Notes Taken By

- Muhammad Raaid Khan
- Data Science and AI (Batch - 05)
- NED - CCEE

Environment Variable

- Information Stored in `.evn` file.
- Information is loaded into Operating System (OS)
- To get this information, the Hacker will have to Hack the Operating System or Cloud Platform on which that application is Hosted.

Docker Compose Commands

- `docker compose up`
    - Build all Services in compose.yaml file.
    - First Time
- `docker compose up --build`
    - Re-Build Services in case of any change in Source Code.

Docker Database Connection

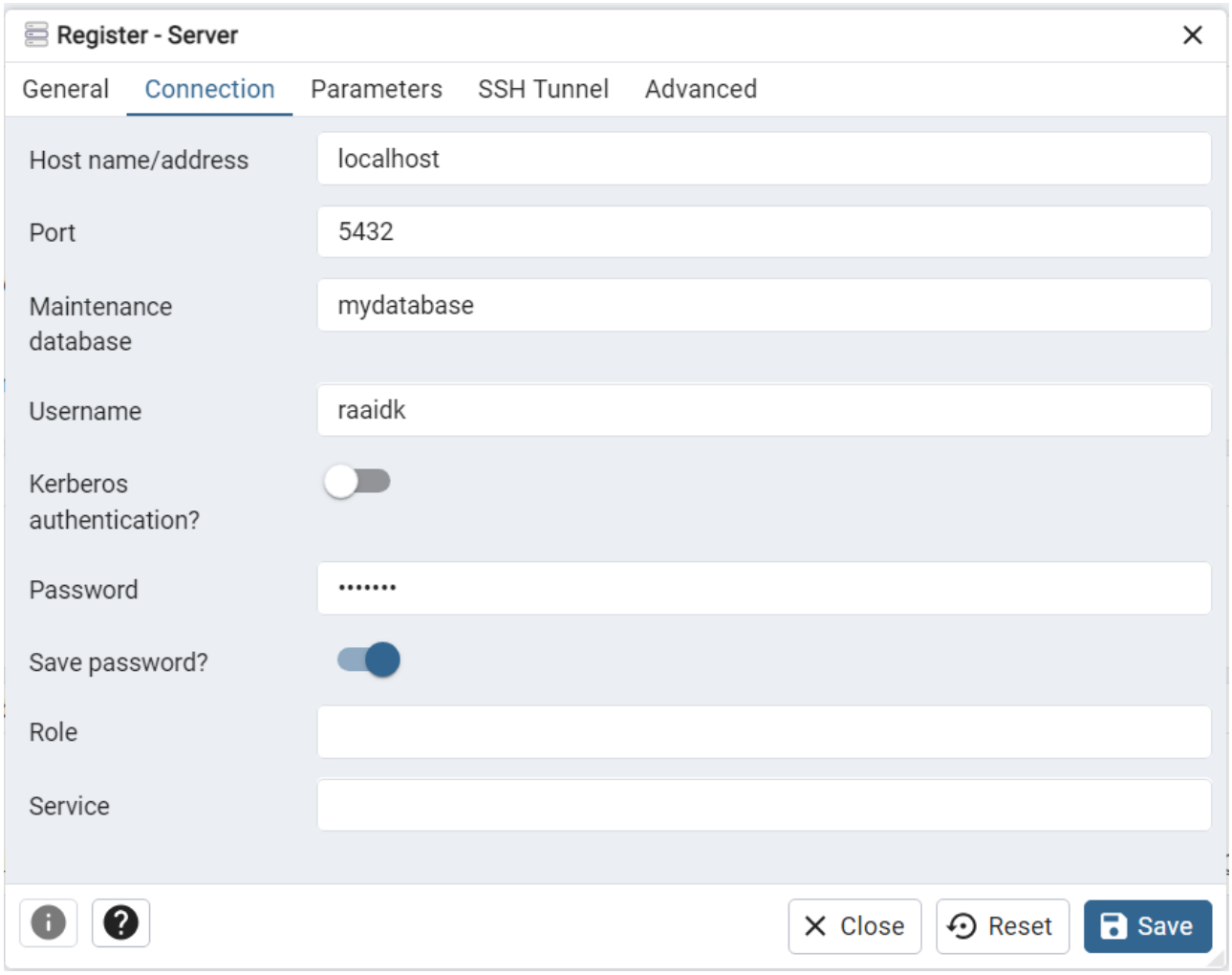
```
conn_url = 'postgresql+psycopg2://yourUserDBName:yourUserDBPassword@yourDBDockerContainerName/yourDBName
```

pgAdmin

- Create a Database connection using environment variable of Postgres_db service from our `compose.yaml` file.

```
  postgres_db:
    image: postgres:latest
    restart: always
    container_name: PostgresCont  # Name Used for Database Connectivity

    # These parameters will be made part of OS Environment on Docker Runtime
    # In Actual, we use .env file for Security Purpose
    environment:
      - POSTGRES_USER=raaidk
      - POSTGRES_PASSWORD=pass123
      - POSTGRES_DB=mydatabase
```



`Note:` Port = **5433** (i.e. Port of Local Computer on which the Docker Database is Mapped)

- This will connect pgAdmin to `PostgresCont` in Docker.

Database for Testing

- Create a Separate Database for Testing Functions.

```
from app.main import app

def test_write_main():
```

```
    connection_string = str(settings.TEST_DATABASE_URL)

    engine = create_engine(
        connection_string, connect_args={"sslmode": "require"}, pool_recycle=300)

    SQLModel.metadata.create_all(engine)

    with Session(engine) as session:

        def get_session_override():
                return session

        # Override the Database session in Main App to TEST_DATABASE
        app.dependency_overrides[get_session] = get_session_override

        client = TestClient(app=app)

        todo_content = "buy bread"

        response = client.post("/todos/",
            json={"content": todo_content}
        )

        data = response.json()

        assert response.status_code == 200
        assert data["content"] == todo_content
```

Event Driven Architecture

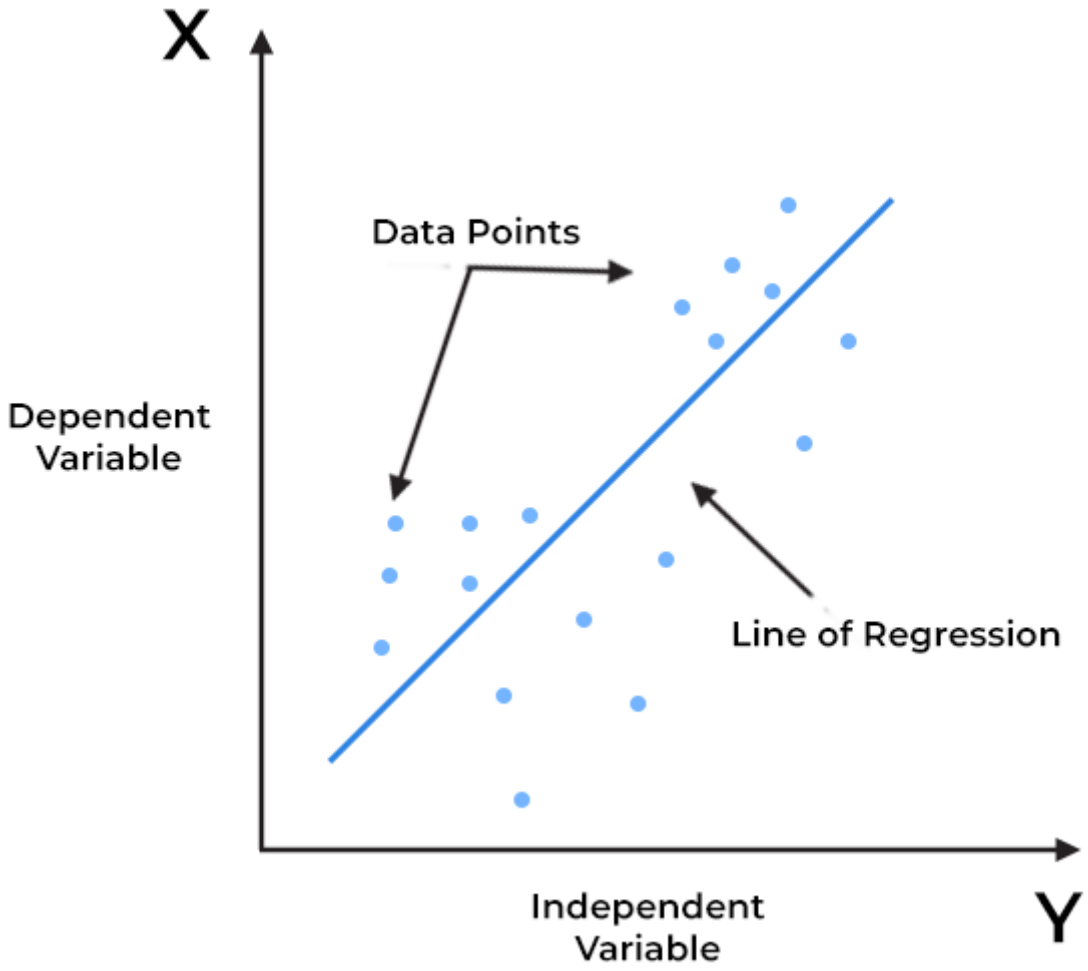
- Apache Kafka
- Home Work

Project

- At least 03 Microservices
    - FastAPI
    - Database
    - User Interface
- All services to start with single `docker compose up` command.

Regression

- X = Independent Variable
- Y = Dependent Variable



Equation of Line

$y = wx + b$

- w & b are weights of Linear Regression
    - w = Slope of Line
    - b = Y Intercept of Line
- Machine Learning find the appropriate values of *w* & *b*
- This can be done using Libraries such as *Tensorflow*.

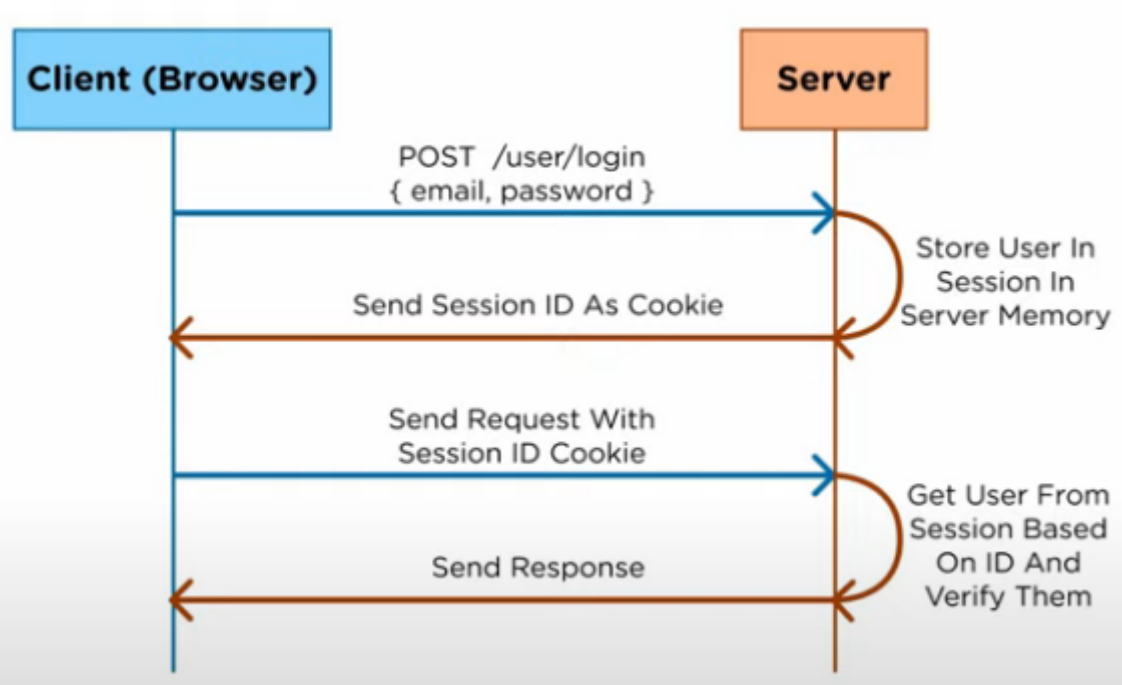
Types of Data

- Qualitative Data
- Quantitative Data
    - Discrete
        - Ordinal (In Order)
        - Nominal
        - Categorical Data
    - Continuous

Authentication & Authorization

Session Based

- Verifying User from Username & Password
- Create **Sessions** on Server using Database / Memory.
- Sends Session ID as Cookie to Client.
- Cookie is stored on Client (Front End)
- Users can interact with Backend after Authentication using Session Cookie.
    - i.e. Authorized

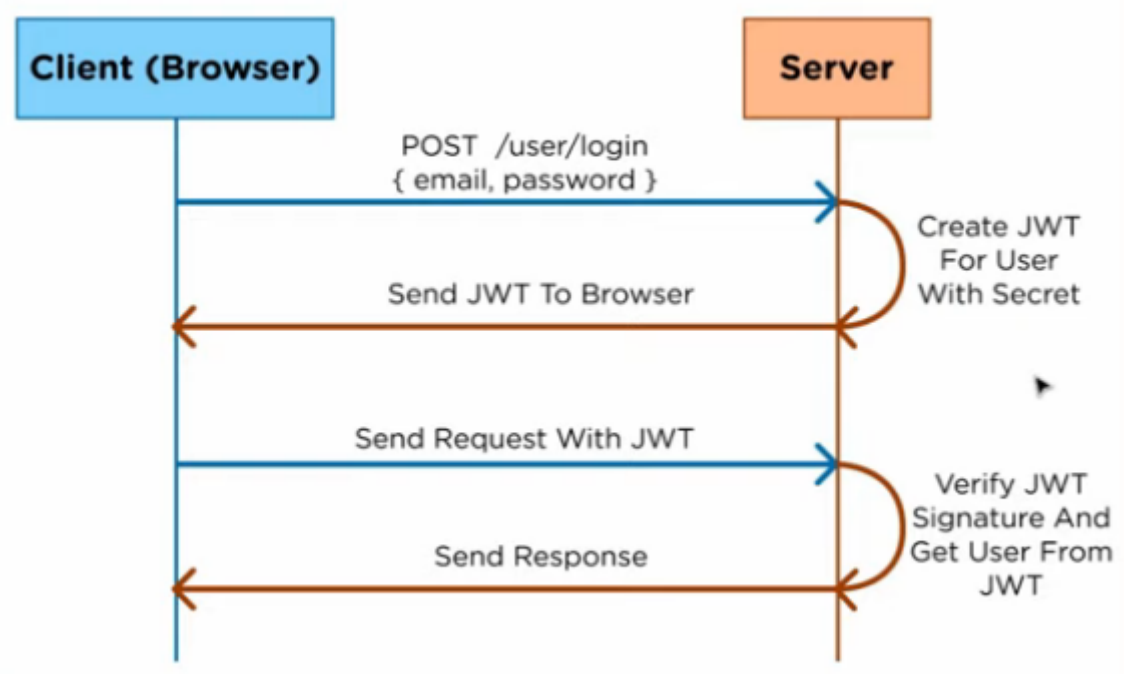


Cons:

- Server has to do lookup in Database to get the correct user. (Will take time for a large System)

Token Based

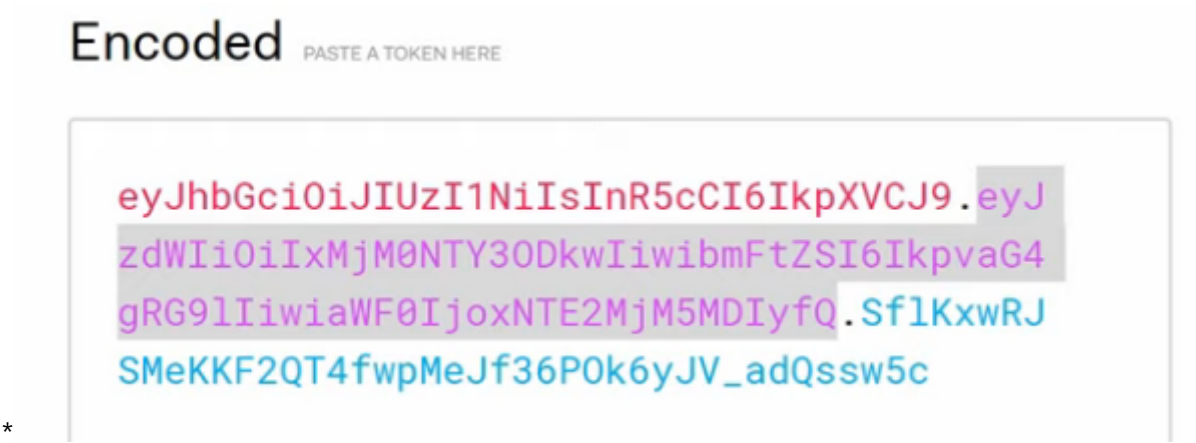
- Verifying User from Username & Password.
- Server creates a **Json Web Token (JWT)** and sign it with its own secret key.
    - Since JWT is signed with secret Key of Server, it will become *invalid* if it is tampered.
    - No information is stored on Server.
    - JWT has all the information about the user.
- JWT is sent back to Client.
    - Client can store JWT in any way i.e. Cookie / Local etc.
- Future requests from Client will contain JWT.
- Server will get User information after De-Coding (De-Serializing) the JWT.



How JWT Works

Encoding

- How to Encode/Decode is defined in Header
- Encode Header and Payload using Header Info.
- Use `HMACSHA256` to create a Verification Signature of (Header + Payload) using Secret Key of server and append it to last (blue) section of JWT.



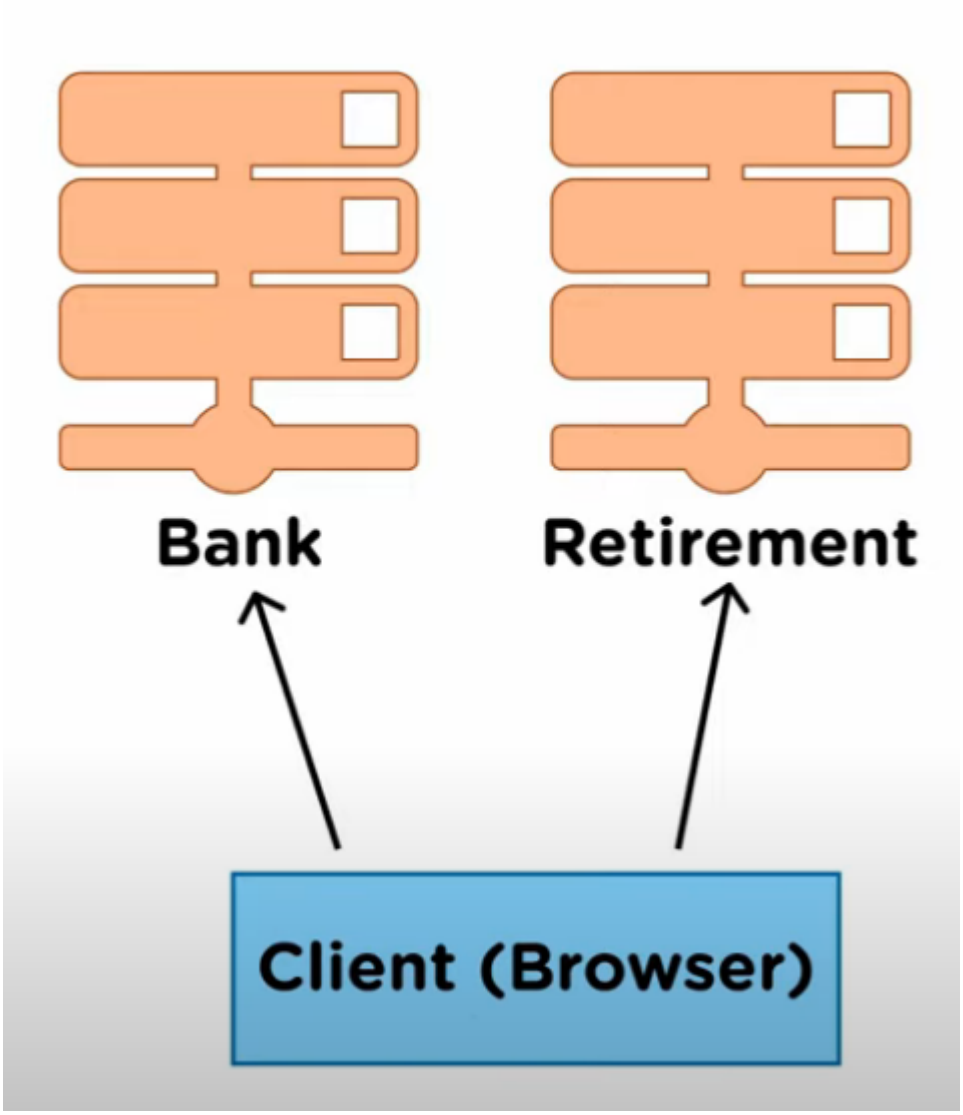
*

Decoding

- Decode Header
- Decode Payload
- Create a Verification Signature (using its Secret Key) and Match it with the signature provided in JWT.
- Secret Key has to be safely stored on Server.

Why Use JWT

- If a Client logs in to One Server, it will get a JWT.
- Client can login in to another Server of same bank with authentication by using the same JWT.
    - Since login information is stored on Client and not on Server.


Bank
Retirement
Client (Browser)