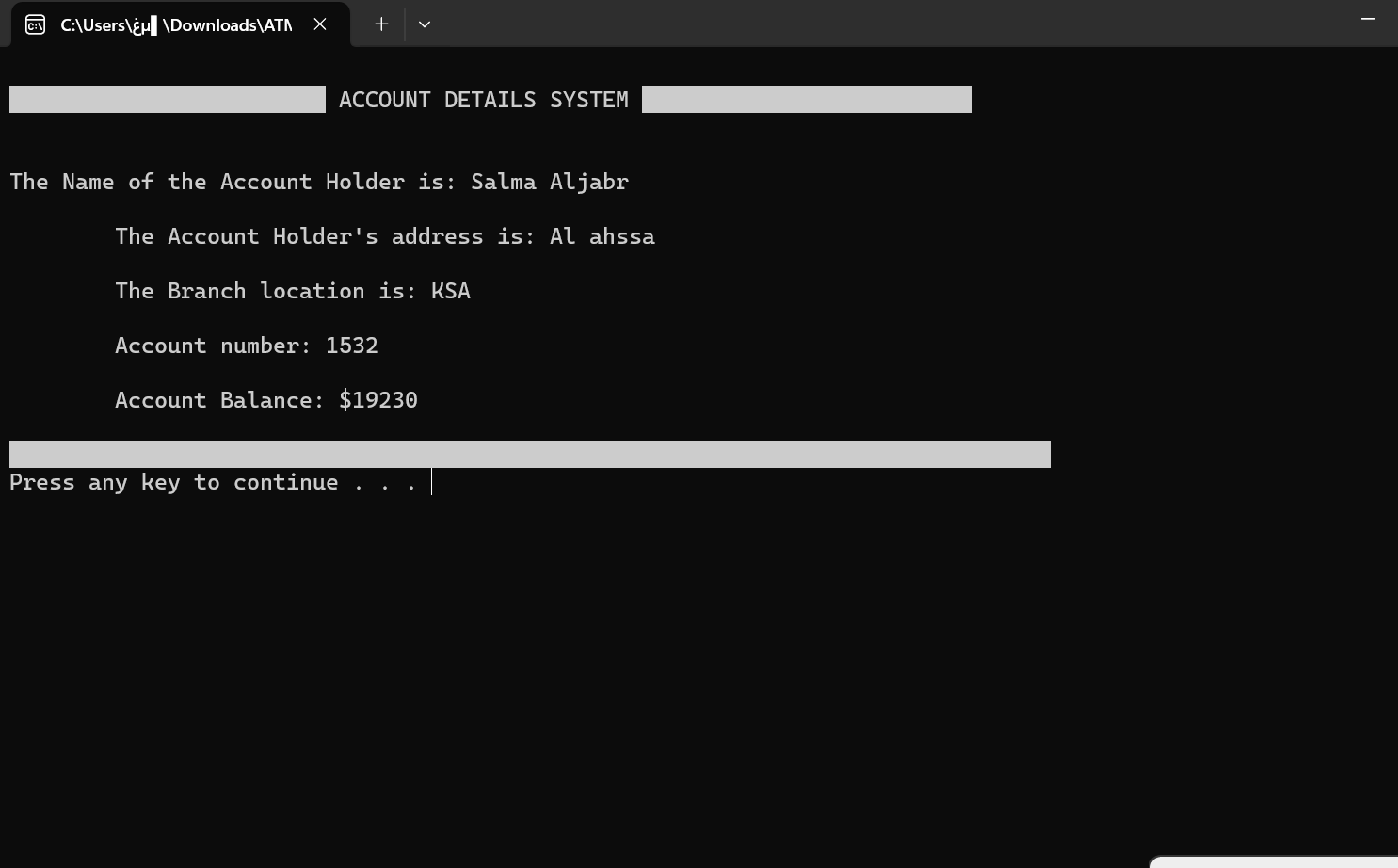
1. Deposit Cash

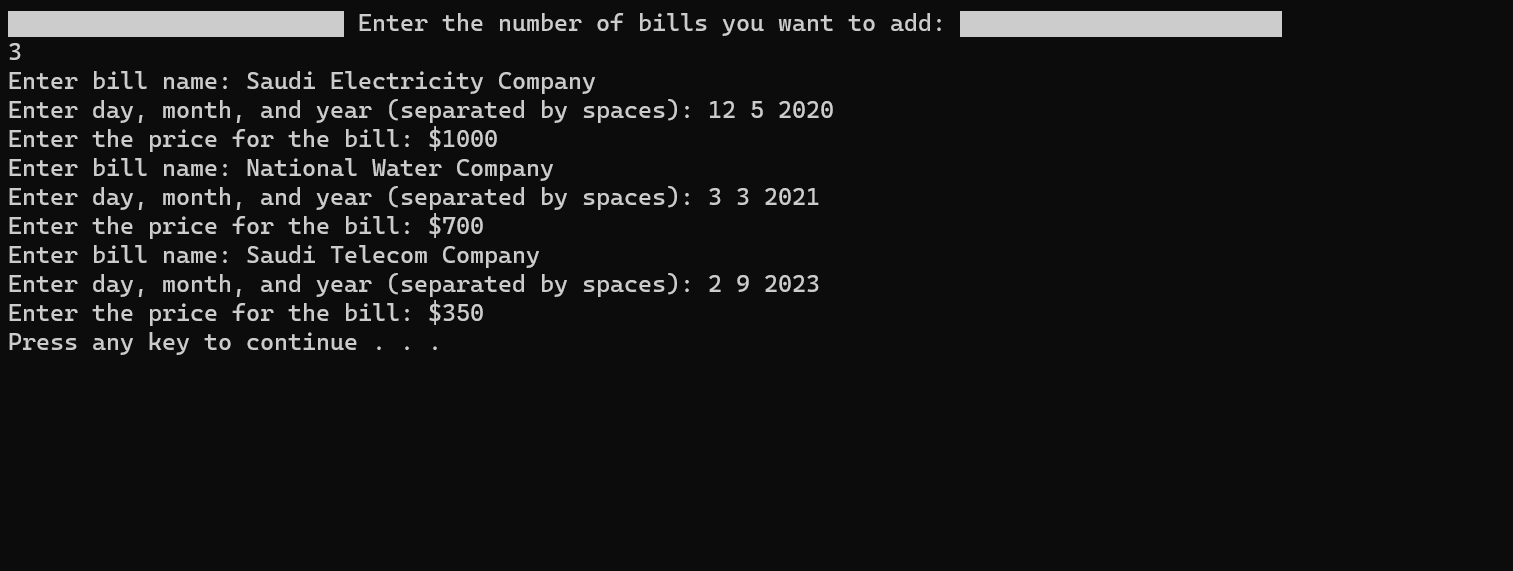
****

# 2. Withdraw Cash

# 

3. Balance Inquiry

****

**** 4. Add new bills

# 5. Pay bills

# 6. Exit ATM

# صورة تحتوي على نص, لقطة شاشة, برمجيات تم إنشاء الوصف تلقائياً

# **Conclusion**

In conclusion, introducing the advanced ATM system is a useful and effective way to overcome issues in traditional banking. By fixing problems and making it easier for people to use, our solution brings a noticeable and quick improvement to banking. Using advanced data structures has played a big part in making operations smoother, and making the financial process work better. When we think about the original problem and the solutions we've suggested, bringing modern technology into banking makes sense and is needed to keep up with what people want. The good things about this solution, which are based on being practical, make banking more convenient and user-friendly for people.

# **References**

* Data structure and algorithm analysis in C++/ Mark Allen Weiss, Florida International University — Fourth edition.