# Cassandra project

Maksymilian Żmuda-Trzebiatowski 156 051

## Descritpion:

This project implements a distributed cinema seat reservation system using Apache Cassandra and a Python CLI application. It simulates real-world scenarios where multiple users reserve movie seats concurrently. The CLI offers regular operations (viewing movies, reserving seats, updating reservations) and several stress tests to evaluate the system’s behavior under heavy load or contention.

The system is deployed using Docker with a multi-node Cassandra cluster to ensure high availability and fault tolerance.

## Database schema:

***Keyspace:***

```
CREATE KEYSPACE IF NOT EXISTS cinema
WITH replication = {
  'class': 'SimpleStrategy',
  'replication_factor': 2
};
```

***Tables:***

```
CREATE TABLE IF NOT EXISTS movies (
  movie_id TEXT PRIMARY KEY,
  title TEXT,
  description TEXT
);


CREATE TABLE IF NOT EXISTS showtimes (
  showtime_id TEXT PRIMARY KEY,
  movie_id TEXT,
```

```
  start_time TIMESTAMP,
  auditorium TEXT
);

CREATE TABLE IF NOT EXISTS showtimes_by_movie (
  movie_id TEXT,
  showtime_id TEXT,
  start_time TIMESTAMP,
  auditorium TEXT,
  PRIMARY KEY (movie_id, showtime_id)
);

CREATE TABLE IF NOT EXISTS reservations (
  showtime_id TEXT,
  seat_number TEXT,
  user_id TEXT,
  reserved_at TIMESTAMP,
  PRIMARY KEY (showtime_id, seat_number)
);
```

# Problems encountered:

1. The app lost data when the nodes were shut down. Used volumes to store data locally.

2. Stress test 3: sometimes failed CAS (Compare-And-Set) operations due to lack of quorum. Resolved by adding a small delay between request. (real-world scenario)

3. Stress test 4:

Resolved by adding a small delay between request. (real-world scenario).

4. Running multiple stress test would fill the reservation list. Resolved by clearing the list after each stress test.