**PROJECT SPECIFICATION**

**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 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. Select file type, .cpp 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\_t {  uint8\_t cardHolderName[25];  uint8\_t primaryAccountNumber[20];  uint8\_t cardExpirationDate[6]; } ST\_cardData\_t;  typedef enum EN\_cardError\_t {  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. |
| 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 "05/25". 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. |
| Implement getCardPAN function | 1. This function will ask for the card's Primary Account Number and store it in card data. 2. PAN is 20 numeric characters string, 19 character max, and 16 character min. 3. If the PAN is NULL, less than 16 or more than 19 characters, 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 all functions |

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\_t {  float transAmount;  float maxTransAmount;  uint8\_t transactionDate[11]; } ST\_terminalData\_t;  typedef enum EN\_terminalError\_t {  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 data 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. |
| Implement isCardExpried 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. |
| 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. |
| 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. |
| 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 all functions |

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 recieveTransactionData(ST\_transaction\_t \*transData); 2. EN\_serverError\_t isValidAccount(ST\_cardData\_t *cardData, ST\_accountsDB\_t*accountRefrence); 3. EN\_serverError\_t isBlockedAccount(ST\_accountsDB\_t \*accountRefrence); 4. EN\_serverError\_t isAmountAvailable(ST\_trminalData\_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\_t { APPROVED, DECLINED\_INSUFFECIENT\_FUND, DECLINED\_STOLEN\_CARD, FRAUD\_CARD, INTERNAL\_SERVER\_ERROR }EN\_transStat\_t;  typedef struct ST\_transaction\_t {  ST\_cardData\_t cardHolderData;  ST\_trminalData\_t terminalData;  EN\_transState\_t transState;  uint32\_t transactionSequenceNumber; } ST\_transaction;  typedef enum EN\_serverError\_t { SERVER\_OK, SAVING\_FAILED, TRANSACTION\_NOT\_FOUND, ACCOUNT\_NOT\_FOUND, LOW\_BALANCE, BLOCKED\_ACCOUNT } EN\_serverError\_t ;  typedef enum EN\_accountState\_t {  RUNNING,  BLOCKED } EN\_accountState\_t;  typedef struct ST\_accountsDB\_t {  float balance;  EN\_accountState\_t state;  uint8\_t primaryAccountNumber[20]; } ST\_accountsDB\_t;  You should deliver a screenshot for your server.h file. |
| Implement server-side accounts' database | 1. Create a global array of ST\_accountsDB\_t for the valid accounts database 2. Fill in the array initially with any valid data 3. This array has a maximum of 255 element/account data 4. You can fill up to 10 different accounts for the sake of testing 5. Example of a running account: {2000.0, RUNNING, "8989374615436851"} 6. Example of a blocked account, **its card is stolen**: {100000.0, BLOCKED, "5807007076043875"}   You should deliver a screenshot of your accounts database array with a minimum of at least 5 different accounts for the different test cases, check all needed test cases in the **"Testing the application"** section |
| Implement server-side transactions' database | 1. Create a global array of ST\_transaction\_t 2. Fill in the array initially with 0s. 3. This array has a maximum of 255 element/transaction data   You should deliver a screenshot of your transaction database array |
| Implement recieveTransactionData 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\_INSUFFECIENT\_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 returns 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 balance 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 any type of transactions, APPROVED or DECLINED, with the specific reason for declining/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 withdrawn 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 and larger 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 Luhun check) is declined.