PAGE - 1

API

- COMPOSITE
  - `ProductAggregate (1.1)` - PAGE 2
  - `RecommendationSummary (1.2)` - PAGE 3
  - `ReviewSummary (1.3)` - PAGE 3
  - `ServiceAddresses (1.4)` - PAGE 4
  - `ProductCompositeServices <interface> (1.11)` - PAGE 4
- CORE
  - REVIEW
    - `Review (1.5)` - PAGE 5
    - `ReviewService <interface> (1.6)` - PAGE 5
  - RECOMMENDATION
    - `Recommendation (1.7)` - PAGE 6
    - `RecommendationService <interface> (1.8)` - PAGE 6
  - PRODUCT
    - `Product (1.9)` - PAGE 7
    - `ProductService (1.10)` - PAGE 7

UTIL

- http
  - `GlobalControllerExceptionHandler (2.1)` - PAGE 8
  - `HttpErrorInfo (2.2)` - PAGE 9
  - `ServiceUtil (2.3)` - PAGE 10
- Exceptions
  - `InvalidInputException (2.4)` - PAGE 11
  - `NotFoundException (2.5)` - PAGE 11

MICROSERVICES

- Review Service
  - `ReviewServiceImpl (3.1)` - PAGE 12
- Recommendation Service
  - `RecommendationServiceImpl (3.2)` - PAGE 13
- Product Service
  - `ProductServiceImpl (3.3)` - PAGE 14
- Product Composite Service
  - `ProductCompositeIntegration (3.4)` - PAGE 15
  - `ProductCompositeServiceImpl (3.5)` - PAGE 16

PRODUCT AGGREGATE (1.1)

```
package se.magnus.api.composite.product;

import java.util.List;

public class ProductAggregate {
    private final int productId;
    private final String name;
    private final int weight;
    private final List<RecommendationSummary> recommendations;
    private final List<ReviewSummary> reviews;
    private final ServiceAddresses serviceAddresses;

    public ProductAggregate(
        int productId,
        String name,
        int weight,
        List<RecommendationSummary> recommendations,
        List<ReviewSummary> reviews,
        ServiceAddresses serviceAddresses) {

        this.productId = productId;
        this.name = name;
        this.weight = weight;
        this.recommendations = recommendations;
        this.reviews = reviews;
        this.serviceAddresses = serviceAddresses;
    }

    public int getProductId() {
        return productId;
    }

    public String getName() {
        return name;
    }

    public int getWeight() {
        return weight;
    }

    public List<RecommendationSummary> getRecommendations() {
        return recommendations;
    }

    public List<ReviewSummary> getReviews() {
        return reviews;
    }

    public ServiceAddresses getServiceAddresses() {
        return serviceAddresses;
    }
}
```

RECOMMENDATION SUMMARY (1.2)

```java
package se.magnus.api.composite.product;

public class RecommendationSummary {

    private final int recommendationId;
    private final String author;
    private final int rate;

    public RecommendationSummary(int recommendationId, String author, int rate) {
        this.recommendationId = recommendationId;
        this.author = author;
        this.rate = rate;
    }

    public int getRecommendationId() {
        return recommendationId;
    }

    public String getAuthor() {
        return author;
    }

    public int getRate() {
        return rate;
    }
}
```

REVIEW SUMMARY (1.3)

```java
package se.magnus.api.composite.product;

public class ReviewSummary {

    private final int reviewId;
    private final String author;
    private final String subject;

    public ReviewSummary(int reviewId, String author, String subject) {
        this.reviewId = reviewId;
        this.author = author;
        this.subject = subject;
    }

    public int getReviewId() {
        return reviewId;
    }

    public String getAuthor() {
        return author;
    }

    public String getSubject() {
        return subject;
    }
}
```

SERVICE ADDRESSES (1.4)

```
package se.magnus.api.composite.product;

public class ServiceAddresses {
    private final String cmp;
    private final String pro;
    private final String rev;
    private final String rec;

    public ServiceAddresses() {
        cmp = null;
        pro = null;
        rev = null;
        rec = null;
    }

    public ServiceAddresses(String compositeAddress, String productAddress,
                            String reviewAddress, String recommendationAddress) {
        this.cmp = compositeAddress;
        this.pro = productAddress;
        this.rev = reviewAddress;
        this.rec = recommendationAddress;
    }

    public String getCmp() {
        return cmp;
    }

    public String getPro() {
        return pro;
    }

    public String getRev() {
        return rev;
    }

    public String getRec() {
        return rec;
    }
}
```

PRODUCT COMPOSITE SERVICE (1.11)

```
package se.magnus.api.composite.product;

import org.springframework.web.bind.annotation.GetMapping;
import org.springframework.web.bind.annotation.PathVariable;

public interface ProductCompositeService {

    /**
     * Sample usage: curl $HOST:$PORT/product-composite/1
     *
     * @param productId
     * @return the composite product info, if found, else null
     */
    @GetMapping(
        value    = "/product-composite/{productId}",
        produces = "application/json")
    ProductAggregate getProduct(@PathVariable int productId);
}
```

REVIEW (1.5)

```java
package se.magnus.api.core.review;

public class Review {
    private final int productId;
    private final int reviewId;
    private final String author;
    private final String subject;
    private final String content;
    private final String serviceAddress;

    public Review() {
        productId = 0;
        reviewId = 0;
        author = null;
        subject = null;
        content = null;
        serviceAddress = null;
    }

    public Review(int productId, int reviewId, String author, String subject,
                  String content, String serviceAddress) {
        this.productId = productId;
        this.reviewId = reviewId;
        this.author = author;
        this.subject = subject;
        this.content = content;
        this.serviceAddress = serviceAddress;
    }

    public int getProductId() {
        return productId;
    }

    public int getReviewId() {
        return reviewId;
    }

    public String getAuthor() {
        return author;
    }

    public String getSubject() {
        return subject;
    }

    public String getContent() {
        return content;
    }

    public String getServiceAddress() {
        return serviceAddress;
    }
}
```

REVIEW SERVICE (1.6)

```java
package se.magnus.api.core.review;

import org.springframework.web.bind.annotation.GetMapping;
import org.springframework.web.bind.annotation.RequestParam;

import java.util.List;

public interface ReviewService {

    /**
     * Sample usage: curl $HOST:$PORT/review?productId=1
     *
     * @param productId
     * @return
     */
    @GetMapping(
        value    = "/review",
        produces = "application/json")
    List<Review> getReviews(@RequestParam(value = "productId", required = true) int productId);
}
```

RECOMMENDATION (1.7)

```java
package se.magnus.api.core.recommendation;

public class Recommendation {
    private final int productId;
    private final int recommendationId;
    private final String author;
    private final int rate;
    private final String content;
    private final String serviceAddress;

    public Recommendation() {
        productId = 0;
        recommendationId = 0;
        author = null;
        rate = 0;
        content = null;
        serviceAddress = null;
    }

    public Recommendation(int productId, int recommendationId, String author, int rate,
                          String content, String serviceAddress) {
        this.productId = productId;
        this.recommendationId = recommendationId;
        this.author = author;
        this.rate = rate;
        this.content = content;
        this.serviceAddress = serviceAddress;
    }

    public int getProductId() {
        return productId;
    }

    public int getRecommendationId() {
        return recommendationId;
    }

    public String getAuthor() {
        return author;
    }

    public int getRate() {
        return rate;
    }

    public String getContent() {
        return content;
    }

    public String getServiceAddress() {
        return serviceAddress;
    }
}
```

RECOMMENDATION SERVICE (1.8)

```java
package se.magnus.api.core.recommendation;

import org.springframework.web.bind.annotation.GetMapping;
import org.springframework.web.bind.annotation.RequestParam;

import java.util.List;

public interface RecommendationService {

    /**
     * Sample usage: curl $HOST:$PORT/recommendation?productId=1
     *
     * @param productId
     * @return
     */
    @GetMapping(
        value    = "/recommendation",
        produces = "application/json")
    List<Recommendation> getRecommendations(@RequestParam(value = "productId", required = true) int productId);
}
```

PRODUCT (1.9)

```java
package se.magnus.api.core.product;

public class Product {
    private final int productId;
    private final String name;
    private final int weight;
    private final String serviceAddress;

    public Product() {
        productId = 0;
        name = null;
        weight = 0;
        serviceAddress = null;
    }

    public Product(int productId, String name, int weight, String serviceAddress) {
        this.productId = productId;
        this.name = name;
        this.weight = weight;
        this.serviceAddress = serviceAddress;
    }

    public int getProductId() {
        return productId;
    }

    public String getName() {
        return name;
    }

    public int getWeight() {
        return weight;
    }

    public String getServiceAddress() {
        return serviceAddress;
    }
}
```

PRODUCT SERVICE (1.10)

```java
package se.magnus.api.core.product;

import org.springframework.web.bind.annotation.GetMapping;
import org.springframework.web.bind.annotation.PathVariable;

public interface ProductService {

    /**
     * Sample usage: curl $HOST:$PORT/product/1
     *
     * @param productId
     * @return the product, if found, else null
     */
    @GetMapping(
        value    = "/product/{productId}",
        produces = "application/json")
     Product getProduct(@PathVariable int productId);
}
```

GLOBAL CONTROLLER EXCEPTION HANDLER (2.1)

```java
package se.magnus.util.http;

import org.slf4j.Logger;
import org.slf4j.LoggerFactory;
import org.springframework.http.HttpStatus;
import org.springframework.http.server.reactive.ServerHttpRequest;
import org.springframework.web.bind.annotation.*;
import se.magnus.util.exceptions.InvalidInputException;
import se.magnus.util.exceptions.NotFoundException;

import static org.springframework.http.HttpStatus.NOT_FOUND;
import static org.springframework.http.HttpStatus.UNPROCESSABLE_ENTITY;

@RestControllerAdvice
class GlobalControllerExceptionHandler {

    private static final Logger LOG = LoggerFactory.getLogger(GlobalControllerExceptionHandler.class);

    @ResponseStatus(NOT_FOUND)
    @ExceptionHandler(NotFoundException.class)
    public @ResponseBody HttpErrorInfo handleNotFoundExceptions(ServerHttpRequest request, Exception ex) {

        return createHttpErrorInfo(NOT_FOUND, request, ex);
    }

    @ResponseStatus(UNPROCESSABLE_ENTITY)
    @ExceptionHandler(InvalidInputException.class)
    public @ResponseBody HttpErrorInfo handleInvalidInputException(ServerHttpRequest request, Exception ex) {

        return createHttpErrorInfo(UNPROCESSABLE_ENTITY, request, ex);
    }

    private HttpErrorInfo createHttpErrorInfo(HttpStatus httpStatus, ServerHttpRequest request, Exception ex) {
        final String path = request.getPath().pathWithinApplication().value();
        final String message = ex.getMessage();

        LOG.debug("Returning HTTP status: {} for path: {}, message: {}", httpStatus, path, message);
        return new HttpErrorInfo(httpStatus, path, message);
    }
}
```

HTTP ERROR INFO (2.2)

```java
package se.magnus.util.http;

import org.springframework.http.HttpStatus;
import java.time.ZonedDateTime;

public class HttpErrorInfo {
    private final ZonedDateTime timestamp;
    private final String path;
    private final HttpStatus httpStatus;
    private final String message;


    public HttpErrorInfo() {
        timestamp = null;
        this.httpStatus = null;
        this.path = null;
        this.message = null;
    }

    public HttpErrorInfo(HttpStatus httpStatus, String path, String message) {
        timestamp = ZonedDateTime.now();
        this.httpStatus = httpStatus;
        this.path = path;
        this.message = message;
    }

    public ZonedDateTime getTimestamp() {
        return timestamp;
    }

    public String getPath() {
        return path;
    }

    public int getStatus() {
        return httpStatus.value();
    }

    public String getError() {
        return httpStatus.getReasonPhrase();
    }

    public String getMessage() {
        return message;
    }
}
```

SERVICE UTIL (2.3)

```
package se.magnus.util.http;

import org.slf4j.Logger;
import org.slf4j.LoggerFactory;
import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.beans.factory.annotation.Value;
import org.springframework.stereotype.Component;

import java.net.InetAddress;
import java.net.UnknownHostException;

@Component
public class ServiceUtil {
    private static final Logger LOG = LoggerFactory.getLogger(ServiceUtil.class);

    private final String port;

    private String serviceAddress = null;

    @Autowired
    public ServiceUtil(
        @Value("${server.port}") String port) {

        this.port = port;
    }

    public String getServiceAddress() {
        if (serviceAddress == null) {
            serviceAddress = findMyHostname() + "/" + findMyIpAddress() + ":" + port;
        }
        return serviceAddress;
    }

    private String findMyHostname() {
        try {
            return InetAddress.getLocalHost().getHostName();
        } catch (UnknownHostException e) {
            return "unknown host name";
        }
    }

    private String findMyIpAddress() {
        try {
            return InetAddress.getLocalHost().getHostAddress();
        } catch (UnknownHostException e) {
            return "unknown IP address";
        }
    }

}
```

INVALID INPUT EXCEPTION (2.4)

```
package se.magnus.util.exceptions;

public class InvalidInputException extends RuntimeException {
    public InvalidInputException() {
    }

    public InvalidInputException(String message) {
        super(message);
    }

    public InvalidInputException(String message, Throwable cause) {
        super(message, cause);
    }

    public InvalidInputException(Throwable cause) {
        super(cause);
    }
}
```

NOT FOUND EXCEPTION (2.5)

```
package se.magnus.util.exceptions;

public class NotFoundException extends RuntimeException {
    public NotFoundException() {
    }

    public NotFoundException(String message) {
        super(message);
    }

    public NotFoundException(String message, Throwable cause) {
        super(message, cause);
    }

    public NotFoundException(Throwable cause) {
        super(cause);
    }
}
```

REVIEW SERVICE IMPL (3.1)

```java
package se.magnus.microservices.core.review.services;

import org.slf4j.Logger;
import org.slf4j.LoggerFactory;
import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.web.bind.annotation.RestController;
import se.magnus.api.core.review.Review;
import se.magnus.api.core.review.ReviewService;
import se.magnus.util.exceptions.InvalidInputException;
import se.magnus.util.http.ServiceUtil;

import java.util.ArrayList;
import java.util.List;

@RestController
public class ReviewServiceImpl implements ReviewService {

    private static final Logger LOG = LoggerFactory.getLogger(ReviewServiceImpl.class);

    private final ServiceUtil serviceUtil;

    @Autowired
    public ReviewServiceImpl(ServiceUtil serviceUtil) {
        this.serviceUtil = serviceUtil;
    }

    @Override
    public List<Review> getReviews(int productId) {

        if (productId < 1) throw new InvalidInputException("Invalid productId: " + productId);

        if (productId == 213) {
            LOG.debug("No reviews found for productId: {}", productId);
            return  new ArrayList<>();
        }

        List<Review> list = new ArrayList<>();

        list.add(new Review(productId, 1, "Author 1", "Subject 1", "Content 1",serviceUtil.getServiceAddress()));

        list.add(new Review(productId, 2, "Author 2", "Subject 2", "Content 2", serviceUtil.getServiceAddress()));

        list.add(new Review(productId, 3, "Author 3", "Subject 3", "Content 3", serviceUtil.getServiceAddress()));

        LOG.debug("/reviews response size: {}", list.size());

        return list;
    }
}
```

RECOMMENDATION SERVICE IMPL (3.2)

```
package se.magnus.microservices.core.recommendation.services;

import org.slf4j.Logger;
import org.slf4j.LoggerFactory;
import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.web.bind.annotation.RestController;
import se.magnus.api.core.recommendation.Recommendation;
import se.magnus.api.core.recommendation.RecommendationService;
import se.magnus.util.exceptions.InvalidInputException;
import se.magnus.util.http.ServiceUtil;

import java.util.ArrayList;
import java.util.List;

@RestController
public class RecommendationServiceImpl implements RecommendationService {

    private static final Logger LOG = LoggerFactory.getLogger(RecommendationServiceImpl.class);

    private final ServiceUtil serviceUtil;

    @Autowired
    public RecommendationServiceImpl(ServiceUtil serviceUtil) {
        this.serviceUtil = serviceUtil;
    }

    @Override
    public List<Recommendation> getRecommendations(int productId) {

        if (productId < 1) throw new InvalidInputException("Invalid productId: " + productId);

        if (productId == 113) {
            LOG.debug("No recommendations found for productId: {}", productId);
            return  new ArrayList<>();
        }

        List<Recommendation> list = new ArrayList<>();
        list.add(new Recommendation(productId, 1, "Author 1", 1, "Content 1", serviceUtil.getServiceAddress()));
        list.add(new Recommendation(productId, 2, "Author 2", 2, "Content 2", serviceUtil.getServiceAddress()));
        list.add(new Recommendation(productId, 3, "Author 3", 3, "Content 3", serviceUtil.getServiceAddress()));

        LOG.debug("/recommendation response size: {}", list.size());

        return list;
    }
}
```

PRODUCT SERVICE IMPL (3.3)

```
package se.magnus.microservices.core.product.services;

import org.slf4j.Logger;
import org.slf4j.LoggerFactory;
import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.web.bind.annotation.RestController;
import se.magnus.api.core.product.Product;
import se.magnus.api.core.product.ProductService;
import se.magnus.util.exceptions.InvalidInputException;
import se.magnus.util.exceptions.NotFoundException;
import se.magnus.util.http.ServiceUtil;

@RestController
public class ProductServiceImpl implements ProductService {

    private static final Logger LOG = LoggerFactory.getLogger(ProductServiceImpl.class);

    private final ServiceUtil serviceUtil;

    @Autowired
    public ProductServiceImpl(ServiceUtil serviceUtil) {
        this.serviceUtil = serviceUtil;
    }

    @Override
    public Product getProduct(int  roductid) {
        LOG.debug("/product return the found product for  roductid={}",  roductid);

        if ( roductid < 1) throw new InvalidInputException("Invalid  roductid: " +  roductid);

        if ( roductid == 13) throw new NotFoundException("No product found for  roductid: " +  roductid);

        return new Product( roductid, "name-" +  roductid, 123, serviceUtil.getServiceAddress());
    }
}
```

PRODUCT COMPOSITE INTEGRATION (3.4)

```
package se.magnus.microservices.composite.product.services;

import static org.springframework.http.HttpMethod.GET;
import static org.springframework.http.HttpStatus.NOT_FOUND;
import static org.springframework.http.HttpStatus.UNPROCESSABLE_ENTITY;

@Component
public class ProductCompositeIntegration implements ProductService, RecommendationService, ReviewService {

    private static final Logger LOG = LoggerFactory.getLogger(ProductCompositeIntegration.class);

    private final RestTemplate restTemplate;
    private final ObjectMapper mapper;

    private final String productServiceUrl;
    private final String recommendationServiceUrl;
    private final String reviewServiceUrl;

    @Autowired
    public ProductCompositeIntegration(
        RestTemplate restTemplate,
        ObjectMapper mapper,

        @Value("${app.product-service.host}") String productServiceHost,
        @Value("${app.product-service.port}") int    productServicePort,

        @Value("${app.recommendation-service.host}") String recommendationServiceHost,
        @Value("${app.recommendation-service.port}") int    recommendationServicePort,

        @Value("${app.review-service.host}") String reviewServiceHost,
        @Value("${app.review-service.port}") int    reviewServicePort
    ) {

        this.restTemplate = restTemplate;
        this.mapper = mapper;

        productServiceUrl        = "http://" + productServiceHost + ":" + productServicePort + "/product/";
        recommendationServiceUrl = "http://" + recommendationServiceHost + ":" + recommendationServicePort +
"/recommendation?productId=";
        reviewServiceUrl         = "http://" + reviewServiceHost + ":" + reviewServicePort + "/review?productId=";
    }
    public Product getProduct(int productId) {
        try {
            String url = productServiceUrl + productId;
            LOG.debug("Will call getProduct API on URL: {}", url);

            Product product = restTemplate.getForObject(url, Product.class);
            LOG.debug("Found a product with id: {}", product.getProductId());

            return product;

        } catch (HttpClientErrorException ex) {

            HttpStatusCode statusCode = ex.getStatusCode();
            if (NOT_FOUND.equals(statusCode)) {
                throw new NotFoundException(getErrorMessage(ex));
            } else if (UNPROCESSABLE_ENTITY.equals(statusCode)) {
                throw new InvalidInputException(getErrorMessage(ex));
            }
            LOG.warn("Got a unexpected HTTP error: {}, will rethrow it", ex.getStatusCode());
            LOG.warn("Error body: {}", ex.getResponseBodyAsString());
            throw ex;
        }
    }
    private String getErrorMessage(HttpClientErrorException ex) {
        try {
            return mapper.readValue(ex.getResponseBodyAsString(), HttpErrorInfo.class).getMessage();
        } catch (IOException ioex) {
            return ex.getMessage();
        }
    }
    public List<Recommendation> getRecommendations(int productId) {

       try {
            String url = recommendationServiceUrl + productId;

            LOG.debug("Will call getRecommendations API on URL: {}", url);
            List<Recommendation> recommendations = restTemplate.exchange(url, GET, null,
                                                                         new ParameterizedTypeReference<List<Recommendation>>() {}).getBody();

            LOG.debug("Found {} recommendations for a product with id: {}", recommendations.size(), productId);
            return recommendations;

        } catch (Exception ex) {
            LOG.warn("Got an exception while requesting recommendations, return zero recommendations: {}", ex.getMessage());
            return new ArrayList<>();
        }
    }
```

```
    public List<Review> getReviews(int productId) {

        try {
            String url = reviewServiceUrl + productId;

            LOG.debug("Will call getReviews API on URL: {}", url);
            List<Review> reviews = restTemplate.exchange(url, GET, null,
                                                         new ParameterizedTypeReference<List<Review>>() {}).getBody();

            LOG.debug("Found {} reviews for a product with id: {}", reviews.size(), productId);
            return reviews;

        } catch (Exception ex) {
            LOG.warn("Got an exception while requesting reviews, return zero reviews: {}", ex.getMessage());
            return new ArrayList<>();
        }
    }
}
```

PRODUCT COMPOSITE SERVICE IMPL (3.5)

```
package se.magnus.microservices.composite.product.services;

import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.web.bind.annotation.RestController;
import se.magnus.api.composite.product.*;
import se.magnus.api.core.product.Product;
import se.magnus.api.core.recommendation.Recommendation;
import se.magnus.api.core.review.Review;
import se.magnus.util.exceptions.NotFoundException;
import se.magnus.util.http.ServiceUtil;

import java.util.List;
import java.util.stream.Collectors;

@RestController
public class ProductCompositeServiceImpl implements ProductCompositeService {

    private final ServiceUtil serviceUtil;
    private  ProductCompositeIntegration integration;

    @Autowired
    public ProductCompositeServiceImpl(ServiceUtil serviceUtil, ProductCompositeIntegration integration) {
        this.serviceUtil = serviceUtil;
        this.integration = integration;
    }
    @Override
    public ProductAggregate getProduct(int productId) {

        Product product = integration.getProduct(productId);
        if (product == null) throw new NotFoundException("No product found for productId: " + productId);

        List<Recommendation> recommendations = integration.getRecommendations(productId);

        List<Review> reviews = integration.getReviews(productId);

        return createProductAggregate(product, recommendations, reviews, serviceUtil.getServiceAddress());
    }
    private ProductAggregate createProductAggregate(Product product,
                                                    List<Recommendation> recommendations,
                                                    List<Review> reviews, String serviceAddress) {
        // 1. Setup product info
        int productId = product.getProductId();
        String name = product.getName();
        int weight = product.getWeight();

        // 2. Copy summary recommendation info, if available
        List<RecommendationSummary> recommendationSummaries = (recommendations == null) ? null :
             recommendations.stream()
                .map(r -> new RecommendationSummary(r.getRecommendationId(), r.getAuthor(), r.getRate()))
                .collect(Collectors.toList());

        // 3. Copy summary review info, if available
        List<ReviewSummary> reviewSummaries = (reviews == null)  ? null :
            reviews.stream()
                .map(r -> new ReviewSummary(r.getReviewId(), r.getAuthor(), r.getSubject()))
                .collect(Collectors.toList());
```

```
        // 4. Create info regarding the involved microservices addresses
        String productAddress = product.getServiceAddress();
        String reviewAddress = (reviews != null && reviews.size() > 0) ? reviews.get(0).getServiceAddress() :
"";
        String recommendationAddress = (recommendations != null && recommendations.size() > 0) ?
recommendations.get(0).getServiceAddress() : "";
        ServiceAddresses serviceAddresses = new ServiceAddresses(serviceAddress, productAddress, reviewAddress,
recommendationAddress);

        return new ProductAggregate(productId, name, weight, recommendationSummaries, reviewSummaries,
serviceAddresses);
    }
}
```