

# Final Project Work

## Git Repo with all Java Code:

<https://github.com/Christinalytle/movieAPI/tree/main/src/main/java/com/promineotech/movieApi>

```
Auditorium.java
1 package com.promineotech.movieApi.entity;
2
3 import java.util.Set;
4
12
13 @Entity
14 public class Auditorium {
15
16     private Long auditoriumId;
17     private Long auditoriumNumber;
18
19     @JsonIgnore
20     private Set<Seat> seats;
21
22     @JsonIgnore
23     private Set<Screening> screenings;
24
25     //Primary Key
26     @Id
27     @GeneratedValue(strategy = GenerationType.AUTO)
28     public Long getAuditoriumId() {
29         return auditoriumId;
30     }
31
32     public void setAuditoriumId(Long auditoriumId) {
33         this.auditoriumId = auditoriumId;
34     }
35
36     public Long getAuditoriumNumber() {
37         return auditoriumNumber;
38     }
39
40     public void setAuditoriumNumber(Long auditoriumNumber) {
41         this.auditoriumNumber = auditoriumNumber;
42     }
43
44     //Foreign Key is seatId
45     @OneToMany(mappedBy = "auditorium")
46     public Set<Seat> getSeats() {
47         return seats;
48     }
49
50     public void setSeats(Set<Seat> seats) {
51         this.seats = seats;
52     }
53
54     //Foreign Key is screeningId
55     @OneToMany(mappedBy = "auditorium")
56     public Set<Screening> getScreenings() {
57         return screenings;
58     }
59
60     public void setScreenings(Set<Screening> screenings) {
61         this.screenings = screenings;
62     }
63
64 }
65
```

```
Credentials.java
1 package com.promineotech.movieApi.entity;
2
3 public class Credentials {
4
5     private String email;
6     private String password;
7
8
9     public String getPassword() {
10         return password;
11     }
12
13     public void setPassword(String password) {
14         this.password = password;
15     }
16
17     public String getEmail() {
18         return email;
19     }
20
21     public void setEmail(String email) {
22         this.email = email;
23     }
24
25 }
26
```

Screening.java

```

1 package com.promineotech.movieApi.entity;
2
3 import java.util.Set;
4
5
6
7
8
9
10
11
12
13
14 @Entity
15 public class Screening {
16
17     private Long screeningId;
18     private Movie movies;
19     private Auditorium auditoriums;
20     private String time;
21
22     @JsonIgnore
23     private Set<Reservation> reservations;
24
25     //Primary Key
26     @Id
27     @GeneratedValue(strategy = GenerationType.AUTO)
28     public Long getScreeningId() {
29         return screeningId;
30     }
31     public void setScreeningId(Long screeningId) {
32         this.screeningId = screeningId;
33     }
34
35     //Foreign Key movieId
36     @ManyToOne
37     @JoinColumn(name = "movieId")
38     public Movie getMovie() {
39         return movies;
40     }
41     public void setMovie(Movie movies) {
42         this.movies = movies;
43     }
44
45     //Foreign Key auditoriumId
46     @ManyToOne
47     @JoinColumn(name = "auditoriumId")
48     public Auditorium getAuditorium() {
49         return auditoriums;
50     }
51     public void setAuditorium(Auditorium auditoriums) {
52         this.auditoriums = auditoriums;
53     }
54
55     public String getTime() {
56         return time;
57     }
58     public void setTime(String time) {
59         this.time = time;
60     }
61
62     //A reservation can have many screenings
63     @OneToMany(mappedBy = "screening")
64     public Set<Reservation> getReservation() {
65         return reservations;
66     }
67     public void setReservation(Set<Reservation> reservations) {
68         this.reservations = reservations;
69     }
70
71 }
72
73
74
75
76
77
78
79
80

```

Reservation.java

```

1 package com.promineotech.movieApi.entity;
2
3 import java.util.Set;
4
5
6
7
8
9
10
11
12
13
14 @Entity
15 public class Reservation {
16
17     private Long reservationId;
18
19     private Customer customer;
20
21     private Screening screenings;
22
23     private Set<Seat> seats;
24
25     private double reservationAmount;
26
27     //Primary Key
28     @Id
29     @GeneratedValue(strategy = GenerationType.AUTO)
30     public Long getReservationId() {
31         return reservationId;
32     }
33     public void setReservationId(Long reservationId) {
34         this.reservationId = reservationId;
35     }
36
37     //Foreign Key screeningId
38     @ManyToOne
39     @JoinColumn(name = "screeningId")
40     public Screening getScreening() {
41         return screenings;
42     }
43     public void setScreening(Screening screenings) {
44         this.screenings = screenings;
45     }
46
47     //Foreign Key a set of seatIds
48     @ManyToMany(mappedBy = "reservations")
49     public Set<Seat> getSeats() {
50         return seats;
51     }
52     public void setSeats(Set<Seat> seats) {
53         this.seats = seats;
54     }
55
56     public double getReservationAmount() {
57         return reservationAmount;
58     }
59     public void setReservationAmount(double reservationAmount) {
60         this.reservationAmount = reservationAmount;
61     }
62
63 }
64
65
66
67
68
69
70
71
72
73
74
75
76
77
78
79
80

```

Movie.java

```

1 package com.promineotech.movieApi.entity;
2
3 import java.util.Set;
4
12 @Entity
13 public class Movie {
14
15     private Long movieId; //primary key
16     private String name;
17     private String description;
18
19     //Screenings can have many movies
20     @JsonIgnore
21     private Set<Screening> screenings;
22
23
24
25     @Id
26     @GeneratedValue(strategy = GenerationType.AUTO)
27     public Long getMovieId() {
28         return movieId;
29     }
30
31     public void setMovieId(Long movieId) {
32         this.movieId = movieId;
33     }
34
35     public String getName() {
36         return name;
37     }
38
39     public void setName(String name) {
40         this.name = name;
41     }
42
43     public String getDescription() {
44         return description;
45     }
46
47     public void setDescription(String description) {
48         this.description = description;
49     }
50
51     //Foreign key screeningId
52     @OneToMany(mappedBy = "movie")
53     public Set<Screening> getScreenings() {
54         return screenings;
55     }
56
57     public void setScreenings(Set<Screening> screenings) {
58         this.screenings = screenings;
59     }
60
61
62
63
64 }
65

```

Customer.java

```

1 package com.promineotech.movieApi.entity;
2
3 import java.util.Set;
4
14
15
16 @Entity
17 public class Customer {
18
19     private Long customerId;
20     private String hash; //password
21     private String email;
22
23     @JsonIgnore
24     private Set<Reservation> reservation;
25
26     //Primary Key
27     @Id
28     @GeneratedValue(strategy = GenerationType.AUTO)
29     public Long getCustomerId() {
30         return customerId;
31     }
32
33     public void setCustomerId(Long customerId) {
34         this.customerId = customerId;
35     }
36
37     //Hash = password
38     @Column(name="Password")
39     public String getHash() {
40         return hash;
41     }
42
43     public void setHash(String hash) {
44         this.hash = hash;
45     }
46
47     public String getEmail() {
48         return email;
49     }
50
51     public void setEmail(String email) {
52         this.email = email;
53     }
54
55     //Foreign Key to reservationId
56     @OneToMany(mappedBy = "customer")
57     public Set<Reservation> getReservation() {
58         return reservation;
59     }
60
61     public void setReservation(Set<Reservation> reservation) {
62         this.reservation = reservation;
63     }
64
65
66
67
68
69     //Foreign key customerId
70     @ManyToOne
71     @JoinColumn(name = "customerId")
72     public Customer getCustomer() {
73         return customer;
74     }
75     public void setCustomer(Customer customer) {
76         this.customer = customer;
77     }
78
79
80 }
81

```

```

1 package com.promineotech.movieApi.entity;
2
3 import java.util.Set;
18
19
20 @Entity
21 public class Seat {
22
23     private Long seatId;
24     private Long seatNumber;
25     private double seatPrice;
26     private String rowName;
27
28     private Auditorium auditorium;
29
30     @JsonIgnore
31     private Set<Reservation> reservations;
32
33     //Primary Key
34     @Id
35     @GeneratedValue(strategy = GenerationType.AUTO)
36     public Long getSeatId() {
37         return seatId;
38     }
39
40     public void setSeatId(Long seatId) {
41         this.seatId = seatId;
42     }
43
44     public Long getSeatNumber() {
45         return seatNumber;
46     }
47
48     public void setSeatNumber(Long seatNumber) {
49         this.seatNumber = seatNumber;
50     }
51
52     public String getRowName() {
53         return rowName;
54     }
55
56     public void setRowName(String rowName) {
57         this.rowName = rowName;
58     }
59
60     public double getSeatPrice() {
61         return seatPrice;
62     }
63
64     public void setSeatPrice(double seatPrice) {
65         this.seatPrice = seatPrice;
66     }
67
68     //Foreign Key auditoriumID
69     @ManyToOne
70     @JoinColumn(name = "auditoriumId")
71     public Auditorium getAuditorium() {
72         return auditorium;
73     }
74
75     public void setAuditorium(Auditorium auditorium) {
76         this.auditorium = auditorium;
77     }
78
79     /*This ManyToMany table is used to connect a set of seats to a reservation number. This will
80     * allow customers to pick a set of seats that will connect to their reservation ID. */
81
82     @JsonIgnoreProperties({"hibernateLazyInitializer", "handler", "client", "clientMatter"})
83     @JsonFormat(with = JsonFormat.Feature.ACCEPT_SINGLE_VALUE_AS_ARRAY)
84     @ManyToMany(fetch = FetchType.LAZY)
85     @JoinTable(name = "seats_reserved",
86               joinColumns = @JoinColumn(name = "seatId", referencedColumnName="seatId"),
87               inverseJoinColumns = @JoinColumn(name = "reservationId", referencedColumnName="reservationId"))
88     public Set<Reservation> getReservations() {
89         return reservations;
90     }
91
92     public void setReservations(Set<Reservation> reservations) {
93         this.reservations = reservations;
94     }
95 }

```

ReservationDto.java

```
1 package com.promineotech.movieApi.entity;
2
3 import java.util.Set;
4
5 public class ReservationDto {
6
7     //The DTO is used to input variables in PostMan so they can connect to the right object
8
9     private Long customerId;
10    private Long screeningId;
11    private Set<Long> seatIds;
12    private double reservationAmount;
13
14    public Long getCustomerId() {
15        return customerId;
16    }
17    public void setCustomerId(Long customerId) {
18        this.customerId = customerId;
19    }
20
21    public Long getScreeningId() {
22        return screeningId;
23    }
24    public void setScreeningId(Long screeningId) {
25        this.screeningId = screeningId;
26    }
27
28    public Set<Long> getSeatIds() {
29        return seatIds;
30    }
31    public void setSeatIds(Set<Long> seatIds) {
32        this.seatIds = seatIds;
33    }
34
35    public double getReservationAmount() {
36        return reservationAmount;
37    }
38    public void setReservationAmount(double reservationAmount) {
39        this.reservationAmount = reservationAmount;
40    }
41
42 }
43
44
```

ScreeningDto.java

```
1 package com.promineotech.movieApi.entity;
2
3
4 public class ScreeningDto {
5
6     private Long movieId;
7     private Long auditoriumId;
8     private String time;
9
10
11    //The DTO is used to input variables in PostMan so they can connect to the right object
12
13    public Long getAuditoriumId() {
14        return auditoriumId;
15    }
16    public void setAuditoriumId(Long auditoriumId) {
17        this.auditoriumId = auditoriumId;
18    }
19
20
21    public Long getMovieId() {
22        return movieId;
23    }
24    public void setMovieId(Long movieId) {
25        this.movieId = movieId;
26    }
27
28
29    public String getTime() {
30        return time;
31    }
32    public void setTime(String time) {
33        this.time = time;
34    }
35
36 }
37
```

AuditoriumRepository.java

```
1 package com.promineotech.movieApi.repository;
2
3 import org.springframework.data.jpa.repository.JpaRepository;
4
5 import com.promineotech.movieApi.entity.Auditorium;
6
7
8 public interface AuditoriumRepository extends JpaRepository <Aud
9 }
10
11
```

MovieRepository.java

```
1 package com.promineotech.movieApi.repository;
2
3 import org.springframework.data.jpa.repository.
4 JpaRepository;
5
6 import com.promineotech.movieApi.entity.Movie;
7
8 public interface MovieRepository extends JpaRepository
9 <Movie, Long> {
10
11 }
12
```

ScreeningRepository.java

```
1 package com.promineotech.movieApi.repository;
2
3 import org.springframework.data.jpa.repository.
4 JpaRepository;
5
6 import com.promineotech.movieApi.entity.Screening;
7
8 public interface ScreeningRepository extends JpaRepository
9 <Screening, Long> {
10
11 }
12
```

CustomerRepository.java

```
1 package com.promineotech.movieApi.repository;
2
3 import org.springframework.data.jpa.repository.JpaRepository;
4
5 import com.promineotech.movieApi.entity.Customer;
6
7
8 public interface CustomerRepository extends JpaRepository
9 <Customer, Long> {
10
11     public Customer findByEmail(String email);
12
13 }
14
```

ReservationRepository.java

```
1 package com.promineotech.movieApi.repository;
2
3 import org.springframework.data.jpa.repository.JpaRepository;
4
5 import com.promineotech.movieApi.entity.Reservation;
6
7 public interface ReservationRepository extends
8 JpaRepository <Reservation, Long> {
9
10 }
11
```

SeatRepository.java

```
1 package com.promineotech.movieApi.repository;
2
3 import org.springframework.data.jpa.repository.JpaRepository;
4
5 import com.promineotech.movieApi.entity.Seat;
6
7 public interface SeatRepository extends JpaRepository
8 <Seat, Long> {
9
10 }
11
```



```

1 package com.promineotech.movieApi.service;
2
3 import org.apache.logging.log4j.LogManager;
4
5
6
7
8
9
10
11 @Service
12 public class AuditoriumService {
13
14     private static final Logger Logger = LogManager.getLogger(AuditoriumService.class);
15
16     @Autowired
17     private AuditoriumRepository repo;
18
19     //GET all auditoriums
20     public Iterable<Auditorium> getAuditoriums(){
21         return repo.findAll();
22     }
23
24     //POST (create) an auditorium
25     public Auditorium createAuditorium (Auditorium auditorium) {
26         return repo.save(auditorium);
27     }
28
29     //PUT (update) an auditorium
30     public Auditorium updateAuditorium (Auditorium auditorium, Long id) throws Exception {
31         try {
32             Auditorium oldAuditorium = repo.findById(id).orElseThrow();
33             oldAuditorium.setAuditoriumNumber(auditorium.getAuditoriumNumber());
34             return repo.save(oldAuditorium);
35         } catch (Exception e) {
36             Logger.error("Exception occurred while trying to update auditorium "+ id, e);
37             throw new Exception ("Unable to update auditorium");
38         }
39     }
40
41     //DELETE an auditorium
42     public void deleteAuditorium(Long id) throws Exception {
43         try {
44             repo.deleteById(id);
45         } catch (Exception e) {
46             Logger.error("Exception occurred while trying to delete movie "+ id, e);
47             throw new Exception ("Unable to delete auditorium");
48         }
49     }
50
51 }
52
53

```

AuthService.java

```
1 package com.promineotech.movieApi.service;
2
3 import javax.naming.AuthenticationException;
4
5 @Service
6 public class AuthService {
7
8     @Autowired
9     private CustomerRepository customerRepo;
10
11     //POST (create) a new customer
12     public Customer register(Credentials cred) throws AuthenticationException {
13         Customer customer = new Customer();
14         customer.setEmail(cred.getEmail());
15         customer.setHash(BCrypt.hashpw(cred.getPassword(), BCrypt.gensalt()));
16         try {
17             customerRepo.save(customer);
18             return customer;
19         } catch (DataIntegrityViolationException e) {
20             throw new AuthenticationException("Username not available.");
21         }
22     }
23
24     //POST (login) login a customer
25     public Customer login (Credentials cred) throws AuthenticationException {
26         Customer foundCustomer = customerRepo.findByEmail(cred.getEmail());
27         if (foundCustomer != null && BCrypt.checkpw(cred.getPassword(), foundCustomer.getHash())) {
28             return foundCustomer;
29         }
30         throw new AuthenticationException("Incorrect username or password");
31     }
32 }
33
34
35
36
37
38
39
40
41
42
43
44
45
46
47 }
48
```



```
1 package com.promineotech.movieApi.service;
2
3 import org.apache.logging.log4j.LogManager;
4
10
11 @Service
12 public class MovieService {
13
14     private static final Logger Logger = LogManager.getLogger(MovieService.class);
15
16     @Autowired
17     private MovieRepository repo;
18
19     //GET (find) a movie by its id
20     public Movie getMovieById (Long id) throws Exception {
21         try {
22             return repo.findById(id).orElseThrow();
23         } catch (Exception e) {
24             Logger.error("Exception occurred while trying to find movie "+ id, e);
25             throw e;
26         }
27     }
28
29     //POST (create) a movie
30     public Movie createMovie (Movie movie) {
31         return repo.save(movie);
32     }
33
34     //GET all movies
35     public Iterable<Movie> getMovies() {
36         return repo.findAll();
37     }
38
39     //PUT (update) a movie by its id
40     public Movie updateMovie (Movie movie, Long id) throws Exception {
41         try {
42             Movie oldMovie = repo.findById(id).orElseThrow();
43             oldMovie.setDescription(movie.getDescription());
44             oldMovie.setName(movie.getName());
45             return repo.save(oldMovie);
46         } catch (Exception e) {
47             Logger.error("Exception occurred while trying to update movie " + id, e);
48             throw new Exception ("Unable to update product.");
49         }
50     }
51
52     //DELETE a movie by its id
53     public void deleteMovie(Long id) throws Exception {
54         try {
55             repo.deleteById(id);
56         } catch (Exception e) {
57             Logger.error("Exception occurred while trying to delete movie "+id, e);
58             throw new Exception("Unable to delete product.");
59         }
60     }
61 }
62
63
```

```
2
3 import java.util.HashSet;
25
26 @Service
27 public class ReservationService {
28
29     private static final Logger Logger = LogManager.getLogger(ReservationService.class);
30
31     @Autowired
32     private ReservationRepository resRepo;
33
34     @Autowired
35     private ScreeningRepository screeningRepo;
36
37     @Autowired
38     private SeatRepository seatRepo;
39
40     @Autowired
41     private CustomerRepository customerRepo;
42
43
44     //GET all reservations
45     public Iterable<Reservation> getReservation() {
46         return resRepo.findAll();
47     }
48
49     //GET reservation by its id
50     public Reservation getReservation (Long id) {
51         return resRepo.findById(id).orElseThrow();
52     }
53
54     //POST (create) a reservation.
55     //Find the screening number and seat numbers
56     //Find the customerId from the URL
57     //Get the total amount by looping thru the seats and adding their prices together
58     //Add the seats to the reservation to satisfy the ManyToMany table
59     //Save the reservation
60     //Convert reservation to a ReservationResponse (to keep it from doing an infinite recursion)
61     //Return a ReservationResponse
62     public ReservationResponse startReservation(Set<Long> seatIds, Long screeningId , Long customerId) throws AuthenticationException {
63         try {
64             Reservation reservation = new Reservation();
65             Screening screen = screeningRepo.findById(screeningId).orElseThrow();
66             Customer customer = customerRepo.findById(customerId).orElseThrow();
67             reservation.setSeats(convertToSeatSet(seatRepo.findAllById(seatIds)));
68             reservation.setScreening(screen);
69             reservation.setCustomer(customer);
70             reservation.setReservationAmount(calculateReservationTotal(reservation.getSeats()));
71             addSeatsToReservation(reservation);
72             resRepo.save(reservation);
73             return convertToReservationResponse(reservation);
74         } catch (DataIntegrityViolationException e) {
75             Logger.error("Exception occurred while trying to create a screening");
76             throw new AuthenticationException("Seats not available");
77         }
78     }
79 }
```

```

79
80 //Delete the reservation
81 public void deleteReservation (Long reservationId) throws Exception {
82     try {
83         resRepo.deleteById(reservationId);
84     } catch (Exception e) {
85         Logger.error("Exception occurred while trying to delete reservation: " + reservationId, e);
86         throw new Exception("Unable to delete reservation.");
87     }
88 }
89
90 private ReservationResponse convertToReservationResponse(Reservation reservation) {
91     ReservationResponse response = new ReservationResponse();
92     Set<Seat> seats = reservation.getSeats();
93     response.setReservationId(reservation.getReservationId());
94     response.setCustomerEmail(reservation.getCustomer().getEmail());
95     response.setPrice(calculateReservationTotal(reservation.getSeats()));
96     response.setScreeningId(reservation.getScreening().getScreeningId());
97     response.setSeats(convertToSeatResponse(seats));
98     return response;
99 }
100
101 private Set<SeatResponse> convertToSeatResponse(Set<Seat> seats) {
102     Set<SeatResponse> seatList = new HashSet<SeatResponse>();
103     for(Seat seat : seats) {
104         SeatResponse response = new SeatResponse();
105         response.setRowName(seat.getRowName());
106         response.setSeatNumber(seat.getSeatNumber());
107         response.setSeatPrice(seat.getSeatPrice());
108         seatList.add(response);
109     }
110     return seatList;
111 }
112
113 private void addSeatsToReservation(Reservation reservation) {
114     Set<Seat> seats = reservation.getSeats();
115     for (Seat seat : seats) {
116         seat.getReservations().add(reservation);
117     }
118 }
119
120 private Set<Seat> convertToSeatSet(Iterable<Seat> iterable) {
121     Set<Seat> set = new HashSet<Seat>();
122     for (Seat seat : iterable) {
123         set.add(seat);
124     }
125     return set;
126 }
127
128 private double calculateReservationTotal(Set<Seat> seats) {
129     double total = 0;
130     for (Seat seat : seats) {
131         total += seat.getSeatPrice();
132     }
133     return total;
134 }
135 }

```

```

1  package com.promineotech.movieApi.service;
2
3  import org.apache.logging.log4j.LogManager;
14
15  @Service
16  public class ScreeningService {
17
18      private static final Logger Logger = LogManager.getLogger(ScreeningService.class);
19
20  @Autowired
21  private ScreeningRepository repo;
22
23  @Autowired
24  private AuditoriumRepository audRepo;
25
26  @Autowired
27  private MovieRepository movieRepo;
28
29  //GET a screening by its Id
30  public Screening getScreeningById (Long id) throws Exception {
31      try {
32          return repo.findById(id).orElseThrow();
33      } catch (Exception e) {
34          Logger.error("Exception occurred while trying to get Screenings", e);
35          throw e;
36      }
37  }
38
39  //GET all screenings
40  public Iterable<Screening> getScreenings() {
41      return repo.findAll();
42  }
43
44  //POST (create) a screening by finding the auditorium and movie ID, and saving
45  //it to the reservation while adding a time.
46  public Screening createScreening (Long auditoriumId, Long movieId, String time) {
47      Screening screening = new Screening();
48      Auditorium aud = audRepo.findById(auditoriumId).orElseThrow();
49      Movie mov = movieRepo.findById(movieId).orElseThrow();
50      screening.setAuditorium(aud);
51      screening.setMovie(mov);
52      screening.setTime(time);
53      return repo.save(screening);
54  }
55
56
57  //PUT (update) a screening by its id
58  public Screening updateScreening (Screening screening, Long id) throws Exception {
59      try {
60          Screening oldScreening = repo.findById(id).orElseThrow();
61          oldScreening.setAuditorium(screening.getAuditorium());
62          oldScreening.setMovie(screening.getMovie());
63          oldScreening.setTime(screening.getTime());
64          return repo.save(oldScreening);
65      } catch (Exception e) {
66          Logger.error("Exception occurred while trying to update Screening " + id, e);
67          throw new Exception ("Unable to update product.");
68      }
69  }
70
71  //DELETE a screening by its id
72  public void deleteScreening (Long id) throws Exception {
73      try {
74          repo.deleteById(id);
75      } catch (Exception e) {
76          Logger.error("Exception occurred while trying to delete Screening " + id, e);
77          throw new Exception ("Unable to delete screening.");
78      }
79  }
80
81  }
82
83  }
84

```

```
1 package com.promineotech.movieApi.service;
2
3 import org.apache.logging.log4j.LogManager;
4
10
11 @Service
12 public class SeatService {
13
14     private static final Logger Logger = LogManager.getLogger(SeatService.class);
15
16     @Autowired
17     private SeatRepository repo;
18
19
20     //GET all seats
21     public Iterable<Seat> getSeats() {
22         return repo.findAll();
23     }
24
25     //POST (create) a seat
26     public Seat createSeat(Seat seat) {
27         return repo.save(seat);
28     }
29
30     //PUT (update) a seat by its id
31     public Seat updateSeat (Seat seat, Long id ) throws Exception {
32         try {
33             Seat oldSeat = repo.findById(id).orElseThrow();
34             oldSeat.setSeatNumber(seat.getSeatNumber());
35             oldSeat.setRowName(seat.getRowName());
36             oldSeat.setSeatPrice(seat.getSeatPrice());
37             return repo.save(oldSeat);
38         } catch (Exception e) {
39             Logger.error("Exception occurred while tryin to update seat " + id, e);
40             throw new Exception("Unable to update product.");
41         }
42     }
43
44     //DELETE a seat by its id
45     public void deleteSeat (Long id) throws Exception {
46         try {
47             repo.deleteById(id);
48         } catch (Exception e) {
49             Logger.error("Exception occurred while trying to delete seat: " + id, e);
50             throw new Exception("Unable to delete product.");
51         }
52     }
53
54 }
55
```



```
1 package com.promineotech.movieApi.controller;
2
3 import org.springframework.beans.factory.annotation.Autowired;
4
14 @RestController
15 @RequestMapping("/auditoriums")
16 public class AuditoriumController {
17
18     @Autowired
19     private AuditoriumService service;
20
21
22     @RequestMapping (method = RequestMethod.GET)
23     public ResponseEntity<Object> getAuditoriums() {
24         return new ResponseEntity<Object>(service.getAuditoriums(), HttpStatus.OK) ;
25     }
26
27     @RequestMapping (method=RequestMethod.POST)
28     public ResponseEntity<Object> createAuditorium(@RequestBody Auditorium auditorium) {
29         return new ResponseEntity<Object>(service.createAuditorium(auditorium), HttpStatus.CREATED);
30     }
31
32     @RequestMapping (value="/{id}", method = RequestMethod.PUT)
33     public ResponseEntity<Object> updateAuditorium (@RequestBody Auditorium auditorium, @PathVariable Long id) {
34         try {
35             return new ResponseEntity<Object>(service.updateAuditorium(auditorium, id), HttpStatus.OK);
36         } catch (Exception e) {
37             return new ResponseEntity<Object>("Unable to update product.", HttpStatus.BAD_REQUEST);
38         }
39     }
40
41     @RequestMapping (value = "/{id}", method = RequestMethod.DELETE)
42     public ResponseEntity<Object> deleteAuditorium(@PathVariable Long id) {
43         try {
44             service.deleteAuditorium(id);
45             return new ResponseEntity<Object>("Successfully deleted product with id: " + id, HttpStatus.OK);
46         } catch (Exception e) {
47             return new ResponseEntity<Object>("Unable to delete product.", HttpStatus.BAD_REQUEST);
48         }
49     }
50 }
51
52
```



CustomerController.java

```
1 package com.promineotech.movieApi.controller;
2
3 import javax.naming.AuthenticationException;
4
5
6 @RestController
7 @RequestMapping("/customers")
8 public class CustomerController {
9
10
11     @Autowired
12     private AuthService authService;
13
14
15     @RequestMapping (value = "/register", method = RequestMethod.POST)
16     public ResponseEntity<Object> register (@RequestBody Credentials cred) {
17         try {
18             return new ResponseEntity<Object>(authService.register(cred), HttpStatus.CREATED);
19         } catch (AuthenticationException e) {
20             return new ResponseEntity<Object>(e.getMessage(), HttpStatus.BAD_REQUEST);
21         }
22     }
23
24
25     @RequestMapping (value = "/login", method=RequestMethod.POST)
26     public ResponseEntity<Object> logIn(@RequestBody Credentials cred) {
27         try {
28             return new ResponseEntity<Object>(authService.login(cred), HttpStatus.OK);
29         } catch (AuthenticationException e) {
30             return new ResponseEntity<Object>(e.getMessage(), HttpStatus.UNAUTHORIZED);
31         }
32     }
33
34
35 }
36
37
38
39
40
41
42
43
44
45
46 }
47
```

```
1 package com.promineotech.movieApi.controller;
2
3 import org.springframework.beans.factory.annotation.Autowired;
4
14
15 @RestController
16 @RequestMapping("/movies")
17 public class MovieController {
18
19     @Autowired
20     private MovieService service;
21
22     @RequestMapping (method=RequestMethod.GET)
23     public ResponseEntity<Object> getMovies() {
24         return new ResponseEntity<Object>(service.getMovies(), HttpStatus.OK);
25     }
26
27     @RequestMapping (method = RequestMethod.POST)
28     public ResponseEntity<Object> createMovie(@RequestBody Movie movie) {
29         return new ResponseEntity<Object>(service.createMovie(movie), HttpStatus.CREATED);
30     }
31
32     @RequestMapping(value =("/{id}", method=RequestMethod.PUT)
33     public ResponseEntity<Object> updateMovie(@RequestBody Movie movie, @PathVariable Long id) {
34         try {
35             return new ResponseEntity<Object>(service.updateMovie(movie, id), HttpStatus.OK);
36         } catch (Exception e) {
37             return new ResponseEntity<Object>("Unable to update product.", HttpStatus.BAD_REQUEST);
38         }
39     }
40
41     @RequestMapping(value="/{id}", method=RequestMethod.DELETE)
42     public ResponseEntity<Object> deleteMovie (@PathVariable Long id) {
43         try {
44             service.deleteMovie(id);
45             return new ResponseEntity<Object>("Successfully delete movie with id: " + id, HttpStatus.OK);
46         } catch (Exception e) {
47             return new ResponseEntity<Object>("Unable to delete product.", HttpStatus.BAD_REQUEST);
48         }
49     }
50 }
51
52
```

ReservationController.java

```
1 package com.promineotech.movieApi.controller;
2
3 import org.springframework.beans.factory.annotation.Autowired;
14
15 @RestController
16 @RequestMapping("/customers/{customerId}/reservations")
17 public class ReservationController {
18
19     @Autowired
20     private ReservationService service;
21
22     @RequestMapping (method = RequestMethod.POST)
23     public ResponseEntity<Object> startReservation(@RequestBody ReservationDto res, @PathVariable Long customerId){
24         try {
25             return new ResponseEntity<Object>(service.startReservation(res.getSeatIds(), res.getScreeningId(), customerId),
26                 HttpStatus.CREATED);
27         } catch (Exception e) {
28             return new ResponseEntity<Object>(e, HttpStatus.BAD_REQUEST);
29         }
30     }
31
32
33
34
35 }
```

ScreeningController.java

```
1 package com.promineotech.movieApi.controller;
2
3 import org.springframework.beans.factory.annotation.Autowired;
4
5
6
7
8 @RestController
9 @RequestMapping("/screenings")
10 public class ScreeningController {
11
12     @Autowired
13     private ScreeningService service;
14
15
16
17     @RequestMapping (method=RequestMethod.GET)
18     public ResponseEntity<Object> geScreenings() {
19         return new ResponseEntity<Object>(service.getScreenings(), HttpStatus.OK);
20     }
21
22     @RequestMapping (method = RequestMethod.POST)
23     public ResponseEntity<Object> createScreening(@RequestBody ScreeningDto screeningDto) {
24         return new ResponseEntity<Object>(service.createScreening(screeningDto.getAuditoriumId(), screeningDto.getMovieId(), screeningDto.getTime()), HttpStatus.CREATED);
25     }
26
27     @RequestMapping(value =("/{id})", method=RequestMethod.PUT)
28     public ResponseEntity<Object> updateScreening(@RequestBody Screening screening, @PathVariable Long id) {
29         try {
30             return new ResponseEntity<Object>(service.updateScreening(screening, id), HttpStatus.OK);
31         } catch (Exception e) {
32             return new ResponseEntity<Object>("Unable to update product.", HttpStatus.BAD_REQUEST);
33         }
34     }
35
36     @RequestMapping(value="/{id}", method=RequestMethod.DELETE)
37     public ResponseEntity<Object> deleteScreening (@PathVariable Long id) {
38         try {
39             service.deleteScreening(id);
40             return new ResponseEntity<Object>("Successfully deleted screening with id: " + id, HttpStatus.OK);
41         } catch (Exception e) {
42             return new ResponseEntity<Object>("Unable to delete screening", HttpStatus.BAD_REQUEST);
43         }
44     }
45 }
46
47
48 }
```

SeatController.java

```
1 package com.promineotech.movieApi.controller;
2
3 import org.springframework.beans.factory.annotation.Autowired;
4
14
15 @RestController
16 @RequestMapping("/seats")
17 public class SeatController {
18
19     @Autowired
20     private SeatService service;
21
22     @RequestMapping(method = RequestMethod.GET)
23     public ResponseEntity<Object> getSeats() {
24         return new ResponseEntity<Object>(service.getSeats(), HttpStatus.OK);
25     }
26
27     @RequestMapping(method=RequestMethod.POST)
28     public ResponseEntity<Object> createSeat (@RequestBody Seat seat) {
29         return new ResponseEntity<Object>(service.createSeat(seat), HttpStatus.CREATED);
30     }
31
32     @RequestMapping(value =("/{id})", method=RequestMethod.PUT)
33     public ResponseEntity<Object> updateSeat (@RequestBody Seat seat, @PathVariable Long id) {
34         try {
35             return new ResponseEntity<Object>(service.updateSeat(seat, id), HttpStatus.OK);
36         } catch (Exception e) {
37             return new ResponseEntity<Object>("Unable to update product.", HttpStatus.BAD_REQUEST);
38         }
39     }
40 }
41
42
```

App.java

```
1 package com.promineotech.movieApi;
2
3 import org.springframework.boot.SpringApplication;
4
6
7 @ComponentScan("com.promineotech.movieApi")
8 @SpringBootApplication
9 public class App
10 {
11     public static void main( String[] args )
12     {
13         SpringApplication.run(App.class, args);
14     }
15 }
16
```

**POST** ▼ http://localhost:8080/auditoriums

Params Authorization Headers (8) **Body** ● Pre-req

☐ none ☐ form-data ☐ x-www-form-urlencoded ☒ raw

```
1 [
2   "auditoriumNumber": 2
3 ]
4
```

**Body** Cookies Headers (5) Test Results

Pretty Raw Preview Visualize JSON ▼

```
1 [
2   ... "auditoriumId": 9,
3   ... "auditoriumNumber": 2
4 ]
```

**GET** ▼ http://localhost:8080/auditoriums

Params Authorization Headers (8) **Body** ● Pre-req

☐ none ☐ form-data ☐ x-www-form-urlencoded ☒ raw

```
1 [
2   "auditoriumNumber": 2
3 ]
4
```

**Body** Cookies Headers (5) Test Results

Pretty Raw Preview Visualize JSON ▼

```
1 [
2   ... {
3     ... "auditoriumId": 1,
4     ... "auditoriumNumber": 1
5   },
6   ... {
7     ... "auditoriumId": 9,
8     ... "auditoriumNumber": 2
9   }
10 ]
```

**PUT** ▼ http://localhost:8080/auditoriums/9

Params Authorization Headers (8) **Body** ● Pre-req

☐ none ☐ form-data ☐ x-www-form-urlencoded ☒ raw

```
1 [
2   "auditoriumNumber": 4
3 ]
4
```

**Body** Cookies Headers (5) Test Results

Pretty Raw Preview Visualize JSON ▼

```
1 [
2   ... "auditoriumId": 9,
3   ... "auditoriumNumber": 4
4 ]
```

**DELETE** ▼ http://localhost:8080/auditoriums/9

Params Authorization Headers (8) **Body** ● Pre-request Sc

☐ none ☐ form-data ☐ x-www-form-urlencoded ☒ raw ☐ bi

```
1 [
2   "auditoriumNumber": 4
3 ]
4
```

**Body** Cookies Headers (5) Test Results

Pretty Raw Preview Visualize Text ▼

1 Successfully deleted product with id: 9



POST

http://localhost:8080/customers/register

Params

Authorization

Headers (8)

Body

Pre-request Script

Tests

Settings

none

form-data

x-www-form-urlencoded

raw

binary

GraphQL

JSON

1

{

2

"email": "lytle456@gmail.com",

3

"password": "123abc"

4

}

5

Body

Cookies

Headers (5)

Test Results

Pretty

Raw

Preview

Visualize

JSON

1

...

2

"customerId": 10,

3

"hash": "\$2a\$10\$hvb84Z912Prfm2VnxjBUKuuk6ArGiHosSJMmxdsEr4f8asPIEl0TC",

4

"email": "lytle456@gmail.com"

5

POST

http://localhost:8080/customers/login

Params

Authorization

Headers (8)

Body

Pre-request Script

Tests

Settings

none

form-data

x-www-form-urlencoded

raw

binary

GraphQL

JSON

1

{

2

"email": "lytle456@gmail.com",

3

"password": "123abc"

4

}

5

Body

Cookies

Headers (5)

Test Results

Pretty

Raw

Preview

Visualize

JSON

1

...

2

"customerId": 10,

3

"hash": "\$2a\$10\$hvb84Z912Prfm2VnxjBUKuuk6ArGiHosSJMmxdsEr4f8asPIEl0TC",

4

"email": "lytle456@gmail.com"

5

POST

http://localhost:8080/movies

Params

Authorization

Headers (8)

Body

Pre-request Script

none

form-data

x-www-form-urlencoded

raw

binary

1

2

3

4

5

```
1 {
2   "name": "Die Hard",
3   "description": "Action Christmas Movie"
4 }
5
```

Body

Cookies

Headers (5)

Test Results

Pretty

Raw

Preview

Visualize

JSON

1

2

3

4

5

```
1 {
2   ... "movieId": 11,
3   ... "name": "Die Hard",
4   ... "description": "Action Christmas Movie"
5 }
```

PUT

http://localhost:8080/movies/11

Params

Authorization

Headers (8)

Body

Pre-request Script

none

form-data

x-www-form-urlencoded

raw

binary

1

2

3

4

5

```
1 {
2   "name": "Die Hard",
3   "description": "The Best Action Christmas Movie"
4 }
5
```

Body

Cookies

Headers (5)

Test Results

Pretty

Raw

Preview

Visualize

JSON

1

2

3

4

5

```
1 {
2   ... "movieId": 11,
3   ... "name": "Die Hard",
4   ... "description": "The Best Action Christmas Movie"
5 }
```

GET

http://localhost:8080/movies

Params

Authorization

Headers (8)

Body

Pre-request Script

none

form-data

x-www-form-urlencoded

raw

binary

1

2

3

4

5

```
1 {
2   "name": "Die Hard",
3   "description": "Action Christmas Movie"
4 }
5
```

Body

Cookies

Headers (5)

Test Results

Pretty

Raw

Preview

Visualize

JSON

1

2

3

4

5

6

7

8

9

10

11

12

```
1 [
2   ... {
3     ... "movieId": 3,
4     ... "name": "titanic",
5     ... "description": "ship sinks"
6   },
7   ... {
8     ... "movieId": 11,
9     ... "name": "Die Hard",
10    ... "description": "Action Christmas Movie"
11  }
12 ]
```

DELETE

http://localhost:8080/movies/11

Params

Authorization

Headers (8)

Body

Pre-request Script

none

form-data

x-www-form-urlencoded

raw

binary

1

2

3

4

5

```
1 {
2   "name": "Die Hard",
3   "description": "The Best Action Christmas Movie"
4 }
5
```

Body

Cookies

Headers (5)

Test Results

Pretty

Raw

Preview

Visualize

Text

1

Successfully delete movie with id: 11

POST

http://localhost:8080/seats

GET

http://localhost:8080/seats

Params

Authorization

Headers (8)

Body

Pre-reques

Params

Authorization

Headers (8)

Body

Pre-request Script

T

none

form-data

x-www-form-urlencoded

raw

none

form-data

x-www-form-urlencoded

raw

binary

```
1 {
2   "rowName": "B",
3   "seatNumber": 1,
4   "seatPrice": 5.99,
5   "auditorium": {
6     "auditoriumId": 1 }
7 }
8
```

```
1 {
2   "rowName": "B",
3   "seatNumber": 1,
4   "seatPrice": 5.99,
5   "auditorium": {
6     "auditoriumId": 1 }
7 }
8
```

Body

Cookies

Headers (5)

Test Results

Body

Cookies

Headers (5)

Test Results

Pretty

Raw

Preview

Visualize

JSON

```
1 {
2   .... "seatId": 12,
3   .... "seatNumber": 1,
4   .... "seatPrice": 5.99,
5   .... "rowName": "B",
6   .... "auditorium": {
7     .... "auditoriumId": 1,
8     .... "auditoriumNumber": null
9   },
10  .... "reservations": null
11 }
```

Pretty

Raw

Preview

Visualize

JSON

```
1 {
2   .... {
3     .... "seatId": 4,
4     .... "seatNumber": 1,
5     .... "seatPrice": 5.99,
6     .... "rowName": "A",
7     .... "auditorium": {
8       .... "auditoriumId": 1,
9       .... "auditoriumNumber": 1
10    },

```

POST http://localhost:8080/screenings

Params Authorization Headers (8) **Body** Pre-req

none form-data x-www-form-urlencoded **raw**

```
1 {
2   ... "auditoriumId": 1,
3   ... "movieId": 3,
4   ... "time": "6:00pm"
5 }
6
```

Body Cookies Headers (5) Test Results

Pretty Raw Preview Visualize JSON

```
1 {
2   ... "screeningId": 13,
3   ... "time": "6:00pm",
4   ... "reservation": null,
5   ... "auditorium": {
6     ... "auditoriumId": 1,
7     ... "auditoriumNumber": 1
8   },
9   ... "movie": {
10    ... "movieId": 3,
11    ... "name": "titanic",
12    ... "description": "ship sinks"
13  }
14 }
```

GET http://localhost:8080/screenings

Params Authorization Headers (8) **Body** Pre-request Script Tests Settings

none form-data x-www-form-urlencoded **raw** binary GraphQL JSON

```
1 {
2   ... "auditoriumId": 1,
3   ... "movieId": 3,
4   ... "time": "6:00pm"
5 }
6
```

Body Cookies Headers (5) Test Results

Pretty Raw Preview Visualize JSON

Status: 200 OK

```
1 {
2   ... {
3     ... "screeningId": 7,
4     ... "time": "2:00pm",
5     ... "reservation": [
6       ... {
7         ... "reservationId": 8,
8         ... "customer": {
9           ... "customerId": 2,
10          ... "hash": "$2a$10$gl7FH0M2fR.5E4Kov4kIYe5HPN3RhAYLj04ZGrKdT.EIpls7v4QfK",
11          ... "email": "lytle@gmail.com"
12        },
13        ... "seats": [
14          ... {
15            ... "seatId": 6,
16            ... "seatNumber": 3,
17            ... "seatPrice": 5.99,
18            ... "rowName": "A",
19            ... "auditorium": {
20              ... "auditoriumId": 1,
21              ... "auditoriumNumber": 1
22            }
15          }
16        ]
17      }
18    ]
19  }
20 }
```

POST



http://localhost:8080/customers/10/reservations

Params

Authorization

Headers (8)

**Body**

Pre-request Script

☐ none ☐ form-data ☐ x-www-form-urlencoded ☒ raw ☐ binary

```
1 {  
2   ... "seatIds": [4,5,6],  
3   ... "screeningId": 13  
4 }
```

**Body**

Cookies

Headers (5)

Test Results

Pretty

Raw

Preview

Visualize

JSON



```
1 {  
2   ... "reservationId": 14,  
3   ... "customerEmail": "lytle456@gmail.com",  
4   ... "seats": [  
5     ... {  
6       ... "seatNumber": 3,  
7       ... "seatPrice": 5.99,  
8       ... "rowName": "A"  
9     },  
10    ... {  
11      ... "seatNumber": 2,  
12      ... "seatPrice": 5.99,  
13      ... "rowName": "A"  
14    },  
15    ... {  
16      ... "seatNumber": 1,  
17      ... "seatPrice": 5.99,  
18      ... "rowName": "A"  
19    }  
20  ],  
21  ... "screeningId": 13,  
22  ... "price": 17.97  
23 }
```