FitFact: Research-Backed Fitness Q&A Chatbot

Satya Harish(002300329), Elenta Suzan Jacob(002530281), Rahul Gudivada(002560822)
DS 5110 - Essentials of Data Science

1. Project Kickoff

Primary Goal: Build an AI-powered fitness chatbot that provides evidence-based fitness advice by leveraging PubMed research data.

Expected Outcomes:

- A functional chatbot that can answer fitness-related questions (nutrition, exercise, recovery, etc.)
- Responses backed by peer-reviewed scientific research from PubMed
- Proper citations for all scientific claims (maintaining academic integrity)
- Local caching system to minimize API calls and improve response time
- User-friendly interface where people can ask natural language questions
- System that respects PubMed rate limits and usage policies

Success Metrics:

- Response accuracy (backed by valid research)
- Response time (under 5 seconds for cached queries, under 15-20 seconds for new queries requiring PubMed search and Claude synthesis)
- Cache hit rate (target: 60%+ for common questions)
- User satisfaction with answer quality
- Proper citation formatting in 100% of responses

Clear Project Scope

In Scope:

- Text-based chatbot interface (web or terminal-based)
- Integration with PubMed E-utilities API
- Local PostgreSQL database for caching PubMed articles and searches
- Rate-limiting mechanism to respect API constraints
- Natural language processing for question understanding
- Citation generation and metadata preservation
- Basic fitness domains: strength training, cardio, nutrition, recovery, injury prevention

Out of Scope:

- Personalized workout plan generation
- Payment processing or user accounts
- Mobile app development
- Multi-language support (English only initially)

- Production deployment to cloud platforms (Streamlit Cloud, Heroku, etc.)

Key Deliverables by Phase

Phase 1: Foundation (Week 1: Oct 27-Nov 2)

- Development environment setup (All: Satya, Elenta, Rahul)
- API keys obtained (All: PubMed, Claude)
- PostgreSQL database with 3 tables created (Satya)
- PubMed API wrapper with Biopython (Elenta)
- Claude API integration prototype (Rahul)
- Initial prompt template design (Rahul)

Phase 2: Data Pipeline & Integration (Week 2: Nov 3-9)

- 50 PubMed articles pre-seeded across 10 fitness topics (Elenta)
- Response caching system with question normalization (Satya)
- Cache lookup and storage functions (Satya)
- Citation formatting module (Elenta)
- End-to-end pipeline: Question → PubMed → Cache → Claude → Response (Elenta with team)

Phase 3: Interface & Robustness (Week 3: Nov 10-16)

- Streamlit web-based chat interface (Rahul)
- Loading indicators and status messages (Rahul)
- Rate limiting enforcement (3 requests/sec) (Elenta)
- API failure error handling and retry logic (Elenta)
- Performance monitoring queries (Satya)

Phase 4: Testing & Documentation (Week 4: Nov 17-23)

- Integration testing with 20+ diverse fitness questions (Elenta)
- Complete documentation package (README, setup guide, user guide) (Rahul)
- Database schema diagrams (Satya)
- Test report with cache hit rate and response time metrics (Satya)
- Edge case validation (Elenta)

Phase 5: Finalization & Presentation (Week 5: Nov 24-30)

- Bug fixes and system polish (All: Satya, Elenta, Rahul)
- Presentation slides (8-10 slides) (All: Satya, Elenta, Rahul)
- Backup demo video (2-3 minutes) (All: Satya, Elenta, Rahul)
- 3-page Overleaf PDF report (All: Satya, Elenta, Rahul)
- Excel progress tracker finalized (All: Satya, Elenta, Rahul)
- Presentation rehearsal (2-3 run-throughs) (All: Satya, Elenta, Rahul)

Major Milestones and Deadlines

Week 1 (Oct 27-Nov 2): Foundation Setup

Milestone: All development environments and API connections working

- **Deliverable:** Database created (Satya), can fetch 1 PubMed article (Elenta), Claude responds to 1 test prompt (Rahul)
- **Checkpoint:** Each team member completes their prototype independently

Week 2 (Nov 3-9): Pipeline Operational

Milestone: End-to-end question-answering pipeline functional

- **Deliverable:** 50 articles cached (Elenta), response caching working (Satya), complete flow tested (Rahul)
- **Checkpoint:** Successfully answer 3 test questions with proper citations

Week 3 (Nov 10-16): User Interface Complete

Milestone: Streamlit app deployed (Rahul) and robust error handling implemented (Elenta)

- **Deliverable:** Users can interact via web interface, system handles failures gracefully
- **Checkpoint:** Demo app to 2-3 classmates for initial feedback

Week 4 (Nov 17-23): Testing & Documentation

Milestone: System validated (Elenta) and fully documented (Rahul, Satya)

- **Deliverable:** Test report showing 60% cache hit rate, ¡5s cached response time, 100% citation accuracy
- **Checkpoint:** Successfully answer 20+ diverse fitness questions without crashes

Week 5 (Nov 24-30): Presentation Ready

Milestone: All materials finalized for presentation (All: Satya, Elenta, Rahul)

- **Deliverable:** Slides, demo video, documentation, GitHub repo complete
- **Final demo date:** December 1, 2024

Is there an existing dataset available for this project, or is no dataset required for the current iteration?

No dataset is required.

2. Team Discussions

What core skills does each team member contribute?

Satya Harish:

- PostgreSQL database design and optimization
- Advanced SQL and query performance tuning
- Cache management and eviction strategies
- Schema design and ER modeling
- Database indexing fundamentals

Elenta Suzan Jacob:

- Python programming and data processing
- REST API integration fundamentals
- ETL pipelines with pandas
- Rate limiting and error handling
- Data validation and quality checks
- Automated scheduling and monitoring

Rahul Gudivada:

- Python programming
- API integration basics
- Frontend development concepts
- User interface design
- Analytics and logging systems

How will each member's expertise support specific tasks or components of the project?

Satya:

- Design cache-first schema (Papers, Queries, Mappings, API Logs, Synonyms)
- Create indexes for efficient lookups
- Implement cache eviction queries (remove papers unused 50+ days)
- Write 8+ SQL analytical reports
- Optimize database performance

Elenta:

- Integrate PubMed API to fetch top 10 papers (prioritize meta-analyses + recent publications)
- Build cache-check logic: search DB first, call API on miss
- Track paper usage (times_used counter) and auto-cache at 20-30 threshold
- Implement quality scoring algorithm based on: journal impact factor, publication date recency, and study type (meta-analyses and systematic reviews ranked highest)
- Maintain synonym dictionary for query normalization
- Run daily cleanup jobs

Rahul:

- Build Streamlit chat interface
- Coordinate cache lookups and API calls
- Use sentence-transformers to match incoming queries against cached similar questions for improved cache hit rates
- Format papers into Claude prompts
- Parse Claude responses and display with citations
- Log paper usage and implement feedback collection

Are there missing skills that could create challenges or delay completion?

Critical learning gaps:

Biopython (Elenta):

- **Challenge:** Elenta needs to learn Biopython for PubMed XML parsing
- **Current:** Comfortable with REST API concepts but PubMed's E-utilities requires XML parsing via Biopython
- **Time needed:** 2-3 days to learn E-utilities (eSearch, eFetch) and XML parsing
- **Mitigation:** Start with Biopython tutorials, use documentation examples, practice with sample queries

PostgreSQL Full-Text Search (Satya):

- **Challenge:** Satya needs to learn tsvector/tsquery for efficient text searching
- **Current:** Knows basic PostgreSQL but not full-text search features
- **Time needed:** 3-4 days to learn and implement GIN indexes, ts_rank, and full-text queries
- **Mitigation:** Study PostgreSQL documentation, experiment with sample data, test performance

Streamlit (Rahul):

- **Challenge:** Learning Streamlit framework for chat interface
- **Current:** General programming knowledge but not Streamlit-specific
- **Time needed:** 2-3 days for basic app structure and components
- **Mitigation:** Streamlit has excellent documentation and chat component examples

Total learning overhead: 1-2 weeks distributed across team members working in parallel

What tools and technologies does the team already have experience with, and what must be learned?

Current Experience:

- Python (team's strongest skill)
- PostgreSQL basics (Satya)
- REST API concepts (team understands HTTP requests, JSON responses)
- pandas (Elenta)
- Git/GitHub (all members)
- General data processing workflows

Must Learn:

- Biopython (Elenta) - PubMed E-utilities wrapper and XML parsing (different from standard REST APIs)
- PostgreSQL full-text search (Satya) - tsvector, tsquery, GIN indexes, ts_rank
- Streamlit (Rahul) - Chat interface components
- sentence-transformers (Rahul) - Semantic similarity matching for query optimization

Already Manageable (minimal learning):

- Anthropic SDK (Rahul) - Straightforward API, good documentation
- NLTK basics (Rahul) - Simple tokenization and stopword removal

Based on the project's needs and the team's background, which programming languages and platforms should be used?

Final Stack:

Backend & Data Pipeline:

- Python 3.9+ (team's strength)
- PostgreSQL 14+ with pg_trgm extension
- Biopython (PubMed API - learning required)
- psycopg2 (database connection)
- pandas (ETL transformations)

LLM & NLP:

- Anthropic Python SDK (straightforward to learn)
- NLTK (keyword extraction and stopword removal)
- sentence-transformers (semantic similarity for cache optimization)

Frontend:

- Streamlit (learning required - but fastest option given Python strength)

Development Tools:

- Git/GitHub (version control)
- pytest (unit testing)
- python-dotenv (API key management)
- DBeaver (database visualization)
- Postman (API testing)

3. Skills and Tools Assessment

External Resources:

Documentation & Technical Resources:

- PostgreSQL official docs and community forums for database optimization
- Anthropic Claude API documentation and support for LLM integration challenges
- PubMed E-utilities API technical documentation for rate limiting strategies
- Biopython documentation for XML parsing and API wrapper usage

Community Resources:

- Stack Overflow and Reddit communities (r/PostgreSQL, r/datascience) for troubleshooting
- GitHub repositories with similar PubMed integration projects as reference examples

Tools, Frameworks, and Libraries:

Backend & Data Pipeline:

- Python 3.9+ (team's strongest language, excellent library ecosystem)
- PostgreSQL 14+ (robust full-text search, handles structured + unstructured data)
- Biopython (PubMed API wrapper with built-in XML parsing)
- psycopg2 (PostgreSQL Python adapter)
- pandas (ETL data manipulation)

LLM & NLP:

- Anthropic Python SDK (official Claude API integration)
- sentence-transformers (semantic similarity for cache lookup optimization - matches user queries to previously cached similar questions)
- NLTK (keyword extraction and stopword removal for query normalization)

Frontend:

- Streamlit (Python-native, rapid prototyping, built-in chat components)

Development Tools:

- Git/GitHub (version control and collaboration)
- pytest (testing framework)
- python-dotenv (secure API key management)
- DBeaver (database visualization for Satya)

Why This Stack:

- Leverages existing Python expertise across all team members
- PostgreSQL's full-text search perfect for abstract searching
- Streamlit eliminates need to learn separate frontend framework
- All tools have strong documentation and community support

Team Coordination

Ongoing Proficiency Strategies:

- Weekly code reviews - share knowledge and catch issues early
- Pair programming for integrations - Elenta + Satya on database, Elenta + Rahul on APIs
- Shared documentation in GitHub wiki - each person documents their component
- Troubleshooting sessions - open help desk for any blockers

Knowledge Repository:

- Shared Google Doc with common errors, solutions, and code snippets
- Links to helpful tutorials organized by component
- API credential management guide

Strategic Roles and Responsibilities

Satya: Foundation Layer

- **Strengths:** SQL expertise, data modeling
- **Tasks:** Schema design, indexing for text search, analytics queries, performance optimization
- **Why aligned:** Database architecture is critical infrastructure; his SQL background ensures robust foundation

Elenta: Data Collection Engine

- **Strengths:** API integration, ETL processing
- **Tasks:** PubMed API wrapper, automated extraction pipeline, quality scoring, duplicate detection
- **Why aligned:** Her pipeline skills bridge external data sources to our database; she owns data flow

Rahul: User-Facing Intelligence

- **Strengths:** LLM integration, UI development
- **Tasks:** Claude API integration, prompt engineering, chat interface, caching system, user analytics
- **Why aligned:** His NLP and frontend skills create the user experience; he translates questions into answers

4. Initial Setup

Development Environment

Required Software:

- Python 3.9+ with virtual environment support
- PostgreSQL 14+ database server
- Git for version control
- VS Code or PyCharm (code editor)
- Postman for API testing
- DBeaver or pgAdmin for database management

Python Dependencies (requirements.txt):

- anthropic (Claude API integration)
- biopython (PubMed API wrapper)
- psycopg2-binary (PostgreSQL adapter)
- pandas (data manipulation)
- streamlit (web interface)
- sentence-transformers (semantic search)
- python-dotenv (environment variables)
- pytest (testing)
- nltk (keyword extraction)