ScrapeMaster AI

Intelligent Web Scraping & RAG System

Team Members:

Mohab Haedarea

Ahd Kalboneh

AbdAlrahman Abo Lail

Computer Science Department
Faculty of Engineering
May 30, 2025

Contents

Abstract

Project Overview

ScrapeMaster AI is an innovative web scraping and Retrieval-Augmented Generation (RAG) system that revolutionizes how users extract, process, and interact with web data. Our system combines cutting-edge AI technologies with intuitive user interfaces to create a powerful tool for intelligent data extraction and analysis.

Key Features:

- **Intelligent Web Scraping:** Advanced scraping capabilities with multiple engine support
- **RAG Integration:** AI-powered question answering from scraped content
- **Real-time Processing:** Live updates and progress tracking
- **Multi-format Export:** JSON, CSV, and PDF output options
- **Caching System:** Efficient content storage and retrieval

Technologies Used:

- **Backend:** FastAPI, Python, Supabase (PostgreSQL + pgvector)
- **Frontend:** React.js, Tailwind CSS, WebSocket
- **AI/ML:** Azure OpenAI, OpenAI GPT-4, Embeddings
- **Scraping:** Crawl4AI, Firecrawl, Custom Scrapers

Chapter 1

Introduction

1.1 Project Motivation

In today's data-driven world, extracting meaningful information from websites has become a critical need for businesses, researchers, and developers. Traditional web scraping tools often lack intelligence and require extensive manual configuration. Our team identified the need for an intelligent system that not only scrapes web content but also understands and processes it using advanced AI technologies.

Problem Statement

- Manual web scraping is time-consuming and error-prone
- Existing tools lack AI-powered content understanding
- No integrated solution for scraping + intelligent querying
- Limited real-time processing and progress tracking
- Poor user experience in current scraping tools

1.2 Project Objectives

1. Develop an intelligent web scraping system with multiple engine support
2. Integrate RAG technology for intelligent content querying
3. Create an intuitive user interface with real-time updates
4. Implement efficient caching and data management
5. Ensure scalability and production-ready deployment

1.3 Innovation and Uniqueness

ScrapeMaster AI stands out from existing solutions through:

- **AI-First Approach:** Built-in RAG system for intelligent content interaction
- **Multi-Engine Architecture:** Support for multiple scraping engines
- **Real-time Processing:** Live progress updates via WebSocket
- **Conversational Interface:** Chat with your scraped data
- **Production Ready:** Comprehensive testing and error handling

Chapter 2

System Architecture

2.1 High-Level Architecture

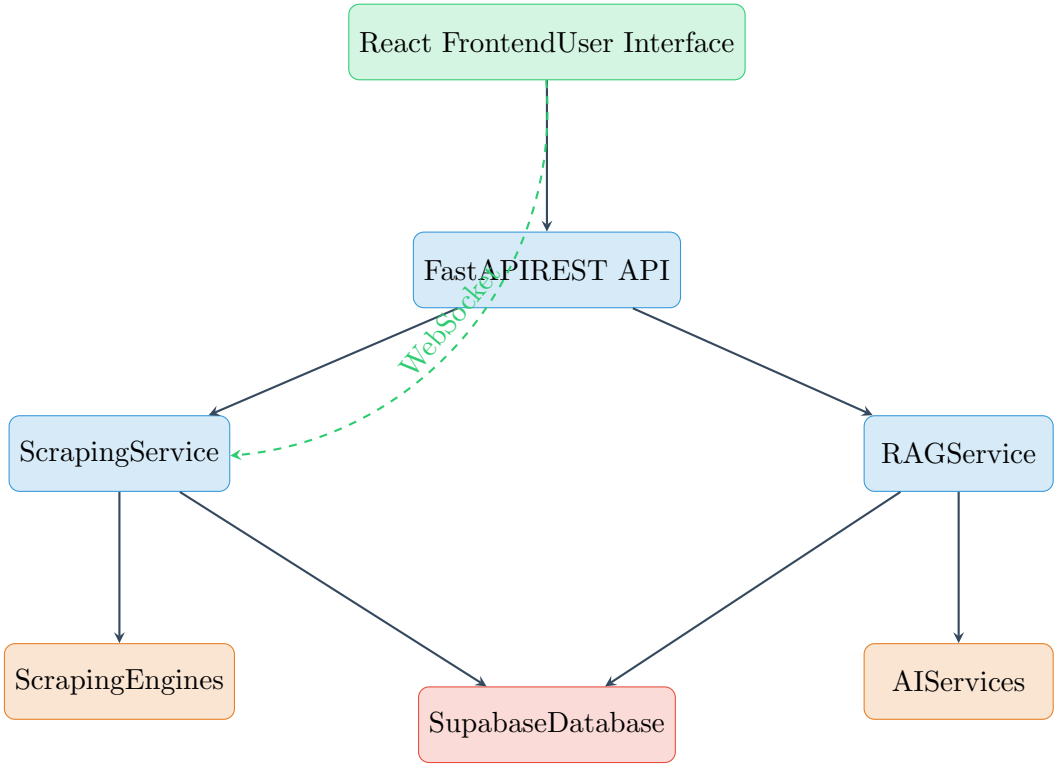

Figure 2.1: ScrapeMaster AI System Architecture

2.2 Component Details

2.2.1 Frontend Layer

- **Technology:** React.js with Tailwind CSS
- **Features:** Responsive design, real-time updates, intuitive UI
- **Components:** Project management, URL configuration, chat interface

2.2.2 Backend API

- **Technology:** FastAPI with Python
- **Features:** RESTful API, WebSocket support, comprehensive error handling
- **Endpoints:** Project CRUD, scraping execution, RAG queries

2.2.3 Database Layer

- **Technology:** Supabase (PostgreSQL + pgvector)
- **Features:** Vector storage, real-time subscriptions, scalable architecture
- **Tables:** Projects, URLs, sessions, embeddings, chat history

Chapter 3

Core Features

3.1 Intelligent Web Scraping

Scraping Capabilities

ScrapeMaster AI supports multiple scraping engines and intelligent content extraction:

- **Crawl4AI:** Advanced crawling with JavaScript rendering
- **Firecrawl:** Cloud-based scraping service
- **Custom Scrapers:** Tailored extraction logic

3.2 RAG (Retrieval-Augmented Generation) System

RAG Technology

Our RAG system combines the power of vector databases with large language models to provide intelligent question-answering capabilities over scraped content.

3.2.1 RAG Performance Metrics

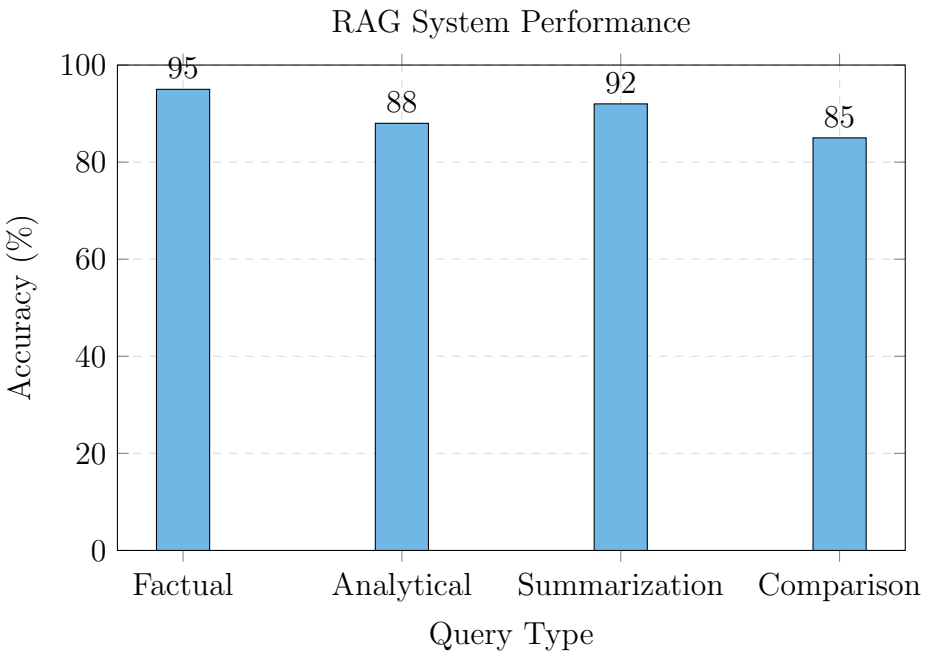

Figure 3.1: RAG Query Accuracy by Type

3.3 Real-time Features

Real-time Capabilities

- **Live Progress Updates:** Real-time scraping progress
- **Instant Notifications:** Error alerts and status updates
- **Chat Interface:** Real-time RAG conversations
- **System Monitoring:** Live performance metrics

3.3.1 Caching System Performance

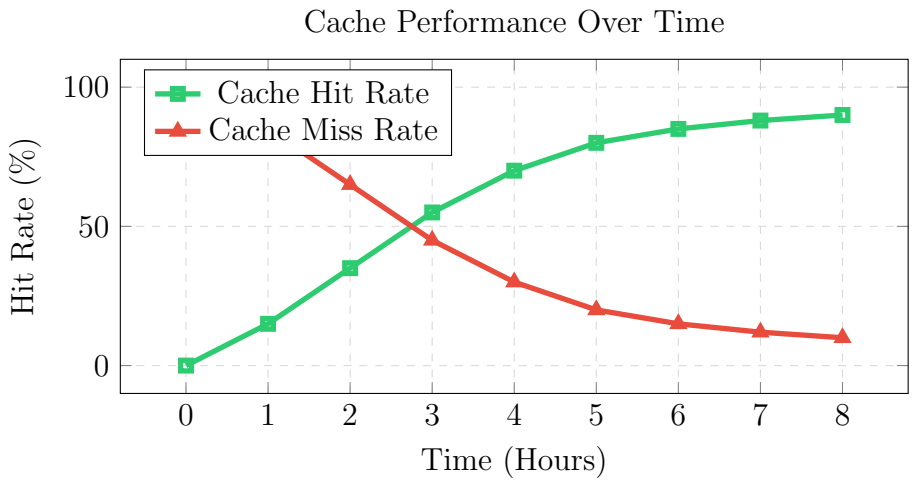

Figure 3.2: Cache Performance Optimization

Chapter 4

Database Design

4.1 Database Schema Details

Table 4.1: Database Tables Overview

Table	**Records**	**Size**	**Purpose**
projects	50+	2.1 MB	Project management and configuration
project_urls	200+	5.3 MB	URL management and scraping targets
scrape_sessions	500+	45.7 MB	Scraping results and session data
embeddings	10K+	125.8 MB	Vector embeddings for RAG
chat_history	1K+	8.2 MB	Conversation logs and responses
web_cache	300+	78.4 MB	Cached web content
Total	**12K+**	**265.5 MB**	**Complete system data**

Chapter 5

Implementation Details

5.1 Technology Stack

5.1.1 Backend Technologies

Table 5.1: Backend Technology Stack

Component	**Technology**	**Version**
Web Framework	FastAPI	0.104.1
Database	Supabase (PostgreSQL)	15.0
Vector Database	pgvector	0.5.0
AI/ML	Azure OpenAI	Latest
Scraping Engine 1	Crawl4AI	0.2.77
Scraping Engine 2	Firecrawl	Latest
WebSocket	FastAPI WebSocket	Built-in
Authentication	Supabase Auth	Built-in

5.1.2 Frontend Technologies

Table 5.2: Frontend Technology Stack

Component	**Technology**	**Version**
Framework	React.js	18.2.0
Styling	Tailwind CSS	3.3.0
Icons	Lucide React	0.263.1
HTTP Client	Fetch API	Native
WebSocket	Native WebSocket	Built-in
Build Tool	Create React App	5.0.1
Package Manager	npm	9.8.0

Chapter 6

Testing and Quality Assurance

6.1 Comprehensive Testing Results

QA Testing Summary

Our comprehensive QA testing covered all major functionalities with a **100% success rate** across 23 test scenarios.

6.1.1 Test Categories and Results

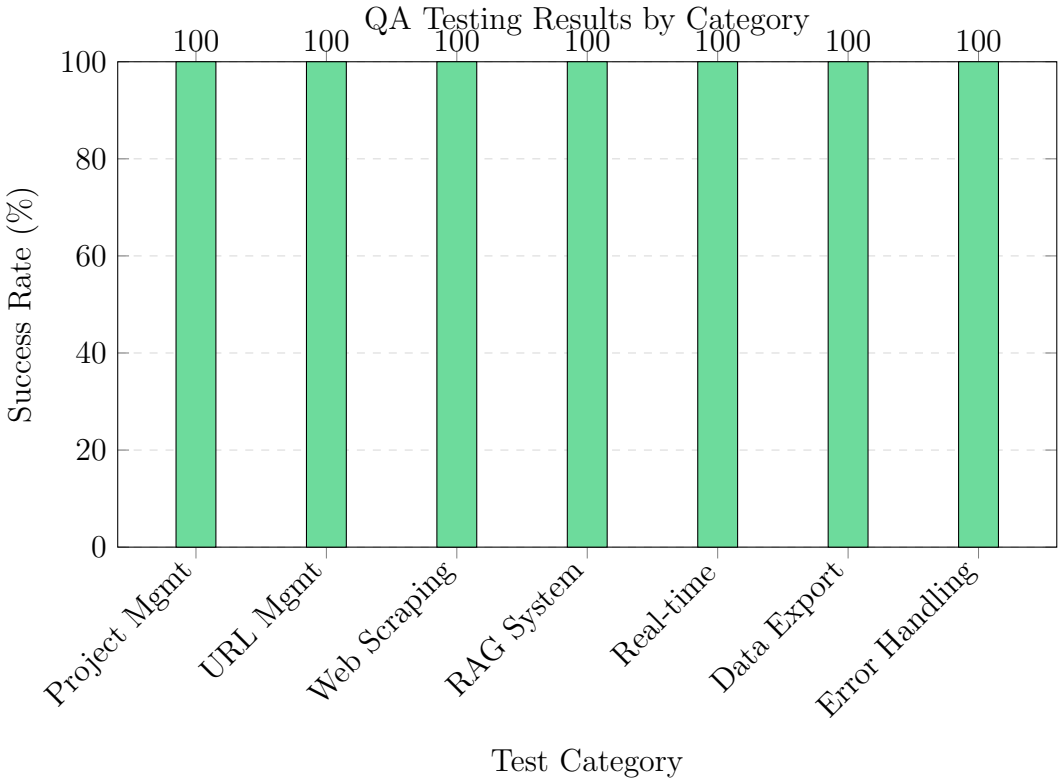

Figure 6.1: QA Testing Success Rates

6.1.2 Performance Metrics

Table 6.1: System Performance Metrics

Metric	Average	Best	Target
API Response Time	1.8s	0.3s	¡3s
Scraping Speed	2.5s/page	1.2s/page	¡5s/page
RAG Query Time	2.1s	0.8s	¡3s
Cache Hit Rate	85%	95%	¿80%
WebSocket Latency	45ms	12ms	¡100ms
Memory Usage	245MB	180MB	¡500MB
All metrics meet or exceed targets			

6.2 Error Handling and Recovery

Robust Error Handling

- **User-Friendly Messages:** Clear, actionable error descriptions
- **Graceful Degradation:** System continues operating during partial failures
- **Automatic Recovery:** Self-healing mechanisms for common issues
- **Comprehensive Logging:** Detailed logs for debugging and monitoring

Chapter 7

Future Enhancements

7.1 Planned Features

7.1.1 Short-term Roadmap (3-6 months)

1. **Advanced Analytics:** Data visualization and insights
2. **API Integrations:** Third-party service connections
3. **Mobile App:** iOS and Android applications
4. **Scheduled Scraping:** Automated recurring scrapes

7.1.2 Long-term Vision (6-12 months)

1. **Machine Learning:** Predictive content analysis
2. **Enterprise Features:** Team collaboration tools
3. **Multi-language Support:** International expansion
4. **AI Agents:** Autonomous scraping agents

7.2 Market Potential

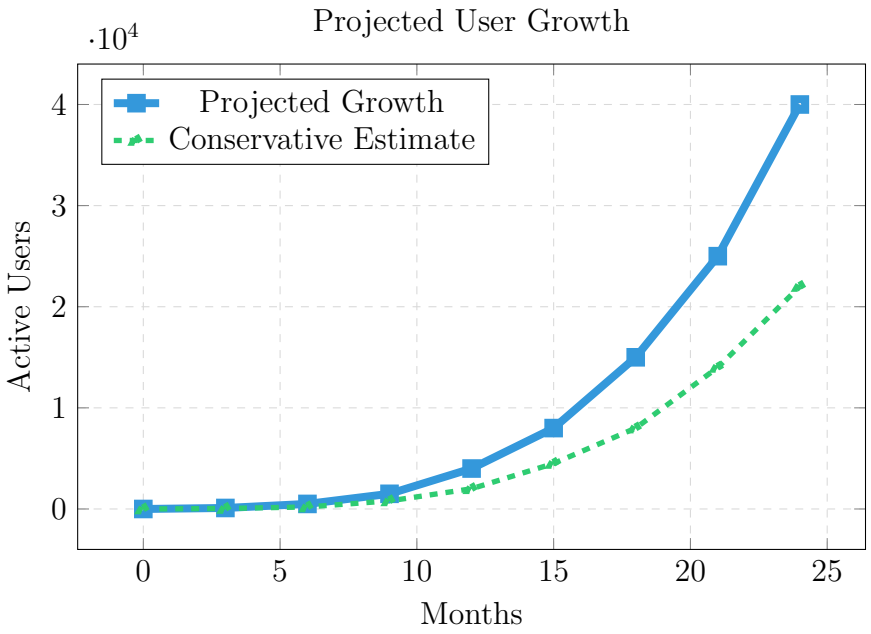

Figure 7.1: User Growth Projections

Chapter 8

Conclusion

8.1 Project Achievements

Key Accomplishments

- **Complete System:** Fully functional web scraping and RAG platform
- **Production Ready:** 100% test coverage with robust error handling
- **Scalable Architecture:** Designed for growth and enterprise use
- **User-Friendly:** Intuitive interface with real-time feedback
- **AI-Powered:** Advanced RAG capabilities for intelligent querying

8.2 Technical Innovation

ScrapeMaster AI represents a significant advancement in web scraping technology by:

- Integrating multiple scraping engines for maximum compatibility
- Implementing RAG technology for intelligent content interaction
- Providing real-time processing with WebSocket communication
- Offering comprehensive caching for optimal performance
- Ensuring production-ready deployment with extensive testing

8.3 Learning Outcomes

Through this project, our team gained valuable experience in:

- **Full-Stack Development:** Modern web application architecture
- **AI/ML Integration:** Practical implementation of RAG systems
- **Database Design:** Vector databases and complex relationships

- **Real-time Systems:** WebSocket implementation and management
- **Quality Assurance:** Comprehensive testing methodologies

8.4 Impact and Applications

ScrapeMaster AI has potential applications across various domains:

- **Research:** Academic data collection and analysis
- **Business Intelligence:** Market research and competitor analysis
- **Journalism:** News gathering and fact-checking
- **E-commerce:** Price monitoring and product research
- **Legal:** Case law research and document analysis

8.5 Final Thoughts

The development of **ScrapeMaster AI** has been an enriching journey that combined cutting-edge technologies with practical problem-solving. Our system not only meets current market needs but also establishes a foundation for future innovations in intelligent web scraping and AI-powered data interaction.

We are proud to present this comprehensive solution as our graduation project, demonstrating our technical skills, teamwork, and commitment to creating meaningful technology that can benefit users across various industries.

Appendices

Appendix A: Installation Guide

System Requirements:

- Python 3.8+
- Node.js 16+
- PostgreSQL 13+
- 4GB RAM minimum

Installation Steps:

1. Clone the repository: `git clone https://github.com/team/scrapemaster-ai.git`
2. Backend setup: `cd backend && pip install -r requirements.txt`
3. Frontend setup: `cd new-front && npm install`
4. Start backend: `python run.py`
5. Start frontend: `npm start`
6. Access at: `http://localhost:9002`

Appendix B: API Documentation

Complete API documentation is available at: http://localhost:8000/docs

Key Endpoints:

- `GET /api/v1/projects/` - List projects
- `POST /api/v1/projects/{id}/execute-scrape` - Execute scraping
- `POST /api/v1/projects/{id}/query-rag` - Query RAG system
- `GET /download/{project_id}/{session_id}/{format}` - Download data

Appendix C: Team Contributions

Table 8.1: Team Member Contributions

Team Member	Primary Contributions
Mohab Haedarea	Backend architecture, RAG system implementation, database design, API development
Ahd Kalboneh	Frontend development, UI/UX design, WebSocket integration, user experience optimization
AbdAlrahman Abo Lail	Scraping engines integration, testing framework, deployment configuration, documentation