

## Документација за лабораториска вежба III

### Слики и објаснувања за кодот

```
app.config['MAIL_SERVER'] = 'smtp.sendgrid.net'
app.config['MAIL_PORT'] = 587
app.config['MAIL_USERNAME'] = 'apikey'
app.config['MAIL_PASSWORD'] = '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'
app.config['MAIL_USE_TLS'] = True
app.config['MAIL_USE_SSL'] = False
app.config['MAIL_DEFAULT_SENDER'] = 'tamarageorgieva2001@gmail.com'
app.secret_key = os.urandom(24)
```

Во првиот дел од кодот ги конфигурирам поставките за е-пошта. Се поставуваат подесувањата за SMTP серверот на sendgrid како што се порта, информации за автентикација (корисничко име и лозинка), како и се наведува користење на TLS и се дефинира стандардна е-пошта од која ќе бидат испратени е-маилите.

```
def send_email(email, subject, content):
    message = Mail(
        from_email=app.config['MAIL_DEFAULT_SENDER'],
        to_emails=email,
        subject=subject,
        html_content=content
    )

    try:
        sg = SendGridAPIClient(app.config['MAIL_PASSWORD'])
        response = sg.send(message)
        if response.status_code == 202:
            return True
        else:
            return False

    except Exception as e:
        traceback.print_exc()
        return False
```

Оваа функција претставува делот за испраќање на е-пораки преку SendGridAPI во Flask апликацијата. Функцијата ги користи конфигурациските поставки за е-пошта кои ги поставив во предходната слика.

```
class User(UserMixin):
    def __init__(self, user_id, role, is_vip=False, is_admin=False):
        self.id = user_id
        self.role = role
        self.is_vip = is_vip
        self.is_admin = is_admin

    @staticmethod
    def get(user_id):
        users_file_path = create_users_file()
        admin_file_path = create_admin_users_file()
        vip_file_path = create_vip_users_file()

        with open(admin_file_path, 'r') as admin_file:
            for line in admin_file:
                stored_info = line.strip().split(':')
                stored_username = stored_info[0].lower()
                if stored_username.lower() == user_id.lower():
                    return User(user_id, role='admin', is_admin=True)

        with open(vip_file_path, 'r') as vip_file:
            for line in vip_file:
                stored_info = line.strip().split(':')
                stored_username = stored_info[0].lower()
                if stored_username.lower() == user_id.lower():
                    return User(user_id, role='vip', is_vip=True)

        with open(users_file_path, 'r') as users_file:
            for line in users_file:
                stored_info = line.strip().split(':')
                stored_username = stored_info[0].lower()
                if stored_username.lower() == user_id.lower():
                    if "_admin" in stored_username:
                        return User(user_id, role='admin')
                    else:
                        return User(user_id, role='user')

        return None
```

Оваа класа User се користи за да се создаваат објекти кои ги претставуваат корисниците во системот.

```
def hash_password(password):  
    return bcrypt.hashpw(password.encode('utf-8'), bcrypt.gensalt()).decode('utf-8')
```

Оваа функција, `hash_password(password)`, е дизајнирана да ја хешира корисничка лозинка со корисрење на `bcrypt` алгоритмот. Овој алгоритам работи на принципот на првично претворање на пасвордот во битови, потоа генерира и додава “сол” на пасвордот за поголема безбедност и на крајот го враќа пасвордот како стринг од хешираната верзија на пасвордот.

```
def verify_password(provided_password, stored_password_hash):  
    return bcrypt.checkpw(provided_password.encode('utf-8'), stored_password_hash.encode('utf-8'))
```

Оваа функција `verify_password` проверува дали дадената лозинка (како текст) се совпаѓа со зачуваната хеш вредност на лозинката.

```
def generate_verification_code(email):  
    return secrets.randbelow(1000000)
```

Оваа функција генерира шестцифрен број кој се користи како верификациски код за дадената е-пошта.

```
2 usages  
def is_admin_username(username):  
    return "_admin" in username.lower()  
  
5 usages  
def is_vip_username(username):  
    return "_vip" in username.lower()
```

Тука имаме две слични функции: `is_admin_username` и `is_vip_username`. Првата функција проверува дали во даденото корисничко име постои дел `"_admin"`, додека втората функција проверува дали во корисничкото име постои дел `"_vip"`. Тие враќаат `True` ако соодветниот дел се наоѓа во корисничкото име, во спротивно враќаат `False`.

```
def register_admin(username, password):
    admin_file_path = create_admin_users_file()

    with open(admin_file_path, 'a+') as admin_file:
        admin_file.seek(0)
        lines = admin_file.readlines()

        for line in lines:
            stored_info = line.strip().split(':')
            if len(stored_info) >= 1:
                stored_username = stored_info[0].lower()

                if stored_username == username.lower():
                    return "Registration failed: Username already exists for admin users. Please choose another one."

        hashed_password = hash_password(password)
        admin_file.write(f"{username.lower()}:{hashed_password}\n")
    return "Admin User registered successfully!"

1 usage
def register_vip(username, email):
    vip_file_path = create_vip_users_file()

    with open(vip_file_path, 'a+') as vip_file:
        vip_file.seek(0)
        lines = vip_file.readlines()

        for line in lines:
            stored_info = line.strip().split(':')
            if len(stored_info) >= 2:
                stored_username = stored_info[0].lower()
                stored_email = stored_info[1].lower()

                if stored_username == username.lower():
                    return "Registration failed: Username already exists for VIP users. Please choose another one."
                if stored_email == email.lower():
                    return "Registration failed: Email already exists for VIP users. Please enter another email."

        verification_code = generate_verification_code(email)
        if not send_email(email, subject='Your Verification Code', content=f"Your verification code is: {verification_code}"):
            return "Failed to send verification code"
        vip_file.write(f"{username.lower()}:{email.lower()}:{verification_code}\n")
    return "VIP User registered successfully!"
```

Овие две функции се за регистрирање на admin и VIP корисниците во системот.

register\_admin проверува дали корисничкото име веќе постои во датотеката за администратори и ако не, го зачувува заедно со хешираната лозинка.

register\_vip проверува дали корисничкото име и е-поштата веќе постојат во датотеката за VIP корисници. Ако не постојат, генерира верификациски код и го зачувува заедно со корисничкото име и е-поштата, а потоа праќа е-пошта со верификацискиот код на корисникот.

```

def register(username, password, email):
    file_path = create_users_file()

    is_admin = is_admin_username(username)
    is_vip_user = is_vip_username(username)

    if is_admin:
        result = register_admin(username, password)
        if result == "Admin User registered successfully!":
            return "Admin registered successfully!"
        else:
            return result

    if is_vip_user:
        return register_vip(username, email)

    with open(file_path, 'r') as file:
        lines = file.readlines()
        for line in lines:
            stored_info = line.strip().split(':')
            if len(stored_info) >= 4:
                stored_username, _, stored_email, _ = stored_info[:4]
                if stored_username.lower() == username.lower():
                    return "Registration failed: Username already exists. Please choose another one."
                if stored_email.lower() == email.lower():
                    return "Registration failed: Email already exists. Please enter another email."

    if not re.match(pattern: r'^[\w.-]+@[\w.-]+\.[a-zA-Z]+$', email):
        return "Registration failed: Invalid email address. Please enter a valid email address."

    hashed_password = hash_password(password)
    if not all([
        re.search(pattern: r'[A-Z]*', password),
        re.search(pattern: r'[!@#$%^&*()+}{\[\]\.:;<>.,?/~`"]', password),
        re.search(pattern: r'[0-9]', password)
    ]):
        return "Password must contain at least one uppercase letter, one symbol, and one digit. Please choose another password."

    verification_code = generate_verification_code(email)
    if not send_email(email, subject: 'Your Verification Code', content: f"Your verification code is: {verification_code}"):
        return "Failed to send verification code"

    with open(file_path, 'a') as file:
        file.write(f"{username.lower()}:{hashed_password}:{email}:{verification_code}\n")

    return "User registered successfully!"

```

Оваа функција register има за цел да регистрира нови корисници. Таа прво ги повикува функциите за регистрација на admin и VIP корисници. Потоа ги проверува податоците во текстуралната датотека users.txt, односно дали корисничкото име или е-поштата веќе постојат, го валидира форматот на е-поштата и лозинката, генерира верификациски код и праќа е-пошта за потврда на корисникот. Ако сè е успешно, регистрира нов корисник и го зачувува во датотеката. Ако настане некоја грешка, враќа соодветна порака за неуспешна регистрација.

```
def login_admin(username, password):
    admin_file_path = create_admin_users_file()
    with open(admin_file_path, 'r') as admin_file:
        for line in admin_file:
            stored_admin_username, stored_admin_password = line.strip().split(':')
            if stored_admin_username.lower() == username.lower():
                if verify_password(password, stored_admin_password):
                    user = User.get(username)
                    if user and user.is_admin:
                        login_user(user)
                        return {"redirect": True, "redirect_url": '/admin_section'}
                    else:
                        return "Admin user not found or not authorized."
                else:
                    return "Invalid admin credentials."
        return "Admin user not found."

2 usages
def login_vip(username, entered_code):
    vip_file_path = create_vip_users_file()
    with open(vip_file_path, 'r') as vip_file:
        for line in vip_file:
            stored_info = line.strip().split(':')
            if len(stored_info) >= 3:
                stored_username = stored_info[0].lower()
                stored_code = stored_info[2]

                if stored_username == username.lower():
                    if entered_code == stored_code:
                        user = User.get(username)
                        if user and user.is_vip:
                            login_user(user)
                            return {"redirect": True, "redirect_url": '/vip_section'}
                        else:
                            return "VIP user not found."
                    else:
                        return "Invalid verification code for VIP user."
            else:
                return "Invalid VIP user information in the database."
```

Овие две функции се за логирање на admin и VIP корисници во системот.

login\_admin проверува дали внесеното корисничко име и лозинка се совпаѓаат со податоците за администратори во системот. Ако се совпаѓаат, проверува дали корисникот е администратор и ако е, го најавува во системот и го пренасочува кон делот за администратори.

login\_vip проверува дали внесеното корисничко име и верификацискиот код се совпаѓаат со податоците за VIP корисници во системот. Ако се совпаѓаат, проверува дали корисникот е VIP корисник и ако е, го најавува во системот и го пренасочува кон делот за VIP корисници.

```
def login(username, password, entered_code):
    is_admin = is_admin_username(username)
    is_vip_user = is_vip_username(username)

    if is_admin:
        return login_admin(username, password)
    elif is_vip_user:
        return login_vip(username, entered_code)
    else:
        file_path = create_users_file()
        with open(file_path, 'r') as file:
            for line in file:
                stored_info = line.strip().split(':')
                if len(stored_info) >= 4:
                    stored_username = stored_info[0].lower()
                    stored_password = stored_info[1]
                    stored_code = stored_info[3]

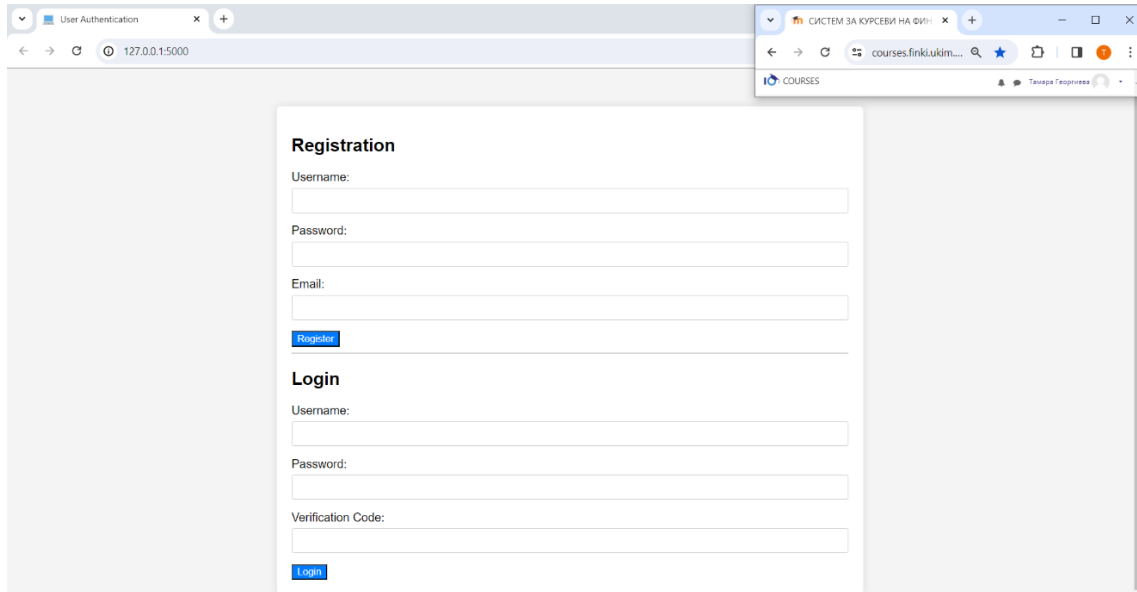
                    if stored_username == username.lower():
                        if verify_password(password, stored_password):
                            if entered_code == stored_code:
                                user = User.get(username)
                                if user and user.role == 'user':
                                    login_user(user)
                                    return {"redirect": True, "redirect_url": '/user_dashboard'}
                                else:
                                    return "User not found."
                            else:
                                return "Invalid verification code for user."
                        else:
                            return "Invalid password for user."
                    else:
                        return "Invalid user information in the database."

        return "Invalid username or verification code."
```

Оваа функција прво ги повикува функциите за најава на admin и VIP корисници. Потоа ги чита податоците од текстуалната датотека users.txt каде што се чуваат информациите за стандардните корисници. За секој ред од датотеката, го споредува внесеното корисничко име со зачуваното во датотеката. Ако корисничкото име е пронајдено, функцијата проверува дали внесената лозинка се совпаѓа со зачуваната лозинка за тоа корисничко име. Ако лозинката е валидна, проверува дали внесениот верификациски код се совпаѓа со зачуваниот за истиот корисник. Во зависност од внесеното, враќа соодветна порака за успешна или неуспешна најава.

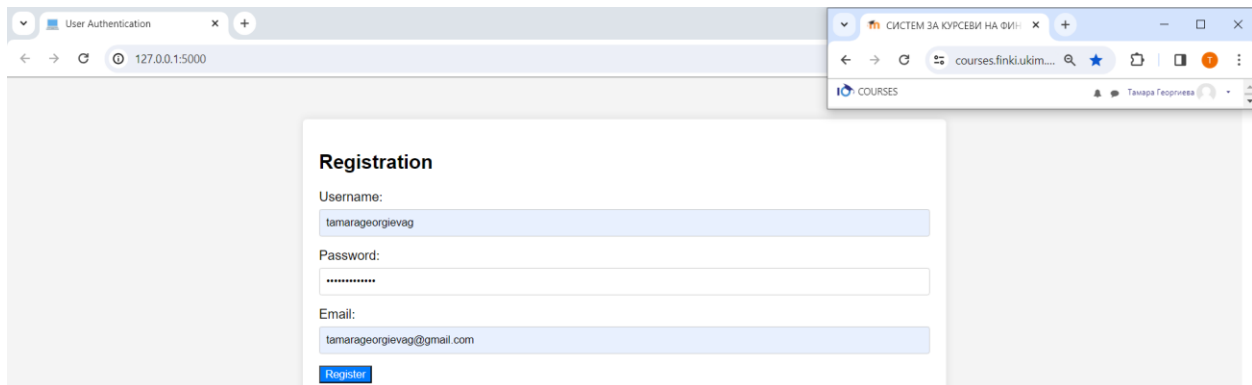
## Слики од изгледот и работата на веб страната

### Воведна страна



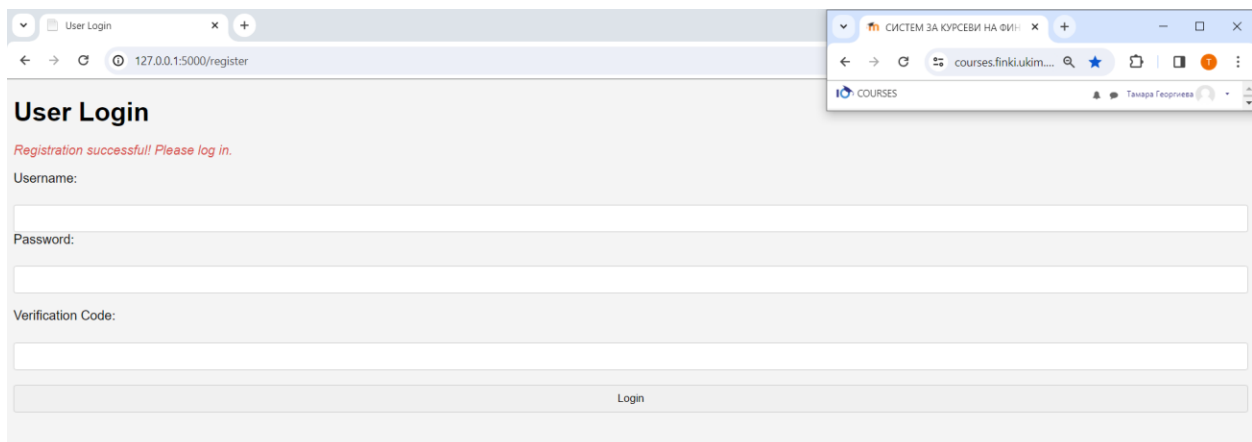
The screenshot shows a web browser window with two tabs. The active tab is titled 'User Authentication' and shows a registration and login form. The form has two sections: 'Registration' and 'Login'. The 'Registration' section has fields for 'Username:', 'Password:', and 'Email:', followed by a 'Register' button. The 'Login' section has fields for 'Username:', 'Password:', and 'Verification Code:', followed by a 'Login' button. The browser's address bar shows '127.0.0.1:5000'.

### Изглед на пополнет дел за регистрација



The screenshot shows the same 'User Authentication' page, but the 'Registration' form is now filled out. The 'Username' field contains 'tamarageorgievag', the 'Password' field contains '\*\*\*\*\*', and the 'Email' field contains 'tamarageorgievag@gmail.com'. The 'Register' button is still visible. The browser's address bar shows '127.0.0.1:5000'.

Откако корисникот успешно ќе се регистрира, веднаш добива можност да се најави



The screenshot shows a web browser window with two tabs. The active tab is titled 'User Login' and shows a login form. The form has fields for 'Username:', 'Password:', and 'Verification Code:', followed by a 'Login' button. Above the form, there is a message: 'Registration successful! Please log in.' The browser's address bar shows '127.0.0.1:5000/register'.



Тамара Георгиева 213207

Ако корисникот не се регистрира точно, веднаш ја добива опција повторно да се обиде да се регистрира

User Registration

Registration failed: Username already exists. Please choose another one.

Username:

Password:

Email:

Register

По успешна регистрација, корисникот добива верификациски код на емаил.

Your Verification Code

tamarageorgieva2001@gmail.com via sendgrid.net  
to me

Your verification code is: 612923

Reply Forward

СИСТЕМ ЗА КУРСЕВИ НА ФИНИ

courses.finki.ukim....

COURSES

Тамара Георгиева

Тамара Георгиева 213207

Откако корисникот ќе го добие кодот за регистрација, тој/таа може да се логира во системот со своето корисничко име, пасворд и верификацискиот код.

The screenshot shows a web browser window with two tabs. The active tab is titled "СИСТЕМ ЗА КУРСЕВИ НА ФИНИ" and has the URL "courses.finki.ukim.mk". The browser's address bar shows "127.0.0.1:5000". The page content is divided into two sections: "Registration" and "Login".

**Registration Form:**

- Username:
- Password:
- Email:
- Register:

**Login Form:**

- Username:
- Password:
- Verification Code:
- Login:

Ако корисникот направи грешка при логирањето, системот ќе му каже дека има грешка и ќе му се даде можност на повторно да се обиде да се логира.

The screenshot shows a web browser window with a single tab titled "User Login". The browser's address bar shows "127.0.0.1:5000/login". The page content is titled "User Login" and displays an error message in red text: "Invalid username or password." Below the error message, there are three input fields for "Username:", "Password:", and "Verification Code:". At the bottom of the form, there is a "Login" button.

User Login

*Invalid admin credentials.*

Username:

Password:

Verification Code:

Login

User Login

*Invalid verification code for VIP user.*

Username:

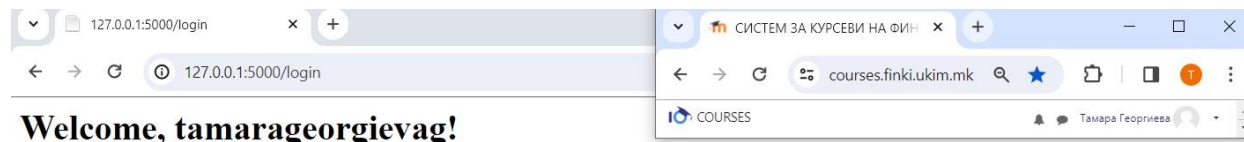
Password:

Verification Code:

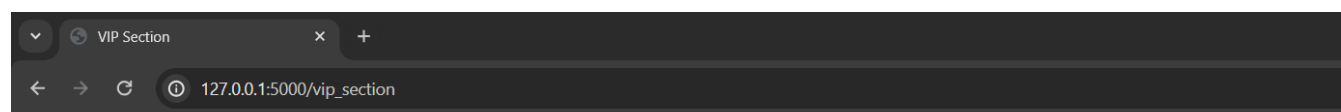
Login

Тамара Георгиева 213207

Ако корисникот успешно се логира, добива адекватна порака за добредојде според нивниот рол.



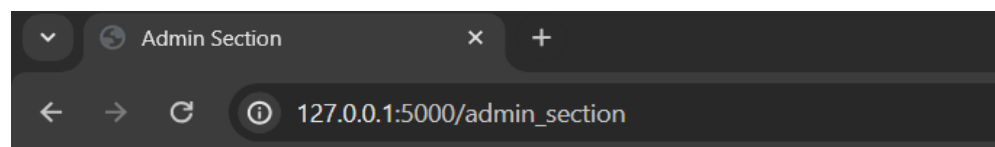
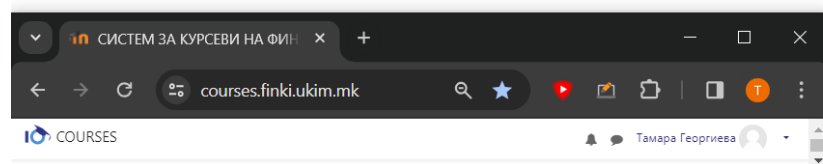
Ако корисникот што се логира во системот е админ или вип, добива посебна порака на своја веб старна која не може да се отвори од други корисници



## Welcome, VIP User!

### Special Features

As a special thank you for supporting us, you don't need to needlessly remember your password when logging in, you can just log in with the code you get on your email!



## Welcome, Admin!

