UNIVERZITET U NOVOM SADU
PRIRODNO-MATEMATIČKI FAKULTET
DEPARTMAN ZA MATEMATIKU I INFORMATIKU

Stefan Kalafatic, 113/19

Sistem za podršku vremenskog planiranja u zdravstvenoj ustanovi

- seminarski rad iz predmeta Skript jezici-

Novi Sad, 2020.

Sadržaj

Sadržaj

Uvod

Sistem za podršku vremenskog planiranja u zdravstvenoj ustanovi je jednostavna aplikacija napisana u programskom jeziku Python kao seminarski rad u okviru teme Skript jezici. Napisan je sa ciljem da olakša i ubrza organizaciju vremena u zdravstvenoj ustanovi.

Opis programa

U programskom jeziku Python napisan je program pod nazivom "Sistem za podršku vremenskog planiranja unutar zdravstvene ustanove".

U ovakvom softverskom sistemu učesnici su
- -Administrator
- -Lekar
- -Pacijent

Svaki korisnik sistema, bez obzira na ulogu opisan je sledećim podacima:
- -Korisničko ime
- -Lozinka
- -Ime
- -Prezime

Pored osnovnih poodataka u zavisnosti od uloge, neki korisnici su opisani dodatnim podacima.
Lekar:
- -Spisak zakazanih pregleda
- -Lista pacijenata

Pacijent:
- -JMBG
- -Lista pregleda

U nastavku ćemo detaljnije opisati svaku od datih funkcionalnosti.

Class Admin

Administrator sistema rukuje mnogim funkcionalnostima kao što su: dodavanje/zapošljavanje novih lekara, štampanje svih lekara, brisanje lekara/otpuštanje, štampanje liste lekara sa šiframa i iscrtavanje grafika. Krenućemo sa opisom prve (Figure 1.).

```
195  class Admin:
196
197
198      lista_lekara = []
199
200
201
202      def dodavanje_lekara(self):
203          from random import randrange
204
205          rand = randrange(999, 10000)
206
207          ime = input("Unesite ime lekara: ")
208          prezime = input("Unesite prezime lekara: ")
209          ime = ime.capitalize()
210          prezime = prezime.capitalize()
211          s = ime + "-" + prezime
212          ok = False
213          for i in self.lista_lekara:
214              if(s == i):
215                  ok = True
216              else:
217                  continue
218          if(ok):
219              print("Lekar sa tim imenom vec postoji.")
220          else:
221              fajl = open(ime + prezime + "_lista_pregleda.txt","w+")
222              fajl.close()
223
224              fajl = open(ime + prezime + "_lista_pacijenata.txt","w+")
225              fajl.close()
226
227
228              fajl = open("Lista lekara.txt","a")
229              fajl.write("\n" + ime + prezime  +"/"+ str(rand) +  "\n")
230              fajl.close()
231              print()
232              print("Šifra Vašeg lekara je " + str(rand) + ", a username - " + ime + prezime)
233              self.lista_lekara.append(str(s))
```

Figure 1 dodavanje_lekara()

Funkcija `dodavanje_lekara()` može da registruje novog korisnike sistema (Lekara). Prilikom registracije novih korisnika administrator unosi: ime i prezime koji su ujedno i jedinstveno korisničko ime dok se automatski generiše lozinka.

Prilikom „zapošljavanja“ novog lekara, njemu se otvaraju 2 fajla: prvi u kojima ce čuvati listu svojih pacijenata, drugi u kom će imati uvid u svoje preglede.

Sledeća funkcija je `stampanje_liste_lekara()` (Figure 2.) u kojoj prolazimo kroz listu lekara i stampamo svaki element te liste.

```
199
200     def stampanje_liste_lekara(self):
201         for n in self.lista_lekara:
202             print(n)
203
```

Figure 2. stampanje_liste_lekara()

U funkciji `brisanje_lekara()`(Figure 3.) unosimo ime lekara kog zelimo da otpustimo. Pošto su u listi lekara oni zapisani u obliku IME-PREZIME, sa funkcijom split() uzimamo ime i prezime tog lekara da bi mogli da obrisemo njegove fajlove sa listom pregleda i listom pacijenata. Datog lekara brisemo i iz liste i iz fajla Lista_lekara.txt.

```
241     def brisanje_lekara(self):
242         import os
243         ime = input("Unesite kog lekara želite da otpustite ")
244         ime = ime.replace(" ", "")
245
246
247
248         for n in self.lista_lekara:
249             sp = n.split("-")
250             if((sp[0] + sp[1]) == ime):
251                 if os.path.exists(ime + "_lista_pregleda.txt"):
252                     os.remove(ime + "_lista_pregleda.txt")
253                 if os.path.exists(ime + "_lista_pacijenata.txt"):
254                     f = open(ime + "_lista_pacijenata.txt", "r")
255                     for line in f.readlines():
256                         s = line.strip().split("|")
257                         if os.path.exists(ime + "_lista_pacijenata.txt"):
258                             os.remove(s[0] + s[1] + ".txt")
259                     f.close()
260                     os.remove(ime + "_lista_pacijenata.txt")
261                     self.lista_lekara.remove(n)
262             self.brisanje_lekara_iz_fajla(ime)
263
264
```

Figure 3. brisanje_lekara()

Fukcija `brisanje_lekara_iz_fajla()` je pomoćna funkcija koja se koristi u gore navedenoj `brisanje_lekara()` koja brise datog lekara iz fajla, tako što ispisuje novi (ustvari isti taj fajl) bez tog lekara.(Figure 4.)

```
266     def brisanje_lekara_iz_fajla(self,ime):
267
268         fajl = open("Lista lekara.txt", "r")
269         lines = fajl.readlines()
270         fajl.close()
271
272         novi_fajl = open("Lista lekara.txt", "w")
273
274         for line in lines:
275             if ime not in line.strip("123456789|\n"):
276                 novi_fajl.write(line)
277         print("Lekar otpušten!")
278         novi_fajl.close()
279
280
```

Figure 4. brisanje_lekara_iz_fajla()

Sledeca nešto jednostavnija funkcija je `stampanje_liste_lekara_sa_siframa()`. Radi na principu otvaranja fajla "Lista lekara.txt" i štampa sve lekare sa njihovim šiframa. (Figure 5.).

```
249     def stampanje_liste_lekara_sa_siframa(self):
250         lista_lekara = open("Lista lekara.txt" , "r")
251         for linija in lista_lekara:
252             strip_linija = linija.strip()
253             print(strip_linija)
254         lista_lekara.close()
```

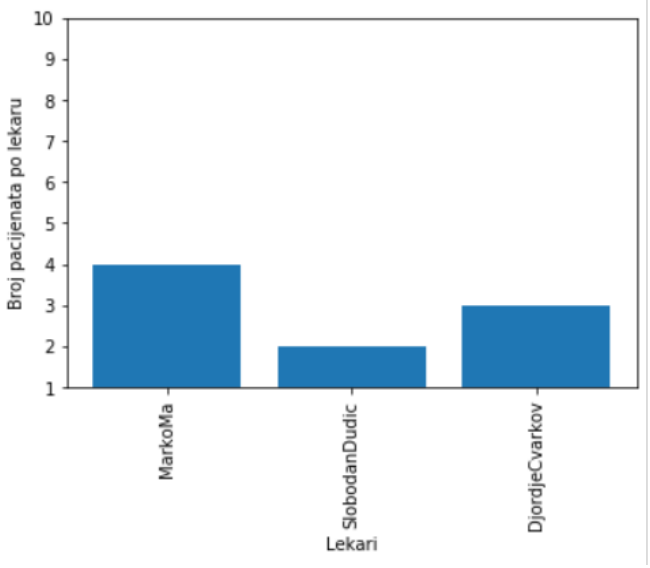
Figure 5. stampanje_liste_lekara_sa_siframa()

Takođe jos jedna funkcionalnost koju poseduje klasa Admin, koja je ujedno i poslednja, je iscrtavanje stubičastog grafika- funkcija `grafik()`. Iscrtava odnos lekara i njegov broj pacijenata. (Figure 6.)

```
272     def grafik(self):
273         import matplotlib.pyplot as plt
274         x_podaci = []
275         y_podaci = []
276         f = open("Lista lekara.txt", "r")
277         for line in f.readlines():
278             s = line.strip().split("|")
279             x_podaci.append(s[0])
280             fajl = open(s[0] + "_lista_pacijenata.txt","r")
281             lista = fajl.readlines()
282             print(len(lista))
283             y_podaci.append(len(lista))
284             fajl.close()
285         plt.bar(x_podaci, y_podaci)
286         plt.xlabel('Lekari')
287         plt.xticks(rotation=90)
288         plt.ylabel('Broj pacijenata po lekaru')
289         plt.ylim(ymin=1, ymax=30)
290         plt.show()
```

Figure 6. grafik()

Podatke za iscrtavanje grafika dobija prvo otvaranjem fajla „Lista lekara.txt“, u kom uzima ime prvog lekara, zatim otvara fajl tog lekara i uzima sve pacijente kojima je on lekar. I tako za svakog lekara koji je zaposlen u bolnici.



Primer Grafika

Funkcija koja nam pomaže da učitamo sve prethodno unete informacije, kada započinjemo ponovo korišćenje programa, je `loadAdin()`. (Figure 7.)

```
293     def loadAdmin(self):
294         import re
295         self.lista_lekara.clear()
296         fajl = open("Lista lekara.txt","r")
297         for line in fajl.readlines():
298             if(line == "\n"):
299                     continue
300             if("/" not in line):
301                 continue
302             s = line.strip().split("/")
303             user = s[0]
304             lista = re.findall('[A-Z][^A-Z]*', user)
305             lista[1] = lista[1].split("/")
306             self.lista_lekara.append(str(lista[0] + "-" +  lista[1][0]))
307         fajl.close()
308
```

Figure 7. loadAdin()

Class Lekar

Takođe, kao i Administrator, lekar ima različite ugrađene funkcionalnosti. Zapošljavanjem novog lekara otvara se lista pacijenata, i lista pregleda (pregledi se deklarišu kao rečnici).

```
7   class Lekar:
8
9
10
11      def __init__(self, ime, prezime, lozinka):
12          self.ime = ime
13          self.prezime = prezime
14          self.spisak_zakazanih_pregleda = []
15          self.lozinka = lozinka
16          self.lista_pacijenata = []
17
```

Figure 1 Prvi deo dodavanje_pacijenta()

Dodavanje pacijenta, zakazivanje pregleda, štampanje pregleda za neki datum, štampanje svih pregleda, postavi dijagnozu pacijenta, štampaj pacijente sa dijagnozama su sve mogucnosti kojima može da se koristi zaposleni lekar.

Prva funkcija pod imenom `dodaj_pacijenta()` je komplikovanija od ostalih, stoga ćemo je podeliti u tri dela

U prvom delu od korisnika se traže ime i prezime pacijenta, kao i njegov jmbg. Sledeći korak je dodavanje pacijenta u lekarov fajl (fajl pacijenta koji je memorisan kao njegovo ime i prezime). Proverava se da li fajl postoji, ako postoji upisuje se ime i prezime pacijenta kao i njegov jmbg.(Figure 1)

```
61      def dodavanje_pacijenta(self):
62          import os
63
64          ime = input("Unesite ime pacijenta: ")
65          prezime = input("Unesite prezime pacijenta: ")
66          jmbg = input("Unesite jmbg pacijenta: ")
67          ime = ime.capitalize()
68          prezime = prezime.capitalize()
69
70          ok = os.stat(self.ime + self.prezime + "_lista_pacijenata.txt").st_size == 0
71          if(ok):
72              fajl = open(self.ime + self.prezime + "_lista_pacijenata.txt","a")
73              fajl.write(ime +"|" + prezime  +"|"+ jmbg + "|"  + "\n")
74              f = open(ime+prezime + ".txt","w+")
75              f.close()
76              fajl.close()
77              self.lista_pacijenata.append(self.pomocna(ime,prezime,jmbg))
78              return True
```

Figure 1 Prvi deo dodavanje_pacijenta()

U drugom delu proveravamo da li taj pacijent postoji. Ponovo otvaramo fajl sa listom pacijenata za datog lekara i za svaku liniju fajla ispitujemo da li se poklapa sa novim imenom.(Figure 2)

```
79          fajl = open(self.ime + self.prezime + "_lista_pacijenata.txt","r")
80          lines = fajl.readlines()
81          for line in lines:
82                  if(line == "\n"):
83                      continue
84                  s = line.strip().split("|")
85                  if (ime == s[0]) and (s[1] == prezime) and (s[2] == jmbg):
86                      print("Pacijent vec postoji! ")
87                      ok = True
88                      break
```

Figure 2 Drugi deo dodavanje_pacijenta()

U trećem, ujedno i poslednjem delu funkcije `dodavanje_pacijenta()`, pošto smo prošli proveru jedinstvenosti pacijenta, otvaramo mu fajl u kojem će kasnije biti smešteno: ime lekara, vreme, datum i cena pregleda. Takođe, pojavljuje se dodatna funkcija `pomocna()` koju ćemo kasnije detaljnije opisati(Figure 3.)

```
                    break
        if(ok != True):
            fajl = open(self.ime + self.prezime + "_lista_pacijenata.txt","a")
            fajl.write(ime +"|" + prezime  +"|"+ jmbg + "\n")
            f = open(ime+prezime + ".txt","w+")
            f.close()
            fajl.close()
            self.lista_pacijenata.append(self.pomocna(ime,prezime,jmbg))
            print()
            print("Šifra Vašeg pacijenta je njegov jmbg, a username - " + ime + prezime)
            return True

```

Figure 3 Treći deo dodavanje_pacijenta()

Funkciju `zakazivanje_pregleda()` smo spomenuli u prethodnoj. Ona omogućava lekaru da fajlu (kartonu) pacijenta dodeli: ime lekara, datum, vreme i cenu pregleda.

```
    def zakazivanje_pregleda(self,datum,vreme):
        ime = input("Unesite ime pacijenta: ")
        ime = ime.replace(" ","")
        prezime = input("Unesite prezime pacijenta: ")
        prezime = prezime.replace(" ","")
        ok = True
        for d in self.lista_pacijenata:
           i = d.get('Ime')
           p = d.get('Prezime')
           if((i == ime) and (p == prezime)):
               d["Datum"] = datum
               d["Vreme"] = vreme
               d["Cena"] = input("Unesite cenu pregleda")
               fajl = open(self.ime + self.prezime + "_lista_pregleda", "a")
               fajl.write(ime + "-" + prezime + "|" + datum + "|" + vreme)
               fajl.close()
               self.spisak_zakazanih_pregleda.append(d)
               f = open(ime + prezime + ".txt", "a")
               f.write(self.ime +"|" + self.prezime  +"|"+ str(datum) + "|" + str(vreme) + "|" + d["Cena"] + "\n")
               f.close()
               ok = False
        if(ok):
            print("Molimo prvo dodajte pacijeta, pa onda zakazite pregled.")


```

Figure 4 zakazivanje_pregleda()

Slede dve veoma slične funkcije koje cemo ovde opisati.

Prva, `stampaj_spisakk_pregleda()`, stampa spisak pregleda za zadati datum. Importujemo modul date time sa kojim smo memorisali datum i kroz listu rečnika (spisak_zakazanih_pregleda) proveravamo sve preglede koje imaju zajednički ključ (datum) pa ih štampamo.(Figure 5)

Dok druga štampa svaki pregled (rečnik) iz liste spisak_zakazanih_pregleda.

```
128     def stampaj_spisakk_pregleda(self):
129         import datetime
130         g = eval(input("Za koju godinu zelite da proverite preglede? "))
131         m = eval(input("Za koji mesec zelite da proverite preglede? "))
132         d = eval(input("Za koji dan zelite da proverite preglede? "))
133         d = datetime.date(g, m, d)
134         d = str(d)
135         for n in self.spisak_zakazanih_pregleda:
136             print(n.get("Datum"))
137             print(d)
138             if(n.get("Datum") == d):
139                 print(n.get("Ime") + n.get("Prezime") + ": " + str(n.get("Datum")) + "-" + str(n.get("Vreme")))
140 
141     def stampaj_spisak_svih_pregleda(self):
142         ok = True
143         for n in self.spisak_zakazanih_pregleda:
144             print(n.get("Ime") + n.get("Prezime") + ": " + str(n.get("Datum")) + "-" + str(n.get("Vreme")))
145             ok = False
146         if(ok):
147             print("Trenutno nema zakazanih pregleda.")
148 
```

Figure 5 stampaj_spisakk_pregleda()

Sistem ne bi imao smisla ako se pacijentu ne mogu dodeljivati dijagnoze. To je rešeno funkcijom `postavi_dijagnozu()`, koja "prolazi" kroz listu pregleda (lista rečnika) pronalazi dato ime i prezime pacijenta kojem zelimo postaviti dijagnozu i dodeljuje mu je.(Figure 6)

```
149     def postavi_dijagnozu(self):
150         ime = input("Unesite ime pacijenta: ")
151         ime = ime.replace(" ","")
152         prezime = input("Unesite prezime pacijenta: ")
153         prezime = prezime.replace(" ","")
154         ok = True
155         for d in self.spisak_zakazanih_pregleda:
156             i = d.get('Ime')
157             p = d.get('Prezime')
158             if((i == ime) and (p == prezime)):
159                 dijagnoza = input("Unesite dijagnozu pacijenta: ")
160                 d["Dijagnoza"] = dijagnoza
161                 ok = False
162                 break
163         if(ok):
164             print("Ne postoji uneti pacijent!")
165
```

Figure 6 postavi_dijagnozu()

Takodje funkcijom `stampaj_pacijente_sa_dijagnozama()`, možemo videti sve pacijente I njihove dijagnoze.(Figure 7)

```
142     def stampaj_pacijente_sa_dijagnozama(self):
143         for d in self.spisak_zakazanih_pregleda:
144             ime = d.get('Ime')
145             prezime = d.get('Prezime')
146             print (ime + prezime + " : %s" %  d.get('Dijagnoza'))
147
```

Figure 7 stampaj_pacijente_sa_dijagnozama()

Poslednja funkcija koja je veoma bitna je `loadLekar()` . Prvo proveravamo da li taj fajl uopšte ima pacijenta, ako ima učitavamo ih. Učitavamo listu pacijenata iz fajla tog lekara, zatim pomoću imena tog pacijenta pronalazimo njegov fajl sa njegovim pregledima i učitavamo ih u listu spisak_zakazanih_pregleda.(Figure 8)

```
    def loadLekar(self,ime,prezime):
        import os
        ok = os.stat(ime + prezime + "_lista_pacijenata.txt").st_size == 0
        if(ok):
            pass
        else:
            fajl = open(ime + prezime + "_lista_pacijenata.txt","r")
            for line in fajl.readlines():
                s = line.strip().split("|")
                self.lista_pacijenata.append(self.pomocna(s[0], s[1], s[2]))
                f = open(s[0] + s[1] + ".txt", "r")
                linija = f.readlines()
                for i in linija:
                    p = line.strip().split("|")
                    self.spisak_zakazanih_pregleda.append(self.pomocna2(s[0], s[1], s[2], p[0], p[1], p[2]))

```

Figure 8 loadLekar()

Class Pacijent

Klasa Pacijent nije složena kao ostale.Pacijent ima svoju listu pregleda kao i jednu funkciju koja otvara fajl pacijenta, i stampa sve preglede za datog pacijenta. (Figure 1)

```
class Pacijent:


    def __init__(self, ime, prezime, jmbg):
        self.ime = ime
        self.prezime = prezime
        self.jmbg = jmbg
        self.lista_pregleda = []

    def stampaj_listu_pregleda(self):
        f = open(self.ime + self.prezime + ".txt", "r")
        for line in f.readlines():
            print(line)
        f.close()
```

Figure 1 stampaj_listu_pregleda()

Main()

Počinjemo program pozivom funkcije `main()`. U funkciji `main()` postoje 3 opcije: prijava kao admin, prijava kao lekar i prijava kao pacijent. Prvo se unosi odredjeni karakter u zavisnosti od opcije koju želimo. Zatim se poziva pomocna funkcija `login()` , koju cemo detaljnije opisati u nastavku, koja nam vraca vrednost tipa `tuple()`. "Raspakivanjem" tupla dobijamo 3 vrednosti, da li je prijavljivanje uspesno, ime i prezime korisnika. Proverom prvobinto unetog karaktera kao i vrednosti koje smo raspakovali tupla funkcija nas salje na odgovarajući meni.(Figure 1)

```
449 def main():
450         s = input("Da li se prijavljujete kao admin, lekar ili pacijent? Unesite A za admin, L za lekar, P za pacijent ili K za kraj: ")
451         s = s.lower()
452 
453         touple = login(s)
454         log = touple[0]
455         user = touple[1]
456         passw = touple[2]
457         if ((log == True) and (s == 'a')):
458              menu1()
459         elif((log == True) and (s == 'l')):
460             menu2(user, passw)
461         elif((log == True) and (s == 'p')):
462             menu3(user, passw)
463         else:
464             print("Kraj")
```

Figure 1 main()

Menu1()

Meni Admina:

Ukoliko se korisnik prijavi na meni1(), u mogućnosti je da koristi sve funkcionalnosti kao Admin. Ako odabere neku od opcija: '1', '2', '3', '4', '5', '6' izvršiće se jedan od sledećih metoda:

Opcija '1' – Pozivanje funkcije dodavanje lekara.

Opcija '2' – Štampanje liste lekara

Opcija '3' – Brisanje lekara

Opcija '4' – Štampanje liste lekara sa šiframa

Opcija '5' – Iscrtavanje grafika

Opcija '6' – Pozivanje funkcije logout() i vraćanje na početni meni.

```
def menu1():

        print("Dobrodošli! Admin")
        br = 99
        a = Admin()
        a.loadAdmin()
        while(br!='10'):
            print(br)
            print(" 1 - Dodajte/Zaposlite lekara\n 2 - Štampajte listu lekara\n 3 - Obrišite lekara\n 4 - Štampajte liste lekara sa šiframa\n 5 - Šta
            br = input("Unesite vašu opciju: ")
            if(br == '1'):
                a.dodavanje_lekara()
            elif(br == '2'):
                a.stampanje_liste_lekara()
            elif(br == '3'):
                a.brisanje_lekara()
            elif(br == '4'):
                a.stampanje_liste_lekara_sa_siframa()
            elif(br == '5'):
                a.grafik()
            elif(br == '6'):
                print("Odlogovani ste!")
                br = '10'
                logout()
            else:
                print("Pogresan unos, molimo pokusajte ponovo!")
                br = '99'
```

Figure 1 Menu1()

Menu2()

Meni lekara:

`Meni2()` je nešto komplikovaniji od ostalih, zbog toga sto moramo znati ime i prezime lekara koji se prijavljuje. To smo rešili odvajamo string po velikim slovima. (Figure 1). Elemente koje dobijamo prosledjujemo klasi lekar. Nakon toga pozivamo funkciju loadLekar() koju smo gore opisali.

```
def menu2(user,passw):
        import re
        import datetime
        lista = re.findall('[A-Z][^A-Z]*', user)
        lista[1] = lista[1].split("|")
        l = Lekar(lista[0],lista[1][0],passw)
        l.loadLekar(lista[0],lista[1][0])
```

Figure 15 Menu2()

Nakon toga meni postaje sličan prošlom. Biramo neku od opcija koju program može da izvrši.(Figure 2)

Figure 26 Menu2()

Opcija '1' – Dodavanje novog pacijenta.

Opcija '2' – Zakazivanje pregleda, biramo datum i vreme pregleda i zakazujemo pregled za datog pacijenta.

Opcija '3' – Štampanje spiska pregleda za uneti datum.

Opcija '4' – Štampanje liste svih pregleda za datog lekara.

Opcija '5' – Postavljanje dijagnoze za unetog pacijenta

Opcija '6' – Štampanje svih pacijenta sa njihovim dijagnozama

Opcija '7' – Pozivanje funkcije `logout()` i vraćanje na prvobitni meni.

Menu3()

Meni Pacijenta:

Analogno meniju2, razdvajamo ime i prezime pacijenta i prosledjujemo parametre klasi Pacijent.(Figure 1)

```
379 def menu3(user,passw):
380         import re
381         lista = re.findall('[A-Z][^A-Z]*', user)
382         lista[1] = lista[1].split("/")
383         l = Pacijent(lista[0],lista[1][0],passw)
384 
385         print("Dobrodošli! ")
386         br = '99'
387         while(br!='10'):
388 
389             print(" 1 - Štampaj listu pregleda\n 2 - Logout")
390             br = input("Unesite vašu opciju: ")
391             if(br == '1'):
392                 l.stampaj_listu_pregleda()
393             elif(br == '2'):
394                 print("Odlogovani ste!")
395                 br = '10'
396                 logout()
397             else:
398                 print("Pogresan unos, molimo pokusajte ponovo!")
399                 br = '99'
```

Figure 17 Menu3()

Ponovo korisnik bira neku od opcija:

Opcija '1' – Štampanje liste pregleda pacijenta.

Opcija '2'- Pozivanje funkcije logout() i vraćanje na prvobitni meni.

Login()

Funkcija `login()` je podeljena na tri dela. U zavisnosti od opcije koju je prethodno uneo proverava se dato korisničko ime i lozinka u odgovarajućim fajlovima:

Opcija '1' – Korisnik se loguje kao admin:

Otvara se odgovarajući fajl u kojem je smeštena šifra kao i korisničko ime admina koje je definisano kao – KORISNIČKOIME|PASSWORD. Prolaskom kroz fajl, linija se deli na KORISNIČKOIME i PASSWORD koje se nakon toga proveravaju sa unetim vrednostima. Ukoliko je sve tačno funkcija vraća vrednost True.

Opcija ‘2’ – Korisnik se loguje kao lekar:

Analogno opciji 1.

Opcija ‘3’ – Korisnik se loguje kao pacijent:

Analogno opciji 1.

Zaključak

Softver je baziran na principu objeknto-orijentisanog programiranja, pokriva mnoge mogućnosti rada i olakšava brzinu funkcionisanja zdravstvene ustanove. Poboljšanje programa moglo bi se naći u korišćenju baza podataka umesto fajlova kao i proširivanjem obima funkcionalnosti. Npr. Dodavanjem funkcija za rezervisanje sala, dodavanje novih lica i drugih zaposlenih pored lekara itd… Funkcionalnost ovog programa mogu pronaći razne drzavne pa i privatne zdravstvene ustanove koji bi u potpunosti olakšao rad zaposlenih.

Literatura

1. https://stackoverflow.com/