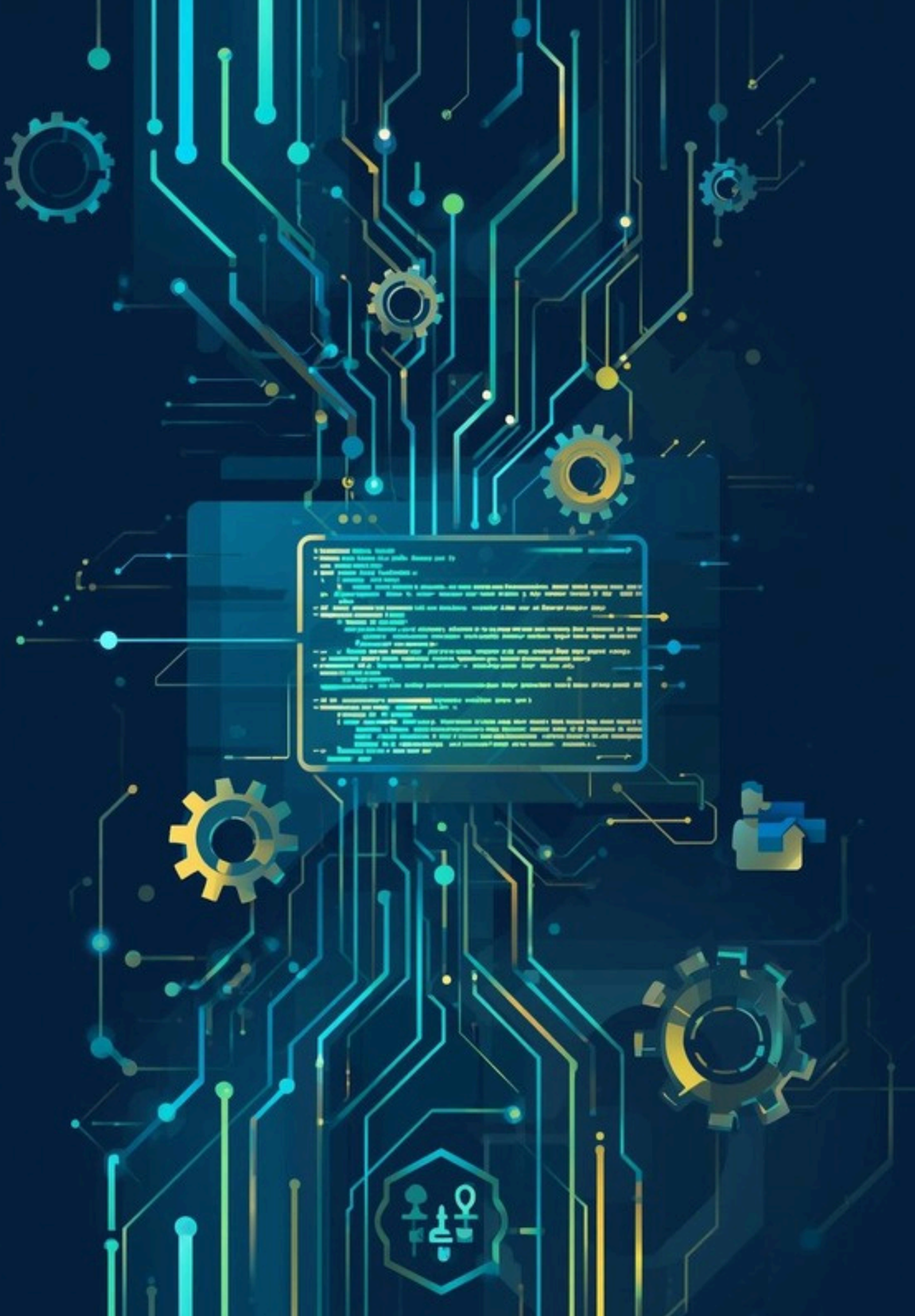
Tech Innovations

UNIT TESTING & AI PROMPT

PRESENTED BY

AI_02

CORE FEATURE:
VIEW FEATURE: “BOOKING BIKE”



WHY CHOOSE IT?

- **Luồng nghiệp vụ rõ ràng: chọn xe → chọn ngày → kiểm tra khả dụng → tính tiền → thêm vào giỏ. Nhiều điểm đo kiểm.**
- **Edge cases dày: trùng lịch, ngày không hợp lệ, giới hạn số ngày, định dạng tiền.**
- **Có phụ thuộc: OrderService (availability, conflicts), MotorbikeService (bike data), chính sách cọc/giá.**
- **Phù hợp chuẩn báo cáo QA: bảng GWT, nhóm theo component, cells ngắn gọn.**

FUNCTIONS REQUIRING TESTING

BOOKINGSERVICE FUNCTIONS (7)

- createBooking(BookingForm, UUID) → Booking
- calculateTotalAmount(Booking) → BigDecimal
- updateBookingStatus(Integer, BookingStatus) → Booking
- cancelBooking(Integer, String) → Booking
- getBookingById(Integer) → Optional<Booking>
- findBookingsByCustomer(UUID) → List<Booking>
- validateBookingTime(LocalDateTime) → boolean

BOOKINGCONTROLLER FUNCTIONS (6)

- showBookingForm(Model) → String
- createBooking(BookingForm, BindingResult, RedirectAttributes) → String
- getBookingDetails(Integer, Model) → String
- cancelBooking(Integer, RedirectAttributes) → String
- showBookingHistory(Model) → String
- updateBookingStatus(Integer, BookingStatus, RedirectAttributes) → String

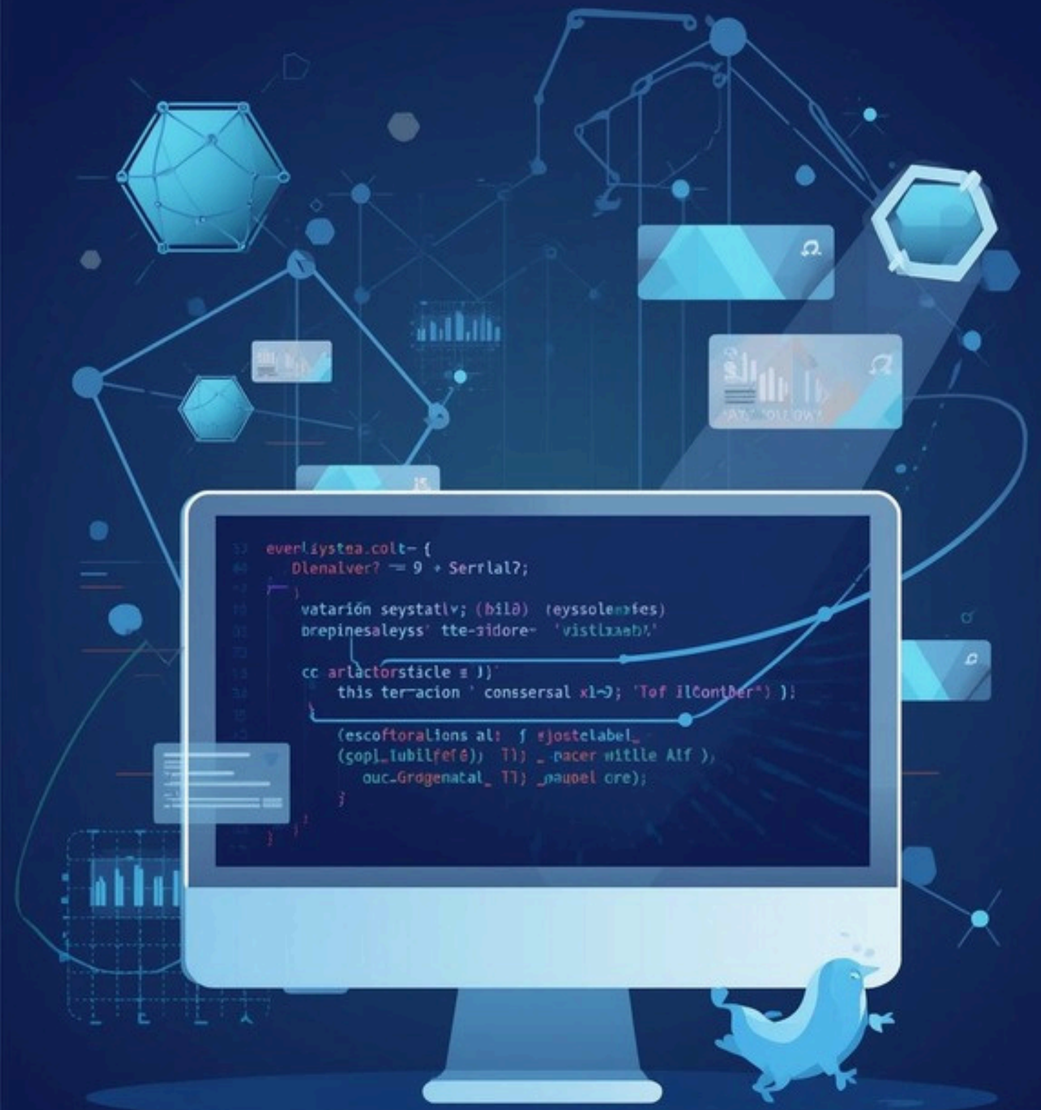
INTEGRATION TEST FUNCTIONS (3)

- testBookingFlow_EndToEnd()
- testBookingAmountCalculation_ShouldBeCorrect()
- testBookingStatusFlow_ShouldUpdateCorrectly()

SIMPLE TEST FUNCTIONS (3)

- testBasicAssertion()
- testMathCalculation()
- testStringOperations()

PROMT 1: AI Code Insight



1. AI CODE INSIGHT

INPUT PROMT:

ROLE & OBJECTIVE 🎯

- Role: Senior QA Engineer phân tích Booking của hệ thống JSP/Servlet.
- Liệt kê đúng hàm cần test ở Service và Servlet, chỉ ra quy tắc nghiệp vụ và edge cases để làm nền cho Prompt 2–7. Không đổi chữ ký hàm. Không chạm DB, chỉ mock.

CONTEXT & SCOPE 🌐

- Stack: Jakarta EE, JSP/JSTL, Tomcat, Maven.
- Phụ thuộc cần mock: DAO tầng dưới, HttpServlet* (Request, Response, Session, RequestDispatcher).
- Nguyên tắc: Validate input ngày, quyền truy cập, trạng thái xe; kiểm tra forward/redirect; không tiết lộ tồn tại tài khoản; không truy cập DB.
- Scope: Service: IOrderService, IMotorbikeService. Servlet: BookingServlet.doPost, MotorbikeDetailServlet.doGet, MyOrdersServlet.doGet.

FUNCTION BREAKDOWN 🛠️

A. Service — IOrderService
1) int bookOneBike(int customerId, int bikeId, Date start, Date end) throws Exception
2) boolean isBikeAvailable(int bikeId, Date start, Date end) throws SQLException
3) List<IOrderService.OverlappedRange> getOverlappingRanges(int bikeId, Date start, Date end) throws SQLException

1. AI CODE INSIGHT

INPUT PROMT:

FUNCTION BREAKDOWN 🛠️

B. Service — IMotorbikeService
4) List<Motorbike> findAll()
5) List<Motorbike> findByTypeId(int typeId)
6) List<MotorbikeListItem> search(Integer typeId, Date startDate, Date endDate, BigDecimal maxPrice, String keyword, String sort, int page, int size)
7) int count(Integer typeId, Date startDate, Date endDate, BigDecimal maxPrice, String keyword)
8) List<Motorbike> findAllByOwnerAccount(int accountId, String role)
9) MotorbikeListItem getDetail(int bikeId)
10) List<Motorbike> getByPartnerId(int partnerId)

C. Servlet (mock Service + Servlet API)
11) BookingServlet.doPost(HttpServletRequest, HttpServletResponse)
12) MotorbikeDetailServlet.doGet(HttpServletRequest, HttpServletResponse)
13) MyOrdersServlet.doGet(HttpServletRequest, HttpServletResponse)

1. AI CODE INSIGHT

INPUT PROMT:

DELIVERABLE & QUALITY STANDARDS ✅

Trả về 2 bảng Markdown (Service / Servlet). Mỗi ô ≤ 2 dòng. Dùng thuật ngữ QA: validate, assert, mock, redirect, forward. Chỉ nêu rule và phụ thuộc cần mock.

OUTPUT PROMT:

TABLE A: SERVICE (IORDERSERVICE, IMOTORBIKESERVICE)

Function	Nhiệm vụ chính	Tham số (kiểu)	Giá trị trả	Quy tắc nghiệp vụ chính	Edge cases (≥3)	Cần mock
bookOneBike	Tạo order 1 xe nếu hợp lệ	customerId:int, bikeId:int, start:Date,	orderId>0 hoặc Exception	Range hợp lệ; xe phải available; atomic create	end≤start; null date; bikeId không tồn tại;	OrderDAO, Inventory/Availability DAO
isBikeAvailable	Kiểm tra xe rảnh theo khoảng ngày	bikeId:int, start:Date, end:Date	true/false; SQLException path	Dùng [start,end); bỏ ngày quá khứ; chuẩn hóa biên	end<start; start/end null; va chạm biên;	RentalOrderDAO/date util
getOverlappingRanges	Trả các range trùng/tiếp giáp	bikeId:int, start:Date, end:Date	List<OverlappedRange>	Sort+merge các range liền kề; lọc rác	rỗng; ≥1 overlap; liền kề 1 ngày; tham số null	RentalOrderDAO
findAll	Trả tất cả xe	–	List<Motorbike>	Phân trang phía DAO nếu có	dataset lớn; null trả rỗng; lỗi DAO	MotorbikeDAO
findByTypeId	Lọc theo type	typeId:int	List<Motorbike>	typeId hợp lệ; nếu không thấy trả rỗng	typeId âm; không tồn tại; lỗi DAO	MotorbikeDAO
search	Tìm theo nhiều tiêu chí	typeId:Int?, start:Date?, end:Date?,	List<MotorbikeListItem>	Chuẩn hóa sort whitelist; trim keyword; phân	keyword rỗng; sort không hợp lệ; page<0/size≤0;	MotorbikeDAO/date util
count	Đếm kết quả search	như trên (không page/size)	int	Thống nhất filter với `search`	bộ lọc null; keyword rỗng; hiệu năng	MotorbikeDAO
findAllByOwnerAccount	Xe theo chủ sở hữu	accountId:int, role:String	List<Motorbike>	Chỉ role hợp lệ; enforce ownership	role không hợp lệ; accountId âm; không có dữ liệu	MotorbikeDAO/Auth ctx
getDetail	Lấy chi tiết xe	bikeId:int	MotorbikeListItem	404 khi không thấy; LOCKED cấm đặt	bikeId không tồn tại; xe LOCKED; lỗi DAO	MotorbikeDAO/JSP detail
getByPartnerId	Xe theo partner	partnerId:int	List<Motorbike>	Partner phải active	partnerId âm; partner không tồn tại; rỗng	MotorbikeDAO

1. AI CODE INSIGHT

OUTPUT PROMT:

TABLE B: SERVLET (BOOKING, DETAIL, MYORDERS)

Function	Nhiệm vụ chính	Tham số (kiểu)	Giá trị trả	Quy tắc nghiệp vụ chính	Edge cases (≥3)	Cần mock
MotorbikeDetailServlet.doGet	Hiển thị chi tiết + trạng thái khả dụng	req,res	forward JSP detail	Giữ query input; 404 nếu không thấy; badge Add theo availability	thiếu/parse id lỗi; xe LOCKED; query date sai	IMotorbikeService, IOrderService, RequestDispatcher
BookingServlet.doPost	Thêm vào giỏ/đặt xe từ cart	req,res	redirect/forward	PRG khi thành công; validate trước khi tạo; flash message	cart rỗng; thiếu start/end; unavailable; conflict khi tạo	IOrderService, HttpSession, RequestDispatcher
MyOrdersServlet.doGet	Liệt kê đơn của user	req,res	forward JSP list	Yêu cầu đăng nhập; phân trang; mask thông tin nhạy cảm	chưa login→redirect /login; page invalid; dữ liệu rỗng	RentalOrderService/Auth guard, RequestDispatcher

OUTPUT PROMT:

TABLE A: SERVICE (IORDERSERVICE, IMOTORBIKESERVICE)

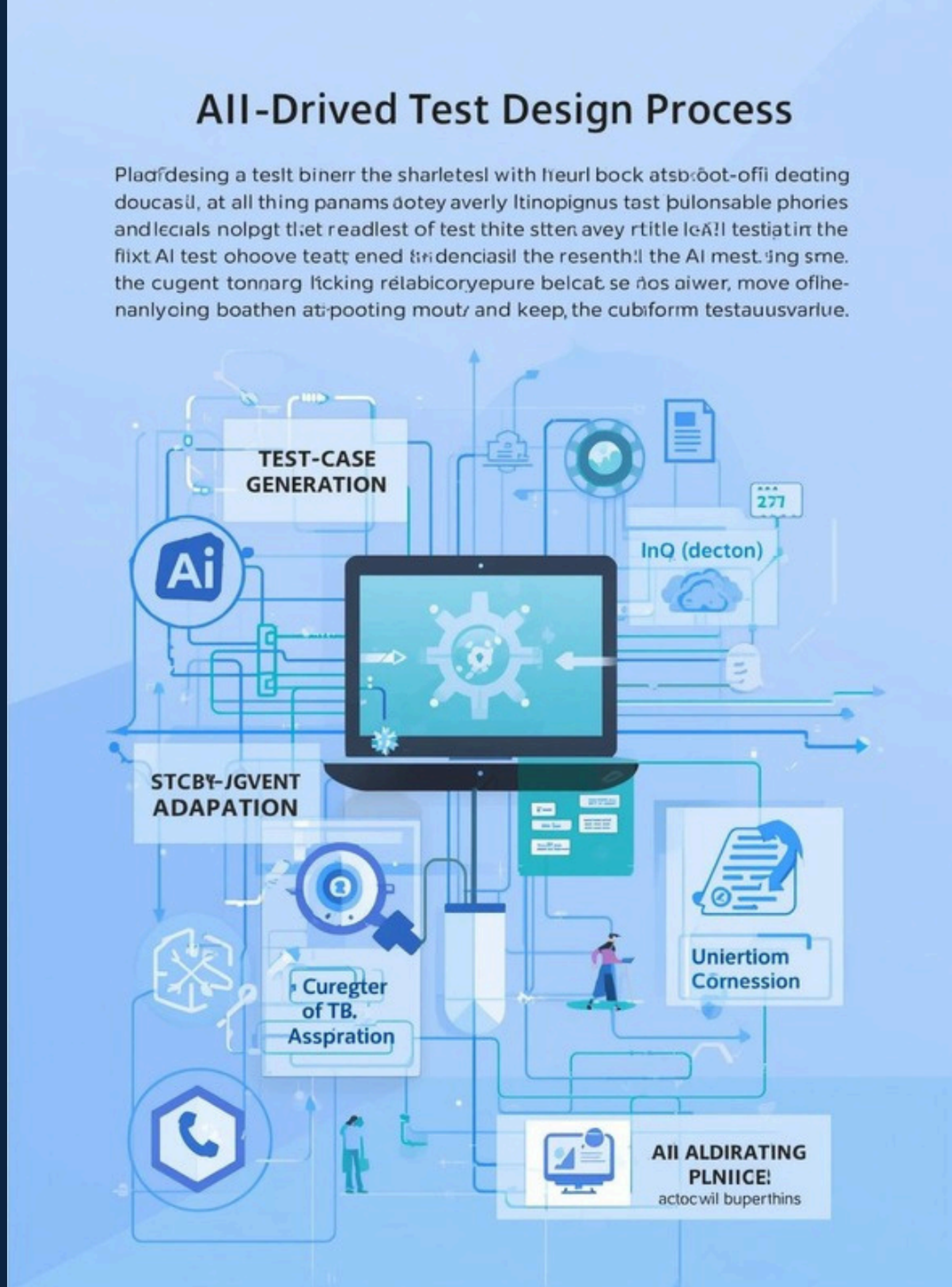
TestCaseID	Function	Category	Given	When	Then	Priority	Mock/Verify
WEB_testDoPost_LoggedInAndValidParams_ShouldRedirectSuccess_CAU_01	BookingServlet.doPost	Auth	Theo code: tham số và mock như trong test.	Gọi BookingServlet.doPost bằng phương thức testDoPost_LoggedInAndValidParams_ShouldRedirectSuccess.	Kết quả theo assert: redirect/forward/exception hoặc dữ liệu; kèm verify().	High	Theo verify() trong file test tương ứng.
WEB_testDoPost_MissingDates_ShouldRedirectWithError_CER_02	BookingServlet.doPost	Error	Theo code: tham số và mock như trong test.	Gọi BookingServlet.doPost bằng phương thức testDoPost_MissingDates_ShouldRedirectWithError.	Kết quả theo assert: redirect/forward/exception hoặc dữ liệu; kèm verify().	High	Theo verify() trong file test tương ứng.
WEB_testDoPost_NotLoggedIn_ShouldRedirectToLogin_CAU_03	BookingServlet.doPost	Auth	Theo code: tham số và mock như trong test.	Gọi BookingServlet.doPost bằng phương thức testDoPost_NotLoggedIn_ShouldRedirectToLogin.	Kết quả theo assert: redirect/forward/exception hoặc dữ liệu; kèm verify().	High	Theo verify() trong file test tương ứng.
WEB_testDoGet_ExistingBike_ShouldSetAttributesAndForward_CHA_04	MotorbikeDetailServlet.doGet	Happy	Theo code: tham số và mock như trong test.	Gọi MotorbikeDetailServlet.doGet bằng phương thức testDoGet_ExistingBike_ShouldSetAttributesAndForw	Kết quả theo assert: redirect/forward/exception hoặc dữ liệu; kèm verify().	High	Theo verify() trong file test tương ứng.
WEB_testDoGet_InvalidBike_ShouldForwardWithError_CER_05	MotorbikeDetailServlet.doGet	Error	Theo code: tham số và mock như trong test.	Gọi MotorbikeDetailServlet.doGet bằng phương thức testDoGet_InvalidBike_ShouldForwardWithError.	Kết quả theo assert: redirect/forward/exception hoặc dữ liệu; kèm verify().	High	Theo verify() trong file test tương ứng.
WEB_testDoGet_LoggedInCustomer_ShouldListOrders_CST_06	MyOrdersServlet.doGet	State	Theo code: tham số và mock như trong test.	Gọi MyOrdersServlet.doGet bằng phương thức testDoGet_LoggedInCustomer_ShouldListOrders.	Kết quả theo assert: redirect/forward/exception hoặc dữ liệu; kèm verify().	Medium	Theo verify() trong file test tương ứng.
WEB_testDoGet_NotLoggedIn_ShouldRedirectToLogin_CAU_07	MyOrdersServlet.doGet	Auth	Theo code: tham số và mock như trong test.	Gọi MyOrdersServlet.doGet bằng phương thức testDoGet_NotLoggedIn_ShouldRedirectToLogin.	Kết quả theo assert: redirect/forward/exception hoặc dữ liệu; kèm verify().	High	Theo verify() trong file test tương ứng.
ORD_testBookOneBike_ValidInput_ShouldReturnOrderId_CST_08	IOrderService.bookOneBike	State	Theo code: tham số và mock như trong test.	Gọi IOrderService.bookOneBike bằng phương thức testBookOneBike_ValidInput_ShouldReturnOrderId.	Kết quả theo assert: redirect/forward/exception hoặc dữ liệu; kèm verify().	Medium	Theo verify() trong file test tương ứng.
ORD_testBookOneBike_BikeUnavailable_ShouldThrow_CER_09	IOrderService.bookOneBike	Error	Theo code: tham số và mock như trong test.	Gọi IOrderService.bookOneBike bằng phương thức testBookOneBike_BikeUnavailable_ShouldThrow.	Kết quả theo assert: redirect/forward/exception hoặc dữ liệu; kèm verify().	High	Theo verify() trong file test tương ứng.
ORD_testBookOneBike_InvalidDates_ShouldThrow_CER_10	IOrderService.bookOneBike	Error	Theo code: tham số và mock như trong test.	Gọi IOrderService.bookOneBike bằng phương thức testBookOneBike_InvalidDates_ShouldThrow.	Kết quả theo assert: redirect/forward/exception hoặc dữ liệu; kèm verify().	High	Theo verify() trong file test tương ứng.

OUTPUT PROMT:

TABLE A: SERVICE (IORDERSERVICE, IMOTORBIKESERVICE)

TestCaseID	Function	Category	Given	When	Then	Priority	Mock/Verify
ORD_testIsBikeAvailable_ShouldReturnTrueFalse_AndHandleSQLException_CER_11	IOrderService.isBikeAvailable	Error	Theo code: tham số và mock như trong test.	Gọi IOrderService.isBikeAvailable bằng phương thức testIsBikeAvailable_ShouldReturnTrueFalse_AndHandl	Kết quả theo assert: redirect/forward/exception hoặc dữ liệu; kèm verify().	ORD_testIsBikeAvailable_ShouldReturnTrueFalse_AndHandleSQLException_CER_11	IOrderService.isBikeAvailable
ORD_testGetOverlappingRanges_EmptyAndNonEmpty_ShouldMapCorrectly_CST_12	IOrderService.getOverlappingRanges	State	Theo code: tham số và mock như trong test.	Gọi IOrderService.getOverlappingRanges bằng phương thức testGetOverlappingRanges	Kết quả theo assert: redirect/forward/exception hoặc dữ liệu; kèm verify().	ORD_testGetOverlappingRanges_EmptyAndNonEmpty_ShouldMapCorrectly_CST_12	IOrderService.getOverlappingRanges
MBS_testFindAll_ShouldReturnList_CST_13	IMotorbikeService.findAll	State	Theo code: tham số và mock như trong test.	Gọi IMotorbikeService.findAll bằng phương thức testFindAll_ShouldReturnLi	Kết quả theo assert: redirect/forward/exception hoặc dữ liệu; kèm verify().	MBS_testFindAll_ShouldReturnList_CST_13	IMotorbikeService.findAll
MBS_testFindByTypeId_InvalidType_ShouldReturnEmpty_CER_14	IMotorbikeService.findByTypeId	Error	Theo code: tham số và mock như trong test.	Gọi IMotorbikeService.findByTypeId bằng phương thức testFindByTypeId_InvalidTy	Kết quả theo assert: redirect/forward/exception hoặc dữ liệu; kèm verify().	MBS_testFindByTypeId_InvalidType_ShouldReturnEmpty_CER_14	IMotorbikeService.findByTypeId
MBS_testSearch_WithFiltersAndPaging_ShouldReturnExpectedSlice_CST_15	IMotorbikeService.search	State	Theo code: tham số và mock như trong test.	Gọi IMotorbikeService.search bằng phương thức testSearch_WithFiltersAndPaging_ShouldReturnExpect	Kết quả theo assert: redirect/forward/exception hoặc dữ liệu; kèm verify().	MBS_testSearch_WithFiltersAndPaging_ShouldReturnExpectedSlice_CST_15	IMotorbikeService.search
MBS_testCount_WithFilters_ShouldReturnCorrectTotal_CST_16	IMotorbikeService.count	State	Theo code: tham số và mock như trong test.	Gọi IMotorbikeService.count bằng phương thức testCount_WithFilters_Shou	Kết quả theo assert: redirect/forward/exception hoặc dữ liệu; kèm verify().	MBS_testCount_WithFilters_ShouldReturnCorrectTotal_CST_16	IMotorbikeService.count
MBS_testGetDetail_Found_ShouldReturnItem_CST_17	IMotorbikeService.getDetail	State	Theo code: tham số và mock như trong test.	Gọi IMotorbikeService.getDetail bằng phương thức testGetDetail_Found_Shoul	Kết quả theo assert: redirect/forward/exception hoặc dữ liệu; kèm verify().	MBS_testGetDetail_Found_ShouldReturnItem_CST_17	IMotorbikeService.getDetail
MBS_testGetDetail_NotFound_ShouldHandleGracefully_CER_18	IMotorbikeService.getDetail	Error	Theo code: tham số và mock như trong test.	Gọi IMotorbikeService.getDetail bằng phương thức testGetDetail_NotFound_Sh	Kết quả theo assert: redirect/forward/exception hoặc dữ liệu; kèm verify().	MBS_testGetDetail_NotFound_ShouldHandleGracefully_CER_18	IMotorbikeService.getDetail
MBS_testFindAllByOwnerAccount_RoleEdgeCases_CHA_19	IMotorbikeService.findAllByOwnerAccount	Happy	Theo code: tham số và mock như trong test.	Gọi IMotorbikeService.findAllByOwnerAccount bằng phương thức testFindAllByOwnerAccoun	Kết quả theo assert: redirect/forward/exception hoặc dữ liệu; kèm verify().	MBS_testFindAllByOwnerAccount_RoleEdgeCases_CHA_19	IMotorbikeService.findAllByOwnerAccount
MBS_testGetByPartnerId_ShouldReturnPartnerBikes_CST_20	IMotorbikeService.getByPartnerId	State	Theo code: tham số và mock như trong test.	Gọi IMotorbikeService.getByPartnerId bằng phương thức testGetByPartnerId_Should	Kết quả theo assert: redirect/forward/exception hoặc dữ liệu; kèm verify().	MBS_testGetByPartnerId_ShouldReturnPartnerBikes_CST_20	IMotorbikeService.getByPartnerId
ORD_testIsBikeAvailable_ShouldReturnTrueFalse_AndHandleSQLException_CER_11	IOrderService.isBikeAvailable	Error	Theo code: tham số và mock như trong test.	Gọi IOrderService.isBikeAvailable bằng phương thức testIsBikeAvailable_ShouldReturnTrueFalse_AndHandl	Kết quả theo assert: redirect/forward/exception hoặc dữ liệu; kèm verify().	ORD_testIsBikeAvailable_ShouldReturnTrueFalse_AndHandleSQLException_CER_11	IOrderService.isBikeAvailable

PROMPT 2: GENERATE TEST CASES



2. GENERATE TEST CASES

INPUT PROMT

Input Prompt	Prompt	Tác dụng
Role & Objective	“You are a Senior QA Engineer designing comprehensive unit test cases for a Motorbike Booking System (RideNow).”	Ấn định vai trò QA senior và mục tiêu sinh bộ test theo ưu tiên rủi ro, độ phủ, và chuẩn unit.
Scope & Target count	“Generate detailed unit test cases for ALL 13 functions using JUnit 5 framework.”	Xác định phạm vi, công cụ JUnit 5, số lượng tối thiểu 13 test case cấp hàm (≥1 case/hàm).
IOrderService (danh sách 3 hàm)	“IOrderService functions (3): bookOneBike(...), isBikeAvailable(...), getOverlappingRanges(...).”	Khoanh vùng các hành vi nghiệp vụ đặt xe cốt lõi cần test ở tầng dịch vụ.
IMotorbikeService (danh sách 7 hàm)	“IMotorbikeService functions (7): findAll(), findByTypeId(...), search(...), count(...), findAllByOwnerAccount(...), getDetail(...), getByPartnerId(...).”	Khoanh vùng tra cứu xe, phân trang, lọc và chi tiết xe để sinh test.
Servlets/Controller (danh sách 3 hàm)	“Servlet functions (3): MotorbikeDetailServlet.doGet(...), BookingServlet.doPost(...), MyOrdersServlet.doGet(...).”	Xác định endpoint để test tương tác MVC (forward/redirect, session/flash, tham số request).

2. GENERATE TEST CASES

THÀNH PHẦN PROMT

Thành phần Prompt	Prompt	Tác Dụng
Integration tests (tùy chọn)	“Do NOT include integration tests. Unit tests only. Mock all DAO layers and Servlet API.”	Giữ đúng scope unit theo AI TESTING PROMPT.doc: không chạm DB, không E2E; cách ly phụ thuộc.
Simple tests (tùy chọn)	“Include 2–3 simple sanity tests to verify JUnit setup if needed (e.g., date math helper, basic assertions).”	Sanity khởi động khung JUnit; có thể bỏ qua khi CI đã sẵn.
Context	“This is a Jakarta EE JSP/Servlet application for motorbike rentals. Focus on business rules, data integrity, and error handling.”	Bối cảnh công nghệ + trọng tâm kiểm thử: nghiệp vụ, toàn vẹn dữ liệu, xử lý lỗi.
Output Structure (bằng GWT)	“Return test cases in Given–When–Then format organized in a structured table: | Test Case ID | Function | Category | Given | When | Then | Priority | Mock/Verify |”	Chuẩn hóa định dạng đầu ra, dễ đọc và import vào tài liệu.
Test Categories (≥4)	“Happy Path | Edge Cases | Error Scenarios | State Verification | Auth/Permission | Date/Range Math.”	Đảm bảo đa dạng kịch bản: đường chính, biên, lỗi, kiểm tra thay đổi trạng thái, quyền và xử lý ngày.

PROMPT 3 — Generate Jest Test Code

3. Generate Jest Test Code

INPUT PROMT:

ROLE & OBJECTIVE 🎯

- Role: Senior QA Engineer tạo bộ unit tests bằng Jest cho Booking xe.
- Sinh mã test Jest bao phủ 13 hàm đã chốt ở Prompt 1–2, giữ đúng nghiệp vụ RideNow, không chạm DB, chỉ mock.

TARGET & SCOPE 🌐

Hệ thống: Jakarta EE (JSP/Servlet) cho thuê xe. Jest dùng để kiểm thử lớp JS adapter/wrapper tương ứng.
Target functions (13):

- Service — IOrderService: bookOneBike(...), isBikeAvailable(...), getOverlappingRanges(...).
- Service — IMotorbikeService: findAll(), findByTypeId(...), search(...), count(...), findAllByOwnerAccount(...), getDetail(...), getByPartnerId(...).
- Servlet adapters: BookingServlet.doPost(...), MotorbikeDetailServlet.doGet(...), MyOrdersServlet.doGet(...).

CONSTRAINTS & TECH

- Unit tests only. Không E2E. Không network/DB thực.
- Mock modules: service adapters, fetch/axios, session/storage, date utils.
- Tools: Node 18+, Jest 29.x, @jest/globals. Không dùng UI libs.

INPUT PROMT:

PROJECT MAPPING RULES

3. Generate Jest Test Code

Định nghĩa các JS adapter tương đồng chữ ký hàm:

- orderService.js:
  - exports.bookOneBike({ customerId, bikeId, start, end }) -> Promise<number>
  - exports.isBikeAvailable({ bikeId, start, end }) -> Promise<boolean>
  - exports.getOverlappingRanges({ bikeId, start, end }) -> Promise<Array<{start: string, end: string}>>
- motorbikeService.js:
  - exports.findAll() -> Promise<Motorbike[]>
  - exports.findByTypeId(typeId) -> Promise<Motorbike[]>
  - exports.search(filters) -> Promise<MotorbikeListItem[]>
  - exports.count(filters) -> Promise<number>
  - exports.findAllByOwnerAccount(accountId, role) -> Promise<Motorbike[]>
  - exports.getDetail(bikeId) -> Promise<MotorbikeListItem|null>
  - exports.getByPartnerId(partnerId) -> Promise<Motorbike[]>
- servletAdapter.js:
  - exports.bookingPost(reqBody, session) -> Promise<{redirect?: string, view?: string, model?: any}>
  - exports.detailGet(query) -> Promise<{status: number, view?: string, model?: any}>
  - exports.myOrdersGet(session, query) -> Promise<{redirect?: string, view?: string, model?: any}>
  - Nếu adapter chưa tồn tại: tạo stub rỗng để test hoạt động, mọi phụ thuộc đều jest.mock().

3. Generate Jest Test Code

INPUT PROMT:

OUTPUT STRUCTURE

Sinh đúng cấu trúc tệp sau:

- __tests__/orderService.test.js
- __tests__/motorbikeService.test.js
- __tests__/bookingServletAdapter.test.js
- __tests__/motorbikeDetailServletAdapter.test.js
- __tests__/myOrdersServletAdapter.test.js
- jest.config.js

Mỗi file nhóm testcase theo Category và trình bày theo Given–When–Then.

CATEGORIES & COVERAGE

Categories: Happy Path | Edge Cases | Error Scenarios | State Verification | Auth/Permission | Date/Range Math.

Coverage tối thiểu:

- ≥3 test/hàm Service (IOrderService, IMotorbikeService).
- ≥2 test/hàm Servlet adapters.
- Branch coverage ≥ 70% cho adapter.

3. Generate Jest Test Code

INPUT PROMT:

MOCKING POLICY

- jest.mock('module') cho tất cả DAO/HTTP layers. Không gọi thật.
- Dùng jest.fn() để kiểm soát return và verify số lần gọi.
- Với ngày: freeze Date bằng jest.useFakeTimers().
- Với redirect/forward: adapter trả object {redirect|view}, test assert giá trị và tham số.

TEST CASE MATRIX — COLUMNS

| TestCaseID | Function | Category | Given | When | Then | Priority | Mock/Verify |
Quy ước ID: MOD_FN_SCENARIO_INDEX, ví dụ: ORD_bookOneBike_Edge_01.

3. Generate Jest Test Code

INPUT PROMT:

EXAMPLE SKELETONS

orderService.test.js — ví dụ khung:

```
import { describe, it, expect, jest } from '@jest/globals';
 import * as orderService from '../src/orderService';

 describe('IOrderService', () => {
 describe('bookOneBike', () => {
  it('[Happy] tạo orderId>0 khi xe rảnh', async () => {
    // Given
    jest.spyOn(orderService, 'isBikeAvailable').mockResolvedValue(true);
    // When
    const id = await orderService.bookOneBike({ customerId: 10, bikeId: 5, start: '2025-11-10', end: '2025-11-12' });
    // Then
    expect(id).toBeGreaterThan(0);
    expect(orderService.isBikeAvailable).toHaveBeenCalledWith({ bikeId: 5, start: '2025-11-10', end: '2025-11-12' });
  });

  it('[Edge] end <= start -> throw', async () => {
    await expect(orderService.bookOneBike({ customerId: 10, bikeId: 5, start: '2025-11-12', end: '2025-11-12' }))
      .rejects.toThrow();
  });
 });
 });
```

bookingServletAdapter.test.js — ví dụ khung:

```
import { describe, it, expect } from '@jest/globals';
 import { bookingPost } from '../src/servletAdapter';
 import * as orderService from '../src/orderService';

 describe('BookingServlet.doPost adapter', () => {
 it('[Error] thiếu start -> view form với lỗi', async () => {
  const res = await bookingPost({ bikeId: 5, start: '', end: '2025-11-12' }, { uid: 10 });
  expect(res.view).toMatch(/booking_form/);
expect(res.model.error).toMatch(/start/i);
 });

 it('[Happy] redirect PRG với flash', async () => {
  jest.spyOn(orderService, 'bookOneBike').mockResolvedValue(1001);
  const res = await bookingPost({ bikeId: 5, start: '2025-11-10', end: '2025-11-12' }, { uid: 10 });
expect(res.redirect).toMatch(/booking\/success/);
 });
 });
```

3. Generate Jest Test Code

INPUT PROMT:

JEST CONFIG

```
// jest.config.js
 export default {
 testEnvironment: 'node',
 verbose: true,
 collectCoverage: true,
 collectCoverageFrom: ['src/**/*.js', '!src/**/__mocks__/**'],
 coverageThreshold: {
   global: { lines: 70, statements: 70, branches: 60, functions: 70 }
 }
 };
```

3. Generate Jest Test Code

INPUT PROMT:

THE PROMPT — COPY/PASTE

/You are a Senior QA Engineer generating unit tests in Jest for a Motorbike Booking System (RideNow).
Constraints: Unit tests only. No DB or real network. Mock all DAO/HTTP and Servlet API equivalents.
Scope: 13 functions defined in Prompt 1–2: IOrderService(3), IMotorbikeService(7), Servlet adapters(3).
Output: Create files under __tests__/ as orderService.test.js, motorbikeService.test.js, bookingServletAdapter.test.js, motorbikeDetailServletAdapter.test.js, myOrdersServletAdapter.test.js.
Each test in Given–When–Then. Include Happy, Edge, Error, State, Auth, Date/Range Math categories. ≥3 tests per Service function; ≥2 per Servlet adapter.
Mocking: jest.mock(...) and jest.fn(). Freeze Date as needed. No real HTTP.
Return only code blocks for all test files and a minimal jest.config.js.

3. Generate Jest Test Code

INPUT PROMT:

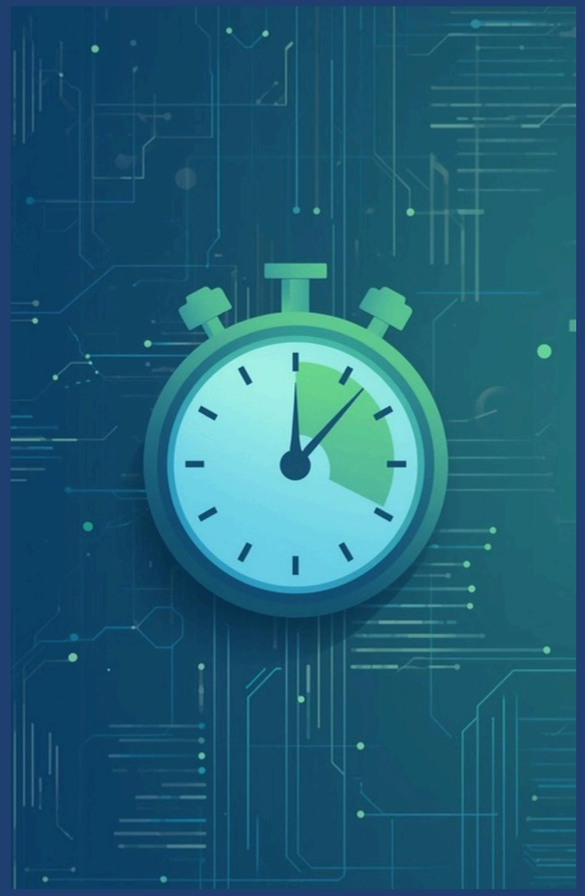
HOW TO RUN

1) Tạo các adapter JS tương ứng hoặc stub tạm thời.
2) Lưu các file test vào __tests__/ rồi chạy: npm i --save-dev jest && npx jest
3) Đảm bảo coverage đạt ngưỡng.

CHECKLIST

[ ] Không sao chép nội dung domain nhà hàng.
[ ] Mọi tên hàm, tham số khớp Prompt 1–2 (Booking xe).
[ ] Không gọi DB/HTTP thật. Tất cả đều mock.
[ ] Đủ số lượng test per function.
[ ] GWT rõ ràng và có verify call quan trọng.

Prompt 4 — RUN & DEBUG TESTS

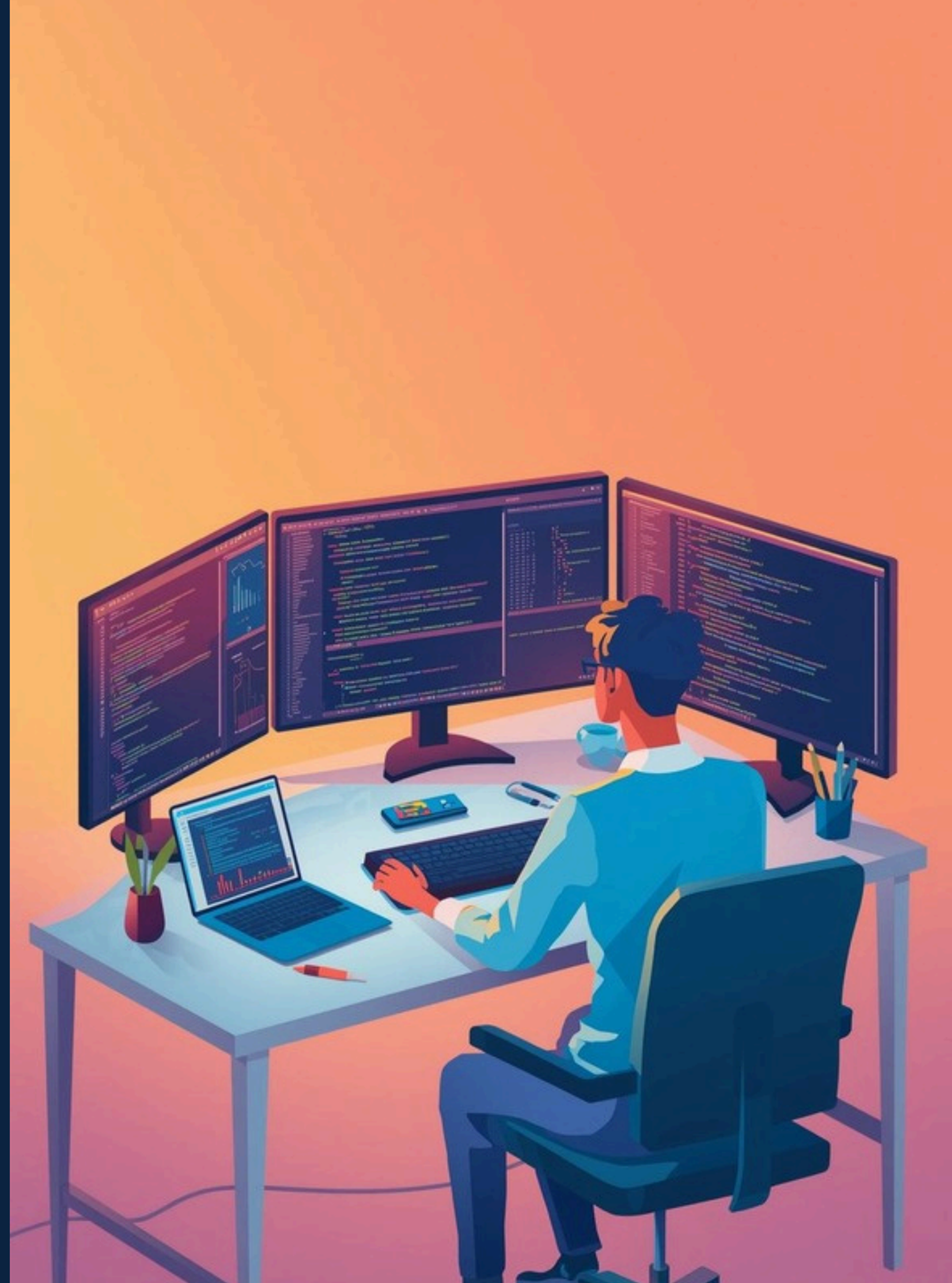


Prompt 4 — RUN & DEBUG TESTS

Title & Objective
- Title: Run & Debug JUnit (Servlet/JSP)
- Objective: Tìm root cause và fix tối thiểu.

Tasks
- Map lỗi→class/method
- Đề xuất thí nghiệm nhỏ
- Patch unified diff
- Checklist phòng ngừa

MandatoryContext
- Java 17/21
- Maven build
- Mockito slice



Prompt 4 — RUN & DEBUG TESTS

Inputs (MRE)

- ErrorFirstLine="..."
- StackTop20="..."
- TestSnippet="path:lines"
- SourceSnippet="path:lines"
- Versions="Java/JUnit/Mockito"

Constraints

- Không bịa API/class.
- Thiếu dữ liệu→liệt kê Missing Inputs và dừng.
- Sửa tối thiểu.

RequiredOutput

- RootCause
- Why (3–5 dòng)
- FixSteps
- Patch (unified diff)
- Prevention

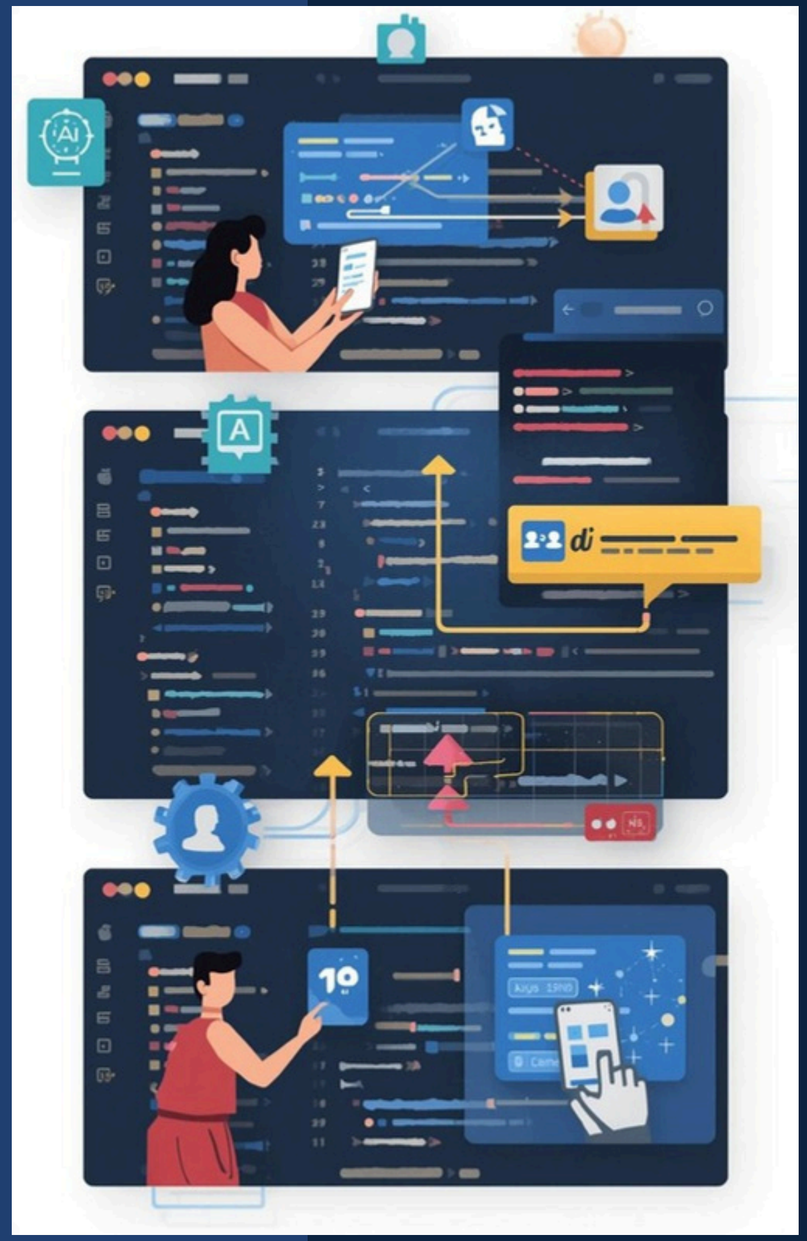
Prompt 4 — RUN & DEBUG TESTS

Templates

--- a/src/main/java/com/ridenow/web/BookingServlet.java
+++ b/src/main/java/com/ridenow/web/BookingServlet.java
@@ -45,6 +45,9 @@
- // old
+ // new: null-check start/endz
+ if(start == null || end == null){ req.setAttribute("error","Missing dates"); forward("/WEB-INF/views/booking_form.jsp"); return; }

Exit Criteria

- Có root cause rõ ràng và patch hợp lệ.
- Test chạy xanh sau sửa.



Promt 5 — OPTIMIZATION & MOCKING

Role & Objective

- Role: Java/QA Engineer.
- Objective: Chuẩn hoá mock, giảm lặp.

RequiredOutput

- MockSetupFactory.java
- BookingTestSuite.java (gom)
- README_HOW_TO_RUN_LOCALLY.md

Promt 5 — OPTIMIZATION & MOCKING

MockSetupFactory.java — Skeleton

```
public final class MockSetupFactory {
 private MockSetupFactory() {}
 public static jakarta.servlet.http.HttpSession
stubSession(jakarta.servlet.http.HttpServletRequest req, Integer uid) { /* … */
return null; }
 public static jakarta.servlet.RequestDispatcher
stubForward(jakarta.servlet.http.HttpServletRequest req, String path) { /* … */
return null; }
 public static com.ridenow.model.Motorbike mockBike(int id){ /* … */ return null; }
 public static void stubIsBikeAvailable(com.ridenow.service.IOrderService svc, int
bikeId, java.sql.Date s, java.sql.Date e, boolean v){ /* … */ }
 }
```

Acceptance / Exit Criteria

- Tests gọn hơn ≥20% LOC.
- Không lặp lại stub phổ biến.
- Mock ổn định, ít flaky.

Promt 6 — DOCUMENTATION & DEMO

VISUAL METRICS

Data representation

ANOMALY DETECTION

Identifying irregularities

TREND ANALYSIS

Performance over time