Report

✅ Design Decisions

- **Serverless Architecture**: Reduces operational complexity by using Lambda for logic.
- **Cognito for Authentication**: Simplifies user auth with secure token handling.
- **S3 for File Storage**: Best choice for storing user uploads with signed URLs.
- **SQS + SES**: Enables asynchronous email notifications with scalability.
- **React Frontend + EC2**: Easy to build and deploy, NGINX handles static assets.

⚠️ Challenges

- CORS configuration for API Gateway.
- Mapping Cognito user to internal MySQL database.
- Handling dynamic email targets with SES
- Debugging NGINX rewrite issues when routing SPA paths.

📓 Key Takeaways

- IAM role design is crucial for minimal privilege yet full functionality.
- Designing a custom VPC with separate public and private subnets improved security by exposing only necessary services like the React frontend via EC2, while keeping the database and backend logic protected.
- **ACL Rules for Inbound Traffic**:
  Setting up Network ACLs provided an additional layer of stateless traffic filtering for the subnets. By configuring inbound rules to only allow traffic on specific ports, and blocking all other traffic

- SQS decouples logic and boosts resilience.
- Logging via CloudWatch is vital for debugging Lambda and API Gateway.
- Hosting React on EC2 with NGINX is simple and effective for internal projects.