VINUNIVERSITY

Operating Systems Project
Theme: Android OS

Course code: COMP2040 **AY 2022 – 2023**

A. Team Members

#	Student Name	Student ID	Email
1	Nguyen Trong Nhan	V202000224	20nhan.nt@vinuni.edu.vn
2	Vo Khoi Thanh Lam	V202000120	20lam.vkt@vinuni.edu.vn
3	Tong Duy Hai	V202000178	20hai.td@vinuni.edu.vn

B. Project Descriptions & Objective

1. Description: In this project, first of all, we will briefly explore an overview of the Android operating system ranging from the history, purpose, market share to its wide range of applications, etc. Next, we will dig deeper into the operating system's important features and components, together with its development upon the release of new versions over time. In addition, we will also discuss Android's benefits and drawbacks, especially when compared to its competitive mobile operating systems namely Apple iOS. The most important topics to be discovered in our project, however, will be centered on the services that the Android operating system provides, including system calls, memory management, error detection, and many other concepts that we already learned in class. Furthermore, to have a better understanding and clearer explanation of the services and how the Android OS works, we will try to install and implement several of its common services using the Java programming language.

2. Objectives: In this project, we have three main objectives. Firstly, we will link theoretical concepts to a particular operating system of Android by explaining which function. Secondly, we will determine the distinguished characteristics of Android by proving the key components and comparing and contrasting differences between it and other operating systems. Finally, we will learn how Android works, demonstrate one of its unique services, and test how it runs. From our team's perspective, the first main reason we choose to explore the Android operating system is that Android is an open-source platform based on the Linux kernel and multiple open-source libraries. Due to that reason, developers like us can freely get access to source code and development tools Android SDK and Eclipse IDE to learn and try to develop an application ourselves. Secondly, the most popular programming language for Android is Java, which is more familiar to VinUni students than C or C++. Therefore, we can install and perform some services in a more comfortable way. Finally, the last reason we choose Android is that our team mostly uses Iphones; thus, we aim to understand which features help Android attract that many users in the marketplace, so we may consider switching our Iphones to Android-based ones.

C. Main Report

1. **Overview** Google's Android is an open-source operating system (OS) for mobile devices such as smartphones and tablets. It debuted in 2008 and is based on the Linux kernel. With nearly 2 billion active devices worldwide as of 2021, Android has rapidly emerged as one of the most popular mobile operating systems (Taylor, 2022) (1).

   Android development began in 2003 at Android Inc., a company created by Andy Rubin, Rich Miner, Nick Sears, and Chris White. They initially intended to develop a software platform for digital cameras, but the company later shifted its focus to developing a mobile

operating system. Google acquired Android Inc. in 2005, and the development of Android OS began in earnest. The T-Mobile G1 (also known as the HTC Dream) was the first Android-powered device released in 2008.

Android has gone through several crucial updates and versions throughout its existence. The first version of Android 1.0, was released in September 2008, and since then, Android has received several major updates, the most recent of which is Android 11, which will be released in September 2020. With each new version of Android, the operating system gains new features and improvements, making it more powerful and versatile. Among the key features introduced in recent Android versions have included a more robust and efficient multitasking system, enhanced security and privacy controls, and device performance and battery life improvements.

Android OS has a dominant market share of the mobile OS market, with a market share of more than 71 percent as of 2022 (Laricchia, 2023) (2). This dominance is due to the wide range of Android-powered devices, which include smartphones, tablets, and even smartwatches. Furthermore, the open-source nature of Android has resulted in a large and active community of developers, constructing a wealth of resources for developing apps and customizing the operating system. As a result, a massive ecosystem of apps and services has developed, which is one of the primary drivers of Android's popularity. It is also popular among users due to its integration with Google services such as Gmail, Google Maps, and the Google Play Store.

Another important feature of Android is that it allows for extensive customization via rooting, flashing custom ROMs, and other methods. Because of its adaptability, Android has become a popular choice among power users who seek to tailor the operating system to their specific requirements.

2. **Key components, architecture, and important features.**

   (a) Key components and architecture:

      i. **Linux Kernel:** The Linux kernel is the fundamental basis of Android OS, serving as an abstraction layer between both the hardware and the remainder of the software stack (Android, 2022) (3). It handles system resources including memory and processors and delivers a stable and secure environment for the operating system's other components to run in. The Linux kernel has the following features:
      - **Security:** The Linux kernel manages security between the application and the system
      - **Memory Management:** It effectively controls memory management, giving us the freedom to design our apps.
      - **Process Management:** It oversees the process well and distributes resources to processes as needed.
      - **Network Stack:** It closely monitors network communication.
      - **Driver Model:** It guarantees that the program runs smoothly on the device and hardware manufacturers are accountable for integrating their drivers into

the Linux build.

ii. **Platform libraries:** Android has a set of libraries that provide a variety of functions such as data storage, web browsing, and multimedia playback (Android, 2022) (3). These libraries, which are written in C and C++, are intended for use by the Android runtime and application framework, listed as below:
- **app:** Accesses the application model and serves as the foundation for all Android programs.
- **content:** Accesses, publishes, and communicates content among applications and application elements.
- **database:** SQLite database and administration classes are employed to extract data published by content providers.
- **OpenGL:** A Java interface to the 3D graphics rendering API OpenGL ES. os: Gives conventional operating system services to applications, such as messages, system services, and inter-process communication.
- **text:** Text that is rendered and manipulated on a device display.
- **view:** The basic building blocks of application user interfaces.
- **widget:** Widgets are pre-built user interface components such as buttons, labels, list views, layout managers, radio buttons, and so on.
- **WebKit:** A collection of classes designed to enable the incorporation of web-browsing features into applications.
- **media:** A media library allows you to play and record music and video formats.
- **surface manager:** The surface manager is in charge of controlling access to the display subsystem.
- **SQLite:** It supports databases, and FreeType supports fonts.
- **SSL:** is a security technique that creates an encrypted link between a web server and a web browser.

iii. **Application runtime:** The Android Runtime (ART) is in charge of executing applications on Android devices (Android, 2022) (3). It comprises a set of core libraries, such as the Dalvik Virtual Machine (DVM) and the Android Core Libraries, that serve as the fundamental building blocks for Android apps. Threading and low-level memory management are handled by the Linux kernel. The basic libraries allow us to build Android apps with the usual JAVA or Kotlin programming languages. It covers various sorts of services, like activity manager, notification manager, view system, package manager, and so on, that are useful for developing our application in accordance with the requirements.

iv. **Application Framework:** An application framework is a collection of services and APIs that developers can utilize to create apps (Android, 2022) (3). Views and layouts, resource management, and data storage are examples of these services. The Android UI toolkit, which offers a variety of UI components that developers can employ to construct the user interface of their apps, is also included in the Application Framework. In the format of Java classes, the Application Frame-

work layer equips various higher-level services to applications. These services may be used by software developers in their applications. The following major services are included in the Android framework:

- **Activity Manager:** Manages the entire application lifecycle and activity stack.
- **Content Providers:** Enables applications to bring out and exchange data with one another.
- **Resource Manager:** Entry to non-code embedded resources like strings, color settings, and user interface layouts is provided through the resource manager.
- **Notifications Manager:** Allows programs to show the user alerts and notifications.
- **View System:** A set of views that can be extended to construct application user interfaces.

v. **Applications:** The applications themselves are the final layer of the Android stack (Android, 2022) (3). Users interact with these apps, which include the phone dialer, web browser, and camera app. The services and APIs issued by the other layers of the stack are used to build these apps. It uses the classes and services supplied by the application framework to operate within the Android run time.

(b) Important features:

i. **Near Field Communication (NFC):** Android devices promote NFC, which enables electronic devices to communicate easily over small distances (Android, 2023) (4). This makes it possible for a more convenient payment option than carrying cash or credit cards, and it has great promise to be a future alternative to Bluetooth Low Energy (BLE).

ii. **Infrared Transmission:** The Android operating system contains an infrared transmitter, allowing you to use your phone or tablet as a remote control (JavaTpoint) (5). This feature allows you to control electronic devices like televisions, air conditioners, and other household appliances.

iii. **Automation:** The Tasker app on Android devices means allowing for app permission control and task automation (JavaTpoint) (5). This feature enables users to control their apps and automate repetitive tasks more easily.

iv. **Wireless App Downloads:** Android supports wireless app downloads via the Android Market or third-party services such as AppBrain (JavaTpoint) (5). Users can utilize this feature to download apps on their PC and then sync them to their Android devices without the need for a plug.

v. **Storage and Battery Swapping:** Android phones have the option to upgrade, replace, and discard a dead battery (Android, 2023) (4). Furthermore, Android phones include SD card ports for expandable storage. This feature gives users greater control over the storage and battery capacity of their devices.

vi. **Custom Home Screens:** Android authorises for home screen customization via third-party launchers such as Apex, Nova, and others (JavaTpoint) (5). Users

can use this feature to add gestures, new shortcuts, or even performance improvements to older-model devices.

vii. **Widgets:** With Android widgets, users can display any feature on their home screen, such as weather apps, music widgets, or productivity tools (Android, 2023) (4). This feature enables users to quickly access important information without opening an app and waiting for it to load.

Because the Android operating system is open source, developers can generate their own editions of the OS, defined as custom ROMs. These custom ROMs can add new features, alter the look and feel of a device, and improve performance. For advanced users who wish to adapt the operating system to their specific needs, this feature provides greater flexibility and customization.

3. **OS services**

(a) **General OS services:**

i. **System calls:** Android employs a series of system calls to allow applications to interact with the Linux kernel. These applications then can use system calls to execute activities like file I/O, process management, and inter-process communication. Besides, Android allocates a Java-based wrapper for Linux kernel system calls, allowing applications to utilize these functions without needing to understand the hidden Linux kernel intricacies.

ii. **Program execution:** Android manages the running of programs and applications by using the Linux kernel. The kernel is in charge of establishing, controlling, and scheduling processes for operation on the device's CPU. Android applications are controlled by the kernel's scheduler as Linux processes.

iii. **Memory management:** Android manages memory allocation and deallocation for applications and the system using the Linux kernel's memory management mechanism. The memory manager in the kernel is in charge of allocating and deallocating memory for programs, as well as handling the page cache, which serves to quicken file I/O operations. Furthermore, Android includes a memory management mechanism based on Java that enables applications to establish and govern their own memory banks.

iv. **Virtual memory:** Virtual memory is used by Android to permit programs to utilise extra memory than is readily available on the device. The operating system can generate more memory with virtual memory by utilising a piece of the device's storage like a "swap" space. When a program wants additional memory than is physically accessible, the kernel will switch out a portion of the application's memory to exchange space and then swap it back in when needed.

v. **Multitasking:** Android authorises many applications to occur simultaneously and manages the execution of numerous threads using the Linux kernel's scheduler. The scheduler is in charge of assigning CPU time to every running process and deciding which process should operate next. Moreover, Android includes a

threading mechanism based on Java that allows programs to build and handle their own threads.

vi. **Handling I/O operations:** Android handles input/output operations such as reading and writing to storage devices using the Linux kernel's I/O subsystem. The I/O subsystem is in charge of coordinating data flow across storage devices and programs, as well as the file system. Additionally, Android includes a Java-based I/O system that enables programs to conduct file I/O operations without needing to understand the underlying Linux kernel specifics.

vii. **Error detection and handling:** Android employs a wide range of options to identify and resolve errors, including exception handling and logging. Exception handling is applied to identify and manage issues within apps, whereas logging serves to monitor system events and errors. Besides, Android includes a Java-based error management system that enables applications to handle issues consistently.

viii. **Resource allocation:** The Linux kernel's process management system is employed by Android to control the allocation of resources including memory and CPU time to applications and the system. The process management system is in charge of assigning resources to every active process and deciding how much a process should obtain. Moreover, Android includes a resource allocation system based on Java that permits applications to administer their own resources.

Overall, these services are provided by the Linux kernel, which serves as Android's underlying operating system. These services are comparable to those provided by other operating systems, but may differ in implementation specifics, and are frequently wrapped by Java level APIs for simple access by Android apps.

(b) **Particular Android OS services:**

i. **Foreground services:** These are services that operate in the foreground of an Android app and alert the user of their current operations (Android, 2022) (6). They are commonly used for long-running processes that the user should be aware of, such as playing music, downloading data, or delivering navigation information. Foreground services are always active, which means they have a greater priority than background services and are displayed to the user via a notification. This enables the user to engage with the service by halting or restarting an ongoing task. Because foreground services are seen as more critical than background services, they are less likely to be terminated by the system when resources are scarce.

ii. **Background services:** Background services, in contrast, are services that operate in the background of an Android app without informing the user of their existence (Android, 2022) (6). They are often used for operations that do not require human participation, such as data synchronization on a schedule, data storage, as well as other background tasks. They do not display any notifica-

tions to the user, and the user cannot access them directly. Therefore, the user is unaware that a background service is operating. Background services are regarded as less critical than foreground services, and hence are more likely to be terminated by the system when resources are scarce.

iii. **Bound Services:** A bound service is one that continues to function as long as another component (such as an activity) is bound to it (Android, 2022) (6). The bound component can interact with the service and demand that it execute certain activities. Multiple components can bind to this kind of service at the same time. The bindService() method is used to associate an application component with a service; multiple components can be associated with the same service at the same time. These services are useful for tasks that must be completed in collaboration with other application components, as well as for data sharing between application components.

In summary, Forefront services are services that operate in the foreground of an Android app and alert the user to ongoing actions. Background services, on the other hand, are services that operate in the background of an Android app without informing the user of their existence. Furthermore, bound services are services that run as long as another component is bound to them, allow several components to bind to it at the same time, and are ideal for tasks that require coordination with other app components.

4. **Discuss advantages and disadvantages**

(a) **Advantages:** We evaluated all of the factors that make Android superior to other platforms. The following are some significant advantages of Android OS (JavaTpoint) (5):

i. **Google Developer:** One of the primary benefits of Android is that it is created by Google, which is one of the world's most reliable and trustworthy organizations. Using an Android device, people possess a sense of trust and dependability.

ii. **Android users:** Android has a big user base, with over a billion people using it. This makes it the world's most popular mobile operating system. It is also the fastest-growing operating system, which indicates that more and more people are adopting it. Because of the enormous user base, there is a diverse assortment of apps, games, and other material accessible for download on the Google Play Store.

iii. **Multitasking:** The ability to multitask is a crucial feature of Android. This means that users can open many programs at the same time and effortlessly transition between them, making it possible to complete multiple tasks at the same time. Multitasking is also made simple and intuitive by the user interface.

iv. **Access to the Google Play Store:** Android gives access to the world's largest mobile app store, the Google Play Store. It offers a vast range of apps, games, and other material for download, making it simple for customers to search and

install the apps they require.

v. **Simple Notification and Retrieve:** The Android operating system makes it simple to access alerts such as SMS and phone calls from the home screen or notification board. This allows users to view and reply to notifications without having to launch the appropriate app.

vi. **Widgets:** Android has a wide choice of widgets that may be placed on the home screen, allowing users to view information and conduct actions without having to open apps. This can save consumers time and make it easier for them to access the information they require.

(b) **Disadvantages:** We all understand that the Android operating system is very popular among users nowadays. However, it is very likely to have a few flaws. The following are some of the downsides of the Android operating system (JavaTpoint) (5):

i. **Advertisement Pop-Ups:** One of the major drawbacks of Android is that some programs on the Google Play Store may exhibit a large number of advertisements, which can be bothersome and disrupt the user experience. This is especially aggravating when the advertisement appears in the notification bar or throughout the app.

ii. **Gmail ID is required:** To access an Android smartphone, a user must have a Gmail ID and password. This can be a drawback for individuals who do not have or decide not to utilize a Gmail account.

iii. **Battery Drain:** Because of the numerous programs running in the background, the Android operating system can quickly deplete the battery. This can be an issue for users who depend on their cellphones for an extended period of time and do not have a charger near hand.

iv. **Malware/Virus/Security:** Because Android devices are less secure than other operating systems, they are more vulnerable to malware and viruses. Additionally, hackers may attempt to target Android devices in order to gain personal information. While security measures can be implemented to lessen these risks, consumers should be aware of the possibility of security vulnerabilities when running an Android smartphone.

5. **Implementation**

Android services are a special type of component that enables an application to execute tasks in the background. They are designed to keep the application running even when the user is using other apps or has closed the app. Services are not meant to have a user interface as they are meant to run operations without any user input. They can run continuously in the background, ensuring that the app remains active.

In this report, we will implement one of the common foreground services in android, which is playing music in the background. When a user initiates the service, the music will continue to play in the background even if the user switches to another app. In order to pause the music, the user must manually stop the service. The following is a detailed

Java implementation of this type of android service, including the use of various callback methods.

Since the service is created for Android devices, we will install and implement it on Android Studio. First, we design a simple layout in the activity_main XML file, which contains a heading and two buttons for starting and stopping the service:

```
<?xml version="1.0" encoding="utf-8"?>
<androidx.constraintlayout.widget.ConstraintLayout
    xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:app="http://schemas.android.com/apk/res-auto"
    xmlns:tools="http://schemas.android.com/tools"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:background="#168BC34A"
    tools:context=".MainActivity">

    <LinearLayout
        android:id="@+id/linearLayout"
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:layout_centerVertical="true"
        android:orientation="vertical"
        app:layout_constraintBottom_toBottomOf="parent"
        app:layout_constraintEnd_toEndOf="parent"
        app:layout_constraintStart_toStartOf="parent"
        app:layout_constraintTop_toTopOf="parent"
        app:layout_constraintVertical_bias="1.0"
        tools:ignore="MissingConstraints">

        <TextView
            android:id="@+id/textView1"
            android:layout_width="match_parent"
            android:layout_height="wrap_content"
            android:layout_marginBottom="170dp"
            android:fontFamily="sans-serif-black"
            android:text="@string/heading"
            android:textAlignment="center"
            android:textAppearance="@style/TextAppearance.AppCompat.Large"
            android:textColor="@android:color/holo_green_dark"
            android:textSize="36sp"
            android:textStyle="bold" />

        <Button
            android:id="@+id/startButton"
            android:layout_width="match_parent"
            android:layout_height="match_parent"
```

```
            android:layout_marginStart="20dp"
                        android:layout_marginTop="10dp"
            android:layout_marginEnd="20dp"
            android:layout_marginBottom="20dp"
            android:background="#4CAF50"
            android:fontFamily="sans-serif-black"
            android:text="@string/startButtonText"
            android:textAlignment="center"
            android:textAppearance="@style/TextAppearance.AppCompat.Display1"
            android:textColor="#FFFFFF"
            android:textStyle="bold" />

        <Button
            android:id="@+id/stopButton"
            android:layout_width="match_parent"
            android:layout_height="match_parent"
            android:layout_marginStart="20dp"
            android:layout_marginTop="10dp"
            android:layout_marginEnd="20dp"
            android:layout_marginBottom="20dp"
            android:background="#4CAF50"
            android:fontFamily="sans-serif-black"
            android:text="@string/stopButtonText"
            android:textAlignment="center"
            android:textAppearance="@style/TextAppearance.AppCompat.Display1"
            android:textColor="#FFFFFF"
            android:textStyle="bold" />

        <ImageView
            android:id="@+id/imageView"
            android:layout_width="match_parent"
            android:layout_height="wrap_content"
            android:layout_marginTop="80dp"
            />
    </LinearLayout>

</androidx.constraintlayout.widget.ConstraintLayout>
```

Next, we define a NewService class in the same directory as the MainActivity class, which extends the Service class. Also, we will use the MediaPlayer object to play music and callback methods to start and stop the service:

```
package com.example.myapplication;

import android.app.Service;
import android.content.Intent;
```

```
5  import android.media.MediaPlayer;
6  import android.os.IBinder;
7  import android.provider.Settings;
8  import android.widget.Toast;
9
10 import androidx.annotation.Nullable;
11
12 public class NewService extends Service {
13
14     // declaring object of MediaPlayer
15     private MediaPlayer player;
16
17     @Override
18
19     // execution of service will start
20     // on calling this method
21     public int onStartCommand(Intent intent, int flags, int ↩
           startId) {
22
23         // creating a media player which
24         // will play the audio of Default
25         // ringtone in android device
26         Toast.makeText(this, "Service started!", ↩
               Toast.LENGTH_SHORT).show();
27         player = MediaPlayer.create( this, R.raw.relax );
28
29         // providing the boolean
30         // value as true to play
31         // the audio on loop
32         player.setLooping( true );
33
34         // starting the process
35         player.start();
36
37         // returns the status
38         // of the program
39         return START_STICKY;
40     }
41
42     @Override
43
44     // execution of the service will
45     // stop on calling this method
46     public void onDestroy() {
47         Toast.makeText(this, "Service stopped!", ↩
               Toast.LENGTH_SHORT).show();
48         super.onDestroy();
```

```
49
50          // stopping the process
51          player.stop();
52      }
53
54      @Nullable
55      @Override
56      public IBinder onBind(Intent intent) {
57          return null;
58      }
59  }
```

Now, we start to create the buttons and functions for handling button click in the MainActivity class:

```
1   package com.example.myapplication;
2
3   import androidx.appcompat.app.AppCompatActivity;
4   import android.content.Intent;
5   import android.os.Bundle;
6   import android.view.View;
7   import android.widget.Button;
8
9   public class MainActivity extends AppCompatActivity implements ↩
        View.OnClickListener {
10
11      // declaring objects of Button class
12      private Button start, stop;
13
14      @Override
15      protected void onCreate(Bundle savedInstanceState) {
16          super.onCreate( savedInstanceState );
17          setContentView( R.layout.activity_main );
18
19          // assigning ID of startButton
20          // to the object start
21          start = (Button) findViewById( R.id.startButton );
22
23          // assigning ID of stopButton
24          // to the object stop
25          stop = (Button) findViewById( R.id.stopButton );
26
27          // declaring listeners for the
28          // buttons to make them respond
29          // correctly according to the process
30          start.setOnClickListener( this );
```

```
        stop.setOnClickListener( this );
    }

    public void onClick(View view) {

        // process to be performed
        // if start button is clicked
        if(view == start){

            // starting the service
            startService(new Intent( this, NewService.class ) );
        }

        // process to be performed
        // if stop button is clicked
        else if (view == stop){

            // stopping the service
            stopService(new Intent( this, NewService.class ) );

        }
    }
}
```

We modify the strings used in the strings.xml file as below:

```
<resources>
    <string name="app_name">My Application</string>
    <string name="heading">Services In Android</string>
    <string name="startButtonText">Start the Service</string>
    <string name="stopButtonText">Stop the Service</string>
</resources>
```

Finally, in order to effectively utilize services on an android device, it is important to include the service within the AndroidManifest.xml file.

```
<?xml version="1.0" encoding="utf-8"?>
<manifest ↩
    xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:tools="http://schemas.android.com/tools">

    <application
        android:allowBackup="true"
        android:dataExtractionRules="@xml/data_extraction_rules"
        android:fullBackupContent="@xml/backup_rules"
```

```
        android:icon="@mipmap/ic_launcher"
        android:label="@string/app_name"
        android:supportsRtl="true"
        android:theme="@style/Theme.MyApplication"
        tools:targetApi="31">
        <activity
            android:name=".MainActivity"
            android:exported="true">
            <intent-filter>
                <action android:name="android.intent.action.MAIN" />

                <category android:name="android.intent.category.LAUNCHER" />
            </intent-filter>
        </activity>

        <!-- Mention the service name here -->
        <service android:name=".NewService"/>

    </application>

</manifest>
```

We can run this Java implementation on an emulator virtual device, or connect it with an Android phone, and get the demo video of running the service which can be accessed via this Google drive link: Services in Android Demo

D. Reference List

1. Taylor, P. (2022, May 23). Internet users worldwide by OS 2012-2021. Statista. Retrieved January 18, 2023, from https://www.statista.com/statistics/543185/worldwide-internet-connected-operating-system-population/
2. Laricchia, F. (2023, January 17). Global Mobile OS Market Share 2022. Statista. Retrieved January 18, 2023, from https://www.statista.com/statistics/272698/global-market-share-held-by-mobile-operating-systems-since-2009/
3. Platform architecture: android developers. Android Developers. (n.d.). Retrieved January 18, 2023, from https://developer.android.com/guide/platform
4. Features and apis overview: android developers. Android Developers. (n.d.). Retrieved January 18, 2023, from https://developer.android.com/about/versions/12/features

5. Android operating system- javatpoint. www.javatpoint.com. (n.d.). Retrieved January 18, 2023, from https://www.javatpoint.com/android-operating-system

6. Services overview: android developers. Android Developers. (n.d.). Retrieved January 18, 2023, from https://developer.android.com/guide/components/services