Pseudo-code Algorithm Design

Jake Sundstrom 1-8, Jordan Brandt 9-16, Juni Bautista & Junjie Lin & David Zhan 17-24
and
Rakan Abu Awwad: UI Design

Use case 1: Expense Management

```
FUNCTION manageExpense():
    DISPLAY "Enter the details of the expense(Amount, Category, Date, Description): "
    INPUT amount, category, date, description
    //Checks if the expense if valid and categorizes it if auto is selected
    if amount <=  0:
        DISPLAY "Error: The amount must be greater than 0"
        RETURN
    if category NOT IN allowedCategories:
        DISPLAY "Error: The Category you inputted isn't allowed. Allowed  Categories:
        {allowedCategories}"
        RETURN
     if NOT isValidDate(date):
        DISPLAY "Error: The date entered is not a valid date"
        RETURN
    if category == "AUTO":
        category = categorizeExpense(description)
    expense = {
            "amount": amount,
            "category": category,
            "date": date,
            "description": description
          }
    ADD expense TO userExpensesList //adds expense to total expenses list
    monthlyTotal = calculateMonthlyTotal()
    monthlyLimit = getMonthlyLimit()
    if monthlyTotal > monthlyLimit: //checks if the monthly limit has been exceeded
        DISPLAY: "Well that expense put you past your set monthly limit"
    DISPLAY: "Expensive successfully added"
FUNCTION calculateMonthlyTotal(): //adds expenses together to find amount spend that month
    total = 0
    FOR expense IN userExpensesList: //checks if that the expense is for the current month
        if isCurrentMonth(expense["date"]):
            total += expense["amount"]
    RETURN total //returns total amount
FUNCTION categorizeExpense(description): //checks what category the expense belongs to
    FOR keyword, suggestedCategory IN categorizationRules:
        if keyword IN description:
            RETURN suggestedCategory
    RETURN “Uncategorized”
```

Use case 2: Emergency Fund Allocation

```
FUNCTION emergencyFundAllocation(): //allows for additions and subtractions to the
emergency fund
```

```
DISPLAY "Welcome to Emergency Fund Allocation Portal"
AUTHENTICATE user USING FERPA-compliant authentication //checks the user is authenticated
if authenticationFail:
    DISPLAY "Authentication has failed. Please retry or reset password."
    RETURN
DISPLAY  “Welcome to your Emergency Fund”
DISPLAY ”Would you like to Withdraw or Deposit” //allows user to choose which action
INPUT userAction
//either performs the needed actions for withdrawal or deposit and checks conditions are met
if userAction.equals(“Withdraw”):
    DISPLAY “How much would you like to Withdraw?: ”
    INPUT requestedAmount
    if requestedAmount > availableAmount:
        DISPLAY “You have insufficient funds, you have: {availableAmount}”
        RETURN
    PROCESS transferFunds(requestedAmount)
    DISPLAY “Funds have been taken, your current balance is {availableAmount}”
    SEND emailReceipt TO userEmailAddress

else if userAction.equals(“Deposit”):
    DISPLAY “How much would you like to Deposit?: ”
    INPUT requestedAmount
```

```
        PROCESS transferFunds(requestedAmount)

        DISPLAY “Funds have been added, your current balance is {availableAmount}”
        SEND emailReceipt TO userEmailAddress //sends an email confirmation of
action
    else: //user imputed an invalid option it informs them
        DISPLAY “Sorry that’s not a valid option”
        RETURN
```

Use case 3: Dark/ Light Mode

```
FUNCTION darkLightModeSetting(): //allows user to swap from light mode and dark mode
    DISPLAY “Welcome to Display settings”
    AUTHENTICATE user USING FERPA-compliant authentication
    if authenticationFail:
        DISPLAY "Authentication has failed. Please retry or reset password."
        RETURN
    DISPLAY “Would you like Dark or Light mode?: ”
    INPUT modeChoice
    // If valid option is given it makes the needed change and saves it, otherwise it gives an
    error message
    if modeChoice.equals(“Dark”):
        APPLY darkMode()
        SAVE userThemePref = “Dark Mode”
        DISPLAY “Enjoy {userThemePref}”
```

```
else if modeChoice.equals(“Light”):
    APPLY darkMode()
    SAVE userThemePref = “Light Mode”
    DISPLAY “Enjoy {userThemePref}”
else:
    DISPLAY “Sorry that’s not a valid option”
    RETURN
```

Use case 4: Debt Tracking Mode

```
FUNCTION debtTrackingMode(): //gives the user information about their current debts
    DISPLAY "Welcome to Debt Tracking Dashboard"
    AUTHENTICATE user USING FERPA-compliant authentication
    if authenticationFail:
        DISPLAY "Authentication has failed. Please retry or reset password."
        RETURN
    debts = GET userDebts()
    DISPLAY “Here are your current debts: ”
    FOR each debt in debts: //lists information about each debt
        DISPLAY “Debt title: ”, debt.name
        DISPLAY “Principal: ”, debt.principal
        DISPLAY “Interest Rate: ”, debt.interestRate
        DISPLAY “Remaining Amount Due: ”, debt.remainingAmount
        DISPLAY “Due Date: ”, debt.dueDate
        DISPLAY “Payment History: ”, debt.paymentHistory

    DISPLAY “Progress Visualization: ”, debtRepaymentGraph(debts) //creates graph of
changes made to debts

    if userMakesPayment: //updates debts when changes are made

        UPDATE debtBalance(userPayment)

        UPDATE interestCalc()

        DISPLAY “Balance has been updated”

    FOR each debt in debts: //notifies the user when payment for debts is coming or changed

        if dueDateComing(dept):

            SEND notification TO USER(“Warning, you have a debt payment almost
            due for: ”, debt.name)

        if interestRateChanged(debt):

            SEND notification TO USER(“Warning, you have a debt payment almost
            due for: ”, debt.name)

    //User is adding new debt

    DISPLAY “Enter new debt details(Title, Principal, Interest Rate, Amount due, Due
    Date,): ”

    INPUT debtDetails

    if validateDebtDetails(debtDetails): //checks the details for the debt are valid and adds
debt if it is

        ADD newDebt TO userDebts()

        DISPLAY “New Debt has been added”

    else:

        DISPLAY "Invalid debt details. Please check your input."

    report = GENERATE debtReport(debts) //generates then displays a debt report
    DISPLAY “Current Debt Report: ”, report

    if externalServiceConnected: //syncs changes with their external banking service

        SYNC userDebts with bankService()

        DISPLAY “Debt data is up to date with your financial services”
```

Use case 5: Parent Mode

```
FUNCTION parentMode(): //Gives parents the option to control limits and get notifications of financial activities

    DISPLAY "Welcome to Debt Tracking Dashboard"

    AUTHENTICATE parent using secure credentials //ensures the parent is allowed access

    if authenticationFail:

        DISPLAY "Authentication has failed. Please retry or reset password."

        RETURN

    financialActivity = GET studentFinancialActivity()

    FOR each activity IN financialActivity: //every financial transaction is logged and a brief description is sent to the parent

        LOG activity IN studentFinancialRecord()

        SEND notification TO PARENT(“New transaction: ”), activity.description

    DISPLAY “Recent Transactions: ”

    FOR each transaction IN financialActivity: //Shows all transactions in order of entry

        DISPLAY transaction.description, transaction.amount, transaction.category

    FOR each transaction IN financialActivity: // If a transaction exceeds the monthly limit for that category the parent is informed

        if transaction.amount > getMonthlyLimit(transaction.category):

            SEND notification TO PARENT and STUDENT(“WARNING: spending
            limit exceeded for ”), transaction.category
    DISPLAY “Would you like to update the monthly limit for a category?(Yes or No) ”
    INPUT adjustChoice
    if adjustChoice.equals(“Yes”): //allows the parent to change a category’s limit if valid
        DISPLAY “Please enter the category and updated limit(category, new limit): ”
        INPUT updatedLimit
        if allowedCategory(updatedLimit[0]):
            if allowedLimit(updatedLimit[1]):
                UPDATE category(updatedLimit[0]) with updatedLimit
            else:
                DISPLAY “You can’t set the limit to that amount”
                RETURN
        else:
            DISPLAY “That is not a valid category”
            RETURN
    DISPLAY “Updated Financial Overview: ”, GENERATE
    financialOverview(studentFinancialActivity) //gives an overview of financial activity
    if parentsRequestReport: //creates a financial report if requested
        report = GENERATE financialReport(financialActivity)
        DISPLAY “Financial Report: ”, report
    if externalServiceConnected: //syncs changes with external banking service if applicable
        success = SYNC studentFinancialDate WITH bankService()
        if NOT success:
            DISPLAY “Sync failed”
        else:
            DISPLAY “Sync succeeded”
```

Use case 6: Income Management

```
FUNCTION incomeManagement(): //allows for students to add new income
    DISPLAY “Welcome to Income Management”
    AUTHENTICATE user USING FERPA-compliant authentication
    if authenticationFail:
        DISPLAY "Authentication has failed. Please retry or reset password."
        RETURN
    While userAddingIncome: //allows the user to keep adding valid incomes which are
    sorted based on frequency paid until they select no
        DISPLAY “Enter income amount: ”
        INPUT incomeAmount
        DISPLAY “Enter income label: ”
        INPUT incomeLabel
        DISPLAY “Enter income frequency: ”
        INPUT incomeFrequency
        if incomeAmount <= 0 OR NOT isNumber(incomeAmount):
            DISPLAY “Not a valid amount, please enter a positive number”
            CONTINUE
        if incomeLabel == "" OR NOT isValidText(incomeLabel):
            DISPLAY “Not a valid label, please enter a valid description”
            CONTINUE
        if incomeFrequency == "" OR NOT isValidFreq(incomeFrequency):
            DISPLAY “Not a valid Frequency, please enter a valid frequency”
            CONTINUE
        ADD incomeAmount, incomeLabel, incomeFrequency TO userIncomeDatabase
        SORT userIncomeDatabase BY incomeFrequency
        DISPLAY “Income has been added!”
        DISPLAY “Would you like to add another source of income(Yes/No)? ”
        INPUT answer
        if answer.equals(“No”):
            userAddingIncome = FALSE
    DISPLAY “Income Summary: ” //give a summary of the user’s income
    FOR each income IN userIncomeDatabase:
        DISPLAY income.label, income.amount, income.frequency
```

Use case 7: Budget Account Creation (User ID/ Password)

```
FUNCTION budgetAccountCreation(): //allows a new user to create an account
    DISPLAY “Welcome to the Hokie Budgeting Tool, let’s create your account”
    AUTHENTICATE user USING FERPA-compliant authentication
    if authenticationFail:
        DISPLAY "Authentication has failed. Please retry or reset password."
        RETURN
    DISPLAY “Enter your desired username/ID: ”
    INPUT username
    DISPLAY “Enter your desired password: ”
    INPUT password
    //checks that the username and password are valid and asks for new ones until they are
    WHILE userIDAlreadyExists(username):
        DISPLAY “Username/ID is taken, please try again: ”
        INPUT username
    WHILE passwordIsWeak(password):
        DISPLAY “TIP: Strong Passwords should have at least 8 characters”
        DISPLAY “Password is weak, please try again: ”
        INPUT username
    ENCRYPT password
    STORE username, encryptedpassword in userDatabase
```

Use case 8: Income Simulator

```
FUNCTION incomeSimulator(): //allows the user to test an the results of a different income
    DISPLAY “Welcome to the Income Simulator!”
    AUTHENTICATE user USING FERPA-compliant authentication
    if authenticationFail:
        DISPLAY "Authentication has failed. Please retry or reset password."
        RETURN
    DISPLAY “Enter income details(Sources, Frequencies, Time Frame(in days)):”
    INPUT incomeSources //array of incomes
    INPUT incomeFrequencies //array of income frequencies
    INPUT timeFrame //How long the simulation goes on

    FOR each income IN incomeSources: //checks that the incomes are valid

        if income <= 0 OR NOT isNumber(income):

            DISPLAY “Not a valid amount, please enter a positive number”

            RETURN

    totalIncome = 0

    FOR i = 0 TO LEN(incomeSources) - 1: // finds the total income over the time period

        income =  calculateProjection(incomeSources[i], incomeFrequencies[i],
        timeFrame)

        totalIncome += income

    balance = calculateBalance(totalIncome, expenses) //finds the balance based on user’s
real expenses

    dailyDisposable = totalIncome / timeFrame //finds user daily income

//displays the information found from the simulation

    DISPLAY “Total project income over ” + timeframe + “ days is ” + totalIncome

    DISPLAY “Balance: ”, balance

    DISPLAY “Daily Disposable Income: ”, dailyDisposable

    DISPLAY “Simulation Complete, hope you enjoyed it!”
```

Use Case 9: Budget Setting

```
Function SetBudget():

    Display("Enter the amount you would like to save: ")

    UserSaveAmount = Input()

    Display("Enter the frequency of saving (e.g., weekly, monthly): ")
```

```
UserSaveFrequency = Input()

#User needs to have a valid a previous valid income and expense from other methods

IF (UserIncome IS NULL OR UserExpenses IS NULL) THEN

    NotifyUser("Valid income and expenses are required to create a budget.")

    EXIT

BudgetReccomendations = CalculateBudget(UserSaveAmount, UserSaveFrequency,
UserIncome, UserExpenses)

#Third party api call to calculate a recommended budget given budget parameters

DisplyBudget(BudgetReccomendations)

#another third party api call to display the recommended budget

Display("Would you like to adjust these recommendations? (Yes/No)")

AdjustmentResponse = Input()

IF UserAdjustmentResponse == "Yes" THEN

    setBudget()

Else:

    UserBudget = BudgetReccomendations

    EXIT
```

Use case 10: Automatic Categorization

```
Function AutomaticCategorization()

    IF (UserIncome IS NULL AND UserExpense IS NULL AND UserBudget) THEN

        Display("You must have income, expense, or budget inputted to categorize.")

        EXIT

    InfoSet = {UserIncome, UserExpense, UserBudget}
    For each in InfoSet:
        If not NULL:
            DisplayData()
            #Api to display data breakdowns
    EXIT
```

Use case 11: Monthly Budget Overview

```
Function MonthlyBudget():
    IF UserHasBudget() THEN
        Display("You already have a budget. Do you want to create a new one or edit
your current budget?")
    Input = Input()
    IF Input == "Edit" THEN
        EditCurrentMonthBudget()
        #not included here, but basically would just be like CreateNewMonthBudget
    ELSE
            CreateNewMonthBudget()


Function CreateNewMonthBudget:
  DisplayPrompt("Please enter your expected income:")
  UserIncome = Input()
  DisplayPrompt("Please enter your expected expenditures:")
  UserExpenditures = Input()
  DisplayPrompt("Please enter your savings goals for the month:")
```

```
 UserSavingsGoal = Input()

 IF (UserIncome < 0 OR UserExpenditures < 0 OR UserSavingsGoal < 0) THEN

     DisplayErrorMessage("Invalid input detected. Please enter valid (non-negative) numbers for
income, expenditures, and savings.")

     EXIT

CreateBudget(UserIncome, UserExpenditures, UserSavingsGoal)

SaveBudgetToUserProfile(UserIncome, UserExpenditures, UserSavingsGoal

DisplayBudgetOverview(UserIncome, UserExpenditures, UserSavingsGoal)
```

Use case 12: Annual Budget Planning

```
FUNCTION AnnualBudget():

        IF UserHasAnnualBudget() THEN

                Display("You already have a budget. Do you want to create a new one or edit
                your current budget?")

        Input = Input()

        IF Input == "Edit" THEN

                EditAnnualBudget()

        ELSE

                        CreateNewAnnualBudget()


Function CreateNewAnnualBudget():

        MonthlyBudget = GetMonthlyBudget()

        AnnualBudget = CalculateAnnualBudget(MonthlyBudget)

        DisplayAnnualBudgetOverview(AnnualBudget)
```

```
FUNCTION EditAnnualBudget()

    Display("Editing your annual budget...")

    Display("Enter edited annual income:")

    EditedIncome = Input

    Display("Enter edited annual expenditures:")

    EditedExpenditures = Input

    Display("Enter edited savings goal:")

    EditedSavingsGoal = Input

    AnnualBudget.Income = EditedIncome

    AnnualBudget.Expenditures = EditedExpenditures

    AnnualBudget.SavingsGoal = EditedSavingsGoal
```

Use case 13: Cash Flow Management

```
Function CashFlow():

    DisplayMessage("Displaying Cash Flow Dashboard...")

    DisplayDashboard()

    DisplayMessage("Select a time period for cash flow overview.")

    TimePeriod = GetUserInput("Enter 'Weekly', 'Monthly', or 'Quarterly'")

    IF TimePeriod == "Weekly" OR TimePeriod == "Monthly" OR TimePeriod ==
    "Quarterly" THEN

        FilterTransactionsByTimePeriod(TimePeriod)

    ELSE

        DisplayErrorMessage("Invalid time period. Please enter a valid option.")
        EXIT CashFlowManagementProcess()

    IncomeData, ExpenseData = GetIncomeAndExpensesForPeriod(TimePeriod)

    DisplayCashFlowData(IncomeData, ExpenseData)

    NetCashFlow = CalculateNetCashFlow(IncomeData, ExpenseData)

    DisplayMessage("Net Cash Flow: " + NetCashFlow)
```

Use case 14: Saving Goals

```
FUNCTION SavingGoal():

    Display("Enter the goal name")

    GoalName = Input()

    Display("Enter the target amount for this goal:")

    TargetAmount = Input()

    Display("Enter the target date for achieving the goal (MM/DD/YYYY):")

    TargetDate = Input()

    AvailableBudget = GetAvailableBudget()  // Monthly contributions available from budget

    Display("Available monthly contribution options from your budget categories: " +
    AvailableBudget)

    Display("Enter the monthly contribution amount to this goal:")

    ContributionAmount = Input()

    DisplayMessage("To reach your goal by the target date, you need to contribute " +
    ContribuitionAmount + " per month.")

    UserConfirmation = GetUserInput("Do you want to proceed with this goal? Enter 'Yes' to
    confirm or 'No' to cancel.")

    IF UserConfirmation == "Yes" THEN

        LogSavingGoal(GoalName, TargetAmount, TargetDate, ContributionAmount)
        StartTrackingProgress(GoalName, TargetAmount, ContributionAmount)
        DisplayMessage("Your saving goal has been created and is now being tracked.")
        EXIT
    ELSE
        DisplayMessage("Saving goal creation canceled.")
        EXIT
```

Use case 15: Category Expenditure Pie Chart

```
FUNCTION ExpenditurePieChart():
    Expenditures = GetUserExpenditures()
    IF (Expenditures isEmpty) THEN
        DisplayMessage("No expenditure data found. Please add some expenditures.")
        EXIT
    CategorizedExpenditures = CategorizeExpenditures(Expenditures)
    Display("Expenditures categorized successfully.")
    PieChart = GeneratePieChart(CategorizedExpenditures)
    DisplayPieChart(PieChart)
```

Use case 16: Financial Transaction Reminders

```
FUNCTION TransactionReminders():
    Display("Enter the details for your financial transaction reminder.")
    Display("Enter the name of the reminder:")
    ReminderName = Input()
    Display("Enter the amount due for the transaction:")

    AmountDue = Input()

    Display("Enter the due date (e.g., YYYY-MM-DD):")

    DueDate = Input()

    Display("Enter the frequency of the reminder (e.g., 'One time' or 'Recurring'):")

    Frequency = Input()

    IF (ReminderName == "" OR AmountDue <= 0 OR DueDate == "" OR Frequency == "")
THEN

        DisplayMessage("Invalid input. Please enter valid information for all fields.")

        EXIT SetUpTransactionReminder()

    Reminder = CreateReminder(ReminderName, AmountDue, DueDate, Frequency)

    SaveReminderToUserProfile(Reminder)

    Display("Your reminder has been successfully set up!")

    DisplayReminderDetails(Reminder)
```

Use case 17: Investment Tracking

```
FUNCTION investmentTracking():

    authentication()

    If authenticationFail:

        Display "Authentication failed. Please retry or reset your password."

        Return

    If userClickInvestmentTracking()

        GET userInvestmentPortfolio() = investment

        formatInvestmentDetails(investment)
        Display investmentHistory(formatInvestmentDetails)
    If userClickExportToPDF():
        FileO(formatInvestmentDetails)
        Display “Investment Tracking has been saved successfully as PDF file
```

Use case 18: Share, print, or export budget sheet

```
FUNCTION sharePrintExportBudget():
    INPUT: budget data, action (share, print, export), options (email, file, etc)
    OUTPUT: status displaying action success or failure
    If userClickShare():
        Email = options[email]
        shareFile(Email)
        Display “Budget successfully sent to *email*”
    If userClickPrint():
        Display “Budget successfully exported to *Printer name*”
    If userClickExport():
        Display “Budget successfully exported as PDF”
```

Use case 19: Event based budgeting

```
FUNCTION generateCategoryExpenditurePieChart():
INPUT: expenditureData (JSON file)
OUTPUT: pieChart (visual representation of expenditures by category) or error message
```

```
IF userLogin(userCredentials):
    IF userNavigateTo("Monthly Expenditure"):
        IF userChoose("Pie Chart View"):
            expenditureData = decodeJSONFile("expenditures.json")


            IF expenditureData IS NOT EMPTY:
                categorizedData = categorizeExpenditures(expenditureData)
                pieChart = createPieChart(categorizedData)
                DISPLAY "Expenditure Pie Chart:"
                DISPLAY pieChart


                IF userClickExport():
                    exportFormat = getUserExportChoice()  # e.g., "PDF" or "Image"
                    exportPieChart(pieChart, exportFormat)
                    DISPLAY "Pie Chart successfully exported as", exportFormat
            ELSE:
                DISPLAY "No expenditure data found. Please add your expenditures to view the pie
chart."
        ELSE:
            DISPLAY "Error: Pie Chart View not selected."
    ELSE:
        DISPLAY "Error: Unable to navigate to Monthly Expenditure section."
ELSE:
    DISPLAY "Error: Invalid login credentials."
```

```
FUNCTION decodeJSONFile(filePath):
    TRY:
        OPEN filePath
        RETURN parsedJSONContent
    CATCH error:
        DISPLAY "Error decoding expenditure file."
        RETURN []

FUNCTION categorizeExpenditures(expenditureData):
    categories = {}
    FOR EACH expenditure IN expenditureData:
        category = expenditure.category
        amount = expenditure.amount
        IF category EXISTS IN categories:
            categories[category] += amount
        ELSE:
            categories[category] = amount
    RETURN categories

FUNCTION createPieChart(categorizedData):
    USE thirdPartyLibraryToCreatePieChart(categorizedData)
    SET pieChartColors = highContrastColors(categorizedData)
    ADD labelsToPieChart(pieChartColors)
    RETURN pieChart

FUNCTION exportPieChart(pieChart, format):
    IF format == "PDF":
        SAVE pieChart AS PDF
    ELSE IF format == "Image":
        SAVE pieChart AS Image
    ELSE:
        DISPLAY "Invalid export format."
```

Use case 20: Budget Analysis

```
FUNCTION incomeSimulator():
INPUT: incomeData (sources, amounts, frequencies), timeFrame
OUTPUT: simulated income scenarios, estimated balance, daily disposable, or error messages

            IF userClickTools():
            IF userClickIncomeSimulator():
                        DISPLAY "Enter income details (source, amount, frequency):"
                        incomeData = getUserInput()
                        DISPLAY "Select simulation time frame:"
                        timeFrame = getTimeFrame()
```

```
 IF validateIncomeData(incomeData):
     simulatedIncome = calculateIncomeProjection(incomeData, timeFrame)
     estimatedBalance = calculateBalance(simulatedIncome, timeFrame)
     dailyDisposable = calculateDailyDisposable(estimatedBalance, timeFrame)
     DISPLAY "Projected Income Scenarios:"
     DISPLAY simulatedIncome
     DISPLAY "Estimated Balance: ", estimatedBalance
     DISPLAY "Daily Disposable Income: ", dailyDisposable
     ELSE:
         DISPLAY "Error: Please enter valid income details (numeric and
         non-negative)."
     ELSE:
         DISPLAY "Error: Income Simulator tool not found."
     ELSE:
         DISPLAY "Error: Tools menu not clicked."


FUNCTION validateIncomeData(incomeData):
    FOR EACH source IN incomeData:
        IF source.amount IS NOT numeric OR source.amount < 0:
        RETURN False
        RETURN True


FUNCTION calculateIncomeProjection(incomeData, timeFrame):
    totalIncome = 0
    FOR EACH source IN incomeData:
    IF source.frequency == "daily":
    totalIncome += source.amount * numberOfDays(timeFrame)
    IF source.frequency == "weekly":
    totalIncome += source.amount * numberOfWeeks(timeFrame)
    RETURN totalIncome


FUNCTION calculateBalance(simulatedIncome, timeFrame):
    RETURN simulatedIncome - calculateExpenses(timeFrame)


FUNCTION calculateDailyDisposable(estimatedBalance, timeFrame):
    RETURN estimatedBalance / numberOfDays(timeFrame)
```

Use case 21: Separate Card and Cash Tracking

```
FUNCTION investmentTracking():
INPUT: userCredentials, linkedAccounts (optional), manualInvestments (optional)
OUTPUT: portfolioOverview (asset allocation, current values, gains/losses, performance history)

IF userLogin(userCredentials):
  IF userNavigateTo("Investment Tracking"):
    investmentData = retrieveInvestmentData(linkedAccounts, manualInvestments)

    IF investmentData IS NOT EMPTY:
            portfolioOverview = generatePortfolioOverview(investmentData)
            DISPLAY "Investment Portfolio Overview:"
            DISPLAY portfolioOverview
            DISPLAYGraphsAndCharts(portfolioOverview)

            IF userAdjustInvestments():
                newInvestments = getUserInvestmentChanges()
                updateInvestmentRecords(newInvestments)
                DISPLAY "Investment records successfully updated."
        ELSE:
            DISPLAY "No investment data found. Please link accounts or add investments manually."
    ELSE:
        DISPLAY "Error: Unable to navigate to Investment Tracking section."
    userLogout()
ELSE:
    DISPLAY "Error: Invalid login credentials."


FUNCTION retrieveInvestmentData(linkedAccounts, manualInvestments):
    investmentData = []
    IF linkedAccounts IS NOT EMPTY:
        FOR EACH account IN linkedAccounts:
            accountData = fetchAccountData(account)
            IF accountData IS VALID:
```

```
            investmentData.append(accountData)
        ELSE:
            DISPLAY "Error: Unable to access linked account. Please update account credentials."
    IF manualInvestments IS NOT EMPTY:
        investmentData.append(manualInvestments)
    RETURN investmentData

FUNCTION generatePortfolioOverview(investmentData):
    portfolioOverview = {}
    portfolioOverview["Asset Allocation"] = calculateAssetAllocation(investmentData)
    portfolioOverview["Current Values"] = calculateCurrentValues(investmentData)
    portfolioOverview["Gains/Losses"] = calculateGainsAndLosses(investmentData)
    portfolioOverview["Performance History"] = generatePerformanceHistory(investmentData)
    RETURN portfolioOverview

FUNCTION updateInvestmentRecords(newInvestments):
    FOR EACH investment IN newInvestments:
        updateRecord(investment)
    RETURN True
```

Use case 22: Peer Financial Advice

```
FUNCTION peerFinancialAdvice():
        INPUT: pie chart, income and expense history
```

```
    OUTPUT: reply to requested advice

    Display “Welcome to Peer Financial Advice Platform”

        Authenticate()

        If authenticationFail()

            Display "Authentication has failed. Please retry or reset your password."

            RETURN

        Display “Please agree to the user data sharing contract.”

        If userClickDisagree()

            Display “Unable to use Peer Financial Advice due to refusing to share
            personal data.”

            RETURN

        Question = GET userInput()

        FOR each Question IN Questions: DISPLAY Question.ID + ": " + Question.text

        Advice = GET adUserInput()

        Discussions = FETCH recentDiscussions(FOR each Advice in Question)

        Display “"Here are some recent financial discussions:"

        Display Discussions()
```

Use case 23: Importing Bill and OCR PDF

```
FUNCTION importBillThroughOCR():

    INPUT: Camera API, PDF, PNG, JPN image file format

    OUTPUT: Words and digits detected from image files and JSON file of OCRResults

    If userClickCam():

        Display “Please place your bill on the table and point your camera at the bill. Or
        click File button to upload your bill from your photo album.”
        If userClickFile():
            Display “Please choose your bill from the photo album and click Upload.”
                If userClickUpload():
                    OCRResults = performOCR(ImageFile)
        If userClickShoot():
            OCRResults = performOCR(ImageFile)
        BillData = parseOCRData(OCRResults)
```

Use case 24: Virtual Account Assistant

```
FUNCTION virtualAssistant():
    Display “Welcome to your Virtual Account Assistant”
        Authenticate()
        If authenticationFail
            Display "Authentication has failed. Please retry or reset your password."
            RETURN
        UserSession = TRUE
        Display "How can I assist you with your accounts today? You can ask about
        balances, recent transactions, budgeting tips, or more."
        WHILE (userState == TRUE) {
            UserQuery = GET userTextInput()
            POST UserQuery to ChatGPT API
            GET ChatGPT API answer return
            If (userClickQuit) {
```

```
        setUserState == FALSE

        RETURN mainPage()

    }

}
```