For Oblig 4 we have implemented Cassandra db implementation, this means that we have set up a Cassandra Db, that we run through cassandras docker file. Cassandra Db Is a nosql database that is made for scalability and availability as its main objective.

When implementing CassandraDB or any other database solution in a distributed system, it is important to consider the security challenges faced by such systems. While CassandraDB offers scalability and availability, it does not inherently address all the security challenges. Authentication, authorization, confidentiality, integrity, and availability are all important aspects of securing a distributed system.

To deal with the problems of authentication and authorization, Kerberos can help. It's a network authentication protocol that uses symmetric-key cryptography to confirm that users are who they say they are and have the correct permissions to access resources.

Kerberos provides a secure way to manage authentication and authorization in a distributed system, which helps to keep sensitive data safe.

On the other hand, Bitcoin can help with traceability by creating an unchangeable record of transactions using blockchain technology. But it doesn't address issues of confidentiality, integrity, and availability.

But Blockchain technology can help with confidentiality by using encryption to secure sensitive data. It also provides integrity by verifying every transaction multiple times, making it hard for the wrong user to change the data.

However, you should be aware that blockchain technology isn't always the best solution. Scalability, interoperability, and regulatory compliance are still problems that need to be considered.

The consensus mechanism of blockchain technology ensures that every transaction is validated by multiple nodes, making it difficult to alter or delete data. However, blockchain technology has some limitations, such as scalability, interoperability, and regulatory compliance, that need to be considered before implementing it.

In summary, blockchain technology can help with security in a distributed system, but it's essential to understand its benefits and limitations before using it.

Overall, it's important to think about both scalability and security when setting up distributed systems. While tools like CassandraDB offer scalability, tools like Kerberos and blockchain can help solve security problems.

For our implementation Cassandra db we have made separate tables for each of our previous data classes(patient, symptom, department and doctor) in the project these files are named

PatientCass, symptomCass and so on. These set up the tables in cassandraDb and have their previous ids as primary keys

Each of these new tables have repositories setup to use CassandraDB's methods, such as find, findall, findallbyid and so on, and some have their own methods implemented. Which allows us to save, and receive data to and from these tables. The naming scheme of these Repositories on the form of the normal repository with Cass after them. Eg. PatientRepositoryCass.

To intialise these tables we have set up a codelinerunner which saves some data in these tables to make it easy to test the methods for getting out data from the api. In the codelinerunner you will find that we have opted to clear the data from Cassandra db on startup of the program. In a final deployment this would not be optimal, but this is an easy way to be able to run several instances of the app and not create the same tables on startup. This setup would have to be changed for a full deployment, as of course deleting data for each new server instance would be detrimental.

You will also find, as we have noted in the readme that we only have a replication-factor of one. This has to do with us only setting up one node of Cassandra-db, and therefore there is no other nodes to replicate on. This could be implroved upon by for example setting up a Kubernetes cluster, with several cassandraDb nodes, so that data could be replicated. The replication of nodes is highly important to both the availability of the api(as one node could go down and the api would still be available), as well as for the data security. If your one single node goes down, that data is lost forever, and thus the Recommended way of setting it up is to have nodes on different servers, on different locations so that the whole service will not be effected by a location losing power, going down another way. This is the main point of a distributed system.

Even though we do not have a distributed system at this time(as previously discussed), it could easily with the previously mentioned methods be made distributed. To conclude the part of our database implementation, it does not yet have high availability, but is set up to be scalable, which in turn will provide availability. However, as you will find in the code, the database tables are not fully implemented, and so not all the calls will return data from the database, and not all the post methods will insert it into the database. This is a flaw that stems from us not changing all the old repositories to use the Cassandra repositories classes

Security

as you will see in the finished solution we have not delivered on all the security features that we had planned to, and that we wanted to. This was poor planning on our part.

For now we have implemented a pre-set username and password as authentication

We do however have the unfinished work in dev2 as it may be interesting to look at our logic for audit logging, our authentication controller and the way we have Implemented the user table and repository for Cassandra.

Using Cassandra this branch uses spring and JJwt to compare the hashed password values of the Cassandra db table and the user input.

Users in Cassandra database are set up so that they have their unique id, username, name, and password and role.
our intention as you will se commented out in the security config files was to filter the allowed methods and endpoints by the users role, either USER or ADMIN.
in the finished solution this would work in the way Users would have right to most methods, but not delete, while admins would be able to access all methods as well as create new users.

This combined with the audit logging would enable us to see which users had done which operation at which endpoint and also their role.

.