Automotas AI API Test Report

Test Date: July 28, 2025
Test Duration: Comprehensive API endpoint testing
Environment: automotas-ai test environment

Executive Summary

This report provides a comprehensive analysis of the Automotas AI API endpoints, including test results, discovered issues, and recommendations for system improvement.

Overall Test Results

- **Total Endpoints Tested:** 39
- **Successful Tests:** 6 (15.4%)
- **Failed Tests:** 33 (84.6%)
- **Services Discovered:** 2 active services

Service Discovery

Active Services Found

1. **MCP (Model Context Protocol) Service**
   - **Port:** 8000
   - **Status:** ✅ OPERATIONAL
   - **Description:** REST API for managing Model Context Protocol servers
   - **Version:** 1.0.0

2. **Computer Tools API Service**
   - **Port:** 1000
   - **Status:** ⚠️ LIMITED FUNCTIONALITY
   - **Description:** API for computer automation tools
   - **Issues:** Limited endpoint availability

3. **Main Orchestrator API**
   - **Expected Port:** 8002
   - **Status:** ❌ NOT RUNNING
   - **Issues:** Missing dependencies (langchain_text_splitters)

Detailed Test Results

✅ MCP Service Endpoints (Port 8000)

Success Rate: 75% (3/4 tests passed)

Endpoint	Method	Status	Response Time	Description
`/listServers`	GET	✅ 200	0.008s	List MCP servers
`/listTools`	POST	✅ 200	0.002s	List available tools
`/stopAllServers`	POST	✅ 204	0.001s	Stop all servers
`/startServer`	POST	❌ 500	0.007s	Start MCP server

Working Endpoints:
- Server listing and management
- Tool discovery
- Bulk server operations

Issues Found:
- Server startup functionality returns HTTP 500 errors
- Individual server management may be problematic

❌ Main Orchestrator API Endpoints (Port 8002)

Status: Service Not Available

Expected Endpoints (Not Accessible):
- Agent Management (`/api/agents/`)
- Skill Management (`/api/skills/`)
- Workflow Management (`/api/workflows/`)
- Document Management (`/api/documents/`)
- System Management (`/api/system/`)
- Health Check (`/health`)

Root Cause: Missing Python dependencies preventing service startup
- `langchain_text_splitters` module not installed
- Service fails to initialize due to import errors

⚠️ Documentation Endpoints (Port 8000)

Success Rate: 50% (2/4 tests passed)

Endpoint	Method	Status	Response Time	Description
`/docs`	GET	✅ 200	0.001s	Swagger UI documentation
`/openapi.json`	GET	✅ 200	0.001s	OpenAPI schema
`/health`	GET	❌ 404	0.023s	Health check
`/`	GET	❌ 404	0.001s	Root endpoint

Performance Analysis

Response Time Metrics

- **Average Response Time:** 0.003s
- **Fastest Response:** 0.001s
- **Slowest Response:** 0.008s
- **Responses Under 1s:** 100% (6/6 successful requests)

Performance Insights

- ✅ Excellent response times for working endpoints
- ✅ No timeout issues observed
- ✅ Consistent performance across multiple requests
- ⚠️ Limited sample size due to service availability issues

Test Plan Review

Existing Test Plans Found

1. **Professional Agent Testing Suite** (`test_professional_agents.py`)
   - **Purpose:** Comprehensive testing of agent types and capabilities
   - **Coverage:** Agent creation, skill execution, collaboration scenarios
   - **Status:** Available but requires running orchestrator service

2. **Complex Development Task Testing** (`test_complex_request.py`)
   - **Purpose:** End-to-end testing of complex development workflows
   - **Coverage:** API development, full-stack applications, microservices
   - **Status:** Available but requires orchestrator service

Test Coverage Analysis

Planned Test Categories:
- ✅ Health and connectivity tests
- ✅ MCP service functionality
- ❌ Agent management (service unavailable)
- ❌ Skill management (service unavailable)
- ❌ Workflow operations (service unavailable)
- ❌ Document processing (service unavailable)
- ❌ System management (service unavailable)
- ✅ Performance testing (limited scope)
- ❌ Integration testing (service dependencies unavailable)

API Endpoint Inventory

MCP Service API (Port 8000) - OPERATIONAL

```
GET    /listServers          - List managed MCP servers
POST   /startServer          - Start a new MCP server
POST   /startServers         - Start multiple MCP servers
POST   /stopServer           - Stop a specific server
POST   /stopAllServers       - Stop all servers
POST   /listTools            - List tools on servers
POST   /callTool             - Execute a tool on a server
GET    /docs                 - API documentation
GET    /openapi.json         - OpenAPI schema
```

Main Orchestrator API (Port 8002) - NOT OPERATIONAL

```
# Agent Management
GET    /api/agents/          - List agents
POST   /api/agents/          - Create agent
GET    /api/agents/{id}      - Get agent details
PUT    /api/agents/{id}      - Update agent
DELETE /api/agents/{id}      - Delete agent

# Skill Management
GET    /api/skills/          - List skills
POST   /api/skills/          - Create skill
GET    /api/skills/{id}      - Get skill details
PUT    /api/skills/{id}      - Update skill
DELETE /api/skills/{id}      - Delete skill

# Workflow Management
GET    /api/workflows/       - List workflows
POST   /api/workflows/       - Create workflow
GET    /api/workflows/{id}   - Get workflow details
POST   /api/workflows/{id}/execute - Execute workflow
GET    /api/workflows/executions/ - List executions

# Document Management
GET    /api/documents/       - List documents
POST   /api/documents/upload - Upload document
GET    /api/documents/{id}   - Get document details
POST   /api/documents/search - Search documents
POST   /api/context/search   - Context search
GET    /api/documents/stats  - Document statistics

# System Management
GET    /api/system/status    - System status
GET    /api/system/metrics   - System metrics
GET    /api/system/config    - System configuration
POST   /api/system/config    - Update configuration
GET    /api/system/logs      - System logs

# Core Endpoints
GET    /health               - Health check
GET    /                     - Root endpoint
WebSocket /ws                - Real-time updates
```

Issues and Recommendations

Critical Issues

1. **Missing Dependencies**
   - **Issue:** Main orchestrator service fails to start due to missing `langchain_text_splitters`
   - **Impact:** 85% of planned API functionality unavailable
   - **Recommendation:** Install missing Python packages
   `bash`
   ```
     pip install langchain-text-splitters langchain-community
   ```

2. **Service Configuration**
   - **Issue:** Services running on non-standard ports
   - **Impact:** Confusion about service availability
   - **Recommendation:** Standardize port configuration and document service architecture

3. **Database Dependencies**
   - **Issue:** Document management requires PostgreSQL with pgvector
   - **Impact:** Document processing and context engineering features unavailable
   - **Recommendation:** Ensure database services are properly configured

Performance Issues

1. **MCP Server Startup**
   - **Issue:** `/startServer` endpoint returns HTTP 500 errors
   - **Impact:** Limited MCP server management capabilities
   - **Recommendation:** Debug server startup process and error handling

Documentation Issues

1. **API Documentation Gaps**
   - **Issue:** Health and root endpoints not documented in OpenAPI schema
   - **Impact:** Incomplete API documentation
   - **Recommendation:** Update OpenAPI schema to include all endpoints

2. **Service Discovery**
   - **Issue:** No clear documentation of which services run on which ports
   - **Impact:** Difficulty in testing and integration
   - **Recommendation:** Create service architecture documentation

Test Recommendations

Immediate Actions

1. **Resolve Dependencies**
   `bash`
   ```
     cd /home/ubuntu/automotas-ai-v2.5/automotas-ai/automotas-ai/orchestrator
     pip install langchain-text-splitters langchain-community sqlalchemy psycopg2-binary
   ```

2. **Start Main API Service**
   `bash`
   ```
     python main.py
   ```

3. **Verify Database Services**
   `bash`

```
systemctl status postgresql
systemctl status redis
```

Comprehensive Testing Plan

1. **Phase 1: Service Startup**
   - Resolve all dependency issues
   - Start all required services
   - Verify basic connectivity

2. **Phase 2: Functional Testing**
   - Test all CRUD operations for each entity type
   - Verify authentication and authorization
   - Test file upload and processing

3. **Phase 3: Integration Testing**
   - Test agent-skill relationships
   - Test workflow execution end-to-end
   - Test document processing pipeline

4. **Phase 4: Performance Testing**
   - Load testing with concurrent users
   - Response time optimization
   - Resource utilization monitoring

Monitoring and Alerting

1. **Health Checks**
   - Implement comprehensive health checks for all services
   - Monitor service dependencies (database, Redis, etc.)
   - Set up automated alerts for service failures

2. **Performance Monitoring**
   - Track API response times
   - Monitor resource utilization
   - Set up performance baselines

Conclusion

The Automotas AI system shows a well-designed API architecture with comprehensive endpoint coverage for agent management, workflow orchestration, and document processing. However, the current test environment has significant deployment issues that prevent full functionality testing.

Key Findings:
- ✅ MCP service is operational with good performance
- ❌ Main orchestrator API is not running due to dependency issues
- ✅ API design appears comprehensive and well-structured
- ⚠️ Service configuration needs standardization

Next Steps:
1. Resolve dependency issues to enable full API testing
2. Implement comprehensive test suite once services are operational
3. Establish monitoring and alerting for production readiness
4. Document service architecture and deployment procedures

Test Coverage Achieved: 15.4% (limited by service availability)
Recommended Retest: After resolving dependency and service startup issues

---

Report generated by Automotas AI API Test Suite
For questions or issues, please refer to the technical documentation or contact the development team.