ASMC API - Installation & Setup Guide

This comprehensive guide covers the complete installation and setup process for the ASMC API, from development to production environments.

Table of Contents

System Requirements

Minimum Requirements

Component	Minimum	Recommended
OS	Ubuntu 18.04+ / macOS 10.15+ / Windows 10+	Ubuntu 20.04+ / macOS 11+ / Windows 11+
Node.js	16.0.0+	18.0.0+
MongoDB	4.4.0+	6.0.0+
RAM	2GB	4GB+
Storage	10GB free	20GB+ free
CPU	2 cores	4+ cores

Supported Platforms

- **Ubuntu 20.04+** (Recommended for production)
- **macOS 10.15+** (Development)
- **Windows 10+** (Development)
- **Docker** (Any platform)
- **AWS EC2** (Production)
- **DigitalOcean Droplet** (Production)

Development Setup

Step 1: Install Prerequisites

Install Node.js

Using Node Version Manager (NVM) - Recommended:

```
# Install NVM
curl -o- https://raw.githubusercontent.com/nvm-sh/nvm/v0.39.0/install.sh | bash

# Reload shell
```

```bash
source ~/.bashrc

# Install and use Node.js 18
nvm install 18
nvm use 18
nvm alias default 18

# Verify installation
node --version  # Should show v18.x.x
npm --version   # Should show 9.x.x
```

Direct Installation:

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

# macOS
brew install node@18

# Windows
# Download from https://nodejs.org/en/download/
```

Install MongoDB

Ubuntu/Debian:

```bash
# Import MongoDB public GPG key
wget -qO - https://www.mongodb.org/static/pgp/server-6.0.asc | sudo apt-key add -

# Add MongoDB repository
echo "deb [ arch=amd64,arm64 ] https://repo.mongodb.org/apt/ubuntu focal/mongodb-
org/6.0 multiverse" | sudo tee /etc/apt/sources.list.d/mongodb-org-6.0.list

# Update package database
sudo apt-get update

# Install MongoDB
sudo apt-get install -y mongodb-org

# Start MongoDB
sudo systemctl start mongod
sudo systemctl enable mongod

# Verify installation
sudo systemctl status mongod
```

macOS:

```bash
# Install using Homebrew
brew tap mongodb/brew
brew install mongodb-community
```

```
# Start MongoDB
brew services start mongodb/brew/mongodb-community

# Verify installation
brew services list | grep mongodb
```

Windows:

1. Download MongoDB Community Server from MongoDB Download Center
2. Run the installer and follow the setup wizard
3. MongoDB will start automatically as a Windows service

Step 2: Clone and Setup Project

```
# Clone the repository
git clone <repository-url>
cd asmc-api

# Install dependencies
npm install

# Verify installation
npm list --depth=0
```

Step 3: Environment Configuration

```
# Copy environment template
cp .env.example .env.development

# Edit environment variables
nano .env.development
```

Minimum Development Configuration:

```
# Server Configuration
PORT=7055
NODE_ENV=development

# Database
MONGO_URI=mongodb://localhost:27017/asmc_dev

# Authentication
JWT_SECRET=your-super-secret-jwt-key-minimum-32-characters-long
JWT_EXPIRE=7d

# Email (Optional for development)
SMTP_HOST=smtp.gmail.com
SMTP_PORT=587
SMTP_USER=your-email@gmail.com
SMTP_PASS=your-app-password

# CORS (Development)
CORS_ORIGINS=http://localhost:3000,http://localhost:3001
```

Step 4: Start Development Server

```
# Start development server with auto-restart
npm run dev

# Alternative: Start without auto-restart
npm start
```

Expected Output:

```
== Server running on Port == 7055
API Documentation available at http://localhost:7055/api-docs
```

Step 5: Verify Installation

```
# Test health endpoint
curl http://localhost:7055/health

# Expected response: "ok"

# Test API documentation
open http://localhost:7055/api-docs
```

Production Setup

Ubuntu Server Setup

Step 1: Server Preparation

```
# Update system packages
sudo apt update && sudo apt upgrade -y

# Install essential packages
sudo apt install -y curl wget git unzip software-properties-common

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

# Install PM2 globally
sudo npm install -g pm2

# Install MongoDB
wget -qO - https://www.mongodb.org/static/pgp/server-6.0.asc | sudo apt-key add -
echo "deb [ arch=amd64,arm64 ] https://repo.mongodb.org/apt/ubuntu focal/mongodb-
org/6.0 multiverse" | sudo tee /etc/apt/sources.list.d/mongodb-org-6.0.list
sudo apt-get update
sudo apt-get install -y mongodb-org

# Start and enable MongoDB
```

```
sudo systemctl start mongod
sudo systemctl enable mongod
```

Step 2: Application Deployment

```
# Create application directory
sudo mkdir -p /opt/asmc-api
sudo chown $USER:$USER /opt/asmc-api

# Clone repository
cd /opt/asmc-api
git clone <repository-url> .

# Install dependencies
npm ci --production

# Set up environment
cp .env.example .env.production
nano .env.production
```

Production Environment Configuration:

```
# Server Configuration
PORT=7055
NODE_ENV=production

# Database
MONGO_URI=mongodb://localhost:27017/asmc_prod

# Authentication
JWT_SECRET=your-super-secure-jwt-secret-minimum-64-characters-long
JWT_EXPIRE=7d

# Email Configuration
SMTP_HOST=smtp.gmail.com
SMTP_PORT=587
SMTP_USER=your-email@gmail.com
SMTP_PASS=your-app-password
MSG91_AUTH_KEY=your-msg91-auth-key
MSG91_SENDER_ID=ASMC

# Image Processing
IMAGEKIT_PUBLIC_KEY=your-imagekit-public-key
IMAGEKIT_PRIVATE_KEY=your-imagekit-private-key
IMAGEKIT_URL_ENDPOINT=https://ik.imagekit.io/your-imagekit-id

# Payment Gateway
CCAVENUE_MERCHANT_ID=your-merchant-id
CCAVENUE_ACCESS_CODE=your-access-code
CCAVENUE_WORKING_KEY=your-working-key

# Biometric Integration
```

```
BIOMETRIC_IP=192.168.1.100
BIOMETRIC_PORT=4370

# Security
CORS_ORIGINS=https://asmcdae.in,https://admin.asmcdae.in
```

Step 3: Start with PM2

```
# Start application with PM2
npm run start:prod

# Verify PM2 processes
pm2 list

# Monitor application
pm2 monit

# View logs
pm2 logs asmc-api
```

Step 4: Configure Nginx (Optional)

```
# Install Nginx
sudo apt install -y nginx

# Create Nginx configuration
sudo nano /etc/nginx/sites-available/asmc-api
```

Nginx Configuration:

```
server {
    listen 80;
    server_name api.asmcdae.in;

    location / {
        proxy_pass http://localhost:7055;
        proxy_http_version 1.1;
        proxy_set_header Upgrade $http_upgrade;
        proxy_set_header Connection 'upgrade';
        proxy_set_header Host $host;
        proxy_set_header X-Real-IP $remote_addr;
        proxy_set_header X-Forwarded-For $proxy_add_x_forwarded_for;
        proxy_set_header X-Forwarded-Proto $scheme;
        proxy_cache_bypass $http_upgrade;

        # Timeout settings
        proxy_connect_timeout 60s;
        proxy_send_timeout 60s;
        proxy_read_timeout 60s;
    }
}
```

```
# Enable site
sudo ln -s /etc/nginx/sites-available/asmc-api /etc/nginx/sites-enabled/
sudo nginx -t
sudo systemctl restart nginx
```

Step 5: SSL Certificate (Optional)

```
# Install Certbot
sudo apt install -y certbot python3-certbot-nginx

# Get SSL certificate
sudo certbot --nginx -d api.asmcdae.in

# Auto-renewal (already configured)
sudo certbot renew --dry-run
```

Database Setup

MongoDB Configuration

Basic Setup

```
# Start MongoDB
sudo systemctl start mongod

# Connect to MongoDB
mongo

# Create database and user
use asmc_prod
db.createUser({
  user: "asmc_user",
  pwd: "secure_password",
  roles: [
    { role: "readWrite", db: "asmc_prod" }
  ]
})
```

Security Configuration

```
# Edit MongoDB configuration
sudo nano /etc/mongod.conf
```

Security Settings:

```
# /etc/mongod.conf
security:
    authorization: enabled

net:
    port: 27017
    bindIp: 127.0.0.1
```

```yaml
storage:
    dbPath: /var/lib/mongodb
    journal:
        enabled: true

systemLog:
    destination: file
    logAppend: true
    path: /var/log/mongodb/mongod.log
```

```bash
# Restart MongoDB
sudo systemctl restart mongod

# Test connection with authentication
mongo -u asmc_user -p --authenticationDatabase asmc_prod
```

Database Indexes

```bash
# Connect to database
mongo -u asmc_user -p asmc_prod

# Create indexes for performance
db.members.createIndex({ "memberId": 1 }, { unique: true })
db.members.createIndex({ "personalInfo.email": 1 })
db.members.createIndex({ "personalInfo.phone": 1 })
db.payments.createIndex({ "memberId": 1, "createdAt": -1 })
db.hallBookings.createIndex({ "hallId": 1, "bookingDate": 1 })
```

Database Backup Setup

```bash
# Create backup directory
sudo mkdir -p /opt/asmc-api/backups
sudo chown $USER:$USER /opt/asmc-api/backups

# Test backup script
npm run backup

# Verify backup
ls -la /opt/asmc-api/backups/
```

Environment Configuration

Environment Files Structure

```
asmc-api/
├── .env.example          # Template file
├── .env.development      # Development environment
├── .env.staging         # Staging environment
├── .env.production      # Production environment
└── .env.local          # Local overrides (git ignored)
```

Complete Environment Variables

Server Configuration

```
# Server
PORT=7055
NODE_ENV=development|staging|production

# Logging
LOG_LEVEL=debug|info|warn|error
LOG_FILE=./logs/app.log
```

Database Configuration

```
# MongoDB
MONGO_URI=mongodb://localhost:27017/asmc
MONGO_TEST_URI=mongodb://localhost:27017/asmc_test

# Connection Options
MONGO_MAX_POOL_SIZE=10
MONGO_SERVER_SELECTION_TIMEOUT=5000
MONGO_SOCKET_TIMEOUT=45000
```

Authentication Configuration

```
# JWT
JWT_SECRET=your-super-secret-jwt-key-minimum-32-characters
JWT_EXPIRE=7d
JWT_ISSUER=asmc-api
JWT_AUDIENCE=asmc-clients

# Password Policy
PASSWORD_MIN_LENGTH=8
PASSWORD_REQUIRE_UPPERCASE=true
PASSWORD_REQUIRE_LOWERCASE=true
PASSWORD_REQUIRE_NUMBERS=true
PASSWORD_REQUIRE_SYMBOLS=true
```

Email Configuration

```
# SMTP
SMTP_HOST=smtp.gmail.com
SMTP_PORT=587
SMTP_SECURE=false
SMTP_USER=your-email@gmail.com
SMTP_PASS=your-app-password

# MSG91 (SMS)
MSG91_AUTH_KEY=your-msg91-auth-key
MSG91_SENDER_ID=ASMC
MSG91_ROUTE=4
```

Image Processing Configuration

```
# ImageKit
IMAGEKIT_PUBLIC_KEY=your-imagekit-public-key
IMAGEKIT_PRIVATE_KEY=your-imagekit-private-key
IMAGEKIT_URL_ENDPOINT=https://ik.imagekit.io/your-imagekit-id

# Sharp (Local processing)
IMAGE_MAX_WIDTH=1920
IMAGE_MAX_HEIGHT=1080
IMAGE_QUALITY=80
IMAGE_FORMAT=jpeg
```

Payment Gateway Configuration

```
# CCAvenue
CCAVENUE_MERCHANT_ID=your-merchant-id
CCAVENUE_ACCESS_CODE=your-access-code
CCAVENUE_WORKING_KEY=your-working-key
CCAVENUE_CURRENCY=INR
CCAVENUE_LANGUAGE=en
CCAVENUE_COUNTRY=IND
```

Biometric Integration

```
# ZKTeco Integration
BIOMETRIC_IP=192.168.1.100
BIOMETRIC_PORT=4370
BIOMETRIC_TIMEOUT=5000
BIOMETRIC_RETRY_COUNT=3
```

File Upload Configuration

```
# File Upload Limits
MAX_FILE_SIZE=10485760        # 10MB
MAX_FILES_COUNT=5
ALLOWED_FILE_TYPES=image/jpeg,image/png,image/gif,application/pdf

# Upload Paths
UPLOAD_PATH=./uploads
TEMP_PATH=./uploads/temp
BACKUP_PATH=./backups
```

CORS Configuration

```
# CORS Origins
CORS_ORIGINS=http://localhost:3000,https://asmcdae.in
CORS_METHODS=GET,POST,PUT,DELETE,OPTIONS,PATCH
CORS_ALLOWED_HEADERS=Content-Type,Authorization,X-Requested-With
CORS_CREDENTIALS=true
CORS_MAX_AGE=86400
```

Rate Limiting

```
# Rate Limiting
RATE_LIMIT_WINDOW_MS=900000      # 15 minutes
RATE_LIMIT_MAX_REQUESTS=100
RATE_LIMIT_SKIP_SUCCESSFUL_REQUESTS=false
RATE_LIMIT_SKIP_FAILED_REQUESTS=false
```

Service Configuration

PM2 Configuration

PM2 Ecosystem File

Create `ecosystem.config.js`:

```
module.exports = {
    apps: [
        {
            name: 'asmc-api',
            script: 'app.js',
            instances: 'max',
            exec_mode: 'cluster',
            env: {
                NODE_ENV: 'development',
                PORT: 7055,
            },
            env_staging: {
                NODE_ENV: 'staging',
                PORT: 7055,
            },
            env_production: {
                NODE_ENV: 'production',
                PORT: 7055,
            },
            error_file: './logs/err.log',
            out_file: './logs/out.log',
            log_file: './logs/combined.log',
            time: true,
            max_memory_restart: '1G',
            node_args: '--max-old-space-size=2048',
        },
    ],
};
```

PM2 Commands

```
# Start with ecosystem file
pm2 start ecosystem.config.js --env production

# Monitor processes
pm2 monit
```

```
# View logs
pm2 logs asmc-api

# Restart application
pm2 restart asmc-api

# Stop application
pm2 stop asmc-api

# Delete application
pm2 delete asmc-api

# Save PM2 configuration
pm2 save

# Setup PM2 startup
pm2 startup
```

Systemd Service (Alternative)

Create `/etc/systemd/system/asmc-api.service`:

```
[Unit]
Description=ASMC API Server
After=network.target mongod.service

[Service]
Type=simple
User=www-data
WorkingDirectory=/opt/asmc-api
Environment=NODE_ENV=production
Environment=PORT=7055
ExecStart=/usr/bin/node app.js
Restart=always
RestartSec=10

[Install]
WantedBy=multi-user.target
```

```
# Enable and start service
sudo systemctl enable asmc-api
sudo systemctl start asmc-api

# Check status
sudo systemctl status asmc-api

# View logs
sudo journalctl -u asmc-api -f
```

Verification & Testing

Health Checks

Application Health

```
# Test health endpoint
curl http://localhost:7055/health

# Expected response: "ok"
```

Database Health

```
# Test database connection
mongo --eval "db.adminCommand('ismaster')"

# Expected response: { "ismaster" : true, ... }
```

API Documentation

```
# Open API documentation
open http://localhost:7055/api-docs

# Should display Swagger UI interface
```

Functional Testing

Authentication Test

```
# Test admin login
curl -X POST http://localhost:7055/auth/admin-login \
  -H "Content-Type: application/json" \
  -d '{
    "username": "admin@asmc.com",
    "password": "password123"
  }'

# Expected: JWT token in response
```

API Endpoint Test

```
# Test members endpoint (requires authentication)
curl -X GET http://localhost:7055/members \
  -H "Authorization: Bearer YOUR_JWT_TOKEN"

# Expected: Members list or empty array
```

Performance Testing

Load Testing with Artillery

```
# Install Artillery
npm install -g artillery
```

```
# Create test configuration
cat > load-test.yml << EOF
config:
  target: 'http://localhost:7055'
  phases:
    - duration: 60
      arrivalRate: 10

scenarios:
  - name: "Health check"
    flow:
      - get:
          url: "/health"
EOF

# Run load test
artillery run load-test.yml
```

Troubleshooting

Common Issues

1. Port Already in Use

```
# Error: EADDRINUSE: address already in use :::7055

# Solution 1: Find and kill process
sudo lsof -i :7055
sudo kill -9 <PID>

# Solution 2: Use different port
PORT=7056 npm run dev
```

2. MongoDB Connection Failed

```
# Error: MongoNetworkError: failed to connect to server

# Check MongoDB status
sudo systemctl status mongod

# Start MongoDB
sudo systemctl start mongod

# Check MongoDB logs
sudo tail -f /var/log/mongodb/mongod.log

# Test connection
mongo --eval "db.adminCommand('ismaster')"
```

3. Permission Denied

```
# Error: EACCES: permission denied

# Fix file permissions
sudo chown -R $USER:$USER /opt/asmc-api
chmod -R 755 /opt/asmc-api
```

4. Memory Issues

```
# Error: JavaScript heap out of memory

# Increase Node.js memory limit
node --max-old-space-size=4096 app.js

# Or in PM2
pm2 start app.js --node-args="--max-old-space-size=4096"
```

5. JWT Token Issues

```
# Error: JsonWebTokenError: invalid token

# Check JWT_SECRET
echo $JWT_SECRET

# Ensure secret is minimum 32 characters
# Regenerate token if needed
```

Debug Mode

Enable Debug Logging

```
# Set debug environment
DEBUG=* npm run dev

# Or specific modules
DEBUG=express:router npm run dev
DEBUG=mongoose:* npm run dev
```

PM2 Debug Mode

```
# Start with debug logs
pm2 start app.js --name asmc-api --node-args="--inspect"

# View debug logs
pm2 logs asmc-api
```

Log Analysis

Application Logs

```
# View application logs
tail -f logs/app.log
```

```
# Search for errors
grep -i error logs/app.log

# Search for specific endpoints
grep "POST /members" logs/app.log
```

System Logs

```
# View system logs
sudo journalctl -u asmc-api -f

# View MongoDB logs
sudo tail -f /var/log/mongodb/mongod.log

# View Nginx logs
sudo tail -f /var/log/nginx/access.log
sudo tail -f /var/log/nginx/error.log
```

Performance Issues

Database Performance

```
# Check MongoDB performance
mongo --eval "db.stats()"

# Check slow queries
mongo --eval "db.setProfilingLevel(2, { slowms: 100 })"
mongo --eval "db.system.profile.find().sort({ts: -1}).limit(5)"
```

Application Performance

```
# Monitor CPU and memory
pm2 monit

# Check process details
pm2 show asmc-api

# Profile Node.js application
node --prof app.js
```

Next Steps

After successful installation:

1. **Configure Production Environment**: Set up proper environment variables
2. **Set up Monitoring**: Implement logging and monitoring solutions
3. **Configure Backup**: Set up automated database backups
4. **Security Hardening**: Implement security best practices
5. **Load Testing**: Test application under load
6. **Deploy Frontend**: Connect admin panel and mobile app

Support

If you encounter issues during installation:

1. **Check Logs**: Review application and system logs
2. **Verify Dependencies**: Ensure all prerequisites are installed
3. **Environment Variables**: Verify all required environment variables are set
4. **Network Issues**: Check firewall and network connectivity
5. **Documentation**: Refer to other documentation files for specific topics

---

Congratulations! Your ASMC API is now properly installed and configured!