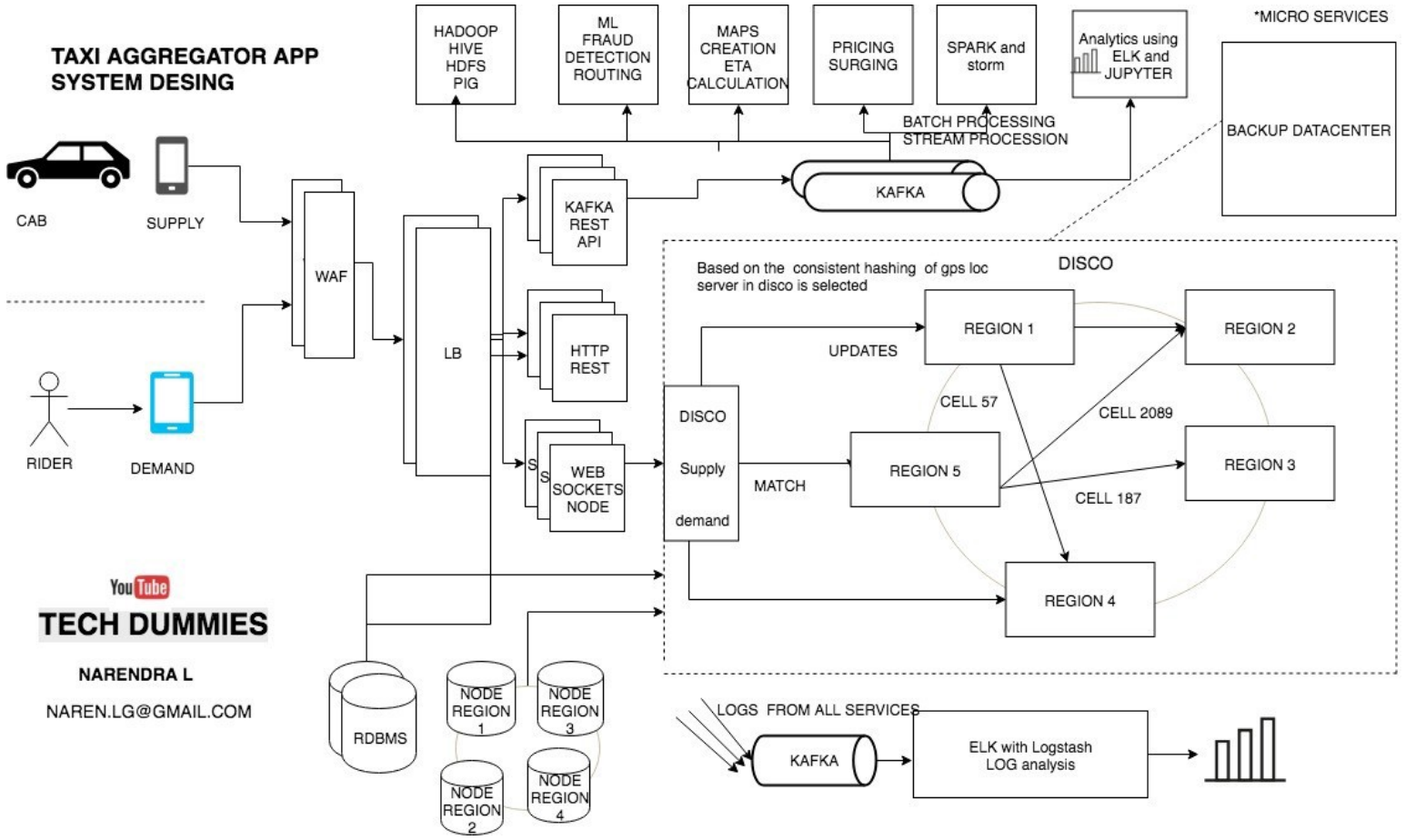
How Uber Works?

Ummi Izzati binti Mustapha

Uber – System Design



Uber - Services

SUPPLY SERVICE (Driver)

- Track cars using geolocation (lat and lang) Every cab which is active keep on sending lat-long to the server every 5 sec once
- The state machines of all of the supply also kept in memory
- To track vehicles there are many attributes to model: number of seats, type of vehicle, the presence of a car seat for children, can a wheelchair be fit, and so on.
- Allocation needs to be tracked. A vehicle, for example, may have three seats but two of those are occupied.

Uber -Services

DEMAND SERVICES (Customer)

- Tracks the GPS location of the user when requested
- Tracks requirements of the orders like Does a rider require small car/big car or pool etc

Uber- Architecture

Microservices and Monolothic Services

	Monolothic Services	Micro Services
Productivity, when teams and codebases are small	High	Low
Productivity, when teams and codebases are large	Low	High(Conway's law)
Requirements on Engineering Quality	High (under-qualified devs break down the system easily)	Low (runtimes are segregated)
Dependency Bump	Fast (centrally managed)	Slow
Multi-tenancy support / Production-staging Segregation	Easy	Hard (each individual service has to either 1) build staging env connected to others in staging 2) Multi-tenancy support across the request contexts and data storage)
Debuggability, assuming same modules, metrics, logs	Low	High (with distributed tracing)
Latency	Low (local)	High (remote)
DevOps Costs	Low (High on building tools)	High (capacity planning is hard)

Uber – Platform (Web)

Web and product teams collaborate to create and promote modular, separately-deployed web apps with shared user interfaces and a unified user experience.

Language	Node.js (large and vibrant community of web engineers)
Web Server	Bedrock, is built on top of the widely popular web framework Express.js, which has a set of default middleware to provide security, internationalization, and other Uber-specific pieces that handle infrastructure integration.
Rendering, State Handling, and Building	Use React.js and standard Flux for our application rendering and state handling. In the build system, Core Tasks, is a standard set of scripts to compile and version front-end assets, created on top of Gulp.js, that publishes to the file storage web service, allowing us to take advantage of a CDN service. Use an internal NPM registry to access the huge collection of public-registry packages as well as publish internal-only packages.

Uber-Paltform (Mobile)

Uber once had a strictly mobile org. Now, they have a cross-functional organization for what they call program teams. Each interdisciplinary program team has members from back end to design and data science.

Language	Uber's iOS engineers write in Objective C and Swift, while Android engineers write in Java.
Libraries	Use third-party libraries or build our own to fit specific needs. Many open source libraries available are general-purpose, which can create binary bloat. For mobile engineering, every kilobyte matters.
Android	On the Android side, Gradle is their build system. They use OkHttp, Retrofit, and Gson for networking. Dagger is their dependency injection framework.
iOS	iOS code lives in a monorepo that they build with Buck. Masonry and SnapKit with Auto Layout help with component placement and sizing. For crash detection, they use KSCrash and report the crashes using their internal reporting framework. For testing in Objective-C, they use OCMock to mock and stub classes. For testing in Swift, they generate mocks via protocols.
Storage	Uber use LevelDB. On the back end, they use the standard Schemaless and MySQL, gradually moving toward all-Schemaless.