Table of Contents

Overview

SAMFMS (Smart Autonomous Fleet Management System) is a microservices-based fleet management solution by Team Firewall Five. The system consists of:

- **Core Service**: API gateway and request router
- **Service Blocks**: 7 microservices (Security, GPS, Management, Trip Planning, Vehicle Maintenance, Utilities, Micro Frontend)
- **Data Blocks**: 3 data services (GPS, Users, Vehicles)
- **Frontend**: React web interface
- **Infrastructure**: RabbitMQ, MongoDB, Redis

Installation Options:

- **Production**: Docker Compose (recommended)
- **Development**: Local services + containerized infrastructure
- **Testing**: Quick Docker setup

System Requirements:

- **CPU**: 4 cores (Intel i5/AMD Ryzen 5)
- **RAM**: 8GB minimum, 16GB recommended
- **Storage**: 20GB free space
- **OS**: Windows 10+, macOS 10.15+, Ubuntu 20.04+
- **Network**: Broadband connection, ports 21000-21020

Prerequisites

Required Software (with Version Numbers)

Software	Windows	macOS	Linux	Version Required
Docker Desktop	winget install Docker.DockerDesktop	brew install --cask docker	Install Guide	20.10.0+
Git	winget install Git.Git	brew install git	sudo apt install git	2.30.0+
Node.js	winget install OpenJS.NodeJS	brew install node@18	NodeSource	18.17.0+
Python	winget install Python.Python.3.9	brew install python@3.9	sudo apt install python3.9	3.9.0+

Linux-Specific Installation

Docker Engine (Ubuntu/Debian):

```
# Install Docker
 curl -fsSL https://get.docker.com -o get-docker.sh
 sudo sh get-docker.sh
 sudo usermod -aG docker $USER

 # Install Docker Compose
 sudo curl -L
"https://github.com/docker/compose/releases/download/v2.20.2/docker-c
ompose-$(uname -s)-$(uname -m)" -o /usr/local/bin/docker-compose
 sudo chmod +x /usr/local/bin/docker-compose
```

Node.js (Ubuntu/Debian) {#linux-node}:

```
curl -fsSL https://deb.nodesource.com/setup_18.x | sudo -E bash -
 sudo apt-get install -y nodejs
```

Optional Tools

- **VS Code**: winget install Microsoft.VisualStudioCode
- **MongoDB Compass**: Database management GUI
- **Postman/Hopscotch**: API testing tool

Version Compatibility Matrix

Component	Minimum	Recommended	Tested
Docker Engine	20.10.0	24.0.0	24.0.6
Docker Compose	2.0.0	2.20.0	2.20.2
Python	3.9.0	3.9.18	3.9.18
Node.js	18.0.0	18.17.0	18.17.1

[Screenshot needed: Version verification commands showing successful installation]

Installation

Method 1: Production (Docker) - Recommended

```
# 1. Clone repository
git clone https://github.com/COS301-SE-2025/SAMFMS.git
cd SAMFMS

# 2. Configure environment (optional)
copy .env.example .env
notepad .env  # Edit if needed

# 3. Start all services
docker-compose up -d

# 4. Verify installation
docker-compose ps  # All services should show "Up (healthy)"
curl http://localhost:21004/health  # Core API health check
```

[Screenshot needed: Docker services starting and health check response]

Access Points:

- **Frontend**: http://localhost:3000
- **Core API**: http://localhost:21004/docs
- **RabbitMQ Management**: http://localhost:21001 (samfms_rabbit / RabbitPass2025!)

Method 2: Development Setup

```
# 1. Clone and setup
git clone https://github.com/COS301-SE-2025/SAMFMS.git
cd SAMFMS


# 2. Start infrastructure only
docker-compose up -d rabbitmq mongodb redis

# 3. Setup Python environment
python -m venv venv
.\venv\Scripts\activate  # Windows
pip install -r Core/requirements.txt

# 4. Install service dependencies (repeat for each service)
cd Sblocks/security && pip install -r requirements.txt && cd ../..
cd Sblocks/gps && pip install -r requirements.txt && cd ../..
# ... repeat for management, vehicle_maintenance, trip_planning, utilities

# 5. Run services (separate terminals)
# Terminal 1: cd Core && python main.py
# Terminal 2: cd Sblocks/security && python main.py
# Terminal 3: cd Frontend/samfms && npm install && npm start
```

Method 3: Manual Installation

For custom configurations without Docker:

1. **Install Infrastructure**:
   - MongoDB Community Server 7.0+
   - RabbitMQ Server 3.12+ with management plugin
   - Redis Server 7.0+
2. **Configure Services**:

```
# Set environment variables
set MONGODB_URL=mongodb://username:password@localhost:27017
set RABBITMQ_URL=amqp://username:password@localhost:5672/
set REDIS_HOST=localhost
```

3. **Install Application Dependencies**: Follow Development Setup steps 3-5

Configuration

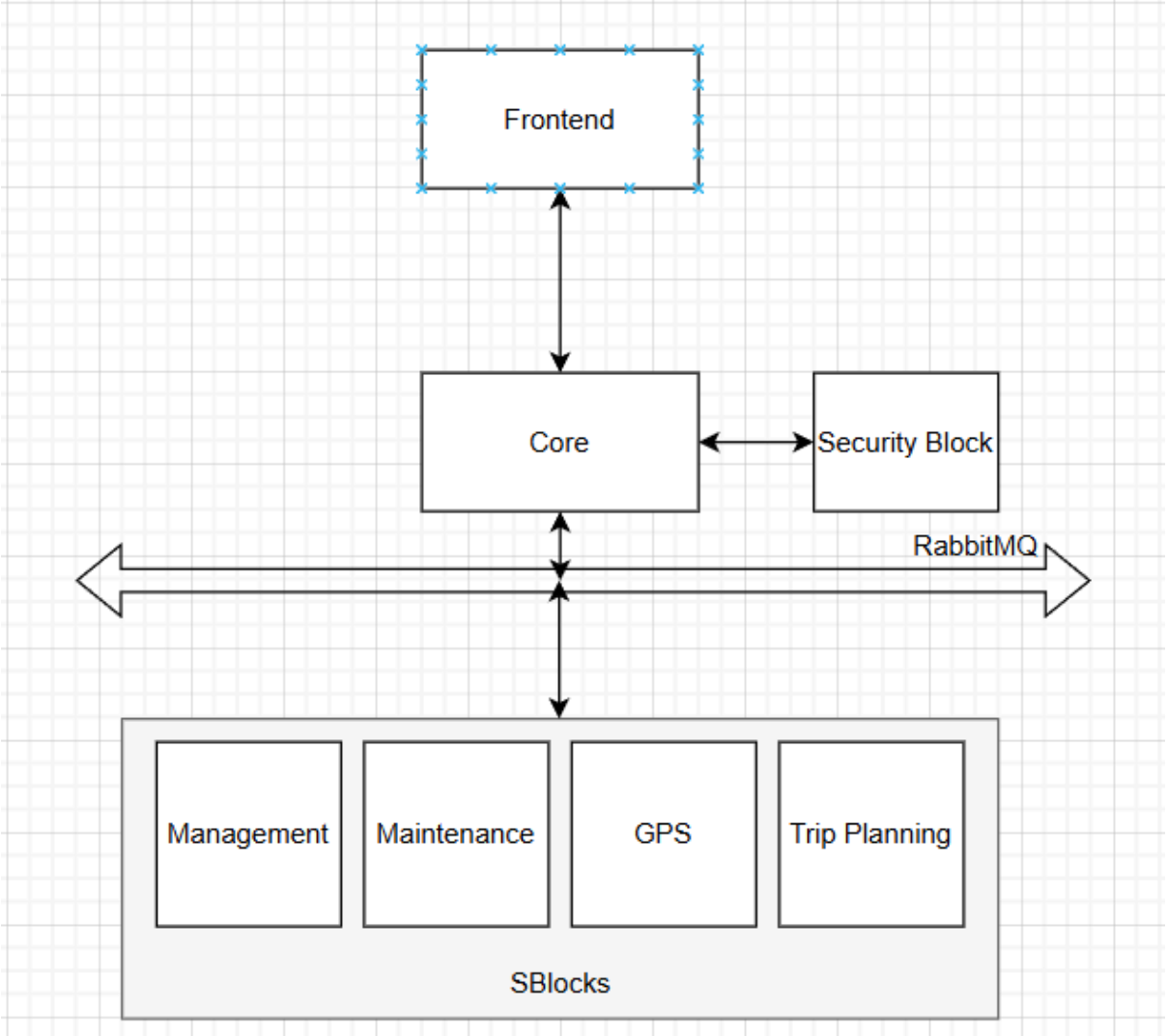
Environment Variables

Configure the system using .env file:

```
# Core Service
 CORE_PORT=21004
 JWT_SECRET_KEY=your-secure-secret-key-here
 ACCESS_TOKEN_EXPIRE_MINUTES=15

 # Database

MONGODB_URL=mongodb://samfms_admin:SafeMongoPass2025%21SecureDB%40SAMFMS@mongodb:27017
 DATABASE_NAME=samfms_core

 # Message Queue

RABBITMQ_URL=amqp://samfms_rabbit:RabbitPass2025%21@rabbitmq:5672/
 RABBITMQ_CONNECTION_RETRY_ATTEMPTS=30


# Infrastructure Ports
 RABBITMQ_PORT=21000
 RABBITMQ_MANAGEMENT_PORT=21001
 REDIS_EXTERNAL_PORT=21002
 MONGODB_PORT=21003

 # Service Ports
 GPS_SERVICE_PORT=21005
 TRIP_PLANNING_SERVICE_PORT=21006
 VEHICLE_MAINTENANCE_SERVICE_PORT=21007
 SECURITY_SERVICE_PORT=21008
 MANAGEMENT_SERVICE_PORT=21009
 UTILITIES_SERVICE_PORT=21010

 # System Configuration
 LOG_LEVEL=INFO
 ENVIRONMENT=production
```

Service Architecture

Component	Port	Purpose	Dependencies
Core	21004	API Gateway & Router	FastAPI, Motor, aio-pika
Security	21008	Authentication & Authorization	PyJWT, Passlib, Motor
GPS	21005	Vehicle Tracking	Motor, Redis, aio-pika
Management	21009	Fleet Management	Motor, aio-pika
Trip Planning	21006	Route Optimization	Motor, aio-pika
Vehicle Maintenance	21007	Service Scheduling	Motor, aio-pika
Utilities	21010	Email & Notifications	SMTP, aio-pika



Deployment and Running

Production Deployment

```
# Start complete system
docker-compose up -d

# Verify all services are healthy
docker-compose ps

# Access the application
# - Frontend: http://localhost:3000
# - Core API: http://localhost:21004/docs
# - Health Check: http://localhost:21004/health
```

System Startup Sequence

1. **Infrastructure** (30-60s): RabbitMQ → MongoDB → Redis
2. **Core Service** (10-15s): API Gateway waits for infrastructure
3. **Service Blocks** (5-10s each): All microservices start
4. **Frontend** (5-10s): React application connects to Core API

First-Time Setup

```
# 1. Create admin account
curl -X POST http://localhost:21008/auth/register/admin \
  -H "Content-Type: application/json" \
  -d
'{"username":"admin","email":"admin@samfms.com","password":"SecurePassword123!"}'

# 2. Test core functionality
# Login at: http://localhost:3000
# API docs: http://localhost:21004/docs
```

System Administration

Health Monitoring:

```
# Check service health
curl http://localhost:21004/health  # Core
curl http://localhost:21005/health  # GPS
curl http://localhost:21008/health  # Security

# Monitor resources
docker stats

# View logs
docker-compose logs -f [service_name]
```

Stopping the System:

```
docker-compose down                    # Graceful shutdown
 docker-compose down --volumes              # Complete cleanup
```

Troubleshooting

Common Issues & Solutions

Issue	Solution
Docker Build Failures	docker system prune -a && docker-compose build --no-cache
Port Conflicts	netstat -an | findstr "21000" then docker-compose down
Service Won't Start	docker-compose logs [service_name] then docker-compose restart [service]
Database Connection	docker logs mongodb and verify credentials in .env
RabbitMQ Issues	Access http://localhost:21001 (samfms_rabbit / RabbitPass2025!)
Performance Issues	docker stats to monitor resource usage

Platform-Specific Notes

Windows:

- Enable WSL2 for Docker Desktop
- Set PowerShell execution policy: Set-ExecutionPolicy RemoteSigned
- Add Docker to Windows Defender exclusions

macOS:

- Install Xcode Command Line Tools: xcode-select --install
- Allocate 4GB+ RAM to Docker Desktop in preferences

Linux:

- Add user to docker group: sudo usermod -aG docker $USER
- Configure SELinux if enforcing: sudo setsebool -P container_manage_cgroup on

Installation Verification Checklist

Prerequisites ✓

- Docker Desktop installed and running (v20.10+)
- Git installed (v2.30+): git --version
- Python 3.9+: python --version
- Node.js 18+: node --version
- Ports 21000-21020 available

Installation ✓

- Repository cloned successfully
- Environment configured: .env file exists
- All services healthy: docker-compose ps
- Core API responds: http://localhost:21004/health
- Frontend loads: http://localhost:3000

Functionality ✓

- Admin account created
- User login works
- API documentation accessible: http://localhost:21004/docs
- Service-to-service communication working

Support

Documentation Links

- **User Manual**: User_Manual.md - End-user operation guide
- **API Documentation**: Available at /docs endpoint of each service
- **Developer Guide**: Developer_Docker_Guide.md
- **Configuration Guide**: Configuration_Guide.md

Support Channels

- **GitHub Issues**: https://github.com/COS301-SE-2025/SAMFMS/issues
- **Project Repository**: https://github.com/COS301-SE-2025/SAMFMS
- **Team Contact**: Firewall Five via project repository

Version Information

- **System Version**: 1.0.0
- **Python**: 3.9.18
- **Docker**: 24.0.6
- **Node.js**: 18.17.1

Last Updated: July 2025
Authors: Team Firewall Five
License: See main project repository