Term Project: Smart University Card System

The Smart University Card is a multifunctional identification and access card for students, faculty members, and staff. This system combines identification, access control, payment, attendance tracking, and various campus services into one secure and easy-to-use medium. Such system enhances convenience, security, and operational efficiency within the university and provides reliable and smooth method to ease the student-related and faculty-related functions.

The technology used in university card systems is evolving from traditional magnetic stripe cards, to physical cards, and finally to digital identities on personal devices such as mobile credentials.

Through this project, we aim to design, analysis, implement, and test a simple system that fulfils basic functionalities for three primary types of users: faculty members, students, and administrator. The involved card acts as a single credential used by student for accessing buildings, borrowing library materials, paying for meals, attending classes, printing documents, and participating in campus events. Likewise, the faculty member can use the card to access parking slots in campus buildings and generate attendance report for classes. It also supports card and transaction management functions for system administrator.

The system's database maintains **<u>I/O files</u>** (not temporary lists or arrays) for all involved students, faculty members, cards, transactions, and lecture attendance. This information must be permanent and is accessed by different users to achieve their desired functionalities and be updated continuously throughout the running time of the system. Every **<u>student</u>** has the following information: user ID (unique), name, and list of registered courses. Whereas every **<u>faculty member</u>** has user ID (unique), name, and list of taught courses. Regarding the **<u>card</u>**, it has the following information: number (unique), balance, status (blocked or unblocked), user ID, and type (student or faculty member). For every **<u>transaction</u>** entry, the system records the following information: ID (unique), card number, type (recharge, attendance, or payment), and amount (for recharge and payment only). **<u>Attendance</u>** record is identified by: course ID (unique), date, and list of attendees (students' user IDs).

As a prerequisite, your implementation program must initially load the database with a set of students, faculty members, and cards at the beginning of first running and reflect any changes in future. The following data must be loaded as is upon running your program for the first time:

[**STUDENTS**]

User ID	Name	Registered Courses
S01	Ali	CPE100,SE400
S02	Omar	CPE100,NES200
S03	Reem	NES200,CIS300,SE400
S04	Maher	CPE100,SE400

[**CARDS**]

Number	Balance	Type	Status	User ID
10	80	faculty member	unblocked	F02
20	110	student	unblocked	S02
30	95	student	blocked	S03
40	160	student	unblocked	S04

[**FACULTY MEMBERS**]

User ID	Name	Taught Courses
F01	Sami	CPE100, CIS300
F02	Eman	NES200, SE400

The system has only one administrator that logs in the system without user name and password.

Requirement Specifications

The two types of smart card holders (student and faculty member) access the system by tapping their cards at card readers that are located at different points where services are offered to them inside the university campus, then they get what they want. However, we are to simulate the operations of the system by developing console application that acts as a platform to system's users: student, faculty member, and administrator. So, the card holder must access the system using card number, then reading and writing data to console screen.

Administrator is responsible for issuing cards for both students and faculty members. Also he can block or unblock cards in case of emergent conditions. Once the card is blocked, the card holder can't use it at all and the system must notify the card holder when he try to log in. Hence, the system will not display options for the card holder.

Student can record his attendance to one of the registered courses on different dates. On the other side, the faculty member can generate an attendance report that contains all attendees for a specific taught course on specific date.

The cafeteria of the university presents the following menu:

[**CAFETERIA MENU**]

ITEM	PRICE
Steak	8
Soup	2
Sandwich	3
Salad	4

ITEM	PRICE
Tea	2
Juice	3
Cake	5
Water	1

This menu will appear when student chooses "Pay for cafeteria" service such that he can select as many items as he wants for every payment.

Regarding the payment for bus ride, the system offers 3 round-trip tracks inside the university campus. Student can select one track for each time.

Track 1→ origin: main gate destination: Northern buildings cost: 3 JD
Track 2→ origin: main gate destination: Southern buildings cost: 4 JD
Track 3→ origin: main gate destination: Library cost: 5 JD

The system offers a car parking service to faculty members. This service calculates the cost based on duration as follows:
1st hour: 5 JD
2nd hour: 4 JD
3rd hour: 3 JD
4th hour: 2 JD
5th hour: 1 JD
Above 5 hours for free. Example: a 3-hour parking costs 12 JD

For each time a card holder makes a payment, the system must check if the card has enough balance to proceed the payment, else the system must deny the payment and notify the student or faculty member.

The system is required to satisfy the main functional requirements for all users (student, faculty member, and administrator). This necessitate to explore high-level use cases then decomposing them into sub use cases to draw a full image of the involved implementation steps. Consequently, our system has the following high-level use cases:

Student functions

1. Recharge card
2. Record lecture attendance
3. Pay for cafeteria
4. Pay for bus ride
5. View transaction history

Faculty Member functions

6. Recharge card
7. Access car parking
8. Generate attendance report

Administrator functions

9. Issue card
10. Block card
11. Unblock card
12. View all cards
13. View all transactions

YOUR PROGRAM MUST SHOW ONLY THE FUNCTIONS RELATED TO THE CURRENTLY LOGGED-IN USER. THEREFORE, UPON RUNNING YOUR PROGRAM, THE SYSTEM MUST DISPLAY TWO LOGIN OPTIONS (LOGIN SCREEN): AS ^{1}ADMIN, ^{2}OR AS CARD HOLDER.

IF CARD HOLDER IS SELECTED, THEN SYSTEM MUST DISPLAY TWO TYPES OF HOLDERS: STUDENT OR FACULTY MEMBER. BASED ON THE SELECTED USER TYPE (ADMIN, STUDENT, OR FACULTY MEMBER), THE SYSTEM MUST DISPLAY THE OPTIONS RELATED TO THE LOGGED-IN USER.
THE OPTIONS MENU FOR THE THREE USER TYPES MUST PROVIDE LOGOUT OPTION TO RETURN TO THE LOGIN SCREEN TO ENABLE LOGGING IN WITH DIFFERENT USER TYPE.

In details, below are the involved steps for each use case:

Use case 1: Recharge card

- The system displays full information of the card.
- The system asks the card holder to enter new balance.
- The system updates the card balance and display the updated balance (old balance + new balance) on screen.
- The system asks the card holder to enter transaction ID.
- The system creates transaction record, sets transaction type to “recharge” and transaction amount to the new balance.
- The system adds the transaction record to “transactions” file.

Use case 2: Record lecture attendance

- The system displays all courses the currently logged-in student registers in.
- The system asks the student to enter course ID.
- The system asks the student to enter the date of lecture.
- The system **either** adds user ID of the student to list of attendees on that date (if there is an attendance record in “attendance” file with the same course ID and date), **or** creates new attendance record, adds user ID of the student to list of attendees, and finally adds the record to “attendance” file.
- The system creates transaction record, sets transaction type to “attendance” and transaction amount to “N/A”.
- The system asks student to enter transaction ID.
- The system adds the transaction record to “transactions” file.

Use case 3: Pay for cafeteria

- The system displays the food items in the menu along with prices, and asks student to enter what he wants to eat and drink.
- The student chooses as many items as he wants from the menu.
- The system creates transaction record, sets transaction type to “payment”, and calculates transaction amount.
- The system asks student to enter transaction ID.
- The system adds the transaction record to “transactions” file.

Use case 4: Pay for bus ride

- The system displays available bus tracks along with costs, and asks student to enter one track.
- The system creates transaction record, sets transaction type to “payment”, and calculates transaction amount.
- The system asks student to enter transaction ID.
- The system adds the transaction record to “transactions” file.

Use case 5: View transaction history

- The system displays full information of all transactions related to the currently logged-in student, sorted by transaction type.

Use case 7: Access car parking

- The system displays parking fees based on duration, and asks faculty member to enter number of hours.
- The system creates transaction record, sets transaction type to “payment”, and calculates transaction amount.
- The system asks faculty member to enter transaction ID.
- The system adds the transaction record to “transactions” file.

Use case 8: Generate attendance report

- The system displays ID of all courses taught by the currently logged-in faculty member.
- The system asks faculty member to enter course ID.
- The system asks faculty member to enter date.
- The system displays all attendees for course on the entered date.

Use case 9: Issue card

- The system asks administrator to enter the following information: card number, card type, and user ID.
- The system creates card record and sets status to “unblocked” and balance to 50.
- The system saves the card record to “cards” file.

Use case 10: Block card

- The system displays full information of all unblocked cards.
- The system asks administrator to enter card number.
- The system changes the status of this card to “blocked”.

Use case 11: Unblock card

- The system displays full information of all blocked cards.
- The system asks administrator to enter card number.
- The system changes the status of this card to “unblocked”.

Use case 12: View all cards

- The system displays full information of all cards, sorted by card type.

Use case 13: View all transactions

- The system displays full information of all transactions, sorted by transaction type.

You are required to satisfy and implement every step above, else your grade will be affected accordingly. Please don’t assume anything even if it makes sense. In case you do not understand something, please feel free to ask about it.