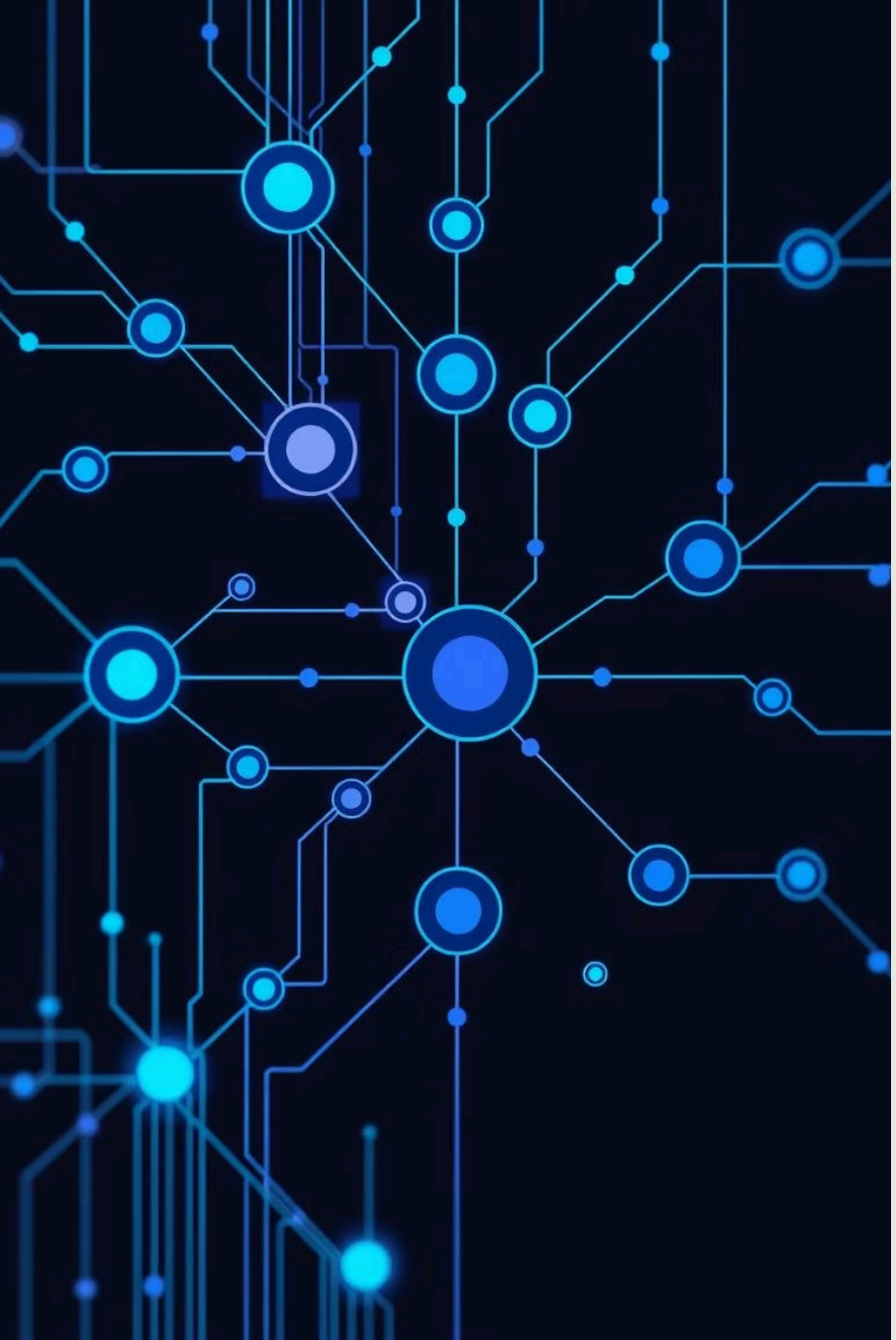
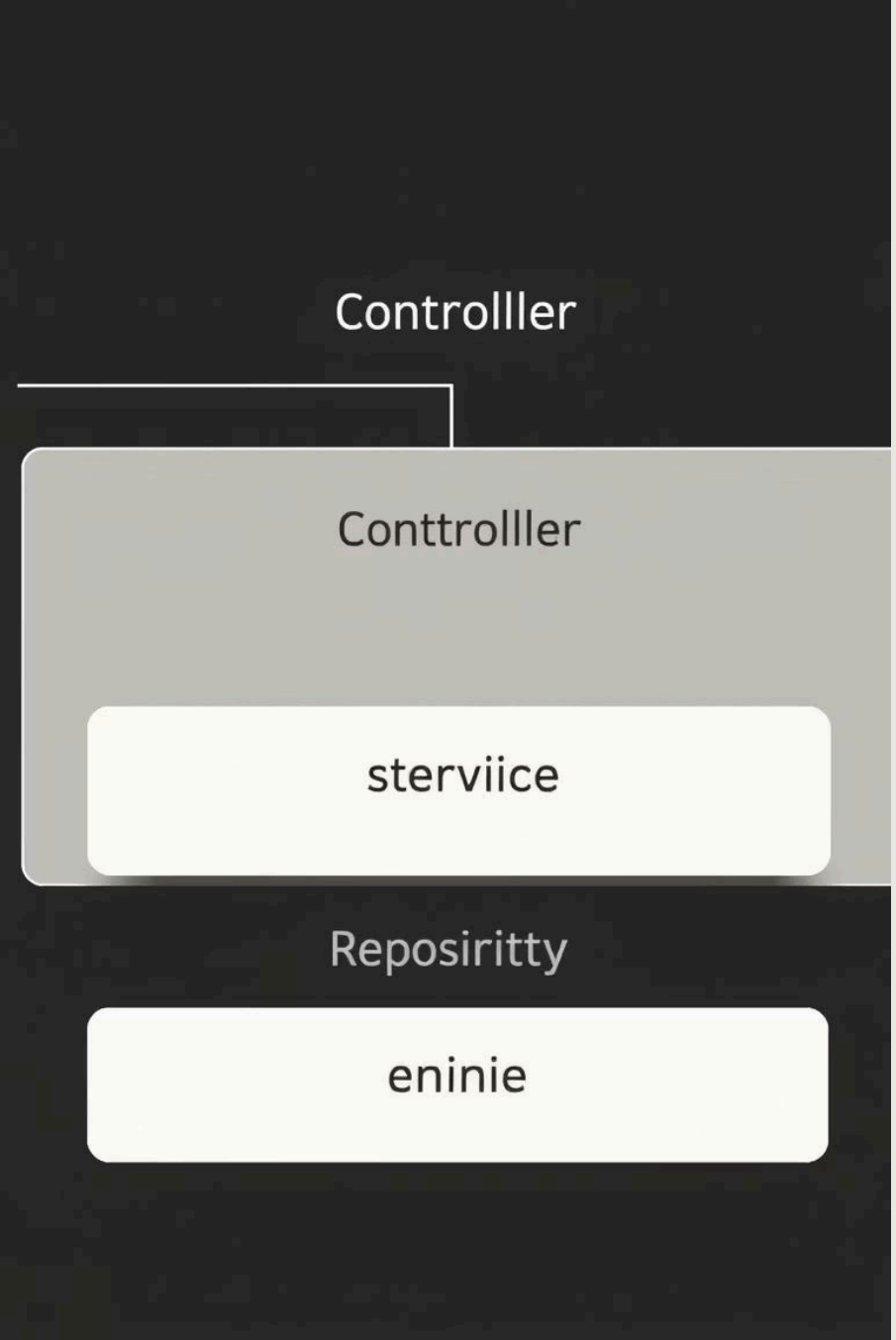


DETAILED OVERVIEW OF MYNTRA CLONE BACKEND PROJECT

This presentation provides an in-depth analysis of the Myntra Clone Backend Project. We will explore its architecture, key Java features, robust validation mechanisms, and comprehensive error handling strategies. Additionally, we will touch upon the security configurations and the integration with frontend technologies, offering a holistic view of the project's capabilities and design principles.

A by Aadvik Krishna



PROJECT INTRODUCTION

PURPOSE

The project aims to replicate core backend features of the Myntra e-commerce platform, focusing on robust and scalable solutions.

ARCHITECTURE

It employs a layered architecture (Controller → Service → Repository) with a RESTful API structure, built on Spring Boot 2.7.14 and Java SE 17.

FOLDER STRUCTURE RECAP

Folder	Purpose
controller/	Manages API requests/responses
service/	Business logic layer
repository/	DAO, database interaction
model/	Entities, data structure
security/	Auth & JWT
util/	Helper utilities
resources/	application.properties, static files

JAVA CODE - KEY CONCEPTS USED

LOMBOK ANNOTATIONS

Utilizes @Data, @Getter, @Setter, @NoArgsConstructor, @AllArgsConstructor to significantly reduce boilerplate code, enhancing readability and maintainability.

JPA ANNOTATIONS

Employs @Entity, @Id, @GeneratedValue, @ManyToOne, and others for seamless Object-Relational Mapping (ORM), simplifying database interactions.

JAVA 17 FEATURES

Leverages modern Java 17 features such as Records, Enhanced Switch Expressions, and Text Blocks for cleaner and more efficient code construction.

DATA VALIDATION

JSR-380 BEAN VALIDATION

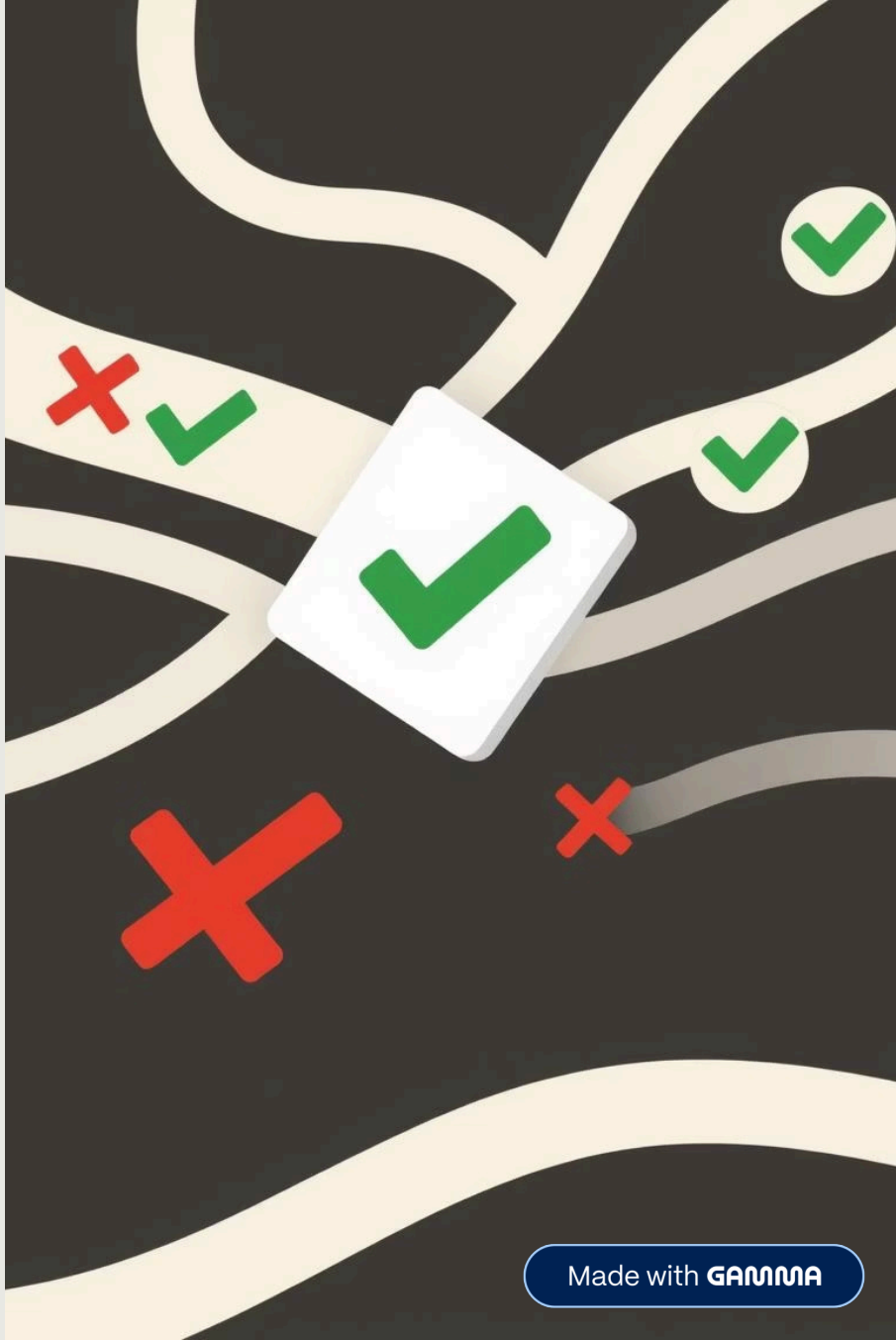
Uses Hibernate Validator with annotations like @NotBlank, @Size, @Email, and @Pattern for robust data integrity.

EXAMPLE IN USER.JAVA

```
@NotBlank(message = "Email is mandatory")
@Email(message = "Invalid email format") private String email;
```

CUSTOM VALIDATORS

(If present) Implements custom validation logic for specific rules like password strength or username formats.



ERROR HANDLING

GLOBAL EXCEPTION HANDLING

Centralized error management using @ControllerAdvice for consistent responses across the application.

EXCEPTIONHANDLER METHODS

Utilizes @ExceptionHandler methods, e.g.,

```
@ExceptionHandler(MethodArgu
mentNotValidException.class)
public ResponseEntity
handleValidationError(...) { ... }
```

for specific error types.

STANDARDIZED ERROR RESPONSE

Custom ErrorResponse class includes timestamp, status, message, and details for clear error communication.

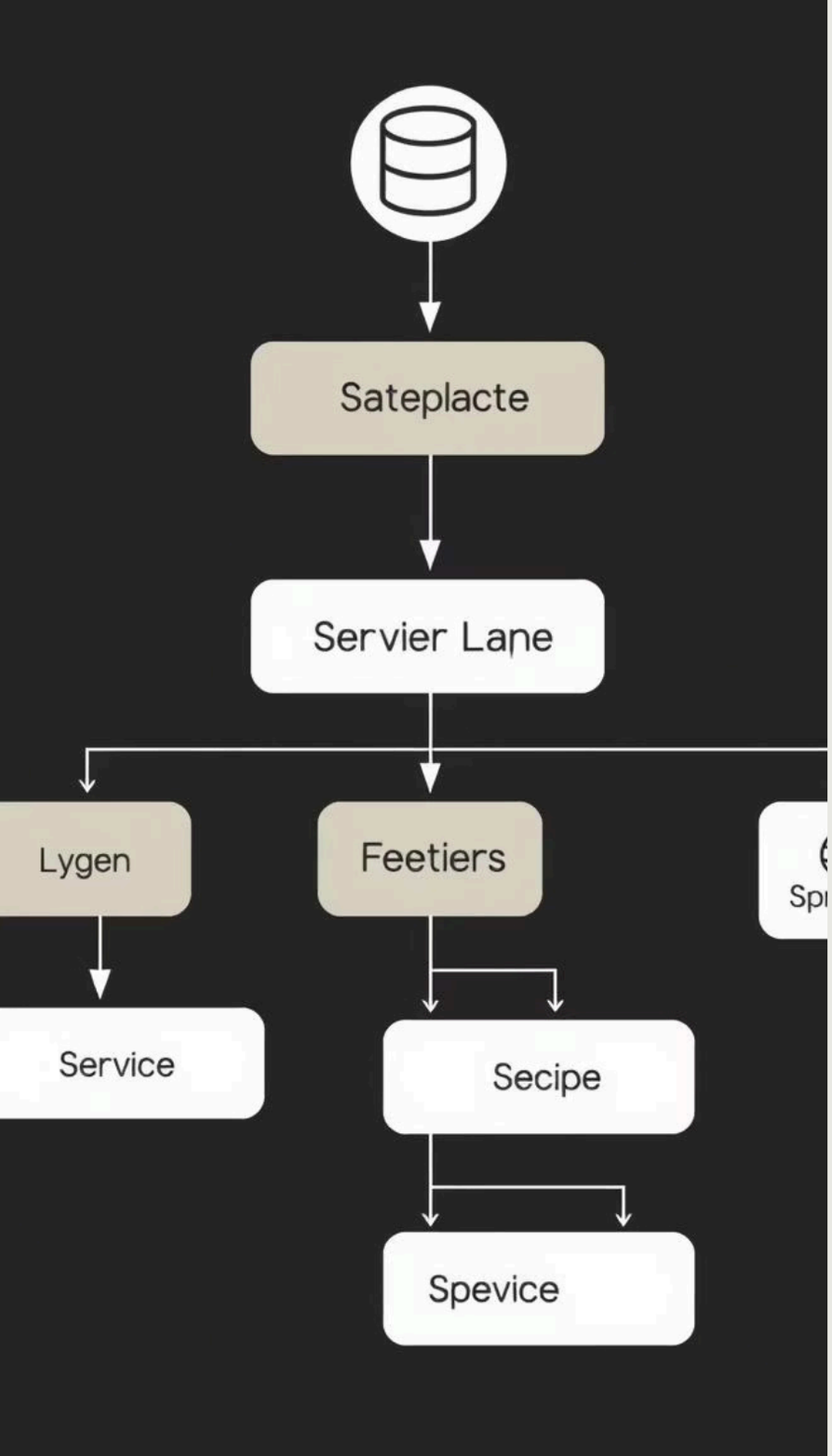
SECURITY CONFIGURATION

JWT AUTHENTICATION

JWTUtil.java handles token generation and validation, while AuthenticationFilter intercepts HTTP requests for secure access.

SPRING SECURITY SETUP

SecurityConfig.java defines role-based access, public/private endpoints, and integrates BCryptPasswordEncoder for secure password hashing.



SERVICE LAYER EXPLAINED

BUSINESS LOGIC

Core business logic resides in services like UserService and ProductService, annotated with @Service.

DEPENDENCY INJECTION

Services use @Autowired or constructor injection to manage dependencies and interact with repositories.

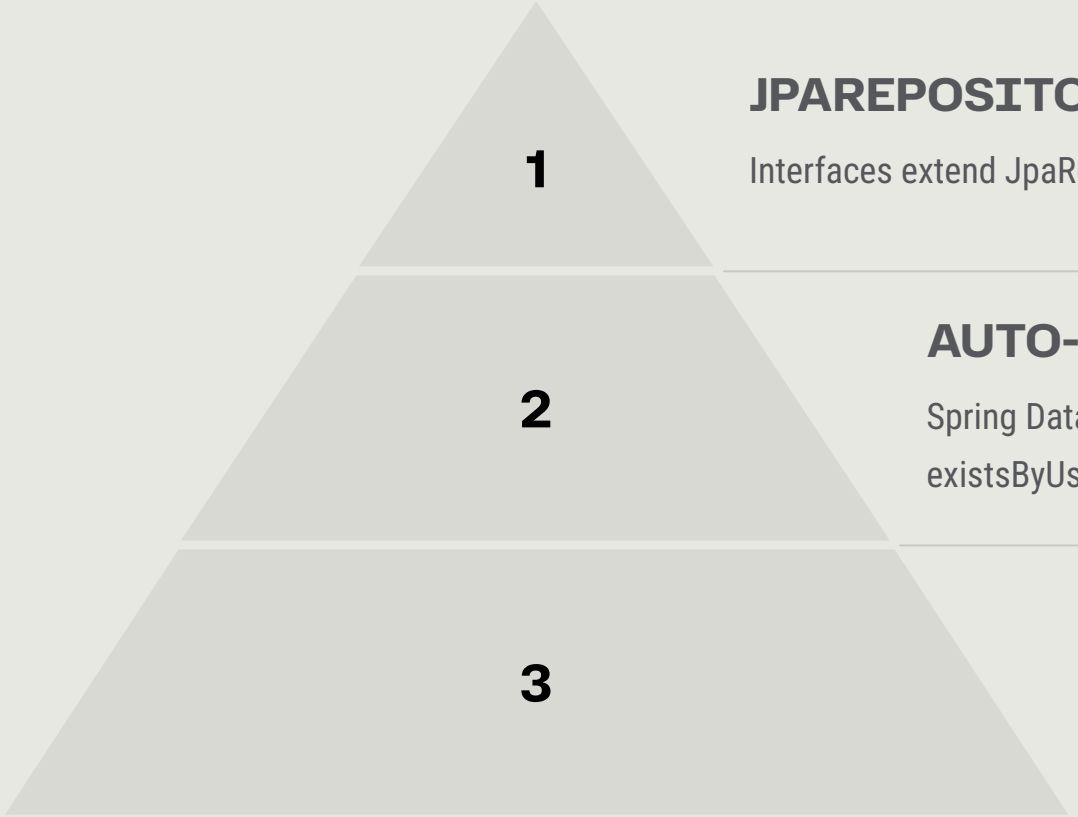
DATABASE OPERATIONS

Service methods call repository methods to perform database operations, ensuring data integrity and consistency.

EXAMPLE METHOD

```
public User register(User user) { if (userRepository.existsByEmail(user.getEmail())) { throw new UserAlreadyExistsException(...); } return userRepository.save(user); }
```

REPOSITORY LAYER



JPAREPOSITORY EXTENSION

Interfaces extend JpaRepository, inheriting powerful CRUD operations and query capabilities.

AUTO-IMPLEMENTED METHODS

Spring Data automatically implements methods like findByEmail() and existsByUsername(), reducing manual coding.

CUSTOM QUERIES

(If used) Custom queries can be defined with @Query for complex data retrieval scenarios.

CONCLUSION & NEXT STEPS

KEY TAKEAWAYS

The Myntra Clone Backend Project demonstrates a robust, secure, and scalable architecture, leveraging modern Java and Spring Boot features for efficient e-commerce operations.

FUTURE ENHANCEMENTS

Potential next steps include implementing advanced search functionalities, integrating payment gateways, and expanding user management features.

COLLABORATION

We encourage further collaboration and contributions to enhance the project's capabilities and address new challenges.