**Points to Discuss in the Presentation**

1. **HSBC (< 1 min)**

Principles?

1. **Overview of the Client Requirements (Long-ish)**

* R1: Detect Bluetooth pulses. Transfer limited information (1024 bytes) from one phone to another based on interaction time and proximity.
* R2: Allow self reporting. Link users to testing facilities.
* R3: Find a way to update results for large numbers of users that are tested in batches (usual form of testing in labs) at a time.
* R4: Find a way to notify all users that have been in contact with a suspected/confirmed COVID-19 patient. Problem of sheer scale – not so bad due to lockdown guidelines.
* R5: Provide relevant data in required format to dashboard.
* Keeping track of “social hops” from initial infected person, and time since interaction with infected person for all relevant users as per guidelines (max 14 days, max 5 “hops”).
* Ensuring minimal private/personal data of users is stored outside users’ phones – prevent access by third parties. Unnecessary data must be deleted. Public trust.
* Prevent “gaming” of the system as much as possible – prevent a single person reporting multiple times, possibly monitor for dishonest self-reporting.
* Scale of system – 80% of mobile phone users all over UK – 50.16 million users. Up to 500,000 self-reports, 500,000 test results a day. Need for highly dynamic, error-proof system.
* Reliability – 4 9’s on medical results – less than 50 failures (0.01% of 500,000) a day. Even better performance on notifying users. Should perform even in absence of internet/range (buffering?)
* Availability – Ensure servers/ phones can handle their loads. Need to account for failure of devices/ storage.
* Rollout: 8 weeks. Ease of implementation, piloting, testing, debugging by engineers.

1. **Overview of System and How it Meets the Requirements (Probably longest part)**

* Overview of our architecture using AOD and SCD.
* Mention “users register with a unique ID linked with their phone”
* As we discuss different parts of the system we highlight how a particular part solves a particular problem.

How Different Problems are Solved:

* R1: Bluetooth system, as per client spec. Transfer data to other phone: user ID, status, location, time.
* R2: Medical practitioner receives ID of self-reporting user. Links with result.
* R3: Integrate software component with device used to evaluate test samples. Automatically sends result, linked with ID, to system central storage – similar to sending automated emails from an excel sheet. Medics can also update information from a special interface. Login would be required in either case.
* R4: We reach every relevant user by a “spreading” of the status that mirrors the infection cone. First the system notifies a user of a change in his status (due to self-report/confirmed test). The user’s phone sends interaction data to a local cluster of high-speed servers – these then send notifications to all relevant contacted users, as per the guidelines (14 days, 5 hops).
* R5: Storage of users is already sorted by location (level 1 storages). Data is sent to dashboard once a day.
* Hops and time since interaction: When sending notifications, the severs calculate the number of hops from the initial user. Data older than 14 days is already deleted on every phone.
* User privacy: All user interaction data is stored on phones. Server only keeps interaction data as long as is necessary to propagate notifications. Central storage only has list of user ID’s, and status.
* A user trying to self-report multiple times will be sent an error message. Difficult to guard against dishonest reporting as we need to give people the benefit of the doubt to prevent spreading of the virus. Can maintain percentage of confirmed cases with respect to self-reports? (don’t need to put this).
* Scale: 3 tier “cache” system. Divides users by location. Can easily be scaled, as no need for synchronisation. Reduces load as self-reports/results are distributed.
* Reliability: Buffering of self-reports/ notifications in the absence of internet. Cloud backup for each phone. In 3-tier system, level 2 backs up level 1. Level 3 backs up level 2. Highly resilient to failure.
* Availability: Distributed clusters of high-speed servers handle load of spreading notifications.
* Rollout: Easy since most of the processing is on the phone. Data storage/servers can be set up quickly.

1. **What Makes Our System Special? (Relatively short)**

* Ease: It simplifies the experience for all users of the system. General users only need to self-report and receive results. Allows us to focus on an attractive UI. Medical practitioners often don’t need to lift a finger. Low latency.
* Privacy and Scalability Together: More data is stored on phones than on server, which stores the bare minimum. Unique as compared to most client-server systems. Useful for privacy. This also makes the system more scalable, as storage will automatically grow as more phones join. This allows us to focus on managing load.
* Emphasis on failure tolerance: The unique 3-tier “cache” system ensures high resistance to loss of data. Backup provided for each phone. Buffering in absence of internet.
* Emphasis on Availability: Nationwide distribution of level 1 storage and notification server clusters designed to handle load and accounts for higher transmission rates in cities.
* Will Work When Lockdown is Lifted: We estimate a slight increase in failure rates and increased latency when the lockdown is lifted due to increased load and people travelling. However, the system will work. It is designed to be scaled up in the face of such a situation.

1. **Modelling Use Cases (Relatively Short)**

* Cover two submitted use cases. Illustrate possible interface (Maelle’s website). Illustrate how easy it is?
* Maybe cover how support staff would use the system? (May not be important to C-suite folk)

1. **Recap ( < 1 min)**

* Catchy final message – need good delivery – “Our system is fast, flexible and fun.” (3 F’s)

END