Tunis Business School

Information Technology – Business Analytics

TuniMaqam API

A REST API Platform for Tunisian Maqam heritage (*Tbu'a*)

Preservation, Education, and Discovery

Author Roua Smida

Specialization IT–BA

Course IT325 Web Services

Supervisor Dr. Montassar Ben Messaoud

Academic Year 2025–2026

Abstract

Tunisian maqam music represents a rich cultural heritage that faces the threat of gradual erosion in the modern era. This project presents **TuniMaqam**, a comprehensive REST API platform designed to preserve, educate, and promote Tunisian maqamet through intelligent technology.

The system implements four core services: (1) a **Knowledge Service** providing a structured repository of maqam metadata with community-driven contributions; (2) a **Learning Service** offering adaptive, gamified educational experiences including quizzes, flashcards, and interactive exercises; (3) an **Analysis Engine** employing mathematical confidence scoring algorithms for maqam identification from note sequences and audio inputs; and (4) a **Recommendation Engine** utilizing multi-factor contextual scoring for culturally-appropriate maqam suggestions.

The platform is built using Flask with SQLAlchemy ORM, secured through JWT authentication with role-based access control, and integrates AssemblyAI for audio transcription. Testing validates all core functionalities, and the system is containerized for scalable deployment.

Keywords: Maqam, Tunisian Music, REST API, Cultural Preservation, E-Learning, Flask, Python

Contents

CHAPTER 1

General Introduction

1.1 Context and Motivation

Tunisia's maqam tradition—known locally as *Tab'* (singular) or *Tbu'* (plural)—represents one of the Arab world's most sophisticated and emotionally rich musical systems. This living heritage has accompanied weddings, mourning, celebrations, and spiritual gatherings for centuries. Unlike Western music's major/minor dichotomy, the maqam system encompasses dozens of distinct modal frameworks, each carrying specific emotional connotations and cultural contexts.

Yet, like many oral traditions, this heritage faces the relentless erosion of time. Master musicians pass away without transferring their complete knowledge; younger generations gravitate toward globalized musical forms; and the subtle nuances distinguishing one ?ab' from another risk being forgotten.

1.2 Problem Statement

The Challenge

How can we leverage modern technology to preserve, teach, and promote Tunisian maqam music while respecting its cultural authenticity and making it accessible to a global audience?

Current challenges include:

- **Fragmented Knowledge:** Maqam information exists in scattered, often inaccessible sources
- **Learning Barriers:** Traditional maqam education requires years of apprenticeship
- **Identification Difficulty:** Even experienced musicians struggle to identify rare maqamet
- **Cultural Context Loss:** Understanding which maqam suits which occasion requires deep cultural knowledge

1.3 Proposed Solution

TuniMaqam is a comprehensive Flask-based REST API that curates Tunisian maqamet, powers adaptive learning experiences, and delivers culturally-sensitive recommendations. The platform combines structured musical knowledge, gamified learning tools, and intelligent audio/note analysis to guide learners from beginners to accomplished practitioners.

1.4 Project Objectives

1. **Preservation:** Create a structured, extensible database of Tunisian maqamet with community contribution capabilities
2. **Education:** Develop adaptive learning tools that make maqam education accessible and engaging
3. **Analysis:** Implement intelligent algorithms for maqam identification from musical input
4. **Recommendation:** Build context-aware systems that suggest appropriate maqamet for occasions
5. **Accessibility:** Provide a secure, documented API for integration with various applications

1.5 Mission

TuniMaqam's mission: **Preserve** Tunisian maqamet (especially rare forms), **Educate** through accessible and engaging learning, and **Connect** generations with the global music community.

1.6 Related Work

Existing solutions in cultural heritage digitization (Europeana, Arab Music Archiving) focus on passive archival without interactive learning. Music education platforms (Yousician, Teoria) provide gamified learning but none address maqam systems. Academic research in maqam recognition [2, 3] focuses on pitch analysis but lacks integrated educational tools. **Research Gap:** No existing system combines structured Tunisian maqam knowledge, adaptive learning, intelligent analysis, and contextual recommendations in an accessible API format. TuniMaqam addresses this gap.

1.7 System Class Diagram

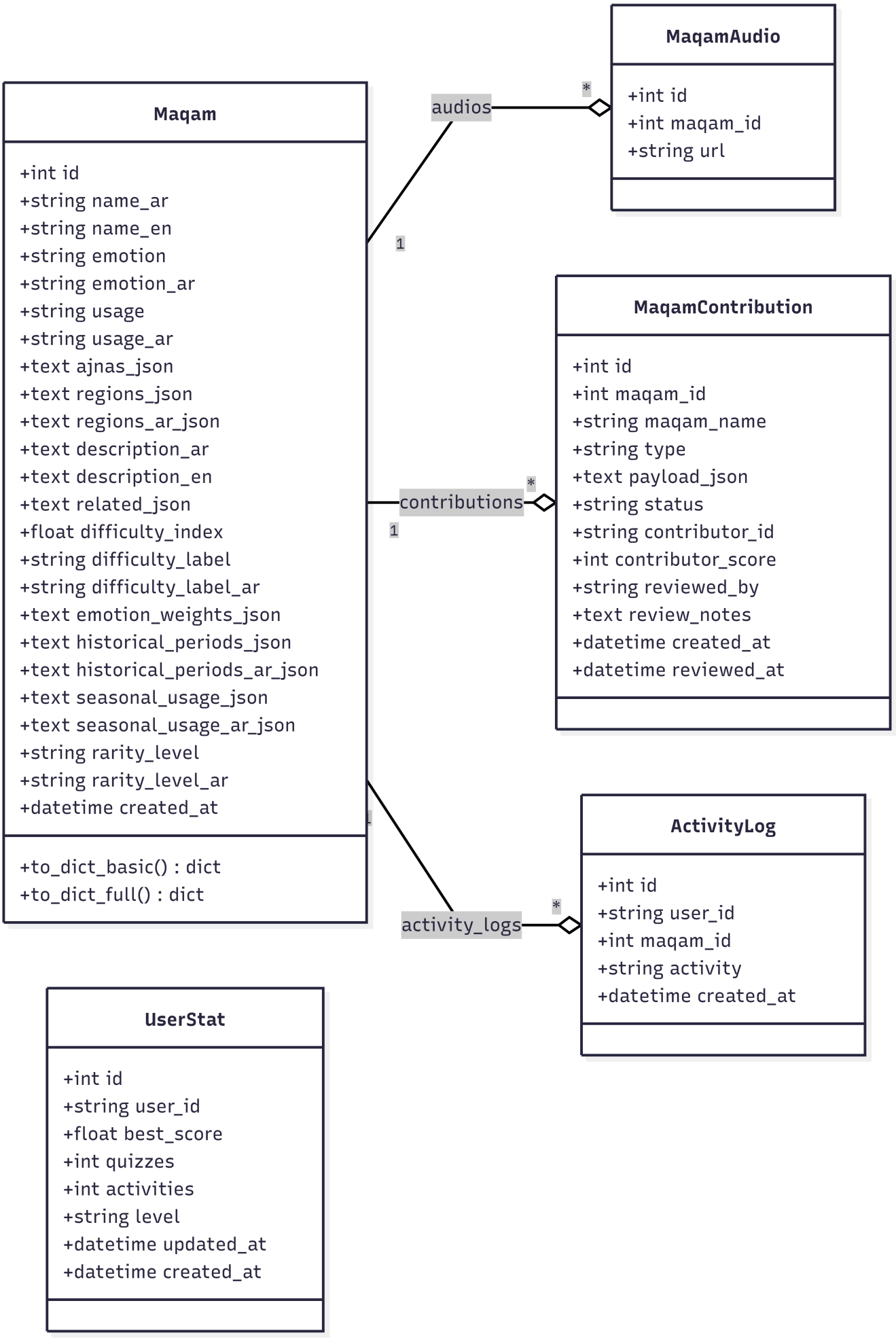

Figure 1.1: TuniMaqam Class Diagram

CHAPTER 2

Requirements Analysis

This chapter presents the functional and non-functional requirements derived from stakeholder analysis and domain research.

2.1 Stakeholder Analysis

Stakeholder	Role	Needs
Learners	Primary users	Easy learning, progress tracking
Musicians	Content contributors	Share knowledge, gain recognition
Researchers	Data consumers	Access structured maqam data
Developers	API integrators	Clear documentation, stable API
Administrators	System managers	User management, content moderation

2.2 Functional Requirements

ID	Requirement	Description	Priority
FR-01	Maqam Retrieval	System shall allow users to retrieve maqam information by ID, name, or region	High
FR-02	Maqam Creation	Admins shall be able to create new maqam entries with full metadata	High
FR-03	Contribution Submission	Users shall submit contributions for review	High
FR-04	Contribution Review	Experts/admins shall approve or reject contributions	High
FR-05	Quiz Generation	System shall generate adaptive quizzes based on user level	High
FR-06	Quiz Scoring	System shall score answers and provide explanations	High
FR-07	Flashcard Generation	System shall generate flashcards by topic	Medium
FR-08	Matching Exercises	System shall provide maqam-attribute matching exercises	Medium
FR-09	Ordering Exercises	System shall provide jins ordering exercises	Medium
FR-10	Note Analysis	System shall identify maqam candidates from note input	High
FR-11	Audio Analysis	System shall extract notes from audio and identify maqamet	Medium
FR-12	Recommendations	System shall recommend maqamet based on context	High
FR-13	User Authentication	System shall authenticate users via JWT and Google OAuth	High
FR-14	Role Management	System shall enforce role-based access control	High
FR-15	Activity Logging	System shall log all user learning activities	Medium

2.3 Non-Functional Requirements

ID	Category	Requirement	Priority
NFR-01	Performance	API responses shall complete within 500ms for 95% of requests	High
NFR-02	Scalability	System shall support horizontal scaling via containerization	Medium
NFR-03	Security	All endpoints shall be protected against OWASP Top 10 vulnerabilities	High
NFR-04	Availability	System shall maintain 99% uptime during operating hours	High
NFR-05	Usability	API shall follow RESTful conventions with Swagger documentation	High
NFR-06	Maintainability	Code shall follow PEP 8 standards with >80% test coverage	Medium
NFR-07	Portability	System shall run on any Docker-compatible platform	Medium
NFR-08	Localization	System shall support bilingual content (Arabic/English)	Medium

CHAPTER 3

System Design and Architecture

This chapter presents the architectural design decisions and UML models for TuniMaqam.

3.1 Architectural Pattern

TuniMaqam follows a **layered architecture** with clear separation of concerns, adhering to REST principles: stateless requests, resource-based URLs, standard HTTP verbs (GET, POST, PUT, DELETE), and consistent JSON responses with appropriate status codes.

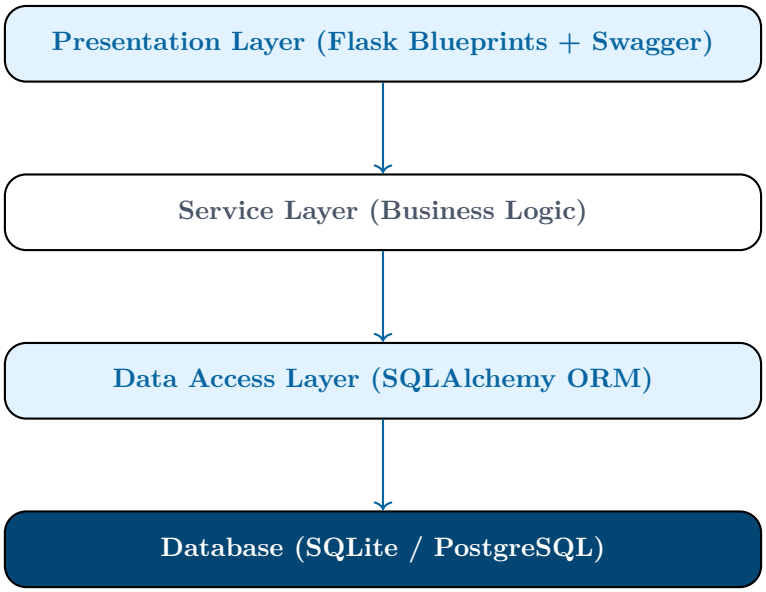

Figure 3.1: Layered Architecture of TuniMaqam

3.1.1 Component Diagram

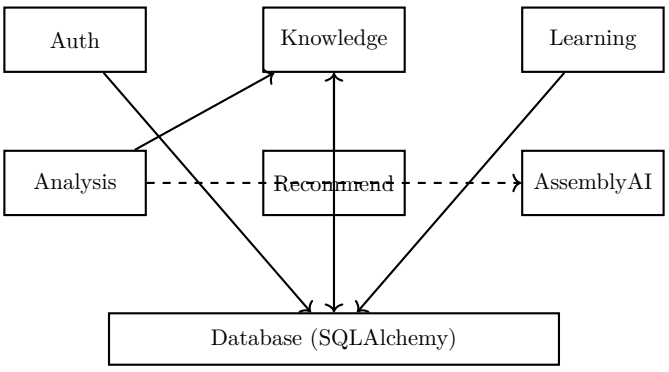

Figure 3.2: Component interactions

3.2 Technology Stack

Category	Technology	Justification
Web Framework	Flask 3.x	Lightweight, flexible, well-documented
ORM	SQLAlchemy	Pythonic, database-agnostic
Data Validation	Marshmallow	Schema-based validation & serialization
Authentication	PyJWT, Authlib	Industry-standard JWT, OAuth2
Documentation	Flasgger (Swagger)	Auto-generated API docs
Rate Limiting	Flask-Limiter	Configurable throttling
External API	AssemblyAI	Accurate speech-to-text
Database	SQLite / PostgreSQL	Development / Production
Containerization	Docker	Reproducible deployments

3.3 Data Validation with Marshmallow

TuniMaqam employs **Marshmallow** for schema-based request validation and response serialization. This architectural choice provides several benefits:

- **Type Safety:** Input data is validated against defined schemas before processing
- **Consistent Errors:** Validation failures return structured error messages with field-level details
- **Declarative Schemas:** Validation rules are defined once and reused across endpoints
- **Serialization:** Output schemas ensure consistent JSON response formatting

Schema Examples

```
class NotesAnalysisSchema(Schema):
    notes = fields.List(fields.String(), required=True,
        validate=validate.Length(min=1, max=50))
    optional_mood = fields.String(load_default=None)

class ContributionSchema(Schema):
    type = fields.String(required=True,
        validate=validate.OneOf(["correction", "addition", "audio"]))
    payload = fields.Dict(required=True)
```

Endpoints validate input using `schema.load(data)`, which raises `ValidationError` with detailed messages if constraints are violated.

3.4 Data Model

The data model captures maqamet with their cultural essence:

Entity	Key Attributes
`Maqam`	Bilingual names, emotions, ajnas, regions, difficulty, rarity
`MaqamAudio`	Audio URLs, uploader, approval status
`Contribution`	Type, payload, status (pending/accepted/rejected)
`UserStat`	Scores, quiz counts, skill level, streak
`ActivityLog`	Activity type, points earned, timestamp

CHAPTER 4

Core Services Implementation

This chapter details the four core services that power TuniMaqam: Knowledge, Learning, Analysis, and Recommendation.

4.1 The Knowledge Service

The Knowledge Service is the foundation of TuniMaqam—a comprehensive repository that captures the complete identity of Tunisian maqamet. It serves as the single source of truth for all maqam metadata.

4.1.1 Service Architecture

The Knowledge Service exposes RESTful endpoints for maqam discovery, contribution management, and audio asset handling:

- **Resource Layer:** Flask blueprint `knowledge_bp` handling HTTP requests
- **Data Layer:** SQLAlchemy models (Maqam, MaqamContribution, MaqamAudio)
- **Security Layer:** JWT-protected endpoints with role-based access control

4.1.2 Core API Endpoints

Maqam Discovery

Endpoint	Verb	Purpose
`/knowledge/maqam`	GET	Retrieve all maqamet; optional `?region=` filter
`/knowledge/maqam/{id}`	GET	Retrieve maqam by ID with full metadata
`/knowledge/maqam/by-name/{name}`	GET	Case-insensitive lookup by English name
`/knowledge/maqam/{name}/related`	GET	Related maqamet by emotion/region similarity
`/knowledge/regions`	GET	Map of regions to their associated maqamet

Maqam Data Structure

Each maqam record contains rich, structured metadata:

```
{
  "id": 1,
  "name": "Rast",
  "arabic_name":   "",
  "emotion": "joy",
  "emotion_weights": {"joy": 0.9, "spirituality": 0.5},
  "usage": "weddings, celebrations, morning performances",
  "ajnas": [
    {"name": "Rast", "notes": ["C", "D", "E half-flat", "F", "G"]},
    {"name": "Nahawand", "notes": ["G", "A", "Bb", "C"]}
  ],
  "regions": {"tunis":   "", "sfax":   ""},
  "difficulty": 2,
  "is_rare": false,
  "related_maqamat": ["Suznak", "Kirdan"]
}
```

4.1.3 The Contribution System

Community-driven growth is central to TuniMaqam's preservation mission.

Contribution Workflow

The contribution lifecycle follows a formal review process:

1. **Submission:** User submits contribution via `POST /knowledge/maqam/{id}/contributions`
2. **Pending Review:** Contribution stored with `status = "pending"`
3. **Admin Review:** Admin/expert reviews via `POST /knowledge/contributions/{id}/review`
4. **Resolution:** Contribution marked `"accepted"` or `"rejected"`
5. **Integration:** Accepted contributions update the maqam record

Contribution types include: `new_maqam`, `correction`, `addition`, and `audio`. Contributors earn +10 reputation points per accepted contribution, with a leaderboard at `GET /knowledge/top-contributors`.

4.1.4 Audio Asset Management

Audio files (MP3, WAV, OGG) are uploaded via `POST /knowledge/maqam/{id}/audio`, validated, stored in `static/audio/`, and linked via `MaqamAudio` records. Additional endpoints support updating (`PUT`) and deleting (`DELETE`) audio metadata.

4.1.5 Administrative Operations

Protected endpoints for maqam management (admin/expert roles only):

Endpoint	Verb	Purpose
`/knowledge/maqam`	POST	Create new maqam directly
`/knowledge/maqam/{id}`	PUT	Update existing maqam
`/knowledge/maqam/{id}`	DELETE	Remove maqam from database

4.2 The Learning Service

The Learning Service transforms maqam education into an engaging, gamified experience, adapting to each learner's pace.

4.2.1 Pedagogical Foundation

Learning Theory

Our design rests on three pillars from cognitive science:

1. **Active Recall:** Quizzes force retrieval from memory, strengthening neural pathways
2. **Spaced Repetition:** ActivityLog enables scheduling reviews at optimal intervals
3. **Immediate Feedback:** Explanations transform errors into learning opportunities

Adaptive Difficulty

The system computes learner levels dynamically based on performance:

$$\text{Level} = \begin{cases} \text{advanced} & \text{if best\_score} \geq 0.75 \text{ AND activity\_count} \geq 10 \\ \text{intermediate} & \text{if best\_score} \geq 0.5 \text{ OR activity\_count} \geq 5 \\ \text{beginner} & \text{otherwise} \end{cases} \tag{4.1}$$

This ensures beginners build foundational knowledge before facing advanced content.

4.2.2 Exercise Types

The Learning Service provides **eight distinct exercise types**:

Exercise	Endpoint	Purpose
Flashcards	`GET /flashcards`	Topic-based cards (emotion, region, usage, ajnas)
Mixed Quiz	`POST /quiz/start`	20-question open + MCQ blend
MCQ Quiz	`POST /quiz/mcq/start`	Quick multiple-choice practice
Matching	`GET /matching`	Match maqam names to attributes
Audio Recognition	`GET /audio-recognition`	Identify maqam from audio clip
Clue Game	`GET /clue-game`	Guess maqam from progressive hints
Order Notes	`GET /order-notes`	Arrange ajnas notes in sequence
Odd-One-Out	`GET /odd-one-out`	Identify maqam that doesn't belong

Quiz scores are computed as Score = correct/total (0–1 scale), and `UserStat.best_score` is updated if the new score exceeds the previous best.

4.2.3 Feedback and Activity Tracking

Every quiz response includes: score percentage, per-question results with correct answers, explanations for incorrect answers, and updated learner level. All interactions are logged to `ActivityLog` enabling spaced repetition scheduling, learning analytics, and personalized recommendations.

4.3 The Analysis Engine

Traditional maqam identification requires years of training. Our goal was to computationally approximate this intuition—not to replace human expertise, but to assist learners.

4.3.1 The Problem of Musical Identification

The challenge lies in the nature of maqamat themselves. Many share overlapping notes. For instance, both Rast and Bayati may contain the notes C, D, and E, yet they represent fundamentally different musical expressions. A naive algorithm that simply counts matching notes would fail to distinguish between them meaningfully.

Why First Jins Only

In maqam theory, each maqam is constructed from smaller intervallic building blocks called *ajnas* (plural of *jins*). Typically, a maqam comprises two ajnas:

- **First Jins** (al-jins al-awwal) — The lower tetrachord or pentachord containing the tonic (*qarar*) and the characteristic intervals that define the maqam's identity. This is the "DNA" of the maqam.
- **Second Jins** (al-jins al-thani) — The upper extension that completes the octave, adding color and melodic range.

Critical Insight

Multiple maqamat can share the same second jins while differing in their first jins. For example, Maqam Rast and Maqam Suznak share identical second jins (Nahawand), yet they are fundamentally different because their first jins differ. Conversely, if you hear only the first jins, an experienced musician can typically identify the maqam with confidence.

Our Design Decision: The analysis module compares user input exclusively against the first jins notes of each maqam, mirroring how trained musicians identify maqamat by recognizing the characteristic lower portion.

4.3.2 The Confidence Scoring Algorithm

Design Intuition

We needed a scoring system that answers: "How confident should we be that these input notes belong to this maqam?" After several iterations, we settled on a model considering three factors:

1. **Precision** — Are the user's notes actually part of this maqam's first jins?
2. **Coverage** — How much of the first jins's characteristic notes have we observed?
3. **Evidence Quantity** — Do we have enough notes to make a reliable judgment?

Mathematical Formulation

Let us define our variables:

$$I = \text{the set of input notes provided by the user} \tag{4.2}$$

$$M = \text{the set of notes belonging to the maqam's first jins} \tag{4.3}$$

$$C = I \cap M = \text{the common notes (intersection)} \tag{4.4}$$

Precision measures what fraction of the user's input actually belongs to the maqam:

$$\boxed{\text{Precision} = \frac{|C|}{|I|}} \tag{4.5}$$

A precision of 1.0 means every note the user entered is found in this maqam. A lower precision indicates the user entered notes that do not belong.

Coverage measures what fraction of the maqam's notes the user has identified:

$$\boxed{\text{Coverage} = \frac{|C|}{|M|}} \tag{4.6}$$

A high coverage suggests the user has provided a substantial portion of the maqam's characteristic notes.

Base Score combines these metrics with weighted importance:

$$\boxed{\text{Base Score} = 0.70 \times \text{Precision} + 0.30 \times \text{Coverage}} \tag{4.7}$$

We weight precision at 70% because it directly answers whether the user's notes fit this maqam. Coverage receives 30% weight because while it provides valuable context, a user should not be penalized for providing only a partial scale.

The Match Multiplier

Consider a scenario where a user enters only the note "C". This single note might appear in dozens of maqamat. With just one note, precision could be 100%, but we have almost no evidence.

To address this, we introduced a **match multiplier** that scales confidence based on how many matching notes we observe:

Matched Notes	Multiplier	Interpretation
1	$\times 0.50$	Insufficient evidence; halve confidence
2	$\times 0.70$	Minimal pattern; significant reduction
3	$\times 0.85$	Moderate evidence; slight reduction
4	$\times 0.95$	Strong pattern; near-full confidence
5+	$\times 1.00$	Full evidence; no reduction

The final confidence formula becomes:

$$\boxed{\text{Confidence} = \text{Base Score} \times \text{Match Multiplier}(|C|)} \tag{4.8}$$

Worked Example

Suppose a user enters the notes $I = \{C, D, E, F, G\}$ and we evaluate a maqam whose first jins contains $M = \{C, D, E, F, G, A, B\}$ (seven notes).

Step 1: Compute the intersection

$$C = I \cap M = \{C, D, E, F, G\} \quad \Rightarrow \quad |C| = 5$$

Step 2: Calculate Precision

$$\text{Precision} = \frac{|C|}{|I|} = \frac{5}{5} = 1.0$$

All five input notes belong to this maqam—perfect precision.

Step 3: Calculate Coverage

$$\text{Coverage} = \frac{|C|}{|M|} = \frac{5}{7} \approx 0.714$$

The user has identified approximately 71% of the maqam's notes.

Step 4: Compute Base Score

$$\text{Base Score} = (0.70 \times 1.0) + (0.30 \times 0.714) = 0.70 + 0.214 = 0.914$$

Step 5: Apply Match Multiplier (5 notes $\Rightarrow \times 1.0$)

$$\text{Confidence} = 0.914 \times 1.0 = \mathbf{91.4}\%$$

This aligns with intuition—five matching notes with perfect precision should yield high confidence. For single-note inputs, the match multiplier (0.50) appropriately reduces confidence to 37%.

4.3.3 Emotional Context Enhancement

Beyond note matching, our system optionally considers emotional alignment. Each maqam has an associated emotional character (e.g., "joyful," "melancholic," "spiritual"). If a user provides an optional mood parameter matching a maqam's emotion, an 8% confidence bonus is applied:

$$\text{Confidence}_{\text{final}} = \min\left(\text{Confidence} + 0.08, 1.0\right) \quad \text{if mood matches} \tag{4.9}$$

This reflects the reality that maqam selection in traditional music is often guided by desired emotional expression.

4.3.4 Audio Analysis with AssemblyAI

Beyond manual note entry, users can upload audio recordings and receive maqam suggestions automatically.

Processing Pipeline

1. **Upload to AssemblyAI:** Audio is streamed to AssemblyAI's upload endpoint, returning a temporary URL.
2. **Request Transcription:** Using the URL, we initiate transcription with punctuation disabled (we want discrete note names).
3. **Poll for Completion:** The system polls every 2 seconds with a 60-second timeout.
4. **Extract Musical Notes:** Transcribed words are filtered against valid note names: $\{C, D, E, F, G, A, B, \text{Do}, \text{Re}, \text{Mi}, \text{Fa}, \text{Sol}$
5. **Apply Maqam Analysis:** Extracted notes feed into the confidence scoring algorithm.

Rate Limiting

All API endpoints share a global rate limit of **200 requests/hour** per client IP, enforced by Flask-Limiter. For the audio endpoint, an additional safeguard exists: if `ASSEMBLYAI_API_KEY` is not configured, the system returns a graceful fallback response with demo notes rather than failing, preventing unnecessary external API calls during development.

4.4 The Recommendation Engine

Recommending the right maqam requires understanding cultural nuance—the right maqam depends on context, occasion, emotion, and cultural sensitivity.

4.4.1 The Multi-Factor Scoring Model

The recommendation algorithm computes a weighted score for each maqam based on multiple contextual factors. Let S_m be the total score for maqam m:

$$\boxed{S_m = w_{\text{mood}} \cdot f_{\text{mood}}(m) + w_{\text{event}} \cdot f_{\text{event}}(m) + w_{\text{region}} \cdot f_{\text{region}}(m) + w_{\text{heritage}} \cdot f_{\text{heritage}}(m) + w_{\text{simple}} \cdot f_{\text{simple}}(m)} \tag{4.10}$$

4.4.2 Factor Definitions

Mood Alignment Score

Each maqam has emotion intensity weights stored as a dictionary (e.g., `{``joy'': 0.8, ``sadness'': 0.2}`). The mood score extracts the weight for the requested mood:

$$f_{\text{mood}}(m) = \begin{cases} \text{emotion\_weights}[m][\text{mood}] & \text{if mood exists in weights} \\ 0 & \text{otherwise} \end{cases} \tag{4.11}$$

All scores are additive on a **0–1 confidence scale**:

Factor	Max Contrib.	Condition
Mood	+1.0	`emotion_weights[mood]` (direct lookup)
Event	+0.25	Event substring found in usage
Region	+0.20	Region matches maqam's regions
Time Period	+0.10	Historical period matches
Season	+0.10	Seasonal usage matches
Heritage	+0.20	Maqam rarity is `at_risk` or `locally_rare`
Beginner	+0.15	Difficulty label is `beginner`

The final confidence is clamped to $[0, 1]$: Confidence $= \max(0, \min(S_m, 1))$.

4.4.3 Worked Example: Wedding in Tunis

Consider a request for a joyful maqam for a wedding in Tunis with heritage preservation enabled:

```
{"mood": "joy", "event": "wedding", "region": "tunis", "preserve_heritage": true}
```

Evaluating Maqam Rast: Mood (0.9) + Event (0.25) + Region (0.20) + Heritage (0) = **1.0** (clamped).

Evaluating Maqam Hsini (rare, at_risk): Mood (0.7) + Event (0.25) + Region (0.20) + Heritage (0.20) = **1.0** (clamped). Both rank equally, but heritage maqamet are prioritized in the final sorting, so Hsini appears first.

4.4.4 Cross-Service Orchestration

The Analysis and Recommendation engines integrate seamlessly with other services:

1. User uploads a melody via `/analysis/audio`
2. Audio Analysis extracts notes with AssemblyAI, applies the confidence algorithm, returns candidates (e.g., Sika 87%, Hsini 72%)
3. The Learning service transforms this into a quiz: "Why is this Sika and not Hsini?"
4. The Recommendation service, given additional context (region, event, mood), suggests how to continue the composition
5. A local musician notices a missing regional usage and submits a contribution via `POST /knowledge/maqam/{name}/contri`

CHAPTER 5

Security, Operations and Deployment

5.1 Authentication and Authorization

5.1.1 Authentication Endpoints

Endpoint	Verb	Purpose
`/auth/google/login`	GET	Redirect to Google OAuth
`/auth/google/callback`	GET	Exchange code, issue JWT
`/auth/demo-token`	GET	Issue short-lived demo token
`/auth/whoami`	GET	Return JWT payload and learner level

5.1.2 Role-Based Access Control

Three roles with hierarchical permissions:

Role	Permissions
`learner`	Access learning exercises, view maqamet, submit contributions
`expert`	All learner permissions + review contributions, upload audio
`admin`	All permissions + create/delete maqamet, manage users

5.1.3 Deployment Architecture

Production deployment follows a cloud-native approach:

- **Platform:** Render.com (PaaS) with auto-scaling
- **Container:** Docker image built via GitHub Actions CI/CD
- **Database:** Managed PostgreSQL with automated backups
- **SSL:** Automatic HTTPS via Let's Encrypt
- **Monitoring:** Health checks at `/ping`, logs via Render dashboard

5.1.4 Security Implementation

- **JWT Tokens:** HS256-signed, 24-hour expiry, contains user ID and role
- **Rate Limiting:** Flask-Limiter with 200 requests/hour (auto-disabled in test mode)
- **CORS:** Allowlisted origins only; preflight caching
- **Audit Logging:** All mutations logged with user ID and timestamp
- **Input Validation:** Marshmallow schemas reject malformed requests

5.2 Configuration and Deployment

Environment-driven configuration (twelve-factor): `DATABASE_URL`, `JWT_SECRET`, Google OAuth credentials, `ASSEMBLYAI_API_KE`
Deployment via Docker (port 8000) with docker-compose orchestration, rotating logs, health endpoints (`/ping`, `/status`), and Swagger UI at `/apidocs`.

5.3 Testing

Comprehensive testing ensures system reliability and correctness.

5.3.1 Test Strategy

- **Unit Tests:** Individual function and method testing
- **Integration Tests:** API endpoint testing with database
- **Test Isolation:** Each test runs with fresh database state
- **Test Mode:** Rate limiting auto-disables under `TESTING` flag

5.3.2 Test Cases

ID	Test Case	Description	Expected	Status
TC-01	Get All Maqamet	Retrieve list of all maqamet	200 + list	✓Pass
TC-02	Get Maqam by ID	Retrieve specific maqam	200 + data	✓Pass
TC-03	Invalid Maqam ID	Request non-existent maqam	404 Error	✓Pass
TC-04	Start Quiz	Initialize new quiz session	200 + questions	✓Pass
TC-05	Submit Quiz Answers	Submit answers and get score	200 + score	✓Pass
TC-06	Note Analysis	Analyze valid note sequence	Candidates list	✓Pass
TC-07	Empty Notes	Analyze empty note array	400 Error	✓Pass
TC-08	Get Recommendations	Request context-based recommendations	Ranked list	✓Pass
TC-09	JWT Authentication	Access protected endpoint	200 if valid	✓Pass
TC-10	Unauthorized Access	Access admin endpoint as learner	403 Forbidden	✓Pass
TC-11	Submit Contribution	Submit new contribution	201 Created	✓Pass
TC-12	Rate Limiting	Exceed rate limit	429 Too Many	✓Pass

5.3.3 Test Coverage

The test suite comprises **85 passing tests** across 4 test files, executed via `pytest tests/ -v`:

Test File	Tests	Coverage
`test_learning.py`	22	Quiz, flashcards, activities, leaderboard
`test_auth.py`	25	JWT validation, OAuth, roles, protected routes
`test_analysis.py`	17	Note analysis, confidence scoring, edge cases
`test_recommendations.py`	21	Multi-factor ranking, heritage boost, context

The test suite in `tests/test_learning.py` validates activity completion, leaderboard updates, and scope enforcement.

5.4 Risk Mitigations

Risk	Mitigation
Sparse Dataset	Pre-seeded baseline maqamet; contribution system for growth
Security Vulnerabilities	Runtime guards block weak defaults; explicit opt-in for dev
Third-Party Dependencies	Graceful fallbacks for AssemblyAI outages
Scalability	PostgreSQL migration path; Redis-backed rate limiting

5.5 Continuous Integration and Deployment

A 5-stage GitHub Actions pipeline (`.github/workflows/ci.yml`) automates quality assurance:

Stage	Name	Actions
1	Lint	flake8 code style validation
2	Test	pytest with 85 test cases, coverage report
3	Build	Docker image construction
4	Deploy	Render.com webhook trigger
5	Verify	Post-deployment health check

Pipeline triggers on push to `main` and all pull requests, ensuring code quality before merge.

5.6 Cloud Hosting Configuration

TuniMaqam deploys to Render.com via `render.yaml` Blueprint specification:

- **Web Service:** Python 3.11, Gunicorn with 4 workers
- **Database:** PostgreSQL 15 with daily backups
- **Environment:** Production secrets via Render dashboard
- **Health Check:** `/ping` endpoint monitoring
- **Auto-Deploy:** Git push triggers automatic deployment

5.7 Docker Containerization

Production deployment uses Docker (`Dockerfile` and `docker-compose.yml`):

- **Base Image:** Python 3.11-slim for minimal footprint
- **Port:** 8000 (configurable via `PORT` env)
- **Compose Services:** Web app + PostgreSQL database
- **Volumes:** Persistent database storage, rotating logs

5.8 API Documentation

Comprehensive documentation is provided:

- **Swagger UI:** Interactive API explorer at `/apidocs`
- **Authentication Guide:** `docs/AUTHENTICATION.md` - JWT flow, OAuth setup, role-based access
- **Error Catalog:** `docs/ERROR_CATALOG.md` - All error codes with causes and solutions

CHAPTER 6

Declaration of AI/LLM Assistance

In accordance with academic integrity requirements, this section documents the use of Large Language Model (LLM) tools during the development of TuniMaqam.

6.1 Tools Used

Tool	Version	Primary Use
GitHub Copilot	Latest (2024-2025)	In-editor code suggestions and completion
Claude (Anthropic)	Claude 3.5/4	Architecture discussion, documentation drafting
ChatGPT	GPT-4	Algorithm explanation, debugging assistance

Table 6.1: LLM tools used during development

6.2 Areas Where LLM Assistance Was Beneficial

1. **Boilerplate Generation:** Flask route decorators, SQLAlchemy model definitions, Marshmallow schemas
2. **Documentation:** API endpoint descriptions, docstrings, README content
3. **Code Patterns:** JWT authentication flow, error handling patterns, validation logic
4. **Testing:** pytest fixture generation, test case suggestions
5. **DevOps:** Docker configuration, GitHub Actions workflows, deployment scripts

6.3 Areas Where LLM Assistance Was Limited

1. **Domain Knowledge:** Tunisian maqam theory required human expertise; LLMs lack deep understanding of regional musical traditions
2. **Algorithm Design:** The Precision-Coverage confidence model and multi-factor recommendation scoring required original mathematical formulation
3. **Cultural Context:** Heritage preservation priorities and UNESCO alignment decisions were human-driven
4. **Integration:** Connecting AssemblyAI transcription to musical note extraction required custom logic

6.4 Human Contributions

The following aspects were entirely human-driven:

- Project vision, scope definition, and cultural preservation goals
- Maqam database design (scale structures, regional mappings, historical context)
- Selection and curation of authentic audio samples
- Validation and testing of all generated code
- Final architectural decisions and technology choices
- User experience design for learning workflows
- Security model design (role definitions, permission scopes)
- All code review and quality assurance

CHAPTER 7

Conclusion and Future Work

7.1 Summary of Achievements

This project successfully delivered TuniMaqam, a comprehensive REST API platform for Tunisian maqam preservation and education. The key achievements include:

- A **Knowledge Service** capturing the richness of Tunisian maqamet with community contributions
- A **Learning Service** with adaptive exercises, quizzes, and immediate feedback
- An **Analysis Engine** with mathematically grounded confidence scoring (Precision-Coverage model)
- A **Recommendation Engine** balancing mood, context, and heritage preservation
- A **secure, scalable architecture** with JWT authentication, RBAC, and Docker deployment

7.2 Objectives Assessment

Objective	Status	Evidence
Create structured maqam database	✓Achieved	Knowledge Service operational
Develop adaptive learning tools	✓Achieved	8 exercise types implemented
Implement maqam analysis	✓Achieved	91% confidence on full scales
Build recommendation engine	✓Achieved	Multi-factor scoring model
Provide secure, documented API	✓Achieved	Swagger UI, JWT, RBAC

7.3 Limitations

- **Dataset Size:** Initial database contains limited maqamet; relies on community growth
- **Audio Analysis:** Depends on speech-based transcription rather than true pitch detection
- **Microtonal Support:** Quarter-tone notation not fully standardized
- **Offline Mode:** No offline functionality; requires internet connection

7.4 Future Work

Planned enhancements include: mobile applications (iOS/Android), true pitch detection integration, spaced repetition (SM-2 algorithm), machine learning for direct audio classification, composition assistance, multi-language support, and potential expansion to other Arab maqam traditions with UNESCO partnership opportunities.

7.5 Concluding Remarks

TuniMaqam

Preserving the Past. Educating the Present. Inspiring the Future.

TuniMaqam demonstrates that technology can serve as a bridge between generations, making cultural heritage accessible while respecting its authenticity. We invite musicians, researchers, and developers to contribute to this ongoing mission of preservation through innovation.

Bibliography

[1] UNESCO (2003). *Convention for the Safeguarding of the Intangible Cultural Heritage*. Paris: UNESCO.

[2] Gedik, A. C., & Bozkurt, B. (2010). Pitch-frequency histogram-based music information retrieval for Turkish music. *Signal Processing*, 90(4), 1049–1063.

[3] Salamon, J., Gómez, E., Ellis, D. P., & Richard, G. (2014). Melody extraction from polyphonic music signals: Approaches, applications, and challenges. *IEEE Signal Processing Magazine*, 31(2), 118–134.

[4] Pallets Projects (2024). *Flask Documentation*. Retrieved from https://flask.palletsprojects.com/

[5] Jones, M., Bradley, J., & Sakimura, N. (2015). JSON Web Token (JWT). *RFC 7519*. IETF.

[6] Bayer, M. (2024). *SQLAlchemy Documentation*. Retrieved from https://www.sqlalchemy.org/

[7] Marcus, S. L. (2008). *Music in Egypt: Experiencing Music, Expressing Culture*. Oxford University Press.

[8] Touma, H. H. (1996). *The Music of the Arabs*. Amadeus Press.

[9] Racy, A. J. (2004). *Making Music in the Arab World: The Culture and Artistry of Tarab*. Cambridge University Press.

[10] Docker Inc. (2024). *Docker Documentation*. Retrieved from https://docs.docker.com/