# 🎸 Jammr – Product Requirements Document (PRD)

**Project Title:** Jammr
**Version:** 1.0 (MVP)
**Authors:** Gabriel Khouri, Benjamin Lerner, Eitan Ben Shitrit
**Date:** October 2025
**Course:** EECS 497 – Major Design Project

---

## 1. Overview

**Jammr** is a **mobile-first networking platform for musicians** to connect, collaborate, and form bands. It offers a lightweight, intuitive interface that emphasizes intentional, frictionless discovery.

Unlike dating-style swipe apps, Jammr uses a **dynamic list view** filtered by instruments, genre, skill level, and distance. Once users identify someone compatible, they can send a **"Request to Chat."** Upon acceptance, both users can communicate within the app to plan jam sessions or collaborations.

**Goal:** Deliver a polished MVP that connects musicians quickly and safely, validated through user testing by Dec 8, 2025.

# 🎸 Jammr – Product Requirements Document (PRD)

**Project Title:** Jammr
**Version:** 1.0 (MVP)
**Authors:** Gabriel Khouri, Benjamin Lerner, Eitan Ben Shitrit
**Date:** October 2025
**Course:** EECS 497 – Major Design Project

---

## 2. Data Flow Diagram

[User]
  ↓ (create account)
[Firebase Auth]
  ↓ (generate UID)
[Firestore: Users collection]
  ↓ (read profiles / apply filters)
[Client: React Native App]
  ↓ (request to chat)
[Firestore: Matches collection]
  ↓ (on acceptance)
[Firestore: Chats collection]

## 3. Database Schema (Firestore)

| Field | Type | Description |
|---|---|---|
| user_id | string (UID) | Unique Firebase Auth user ID |
| name | string | Display name |
| instrument | string | e.g., Guitar, Drums |
| genres | array[string] | e.g., ["Rock", "Blues"] |
| skill_level | string | Beginner / Intermediate / Advanced |
| bio | string | Short self-description |
| location | geopoint | User’s approximate location |
| audio_clips | array[string] | Firebase Storage URLs |
| image_url | string | Profile image |
| visibility | boolean | Whether the user’s profile is visible |
| created_at | timestamp | Auto-generated |

| Field | Type | Description |
| --- | --- | --- |
| match_id | string | Unique match ID |
| requester_id | string | UID of sender |
| receiver_id | string | UID of receiver |
| status | enum("pending", "accepted", "declined") | |
| created_at | timestamp | Date created |

| Field | Type | Description |
| --- | --- | --- |
| chat_id | string | Unique ID |
| | | Associated match |
| | | { sender_id, text, timestamp } |

# 4. API Endpoints (via Firebase Cloud Functions)

| Endpoint | Method | Description |
| --- | --- | --- |
| `/createUserProfile` | POST | Create or update user profile |
| `/getUserProfiles` | GET | Fetch list of nearby users (filtered) |
| `/sendMatchRequest` | POST | Initiate match request |
| `/respondMatchRequest` | POST | Accept or decline request |
| `/getMatches` | GET | Retrieve matches for logged-in user |
| `/sendMessage` | POST | Push message to chat thread |

| `/getMessages` | GET | Retrieve message history for a chat |

| `/uploadMedia` | POST | Upload user audio/video to Firebase Storage |

| `/sendNotification` | POST | Send push notification via Firebase Messaging |

# 5. Matching Algorithm (Naïve MVP)

1. Retrieve all visible users within a configurable distance (default: 25 miles).
2. Filter users matching:
    - at least one shared genre
    - compatible skill levels (±1 tier difference)
    - not previously matched or blocked
3. Rank users by distance, then by number of shared genres.
4. Return top N (default: 25) profiles per query.

---

# 6. UI/UX Wireframe Overview

Key Screens

1. Login / Register
    - Email/password or Google sign-in
2. Profile Setup
    - Instrument dropdown, genre multiselect, bio, photo/audio upload
3. Discovery Feed

    - Scrollable list view with profile cards and filters
4. Profile View
    - Details + “Request to Chat” button
5. Chat Interface
    - Message bubbles, media preview, timestamp
6. Notifications
    - Push alerts for match requests & messages

Design Language:

- Clean, minimalist aesthetic
- Color palette: Deep purple + off-white
- Typography: Sans-serif (Inter or Poppins)
- Accessibility: ≥ WCAG AA compliance