IMPORTANT ANNOTATIONS

Annotations	**Usage**
@SpringBootApplication	main annotation used in Spring Boot applications
@ComponentScan	Scans for beans/components (@Component, @Service, @Repository, @Controller)
@Configuration	Used to mark a class as a configuration class where Spring beans are defined. Inside this class, you can use @Bean methods to create beans manually.
@EnableAutoConfiguration	Tells Spring Boot to automatically configure beans depending on the dependencies
@Bean	Declares a Spring bean inside a @Configuration class.
@Autowired	Enables dependency injection. Spring automatically injects the required bean by type.
@Component	annotation that marks a class as a Spring-managed bean.
@Service	A specialized form of @Component, used for service layer classes (business logic). It improves readability and indicates the role of the class.

@Repository	Used for Data Access Layer classes. Provides exception translation (converts SQL exceptions into Spring's DataAccessException).
@Controller	Marks a class as a Spring MVC controller to handle web requests.
@RestController	Combination of @Controller + @ResponseBody.Returns data (JSON/XML) instead of views. Mostly used in REST APIs.
@RequestMapping	Maps HTTP requests to a handler method. Can be used at class level (base URL) and method level (specific endpoints).
@GetMapping, @PostMapping, @PutMapping, @DeleteMapping	@GetMapping → For GET requests @PostMapping → For POST requests @PutMapping → For PUT requests @DeleteMapping → For DELETE requests@PathVariable
@PathVariable	Extracts values from the URI path.
@RequestParam	Extracts query parameters from the URL.
@RequestBody	Maps the HTTP request body (JSON, XML) to a method parameter.

@ResponseBody	Tells Spring that the method's return value should go directly to the HTTP response body (not a view).
@Entity	Marks a class as a JPA entity. A JPA entity is a Java class mapped to a database table
@Id	Marks a field as the **primary key** of the entity.
@NotBlank	Ensures that a **string field is not null, not empty**
@DiscriminatorValue	JPA uses a **discriminator column** in the database to identify which child type a row belongs to
@Embeddable	defines a class whose fields can be embedded into another entity.
@Embedded	used in an entity to include the embeddable class.
@ModelAttribute	Used to bind request parameter to method parameter

@Valid	Validation on object passed to controller.This ensures not null,blank etc.
@Table	Used to define name od db
@GeneratedValue	Specify auto generation of primary key values
@FeignClient	Allows to declare REST Clients. Helps to make HTTP request to other microservices
@CircuitBreaker	Used to prevent system from repeatedly trying to perform operation that is likely to fail
@Qualifier	Helps resolve conflict when multiple beans of same type exist.
@PropertySource	Loads external properties file.
@ConfigurationProperties	Maps properties from application.properties to a class
@RestControllerAdvice	Global exception handler + data advice for REST APIs

@EnableScheduling	Enables scheduled tasks(@Scheduled)
@EnableAsync	Enable asynchronous methods(@Async)
@Profile	Activates beans based on environment
@EnableEurekaServer	Turns an app into a Eureka Discovery server
@EnableEurekaClient	Registers a service as Eureka clients
@EnableFeignClients	Enables Feign
@EnableConfigServer	Enables Spring cloud config server
@EnableDiscoveryClient	General discovery client
@ControllerAdvice	Global exception handling for MVC controllers.

@ExceptionHandler	Handles a specific exception in a controller/advice.
@LoadBalanced	Used with RestTemplate for client side load balancing
@ResponseStatus	Maps an exception to an HTTP status code
@ResponseBody	Returns data directly in the response body (used in REST).