MasjidSuite - Complete Technical Specification

Version 1.0 | Implementation Guide for AI Development

🎯PROJECT OVERVIEW

Project Name: MasjidSuite
Purpose: Replace expensive monthly subscription mosque management software with a free, offline-capable alternative
Target Savings: 6,000+ MYR annually per mosque
Current Problem: Mosques paying 500 MYR/month for basic prayer time display + technician "updates"

📋REQUIREMENTS SUMMARY

Core Requirements

- ✅**Offline-first**: Must work without internet
- ✅**Zero monthly costs**: No subscriptions, no licensing fees
- ✅**Low-maintenance**: Imam can operate without technical support
- ✅**Secure**: Encrypted backups, protected community data
- ✅**Multi-display**: Control panel + external display support
- ✅**Prayer automation**: Auto-calculation, countdown, Azan playback

Hardware Requirements

- **Minimum**: Any laptop/desktop with HDMI output
- **Recommended**: Windows 10/11, 4GB RAM, 500MB storage
- **Alternative**: Raspberry Pi 4 for dedicated kiosk mode

🏗️SYSTEM ARCHITECTURE

Technology Stack

```
Frontend: React 18 + TypeScript
Backend: Electron 28+ (Node.js)
Database: SQLite3 (embedded)
Encryption: Node.js crypto module (AES-256-GCM)
Prayer Times: Custom calculation library
Audio: Web Audio API
Deployment: Electron Builder (Windows .exe)
```

Project Structure

```
masjid-suite/
├── src/
│   ├── main/          # Electron main process
│   ├── renderer/      # React frontend
│   ├── shared/        # Shared utilities
│   └── assets/        # Images, audio files
├── database/          # SQLite schemas
├── docs/              # User documentation
├── build/             # Build configurations
└── releases/          # Distribution packages
```

🗄️DATABASE DESIGN

SQLite Schema

sql

```sql
-- Settings table
CREATE TABLE settings (
  id INTEGER PRIMARY KEY,
  location TEXT NOT NULL DEFAULT 'Kuala Lumpur',
  prayer_method TEXT NOT NULL DEFAULT 'MuslimWorldLeague',
  azan_volume INTEGER DEFAULT 80,
  selected_azan TEXT DEFAULT 'default.mp3',
  admin_pin_hash TEXT,
  language TEXT DEFAULT 'English',
  created_at DATETIME DEFAULT CURRENT_TIMESTAMP,
  updated_at DATETIME DEFAULT CURRENT_TIMESTAMP
);

-- Prayer times cache
CREATE TABLE prayer_times (
  id INTEGER PRIMARY KEY,
  date TEXT NOT NULL UNIQUE,
  fajr TEXT NOT NULL,
  dhuhr TEXT NOT NULL,
  asr TEXT NOT NULL,
  maghrib TEXT NOT NULL,
  isha TEXT NOT NULL,
  created_at DATETIME DEFAULT CURRENT_TIMESTAMP
);

-- Event banners
CREATE TABLE event_banners (
  id INTEGER PRIMARY KEY,
  title TEXT NOT NULL,
  date_time TEXT NOT NULL,
  location TEXT NOT NULL,
  guest_of_honor TEXT,
  background_template TEXT,
  notes TEXT,
  created_at DATETIME DEFAULT CURRENT_TIMESTAMP
);

-- Community database (Qariyah)
CREATE TABLE qariyah_members (
  id INTEGER PRIMARY KEY,
  name TEXT NOT NULL,
  contact TEXT,
  address TEXT,
  household_size INTEGER DEFAULT 1,
  needs TEXT,
  notes TEXT,
  created_at DATETIME DEFAULT CURRENT_TIMESTAMP,
  updated_at DATETIME DEFAULT CURRENT_TIMESTAMP
);

-- Aid history
CREATE TABLE aid_history (
  id INTEGER PRIMARY KEY,
  member_id INTEGER NOT NULL,
  date TEXT NOT NULL,
  item TEXT NOT NULL,
  notes TEXT,
  created_at DATETIME DEFAULT CURRENT_TIMESTAMP,
  FOREIGN KEY (member_id) REFERENCES qariyah_members(id)
);

-- Backup logs
CREATE TABLE backup_logs (
  id INTEGER PRIMARY KEY,
  backup_type TEXT NOT NULL,
  filename TEXT NOT NULL,
  status TEXT NOT NULL,
  created_at DATETIME DEFAULT CURRENT_TIMESTAMP
);
```

🕌PRAYER TIME CALCULATION

Implementation Requirements

javascript

```javascript
// Prayer time calculation using astronomical formulas
class PrayerTimeCalculator {
  constructor(latitude, longitude, method = 'MuslimWorldLeague') {
    this.latitude = latitude;
    this.longitude = longitude;
    this.method = method;
    this.methods = {
      'MuslimWorldLeague': { fajr: 18, isha: 17 },
      'IslamicSocietyNorthAmerica': { fajr: 15, isha: 15 },
      'UmmAlQura': { fajr: 18.5, isha: 90 }
    };
  }

  calculatePrayerTimes(date) {
    // Implement astronomical calculations
    // Return: { fajr, dhuhr, asr, maghrib, isha }
  }
}
```

Key Features

- **Automatic calculation**: Generate prayer times for next 30 days
- **Multiple methods**: Support MWL, ISNA, Umm al-Qura
- **Timezone handling**: Automatic Malaysia timezone (GMT+8)
- **Daylight saving**: Handle time changes automatically
- **Validation**: Ensure realistic prayer time intervals

🎨EVENT BANNER CREATOR

Component Requirements

typescript

```typescript
interface BannerData {
  title: string;
  dateTime: string;
  location: string;
  guestOfHonor?: string;
  backgroundTemplate: string;
  notes?: string;
}

interface BannerTemplate {
  id: string;
  name: string;
  backgroundImage: string;
  layout: 'standard' | 'minimal' | 'formal';
  dimensions: { width: number; height: number };
}
```

Implementation Steps

1. **Template Selection**: 5+ pre-designed backgrounds
2. **Form Interface**: Clean input fields for event details
3. **Live Preview**: Real-time banner preview
4. **Export Options**: PNG/JPEG at 1920x1080 resolution
5. **Storage**: Save banner configs for reuse

Banner Templates Required

- **Template 1**: Islamic geometric pattern (general events)
- **Template 2**: Mosque silhouette (formal events)
- **Template 3**: Arabic calligraphy border (lectures)
- **Template 4**: Minimal design (announcements)
- **Template 5**: Ramadan/Eid themed (seasonal)

📜QARIYAH DATABASE

Data Model

typescript

```typescript
interface QariyahMember {
  id: number;
  name: string;
  contact?: string;
  address?: string;
  householdSize: number;
  needs?: string;
  notes?: string;
  aidHistory: AidRecord[];
  createdAt: Date;
  updatedAt: Date;
}

interface AidRecord {
  id: number;
  date: string;
  item: string;
  notes?: string;
}
```

Required Features

- **CRUD Operations**: Add, edit, delete, view members
- **Search & Filter**: By name, address, needs, household size
- **Aid Tracking**: Record and view aid distribution history
- **Privacy Protection**: Encrypt sensitive data in backups
- **Export Options**: Generate aid recipient lists

🔐SECURITY & BACKUP SYSTEM

Security Questions Setup

javascript

```javascript
const securityQuestions = [
  { id: 'color', question: 'What is your favorite color?', placeholder: 'e.g., blue' },
  { id: 'animal', question: 'What is your favorite animal?', placeholder: 'e.g., cat' },
  { id: 'developer', question: 'What is the developer\'s name?', placeholder: 'Your name' }
];
```

Encryption Implementation

javascript

```javascript
// Key derivation from security answers
function deriveKey(answers, salt) {
  const passphrase = answers.join('|').toLowerCase();
  return crypto.pbkdf2Sync(passphrase, salt, 100000, 32, 'sha512');
}

// Backup encryption
function encryptBackup(data, answers) {
  const salt = crypto.randomBytes(16);
  const key = deriveKey(answers, salt);
  const iv = crypto.randomBytes(16);
  const cipher = crypto.createCipher('aes-256-gcm', key);

  let encrypted = cipher.update(JSON.stringify(data), 'utf8', 'base64');
  encrypted += cipher.final('base64');

  return {
    salt: salt.toString('base64'),
    iv: iv.toString('base64'),
    data: encrypted,
    authTag: cipher.getAuthTag().toString('base64')
  };
}
```

Backup File Format

json

```json
{
  "🐪spiceBucket": "base64-encoded-salt",
  "🧠vectorSpace": "base64-encoded-iv",
  "🧩doodle64": "base64-encoded-encrypted-data",
  "🔐sealStamp": "base64-encoded-auth-tag",
  "version": "1.0",
  "timestamp": "2025-07-03T12:00:00Z"
}
```

🖥️USER INTERFACE DESIGN

Main Application Layout

```
┌──────────────────────────────────────────────────────────
│ MasjidSuite                    [_] [□] [X] │
├──────────────────────────────────────────────────────────
│  ┌───────────────┐
┌──────────────────────────────────────────────────┐ │
││🕌 Prayer   ││                   ││
││🎨 Events   ││    Main Content Area    ││
││📜 Qariyah  ││                   ││
││🔐 Backup   ││                   ││
││⚙️ Settings ││                   ││
│  └───────────────┘
└──────────────────────────────────────────────────┘ │
└──────────────────────────────────────────────────────────
```

Prayer Display (External Monitor)

```
┌──────────────────────────────────────────────────────────
│                                  │
│        🕌 MASJID AL-IKHLAS              │
│                                  │
│        Next Prayer: MAGHRIB             │
│                                  │
│           ⏰ 02:34:17             │
│                                  │
│ Fajr   Dhuhr   Asr   Maghrib   Isha      │
│ 05:45  13:15  16:30   19:15   20:30       │
│                                  │
│        🕐 Mecca Time: 14:15           │
│                                  │
└──────────────────────────────────────────────────────────
```

Component Specifications

Prayer Display Component

```typescript
interface PrayerDisplayProps {
  isFullscreen: boolean;
  currentTime: Date;
  nextPrayer: {
    name: string;
    time: string;
    countdown: string;
  };
  todaysPrayers: PrayerTime[];
  meccaTime: string;
}
```

Event Banner Creator

```typescript
interface BannerCreatorProps {
  templates: BannerTemplate[];
  onSave: (banner: BannerData) => void;
  onExport: (banner: BannerData) => void;
}
```

🔊 AUDIO SYSTEM

Azan Playback Requirements

javascript

```javascript
class AzanPlayer {
  constructor() {
    this.audioContext = new AudioContext();
    this.currentAudio = null;
    this.volume = 0.8;
  }

  async playAzan(filename) {
    // Load and play Azan audio file
    // Support MP3, WAV formats
    // Respect system volume settings
  }

  setVolume(level) {
    // Volume control 0-100
  }

  stop() {
    // Stop current playback
  }
}
```

Audio Files Required

- **default.mp3**: Standard Azan (5-7 minutes)
- **short.mp3**: Brief Azan (2-3 minutes)
- **madinah.mp3**: Madinah-style Azan
- **notification.mp3**: Simple notification sound

🚀 DEVELOPMENT IMPLEMENTATION STEPS

Phase 1: Core Infrastructure (Week 1)

1. **Project Setup**
   - Initialize Electron + React project
   - Configure TypeScript
   - Set up SQLite database
   - Create basic window management
2. **Database Layer**
   - Implement SQLite schemas
   - Create data access layer
   - Add migration system
   - Test CRUD operations
3. **Prayer Time Engine**
   - Implement calculation algorithms
   - Add timezone handling
   - Create caching system
   - Test with real dates

Phase 2: User Interface (Week 2)

1. **Main Application Shell**
   - Create sidebar navigation
   - Implement routing
   - Add responsive layout
   - Theme system setup
2. **Prayer Display**
   - Real-time countdown
   - Prayer schedule display
   - Fullscreen mode
   - External monitor support
3. **Basic Settings**
   - Location configuration
   - Prayer method selection
   - Audio settings
   - Display preferences

Phase 3: Event Management (Week 3)

1. **Banner Creator**
    - Template system
    - Form inputs
    - Live preview
    - Export functionality
2. **Template Assets**
    - Design 5 banner templates
    - Optimize for 1920x1080
    - Test font rendering
    - Ensure print quality

Phase 4: Community Database (Week 4)

1. **Qariyah Management**
    - Member CRUD interface
    - Search and filtering
    - Aid history tracking
    - Data validation
2. **Privacy Features**
    - Sensitive data handling
    - Access controls
    - Data encryption

Phase 5: Security & Backup (Week 5)

1. **Security Questions**
    - Setup interface
    - Answer validation
    - Key derivation
    - Session management
2. **Backup System**
    - Data serialization
    - Encryption implementation
    - File generation
    - Restore functionality

Phase 6: Testing & Polish (Week 6)

1. **Integration Testing**
    - End-to-end workflows
    - Error handling
    - Performance testing
    - Memory leak detection
2. **User Experience**
    - Loading states
    - Error messages
    - Keyboard shortcuts
    - Accessibility features

📦BUILD & DEPLOYMENT

Build Configuration

json

```json
// package.json
{
  "name": "masjid-suite",
  "version": "1.0.0",
  "main": "dist/main.js",
  "scripts": {
    "dev": "electron-forge start",
    "build": "electron-forge make",
    "package": "electron-forge package",
    "publish": "electron-forge publish"
  },
  "devDependencies": {
    "@electron-forge/cli": "^7.0.0",
    "@electron-forge/maker-squirrel": "^7.0.0",
    "@electron-forge/maker-zip": "^7.0.0",
    "@electron-forge/plugin-auto-unpack-natives": "^7.0.0",
    "@electron-forge/plugin-fuses": "^7.0.0"
  }
}
```

Distribution Requirements

- **Windows**: .exe installer with auto-updater
- **File Size**: < 100MB total
- **Install Location**: Program Files/MasjidSuite
- **Desktop Shortcut**: Yes
- **Start Menu**: Yes
- **Uninstaller**: Standard Windows uninstaller

🎯TESTING CHECKLIST

Functional Testing

- [ ] Prayer times calculate correctly for Kuala Lumpur
- [ ] Countdown timer updates every second
- [ ] Azan plays automatically at prayer time
- [ ] External monitor displays correctly
- [ ] Banner creator exports proper images
- [ ] Qariyah database handles 1000+ records
- [ ] Backup encryption/decryption works
- [ ] Security questions lock access properly
- [ ] Application starts on system boot (optional)
- [ ] Works offline completely

Performance Testing

- [ ] Application starts in < 5 seconds
- [ ] Memory usage < 200MB
- [ ] CPU usage < 5% when idle
- [ ] Database queries < 100ms
- [ ] Export operations < 10 seconds
- [ ] Runs continuously for 24+ hours

Error Handling

- [ ] Graceful audio failure handling
- [ ] Database corruption recovery
- [ ] Invalid backup file handling
- [ ] Network unavailable scenarios
- [ ] Disk space limitations
- [ ] Permission errors

📋USER DOCUMENTATION

Installation Guide

1. **Download**: Get MasjidSuite-Setup.exe
2. **Install**: Run installer as administrator
3. **First Run**: Complete initial setup wizard
4. **Location**: Confirm mosque location
5. **Security**: Set up security questions
6. **Audio**: Test Azan playback
7. **Display**: Configure external monitor

Daily Operation

1. **Startup**: Application auto-starts with Windows
2. **Prayer Display**: Shows automatically on external monitor
3. **Azan**: Plays automatically at prayer times
4. **Manual Control**: Use admin interface for adjustments

Maintenance Tasks

- **Weekly**: Review prayer times for accuracy
- **Monthly**: Create backup file
- **Quarterly**: Update community database
- **Annually**: Review security questions

🔧TROUBLESHOOTING GUIDE

Common Issues

1. **Prayer times incorrect**
   - Check location settings
   - Verify calculation method
   - Confirm timezone
2. **Azan not playing**
   - Check audio device
   - Verify file paths
   - Test volume settings
3. **External monitor not working**
   - Check HDMI connection
   - Verify display settings
   - Try different resolution
4. **Backup won't restore**
   - Verify security answers
   - Check file integrity
   - Try different backup file

Support Information

- **Developer**: [Your Name]
- **Contact**: [Your Email]
- **Source Code**: [GitHub Repository]
- **License**: MIT Open Source

📈 FUTURE ENHANCEMENTS

Phase 2 Features (Optional)

- **Multi-language**: Arabic, Malay support
- **Cloud Sync**: Optional Google Drive backup
- **Mobile App**: Android companion app
- **Donation Tracking**: Community fund management
- **Event Calendar**: Full calendar integration
- **SMS Notifications**: Prayer time reminders

Technical Improvements

- **Auto-updater**: Seamless software updates
- **Plugin System**: Third-party integrations
- **API Support**: External system integration
- **Performance Monitoring**: Usage analytics
- **Crash Reporting**: Automated error reporting

💰 COST COMPARISON

Current Solution (Annual)

- **Software License**: 6,000 MYR
- **Technician Visits**: 1,200 MYR
- **Updates/Maintenance**: 800 MYR
- **Total**: 8,000 MYR per year

MasjidSuite (Annual)

- **Software License**: 0 MYR
- **Technician Visits**: 0 MYR
- **Updates/Maintenance**: 0 MYR
- **Total**: 0 MYR per year

Annual Savings: 8,000 MYR per mosque **5-Year Savings**: 40,000 MYR per mosque

---

🎉 PROJECT SUCCESS CRITERIA

Technical Success

- [ ] Application runs 24/7 without crashes
- [ ] Prayer times accurate to within 1 minute
- [ ] Backup/restore works 100% reliably
- [ ] No monthly subscription fees
- [ ] Imam can operate without technical support

Business Success

- [ ] Mosque saves 6,000+ MYR annually
- [ ] Imam confident in system operation
- [ ] Community features actively used
- [ ] Positive feedback from mosque committee
- [ ] Successful deployment without vendor support

Project Completion

- [ ] All features implemented and tested
- [ ] User documentation complete
- [ ] Installation successful at mosque
- [ ] Imam trained on all features
- [ ] Source code delivered with MIT license

🚀QUICK START COMMANDS

```bash
# Initialize project
npm create electron-app@latest masjid-suite -- --template=typescript
cd masjid-suite

# Install dependencies
npm install react react-dom sqlite3 crypto-js
npm install --save-dev @types/react @types/react-dom

# Development
npm run dev

# Build for production
npm run build

# Package for distribution
npm run package
```

END OF SPECIFICATION

This document provides complete implementation details for building MasjidSuite using Cursor AI or any other development tool. All components, interfaces, and requirements are specified in detail to enable autonomous development.