Chat analysis part: (Robert) In the app, as discussed with other teams, users will have the ability to chat with other users that are using the app. In the case of abusive language from a user, the recipient can report that message (this is done by the social team). We will have a database table with chat reports where we get the reported message and the reported user's cnp from. If the reported message is offensive (call an api function to decide if the language used contains swear words) then we will get that user's cnp and lower his credit score by 15. Each time we tax someone's credit score for offensive chats the count of punishments in the users table for this user is incremented. If that person has at least 3 punishments in the past then the credit score will instead drop by that 15 * number of previous punishments.

Subtasks:

1. Connect to an API that can detect bad words (PurgoMalum is a free option https://publicapi.dev/purgo-malum-api)
2. Extract the data from the chat reports table (the reported message and the user's cnp)
3. Analyze the message and find out if there are words in it that are offensive (the api will have a function for this)
4. Check if the user has at least 3 previous offenses or not
5. Lower the user's credit score if the message is offensive by 15 (if he has less than 3 offenses) or by 15 * noOffenses if he has at least 3 / or no further action shall be taken if it's not offensive
6. Update the user's credit score in the credit store history table with the current date. Each time the credit score of a user is modified, in the history table will be added / updated the new credit score. If there exists another credit score for the current day, then the record of the current day from the history table will only be updated with the new score. If there is no record for the current date, then the data will be added in the table.
7. Check if in the activityLog table, the user appears at least once with the activity name "verbal abuse". If he does, update the lastModifiedAmount with however much you decreased now from the credit score. If he doesn't, then make a new entry in that table and put the lastModifiedAmount the amount you lowered the credit score with now.

Bill Splitting analysis: (Bolos Mihai) When a registered user wants to split a bill along with other registered users, a new group is created (this group will have the same structure as the group implemented by the social part team, we will take it from them) with all the users that are part of the bill split. In this group there will be, right next to each user, in the list, the share of their bill in bold and the date when they paid their share in italic, right next to the share. If they did not pay their share yet, the field of the date will be empty (no data is shown). Also the payer user will have the report option, as a button, for each user that did not pay their share. When the user chooses to report the user (press the button), the report is sent and the software analyzes the following data: date of the transaction, owed money, history of transactions of both the reported user and the report initiator. It checks the reported user transaction history from the date the bill split was and when the report is made, to see if the reported user made a transaction with the exact amount of money owed to the user with a description that contains any of the words "bill" or "share", but the payment is not made through the bill splitting group. After the check, a gravity factor will be computed (which is a score from 0 to 100) where half of it is based on the time since the share was supposed to be paid (1 day - 3 weeks). This is an interval from 0 to 100, taken proportionally from the 1 - 21 days (so 10 days = 25 gravity score as an example, and anything over 21 days is considered maximum and will get a 50 score). The other half of the risk factor is based on the money owed(1 currency - 1000 currency). The same logic applies here, where this is proportionally from the 1 - 1000 (for example 500 currency is 25 gravity score), and anything over is considered maximum from this interval.

GravityFactor = 1/2(min(50,(T-1) * 50 /20)+min(50,(M - 1) * 50/999))
T = number of days past due
M = amount of money owed

The software will also check to see if the user currently has enough money to pay their share or if the sum of the outgoing transactions made since the report was initiated is greater than their share of the

bill (if the reported user couldn't pay their share, even if they wanted to, the gravity factor is decreased by 10%). Also, the software will check if the user paid their shares of other bills at least on 3 different occasions in the past (-5% on the gravity factor), if the user made at least 5 transaction in the previous month to the reporter (-5% on the gravity factor because this might indicate they are friends) and the number of offences(increase the gravity factor with the floor ot 10% of the number of offences (eg: if they have 50 offences +floor(10% of 50)% = +5% to the gravity factor)). These checks will help the software understand the report and make a calculated decision on how much to decrease the credit score. When the user pays their share or 21 days have passed (the maximum time allowed for the user to pay their share), the new credit score is computed based on the gravity factor: newCreditScore = floor(oldCreditScore - 0.2 * gravityFactor). After that the user's credit score is updated, the report and the group are closed.

Subtasks:

1. Extract data from the report.
   - The report initiator
   - The reported user
   - The date and time of the report
   - The owed money
2. Verify the report.
   - Check the reported users transaction history to verify if the user made a separate payment to the reporter user with the exact sum of money owed with the date of the transaction being after the bill split was initiated. If he did, the report is dismissed. Otherwise, it is a valid report.
3. Analyse data.
   Check if the user could have paid the bill. Here are the two cases described. One that checks the current money in the reported user account, and one that sums the transactions since the report was initiated to see if the user intentionally did not pay their share.
   - Check if the user paid at least 3 shares of bills on other occasions (if this is their first time forgetting to pay it, we will assume it is an honest mistake of not paying it this time)
   - Check if the user usually transfers money to the report initiator (if the reported user sent money to the reporter at least 5 times in the last month, they might be friends and the decreasing process should not be considered that serious)
   - Check the database with the "noOffences" field to check how naughty the user is (this will be calculated based on the previously described formula)
4. Computer the new credit score.
   - Based on the previous checks, compute the gravity factor.
     - - 50% based on the time since the bill share was initiated, 50% based on the amount of money owed. After that we modify the gravity factor in this way:
     - - 10% of the gravity factor if the user could have paid their share of the bill
     - - 5% if the user paid at least 3 shares of bills on other occasions
     - - 5% if the user made at least 5 transaction to the report initiator in the last month
     - +floor(10% of the number of offences)%

   -Compute the new credit score based on the previously described formula.
     - newCreditScore = floor(oldCreditScore - 0.2 * gravityFactor)
5. Update the database
   - Update the database with the new credit score computed
   - Increase the "noOffences" counter in the database such that the chat analysis part has a shared record of offences.
   - Check if in the activityLog table, the user appears at least once with the activity name "bill split". If he does, update the lastModifiedAmount with however much you decreased now from the credit score. If he doesn't, then make a new entry in that table and put the lastModifiedAmount the amount you lowered the credit score with now.

Stocks transaction history analysis part: (Mihai Balau): A user's credit score will be adjusted based on their stock trading performance, represented by the risk score (an integer in the range [0, 100]) and ROI (return of investment). If they lose more than 35% of their trades, they are considered a bad trader, and their risk score will increase by 5 (cannot exceed 100) for each new investment until they have fewer than 35% losing trades (A losing trade refers to a transaction where the investor sells an asset for less than

its purchase price, in our case the stock market). Their risk score decreases by 5 points per profitable trade, regardless of whether they are considered bad traders or not. Investment frequency risk is classified as low risk for 1-2 trades per day (-5 risk score added), moderate risk for 3-4 trades per day (no changes to risk score), and high risk for more than 5 trades per day (5 risk score added), averaged over the last week and applied at the end of the week. Investment size relative to monthly income is considered low risk if below 5% (-5 risk score), moderate risk between 5-10% (no changes to risk score), and high risk if exceeding 10% (+5 risk score), also applied weekly. The risk score will be applied to the credit score weekly in the following way: the risk score, ranging from 1 to 100, will be divided by 100 and subtracted as a percentage from the credit score. For example, if the user has a credit score of 400 and a risk factor of 20, the new credit score will be 400 - 400 * 0.2 = 320. The credit score cannot go below 100. The base formula for ROI is Amount Gained / Cost of Investment. For each week, it will be calculated for each investment and after this will be an average expressed. Then, if it is lower than 1, the credit score will be decreased with the formula CreditScore = CreditScore - 25 / ROI. If it is greater than 1, the credit score will be increased with the formula CreditScore = CreditScore + 25 * ROI, again credit score must be between [100, 700].

Subtasks:

1. Data Collection and Integration: Collect data from the stock investment module, focusing on financial transactions. For each user, we need their monthly wage. For each user investment, we require the date, amount invested, and amount returned. The system will update in real-time (< 1s) whenever a user closes an investment, ensuring that only new investments are added to the existing internal database and used for calculations.
2. Risk Score Calculation: To calculate a risk score, we analyze a user's stock trading history, focusing on factors like trading performance, investment frequency, and investment size relative to income. If a user loses more than 35% of trades, their risk score increases by 5 per new investment until this rate improves. Each profitable trade decreases the risk score by 5. Investment frequency impacts the risk score: low risk (1-2 trades/day) reduces it by 5, moderate risk (3-4 trades/day) has no effect, and high risk (over 5 trades/day) increases it by 5, all averaged weekly. Investment size also affects the risk score: below 5% of income reduces it by 5, between 5-10% has no effect, and above 10% increases it by 5. The risk score is then used to adjust the user's credit score (this will be presented in Weighted Scoring Model task)
3. Advanced Metrics and Indicators: To evaluate a user's financial responsibility through their stock investment activities, we need to calculate their Return on Investment (ROI). The ROI formula is straightforward: it involves dividing the net profit from an investment by its total cost and expressing the result as a percentage. For example, if an investment of $1,000 yields a net profit of $200, the ROI is 1.2 [Amount Gained / Cost of Investment]. For each week, it will be calculated for each investment and after this will be an average expressed as a percentage and applied to the user's credit score (this will be presented in Weighted Scoring Model task).
4. Weighted Scoring Model: The risk score and ROI are used to adjust the credit score. The risk score, ranging from 1 to 100, is applied by subtracting a percentage from the credit score, where the risk score is divided by 100 and then subtracted as a percentage, weekly. For example, a risk score of 20 would reduce a credit score of 400 to 320. Additionally, the ROI impacts the credit score weekly: if the ROI is less than 1, the credit score decreases by CreditScore - 10 / ROI; if the ROI is greater than 1, it increases by CreditScore + 10 * ROI.
5. Investment Portfolio: Create an investment portfolio that provides a clear overview of each investment, including how it contributes to the overall risk score and ROI. This portfolio should allow users to see the specific impact of each investment on their risk profile and financial performance, enabling them to make informed decisions about their investment strategy. Regularly rebalance the portfolio to ensure it remains aligned with investment goals and adapts to market changes.

Credits/loans analysis: (Paul) Whenever a user wants to get a new loan, we have to decide if they are eligible for it or not and what interest rate we can give them.

- When deciding eligibility of a user and computing interest rate we need to take into account the next factors:
  - User Credit Score will be in the range [100, 700]
  - User Risk Score will be in the range [1, 100]
  - Interest rate will be computed with the formula
    IR = UserRiskScore / UserCreditScore * 100. This ensures that interest rate is proportional to risk score and credit score
  - Monthly payments are computed by the formula:
    MP = LoanAmount * InterestRate / NoOfMonths + PenaltyFactor
    - Number of Months is computed by the formula:

    NoOfMonths = (RepaymentDate - ApplicationDate) / 12
- A user is eligible for a loan if they meet all of these conditions
  - LoanAmount <= MonthlyIncome * 10
  - UserCreditScore >= 300
  - Monthly Income > 0
  - No open credits that are past their repayment period and the user has yet to pay
  - DebtToIncomeRatio <= 60%. This is computed with the formula:
    DTIR = Total Monthly Debt Amount / Monthly Income * 100
  - User Risk Score <= 70
- Whenever the user misses a monthly payment, a penalty factor will be added to their payment
  - Penalty factor will be computed by the formula:
    PF = DaysOverdue/10
- They will also be assigned a standard time frame in which they can pay off the loan, and if they manage to pay it off in that interval, then their CS will go up, if not, it will go down.
  - CS updates will be computed by the formula:

NewUserCreditScore = min(UserCreditScore + (Loan amount / Monthly Income) +
min(TotalDaysInAdvance, 30) - min(TotalDaysOverdue, 100)), 700

- TotalDaysInAdvance is computed by the formula:
  TDIA = RepaymentDate - CurrentDate

Loan Data:

- Loan ID, unique for every loan
- User ID, the user identifier
- Loan Amount, which will be given to the user

- Application Date, for when the loan was requested.
- Approval Status: pending, approved, rejected. (initial value will be “pending”)
- Interest Rate (initial value will be 0 until it is assigned)
- Repayment Date, until which the user must repay the loan
- Monthly Payment Amount
- Monthly Payments Completed (initial value = 0)
- Repaid Amount (initial value = 0)
- Penalty, for missing monthly payments (initial value = 0)

Subtasks:

1. Get user Request:
    - Get UserCNP of the user requesting the loan
    - Get LoanID of the requested loan
    - Get requested LoanAmount
2. Retrieve user data
    - Income
    - Credit score
    - Previous loans - so we know the user has no other open credits that are past their repayment period and the user has yet to pay
    - Risk score
3. Check Loan Eligibility
    - Decide if user qualifies for the loan based on following conditions:
        - LoanAmount <= MonthlyIncome * 10
        - UserCreditScore >= 300
        - No open credits that are past their repayment period and the user has yet to pay
        - DebtToIncomeRatio <= 60%. This is computed with the formula:
          DTIR = Total Monthly Debt Amount / Monthly Income * 100
        - UserRiskScore <= 70
    - If user qualifies for the loan, ApprovalStatus will be true
    - Otherwise the process ends here and the loan is denied
4. Compute Interest Rate dynamically
    - Implement risk-based interest adjustments, interest rates will be proportional to the risk factor of clients and their credit scores
    - IR = UserRiskScore / UserCreditScore * 100
5. Compute Monthly Payment rate
    - Compute number of months over which the payments are spread out
        - Number of Months is computed by the formula:

NoOfMonths = (RepaymentDate.year - ApplicationDate.year) * 12 + (RepaymentDate.month - ApplicationDate.month)

- Monthly payments are computed by the formula:
  MP = LoanAmount * InterestRate / NoOfMonths + PenaltyFactor
- PenaltyFactor has a base value of 0

6. Track loan repayments and updates
   - Check if user managed to pay off monthly payment
   - The penalty will be increased from 0 for each missed payment
   - Payments are considered missed when the current number of months that have passed since the start of the loan is greater than the amount of monthly payments made
7. Late Payment Penalties
   - Penalty factor is computed by the formula:

     PF = DaysOverdue/10
8. Credit Score Adjustment System
   - After the Repayment Date has passed, update CS with the formulas:

NewUserCreditScore = min(UserCreditScore + (Loan amount / Monthly Income) + min(TotalDaysInAdvance, 30) - min(TotalDaysOverdue, 100)), 700)

   - TotalDaysInAdvance is computed by the formula:

     TDIA = RepaymentDate - CurrentDate
9. Insert activity into ActivityLog table
   - ActivityID will be unique
   - UserCNP - the user that completed the loan
   - ActivityName - “loan”
   - LastModifiedAmount - from the formula:

min(UserCreditScore + (Loan amount / Monthly Income) + min(TotalDaysInAdvance, 30) - min(TotalDaysOverdue, 100)), 700) - UserCreditScore

   - This computes the delta of the previous user score and the updated one, and this is used in the user analysis profile functionality

Zodiac sign and Chuck Norris based coin flip system for updating credit system (Boar Victor):
Each user will have a zodiacSign attribute in the db (one of: aries, taurus, gemini, cancer, leo, virgo, libra, scorpio, sagittarius, capricorn, aquarius, pisces) and a zodiacAttribute field (both in the user table). There will be 15 hardcoded attributes (Prosperity: 5, Instability: -4, Determination: 4, Impulsiveness: -10, Ambition: 3, Stubbornness: -6, Intuition: 3, Confidence: -1, Independence: 0, Creativity: 5, Optimism: 4, Persistence: 5, Sensitivity: -2, Spontaneity: -1, Jealousy: -6, Luck: 3), each with a credit score gravity noted next to them. At the press of a round red button in the UI, one attribute will be randomly assigned to each zodiac sign, updating the zodiacAttribute for each user according to their zodiac sign, and the credit score gravity of each will be added to the user’s credit score (do keep in mind that this gravity can be negative and it can lower credit score when adding it). Also, we will connect to the Chuck Norris joke api (https://api.chucknorris.io/) and extract a joke at the press of that button too. We will add the ascii values of all the characters in that joke, modulo them by 10, and subtract or add that amount based on a coin flip (a random number is generated between 0 and 1, anything < 0,5 is heads and the rest is tails). If it is heads we increase the credit score otherwise we lower it.

Subtasks:

1. Assign the user the zodiac sign that corresponds to their birthday (there is a field for the birthday in the db user table)
2. Create the function that when is called, assigns a random attribute to each zodiac sign. So if “creativity” is assigned to the aries zodiac sign, then each user in the db having the aries zodiac sign will have the attribute creativity, and their credit score will be affected by + 5 (described in the feature description)
3. Connect the Chuck Norris API.
4. Make a function that makes a get request and gets a joke from the api
5. Compute the sum of every character in the joke, make that modulo 10 and then generate a random number between 0 and 1 to simulate the coinflip described above
6. Compute the new credit score taking in account the zodiac sign, the coin flip value and the joke as described above.

User analysis profile: (Cristiana) The system shall allow registered users to view their credit score changes on a bar chart, as well as the account information and a list with the activities from which the credit score increased or decreased. The bar chart will be done using clear visual indicators green for increases, red for decreases and yellow for neither, for changes of the credit score based on the previous value (e.g if in day 1 the credit score is 50, and in day 2 the credit score is 60/ 50/ 40, the bar of day 2 is green/ yellow/ red).The shown chart data can be filtered to be weekly, monthly or yearly. The account information that will be displayed is: userId, First name, Last name, CNP, Phone number, email; the displayed information will be read-only. In the list with the activities, the user will see each action that modified the user score, with the specified motive and the amount of score gained/ lost in green/ red. The data will reflect real-time credit score updates based on users' financial activities and the interaction with the other users. The system will maintain the history of the credit score changes of each user in a time span of a year and a half.

1. When clicking on one user, a new page will appear.
2. At the top of the new page, the information about the user will be displayed (userId, First name, Last name, CNP, Phone number, email). The details should match the user’s profile and should be read-only.
3. Create 2 tables:
    a. CreditScoreHistory, containing:
        i. historyId(PK)(int)
        ii. userId(FK)(int)
        iii. currentDate(date)
        iv. currentScore(int)
    b. ActivitiesLog, containing:
        i. activityId(PK)(int)
        ii. activityName(string)
        iii. userId(FK)(int)
        iv. lastModifiedAmount(int)
        v. Details(string)
4. Each time the credit score of a user is modified, in the history table will be added / updated the new credit score. If there exists another credit score for the current day, then the record of the current day from the history table will only be updated with the new score. If there is no record for the current date, then the data will be added in the table.
5. Under the account information of the user, a bar chart will be displayed.
6. When the page will be accessed, the chart will contain the changes of the credit score based on the current month. It will be populated using the Monthly button implementation.

7. The data displayed in the chart will be checked such that the increasing values to be displayed with green, the decreasing values to be displayed with red and if there is no change to be displayed with yellow. The comparison between values will be done considering the previous day (weekly or monthly mode) or month (yearly mode).
8. On the new page, on the right part of the chart space, three buttons should be displayed vertically, having the button text in this order: Weekly, Monthly, Yearly.
9. When the user presses the Weekly button, the chart's data will be updated based on the current week, data being separated based on days of the week. If the week isn't finished yet, the data of the remaining days will be empty. For example, if the current day is friday, the bar for saturday and sunday will be at 0. The data will be taken from the history table, for the current week.
10. When the user presses the Monthly button, the chart's data will be updated based on the current month, data being separated based on days of the month. If the month isn't finished yet, the data of the remaining days will be empty. For example, if the current day is 29.01, the bar for 30.01 and 31.01 will be at 0. The data will be taken from the history table, for the current month.
11. When the user presses the Yearly button, the chart's data will be updated based on the current year, data being separated based on months of the year. If the year isn't finished yet, the data of the remaining months will be empty. For example, if the current month is october, the bar for november and december will be at 0. The data will be taken from the history table, for the current year.
12. Under the chart, the user can see a list with the activities that changed the credit score.
13. Each item in the list should contain: Name of activity, details of the activity (reasons / motives of why the score increased or decreased) and the amount of the score that increased or decreased.
14. The text of the score should be green if the score increased and red if the score decreased.

Credit score tips: (Haj Laith) Users will receive tips on how to improve their credit score, and also congratulations (whenever their credit score increases) and shaming messages (whenever their credit score decreases). Tips will be given according to the user's credit bracket. Credit brackets are as follows: 100-299 low, 300-549 medium, 550 to 700 high. Tips will be given when someone's credit score changes. So we must look at all the functions from other features that change the credit score, and in there send tips according to the user's new bracket. Congratulations and shaming messages will be sent every 3rd tip, the low bracket and medium brackets will get shaming messages (we are dystopian after all), and the high bracket gets congratulations messages. Each user will have a tips tab in their menu that contains all the tips and congratulations/shaming messages given to them with the date they were sent next to them.

Credit Brackets:

- Low Credit: 100 to 299
- Medium Credit: 300 to 549
- High Credit: 550 to 700

Subtasks:

1. Create and store at least 5 tips for the lower credit score population

The tips are:

-" Pay your bills on time,Late payments can decrease your score"

-" Credit cards aren't free money, if you cannot afford something, dont put it on Credit hoping you will figure it out after"

-"Avoid doing your past mistakes "

-"if you can't pay off a card, stop using it. Swiping more won't fix your situation"

-"Start small if you have no credit history."

2. Create and store at least 5 tips for the middle credit score population

The tips are:

- “ Set up automatic payments. One missed bill can set you back years “
- “Stop applying for new credit so often. Every application lowers your score a bit”
-”Don’t let unpaid debts sit for too long. Old debts still impact your credit, so pay them off.”
-”Pay off debts with the highest interest first. It saves you money and helps with your score”
-”Don’t apply for loans you don’t need.Every application slightly lowers your score”

3. Create and store at least 5 tips for the higher credit score population

The tips are:

-”Pay your full balance every month.“
-”Set up autopay for all bills. Even one missed payment can drop your score fast”
-”Avoid unnecessary hard inquiries. Only apply for new credit when you truly need it”.
-”Use credit smartly. A mix of credit types(loans, credit cards) can boost your score”
-”Keep your credit card utilization low.Even if your limit is high, don’t use more than 10-20%”

4. Create and store at least 3 congratulations messages

The messages are:

-”Your credit score just went up! Keep making smart moves, and you will unlock better opportunities.”
-”You’re roving you know how to handle money :).Lenders trust you more, and so should you.”
-”Great progress! A higher credit score means better deals, lower interest rates, and more financial freedom.”

5. Create and store at least 3 roast messages

The messages are:

-”Your credit score is lower than your phone battery at 2%.Charge it up before it’s too late!”
-”Even monopoly money has better credit than you right now. Get it together!”
-”Your credit score is so low that even a charity wouldn’t loan you a dollar!”

6. Make a tips tab.

The user will have a credit tips section in their menu, where they can see all their past tips, congratulations , and roast messages displayed in most recent first(newest at the top, oldest at the bottom), with each entry showing the message on the left side and the date on the right side, and for every three tips sent either a congratulations message (for high credits) or a roast (for low/medium scores), ensuring a complete history of their credit advice unless manually cleared by the user.