Backend Test

1. Write Test Cases for the above.

(i) For Ops User :

1. Upload PPTX File :

- The Ops User uploads a valid PPTX file
- Verify that the file is uploaded successfully

2. Upload DOCX File :

- The Ops User uploads a valid DOCX file
- Verify that the file is uploaded successfully

3. Upload XLSX File :

- The Ops User uploads a valid XLSX file
- Verify that the file is uploaded successfully

4. Upload Invalid File Type :

- The Ops User attempts to upload an unsupported type of file
- Verify that the upload is rejected by the server with the relevant error message

(ii) For Client User :

1. User Registration :

- After signing up, the Client User receives an encrypted URL
- Verify that the registration is successful, and the URL is valid

2. Email Verification :

- The Client User clicks on the link received via email for verification
- Ensure that the user's email is successfully verified

3. User Login :

- Client User logs in with valid credentials
- Verify that the login is successful

4. File Upload :

- Client User uploads a file
- Verify that the file is successfully uploaded

5. Successful File Download :

- Client User uses a valid Assignment ID to access the download API.
- Verify that a secure encrypted URL is returned, and the download link is accessible

6. File Download - Unauthorized Access :

- Another user (not a Client User) attempts to access the download link
- Verify that access is denied with an appropriate error message

7. List Uploaded Files :

- Client User requests the list of all uploaded files
- Verify that the list is returned successfully

2. How do you plan on deploying this to the production environment?

Deployment Plan :

1. Environment :

- Set up separate environments for development, testing, and production.
- Use containerization (Docker) for consistent deployment across environments.

2. Database :

- Choose an appropriate SQL or NoSQL database (e.g. MongoDB).
- Set up database schema and indices for optimal performance.

3. Web Server :

- Deploy the application on a server
- Use a reverse proxy (e.g., Nginx or Apache) for additional security.

4. Security Measures :

- Configure firewalls and security groups.
- Use HTTPS with a valid SSL certificate.
- Regularly update dependencies and perform security audits.

5. Monitoring and Logging :

- Implement logging for errors and access.
- Set up monitoring tools for performance tracking.

6. Scaling :

- Plan for horizontal scaling to handle increased traffic.
- Use load balancers for distributing traffic.

7. Continuous Integration/Continuous Deployment (CI/CD) :

- Set up CI/CD pipelines for automated testing and deployment.

- Use tools like GitLab CI, or GitHub Actions.

8. Backup and Recovery :

- Implement regular backups of the database and file storage.
- Have a disaster recovery plan in place.

9. Documentation :

- Document the deployment process, environment variables, and API documentation.
- Provide clear instructions for maintenance and troubleshooting.

10. Testing in Production :

- Implement a controlled rollout strategy.
- Monitor performance and user feedback in the production environment.