Comprehensive File Structure Recommendations for Integrated AI System

Project Structure Overview

```
integrated-ai-system/
├── README.md
├── package.json
├── tsconfig.json
├── .gitignore
├── .env.example
├── .env
├── docker-compose.yml
├── Dockerfile
├── docs/
│   ├── api/
│   ├── architecture/
│   ├── deployment/
│   └── user-guide/
├── src/
│   ├── components/
│   ├── services/
│   ├── utils/
│   ├── types/
│   ├── hooks/
│   ├── config/
│   └── tests/
├── public/
├── assets/
├── data/
├── scripts/
├── config/
└── dist/
```

Detailed File Structure with Descriptions

1. Root Configuration Files

package.json

```
{
  "name": "integrated-ai-system",
  "version": "1.0.0",
  "description": "Advanced integrated AI system with memory
management and file processing",
  "main": "dist/index.js",
  "scripts": {
    "dev": "vite",
    "build": "tsc && vite build",
    "preview": "vite preview",
    "test": "jest",
    "test:watch": "jest --watch",
    "lint": "eslint src --ext .ts,.tsx",
    "lint:fix": "eslint src --ext .ts,.tsx --fix",
    "type-check": "tsc --noEmit",
    "docker:build": "docker build -t integrated-ai-system .",
    "docker:run": "docker run -p 3000:3000 integrated-ai-system"
  },
  "dependencies": {
    "react": "^18.2.0",
    "react-dom": "^18.2.0",
    "@types/react": "^18.2.0",
    "@types/react-dom": "^18.2.0",
    "typescript": "^5.0.0",
    "vite": "^4.4.0",
    "axios": "^1.4.0",
    "lodash": "^4.17.21",
    "date-fns": "^2.30.0",
    "uuid": "^9.0.0",
    "zod": "^3.21.4",
    "zustand": "^4.3.9"
  },
  "devDependencies": {
    "@vitejs/plugin-react": "^4.0.0",
    "@types/lodash": "^4.14.195",
    "@types/uuid": "^9.0.2",
    "jest": "^29.5.0",
    "@testing-library/react": "^13.4.0",
    "@testing-library/jest-dom": "^5.16.5",
    "eslint": "^8.45.0",
    "@typescript-eslint/eslint-plugin": "^6.0.0",
    "@typescript-eslint/parser": "^6.0.0",
    "prettier": "^3.0.0"
```

```
  }
}
```

tsconfig.json

```
{
  "compilerOptions": {
    "target": "ES2020",
    "useDefineForClassFields": true,
    "lib": ["ES2020", "DOM", "DOM.Iterable"],
    "module": "ESNext",
    "skipLibCheck": true,
    "moduleResolution": "bundler",
    "allowImportingTsExtensions": true,
    "resolveJsonModule": true,
    "isolatedModules": true,
    "noEmit": true,
    "jsx": "react-jsx",
    "strict": true,
    "noUnusedLocals": true,
    "noUnusedParameters": true,
    "noFallthroughCasesInSwitch": true,
    "baseUrl": ".",
    "paths": {
      "@/*": ["src/*"],
      "@/components/*": ["src/components/*"],
      "@/services/*": ["src/services/*"],
      "@/utils/*": ["src/utils/*"],
      "@/types/*": ["src/types/*"],
      "@/hooks/*": ["src/hooks/*"],
      "@/config/*": ["src/config/*"]
    }
  },
  "include": ["src"],
  "references": [{ "path": "./tsconfig.node.json" }]
}
```

.gitignore

```
# Dependencies
node_modules/
npm-debug.log*
yarn-debug.log*
yarn-error.log*

# Production builds
dist/
build/
```

```
# Environment variables
.env
.env.local
.env.development.local
.env.test.local
.env.production.local

# IDE files
.vscode/
.idea/
*.swp
*.swo

# OS generated files
.DS_Store
.DS_Store?
._*
.Spotlight-V100
.Trashes
ehthumbs.db
Thumbs.db

# Logs
logs/
*.log

# Runtime data
pids/
*.pid
*.seed
*.pid.lock

# Coverage directory used by tools like istanbul
coverage/

# Temporary folders
tmp/
temp/

# Docker
.dockerignore
```

.env.example

```
# API Configuration
API_BASE_URL=http://localhost:3001
API_KEY=your_api_key_here

# Memory System Configuration
MEMORY_COMPRESSION_RATIO=0.7
```

```
MEMORY_RETENTION_THRESHOLD=0.8
MEMORY_CLEANUP_INTERVAL=300000

# File Processing Configuration
MAX_FILE_SIZE=50MB
SUPPORTED_EXTENSIONS=tsx,rs,js,py,cpp,html,css,md,pdf,json,csv,xml,yaml,pr
PROCESSING_QUEUE_SIZE=100

# Performance Monitoring
ENABLE_PERFORMANCE_LOGGING=true
LOG_LEVEL=info
METRICS_ENDPOINT=http://localhost:9090

# Security
CORS_ORIGIN=*
RATE_LIMIT_REQUESTS=1000
RATE_LIMIT_WINDOW=900000
```

2. Source Code Structure

src/types/index.ts

```typescript
export interface MemorySystemState {
  shortTerm: MemoryItem[];
  longTerm: MemoryItem[];
  archive: MemoryItem[];
  compressionRatio: number;
  retentionScore: number;
  cyclicCleanup: number;
}

export interface MemoryItem {
  id: string;
  content: any;
  timestamp: Date;
  accessCount: number;
  importance: number;
  tags: string[];
  metadata: Record<string, any>;
}

export interface FileProcessingState {
  queue: ProcessingFile[];
  processed: ProcessedFile[];
  categories: FileCategories;
  locations: Map<string, string>;
  encoding: Map<string, string>;
}

export interface ProcessingFile {
```

```
  id: string;
  name: string;
  path: string;
  size: number;
  type: string;
  status: 'pending' | 'processing' | 'completed' | 'error';
  priority: number;
}

export interface ProcessedFile extends ProcessingFile {
  processedAt: Date;
  metadata: FileMetadata;
  content?: any;
  errors?: string[];
}

export interface FileCategories {
  code: FileCategory;
  documents: FileCategory;
  data: FileCategory;
  multimedia: FileCategory;
  archives: FileCategory;
  executables: FileCategory;
}

export interface FileCategory {
  count: number;
  types: string[];
  totalSize: number;
  lastUpdated: Date;
}

export interface FileMetadata {
  size: number;
  createdAt: Date;
  modifiedAt: Date;
  author?: string;
  encoding?: string;
  checksum: string;
  contentType: string;
  extractedText?: string;
  semanticTags?: string[];
}

export interface PerformanceLogEntry {
  id: string;
  timestamp: Date;
  eventType: string;
  duration?: number;
  resourceUsage: ResourceUsage;
  status: 'success' | 'failure' | 'warning';
  details: Record<string, any>;
```

```
}

export interface ResourceUsage {
  cpu: number;
  memory: number;
  diskIO: number;
  networkIO: number;
}
```

src/config/fileTypes.json

```
{
  "categories": {
    "code": {
      "extensions": [
        "tsx", "ts", "jsx", "js", "py", "cpp", "c", "h",
"java", "kt",
        "go", "php", "rb", "pl", "lua", "dart", "R", "jl", "f",
"vhd",
        "sv", "asm", "wasm", "html", "css", "scss", "less",
"vue", "svelte",
        "sql", "sh", "bash", "ps1", "bat", "cmd", "swift", "m"
      ],
      "mimeTypes": [
        "text/javascript", "text/typescript", "text/x-python",
        "text/x-c", "text/x-java-source", "text/html", "text/
css"
      ]
    },
    "documents": {
      "extensions": [
        "md", "pdf", "docx", "doc", "txt", "rtf", "odt",
"epub", "tex",
        "xlsx", "xls", "ods", "pptx", "ppt", "odp", "log", "nfo"
      ],
      "mimeTypes": [
        "application/pdf", "text/plain", "text/markdown",
        "application/vnd.openxmlformats-
officedocument.wordprocessingml.document",
        "application/vnd.openxmlformats-
officedocument.spreadsheetml.sheet"
      ]
    },
    "data": {
      "extensions": [
        "json", "csv", "xml", "yaml", "yml", "toml", "ini",
"parquet",
        "avro", "orc", "sqlite", "db", "hdf5", "h5", "feather",
"pickle",
        "pkl", "geojson", "topojson", "gpx", "kml", "rdf",
```

```
"ttl", "graphml"
      ],
      "mimeTypes": [
        "application/json", "text/csv", "application/xml",
"text/yaml",
        "application/x-sqlite3"
      ]
    },
    "multimedia": {
      "extensions": [
        "png", "jpg", "jpeg", "gif", "bmp", "tiff", "webp",
"svg", "ico",
        "psd", "ai", "eps", "raw", "dng", "heic", "avif",
"mp3", "wav",
        "aac", "flac", "ogg", "m4a", "wma", "aiff", "opus",
"mp4", "mov",
        "avi", "wmv", "flv", "webm", "mkv", "3gp", "obj",
"fbx", "gltf",
        "glb", "stl", "dae", "blend", "ttf", "otf", "woff",
"woff2"
      ],
      "mimeTypes": [
        "image/png", "image/jpeg", "image/gif", "image/svg+xml",
        "audio/mpeg", "audio/wav", "video/mp4", "video/webm",
        "font/ttf", "font/otf"
      ]
    },
    "archives": {
      "extensions": [
        "zip", "tar", "gz", "rar", "7z", "bz2", "xz", "iso",
"dmg"
      ],
      "mimeTypes": [
        "application/zip", "application/x-tar", "application/
gzip",
        "application/x-rar-compressed"
      ]
    },
    "executables": {
      "extensions": [
        "exe", "dll", "so", "dylib", "apk", "deb", "rpm"
      ],
      "mimeTypes": [
        "application/x-executable", "application/x-sharedlib",
        "application/vnd.android.package-archive"
      ]
    }
  },
  "security": {
    "allowedExecutables": false,
    "maxFileSize": "50MB",
    "quarantineExtensions": ["exe", "dll", "bat", "cmd", "scr"],
```

```
    "scanForMalware": true
  }
}
```

3. Documentation Files

docs/README.md

```
# Integrated AI System Documentation

## Overview
This documentation provides comprehensive information about the
Integrated AI System, including architecture, API references,
deployment guides, and user manuals.

## Documentation Structure

- **API Documentation**: Detailed API endpoints and usage
examples
- **Architecture**: System design and component interactions
- **Deployment**: Installation and deployment instructions
- **User Guide**: End-user documentation and tutorials

## Quick Start
1. [Installation Guide](deployment/installation.md)
2. [Configuration](deployment/configuration.md)
3. [API Reference](api/endpoints.md)
4. [User Guide](user-guide/getting-started.md)
```

docs/api/endpoints.md

```
# API Endpoints Documentation

## Memory System API

### GET /api/memory/status
Returns the current status of the memory system.

**Response:**
```json
{
  "shortTermCount": 150,
  "longTermCount": 1200,
  "archiveCount": 5000,
  "compressionRatio": 0.75,
  "retentionScore": 0.82,
```
```

```
  "lastCleanup": "2024-01-15T10:30:00Z"
}
```

POST /api/memory/store

Stores a new memory item.

Request Body:

```
{
  "content": "Memory content",
  "importance": 0.8,
  "tags": ["important", "user-interaction"],
  "metadata": {
    "source": "user-input",
    "context": "conversation"
  }
}
```

File Processing API

POST /api/files/upload

Uploads and processes a file.

Request: - Multipart form data with file - Optional metadata in JSON format

Response:

```
{
  "fileId": "uuid-string",
  "status": "processing",
  "estimatedTime": "30s",
  "category": "document"
}
```

GET /api/files/{fileId}/status

Returns the processing status of a file.

GET /api/files/categories

Returns file processing statistics by category.

```
#### docs/architecture/system-design.md
```markdown
# System Architecture

## Overview
The Integrated AI System follows a modular, event-driven
architecture designed for scalability and maintainability.

## Core Components

### 1. Memory Management System
- **Short-term Memory**: Fast access for current session data
- **Long-term Memory**: Persistent storage for important
information
- **Archive**: Compressed historical data
- **Compression Engine**: Intelligent data compression and
retention

### 2. File Processing Engine
- **Asynchronous Queue**: Non-blocking file processing
- **Content Analysis**: Intelligent categorization and metadata
extraction
- **Multi-format Support**: Handles diverse file types
- **Security Scanning**: Malware detection and quarantine

### 3. Performance Monitoring
- **Real-time Metrics**: CPU, memory, and I/O monitoring
- **Event Logging**: Structured logging for all system events
- **Anomaly Detection**: Automatic detection of performance
issues

## Data Flow
1. Input → Validation → Processing Queue
2. Processing → Content Analysis → Categorization
3. Storage → Memory System → Indexing
4. Retrieval → Search → Response

## Security Considerations
- Input validation and sanitization
- File type restrictions and scanning
- Access control and authentication
- Data encryption at rest and in transit
```

4. Configuration and Scripts

docker-compose.yml

```
version: '3.8'
```
```

```yaml
services:
  app:
    build: .
    ports:
      - "3000:3000"
    environment:
      - NODE_ENV=production
      - API_BASE_URL=http://api:3001
    depends_on:
      - api
      - redis
      - postgres
    volumes:
      - ./data:/app/data

  api:
    build: ./api
    ports:
      - "3001:3001"
    environment:
      - DATABASE_URL=postgresql://user:password@postgres:5432/
aidb
      - REDIS_URL=redis://redis:6379
    depends_on:
      - postgres
      - redis

  postgres:
    image: postgres:15
    environment:
      - POSTGRES_DB=aidb
      - POSTGRES_USER=user
      - POSTGRES_PASSWORD=password
    volumes:
      - postgres_data:/var/lib/postgresql/data

  redis:
    image: redis:7-alpine
    volumes:
      - redis_data:/data

  prometheus:
    image: prom/prometheus
    ports:
      - "9090:9090"
    volumes:
      - ./config/prometheus.yml:/etc/prometheus/prometheus.yml

  grafana:
    image: grafana/grafana
    ports:
      - "3001:3000"
```

```
    environment:
      - GF_SECURITY_ADMIN_PASSWORD=admin
    volumes:
      - grafana_data:/var/lib/grafana

volumes:
  postgres_data:
  redis_data:
  grafana_data:
```

scripts/setup.sh

```
#!/bin/bash

# Setup script for Integrated AI System

echo "Setting up Integrated AI System..."

# Check Node.js version
if ! command -v node &> /dev/null; then
    echo "Node.js is required but not installed. Please install
Node.js 18 or higher."
    exit 1
fi

# Install dependencies
echo "Installing dependencies..."
npm install

# Setup environment variables
if [ ! -f .env ]; then
    echo "Creating environment file..."
    cp .env.example .env
    echo "Please edit .env file with your configuration"
fi

# Create necessary directories
mkdir -p data/{uploads,processed,logs}
mkdir -p dist
mkdir -p tmp

# Set permissions
chmod +x scripts/*.sh

# Build the project
echo "Building project..."
npm run build
```

```
echo "Setup complete! Run 'npm run dev' to start development
server."
```

5. Testing Files

src/tests/memory.test.ts

```
import { describe, it, expect, beforeEach } from '@jest/
globals';
import { MemorySystem } from '../services/MemorySystem';
import { MemoryItem } from '../types';

describe('MemorySystem', () => {
  let memorySystem: MemorySystem;

  beforeEach(() => {
    memorySystem = new MemorySystem();
  });

  it('should store memory items correctly', () => {
    const item: MemoryItem = {
      id: '1',
      content: 'Test memory',
      timestamp: new Date(),
      accessCount: 0,
      importance: 0.8,
      tags: ['test'],
      metadata: {}
    };

    memorySystem.store(item);
    expect(memorySystem.getShortTermCount()).toBe(1);
  });

  it('should compress memories when threshold is reached', ()
=> {
    // Add multiple items to trigger compression
    for (let i = 0; i < 100; i++) {
      const item: MemoryItem = {
        id: i.toString(),
        content: `Test memory ${i}`,
        timestamp: new Date(),
        accessCount: 0,
        importance: Math.random(),
        tags: ['test'],
        metadata: {}
      };
      memorySystem.store(item);
    }
```

```
    memorySystem.compress();
    expect(memorySystem.getLongTermCount()).toBeGreaterThan(0);
  });
});
```

jest.config.js

```
module.exports = {
  preset: 'ts-jest',
  testEnvironment: 'jsdom',
  setupFilesAfterEnv: ['<rootDir>/src/tests/setup.ts'],
  moduleNameMapping: {
    '^@/(.*)$': '<rootDir>/src/$1',
  },
  collectCoverageFrom: [
    'src/**/*.{ts,tsx}',
    '!src/**/*.d.ts',
    '!src/tests/**/*',
  ],
  coverageThreshold: {
    global: {
      branches: 80,
      functions: 80,
      lines: 80,
      statements: 80,
    },
  },
};
```

This comprehensive file structure provides a solid foundation for the integrated AI system with proper organization, documentation, testing, and deployment configurations.