**داکیومنت پروژه نرم افزار مدیریت حسابداری**

**بخش های توسعه داده شده توسط مانی مستعلی**

**پروژه درس ap**

**اعضای تیم: مانی مستعلی و محمد صادق همتی**

**استاد راهنما: دکتر داوودآبادی**

**منتور: آقای مرتضی قندچی**

**گیت**

برای توسعه یک پروژه به خصوص اگر شامل بخش های متنوع باشد و همچنین توسط بیشتر از یک نفر مدیریت شود, از ابزاری به نام گیت استفاده می کنیم. به این منظور ریپازیتوری گیت هاب <https://github.com/mani-masteali/accounting-ap.git> را ایجاد کردیم. (منبع راهنمای پیاده سازی گیت: مطالب ارائه شده توسط آقای ذبیحی تی ای درس ap و همچنین کوپایلت vscodeبرای آسانی دسترسی به سینتکس ها و کد های مربوطه و همچنین راهنمایی های ارائه شده در خود سایت گیت هاب). این پروژه یک شاخه اصلی است که کد اصلی در آنجا توسعه داده می شود اما نکته مهم این است که هر شخص در این پروژه مسئولیت های متمایز اما مرتبط به هم داشت پس نیاز بود که یک شاخه فرعی برای بنده (masteali) و یک شاخه برای هم تیمی ام (hemati) ایجاد شود تا هنگام اتمام پروژه شاخه ها در یکدیگر ادغام شوند. هر تغییری که در برنامه ایجاد شده به صورت یک کامیت در این شاخه ها ذخیره و در نهایت در شاخه اصلی ادغام شده اند. همچنین نکته حائز اهمیت استفاده از gitignore برای نادیده گرفتن یک سری از فایل های سیستمی و ثابت است.

**منوی اولیه**

**ثبت نام (sign up):**

بعد از اینکه کاربر گزینه ثبت نام را انتخاب می کند, از او نام, نام خانوادگی, شماره موبایل,یوزرنیم, پسورد, تکرار پسورد(گزینه ای برای تایید پسورد انتخاب شده), شهر محل زندگی کاربر, ایمیل, تاریخ تولد, و همچنین ثبت جواب یک سوال امنیتی که درصورت فراموشی رمز عبور کاربر با وارد کردن ایمیل یا یوزنیم به آن پاسخ داده و به رمز خود دسترسی خواهد داشت.اطلاعات یوزر ساخته شده به صورت یک آبجکت از کلاس User ساخته شده و در یک فایل csv ذخیره خواهد شد.

در ابتدا در کلاس User مقادیر مطلوب که در عکس زیر نشان داده شده اند initialize می شوند. دلیل None گذاشتن همه مقادیر در ابتدا این است که هر یک از این مقادیر باید در یک چهارچوب مشخص باشند واستفاده از توابع این کار را ساده تر می کند.در هر یک از این توابع یک سری محدودیت ها با استفاده از کتابخانه regex تعریف شده است.

class User:

    def \_\_init\_\_(self):

        self.firstName=None

        self.lastName=None

        self.nationalId=None

        self.phoneNumber=None

        self.userName=None

        self.password=None

        self.city=None

        self.email=None

        self.birthDate=None

        self.securityQAnswer=None

        self.savedcities=['Tehran','Isfahan','Mashhad','Tabriz','Shiraz','Karaj','Qom','Yazd','Rasht','Sari']

برای نام و نام خانوادگی بررسی کردیم که آیا مجموع تعداد حروف بزرگ و کوچک انگلیسی با تعداد حروف خود رشته برابر هست یا خیر, زیرا در این صورت حتما یک کاراکتر غیر انگلیسی مانند عدد یا حروف دیگر در آن وجود دارد, که این مهم در تابع get\_first\_name و get\_last\_name تعریف و بررسی شده است. همچنین با توجه به اینکه در این فاز مانور تیم ما روی ترمینال بود و در فاز دوم قصد استفاده از رابط گرافیکی را داریم, از کتابخانه rich برای رنگی نشان دادن و تغییر ظاهنی متن ها استفاده کردیم.

    def get\_first\_name(self,firstName):

        if len(re.findall('[a-z]',firstName))+len(re.findall('[A-Z]',firstName)) == len(firstName):

            self.firstName=firstName

        else:

            raise ValueError(Console().print('[bold red]first name must only consist of English letters.'))

        return

    def get\_last\_name(self,lastName):

        if len(re.findall('[a-z]',lastName))+len(re.findall('[A-Z]',lastName)) == len(lastName):

            self.lastName=lastName

        else:

            raise ValueError(Console().print('[bold red]last name must only consist of English letters.'))

        return

برای کد ملی بررسی کردیم که آیا همه مقادیر آن ارقام بین 0 تا 9 هستند یا خیر زیرا کد ملی تنها شامل اعداد است. همچنین با توجه به اینکه کد ملی 10 رقمی است, بررسی کردیم که آیا این شرط برقرار است یا خیر, که این شروط در تابع get\_code\_meli بررسی شده اند.

    def get\_code\_meli(self,nationalId):

        if len(nationalId)==10 :

            if len(re.findall('[0-9]',nationalId))==len(nationalId):

                self.nationalId=nationalId

            else:

                raise ValueError(Console().print('[bold red]National ID only has numbers in it.'))

        else:

            raise ValueError(Console().print('[bold red]National ID must have ten numbers.'))

        return

برای شماره تلفن اما شروط به این صورت بودند که 11 رقمی باشد, حتما از ارقام تشکیل شده باشد و با 09 شروع شده باشد. برای 11 رقمی به طور ساده از طول استفاده کردیم. برای تشکیل از ارقام از متدی مشابه کد ملی استفاده کردیم. اما برای شروع با 09 از متود search در ماژول استفاده کردیم که بررسی میکند آیا یک عبارت با تعریف مشخص شده پیدا می کند یا خیر. همه این شروط در تابع get\_phone\_number پیاده سازی شده اند.

    def get\_phone\_number(self,phoneNumber):

        if len(phoneNumber)==11 and len(re.findall('[0-9]',phoneNumber))==len(phoneNumber) and  bool(re.search('^09.',phoneNumber))==True:

            self.phoneNumber=phoneNumber

        elif len(phoneNumber)!=11:

            raise ValueError(Console().print('[bold red]phone number must have eleven digits.'))

        elif len(re.findall('[0-9]',phoneNumber))!=len(phoneNumber):

            raise ValueError(Console().print('[bold red]phone number must only consist of digits.'))

        elif bool(re.search('^09.',phoneNumber))==False:

            raise ValueError(Console().print('[bold red]phone number must begin with 09.'))

        return

برای نام کاربری با توجه به اینکه می تواند شامل هر عبارتی باشد,شرط خاصی برای آن در نظر نگرفته و به تعریف کردن مقدار آن به کاربر بسنده کردیم.

    def get\_username(self,userName):

        self.userName=userName

برای رمز عبور چندین شرط مختلف وجود دارد که در تابع get\_password بررسی شده اند. در ابتدا بررسی می شود که رمز بیشتر از 6 کاراکتر داشته باشد. سپس بررسی می شود که حداقل شامل یک حرف انگلیسی کوچک, یک حرف انگلیسی بزرگ, یک رقم, و یک کاراکتر خاص(منبع کاراکتر های خاص در پایین) داشته باشد. همچنین برای ورودی گرفتن رمز به صورت ستاره چه در قسمت ثبت نام و چه در قسمت ورود از کتابخانه getpass (source: <https://stackoverflow.com/questions/35805078/how-do-i-convert-a-password-into-asterisks-while-it-is-being-entered)> و با کد :

getpass.getpass('enter password:')

این مدل پیاده سازی می شود.

    def get\_password(self,password):

        if len(password)>=6 and len(re.findall('[a-z]',password))>=1 and len(re.findall('[A-Z]',password))>=1 and len(re.findall("[!#$%&'()\*+,-./:;<=>?@[\]^\_`{|}~]",password))>=1 and len(re.findall('[0-9]',password))>=1:

            self.password=password

        elif len(password)<6:

            raise ValueError(Console().print('[bold red]password must be at least 6 characters long.'))

        elif len(re.findall('[a-z]',password))<1:

            raise ValueError(Console().print('[bold red]password must contain at least one lowercase letter.'))

        elif len(re.findall('[A-Z]',password))<1:

            raise ValueError(Console().print('[bold red]password must contain at least one uppercase letter.'))

        elif len(re.findall("!#$%&'()\*+,-./:;<=>?@[\]^\_`{|}~"),password)<1:

            raise ValueError(Console().print('[bold red]password must contain at least one digit.'))

        elif len(re.findall('[0-9]',password))<1:

            raise ValueError(Console().print('[bold red]password must contain at least one special character (!#$%&\'()\*+,-./:;<=>?@[\\]^\_`{|}~).'))

        return

همچنین برای آنکه مطمئن شویم که کاربر رمز خود را تایید می کند و بعدا آن را فراموش نمی کند, از او درخواست می کنیم که رمز خود را دوباره وارد کند و در صورت عدم تطابق خطای مربوطه نمایش داده خواهد شد. این مورد در تابع check\_repeated\_password بررسی می شود.

    def check\_repeated\_password(self,repeatedPassword):

        if repeatedPassword!=self.password:

            Console.print('passwords do not match',style=red)

همانطور که در بخش قبل دیدیم, یک لیست ده تایی از شهر ها از قبل برای برنامه تعریف شده و به کاربر نمایش داده می شود و کاربر باید از میان آنها انتخاب کند. اگر شهری به غیر موارد گفته شده را انتخاب نماید به او اخطار داده می شود که حتما از میان مقادیر پیش فرض انتخاب کند. در تابع get\_city این شرط بررسی می شود.

    def get\_city(self,city):

        if city in self.savedcities:

            self.city=city

        else:

            raise ValueError(Console().print(f'[bold red]Invalid city. Please choose the city from this list: [cyan]{self.savedcities}.'))

        return

حال در قسمت بعدی ایمیلی که از کاربر دریافت کرده ایم را بررسی میکنیم. ایمیل به صورت ترکیبی از اعداد و حروف به علاوه @ و در نهایت gmail.com یا yahoo.com که دامنه ایمیل هستند می باشند و هر ایمیلی غیر از این مورد از نظر برنامه غیر مجاز است. اعتبار ایمیل از طریق تابع get\_email بررسی می شود.

    def get\_email(self,email):

        if bool(re.findall(r'[A-Za-z0-9]+@(gmail|yahoo)\.com',email))==True:

            self.email=email

        else:

            raise ValueError(Console().print(f'[bold red]Invalid email'))

        return

مورد بعدی که یکی از چالش برانگیز ترین بخش های ثبت نام بود گرفن تاریخ تولد از کاربر است. با وجود اینکه توانایی استفاده از ماژول datetime وجود داشت اما به دلیل حجم پروژه از ایجاد بار اضافی روی این بخش از پروژه خودداری کردم. در این قسمت سال, ماه, و روز تولد بررسی می شوند که سال باید میلادی و بین 1920 تا 2005 باشد و همچنین ماه ها مطابق فرض در سال کبیسه میلادی استاندارد سازی شده اند که شامل 12 ماه هستند.

    def get\_birth\_date(self,birthDate):

        if bool(re.findall('[1-2][0-9][0-9][0-9]/[0-1][0-9]/[0-3][0-9]',birthDate))==True:

            year=birthDate[0:4]

            month=birthDate[5:7]

            day=birthDate[8:10]

            if int(year)>=1920 and int(year)<=2005 and 1<=int(month)<=12:

                max\_days={1:31,2:29,3:31,4:30,5:31,6:30,7:31,8:31,9:30,10:31,11:30,12:31}

                if 1<=int(day)<=max\_days[int(month)]:

                    self.birthDate=birthDate

                else:

                    raise ValueError(Console().print(f'[bold red] invalid day. this month has only {max\_days[month]}'))

            elif int(year)<1920 or int(year)>2005:

                raise ValueError(Console().print(f'[bold red]birth year must be between 1920 and 2005'))

            elif int(month)<1 or int(month)>12:

                raise ValueError(Console().print(f'[bold red]month must be between 1 and 12'))

        else:

            raise ValueError(Console().print(f'[bold red] invalid birth date format'))

        return

همچنین یک سوال امنیتی از کاربر پرسیده می شود که پاسخ آن به عنوان کلیدی برای دسترسی به رمز فراموش شده قابل استفاده خواهد بود که در برنامه ما خودروی مورد علاقه کاربر به دلخواه سوال شده است. از تابع get\_security\_questions\_answer برای ذخیره این مقدار استفاده می شود.

    def get\_security\_questions\_answer(self,answer):

        self.securityQAnswer=answer

در نهایت در صورت موفقیت همه این موارد, اطلاعات مربوط به کاربر در یک فایل csv به نام users.csv ذخیره می شود. برای این منظور از کتابخانه pandas استفاده شد که اطلاعات را در یک دیتافریم ذخیره کرده و به فایل users کانکت می کند تا به عنوان یک ردیف از آن قرار گیرد.

    def save\_csv(self):

        users=pandas.read\_csv('users.csv')

        new\_user={

            'firstName':[self.firstName],

            'lastName':[self.lastName],

            'nationalId':[self.nationalId],

            'phoneNumber':[self.phoneNumber],

            'username':[self.userName],

            'password':[self.password],

            'city':[self.city],

            'email':[self.email],

            'birthDate':[self.birthDate],

            'securityQAnswer':[self.securityQAnswer]

        }

        new\_user=pandas.DataFrame(new\_user)

        users=pandas.concat([users,new\_user])

        users.to\_csv('users.csv',index=False)

regex sources: [https://www.w3schools.com/python/python\_regex.asp](https://www.w3schools.com/python/python_regex.asp%20) [https://stackoverflow.com/questions/89909/how-do-i-verify-that-a-string-only-contains-letters-numbers-underscores-and-da](https://stackoverflow.com/questions/89909/how-do-i-verify-that-a-string-only-contains-letters-numbers-underscores-and-da%20) <https://www.geeksforgeeks.org/how-to-check-a-valid-regex-string-using-python>/

<https://www.geeksforgeeks.org/extracting-email-addresses-using-regular-expressions-python/>

special characters sources: <https://owasp.org/www-community/password-special-characters>

<https://stackoverflow.com/questions/72221125/regex-to-match-string-and-special-characters>

rich source: <https://rich.readthedocs.io/en/stable/introduction.html>

pandas sources: [https://www.w3schools.com/python/pandas/pandas\_dataframes.asp#gsc.tab=0](https://www.w3schools.com/python/pandas/pandas_dataframes.asp%23gsc.tab=0)

<https://stackoverflow.com/questions/66987199/how-to-add-row-of-one-dataframe-to-column-of-another-dataframe>

**ورود (login) :**

در صورتی که کاربر از قبل ثبت نام کرده باشد و بخواهد وارد اکانت خود شد, از این بخش از منو استفاده می کند. در این بخش تنها سه تابع تعریف شده اند که ورود کاربر را ممکن, و در صورت فراموشی رمز عبور کاربر می تواند از آپشن مربوطه استفاده کند.

در ابتدا به تابع login\_user می پردازیم. این تابع مطابق نام کاربری و رمز عبور وارد شده را با ورودی کاربر بررسی می کنند و در صورت مطابق به او اجازه ورود می دهد( سیگنال درستی را به برنامه ارسال می کند), و در غیر این صورت به عنوان یک عدم ورود لحاظ می شود و سیگنال نادرستی را به برنامه ارسال می کند.

def login\_user(userName, passWord):

    users = pandas.read\_csv('users.csv')

    for \_, user in users.iterrows():

        if user['userName'] == userName and user['password'] == passWord:

            return True

    return False

نکته حائز اهمیت این است که اگر کاربر رمز را اشتباه وارد می کند, یک شمارشگر وجود دارد که در صورتی که کاربر 3 بار یا مضربی از 3 بار رمز را اشتباه وارد کند, برنامه به مدت یک دقیقه متوقف شود و بعد از آن کاربر بتواند دوباره مقادیر را وارد کند. این کار با استفاده از تابع count\_wrong\_enters و به کمک ماژول time اعمال می شود.

def count\_wrong\_enters(count):

    if count%3==0:

        Console.print(Text('try again in one minute!'),style=red)

        time.sleep(60)

    else:

        pass

در نهایت ما می خواهیم که کاربر در صورت فراموشی رمز عبور خود, با استفاده از متود تعریف شده در تابع forgot\_password, ایمیل یا یوزرنیم خود را وارد می کند. در صورت وجود داشتن چنین کاربری, سوال امنیتی از کاربر پرسیده می شود و در صورت ارائه پاسخ صحیح, رمز کاربر به او نمایش داده می شود. در غیر این صورت در هر مورد خطای مربوطه چاپ خواهد شد.

def forgot\_password(userNameOrEmail,securityQAnswer):

    users = pandas.read\_csv('users.csv')

    users['userName'] = users['userName'].astype(str).str.lower()

    users['email'] = users['email'].astype(str).str.lower()

    matching\_users = users[

        (users['userName'] == userNameOrEmail.lower()) |

        (users['email'] == userNameOrEmail.lower())

    ]

    if not matching\_users.empty:

        user = matching\_users.iloc[0]

        if user['SecuityQAnswer'] == securityQAnswer:

            Console().print(Text(f"Your password is: {user['password']}"), style=green)

        else:

            Console().print(Text("Security question answer is wrong!"), style=red)

    else:

        Console().print(Text("Username or email does not exist!"), style=red)