Media Storage:

Minio:

- Version : minio version RELEASE.2021-03-17T02-33-02Z
- Java Driver : io.minio 8.2.1
- MinioInstance Class was created where a persistent connection to minio server was made . This Class was instantiated with each service
- A MediaHandler was added to the netty pipeline to handle requests related to the media storage

NoSQL:

Arangodb:

- Version: ArangoDB 3.7.1
- Java Driver : `arangodb-java-driver 6.11.1`
- ArangoInstance Class was created where a persistent connection to the NoSQL database was made. This Class was instantiated with each service

SQL:

PostgreSQL:

- Version: postgres (PostgreSQL) 13.2
- Java Driver : `postgresql 42.2.1`
- PostgresConnection Class was created where a persistent connection to the SQL database was made. This Class was instantiated with each service

DB Pooling :

DBCP2:

- Version: 2.8.0
- Java Driver: org.apache.commons (commons-dbcp2)
- PoolingDataSource was used in (PostGresql) class to access the PostGresql Database after defining a Connection Pool with Maximum Connections allowed per DB instance.

Cache:

Redis:

- Java Driver : `redisson 3.6.1`
- RedisConnection Class was created where a persistent connection to the redis cluster was made. This Class was instantiated with each service.
- A cluster consisting of 6 nodes is started on the server startup. Redis clusters use a master-slave schema, where master nodes are the main nodes and slave nodes are used for data replication to account for potential failures.
- In our configuration, there are 3 master nodes and 3 slave nodes (a unique slave node for each master node).

Load Balancer:

HAProxy:

- Version: 2.4.0
- Name: HAProxy
- Used haproxy.cfg file to define the Load Balancer Server at port 90 which directs the requests to two servers in a round-robin fashion.

Messaging Queue:

RabbitMQ:

- Version: RabbitMQ 3.8.12
- Maven Dependency: amqp-client 5.1.2
- RabbitMQ was used to create request and response queues for each micro-service, handling the produce and consume functions was done in RequestHandler, and ServiceControl.

Chat:

Netty:

- Version: Netty 4.1.21

- The built-in WebSocketServerProtocolHandler Class was used to handle websocket connections.
- A TextWebSocketFrameHandler Class was created to handle websocket frames sent through the websocket connection and forward them to the correct recipient.

Threading Pool:

Java ThreadPoolExecutor:

- Package: java.util.concurrent.ThreadPoolExecutor
- ThreadPoolExecutor was set to define the number of threads used to handle the execution of commands consumed from the request queues after reflection.

Independent Application Framework:

Netty:

- Version: Netty 4.1.21
- We used Netty ServerBootstrap, and Channel in NettyWebServer.java in order to initialize the Web Servers.
- Used io.netty.channel.ChannelInitilializer and io.netty.channel.ChannelPipeline in NettyWebServerInitilizer to initialize our own handler Pipeline.
- Used `io.netty.channel.ChannelInboundHandlerAdapter in RequestHandler and HTTPHandler to decode the HTTP requests received from client and direct them to corresponding queue.`

Notifications:

Firebase:

- Notifications are sent via the firebase HTTP-API.
- Users who accept to receive notifications are assigned a ***firebase cloud messaging*** (FCM) token that is sent via an HTTP request an FCM registered account (FCM authentication is done by sending the account-unique key in the request header).

Authentication:

Java-JWT:

- Version: 3.15.0
- The HMAC256 algorithm was used to create JWT tokens with payload containing either the userId or moderatorId

Bcrypt:

- Version: 0.9.0
- The library was used to implement the bcrypt algorithm to hash passwords during users and moderators sign up, and verify the passwords sent during sign in.

Geolocation:

GeoIP2:

- Version: 2.8.0
- The geoip2 library and database were used to extract location information from the IP address of the client to be able to set the user's location in the database.

Logger:

Java Logging:

- Package: `java.util.logging`
- `Used Logger and LogManager in MyLogger.java in order to set-up logging properties and log all events on console and in .log file on disk.`

Unit Tests:

JUnit:

- Version : JUnit 4.12
- Test Files can be found in ‘tests’ package.