

# SOFTWARE APPLICATION TEST DOCUMENT

## reliserv

---

**Prepared By:**

QA Team / Nyo Me Han

**Organization:**

ServiceBridge Team

**Repository:**

<https://github.com/ServiceBridgeTeam/reliserv>

**Date:**

[02-28-2026]

**Version:**

1.0

---

CONFIDENTIAL — FOR INTERNAL USE ONLY

---

## 1. Document Revision History

---

Version	Date	Description	Author
1.0	02-28-2026	Initial draft	Nyo Me Han

## 2. Introduction

---

### 2.1 Purpose

This document describes the test plan, test cases, and test results for the reliserv application maintained by ServiceBridge Team. It is intended to verify that all functional and non-functional requirements are correctly implemented and that the system behaves reliably under expected and edge-case conditions.

### 2.2 Scope

This test document covers the following areas of the reliserv application:

- Core service reliability and uptime functionality
- API endpoint behavior and response validation
- Service registration, discovery, and health-check mechanisms
- Authentication and authorization workflows
- Error handling and fault tolerance
- Performance under normal and peak load conditions
- Integration with dependent external services

### 2.3 Intended Audience

This document is intended for QA engineers, developers, project managers, and stakeholders involved in the reliserv project.

### 2.4 References

Document / Resource	Location / URL
GitHub Repository	<a href="https://github.com/ServiceBridgeTeam/reliserv">https://github.com/ServiceBridgeTeam/reliserv</a>
API Documentation	[Link to API docs]
Requirements Specification	[Link to requirements doc]
Architecture Document	[Link to architecture doc]
Project Plan	

### 3. Test Objectives

---

The primary objectives of this test effort are:

1. Verify that all specified functional requirements are correctly implemented.
2. Validate that the application handles error states and edge cases gracefully.
3. Confirm that performance benchmarks are met under load.
4. Ensure security controls are enforced and cannot be bypassed.
5. Confirm compatibility with supported environments and configurations.
6. Identify and document defects for remediation before release.

## 4. Test Environment

---

### 4.1 Environment Summary

Parameter	Value
Environment Type	Staging / Pre-Production
Operating System	[e.g., Ubuntu 22.04 LTS / Windows Server 2022]
Runtime / Language Version	[e.g., Node.js 20.x / Go 1.22 / Java 21]
Database	[e.g., PostgreSQL 15 / Redis 7]
Containerization	[e.g., Docker 24 / Kubernetes 1.29]
CI/CD Pipeline	[e.g., GitHub Actions]
Test Framework	[e.g., Jest / Pytest / JUnit]
Base URL	<a href="https://[staging-host]/api">https://[staging-host]/api</a>

### 4.2 Test Tools

Tool	Purpose	Version
Postman / REST Client	API endpoint testing	1.0
GitHub Actions	CI pipeline and automated test runs	N/A

### 4.3 Test Data

All test data used in this document is synthetic and does not contain real production data. Test data sets should be prepared in the test fixtures folder of the repository before execution.

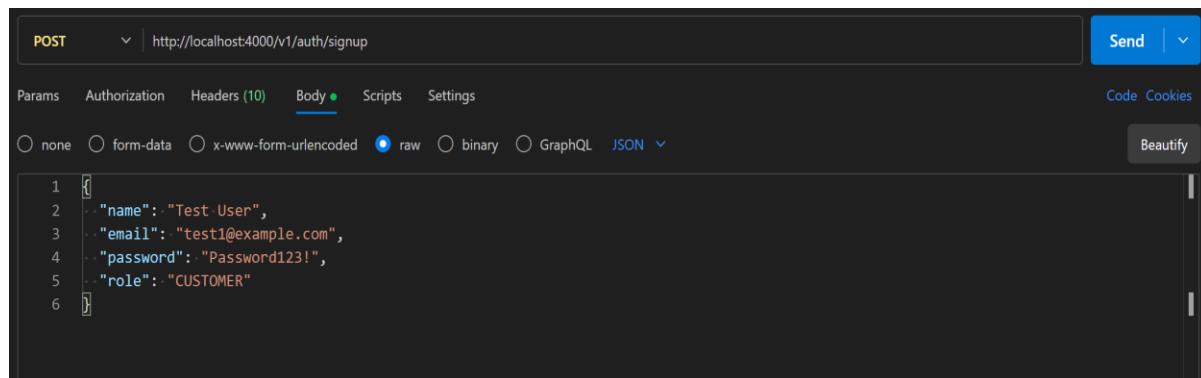
## 5. Test Cases

---

### 5.1 API Validation

#### 5.1.1 Authentication - Signup

## Request



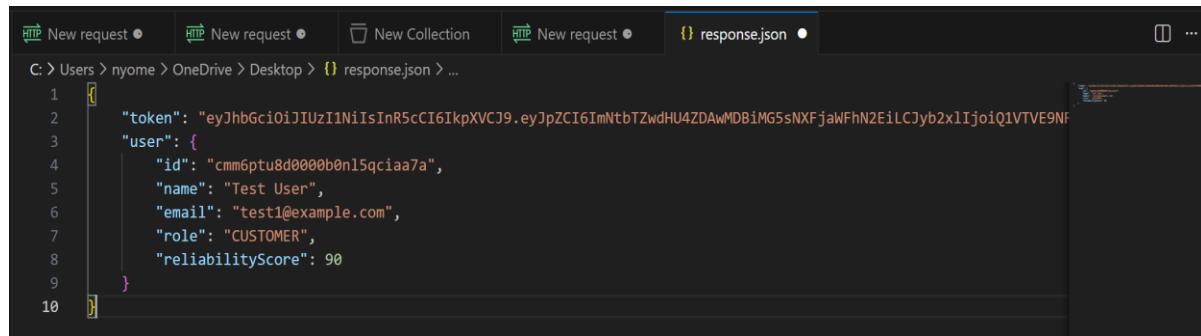
The screenshot shows the reliserv API testing interface. A POST request is being made to `http://localhost:4000/v1/auth/signup`. The **Body** tab is selected, showing a raw JSON payload:

```

1  [
2   . "name": "Test-User",
3   . "email": "test1@example.com",
4   . "password": "Password123!",
5   . "role": "CUSTOMER"
6 ]

```

## Response



The screenshot shows the response to the previous POST request. The response is a JSON object:

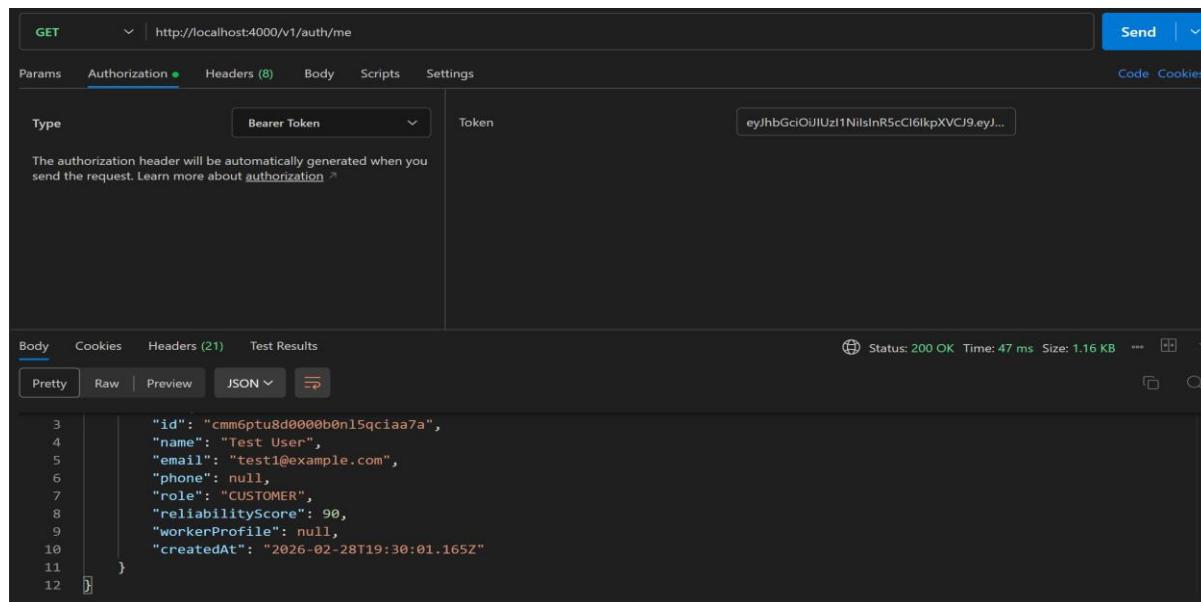
```

1  [
2   . "token": "eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9.eyJpZCI6ImNtbTZwdHU4ZDAwMDB1MG5sNXFjaWFhN2EiLCJyb2x1IjoiQ1VTVE9NF...",
3   . "user": {
4     . "id": "cmm6ptu8d0000b0nl5qciaa7a",
5     . "name": "Test User",
6     . "email": "test1@example.com",
7     . "role": "CUSTOMER",
8     . "reliabilityScore": 90
9   }
10 ]

```

## 5.2 Configuration & Environment

### 5.2.1 JWT Middleware (Request - Response)



The screenshot shows a GET request to `http://localhost:4000/v1/auth/me`. The **Authorization** tab is selected, showing a **Type** of **Bearer Token** and a **Token** field containing `eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9.eyJ...`.

The response body is a JSON object:

```

3  . "id": "cmm6ptu8d0000b0nl5qciaa7a",
4  . "name": "Test User",
5  . "email": "test1@example.com",
6  . "phone": null,
7  . "role": "CUSTOMER",
8  . "reliabilityScore": 90,
9  . "workerProfile": null,
10 . "createdAt": "2026-02-28T19:30:01.165Z"
11 }
12 ]

```

## 5.2.2 DB Proof — Prisma Studio (visual verification)

The screenshot shows the Prisma Studio interface. On the left, there's a sidebar with 'Schema: public' selected. Below it are sections for 'Studio', 'Visualizer', 'Console', and 'Tables'. Under 'Tables', 'User' is selected, showing a list of users with columns: id, createdAt, email, name, and passwordHash. The data includes entries for Alex Rivera, John Martinez, and a test user.

	id	createdAt	email	name	passwordHash
	cmm6pl5020000qsnl4jx193sx	2026-03-01T01:23:15.217Z	alex@example.com	Alex Rivera	\$2b\$10\$ovbClDRlomIX5A4uHkIIE.STC40GERNTdrLSCv0Mx7y.y.RiknCPy
	cmm6pl50y0001qsnl59mx5ef9	2026-03-01T01:23:15.248Z	john.worker@example.com	John Martinez	\$2b\$10\$ovbClDRlomIX5A4uHkIIE.STC40GERNTdrLSCv0Mx7y.y.RiknCPy
	cmm6ptu8d0000b0nl5qciaa7a	2026-03-01T01:30:01.165Z	test1@example.com	Test User	\$2b\$10\$ovbClDRlomIX5A4uHkIIE.STC40GERNTdrLSCv0Mx7y.y.RiknCPy

## 5.3 Core Service Functionality

### 5.3.1 Create Job

#### Request

The screenshot shows a Postman request for creating a job. The method is 'POST', the URL is 'http://localhost:4000/v1/jobs', and the body is set to 'raw' JSON. The JSON payload is:

```

1 {
2   "title": "Fix leaking faucet",
3   "description": "Kitchen faucet dripping for 3 days, worse when turned off.",
4   "jobType": "plumbing",
5   "urgency": "NORMAL",
6   "priceMin": 90,
7   "priceMax": 150,
8   "locationText": "0.8 miles away"
9 }

```

## Response

```

1  {
2      "job": {
3          "id": "cmm6vv5c80001b0nl1p1y40hz",
4          "createdAt": "2026-02-28T22:18:59.897Z",
5          "updatedAt": "2026-02-28T22:18:59.897Z",
6          "title": "Fix leaking faucet",
7          "description": "Kitchen faucet dripping for 3 days, worse when turned off.",
8          "jobType": "plumbing",
9          "urgency": "NORMAL",
10         "status": "OPEN",
11         "lat": null,
12         "lng": null,
13         "locationText": "0.8 miles away",
14         "priceMin": 90,
15         "priceMax": 150,
16         "priceFinal": null,
17         "lockedScope": null,
18         "createdById": "cmm6ptu8d0000b0nl5qcacia7a",
19         "assignedWorkerId": null,
20         "events": [
21             {
22                 "id": "cmm6vv5cb0002b0nl150i6dlsw",
23                 "createdAt": "2026-02-28T22:18:59.897Z",
24                 "jobId": "cmm6vv5c80001b0nl1p1y40hz",
25                 "actorId": "cmm6ptu8d0000b0nl5qcacia7a",
26                 "type": "CREATED",
27                 "note": "Job created"
28             }
29         ]
30     }
31 }
```



### 5.3.2 List Jobs (Request – Response)

GET | http://localhost:4000/v1/jobs?mine=true | Send

Params • Authorization • Headers (8) Body Scripts Settings Code Cookies

Type Bearer Token Token eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9.eyJ...

The authorization header will be automatically generated when you send the request. Learn more about [authorization](#)

Body Cookies Headers (21) Test Results Status: 200 OK Time: 39 ms Size: 1.27 KB

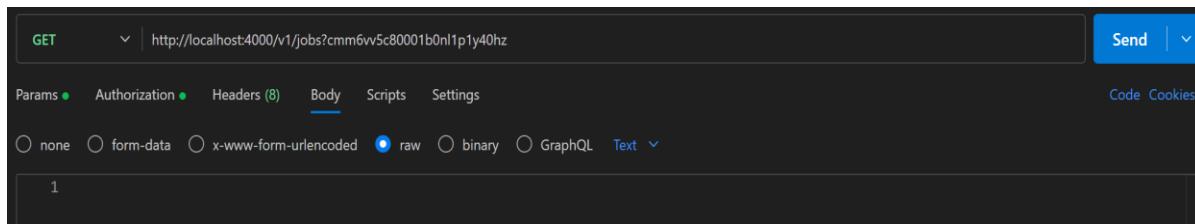
Pretty Raw Preview JSON

```

1  {
2      "jobs": [
3          {
4              "id": "cmm6vv5c80001b0nl1p1y40hz",
5              "title": "Fix leaking faucet",
6              "jobType": "plumbing",
7              "urgency": "NORMAL",
8              "status": "OPEN",
9              "priceMin": 90,
10             "priceMax": 150,
11             "priceFinal": null,
12             "locationText": "0.8 miles away",
13             "createdAt": "2026-02-28T22:18:59.897Z",
14             "createdById": "cmm6ptu8d0000b0nl5qcacia7a",
15             "assignedWorkerId": null
16         }
17     ]
18 }
```

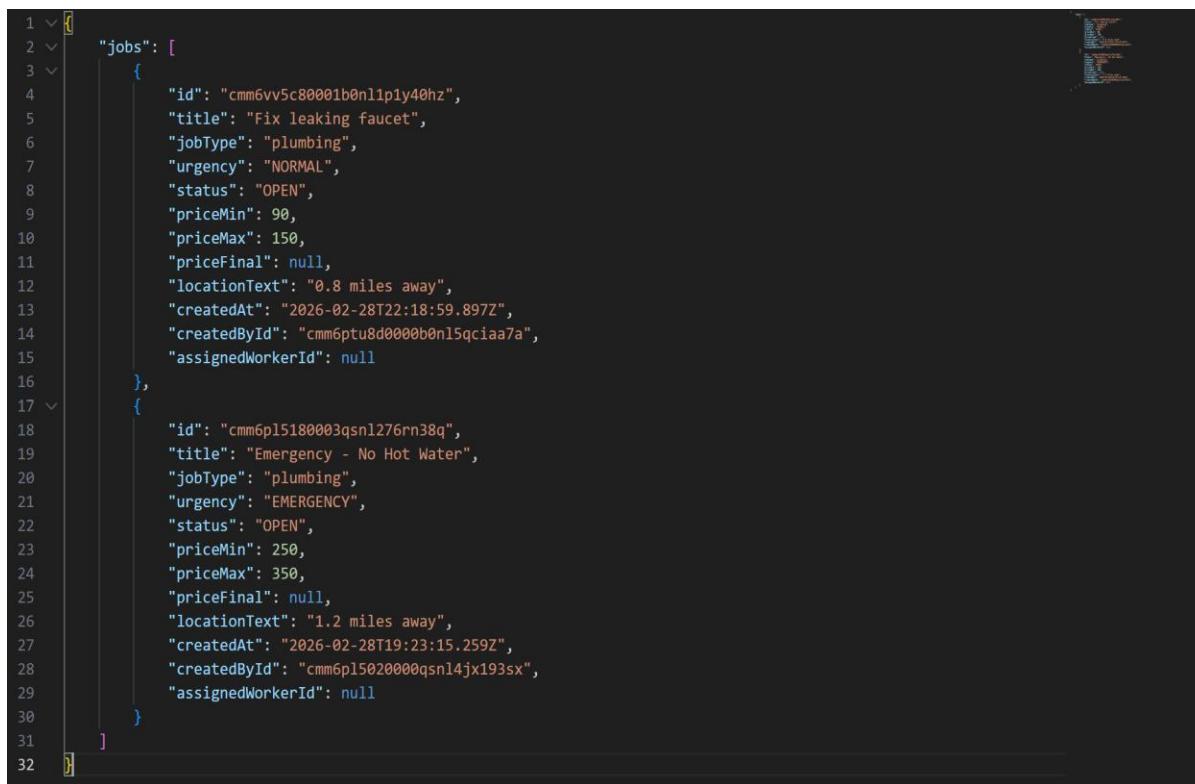
### 5.3.3 Get Job by ID

#### Request



A screenshot of the Postman application interface. The top bar shows 'GET' and the URL 'http://localhost:4000/v1/jobs?cmn6vv5c80001b0nl1p1y40hz'. Below the URL are tabs for 'Params', 'Authorization', 'Headers (8)', **Body**, 'Scripts', and 'Settings'. Under 'Body', there are options for 'none', 'form-data', 'x-www-form-urlencoded', 'raw' (which is selected), 'binary', 'GraphQL', and 'Text'. A dropdown menu next to 'Text' shows '1'. The main body area contains the number '1'.

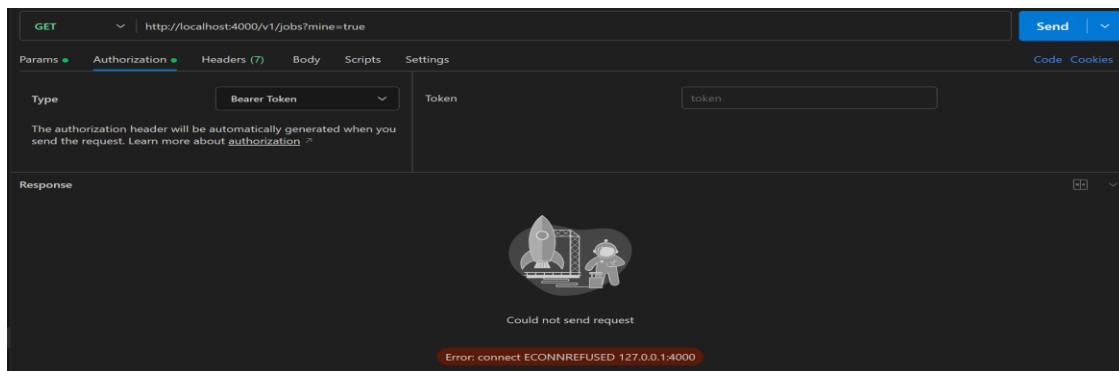
#### Response



```
1 < [ { "id": "cmn6vv5c80001b0nl1p1y40hz", "title": "Fix leaking faucet", "jobType": "plumbing", "urgency": "NORMAL", "status": "OPEN", "priceMin": 90, "priceMax": 150, "priceFinal": null, "locationText": "0.8 miles away", "createdAt": "2026-02-28T22:18:59.897Z", "createdById": "cmn6ptu8d000b0nl5qciao7a", "assignedWorkerId": null }, { "id": "cmn6pl5180003qsnl276rn38o", "title": "Emergency - No Hot Water", "jobType": "plumbing", "urgency": "EMERGENCY", "status": "OPEN", "priceMin": 250, "priceMax": 350, "priceFinal": null, "locationText": "1.2 miles away", "createdAt": "2026-02-28T19:23:15.259Z", "createdById": "cmn6pl502000qsnl4jx193sx", "assignedWorkerId": null } ]
```

## 5.5 Security Proof

### 5.5.1 Validation Without Token



## 6. Defect Reporting

### 6.1 Defect Log

*Instructions: Record all defects discovered during testing below. Link to GitHub Issues where applicable.*

Defect ID	Title	TC ID(s)	Severity	Steps to Reproduce	Expected	Actual	Status	GitHub Issue
DEF-001							N.A	
DEF-002							N.A	
DEF-003							N.A	
DEF-004							N.A	

### 6.2 Severity Definitions

Severity	Definition
Critical	System crash, data loss, security breach, or complete feature failure blocking core workflows.
High	Major feature broken with no workaround; significantly impacts usability.
Medium	Feature partially broken; workaround exists but is burdensome.
Low	Minor cosmetic or non-blocking issue; minimal user impact.

## 7. Test Execution Summary

### 7.1 Execution Details

Parameter	Value
Test Cases	5.1 – 5.5
Test Start Date	02-28.2026

Parameter	Value
Test End Date	02-28.2026
Executed By	Nyo Me Han
Environment	Development / Staging

## 7.2 Test Results Summary

Test Category	Total	Passed	Failed	Blocked	Not Run	Pass %
Authentication & Authorization	1	Yes				100
Core Service Functionality	3	Yes				100
API Validation & Error Handling	0					
Security Validation	1	Yes				100
Performance Tests	0					
Configuration & Environment	2	Yes				100
TOTAL	7					

## 7.3 Outstanding Risks

Risk	Likelihood	Impact	Mitigation
NA	NA		
NA	NA		

## 8. Sign-Off

The following stakeholders have reviewed and approved this test document:

Role	Name	Signature	Date
QA Lead	Nyo Me Han	NYO ME HAN	02-28-2026
Development Lead			
Product Owner / Manager			
Release Manager			

## 9. Abbreviations

Abbreviation	Definition
API	Application Programming Interface
CI/CD	Continuous Integration / Continuous Delivery
JWT	JSON Web Token
OOM	Out Of Memory
QA	Quality Assurance
RPS	Requests Per Second
SLA	Service Level Agreement
TC	Test Case
DEF	Defect
P95/P99	95th / 99th Percentile Latency

## 10. Notes & Additional Observations

*Use this section to capture any additional observations, out-of-scope items, or information not covered elsewhere in this document.*

[Add notes here]