Unit Tests using Junit

In CarServiceTest.java

```
@SpringBootTest
public class CarServiceTest {

    @Autowired private CarService carService;

    @MockBean private CarRepository carRepository;

    @Test
    public void testRegisterCar() {
        Car car = new Car();
        car.setModel("Hyundai");
        car.setStatus("IDEAL");

        when(carRepository.save(any(Car.class))).thenReturn(car);
        Car result = carService.registerCar(car);

        assertEquals("Hyundai", result.getModel());
    }
}
```

Functional API Tests using MockMvc

In CarControllerTest.java

```
@WebMvcTest(CarController.class)
public class CarControllerTest {

    @Autowired private MockMvc mockMvc;

    @MockBean private CarService carService;

    @Test
    public void shouldRegisterCar() throws Exception {
        Car car = new Car();
        car.setModel("Tesla");

        when(carService.registerCar(any())).thenReturn(car);

        mockMvc.perform(post("/api/car/register")
                .contentType(MediaType.APPLICATION_JSON)
                .content("{\"model\":\"Tesla\"}"))
            .andExpect(status().isOk())
            .andExpect(jsonPath("$.model").value("Tesla"));
    }
}
```

Test Reports with Gradle

```
plugins {
```

```
    id 'java'
    id 'jacoco'
}

test {
    useJUnitPlatform()
    finalizedBy jacocoTestReport
}

jacocoTestReport {
    dependsOn test
    reports {
        xml.required = true
        html.required = true
        csv.required = false
    }
}
```