1. **Introduction**

In today’s technology genomics research is widely used for discovery of mutations and by that way, cures for human diseases can be further improved. Data sharing is crucial for this kind genomics research; however the data must be safe from being removed from existing databases by financial and political reasons, required by the organizations []. Our project, CrypDist, provides a way to access genomics data more securely. It is a decentralized distributed system which uses a distributed ledger called blockchain to record the URL links of datasets.

Blockchain is a cryptographic data structure which ensures immutability of data and avoids third-party access. It basically provides synchronization of the data links among many users and it also includes data summaries. The mentioned data is not kept in the blockchain because of its size.

1. **Packages and Tools**

* There is a local database in each of the machines which is developed by using PostgreSQL package [].
* For managing logging information, Log4j package is used which belongs to Apache Software Foundation [].
* For testing purposes, a stub server is used for uploading the data which is provided by Amazon Web Services []. In the future, Akamai services [] is planned to be configured for the project for more efficient usage.
* //TODO Timestamp server
* For build automation, we implemented the project by using Apache Maven [].

1. **User Interface**
2. **Hardware/Software Mapping**
3. **Software Architecture**
   1. **Subsystem Decomposition**

Figure x shows the subsystem decomposition of the system. Class diagrams are drawn after that for clarity.

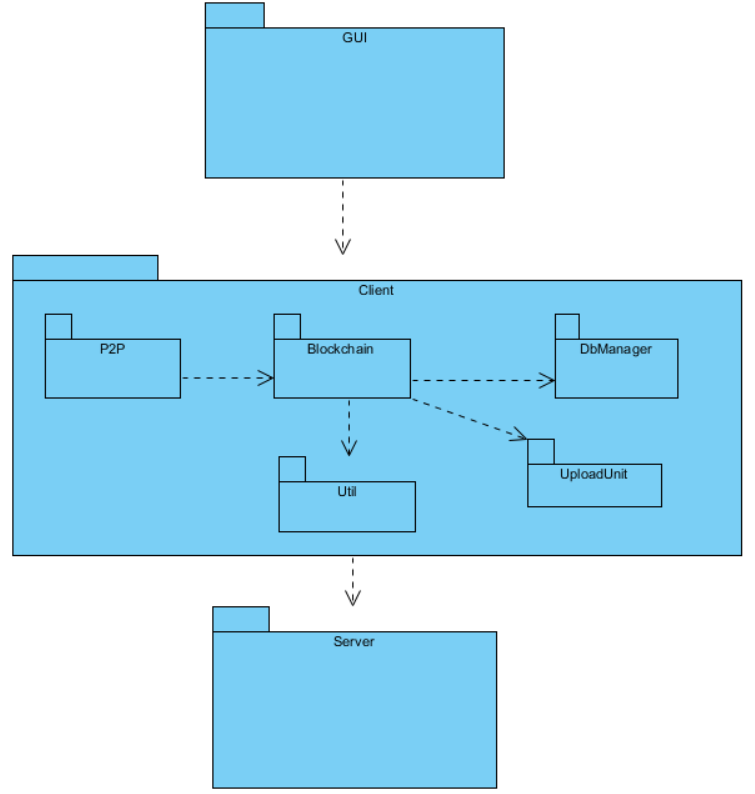
****

Figure x

The system has a 3-tier architectural style. The top layer includes the graphical user interface of the system. The middle layer includes the Client interfaces where client is a peer in the system. Blockchain of peer is managed by the Blockchain subsystem and it is kept in a local database which is managed by the DbManager subsystem. By the P2P subsystem, peer can contribute to peer-to-peer connection by other peers. UploadUnit subsystem is for uploading genome data to the off-the-shelf server and Util subsystem is used for managing client operations such as parallel download and receiving hash. Finally, the Server subsystem acts as a registrar which keeps the IP addresses and authentication credentials of the clients and peers can get those information from the registrar as necessary.

* 1. **Subsystem Interfaces**

**GUI Subsystem**

Figure x shows the GUI subsystem.

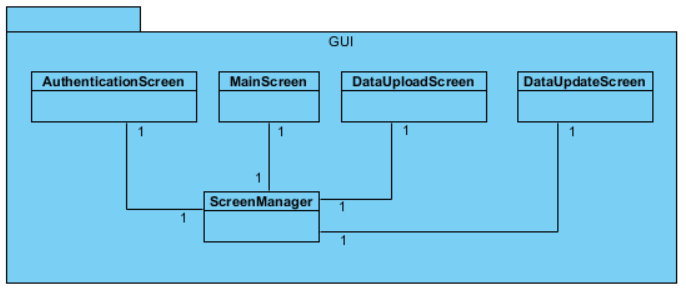


Figure x – GUI subsystem

Clients authenticate themselves via AuthenticationScreen. In MainScreen, they can query for the data in the blockchain. They can upload new data by DataUploadScreen and provide new versions for it by DataUpdateScreen. ScreenManager provides the interface between Client and GUI subsystems that it forwards the requests coming from the user interface to the bottom layer.

**Client Subsystem**

* **P2P Subsystem**
* **Blockchain Subsystem**
* **DbManager Subsystem**
* **UploadUnit Subsystem**
* **Util Subsystem**

**Server Subsystem**

* 1. **Design Patterns**

1. **Data Structures and Algorithms**
   1. **Blockchain Structure**
   2. **Block Mining Algorithm**
   3. **Digital Signature Algorithm**
   4. **Hash Choosing Algorithm**
   5. **Parallel Download Algorithm**
2. **Impact of Engineering Solutions**
3. **Contemporary Issues**
4. **User’s Manual**