**Open-Source Software Development**

**Submitted By**

Shehryar Aslam(21-CS-421)

Mashood(21-CS-423)

Usama Bin Adil(21-CS-447)

**Submitted To**

Dr. Abdul Jaleel (Chairman)

**Title: Serverless computing support in fogbus2**

**Abstraction:**

**Overview:**

Serverless computing is a cloud computing model where developers write and deploy code without managing underlying servers. It offers scalability, cost-effectiveness, and faster development cycles, making it ideal for event-driven applications, microservices architecture, and backend services for web and mobile app. Its recent usability has surged due to its ability to handle fluctuating workloads efficiently and facilitate quicker time-to-market for applications.

**Motivation:**

Serverless computing in FogBus2 streamlines resource management, enhancing scalability and reducing operational complexities. By leveraging serverless architecture, FogBus2 optimizes resource utilization and accelerates the deployment of edge computing applications, promoting efficiency and agility in distributed environments.

**State of the art:**

In FogBus2, serverless computing integrates seamlessly with edge environments, optimizing resource allocation and enabling efficient execution of event-driven functions. Its state-of-the-art approach emphasizes dynamic scaling, cost-effectiveness, and simplified development, empowering rapid deployment of applications across distributed fog computing infrastructures.

**Destabilizing:**

In FogBus2, the rapid scalability of serverless computing can potentially destabilize edge environments due to sudden resource fluctuations. Currently, FogBus2 does not fully address the challenges of latency-sensitive applications or the complexities of orchestrating serverless functions across heterogeneous edge devices, which remain areas for further development and research.

**Problem Statement:**

This work aims to propose a novel framework for optimizing serverless computing in FogBus2 by addressing resource allocation challenges and ensuring efficient function execution across distributed edge environments, thus mitigating latency issues and enhancing overall system performance.

**Our methodology:**

Our methodology in serverless computing for FogBus2 involves iteratively refining resource allocation algorithms and function deployment strategies through empirical evaluation and simulation, aiming to optimize performance and scalability in distributed edge environments.

**What is Serverless Computing?**

Serverless computing, also known as Function-as-a-Service (FaaS), is a cloud computing paradigm where developers can build and deploy applications without the need to manage the underlying infrastructure. In traditional computing models, developers are responsible for provisioning and managing servers, configuring networking, and ensuring scalability and availability. However, with serverless computing, developers can focus solely on writing application code, while the cloud provider takes care of server management, scaling, and maintenance.

**How Does Serverless Computing Work?**

**1. Code Deployment:** Developers write their application code as functions, which are small, self-contained units of logic designed to perform a specific task. These functions are then packaged and deployed to a serverless platform provided by cloud vendors like AWS Lambda, Microsoft Azure Functions, or Google Cloud Functions.

**2. Event Triggering:** Serverless functions are typically triggered by events such as HTTP requests, database changes, file uploads, or scheduled events. When an event occurs, the serverless platform automatically invokes the corresponding function to handle it.

**3. Automatic Scaling:** One of the key features of serverless computing is automatic scaling. The cloud provider dynamically allocates resources to handle incoming requests based on demand. If there is a sudden increase in traffic, the platform automatically provisions additional instances of the function to handle the load. Conversely, if the workload decreases, the platform deallocates resources to optimize costs.

**4. Pay-per-Use Billing:** Serverless platforms operate on a pay-per-use billing model, where users are charged only for the resources consumed by their functions and the execution time. Since there are no fixed costs associated with maintaining servers, serverless computing can be more cost-effective, especially for applications with unpredictable or sporadic traffic patterns.

**5. Statelessness:** Serverless functions are inherently stateless, meaning they do not retain any information between invocations. Any required state must be stored externally, such as in a database or a distributed cache. While this statelessness simplifies scalability and fault tolerance, it also introduces challenges for applications that require persistent connections or shared state.

Overview of Serverless computing, epitomized by Function-as-a-Service (FaaS), liberates developers from managing infrastructure, focusing on writing code while cloud providers handle server management and scaling. Functions are deployed to platforms like AWS Lambda, triggered by events, and dynamically scaled based on demand, offering cost-efficiency under a pay-per-use model.

**Code of User interface:**

import 'package:flutter/material.dart';

import 'package:interface\_gui/watch.dart';

void main() {

  runApp(MyApp());

}

class MyApp extends StatelessWidget {

  @override

  Widget build(BuildContext context) {

    return MaterialApp(

      title: 'Cloud Service UI',

      theme: ThemeData(

        primarySwatch: Colors.blue,

      ),

      home: MyHomePage(),

    );

  }

}

class MyHomePage extends StatefulWidget {

  @override

  \_MyHomePageState createState() => \_MyHomePageState();

}

class \_MyHomePageState extends State<MyHomePage> {

  int \_selectedIndex = 0;

  void \_onItemTapped(int index) {

    if (index == 2) {

      Navigator.push(

        context,

        MaterialPageRoute(builder: (context) => SmartWatchApp()),

      );

    } else {

      setState(() {

        \_selectedIndex = index;

      });

    }

  }

  Widget \_buildPage(int index) {

    switch (index) {

      case 0:

        return CloudServicePage1();

      case 1:

        return CloudServicePage2();

      case 2:

        return SmartWatchApp();

      case 3:

        return smartmobile();

      default:

        return Container();

    }

  }

  @override

  Widget build(BuildContext context) {

    return Scaffold(

      appBar: AppBar(

        title: Text('Cloud Service UI'),

      ),

      body: \_buildPage(\_selectedIndex),

      bottomNavigationBar: BottomNavigationBar(

        items: const <BottomNavigationBarItem>[

          BottomNavigationBarItem(

            icon: Icon(Icons.cloud),

            label: 'Cloud Service Provider 1',

          ),

          BottomNavigationBarItem(

            icon: Icon(Icons.phone\_android),

            label: 'Smart Mobile',

          ),

          BottomNavigationBarItem(

            icon: Icon(Icons.watch),

            label: 'IoT Devices',

          ),

        ],

        currentIndex: \_selectedIndex,

        selectedItemColor: Colors.blue,

        onTap: (index) {

          \_onItemTapped(index);

        },

      ),

    );

  }

}

class CloudServicePage1 extends StatelessWidget {

  @override

  Widget build(BuildContext context) {

    return Container(

      color: Colors.blue,

      child: Center(

        child: Text('Cloud Service Provider 1'),

      ),

    );

  }

}

class CloudServicePage2 extends StatelessWidget {

  @override

  Widget build(BuildContext context) {

    return Container(

      color: Colors.green,

      child: Center(

        child: Text('Cloud Service Provider 2'),

      ),

    );

  }

}

class smartmobile extends StatelessWidget {

  @override

  Widget build(BuildContext context) {

    return Container(

      color: const Color.fromARGB(255, 243, 33, 100),

      child: Center(

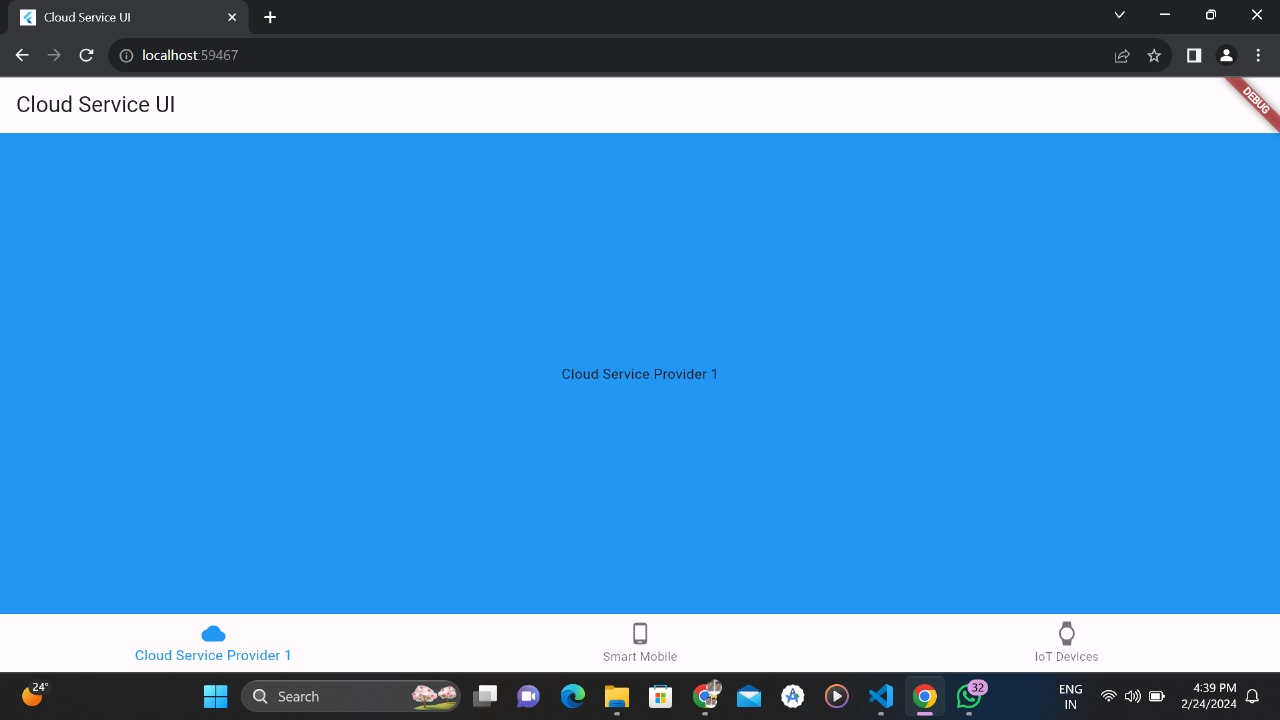
        child: Text('Here it is the Summary of the Smart Mobile'),

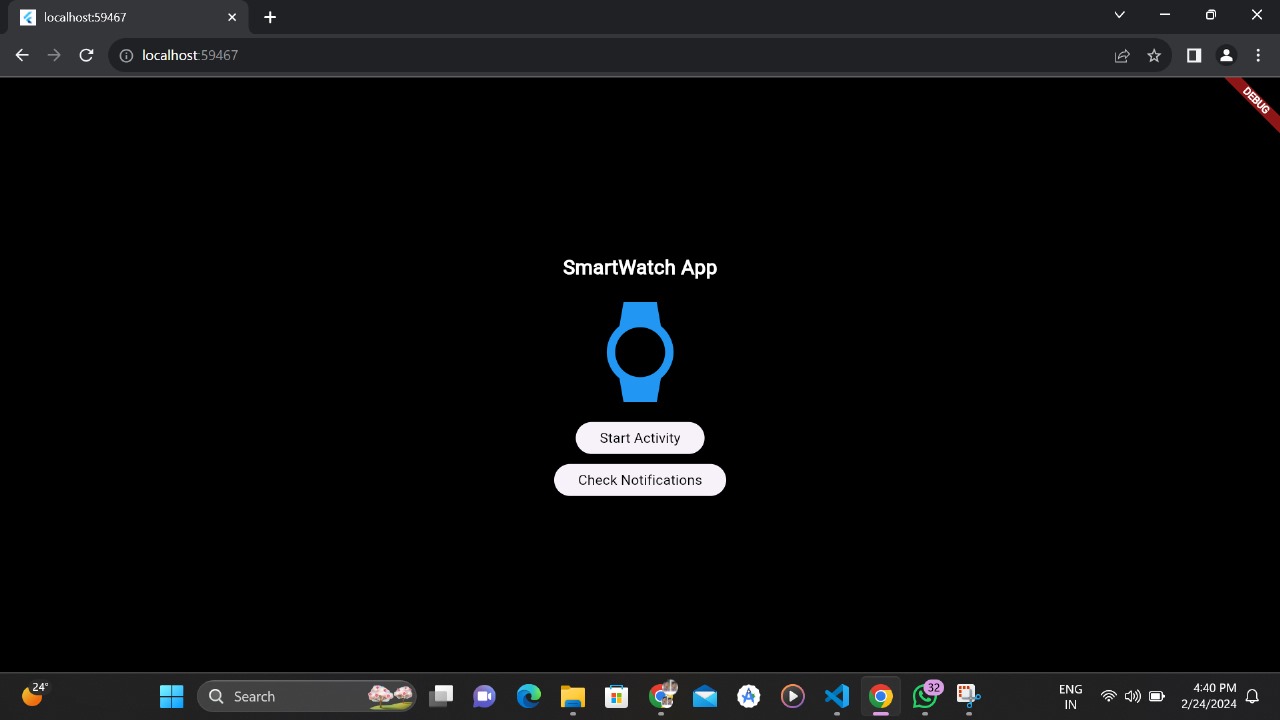
      ),

    );

  }

}





**Next Task:**

We have designing UI and also read about serverless computing i.e., Amazon, Microsoft Azure, Cloud computing and 10 use case of amazon.