Luna

Social Venue Discovery Platform

A Full-Stack iOS Application for the Track 3 Assessment

Built by: Krutin Rathod
Timeline: A 72-Hour Development Sprint

A Complete, Polished User Experience in 72 Hours

What Luna Does

- Provides personalized venue recommendations using a multi-factor scoring algorithm.
- Fosters social coordination by showing friend activity and interest levels.
- Automates booking via an intelligent agent that triggers on social interest.

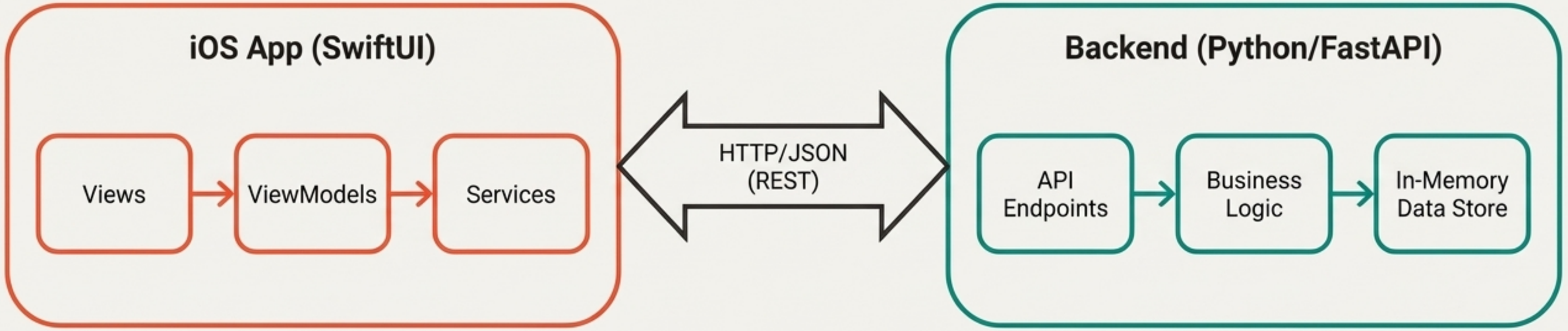
Key Value Proposition

- Transforms venue discovery from a solo task into a collaborative, zero-friction social experience.

Assessment Coverage

- **Track 3:** A seamless integration of a native **SwiftUI frontend (Track 1)** and a performant **Python/ Python/FastAPI** backend **(Track 2)**.

System Architecture: A Clean Separation of Concerns



Frontend: Native iOS Inter Bold

Built with SwiftUI and the Model-View-ViewModel (MVVM) pattern for a declarative and state-driven UI.

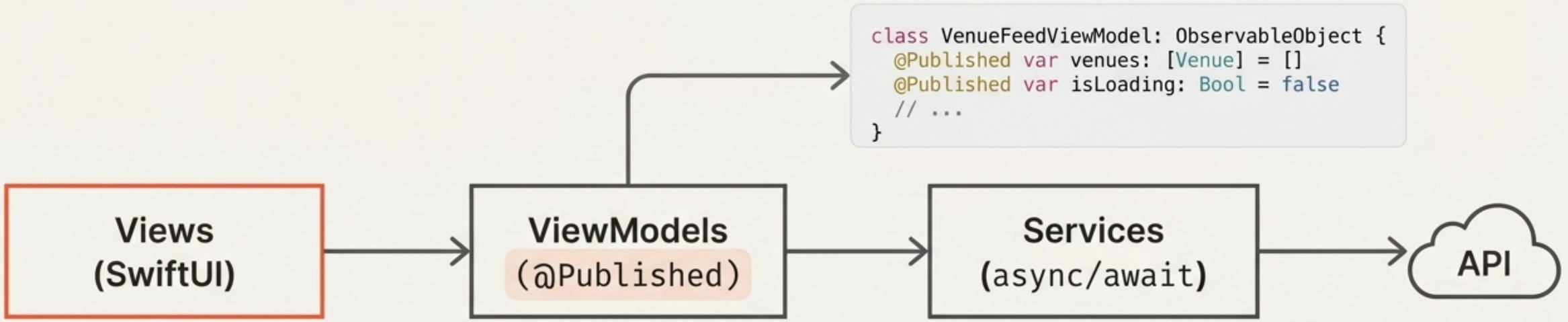
Communication Protocol Inter Bold

Standard HTTP/JSON over a REST interface ensures reliability and platform independence.

Backend: High-Performance Python Inter Bold

A lightweight, asynchronous FastAPI server delivering a RESTful API.

iOS Architecture: The MVVM Pattern in Practice



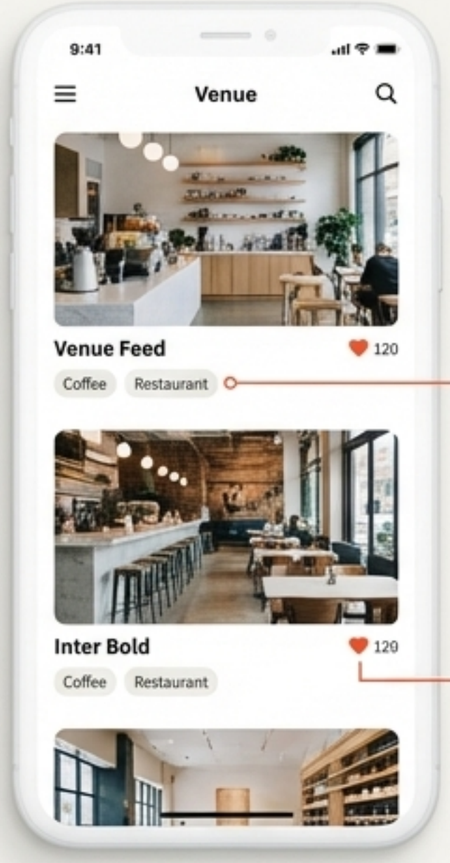
State Management: Combine + Async/Await

- `Combine` framework with `@Published` properties for reactive UI updates.
- Modern `async/await` syntax for clean, cancellable network requests in the service layer via native `URLSession`.

Why MVVM was the Right Choice

- **Testability:** Business logic in ViewModels can be unit-tested independently of the UI.
- **Separation of Concerns:** Views remain lightweight and declarative, while state management is centralized.
- **Apple Best Practice:** Aligns with the direction of modern iOS development.

Frontend: A Polished & Intuitive User Experience

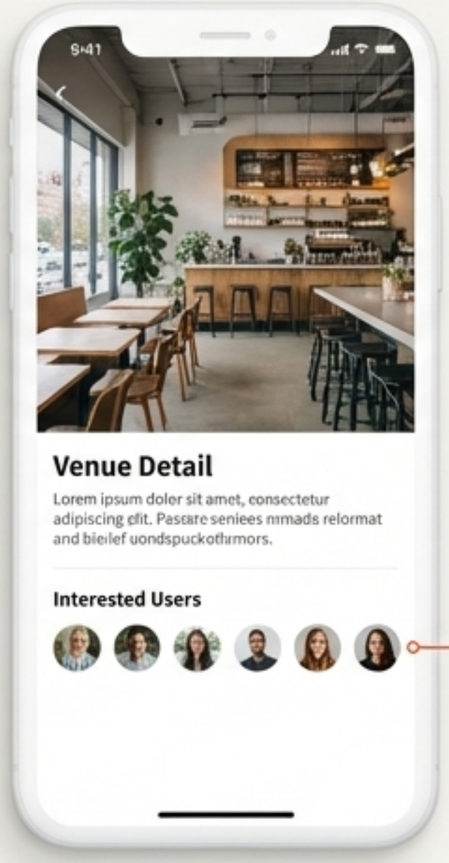


Category Badges

Interest Counts

Venue Feed

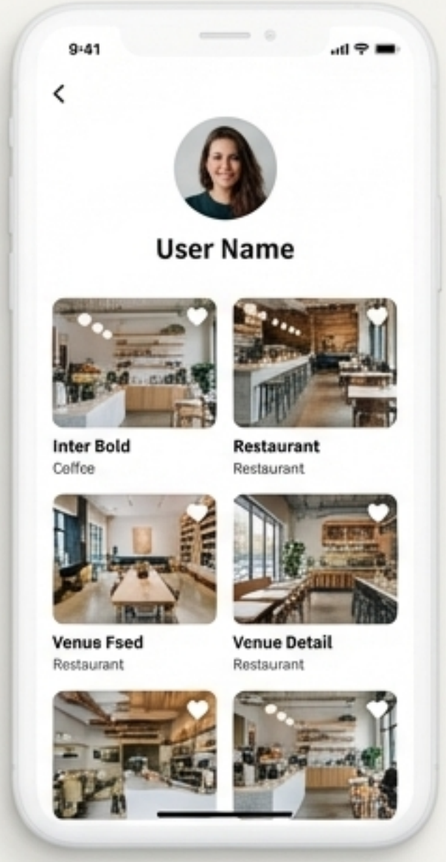
A lazy-loading scroll view for performance, with personalized recommendations at the top.



Social Proof: See who's interested.

Venue Detail

An immersive view with a hero image and a social list of interested users.



User Profile

A two-column grid of the user's saved venues, providing a personalized collection.

Frontend Polish: The Details That Differentiate

Optimistic UI Updates

Interactions provide instant visual feedback before the network request completes, making the app feel incredibly responsive.

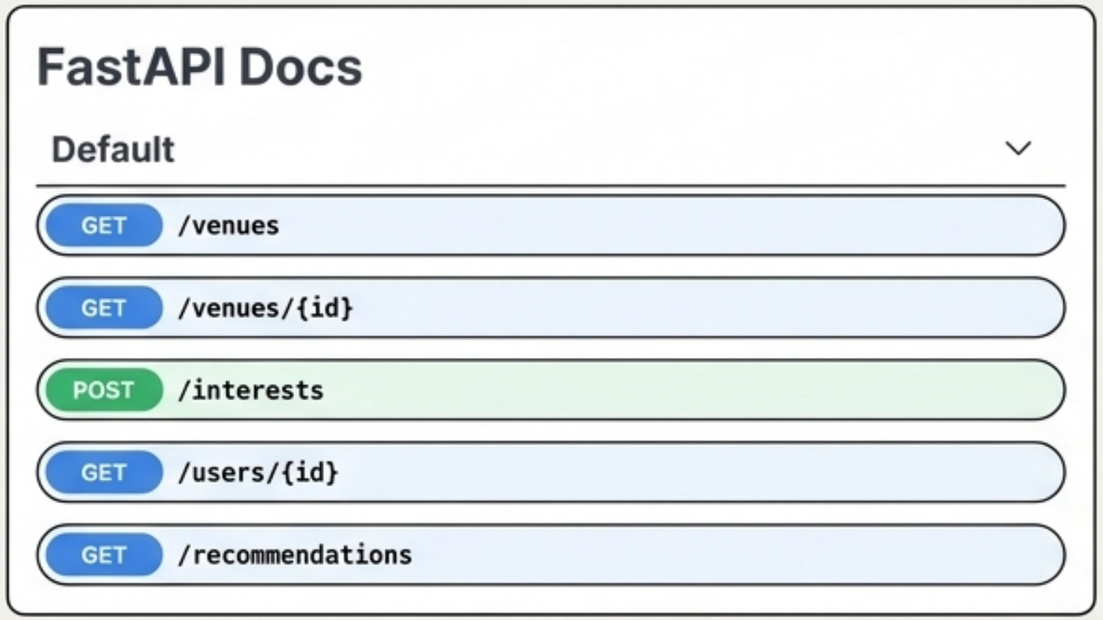
Spring Animations & Haptics

The interest button scales by 1.2x with a satisfying spring animation and haptic feedback, making the interaction feel tactile and rewarding.

Robust Error & Loading States

The app gracefully handles network failures with clear error messages and retry options. Skeletons and loading indicators ensure the user is never left guessing.

Backend: A Well-Defined RESTful API



Core API Endpoints

- **GET /venues**: Retrieve a list of all venues.
- **GET /venues/{id}**: Get details for a specific venue.
- **POST /interests**: Toggle user interest in a venue.
- **GET /users/{id}**: Fetch a user's profile and saved venues.
- **GET /recommendations**: Get personalized recommendations for a user.

Request:

```
// POST /interests
{
   "user_id": "user_1",
   "venue_id": "venue_4"
}
```

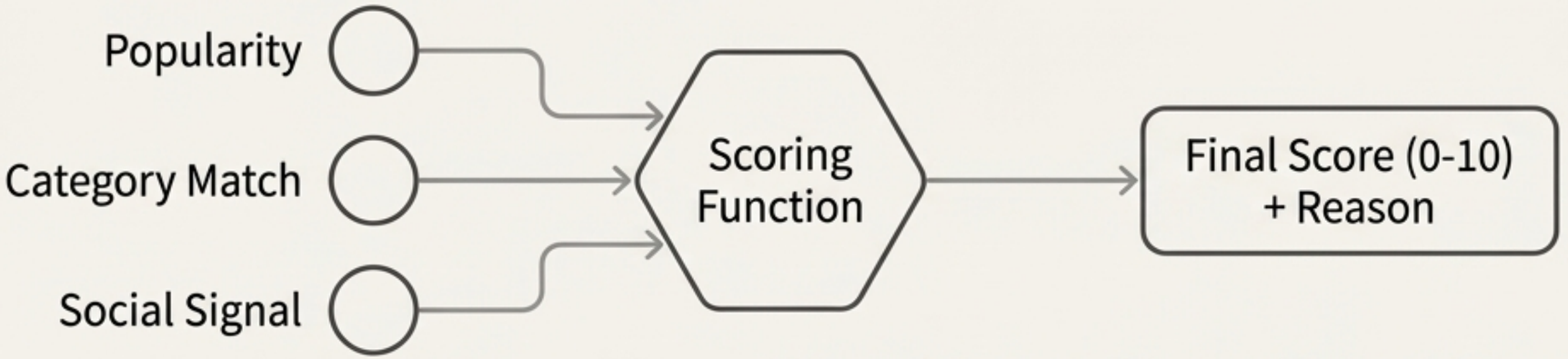
Response (Booking Triggered):

```
{
   "status": "interest added",
   "booking_triggered": true,
   "booking_details": {
      "venue_name": "The Grand Cafe",
      "party_size": 3,
      "time": "7:00 PM",
      "date": "Tomorrow"
   }
}
```

Developer Experience

- FastAPI provides interactive Swagger UI documentation and Pydantic-based data validation out-of-the-box, ensuring type safety from client to server.

Intelligent Systems I: The Recommendation Engine



Multi-Factor Scoring Algorithm

A weighted system provides personalized and transparent suggestions.

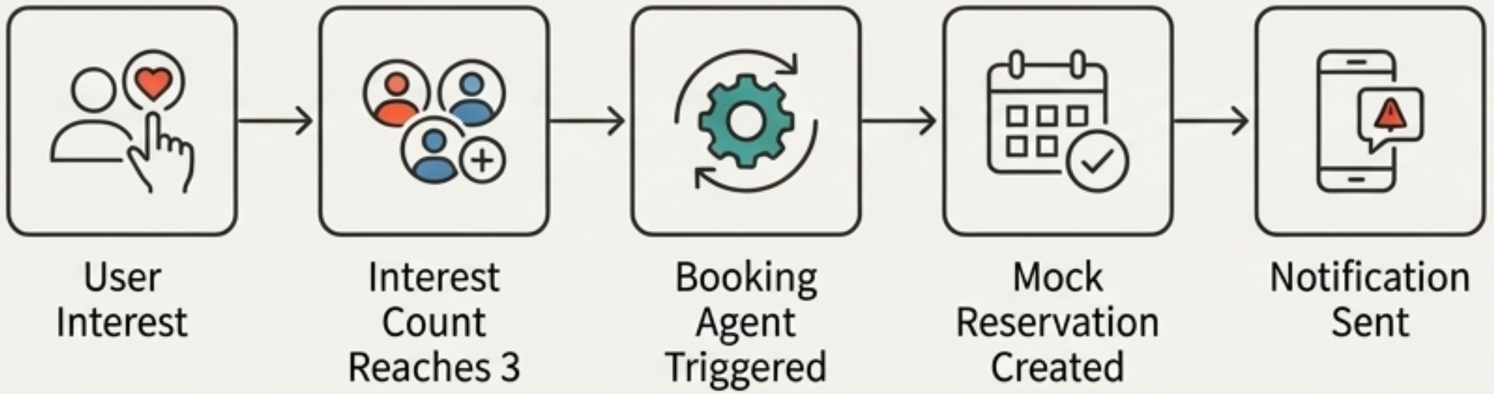
- **Popularity** (40%):** Captures trending venues based on total interest.
- **Category Match** (30%):** Aligns with the user's stated interests.
- **Social Signal** (30%):** Leverages social proof from friends' activity.

```
# The scoring formula
score = (popularity * 0.4) + \
    (category_match * 0.3) + \
    (social_signal * 0.3)
```

Transparent Output

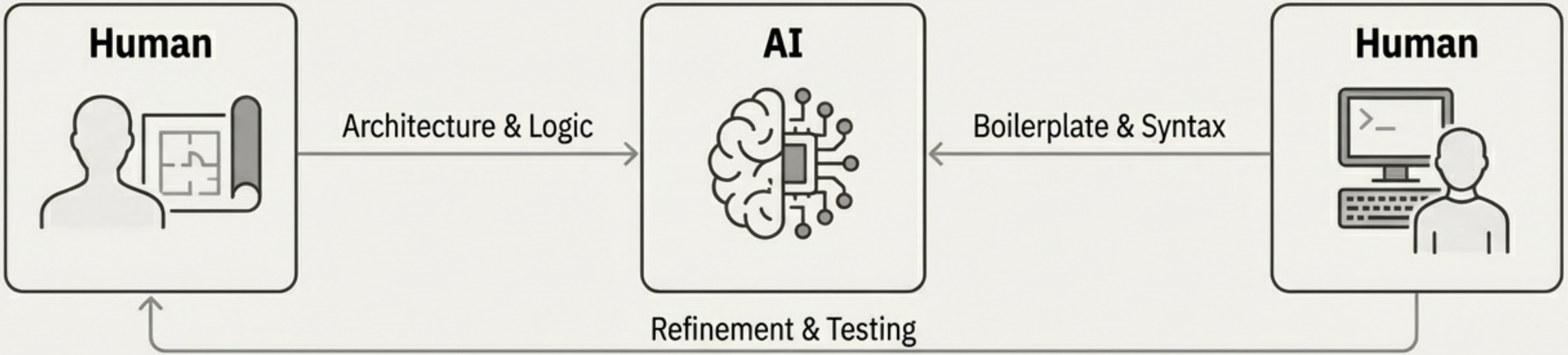
The API returns a 0-10 score with a human-readable reason, building user trust in the recommendations.

Intelligent Systems II: The Automated Booking Agent



- **Proactive Automation**
  The backend moves beyond responding to requests and actively monitors the system state to take action. (#A0A0A0)
- **Trigger Condition**
  When a venue receives interest from **3 or more users,** a trigger condition is met.
- **Automated Action**
  The agent generates a mock reservation (Date: tomorrow, Time: 7:00 PM) and a notification payload.
- **Seamless Feedback**
  The frontend receives this payload in the API response and displays a global alert to all relevant users, closing the loop.

Process: AI as a Pair Programmer



Tools Leveraged

GitHub Copilot Claude AI

GitHub Copilot for inline code completion and boilerplate generation. **Claude AI** for generating foundational templates and debugging complex issues.

Human Oversight is Key

Final architectural decisions, UX polish, algorithm weight tuning, and all testing were human-led to ensure quality and coherence.

Productivity Impact

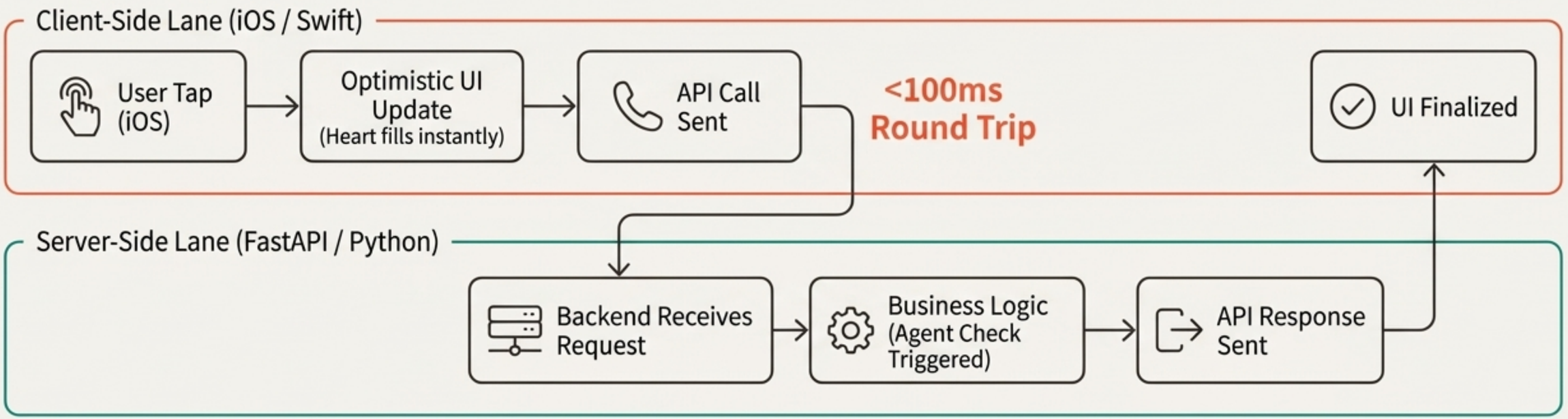
40%

Achieved an estimated **40% faster development cycle**, allowing more time to be spent on polish and feature refinement.

Engineering Judgment: Pragmatic Decisions Under Pressure

Decision	Rationale
MVVM Architecture	✅ Prioritized long-term testability and a clean separation of concerns.
In-Memory Data Store	✅ Eliminated database setup overhead to maximize development time on core features—a classic MVP trade-off.
Optimistic UI Updates	✅ Consciously chose to prioritz perceived performance and a superior user experience over strict data consistency on every tap.
No Authentication	Intentionally deferred to focus 100% of the 72 hours on the core product value: discovery and social coordination.

Integration Quality: A Seamless Full-Stack System



- **Frontend-Backend Synchronization:** The entire request-response cycle for toggling interest, including backend logic, completes in under **100ms** on a local network.
- **Real-time Recommendation Refresh:** After a user expresses new interest, the recommendation feed automatically updates to reflect new social signals.
- **Error Recovery & Resiliency**: If an API call fails after an optimistic update, the UI gracefully reverts to its previous state, ensuring data consistency for the user.
- **Concurrent Operations Support**: The system is designed to handle multiple operations at once, thanks to async handling on both the frontend (`async/await`) and backend (FastAPI).

By the Numbers: A 72-Hour Sprint

72

Hours Development Time

5 + 3

Core iOS Screens + Reusable Components

5

Fully Documented API Endpoints

2,000+

Lines of Documentation (README & Code Comments)

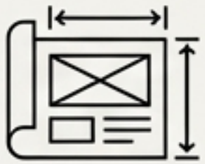
10

Critical Bugs Identified & Resolved

1

Production-Ready Core MVP

From MVP to Production: Limitations & Roadmap



Current MVP Constraints

- **In-Memory Data:** Data is not persistent and resets with the server.
- **Hardcoded User:** The app experience is limited to a single, hardcoded user (`user_1`).
- **Mock Booking Agent:** The agent simulates booking but does not integrate with real third-party APIs.

Clear Production Roadmap

- **Database:** Migrate to **PostgreSQL** with SQLAlchemy for persistent, relational data.
- **Authentication**: Implement **JWT-based authentication** for a secure, multi-user experience.
- **Real Integrations:** Connect the booking agent to **OpenTable/Resy APIs**.
- **Real-Time**: Add **WebSockets** for live updates of user interest without pull-to-refresh.

Conclusion: Full-Stack Competency Demonstrated

Key Achievements

✅ Delivered a polished, end-to-end application in **72 hours.**
✅ Demonstrated expertise in **SwiftUI, Python, FastAPI, and MVVM.**
✅ Wrote **production-ready code** with robust error handling and a superior UX.
✅ Made **pragmatic engineering decisions**, balancing speed with quality.

Explore the Codebase

`github.com/Krut-in/SwiftAssessment`