🪐 Software Requirement Specification (SRS)

Project Title: WhatsApp-Based Astrology Automation Software

Technology Stack: Node.js, PostgreSQL, WhatsApp Cloud API, Prokerala Astrology API, Google Geocoding API
Hosting: Render (Backend), GitHub (Repository)
Version: 1.0
Author: Shishir Srivastava

---

1. Introduction

1.1 Purpose

The purpose of this project is to build a **WhatsApp-based automated astrology system** that collects users' birth details, determines their Mahadasha–Antardasha periods using the **Prokerala Astrology API**, and provides personalized **YouTube playlists** based on planetary periods and user problems. The system automates astrology consultation through WhatsApp without any human intervention.

1.2 Scope

The system will:

- Collect birth details via WhatsApp chat.
- Fetch latitude & longitude using **Google Geocoding API**.
- Retrieve Mahadasha–Antardasha information from **Prokerala API**.
- Store all data securely in **PostgreSQL**.

- Send personalized predefined **YouTube playlists** based on astrology and problem type.
- Use a **cron job** to check for dasha changes daily and send new playlists automatically.
- Provide admin flexibility to add/update problem playlists later.

1.3 System Users

- **End Users (Public):** Individuals interacting via WhatsApp.
- **Admin:** Maintains playlist database and monitors user data.

1.4 Goals and Objectives

- To create a **fully automated astrology solution** running through WhatsApp.
- To minimize manual intervention and offer instant personalized responses.
- To maintain all astrology and user data in an organized database.
- To ensure daily monitoring of user's current dasha and timely playlist updates.

---

2. Overall Description

2.1 System Environment

The system will function as a **cloud-based backend** application connected to:

- **WhatsApp Cloud API** for user interaction.
- **Prokerala API** for astrology calculations.
- **Google Geocoding API** for latitude/longitude.

- **PostgreSQL** for persistent data storage.
- **Cron job scheduler** for periodic automated updates.

2.2 System Architecture (Layered)

1. **Presentation Layer:** WhatsApp message interface (user-side).
2. **Application Layer:** Node.js backend managing APIs, logic, and automation.
3. **Data Layer:** PostgreSQL database with structured tables for users, astrology data, playlists, and locations.
4. **Integration Layer:** Prokerala API + Google Geocoding API.

2.3 Constraints

- Only free-tier APIs and hosting platforms (Render, Netlify, GitHub).
- Limited WhatsApp Cloud API message templates (as per Meta policy).
- Internet access required for API communication.

2.4 Assumptions

- Admin will provide predefined YouTube playlists for each Mahadasha/Antardasha and problem type.
- APIs will return valid and accurate responses.
- Users will provide correct date/time/place of birth.

---

3. Functional Requirements

3.1 WhatsApp Interaction Flow

1. User sends “Hi” to the WhatsApp number.
2. Bot replies asking for preferred language (Hindi/English).
3. Bot collects the following details sequentially:
    - Name
    - Date of Birth (DD/MM/YYYY)
    - Time of Birth (HH:MM)
    - Place of Birth (City)
    - Problem Type (dropdown-like options: Career, Marriage, Health, etc.)
4. All responses stored in PostgreSQL.

3.2 Location Handling (Google Geocoding)

- When place of birth is received:
    - Check if it exists in `locations` table.
    - If yes → fetch latitude & longitude from DB.
    - If no → call **Google Geocoding API**, get coordinates, store them in `locations` table for future use.

3.3 Astrology Data Fetch (Prokerala API)

- Using the collected DOB, TOB, and coordinates:

- Call Prokerala API endpoint for **Mahadasha and Antardasha** details.
- Receive:
  - Mahadasha name
  - Antardasha name
  - Antardasha end date
- Store in `astrology_data` table.

3.4 Playlist Generation Logic

- Each Mahadasha + Antardasha combination (≈81) has a predefined playlist.
- Each problem type (≈50) also has a predefined playlist.
- The system will select:
  - Playlist 1 → for Mahadasha combination
  - Playlist 2 → for Problem Type
- Both playlist URLs + personalized message will be sent to the user.

3.5 Daily Cron Job Automation

- A **node-cron job** runs daily at midnight.
- It checks if the `antardasha_end_date` has passed for any user.
- If yes → fetch new Mahadasha/Antardasha data from API and send the new playlists.

3.6 Admin Functionalities

- Manage predefined playlist data (insert/update/delete).
- Add new problem categories.
- Monitor user interaction logs (optional future enhancement).

---

4. Non-Functional Requirements

Requirement	Description
Performance	System should handle up to 1000 concurrent WhatsApp users.
Security	User data stored securely, API keys hidden in `.env` file.
Scalability	Can be deployed on multiple instances using Render.
Maintainability	Clear folder structure, comments, and documentation.
Reliability	Automatic retries for API failures.
Usability	Simple WhatsApp-based conversation flow.

---

5. Database Design

5.1 Tables

🧍 **users**

Column	Type	Description
id	SERIAL	Primary key
name	TEXT	User name
phone	TEXT	WhatsApp number
dob	DATE	Date of birth
tob	TIME	Time of birth
place	TEXT	Birth city

problem_type	TEXT	User problem
language	TEXT	Preferred language
created_at	TIMESTAMP	Record created date

🌎 locations

Column	Type	Description
id	SERIAL	Primary key
place_name	TEXT	City name
latitude	FLOAT	Latitude
longitude	FLOAT	Longitude

🪐 astrology_data

Column	Type	Description
id	SERIAL	Primary key
user_id	INT	FK → users.id
mahadasha	TEXT	Current Mahadasha
antardasha	TEXT	Current Antardasha
antardasha_end_date	DATE	End date
created_at	TIMESTAMP	Record created time

🎵 playlists

Column	Type	Description
id	SERIAL	Primary key
type	TEXT	“dasha” or “problem”

name	TEXT	Dasha or Problem name
youtube_link	TEXT	Playlist URL

---

6. API Integration Details

6.1 WhatsApp Cloud API

- Used for message automation and user input collection.
- Endpoints:
  - `/webhook` – to receive user messages.
  - `/send-message` – to send messages to users.

6.2 Google Geocoding API

- Endpoint: `https://maps.googleapis.com/maps/api/geocode/json`
- Input: `place_name`
- Output: `latitude`, `longitude`

6.3 Prokerala Astrology API

- Endpoint example:

  `https://api.prokerala.com/v2/astrology/dasha?ayanamsa=1&datetime=YYYY-MM-DDTHH:MM:SS&coordinates=LAT,LONG`

- Output: JSON containing Mahadasha, Antardasha, End Date.

---

7. Process Flow (End-to-End)

```
User (WhatsApp)
      ↓
WhatsApp Cloud API
      ↓
Node.js Backend
      ↓
→ Check & Store User Details
→ Call Google Geocode API
→ Call Prokerala API
→ Save Dasha Data in PostgreSQL
→ Fetch Matching Playlists
→ Send WhatsApp Message (with playlists)
      ↓
Daily Cron Job → Auto-check Dasha → Send Updated Playlist
```

---

8. Deployment & Hosting

- **Backend:** Render (free tier Node.js app)
- **Database:** PostgreSQL (Render / Supabase free tier)
- **Version Control:** GitHub private repo
- **Environment Variables:** `.env` file (contains API keys, DB credentials)

---

9. Future Enhancements

- Admin panel for playlist management.
- Android/iOS mobile app using same APIs.
- Web dashboard for analytics and user monitoring.

---

10. Deliverables

✅ Complete Node.js backend source code
✅ PostgreSQL schema (SQL dump)
✅ Integrated WhatsApp, Google, and Prokerala APIs
✅ Cron job automation for dasha change
✅ Render deployment & documentation
✅ GitHub repository with README and setup guide