**VYDEHI SCHOOL OF EXCELLENCE**

Affiliated to CBSE, Delhi

Vydehi campus, Whitefield, Bengaluru

Karnataka

A logo of a school

Description automatically generated

**COMPUTER SCIENCE (083)**

Project on:

**Banking System implemented using Python and MySQL**

Year: 2024-25

Submitted to Submitted by–Nandan Goyal

**Ms. Ranjeeta Shrivastava** Class – XII  **A**

**VYDEHI SCHOOL OF EXCELLENCE**A logo of a school

Description automatically generated

**DEPARTMENT OF COMPUTER SCIENCE**

**CERTIFICATE**

This is to certify that **NANDAN GOYAL** of class **XII-A** has successfully completed the project under the guidance of **MS. RANJEETA SHRIVASTAVA** during the academic year **2024-25** in partial fulfilment of **COMPUTER SCIENCE** practical examination conducted by AISSCE, New Delhi**.**

***Signature of the external examiner Signature of the internal examiner***

External examiner no:  ***Signature of the Principal***

**ACKNOWLEDGMENT**

I want to express my most sincere thanks and gratitude to Ms. Ranjeeta, my computer science teacher, who's valuable feedback and guidance in this project was a major contribution. Her constant motivation and advice were the foundation to this project.

Next, I'd like to thank Ms. Chhavi Garg, our principal ma'am, for extending support and coordination wherever possible which aided in the productive and successful completion of this project.

Finally, I'd like to thank everyone who helped me directly, or indirectly in the completion of this project.

**INDEX**

|  |  |  |
| --- | --- | --- |
| **S.no** | **Title** | **Page no.** |
| 1 | Introduction | 1 |
| 2 | Why Python? | 2 |
| 3 | System Specs | 3 |
| 4 | Aim | 4 |
| 5 | Some Background | 5 |
| 6 | Program Code | 7 |
| 7 | Flowchart | 21 |
| 8 | Sample Output | 22 |
| 9 | Bibliography | 30 |

**INTRODUCTION**

This project is a terminal-based net-banking app, in which users can:

1. Create and log into a password protected account
2. Deposit money
3. Make transactions with other users
4. Create fixed deposits and gain interest

All user-data is stored in and retrieved from a MySQL database. For the application itself, Python is used.

One of the main features of this project in terms of development is its state machine architecture, which is discussed in detail later in this document.

**WHY PYTHON?**

* **Cross-platform Language**: Python can run equally well on variety of platforms – Windows, Linus/UNIX, smartphones, etc.
* **Simple and expressive syntax:** Python has a simple syntax similar to the English language. It is thus very expressive with fewer lines of code and simplicity compared to other popular languages like C++, Java etc.
* **Quick prototyping:** Python runs on an interpreter system, so the code can be executed as soon as it is written. This, along with its simplicity, means that prototyping can be very quick.
* **Multi-paradigm:** Python can be written in a procedural way, an object-oriented way or a functional way.

**SYSTEM SPECS**

|  |  |
| --- | --- |
| **Operating System** | Windows 10 |
| **Processor** | Intel Core i3 7th gen @ 2.30 Ghz |
| **RAM** | 12 GB |
| **Hard disk** | SSD 233 GB, HDD 932 GB |

**AIM**

To create a net-banking client application with terminal-based UI.

**Some Background**

As mentioned before, the state machine architecture of this application is a highlight of this project.

Based on what functionality the user wants to access, different kinds of processing have to be done. The idea of a state arises from this situation naturally. Based on user input, we will set a certain “state”, and based on the current state, some processing will be done. Each state can also change the current state to a different one, allowing navigation between different states.

A naïve implementation would declare constants that represent different states, and would check in an if-elif chain what state is currently set, and run code based on that, like so:

1 | STATE0 = 0

2 | STATE1 = 1

3 | currentState = 0

4 | ​

5 | while True: *# main process-loop*

6 |  if currentState == STATE0:

7 |  *# STATE0'S processing*

8 |

9 |  inp = userInput()

10|

11|  if inp == "change\_state":

12|             *# some user-input condition*

13|  currentState = STATE1

14|

15|  continue

16|

17|  elif currentState == STATE1:

18|  *# STATE1'S processing*

19|

20|  inp = userInput()

21|

22|  if inp == "change\_state":

23|  currentState = STATE0

24|

25|  continue

However, this if-elif chain can quickly grow very large. Since the states are being set by the code itself explicitly, there shouldn’t be any need to check for the state in each process-loop. Moreover, the implementation for different states cannot be separated and thus modularization cannot be achieved, which would be desirable from a code-design standpoint.

The problem is that the state in the above code is represented by an integer object, which does not contain any information about what kind of processing it needs. Thus, the current state needs to be checked and its implementation has to be provided by the main process-loop itself. However, if the state was represented by an object that itself contained information about the required processing, then we could just use that information without caring what the current state exactly is. This state-object can be a class that contains a process() function, which is called by the main process-loop. This eliminates the need of if-checks altogether. Also, since the class definitions can be written separately, it improves code-design by allowing modularization. This design is exemplified by the application code following this page.

Note: We have designed the program so that many in-program days pass in just a few real-life seconds, so that we can demonstrate fixed deposit interests.

The complete development history and project files can be found here :

https://github.com/satwik-krit/banking-system

The general program execution can be expressed as follows:

**A diagram of a flowchart

Description automatically generated**

**PROGRAM**

**CODE**

SQL queries :

1 | **DROP** DATABASE IF EXISTS Bank;

2 | **CREATE** DATABASE Bank;

3 | USE Bank;

4 | ​

5 | **CREATE** **TABLE** Users (

6 |    password VARCHAR(10) **NOT** NULL,

7 |    username VARCHAR(50),

8 |    firstname VARCHAR(50) **NOT** NULL,

9 |    lastname VARCHAR(50) **NOT** NULL,

10|    age INT **NOT** NULL,

11|    phone VARCHAR(15) **NOT** NULL,

12|    inactive TINYINT(1) **NOT** NULL DEFAULT 0,

13| ​

14|    PRIMARY KEY(username),

15|    CHECK(age > 0)

16| );

17| ​

18| **CREATE** **TABLE** Account (

19|    balance INT **NOT** NULL,

20|    created DATE **NOT** NULL,

21|    frozen TINYINT(1) **NOT** NULL DEFAULT 0,

22|    username VARCHAR(50),

23| ​

24|    PRIMARY KEY(username),

25|    CHECK(balance >= 0),

26|    FOREIGN KEY(username) REFERENCES Users(username)

27|     **ON** **DELETE** CASCADE

28|     **ON** **UPDATE** CASCADE

29| );

30| ​

31| **CREATE** **TABLE** FixedDepo (

32|    fdName VARCHAR(30),

33|    username VARCHAR(50),

34|    principal INT **NOT** NULL,

35|    interest INT **NOT** NULL,

36|    creationdate DATE **NOT** NULL,

37|    timeperiod INT **NOT** NULL,

38|    maturedate DATE **NOT** NULL,

39|    withdrawn INT **NOT** NULL DEFAULT 0,

40| ​

41|    PRIMARY KEY(fdName, username),

42|    CHECK(principal > 0),

43|    CHECK(interest > 0),

44|    CHECK(timeperiod **BETWEEN** 0 **AND** 10),

45|    CHECK(maturedate = DATE\_ADD(creationdate, INTERVAL timeperiod

YEAR)),

46|    FOREIGN KEY(username) REFERENCES Users(username)

47|     **ON** **DELETE** CASCADE

48|     **ON** **UPDATE** CASCADE

49| );

50| ​

51| **CREATE** **TABLE** Transactions (

52|    transID INT AUTO\_INCREMENT,

53|    payerID VARCHAR(50) **NOT** NULL,

54|    receiverID VARCHAR(50) **NOT** NULL,

55|    transDate DATE **NOT** NULL,

56|    amount INT **NOT** NULL,

57|    comment TINYTEXT,

58| ​

59|    PRIMARY KEY(transID),

60|    CHECK(amount > 0),

61|    FOREIGN KEY(payerID) REFERENCES Users(username)

62|     **ON** **DELETE** RESTRICT

63|     **ON** **UPDATE** CASCADE,

64|    FOREIGN KEY(payerID) REFERENCES Users(username)

65|     **ON** **DELETE** RESTRICT

66|     **ON** **UPDATE** CASCADE

67| );

68| ​

69| **CREATE** **TABLE** Updates (

70|    username VARCHAR(50) **NOT** NULL,

71|    baseContent TINYTEXT **NOT** NULL,

72|    extraContent TEXT **NOT** NULL,

73|    updateDate DATE,

74| ​

75|    FOREIGN KEY(username) REFERENCES Users(username)

76| );

77| ​

78| **CREATE** **TABLE** EnvInfo (

79|    DBCreationDateTime TIMESTAMP DEFAULT CURRENT\_TIMESTAMP

80| );

81| ​

82| **INSERT** **INTO** EnvInfo

83| **VALUES** ();

Python code :

1 | **import** time

2 | **from** getpass **import** getpass

3 | **import** datetime **as** dt

4 | **from** dateutil.relativedelta **import** relativedelta

5 | **import** mysql.connector **as** sqlconn

6 | **from** mysql.connector **import** DataError, DatabaseError, OperationalError,

NotSupportedError, IntegrityError, ProgrammingError, InternalError

7 | ​

8 | **try**:

9 | ​

10|     currentState = **None**

11|     TIMEDELTA = 0.2

12|     currentDate = **None**

13| ​

14|     db = sqlconn.connect(host="localhost", user="root",

password="root", database="bank", charset="utf8")

15|     crsr = db.cursor(buffered=**True**)

16|

17|     **def** EXIT(code=0):

18|         db.close()

19|         exit(code)

20| ​

21|     **def** execute(query : str, args : tuple) -> **None**:

22|         crsr.execute(query.format(\*args))

23| ​

24|     **def** resultExists(result):

25|         **if** len(result):

26|             **return** **True**

27|         **else**:

28|             **return** **False**

29| ​

30|     **def** getBalance(username : str) -> int:

31|         Q\_GET\_BALANCE = ("SELECT balance "

32|                          "FROM account "

33|                          "WHERE username = '{}';")

34| ​

35|         execute(Q\_GET\_BALANCE, (username,))

36|         **return** crsr.fetchone()[0]

37| ​

38|     **def** c\_changeBalance(username : str, change : int) -> **None**:

39|         QC\_CHANGE\_BALANCE = ("UPDATE Account "

40|                              "SET balance = balance + {1} "

41|                              "WHERE username = '{0}'; ")

42|

43|         execute(QC\_CHANGE\_BALANCE, (username, change))

44| ​

45|     **def** userExists(username : str) -> bool:

46|         Q\_CHECK\_USERNAME = ("SELECT username "

47|                             "FROM Users "

48|                             "WHERE username = '{}';")

49| ​

50|         execute(Q\_CHECK\_USERNAME, (username,))

51| ​

52|         **if** len(crsr.fetchall()) != 0:

53|             **return** **True**

54|         **else**:

55|             **return** **False**

56| ​

57|     **def** checkFDExists(username : str, fdName : str) -> bool:

58|         Q\_CHECK\_FD\_EXISTS = ("SELECT \* FROM FixedDepo "

59|                              "WHERE username = '{}' AND fdName = '{}';

")

60| ​

61|         execute(Q\_CHECK\_FD\_EXISTS, (username, fdName))

62| ​

63|         **if** len(crsr.fetchall()) != 0:

64|             **return** **True**

65|         **else**:

66|             **return** **False**

67| ​

68|     **def** intInput(prompt : str, failMsg : str = "Invalid input.") ->

int:

69|         **while** **True**:

70|             inpStr = input(prompt).strip()

71| ​

72|             **if** **not** inpStr.isdigit():

73|                 print(failMsg)

74|             **else**:

75|                 **return** int(inpStr)

76|

77|     **def** getUpdates(username, date=**None**):

78|         \_Q\_GET\_UPDATES\_ALL = ("SELECT baseContent, extraContent,

updateDate "

79|                               "FROM Updates "

80|                               "WHERE username = '{}';")

81| ​

82|         \_Q\_GET\_UPDATES\_DAY = ("SELECT baseContent, extraContent,

updateDate "

83|                               "FROM Updates "

84|                               "WHERE username = '{}' "

85|                               "AND updateDate = '{}'")

86| ​

87|         **if** date:

88|             execute(\_Q\_GET\_UPDATES\_DAY, (username, date))

89|             **return** crsr.fetchall()

90|         **else**:

91|             execute(\_Q\_GET\_UPDATES\_ALL, (username, ))

92|             **return** crsr.fetchall()

93| ​

94|     **def** c\_createUpdate(username, baseContent, extraContent="No

comment", \_date=**None**):

95|         \_QC\_CREATE\_UPDATE = ("INSERT INTO Updates "

96|                              "VALUES "

97|                              "('{}', '{}', '{}', '{}')")

98| ​

99|         **if** \_date:

100|             execute(\_QC\_CREATE\_UPDATE, (username, baseContent,

extraContent, \_date))

101|         **else**:

102|             execute(\_QC\_CREATE\_UPDATE, (username, baseContent,

extraContent, currentDate))

103| ​

104|     **def** getUserInfo(username):

105|         \_Q\_GET\_USER = ("SELECT firstname, lastname, age, phone,

inactive "

106|                        "FROM Users "

107|                        "WHERE username = '{}' ;")

108| ​

109|         execute(\_Q\_GET\_USER, (username,))

110|         **return** crsr.fetchone()

111| ​

112|     **class** LockedState:

113|         **def** \_\_init\_\_(self):

114|             **pass**

115| ​

116|         **def** process(self):

117|             **global** currentState

118| ​

119|             print("===================================================

===================")

120|             print("Enter username and password to view details or

create a new account")

121|             print("(1) Login")

122|             print("(2) Create an account")

123|             print("(3) Quit")

124|             print()

125| ​

126|             option = intInput("(Option) -> ")

127| ​

128|             **if** option == 1:

129|                 currentState = LoginState()

130| ​

131|             **elif** option == 2:

132|                 currentState = CreateAccountState()

133| ​

134|             **elif** option == 3:

135|                 EXIT()

136| ​

137|             **else**:

138|                 print()

139|                 print("Please choose a valid option.")

140| ​

141|     **class** LoginState:

142|         \_Q\_LOGIN\_USER = ("SELECT username, password "

143|                          "FROM Users "

144|                          "WHERE username = '{}'; ")

145| ​

146|         **def** \_\_init\_\_(self):

147|             **pass**

148| ​

149|         **def** \_login(self, username : str, password : str) -> int:

150|             **global** currentState

151| ​

152|             execute(self.\_Q\_LOGIN\_USER, (username,))

153|             record = crsr.fetchone()

154| ​

155|             **if** record == **None**:

156|                 print("Username not found.")

157|                 currentState = LockedState()

158|                 **return**

159|

160|             **if** record[1] != password:

161|                 print("Incorrect password.")

162|                 currentState = LockedState()

163|                 **return**

164| ​

165|             print("Logged in successfully.")

166| ​

167|             currentState = UnlockedState(username)

168| ​

169|         **def** process(self):

170|             print("=======================================")

171|             username = input("(Enter Username) -> ").strip()

172|             password = getpass("(Enter Password) -> ").strip()

173|             print()

174| ​

175|             self.\_login(username, password)

176| ​

177|     **class** CreateAccountState:

178|         \_QC\_CREATE\_USER = ("INSERT INTO Users VALUES "

179|                            "('{}', '{}', '{}', '{}', {}, '{}', {}); ")

180| ​

181|         \_QC\_CREATE\_ACCOUNT = ("INSERT INTO account "

182|                               "VALUES "

183|                               "({}, '{}', {}, '{}'); ")

184| ​

185|         **def** \_\_init\_\_(self):

186|             **pass**

187|

188|         **def** \_createNewUser(self, username : str, password : str,

firstname : str,

189|                         lastname : str, age : int, phone : int) ->

int:

190|             execute(self.\_QC\_CREATE\_USER, (password, username,

firstname, lastname, age, phone, 0))

191|             execute(self.\_QC\_CREATE\_ACCOUNT, (0, str(currentDate), 0,

username))

192|             db.commit()

193| ​

194|         **def** process(self):

195|             **global** currentState

196| ​

197|             print("========================================")

198|             print("(0) Create account")

199|             print("(1) Abort")

200|             print()

201| ​

202|             option = intInput("(Option) -> ")

203| ​

204|             **if** option == 0:

205|                 print()

206|                 username = input("(Enter NEW Username) -> ").strip()

207| ​

208|                 **if** userExists(username):

209|                     print()

210|                     print("Username not unique.")

211|                     **return**

212| ​

213|                 **while** **True**:

214|                     password = input("(Enter NEW Password) ->

").strip()

215|                     confirmPassword = getpass("(Enter password for

confirmation) -> ").strip()

216| ​

217|                     **if** password == confirmPassword:

218|                         **break**

219| ​

220|                     print("Passwords do not match. Enter again.")

221| ​

222|                 firstname = input("(Enter first name) -> ").strip()

223|                 lastname = input("(Enter last name) -> ").strip()

224|                 age = intInput("(Enter age) -> ")

225|                 phone = intInput("(Enter phone no.) -> ")

226|                 print()

227| ​

228|                 self.\_createNewUser(username, password, firstname,

lastname, age, phone)

229| ​

230|                 currentState = UnlockedState(username)

231| ​

232|             **elif** option == 1:

233|                 currentState = LockedState()

234| ​

235|             **else**:

236|                 print()

237|                 print("Please choose a valid option")

238| ​

239|     **class** UnlockedState:

240|         **def** \_\_init\_\_(self, username : str):

241|             self.\_username = username

242| ​

243|         **def** process(self):

244|             **global** currentState

245|             **global** currentDate

246| ​

247|             # print and remove updates

248|             balance = getBalance(self.\_username)

249|             updates = getUpdates(self.\_username, currentDate)

250| ​

251|             print("===================================")

252|             print(currentDate)

253|             print(f"BALANCE: {balance}")

254|             **if** resultExists(updates):

255|                 print("TODAY'S UPDATES:", end=" ")

256|                 **for** content, \_, \_\_ **in** updates:

257|                     print(f"{content}", end=", ")

258|             print()

259|             print("(0) Logout")

260|             print("(1) Pay")

261|             print("(2) Deposit")

262|             print("(3) Create a fixed deposit")

263|             print("(4) Modify/View fixed deposits")

264|             print("(5) View all updates for your account")

265|             print()

266| ​

267|             option = intInput("(Option) -> ")

268| ​

269|             **if** option == 1:

270|                 currentState = PayState(self.\_username)

271| ​

272|             **elif** option == 2:

273|                 currentState = DepositState(self.\_username)

274| ​

275|             **elif** option == 3:

276|                 currentState = CreateFDState(self.\_username)

277| ​

278|             **elif** option == 0:

279|                 currentState = LockedState()

280| ​

281|             **elif** option == 4:

282|                 currentState = ViewFDState(self.\_username)

283|

284|             **elif** option == 5:

285|                 currentState = ViewUpdatesState(self.\_username)

286| ​

287|             **else**:

288|                 print()

289|                 print("Please choose a valid option.")

290| ​

291|     **class** PayState:

292|         \_QC\_PAY\_USER = ("INSERT INTO transactions "

293|                         "(payerID, receiverID, transDate, amount,

comment) "

294|                         "VALUES "

295|                         "('{}', '{}', '{}', {}, '{}'); ")

296|

297|         \_Q\_GETUSERPASSWORD = ("SELECT password "

298|                               "FROM Users "

299|                               "WHERE username = '{}'; ")

300| ​

301|         **def** \_\_init\_\_(self, username : str):

302|             self.\_username = username

303| ​

304|         **def** \_pay(self, receiverName : str, amount : float, comment:

str) -> int:

305|             **global** currentState

306| ​

307|             **global** currentState

308|             balance = getBalance(self.\_username)

309| ​

310|             **if** receiverName == self.\_username:

311|                 print("You cannot pay yourself.")

312|                 **return**

313| ​

314|             **if** **not** userExists(receiverName):

315|                 print("This receiver does not exist.")

316|                 **return**

317| ​

318|             **if** amount == 0:

319|                 print("Enter a valid amount to pay.")

320|                 **return**

321| ​

322|             **if** amount > balance:

323|                 print("You do not have sufficient balance.")

324|                 **return**

325| ​

326|             inpPwd = getpass("(Enter password to proceed with payment)

-> ")

327|             execute(self.\_Q\_GETUSERPASSWORD, (self.\_username, ))

328|             userPwd = crsr.fetchone()[0]

329| ​

330|             **if** inpPwd != userPwd:

331|                 print("Incorrect password, aborting payment.")

332|                 **return**

333| ​

334|             c\_changeBalance(self.\_username, -amount)

335|             execute(self.\_QC\_PAY\_USER, (self.\_username, receiverName,

str(currentDate), amount, comment))

336|             c\_changeBalance(receiverName, amount)

337| ​

338|             recFirstName = getUserInfo(receiverName)[0]

339|             userFirstName = getUserInfo(self.\_username)[0]

340|             c\_createUpdate(receiverName, f"{userFirstName} paid

{amount}", f"{comment}")

341|             c\_createUpdate(self.\_username, f"Paid {amount} to

{recFirstName}", f"{comment}")

342| ​

343|             db.commit()

344| ​

345|             print("Transaction made successfully.")

346| ​

347|         **def** process(self):

348|             **global** currentState

349| ​

350|             print("===========================")

351|             print("(0) Pay to another user")

352|             print("(1) Abort")

353|             print()

354| ​

355|             option = intInput("(Option) -> ")

356| ​

357|             **if** option == 0:

358|                 print()

359|                 receiverName = input("(Enter username of receiver) ->

").strip()

360|                 amount = intInput("(Enter amount to pay) -> ")

361|                 comment =  input("Enter comment (optional)) ->

").strip()

362|                 print()

363| ​

364|                 **if** **not** comment:

365|                     comment = "No comment"

366| ​

367|                 self.\_pay(receiverName, amount, comment)

368| ​

369|             **elif** option == 1:

370|                 currentState = UnlockedState(self.\_username)

371| ​

372|             **else**:

373|                 print()

374|                 print("Please choose a valid option.")

375| ​

376|     **class** DepositState:

377|         **def** \_\_init\_\_(self, username : str):

378|             self.\_username = username

379| ​

380|         **def** \_deposit(self, amount : int) -> **None**:

381|             c\_changeBalance(self.\_username, amount)

382|             c\_createUpdate(self.\_username, f"Deposit {amount}")

383|             db.commit()

384| ​

385|         **def** process(self):

386|             **global** currentState

387| ​

388|             print("===================================================

===================")

389|             amount = intInput("(Enter amount to deposit (cash to

digital money)) -> ")

390|             self.\_deposit(amount)

391| ​

392|             currentState = UnlockedState(self.\_username)

393| ​

394|     **class** CreateFDState:

395|         \_QC\_CREATE\_FD = ("INSERT INTO FixedDepo "

396|                          "(fdName, username, principal, interest,

creationdate, timeperiod, maturedate) "

397|                          "VALUES('{}', '{}', {}, {}, '{}', {}, '{}');

")

398| ​

399|         **def** \_\_init\_\_(self, username : str):

400|             self.\_username = username

401| ​

402|         **def** \_createFD(self, name : str, amount : int, period : int) ->

**None**:

403|             **if** checkFDExists(self.\_username, name):

404|                 print("FD with this name already exists")

405|                 **return**

406| ​

407|             **if** getBalance(self.\_username) < amount:

408|                 print("You do not have sufficient balance.")

409|                 **return**

410| ​

411|             c\_changeBalance(self.\_username, -amount)

412|             execute(self.\_QC\_CREATE\_FD, (name, self.\_username, amount,

2, str(currentDate), period,

413|                     currentDate + relativedelta(years=period)))

414|             c\_createUpdate(self.\_username, f"Create {name} FD")

415|             db.commit()

416|             print("FD created successfully.")

417| ​

418|         **def** process(self):

419|             **global** currentState

420| ​

421|             print("======================")

422|             print("(0) Create new FD")

423|             print("(1) Return")

424|             print()

425| ​

426|             option = intInput("(Option) -> ")

427| ​

428|             **if** option == 0:

429|                 print()

430|                 name = input("(Enter FD name) -> ")

431|                 amount = intInput("(Enter amount) -> ")

432|                 period = intInput("(Enter time period in years (under

10)) -> ")

433|                 print()

434| ​

435|                 self.\_createFD(name, amount, period)

436| ​

437|             **elif** option == 1:

438|                 currentState = UnlockedState(self.\_username)

439| ​

440|             **else**:

441|                 print()

442|                 print("Please choose a valid option.")

443| ​

444|     **class** ViewFDState:

445|         \_Q\_GET\_FD\_DETAILS = ("SELECT \* FROM FixedDepo "

446|                              "WHERE username = '{}' AND fdName = '{}';

")

447|         \_QC\_WITHDRAW\_FD = ("UPDATE FixedDepo "

448|                            "SET withdrawn = 1 "

449|                            "WHERE username = '{}' AND fdName = '{}';

")

450|         \_Q\_GET\_ALL\_FDS = ("SELECT fdName FROM FixedDepo "

451|                           "WHERE username = '{}'; ")

452| ​

453|         **def** \_\_init\_\_(self, username : str):

454|             self.\_username = username

455| ​

456|         **def** \_getFDComputedDetails(self, record : tuple):

457|                 passedTimeDelta = relativedelta(currentDate,

record[4])

458|                 yearsPassed = int(passedTimeDelta.years +

(passedTimeDelta.months / 12) + (passedTimeDelta.days

/ 365.25))

459|                 matured = **False** **if** yearsPassed < record[5] **else** **True**

460|                 value = (record[2] \* record[3] \* (record[5] **if** matured

**else** yearsPassed) / 100) + record[2]

461| ​

462|                 **return** (yearsPassed, matured, value)

463| ​

464|         **def** \_printFD(self, fdName : str) -> **None**:

465|             **if** **not** checkFDExists(self.\_username, fdName):

466|                 print("FD with this name does not exist.")

467|                 **return**

468|

469|             execute(self.\_Q\_GET\_FD\_DETAILS, (self.\_username, fdName))

470|             record = crsr.fetchone()

471|             computedDetails = self.\_getFDComputedDetails(record)

472| ​

473|             print(f"Principal : {record[2]}")

474|             print(f"Interest : {record[3]}")

475|             print(f"Created : {record[4]}")

476|             print(f"Total time period (years) : {record[5]}")

477|             print(f"Time passed (years) : {computedDetails[0]}")

478|             print(f"Current value : {computedDetails[2]}")

479|             print(f"Mature date : {record[6]}")

480|             print(f"Matured? : {'Yes' **if** computedDetails[1] **else**

'No'}")

481|             print(f"Widthdrawn? : {'Yes' **if** record[7] **else** 'No'}")

482| ​

483|         **def** \_withdrawFD(self, fdName : str) -> **None**:

484|             **if** **not** checkFDExists(self.\_username, fdName):

485|                 print("FD with this name does not exist.")

486|                 **return**

487| ​

488|             execute(self.\_Q\_GET\_FD\_DETAILS, (self.\_username, fdName))

489|             record = crsr.fetchone()

490| ​

491|             **if** record[7]:

492|                 print("You have already withdrawn this FD.")

493|                 **return**

494|

495|             computedDetails = self.\_getFDComputedDetails(record)

496|             execute(self.\_QC\_WITHDRAW\_FD, (self.\_username, fdName))

497|             c\_changeBalance(self.\_username, computedDetails[2])

498|             c\_createUpdate(self.\_username, f"Withdrew amount

{computedDetails[2]} from FD {fdName}.")

499| ​

500|             db.commit()

501| ​

502|             print(f"Withdrew amount {computedDetails[2]} from FD

{fdName}.")

503|

504|         **def** process(self):

505|             **global** currentState

506| ​

507|             # display FDs

508| ​

509|             print("=============================")

510|             print("(0) Show all FDs")

511|             print("(1) View details of a particular FD")

512|             print("(2) Withdraw an FD")

513|             print("(3) Return")

514|             print()

515| ​

516|             option = intInput("(Option) -> ")

517| ​

518|             **if** option == 0:

519|                 execute(self.\_Q\_GET\_ALL\_FDS, (self.\_username,))

520|                 fdNames = crsr.fetchall()

521| ​

522|                 **if** **not** resultExists(fdNames):

523|                     print("You don't have any FDs yet.")

524|                     **return**

525|

526|                 **for** fdName **in** fdNames:

527|                     print(fdName[0])

528| ​

529|             **elif** option == 1:

530|                 print()

531|                 fdName = input("(Enter FD name) -> ").strip()

532|                 print()

533|                 self.\_printFD(fdName)

534| ​

535|             **elif** option == 2:

536|                 print()

537|                 fdName = input("(Enter FD name) -> ").strip()

538|                 print()

539|                 self.\_withdrawFD(fdName)

540| ​

541|             **elif** option == 3:

542|                 currentState = UnlockedState(self.\_username)

543| ​

544|             **else**:

545|                 print()

546|                 print("Please choose a valid option.")

547| ​

548|     **class** ViewUpdatesState:

549|         **def** \_\_init\_\_(self, username):

550|             self.\_username = username

551| ​

552|         **def** \_displayUpdates(self, updates):

553|             **if** **not** resultExists(updates) :

554|                 print("You have no updates for the requested query.")

555|                 **return**

556| ​

557|             # sort updates from most recent to last

558|             updates.sort(key = **lambda** x: x[2])

559|             **for** index, update **in** enumerate(updates):

560|                 baseContent, extraContent, updateDate = update

561|                 print()

562|                 print(f"({index}): {baseContent}")

563|                 print(f"Date: {updateDate}")

564|                 print(f"Comment: {extraContent}")

565| ​

566|         **def** process(self):

567|             **global** currentState

568| ​

569|             print("=============================")

570|             print("(0) View all updates")

571|             print("(1) View all updates for a day")

572|             print("(2) Return")

573|             print()

574| ​

575|             option = intInput("(Option) -> ")

576| ​

577|             **if** option == 0:

578|                 updates = getUpdates(self.\_username)

579|                 self.\_displayUpdates(updates)

580| ​

581|             **elif** option == 1:

582|                 inp = input("(Required date, in YYYY-MM-DD format) ->

")

583| ​

584|                 **try**:

585|                     date = dt.date.fromisoformat(inp)

586|                     updates = getUpdates(self.\_username, date)

587|                     self.\_displayUpdates(updates)

588| ​

589|                 **except** ValueError:

590|                     print("Invalid date.")

591| ​

592|             **elif** option == 2:

593|                 currentState = UnlockedState(self.\_username)

594| ​

595|             **else**:

596|                 print("Please choose a valid option.")

597| ​

598|     **if** \_\_name\_\_ == '\_\_main\_\_':

599|         currentState = LockedState()

600| ​

601|         \_Q\_GETDBCREATIONDATETIME = ("SELECT DBCreationDateTime "

602|                                     "FROM EnvInfo ;")

603| ​

604|         # Get the date and time when we created the database

605|         execute(\_Q\_GETDBCREATIONDATETIME, ())

606| ​

607|         creationDateTime = crsr.fetchone()[0]

608|         creationTime = creationDateTime.timestamp()

609|         creationDate = creationDateTime.date()

610| ​

611|         previousTime = creationTime

612|         currentDate = creationDate

613| ​

614|         **while** **True**:

615|             currentTime = time.time()

616|             elapsedDays = (currentTime - previousTime) // TIMEDELTA

617|             currentDate += dt.timedelta(days=elapsedDays)

618| ​

619|             currentState.process()

620|             previousTime = currentTime

621| ​

622| **except** (DataError, DatabaseError, OperationalError, NotSupportedError,

IntegrityError, ProgrammingError, InternalError) **as** e:

623|     print("DB Error!", e)

624| ​

625| **except** KeyboardInterrupt:

626|     EXIT(0)

627| ​

628| **except** Exception **as** e:

629|     print("ERROR: ", e)

630|     EXIT(1)

**SAMPLE**

**OUTPUT**

Enter username and password to view details or create a new account

(1) Login

(2) Create an account

(3) Quit

​

(Option) -> 2

========================================

(0) Create account

(1) Abort

​

(Option) -> 0

​

(Enter NEW Username) -> ng

(Enter NEW Password) -> ng@2007

(Enter password for confirmation) ->

(Enter first name) -> nandan

(Enter last name) -> goyal

(Enter age) -> 19

(Enter phone no.) -> 9837461850

​

===================================

2025-01-01

BALANCE: 0

​

(0) Logout

(1) Pay

(2) Deposit

(3) Create a fixed deposit

(4) Modify/View fixed deposits

(5) View all updates for your account

​

(Option) -> 2

======================================================================

(Enter amount to deposit (cash to digital money)) -> 5000

===================================

2025-01-10

BALANCE: 5000

​

(0) Logout

(1) Pay

(2) Deposit

(3) Create a fixed deposit

(4) Modify/View fixed deposits

(5) View all updates for your account

​

(Option) -> 3

======================

(0) Create new FD

(1) Return

​

(Option) -> 0

​

(Enter FD name) -> fd1

(Enter amount) -> 1000

(Enter time period in years (under 10)) -> 5

​

FD created successfully.

======================

​

(0) Create new FD

(1) Return

​

(Option) -> 1

===================================

2025-01-22

BALANCE: 4000

​

(0) Logout

(1) Pay

(2) Deposit

(3) Create a fixed deposit

(4) Modify/View fixed deposits

(5) View all updates for your account

​

(Option) -> 4

=============================

(0) Show all FDs

(1) View details of a particular FD

(2) Withdraw an FD

(3) Return

​

(Option) -> 0

fd1

=============================

(0) Show all FDs

(1) View details of a particular FD

(2) Withdraw an FD

(3) Return

​

(Option) -> 1

​

(Enter FD name) -> fd1

​

Principal : 1000

Interest : 2

Created : 2025-01-17

Total time period (years) : 5

Time passed (years) : 0

Current value : 1000.0

Mature date : 2030-01-17

Matured? : No

Widthdrawn? : No

=============================

(0) Show all FDs

(1) View details of a particular FD

(2) Withdraw an FD

(3) Return

​

(Option) -> 3

===================================

​

​

​

​

​

​

​

​

2025-02-02

BALANCE: 4000

​

(0) Logout

(1) Pay

(2) Deposit

(3) Create a fixed deposit

(4) Modify/View fixed deposits

(5) View all updates for your account

​

(Option) -> 5

=============================

(0) View all updates

(1) View all updates for a day

(2) Return

​

(Option) -> 0

​

(0): Deposit 5000

Date: 2025-01-05

Comment: No comment

​

(1): Create fd1 FD

Date: 2025-01-17

Comment: No comment

=============================

(0) View all updates

(1) View all updates for a day

(2) Return

​

(Option) -> 2

===================================

2025-02-10

BALANCE: 4000

​

(0) Logout

(1) Pay

(2) Deposit

(3) Create a fixed deposit

(4) Modify/View fixed deposits

(5) View all updates for your account

​

(Option) -> 0

======================================================================

Enter username and password to view details or create a new account

(1) Login

(2) Create an account

(3) Quit

​

(Option) -> 2

========================================

(0) Create account

(1) Abort

​

(Option) -> sg

Invalid input.

(Option) -> 0

​

​

(Enter NEW Username) -> sg

(Enter NEW Password) -> sg@2007

(Enter password for confirmation) ->

(Enter first name) -> satwik

(Enter last name) -> gupta

(Enter age) -> 18

(Enter phone no.) -> 9487468553

​

===================================

2025-02-27

BALANCE: 0

​

(0) Logout

(1) Pay

(2) Deposit

(3) Create a fixed deposit

(4) Modify/View fixed deposits

(5) View all updates for your account

​

(Option) -> 2

======================================================================

(Enter amount to deposit (cash to digital money)) -> 10000

===================================

2025-03-01

BALANCE: 10000

​

(0) Logout

(1) Pay

(2) Deposit

(3) Create a fixed deposit

(4) Modify/View fixed deposits

(5) View all updates for your account

​

(Option) -> 1

===========================

(0) Pay to another user

(1) Abort

​

(Option) -> 0

​

(Enter username of receiver) -> ng

(Enter amount to pay) -> 2000

Enter comment (optional)) -> first payment

​

(Enter password to proceed with payment) ->

Transaction made successfully.

===========================

(0) Pay to another user

(1) Abort

​

(Option) -> 1

===================================

2025-03-12

BALANCE: 8000

​

(0) Logout

(1) Pay

(2) Deposit

(3) Create a fixed deposit

(4) Modify/View fixed deposits

(5) View all updates for your account

​

(Option) -> 5

=============================

(0) View all updates

(1) View all updates for a day

(2) Return

​

(Option) -> 0

​

(0): Deposit 10000

Date: 2025-02-28

Comment: No comment

​

(1): Paid 2000 to nandan

Date: 2025-03-03

Comment: first payment

=============================

(0) View all updates

(1) View all updates for a day

(2) Return

​

(Option) -> 2

===================================

2025-03-20

BALANCE: 8000

​

(0) Logout

(1) Pay

(2) Deposit

(3) Create a fixed deposit

(4) Modify/View fixed deposits

(5) View all updates for your account

​

(Option) -> 0

======================================================================

Enter username and password to view details or create a new account

(1) Login

(2) Create an account

(3) Quit

​

(Option) -> 1

=======================================

(Enter Username) -> ng

(Enter Password) ->

​

Logged in successfully.

===================================

2025-03-22

BALANCE: 6000

​

(0) Logout

(1) Pay

(2) Deposit

(3) Create a fixed deposit

(4) Modify/View fixed deposits

(5) View all updates for your account

​

(Option) -> 5

=============================

(0) View all updates

(1) View all updates for a day

(2) Return

​

(Option) -> 0

​

(0): Deposit 5000

Date: 2025-01-05

Comment: No comment

​

(1): Create fd1 FD

Date: 2025-01-17

Comment: No comment

​

(2): satwik paid 2000

Date: 2025-03-03

Comment: first payment

=============================

(0) View all updates

(1) View all updates for a day

(2) Return

​

(Option) -> 2

===================================

2025-04-01

BALANCE: 6000

​

(0) Logout

(1) Pay

(2) Deposit

(3) Create a fixed deposit

(4) Modify/View fixed deposits

(5) View all updates for your account

​

(Option) -> 5

=============================

(0) View all updates

(1) View all updates for a day

(2) Return

​

(Option) -> 0

​

​(0): Deposit 5000

Date: 2025-01-05

Comment: No comment

​

(1): Create fd1 FD

Date: 2025-01-17

Comment: No comment

​

(2): satwik paid 2000

Date: 2025-03-03

=============================

(0) View all updates

(1) View all updates for a day

(2) Return

​

(Option) -> 2

===================================

2040-04-19

BALANCE: 6000

​

(0) Logout

(1) Pay

(2) Deposit

(3) Create a fixed deposit

(4) Modify/View fixed deposits

(5) View all updates for your account

​

(Option) -> 4

=============================

(0) Show all FDs

(1) View details of a particular FD

(2) Withdraw an FD

(3) Return

​

(Option) -> 1

​

(Enter FD name) -> fd1

​

Principal : 1000

Interest : 2

Created : 2025-01-17

Total time period (years) : 5

Time passed (years) : 15

Current value : 1100.0

Mature date : 2030-01-17

Matured? : Yes

Widthdrawn? : No

=============================

(0) Show all FDs

(1) View details of a particular FD

(2) Withdraw an FD

(3) Return

​

(Option) -> 2

​

(Enter FD name) -> fd1

​

Withdrew amount 1100.0 from FD fd1.

=============================

(0) Show all FDs

(1) View details of a particular FD

(2) Withdraw an FD

(3) Return

​

(Option) -> 3

===================================

2040-08-22

BALANCE: 7100

​

(0) Logout

(1) Pay

(2) Deposit

(3) Create a fixed deposit

(4) Modify/View fixed deposits

(5) View all updates for your account

​

**BIBLIOGRAPHY**

* Sumita Arora Textbook Computer Science with Python Class XII
* Various online resources