FINAL REVIEW FOR MTE PROJECT ON

OPERATING SYSTEM

BACHELOR OF TECHNOLOGY



DEPARTMENT OF COMPUTER ENGINEERING

# DELHI TECHNOLOGICAL UNIVERSITY, DELHI, INDIA APRIL, 2021

## SUBMITTED BY: SUBMITTED TO:

NITIWIT KULDILOKE DR. Akshi Kumar

(2K19/CO/263) Delhi Technological University POOJA JAISAWAL Operating System

(2K19/CO/275)

## TITLE:

**ABSTRACT:**

### “TICKET BOOKING APPLICATION”

This project is made with objectives to helping people solve the problem of missing the bus or people who want to manage the time and booking ticket in advance and to implement the idea of operating system in order to improving the performance of application developing.

## OBJECTIVE:

To help people solving the problem of the bus ticket booking, we are developing the application which will help people booking the ticket at ease and making them to be feasible for everyone.

## INTRODUCTION:

We could see nowadays the process of getting the ticket of the bus is first come first serve. Thus, the problem has been happening and the people who has the busy time table of life is able to suffer. The solution which our group see and could provide is the application which can book the ticket in advance and see the travelling timetable of each and every bus. Hence, the people have the choice to choose and will be easier for them to manage the time for making the busy timetable to be the freely one.

The application which we are developing as the project of operating system is the kind of demo of what our project could be in the future and can be developed further as the time matter. In the future the cooperation of bus and information of everything of bus timetable might be provided.

## THEORY:

Initially, we should select the field of study and theory which will be included in the project first. Starting at understanding the where and what are we doing?

#### OPERATING SYSTEM

As we developing the application through android studio, the operating system in use is clearly the android one. As the main target of using the application is the lower-bound user thus the suitable field of target is obviously android mobile as the price is cheaper and easier in term of using it.

Android is a mobile operating system based on a modified version of the Linux kernel and other open-source software, designed primarily for touchscreen mobile devices such as smartphones and tablets. Android is developed by a consortium of developers known as the Open Handset Alliance and commercially sponsored by Google.

It was unveiled in November 2007, with the first commercial Android device launched in September 2008. It is free and open source software; its source code is known as Android Open Source Project (AOSP), which is primarily licensed under the Apache License.

However most Android devices ship with additional proprietary software pre- installed, most notably Google Mobile Services (GMS)which includes core apps such as Google Chrome, the digital distribution platform Google Play and associated Google Play Services development platform. About 70 percent of Android smartphones run Google's ecosystem; competing Android ecosystems and forks include Fire OS (developed by Amazon) or Lineage OS.

As the android is one of kernel which provides many services related to I/O. Several services such as scheduling, caching, spooling, device reservation, and error handling – are provided by the kernel, s I/O subsystem built on the hardware and device-driver infrastructure. The I/O subsystem is also responsible for protecting itself from errant processes and malicious users.

#### I/O Scheduling –

To schedule a set of I/O requests means to determine a good order in which to execute them. The order in which the application issues the system call is the best choice. Scheduling can improve the overall performance of the system, can share device access permission fairly to all the processes, reduce the average waiting time, response time, turnaround time for I/O to complete.

OS developers implement schedules by maintaining a wait queue of the request for each device. When an application issues a blocking I/O system call, the request is placed in the queue for that device.

The I/O scheduler rearranges the order to improve the efficiency of the system.

#### Buffering –

A *buffer* is a memory area that stores data being transferred between two devices or between a device and an application. Buffering is done for three reasons.

* 1. The first is to cope with a speed mismatch between producer and consumer of a data stream.
  2. The second use of buffering is to provide adaptation for data that have different data-transfer sizes.
  3. The third use of buffering is to support copy semantics for the application I/O, “copy semantic” means, suppose that an application wants to write data on a disk that is stored in its buffer. it calls the **write ()** system’s call, providing a pointer to the buffer and the integer specifying the number of bytes to write.

#### Caching –

A cache could be a region of quick memory that holds a duplicate of information. Access to the cached copy is way easier than the initial

file. for example, the instruction of the presently running method is keep on the disk, cached in physical memory, and traced once more within the CPU’s secondary and first cache. The main distinction between a buffer and a cache is that a buffer could hold solely the present copy of a knowledge item, whereas a cache, by definition, holds a duplicate on quicker storage of associate item that

resides elsewhere.

#### Spooling and Device Reservation

A spool could be a buffer that holds the output of a tool, like a printer that can't settle for interleaved information streams. though a printer will serve just one job at a time, many applications might need to print their output at the same time, while not having their output mixes along. The OS solves this drawback by preventing all output from continued to the printer. The output of all applications is spooled during a separate computer file. once Associate in Nursing application finishes printing then the spooling system queues the corresponding spool file for output to the printer.

#### Error Handling

An Operating system that uses protected memory will guard against several styles of hardware and application errors so an entire system failure isn't the same old results of every minor mechanical bug, Devices, and I/O transfers will fail in some ways, either for transient reasons, as once a network becomes overladen or for permanent reasons, as once a control becomes defective.

#### I/O Protection

Errors and the issue of protection are closely related. A user process may attempt to issue illegal I/O instructions to disrupt the normal function of a system. We can use the various mechanisms to ensure that such disruption cannot take place in the system.

To prevent illegal, I/O access, we define all I/O instructions to be privileged instructions. The user cannot issue I/O instruction directly.

The IDE we are using is Androids Studio thus the android studio is what the theory we are studying about

Android Studio is the official integrated development environment (IDE) for Google's Android operating system, built on JetBrains' IntelliJ IDEA software and designed specifically for Android development. It is available for download on Windows, macOS and Linux based operating systems or as a subscription- based service in 2020. It is a replacement for the Eclipse Android Development Tools (E-ADT) as the primary IDE for native Android application development. Android Studio was announced on May 16, 2013 at the Google I/O conference. It was in early access preview stage starting from version 0.1 in May 2013, then entered beta stage starting from version 0.8 which was released in June 2014. The first stable build was released in December 2014, starting from version 1.0. On May 7, 2019, Kotlin replaced Java as Google's preferred language for Android app development. Java is still supported, as is C++.

Operating System included

Shared libraries square measure helpful in sharing code that is common across several applications. As an example, it's additional economic to pack all the code associated with TCP/IP implementation in an exceedingly shared library. However, knowledge can’t be shared as each application desires its own set of knowledge.

Applications like, browser, ftp, telnet, etc… build use of the shared ‘network’ library to elevate specific practicality. Every package has its own illustration and tool-set to form shared libraries. Additional or less the ideas square measure same. On Windows each object file (\*.obj, \*.dll, \*.OCX, \*.sys, \*.exe etc…) follow a format known as Portable practicable. Even shared libraries (called as Dynamic connected Libraries or DLL in short) also are painted in letter of the alphabet format. The tool-set that's accustomed produce these libraries got to perceive the binary format. Linux variants follow a format known as practicable and Linkable Format (ELF).

The ELF files square measure position freelance (PIC) format. Shared libraries in Linux square measure referred as shared objects (generally with extension \*.so). These square measure like DLL in Windows platform. Even shared object files follow the ELF binary format. Remember, the file extensions (\*.dll, \*.so, \*.a,

\*.lib, etc…) square measure only for engineer convenience. They don’t have any significance. Of these square measure binary files.

You'll be able to name them as you want. Nevertheless, make sure you offer absolute ways in building applications. In general, once we compile associate

application the steps square measure easy. Compile, Link and cargo. However, it's not easy. These steps square measure additional versatile on fashionable operative systems.

Due to the verification part of the application is needed to verify the identity of user thus we also include Cloud Firestone as the authentication and NoSQL cloud database

Cloud Firestore is a flexible, scalable database for mobile, web, and server development from Firebase and Google Cloud. Like Firebase Realtime Database, it keeps your data in sync across client apps through realtime listeners and offers offline support for mobile and web so you can build responsive apps that work regardless of network latency or Internet connectivity. Cloud Firestore also offers seamless integration with other Firebase and Google Cloud products, including Cloud Functions.

Android File Allocation

In computer operating systems, memory paging is a memory management scheme by which a computer stores and retrieves data from secondary storage[a] for use in main memory. In this scheme, the operating system retrieves data from secondary storage in same-size blocks called pages. Paging is an important part of virtual memory implementations in modern operating systems, using secondary storage to let programs exceed the size of available physical memory. For simplicity, main memory is called "RAM" (an acronym of "random-access memory") and secondary storage is called "disk" (a shorthand for "hard disk drive, drum memory or solid-state drive"), but the concepts do not depend on whether these terms apply literally to a specific computer system.

Random-access memory (RAM) is a valuable resource in any software development environment, but it's even more valuable on a mobile operating system where physical memory is often constrained. Although both the Android Runtime (ART) and Dalvik virtual machine perform routine garbage collection, this does not mean you can ignore when and where your app allocates and releases memory. You still need to avoid introducing memory leaks, usually caused by holding onto object references in static member variables, and release any Reference objects at the appropriate time as defined by lifecycle callbacks.

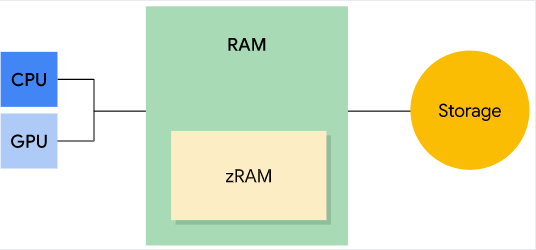
The Android platform runs on the premise that free memory is wasted memory. It tries to use all of the available memory at all times. For example, the system keeps apps in memory after they've been closed so the user can quickly switch back to them. For this reason, Android devices often run with very little free memory.

Memory management is vital to properly allocate memory among important system processes and many user applications.

This page discusses the basics of how Android allocates memory for the system and for user applications. It also explains how the operating system reacts to low memory situations.

# Types of memory

Android devices contain three different types of memory: RAM, zRAM, and storage. Note that both the CPU and GPU access the same RAM.



**Figure 1.** Types of memory - RAM, zRAM, and storage

* RAM is the fastest type of memory, but is usually limited in size. High-end devices typically have the largest amounts of RAM.
* zRAM is a partition of RAM used for swap space. Everything is compressed when placed into zRAM, and then decompressed when copied out of zRAM. This portion of RAM grows or shrinks in size as pages are moved into or taken out of zRAM. Device manufacturers can set the maximum size.
* Storage contains all of the persistent data such as the file system and the included object code for all apps, libraries, and the platform. Storage has much more capacity than the other two types of memory. On Android, storage isn’t used for swap space like it is on other Linux implementations since frequent writing can cause wear on this memory, and shorten the life of the storage medium.

# Memory pages

RAM is broken up into *pages*. Typically each page is 4KB of memory.

Pages are considered either *free* or *used*. Free pages are unused RAM. Used pages are RAM that the system is actively using, and are grouped into the following categories:

* Cached: Memory backed by a file on storage (for example, code or memory- mapped files). There are two types of cached memory:
  + Private: Owned by one process and not shared
    - Clean: Unmodified copy of a file on storage, can be deleted by [kswapd](https://developer.android.com/topic/performance/memory-management#kswapd) to increase free memory
    - Dirty: Modified copy of the file on storage; can be moved to, or compressed in, zRAM by kswapd to increase free memory
  + Shared: Used by multiple processes
    - Clean: Unmodified copy of the file on storage, can be deleted by kswapd to increase free memory
    - Dirty: Modified copy of the file on storage; allows changes to be written back to the file in storage to increase free memory

by kswapd, or explicitly using [msync()](https://developer.android.com/reference/android/system/Os#msync(long%2C%2520long%2C%2520int)) or [munmap()](https://developer.android.com/reference/android/system/Os#munmap(long%2C%2520long))

* Anonymous: Memory **not** backed by a file on storage (for example, allocated by mmap() with the MAP\_ANONYMOUS flag set)
  + Dirty: Can be moved/compressed in zRAM by kswapd to increase free memory

# Garbage collection

A managed memory environment, like the ART or Dalvik virtual machine, keeps track of each memory allocation. Once it determines that a piece of memory is no longer being used by the program, it frees it back to the heap, without any intervention from the programmer. The mechanism for reclaiming unused memory within a managed memory environment is known as *garbage collection*. Garbage collection has two goals: find data objects in a program that cannot be accessed in the future; and reclaim the resources used by those objects.

Android’s memory heap is a generational one, meaning that there are different buckets of allocations that it tracks, based on the expected life and size of an object being allocated. For example, recently allocated objects belong in

the *Young generation*. When an object stays active long enough, it can be promoted to an older generation, followed by a permanent generation.

Each heap generation has its own dedicated upper limit on the amount of memory that objects there can occupy. Any time a generation starts to fill up, the system executes a garbage collection event in an attempt to free up memory. The duration of the garbage collection depends on which generation of objects it's collecting and how many active objects are in each generation.

Even though garbage collection can be quite fast, it can still affect your app's performance. You don’t generally control when a garbage collection event occurs from within your code. The system has a running set of criteria for

determining when to perform garbage collection. When the criteria are satisfied, the system stops executing the process and begins garbage collection. If garbage collection occurs in the middle of an intensive processing loop like an animation or during music playback, it can increase processing time. This increase can potentially push code execution in your app past the recommended 16ms threshold for efficient and smooth frame rendering.

Additionally, your code flow may perform kinds of work that force garbage collection events to occur more often or make them last longer-than-normal. For example, if you allocate multiple objects in the innermost part of a for-loop during each frame of an alpha blending animation, you might pollute your memory heap with a lot of objects. In that circumstance, the garbage collector executes multiple garbage collection events and can degrade the performance of your app.

#### Low memory management

Android has two main mechanisms to deal with low memory situations: the kernel swap daemon and low-memory killer.

kernel swap daemon: The kernel swap daemon (kswapd) is part of the Linux kernel, and converts used memory into free memory. The daemon becomes active when free memory on the device runs low. The Linux kernel maintains low and high free memory thresholds. When free memory falls below the low threshold, kswapd starts to reclaim memory. Once the free memory reaches the high threshold, kswapd stops reclaiming memory.

kswapd can reclaim clean pages by deleting them because they're backed by storage and have not been modified. If a process tries to address a clean page that

has been deleted, the system copies the page from storage to RAM. This operation is known as *demand paging*.

**WORKING**

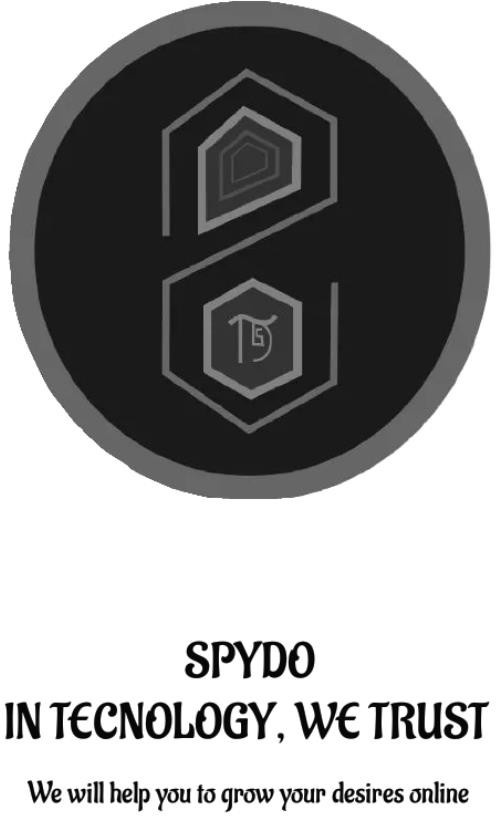
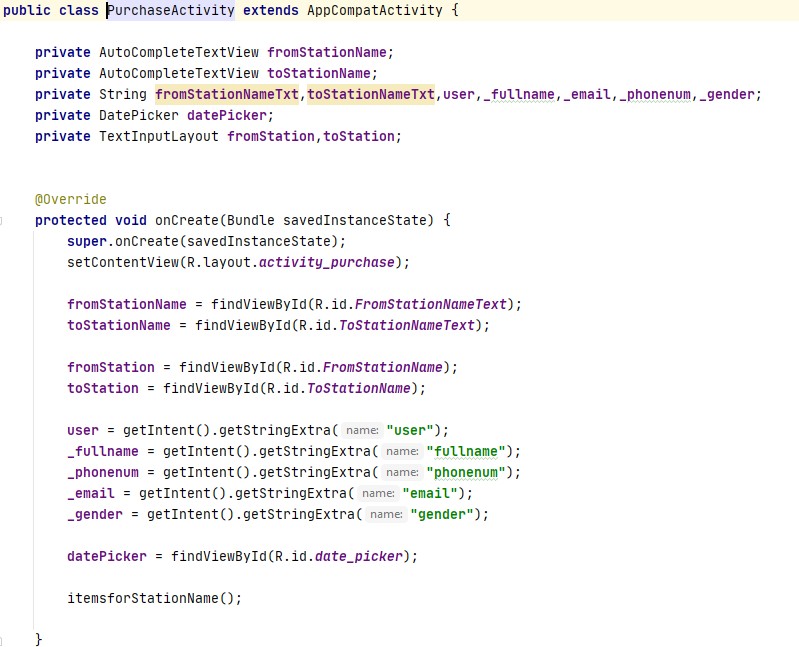


Figure: Interface of our application

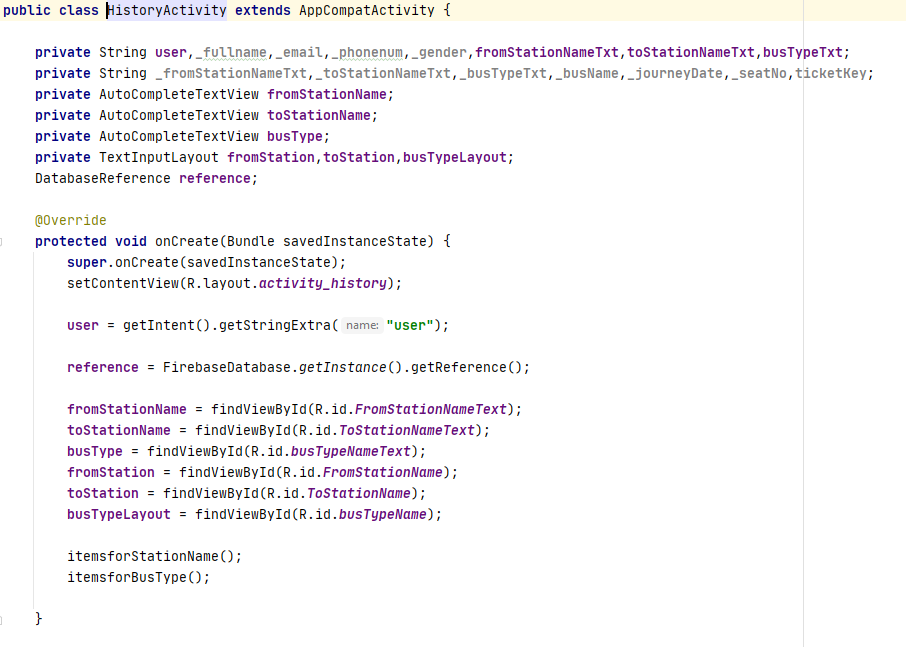
The process of our work will be divided into two part the back-end and the front end by this report we will provide only the back-end part



This class will be work as the cross platform as the authentication is required the shared libraries cloud firebase of Google this will get the input from the user then compile the data and store it in the database then send the verification to the phone number which will be put.



This class will show the Purchase or booking part of the app.



This class will show the history of ticket booking.

As the work have been shown in the back-end the front-end will be shown in the presentation.

**CONCLUSION**

Simplicity is never simple. As we have seen in this project, the process of creating a user friendly and straightforward platform that user can be at ease when using it. The step before working and developing it is hard when the part of understanding is needed to be done but after that implementation of the idea is not that kind of work to be considered as hard part. The understanding of operating system in this project is focusing in the area of file allocation of android operating system and share memory like the Firestone Cloud.

After finishing it, implementation part needs the understanding of the Java Languages too. As the app development is used Java. The Java language need to understand in term of developing.

And as we expected, the application is very ease to use as the reviewer from some group of people who is in need of it giving review to us. Thus, we could say this project is successfully as we expected. The system can still be improving in the future too as the project can get the data of all bus line and time table can be provided also then we can hope for improvement in the future.

**REFERENCE:**

* “Overview of memory management” in the name of developer.android. 2021-02-24 UTC.

[Online]: <https://developer.android.com/topic/performance/memory-overview>

* Tanya Sethi “Resource Allocation Techniques for Processes” in the name of GEEKFORGEEK 14 Aug, 2019 [Online]: [https://www.geeksforgeeks.org/resource-allocation-techniques- for-processes/](https://www.geeksforgeeks.org/resource-allocation-techniques-for-processes/)
* Akash1295” Microkernel in Operating Systems” in the name of GEEKFORGEEK 28 Oct, 2020[Online]: <https://www.geeksforgeeks.org/microkernel-in-operating-systems/>