This solution is for the first system in the assignment.

In my solution there are 6 services, as the following:

Service** (name in assignment)	Base Image	Port	Technology Stack	Volume**	Dependency
Mongo (MongoDB)	Mongo:6.0.3	27017	-	Mongodb_data	-
Mydb2 (MySql DB)	Mysql:latest	3306	-	Mysql_data	-
Collect (Enter Data)	openjdk:17	8080	Spring MVC	-	Mydb2
Info (Show Results)	openjdk:17	8082	Spring MVC	-	mongo
Authentication (Authentication Service)	openjdk:17	8081	Spring Rest	-	-
Analysis (Analytics Service)	openjdk:17	8083	Spring Rest	-	Mydb2 mongo

These services are defined in docker-compose file, so when we execute this file, all these services are running containers, and they can communicate with each other in the default network that is created by docker-compose file and named default**), let's talk about each service and how does it communicate with other services.

**When docker-compose is running, those names are prefixed with the name of the directory where docker-compose is.

1. Enter data (Web app) – collect (Spring MVC):

   This service is based on Spring MVC project, and it contains single controller, two models and two views; login and input. Login view is HTML page contains a form for that receives username and password from the user, and on submit, username and password which is wrapped in Login object, are passed to **createSession** method using HTTP-POST request, this method will send HTTP-POST request using **RestTemplate** to login method in authentication service to check user authority,(more details in authentication section), if the user is authenticated, a session will be created for that user and input page will open prompt him to enter a double value, and will list all entered values before which are stored in **mysql_data** volume which has a database schema called mydata and this schema has one table called Data with two columns (ID and VAL), and that volume is mounted to mydb2 container. If the logged user enter a value, this value will be stored there also via the connection that is established by identifying data source in the web app's configuration, connection string in projects that are containerized is little bit different from traditional connection strings, instead of using localhost as host name of the service or (database in our case), we use container name that is written in docker-compose file as host name, because in docker terms, service names are resolved to their respective container IP addresses. When you define services in a Docker Compose file, the service name acts as a hostname that can be resolved within the Docker internal network.

   After the entered value is stored in MY SQL database, HTTP POST request is send to analysis service (more details in analysis service section), to update mongoDB database that is also another service according new entered values

   In input page, there is a logout button to invalidate (terminate) the session and redirect the user to login page again.

2. Authentication Service – (Spring Rest):

   This service is based on Spring Rest project, it is a Restful API that has one controller called LoginContoller and has one endpoint called login receives a Login object and check if there is such objects in a hard-coded list of Login objects, if such object is exist, this endpoint returns (OK 200) HTTP STATUS CODE, otherwise, it returns (UNAUTHORIZED 401) HTTP STATUS CODE indicates that entered data is invalid login credentials.

   There are two services consume this API, Enter Data (Web app) and Show Results (Web app), I used **CrossOrigin** annotation to allow these services to consume this API.

3. Analytics Service – (Spring Rest):
   This service is based on Spring Rest project, it has two data sources, first one for read from MYSQL database, and second to write and update mongoDB database, and has a AnalysisController with analyze endpoint, this endpoint will read all data in MYSQL database, then calculate minimum, maximum and average values and update mongoDB database with these values, this endpoint is requested after each creation operation in Enter data (Web app), web app will call it using RestTemplate object.

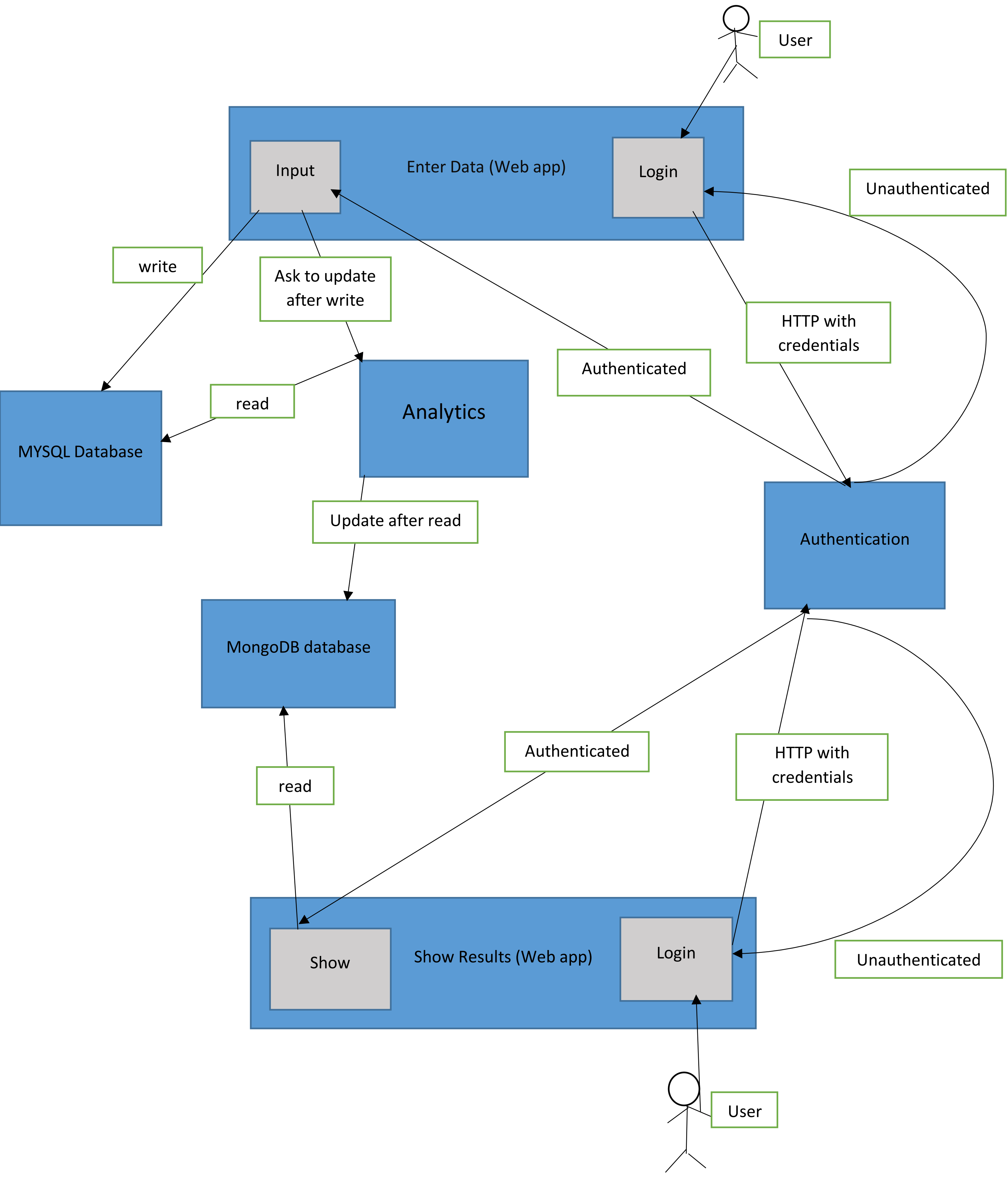
4. Show Results (Web app) – (Spring MVC):
   This service as Enter data (Web app) service, in terms of technology stack and authentication approach used, but It's connected to mongo container to read the processed data from Analyst collection stored in the mounted volume to that container using MongoTemplate object and displays these data for authenticated users in HTML page called show.

5. MySql DB – (docker image):

   This service uses the original image mysql:latest for MYSQL server as base image to run, in docker compose file, mydb2 service is a container exposed port 3306 with a mounted volume called mysql_data to save the data for the long term, we need volumes in this case, and in mongoDB case, because containers are stateless, which means they have no memory, any data desired to be saved after removing the containers must be stored in volumes. This service is used in Enter data (Web app) and in Analytics Service.

6. MongoDB – (docker image):

   This service uses the original image mongo:6.0.3 for mongoDB database as base image to run, in docker compose file, mongo service is a container exposed port 27017 with a mounted volume called mongodb_data, which stores a schema called mydata, has one collection called Analyst. This service is used in Show Results (Web app) and in Analytics Service.



Use Case Diagram

All these services are containerized either by creating a docker file for each of them (Spring MVC and Spring Rest based services), or by mentioning their base images for (Databases services) in docker-compose file.

How we create docker files?

for each service I build, I create a docker file inside its directory, to deodorize a Java application, you need to create a .jar file for that application, command:

```
mvn clean package
```

inside application directory, will create a jar for that app, and store it in target directory, for spring applications that are created using spring initializr, in docker file first, we must choose a base image for your container image, then choose working directory inside the container, copy your jar file to working directory, then specify the command to run when your docker container starts up.

In docker-compose file, build keyword is used to tell docker-compose where to find Dockerfile for some service to build the image for that service, more formally it receives the path of a Dockerfile.