*I have always been interested in **Application Stacks** and I have always admired one particular organisation: **Uber** and the way they:*

- *Handle more than a million transactions a day*
- *Estimate the time and price for a ride from source to destination*
- *Have optimised open source software*

Requirement Code R1

Research and **describe** the technologies used to deliver services or products of your chosen organisation in terms of:
- presentation layer (front-end frameworks, native mobile or desktop applications, PWAs)
- business/application layer (web servers, programming languages and technologies for implementing and supporting business logic and processing of business data)
- data access layer (databases, technologies and services which allow processing and querying of data)

The following technologies are used by Uber to deliver services. A sub flow of food ordering service is inserted in the description below.

Presentation Layer

Swift - Uber has a developed a cross platform mobile architecture called RIB. RIB stands for Route, Interactor and Builder. The architecture includes code generation module in Swift for iOS mobile devices. This is the Uber App for drivers, riders or other customers on iOS to order taxi or food.

Go - Uber RIB architecture also generates code in Go for Android based mobile devices. RIB is open sourced, but the code generation tools and templates are not open sourced. Customer opens the opens the UberEats app on Android phone.

Business/Application Layer

HAProxy- Routes the request to service that processes customer's request. When a customer order through UberEats, it sends to REST-API on Nginx.

Nginx – Web Server used by Uber to provide services using REST-API. One of the REST-API gets customer's address and customer's details and creates a workflow to process the request.

Cadence – Uber's Cadence is an opensource workflow engine to execute multiple asynchronous processes to offer a service. Workflow has a series of steps that are performed in parallel or in sequence to complete a task. A workflow can call a sub workflow to complete a complex step/task. All the procedures that workflow calls are written in Java and Python. When the customer is looking at ordering the food, the sequential steps are getting all the restaurants that can be reached in 15 minutes around customers location, then three tasks can be done in parallel i.e. get all past orders of customer, calculate delivery charges for each restaurant including free delivery (sub workflow) and calculate discounts offered by restaurant. All this data is output from the workflow and sent back to customer to complete order.

Kafka – Uber uses Kafka to process data stream from MySQL database and Uber Key-Val DB. Key-Val data is data coming from logs and web-requests extracted, transformed and loaded (ETL). This data is used by Ingestion Service to be send to storage and analytics. All the activities that the customer ordering the food did on the UberEats app is streamed to Kafka like customer changed the address.

Data Access Layer

MySQL – Uber mainly uses MySQL as their main database. Uber has made the database access layer schema less. Uber Schema less only writes or reads from the database. All the data is time lined and the schema less can query the data using the latest timeline. Some data stored in MySQL database are customer's invoice, payment methods, billing address, etc.

Cassandra – Uber stores all the data streamed and processed using Kafka in Cassandra NoSQL big data storage. Cassandra can provide analytics data for business to improve service. It can help in making decisions like discounting to improve sales. Uber's **Kepler.GL** uses this data to visualise geospatial trends.

Hudi – Hudi is the view on Big Data like a database view for the query engine Presto to use. It can also ingest data into big data system. All the data Kafka produced and transformed is passed through Hudi to be stored in distributed file system. Hudi will be able to provide a food categories view for the customer exactly like a real-time category table with just Italian and Chinese meals.

Hadoop – Hadoop is the distributed file system that Uber uses for Big Data. The processed data produced by data stream like parts for what customer did and considered important for providing service is stored using Hadoop into distributed file systems. Hadoop uses Map/Reduce software framework for big data processing.

Presto – Presto is an SQL query engine. All the past orders and other data is queried using presto.

Reference: https://eng.uber.com/

Requirement Code R2

Identify ONE application stack utilised by an organisation to deliver a product or service.
Describe the role of each technology used within an application stack and how it relates to other applications/technologies in a stack

Uber's uses globalised data with localised view of providing services and profit maximisation. The main technologies that make this happen is Kafka, Cassandra, Presto and Kepler.GL. Safety is a unique selling point of Uber. Uber's data processing technologies helps in keeping drivers and passengers safe.

Kafka – Uber collects a lot of data from different sources. It is both raw and processed data. Raw data is data from uber.com and mobile apps. Raw data includes data from both customers and others.
A subset of data directly collected are –

- Someone opening Uber App in a location not serviced by Uber
- Someone visiting the Uber website in a location that is not serviced by Uber
- Someone clicks the complaints section on the website

Processed data is the data that is stored in MySQL and Key.Val databases, that are processed by micro services.
A subset of processed data is –

- Trips ending in CBD's
- Trips originating in CBD's
- Time of travel

All this data is streamed into Kafka. Kafka uses configuration and Java code to process data to produce messages for the ingestion services to consume. Ingestion services use configuration and logic to store this data in Cassandra.

Cassandra – Cassandra is a big data storage used by Uber for mostly business analytics and Machine Learning. Uber has enough data to get thousands of features to help with business decisions and machine learning models to improve services. Uber's Michelangelo machine learning platform extracts these features from the database and makes it available to ML models that can predict meal delivery times or delivery costs.

Presto – Uber uses Presto for SQL queries on the big data platform. Presto is optimised to make data available near real time for business analytics and business critical operations safely and reliably. Presto's versatility gives Uber an edge over its competition.

Kepler.GL – Kelpler.GL is built on top of deck.GL Web GL data visualisation framework. It is one of the most powerful geospatial data visualisation tools. Uber is a global business run from San Francisco. Decision makers need near real-time data visualisation from around the world to make quick decisions when things are not going as expected. For example, there is hardly any Uber trips in progress right now or in the last few hours in a region of Sydney, the

dashboard used by Operational Analyst on the other side of the world will pick that up, investigate and escalate for the management to shut down all services in the bush fire region. Kepler can process large data sets in terabyte for data visualisation. Kepler's powerful visualisation on the surface of earth can highlight issues so well with animation that it can be missed.
It can also be used for business expansion and improvement. If there are places where people are searching for Uber, Uber can start services there. If there is a lot of demand for Uber in a City or the wait times are higher, Uber can encourage more drivers to join Uber.

Uber's technology stack helps in real-time data driven decision making.

Reference: https://eng.uber.com/

Requirement Code R3

Research the hardware/cloud platform(s) that are utilised by the organisation and **describe** what hardware is required.

Uber uses a combination of data centres and cloud platforms to fulfil its business needs. Data centres leased by Uber are located around the world including Australia. Uber also uses Cloud GPUs from Google cloud for maps and Microsoft Azure for Facial recognition.

There is a total of 576 racks used for data storage and computing in data centres.

- 32 racks for network
- 64 racks for unknown unknowns
- 480 racks remaining for data storage and computing (30 pods of 16 racks each)

Each rack is 42 U with 3 U enclosures. With the optimised power consumption each enclosure can hold 2U rack server.

Each rack uses a combination of servers from different manufacturers to support and store million taxi ride and delivery a day.

Older Dell Servers

Dell PowerEdge 11G R510 Rack Server Intel Xeon Processor 5600 Series
W347K hard disks (hot swap SAS 6Gbps, 600GB, 15K rpm f/w ES64)

Dell PowerEdge R630 Twenty-Core Intel Xeon Processors. 3C1JP Dell R630 Twenty-Core Intel Xeon E5-2698 v4 (2.20GHz) Processor

Newer Dell Servers

Dell R630XD Server 2xE5-2640-V3 2.60GHz 384GB 2X200GB SSD 8X1.2TB SAS 10K PERC H730 RAID
2 X Intel Xeon 10 Core E5-2640 V3, 2 Physical 10 Core Processors 2.60Ghz 20 Cores, 384Gb RAM DDR3 PC10600R ECC, 2X200GB SSD 8X1.2TB SAS 10K 2.5" Hard Drives, DELL PERC H730 / NVRAM RAID Controller Installed, 4 X Gigabit Network Ports 10/100/1000, 2 X Redundant Dual Power Supplies, 2U Rackmount Server

The servers have up to 12 hard drive slots for storage and up to 2 slots for OS, Hypervisor and VM's.

Compatible hard drives like the below are used in the slots.

Seagate SkyHawk AI ST14000VE0008 - Hard drive - 14 TB - internal - 3.5" - SATA 6Gb/s - buffer: 256 MB

Lenova RAC servers are also used in Data Centres where VMWare is not required. VMWare works better with Dell. VMWare is partly owned by Dell EMC.

Lenovo **ThinkSystem SR650** 7X06A057NA 2U Rack Server - 1 x Intel Xeon Silver 4110 Octa-core [8 Core] 2.10 GHz - 16 GB Installed DD

References:

https://www.sec.gov/Archives/edgar/data/1543151/000119312519103850/d647752ds1.htm
https://www.datacenterdynamics.com/analysis/dcdlondon-dean-nelson-details-uber-metal-house-data-centers/

Requirement Code R4

Research the data model of an organisation by looking at their API documentation and any other sources of publicly available information and **describe** the organisational functions that are possible based on the API.

Uber provides developer API's. There are three main API's available and based on that, the following functions are possible.

1. Rider API
   can create applications to order rides on customers behalf, get the list of products and details of each product for a customer to select, get price estimates, get time estimates, get user information, get tracking maps, order and cancel rides

2. Driver API
   create and manage a profile of drivers on their behalf, manage income, manage trips

3. Business API
   manage and upload employees, download transactional data, manage and upload expenses

The following entities help to fulfil organisational functions.

1. **rider**
   Rider is the entity that stores the required personal details of the rider. Uber API can retrieve, and change data stored in this entity. All the entities as created timestamp, as Uber databases only have read and write. As there is no data updated or deleted, created timestamp helps to extract the most recent data.

2. **driver**
   Driver entity stores the required information about the driver and their area of service. Uber API can retrieve and modify stored in this entity. This entity also stores data about their location and the vehicle the driver uses for the service.

3. **vehicle**
   Vehicle entity stores the details of the vehicle used for the service. The accuracy of the data in this entity is important, as this is the information a rider uses to identify the vehicle and start the trip on time.

4. **journey**
   Journey entity has all the information regarding the ride. It has a driver, rider, vehicle, pickup location. There are multiple updates made during the single trip.

5. **product**
   Product entity contains the different vehicles offered for the trip and the pricing associated with product.

6. **price**
   Price has the total cost of the journey and the sub items including discounts.

7. **location**
   Location has latitude and longitude of the location used for picking and dropping of the rider.

8. **product_location**
   Product location is an entity to store the products offered in a location. In database it is a junction table.

9. **localised_business_details**
   Localised business details are information that is used to be included in the customers invoice. Most countries require this information on invoice.

10. **rider_payment**
    Rider payment is the entity that stores rider's payments details. At the end of the journey the rider is charged.

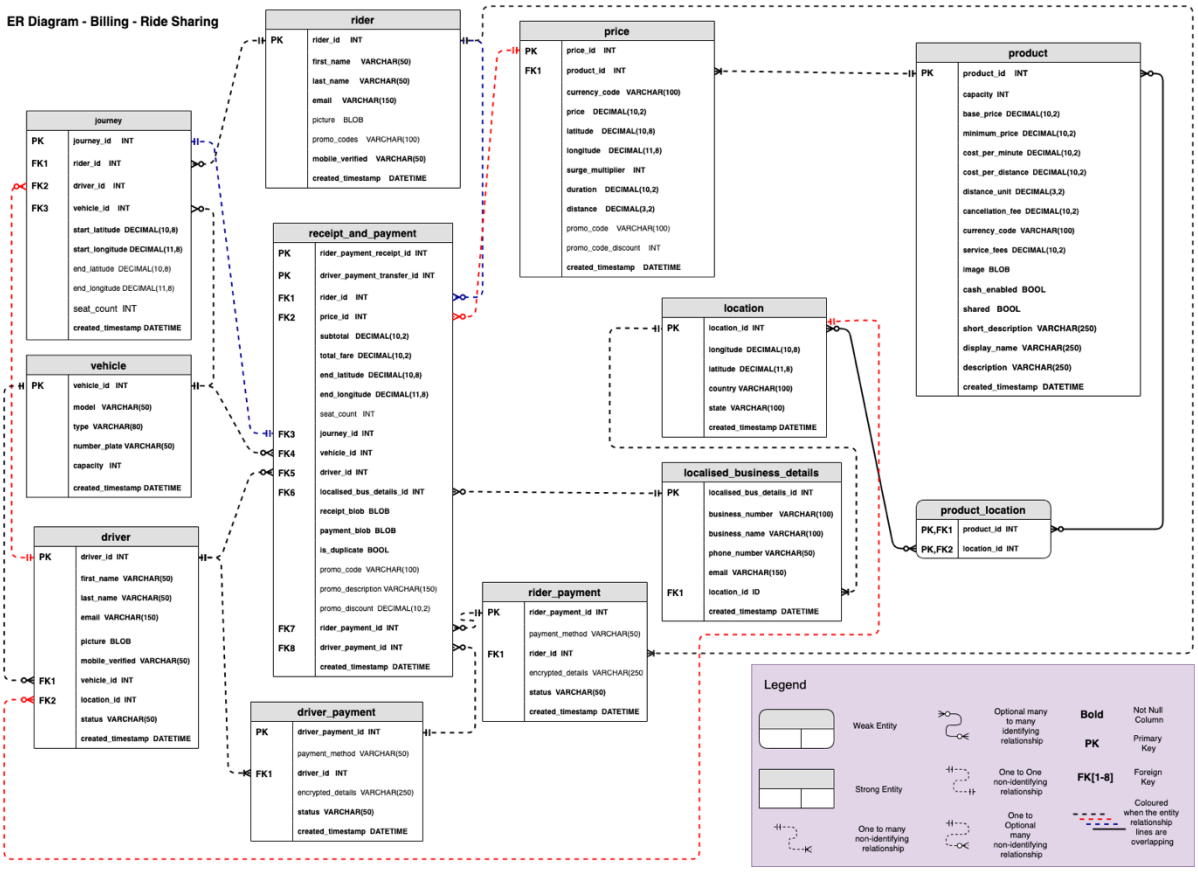
11. **driver_payment**
    Driver payment is the entity used to store the driver's account details. At the end of the trip, drivers share of income is transferred to the driver.

12. **receipt_and_payment**
    Receipt and Payment entity has the details of invoice generated for a trip and the payment made to the driver for the trip. It also contains the actual invoice.

Reference: https://developer.uber.com

Requirement Code R5

Create an entity relationship diagram which represents entities used in a product or service (or part of) and the relationships between them.
You are free to speculate on the properties of entities and the relationships between entities.



Requirement Code R6

Describe TWO processes for the input and output of data based on the company's API and how they achieve organisational objectives.

/estimate/price is Uber's API to provide estimates to a rider to travel from current location to a destination within Uber's limit.

The input data to estimate API is

start_latitude
start_longitude
end_latitude
end_longitude

The output data from the estimate API is

localized_display_name
distance
display_name
product_id
high_estimate
low_estimate
duration
estimate
currency_code

The processes for the input and output of data are processing the estimate using Google Maps or processing the estimate using Uber App on Android.

Processing input output using Google Maps

To get price estimates on Google Maps, the input required is a destination. Google can get the current location from the device and convert it to latitude and longitude. When Directions option is selected, Google Maps gives multiple options – walking, driving, public transport, cycling, ride with the estimated times for each. When ride option is selected, Google Maps gives price range estimates of different Uber vehicles. It displays the lower price estimate and the higher price estimate.

Achieving Organisational Objectives

- Rider can compare Uber's price and time estimates to destination with other modes of transport and other Ride sharing providers. It can encourage rider

to choose Uber when these estimates are compared with the convenience of door to door.

- For Uber brand, it is free marketing, advertising and more business. Google becomes a promoter, reseller or an agent in this case.
- API use by other apps help in the improvement of existing processes.

Processing input output using Uber App on Android

On Uber Android App, a list of all previous rides is provided to select as a destination. If it is a new destination, then there is an option to enter a new address. Once the destination is selected, Uber displays the vehicles and price estimates. Instead of giving the lower and higher estimates, Uber just gives one value that is calculated using a different model and is a value somewhere in the middle of Google's upper and lower estimates.

Achieving Organisational Objectives

- Uber can experiment new machine learning models of price estimation in their own App.
- Collect data from the App based on the estimate queries progressing to booking or not progressing to booking, and it helps in business improvement.
- Better control on the app and the customer.

Reference: https://developer.uber.com

Requirement Code R7

Develop an extension or modification of the existing data model to improve an organisational function. You should provide details about:
- additional entities
- additional relationships between entities
- additional input or outputs
- additional processes
- which entities would be part of a public API

and must **explain** how these changes will lead to an improvement.

The following extensions to the existing data model will lead to business improvement.

- App Partner Discounts
  If a customer chooses Uber over other modes of transport on Google Maps App or Opal App (NSW Transport App), then the customer can be offered a discount. Though there may be a revenue deterioration initially, there will be revenue improvement. This will also help the customers, especially who work late during low availability of public transport and also provide trip to door. Revenue deterioration can be controlled by making more UberPool services available. (Opal App gives the option of OLA rides and Google Maps gives the option of Uber rides)

- Add trip preferences to customer profile
  When Uber estimate prices and duration are compared with other ride share apps, it is slightly cheaper. But there is a catch, the trip may be much more expensive than the estimate. This is because the map or the driver takes toll roads. Price estimate does not include a toll. In Sydney, toll roads can cause a 10-dollar added to estimated price. This will improve a customer's trust in Uber.

Additional entities

- **app_partner**
  The important properties of app_partner entity are global partnership indicator and revenue sharing indicator. If app_partner is a global partnership, then the customer will get discounts anywhere in the world. If app_partner is a local partner, then the discount will only be applied around the location_id up to 100km radius.
  App_partners can have a revenue sharing arrangement, if there is an arrangement, then the revenue sharing percentage should be included.

- **product_app_partner**
  product_app_partner is a junction table that maps the products created specifically for the partner with discounted price.

- **rider_preferences**
  rider_preferences entity has the preference properties with the indicator to specify, if the property should be considered for estimation. The preference available are toll_road_preferred, short_duration_preferred, short_distance_preferred.

Additional relationships between entities

- product_id will be a foreign key in product_app_partner
- rider_id will be a foreign key in rider_preferences

Additional input or outputs

Additional input and outputs are not required if these changes are kept to a minimum. There will be new products created for the partners with the right pricing and mapped to the product_app_partner junction table. This will be used for estimation and Uber does not charge more than the estimates. Similarly, rider_preferences will map the path of the ride which will avoid toll roads.

Additional processes

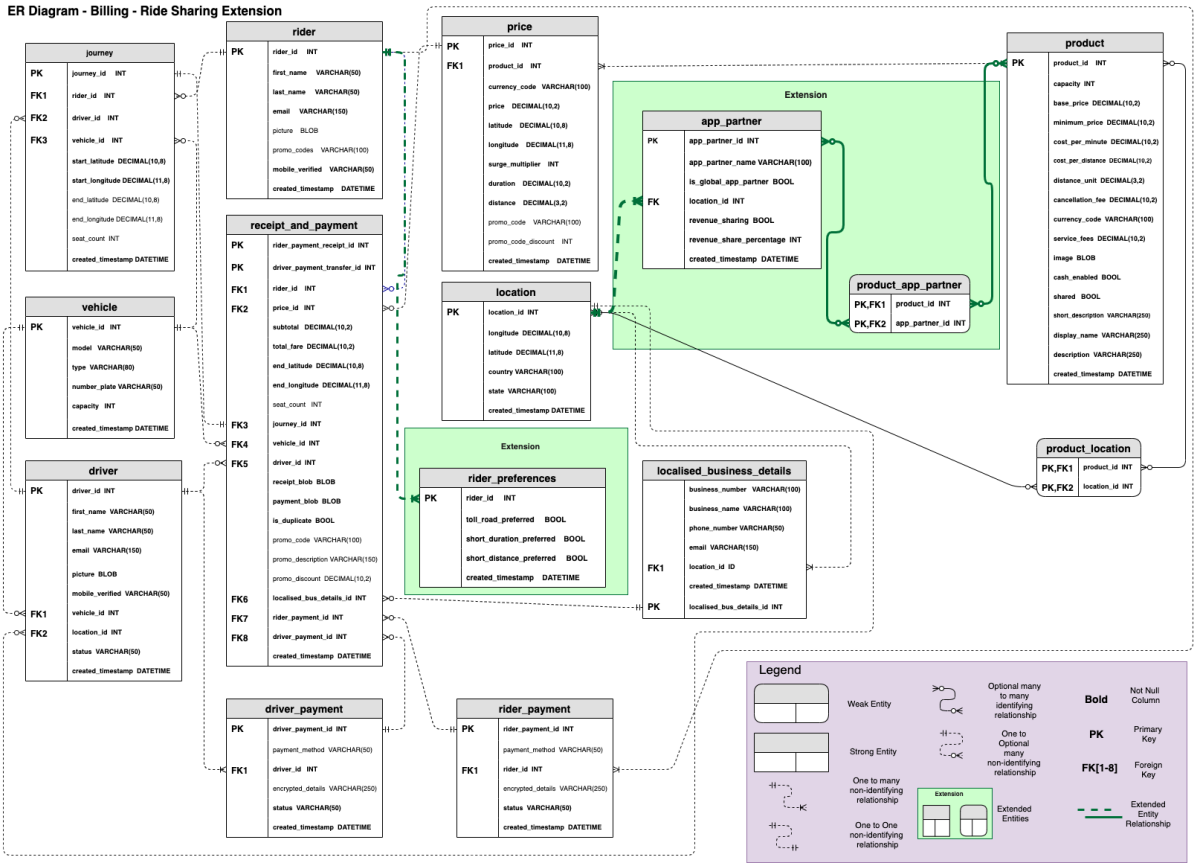
There are no additional processes required, but existing processes will require a change. There should be logic included to use data in new entities while calculating the estimates and mapping the trip to destination.

Which entities would be part of a public API?

rider_preferences should be part of Rider API. This will enable setting, retrieving and changing rider preferences.

Requirement Code R8

Create an entity relationship diagram(s) that **describe** the entities and relationships you propose to extend or modify the system. You must ensure that:
- your entities are normalised to the 1NF
- that it is shown how the new or modified entities connect with the existing system
- provides sufficient detail of data types and properties of entities for someone to write a SQL script



Requirement Code R9

Justify the technical and operational feasibility of your improvement or extension to an organisational function by providing information about:
- how the improvement will be supported technically
- how the improvement will affect stakeholders who utilise the system
- additional costs that may be incurred to implement the improvement

how the improvement will be supported technically

Technical support to implement these changes is required from software architects, business analysts, DBA's, developers, testers and operation teams.

A software architect is required to look into the changes in existing processes, code and API and assess the impact it may have on the current business and systems. Architect will assess the impact of adding new process and API's on performance.

BA's are required to analyse the value addition by introductions of the changes and also the impact of these changes on Systems and Stakeholders.
DBA's are required to help with adding new tables and required constraints, triggers, procedures, and assess if there are any migration activities required.

Developers are required to code the changes.

Testers are required to regression testing, new functionality tests and performance tests.

Operations team are required to make the changes live and support the changes.

Though DevOps has automated some of these activities, technical support from experts ensures safer implementation.

how the improvement will affect stakeholders who utilise the system

First, the app partners who utilise the system have to work with Uber business teams to develop contracts and products. For instance, Google Maps or Opal (Transport NSW) have to decide the base fares for the trip when a rider opts for Uber on their apps. They will also have to work out if the discounts are applicable globally or just in 100 km radius of the business location. In case Google maps, the discounts may apply globally whereas in case of transport NSW, it may just be in Sydney. The partners have to update the new products in their IT systems, as the old products may no longer work. Once these updates are done, partners are ready to offer the new services.

Uber has to notify riders about the preference extension and the riders have to update their preferences on the Uber app to receive the benefits of using non toll roads. For businesses

that manage Uber user profiles on their behalf, they have to change their programmes as there are new API's adding preferences to customer profiles.

Once the preference updates are completed, the end customer or the rider will enjoy informed, cheaper and transparent prices.

additional costs that may be incurred to implement the improvement

In a large organisation like Uber every activity incurs a project or operational cost.

There is no additional cost required to buy hardware.

The main cost is technical support required from software architects, business analysts, DBA's, developers, testers and operation teams

Apart from the technical support costs, there is the cost of project management to manage the change, business and legal teams to work with partners and user to negotiate and contract.

Requirement Code R10

Select TWO technologies in the organisation's stack and provide an alternative solution/technology by comparing the alternative with the existing solution (use of tabular form OR prose is ok).

Alternative Solution/technology

Oracle instead of MySQL

- Oracle has a much better query optimiser compared to MySQL. Oracle query optimiser can do much more than optimising based on indexing. Oracle optimiser can generate the most optimal execution plan for a query with the least cost. Optimiser add a lot of efficiency when the data grows.

- Oracle is a well-tested RDBMS database for large enterprises with large datasets in terms of data recovery, replication for distributed processing, disaster recovery and security. Uber did not have the amount of transactions they have now compared to when they started. MySQL was suitable when they started, but not anymore. Uber has added a layer on the top named "Schema less" to make it suitable for large datasets.

- Cost of having an Oracle database is higher compared to MySQL. But Oracle comes with a profession support package which guarantees support from experts around the world. Having a support package is an insurance.

Activiti Workflow Engine instead of Cadence

- Activiti is much easier to use compared to Cadence. It takes a long time to develop a workflow in Cadence as every step requires a lot of coding in Go or Java. The time required to develop workflows for new products delays bringing new products to market compared to Uber's competitors. Activiti is much easier as it has a well-designed and the steps and interactions can be developed in JSON or XML.

- Ongoing maintenance will be much easier in Activiti compared to Cadence. Uber creates a workflow every time a user orders a ride. Lots of these rides never happen and stale workflows remain in the system as active difficult to archive. Stale workflows use storage and other resources as it is waiting for triggers to progress to completion. Cleaning up these workflows will be a lot of work, as the logic requires coding and testing thoroughly. There is much less effort and time required to clean Activiti stale works as they can be triggered using JSON and pushed to completion with no unintended consequences, like incorrectly sending notification.