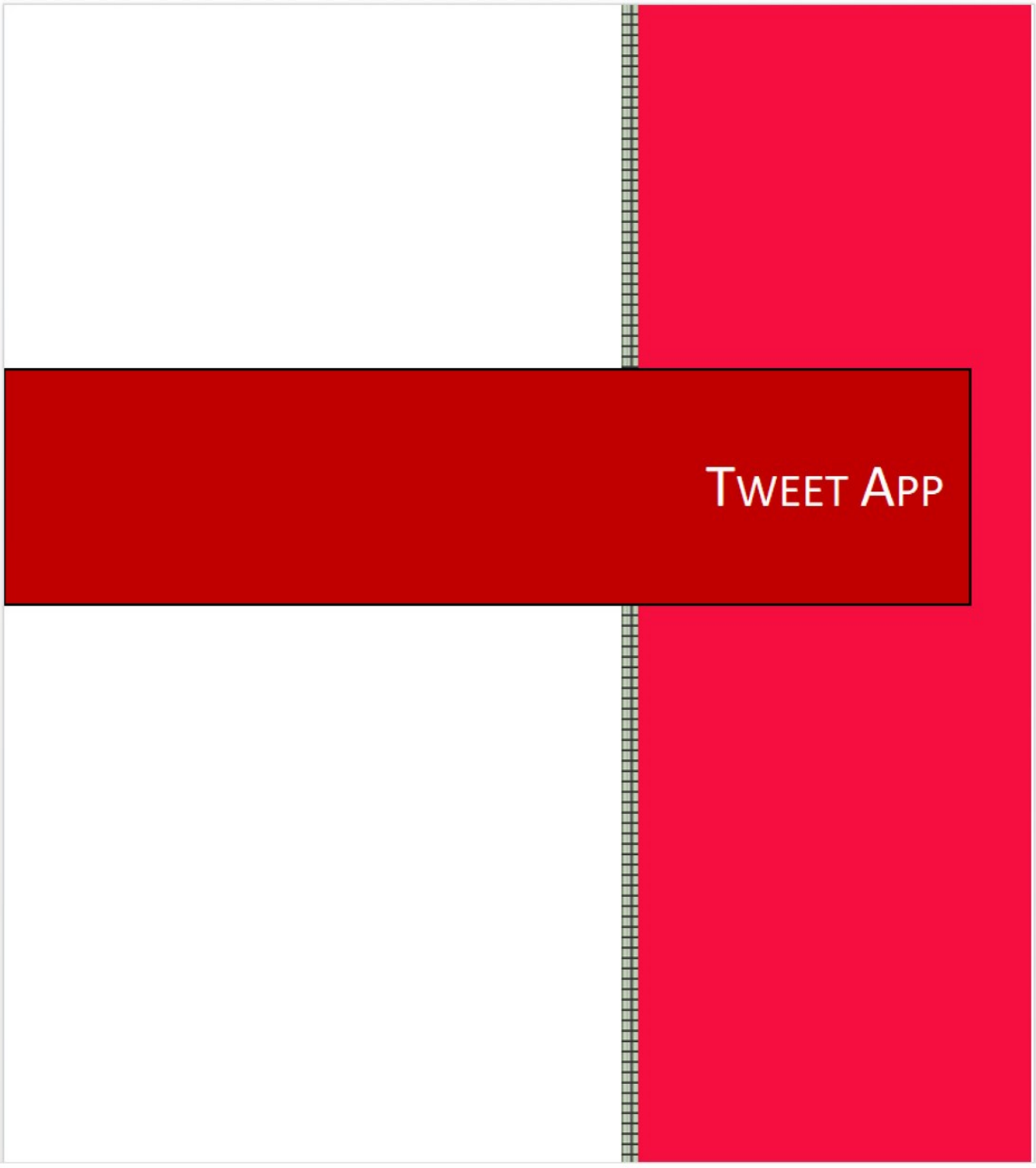

TWEET APP

CONTENTS

1 Problem Statement

Tweet App is SPA (Single Page App) 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:

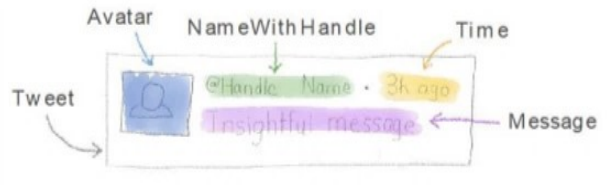
1. User registration and login
2. Post tweet
3. View users/handles and their respective tweets

The scope includes developing the application using toolchain mentioned below.

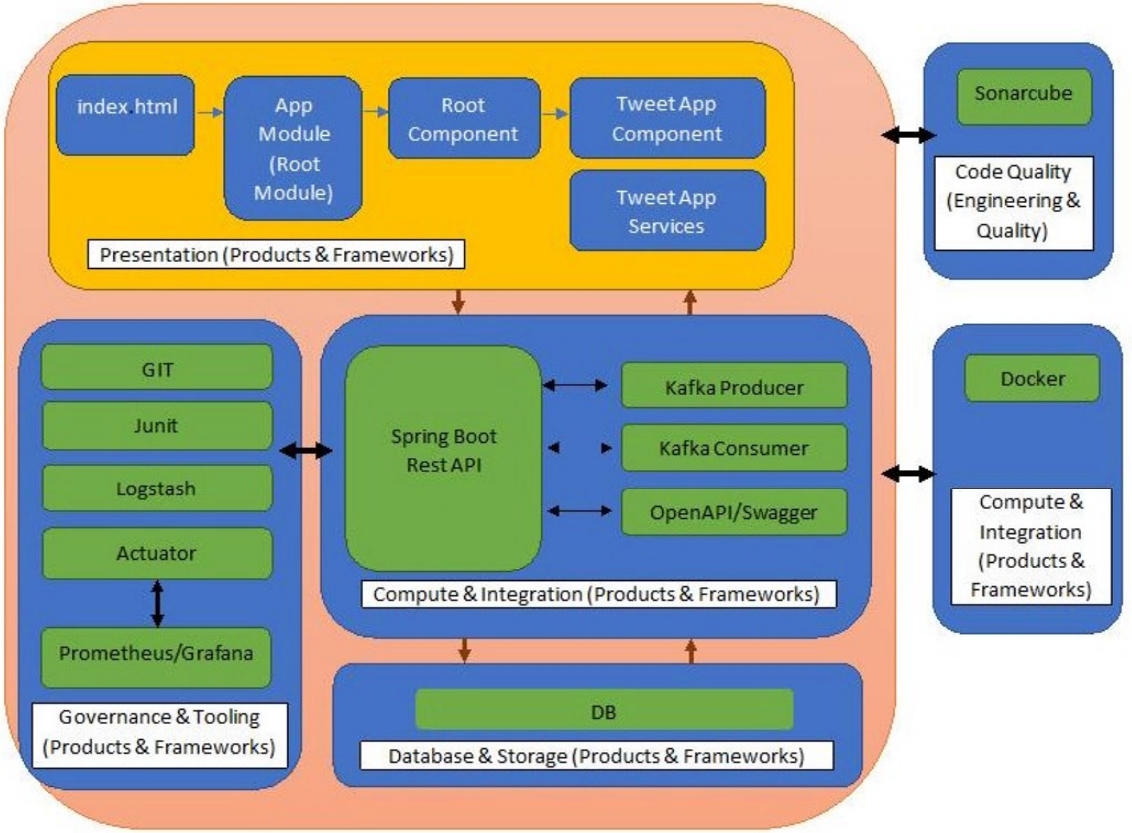
2 Proposed Tweet App Wireframe

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

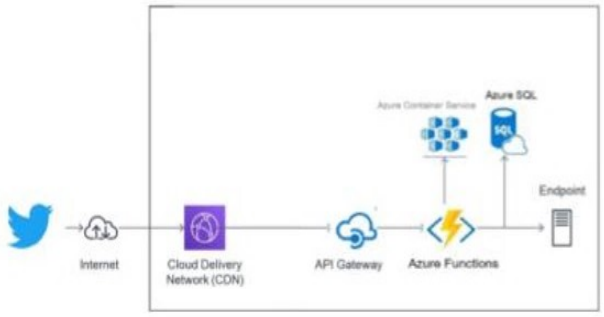
3 Tweet Component Sketch



4 Application Architecture



5 Cloud Architecture



6 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
		Docker
	Database & Storage	MongoDB
	Governance & Tooling	Git
		Junit
		Mockito
		Logstash
		Prometheus & Grafana
Engineering Quality	Code Quality	Sonar Cube
Platform	Cloud Tools	Azure ACS (Container Services)
		Azure CosmosDB/SQL DB
		Azure Automation
		Azure Redis Cache/Storage
		Azure DevOps/Pipeline
		Azure API Gateway
		Azure Load Balancer
		Azure Notification Hubs
		Azure Functions

7 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: 1. A logged-in user can reset their password so they can login, even if they forget their password. 2. A logged-in user: a. Cannot change their username. b. Can logout from their account. 3. 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 4. All details fields must be mandatory 5. Login Id and Email must be unique 6. Password and Confirm Password must be same 7. 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

8 Rubrics/Expected Deliverables

8.1 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

8.2 Database (Products & Frameworks -> Database & Storage):

1. As an application developer:
   a. Implement ORM with Spring Data MongoRepository and MongoDB. For complex and custom queries, create custom methods and use @Query, Aggregations (AggregationOperation, MatchOperation, AggregationResults), implementation of MongoTemplate etc 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.

8.3 API Documentation (Products & Frameworks -> Compute & Integration):

1. As an application developer:
    a. Document REST endpoints with OpenAPI or Swagger

8.4 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 KafkaTemplate for communication with Springboot and topics in kafka.
    d. Use kafka 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.
    f. Configure all kafka related configuration needed for Spring Boot in *.properties or *.yml file.

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

1. As an application developer:

    a. Containerize the complete application, which includes front-end, middleware and kafka (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.

8.6 Debugging & Troubleshooting

1. Generate bug report & error logs - Report must be linked with final deliverables which should also suggest the resolution for the encountered bugs and errors.

9 Platform

9.1 Compute

1. Use Azure CLI 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 Azure CosmosDB as a database for the Tweet Application

9.2 Compute, Identity & Compliance, Security& Content Delivery

1. Use Azure Automation and Azure Cosmos to build a backend process for handling requests for Tweet App.
2. Use Serverless Java Container using Azure ACS and run the tweet app created with Spring Boot inside Azure Automation.
3. Use Azure API Gateway to expose the Azure functions built in the previous step to be accessible on public internet.
4. Use Azure LB to configure the auto-scaling container instances.
5. Configure Azure Notification Hubs to issue messages whenever a Azure LB scales-up and scale-down container instances

Note – Minimum two rest endpoints should be hosted in cloud

10 Methodology

10.1 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