პროექტის აღწერა

პროექტის მიზანი არის დღგ-ს ზღვარის 100 000 ლარის ბრუნვის გადაჭარბების შესახებ ინფორმაციის მიღება აღნიშნული ფაქტის დადგომისთანავე. ამისთვის მომხმარებელი ვალდებულია შევიდეს/დარეგისტრირდეს სისტემაში და მოახდინოს ყოველდღიური ბრუნვის შეტანა , ასევე მას საშუალება აქვს კორექტირება მოახდინოს უკვე შეტანილი გასული პერიოდის ბრუნვის თანხაში. შესაბამისად უნდა განხორციელდეს 100 000 ზღვარის ყოველდღიური დაანგარიშება: მიმდინარე თვის ყველა დღის ბრუნვას უნდა დაემატოს გასული 11 თვის ბრუნვა(ერთობლივი შემოსავალი)

პირველ ეტაპზე ხორციელდება მომხმარებლის რეგისტრაცია ან არსებული მონაცემებით სისტემაში შესვლა:

class User:

    def \_\_init\_\_(self, username, password):

        self.username = username

        self.password = password

    def register(self):

        register\_choice = input("Do you have an account? (y/n): ")

        if register\_choice.lower() == 'n':

            while True:

                username = input("Enter username for registration: ")

                password = input("Enter your password: ")

                try:

                    self.add\_user(username, password)

                    print("User registration successful.")

                    break

                except ValueError:

                    print("Username already exists. Please choose a different username.")

        elif register\_choice.lower() == 'y':

            self.username = input("Enter your username: ")

            self.password = input("Enter your password: ")

            print("Login successful.")

        else:

            print("Invalid choice.")

    def add\_user(self, username, password):

        # Add logic to add user to database

        pass

შემდეგ ეტაპზე უშუალოდ იქმნება დღგ-ს რეგისტრაციის კლასი, რომლითაც გადაცემული მომხმარებლის ექაუნთით აღრიცხული იქნება ყოველდღიური ბრუნვა , რომელიც დაემატება ჯამური ბრუნვის მონაცამებს

class VATRegistration(User):

    def \_\_init\_\_(self, username, password):

        super().\_\_init\_\_(username, password)

        self.turnover\_records = {}

        self.vat\_registered = False

    def add\_turnover(self, date, turnover):

        self.turnover\_records[date] = turnover

    def update\_turnover(self, date, new\_turnover):

        if date in self.turnover\_records:

            self.turnover\_records[date] = new\_turnover

            print(f"Turnover for {date} updated successfully.")

        else:

            print(f"No turnover recorded for {date}.")

აგრეთვე მომხმარებელს ექნება გასული ან მიმდინარე პერიოდის (დღის)კორექტირების საშუალება თუ აღნიშნულ დღეზე ჩანაწერი უკვე არსებობს

შემდეგ ეტაპზე მთლიანი ბრუნვის შესახებ მონაცმების გამოტანა

 def display\_turnover(self):

        total\_turnover = sum(self.turnover\_records.values())

        print(f"Total turnover: {total\_turnover}")

        for date, turnover in self.turnover\_records.items():

            print(f"Turnover for {date}: {turnover}")

მიღებული ბრუნვის საერთო თანხის შედარება კანონმდებლობით განსაზღვრულ ზღვართან (100000):

 def calculate\_vat\_threshold(self, date):

        total\_turnover = sum(self.turnover\_records.values())

        return total\_turnover >= 100000

იმ შემთხვევაში თუ თანხა გადააჭარბებს /ტოლია 100000 ლარს ხორციელდება მიმართვის ბოლო თარიღის გამოთვლა (2 სამუშაო დღის ვადა მიმდინარე დღის ჩაუთვლელად)

 def get\_registration\_deadline(self):

        today = datetime.now().date()

        days\_until\_deadline = 2

        while days\_until\_deadline > 0:

            today += timedelta(days=1)

            if today.weekday() < 5:

                days\_until\_deadline -= 1

        return today

შედეგი რეგისტრაციის დღის დედლაინის გამოტანა

არსებული მონაცემების CSV ფაილში შენახვა