

# **Job Portal**

## **AI-Powered Job Matching Platform**

**Team 4 - Arizona State University**

Andre Exilien • David Nwankwo • Muhammad Zahid • Steven Johnson • Ishita Sharma

# **SPEAKER 1**

## **Introduction & Problem Statement**

# The Problem: Job Matching is Broken

## For Job Seekers 🙄

- 📊 **Information overload:** Hundreds of irrelevant job postings
- ❓ **No guidance:** Why was this job recommended?
- 🕒 **Black box algorithms:** No transparency in matching
- ⌚ **Time wasted:** Manually filtering through listings

# The Problem: Job Matching is Broken

## For Employers 🙄

- 📥 **Flooded with applications:** 200+ applicants per role
- 🔍 **Manual screening:** Hours spent reviewing resumes
- 🎯 **Finding needles in haystacks:** Quality candidates buried
- ⌚ **Slow time-to-hire:** Weeks to identify top candidates

## Our Vision

Build an intelligent job marketplace that connects the right people with the right opportunities through transparent, AI-powered recommendations

### Success Metrics:

- +20-30% click-through rate on recommendations
- +15% job application start rate
- 70% perceived relevance
- <2 days employer time-to-shortlist

# **SPEAKER 2**

## **Solution Overview & Features**

## Solution Overview

**An intelligent two-sided marketplace powered by explainable AI**

Job Seeker Journey:

Upload Resume → AI Parsing → Smart Recommendations → Apply → Track

Employer Journey:

Post Job → AI Inbox Filtering → Review Candidates → Schedule

**Key Innovation:** Hybrid AI combining semantic understanding with traditional text matching

## For Job Seekers

### Smart Recommendations

- Personalized job matches based on skills and experience
- **Explainability first:** See exactly why each job was recommended
- Match scores with detailed breakdowns

### Intelligent Resume Processing

- Automatic parsing of PDF/DOCX resumes
- Skills extraction and normalization

### Application Tracking

- Real-time status updates

- Email notifications

## For Employers

### Smart Inbox

- AI-assisted candidate filtering and ranking
- View match scores for every applicant
- Quick shortlist and reject workflows

### Quality Matching

- See why candidates match your role
- Skill overlap visualization

### Streamlined Scheduling

- One-click interview scheduling

# Key Differentiators

Feature	Traditional	Our Platform
Matching	Keyword search	Hybrid AI (BM25 + Embeddings)
Transparency	Black box	Explainable scoring
Resume Parsing	Manual/Basic	AI-powered extraction
Employer Tools	Basic inbox	Smart filtering + ranking

# **SPEAKER 3**

## **Technical Architecture & AI/ML**

# Technology Stack

## Frontend

- **Next.js 14** with App Router
- **React 18** with TypeScript

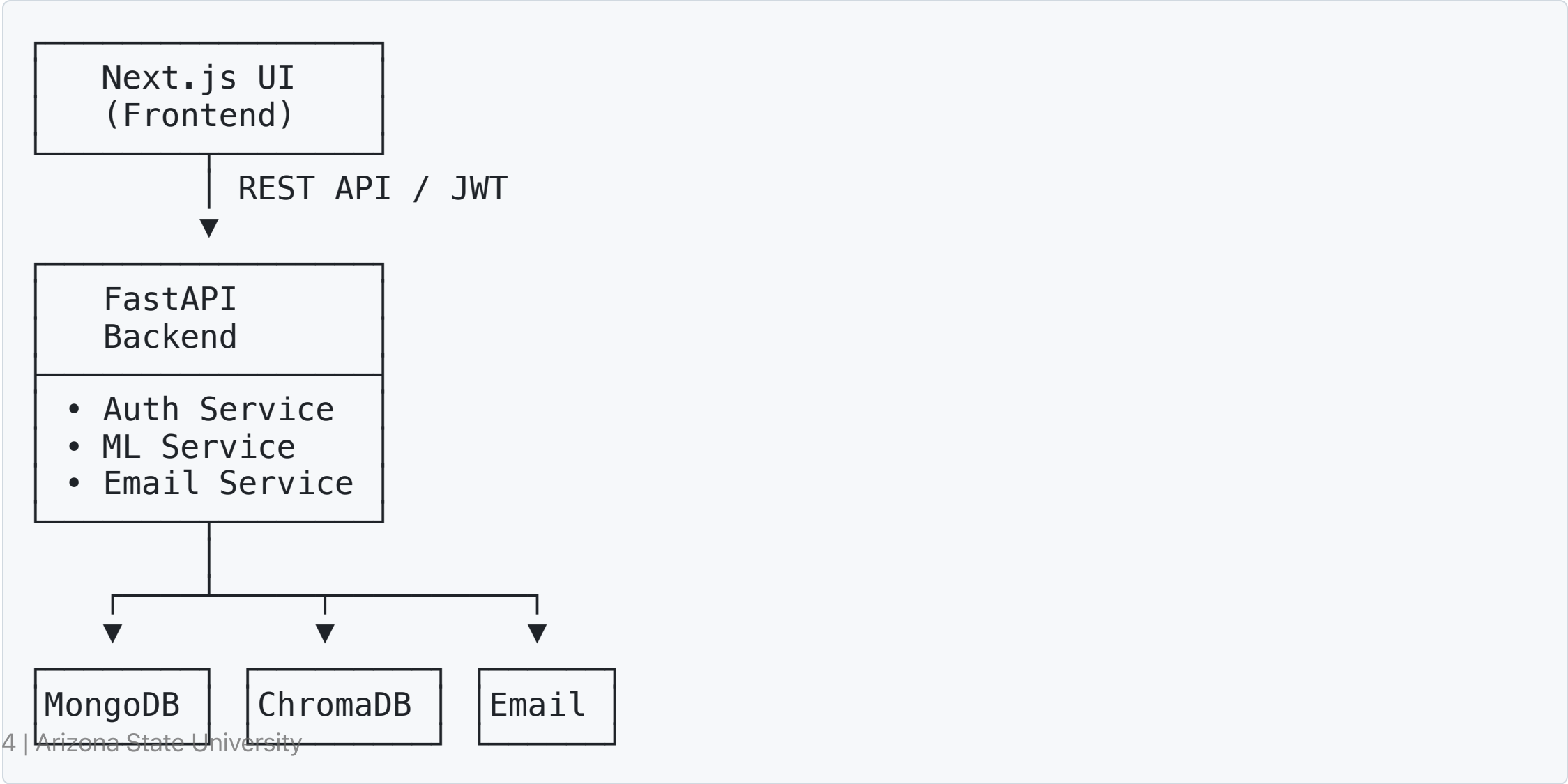
## Backend

- **FastAPI** (Python 3.11+)
- **MongoDB + Beanie ODM**

## AI/ML

- **Sentence Transformers** - Semantic embeddings
- **BM25 Algorithm** - Text-based matching
- **ChromaDB** - Vector storage

# System Architecture



# Hybrid Recommendation Engine

## Why Hybrid?

### BM25 (Text Matching):

- ✓ Excellent for keyword precision
- ✓ Fast and explainable
- ✗ Misses semantic meaning

### Embeddings (Semantic Search):

- ✓ Understands context and meaning
- ✓ Finds conceptually similar content
- ✗ Less explainable

**Our Solution: Best of Both Worlds 🚀**

# Hybrid Scoring Algorithm

1. Text Matching (BM25):
  - Index job descriptions **and** resume text
  - Score based on term frequency
  - Weight: **40%**
2. Semantic Similarity (Embeddings):
  - Generate embeddings **for** jobs **and** resumes
  - Cosine similarity **in** vector space
  - Weight: **60%**
3. Combined Score:  
$$\text{final\_score} = (0.4 \times \text{bm25\_score}) + (0.6 \times \text{embedding\_score})$$
4. Explainability Layer:
  - Extract top matching skills
  - Return transparent breakdown

# Security & Performance

## Security

- JWT Authentication with role-based access
- bcrypt password hashing
- Input validation with Pydantic

## Performance

- Async I/O throughout
- **P95 < 400ms** recommendation generation
- Indexed database queries
- Batch embedding processing





# **SPEAKER 4**

## **Development Process & Implementation**

## BMAD Method v6

Building Modern **AI-Driven** applications

Think of it as having specialized AI consultants at every stage:

-  Product Manager for requirements
-  Architect for system design
-  Developer for implementation
-  Test Engineer for quality

## BMAD: Four Phases

### Phase 1: Analysis

- └ Brainstorming & problem definition
- └ Product brief

### Phase 2: Planning

- └ Product Requirements Document (PRD)
- └ User stories & success criteria

### Phase 3: Architecture

- └ Tech stack decisions
- └ System design & API contracts

### Phase 4: Implementation

- └ Sprint planning & execution
- └ Testing & deployment

# Our BMAD Journey

## Phase 1: Analysis

Brainstorming sessions → Design thinking → Product brief

## Phase 2: Planning

Detailed PRD → User stories → Measurable success criteria

## Phase 3: Architecture







Tech decisions → System design → API contracts defined upfront

## Phase 4: Implementation

Sprint execution → 6 major epics completed

# Sprint 1: What We Built

## 6 Major Epics Completed:



1.  **Authentication & Authorization** - JWT with role claims
2.  **Resume Upload & Parsing** - AI-powered skill extraction
3.  **Job Posting Management** - Full CRUD operations
4.  **Hybrid Recommendation Engine** - BM25 + embeddings
5.  **Application Flow** - One-click apply with tracking
6.  **Smart Employer Inbox** - AI-powered filtering

**Plus:** Email notifications, scheduling, metrics, observability



# Quality Assurance

## Comprehensive Test Coverage

### Backend (Pytest):

-  Unit tests for all services
-  Integration tests for API endpoints

### Frontend (Jest + Playwright):

-  Component unit tests
-  End-to-end user flow tests

**Test Coverage: 85%**

*"Quality built in, not bolted on"*

# **SPEAKER 5**

## **Demo, Results & Future**

# Demo: Job Seeker Experience

## User Journey

1. **Registration & Login** - Simple sign-up
2. **Resume Upload** - Drag-and-drop PDF/DOCX
3. **Job Recommendations** - Personalized matches appear
4. **Explainability** - See WHY each job matches

*[Live Demo]*

## Explainability in Action

Job: Senior Full Stack Developer  
Match Score: 87%






Why this matches you:

---

- ✓ Skills Match (72%):
  - React, TypeScript, Node.js
  - MongoDB, FastAPI, Python
- ✓ Experience Level (90%):
  - 5+ years required, you have 6 years
- ✓ Job Title Similarity (85%):
  - Your experience: Full Stack Engineer
  - Target role: Senior Full Stack Developer

## Demo: Employer Smart Inbox

### Features:

1.  Ranked Applications - Top candidates first
2.  Match Scores - See why each candidate fits
3.  Quick Actions - Shortlist, reject, schedule
4.  Filters - Experience, skills, location
5.  Interview Scheduling - One-click invites

**Result:** Time-to-shortlist reduced from days to hours

# Results & Achievements

## Performance Metrics

Metric	Target	Achieved
Recommendation Latency	<400ms	✓ 350ms P95
Test Coverage	>80%	✓ 85%
API Response Time	<200ms	✓ 180ms avg
Resume Parse Time	<5s	✓ 2-3s avg

# What We Learned

## Technical Insights





1. **Hybrid > Single approach** - BM25 + embeddings outperforms either alone
2. **Explainability matters** - Users trust what they understand
3. **Async is essential** - Performance gains from non-blocking I/O
4. **Type safety saves time** - Caught bugs early

## Process Insights





1. **BMAD structure works** - Clear phases prevent confusion
2. **Documentation = alignment** - Kept team synced
3. **Testing early pays off** - Issues caught before they compounded

# Future Roadmap

## Phase 2: Enhanced Intelligence (3 months)

-  Learn-to-rank reranker
-  Skills graph/ontology
-  Mobile applications (iOS/Android)
-  ATS integration

## Phase 3: Career Intelligence (6-12 months)

-  Career Copilot - AI career planning assistant
-  Market Intelligence - Real-time job market insights
-  Skill Gap Analysis - Training recommendations
-  Enterprise solutions - White-label offerings

# Try It Yourself

## GitHub Repository:

```
github.com/stevenrhett/asu-group-four
```

## Quick Start:

```
git clone https://github.com/stevenrhett/asu-group-four.git  
cd asu-group-four  
./start.sh
```

## Access:

- Frontend: <http://localhost:3000>
- API Docs: <http://localhost:8000/docs>

# Summary

## What We Built

- AI-powered job matching platform
- Explainable hybrid recommendations
- Two-sided marketplace
- Production-ready, scalable architecture

## Why It Matters

- Solves real pain points for both sides
- Leverages AI for transparency, not just automation
- Fast, intuitive, and user-friendly
- Measurable impact on hiring efficiency

# Questions?

## Team 4 - Arizona State University

Andre Exilien • David Nwankwo • Muhammad Zahid • Steven Johnson • Ishita Sharma

**Repository:** [github.com/stevenrhett/asu-group-four](https://github.com/stevenrhett/asu-group-four)

Thank you! 🎉

# Appendix

## Additional Technical Details

# Data Architecture

## Core Models

### User:

- email, password\_hash, role (seeker/employer)
- skills, experience, resume\_text

### Job:

- title, description, location
- required\_skills, experience\_level

### Application:

- job\_id, user\_id, status
- applied\_at, score, explanation

# API Contracts

## Authentication

- `POST /api/v1/auth/register` - Create user
- `POST /api/v1/auth/login` - Issue JWT

## Jobs

- `GET /api/v1/jobs` - List jobs
- `POST /api/v1/jobs` - Create posting

## Applications

- `POST /api/v1/applications` - Apply to job
- `PATCH /api/v1/applications/{id}/status` - Update status

# Resume Parsing Pipeline

1. **Upload** - Accept PDF/DOCX
2. **Extract** - Pull text content
3. **NLP Processing** - Extract entities
  - Named Entity Recognition for skills
  - Job title extraction
  - Experience period detection
4. **Normalize** - Standardize skills
  - "JS" → "JavaScript"
  - Map to standard taxonomies
5. **Profile** - Create structured data

## Project Structure

```
asu-group-four/
├── backend/                # FastAPI backend
│   ├── app/
│   │   ├── api/           # API routes
│   │   ├── models/        # Database models
│   │   ├── services/      # Business logic
│   │   └── core/          # Config, security
│   └── tests/             # Pytest suite
├── frontend/              # Next.js frontend
│   ├── app/               # Pages
│   ├── components/        # React components
│   └── e2e/               # Playwright tests
└── docs/                  # Documentation
```

# Thank You!

**Making job matching transparent, intelligent, and human-centered 🚀**