Aşağıdaki plan Flutter istemcisi hazır kabul edilerek yalnızca **Python tarafını** (FastAPI + Celery + PostgreSQL + MongoDB) kapsar.  
Sıra, Epic-Story-Task kırılımına ve dersin “olmazsa olmaz / bonus” hükümlerine göre (Jira-Kanban + Git-Flow, CI/CD, NoSQL vb.) düzenlenmiştir.

**0 | Başlangıç ve Altyapı**

| **Adım** | **İçerik** | **İlgili Epic-Task** | **Not / Bonus** |
| --- | --- | --- | --- |
| 0.1 | Git repo aç, **Git Flow** dal yapısını (main, develop, feature/\*, release/\*, hotfix/\*) kur. | Bonus (Git Flow) | Branch-issue eşlemesi Jira’yla otomatik açılacak. |
| 0.2 | cookiecutter veya manuel klasör iskeleti oluştur.music\_recognition/├ api/ (FastAPI)├ core/ (domain-service)├ db/ (SQL & NoSQL)├ worker/ (Celery)├ tests/└ deploy/ (Docker, compose) | Mimari özeti | Katmanlı modüler monolit. |
| 0.3 | GitHub Actions yaml: lint-test-build-push, Render/Vercel deploy. | Bonus CI/CD (+15 puan) |  |

**1 | Veri Katmanı – Epic 1 Story 1.3** ✅

Önce şemalar → sonra üst katmanlar

1. **Task 1.3.1-1.3.3**:
   * db/sql/models.py - SQLAlchemy → Song, Artist, Album.
   * db/nosql/collections.py - Motor/MongoEngine → Fingerprint.
   * Alembic migration + Mongo indexleri.
2. **Task 1.3.4-1.3.5**: Repository katmanı (core/repository/{song,fingerprint}.py) – CRUD + bulk-insert.

✅ Bonus “NoSQL” şartı böylece karşılanır.

**2 | Ses Ön-İşleme – Epic 1 Story 1.1** ✅

| **Sıra** | **Görev** | **Sınıf-Fonksiyon** |
| --- | --- | --- |
| 2.1 | **1.1.1** Mikrofondan gelen ham baytı FastAPI UploadFile olarak al, temp dosyaya kaydet. | core/io/recording.py::save\_temp() |
| 2.2 | **1.1.2** RMS-tabanlı normalizasyon. | core/preprocess/audio.py::normalize() |
| 2.3 | **1.1.3** Örnekleme frekansı çevirimi (sox, librosa.resample). | core/preprocess/audio.py::resample() |

**3 | Akustik Parmak İzi – Epic 1 Story 1.2** ✅

1. **1.2.1** Araştırma Spike → ayrı feature/research-fingerprint dalı.
2. **1.2.2** Seçilen algoritmayı (ör. dejavu) sarmalayan core/fingerprint/extractor.py.
3. **1.2.3** Celery görevi worker/tasks.py::store\_fingerprint ⇒ MongoDB.

**4 | Tanı & Eşleştirme – Epic 1 Story 1.4** ✅

| **Sıra** | **Task** | **Modül/Fonksiyon** |
| --- | --- | --- |
| 4.1 | **1.4.1** Ön-işlenmiş kayıttan parmak izi üret. | core/fingerprint/extractor.py |
| 4.2 | **1.4.2** Mongo sorgusu ile benzer hash’ler çek. | core/fingerprint/matcher.py |
| 4.3 | **1.4.3** Eşik hassasiyeti ayarlanabilir Strategy pattern. | core/fingerprint/threshold.py (Bonus: Design Pattern) |
| 4.4 | **1.4.4** Şarkı bilgisi + olasılık JSON döndür. | api/v1/recognition.py endpoint |

**5 | Gürültü Azaltma – Epic 2** ✅

Bu aşama paralel bir dalda başlayabilir (takım üyesi #2).

1. **2.1.x** Literature & model incelemesi (issue label “research”).
2. **2.2.3** Seçilen yöntem (deepfilternet) ⇒ core/noise/reducer.py.
3. Celery görevi zincire eklenir → worker/tasks.py::reduce\_noise ses kaydedilmeden hemen sonra çalışır.

**6 | Öneri & Playlist – Epic 3**

| **Task** | **İçerik** | **Sınıf** |
| --- | --- | --- |
| 3.1.1-3.1.2 | Feature mühendisliği (tempo, genre, embedding). | core/reco/features.py |
| 3.1.3 | K-NN veya cosine benzerliği. | core/reco/engine.py |
| 3.2.1 | PlaylistBuilder service. | core/reco/builder.py |
| 3.2.2 | api/v1/playlists.py endpoint. |  |

**7 | API Dış Yüz – Epic 5**

1. **5.1.x** OpenAPI taslağı → FastAPI routerlar.
2. **5.2.x** Endpoint’ler:
   * POST /recognize
   * GET /songs/{id}
   * GET /recommend/{song\_id}
   * POST /playlist/auto
   * GET /health

Swagger otomatik, mobil istemci entegrasyonu yapılır.

**8 | Test, Kalite ve Dağıtım**

* **Unit Test**: pytest, her service-repository katmanı için.
* **Integration Test**: FastAPI TestClient + docker-compose CI job’ında.
* **Coverage ≥ 80 %** (Jira Definition-of-Done).
* **Dockerfile** + **docker-compose** (Postgres, Mongo, Redis, Celery worker, web).
* GitHub Actions pipeline zaten 0.3’te kuruldu – push → test → build → deploy.

**9 | Bonus Puan Kancaları**

| **Özellik** | **Nerede / Nasıl eklenecek?** |
| --- | --- |
| CI/CD (+15 puan) | deploy/github-ci.yml → Render auto-deploy. |
| Design Pattern zenginliği | Strategy (threshold), Repository, Factory (noise reducer, reco engine). |
| Harici API | Spotify Preview veya Last.fm artist info → core/external/spotify.py, gösterim mobilde. |
| NoSQL Kullanımı | Fingerprint koleksiyonu (Mongo) zaten var. |
| Git Flow | Tüm iş dalları feature/<epic>-<task> konvansiyonu. |

**Önerilen Sprint Dizilimi (3 kişilik ekip, 2-haftalık sprintler)**

| **Sprint** | **Hedef** |
| --- | --- |
| S-0 | Altyapı (0.x) + DB şemaları (1.3) + CI/CD çalışır. |
| S-1 | Ön-işleme (1.1) + Parmak izi çıkarma (1.2) bitir, testler. |
| S-2 | Eşleştirme (1.4) + İlk endpoint /recognize, mobil demo. |
| S-3 | Gürültü azaltma (Epic 2) & refactor, Design Patternlerin yerleştirilmesi. |
| S-4 | Öneri-playlist (Epic 3) + dış API entegrasyonu. |
| S-5 | Hardening: load-test, monitoring, dokümantasyon, final gösterim. |

**Özet Akış**

1. **Veriyi modelle →** SQL+Mongo.
2. **Ses ↑ pipeline →** normalize → resample → noise\_reduce (opsiyon) → fingerprint.
3. **Celery** ağır işleri alır, sonuç Postgres/Mongo’dan okunur.
4. **FastAPI** tek giriş noktasıdır; mobil uygulama bu uçlara bağlanır.

Bu plan modüllerin net sınırlarını, sprint sırasını ve bonus-puan kriterlerini birlikte karşılar; Jira-Kanban üzerinde her **Task** doğrudan karşılık gelen **feature branch** ile eşleşir, CI/CD sayesinde her PR ön-prod’a otomatik taşınır.