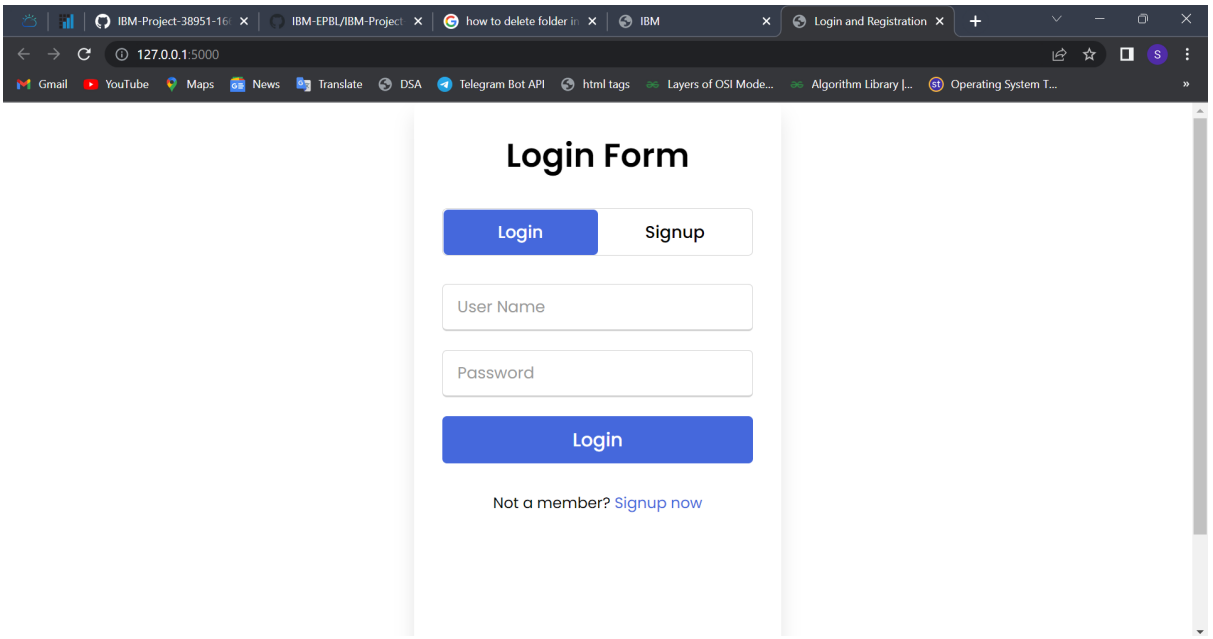
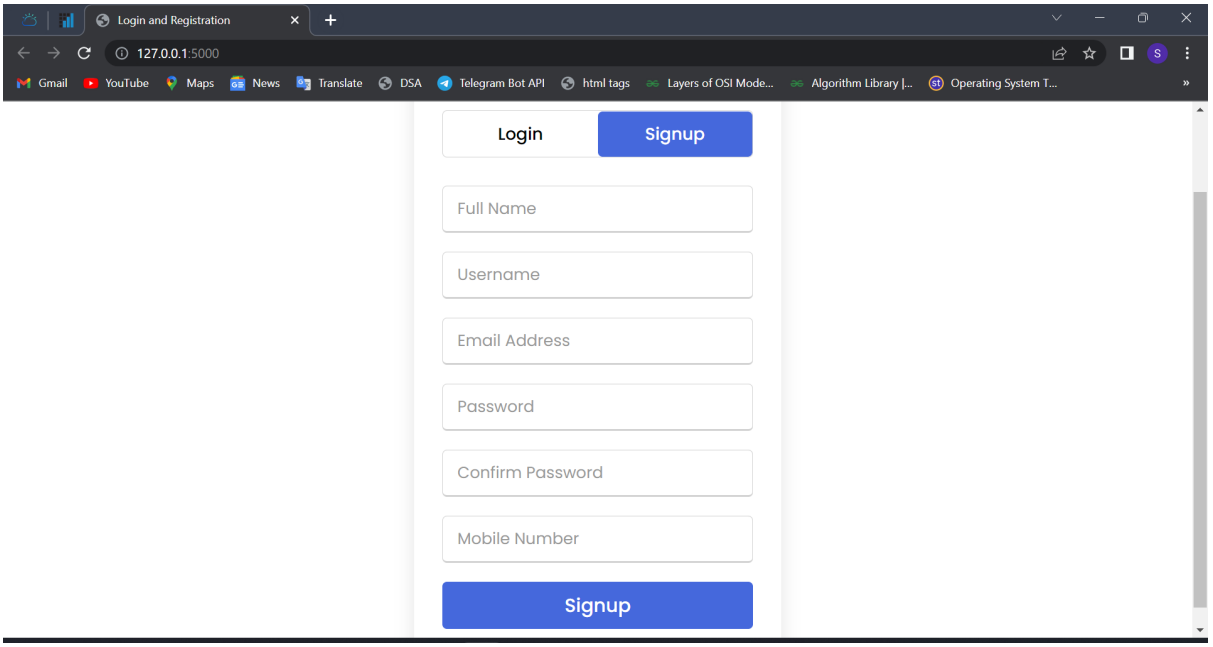
Login and Registration

Date	
Team ID	PNT2022TMID43303
Project Name	Inventory Management System for Retailers





Code

```
@app.route('/login', methods = ['POST'])
def login_check():
    """
    Handles the login process
    :return: Login page
    """

    # capture the username and the password entered by the user and remove trailing and
leading whitespaces
    username = request.form['uname'].rstrip().rstrip()
    entered_password = request.form['password']

    hash_object = sha256(entered_password.encode())
    hashed_entered_password = hash_object.hexdigest()

    # get the expected password corresponding to the username
    expected_password = fetch_password(username)

    if not expected_password:
        # if expected password is not found in the database, it means no such account is
registered yet.
        return render_template('login.html', err_msg = 'Register an account first!')
    elif hashed_entered_password != expected_password:
        # if the entered and expected password don't match, it means the password entered is
wrong.
        return render_template('login.html', err_msg = 'Wrong username or password')
    else:
        # if no errors occur, log the user in and create a session.
        session.update({'username': username})

        # fetch the inventory items corresponding to the user. It's stored in the form of a JSON
as CLOB
        tableContents = fetch_table_contents(username)

        # the value returned above will be of the form list[tuple[str/int/float, str/int/float, ...]]
        if tableContents[0][0] is not None:
            # if the contents are actually present, convert the JSON to python Dict.
            tableContents = loads(tableContents[0][0])
            # print(tableContents['cid'][1])
        else:
            # if no content is present, pass an empty list to the template which will be evaluated
as 'false' by the
            # jinja2 template engine.
            tableContents = []

        return render_template('dashboard.html', greeting=f'Hello, {username}!',
```

```
                        tableContents=tableContents)

    # unreachable code. It's there because I'm paranoid.
    return 'You weren\'t supposed to be here :/'


@app.route('/signup', methods=['POST'])
def signup():
    """
    Handles the signup process.
    :return: Signup Webpage or Dashboard
    """

    # capture the details passed from the form and remove trailing and leading whitespaces.
    fullName = request.form.get('fname').lstrip().rstrip()
    username = request.form.get('uname').lstrip().rstrip()
    email = request.form.get('email').lstrip().rstrip()
    password = request.form.get('password')
    mobile = request.form.get('mobile').lstrip().rstrip()

    hash_object = sha256(password.encode())
    hashed_password = hash_object.hexdigest()

    try:
        # try to create account with the given details
        create_account(fullName, username, email, hashed_password, mobile)
    except IntegrityError:
        # integrity error means either NOT NULL or UNIQUE constraint has been violated.
        # the former being highly unlikely because of the validation checks in the front end.
        return render_template('login.html', err_msg = 'The account exists already. Please log
in.')
    else:
        # if no errors occur, log the user in and create a session.
        # session.update({'username': username})
        # redirect the user to the login procedure, code = 307 specifies to preserve the HTTP
method used originally
        # (POST in this case)
        return redirect(url_for('login_check'), code=307)
        # return render_template('dashboard.html', greeting = f'Welcome, {fullName.split()[0]}!
({username})')

    # unreachable code
    return 'You weren\'t supposed to be here. Please go back'


@app.route('/check-username', methods = ['POST'])
def check_username():
    """
```

```
    Checks if the username is already present in the database or not.
    :return: 'Exists' if present, else return the passed email itself.
    """

    # get the passed email and remove leading and trailing whitespaces, if any.
    passed_username = request.form.get('username').lstrip().rstrip()
    # check if the email exists or not
    exists = check_username_existence(passed_username)

    if exists:
        return 'Exists', 403
    return passed_username, 200


@app.route('/check-email', methods = ['POST'])
def check_email():
    """
    Checks if the email is already present in the database or not.
    :return: 'Exists' if present, else return the passed email itself.
    """

    # get the passed email and remove leading and trailing whitespaces, if any.
    passed_email = request.form.get('email').lstrip().rstrip()
    # check if the email exists or not
    exists = check_email_existence(passed_email)

    if exists:
        return 'Exists', 403
    return passed_email, 200


@app.route('/add-commodity', methods = ['POST'])
def add_commodity():
    # print(request.form)

    details = request.form.to_dict()
    tableContents = fetch_table_contents(session.get('username'))


    if tableContents[0][0] is None:
        tableContents = {field: [] for field in FIELDS}
    else:
        tableContents = loads(tableContents[0][0])

    if details.get('cid') in list(tableContents.values())[0]:
        return f"Item {details.get('cid')} already exists.", 400

    for key, value in tableContents.items():
```

```
        value.append(details.get(key))


    details_json = dumps(tableContents)
    print(details_json)
    print(details)


    try:
        add_details_to_db(session.get('username'), details_json)
    except Exception as exception:
        print(exception)
        return 'Something went wrong, try again', 403
    else:
        return 'Details Added.', 200
        # return redirect(url_for('login_check'), code = 307)

    return 'add req rcvd'
```

Some other codes like handling verification in the front end, storing in the database, verifying in backend etc. have been skipped for the sake of clarity.