# NO FRILLS REMINDER CABS SERVICE

PROJECT REPORT

CS634A (MOBILE COMPUTING)

GUIDE: PROF. R. K. GHOSH

GROUP MEMBER:

MOHD SHAHID KHAN AFRIDI (160407) AND HITESH KUMAR RAWAT(160298)

## Abstract

The main purpose of our project is to learn real life application of mobile computing. We build an android application, the main features of our project is to ease the system of locating and booking cabs online and alongside providing a reminder for customers task to do on the way.

It gives us a hand on experience for variety of advanced technical tools like gRPC server, Google map APIs, Http/2 and database like mongodb.

Flow of the document is : 1. Motivation, 2. Interface and working of application, 3.Technical working: APIs, languages and protocols used , 4. Drawbacks / Sources of improvement.

NO FRILLS REMINDER CABS SERVICE

## MOTIVATION:

The growing need and demand of easy/online cab service is rising rapidly among developing nations like india. Many private companies are gaining the market at a very high rate, but it's seen that soon  after the initial bait of low pricing company started charging more and more with every coming days, misusing the growing habits of customers.

It's important to give these company a good competition in the market, so as to put their prices in control. Our project is mainly focused on this service to society with good features. We are trying to remove these private company as intermediary and let the driver and customer to work under set rules and regulation but no unnecessary additional charges.

## Interfaces and working

***Customer Side*****:** Any customer will be able to download app from cloud, install it in his mobile and open it. Customer **Interface** will provide customer to create its account, mobile phone number will be used to authenticate by sending OTP via sms. Once created account customer is created he can log in to his account and without any delay can avail himself with the cab service. Customer can view map and all the available cabs along with drivers details will be shown to him, customer can select any cab and use the direct dial feature of the app to quickly add call to driver, customer can even bargain with driver accordingly, setting up with an agreed price driver will come to pick up and ride will begin. But it's not only that, customer can set task, he is wishing to do on his way, to tasks list. The **task list** allow customer to add task along with **task's location and due time**, he will be  notified with alarm once he is in the vicinity of that particular location or get passed the due time.

***Driver Side:*** Any driver will be able to download app from cloud, install it in his mobile and open it. Driver **interface** will facilitate driver to create his account filling all his personal details as well as cars detail. Once created a Driver can log in into his account and make himself available/unavailable for any upcoming ride. The Drivers will also be able to locate nearby cabs so as to avoid congestion at one place. Making oneself available driver can sit back wait for a call from a customer and after agreeing upon prices can go for taking the ride.

## Technical working, APIs and Protocols

***Modules and languages**:* Main language used to build different modules are as : **kotlin** being used mainly for the backend and for also for android. Combination of **Java and kotlin** is used for android. **Xml** is used for style and views. **Protobuf** is used for data processing to binary.

***Deployment**:* Server is deployed on google kubernetes engine running on single compute engine instance. The server consists of three docker containers which are described as follow:

***Docker Image 1:*** This docker image consists of the file system configuration that mongodb uses for its storage. This is a statefulset i.e. it maintains it state across multiple deployments.

***Docker Image 2:*** This docker image is the official mongodb docker image that runs inside Google's container optimized OS. The database is exposed via localhost and port 27017.

***Docker Image 3:*** This image is the actual gRPC server that exposes the APIs via data access object model.The API is exposed via an external load balancer which has a public IP.

***Database:*** mongodb is used as a database engine store all the data, the data update will be done when written by the customer or driver and the location of any available cab is pushed every 5 second. Geospatial indexing is done on position/location cabs and tasks.

## Drawbacks / Source Of Improvement

(Mostly machine learning techniques)

**Route prediction :**

API can be setup for showing route from source to destination.

**Time prediction for task:**

Instead of notifying only at the due time we can make an time approximating system, which will ping notification when the the remaining time for due time of a task start decreasing below then expected time of arrival at that location.

**Cost upper bound:**

An upper bound for charges on any route will be suggested, driver can't ask more than that, also giving customer an expected amount.

**Authentication For Images:**

Images being uploaded by driver or customer can be verified to be a non violent and authentic.

## References and sources

Open Source dependencies like android framework, google play services, gRPC , circle image view

Google open source sample for running Location service especially in background or application killed scenario.

Firebase storage, Glide, Image compressor, joda and bugsnag