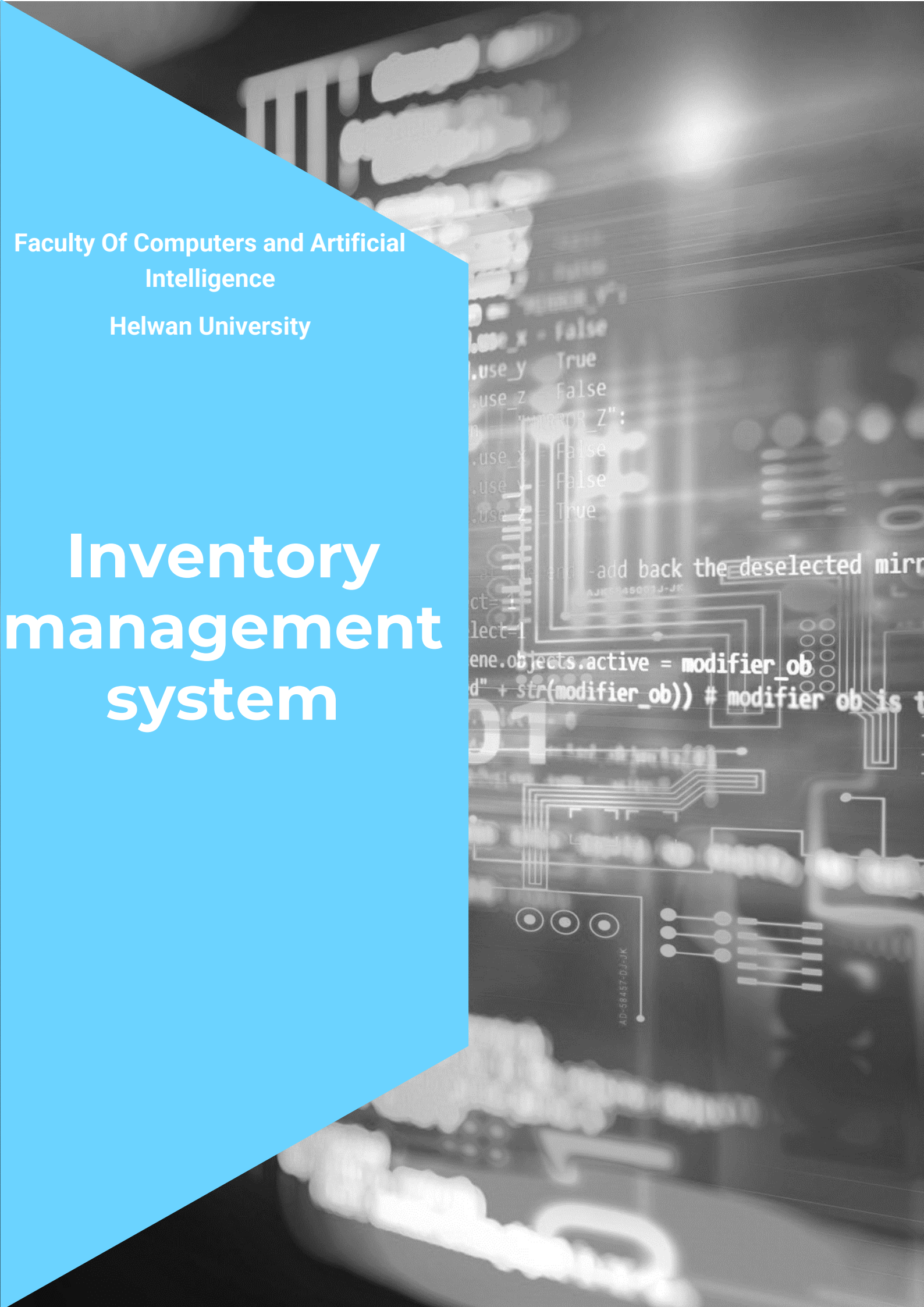
Faculty Of Computers and Artificial Intelligence

Helwan University

Inventory management system



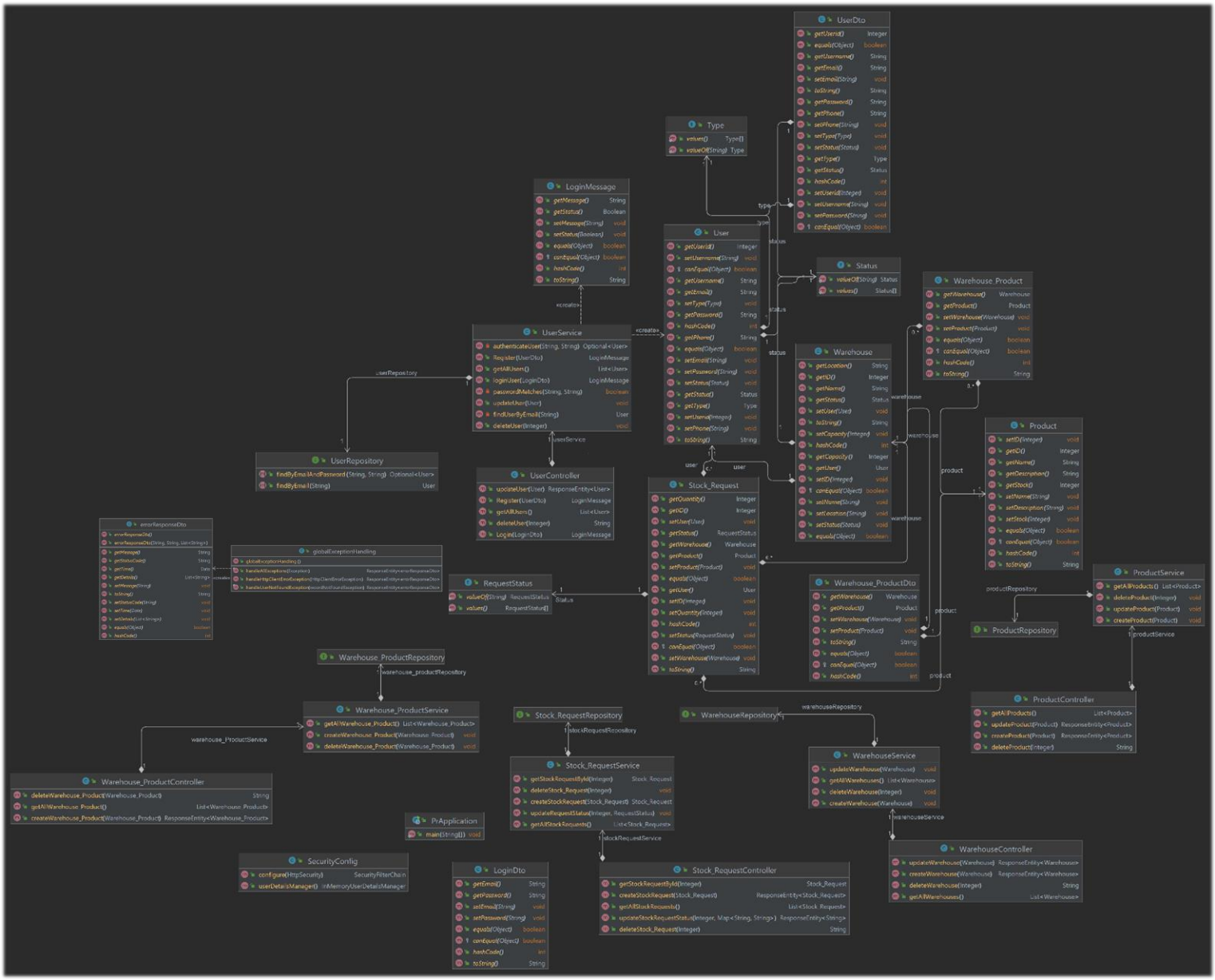
Inventory management system

TABLE OF CONTENTS

Functional Requirements:

FUNCTION NAME	FUNCTION DESCRIPTION
Register	Allows Admins/Supervisors to Create a new account
Login	Allows Admins/Supervisors to Login to the Account they created earlier
getAllUsers	Allows Admins to fetch all users
updateUser	Allows Admins to update user's info
deleteUser	Allows Admins to delete a user
getAllProducts	Allows Admins/Supervisors to fetch all products
createProduct	Allows Admins to Create a new product
updateProduct	Allows Admins to update product's info
deleteProduct	Allows Admins to delete a product
createStockRequest	Allows Supervisors to request stock
getAllStockRequests	Allows Admins to fetch all requests
updateRequestStatus	Allows Admins to update the request status
deleteStock_Request	Allows Admins to delete a request
getAllWarehouse_Product	Allows Admins to fetch all warehouses and their products
createWarehouse_Product	Allows Admins to create a specific product to a specific warehouse
deleteWarehouse_Product	Allows Admin to delete a specific warehouse with its products
getAllWarehouses	Allows Admins to fetch all warehouses
createWarehouse	Allows Admins to create a warehouse
updateWarehouse	Allows Admins to update warehouse's info
deleteWarehouse	Allows Admins to delete a warehouse

Class Diagram:



OCL (Object Constraint Language):

Classes:

- **Product**
- **User**
- **Stock_Request**
- **Warehouse**
- **Warehouse_Product**

Product Class:

Context: Product

invariant: self.ID.size() <= 11

& not self.ID.oclIsUndefined()

invariant: not self.Name.oclIsUndefined()

invariant: not self.Stock.oclIsUndefined()

& self.Stock.size() <=11

```java
public class Product {
    no usages
    @Id
    @GeneratedValue(strategy = GenerationType.IDENTITY)
    @Column(length = 11, nullable = false)
    private Integer ID;
    no usages
    @Column(nullable = false)
    private String Name;
    no usages
    @Column(columnDefinition = "TEXT")
    private String Description;
    no usages
    @Column(nullable = false, length = 11)
    private Integer Stock;

}
```

User Class:

Context: User

Invariant: not self.ID.oclIsUndefined()

& self.ID.size() <= 11

invariant: self.username.size() <= 15

& User.allInstances()->isUnique(username)

Invariant: User.allInstances()->isUnique(email)

&self.email.matches('.+@.+\\..+')

Invariant: not self.password.oclIsUndefined()

& self.password.size() >=8

Invariant: self.Phone.size() <= 11

Invariant: not self.status.oclIsUndefined()

```java
public class User {
    no usages
    @Id
    @GeneratedValue(strategy = GenerationType.IDENTITY)
    @Column(nullable = false, length = 11)
    private Integer ID;
    1 usage
    @Length(max = 15)
    @Column(unique = true)
    private String username;

    1 usage
    @Column(unique = true)
    @Email
    private String email;

    1 usage
    @Column(nullable = false)
    @Length(min = 8)
    private String password;
    1 usage
    @Length(min = 11)
    private String Phone;
    1 usage
    @Column(nullable = false)
    @Enumerated(EnumType.STRING)
    private Status status = Status.Inactive;
    1 usage
    @Column(nullable = false)
    @Enumerated(EnumType.STRING)
    private Type type= Type.Supervisor;
```

& self.status in {Status::Active, Status::Inactive}

init:

self.status = Status::Inactive

Invariant: not self.type.oclIsUndefined()

& self.type in {Type::Admin, Type::Supervisor}

init

self.type=Type::Supervisor

Stock_Request Class:

Context: Stock_Request

Invariant: not self.ID.oclIsUndefined()

& self.ID.size() <= 11

Invariant: self.Quantity.size() <= 11

& self.Quantity.oclIsUndefined()

Invariant: not self.Status.oclIsUndefined()

& self.Status in {RequestStatus::Approved

, RequestStatus::Pending

, RequestStatus::Rejected}

init

self.Status in RequestStatus::Pending

invariant: not self.warehouse.oclIsUndefined()

& self.warehouse.requests->includes(self)

Invariant: not self.product.oclIsUndefined()

& self.product.requests->includes(self)

Invariant: not self.user.oclIsUndefined()

& self.user.requests->includes(self)

```java
public class Stock_Request {
    no usages
    @Id
    @GeneratedValue(strategy = GenerationType.IDENTITY)
    @Column(nullable = false, length = 11)
    private Integer ID;
    no usages
    @Column(length = 11, nullable = false)
    private Integer Quantity;
    no usages
    @Enumerated(EnumType.STRING)
    @Column(nullable = false)
    private RequestStatus Status = RequestStatus.Pending;

    no usages
    @ManyToOne(cascade = CascadeType.DETACH)
    @JoinColumn(nullable = false)
    private Warehouse warehouse;
    no usages
    @ManyToOne(cascade = CascadeType.DETACH)
    @JoinColumn(nullable = false)
    private Product product;
    no usages
    @ManyToOne(cascade = CascadeType.DETACH)
    @JoinColumn(nullable = false)
    private User user;

}
```

Warehouse Class:

Context: Warehouse

Invariant: not self.ID.oclIsUndefined()

& self.ID.size() <= 11

Invariant: not self.Name.oclIsUndefined()

Invariant: Warehouse.allInstances()->isUnique(Location)

Invariant: not self.status.oclIsUndefined()

& self.status in {Status::Active, Status::Inactive}

init:

self.status = Status::Inactive

Invariant: self.Capacity.size() <= 11

Invariant: not self.user.oclIsUndefined()

& self.user.warehouse->includes(self)

```java
public class Warehouse {
    no usages
    @Id
    @GeneratedValue(strategy = GenerationType.IDENTITY)
    @Column(nullable = false, length = 11)
    private Integer ID;
    no usages
    @Column(nullable = false)
    private String Name;
    no usages
    @Column(unique = true)
    private String Location;
    no usages
    @Enumerated(EnumType.STRING)
    @Column(nullable = false)
    private Status status = Status.Inactive;
    no usages
    @Column(length = 11)
    private Integer Capacity;
    no usages
    @OneToOne(cascade = CascadeType.DETACH)
    @JoinColumn(nullable = false)
    private User user;
}
```

Warehouse_Product Class:

Context: Warehouse_Product

Invariant: not self.warehouse.oclIsUndefined()

& self.warehouse.warehouse_Product->includes(self)

Invariant: not self.product.oclIsUndefined()

& self.product.warehouse_Product->includes(self)

```java
public class Warehouse_Product {
    no usages
    @Id
    @ManyToOne(cascade = CascadeType.DETACH)
    @JoinColumn(nullable = false)
    private Warehouse warehouse;
    no usages
    @Id
    @ManyToOne(cascade = CascadeType.DETACH)
    @JoinColumn(nullable = false)
    private Product product;


}
```

OCL pre and post conditions of functions on two classes :

ProductService class:

```
1 usage
public List<product> getAllProducts() { return productRepository.findAll(); }
```

Context: ProductService::getAllProducts() : List<Product>

post: result.oclIsTypeOf(List(Product)) and not result.oclIsUndefined() and result = productRepository

->asSequence()->sortedBy(p | p.id)->toList()

```
1 usage
public void createProduct(product newproduct) { productRepository.save(newproduct); }
```

Context: ProductService::createProduct(newproduct : Product)

pre: not newproduct.oclIsUndefined() and not newproduct.id.oclIsUndefined() and not productRepository

->exists(p | p.id = newproduct.id)

post: productRepository->includes(newproduct) and productRepository->size() = productRepository@pre->size() + 1

```
1 usage
public void updateProduct(product updatedproduct) { productRepository.save(updatedproduct); }
```

context: ProductService::updateProduct(updatedproduct : Product)

pre: not updatedproduct.oclIsUndefined() and not updatedproduct.id.oclIsUndefined() and

productRepository->exists(p | p.id = updatedproduct.id)

post: productRepository->one(p | p.id = updatedproduct.id and

p = updatedproduct) and

productRepository->size() = productRepository@pre->size()

```
1 usage
public void deleteProduct(Integer ID) { productRepository.deleteById(ID); }
```

context: ProductService::deleteProduct(ID : Integer)

pre: not ID.oclIsUndefined() and productRepository->exists(p | p.id = ID)

post: not productRepository->exists(p | p.id = ID) and

productRepository->size() = productRepository@pre->size() -1

userService Class:

```
1 usage
public List<user> getAllUsers() { return userRepository.findAll(); }
```

Context: userService::getAllUsers() : List<user>

post: result.oclIsTypeOf(List(user)) and not result.oclIsUndefined() and result = userRepository

->asSequence()->sortedBy(p | p.id)->toList()

```
public void updateUser(user updatedUser) {
    userRepository.save(updatedUser);
```

context: userService::updateUser(updatedUser : user)

pre: not updatedUser.oclIsUndefined() and not updatedUser.id.oclIsUndefined() and

userRepository->exists(p | p.id = updatedUser.id)

post: userRepository->one(p | p.id = updatedUser.id and

p = updatedUser) and userRepository->size() = userRepository@pre->size()

```
public loginMessage register(userDto userDto) {
    user user = new user(
            userDto.getUsername(),
            userDto.getEmail(),
            userDto.getPassword(),
            userDto.getPhone(),
            userDto.getType(),
            userDto.getStatus()
    );
    userRepository.save(user);
    return new loginMessage( message: "signed up!",  status: true);
}
```

context UserService::register(userDto : UserDto)

pre: not userDto.oclIsUndefined() and

userDto.username.oclIsUndefined() = false and

userDto.email.oclIsUndefined() = false and

not userRepository->exists(u | u.username = userDto.username) and

not userRepository->exists(u | u.email = userDto.email) and

userDto.password.oclIsUndefined() = false and

userDto.password.size() >= 8

post: userRepository->one(u | u.username = userDto.username and u.email = userDto.email) and

userRepository->size() = userRepository@pre->size() + 1

```
public loginMessage loginUser(loginDto loginDto) {
    user user = findUserByEmail(loginDto.getEmail());
    if (user == null) {
        return new loginMessage( message: "Email doesn't exist!",  status: false);
    }

    if (!passwordMatches(loginDto.getPassword(), user.getPassword())) {
        return new loginMessage( message: "Password doesn't match!",  status: false);
    }

    Optional<com.project.pr.models.user> authenticatedUser = authenticateUser(loginDto.getEmail(), user.getPassword());
    if (authenticatedUser.isPresent()) {
        return new loginMessage( message: "Logged in!",  status: true);
    } else {
        return new loginMessage( message: "Login failed!",  status: false);
    }
}
```

context UserService::loginUser(loginDto : LoginDto)

pre: not loginDto.oclIsUndefined() and

loginDto.email.oclIsUndefined() = false and

loginDto.password.oclIsUndefined() = false and

not userRepository->findByEmail(loginDto.email).oclIsUndefined() and

userRepository->exists(u | u.email = loginDto.email and u.password = loginDto.password)

post: if userRepository->exists(u | u.email = loginDto.email) then

if userRepository->any(u | u.email = loginDto.email and u.password = loginDto.password) then

result.success = true and result.message = 'Logged in!'

else

result.success = false and result.message = 'Password doesn\'t match!'

endif

else

result.success = false and result.message = 'Email doesn\'t exist!'

endif

```
1 usage
public void deleteUser(Integer ID) { userRepository.deleteById(ID); }
```

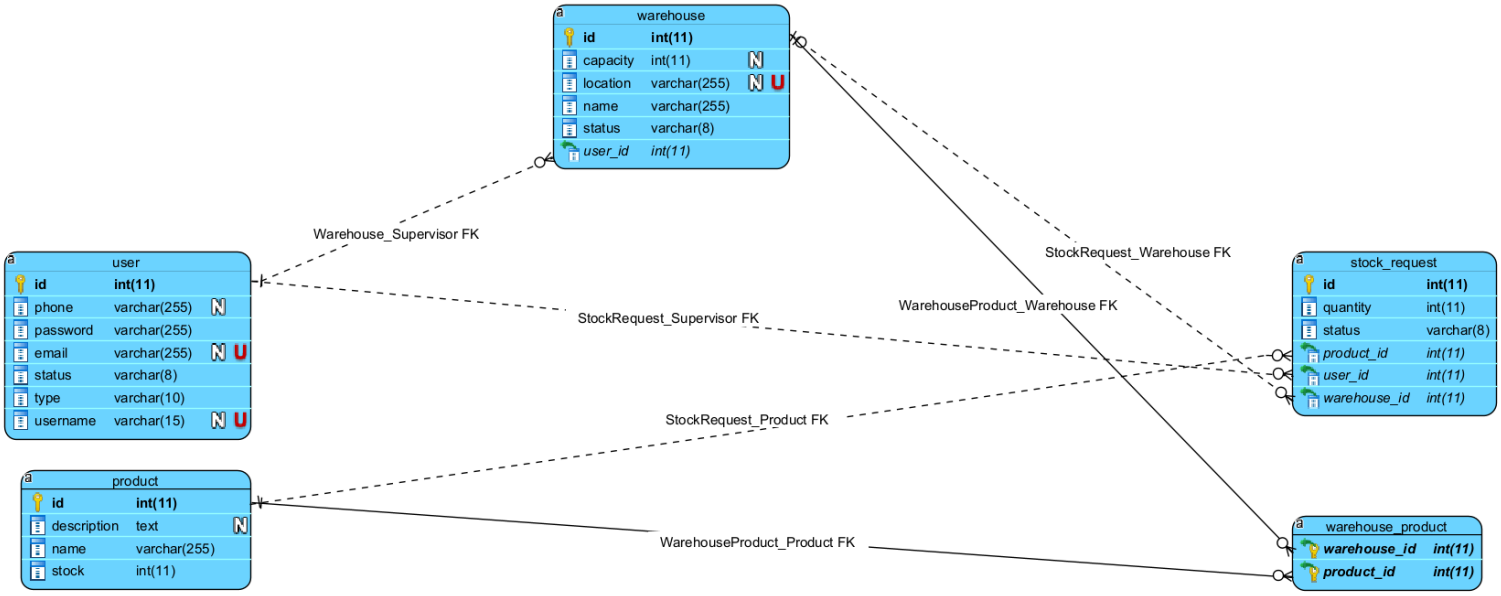
context: userService::deleteUser(ID : Integer)

pre: not ID.oclIsUndefined() and userRepository->exists(p | p.id = ID)

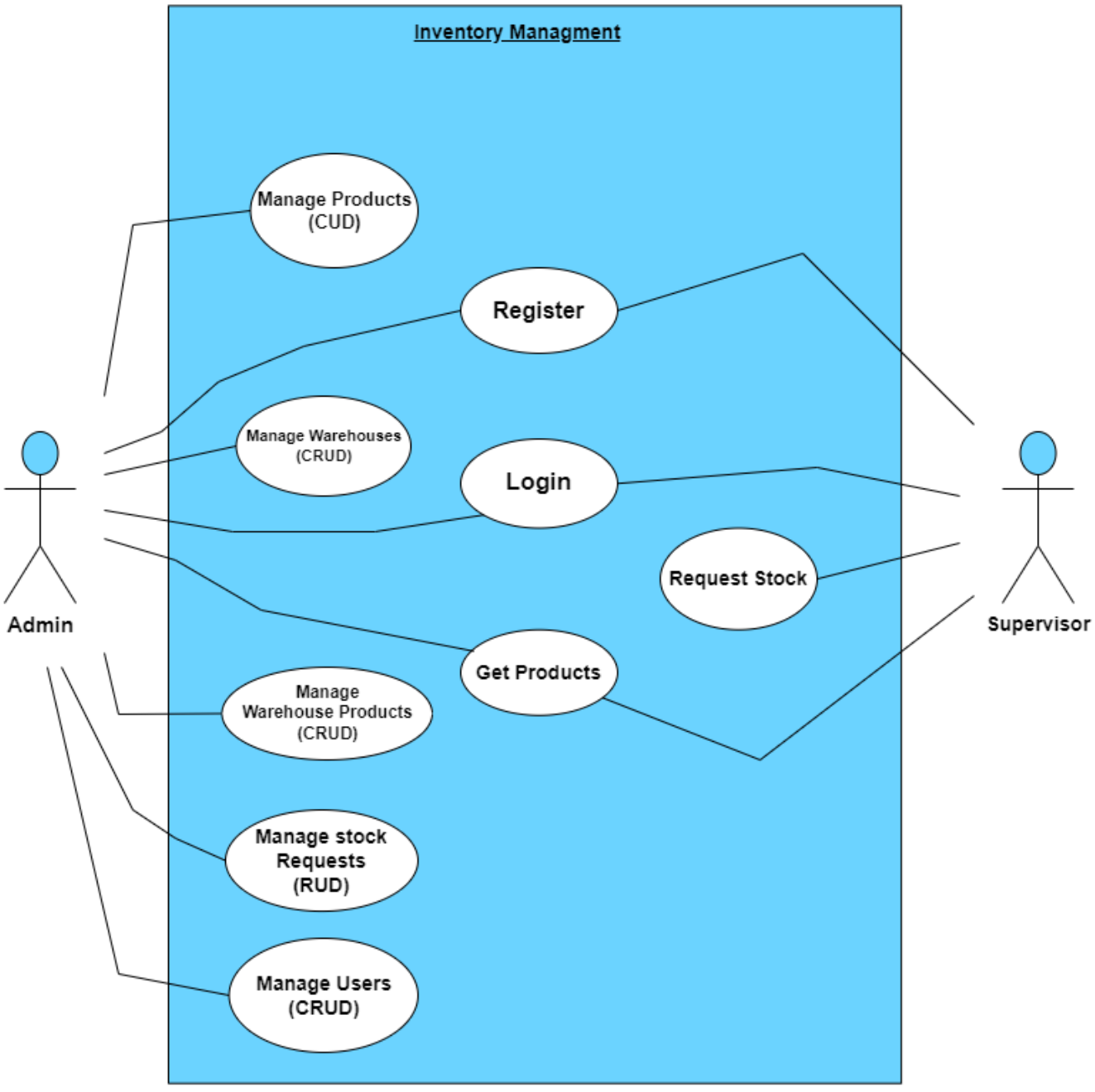
post: not userRepository->exists(p | p.id = ID) and

userRepository->size() = userRepository@pre->size() -1

Database Specification: ERD Tables:

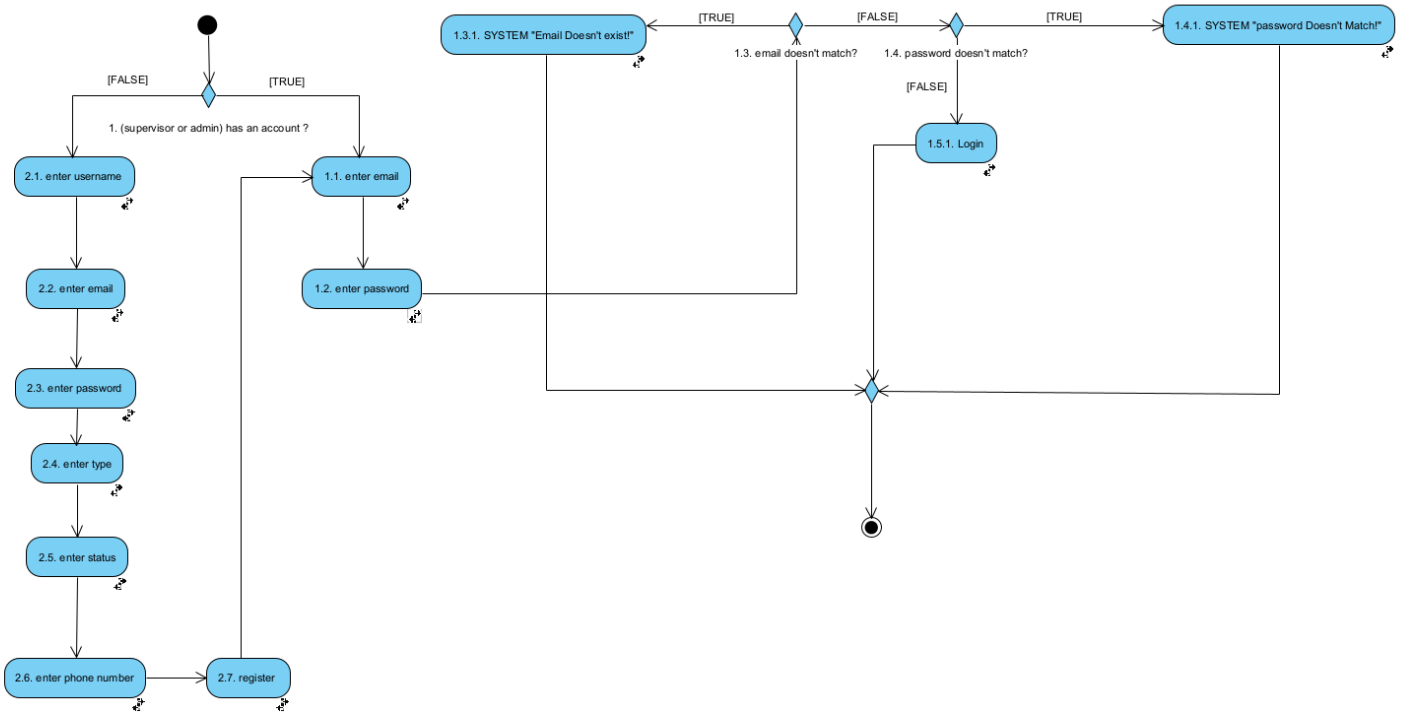


Use Case Diagram:

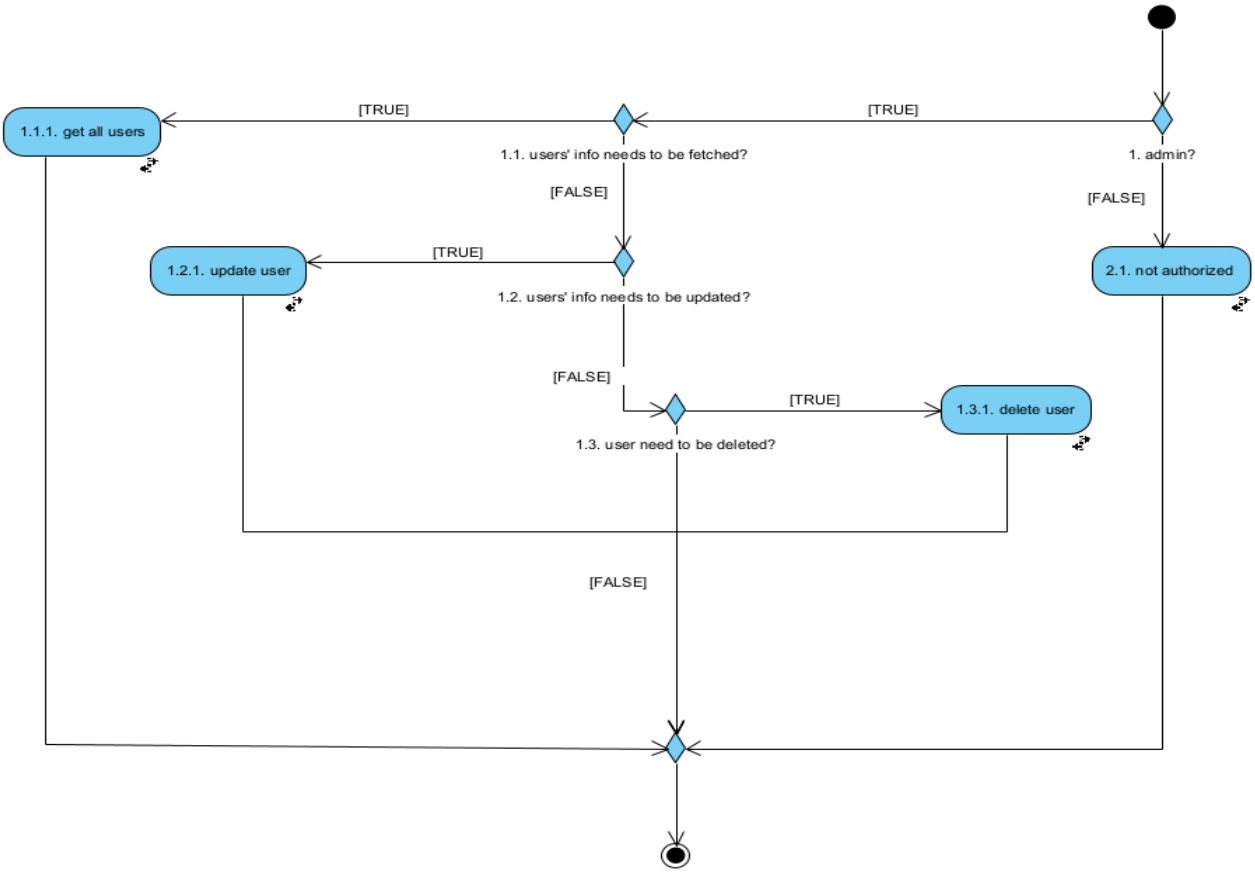


Activity Diagrams:

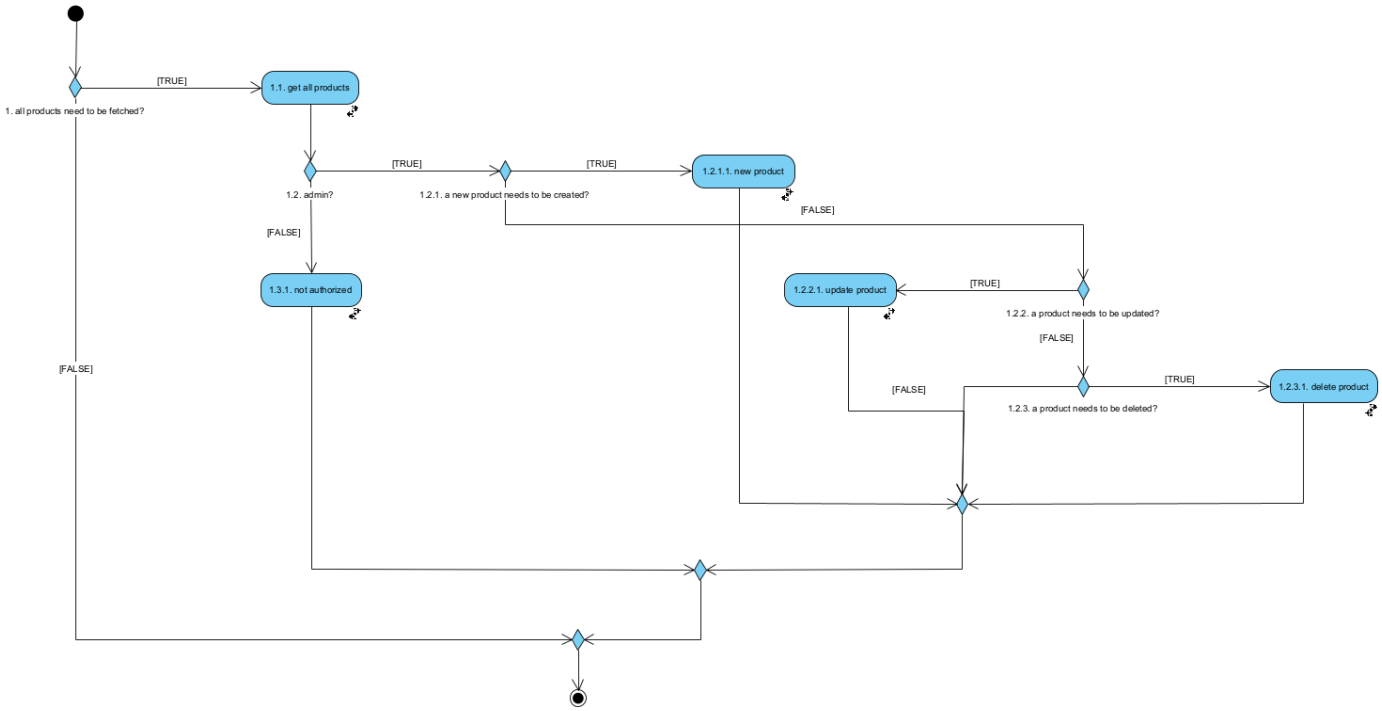
Login&Registration:



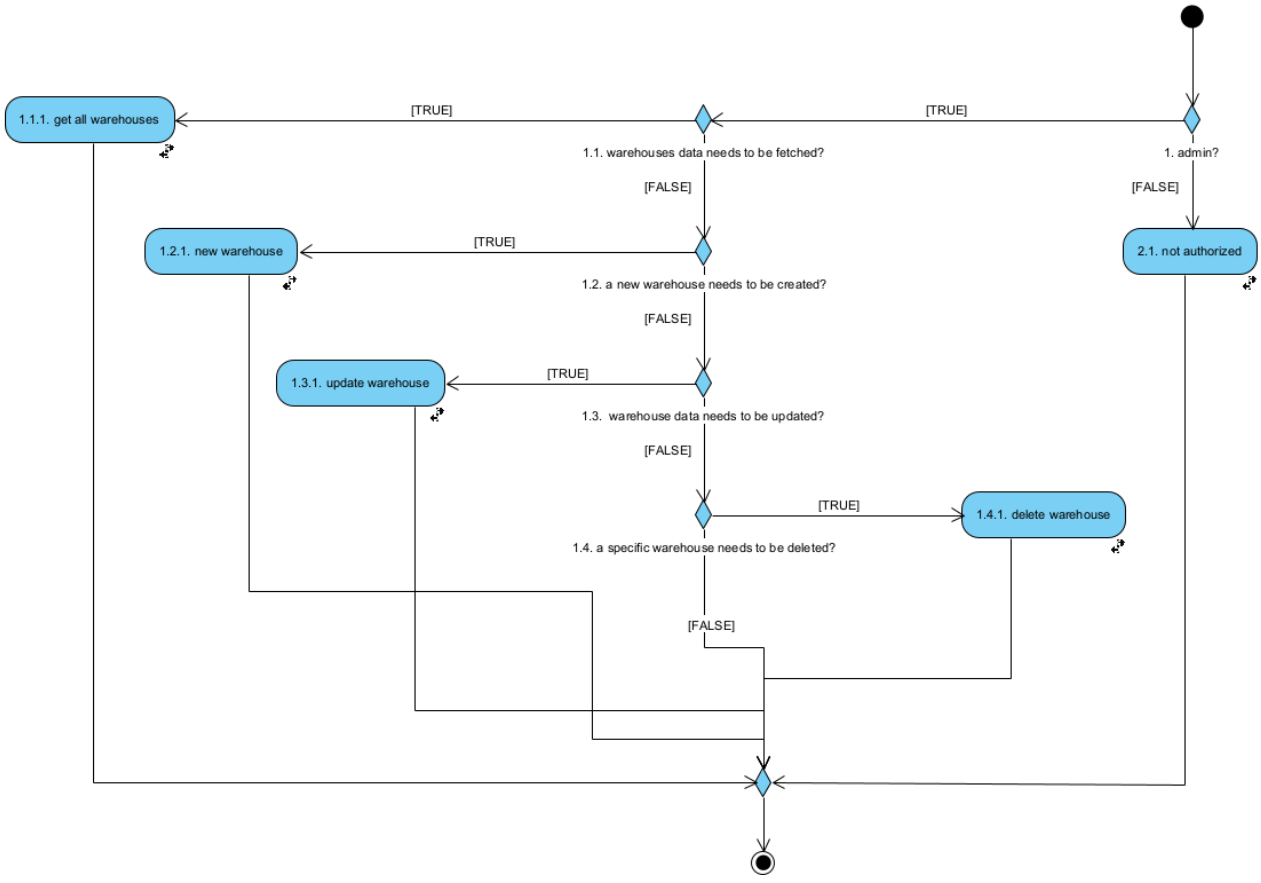
Manage users:



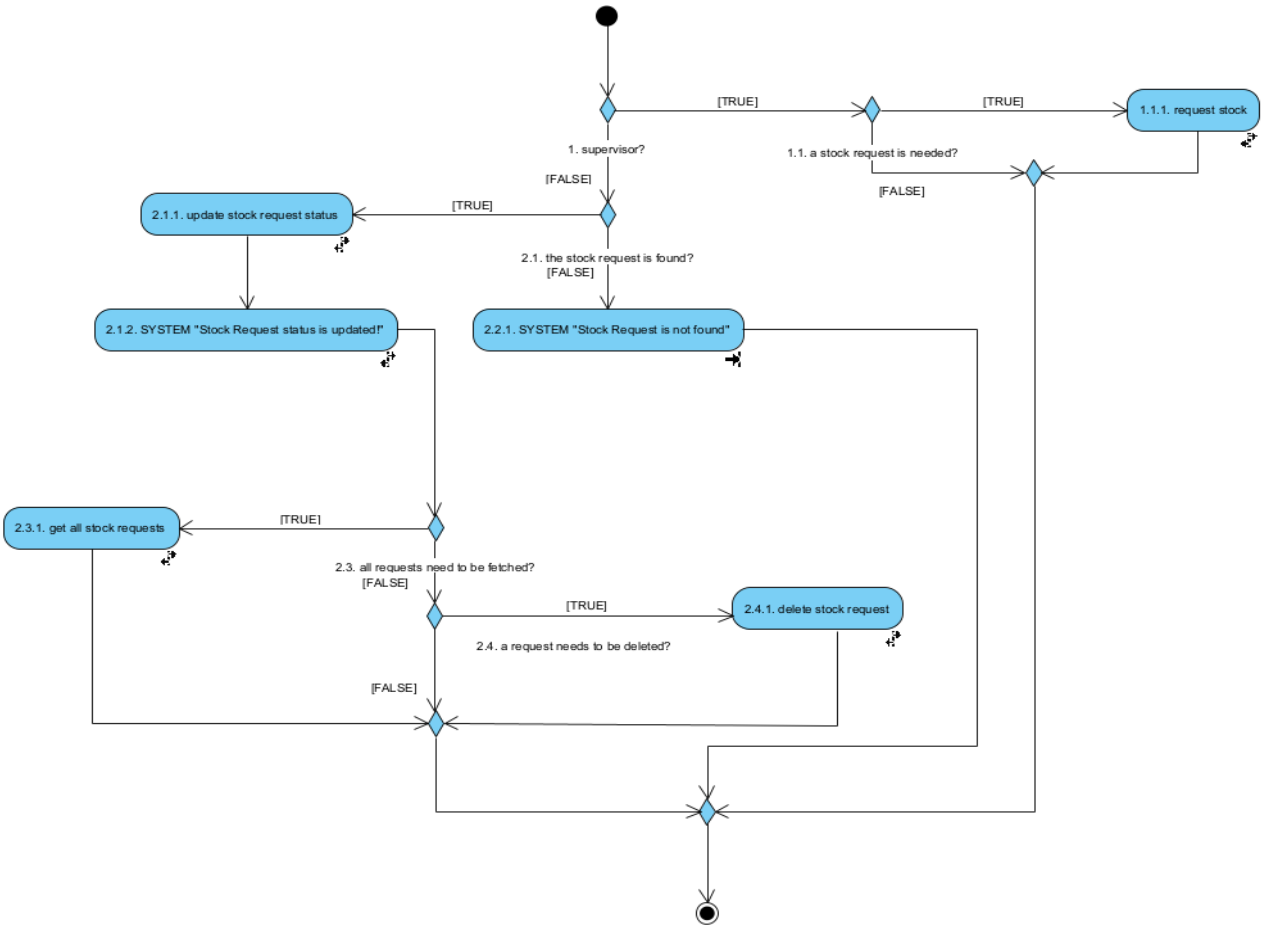
Manage products:



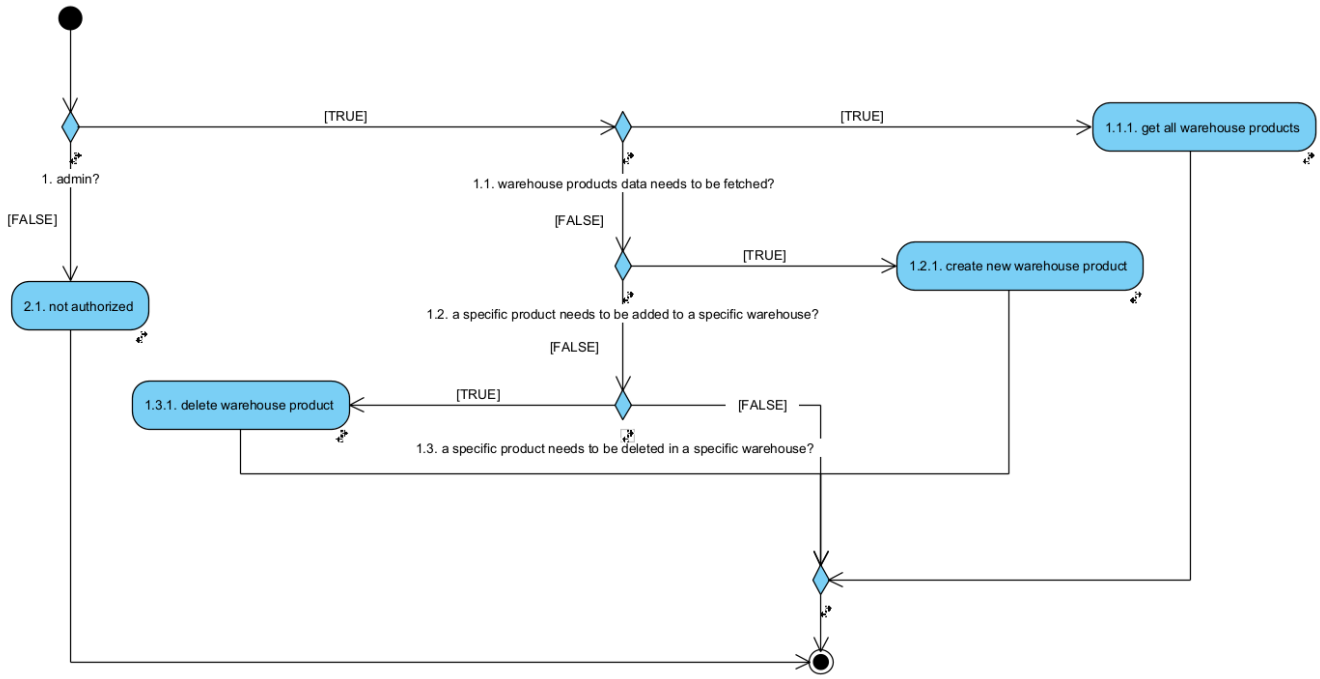
Manage warehouses:



Manage stock requests:

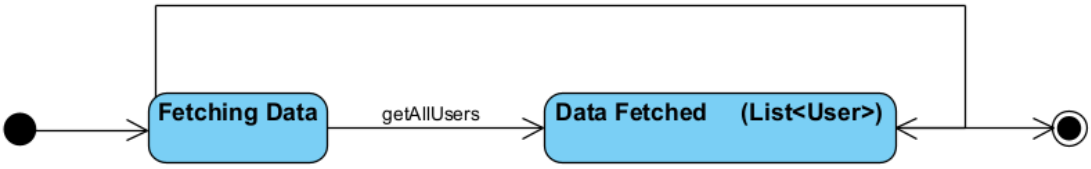


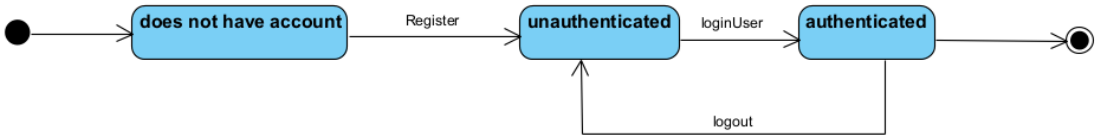
Manage warehouse products:

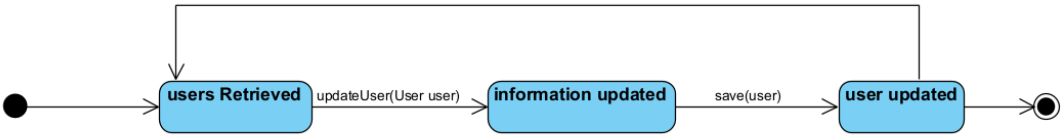


State Diagrams:

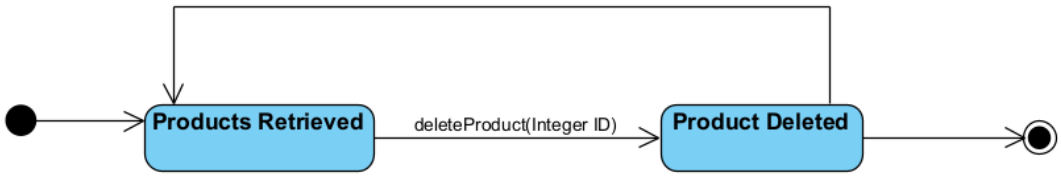
Manage users:

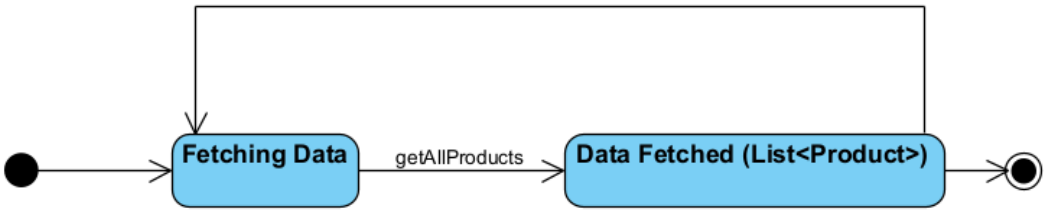


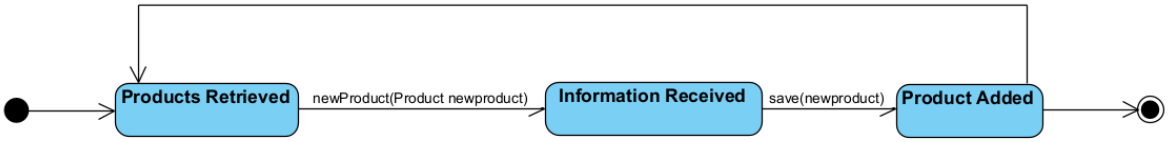


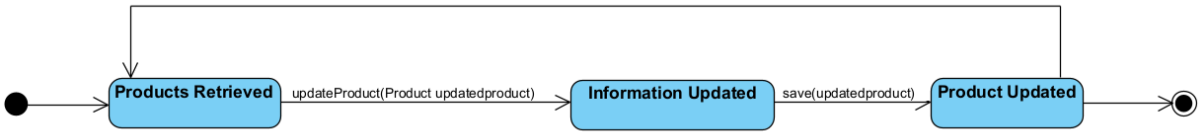


Manage products:

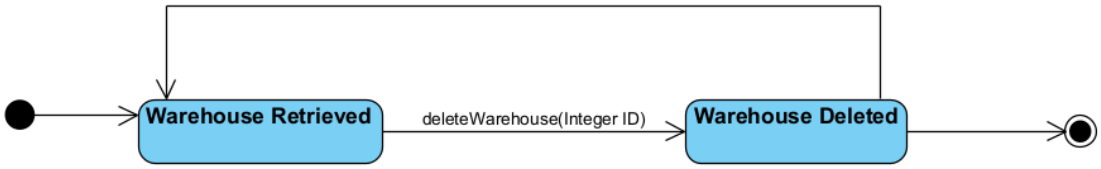


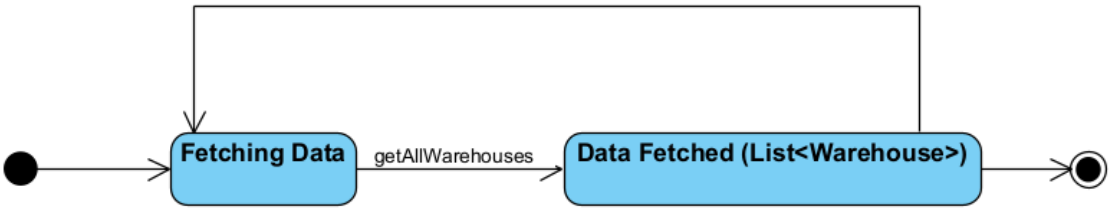


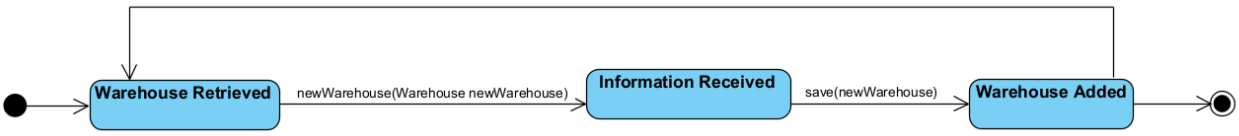


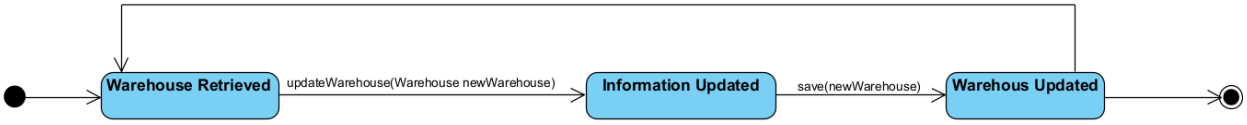


Manage warehouses:

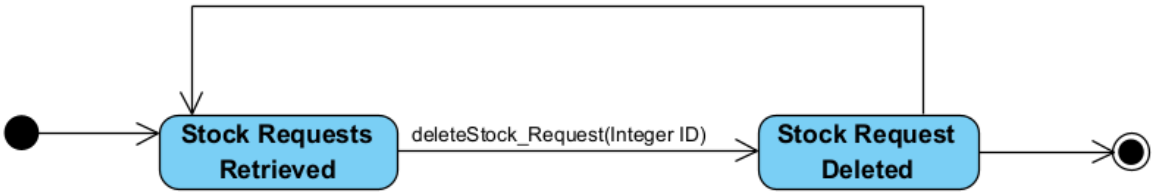


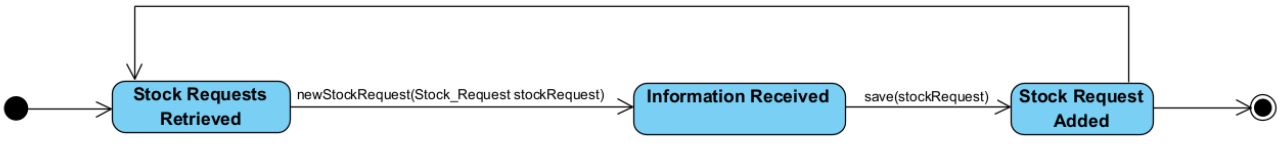


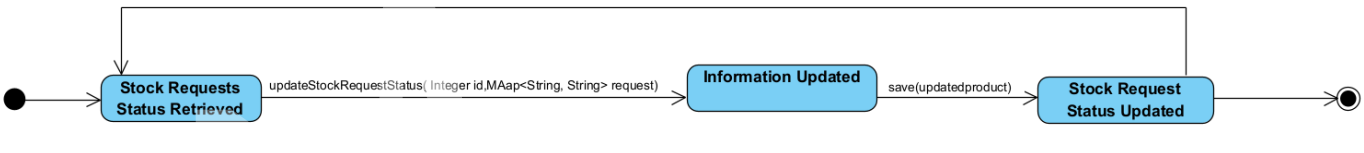




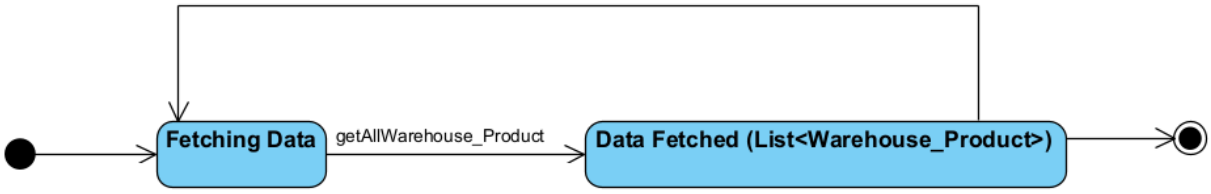
Manage stock requests:

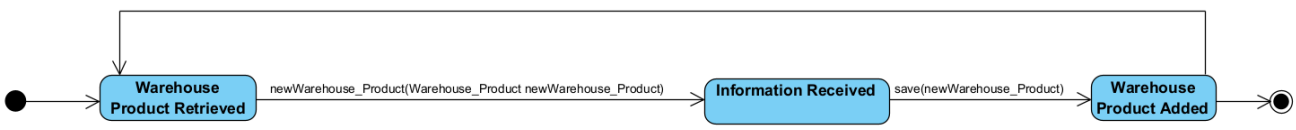






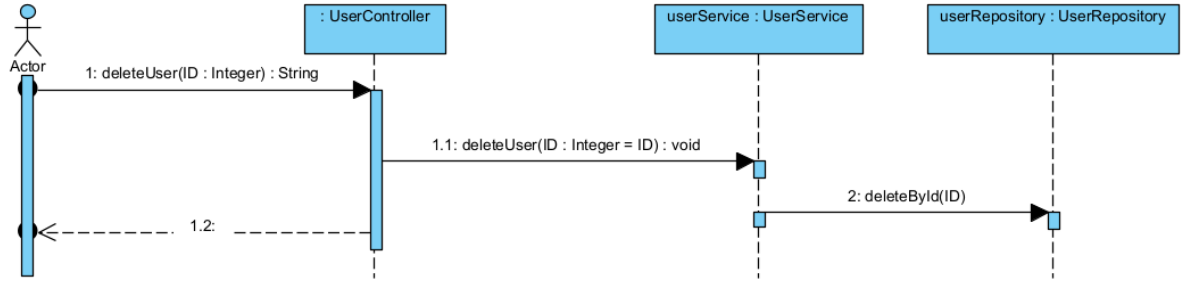
Manage warehouse products:

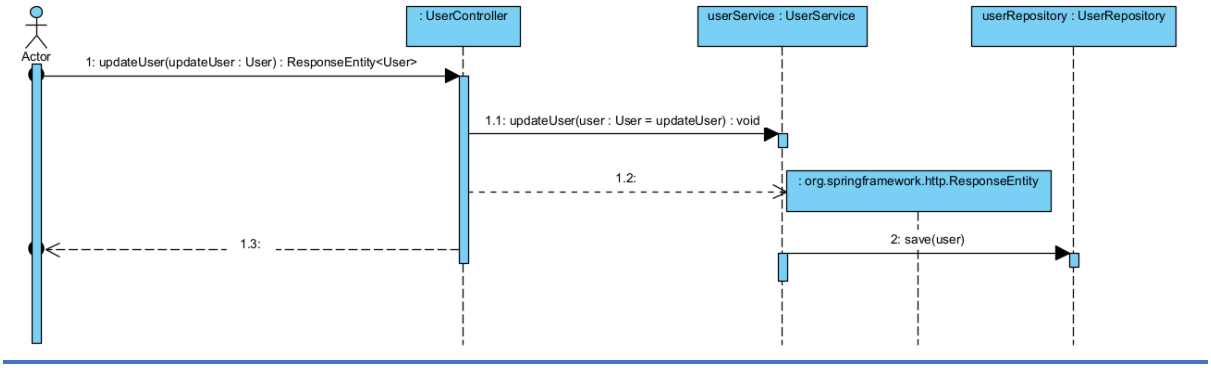


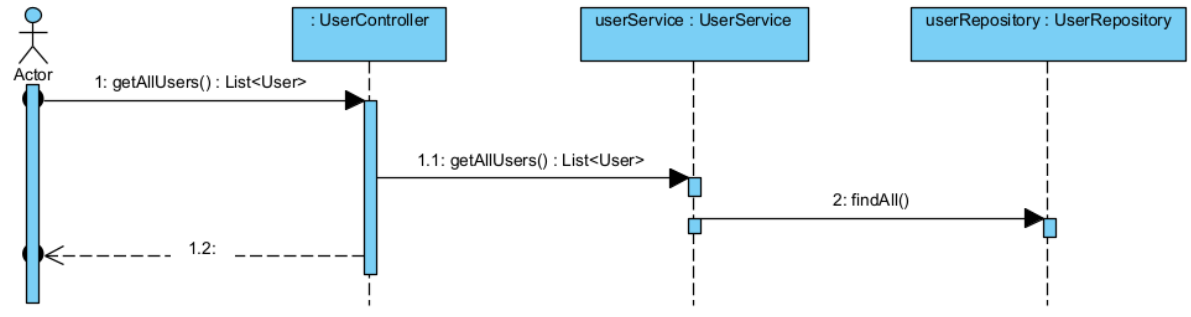


Sequence Diagrams:

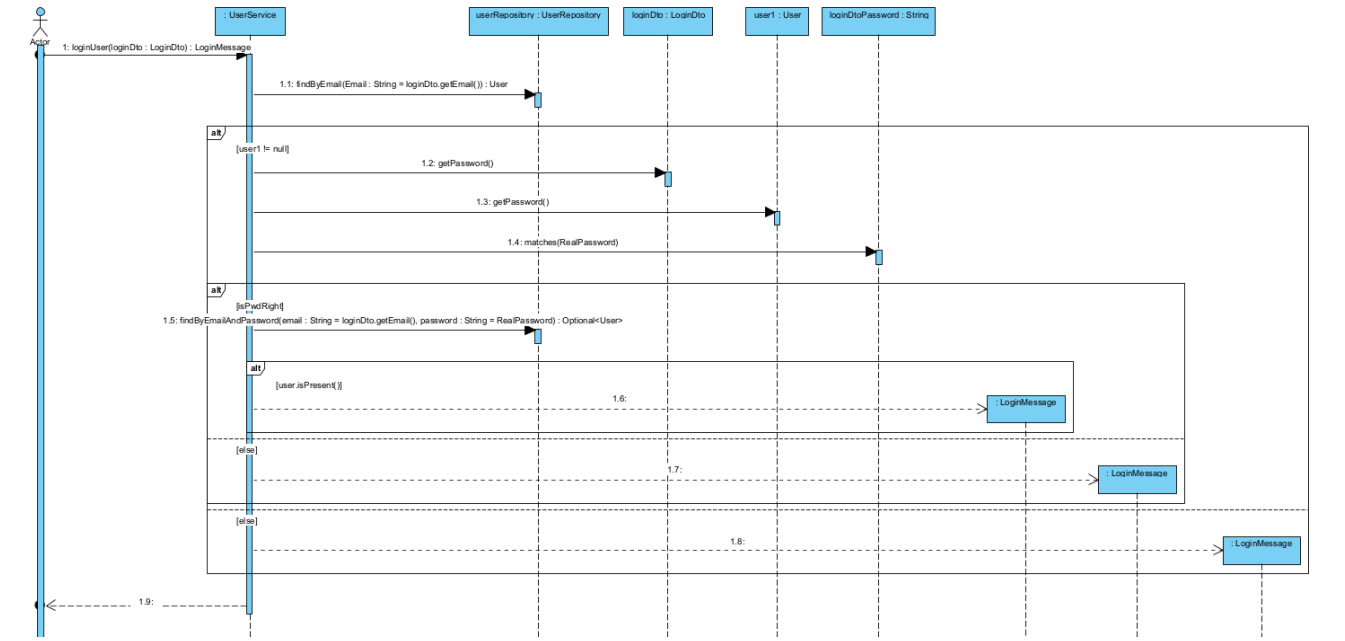
Manage users:

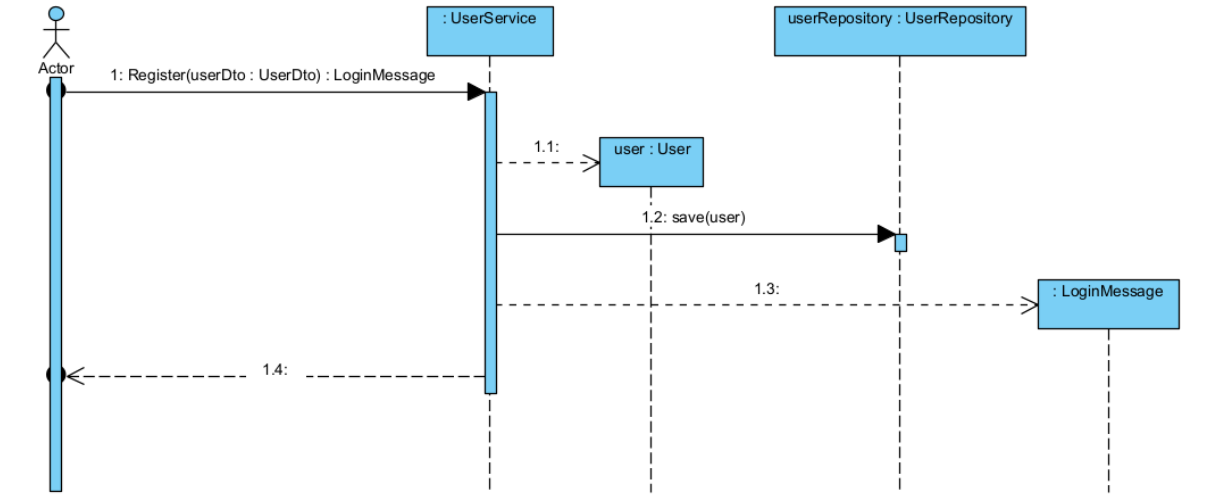




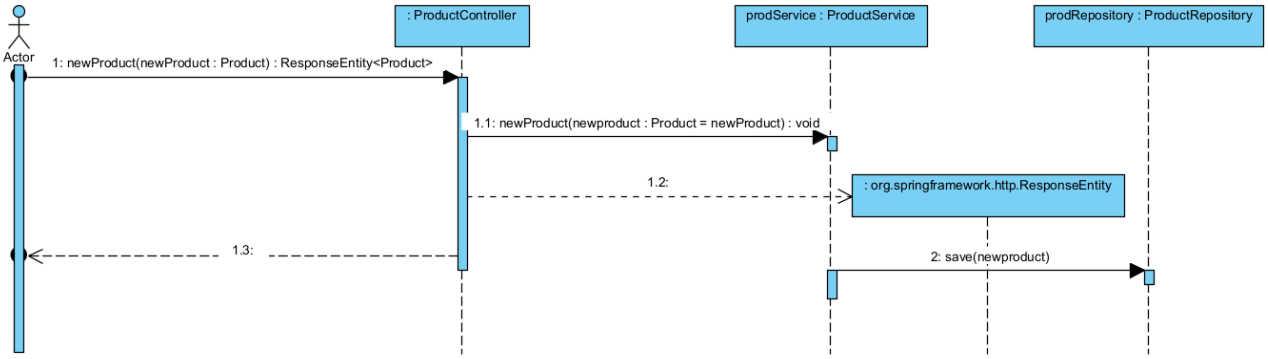


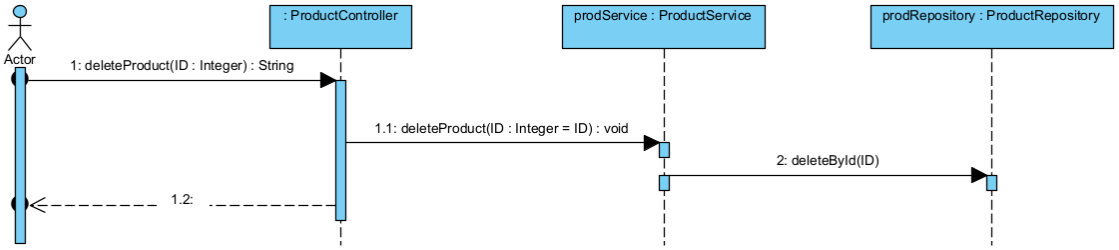
Login&Registration:

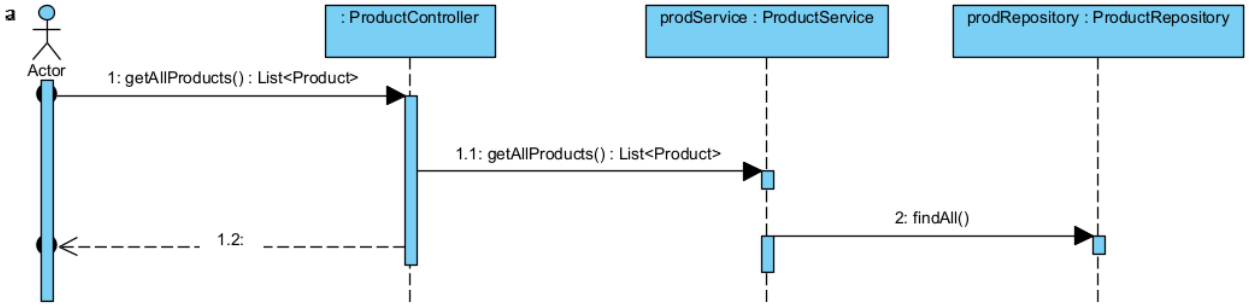


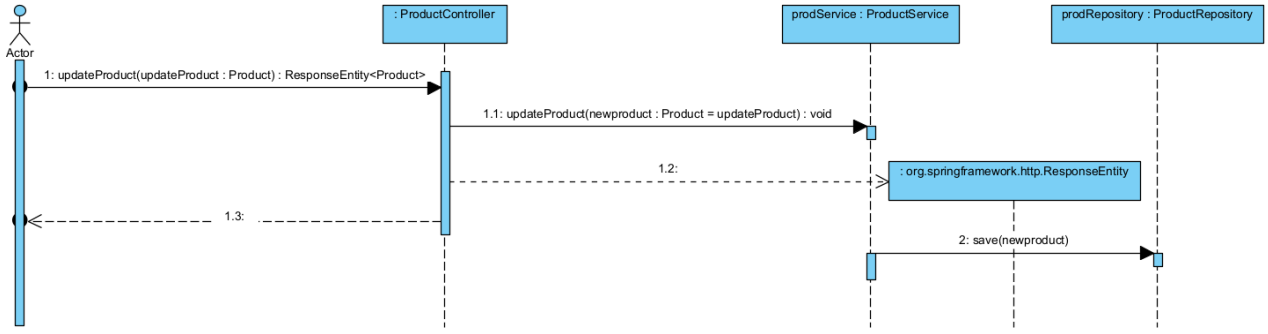


Manage products:

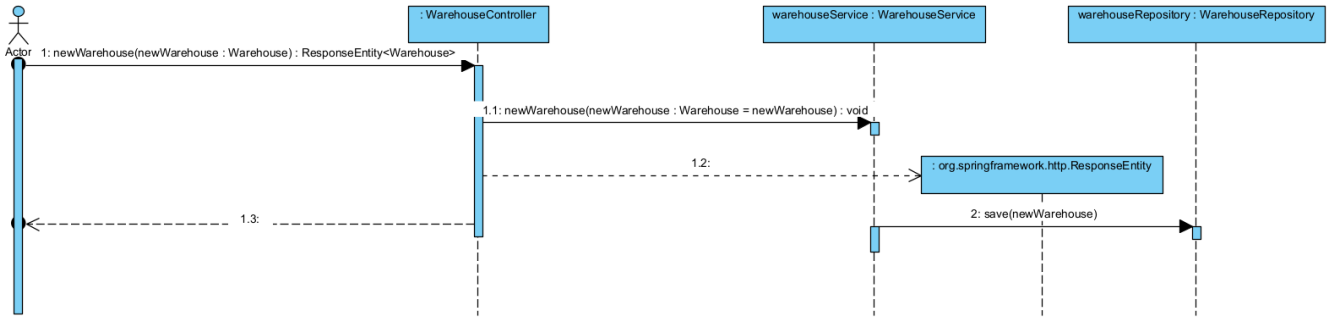


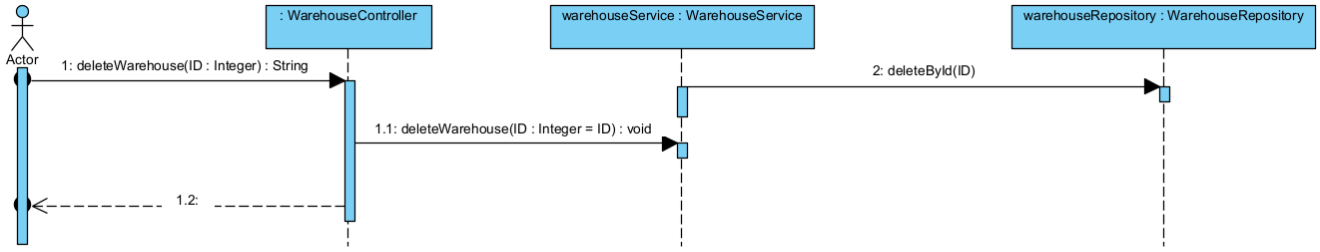


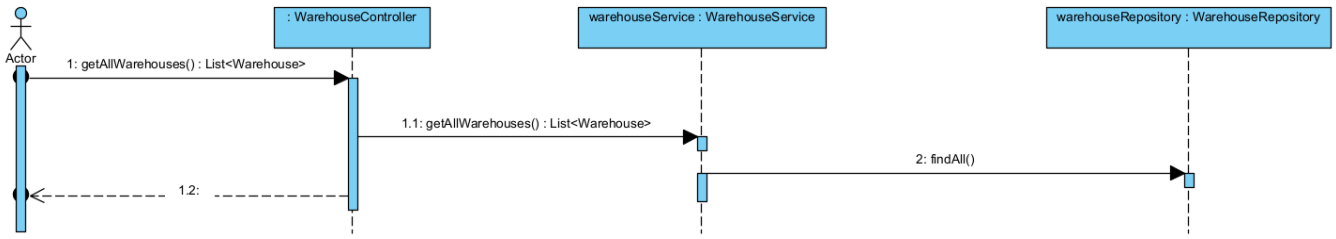


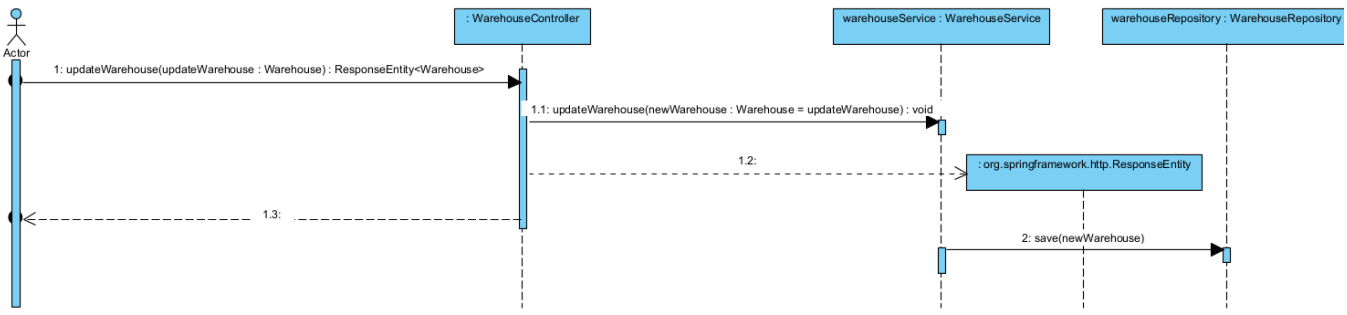


Manage warehouses:

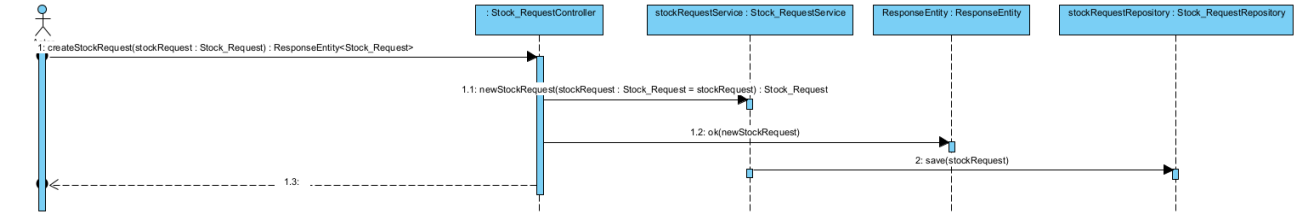


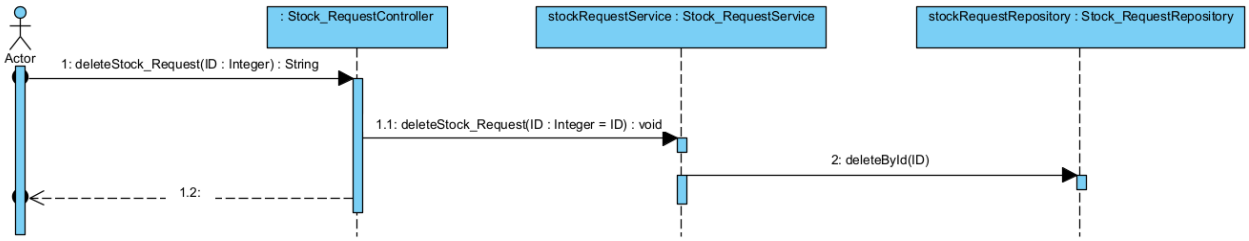


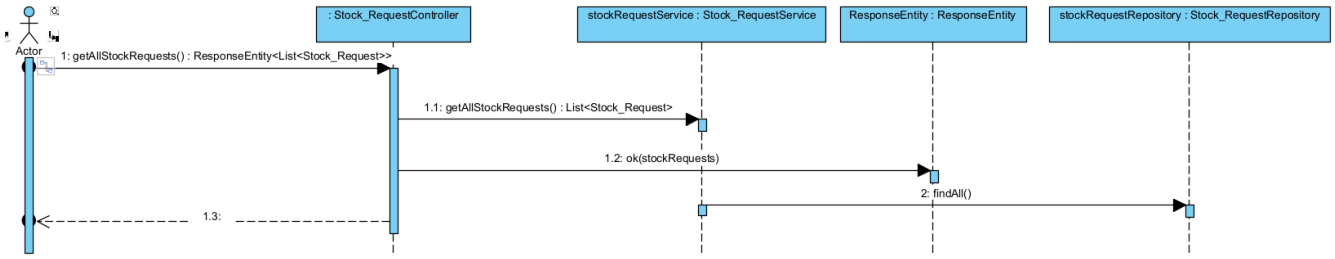


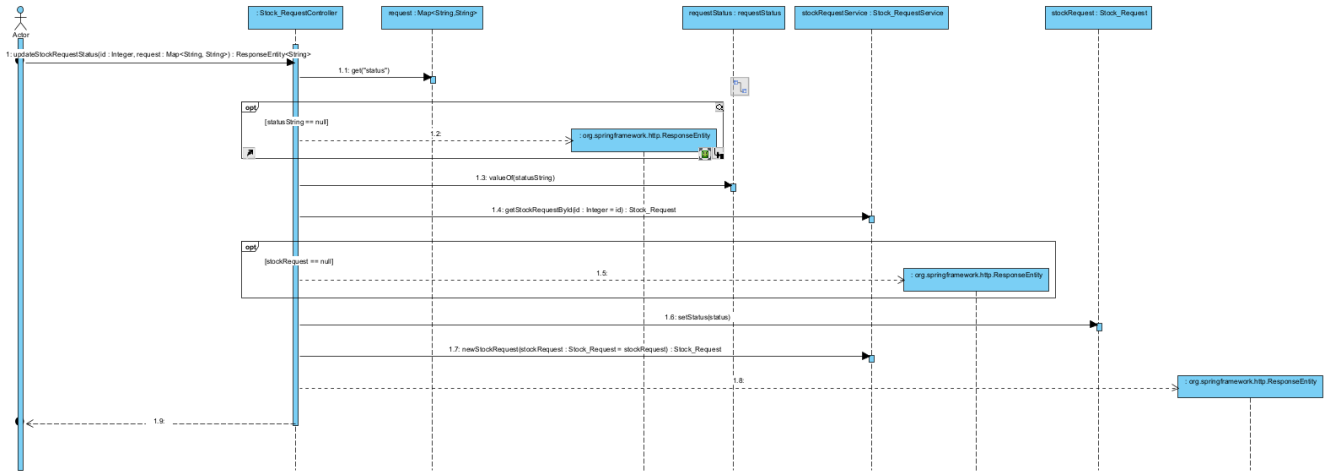


Manage stock requests:

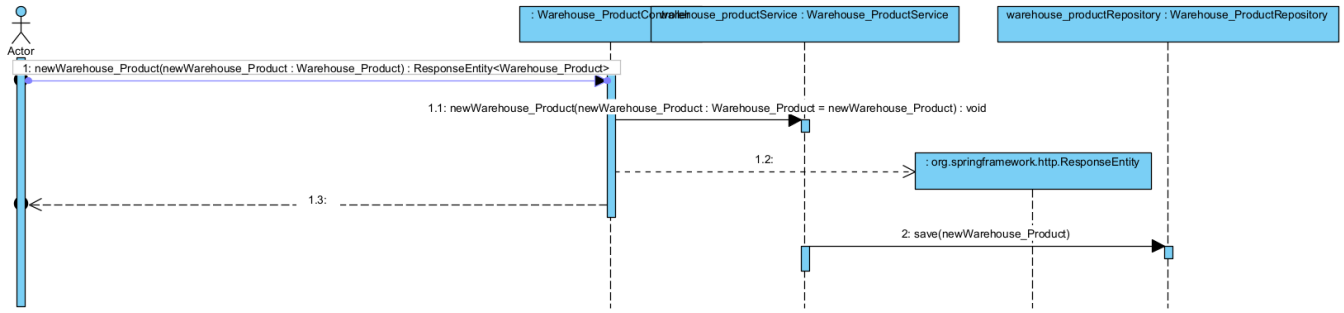


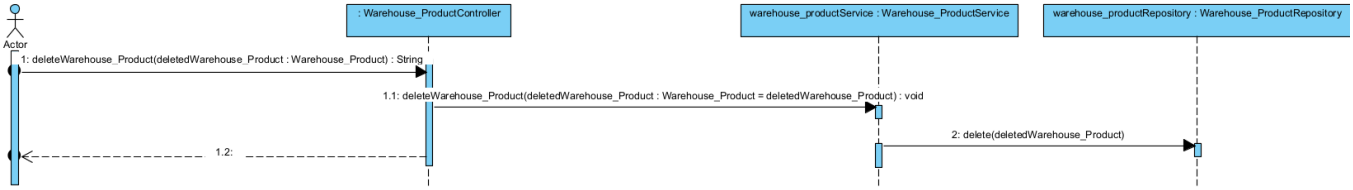


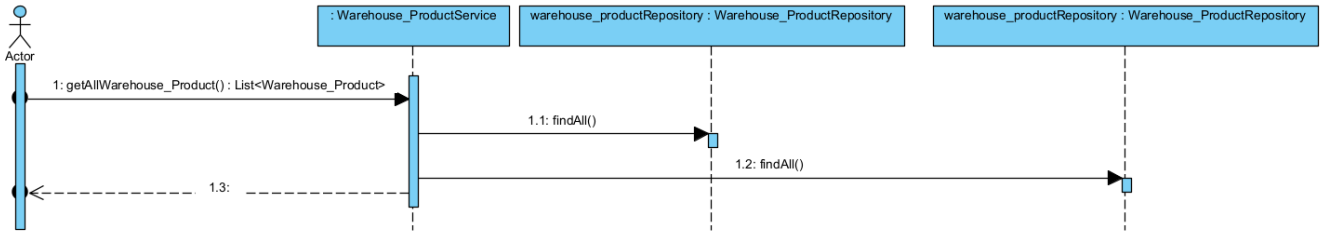




Manage warehouse products:

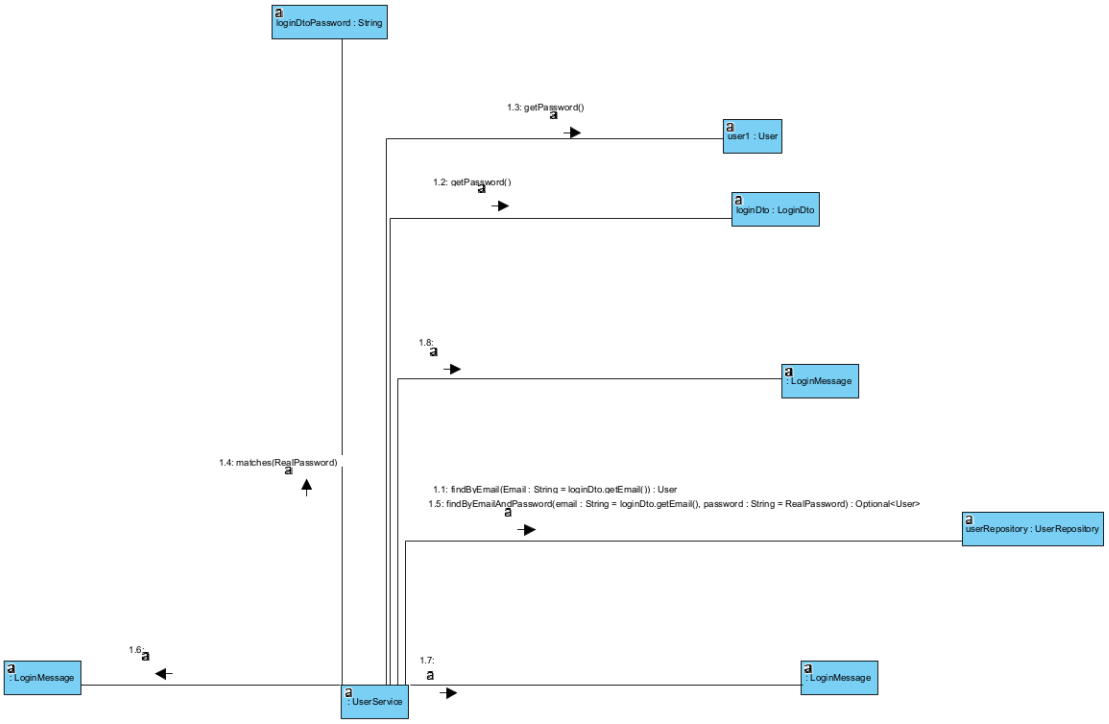


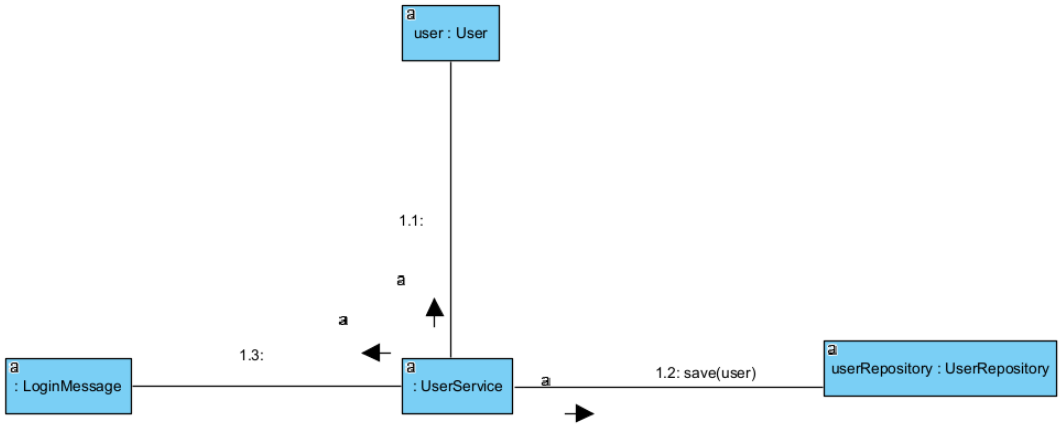




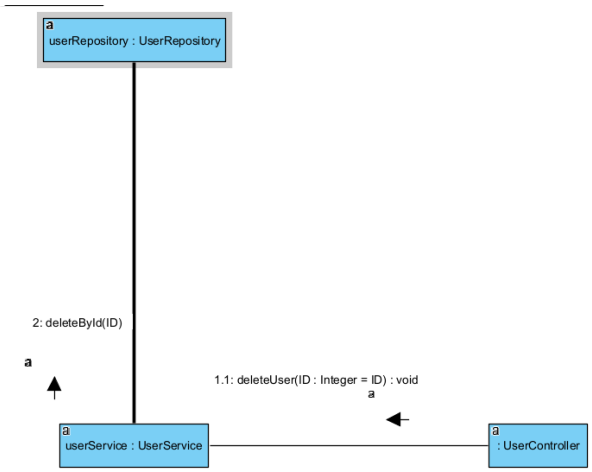
Communication Diagrams:

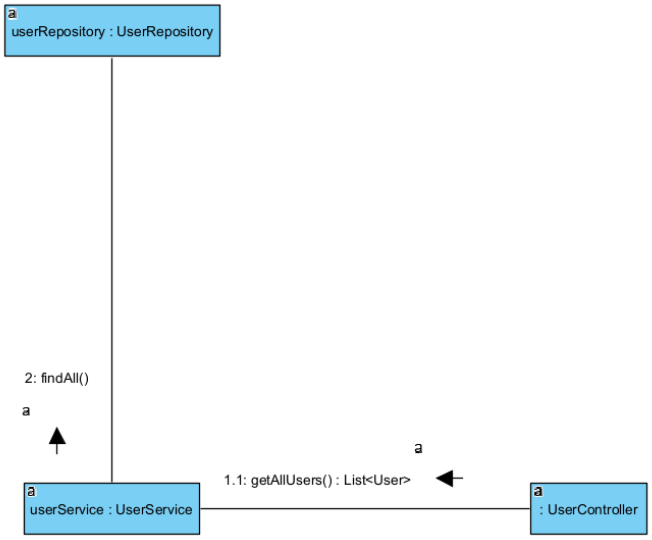
Login&Registration:

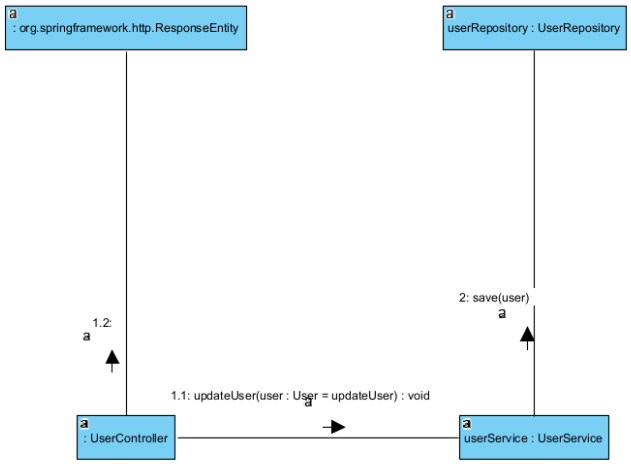




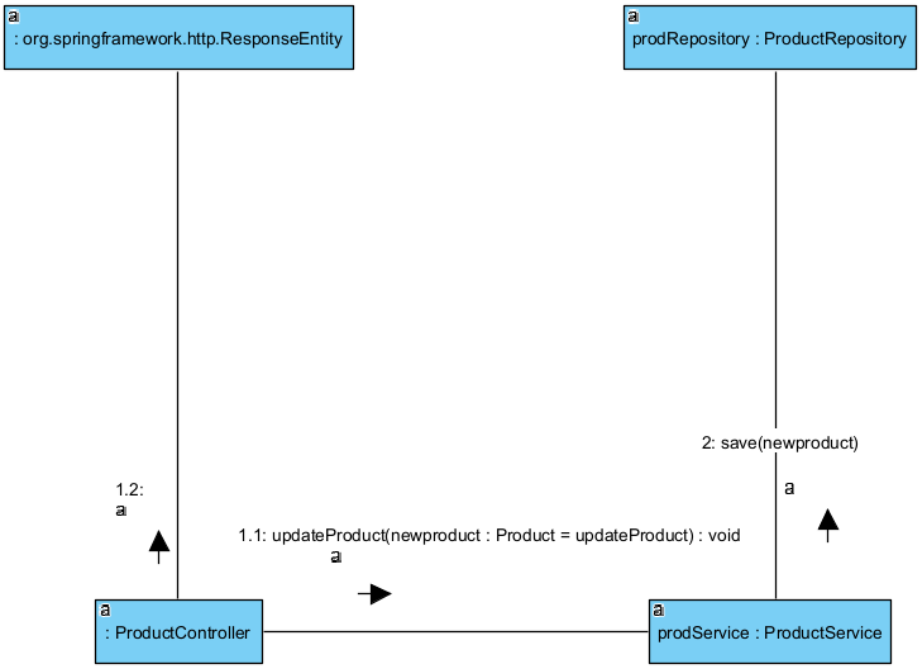
Manage users:

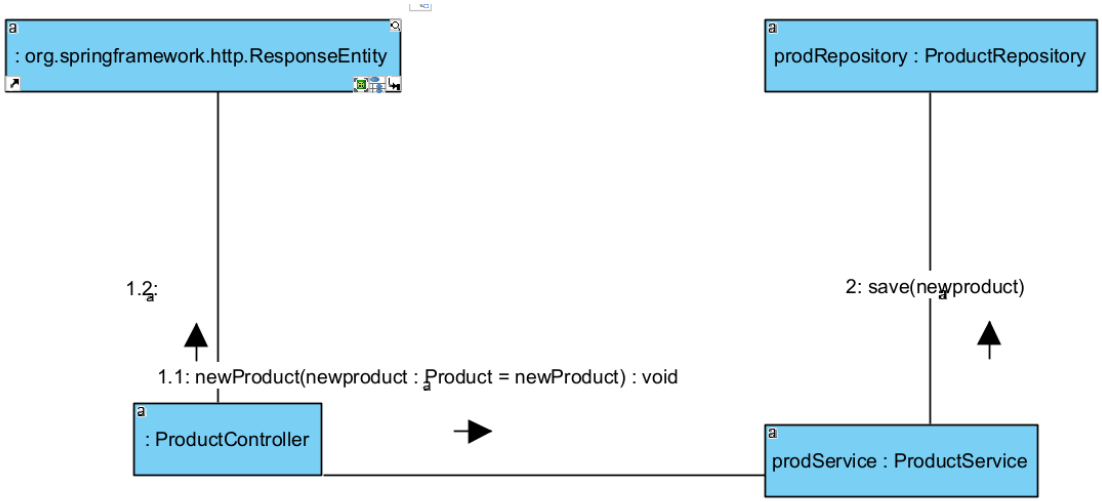


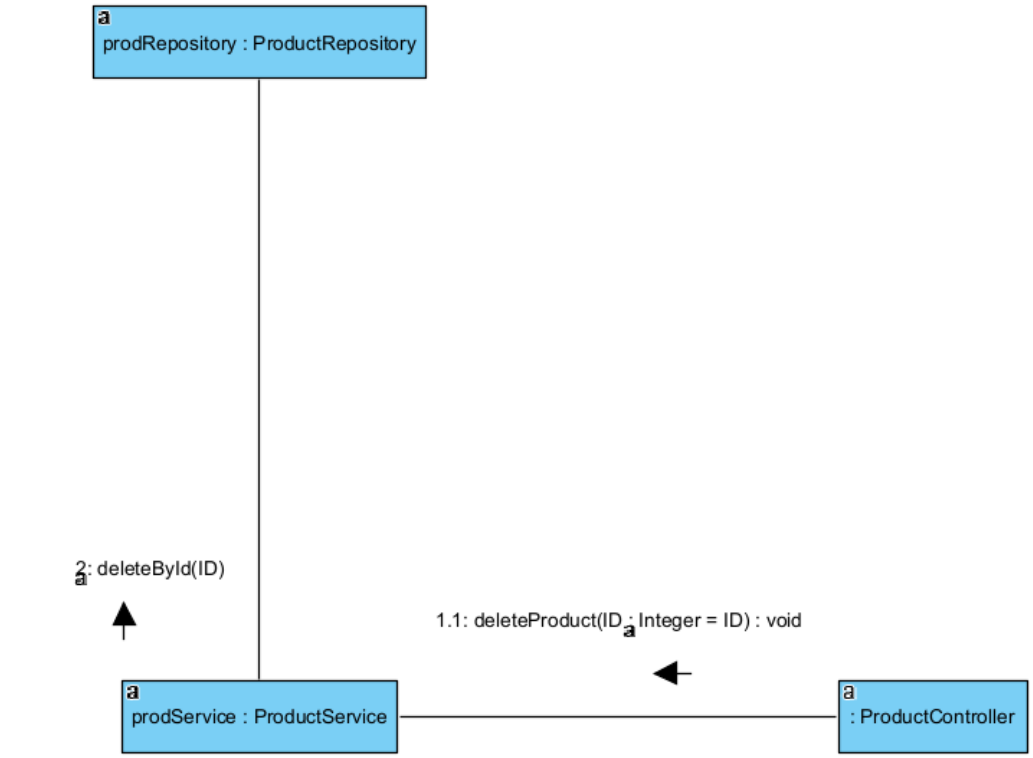


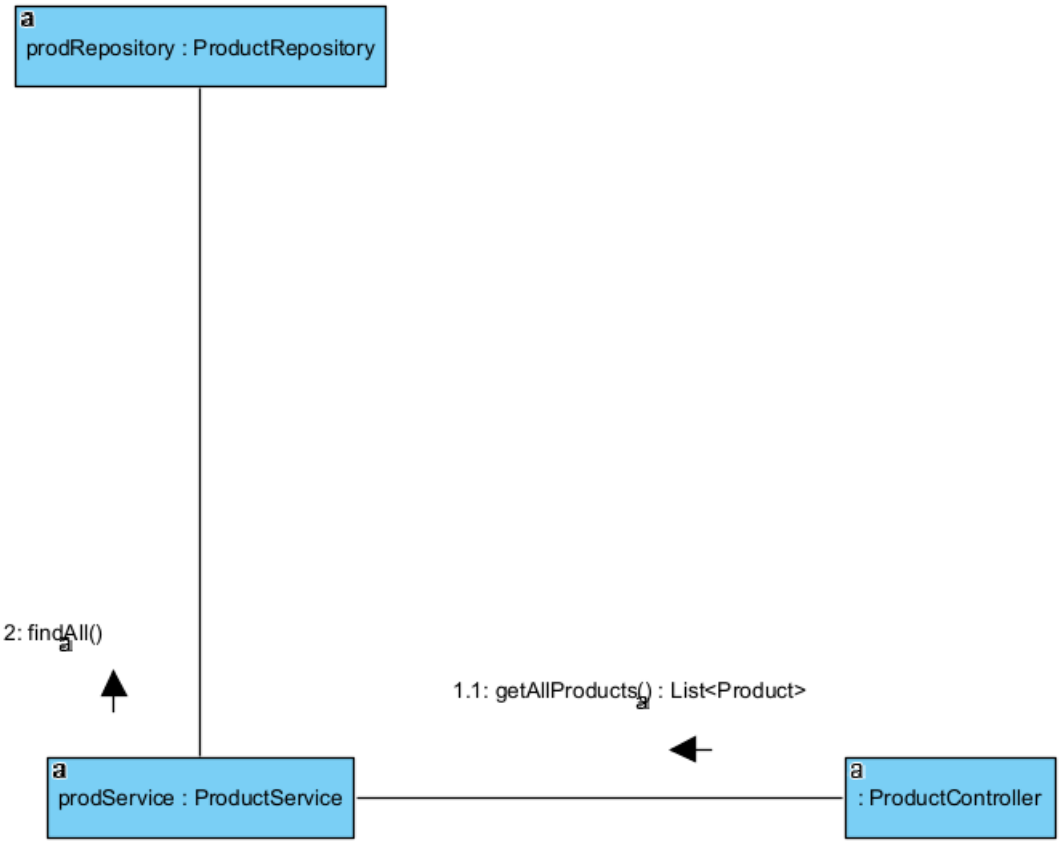


Manage products:

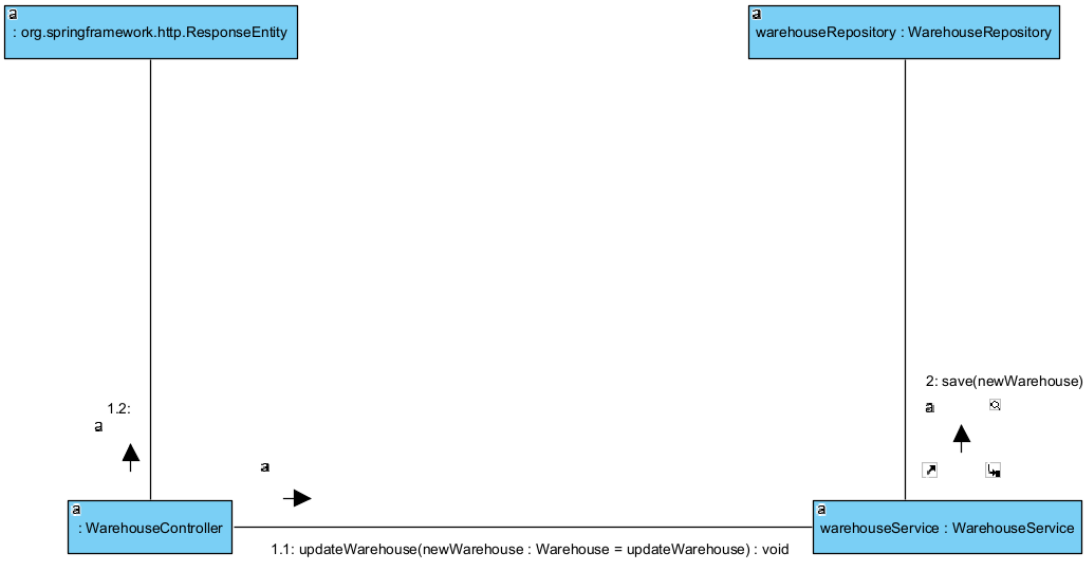


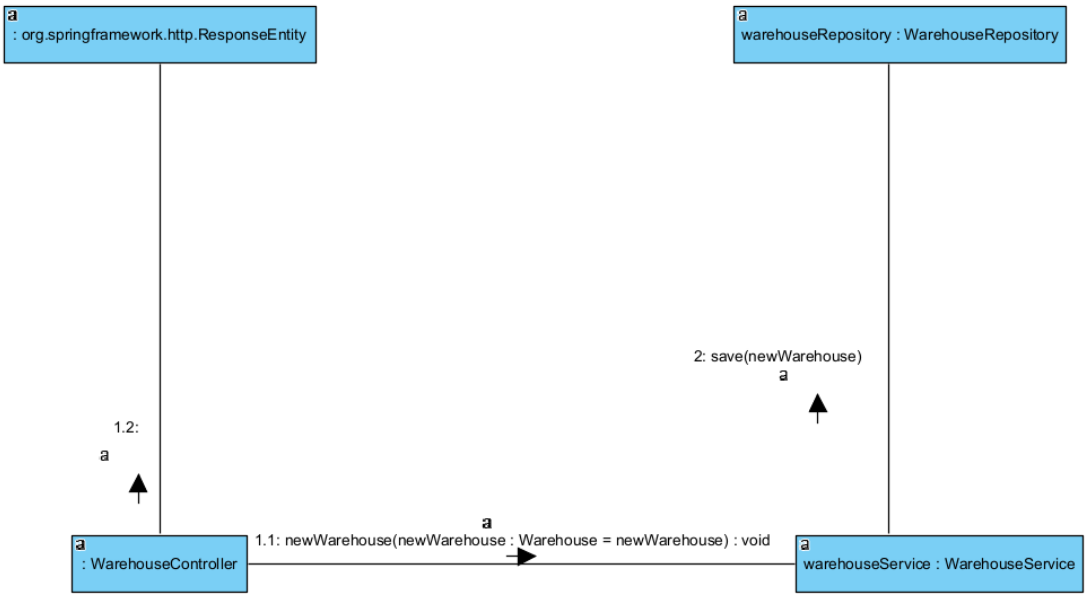


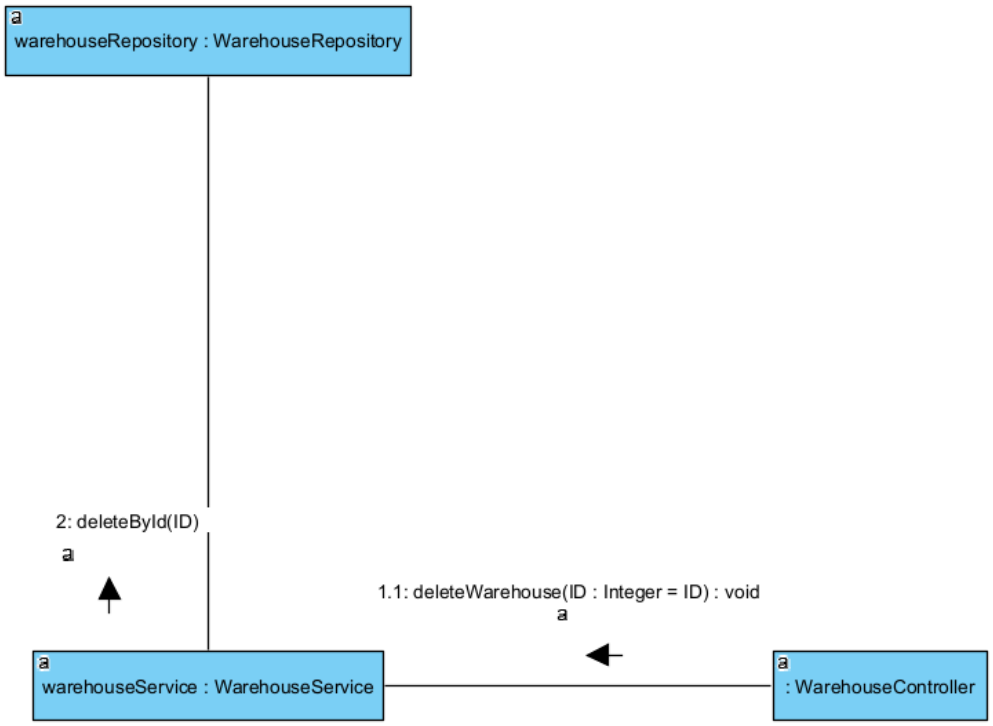

prodRepository : ProductRepository
2: deleteById(ID)
1.1: deleteProduct(ID : Integer = ID) : void
prodService : ProductService
: ProductController

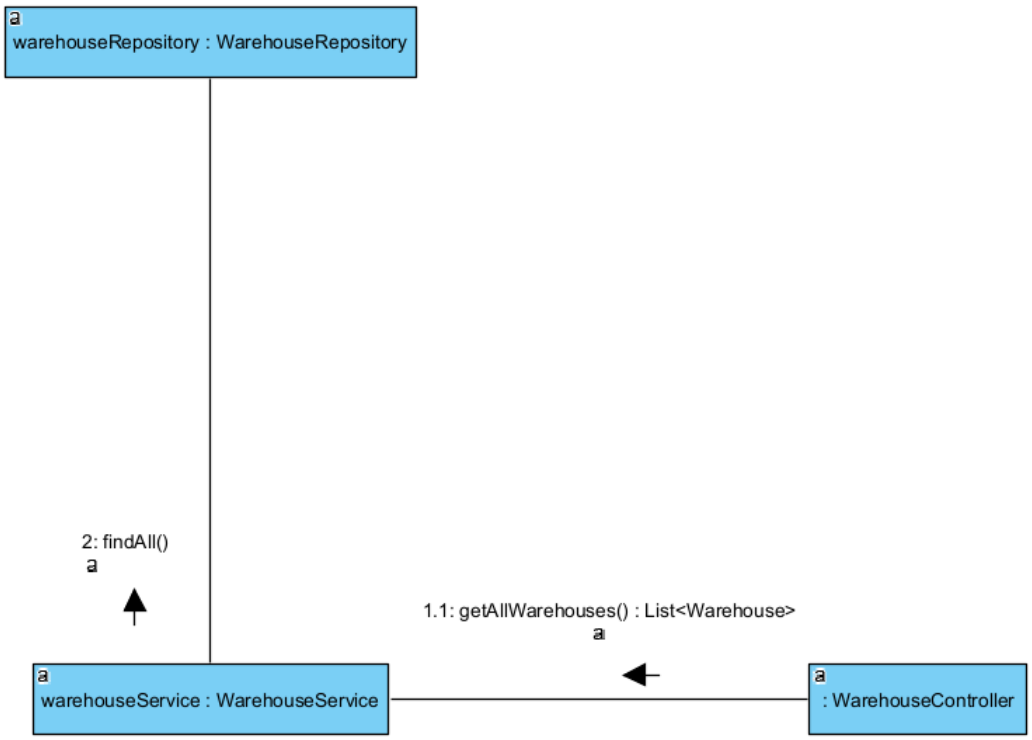

prodRepository : ProductRepository
2: findAll()
1.1: getAllProducts() : List<Product>
prodService : ProductService
: ProductController

Manage warhouses:

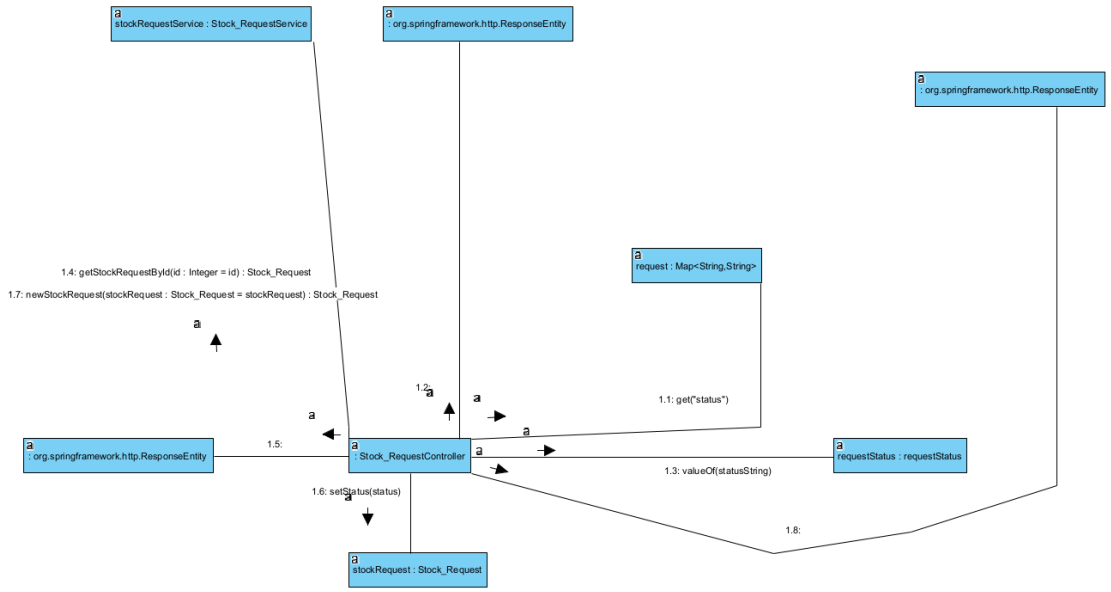


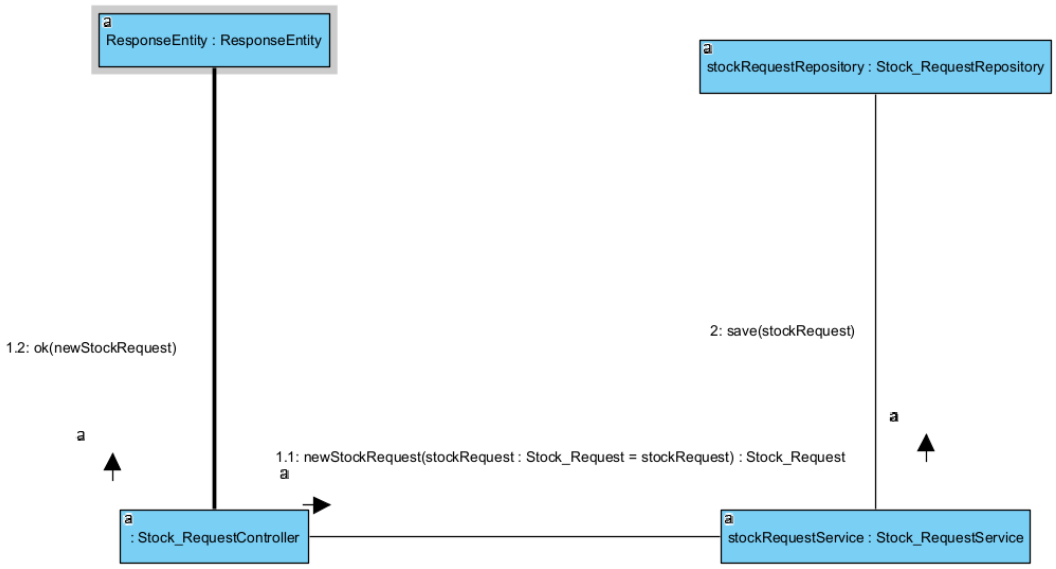


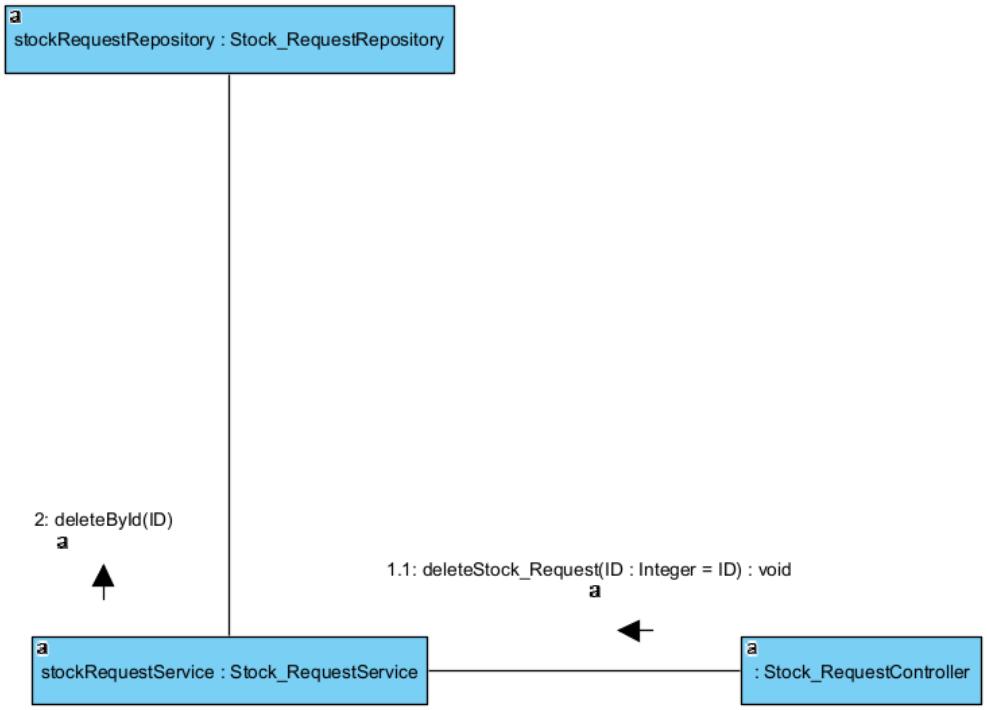

a
warehouseRepository : WarehouseRepository
2: deleteById(ID)
a
1.1: deleteWarehouse(ID : Integer = ID) : void
a
a
warehouseService : WarehouseService
a
: WarehouseController

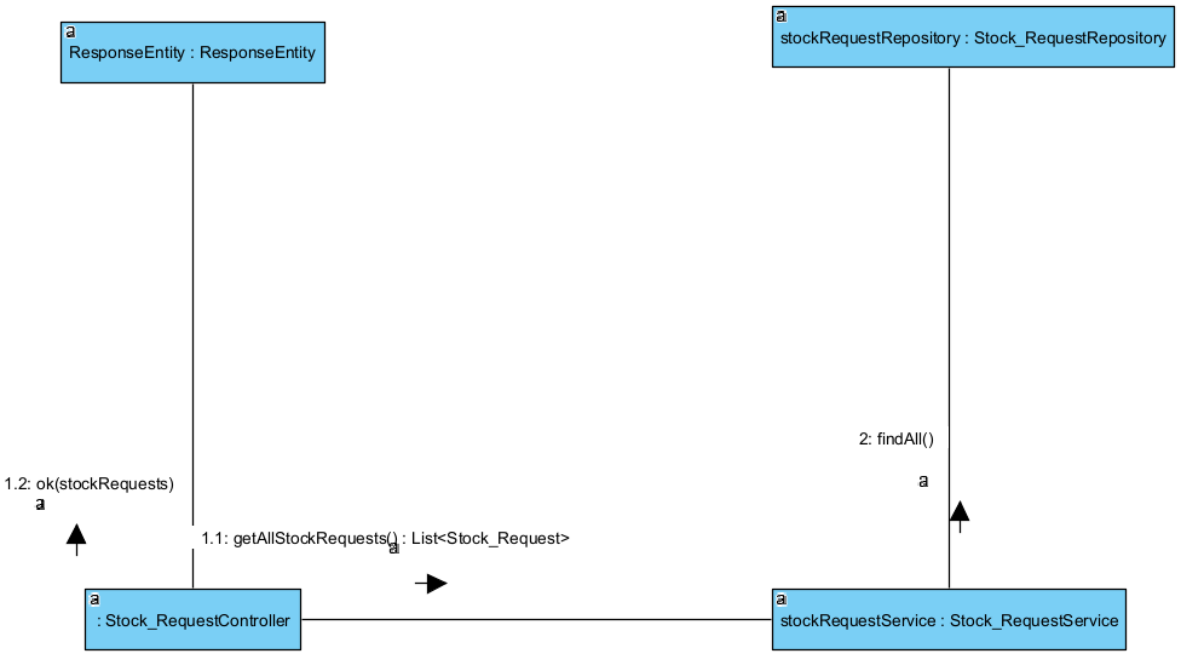

a
warehouseRepository : WarehouseRepository
2: findAll()
a
1.1: getAllWarehouses() : List<Warehouse>
a
a
warehouseService : WarehouseService
a
: WarehouseController

Manage stock requests:






a
stockRequestRepository : Stock_RequestRepository
2: deleteById(ID)
a
1.1: deleteStock_Request(ID : Integer = ID) : void
a
a
stockRequestService : Stock_RequestService
a
: Stock_RequestController


a
ResponseEntity : ResponseEntity
a
stockRequestRepository : Stock_RequestRepository
2: findAll()
a
1.2: ok(stockRequests)
a
1.1: getAllStockRequests() : List<Stock_Request>
a
a
: Stock_RequestController
a
stockRequestService : Stock_RequestService

Manage warehouse products: