Competency Based Learning Material (CBLM)

Android Mobile Application Development

Level-4

Module: Performing Application Programming Interface Integration

Code: CBLM-OU-ICT-AMAD-05-L4-V1

National Skills Development Authority
Prime Minister's Office
Government of the People's Republic of Bangladesh

Copyright

Approved by __ th Authority Meeting of NSDA Held on --------------

How to use this Competency Based Learning Material (CBLM)

The module, Performing Application Programming Interface Integration contains training materials and activities for you to complete. These activities may be completed as part of structured classroom activities or you may be required you to work at your own pace. These activities will ask you to complete associated learning and practice activities in order to gain knowledge and skills you need to achieve the learning outcomes.

1. Review the **Learning Activity** page to understand the sequence of learning activities you will undergo. This page will serve as your road map towards the achievement of competence.

2. Read the **Information Sheets.** This will give you an understanding of the jobs or tasks you are going to learn how to do. Once you have finished reading the **Information Sheets** complete the questions in the **Self-Check.**

3. **Self-**Checks are found after each **Information Sheet**. **Self-Checks** are designed to help you know how you are progressing. If you are unable to answer the questions in the **Self-Check** you will need to re-read the relevant **Information Sheet**. Once you have completed all the questions check your answers by reading the relevant **Answer Keys** found at the end of this module.

4. Next move on to the **Job Sheets. Job Sheets** provide detailed information about *how to do the job* you are being trained in. Some **Job Sheets** will also have a series of **Activity Sheets**. These sheets have been designed to introduce you to the job step by step. This is where you will apply the new knowledge you gained by reading the Information Sheets. This is your opportunity to practice the job. You may need to practice the job or activity several times before you become competent.

5. Specification **sheets**, specifying the details of the job to be performed will be provided where appropriate.

6. A review of competency is provided on the last page to help remind if all the required assessment criteria have been met. This record is for your own information and guidance and is not an official record of competency

When working though this Module always be aware of your safety and the safety of others in the training room. Should you require assistance or clarification please consult your trainer or facilitator.

When you have satisfactorily completed all the Jobs and/or Activities outlined in this module, an assessment event will be scheduled to assess if you have achieved competency in the specified learning outcomes. You will then be ready to move onto the next Unit of Competency or Module

Table of Contents

Module Content

Unit of Competency	**Perform Application Programming Interface integration**
Unit Code	**OU-ICT-AMAD-03-L4-V1**
Module Title	**Performing Application Programming Interface integration**
Module Descriptor	This module covers the knowledge, skills and attitudes required to perform Application Programming Interface integration It includes the task of using API, implementing custom API, firebase API and Firebase Cloud Messaging (FCM)
Nominal Hours	**40** Hours
Lerning Outcome	After completing the practice of the module, the trainees will be able to perform the following jobs: 1. Use API 2. Implement custom API 3. Implement firebase API 4. Implement Firebase Cloud Messaging (FCM)

Assessment Criteria

1. Application Programming Interface (API) is defined.
2. API is integrated A simple activity layout is designed for some basic user operation.
3. Custom API is defined
4. Custom API is integrated
5. Firebase API is defined
6. Firebase API is integrated
7. Firebase authentication is performed
8. Firebase Realtime database is displayed
9. FCM in Android is described.
10. Client and server in FCM is defined.
11. FCM client is implemented.
12. FCM server is implemented.
13. Working with user notification is demonstrated.

Learning Outcome 1: Use API

Assessment Criteria	1. Application Programming Interface (API) is defined. 2. API is integrated
Conditions and Resources	• Actual workplace or training environment • CBLM • Handouts • Job related tools, equipment, and materials • Multimedia Projector • Paper, Pen, Pencil and Eraser • Internet Facilities • Whiteboard and Marker
Contents	1. Application Programming Interface (API) 2. Integrate API
Activities/job/Task	1. Create a simple web application that integrates with a public API. For example, use a weather API to fetch current weather data and display it on your web page. Utilize JavaScript (or a preferred language) to make API requests and handle the response. 2. Build a backend service that serves as an API and integrates with a database. Develop a simple frontend (web or mobile) that communicates with the backend 62 API. Implement CRUD operations, allowing the frontend to interact seamlessly with the backend. 3. Implement an API Gateway that serves as a central entry point for multiple microservices or backend APIs. Configure the API Gateway to route requests to the appropriate services. Create and demonstrate the integration of the API Gateway in a distributed system.
Training Methods	• Blended • Discussion • Presentation • Demonstration • Guided Practice • Individual Practice • Project Work • Problem Solving • Brainstorming
Assessment Methods	Assessment methods may include but not limited to • Written Test • Demonstration • Oral Questioning

Learning Experience 1: Use API

In order to achieve the objectives stated in this learning guide, you must perform the learning steps below. Beside each step are the resources or special instructions you will use to accomplish the corresponding activity.

Learning Activities	Recourses/Special Instructions
1. Trainee will ask the instructor about the learning materials	1. Instructor will provide the learning materials Use API
2. Read the Information sheet and complete the Self Checks & Check answer sheets on "Use API"	2. Read Information sheet 1: Use API 3. Answer Self-check 1: Use API 4. Check your answer with Answer key 1: Use API
3. Read the Job/Task Sheet and Specification Sheet and perform job/Task	5. Job/Task Sheet and Specification Sheet **Job Sheet 1.1:** Create a simple web application that integrates with a public API. For example, use a weather API to fetch current weather data and display it on your web page. Utilize JavaScript (or a preferred language) to make API requests and handle the response. **Specification Sheet 1.1:** Create a simple web application that integrates with a public API. For example, use a weather API to fetch current weather data and display it on your web page. Utilize JavaScript (or a preferred language) to make API requests and handle the response. **Job Sheet 1.2:** Build a backend service that serves as an API and integrates with a database. Develop a simple frontend (web or mobile) that communicates with the backend 62 API. Implement CRUD operations, allowing the frontend to interact seamlessly with the backend. **Specification Sheet 1.2:** Build a backend service that serves as an API and integrates with a database. Develop a simple frontend (web or mobile) that communicates with the backend 62 API. Implement CRUD

	operations, allowing the frontend to interact seamlessly with the backend. **Job Sheet 1.3:** Implement an API Gateway that serves as a central entry point for multiple microservices or backend APIs. Configure the API Gateway to route requests to the appropriate services. Create and demonstrate the integration of the API Gateway in a distributed system. **Specification Sheet 1.1:** Implement an API Gateway that serves as a central entry point for multiple microservices or backend APIs. Configure the API Gateway to route requests to the appropriate services. Create and demonstrate the integration of the API Gateway in a distributed system.

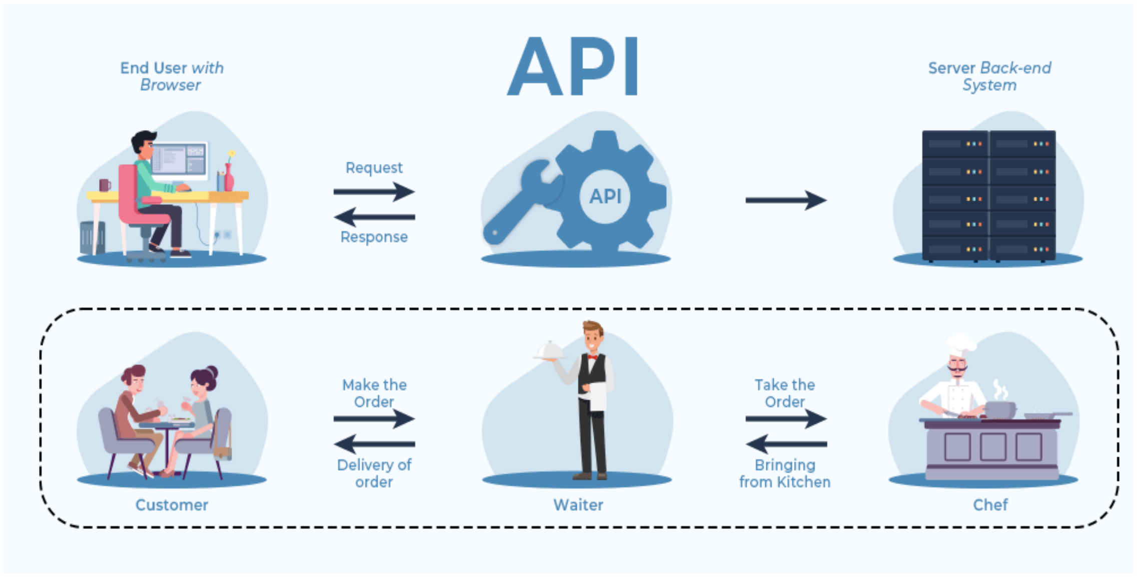
Information Sheet 1: Use API

Learning Objective: After completion of this information sheet, the learners will be able to explain, define and interpret the following contents:

1.1 Application Programming Interface (API)
1.2 Integrate API

1.1 Application Programming Interface (API)

Everyone is in search of the highest-paying job so as to get into it. And, in the list, the web developer has been on the top for years and will remain in the same place due to its demand. If you're the one who's looking to get into it. you must be aware of the most important terms used in it. Out of all the terms, API is yet another term that plays a very important role in building a website. Now, what is an API – (Application Programming Interface)?



To make you clear with the diagram of what is API, let's take a real-life example of an API, you can think of an API as a waiter in a restaurant who listens to your order request, goes to the chef, takes the food items ordered and gets back to you with the order. Also, if you want to look for the working of an API with the example, here's one. You're searching for a course(let's say DSA-Self Paced) on the XYZ website, you send a request(product search requested) through an API, and the database searches for the course and checks if it's available, the API is responsible here to send your request to the database (in search of the course) and responds with the output(best DSA courses)

API full form is an Application Programming Interface that is a collection of communication protocols and subroutines used by various programs to communicate between them. A programmer can make use of various API tools to make their program easier and simpler. Also, an API facilitates programmers with an efficient way to develop their software programs. Thus api meaning is when an API helps two programs or applications to communicate with each other by providing them with the necessary tools and functions. It takes the request from the user and sends it to the service provider and then again sends the result generated from the service provider to the desired user.

A developer extensively uses APIs in his software to implement various features by using an API call without writing complex codes for the same. We can create an API for an operating system, database system, hardware system, JavaScript file, or similar object-oriented files. Also, an API is similar to a GUI(Graphical User Interface) with one major difference. Unlike GUIs, an application program interface helps software developers to access web tools while a GUI helps to make a program easier to understand for users.

API Different from a Web Application?

An API acts as an interface that allows proper communication between two programs whereas a web application is a network-based resource responsible for completing a single task. Also, it's important to know that "All web services are APIs, but not all APIs are web".

The difference between an API and a web application is that API allows two-way communication and web applications are just a way for users to interact through a web browser. A web application may have an API to complete the requests.

Types of APIs

There are three basic forms of API –

a. Web APIs

A **Web API** also called Web Services is an extensively used API over the web and can be easily accessed using the HTTP protocols. A Web application programming interface is an open-source interface and can be used by many clients through their phones, tablets, or PCs.

Example:

Step 1: Get Your API Key

1. Sign up for a free account at OpenWeatherMap](https://home.openweathermap.org/users/sign_up).
2. Go to your API keys section to get your unique API key.

Step 2: Create Your Project Files

- index.html: This will be the HTML file to structure the webpage.
- styles.css: This file will be used for styling the page.
- script.js: This file will contain the JavaScript code to interact with the API.

Step 3: Write the Code

html

```
<!DOCTYPE html>
<html lang="en">
<head>
   <meta charset="UTF-8">
   <meta name="viewport" content="width=device-width, initial-scale=1.0">
   <title>Weather App</title>
   <link rel="stylesheet" href="styles.css">
</head>
<body>
   <div class="container">
      <h1>Weather App</h1>
      <input type="text" id="city" placeholder="Enter city name">
      <button id="getWeather">Get Weather</button>
      <div id="weatherResult"></div>
   </div>
   <script src="script.js"></script>
</body>
</html>
```

styles.css

```
body {
   font-family: Arial, sans-serif;
   display: flex;
   justify-content: center;
   align-items: center;
   height: 100vh;
   margin: 0;
   background-color: #f0f0f0;
}

.container {
   text-align: center;
   background: white;
   padding: 20px;
   border-radius: 8px;
   box-shadow: 0 0 10px rgba(0, 0, 0, 0.1);
}

input {
   padding: 10px;
   width: 200px;
   margin-right: 10px;
}
button {
    padding: 10px;
    cursor: pointer;
}
#weatherResult {
    margin-top: 20px;
}
```

script.js

```
javascript
document.getElementById('getWeather').addEventListener('click', function() {
    const city = document.getElementById('city').value;
    const apiKey = 'YOUR_API_KEY_HERE'; // Replace with your OpenWeatherMap
API key
    const                                   url                                   =
`https://api.openweathermap.org/data/2.5/weather?q=${city}&appid=${apiKey}&
units=metric`;

    fetch(url)
        .then(response => response.json())
        .then(data => {
            if (data.cod === '404') {
                document.getElementById('weatherResult').innerHTML    =    'City    not
found!';
            } else {
                const temperature = data.main.temp;
                const description = data.weather[0].description;
                document.getElementById('weatherResult').innerHTML = `
                    <h2>Weather in ${city}</h2>
                    <p>Temperature: ${temperature} °C</p>
                    <p>Description: ${description}</p>
                `;
            }
        })
        .catch(error => {
            console.error('Error fetching weather data:', error);
            document.getElementById('weatherResult').innerHTML  =  'Error  fetching
weather data.';
        });
});
```

Step 4: Run Your Application

- Save all the files (`index.html`, `styles.css`, `script.js`) in the same directory.
- Open `index.html` in a web browser.

You should see a simple interface where you can input a city name and get the current weather information when you click the "Get Weather" button.

Notes

Make sure to replace `'YOUR_API_KEY_HERE'` in `script.js` with your actual API key from OpenWeatherMap.
This example uses the metric system for temperature. You can change the `units` parameter in the URL to `imperial` if you prefer Fahrenheit.

Feel free to extend this application with more features, such as forecasts or additional weather details!

b. Local APIs
In this type of API, the programmers get the local middleware services. TAPI (Telephony Application Programming Interface), and .NET are common examples of Local APIs.
c. Program APIs
It makes a remote program appear to be local by making use of RPCs (Remote Procedural Calls). SOAP is a well-known example of this type of API.

Few other types of APIs:
- **SOAP (SIMPLE OBJECT ACCESS PROTOCOL):** It defines messages in XML format used by web applications to communicate with each other.
- **REST (Representational State Transfer):** It makes use of HTTP to GET, POST, PUT, or DELETE data. It is basically used to take advantage of the existing data.
- **JSON-RPC:** It uses JSON for data transfer and is a lightweight remote procedural call defining a few data structure types.
- **XML-RPC:** It is based on XML and uses HTTP for data transfer. This API is widely used to exchange information between two or more networks.

REST APIs?
REST stands for Representational State Transfer, and follows the constraints of REST architecture allowing interaction with RESTful web services. It defines a set of functions (GET, PUT, POST, DELETE) that clients use to access server data. The functions used are:
- GET (retrieve a record)
- PUT (update a record)
- POST (create a record)
- DELETE (delete the record)

Its main feature is that REST API is stateless, i.e., the servers do not save clients' data between requests.

Web API
Web API Is simply an API for the web. It is an API that can be accessed using the HTTP protocol. It can be built using Java, .nET, etc. It is implemented to extend the functionality of a browser, simplify complex functions, and provide easy syntax to complex code.
The four main types of web APIs are:

- Open API
- Partner API
- Internal API
- Composite API

API (Application Programming Interface) Integration?
API (Application Programming Interface) Integration is the connection between two or more applications, via APIs, letting you exchange data. It is a medium through which you can share data and communicate with each other by involving APIs to allow web tools to communicate. Due to the rise in cloud-based products, API integration has become very important.

What is API (Application Programming Interface) Testing?
API (Application Programming Interface) testing is a kind of software testing that analyzes an API in terms of its functionality, security, performance, and reliability. It is very important to test an API so as to check whether it's working as expected or not. If not, again changes are made in the architecture and re-verified.
APIs are the center of software development to exchange data across applications. The API testing includes sending requests to single/multiple API endpoints and validating the response. It focuses majorly on business logic, data responses and security, and performance bottlenecks.

Types of Testing

- Unit Testing
- Integration Testing
- Security Testing
- Performance Testing
- Functional Testing

Must Read: API Testing in Software Testing
API Testing Tools

- Postman
- Apigee
- JMeter
- Ping API
- Soap UI
- vREST

How to Create APIs?

Creating an API is an easy task unless you are very well clear on the basic concepts. It's an iterative process (based on feedback) that just includes a few easy steps:

- Plan your goal and the intended users
- Design the API architecture
- Develop (Implement the code) and Test API
- Monitor its working and work on feedback

Must Read: Tips for Building an API

Restrictions of Using APIs

When an API (Application Programming Interface) is made it's not really released as software for download and it has some policies governing its use or restricting its use to everyone, usually, there are three main types of policies governing APIs, are:

- **Private:** These APIs are only made for a single person or entity (like a company that has spent the resources to make it or bought it).
- **Partner:** Just like the name it gives the authority to use APIs to some partners of entities that own APIs for their private use.
- **Public:** You should be aware of them cause you can only find these APIs in the market for your own use if you don't own specific API access from some entity that owns private these APIs for their private use. An example of a Public API is 'Windows API' by Microsoft for more public APIs you can visit this GitHub repository -> https://github.com/public-apis/public-apis.

Advantages of APIs

- **Efficiency:** API produces efficient, quicker, and more reliable results than the outputs produced by human beings in an organization.
- **Flexible delivery of services:** API provides fast and flexible delivery of services according to developers' requirements.
- **Integration:** The best feature of API is that it allows the movement of data between various sites and thus enhances the integrated user experience.
- **Automation:** As API makes use of robotic computers rather than humans, it produces better and more automated results.
- **New functionality**: While using API the developers find new tools and functionality for API exchanges.

Disadvantages of APIs

- **Cost:** Developing and implementing API is costly at times and requires high maintenance and support from developers.
- **Security issues:** Using API adds another layer of surface which is then prone to attacks, and hence the security risk problem is common in APIs.

1.2 Integrate API

API integration is the process of connecting two or more systems, applications, or services by using Application Programming Interfaces (APIs) to exchange data and functionality. APIs provide a standardized way for different systems to communicate with each other, allowing them to share data, resources, and functionality.

Why is API Integration Important?

API integration is important for several reasons:

a. **Data sharing**: APIs enable the sharing of data between different systems, applications, or services, which can help improve the accuracy, completeness, and timeliness of data.
b. **Improved functionality**: API integration allows developers to build new applications or services that leverage the functionality of existing systems, applications, or services.
c. **Increased efficiency**: API integration can automate many manual processes, reducing the need for manual data entry and improving overall efficiency.
d. **Cost savings**: API integration can reduce costs by eliminating the need for custom development or manual data transfer.

Types of API Integration

There are several types of API integration:

a. **Simple API**: A simple API integration involves connecting two systems using a single API endpoint.
b. **Complex API**: A complex API integration involves connecting multiple systems using multiple API endpoints and may require additional logic and processing.
c. **Real-time API**: A real-time API integration involves exchanging data in real-time, often used in applications that require immediate updates.

API Integration Process

The API integration process typically involves the following steps:

a. **Discovery**: Identify the APIs that need to be integrated and determine their capabilities.
b. **Authentication**: Authenticate with the APIs to obtain access tokens or credentials.
c. **Data mapping**: Map the data structures and formats used by the APIs to ensure seamless data exchange.
d. **API endpoint selection**: Select the relevant API endpoints to use for integration.
e. **Data transformation**: Transform the data to conform to the expected format and schema.
f. **Testing**: Test the integrated system to ensure it functions correctly.
g. **Deployment**: Deploy the integrated system in a production environment.

Best Practices for API Integration

a. **Document everything**: Document the APIs, their endpoints, and their usage.
b. **Test thoroughly**: Thoroughly test the integrated system to ensure it functions correctly.
c. **Monitor performance**: Monitor the performance of the integrated system to identify bottlenecks and optimize accordingly.
d. **Use secure protocols**: Use secure protocols (e.g., SSL/TLS) to ensure data transmission security.
e. **Use standard formats**: Use standard formats (e.g., JSON, XML) for data exchange.

Common Challenges in API Integration

a. **Data inconsistencies**: Data inconsistencies can occur due to differences in data formats and structures.
b. **Security concerns**: Security concerns arise from potential vulnerabilities in APIs or insecure data transmission.
c. **Scalability issues**: Scalability issues can occur when large volumes of data are exchanged between systems.
d. **Integration complexity**: Complex integrations can be challenging due to multiple APIs, endpoints, and logic required.

Example:

Here is a step-by-step guide to building a backend service as an API and integrating it with a frontend:

Backend (Node.js, Express.js, and MongoDB)

Step 1: Set up the project

- Create a new Node.js project using npm init and install the required dependencies: express, body-parser, and mongoose.
- Create a new MongoDB database using mongod or a MongoDB cloud service like MongoDB Atlas.

Step 2: Define the API endpoints

Define the API endpoints for CRUD (Create, Read, Update, Delete) operations:

- GET /api/users - Retrieve all users
- GET /api/users/:id - Retrieve a single user by ID
- POST /api/users - Create a new user
- PUT /api/users/:id - Update a user
- DELETE /api/users/:id - Delete a user

Step 3: Implement API endpoints

Use Mongoose to interact with the MongoDB database:

```
const express = require('express');
const app = express();
const bodyParser = require('body-parser');
const mongoose = require('mongoose');
// Connect to the database
```

```
mongoose.connect('mongodb://localhost/mydatabase', { useNewUrlParser: true,
useUnifiedTopology: true });

// Define the User model
const userSchema = new mongoose.Schema({
  name: String,
  email: String
});
const User = mongoose.model('User', userSchema);

// Implement API endpoints
app.use(bodyParser.json());

app.get('/api/users', async (req, res) => {
  const users = await User.find().exec();
  res.json(users);
});

app.get('/api/users/:id', async (req, res) => {
  const id = req.params.id;
  const user = await User.findById(id).exec();
  if (!user) {
    res.status(404).json({ message: 'User not found' });
  } else {
    res.json(user);
  }
});

app.post('/api/users', async (req, res) => {
  const user = new User(req.body);
  try {
    await user.save();
    res.status(201).json(user);
  } catch (err) {
    res.status(400).json({ message: 'Invalid request' });
  }
});

app.put('/api/users/:id', async (req, res) => {
  const id = req.params.id;
  const user = await User.findByIdAndUpdate(id, req.body, { new: true });
  if (!user) {
    res.status(404).json({ message: 'User not found' });
  } else {
    res.json(user);
  }
});

app.delete('/api/users/:id', async (req, res) => {
  const id = req.params.id;
  await User.findByIdAndRemove(id);
  res.status(204).json({ message: 'User deleted' });
});

app.listen(3000, () => {
  console.log('Server listening on port 3000');
});
```

Step 4: Test the API

Use a tool like Postman or cURL to test the API endpoints:

- GET /api/users: Retrieve all users
- GET /api/users/123: Retrieve a single user by ID
- POST /api/users: Create a new user
- PUT /api/users/123: Update a user
- DELETE /api/users/123: Delete a user

Frontend (React Native or React)

Step 1: Set up the project

- Create a new React Native project using npx react-native init or set up a React project using npx create-react-app.
- Install the required dependencies: axios for making HTTP requests to the backend API.

Step 2: Define the UI components

- Create UI components for displaying and editing users:

```
import React from 'react';
import { View, Text, TextInput, Button } from 'react-native';
const UserForm = () => {
  const [name, setName] = React.useState('');
  const [email, setEmail] = React.useState('');
  const handleSubmit = async () => {
    // Send POST request to create a new user
    try {
      const response = await axios.post('http://localhost:3000/api/users', { name, email
});
      console.log(response.data);
    } catch (error) {
      console.error(error);
    }
  };
  return (
    <View>
      <TextInput placeholder="Name" value={name} onChangeText={(text) =>
setName(text)} />
      <TextInput placeholder="Email" value={email} onChangeText={(text) =>
setEmail(text)} />
      <Button title="Submit" onPress={handleSubmit} />
    </View>
  );
};
```

Step 3: Implement CRUD operations

Implement CRUD operations for interacting with the backend API:

```
import React from 'react';
import { View, Text } from 'react-native';
import axios from 'axios';

const UsersList = () => {
  const [users, setUsers] = React.useState([]);

  async componentDidMount() {
    // Get all users from the backend API
    try {
      const response = await axios.get('http://localhost:3000/api/users');
      setUsers(response.data);
    } catch (error) {
      console.error(error);
    }
  }

  const deleteUser = async (id) => {
    // Delete a user by ID
    try {
      await axios.delete(`http://localhost:3000/api/users/${id}`);
      setUsers(users.filter((user) => user._id !== id));
    } catch (error) {
      console.error(error);
    }
  };

  return (
    <View>
      {users.map((user) => (
        <View key={user._id}>
          <Text>{user.name}</Text>
        <Text>{user.email}</Text>
        <Button title="Delete" onPress={() => deleteUser(user._id)} />
      </View>
    ))}
   </View>
 );
};
```

Step 4: Test the frontend

- Run the React Native project on an emulator or physical device.
- Interact with the UI components to test CRUD operations:
  - Create a new user by filling out the form and submitting it.
  - Retrieve all users and display them in the list.
  - Update an existing user by editing the form and submitting it.
  - Delete a user by pressing the delete button.
  -

This is a basic example of building a backend service as an API and integrating it with a frontend. You can further enhance this example by adding more features, error handling, and security measures.

Self-Check Sheet - 1: Use API

Questionnaire

1. What is an API?

   Answer:
2. What does REST stand for in RESTful APIs?

   Answer:
3. What is a RESTful API?

   Answer:
4. What is an endpoint in an API?

   Answer:
5. What is an API key?

   Answer:
6. What is JSON?

   Answer:
7. What is an HTTP method in the context of APIs?

   Answer
8. What is the purpose of the GET method?

   Answer:
9. What is the difference between PUT and POST methods?

   Answer:
10. What is an API endpoint URL?

    Answer:
11. What is authentication in APIs?

    Answer:
12. What is rate limiting?

    Answer:
13. What is CORS?

    Answer:
14. What is an API response?

    Answer:
15. What is a webhook?

    Answer:

Answer Key - 1: Use API

1. What is an API?

   Answer: An API (Application Programming Interface) is a set of rules and tools that allows different software applications to communicate with each other. It defines how requests and responses should be structured.

2. What does REST stand for in RESTful APIs?

   Answer: REST stands for Representational State Transfer. It is an architectural style for designing networked applications, often used for web services.

3. What is a RESTful API?

   Answer: A RESTful API is an API that adheres to the principles of REST, using standard HTTP methods (GET, POST, PUT, DELETE) and typically exchanging data in JSON or XML format.

4. What is an endpoint in an API?

   Answer: An endpoint is a specific URL where an API can access resources or perform operations. It defines the path through which clients interact with the API.

5. What is an API key?

   Answer: An API key is a unique identifier used to authenticate and authorize requests to an API. It helps ensure that only permitted users can access the API.

6. What is JSON?

   Answer: JSON (JavaScript Object Notation) is a lightweight data interchange format that is easy to read and write for humans and easy to parse and generate for machines. It is commonly used in APIs.

7. What is an HTTP method in the context of APIs?

   Answer: HTTP methods are used to specify the desired action on a resource. Common methods include GET (retrieve), POST (create), PUT (update), and DELETE (remove).

8. What is the purpose of the GET method?

   Answer: The GET method is used to retrieve data from a server. It does not modify the resource and is typically used to fetch information.

9. What is the difference between PUT and POST methods?

   Answer: POST is used to create a new resource on the server, while PUT is used to update an existing resource or create a resource if it does not already exist.

10. What is an API endpoint URL?

    Answer: An API endpoint URL is the specific address (URL) at which an API can be accessed. It defines the resource or action being accessed.

11. What is authentication in APIs?

    Answer: Authentication is the process of verifying the identity of a user or application accessing the API. Common methods include API keys, OAuth, and JWT (JSON Web Tokens).

12. What is rate limiting?

    Answer: Rate limiting is a technique used to control the number of requests a client can make to an API within a specific time period. It helps prevent abuse and ensures fair usage.

13. What is CORS?

    Answer: CORS (Cross-Origin Resource Sharing) is a security feature implemented in web browsers that allows or restricts web applications from making requests to a domain different from the one that served the web page.

14. What is an API response?

    Answer: An API response is the data or status returned by an API after a request has been processed. It typically includes a status code, headers, and the response body.

15. What is a webhook?

    Answer: A webhook is a method used by APIs to provide real-time notifications. It allows an API to send data to a specified URL when certain events occur, without the need for the client to make a request.

Job Sheet-1.1: Create a simple web application that integrates with a public API. For example, use a weather API to fetch current weather data and display it on your web page. Utilize JavaScript (or a preferred language) to make API requests and handle the response.

Working Procedure/ Steps:

1. Follow OSH and use Personal Protective Equipment (PPE).
2. Check Electricity & Internet Connections to your Computer.
3. Start the Computer.
4. Collect the resources from your trainer as per the job requirement.
5. Create a new folder for your project and create the following files:
    a. index.html
    b. script.js
    c. style.css (optional)
6. Add the following code to index.html:
7. Sign up for an API key on the OpenWeatherMap website.
8. Add the API key to your script.js file:
9. Use the fetch API to make a GET request to the OpenWeatherMap API:
10. Parse and display response data
11. Open your web page in a browser.
12. Verify that the weather data is displayed correctly.
13. Show your work to the trainer.
14. Clean your work place as per standard procedure.
15. Turn off the computer and clean your workplace.

Specification Sheet-1.1: Create a simple web application that integrates with a public API. For example, use a weather API to fetch current weather data and display it on your web page. Utilize JavaScript (or a preferred language) to make API requests and handle the response.

Conditions for the job: Work must be carried out in a safe manner and according to relevant competency standards.

List of required PPE

S/N	Name of PPE	Specification	Unit	Required Quantity
01	Ergonomic Chair	Wood and foam	No	1
02	Eye protective glass	Metal and Glass	No	1
03	Mask	Surgical mask	1	1

List of required Software

S/N	Name of Materials	Specification	Unit	Required Quantity
01	IntelliJ IDEA	Latest version	…	1
02	Android Studio	Latest version	…	1
03	Java	JDK Latest version	…	1

List of required Tools & Equipment's

S/N	Name	Specification	Unit	Required Quantity
01	Personal Computer or Laptop	Minimum Core i5 7th gen Processor	No	1
04	Internet connection		…	1

Job Sheet-1.2: Build a backend service that serves as an API and integrates with a database. Develop a simple frontend (web or mobile) that communicates with the backend 62 API. Implement CRUD operations, allowing the frontend to interact seamlessly with the backend.

Working Procedure/ Steps:

1. Follow OSH and use Personal Protective Equipment (PPE).
2. Check Electricity & Internet Connections to your Computer.
3. Start the Computer.
4. Collect the resources from your trainer as per the job requirement.
5. Step 1: Choose a framework and setup
6. Create a new project in your chosen framework
7. Set up the project structure and dependencies
8. Step 2: Create a database
9. Choose a database: MySQL or MongoDB
10. Create a new database and schema
11. Create tables or collections for the data you want to store
12. Step 3: Create API endpoints
    a. Define API endpoints for CRUD (Create, Read, Update, Delete) operations
    b. Use HTTP methods (GET, POST, PUT, DELETE) to define the operations
13. Step 4: Implement API logic
    a. Write Java code to implement the API logic for each endpoint
    b. Use Spring Boot's built-in support for database operations (e.g., Hibernate)
14. Step 5: Test the API
15. Step 6: Choose a framework and setup
    a. Choose a framework: React.js or Angular.js
    b. Create a new project in your chosen framework
    c. Set up the project structure and dependencies
16. Step 7: Create UI components
    a. Create UI components for the frontend (e.g., forms, buttons, lists)
    b. Use HTML/CSS/JavaScript to create the UI components
17. Step 8: Implement API communication
    a. Use JavaScript libraries like Axios or Fetch to make HTTP requests to the backend API
    b. Send data to the backend for CRUD operations
    c. Handle responses from the backend and update the frontend accordingly
18. Step 9: Implement business logic
    a. Write JavaScript code to implement business logic for each UI component
19. Step 10: Test the frontend
20. Use your browser or emulator to test the frontend
21. Clean your work place as per standard procedure.
22. Turn off the computer and clean your workplace.

Specification Sheet-1.2: Build a backend service that serves as an API and integrates with a database. Develop a simple frontend (web or mobile) that communicates with the backend 62 API. Implement CRUD operations, allowing the frontend to interact seamlessly with the backend.

Conditions for the job: Work must be carried out in a safe manner and according to relevant competency standards.

List of required PPE

S/N	Name of PPE	Specification	Unit	Required Quantity
01	Ergonomic Chair	Wood and foam	No	1
02	Eye protective glass	Metal and Glass	No	1
03	Mask	Surgical mask	1	1

List of required Software

S/N	Name of Materials	Specification	Unit	Required Quantity
01	IntelliJ IDEA	Latest version	…	1
02	Android Studio	Latest version	…	1
03	Java	JDK Latest version	…	1

List of required Tools & Equipment's

S/N	Name	Specification	Unit	Required Quantity
01	Personal Computer or Laptop	Minimum Core i5 7th gen Processor	No	1
04	Internet connection		…	1

Job Sheet-1.3: Implement an API Gateway that serves as a central entry point for multiple microservices or backend APIs. Configure the API Gateway to route requests to the appropriate services. Create and demonstrate the integration of the API Gateway in a distributed system.

Working Procedure/ Steps:

1. Follow OSH and use Personal Protective Equipment (PPE).
2. Check Electricity & Internet Connections to your Computer.
3. Start the Computer.
4. Collect the resources from your trainer as per the job requirement.
5. Choose an API Gateway framework
6. Create the API Gateway project
    a. Open your preferred IDE (e.g., IntelliJ IDEA) and create a new Spring Boot project.
    b. Choose the "Web" dependency and click "Next".
    c. Add the necessary dependencies:
    d. Configure the project structure
7. Define API routes
    a. Create a new Kotlin file called ApiGatewayConfig.kt and add the following code
8. Implement microservices
    a. Create two separate Kotlin projects for each microservice
9. Integrate API Gateway with microservices
    a. Update the ApiGatewayConfig.kt file to include the microservice URLs
10. Run the API Gateway and microservices
    a. Run the api-gateway project using spring boot run.
    b. Run the user-service and product-service projects using spring boot run.
11. Test the API Gateway
    a. Use a tool like Postman or cURL to test the API Gateway:
    b. Send a GET request to http://localhost:8080/api/users.
    c. Verify that you receive a response with a list of users.
    d. Send a GET request to http://localhost:8080/api/products.
    e. Verify that you receive a response with a list of products.
12. Clean your work place as per standard procedure.
13. Turn off the computer and clean your workplace.

Specification Sheet-1.3: Implement an API Gateway that serves as a central entry point for multiple microservices or backend APIs. Configure the API Gateway to route requests to the appropriate services. Create and demonstrate the integration of the API Gateway in a distributed system.

Conditions for the job: Work must be carried out in a safe manner and according to relevant competency standards.

List of required PPE

S/N	Name of PPE	Specification	Unit	Required Quantity
01	Ergonomic Chair	Wood and foam	No	1
02	Eye protective glass	Metal and Glass	No	1
03	Mask	Surgical mask	1	1

List of required Software

S/N	Name of Materials	Specification	Unit	Required Quantity
01	IntelliJ IDEA	Latest version	…	1
02	Android Studio	Latest version	…	1
03	Java	JDK Latest version	…	1

List of required Tools & Equipment's

S/N	Name	Specification	Unit	Required Quantity
01	Personal Computer or Laptop	Minimum Core i5 7th gen Processor	No	1
04	Internet connection		…	1

Learning Outcome 2: Implement custom (REST) API

Assessment Criteria	1. Custom API is defined 2. Custom API is integrated
Conditions and Resources	• Actual workplace or training environment • CBLM • Handouts • Job related tools, equipment, and materials • Multimedia Projector • Paper, Pen, Pencil, and Eraser • Internet Facilities • Whiteboard and Marker
Contents	1. HTTP Verb (Get, Post, Put, Delete) 2. Custom Application Programming Interface (API) 3. Integrate custom API
Activities/job/Task	1. Build a simple backend server (you can use a framework like Flask, Express, or Django) that exposes the custom API endpoints. Implement logic to handle requests and return appropriate weather data from your sample dataset.
Training Methods	• Blended • Discussion • Presentation • Demonstration • Guided Practice • Individual Practice • Project Work • Problem Solving • Brainstorming
Assessment Methods	Assessment methods may include but not limited to • Written Test • Demonstration • Oral Questioning

Learning Experience 2: Implement custom (REST) API

In order to achieve the objectives stated in this learning guide, you must perform the learning steps below. Beside each step are the resources or special instructions you will use to accomplish the corresponding activity.

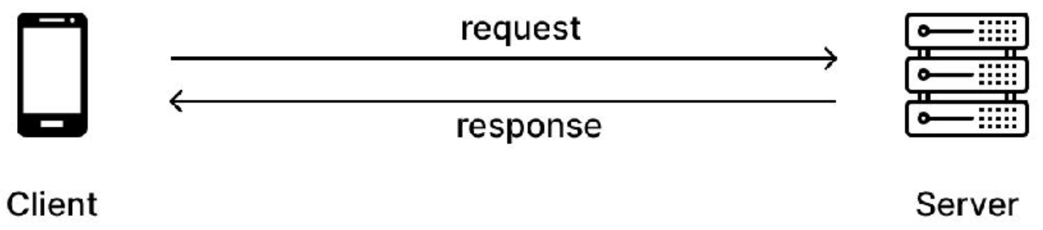
Learning Activities	Recourses/Special Instructions
1. Trainee will ask the instructor about the learning materials	1. Instructor will provide the learning materials Implement custom (REST) API
2. Read the Information sheet and complete the Self Checks & Check answer sheets on “Implement custom (REST) API”	2. Read Information sheet 1: Implement custom (REST) API 3. Answer Self-check 1: Implement custom (REST) API 4. Check your answer with Answer key 1: Implement custom (REST) API
3. Read the Job/Task Sheet and Specification Sheet and perform job/Task	4. Job/Task Sheet and Specification Sheet **Job Sheet 2.1:** Build a simple backend server (you can use a framework like Flask, Express, or Django) that exposes the custom API endpoints. Implement logic to handle requests and return appropriate weather data from your sample dataset. **Specification Sheet 2.1:** Build a simple backend server (you can use a framework like Flask, Express, or Django) that exposes the custom API endpoints. Implement logic to handle requests and return appropriate weather data from your sample dataset.

Information Sheet 2: Implement custom (REST) API

Learning Objective: After completion of this information sheet, the learners will be able to explain, define and interpret the following contents:

2.1 HTTP Verb (Get, Post, Put, Delete)
2.2 Custom Application Programming Interface (API)
2.3 Integrate custom API

2.1 HTTP Verb (Get, Post, Put, Delete)



HTTP which stands for Hyper-Text Transfer Protocol is a protocol that helps us to communicate over the internet, communication in the sense that we can send data from a source to a destination, retrieve or get data from a source(server), update an existing data and delete it when we wish to. For us to successfully carry out these tasks, we have to rely on some of the HTTP paradigms, which include but not limited to the get method, the post, put, patch, read and delete methods. API which stands for Application Programming Interface can be referred to as data source or location, where structured data are kept which can be accessed from different platforms for software development, given that you have the right credentials to access the data.

We can use HTTP methods like GET, POST, PUT, and DELETE to create a new request to these APIs and can perform the following operations:

- GET: Fetch data from API.
- POST: Adding new data to API.
- PUT: Updating existing data.
- DELETE: Deleting data on the server.

Restful web services often transmit data in JSON or XML format. Let us start with Rest API integration with Android application.

Project Overview

Last.fm, The world's largest online music service, provides. The Last.fm API allows developers to access a vast database of music-related data and build their own applications using this data. We will create our application with this API. The name of our application will be MusicWiki. MusicWiki is an unofficial Last.fm app that contains information about different music genres, the albums, artists and tracks listed under the genre or tag.

Goal

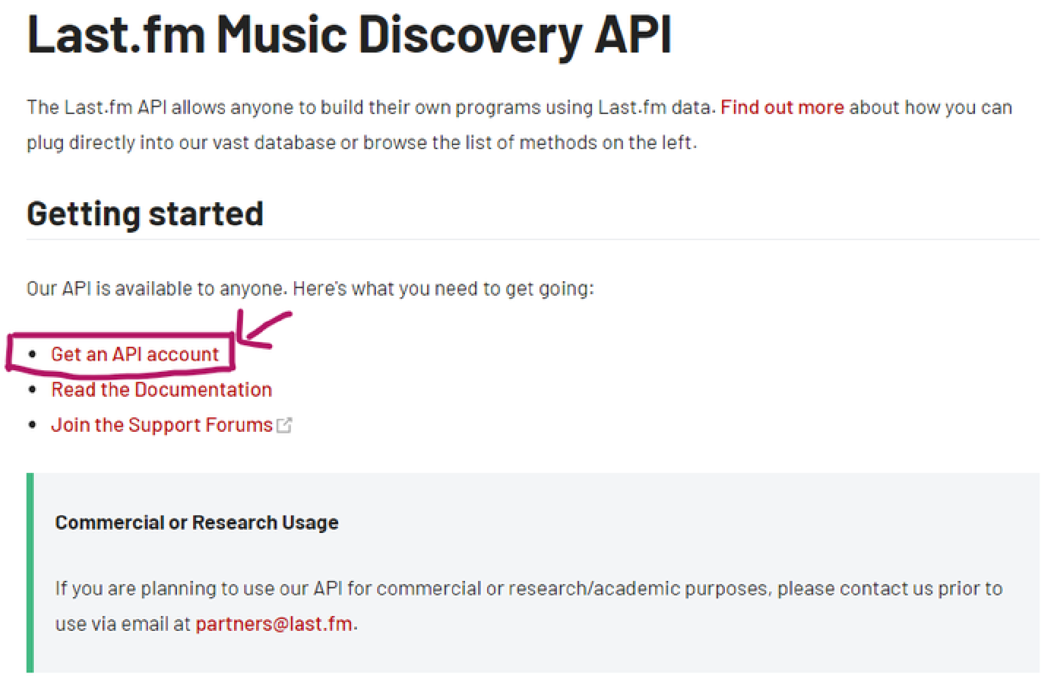
Through this project we will going to cover the following topics:

- Rest API integration with Retrofit Library.
- Learn how to thrott API Documentation.
- Authentication Handling
- Error Handling and Analytics
- Third Party Libraries.
- Step by Step Implementation

Step 1: Download the Project

We will download the Project, In order to proceed further.

Step 2: Set up your last.fm **account in order to get API key**



Last.fm Music Discovery API

The Last.fm API allows anyone to build their own programs using Last.fm data. Find out more about how you can plug directly into our vast database or browse the list of methods on the left.

Getting started

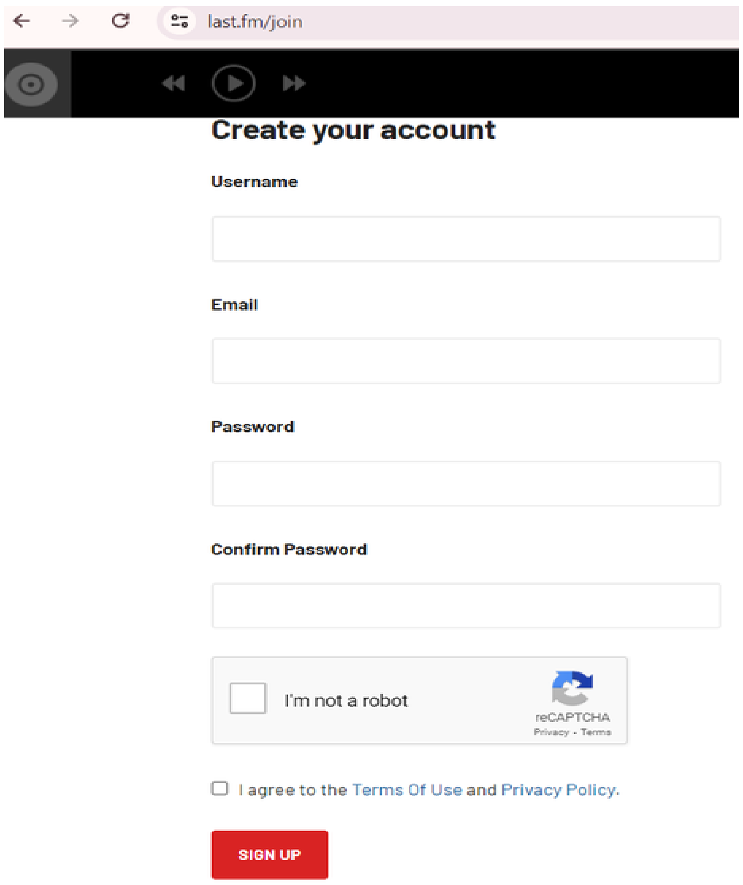
Our API is available to anyone. Here's what you need to get going:

- Get an API account
- Read the Documentation
- Join the Support Forums

Commercial or Research Usage

If you are planning to use our API for commercial or research/academic purposes, please contact us prior to use via email at partners@last.fm.

Create your own account.


last.fm/join

Create your account

Username

Email

Password

Confirm Password

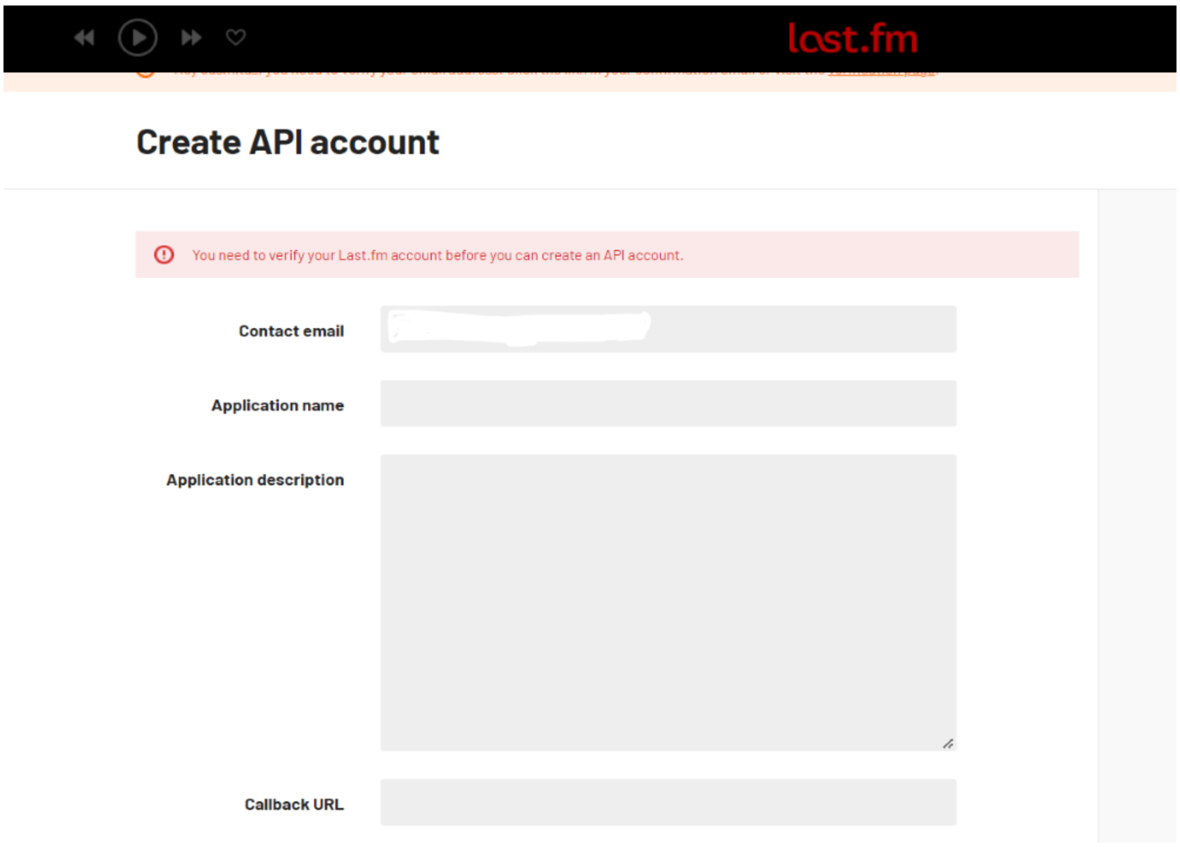
I'm not a robot

reCAPTCHA
Privacy - Terms

I agree to the Terms Of Use and Privacy Policy.

SIGN UP

Fill up the information required for your application. (Don't need to give the Callback URL & Homepage name, if you don't have such information.)


last.fm

Create API account

You need to verify your Last.fm account before you can create an API account.

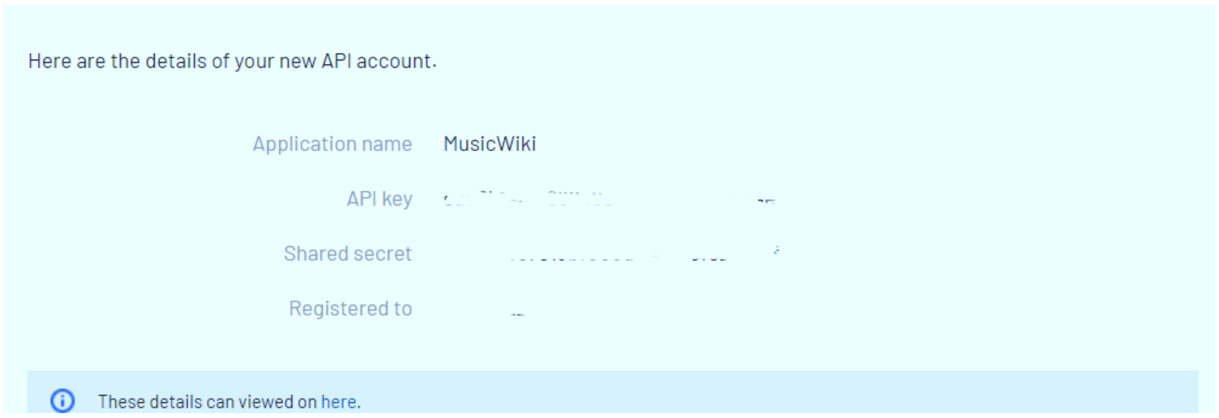
Contact email

Application name

Application description

Callback URL

Click on profile, verify your email and return back to the page. When you successfully cover the following step, you get your API key.



Here are the details of your new API account.

Application name	MusicWiki
API key	[redacted]
Shared secret	[redacted]
Registered to	[redacted]

These details can viewed on here.

Copy the API_KEY and stored it into string.xml file.
<string name="API_KEY"></string>

Do you know, What have you learned for so long? The answer is Throttling API Documentation. As we mentioned earlier, Restful Web Services have their own protocols for accessing them. So our first task was to select one such well-documented and easily accessible API And find out from its documentation how we can use it. In the future, we will see how this API works and where we have to use the API_KEY. (Authentication Handling).

Step 3: Prepare your Android Application for API integration
See here we have two options, we can code from scratch, Here you may need more knowledge and it is also a hectic process. Or we can take the help of third-party in-built libraries. Retrofit is a REST Client library (Helper Library) used in Android and Java to create an HTTP request and also to process the HTTP response from a REST API.

Set up Retrofit for Jetpack-Compose application

Add the following dependencies in your app level **build.gradle** file. and sync the project.

```
// Retrofit for API requests
implementation ("com.squareup.retrofit2:retrofit:2.9.0")
implementation ("com.squareup.retrofit2:converter-gson:2.9.0")
// ViewModel and LiveData for MVVM architecture
implementation ("androidx.lifecycle:lifecycle-viewmodel-compose:2.7.0")
implementation ("androidx.lifecycle:lifecycle-livedata-ktx:2.7.0")
```

Create the API interface as well as Retrofit instance

Keep in mind, That you have entered to the first phase of Error Handling and Analytics. Because, it's important to implement robust error handling and retry mechanisms to handle scenarios like server errors, network failures, or timeouts.

```
// DataModel.kt
package com.example.musicwiki.model
import java.io.Serializable
data class Genre(
   val toptags: Toptags
)
data class Toptags(
   val tag: List<Tag>
)
data class Tag(
   val count: String,
   val name: String,
   val url: String
) : Serializable
data class GenreDetail(
   val tag: GenreInfo
)
data class GenreInfo(
   val name: String,
   val reach: Int,
   val total: Int,
   val wiki: Wiki
)
data class Wiki(
   val content: String,
   val summary: String,
   val published: String,
)
```

Let's try to deep dive onto the following code-snippet.

```
@GET("2.0/")   // API_ENDPOINT
 fun getGenres(@Query("method") method: String, @Query("api_key") apiKey:
String,@Query("format") format:String): Call<Genre>
 @GET("2.0/")
 fun getTagInfo(@Query("method")method: String,
@Query("tag")tagname:String,@Query("api_key") apiKey: String,@Query("format")
format:String)
 : Call<GenreDetail>
```

When you build an application, first structure your requests. It is also applicable to your own APIs. In our application, We only needed to create GET requests to fetch the required data. The second part is to know the api endpoint and see if any authorization is needed for this request or not. This part will always customized according to your API requirement.

Authentication Handling:
In many cases, your API might need an authentication, In that case, you need to include an appropriate authentication header, or you need to pass an API_KEY with requests, like in our case.

Create the ViewModel

```
package com.example.musicwiki.viewModel
import android.util.Log
import androidx.lifecycle.ViewModel
import androidx.lifecycle.viewModelScope
import com.example.musicwiki.model.Genre
import com.example.musicwiki.model.GenreDetail
import com.example.musicwiki.model.GenreInfo
import com.example.musicwiki.model.Tag
import com.example.musicwiki.services.DataService
import kotlinx.coroutines.flow.MutableStateFlow
import kotlinx.coroutines.flow.StateFlow
import kotlinx.coroutines.launch
import retrofit2.Call
import retrofit2.Callback
import retrofit2.Response
class MainViewModel: ViewModel() {
   private val apiService = DataService.lastFmService
   private val _genres = MutableStateFlow<List<Tag>>(emptyList())
   val genres: StateFlow<List<Tag>> = _genres
   val _genreDetail: MutableStateFlow<GenreInfo?> =  MutableStateFlow(null)
   val genreDetail:StateFlow<GenreInfo?> = _genreDetail
   private val apiKey = "xxxxxxxxxxxx"
   fun getAllGenres() {
       viewModelScope.launch {
           try {
               val response = apiService.getGenres("tag.getTopTags",apiKey,"json")
              response.enqueue(object: Callback<Genre> {
                  override fun onResponse(call: Call<Genre>, response: Response<Genre>)
{
                      val body = response.body();

                      Log.d("Response",body.toString())
                      if(response.isSuccessful){
                         _genres.value= body?.toptags?.tag!!
                      }
                  }
                 /** Handle Network Error, Server Error */
                  override fun onFailure(call: Call<Genre>, t: Throwable) {
                      Log.d("GenreError",t.localizedMessage)
                  }
               })
           } catch (e: Exception) {

               /** Handle Error */
               Log.d("GenreError", e.localizedMessage)
           }
       }
   }
   fun getGenreInfo(tagName:String){

       viewModelScope.launch {
           try{
               val                                                         genreInfoResponse=
DataService.lastFmService.getTagInfo("tag.getinfo",tagName,apiKey,"json")
               genreInfoResponse.enqueue(object: Callback<GenreDetail> {
                  override      fun      onResponse(call:      Call<GenreDetail>,      response:
Response<GenreDetail>) {
                      val body = response.body();
                      Log.d("Response",body.toString())
                      if(response.isSuccessful){
                          _genreDetail.value= body?.tag
                      }
                  }
                  /** Handle Network Error, Server Error */
                  override fun onFailure(call: Call<GenreDetail>, t: Throwable) {
                    Log.d("GenreDetailError",t.localizedMessage)
                }
            })
        }catch(e:Exception){
            /** Handle Error */
            Log.d("GenreDetailError", e.localizedMessage)
        }
    }
  }

}
```

Let us deep-dive into ViewModel and try to find out why error handling is one of the crucial parts of API integration.

Error Handling and Analytics:

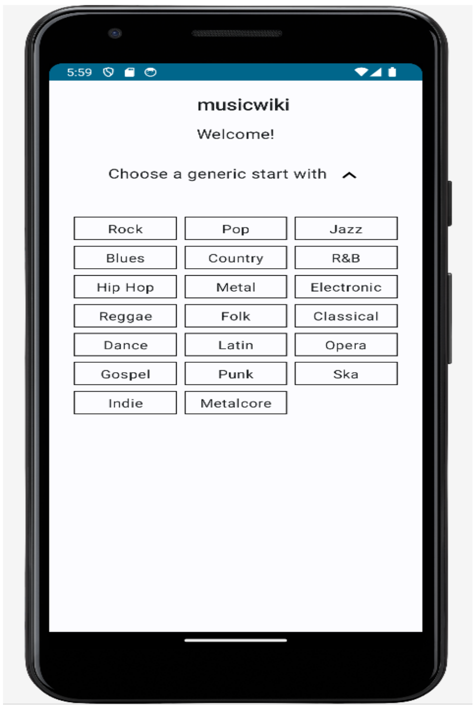
There may be many reasons for a failure request to API, such as a server error, or network error (unavailability of internet connection. If we don't consider those cases, it can result in an unnecessary application crash. Retrofit offers a simple way to manage errors and extract information from the response body. Creating a custom Retrofit CallAdapter allows for centralized handling of API error responses, reducing boilerplate code effectively.

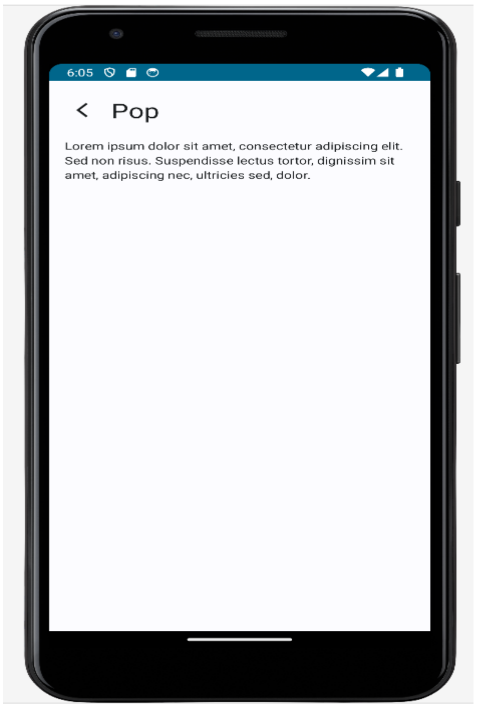
```
** Handle Network Error, Server Error */
override fun onFailure(call: Call<GenreDetail>, t: Throwable) {
            Log.d("GenreDetailError",t.localizedMessage)
        }
    })
  }
  catch(e:Exception){
      /** Handle Error */
      Log.d("GenreDetailError", e.localizedMessage)
}
```

Create the ViewModel object inside MainActivity and pass it to required component

```
// MainActivity.kt
package com.example.musicwiki
import android.os.Bundle
import androidx.activity.ComponentActivity
import androidx.activity.compose.setContent
import androidx.activity.viewModels
import androidx.compose.foundation.layout.fillMaxSize
import androidx.compose.material3.MaterialTheme
import androidx.compose.material3.Surface
import androidx.compose.material3.Text
import androidx.compose.runtime.Composable
import androidx.compose.ui.Modifier
import androidx.compose.ui.tooling.preview.Preview
import com.example.musicwiki.navigation.Navigation
import com.example.musicwiki.ui.theme.MusicWikiTheme
import com.example.musicwiki.viewModel.MainViewModel
class MainActivity : ComponentActivity() {
  private val viewModel: MainViewModel by viewModels()
  override fun onCreate(savedInstanceState: Bundle?) {
    super.onCreate(savedInstanceState)
    setContent {
      MusicWikiTheme {
        // A surface container using the
         // 'background' color from the theme
        Surface(
          modifier = Modifier.fillMaxSize(),
          color = MaterialTheme.colorScheme.background
        ) {
          Navigation(viewModel = viewModel)
        }
      }
    }
  }
}
```

Final Overview of your Application





Lorem ipsum dolor sit amet, consectetur adipiscing elit. Sed non risus. Suspendisse lectus tortor, dignissim sit amet, adipiscing nec, ultricies sed, dolor.

- On Clicking any Button, you can check the data about that Music Form:
- Check what happens when we click Pop on the screen

2.2 Custom Application Programming Interface (API)

A custom application programming interface (API) allows developers to create their own APIs and define the functionality they want to expose to external systems, applications, and services. Custom APIs are the key to communication between applications, third-party services, and backend services. They can be used for many purposes, such as data exchange with cloud software, payment processing, and user registration.

A Custom Application Programming Interface (API) is a software interface that allows different applications to communicate with each other and exchange data in a structured and standardized way. Unlike public APIs, which are widely available and accessible to anyone, custom APIs are developed specifically for a particular organization or application, and are typically used to integrate different systems, services, or applications within an organization.

Custom APIs are designed to meet the specific needs of the organization or application they serve, and can be tailored to provide a unique set of features, functionality, and data exchange protocols. They can be used to:

a. **Integrate different systems:** Custom APIs can be used to integrate different systems, applications, or services within an organization, allowing them to share data and functionality.
b. Automate processes: Custom APIs can automate manual processes and workflows by enabling different systems to communicate with each other automatically.
c. Provide data access: Custom APIs can provide authorized access to data stored in one system or application from another system or application.
d. Enable mobile app development: Custom APIs can provide a way for mobile apps to interact with other systems or applications within an organization.

Types of Custom APIs:

a. Backend API: A backend API is used to integrate different backend systems and services within an organization.
b. Frontend API: A frontend API is used to integrate different frontend systems and applications within an organization.
c. Microservices API: A microservices API is used to integrate different microservices within an organization.
d. Public-facing API: A public-facing API is used to integrate external services or applications with an organization's internal systems.

Benefits of Custom APIs:

a. Improved integration: Custom APIs enable better integration between different systems and applications within an organization.

b. Increased efficiency: Custom APIs can automate manual processes and workflows, reducing the need for manual intervention.
c. Improved data sharing: Custom APIs provide a standardized way for different systems to share data, reducing errors and inconsistencies.
d. Scalability: Custom APIs can be designed to scale with the growing needs of the organization.
e. Security: Custom APIs can be designed with security in mind, providing a secure way for different systems to communicate with each other.

Challenges of Custom APIs:

a. Complexity: Custom APIs can be complex to design and develop, requiring specialized expertise.
b. Integration challenges: Integrating different systems and applications can be challenging, requiring careful planning and testing.
c. Maintenance and updates: Custom APIs require ongoing maintenance and updates to ensure they continue to function correctly.
d. Security risks: Custom APIs present security risks if not designed with security in mind.

Best Practices for Developing Custom APIs:

a. Define clear requirements: Clearly define the requirements for the custom API, including the data exchange protocol, authentication, and authorization.
b. Design for scalability: Design the custom API with scalability in mind, ensuring it can handle increasing traffic and data volumes.
c. Use standard protocols: Use standard protocols such as RESTful API or GraphQL for communication between systems.
d. Implement security measures: Implement robust security measures such as encryption, authentication, and authorization to protect against unauthorized access.
e. Test thoroughly: Thoroughly test the custom API before deploying it in production.

In summary, custom APIs are a powerful tool for integrating different systems, applications, or services within an organization, but require careful planning, design, and implementation to ensure they meet the specific needs of the organization and provide a secure and scalable solution.

2.3 Integrate custom API

Integrating a custom API involves several steps:

- **API Design:** Define the API's endpoints, data structures, and communication protocols. This includes deciding on the API's architecture, data formats, and security measures.
- **API Development:** Develop the API using a programming language such as Java, Python, or Node.js. This involves creating the API's backend, including the server-side logic, database integration, and data storage.
- **API Testing:** Test the API thoroughly to ensure it works correctly and meets the required functionality and performance standards.
- **API Documentation**: Create documentation for the API, including its endpoints, request/response formats, and usage guidelines.
- **API Deploymen**t: Deploy the API in a production-ready environment, ensuring it is secure, scalable, and accessible.
- **API Integration**: Integrate the custom API with other systems, applications, or services within an organization or with external partners.
- **API Maintenance**: Monitor and maintain the API regularly to ensure it remains secure, scalable, and performing well.

Some common integration methods for custom APIs include

- **RESTful API:** Use RESTful API principles to integrate with other systems or applications that support RESTful APIs.
- **GraphQL API:** Use GraphQL APIs to integrate with other systems or applications that support GraphQL.
- **SOAP-based Integration**: Use SOAP-based integration to integrate with other systems or applications that support SOAP-based APIs.
- **Message Queues**: Use message queues like Apache Kafka, RabbitMQ, or Amazon SQS to integrate with other systems or applications that produce and consume messages.
- **Webhooks:** Use webhooks to integrate with other systems or applications that trigger events or callbacks.

Best practices for integrating a custom API:

- **Use standard protocols**: Use standard protocols like RESTful API or GraphQL to ensure compatibility with other systems or applications.
- **Document everything**: Document the API's endpoints, request/response formats, and usage guidelines to ensure easy understanding and use by developers.
- **Test thoroughly**: Thoroughly test the API before deploying it in production to ensure it works correctly and meets required functionality and performance standards.

- **Monitor and maintain**: Monitor and maintain the API regularly to ensure it remains secure, scalable, and performing well.
- **Secure authentication and authorization:** Implement robust authentication and authorization mechanisms to ensure secure access to the API.
- **Use load balancers and caching:** Use load balancers and caching mechanisms to ensure high availability, scalability, and performance of the API.
- **Keep it simple:** Keep the API design simple and intuitive to make it easy for developers to use.

Some common challenges when integrating a custom API include:

- **Data format compatibility:** Ensuring data formats are compatible between different systems or applications.
- **Security concerns:** Ensuring secure authentication and authorization mechanisms are in place.
- **Scalability issues:** Ensuring the API can handle increased traffic and data volumes.
- **Integration complexity:** Integrating with multiple systems or applications can be complex and time-consuming.
- **Testing challenges:** Thoroughly testing the API can be challenging due to its complexity.

Self-Check - 2: Implement custom (REST) API

Questionnaire

1. What is a REST API?

   Answer:
2. What are the common HTTP verbs used in REST APIs?**

   Answer:
3. What is the purpose of the GET verb in a REST API?**

   Answer:
4. What is the purpose of the POST verb in a REST API?**

   Answer:
5. What is the purpose of the PUT verb in a REST API?**

   Answer:
6. What is the purpose of the DELETE verb in a REST API?**

   Answer:
7. How do you handle errors in a REST API?**

   Answer:
8. How do you secure a REST API?**

   Answer:
9. What is JSON (JavaScript Object Notation) in the context of a REST API?**

   Answer:
10. How do you implement pagination in a REST API?**

    Answer:
11. What is rate limiting in a REST API?**

    Answer:
12. How do you integrate a custom API with a web application?**

    Answer:
13. What is caching in a REST API?**

    Answer:
14. How do you handle versioning in a REST API?**

    Answer:
15. What are some best practices for designing a RESTful API?**

    Answer:

Answer Key - 2: Implement custom (REST) API

16. What is a REST API?
    Answer:A REST API (Representational State of Resource) is an architectural style for designing networked applications that uses HTTP requests to manipulate and retrieve data.

17. What are the common HTTP verbs used in REST APIs?**
    Answer: The common HTTP verbs used in REST APIs are GET, POST, PUT, DELETE, PATCH, and OPTIONS.

18. What is the purpose of the GET verb in a REST API?**
    Answer: The GET verb is used to retrieve data from a server without modifying it.

19. What is the purpose of the POST verb in a REST API?**
    Answer: The POST verb is used to create new data on the server.

20. What is the purpose of the PUT verb in a REST API?**
    Answer: The PUT verb is used to update existing data on the server.

21. What is the purpose of the DELETE verb in a REST API?**
    Answer: The DELETE verb is used to delete existing data on the server.

22. How do you handle errors in a REST API?**
    Answer: You can handle errors in a REST API by using HTTP status codes (e.g., 404 for Not Found, 500 for Internal Server Error) and providing error messages or error responses.

23. How do you secure a REST API?**
    Answer: You can secure a REST API by using techniques such as authentication (e.g., username/password), authorization (e.g., role-based access control), and encryption (e.g., SSL/TLS).

24. What is JSON (JavaScript Object Notation) in the context of a REST API?**
    Answer: JSON is a lightweight data interchange format used to represent data in a human-readable format. It is commonly used in REST APIs to exchange data between client and server.

25. How do you implement pagination in a REST API?**
    Answer: You can implement pagination in a REST API by adding query parameters (e.g., page, per_page) to your requests and limiting the number of records returned.

26. What is rate limiting in a REST API?**
    Answer: Rate limiting is a technique used to limit the number of requests an IP address can make to an API within a certain time.

27. How do you integrate a custom API with a web application?**
    Answer: You can integrate a custom API with a web application by using libraries or frameworks that provide HTTP client functionality (e.g., Axios, jQuery).

28. What is caching in a REST API?**
    Answer: Caching is a technique used to store frequently accessed data in memory or on disk to improve performance and reduce latency.

29. How do you handle versioning in a REST API?**
    Answer: You can handle versioning in a REST API by using URI path segments (e.g., /api/v1/users) or query parameters (e.g., ?version=1).

30. What are some best practices for designing a RESTful API?**
    Answer: Some best practices for designing a RESTful API include using resource-based URIs, using standard HTTP methods, using JSON or other standard formats for data exchange, and following uniform resource naming conventions.

Job Sheet-2.1 Build a simple backend server (you can use a framework like Flask, Express, or Django) that exposes the custom API endpoints. Implement logic to handle requests and return appropriate weather data from your sample dataset.

Working Procedure/ Steps:

1. Follow OSH and use Personal Protective Equipment (PPE).
2. Check Electricity & Internet Connections to your Computer.
3. Start the Computer.
4. Collect the resources from your trainer as per the job requirement.
5. Install Flask and required libraries
    a. Install Flask by running pip install flask in your terminal.
    b. Install the json library for handling JSON data: pip install json
6. Create a new Python file for the server
    a. Create a new file called server.py and add the following code:
7. Run the server
    a. Run the server by executing python server.py in your terminal.
    b. The server will start and you should see the message Running on http://127.0.0.1:5000/ (Press CTRL+C to quit).
8. Test the API endpoint
    a. Open a new terminal window and use curl to send a GET request to the /weather endpoint: curl http://127.0.0.1:5000/weather
    b. The response should be in JSON format
9. Clean your work place as per standard procedure.
10. Turn off the computer and clean your workplace.

Specification Sheet-2.1: Build a simple backend server (you can use a framework like Flask, Express, or Django) that exposes the custom API endpoints. Implement logic to handle requests and return appropriate weather data from your sample dataset.

Conditions for the job: Work must be carried out in a safe manner and according to relevant competency standards.

List of required PPE

S/N	Name of PPE	Specification	Unit	Required Quantity
01	Ergonomic Chair	Wood and foam	No	1
02	Eye protective glass	Metal and Glass	No	1
03	Mask	Surgical mask	1	1

List of required Software

S/N	Name of Materials	Specification	Unit	Required Quantity
01	IntelliJ IDEA	Latest version	…	1
02	Android Studio	Latest version	…	1
03	Java	JDK Latest version	…	1

List of required Tools & Equipment's

S/N	Name	Specification	Unit	Required Quantity
01	Personal Computer or Laptop	Minimum Core i5 7th gen Processor	No	1
04	Internet connection		…	1

Learning Outcome 3: Implement firebase API

Assessment Criteria	1. Firebase API is defined 2. Firebase API is integrated 3. Firebase authentication is performed 4. Firebase Realtime database is displayed
Conditions and Resources	• Actual workplace or training environment • CBLM • Handouts • Job related tools, equipment, and materials • Multimedia Projector • Paper, Pen, Pencil, and Eraser • Internet Facilities • Whiteboard and Marker
Contents	1. Firebase API 2. Firebase API integration 3. Firebase authentication 4. Firebase Realtime database
Activities/job/Task	1. Integrate Firebase API in a Web Application with following activity: a. Set Up Firebase Project b. Enable Firebase Authentication c. Integrate Real-Time Database d. Incorporate Firebase Cloud Storage e. Implement Firebase Cloud Functions f. Handle Firebase Authentication State g. Implement Firebase Hosting h. Test and Debug
Training Methods	• Blended • Discussion • Presentation • Demonstration • Guided Practice • Individual Practice • Project Work • Problem Solving • Brainstorming
Assessment Methods	Assessment methods may include but not limited to • Written Test • Demonstration • Oral Questioning

Learning Experience 3: Implement firebase API

In order to achieve the objectives stated in this learning guide, you must perform the learning steps below. Beside each step are the resources or special instructions you will use to accomplish the corresponding activity.

Learning Activities	Recourses/Special Instructions
1. Trainee will ask the instructor about the learning materials	1. Instructor will provide the learning materials Implement firebase API
2. Read the Information sheet and complete the Self Checks & Check answer sheets on "Implement firebase API"	2. Read Information sheet 1: Implement firebase API 3. Answer Self-check 1: Implement firebase API 4. Check your answer with Answer key 1: Implement firebase API
3. Read the Job/Task Sheet and Specification Sheet and perform job/Task	5. Job/Task Sheet and Specification Sheet **Job Sheet 3.1:** Integrate Firebase API in a Web Application with following activity: a. Set Up Firebase Project b. Enable Firebase Authentication c. Integrate Real-Time Database d. Incorporate Firebase Cloud Storage e. Implement Firebase Cloud Functions f. Handle Firebase Authentication State g. Implement Firebase Hosting h. Test and Debug **Specification Sheet 3.1:** Integrate Firebase API in a Web Application with following activity

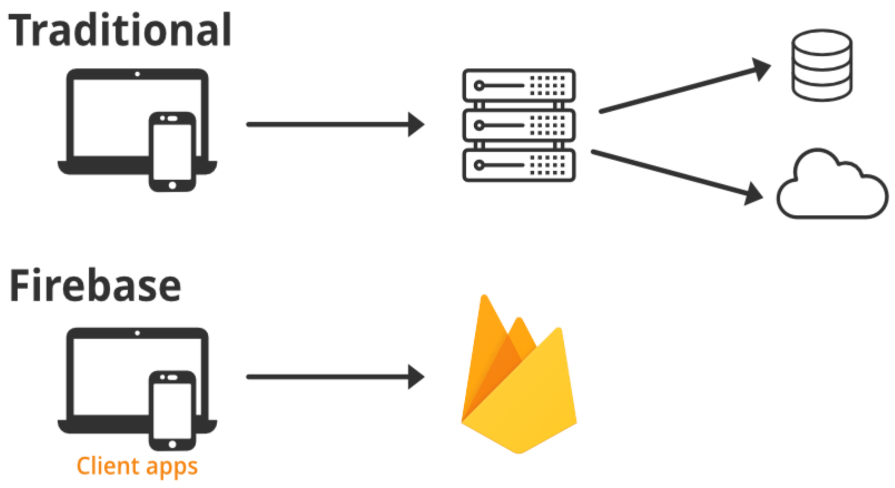
Information Sheet 3: Implement firebase API

Learning Objective: After completion of this information sheet, the learners will be able to explain, define and interpret the following contents:

2.1 Firebase API
2.2 Firebase API integration
2.3 Firebase authentication
2.4 Firebase Realtime database

2.1 Firebase API

Firebase is a product of Google which helps developers to build, manage, and grow their apps easily. It helps developers to build their apps faster and in a more secure way. No programming is required on the firebase side which makes it easy to use its features more efficiently. It provides services to android, ios, web, and unity. It provides cloud storage. It uses NoSQL for the database for the storage of data.



Brief History of Firebase:

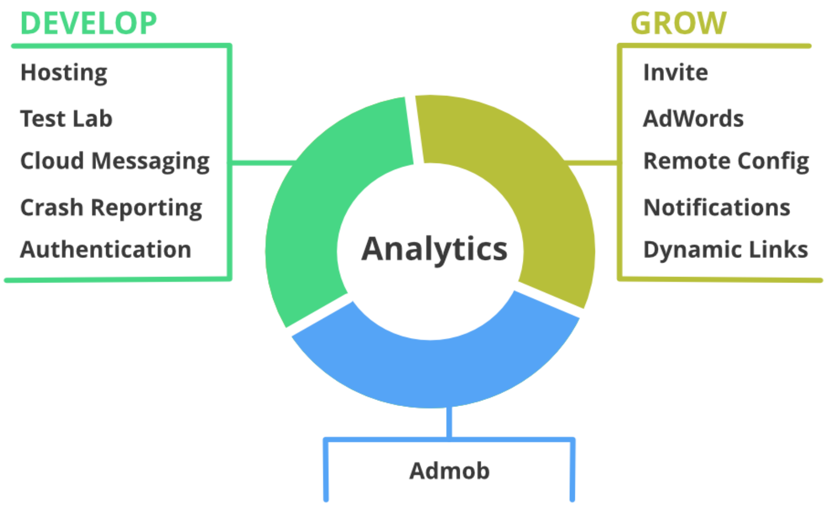
Firebase initially was an online chat service provider to various websites through API and ran with the name Envolve. It got popular as developers used it to exchange application data like a game state in real time across their users more than the chats. This resulted in the separation of the Envolve architecture and it's chat system. The Envolve architecture was further evolved by it's founders James Tamplin and Andrew Lee,to what modern day Firebase is in the year 2012.

Features of Firebase:

Mainly there are 3 categories in which firebase provides its services.

uild better applications

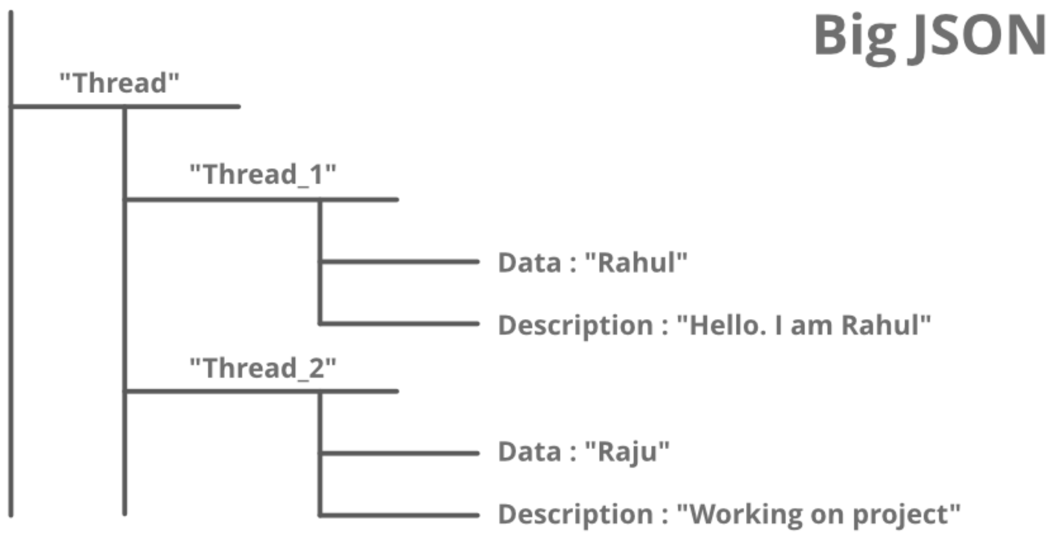
This feature mainly includes backend services that help developers to build and manage their applications in a better way. Services included under this feature are :



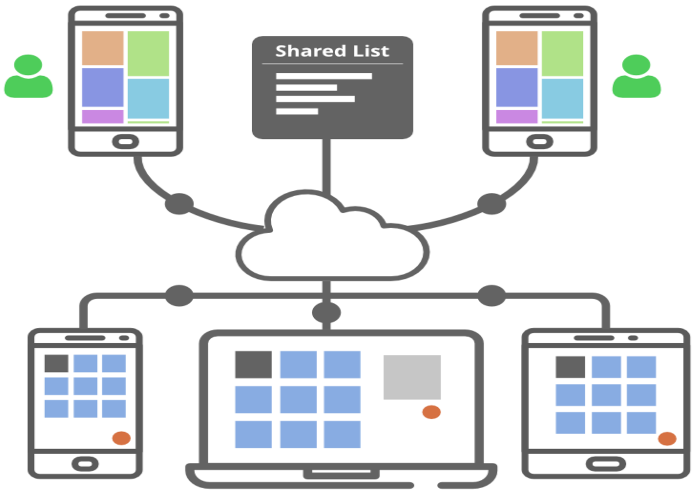
Build better applications

This feature mainly includes backend services that help developers to build and manage their applications in a better way. Services included under this feature are

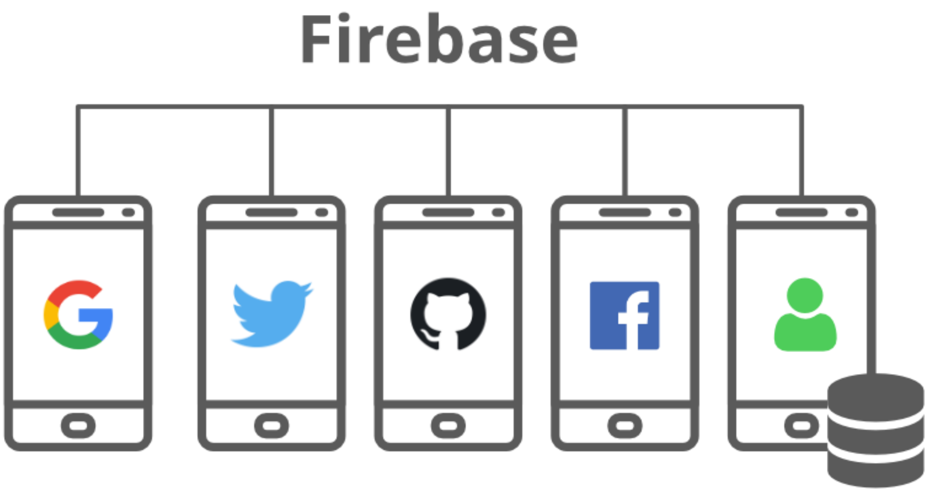
Realtime Database: The Firebase Realtime Database is a cloud-based NoSQL database that manages your data at the blazing speed of milliseconds. In simplest term, it can be considered as a big JSON file.


Big JSON
"Thread"
"Thread_1"
Data : "Rahul"
Description : "Hello. I am Rahul"
"Thread_2"
Data : "Raju"
Description : "Working on project"

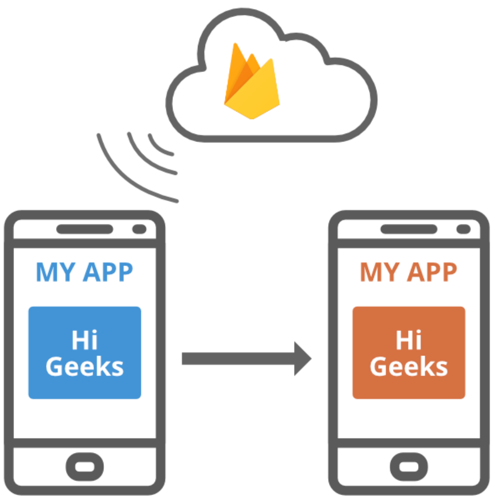
Cloud Firestore: The cloud Firestore is a NoSQL document database that provides services like store, sync, and query through the application on a global scale. It stores data in the form of objects also known as Documents. It has a key-value pair and can store all kinds of data like, strings, binary data, and even JSON trees.



Authentication: Firebase Authentication service provides easy to use UI libraries and SDKs to authenticate users to your app. It reduces the manpower and effort required to develop and maintain the user authentication service. It even handles tasks like merging accounts, which if done manually can be hectic.

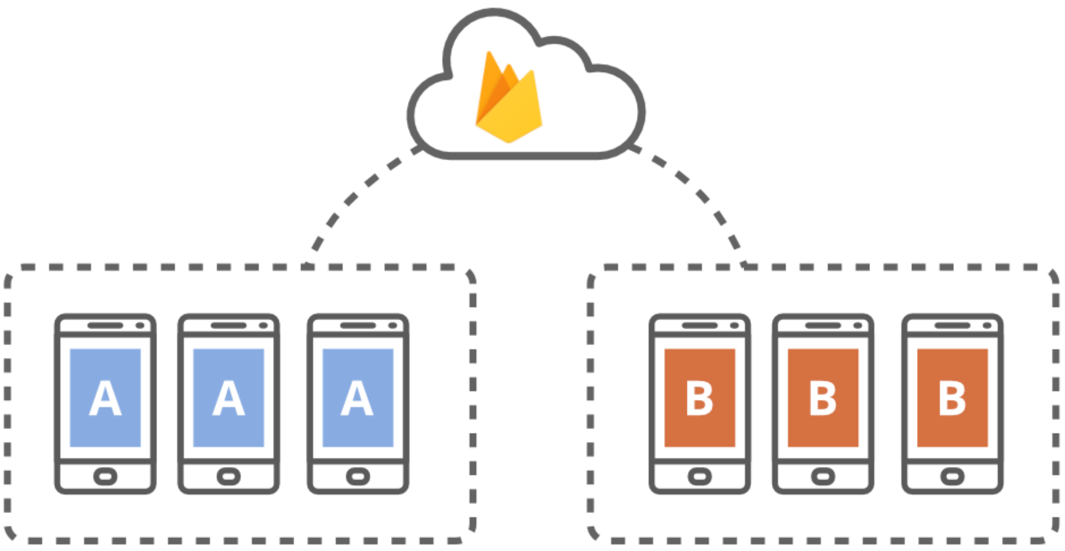


Remote Config: The remote configuration service helps in publishing updates to the user immediately. The changes can range from changing components of the UI to changing the behavior of the applications. These are often used while publishing seasonal offers and contents to the application that has a limited life.



Hosting: Firebase provides hosting of applications with speed and security. It can be used to host Stati or Dynamic websites and microservices. It has the capability of hosting an application with a single command.

Firebase Cloud Messaging(FCM): The FCM service provides a connection between the server and the application end users, which can be used to receive and send messages and notifications. These connections are reliable and battery-efficient.



2.2 Firebase API integration

Open the Firebase Assistant: Tools > Firebase. In the Assistant pane, choose a Firebase product to add to your app. Expand its section, then click the tutorial link (for example, Analytics > Log an Analytics event). Click Connect to Firebase to connect your Android project with Firebase.

Two ways to add firebase

- **Using Firebase Assistant**
- **Manually adding firebase**

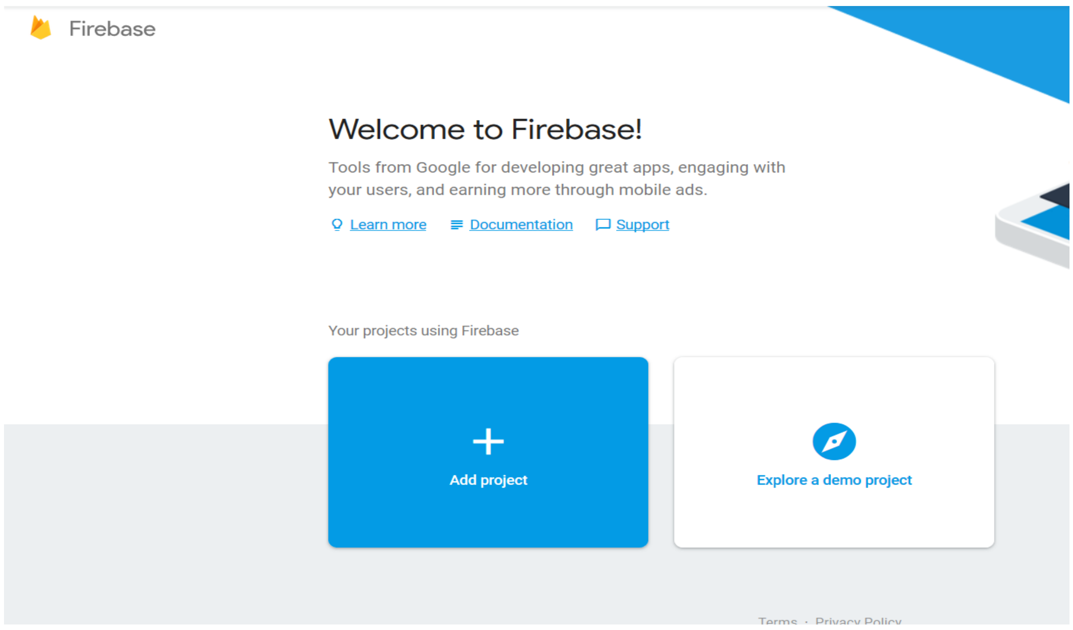
Using Firebase Assistant

Below are the steps to include Firebase to Android project in Android studio:

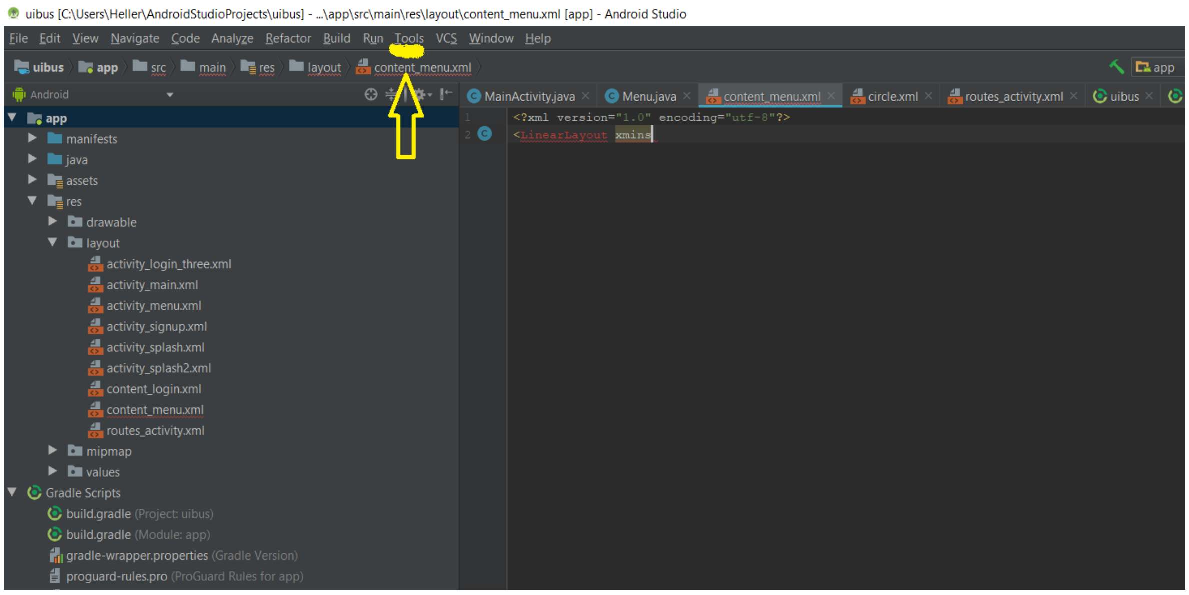
a. Update the android studio (>= 2.2)

b. Create a new project in the firebase by clicking on the Add project.

c.

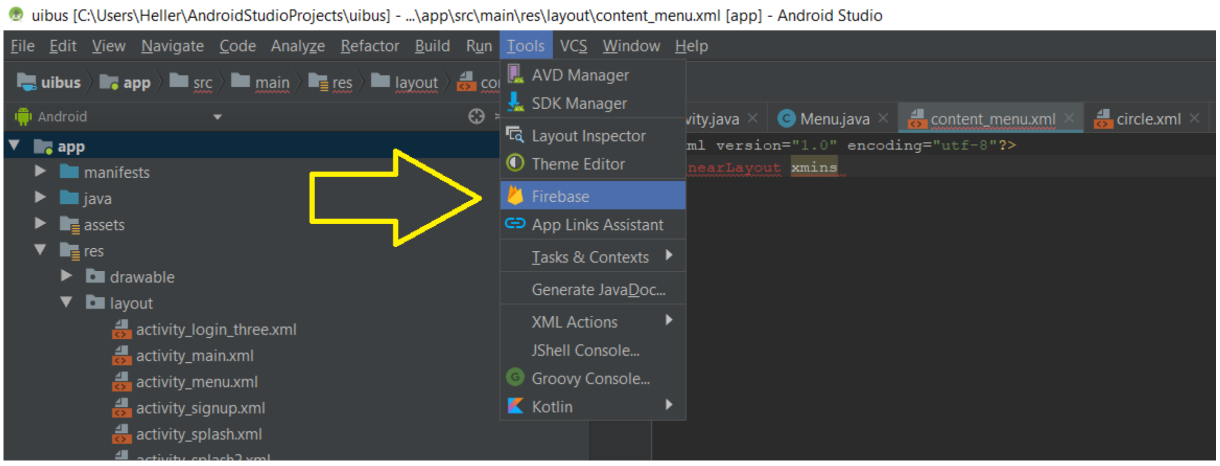

Firebase
Welcome to Firebase!
Tools from Google for developing great apps, engaging with your users, and earning more through mobile ads.
Learn more
Documentation
Support
Your projects using Firebase
Add project
Explore a demo project
Terms · Privacy Policy

d. Now open the android studio and click on Tools in the upper left corner.


uibus [C:\Users\Heller\AndroidStudioProjects\uibus] - ...\app\src\main\res\layout\content_menu.xml [app] - Android Studio
File Edit View Navigate Code Analyze Refactor Build Run Tools VCS Window Help
uibus app src main res layout content_menu.xml
Android
MainActivity.java
Menu.java
content_menu.xml
circle.xml
routes_activity.xml
uibus
app
manifests
java
assets
res
drawable
layout
activity_login_three.xml
activity_main.xml
activity_menu.xml
activity_signup.xml
activity_splash.xml
activity_splash2.xml
content_login.xml
content_menu.xml
routes_activity.xml
mipmap
values
Gradle Scripts
build.gradle (Project: uibus)
build.gradle (Module: app)
gradle-wrapper.properties (Gradle Version)
proguard-rules.pro (ProGuard Rules for app)

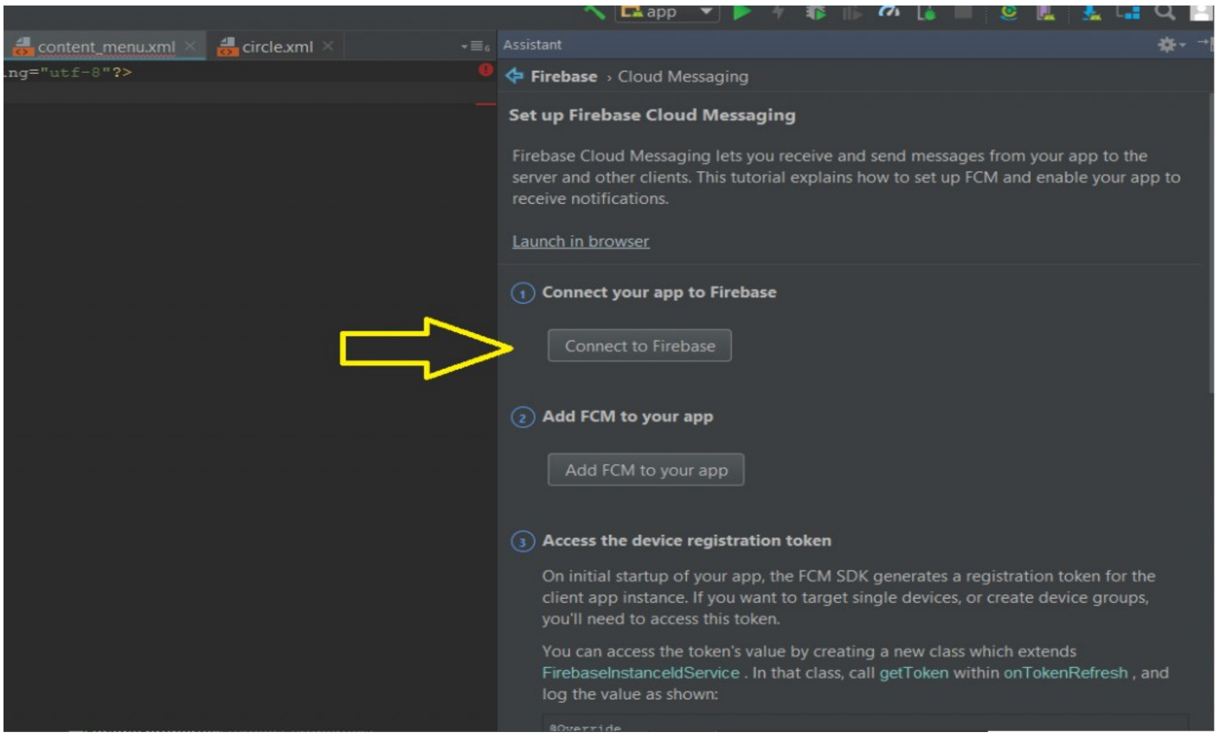
```
<?xml version="1.0" encoding="utf-8"?>
<LinearLayout xmlns
```

e. Now click on the **Firebase** option in the drop down menu.



f. A menu will appear on the right side of screen. It will show various **services** that Firebase offers. Choose the desired service.

g. Now Click on the **Connect to Firebase** option in the menu of desired service.



Set up Firebase Cloud Messaging

Firebase Cloud Messaging lets you receive and send messages from your app to the server and other clients. This tutorial explains how to set up FCM and enable your app to receive notifications.

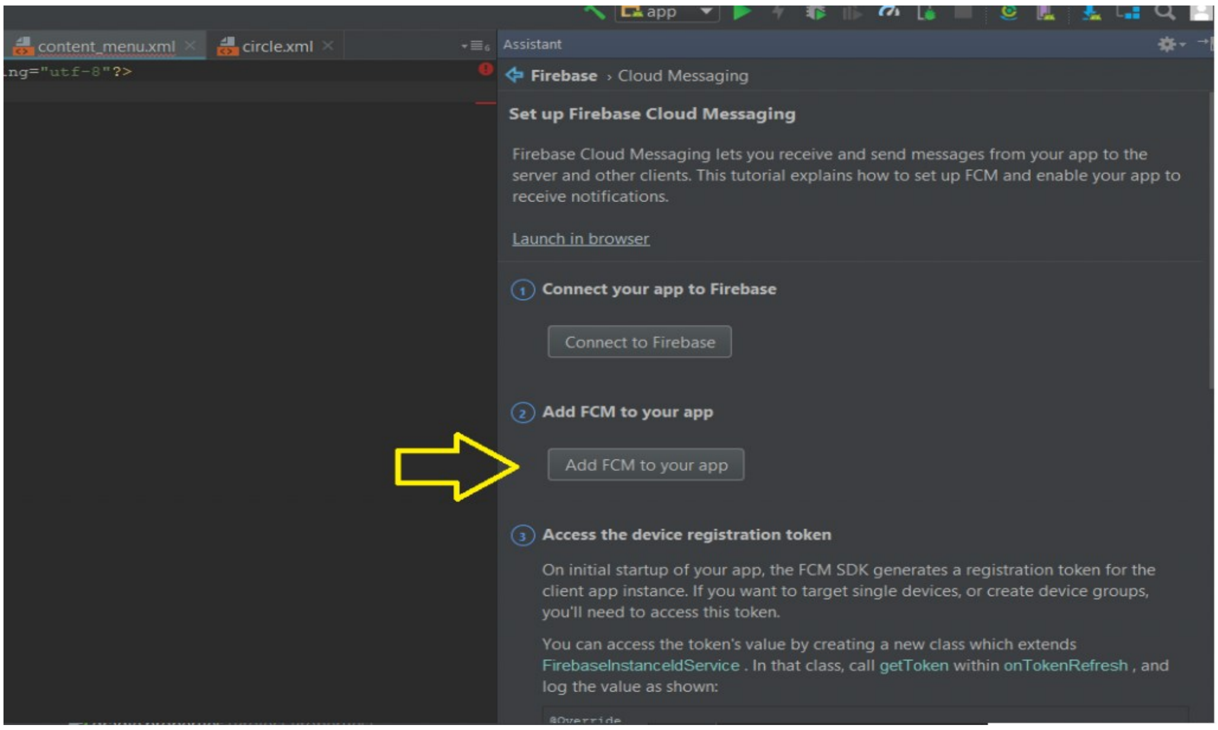
Launch in browser

1. Connect your app to Firebase
2. Add FCM to your app
3. Access the device registration token

On initial startup of your app, the FCM SDK generates a registration token for the client app instance. If you want to target single devices, or create device groups, you'll need to access this token.

You can access the token's value by creating a new class which extends FirebaseInstanceIdService . In that class, call getToken within onTokenRefresh , and log the value as shown:

h. Add the dependencies of your service by clicking on the **Add [YOUR SERVICE NAME] to the app option**. (In the image below, the Firebase cloud messaging service is chosen)



Set up Firebase Cloud Messaging

Firebase Cloud Messaging lets you receive and send messages from your app to the server and other clients. This tutorial explains how to set up FCM and enable your app to receive notifications.

Launch in browser

1. Connect your app to Firebase
2. Add FCM to your app
3. Access the device registration token

On initial startup of your app, the FCM SDK generates a registration token for the client app instance. If you want to target single devices, or create device groups, you'll need to access this token.

You can access the token's value by creating a new class which extends FirebaseInstanceIdService . In that class, call getToken within onTokenRefresh , and log the value as shown:

2.3 Firebase authentication

Firebase is a mobile and web application development platform. It provides services that a web application or mobile application might require. Firebase provides email and password authentication without any overhead of building backend for user authentication.

Core Features:

a. Multiple Authentication Methods:
- **Email and Password:** Users can create accounts with their email addresses and passwords.
- **Phone Authentication:** Users can log in using their phone numbers and receive a verification code via SMS.
- **Federated Identity Providers:** Integration with third-party identity providers like Google, Facebook, Twitter, GitHub, Microsoft, Apple, and others.
- **Anonymous Authentication:** Users can sign in to your app without providing any personal information, useful for temporary user accounts.

b. Secure and Reliable:
- Implements secure protocols for authentication.
- Offers automatic handling of user sessions.
- Provides protection against various security threats like session hijacking, account enumeration, etc.

c. Custom Authentication:
- Supports custom authentication systems where you can integrate your existing authentication mechanisms.
- Ability to create custom tokens to be used with Firebase Authentication.

d. Easy Integration:
- SDKs available for various platforms including iOS, Android, Web, and Unity.
- Integration with other Firebase services (Firestore, Realtime Database, Cloud Functions, etc.) for seamless user data management.

e. User Management:
- Easy-to-use Firebase Console for managing users.
- APIs for programmatically managing users and their credentials.

How It Works

a. Sign-Up and Sign-In:
- Users sign up or log in using their preferred method.
- Firebase Authentication processes the request and verifies the credentials.
- Upon successful authentication, Firebase generates a token for the user session.

b. Token Management:
- Firebase handles the token lifecycle, including refreshing expired tokens.
- Tokens are used to authenticate API requests to other Firebase services.

c. **User Session**
    - Firebase manages user sessions across different devices and platforms.
    - Provides tools for monitoring and controlling user sessions (e.g., session length, session termination).
    -

Implementation Step

a. **Set Up Firebase Project**
    - Create a Firebase project in the Firebase Console.
    - Enable the desired authentication methods in the Authentication section.

b. **Add Firebase SDK to Your App**
    - Integrate the Firebase SDK into your application (iOS, Android, Web, etc.).
    - Initialize Firebase in your app with your project's configuration.

c. **Implement Authentication Logic**
    - Use the SDK methods to implement sign-up, sign-in, and sign-out functionalities.
    - Handle user authentication states and manage user sessions.

d. **Handle Authentication Events**
    - Set up event listeners to respond to authentication state changes.
    - Secure your backend services using Firebase ID tokens.

Example Code (Web)

```
<!DOCTYPE html>
<html>
<head>
  <title>Firebase Authentication</title>
  <script src="https://www.gstatic.com/firebasejs/9.6.1/firebase-app.js"></script>
  <script src="https://www.gstatic.com/firebasejs/9.6.1/firebase-auth.js"></script>
  <script>
    // Your Firebase configuration
    const firebaseConfig = {
      apiKey: "YOUR_API_KEY",
      authDomain: "YOUR_AUTH_DOMAIN",
      projectId: "YOUR_PROJECT_ID",
      storageBucket: "YOUR_STORAGE_BUCKET",
      messagingSenderId: "YOUR_MESSAGING_SENDER_ID",
      appId: "YOUR_APP_ID"
    };

    // Initialize Firebase
    firebase.initializeApp(firebaseConfig);
  </script>
</head>
<body>
```

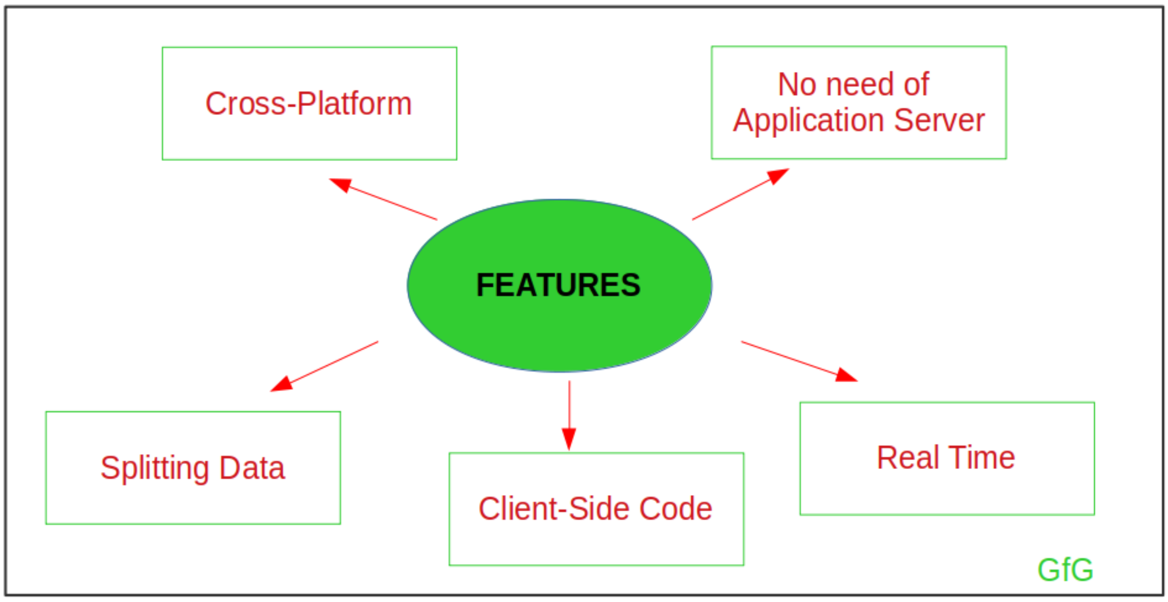
```
  <h1>Firebase Authentication Example</h1>
  <div>
    <input type="email" id="email" placeholder="Email">
    <input type="password" id="password" placeholder="Password">
    <button onclick="signUp()">Sign Up</button>
    <button onclick="signIn()">Sign In</button>
  </div>
  <script>
    const auth = firebase.auth();

    function signUp() {
      const email = document.getElementById('email').value;
      const password = document.getElementById('password').value;
      auth.createUserWithEmailAndPassword(email, password)
        .then(userCredential => {
          console.log('User signed up:', userCredential.user);
        })
        .catch(error => {
          console.error('Error signing up:', error);
        });
    }

    function signIn() {
      const email = document.getElementById('email').value;
      const password = document.getElementById('password').value;
      auth.signInWithEmailAndPassword(email, password)
        .then(userCredential => {
          console.log('User signed in:', userCredential.user);
        })
        .catch(error => {
          console.error('Error signing in:', error);
        });
    }
  </script>
 </body>
</html>
```

This example shows how to integrate Firebase Authentication into a simple web application using email and password authentication.

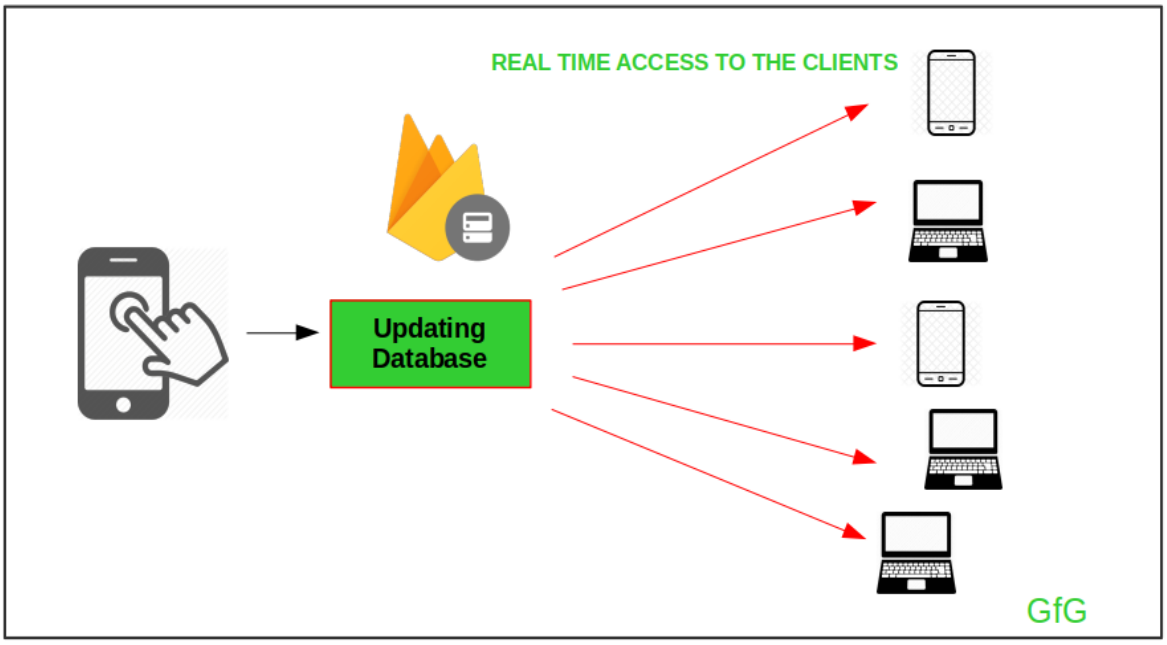
2.4 Firebase Realtime database

Firebase Realtime Database is a Cloud hosted database, i.e. it runs on a cloud and access to the user is provided as a service. It stores data in JSON (Javascript Object Notation) format, a format to store or transport data. All the users connected to it can get access to the data at Real Time.

Features of Firebase Realtime Database?

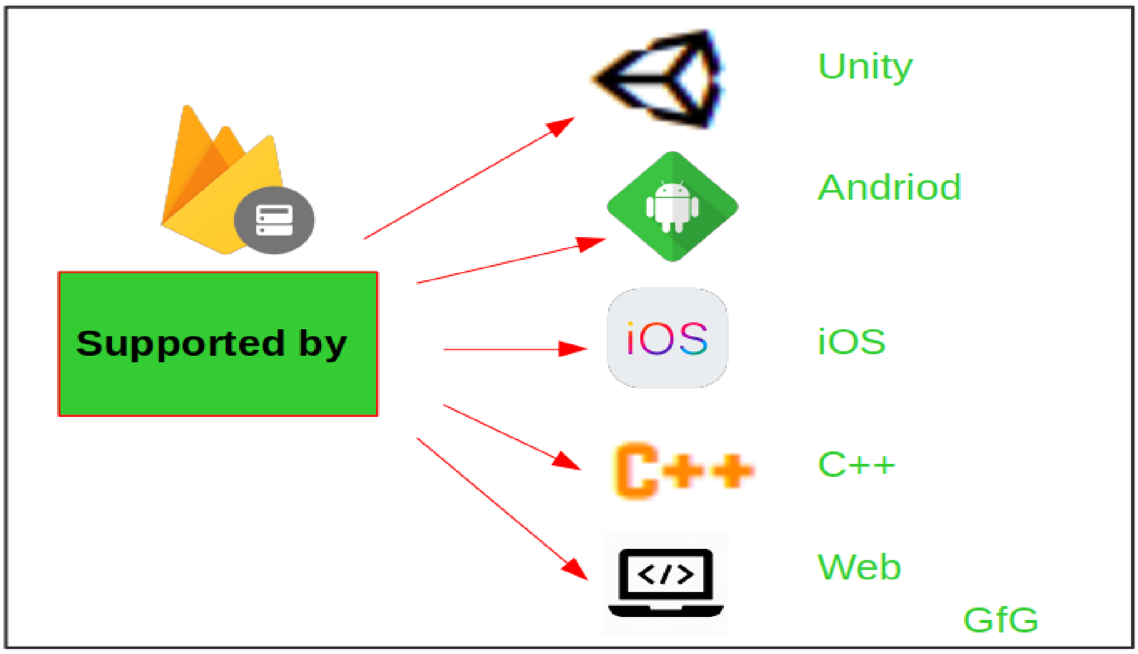


a. Real Time: Due to the Data synchronization used in Real Time, every update is received by the devices/clients in no time.



b. No need of Application Server: As the database can be accessed directly from the mobile device or browser there is no need for an Application Server.

c. Support by various languages and platforms



d. **Splitting Data**: The client can split the data across multiple database instances for the same project.

e. **Client-Side Code**: The dynamic applications with the secured data can be accessed directly from the client-side code.

f. **Cross-Platform**: It can be used for building a back-end for various platforms like Android, iOS, Web, iOS as well as JavaScript SDK.

Structuring the Realtime Database:

Firebase Realtime Database stores data as one large JSON tree. It stores simple data easily, but the unstructured hierarchical data is difficult to organize. Unlike SQL there are no tables here.

What is JSON Tree Model?

JSON Tree Model is inspired from JSON(JavaScript Object Notation) used to represent the JSON document which generally consists of key-value pair in memory .

Why to structure the data (What are the advantages of doing it)?

- If data is stored in well formatted structure then it is easy to save as well as retrieve it easily.
- Querying becomes easy on the structured data.
- It becomes feasible to refer the data in structured format.

Key points to remember while structuring the data:

Before writing and reading the data into the database, the main aim of the developer should be constructing the structure of the database.

Self-Check - 3: Implement firebase API

1. What is Firebase?

   Answer:

2. How do you set up Firebase in a web project?

   Answer:

3. What is Firebase Authentication?

   Answer:

4. How do you enable Firebase Authentication in your project?

   Answer:

5. How do you add a user with email and password in Firebase Authentication?

   Answer:

6. How do you sign in a user with email and password in Firebase Authentication?

   Answer:

7. What is Firebase Realtime Database?

   Answer:

8. How do you read data from Firebase Realtime Database?

   Answer:

9. How do you write data to Firebase Realtime Database?

   Answer:

10. How do you handle authentication state changes in Firebase Authentication?

    Answer:

11. How do you structure data in Firebase Realtime Database?

    Answer:

12. What are Firebase security rules?

    Answer:

13. How do you set Firebase security rules?

    Answer:

14. How do you enable offline capabilities in Firebase Realtime Database?

    Answer

15. How do you authenticate a user using a third-party provider like Google in Firebase?

    Answer:

Answer Key - 3: Implement firebase API

1. What is Firebase?

   Answer: Firebase is a platform developed by Google for creating mobile and web applications. It provides various services like authentication, real-time databases, cloud storage, and hosting.

2. How do you set up Firebase in a web project?

   Answer: To set up Firebase in a web project, create a Firebase project in the Firebase Console, add your web app, and include the Firebase SDK in your HTML file with your project configuration.

3. What is Firebase Authentication?

   Answer: Firebase Authentication is a service that allows you to authenticate users using various methods such as email/password, phone numbers, and third-party providers like Google, Facebook, and Twitter.

4. How do you enable Firebase Authentication in your project?

   Answer: Enable Firebase Authentication by going to the Authentication section in the Firebase Console and turning on the sign-in methods you want to support.

5. How do you add a user with email and password in Firebase Authentication?

   Answer: Use the `createUserWithEmailAndPassword(email, password)` method from the Firebase Authentication SDK.

6. How do you sign in a user with email and password in Firebase Authentication?

   Answer: Use the `signInWithEmailAndPassword(email, password)` method from the Firebase Authentication SDK.

7. What is Firebase Realtime Database?

   Answer: Firebase Realtime Database is a cloud-hosted NoSQL database that stores data in JSON format and syncs it in real-time across all connected clients.

8. How do you read data from Firebase Realtime Database?

   Answer: Use the `on` method (e.g., `on('value', callback)`) to set up a listener for changes at a database reference.

9. How do you write data to Firebase Realtime Database?

   Answer: Use the `set`, `push`, or `update` methods to write data to a database reference.

10. How do you handle authentication state changes in Firebase Authentication?

    Answer: Use the `onAuthStateChanged` method to set up a listener for authentication state changes.

11. How do you structure data in Firebase Realtime Database?

    Answer: Data is structured as a JSON tree, where each node is a key-value pair.

12. What are Firebase security rules?

    Answer: Firebase security rules control access to your Firebase Realtime Database and Cloud Firestore, defining who can read or write to specific parts of your database.

13. How do you set Firebase security rules?

    Answer: Set Firebase security rules in the Firebase Console under the Database section, by editing the rules in the Rules tab.

14. How do you enable offline capabilities in Firebase Realtime Database?

    Answer: Offline capabilities are enabled by default. The Firebase Realtime Database SDK caches data on the device and synchronizes it when the network is available.

15. How do you authenticate a user using a third-party provider like Google in Firebase?

    Answer: Use the `signInWithPopup` or `signInWithRedirect` methods with a `GoogleAuthProvider` object to authenticate a user using Google in Firebase Authentication.

Job Sheet-3.1: Integrate Firebase API in a Web Application

Working Procedure/ Steps:

11. Follow OSH and use Personal Protective Equipment (PPE).
12. Check Electricity & Internet Connections to your Computer.
13. Start the Computer.
14. Collect the resources from your trainer as per the job requirement.
15. Open Android Studio and click on "Start a new Android Studio project".
16. Choose "Empty Activity" and name the project "MyApp".
17. Choose the language as Kotlin and click "Next".
18. Choose the minimum SDK version as API 21 and click "Finish".
19. Set Up Firebase Project
20. Enable Firebase Authentication
21. Integrate Real-Time Database
22. Incorporate Firebase Cloud Storage
23. Implement Firebase Cloud Functions
24. Handle Firebase Authentication State
25. Implement Firebase Hosting
26. Test and Debug.
27. Run the App and Show your work to the trainer.
28. Clean your work place as per standard procedure.
29. Turn off the computer and clean your workplace.

Specification Sheet-3.1: Integrate Firebase API in a Web Application with following activity

Conditions for the job: Work must be carried out in a safe manner and according to relevant competency standards.

List of required PPE

S/N	Name of PPE	Specification	Unit	Required Quantity
01	Ergonomic Chair	Wood and foam	No	1
02	Eye protective glass	Metal and Glass	No	1
03	Mask	Surgical mask	1	1

List of required Software

S/N	Name of Materials	Specification	Unit	Required Quantity
01	IntelliJ IDEA	Latest version	…	1
02	Android Studio	Latest version	…	1
03	Java	JDK Latest version	…	1

List of required Tools & Equipment's

S/N	Name	Specification	Unit	Required Quantity
01	Personal Computer or Laptop	Minimum Core i5 7th gen Processor	No	1
04	Internet connection		…	1

Learning Outcome 4: Implement Firebase Cloud Messaging (FCM)

Assessment Criteria	1. FCM in Android is described. 2. Client and server in FCM is defined. 3. FCM client is implemented. 4. FCM server is implemented. 5. Working with user notification is demonstrated.
Conditions and Resources	• Actual workplace or training environment • CBLM • Handouts • Job related tools, equipment, and materials • Multimedia Projector • Paper, Pen, Pencil, and Eraser • Internet Facilities • Whiteboard and Marker
Contents	1. Firebase Cloud Messaging (FCM) 2. Client and server in FCM 3. FCM client 4. FCM server 5. Topic and channel-based message
Activities/job/Task	1. Integrate Firebase Cloud Messaging (FCM) in an Android App with following activity: a. Create a Firebase Project: b. Add FCM to Your App c. Configure FCM in Your App d. Initialize FCM in Your App e. Implement FCM Token Retrieval f. Handle FCM Message Reception g. Send Test Notification h. Handle Notification Clicks i. Test Real-time Notification Delivery
Training Methods	• Blended • Discussion • Presentation • Demonstration • Guided Practice • Individual Practice • Project Work • Problem Solving • Brainstorming •

Assessment Methods	Assessment methods may include but not limited to • Written Test • Demonstration • Oral Questioning

Learning Experience 4: Implement Firebase Cloud Messaging (FCM)

In order to achieve the objectives stated in this learning guide, you must perform the learning steps below. Beside each step are the resources or special instructions you will use to accomplish the corresponding activity.

Learning Activities	Recourses/Special Instructions
1. Trainee will ask the instructor about the learning materials	1. Instructor will provide the learning materials Implement Firebase Cloud Messaging (FCM)
2. Read the Information sheet and complete the Self Checks & Check answer sheets on "Implement Firebase Cloud Messaging (FCM)"	2. Read Information sheet 1: Implement Firebase Cloud Messaging (FCM) 3. Answer Self-check 1: Implement Firebase Cloud Messaging (FCM) 4. Check your answer with Answer key 1: Implement Firebase Cloud Messaging (FCM)
3. Read the Job/Task Sheet and Specification Sheet and perform job/Task	5. Job/Task Sheet and Specification Sheet **Job Sheet 2.1:** Build a simple backend server (you can use a framework like Flask, Express, or Django) that exposes the custom API endpoints. Implement logic to handle requests and return appropriate weather data from your sample dataset. **Specification Sheet 2.1:** Build a simple backend server (you can use a framework like Flask, Express, or Django) that exposes the custom API endpoints. Implement logic to handle requests and return appropriate weather data from your sample dataset.

Information Sheet 4: Implement Firebase Cloud Messaging (FCM)

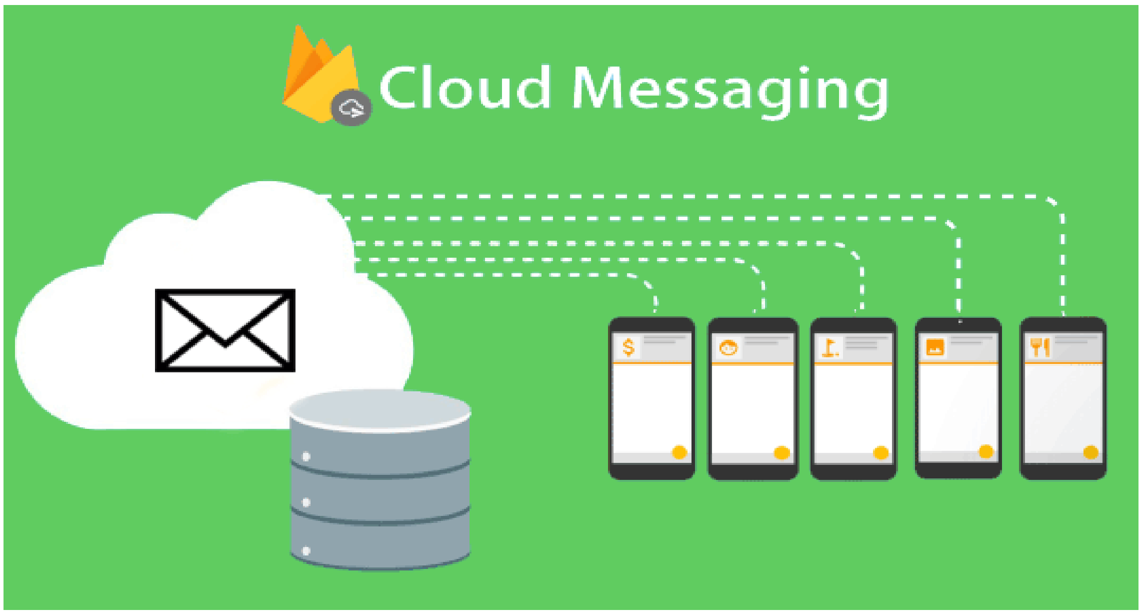
Learning Objective: After completion of this information sheet, the learners will be able to explain, define and interpret the following contents:

2.1 Firebase Cloud Messaging (FCM)
2.2 Client and server in FCM
2.3 FCM client
2.4 FCM server
2.5 Topic and channel-based message.

2.1 Firebase Cloud Messaging (FCM)

Firebase Cloud Messaging

Firebase Cloud Messaging (FCM) is a cross-platform messaging solution which reliably sends the message at no cost. It is formally known as Google Cloud Messaging, which is used for Android, iOS, and web applications.
The service is provided by Google's subsidiary Firebase and on 21 October 2014, Firebase announced that it had been acquired by Google for an undisclosed amount. The official Google Cloud Messaging website demonstrates the Firebase Cloud Messaging (FCM) as the new version of GCM.
If we are using Google Cloud Messaging (GCM) server and client APIs, then there is some bad news, which is that this service has already been removed, and Google plans to discontinue "most" GCM services in April 2019. If we are still using GCM, then we should start migrating our projects to FCM now and complete our migration by April 2019.



How does it work?
FCM implementation contains two main components for sending and receiving. The first one is a trusted environment such as Cloud Function for Firebase or an app server on which to build, target, and send messages, essentially the server-side, and another one is an android client app which receives messages. If we implement our own server code in Cloud Function or Java code, then we can send messages via Firebase Admin SDK or the FCM server protocols.

We can also use the Notification composer for testing or for sending marketing or engagement messages with powerful built-in targeting and analytics.

Implementation path

Set up the FCM SDK
Set up Firebase and FCM on our app according to the setup instruction for our platform.

Develop our client app
In our client app, we have to add message handling, topic subscription logic, or other optional features. During development, we can easily send text messages from the Notification composer.

Develop our app server
We need to decide whether we want to use the Firebase Admin SDK or one of the server protocols to create our sending logic, i.e., logic to authenticate, buildsend requests, handle responses, etc., and build out the logic in our trusted environment.

Concerning the development of our own app server
It will give us the basics of the Server environment, but we will not write any code.

Types of Messages

- **Notification Message**
  Firebase SDK has handled notification messages itself. Typically, the notification message includes the title, icon, message, etc. These messages can also be sent from the Firebase console UI. By sending this type of message, we will not have much control over the information. The notification will appear automatically when the app is in the background.
  The notification design is determined by the system template - the content for each part of the template is defined by our app. Some information on the notification only appears in the expanded view. The most common parts of notification are as follows:
    - **Small icon:** It is essential to set a small icon in the notification. This small icon is set with setSmallIcon().
    - **App name:** The application name is provided by the system.
    - **Timestamp:** The timestamp is also provided by the system, but we can override it with setWhen() function or hide it with **setShowWhen()** function.

    - **Title:** It is also optional and set with setContentTitle() function.
    - **Text:** Text is optional and set with setContentText().
    - **Large icon:** This is optional, and we use it for contact photos. We don't use it for our app icon and set with **setLargeIcon()** function

- **Data Message**
  Data messages are handled by the Android app. If we want to send some additional data along with the information, then we can add such messages. But, it is not possible to send these messages through the Firebase console. To send notifications using the Firebase API, we must have server-side logic. We must use the data key while sending this message.
  We can use data messages to send custom data elements to a client application. However, FCM places a 4KB limit on these data messages, so if our payload is greater than 4KB, we must obtain additional data using the WorkManager or JobScheduler API.

- **Messages with both Notification and Data payload**
  Both Notification and Data payload can also be contained in a message. Sending of these types of messages is handled in two scenarios depending upon the app state, i.e., background and foreground. We can use both the notification and data keys for these messages.

  When the app state is in the background, the apps receive the notification payload when the user taps on the notification, and when in the foreground, the app receives a message object with both payloads available.

2.2 Client and server in FCM

Firebase Cloud Messaging (FCM) allows you to send notifications and messages to users across different platforms (iOS, Android, Web). FCM involves both client-side and server-side components. Here’s a breakdown of the client and server roles in FCM:

Client-side in FCM

1. Initialization

- Initialize Firebase in your app with the Firebase SDK.
- Obtain the FCM token for the device.

2. Receiving Messages:

- Set up listeners to handle incoming messages when the app is in the foreground and background.
- Handle different types of messages: notification messages (automatically displayed) and data messages (handled by the app).

3. Requesting Permissions (Web/iOS):

- For web apps, request permission to display notifications.
- For iOS apps, request permission to send push notifications.

Android Example:

```
// Initializing Firebase in an Android app
FirebaseApp.initializeApp(this);

// Getting the FCM token
FirebaseMessaging.getInstance().getToken().addOnCompleteListener(new
OnCompleteListener<String>() {
    @Override
    public void onComplete(@NonNull Task<String> task) {
        if (!task.isSuccessful()) {
            Log.w(TAG, "Fetching FCM registration token failed", task.getException());
            return;
        }

        // Get new FCM registration token
        String token = task.getResult();

        // Log and toast
        Log.d(TAG, token);
        Toast.makeText(MainActivity.this, token, Toast.LENGTH_SHORT).show();
    }
});
```

Server-side in FCM:

Sending Messages:

- The server can send messages to specific devices, topics, or conditionally using the FCM server API.
- Send messages using HTTP v1 API or the legacy HTTP API.
- Authenticate requests with a server key or OAuth 2.0.

Message Types:

- **Notification Messages:** Automatically displayed by the FCM SDK.
- **Data Messages:** Handled by the app (used for custom notifications or background processing).

```
const admin = require('firebase-admin');

// Initialize the app with a service account, granting admin privileges

admin.initializeApp({

  credential: admin.credential.cert(serviceAccount),

  databaseURL: 'https://<your-database-name>.firebaseio.com'
});
// Send a message to a specific device
const message = {
  notification: {
    title: 'Hello',
    body: 'World'
  },
  token: '<FCM Token>',
};
admin.messaging().send(message)
  .then((response) => {
    console.log('Successfully sent message:', response);
  })
  .catch((error) => {
    console.log('Error sending message:', error);
  });
```

2.3 FCM client

Firebase Cloud Messaging (FCM) client implementation involves setting up your app to receive messages and notifications from FCM. Below are the steps and code examples for setting up an FCM client for Android, iOS, and Web.

FCM Client for Android

Add Firebase to your Android project: Add the Firebase SDK to your build.gradle files.
Project-level build.gradle:

```
buildscript {
    repositories {
        google()
        mavenCentral()
    }
    dependencies {
        classpath 'com.google.gms:google-services:4.3.10' // Check for latest version
    }
}
```

```
allprojects {
   repositories {
      google()
      mavenCentral()
   }
}
```

App-level build.gradle:

```
apply plugin: 'com.android.application'
apply plugin: 'com.google.gms.google-services'

android {
   compileSdkVersion 30
   defaultConfig {
      applicationId "com.example.myapp"
      minSdkVersion 16
      targetSdkVersion 30
      versionCode 1
      versionName "1.0"
   }
   buildTypes {
      release {
         minifyEnabled false
         proguardFiles getDefaultProguardFile('proguard-android-optimize.txt'),
'proguard-rules.pro'
      }
   }
}

dependencies {
   implementation platform('com.google.firebase:firebase-bom:28.4.2') // Check for
latest version
   implementation 'com.google.firebase:firebase-messaging'
}
```

Initialize Firebase in your application

Initialize Firebase in your MainActivity or Application class.

MainActivity.java:

```
import android.os.Bundle;
import androidx.appcompat.app.AppCompatActivity;
import com.google.firebase.messaging.FirebaseMessaging;

public class MainActivity extends AppCompatActivity {
   @Override
```

```
  protected void onCreate(Bundle savedInstanceState) {
    super.onCreate(savedInstanceState);
    setContentView(R.layout.activity_main);

    // Obtain the FCM token
    FirebaseMessaging.getInstance().getToken().addOnCompleteListener(task -> {
      if (!task.isSuccessful()) {
        Log.w(TAG, "Fetching FCM registration token failed", task.getException());
        return;
      }

      // Get new FCM registration token
      String token = task.getResult();

      // Log and toast
      Log.d(TAG, "FCM Token: " + token);
      Toast.makeText(MainActivity.this, token, Toast.LENGTH_SHORT).show();
    });
  }
}
```

2.4 FCM server

To implement an FCM server within Android Studio, you'll typically use Java or Kotlin to create the server-side logic. Here, we'll create a simple FCM server using the Firebase Admin SDK in a Java project within Android Studio. Note that running a server directly within an Android app is not a common practice due to security and architecture concerns. Usually, servers are run on backend services.

However, for learning purposes or small-scale usage, you can create a standalone Java application using the Firebase Admin SDK to send messages to your Android client. Here are the steps:

Step 1: Create a New Java Project in Android Studio

- Open Android Studio.
- Create a new project, and select the "Java" template.
- Name your project, and choose the project location.

Step 2: Add Firebase Admin SDK Dependency

- Add the Firebase Admin SDK to your project's build.gradle file.
- Project-level build.gradle:

```
allprojects {
  repositories {
    google()
    mavenCentral()
  }}
```

App-level build.gradle:

```
apply plugin: 'java'
dependencies {
   implementation 'com.google.firebase:firebase-admin:9.1.1'
}
repositories {
   mavenCentral()
}
```

Step 3: Initialize the Firebase Admin SDK

- Download the service account key file from your Firebase project.
- Save the service account key file (serviceAccountKey.json) in your project directory.

Step 4: Create a Java Class to Send FCM Messages

- Create a class named FCMServer.java:

```
import com.google.firebase.FirebaseApp;
import com.google.firebase.FirebaseOptions;
import com.google.firebase.messaging.FirebaseMessaging;
import com.google.firebase.messaging.Message;
import com.google.firebase.messaging.Notification;
import java.io.FileInputStream;
import java.io.IOException;

public class FCMServer {
   public static void main(String[] args) {
      try {
         // Initialize the app with a service account, granting admin privileges
         FileInputStream serviceAccount = new
FileInputStream("path/to/serviceAccountKey.json");

         FirebaseOptions options = new FirebaseOptions.Builder()
            .setCredentials(GoogleCredentials.fromStream(serviceAccount))
            .build();

         FirebaseApp.initializeApp(options);

         // Send a message to a device
         String registrationToken = "YOUR_REGISTRATION_TOKEN";

         Message message = Message.builder()
            .setNotification(new Notification("Hello", "World"))
            .setToken(registrationToken)
            .build();

         String response = FirebaseMessaging.getInstance().send(message);
```

```
            System.out.println("Successfully sent message: " + response);
        } catch (IOException e) {
            e.printStackTrace();
        }
    }
}
```

Step 5: Run the Java Application

- Ensure you have the service account key file in the correct directory and that the path is correct in the code.
- Run the FCMServer class.

2.5 Topic and channel-based message.

Using topic and channel-based messaging in Android with Firebase Cloud Messaging (FCM) allows you to send messages to multiple devices that have subscribed to a particular topic or channel. Here are the details on how to implement this:

Topic-Based Messaging

Subscribe to a Topic

Devices can subscribe to a topic to receive messages sent to that topic.

```
import com.google.firebase.messaging.FirebaseMessaging
FirebaseMessaging.getInstance().subscribeToTopic("news")
    .addOnCompleteListener { task ->
        var msg = "Subscribed to news topic"
        if (!task.isSuccessful) {
            msg = "Subscription failed"
        }
        Log.d(TAG, msg)
        Toast.makeText(baseContext, msg, Toast.LENGTH_SHORT).show()
    }
```

Channel-Based Messaging

Channels are more applicable in the context of Android notifications starting from Android 8.0 (Oreo) using Notification Channels. You can create multiple channels and assign notifications to these channels.

Create Notification Channels

Create notification channels in your app.

Kotlin Example:

```
if (Build.VERSION.SDK_INT >= Build.VERSION_CODES.O) {
   val channelId = "news_channel"
   val channelName = "News Channel"
   val importance = NotificationManager.IMPORTANCE_DEFAULT
   val channel = NotificationChannel(channelId, channelName, importance).apply {
      description = "Channel for news notifications"
   }
   // Register the channel with the system
   val notificationManager: NotificationManager =
      getSystemService(Context.NOTIFICATION_SERVICE) as NotificationManager
   notificationManager.createNotificationChannel(channel)
}
```

Self-Check - 4: Implement Firebase Cloud Messaging (FCM)

1. What is Firebase Cloud Messaging (FCM)?

   Answer:

2. What is the main difference between FCM and Google Cloud Messaging (GCM)?**

   Answer:

3. What is the FCM client?

   Answer:

4. What is the FCM server?

   Answer:

5. What is a device token in FCM?

   Answer:

6. What are topics in FCM?

   Answer:

7. What are channels in FCM?

   Answer:

8. What is a message in FCM?

   Answer:

9. What are notification messages in FCM?

   Answer:

10. What are data messages in FCM?

    Answer:

11. How do I register a device with FCM?

    Answer:

12. How do I send a message to a device using FCM?

    Answer:

13. How do I handle message delivery retries in FCM?

    Answer:

14. How do I track message delivery statistics in FCM?

    Answer:

15. Is FCM free to use?

    Answer:

Answer Key - 4: Implement Firebase Cloud Messaging (FCM)

1. What is Firebase Cloud Messaging (FCM)?
   Answer: FCM is a cloud-based messaging service offered by Google Cloud that allows developers to send targeted, personalized messages to their users.

2. What is the main difference between FCM and Google Cloud Messaging (GCM)?**
   Answer: FCM is the successor to GCM and offers more features, better performance, and improved reliability.

3. What is the FCM client?
   Answer: The FCM client is the software component that is integrated into your app and interacts with the FCM server to send and receive messages.

4. What is the FCM server?
   Answer: The FCM server is the cloud-based infrastructure that manages device tokens, sends messages, and handles messaging tasks.

5. What is a device token in FCM?
   Answer: A device token is a unique identifier assigned to a device when it registers with the FCM service.

6. What are topics in FCM?
   Answer: Topics are a way to categorize devices based on specific criteria, allowing you to send messages to multiple devices at once.

7. What are channels in FCM?
   Answer: Channels are a way to group devices together based on specific criteria, allowing you to send messages to multiple devices at once.

8. What is a message in FCM?
   Answer: A message is a piece of data sent from the FCM server to a device or group of devices.

9. What are notification messages in FCM?
   Answer: Notification messages are used for displaying notifications to users, such as alerts or updates.

10. What are data messages in FCM?
    Answer: Data messages are used for sending data payloads between the server and client.

11. How do I register a device with FCM?

Answer: You can register a device with FCM using the Firebase console or by implementing the Firebase SDK in your app.

12. How do I send a message to a device using FCM?
    Answer: You can send a message to a device using the Firebase console or by using the Firebase API.

13. How do I handle message delivery retries in FCM?
    Answer: FCM automatically handles message delivery retries for you, but you can also implement custom retry logic if needed.

14. How do I track message delivery statistics in FCM?
    Answer: You can track message delivery statistics using the Firebase console or by implementing analytics in your app.

15. Is FCM free to use?

Answer: Yes, FCM is free to use up to 1 million free messages per day. After that, you can upgrade to a paid plan for additional features and support.

Job Sheet-4.1: Integrate Firebase Cloud Messaging (FCM) in an Android App

Working Procedure/ Steps:

1. Follow OSH and use Personal Protective Equipment (PPE).
2. Check Electricity & Internet Connections to your Computer.
3. Start the Computer.
4. Collect the resources from your trainer as per the job requirement.
5. Open Android Studio and click on "Start a new Android Studio project".
6. Choose "Empty Activity" and name the project "MyApp".
7. Create a Firebase Project:
8. Add FCM to Your App
9. Configure FCM in Your App
10. Initialize FCM in Your App
11. Implement FCM Token Retrieval
12. Handle FCM Message Reception
13. Send Test Notification
14. Handle Notification Clicks
15. Test Real-time Notification Delivery
16. Run the App and Show your work to the trainer.
17. Clean your work place as per standard procedure.
18. Turn off the computer and clean your workplace.

Specification Sheet-4.1: Integrate Firebase Cloud Messaging (FCM) in an Android App with following activity.

Conditions for the job: Work must be carried out in a safe manner and according to relevant competency standards.

List of required PPE

S/N	Name of PPE	Specification	Unit	Required Quantity
01	Ergonomic Chair	Wood and foam	No	1
02	Eye protective glass	Metal and Glass	No	1
03	Mask	Surgical mask	1	1

List of required Software

S/N	Name of Materials	Specification	Unit	Required Quantity
01	IntelliJ IDEA	Latest version	…	1
02	Android Studio	Latest version	…	1
03	Java	JDK Latest version	…	1

List of required Tools & Equipment's

S/N	Name	Specification	Unit	Required Quantity
01	Personal Computer or Laptop	Minimum Core i5 7th gen Processor	No	1
04	Internet connection		…	1

Review of Competency

Below is yourself assessment rating for module "**Performing Application Programming Interface Integration**"

Assessment of performance Criteria	Yes	No
Application Programming Interface (API) is defined.		
API is integrated A simple activity layout is designed for some basic user operation.		
Custom API is defined		
Custom API is integrated		
Firebase API is defined		
Firebase API is integrated		
Firebase authentication is performed		
Firebase Realtime database is displayed		
FCM in Android is described.		
Client and server in FCM is defined.		
FCM client is implemented.		
FCM server is implemented.		
Working with user notification is demonstrated.		

I now feel ready to undertake my formal competency assessment.

Signed:

Date:

Development of CBLM

The Competency based Learning Material (CBLM) of **“Performing Application Programming Interface Integration’ (Occupation: Android Mobile Application Development, Level-4)** for National Skills Certificate is developed by NSDA with the assistance of SIMEC System Ltd., ECF Consultancy & SIMEC Institute of Technology JV (Joint Venture Firm) in the month of July, 2024 under the contract number of package SD-9B dated 15th January 2024.

SL No.	Name & Address	Designation	Contact Number
1	Engr. Md. Zuwel Parves	Writer	01737-278906
2	Md. Abdul Al Hossain	Editor	01778-926438
3	Engr Md. Zuwel Parves	Co-Ordinator	01737-278906
4	Md. Abdur Razzaque	Reviewer	01713-304824

Reference

1. https://reflectoring.io/kotlin-design-patterns/
2. https://www.w3schools.com/kotlin/index.php
3. https://www.tutorialspoint.com/kotlin/index.htm
4. https://medium.com/@oriohac/beginner-friendly-implementation-of-the-http-methods-get-post-put-and-delete-from-apis-in-b4e93952aa29
5. https://www.freecodecamp.org/news/http-request-methods-explained/
6. https://code.tutsplus.com/android-from-scratch-using-rest-apis--cms-27117t
7. https://developer.android.com/games/sdk/performance-tuner/custom-engine/enable-api
8. https://www.geeksforgeeks.org/networking-and-api-integration-in-android/
9. https://firebase.google.com/docs/database/android/start
10. https://www.javatpoint.com/firebase-types-of-message