Issue #9: Environment variable management

Labels: `infrastructure`, `security`, `configuration`
Milestone: Production Ready
Estimated Effort: 3-4 hours

Description

Implement secure and robust environment variable management for different deployment environments (development, staging, production) with validation, documentation, and best practices.

Current State

- ✅ Basic `.env.example` file exists
- ✅ Uses `dotenv` for loading environment variables
- ❌ No environment variable validation
- ❌ No documentation of required vs optional variables
- ❌ No type checking or format validation
- ❌ No secrets rotation strategy
- ❌ No environment-specific configurations

Goals

- ✅ Validate all required environment variables on startup
- ✅ Type-check and format-validate variables
- ✅ Document all environment variables comprehensively
- ✅ Implement secure secrets management
- ✅ Support multiple environments (dev/staging/prod)
- ✅ Fail fast with clear error messages

Tasks

Phase 1: Validation & Type Checking (1-2 hours)

- [ ] Create environment variable schema
- [ ] Implement validation on startup
- [ ] Add type checking (string, number, boolean, URL)
- [ ] Add format validation (tokens, URLs, emails)
- [ ] Fail fast with helpful error messages

Phase 2: Documentation (1 hour)

- [ ] Create comprehensive `.env.example`
- [ ] Document each variable with description
- [ ] Specify required vs optional variables
- [ ] Add format examples and validation rules

- [ ] Document how to obtain sensitive values

Phase 3: Multi-Environment Support (1 hour)

- [ ] Support `.env.development`, `.env.production`
- [ ] Add environment-specific defaults
- [ ] Document environment setup for each stage
- [ ] Add Railway.com environment variable guide

Phase 4: Security Best Practices (1 hour)

- [ ] Document secrets rotation procedure
- [ ] Add .env to .gitignore (verify)
- [ ] Create secrets management guide
- [ ] Document Railway.com secrets setup
- [ ] Add security checklist

Implementation

Environment Variable Validator

```
// config/env.js
require('dotenv').config();

class EnvValidator {
  constructor() {
    this.errors = [];
    this.warnings = [];
  }

  /**
   * Validate a required environment variable
   */
  required(name, options = {}) {
    const value = process.env[name];

    if (!value || value.trim() === '') {
      this.errors.push(`Missing required environment variable: ${name}`);
      return null;
    }

    return this.validate(name, value, { ...options, required: true });
  }

  /**
   * Validate an optional environment variable
   */
  optional(name, defaultValue, options = {}) {
    const value = process.env[name];

    if (!value || value.trim() === '') {
      return defaultValue;
    }

    return this.validate(name, value, { ...options, required: false });
  }

  /**
   * Validate a single environment variable
   */
  validate(name, value, options = {}) {
    const { type, pattern, choices, min, max, required } = options;

    // Type validation
    if (type === 'number') {
      const num = Number(value);
      if (isNaN(num)) {
        this.addError(name, `must be a number, got: ${value}`, required);
        return required ? null : undefined;
      }

      if (min !== undefined && num < min) {
        this.addError(name, `must be >= ${min}, got: ${num}`, required);
      }

      if (max !== undefined && num > max) {
        this.addError(name, `must be <= ${max}, got: ${num}`, required);
      }

      return num;
    }

    if (type === 'boolean') {
```

```
      const lower = value.toLowerCase();
      if (!['true', 'false', '1', '0'].includes(lower)) {
        this.addError(name, `must be a boolean (true/false), got: ${value}`, re-
quired);
        return required ? null : undefined;
      }
      return lower === 'true' || lower === '1';
    }

    if (type === 'url') {
      try {
        new URL(value);
      } catch (error) {
        this.addError(name, `must be a valid URL, got: ${value}`, required);
        return required ? null : value;
      }
    }

    if (type === 'email') {
      const emailRegex = /^[^\s@]+@[^\s@]+\.[^\s@]+$/;
      if (!emailRegex.test(value)) {
        this.addError(name, `must be a valid email, got: ${value}`, required);
        return required ? null : value;
      }
    }

    // Pattern validation
    if (pattern) {
      const regex = new RegExp(pattern);
      if (!regex.test(value)) {
        this.addError(name, `must match pattern ${pattern}, got: ${value}`, required);
        return required ? null : value;
      }
    }

    // Choice validation
    if (choices && !choices.includes(value)) {
      this.addError(name, `must be one of [${choices.join(', ')}], got: ${value}`, re-
quired);
      return required ? null : value;
    }

    return value;
  }

  addError(name, message, required) {
    if (required) {
      this.errors.push(`${name}: ${message}`);
    } else {
      this.warnings.push(`${name}: ${message}`);
    }
  }

  /**
   * Throw if any validation errors occurred
   */
  throwIfInvalid() {
    if (this.errors.length > 0) {
      const errorMessage = [
        '❌ Environment validation failed:',
        '',
        ...this.errors.map(err => `  • ${err}`),
        '',
```

```
        'Please check your .env file and ensure all required variables are set.',
        'See .env.example for reference.',
      ].join('\n');

      throw new Error(errorMessage);
    }

    if (this.warnings.length > 0) {
      console.warn('⚠️  Environment validation warnings:');
      this.warnings.forEach(warn => console.warn(`   • ${warn}`));
      console.warn('');
    }
  }
}

// Validate and export configuration
function loadConfig() {
  const validator = new EnvValidator();

  const config = {
    // Environment
    nodeEnv: validator.optional('NODE_ENV', 'development', {
      choices: ['development', 'staging', 'production', 'test'],
    }),

    // Server
    port: validator.optional('PORT', 3000, { type: 'number', min: 1, max: 65535 }),
    healthPort: validator.optional('HEALTH_PORT', 3001, { type: 'number', min: 1,
max: 65535 }),

    // Slack
    slack: {
      botToken: validator.required('SLACK_BOT_TOKEN', {
        pattern: '^xoxb-[0-9]+-[0-9]+-[a-zA-Z0-9]+$',
      }),
      appToken: validator.required('SLACK_APP_TOKEN', {
        pattern: '^xapp-[0-9]+-[A-Z0-9]+-[0-9]+-[a-z0-9]+$',
      }),
      signingSecret: validator.required('SLACK_SIGNING_SECRET', {
        pattern: '^[a-f0-9]{32}$',
      }),
    },

    // Database
    database: {
      url: validator.required('DATABASE_URL', { type: 'url' }),
      poolSize: validator.optional('DATABASE_POOL_SIZE', 10, { type: 'number', min:
1, max: 100 }),
      ssl: validator.optional('DATABASE_SSL', true, { type: 'boolean' }),
    },

    // Logging
    logging: {
      level: validator.optional('LOG_LEVEL', 'info', {
        choices: ['debug', 'info', 'warn', 'error'],
      }),
    },

    // Monitoring
    sentry: {
      dsn: validator.optional('SENTRY_DSN', null, { type: 'url' }),
      environment: validator.optional('SENTRY_ENVIRONMENT', null),
      sampleRate: validator.optional('SENTRY_SAMPLE_RATE', 0.1, { type: 'number',
```

```
min: 0, max: 1 }),
    },

    // Features
    features: {
      voteHistory: validator.optional('FEATURE_VOTE_HISTORY', false, { type:
'boolean' }),
      analytics: validator.optional('FEATURE_ANALYTICS', false, { type: 'boolean' }),
    },
  };

  // Validate all configuration
  validator.throwIfInvalid();

  // Log loaded configuration (without sensitive values)
  console.log('✓ Configuration loaded:', {
    nodeEnv: config.nodeEnv,
    port: config.port,
    healthPort: config.healthPort,
    slack: {
      botToken: config.slack.botToken ? '[REDACTED]' : 'not set',
      appToken: config.slack.appToken ? '[REDACTED]' : 'not set',
      signingSecret: config.slack.signingSecret ? '[REDACTED]' : 'not set',
    },
    database: {
      url: config.database.url ? '[REDACTED]' : 'not set',
      poolSize: config.database.poolSize,
      ssl: config.database.ssl,
    },
    logging: config.logging,
    sentry: {
      dsn: config.sentry.dsn ? '[REDACTED]' : 'not set',
      environment: config.sentry.environment,
    },
    features: config.features,
  });

  return config;
}

// Singleton instance
let config = null;

function getConfig() {
  if (!config) {
    config = loadConfig();
  }
  return config;
}

module.exports = { getConfig };
```

Usage in Application

```
// index.js
require('dotenv').config();
const { getConfig } = require('./config/env');

// Validate environment variables on startup
const config = getConfig();

// Use configuration throughout application
const app = new App({
  token: config.slack.botToken,
  appToken: config.slack.appToken,
  signingSecret: config.slack.signingSecret,
  socketMode: true,
});

// Database connection
const pool = new Pool({
  connectionString: config.database.url,
  ssl: config.database.ssl ? { rejectUnauthorized: false } : false,
  max: config.database.poolSize,
});

// Logger configuration
const logger = winston.createLogger({
  level: config.logging.level,
  // ...
});
```

Comprehensive .env.example

```
# .env.example
# Copy this file to .env and fill in your values
# DO NOT commit .env to version control

# ================================================================================
# ENVIRONMENT
# ================================================================================
# Current environment: development, staging, or production
# Default: development
NODE_ENV=development

# ================================================================================
# SERVER CONFIGURATION
# ================================================================================
# Port for main application
# Railway.com will set this automatically in production
# Default: 3000
PORT=3000

# Port for health check endpoints
# Default: 3001
HEALTH_PORT=3001

# ================================================================================
# SLACK CONFIGURATION (Required)
# ================================================================================
# Slack Bot User OAuth Token
# Format: xoxb-XXXXXXXXXXXX-XXXXXXXXXXXX-XXXXXXXXXXXXXXXXXXXXXXXX
# How to get: https://api.slack.com/apps → Your App → OAuth & Permissions
# Required Scopes: app_mentions:read, chat:write, channels:history, commands
SLACK_BOT_TOKEN=xoxb-your-bot-token-here

# Slack App-Level Token (for Socket Mode)
# Format: xapp-X-XXXXXXXXXXX-XXXXXXXXXXXX-
XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX
# How to get: https://api.slack.com/apps → Your App → Basic Information → App-Level
Tokens
# Required Scopes: connections:write
SLACK_APP_TOKEN=xapp-your-app-token-here

# Slack Signing Secret
# Format: 32-character hexadecimal string
# How to get: https://api.slack.com/apps → Your App → Basic Information → App Creden-
tials
SLACK_SIGNING_SECRET=your-signing-secret-here

# ================================================================================
# DATABASE CONFIGURATION (Required for Production)
# ================================================================================
# PostgreSQL connection URL
# Format: postgresql://username:password@host:port/database
# Railway.com provides this automatically when you add PostgreSQL
# Example: postgresql://postgres:password@localhost:5432/ppbot
DATABASE_URL=postgresql://user:password@localhost:5432/ppbot

# Database connection pool size
# Default: 10
# Recommended for Railway.com free tier: 5
DATABASE_POOL_SIZE=10

# Enable SSL for database connection
# Set to true for production (Railway.com)
```

```
# Default: true
DATABASE_SSL=true

# ================================================================================
# LOGGING CONFIGURATION
# ================================================================================
# Log level: debug, info, warn, error
# Development: debug
# Production: info
LOG_LEVEL=info

# ================================================================================
# MONITORING & ERROR TRACKING (Optional)
# ================================================================================
# Sentry DSN for error tracking
# How to get: https://sentry.io → Create Project → Copy DSN
# Leave empty to disable Sentry
SENTRY_DSN=

# Sentry environment name
# Helps separate errors by environment in Sentry dashboard
SENTRY_ENVIRONMENT=development

# Sentry performance monitoring sample rate (0.0 to 1.0)
# 0.1 = 10% of transactions, 1.0 = 100% of transactions
# Default: 0.1 for production, 1.0 for development
SENTRY_SAMPLE_RATE=0.1

# ================================================================================
# FEATURE FLAGS (Optional)
# ================================================================================
# Enable vote history tracking
# Requires vote_history table in database
FEATURE_VOTE_HISTORY=false

# Enable analytics and metrics
FEATURE_ANALYTICS=false

# ================================================================================
# SECURITY NOTES
# ================================================================================
# - Never commit .env files to version control
# - Rotate tokens regularly (every 90 days recommended)
# - Use different tokens for development and production
# - Store production secrets in Railway.com environment variables
# - Limit bot token scopes to only what's needed
```

Environment-Specific Files

```
# .env.development
NODE_ENV=development
PORT=3000
HEALTH_PORT=3001
LOG_LEVEL=debug
DATABASE_SSL=false
SENTRY_SAMPLE_RATE=1.0
FEATURE_VOTE_HISTORY=true
FEATURE_ANALYTICS=true

# .env.production
NODE_ENV=production
LOG_LEVEL=info
DATABASE_SSL=true
SENTRY_SAMPLE_RATE=0.1
FEATURE_VOTE_HISTORY=true
FEATURE_ANALYTICS=true

# .env.test
NODE_ENV=test
LOG_LEVEL=error
DATABASE_URL=postgresql://localhost:5432/ppbot_test
DATABASE_SSL=false
```

Load Environment-Specific Config

```
// index.js
const path = require('path');
const dotenv = require('dotenv');

// Load environment-specific .env file
const envFile = process.env.NODE_ENV
  ? `.env.${process.env.NODE_ENV}`
  : '.env';

dotenv.config({ path: path.resolve(process.cwd(), envFile) });
dotenv.config(); // Load .env as fallback

const { getConfig } = require('./config/env');
const config = getConfig();
```

Railway.com Environment Variable Setup

Via Railway Dashboard

1. Navigate to your project on Railway.com
2. Click on your service
3. Go to “Variables” tab
4. Click “Raw Editor”
5. Paste your environment variables:

```
SLACK_BOT_TOKEN=xoxb-your-token
SLACK_APP_TOKEN=xapp-your-token
SLACK_SIGNING_SECRET=your-secret
NODE_ENV=production
LOG_LEVEL=info
SENTRY_DSN=https://your-sentry-dsn
```

1. Click “Save”
2. Railway will automatically redeploy with new variables

Via Railway CLI

```
# Set single variable
railway variables set SLACK_BOT_TOKEN=xoxb-your-token

# Set multiple variables from file
railway variables set -f .env.production

# List all variables
railway variables

# Delete variable
railway variables delete VARIABLE_NAME
```

Secrets Rotation Guide

When to Rotate

- Every 90 days (recommended)
- After a team member leaves
- If a secret is accidentally exposed
- After a security incident

How to Rotate Slack Tokens

1. **Generate New Token**
   - Go to https://api.slack.com/apps
   - Select your app
   - Navigate to OAuth & Permissions (for bot token) or Basic Information (for app token)
   - Click “Regenerate” or create new token

2. **Update Environment Variables**
   ```bash
   # Local development
   # Update .env file with new token

# Railway.com production
railway variables set SLACK_BOT_TOKEN=xoxb-new-token
```

1. **Deploy & Verify**
   - Wait for automatic redeployment
   - Check health endpoint: `curl https://your-app.railway.app/health`
   - Verify bot responds in Slack

2. **Revoke Old Token**
   - Back in Slack API dashboard
   - Revoke/delete the old token
   - Confirm bot still works with new token

How to Rotate Database Password

1. **Create New Database User** (or change password)
   ```sql
   – Connect to Railway PostgreSQL via CLI
   railway run psql $DATABASE_URL

– Create new user
CREATE USER ppbot_new WITH PASSWORD 'new-secure-password';
GRANT ALL PRIVILEGES ON DATABASE ppbot TO ppbot_new;
```

1. **Update DATABASE_URL**
   ```bash
   railway variables set DATABASE_URL=postgresql://ppbot_new:new-password@host:5432/ppbot
   ```

2. **Verify & Clean Up**
   - Wait for redeployment
   - Verify database connectivity
   - Drop old user: `DROP USER old_user;`

Security Checklist

Development

- [ ] `.env` file in `.gitignore`
- [ ] `.env.example` committed without secrets
- [ ] Different tokens for dev and prod
- [ ] Local `.env` file never shared

Production

- [ ] All secrets stored in Railway.com variables
- [ ] No secrets in GitHub repository
- [ ] Secrets rotated within 90 days
- [ ] Access logs reviewed regularly
- [ ] Principle of least privilege applied
- [ ] SSL/TLS enabled for database
- [ ] Environment variables validated on startup

Team Practices

- [ ] Document how to obtain secrets
- [ ] Onboarding includes secrets setup
- [ ] Offboarding includes secrets rotation
- [ ] Secrets shared via secure method (1Password, LastPass)
- [ ] Production access restricted to authorized team members

Testing

Test Environment Validator

```
// config/env.test.js
const { getConfig } = require('./env');

describe('Environment Configuration', () => {
  beforeEach(() => {
    // Save original env
    this.originalEnv = { ...process.env };
  });

  afterEach(() => {
    // Restore original env
    process.env = this.originalEnv;
  });

  test('throws error if required variable missing', () => {
    delete process.env.SLACK_BOT_TOKEN;
    expect(() => getConfig()).toThrow('Missing required environment variable:
SLACK_BOT_TOKEN');
  });

  test('validates Slack bot token format', () => {
    process.env.SLACK_BOT_TOKEN = 'invalid-token';
    expect(() => getConfig()).toThrow(/must match pattern/);
  });

  test('uses default values for optional variables', () => {
    delete process.env.PORT;
    const config = getConfig();
    expect(config.port).toBe(3000);
  });

  test('validates number types', () => {
    process.env.PORT = 'not-a-number';
    expect(() => getConfig()).toThrow(/must be a number/);
  });

  test('validates choices', () => {
    process.env.NODE_ENV = 'invalid-env';
    expect(() => getConfig()).toThrow(/must be one of/);
  });
});
```

Documentation

README.md Section

Add to README:

```
## Environment Variables

See `.env.example` for a complete list of environment variables. Copy it to `.env` and
fill in your values:

```bash
cp .env.example .env
```

Required Variables

Variable	Description	How to Get
`SLACK_BOT_TOKEN`	Bot User OAuth Token	Slack API Dashboard (https://api.slack.com/apps) → OAuth & Permissions
`SLACK_APP_TOKEN`	App-Level Token	Slack API Dashboard (https://api.slack.com/apps) → Basic Information
`SLACK_SIGNING_SECRET`	Signing Secret	Slack API Dashboard (https://api.slack.com/apps) → Basic Information
`DATABASE_URL`	PostgreSQL connection URL	Railway.com provides this automatically

Optional Variables

Variable	Default	Description
`NODE_ENV`	`development`	Environment (development/staging/production)
`PORT`	`3000`	Main application port
`LOG_LEVEL`	`info`	Logging level (debug/info/warn/error)
`SENTRY_DSN`	-	Sentry error tracking DSN

See `.env.example` for complete documentation.
```

Acceptance Criteria

- [ ] Environment validator validates all required variables
- [ ] Type and format validation works correctly
- [ ] Fails fast with clear error messages

- [ ] `.env.example` comprehensively documents all variables
- [ ] Multi-environment support works (.env.development, etc.)
- [ ] Railway.com environment setup documented
- [ ] Secrets rotation guide complete
- [ ] Security checklist documented
- [ ] Tests cover validation logic
- [ ] Configuration loaded and logged on startup

Resources

- The Twelve-Factor App: Config (https://12factor.net/config)
- Node.js Environment Variables Best Practices (https://www.twilio.com/blog/working-with-environment-variables-in-node-js-html)
- Railway.com Environment Variables (https://docs.railway.app/develop/variables)
- dotenv Documentation (https://github.com/motdotla/dotenv)

Dependencies

This issue enhances all other issues but doesn’t strictly block anything.