Technical Design Document (TDD)

# 1. Purpose

* The purpose of this document is to outline the design specifications for new features or systems, ensuring alignment across the development team.

# 2. Introduction

* Title: **Payment Notifier Plugin**
* Version: **1.0.0**
* Author: **Rohan Batra**
* Date: **25 June 2025**

# 3. Objectives

* **Objective 1: Enable Real-Time Audio Messaging**
* Allow users to send and receive real-time, voice-based messages using Firebase Cloud Messaging (FCM) and Text-to-Speech (TTS) technology.
* **Objective 2: Ensure Reliable Cross-Device Communication**
* Guarantee message delivery between different devices using unique FCM tokens for sender and receiver.
* **Objective 3: Automate Message-to-Audio Conversion**
* Automatically convert text messages to audio on the server side using TTS services based on the selected language.
* **Objective 4: Manage Audio File Lifecycle Efficiently**
* Download, play, and delete TTS audio files on the receiver's device to optimize storage and performance.
* **Objective 6: Provide Clear and Accessible Notifications**
* Display system notifications with message metadata and playback support for accessibility and user awareness
* **Objective 7: Support Multilingual Messaging**
* Generate TTS audio based on user-specified language settings for broader user inclusivity.

# 4. Architecture Diagram

A diagram of a system

AI-generated content may be incorrect.

## 6. Design Details

* **FCM Token Manager:**
* ***Description***: Handles the initialization of Firebase Cloud Messaging (FCM) and generates unique tokens for sender and receiver devices.
* ***Responsibilities***:
* Initialize FCM on device startup
* Initialize FCM on device startup
* ***Dependencies:***
* Firebase SDK
* Internet Connectivity
* **Message Composer:**
* ***Description:*** Gathers user input and compiles all data needed to send a message.
* ***Responsibilities:***
* Collect sender name, token, message content, receiver details, date, amount, and language.
* Structure data in JSON payload
* Trigger HTTP post request to backend (/send)
* ***Dependencies:***
* Input Data
* Network Access
* **Backend Server Processor:**
* ***Description:*** Central server that receives the message data and process it using Firebase Admin SDK, language translation and Text To Speech(TTS) generation.
* ***Responsibilities:***
* Authenticate with Firebase Admin SDK
* Parse message content and language text from JSON payload
* Translate message from English to required language
* Generate TTS using translated message
* Save audio file to storage
* Send a push notification to receiver with the title,body,audio url
* ***Dependencies:***
* Firebase Admin SDK
* Bhashini Text Translation API
* Bhashini Text To Seech API
* Network Access
* **Receivers Plugin:**
* ***Description:*** Listen for upcoming FCM payloads and downloads the audio ,plays it and manages cleanup
* ***Responsibilities:***
* Receives push notification from FCM
* Parse payload
* Download audio file
* Play audio and display notification
* Delete local audio file after playback
* ***Dependencies:***
* Firebase Messaging Listener
* Storage access permissions
* Audio player Component

# 7. API Specifications

* **Endpoint: POST /send**
* ***Purpose:*** Accepts message data from the sender’s plugin, processes it (translation+ TTS), and sends a push notification to the receivers via FCM
* ***Request:***
* **Method:** POST
* **Content-Type:** application/json
* **Body:** fromName, fromToken, message, send\_lang, date, toName, toToken, amount
* **Response:**
* **Succes:** 200 {error:false,messageid:xxxxx}
* **Failure:** 500 {error true,messageid:null}
* **Endpoint: GET /audio**
* ***Purpose:*** Accepts the audio filename from the receiver’s plugin and sends the audio file as an attachment
* ***Request:***
* **Method:** GET
* **Content-Type:** text/html
* **Body:** filename
* **Response:**
* **Succes:** 200 <audiofile>.wav
* **Failure:** 404 Audio file not found

# 8. Implementation Plan

* Step 1: **Firebase Setup**
* Register app in Firebase Console.
* Enable FCM and generate service account credentials.
* Step 2: **Token Generation**
* Implement FCM token generation.
* Save sender/receiver tokens securely.
* Step 3: **Message Composer UI**
* Build sender-side UI for sending message.
* Validate inputs before sending.
* Step 4: **API Development**
* Create /send POST, /audio GET endpoint on backend.
* Parse and validate the inputs.
* Step 5: **Translation and Text To Speech integration**
* Implement Bhashini Translation API to translate message to target language.
* Implement Bhashini TTS API for generating audio from translated message.
* Save audio file and generate url.
* Step 6: **Push Notifications**
* Sends FCM notification to receiver’s token with payload.
* Step 7: **Receiver’s Side**
* Implement payload listener.
* Download audio from url in payload.
* Display notification and play audio.
* Delete downloaded audio after playback.
* Ensure notification persists until dismissed.