Capital Ranker - AI ডিল ফ্লো অপটিমাইজার

সম্পূর্ণ প্রজেক্ট বিবরণ ও Backend Architecture

📋 প্রজেক্টের মূল বৈশিষ্ট্যসমূহ (Core Features)

১. স্বয়ংক্রিয় ডেটা একত্রীকরণ (Automated Data Aggregation)

উদ্দেশ্য: বিভিন্ন উৎস থেকে স্টার্টআপ ডেটা স্বয়ংক্রিয়ভাবে সংগ্রহ ও সমন্বয় করা

বৈশিষ্ট্য:

- DealRoom, Crunchbase এবং অন্যান্য তৃতীয় পক্ষের API থেকে রিয়েল-টাইম ডেটা সংগ্রহ
- স্টার্টআপের তহবিল ইতিহাস, টিম তথ্য, পণ্য বিবরণ স্বয়ংক্রিয়ভাবে আপডেট
- বিভিন্ন ডেটা ফরম্যাট একত্রিত করে unified database তৈরি
- ডাটা ডুপ্লিকেশন প্রতিরোধ এবং data quality validation
- Scheduled cron jobs দিয়ে নিয়মিত ডেটা সিঙ্ক

AI/ML ব্যবহার:

- Intelligent data mapping এবং entity resolution
- ডেটা স্ট্রাকচার ম্যাচিং এজেন্ট

---

২. AI ডিল স্কোরিং ও র‍্যাঙ্কিং (AI Deal Scoring & Ranking)

উদ্দেশ্য: স্টার্টআপগুলিকে বিনিয়োগ সম্ভাবনার ভিত্তিতে স্বয়ংক্রিয়ভাবে স্কোর ও র‍্যাঙ্ক করা

বৈশিষ্ট্য:

- **Investment Fit Score (0-100)** তৈরি করা যা নিম্নলিখিত বিষয়ের উপর ভিত্তি করে:
  - বাজারের আকার এবং বৃদ্ধির সম্ভাবনা
  - তহবিলের ইতিহাস এবং runway analysis
  - রাজস্ব বৃদ্ধির মেট্রিক্স (MoM, YoY growth)
  - প্রোডাক্ট-মার্কেট ফিট সূচক
  - প্রতিযোগিতামূলক সুবিধা
- Real-time ranking dashboard
- Customizable scoring parameters প্রতি VC firm এর জন্য
- Historical performance tracking

AI/ML ব্যবহার:

- Predictive ML মডেল (Gradient Boosting, Random Forest)

- Feature engineering স্টার্টআপ মেট্রিক্স থেকে
- Transfer learning ঐতিহাসিক সফল বিনিয়োগ থেকে

---

৩. থিসিস ম্যাচিং ইঞ্জিন (Thesis Matching Engine)

উদ্দেশ্য: বিনিয়োগকারীর থিসিস এবং স্টার্টআপের পিচের মধ্যে সামঞ্জস্যতা বিশ্লেষণ

বৈশিষ্ট্য:

- NLP-powered পিচ ডেক এবং investor thesis বিশ্লেষণ
- **Relevancy Score (0-100)** প্রতিটি ডিলের জন্য
- Sector, technology stack, business model ম্যাচিং
- Investment stage preference alignment
- Geographic focus matching
- থিম্যাটিক keyword এবং concept extraction
- Visual highlighting ম্যাচ করা sections এর

AI/ML ব্যবহার:

- Transformer-based NLP (BERT, GPT-based embeddings)
- Semantic similarity scoring
- Topic modeling (LDA)
- Named Entity Recognition (NER) for sectors/technologies

---

৪. প্রতিষ্ঠাতা মূল্যায়ন স্কোরকার্ড (Founder Evaluation Scorecard)

উদ্দেশ্য: প্রতিষ্ঠাতাদের গুণাবলী এবং পটেনশিয়াল নিরপেক্ষভাবে মূল্যায়ন

বৈশিষ্ট্য:

- **Founder Score (0-100)** নিম্নলিখিত মাপকাঠিতে:
  - পূর্ববর্তী অভিজ্ঞতা এবং track record
  - শিক্ষাগত যোগ্যতা
  - Industry expertise এবং domain knowledge
  - নেতৃত্ব গুণাবলী (leadership indicators)
  - অভিযোজন ক্ষমতা (adaptability metrics)
  - টিম গঠন দক্ষতা
- LinkedIn, AngelList profile analysis
- Co-founder team dynamics evaluation

- Reference check automation
- Red flag detection

AI/ML ব্যবহার:

- NLP-based qualitative analysis
- Sentiment analysis প্রতিষ্ঠাতার communication থেকে
- Pattern recognition সফল প্রতিষ্ঠাতাদের traits থেকে
- Bias-free evaluation algorithms

---

৫. কৌশলগত সতর্কতা এবং ড্যাশবোর্ড (Strategic Alerts & Dashboard)

উদ্দেশ্য: পোর্টফোলিও ঝুঁকি ট্র্যাকিং এবং রিয়েল-টাইম market intelligence

বৈশিষ্ট্য:

- **Real-time Alerts:**
  - নতুন competitive threats
  - Market sentiment shifts
  - Regulatory changes প্রাসঙ্গিক sectors এ
  - Portfolio company performance anomalies
  - Funding round announcements প্রতিদ্বন্দ্বীদের
- **Interactive Dashboard:**
  - Deal flow pipeline visualization
  - Score distribution charts
  - Thesis alignment heatmaps
  - Founder scorecard comparisons
  - Portfolio health metrics
- Customizable notification preferences
- Email/Slack/Teams integration

AI/ML ব্যবহার:

- Anomaly detection algorithms
- Predictive analytics for risk assessment
- Real-time news sentiment analysis
- Time series forecasting

---

৬. পোর্টফোলিও ম্যানেজমেন্ট (Portfolio Management)

উদ্দেশ্য: বিদ্যমান বিনিয়োগ ট্র্যাক এবং মনিটর করা

বৈশিষ্ট্য:

- Portfolio company performance tracking
- KPI monitoring (burn rate, revenue, user growth)
- Quarterly report generation
- Exit scenario modeling
- Follow-on investment recommendations

---

৭. কোলাবোরেশন টুলস (Collaboration Tools)

উদ্দেশ্য: VC টিমের মধ্যে সহযোগিতা সুবিধা

বৈশিষ্ট্য:

- Deal notes এবং comments sharing
- Internal rating এবং voting system
- Meeting scheduling with startups
- Document sharing (pitch decks, financials)
- Task assignment and follow-ups

🏗️ Backend Folder Structure (MVC Architecture)

```
backend/
│
├── src/
│   ├── config/                 # Configuration files
│   │   ├── database.ts         # Database connection setup
│   │   ├── env.ts              # Environment variables validation
│   │   ├── logger.ts           # Winston logger configuration
│   │   └── ml-service.ts       # Python ML service connection config
│   │
│   ├── controllers/            # Request handlers (Controllers)
│   │   ├── auth.controller.ts    # Authentication & authorization
│   │   ├── deal.controller.ts    # Deal CRUD operations
│   │   ├── scoring.controller.ts # AI scoring requests
│   │   ├── thesis.controller.ts  # Thesis matching operations
│   │   ├── founder.controller.ts # Founder evaluation
│   │   ├── alert.controller.ts   # Alert management
│   │   ├── report.controller.ts  # Report generation
│   │   └── portfolio.controller.ts # Portfolio management
│   │
│   ├── models/                # Database models (Mongoose/Sequelize)
│   │   ├── User.ts            # User model (VC investors)
│   │   ├── Startup.ts         # Startup/Deal model
│   │   ├── Score.ts           # Score history model
│   │   ├── InvestorThesis.ts  # Investor thesis model
│   │   ├── Founder.ts         # Founder profile model
│   │   ├── Alert.ts           # Alert configuration model
│   │   ├── Portfolio.ts       # Portfolio company model
│   │   └── ActivityLog.ts     # Audit trail model
│   │
│   ├── services/              # Business logic layer
│   │   ├── auth.service.ts    # Authentication logic (JWT, OAuth)
│   │   ├── deal.service.ts    # Deal processing logic
│   │   ├── aggregation.service.ts # Data aggregation from APIs
│   │   ├── ml-client.service.ts  # Python ML service communication
│   │   ├── scoring.service.ts    # Scoring orchestration
│   │   ├── thesis.service.ts     # Thesis matching logic
│   │   ├── founder.service.ts    # Founder evaluation logic
│   │   ├── notification.service.ts # Email/Slack notifications
│   │   ├── report.service.ts     # Report generation logic
│   │   └── cache.service.ts      # Redis caching logic
│   │
│   ├── routes/                # API routes
│   │   ├── index.ts           # Main router
│   │   ├── v1/                # API version 1
│   │   │   ├── auth.routes.ts    # /api/v1/auth/*
│   │   │   ├── deal.routes.ts    # /api/v1/deals/*
│   │   │   ├── scoring.routes.ts # /api/v1/scoring/*
│   │   │   ├── thesis.routes.ts  # /api/v1/thesis/*
│   │   │   ├── founder.routes.ts # /api/v1/founders/*
│   │   │   ├── alert.routes.ts   # /api/v1/alerts/*
│   │   │   └── portfolio.routes.ts # /api/v1/portfolio/*
│   │
│   ├── middleware/            # Express middleware
│   │   ├── auth.middleware.ts    # JWT verification
│   │   ├── validation.middleware.ts # Request validation
│   │   ├── error.middleware.ts   # Error handling
│   │   ├── rate-limit.middleware.ts # API rate limiting
│   │   └── logger.middleware.ts  # Request logging
│   │
│   ├── validators/            # Request validation schemas
│   │   ├── auth.validator.ts     # Auth request validation
│   │   ├── deal.validator.ts     # Deal request validation
│   │   └── scoring.validator.ts  # Scoring request validation
│   │
│   ├── types/                 # TypeScript interfaces & types
│   │   ├── api.types.ts          # API request/response types
│   │   ├── model.types.ts        # Database model types
│   │   ├── ml.types.ts           # ML service types
│   │   └── common.types.ts       # Common shared types
│   │
│   ├── utils/                  # Utility functions
│   │   ├── response.util.ts      # Standardized API responses
│   │   ├── date.util.ts          # Date manipulation
│   │   ├── crypto.util.ts        # Encryption/hashing
│   │   └── file.util.ts          # File handling
│   │
│   ├── jobs/                   # Background jobs (Bull Queue)
│   │   ├── data-sync.job.ts      # DealRoom/Crunchbase sync
│   │   ├── scoring.job.ts        # Batch scoring tasks
│   │   └── alert.job.ts          # Alert monitoring
│   │
│   ├── integrations/           # External API integrations
│   │   ├── dealroom.ts           # DealRoom API client
│   │   ├── crunchbase.ts         # Crunchbase API client
│   │   ├── linkedin.ts           # LinkedIn API client
│   │   └── slack.ts              # Slack API client
│   │
│   ├── database/               # Database related
│   │   ├── migrations/           # Database migrations
│   │   └── seeders/              # Seed data for development
│   │
│   └── app.ts                  # Express app setup
│
├── tests/                      # Test files
│   ├── unit/                   # Unit tests
│   ├── integration/            # Integration tests
│   └── e2e/                    # End-to-end tests
│
├── .env.example                # Environment variables template
├── .gitignore
├── package.json
├── tsconfig.json
└── README.md
```

📡 API Endpoints বিস্তারিত বিবরণ

১. Authentication APIs

`POST /api/v1/auth/register`

- **কাজ:** নতুন VC investor/user registration
- **Request Body:** `{ email, password, name, firm_name, role }`
- **Response:** JWT token + user details
- **Service:** `auth.service.ts` → User.create() → bcrypt password hashing

`POST /api/v1/auth/login`

- **কাজ:** User login
- **Request Body:** `{ email, password }`
- **Response:** JWT token + refresh token
- **Service:** `auth.service.ts` → JWT sign → Redis session storage

`POST /api/v1/auth/refresh-token`

- **কাজ:** Access token refresh
- **Request Body:** `{ refresh_token }`
- **Response:** New access token

`POST /api/v1/auth/logout`

- **কাজ:** User logout and token invalidation
- **Service:** Redis token blacklisting

---

২. Deal Management APIs

`GET /api/v1/deals`

- **কাজ:** সকল deals list করা (pagination, filtering, sorting)
- **Query Params:** `?page=1&limit=20&sector=fintech&score_min=70`
- **Response:** Paginated list of deals with scores
- **Service:** `deal.service.ts` → Startup.find() + Score.populate()

`GET /api/v1/deals/:id`

- **কাজ:** একটি নির্দিষ্ট deal এর বিস্তারিত তথ্য
- **Response:** Full startup details + scores + founder info
- **Service:** `deal.service.ts` → aggregation pipeline

`POST /api/v1/deals`

- **কাজ:** Manual deal entry
- **Request Body:** Startup details
- **Service:** `deal.service.ts` → Startup.create()

`PUT /api/v1/deals/:id`

- **কাজ:** Deal information update
- **Request Body:** Updated fields
- **Service:** `deal.service.ts` → Startup.update()

`DELETE /api/v1/deals/:id`

- **কাজ:** Deal archive/delete
- **Service:** Soft delete implementation

---

৩. AI Scoring APIs

`POST /api/v1/scoring/deal/:dealId`

- **কাজ:** একটি deal এর জন্য AI scoring trigger করা
- **প্রক্রিয়া:**
  1. `scoring.controller.ts` → deal data fetch
  2. `ml-client.service.ts` → Python ML service call (`POST http://ml-service:8000/api/v1/score_deal`)
  3. Response process এবং Score.create()
- **Response:** `{ investment_fit_score, breakdown, confidence }`

`GET /api/v1/scoring/deal/:dealId/history`

- **কাজ:** একটি deal এর scoring history
- **Response:** Historical score changes with timestamps

`POST /api/v1/scoring/batch`

- **কাজ:** Multiple deals একসাথে score করা
- **Request Body:** `{ deal_ids: [...] }`
- **Service:** Bull queue → background job

`POST /api/v1/scoring/recalculate-all`

- **কাজ:** সকল deals rescore করা (admin only)
- **Service:** Background job scheduling

---

৪. Thesis Matching APIs

`POST /api/v1/thesis`

- **কাজ:** নতুন investor thesis সংরক্ষণ
- **Request Body:** `{ investor_id, thesis_text, sectors, stages, geography }`
- **Service:** `thesis.service.ts` → InvestorThesis.create()

`PUT /api/v1/thesis/:id`

- **কাজ:** Existing thesis update

- **Service:** Thesis update + re-matching trigger

`GET /api/v1/thesis/matches/:dealId`

- **কাজ:** একটি deal এর জন্য thesis matching করা
- **প্রক্রিয়া:**
  1. Deal এবং pitch deck text fetch
  2. Investor thesis fetch
  3. Python ML service call (`POST http://ml-service:8000/api/v1/match_thesis`)
  4. Relevancy score calculation
- **Response:** `{ relevancy_score, matched_keywords, similarity_breakdown }`

`GET /api/v1/thesis/investor/:investorId/matches`

- **কাজ:** একজন investor এর জন্য top matching deals
- **Response:** Ranked list of deals with relevancy scores

---

৫. Founder Evaluation APIs

`POST /api/v1/founders/evaluate/:founderId`

- **কাজ:** Founder evaluation score calculate করা
- **প্রক্রিয়া:**
  1. Founder profile data fetch (LinkedIn, AngelList)
  2. Python ML service call (`POST http://ml-service:8000/api/v1/evaluate_founder`)
  3. Founder score এবং breakdown সংরক্ষণ
- **Response:** `{ founder_score, experience_score, leadership_score, adaptability_score, red_flags }`

`GET /api/v1/founders/:id`

- **কাজ:** Founder এর full profile এবং evaluation
- **Response:** Complete founder information + historical scores

`PUT /api/v1/founders/:id`

- **কাজ:** Founder information update
- **Service:** Manual data update + re-evaluation trigger

---

৬. Alert Management APIs

`GET /api/v1/alerts`

- **কাজ:** Active alerts list
- **Query Params:** `?type=market_shift&priority=high`
- **Response:** Filtered alerts

`POST /api/v1/alerts/configure`

- **কাজ:** Alert preferences configure করা
- **Request Body:** `{ alert_types, threshold_values, notification_channels }`
- **Service:** Alert configuration save + subscription setup

`PUT /api/v1/alerts/:id/read`

- **কাজ:** Alert mark as read
- **Service:** Alert status update

`DELETE /api/v1/alerts/:id`

- **কাজ:** Alert dismiss করা
- **Service:** Soft delete

---

৭. Portfolio Management APIs

`GET /api/v1/portfolio`

- **কাজ:** Portfolio companies list
- **Response:** All portfolio companies with performance metrics

`GET /api/v1/portfolio/:id/performance`

- **কাজ:** Portfolio company এর performance analytics
- **Response:** KPI tracking, burn rate, milestones

`POST /api/v1/portfolio/:id/update`

- **কাজ:** Portfolio company KPI update
- **Request Body:** Updated metrics
- **Service:** Performance tracking update

---

৮. Reporting APIs

`POST /api/v1/reports/generate`

- **কাজ:** Custom report generation
- **Request Body:** `{ report_type, filters, date_range }`

- **Response:** Report ID
- **Service:** Background job → PDF generation

`GET /api/v1/reports/:id`

- **কাজ:** Generated report download
- **Response:** PDF file

`GET /api/v1/reports/deals/:dealId`

- **কাজ:** Deal-specific detailed report
- **Response:** Comprehensive deal analysis

---

৯. Data Aggregation APIs (Internal)

`POST /api/v1/aggregation/sync-dealroom`

- **কাজ:** DealRoom থেকে deals sync করা
- **Service:** `aggregation.service.ts` → DealRoom API → data processing → Startup.bulkCreate()

`POST /api/v1/aggregation/sync-crunchbase`

- **কাজ:** Crunchbase থেকে startup data sync
- **Service:** Crunchbase API integration

`POST /api/v1/aggregation/sync-linkedin`

- **কাজ:** Founder profiles LinkedIn থেকে sync
- **Service:** LinkedIn API → Founder.update()

📦 NPM Packages বিস্তারিত (Backend)

Core Framework & TypeScript

```json
{
  "express": "^4.18.2",         // Web framework
  "typescript": "^5.2.2",       // Type safety
  "@types/express": "^4.17.17", // Express type definitions
  "@types/node": "^20.8.0"      // Node.js type definitions
}
```

ব্যবহার: Express server setup, routing, middleware management

---

Database & ORM

```json
{
  "mongoose": "^7.6.3",          // MongoDB ORM (recommended)
  "@types/mongoose": "^5.11.97",
  // OR
  "sequelize": "^6.33.0",        // SQL ORM (PostgreSQL support)
  "sequelize-typescript": "^2.1.5",
  "pg": "^8.11.3",               // PostgreSQL driver
  "pg-hstore": "^2.3.4"          // PostgreSQL JSON support
}
```

ব্যবহার:

- Database schema definition
- CRUD operations
- Query building
- Relationship management

Authentication & Security

```json
{
  "jsonwebtoken": "^9.0.2",      // JWT token generation
  "@types/jsonwebtoken": "^9.0.3",
  "bcrypt": "^5.1.1",            // Password hashing
  "@types/bcrypt": "^5.0.0",
  "passport": "^0.6.0",          // Authentication middleware
  "passport-jwt": "^4.0.1",      // JWT strategy
  "helmet": "^7.0.0",            // Security headers
  "cors": "^2.8.5",              // CORS handling
  "@types/cors": "^2.8.14"
}
```

ব্যবহার:

- User authentication
- Password encryption
- JWT token generation/verification
- API security
- CORS policy management

Data Validation

```json
{
  "joi": "^17.10.2",              // Schema validation
  "express-validator": "^7.0.1",  // Request validation
  "class-validator": "^0.14.0",   // DTO validation
  "class-transformer": "^0.5.1"   // Object transformation
}
```

ব্যবহার:

- Request body validation
- Query parameter validation
- Type checking
- Error messages

HTTP Client & External APIs

```json
{
  "axios": "^1.5.1",              // HTTP client for ML service & external APIs
  "@types/axios": "^0.14.0",
  "node-fetch": "^3.3.2"          // Alternative HTTP client
}
```

ব্যবহার:

- Python ML microservice communication
- DealRoom API calls
- Crunchbase API integration
- LinkedIn API requests

Logging & Monitoring

```json
{
  "winston": "^3.10.0",           // Logging framework
  "winston-daily-rotate-file": "^4.7.1", // Log rotation
  "morgan": "^1.10.0",            // HTTP request logging
  "@types/morgan": "^1.9.5"
}
```

ব্যবহার:

- Application logging
- Error tracking
- API request/response logging
- Debug information

Environment & Configuration

```json
{
  "dotenv": "^16.3.1",          // Environment variables
  "config": "^3.3.9",           // Configuration management
  "cross-env": "^7.0.3"         // Cross-platform env variables
}
```

ব্যবহার:

- Environment variable loading
- Configuration management
- Different env configs (dev, staging, prod)

Caching

```json
{
  "redis": "^4.6.10",           // Redis client
  "@types/redis": "^4.0.11",
  "ioredis": "^5.3.2"           // Alternative Redis client
}
```

ব্যবহার:

- API response caching
- Session storage
- Token blacklisting
- Rate limiting data storage

Background Jobs

```json
```

```
{
  "bull": "^4.11.4",              // Queue management
  "@types/bull": "^4.10.0",
  "node-cron": "^3.0.2",          // Scheduled tasks
  "@types/node-cron": "^3.0.8"
}
```

ব্যবহার:

- Background scoring tasks
- Data sync jobs (DealRoom, Crunchbase)
- Alert monitoring
- Report generation

File Handling

```json
{
  "multer": "^1.4.5-lts.1",       // File upload
  "@types/multer": "^1.4.8",
  "pdf-lib": "^1.17.1",           // PDF generation
  "pdfkit": "^0.13.0",            // Alternative PDF library
  "xlsx": "^0.18.5"               // Excel file handling
}
```

ব্যবহার:

- Pitch deck upload
- Report PDF generation
- Data export (Excel)

API Documentation

```json
{
  "swagger-jsdoc": "^6.2.8",      // Swagger documentation
  "swagger-ui-express": "^5.0.0"  // Swagger UI
}
```

ব্যবহার:

- API documentation generation
- Interactive API testing

Rate Limiting & Security

```json
{
  "express-rate-limit": "^7.0.1", // API rate limiting
  "express-mongo-sanitize": "^2.2.0", // NoSQL injection prevention
  "xss-clean": "^0.1.4"          // XSS attack prevention
}
```

ব্যবহার:

- API rate limiting
- Security attack prevention

Testing

```json
{
  "jest": "^29.7.0",             // Testing framework
  "@types/jest": "^29.5.5",
  "supertest": "^6.3.3",         // HTTP testing
  "@types/supertest": "^2.0.15",
  "ts-jest": "^29.1.1"           // TypeScript Jest support
}
```

ব্যবহার:

- Unit testing
- Integration testing
- API endpoint testing

Development Tools

```json
```

```
{
  "nodemon": "^3.0.1",           // Auto-restart server
  "ts-node": "^10.9.1",          // TypeScript execution
  "ts-node-dev": "^2.0.0",       // Development server
  "eslint": "^8.51.0",           // Linting
  "@typescript-eslint/parser": "^6.8.0",
  "@typescript-eslint/eslint-plugin": "^6.8.0",
  "prettier": "^3.0.3"           // Code formatting
}
```

ব্যবহার:

- Development environment
- Code quality
- Debugging

Notification Services

```json
{
  "nodemailer": "^6.9.6",        // Email sending
  "@types/nodemailer": "^6.4.11",
  "@slack/web-api": "^6.9.1"     // Slack integration
}
```

ব্যবহার:

- Email notifications
- Slack alerts
- Report delivery

🔄 ML Service Communication Flow

```typescript
```

```
// ml-client.service.ts example
import axios from 'axios';

const ML_SERVICE_URL = process.env.ML_SERVICE_URL || 'http://localhost:8000';

export class MLClientService {
  async scoreDeal(dealData: any) {
    const response = await axios.post(
      `${ML_SERVICE_URL}/api/v1/score_deal`,
      { deal_data: dealData },
      { timeout: 30000 }
    );
    return response.data;
  }

  async matchThesis(pitchText: string, thesisText: string) {
    const response = await axios.post(
      `${ML_SERVICE_URL}/api/v1/match_thesis`,
      { pitch_text: pitchText, thesis_text: thesisText }
    );
    return response.data;
  }

  async evaluateFounder(founderData: any) {
    const response = await axios.post(
      `${ML_SERVICE_URL}/api/v1/evaluate_founder`,
      { founder_data: founderData }
    );
    return response.data;
  }
}
```

🗄️ Database Schema Overview

Users Collection/Table

```
typescript
```

```
{
  _id: ObjectId,
  email: string,
  password_hash: string,
  name: string,
  firm_name: string,
  role: 'admin' | 'investor' | 'analyst',
  preferences: object,
  created_at: Date,
  updated_at: Date
}
```

Startups Collection/Table

typescript

```typescript
{
  _id: ObjectId,
  name: string,
  description: string,
  sector: string[],
  stage: string,
  funding_history: array,
  metrics: {
    revenue: number,
    growth_rate: number,
    burn_rate: number
  },
  team_size: number,
  founded_date: Date,
  website: string,
  pitch_deck_url: string,
  source: 'dealroom' | 'crunchbase' | 'manual',
  last_synced: Date
}
```

Scores Collection/Table

typescript

```
{
  _id: ObjectId,
  startup_id: ObjectId,
  investment_fit_score: number,
  breakdown: {
    market_score: number,
    traction_score: number,
    team_score: number,
    financial_score: number
  },
  confidence: number,
  ml_model_version: string,
  scored_at: Date
}
```

🚀 Complete Project Setup

Installation Steps

```bash
# Backend setup
cd backend
npm install

# Environment configuration
cp .env.example .env
# Edit .env with your configuration

# Database migration
npm run migrate

# Start development server
npm run dev

# Production build
npm run build
npm start
```

Environment Variables (.env)

```env
```

```
# Server
NODE_ENV=development
PORT=5000

# Database
MONGODB_URI=mongodb://localhost:27017/capital_ranker
# OR for PostgreSQL
DATABASE_URL=postgresql://user:password@localhost:5432/capital_ranker

# JWT
JWT_SECRET=your_secret_key
JWT_EXPIRE=7d
REFRESH_TOKEN_SECRET=your_refresh_secret

# ML Service
ML_SERVICE_URL=http://localhost:8000

# Redis
REDIS_HOST=localhost
REDIS_PORT=6379

# External APIs
DEALROOM_API_KEY=your_key
CRUNCHBASE_API_KEY=your_key
LINKEDIN_API_KEY=your_key

# Email
SMTP_HOST=smtp.gmail.com
SMTP_PORT=587
SMTP_USER=your_email
SMTP_PASS=your_password

# Slack
SLACK_WEBHOOK_URL=your_webhook_url
```

এই সম্পূর্ণ আর্কিটেকচার আপনার **Capital Ranker** প্রজেক্ট তৈরি করার জন্য প্রয়োজনীয় সব তথ্য প্রদান করে। প্রতিটি component এর বিস্তারিত ভূমিকা এবং interaction স্পষ্টভাবে ব্যাখ্যা করা হয়েছে।