Architecture Cleanup - Summary of Changes

Date: December 3, 2025 **Completed By**: Claude (Architecture Assistant)

Overview

Completed two major architectural cleanup tasks:

1. ✅ Archived deprecated QuimbiDemoAsApp frontend
2. ✅ Created comprehensive migration plan for ticketing extraction

---

Task 1: Deprecated Frontend Archive

What Was Done

QuimbiDemoAsApp (located at `/Users/scottallen/QuimbiDemoAsApp/`) has been marked as deprecated.

Files Created/Modified:

1. **DEPRECATED.md** - Comprehensive deprecation notice explaining:
   - Why the project was deprecated
   - What replaced it (new architecture)
   - Migration notes
   - Archive strategies
   - When it can be deleted
2. **README.md** - Updated with prominent deprecation warning at the top

Git Commit:

```
commit ab5df7b
Mark project as deprecated - replaced by new architecture

- Added DEPRECATED.md explaining replacement architecture
- Updated README.md to show deprecation notice
- Project replaced by:
  - Quimbi Intelligence Backend (ML/AI)
  - Support Backend (Operations)
  - New Support Frontend
```

Why This Matters

The old QuimbiDemoAsApp was:

- ❌ Monolithic (mixed gaming + e-commerce)
- ❌ Direct calls to monolithic backend
- ❌ No clear separation of concerns

The new architecture is:

- ✅ Separated services (Intelligence, Support, Frontend)
- ✅ Clear API boundaries
- ✅ Scalable and maintainable

Archive Strategy

Recommended: Archive the GitHub repository (make it read-only)

- Preserves history for reference
- Prevents accidental modifications
- Can be unarchived if needed

Timeline: Can be fully archived after Q1 2026 when:

- New Support Frontend is deployed
- All useful components have been extracted
- Team confirms no dependencies remain

Task 2: Ticketing Extraction Migration Plan

What Was Created

TICKETING_EXTRACTION_MIGRATION_PLAN.md - A comprehensive 6-week plan to extract ticketing functionality from Quimbi Intelligence Backend.

Plan Highlights

Current Problem

The Quimbi Intelligence Backend contains:

- ✅ Customer intelligence (belongs here)
- ✅ ML/AI predictions (belongs here)
- ❌ Ticketing system (`support_app` schema) - **DOES NOT BELONG**
- ❌ Ticket messages, assignments - **SHOULD BE IN SUPPORT BACKEND**

Solution: 6-Phase Migration

Phase	Duration	Key Deliverable
Phase 1: Preparation	Week 1	Data audit, dependency map, API contracts
Phase 2: Support Backend Setup	Week 2	Support Backend ticket endpoints working
Phase 3: Intelligence Refactor	Week 3	Stateless generation APIs, tickets deprecated

Phase	Duration	Key Deliverable
Phase 4: Data Migration	Week 4	All ticket data in Support Backend
Phase 5: Frontend Migration	Week 5	Frontend calling Support Backend
Phase 6: Cleanup	Week 6	Intelligence Backend cleaned, schema dropped

Total Timeline: 6 weeks (Target: January 15, 2026)

Key Architecture Changes

Before (Current - Wrong)

```
Frontend → Quimbi Intelligence Backend
              ├── Customer Intelligence ✅
              ├── ML/AI Predictions ✅
              └── Ticketing System ❌  (shouldn't be here)
```

After (Target - Correct)

```
Frontend → Support Backend → Quimbi Intelligence Backend
              ├── Tickets ✅            ├── Intelligence ✅
              ├── Messages ✅           ├── Predictions ✅
              ├── Agents ✅             ├── AI Generation ✅
              └── SLA Tracking ✅       └── Analytics ✅
```

What Gets Moved

Database Tables (from Intelligence DB → Support DB):

- `support_app.tickets` → `support.tickets`
- `support_app.ticket_messages` → `support.ticket_messages`
- `support_app.ticket_ai_recommendations` → Cache in Support Backend
- `support_app.ticket_tags` → `support.ticket_tags`

API Endpoints (from Intelligence Backend → Support Backend):

- `POST /api/tickets` → Support Backend
- `GET /api/tickets` → Support Backend
- `GET /api/tickets/{id}` → Support Backend
- `PATCH /api/tickets/{id}` → Support Backend

What Stays in Intelligence Backend (Refactored):

- `POST /api/generation/message` - Generate AI drafts (no DB access)
- `POST /api/generation/actions` - Generate recommendations (no DB access)
- `POST /api/intelligence/analyze` - Customer intelligence

Critical Success Criteria

Migration complete when:

1. ✅ Support Backend handles all ticket operations
2. ✅ Frontend calls Support Backend (not Intelligence Backend)
3. ✅ Intelligence Backend has no ticketing code/schema
4. ✅ 100% data migrated with zero loss
5. ✅ Integration working (Support → Intelligence)
6. ✅ All tests passing

Documentation Created

Architecture Documents

1. **QUIMBI_BACKEND_ARCHITECTURE.md** - Defines Intelligence Backend boundaries
2. **SUPPORT_BACKEND_ARCHITECTURE.md** - Defines Support Backend responsibilities
3. **COMPLETE_SYSTEM_ARCHITECTURE.md** - Full platform architecture
4. **TICKETING_EXTRACTION_MIGRATION_PLAN.md** - Step-by-step migration plan

Deprecation Documents

1. **QuimbiDemoAsApp/DEPRECATED.md** - Frontend deprecation notice
2. **QuimbiDemoAsApp/README.md** - Updated with deprecation warning

---

Next Steps

Immediate (This Week)

1. Review migration plan with team
2. Get stakeholder approval to proceed
3. Run data audit queries to understand current state
4. Map all dependencies on ticketing system

Week 1 (Preparation Phase)

1. Complete data audit (row counts, storage size)
2. Map all code dependencies on `support_app` schema
3. Define final API contracts between backends
4. Set up project timeline in tracking system

Week 2 (Support Backend Setup)

1. Create Support Backend database schema
2. Implement ticket CRUD endpoints
3. Implement Quimbi Intelligence client
4. Write comprehensive tests

Risk Mitigation

Key Risks Identified

1. **Data Loss**: Mitigated by full backups before migration
2. **Frontend Breaks**: Mitigated by incremental rollout with feature flags
3. **Performance Issues**: Mitigated by load testing before cutover
4. **Missed Dependencies**: Mitigated by thorough dependency mapping

Rollback Plan

- < 1 hour: Revert frontend to call Intelligence Backend
- < 2 hours: Re-enable ticket endpoints in Intelligence Backend
- Data recovery: Restore from CSV backups taken in Phase 4

Questions for Stakeholders

Before proceeding with migration:

1. **Timeline**: Is 6 weeks acceptable, or accelerate?
2. **Data**: Migrate all historical tickets, or only recent (last 6 months)?
3. **Downtime**: Can we have 1-2 hours downtime, or must be zero-downtime?
4. **Support Backend**: Does `q.ai-customer-support` repo exist? What's its state?
5. **Testing**: Do we have staging environments for integration testing?

Files Modified/Created

In This Repository (`unified-segmentation-ecommerce`)

```
docs/
├── QUIMBI_BACKEND_ARCHITECTURE.md ✅ (created earlier)
├── SUPPORT_BACKEND_ARCHITECTURE.md ✅ (created earlier)
├── COMPLETE_SYSTEM_ARCHITECTURE.md ✅ (created earlier)
├── TICKETING_EXTRACTION_MIGRATION_PLAN.md ✅ (created today)
└── ARCHITECTURE_CLEANUP_SUMMARY.md ✅ (this file)
```

In QuimbiDemoAsApp Repository

```
QuimbiDemoAsApp/
├── DEPRECATED.md ✅ (created today)
└── README.md ✅ (updated today)
```

Summary

What We Achieved:

- ✅ Clearly marked deprecated frontend as archived
- ✅ Created detailed 6-week plan to extract ticketing
- ✅ Defined clear service boundaries
- ✅ Identified all components to migrate
- ✅ Established success criteria and rollback plan

Impact:

- 🎯 Clear path to proper architecture separation
- 🎯 Reduced confusion about what belongs where
- 🎯 Better maintainability and scalability
- 🎯 Each service has single responsibility

Current State:

- QuimbiDemoAsApp: Deprecated and documented
- Ticketing Extraction: Plan complete, awaiting approval to execute
- Architecture: Well-documented with clear boundaries

Document Owner: Quimbi Engineering Team **Date Completed**: December 3, 2025 **Status**: Ready for Review