Test report

1. Overview of System Interface

The tested system is the back-end system of Ezparking, which mainly realizes business through SSM framework. Https is used. The total number of tested interfaces is 10, which are described in Readme.md.

2. Purpose and scope of testing

2.1. Test purpose

The purpose of this test is to test whether the interface returns results as expected and to verify Spring security's permission authentication capabilities.

2.2. Range of test objects

It is mainly the single interface test of the interface related to the background system.

2.3. Range of test indicators

u The interface under test receives request and return packets

u The interface under test returns to state

u The interface under test corresponds to business logic processing

3. Testing tools and resources

3.1. Testing tools

Describe the testing tools and auxiliary tools used in this test

Testing Tool: This test will use Postman

Postman is an interface test plugin from Google. It is simple to use, supports use case management, supports get, post, file upload, response verification, variable management, environment parameter management and other functions, can run in batch, and supports use case export and import.

4. Test record and result analysis

4.1. Single Scenario Interface Testing

4.1.1. Test result data

Use case	Scene description	Interface under test	Test result	note
API001	The interface used to submit the form containing waypoint	/insertWP	pass	If the submitted form is empty, 400 is returned to avoid information loss caused by the

				submission of an empty form
API002	An interface used to get Waypoint information	`/loadWP"`	pass	A return of 200 is obtained successfully, but there is no return condition for failure.
API003	An interface used to record the entry and exit of vehicles in a parking lot	`/countUp`	pass	Success is recorded, and the corresponding value is actually modified in the cache
API004	Interface for recording vehicle departures	`/countDown`	pass	
API005	Check the number of remaining vehicles in the parking lot	`/queryCount`	pass	
API006	Check the remaining vehicles in all parking lots	`/getTotalNum`	pass	

API007	Used to modify parking lot information, such as the name of the parking lot and the number of parking Spaces	`/updatePLwp`	pass	
API008	Change the name of destination	`/updateDNwp`	pass	
API009	Get all destination include Lat, longitude,id,etc	`/getAllDestination`	pass	
API010	login	`/Login`	pass	

4.1.2. Test problem and result analysis

There are no measures for getting information and handling exceptions, not many options for returning results, and not a lot of foreseeing problems the code might encounter at runtime. All running logic is premised on smooth operation. There are no solutions that take into account problems caused by server performance or other external factors. The number of operations associated with each insert is too large, and only a few hundred nodes are functional at this stage. There

are potential problems if the size of the scene used increases. Each update requires changing the relevant information. Because the shortest path method is related to the insert method, you need to resubmit the form when some nodes are updated or deleted to get the new solution. This is a design problem of the whole system reflected by the interface. There are not too many security issues, and spring security is currently used to keep users' information secure. The permission of the data interface to be modified has been authenticated.

5. Test conclusion

For now, the system is working normally. However, there are many potential risks, such as concurrency and so on. Because it has not been put into use, the relevant problems cannot be reflected successfully, so the relevant optimization scheme cannot be obtained.