**Entities**.

Elevator – class represents a physical elevator to be used by user to move between different floors. It has a max capacity of 12.

Floor – class represents a floor which has a maximum capacity of 100.

User – class represent a user who take an elevator moves from between floors. It has a name, start and target floor number, status, and the elevator assigned to him if available. If there is no elevator available, it will be put into a queue to wait till one is available.

RushHours – enum class indicates time intervals when there is 300 capacity limit

Status – enum class indicate when user is waiting, in-transit or off the elevator

**Tables:**

create table **elevator** (id integer generated by default as identity, capacity\_left integer not null, start integer not null, stop integer not null, primary key (id))

create table **floor** (id integer generated by default as identity, capacity\_left integer not null, number integer not null, primary key (id))

create table **user** (id integer generated by default as identity, elevator\_id integer not null, name varchar(255), start integer not null, status integer, stop integer not null, primary key (id))

The state of each entity above is stored in the tables.

**Database Access Objects:**

The following database access methods are generated automatically by spring-jpa framework for each above entities.

ElevatorRepository

FloorRepository

UserRepository

findAll()

findAll(Sort)

findAll(Iterable<ID>)

save(Iterable<S>)

flush()

saveAndFlush(S)

deleteInBatch(Iterable<T>)

deleteAllInBatch()

getOne(ID)

findAll(Example<S>)

findAll(Example<S>, Sort)

**Running:**

1. import to project to eclipse
2. run ‘mvn clean install package’ from project
3. run ‘java –jar target/elevator-0.1.0.jar’ from command line

or from eclipse by running hello.Application as a standalone java application.

**Testing**:

Check status:

<http://localhost:8080/checkStatus/mark>

Create a new request:

<http://localhost:8080/createRequest/peter?from=1&to=3>

Change the parameter as necessary

**Assumption:**

The application is fully functioning as expected. However, the backend is not fully implemented. It is returning hard-coded data at this time.