FITNESS TRACKER

Time To Complete: 10 to 12 hr

CONTENTS

1 PROBLEM STATEMENT

Fitness Tracker is SPA (Single Page Application) for placing a request for appointments, view appointments, contact us.

The core modules of fitness tracker app are:

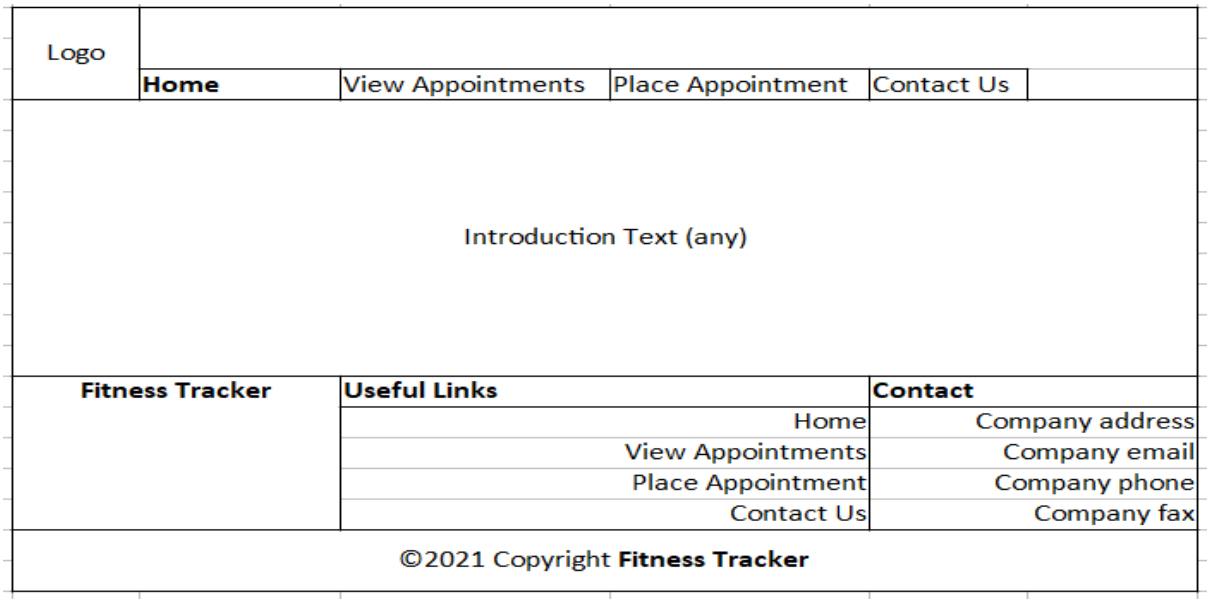
1. Landing Page
2. View Appointments
3. Place Appointment
4. Contact Us

The scope includes developing the application using toolchain mentioned below.

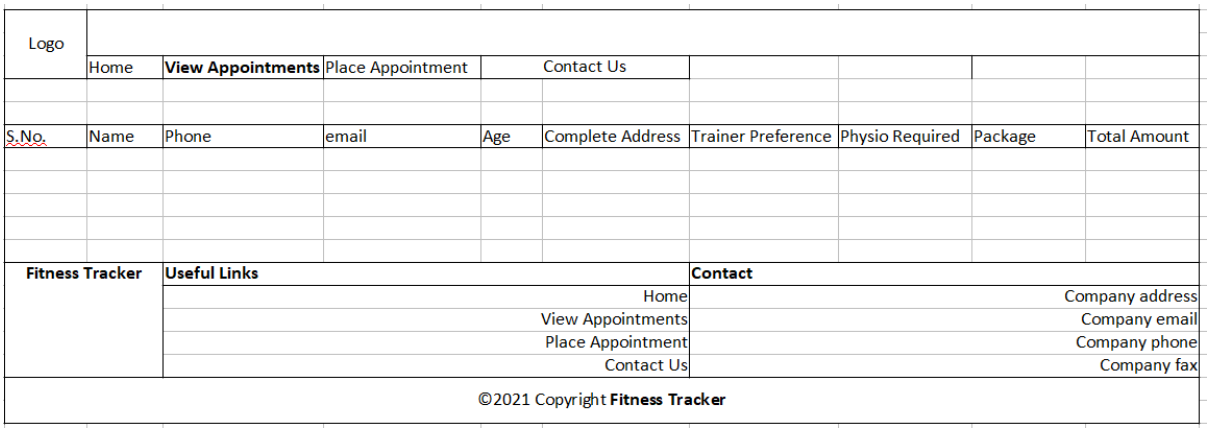
2 PROPOSED FITNESS TRACKER WIREFRAME

UI needs improvisation and modification as per given use case.

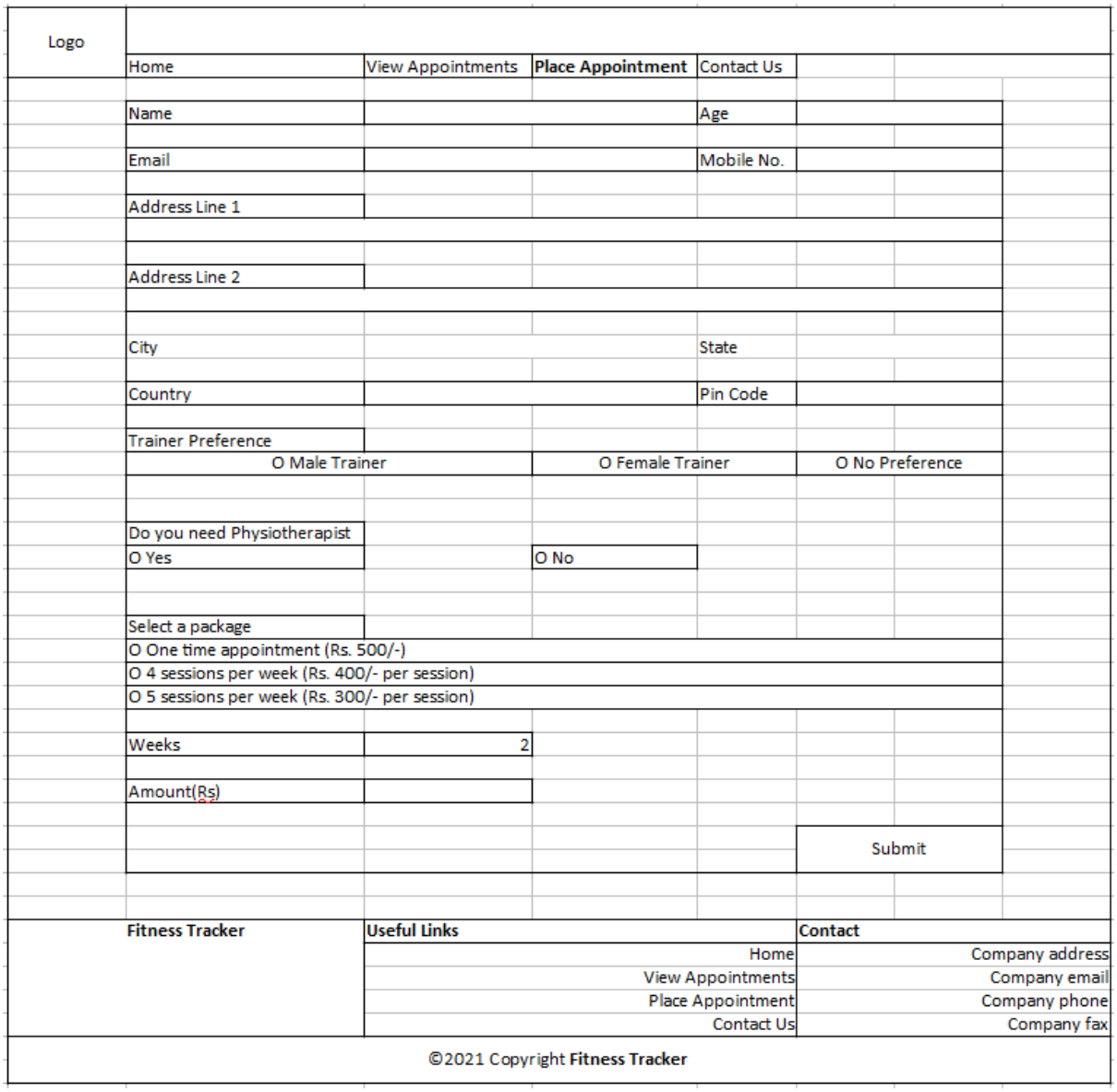
2.1 HOME/LANDING PAGE



2.2 VIEW APPOINTMENTS



2.3 PLACE APPOINTMENT

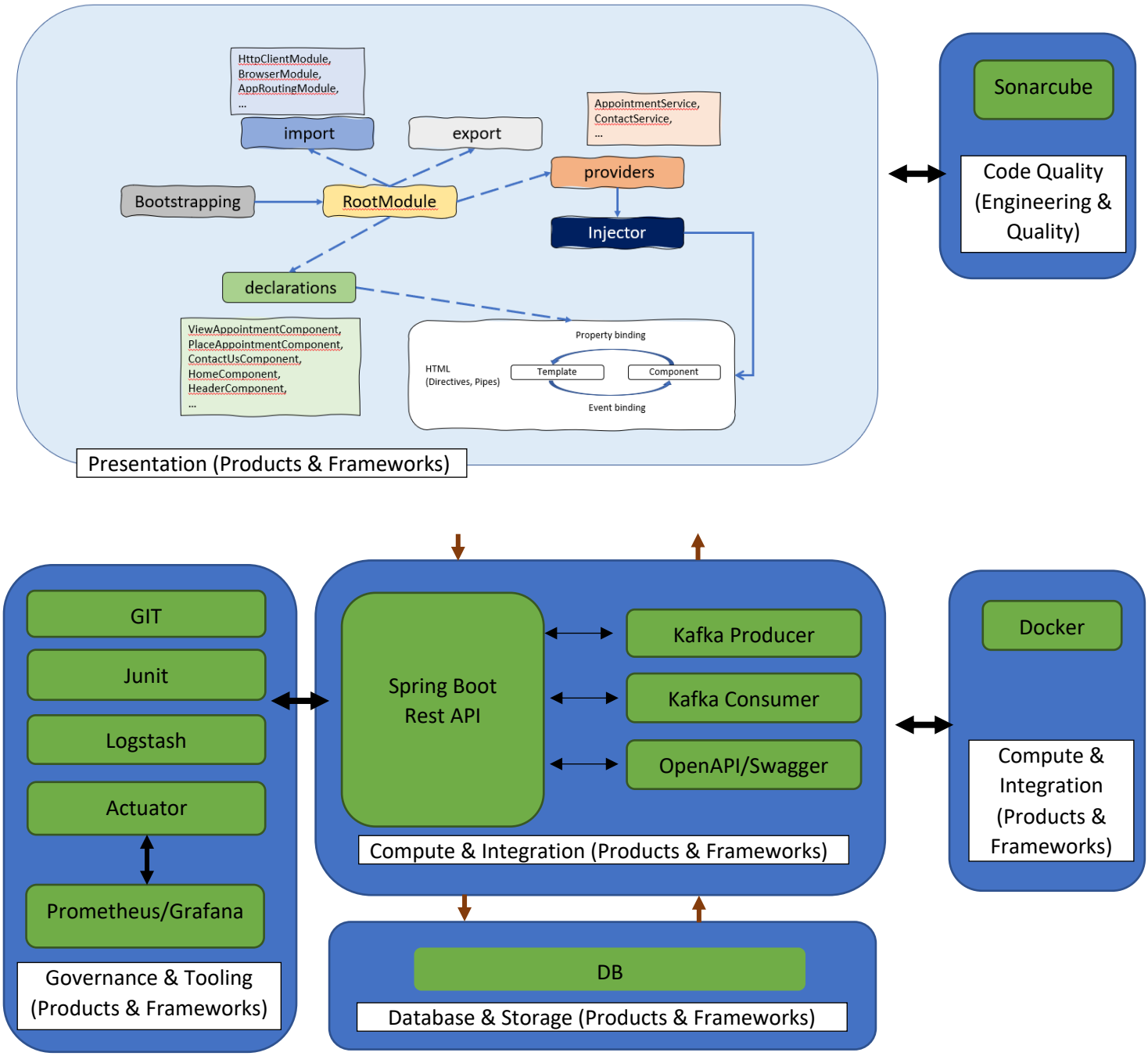


Logo					
	Home	View Appointments	**Place Appointment**	Contact Us	
	Name			Age	
	Email			Mobile No.	
	Address Line 1				
	Address Line 2				
	City			State	
	Country			Pin Code	
	Trainer Preference				
	O Male Trainer		O Female Trainer		O No Preference
	Do you need Physiotherapist				
	O Yes		O No		
	Select a package				
	O One time appointment (Rs. 500/-)				
	O 4 sessions per week (Rs. 400/- per session)				
	O 5 sessions per week (Rs. 300/- per session)				
	Weeks	2			
	Amount(Rs)				
					Submit
	Fitness Tracker	**Useful Links**		**Contact**	
		Home		Company address	
		View Appointments		Company email	
		Place Appointment		Company phone	
		Contact Us		Company fax	
	©2021 Copyright **Fitness Tracker**				

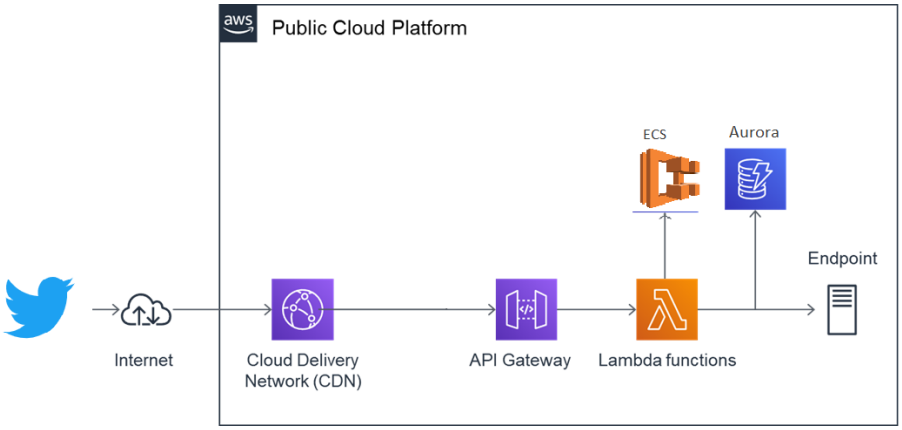
2.4 CONTACT US

Logo				
	Home	View Appointments	Place Appointment	**Contact Us**
		Drop us a message		
	Your Name *	Your Message *		
	Your Email *			
	Your Phone *			
			Send	
Fitness Tracker	**Useful Links**		**Contact**	
	Home		Company address	
	View Appointments		Company email	
	Place Appointment		Company phone	
	Contact Us		Company fax	
©2021 Copyright **Fitness Tracker**				

3 APPLICATION ARCHITECTURE



4 CLOUD ARCHITECTURE



5 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	AWS ECS
		AWS DynamoDB/Aurora
		AWS Lambda
		AWS ElasticCache
		AWS CodeDeploy
		AWS API Gateway
		AWS ELB (Elastic Load Balancer)
		AWS SNS

6 BUSINESS-REQUIREMENT:

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

User Story #	User Story Name	User Story
US_01	Landing Page/Home Page	As a user I should be able to visit the home page as default page. Acceptance criteria: 1. User can click any button given in menu bar.
US_02	Post Appointment	As a user I should be able to post an appointment Acceptance criteria: 1. As a user I should be able to furnish following details at the time of placing an appointment 1.1 Name 1.2 Age 1.3 Email 1.4 Mobile No 1.5 Address Line 1 1.6 Address Line 2 1.7 City 1.8 State 1.9 Country 1.10 Pin Code 1.11 Trainer Preference 1.12 Physiotherapist requirement (Yes or No) 1.13 Select a package 1.14 Weeks 1.15 Amount (Disabled) 2. Weeks number type input box should be visible when 2nd or 3rd package option is selected. 3. Amount should be disabled and should be calculated automatically based on selected package. 2. All details fields must be mandatory. 3. Address line 2 may contain same address as address line 1. 4. Email & Mobile must be unique. 5. If any constraint is not satisfied, validation message must be shown. 6. A success or failure message should be visible after submit button clicked.
US_03	View Appointment	As a user I should be able to view all appointment requests. Acceptance criteria: 1. View all appointment requests. 2. Message should be visible if no appointment is available to show.
US_04	Post Contact Us	As a user I should be able to post a feedback/query/message Acceptance criteria:

		1. As a user I should be able to furnish following details at the time of filling contact us form a. Name b. Email c. Phone d. Message 2. Message should not go beyond 200 characters. 3. All four fields must be mandatory. 4. A success or failure message should be visible after submit button clicked.

7 RUBRICS/EXPECTED DELIVERABLES

7.1 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:

GET	/api/v1.0/fitnesstracker/contacts	Get all contacts
GET	/api/v1.0/fitnesstracker/appointments	Get all appointments
GET	/api/v1.0/fitnesstracker/appointments/<email>	Get all appointments of a user
POST	/api/v1.0/fitnesstracker/appointments	Post new appointment request
PUT	/api/v1.0/fitnesstracker/appointments/<id>	Update appointment request
DELETE	/api/v1.0/fitnesstracker/appointments/<id>	Delete appointment request

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

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

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

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

g. Follow Spring Bean Naming Conventions

7.2 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.

7.3 API DOCUMENTATION (PRODUCTS & FRAMEWORKS -> COMPUTE & INTEGRATION):

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

7.4 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.

7.5 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.

7.6 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.

8 PLATFORM

8.1 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

8.2 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

9 METHODOLOGY

9.1 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