Payment Application

Development environment preparation

CRITERIA	MEETS SPECIFICATIONS
Create modules folders	1. Create a new project 2. Create "Application" folder 3. Create "Card" folder 4. Create "Terminal" folder 5. Create "Server" folder Note: To create a folder in Microsoft Visual Studio 1. In the solution explorer, right click on the project name 2. Go to Add 3. Select New Folder 4. Give a name to that folder You should deliver a screenshot of the solution explorer that clarifies your folder structure.
Create .c and .h file for each module	1. In the "Application" folder create app.c and app.h files 2. In the "Card" folder create card.c and card.h files 3. In the "Terminal" folder create terminal.c and terminal.h files 4. In the "Server" folder create server.c and server.h files. Note: To create a file into a folder in Microsoft Visual Studio 1. In the solution explorer, right click on the folder you want 2. Go to Add 3. Select New item 4. Choose the file type or .h 5. If a .cpp is chosen, change the extension to .c 6. Give a name to that file" You should deliver a screenshot of the solution explorer that clarifies files in each folder.
Add header file guard	1. In the app.h file add the header file guard 2. In the card.h file add the header file guard 3. In the terminal.h file add the header file guard 4. In server.h file add the header file guard You should deliver a screenshot for each .h file, file name must appear in the screenshot and the header file guard

Implement the card module

CRITERIA	MEETS SPECIFICATIONS

Fill in card.h file with functions' prototypes and typedefs	Use the following prototypes as is:
	
	1. EN_cardError_t getCardHolderName(ST_cardData_t *cardData);
	2. EN_cardError_t getCardExpiryDate(ST_cardData_t *cardData);
	3. EN_cardError_t getCardPAN(ST_cardData_t *cardData);
	
	Use the following typedef as is:
	typedef struct ST_cardData_{
	{
	uint8_t cardHolderName[25];
	uint8_t primaryAccountNumber[20];
	uint8_t cardExpirationDate[5];
	}ST_cardData_t;
	
	typedef enum EN_cardError_{
	{
	CARD_OK, WRONG_NAME, WRONG_EXP_DATE, WRONG_PAN
	}EN_cardError_t;
	
	You should deliver a screenshot for your card.h file

Implement getCardHolderName function	1. This function will ask for the cardholder's name and store it into card data. 2. Card holder name is 24 alphabetic characters string max and 20 min. 3. If the cardholder name is NULL, less than 20 characters or more than 24 will return WRONG_NAME error, else return CARD_OK. You should deliver a maximum 2min video to discuss your implementation and run different test cases on this function
Implement getCardExpiryDate function	1. This function will ask for the card expiry date and store it in card data. 2. Card expiry date is 5 characters string in the format "MM/YY" e.g. "05/06". 3. If the card expiry date is NULL, less or more than 5 characters, or has the wrong format will return WRONG_EXP_DATE error, else return CARD_OK. You should deliver a maximum 2min video to discuss your implementation and run different test cases on this function
Implement getCardPAN function	1. This function will ask for the card's Primary Account Number and store it in card data. 2. PAN is 19 numbers string in the format "8989374615436894037" for example. 3. If the PAN is NULL, less than 16 or more than 19 characters, or has the wrong format will return WRONG_PAN error, else return CARD_OK. You should deliver a maximum 2min video to discuss your implementation and run different test cases on this function

Implement the terminal module

CRITERIA	MEETS SPECIFICATIONS

Fill in terminal.h file with functions' prototypes and typedefs	Use the following prototypes as is:
	
	1. EN_terminalError_t getTransactionDate(ST_terminalData_t *termData);
	2. EN_terminalError_t isCardExpired(ST_cardData_t cardData, ST_terminalData_t *termData);
	3. EN_terminalError_t isValidCardPAN(ST_cardData_t *cardData);
	4. EN_terminalError_t getTransactionAmount(ST_terminalData_t *termData);
	5. EN_terminalError_t isBelowMaxAmount(ST_terminalData_t *termData);
	6. EN_terminalError_t setMaxAmount(ST_terminalData_t *termData);
	
	Use the following typedef as is:
	typedef struct ST_terminalData_{
	{
	float transAmount;
	float maxTransAmount;
	uint8_t transactionDate[11];
	}ST_terminalData_t;
	
	typedef enum EN_terminalError_{
	{
	TERMINAL_OK, WRONG_DATE, EXPIRED_CARD, INVALID_CARD, INVALID_AMOUNT, EXCEED_MAX_AMOUNT, INVALID_MAX_AMOUNT
	}EN_terminalError_t;
	
	You should deliver a screenshot for your terminal.h file

Implement getTransactionDate function	1. This function will ask for the transaction date and store it in terminal data. 2. Transaction date is 10 characters string in the format DD/MM/YYYY. e.g. 25/06/2022. 3. If the transaction date is NULL, less than 10 characters or wrong format will return WRONG_DATE error, else return TERMINAL_OK. Optional: The function will read the current date from your computer and store it into terminal data with the mentioned size and format. You should deliver a maximum 2min video to discuss your implementation and run different test cases on this function
Implement isCardExpired function	1. This function compares the card expiry date with the transaction date. 2. If the card expiration date is before the transaction date will return EXPIRED_CARD, else return TERMINAL_OK. You should deliver a maximum 2min video to discuss your implementation and run different test cases on this function
Implement getTransactionAmount function	1. This function asks for the transaction amount and saves it into terminal data. 2. If the transaction amount is less than or equal to 0 will return INVALID_AMOUNT, else return OK. You should deliver a maximum 2min video to discuss your implementation and run different test cases on this function
Implement isBelowMaxAmount function	1. This function compares the transaction amount with the terminal max amount. 2. If the transaction amount is larger than the terminal max amount will return EXCEED_MAX_AMOUNT, else return TERMINAL_OK. You should deliver a maximum 2min video to discuss your implementation and run different test cases on this function
Implement setMaxAmount function	1. This function sets the maximum allowed amount into terminal data. 2. Transaction max amount is a float number. 3. If transaction max amount less than or equal to 0 will return INVALID_MAX_AMOUNT error, else return TERMINAL_OK. You should deliver a maximum 2min video to discuss your implementation and run different test cases on this function

Implement the server module

CRITERIA	MEETS SPECIFICATIONS

Fill in server.h file with functions' prototypes and typedefs	Use the following prototypes as is:
	
	1. EN_transState_t receiveTransactionData(ST_transaction_t *transData);
	2. EN_serverError_t isValidAccount(ST_cardData_t cardData, ST_accountDB_t *accountReference);
	3. EN_serverError_t isBlockedAccount(ST_accountDB_t *accountReference);
	4. EN_serverError_t isAmountAvailable(ST_terminalData_t *termData);
	5. EN_serverError_t saveTransaction(ST_transaction_t *transData);
	6. EN_serverError_t getTransaction(uint32_t transactionSequenceNumber, ST_transaction_t *transData);
	
	Use the following typedef as is:
	
	typedef enum EN_transState_{
	{
	APPROVED, DECLINED_INSUFFICIENT_FUND, DECLINED_STOLEN_CARD, INVALID_CARD, INTERNAL_SERVER_ERROR
	}EN_transState_t;
	
	typedef struct ST_transaction_{
	{
	ST_cardData_t cardHolderData;
	ST_terminalData_t terminalData;
	EN_transState_t transState;
	uint32_t transactionSequenceNumber;
	}ST_transaction;
	
	typedef enum EN_serverError_{
	{
	SERVER_OK, SAVING_FAILED, TRANSACTION_NOT_FOUND, ACCOUNT_NOT_FOUND, LOW_BALANCE, BLOCKED_ACCOUNT
	}EN_serverError_t;
	
	typedef enum EN_accountState_{
	{
	RUNNING,
	BLOCKED
	}EN_accountState_t;
	
	typedef struct ST_accountDB_{
	{
	float balance;
	EN_accountState_t state;
	uint8_t primaryAccountNumber[20];
	}ST_accountDB_t;
	
	You should deliver a screenshot for your server.h file.

Implement server side accounts' database	1. Create a global array of ST_accountDB_t for the valid accounts database 2. Fill in the array initially with very valid data 3. This array has a maximum of 255 elements/transaction data You should deliver a screenshot of your transaction database array.
Implement receiveTransactionData function	1. This function will take all transaction data and validate its data. 2. It checks the account details and amount availability: 3 If the account does not exist return FRAUD_CARD, If the amount is not available will return DECLINED_INSUFFICIENT_FUND, If the account is blocked will return DECLINED_STOLEN_CARD, if a transaction can't be saved will return INTERNAL_SERVER_ERROR and will not save the transaction, else return APPROVED. 4. It will update the database with the new balance. You should deliver a maximum 2min video to discuss your implementation and run different test cases on this function
Implement isValidAccount function	1. This function will take card data and validate if the account related to this card exists or not. 2. It checks if the PAN exists or not in the server's database (searches for the card PAN in the DB). 3 If the PAN doesn't exist will return ACCOUNT_NOT_FOUND, else will return SERVER_OK and return a reference to this account in the DB. You should deliver a maximum 2min video to discuss your implementation and run different test cases on this function
Implement isAmountAvailable function	1. This function will take terminal data and validate these data. 2. It checks if the transaction's amount is available or not. 3 If the transaction amount is greater than the amount in the database will return LOW_BALANCE, else will return SERVER_OK. You should deliver a maximum 2min video to discuss your implementation and run different test cases on this function
Implement saveTransaction function	1. This function will store all transaction data in the transactions database. 2. It gives a sequence number to a transaction, this number is incremented once a transaction is processed into the server, you must check the last sequence number in the server to give the new transaction a new sequence number. 3. It saves the transaction sequence number, PAN, card data, terminal data, and the transaction state. 4 If the transaction can't be saved, for any reason (ex: dropped connection) will return SAVING_FAILED, else will return SERVER_OK, you can simulate this by commenting on the lines you where your code writes the transaction data in the database. 5. It checks if the transaction is saved or not using the getTransaction function. You should deliver a maximum 2min video to discuss your implementation and run different test cases on this function
Implement getTransaction function	1. This function takes the sequence number of a transaction and returns the transaction data if found in the transactions DB. 2. If the sequence number is not found, then the transaction is not found, the function will return TRANSACTION_NOT_FOUND, else return transaction data as well as SERVER_OK You should deliver a maximum 2min video to discuss your implementation and run different test cases on this function

Implement the application

CRITERIA	MEETS SPECIFICATIONS
Fill in application.h file with functions' prototypes	Use the following prototypes as is: void appStart(void); You should deliver a screenshot for your application.h file.
Implement appStart function	Please refere to the flow chart attached under the instructions video in order to implement this application. You should deliver: 1. All project folders and files 2. Video record where you will discuss your implementation (maximum 3min)

Testing the application

CRITERIA	MEETS SPECIFICATIONS
Transaction approved user story	As a bank customer have an account and has a valid and not expired card, I want to withdraw an amount of money less than the maximum allowed and less than or equal to the amount in my balance, so that I am expecting that the transaction is approved and my account balance is reduced by the withdrawal amount. You should deliver a video for testing this user story: 1. Mention test data you are using. 2. Test result must be clear - Is it passed or failed
Exceed the maximum amount user story	As a bank customer have an account, that has a valid and not expired card, I want to withdraw an amount of money that exceeds the maximum allowed amount so that I am expecting that the transaction declined. You should deliver a video for testing this user story: 1. Mention test data you are using. 2. Test result must be clear - Is it passed or failed
Insufficient fund user story	As a bank customer have an account and has a valid and not expired card, I want to withdraw an amount of money less than the maximum allowed amount but greater than the amount in my balance so that I am expecting that the transaction declined. You should deliver a video for testing this user story: 1. Mention test data you are using. 2. Test result must be clear - Is it passed or failed
Expired card user story	As a bank customer have an account, have a valid but expired card, I want to withdraw an amount of money so that I expect that the transaction declined. You should deliver a video for testing this user story: 1. Mention test data you are using. 2. Test result must be clear - Is it passed or failed
Invalid card user story	As a bank customer have an account and has a valid and not expired stolen card, I want to block anyone from using my card so that I am expecting that any transaction made by this card is declined. You should deliver a video for testing this user story: 1. Mention test data you are using. 2. Test result must be clear - Is it passed or failed

Suggestions to Make Your Project Stand Out!

In getCardPAN function:

Give PAN that is a Luhn number, Luhn number generator, and checker

In terminal implement isValidCard function:

1. This function checks if the PAN is a Luhn number or not.
2. If PAN is not a Luhn number will return INVALID_CARD, else return CARD_OK.

In server isBlockedAccount function:

1. This function will take a reference to an existing account in the database.
2. It checks if the account is blocked or not.
3. If the account is blocked, will return BLOCKED_ACCOUNT, else will return SERVER_OK.

Server side accounts DB:

1. Instead of a global array create a text file "Accounts.DB.txt" that stores all account data and read this file into your application
2. Instead of a global array create a text file "Transactions.DB.txt" where you will save all transactions and read if you need

Server side transactions DB:

1. Instead of a global array create a text file "Transactions.DB.txt" where you will save all transactions and read if you need

Fraud card user story:

As a bank administrator, I want to issue my own cards, so that I am expecting that any transaction made by any fraud card (failed in Luhn check) is declined.