TWEET APP

IIHT

CONTENTS

1 Problem Statement 2
2 Application Architecture 2
3 Tool Chain 2
4 Business-Requirement: 3
4.1 Presentation 3
4.2 Database, storage 3
4.3 Debugging & Troubleshooting 3
5 Rubrics/Expected Deliverables 4
6 Problem Statement 4
7 Proposed Tweet App Wireframe 5
8 Tweet Component Sketch 5
9 Application Architecture 6
10 Cloud Architecture 6
11 Tool Chain 7
12 Business-Requirement: 8
13 Rubrics/Expected Deliverables 9
13.1 Rest API (Products & Frameworks -> Compute & Integration): 9
13.2 Database (Products & Frameworks -> Database & Storage): 9
13.3 Maven (Tooling): 10
13.4 Messaging (Products & Frameworks -> Compute & Integration): 10
13.5 Log/ Monitoring (Products & Frameworks -> Governance & Tooling): 10
13.6 Debugging & Troubleshooting 10
13.7 Code Quality 10
13.8 Secured Coding 11
14 Platform 11
14.1 Compute 11
14.2 Compute, Identity & Compliance, Security& Content Delivery 11
15 Methodology 11
15.1 Agile 11

1 Problem Statement

Tweet App is app for registered users to post new tweets, view tweets by other users.

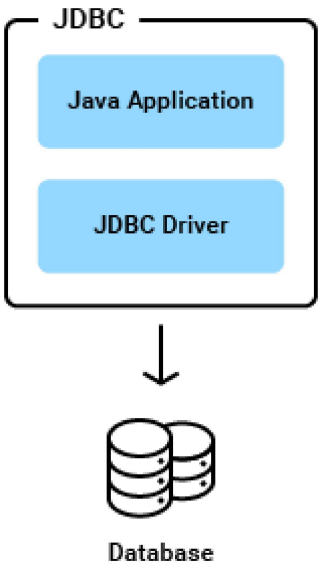
Guest users cannot see any tweets.

The core modules of tweet app are:

1. User registration and login
2. Post tweet
3. View all tweets (all from logged in users account)
4. View users and their respective tweets
5. Forgot password option to reset password.

   Reset password algorithm should be written using typescript and transpile into javascript.
   JavaScript containing reset password algorithm, should be executed by java when reset password is required.

2 Application Architecture



3 Tool Chain

Competency	Skill	Skill Detail
Programming Languages	Application Language	Java
		Typescript Javascript
	Data structures and Algorithms	Data structures and Algorithms
	Database Access	SQL

4 Business-Requirement:

Write a program for Tweet App with below guidelines:

4.1 Presentation

1. A user can register and login.
2. Use below details while registration:
    a. first_name (required)
    b. last_name (optional)
    c. gender (required)
    d. dob (optional, dd-MM-yyyy)
    e. email (required, email will must be used as login-id so must be unique)
    f. password (required)
3. A logged-in user can change his password (old password is required).
4. A logged-in user:
    a. Can post new tweet.
    b. Can view all his tweets.
    c. Can view all registered users and their respective tweets.
5. A user can reset old password.

4.2 Database, storage

1. As the twitter application I should –
    a. Create user when successful registration is done
    b. Create new tweet when user tweets
    c. Find all uses and their tweets
    d. Update user's password

4.3 Debugging & Troubleshooting

1. As the programmer I should:
    a. Debug & Troubleshoot the code which has bugs with breakpoints / pause
    b. Ensure optimal resource utilization & close unneeded connections
    c. Ensure all unhandled exceptions are addressed
    d. Query the database only for the necessary information (user info/tweets etc) with proper indexes & order.

5 RUBRICS/EXPECTED DELIVERABLES

- Implement java 8 features – lambda, stream, map, etc.
- Email id should be used as username/userid
- Use db.properties to store database parameters.
- Package Structure for project will be like com.tweetapp.* with proper naming conventions for package and beans.
- Use dao layer, service layer.
- Use proper layering – database interaction code should be only in DAO classes. DAO classes should not have UI/Business logic. Service classes will have all the business logic.
- Input validations must be handled
- Use custom exception and show a valid error message to user if something goes wrong.
- Use javascript algorithm to reset password. JavaScript code to be executed when reset password is required.
- Build the project.

6 PROBLEM STATEMENT

You've already made a miniature version of the Tweet App in component 1. As a part of component 2, you'll be enhancing this app with a complete middleware & framework. The updated specs are as listed below

Tweet App is an application for registered users to post new tweets, reply to tweets, like/unlike tweets.

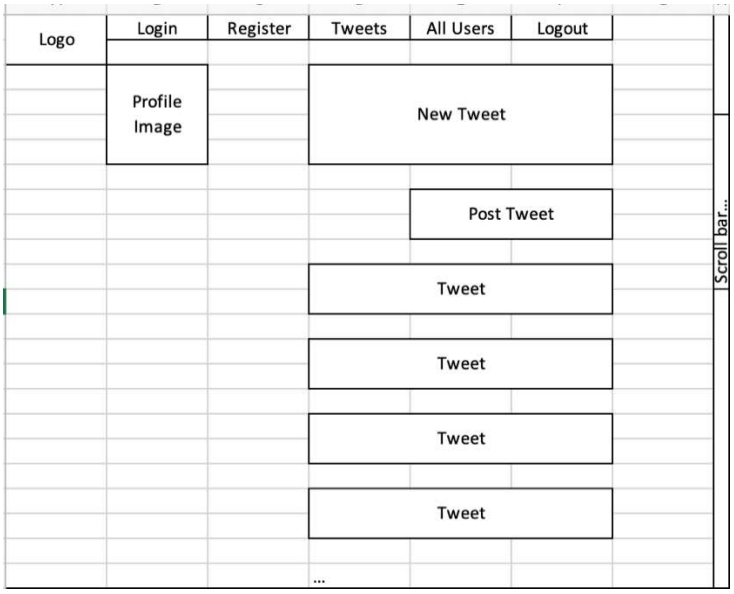
Guest users cannot see any tweets.

The core modules of tweet app are:

1. User registration and login
2. Post tweet
3. View users/handles and their respective tweets

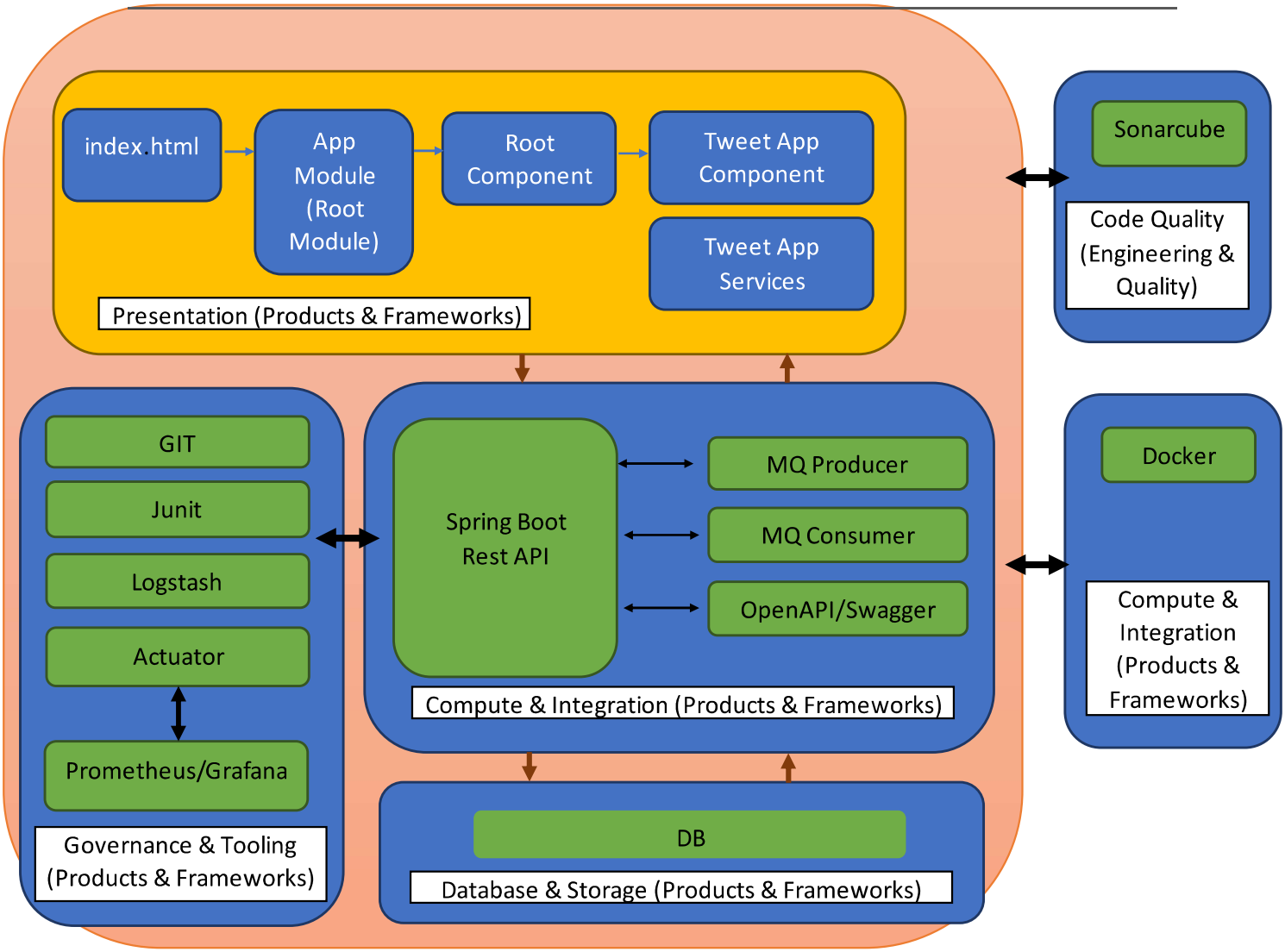
The scope includes developing the application using toolchain mentioned below.

7 PROPOSED TWEET APP WIREFRAME

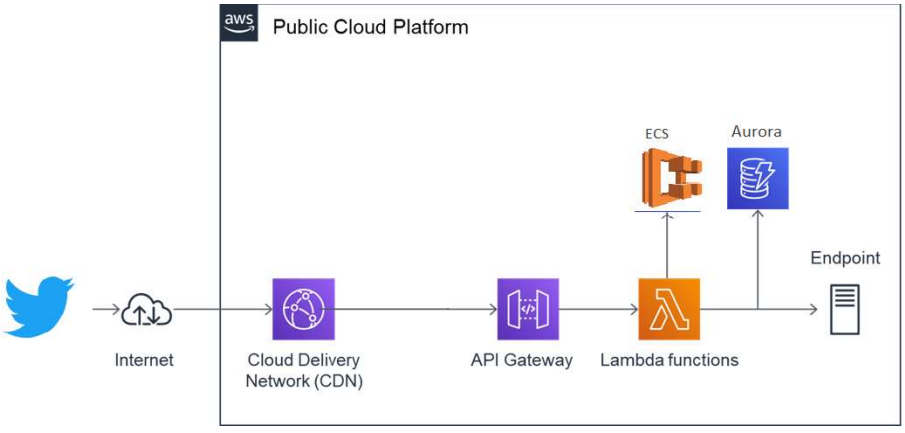
1. UI needs improvisation and modification as per given use case.
2. 

8 TWEET COMPONENT SKETCH

9 Application Architecture



10 Cloud Architecture



11 TOOL CHAIN

Competency	Skill	Skill Detail
Engineering Mindset	Networking and Content Delivery	
	Ways of Working	
	Consulting Mindset	
	DevOps	
Programming Languages	Application Language	Java
Products & Frameworks	Presentation	Angular
		Karma & Jasmine
	Compute & Integration	Spring Boot
		Kafka/RabbitMQ/ActiveMQ/KubeMQ
		Docker
	Database & Storage	MongoDB
	Governance & Tooling	Git
		Maven
		Junit
		Mockito
		Logstash
		Prometheus & Grafana
Engineering Quality	Code Quality	Sonar Cube
Platform	Cloud Tools	AWS ECS
		AWS DynamoDB/Aurora
		AWS Lambda
		AWS ElasticCache
		AWS CodeDeploy
		AWS API Gateway
		AWS ELB(Elastic Load Balancer)
		AWS SNS

12 BUSINESS-REQUIREMENT:

As an application developer, develop frontend, middleware and deploy the Tweet App (Single Page App) with below guidelines:

User Story #	User Story Name	User Story
US_01	Registration and Login	As a user I should be able to login/Register in the tweet application Acceptance criteria: 6. A logged-in user can reset their password so they can login, even if they forget their password. 7. A logged-in user: a. Cannot change their username. b. Can logout from their account. 8. As a user I should be able to furnish following details at the time of registration a. First Name b. Last Name c. Email d. Login Id e. Password f. Confirm Password g. Contact Number 9. All details fields must be mandatory 10. Login Id and Email must be unique 11. Password and Confirm Password must be same 12. If any constraint is not satisfied, validation message must be shown
US_02	Post Tweet	As a user I should be able to post a tweet Acceptance criteria: a. Tweet should not go beyond 144 characters. b. Tweet can optionally be associated with a tag which should not go beyond 50 characters
US_03	View and Reply Tweet	As a user I should be able to view others tweet and reply to it. Acceptance criteria: a. View others tweet and reply b. Others tweet should display original tweet with all the reply c. Tweet and reply must have user name and time of post displayed along. d. Reply should not go beyond 144 characters e. I should be optionally able to add a tag while replying

13 RUBRICS/EXPECTED DELIVERABLES

13.1 REST API (PRODUCTS & FRAMEWORKS -> COMPUTE & INTEGRATION):

a. Use Spring Boot to version and implement the REST endpoints.

b. Implement HTTP methods like GET, POST, PUT, DELETE, PATCH to implement RESTful resources:

POST	**/api/v1.0/tweets/register**	**Register as new user**
GET	**/api/v1.0/tweets/login**	**Login**
GET	**/api/v1.0/tweets/<username>/forgot**	**Forgot password**
GET	**/api/v1.0/tweets/all**	**Get all tweets**
GET	**/api/v1.0/tweets/users/all**	**Get all users**
GET	**/api/v/1.0/tweets/user/search/username***	**Search by username**
GET	**/api/v1.0/tweets/username**	**Get all tweets of user**
POST	**/api/v1.0/tweets/<username>/add**	**Post new tweet**
PUT	**/api/v1.0/tweets/<username>/update/<id>**	**Update tweet**
DELETE	**/api/v1.0/tweets/<username>/delete/<id>**	**Delete tweet**
PUT	**/api/v1.0/tweets/<username>/like/<id>**	**Like tweet**
POST	**/api/v1.0/tweets/<username>/reply/<id>**	**Reply to tweet**

c. ***username may be partial or complete username**

d. Use necessary configuration in place for REST API in application.properties or bootstrap.properties or application.yml; whichever is applicable.

e. Package Structure for Spring Boot Project will be like com.tweetapp.* with proper naming conventions for package and beans.

f. Use configuration class annotated with @Configuration and @Service for business layer.

g. Use constructor-based dependency injection in few classes and setter-based dependency injection in few classes.

h. Follow Spring Bean Naming Conventions

13.2 DATABASE (PRODUCTS & FRAMEWORKS -> DATABASE & STORAGE):

1. As an application developer:
    a. Implement ORM with Spring Data MongoRepositoryand MongoDB. For complex and custom queries, create custom methods and use @Query, Aggregations (AggregationOperation, MatchOperation, AggregationResults), implementation of MongoTemplateetc as necessary.
    b. Have necessary configuration in place for REST API in application.properties or bootstrap.properties or application.yml OR Java based configuration; whichever is applicable.

13.3 Maven (Tooling):

1. As an application developer:
    a. Create the spring boot project using Maven CLI
    b. Generate Surefire test reports and share it as a part of deliverables
    c. Using Maven CLI generate the project documentation, and share it as a part of deliverables

13.4 Messaging (Products & Frameworks -> Compute & Integration):

1. As an application developer:

    a. Have a centralized logging system
    b. Be able to communicate using a messaging infrastructure.
    c. Use Kafka/RabbitMQ/ActiveMQ/KubeMQTemplate for communication with Springboot and topics in Kafka/RabbitMQ/ActiveMQ/KubeMQ.
    d. Use Kafka/RabbitMQ/ActiveMQ/KubeMQ for messaging infrastructure and implement producers to write messages/tweets to topic and consumers to read messages/tweets from topic.
    e. Configure Springboot app to log all logging messages to Kafka/RabbitMQ/ActiveMQ/KubeMQ.
    f. Configure all Kafka/RabbitMQ/ActiveMQ/KubeMQ related configuration needed for Spring Boot in *.properties or *.yml file.

13.5 Log/ Monitoring (Products & Frameworks -> Governance & Tooling):

1. As an application developer:

    a. Containerize the complete application, which includes front-end, middleware and Kafka/RabbitMQ/ActiveMQ/KubeMQ (consumers and producers) using docker and Dockerfile.
    b. Use .dockerignore as necessary to avoid containerizing un-necessary packages.
    c. Integrate Spring Boot Actuator with Prometheus and Grafana to monitor middleware.
    d. Implement logs with logstash.
    e. Open the preconfigured Logstash in Kibana and check if it successfully connect to Elasticsearch Server.

13.6 Debugging & Troubleshooting

1. Generate bug report & error logs - Report must be linkedwith final deliverableswhich should also suggest the resolutionfor the encountered bugs and errors.

13.7 Code Quality

1. Associates should have written efficient code within acceptable limits of time and space complexity
2. Associates should have written clean code that is readable
3. Associates should have written testable code
4. Associate should have used the required Code Analyzer to ensure code quality and standard code style

13.8 Secured Coding

1. Associates should have applied basic secure coding principles
2. Associates should have avoided deprecated functions
3. Associates should have used Checkmarx/Veracode as the secured coding tool

14 Platform

14.1 Compute

1. Use ECS CLI (as an alternative to AWS Management Console) for container management and deployment of spring boot application. You should be able to explain and demonstrate the same in interview.
2. Use NoSQL instance of AWS DynamoDB/Aurora(SQL) as a database for the Tweet Application

14.2 Compute, Identity & Compliance, Security& Content Delivery

1. Use AWS Lambda and AWS Aurora to build a backend process for handling requests for Tweet App.
2. Use Serverless Java Container using AWS ECS and run the tweet app created with Spring Boot inside AWS Lambda.
3. Use Amazon API Gateway to expose the lambda functions built in the previous step to be accessible on public internet.
4. Use AWS ELB to configure the auto-scaling container instances.
5. Configure AWS SNS to issue messages whenever a ELB scales-up and scale-down container instances

Note – Minimum two rest endpoints should be hosted in cloud

15 Methodology

15.1 Agile

1. As an application developer, use project management tool along to update progress as you start implementing solution.
2. As an application developer, the scope of discussion with mentor is limited to:
    a. Q/A
    b. New Ideas, New feature implementations and estimation.
    c. Any development related challenges
    d. Skill Gaps
    e. Any other pointers key to UI/UX and Middleware Development