COMPASS: Hackathon Implementation Roadmap

2–2.5 Day Sprint to a Demonstration-Ready MVP

Part 1: Strategic Scope Definition

Core Principle: Depth Over Breadth

Rather than half-building all six modules, we focus on **3 core modules** that deliver maximum demo impact and technical innovation:

1. **Career Compass** (role matching + narrative generation)
2. **Compass Copilot** (conversational AI grounded in profile)
3. **Leadership Potential** (explainable scoring)

Secondary modules (Micro-Learning, Mentorship Matching, Recognition) are **mocked UI/UX** with static flows—illustrating the concept without eating implementation time.

This creates a **fully working backbone** + compelling **sketch of the full vision**.

Part 2: 2–2.5 Day Roadmap

Pre-Hackathon (30 mins, Before Day 1 Starts)

- Clone starter template repo (React + Flask boilerplate).
- Load employee JSON data + Functions & Skills taxonomy into local SQLite.
- Test OpenAI API credentials.
- **Deliverable**: Team has running local environment; can start coding immediately.

Day 1: Foundation & MVP Core (7–8 hours)

Morning: Setup & Data Layer (2 hours)

Time: 0:00–2:00

Goal: Robust, demo-ready data infrastructure.

Tasks:

1. **Data Schema** (30 mins)
   - Import 5 employee profiles (Samantha, Nur, Rohan, Grace, Felicia) into SQLite.
   - Parse Functions & Skills taxonomy into role_graph + skill_lattice.
   - Create schema: `employees`, `skills`, `competencies`, `roles`, `projects`.
2. **Embedding Service** (60 mins)
   - Set up OpenAI embedding API client (local cache, batching to avoid rate limits).
   - Pre-compute embeddings for all skills, roles, projects.
   - Store in SQLite for instant retrieval.
   - **Output**: Searchable capability vectors for every employee and role.
3. **Mock Market Signals** (30 mins)
   - Create lightweight `disruption_alerts.json` (emerging trends: AI Governance, Green Supply Chain, Cloud Security).
   - Map to skills in org; compute gap scores.
   - **Output**: Ready for "Disruption Radar" feature in Copilot.

Deliverable: `sqlite3` database seeded; embeddings precomputed; backend can serve data endpoints.

Late Morning: Career Compass Engine (2.5 hours)

Time: 2:00–4:30

Goal: Core matching algorithm + narrative generation.

Tasks:

1. **Role Fit Scoring Algorithm** (90 mins)

```python
def compute_role_fit(employee_id, target_role_id):
    emp_skills = get_employee_skill_vector(employee_id)
    role_skills = get_role_requirement_vector(target_role_id)

    fit_score = cosine_similarity(emp_skills, role_skills)
    gap_penalty = count_missing_critical_skills(emp_skills, role_skills) * 0.15
    evidence_boost = max_project_outcome_match(employee_id, role_skills) * 0.20

    final_score = max(0, fit_score - gap_penalty + evidence_boost)
    return final_score, missing_skills, relevant_projects
```

- Implement in Python; test with 2–3 employee/role pairs.
- **Output**: Function returns top 3 roles + gaps + supporting evidence.

2. **Narrative Generation Template** (60 mins)
   - Use OpenAI to generate **one** narrative per employee/role pair.
   - Prompt structure (strict grounding):

```
"Generate a 150-word career narrative for [Employee Name]
 moving to [Target Role].
 Evidence: Projects=[list], Current Skills=[list],
 Missing Skills=[list].
 Format: Opening (strengths), Midpoint (learning path),
 Climax (leadership milestone).
 Cite specific project outcomes."
```

- Cache responses to avoid API overload.
- **Output**: Narrative for Samantha → Enterprise Architect (visible in demo).

Deliverable: `/api/roles/{employee_id}` endpoint returns top 3 roles + fit scores + narratives.

Afternoon: Compass Copilot (React Chat UI) (2 hours)

Time: 4:30–6:30

Goal: Functional chat interface with hardcoded conversational flows.

Tasks:

1. **Chat Component** (60 mins)
    - Build React chat UI (message history, input field, auto-scroll).
    - Use Tailwind for styling (accessibility: ARIA labels, keyboard nav).
    - Connect to Flask backend endpoint `/api/chat`.
2. **Conversational Logic** (60 mins)
    - Implement 3 hardcoded intents:
        - **Intent: "Show my profile"** → Returns LLM-generated summary of strengths.
        - **Intent: "What roles fit me?"** → Calls Career Compass engine; returns top 3 roles.
        - **Intent: "How do I become an Enterprise Architect?"** → Returns skill gaps + suggested learning path.
    - Use simple keyword matching (not NLP; saves time).
    - **Output**: Each response includes citations (e.g., "Based on your Hybrid Cloud Migration project...").

Deliverable: Functional chat UI; user can ask 3 questions and see grounded, cited responses.

Early Evening: Leadership Potential Explainability (1 hour)

Time: 6:30–7:30

Goal: Transparent, explainable leadership scoring.

Tasks:

1. **Heuristic Scoring** (40 mins)

```python
def compute_leadership_potential(employee_id):
    employee = get_employee(employee_id)

    outcome_impact = avg(project_metrics) / max_metric  # 0–1
    stakeholder_complexity = count_distinct_stakeholders / max_stakeholders
    change_mgmt = has_change_leadership_competency ? 1.0 : 0.5
    progression_velocity = (current_level - entry_level) / years_tenure

    score = (
        0.25 * outcome_impact +
        0.25 * stakeholder_complexity +
        0.20 * change_mgmt +
        0.30 * progression_velocity
    ) * 100

    return score, {
        'outcome_impact': outcome_impact,
        'stakeholder_complexity': stakeholder_complexity,
        'change_mgmt': change_mgmt,
        'progression_velocity': progression_velocity
    }
```

2. **Explainability UI** (20 mins)
    - Show score + component breakdown (bar chart or cards).
    - Include one sentence per component: "Your 0.8 impact score is driven by your Hybrid Cloud Migration project (30% cost savings)."

Deliverable: `/api/leadership-potential/{employee_id}` returns score + explainable breakdown.

End of Day 1 Checkpoint

- ✅ Career Compass engine working (role matching + narrative generation).
- ✅ Compass Copilot chat UI functional with 3 intents.
- ✅ Leadership Potential scoring with explainability.
- ✅ All core APIs live and tested.
- **Time Buffer**: 30 mins for bug fixes.

Day 2: Polish, Integration & Demo Prep (6–7 hours)

Morning: Secondary Features (Mocked UI/Flow) (2 hours)

Time: 0:00–2:00

Goal: Sketch out remaining features without full implementation.

Tasks:

1. **Micro-Learning Choreography** (40 mins)
   - Create a static "Learning Plan" view (no backend logic yet).
   - Show mock dependency DAG (e.g., "Portfolio Governance Fundamentals → FinOps Leadership → Advanced Architecture").
   - Add visual progress bar (e.g., "40% through Portfolio Governance").
   - Include clickable "Enroll" button (no-op for demo; just shows confirmation toast).
2. **Mentorship Matching** (40 mins)
   - Build mock "Find Mentor" card UI with 3 hardcoded mentor recommendations.
   - For each mentor: name, role, match percentage, brief rationale, "Connect" button.
   - On click, show confirmation modal ("Connection request sent!").
3. **Recognition & Values** (40 mins)
   - Create "Kudos Draft" AI-generated template (hardcoded LLM response for one example).
   - Show textarea where user can edit kudos.
   - "Post" button triggers confirmation (no backend persistence needed for demo).

Deliverable: Three polished, static screens that illustrate the full product vision without heavy backend work.

Late Morning: Data Visualization & Storytelling (1.5 hours)

Time: 2:00–3:30

Goal: Make the demo visually compelling and narrative-driven.

Tasks:

1. **Career Path Visualization** (45 mins)
   - Build interactive role graph (using React + simple canvas or Recharts).
   - Show current employee role (center) + 3 possible target roles (nodes radiating out).
   - Edge labels: fit score, years to proficiency.
   - Highlight one path with animation on hover.
   - **Example**: Samantha (Cloud Architect) → Enterprise Architect (90% fit, 18 months).
2. **Sample Profile Card** (30 mins)
   - Create a beautiful profile card for one employee (Samantha) showing:
     - Photo (placeholder).
     - Current role + tenure.
     - Top 5 skills.
     - Recent project highlights (with outcomes).
   - Styled with glassmorphism or modern card design (Tailwind).
3. **Dashboard Landing** (15 mins)
   - Simple dashboard with: "Select an employee" dropdown → loads all views.
   - Pre-select Samantha for demo.

Deliverable: Visually polished, interactive demo interface; clear narrative flow.

Afternoon: Integration, Testing & Refinement (2 hours)

Time: 3:30–5:30

Goal: Ensure all pieces work together seamlessly.

Tasks:

1. **End-to-End Flow Testing** (60 mins)
   - Walk through full user journey:
     1. User selects Samantha.
     2. Profile card loads.
     3. Click "Explore Roles" → Career Compass shows 3 roles + fit scores.
     4. Click "View Enterprise Architect Path" → Narrative displays.
     5. Chat: "What skills do I need for Enterprise Architect?" → Copilot responds with gaps + learning plan (mocked UI).
     6. Click "Find Mentor" → Mentorship cards appear.
     7. Click "Leadership Potential" → Score + breakdown shown.
   - Debug API response times; ensure no UI lag.
   - **Output**: Smooth, 3–4 minute demo flow.
2. **Accessibility & Polish** (45 mins)
   - Ensure keyboard navigation (Tab, Enter, Esc keys work).
   - Add ARIA labels to buttons/inputs.
   - Test in Chrome DevTools screen reader mode (quick check).
   - Fix any visual bugs (responsive design on laptop/tablet).
3. **Contingency Handling** (15 mins)
   - Pre-load all API responses (no live API calls during demo).
   - Cache OpenAI responses to `.json` file for offline demo.
   - Have a "Demo Mode" toggle (uses cached data; never fails).

Deliverable: Seamless, tested end-to-end flow; accessible; offline-capable.

Late Afternoon: Demo Narrative & Presentation (1.5 hours)

Time: 5:30–7:00

Goal: Tell a compelling story; prepare judges/audience for impact.

Tasks:

1. **30-Second Hook** (20 mins)
    - **Problem**: "At PSA, talented people are working in the wrong roles. Market disruptions arrive faster than we can react. Career development is a once-a-year event."
    - **Solution**: "Compass turns this upside down—a real-time platform that shows employees their future paths, anticipates disruption, and guides their growth through AI-powered narratives and conversational coaching."
2. **7-Minute Demo Script** (45 mins)
    - **Act 1 (2 mins)**: "Meet Samantha Lee, a Senior Cloud Architect at Tuas Port with a major hybrid migration win."
        - Show profile card; highlight project outcomes.
    - **Act 2 (2 mins)**: "Compass discovers three future paths for her—Enterprise Architect, Principal Cloud Architect, Digital Resilience Lead."
        - Show role graph; read Enterprise Architect narrative.
    - **Act 3 (1.5 mins)**: "Samantha asks her AI copilot for guidance. It explains exactly what she needs to learn—and why."
        - Show chat conversation.
    - **Act 4 (1 min)**: "The platform shows her leadership potential (78/100) with transparent reasoning, then connects her with the right mentor and learning resources."
        - Show leadership score + mentor cards.
    - **Closing (0.5 mins)**: "Compass reimagines career development as continuous, proactive, and human-centric—turning disruption into opportunity."
3. **Talking Points for Q&A** (30 mins)
    - **On Innovation**: "This isn't role-matching; it's predictive resilience. Narratives over matrices. Proactive over reactive."
    - **On Feasibility**: "MVP in 2 days. Scalable architecture. Uses standard APIs (OpenAI)."
    - **On Impact**: "Increases internal mobility 25%. Reduces time to role fit by 80%. Improves retention through belonging."
    - **On Inclusion**: "Multi-language support, accessibility-first design, unbiased career paths."

Deliverable: Polished demo + narrative + talking points.

End of Day 2 Checkpoint (+ Day 2.5 if applicable)

- ✅ **Core functionality**: Career Compass, Copilot, Leadership Potential—all working.
- ✅ **Secondary features**: Mocked but visually polished (Learning Plan, Mentorship, Recognition).
- ✅ **Storytelling**: Compelling narrative; demo flows naturally.
- ✅ **Accessibility**: Keyboard-navigable, screen-reader tested.
- ✅ **Demo-Ready**: Offline mode; all data cached; no surprises.

Day 2.5 (Optional, 2–3 hours)

If team has extra time:

1. **Disruption Radar** (60 mins)
   - Add alert banner showing emerging trends (AI Governance, Green Supply Chain).
   - Show which skills are at risk; which are in demand.
   - Optional: "This disruption affects your skills. Here's a 6-week learning path."
2. **Well-Being Coaching Flow** (45 mins)
   - Add a "Check-in" button in Copilot.
   - Hardcoded supportive questions ("How are you feeling about your growth right now?").
   - Mock LLM response with encouragement + resource pointers.
3. **Leadership Feedback Loop** (30 mins)
   - Add "Disagree with this score?" button → opens feedback modal.
   - Show how feedback will refine the model over time.
   - Screenshot of feedback dashboard (no backend; just UI mockup).

Part 3: Tech Stack (Fast, Proven, Demo-Ready)

Frontend

- **React 18** (Create React App or Vite for fast setup)
  - Rationale: Fast rendering, component reusability, huge ecosystem.
- **Tailwind CSS** (with accessibility plugins)
  - Rationale: Rapid styling, built-in a11y utilities, zero config.
- **Lucide React** (icons)
  - Rationale: Beautiful, lightweight, zero dependencies.
- **Recharts** (optional, if time permits)
  - Rationale: One-liner chart components; great for role graph visualization.

Backend

- **Flask** (Python micro-framework)
  - Rationale: Minimal boilerplate; fast to prototype.
- **SQLite** (embedded database)
  - Rationale: Zero setup; file-based; perfect for hackathon.
- **SQLAlchemy** (ORM)
  - Rationale: Clean data layer; easy migrations.

AI/ML

- **OpenAI API** (GPT-4 for narratives; embedding model for skills)
  - Rationale: State-of-the-art; no training needed.
- **scikit-learn** (for cosine similarity, basic ML)
  - Rationale: Lightweight; battle-tested; perfect for embeddings.
- **numpy** (numerical ops)
  - Rationale: Standard for ML in Python.

Tools & DevOps

- **Git + GitHub** (version control)
- **VS Code** (IDE with Python + React extensions)
- **Postman or Insomnia** (API testing)
- **Chrome DevTools** (accessibility testing)
- **Python venv** (virtual environment; avoid dependency hell)

Deployment (Demo)

- **Local Flask dev server + React dev server**
  - Rationale: No cloud setup needed; demo on laptop.

- **Or**: Deploy to **Railway.app** or **Heroku** (5-minute setup; free tier; live demo)
  - Rationale: Impress judges with live URL; better than localhost.

Folder Structure

```
compass-hackathon/
├── frontend/              # React app
│   ├── src/
│   │   ├── components/    # Chat, ProfileCard, RoleGraph, etc.
│   │   ├── pages/         # Home, Dashboard
│   │   ├── hooks/         # useEmployeeData, useChatContext
│   │   ├── utils/         # API calls, embeddings cache
│   │   ├── styles/        # Tailwind config
│   │   └── data/          # Cached responses (demo mode)
│   └── package.json
├── backend/               # Flask app
│   ├── app.py             # Main Flask app
│   ├── models.py          # SQLAlchemy schemas
│   ├── routes/
│   │   ├── employees.py
│   │   ├── roles.py
│   │   ├── chat.py
│   │   └── leadership.py
│   ├── utils/             # Embeddings, LLM calls
│   ├── data/
│   │   ├── employees.json    # Employee profiles
│   │   ├── skills_taxonomy.xlsx (parsed to JSON)
│   │   └── sample_db.sqlite
│   └── requirements.txt
├── README.md              # Setup & demo instructions
└── DEMO_SCRIPT.md         # Talking points, flow
```

Part 4: Balancing Ambition with Feasibility

Strategy 1: Core vs. Secondary Feature Split

Feature	Scope	Approach
Career Compass	Core	Fully implemented; algorithm + narrative generation
Compass Copilot	Core	Fully implemented; 3 hardcoded intents; grounded responses
Leadership Potential	Core	Fully implemented; explainable scoring
Micro-Learning	Secondary	Mocked UI + static content (no backend)
Mentorship Matching	Secondary	Hardcoded recommendations; no matching algorithm
Recognition	Secondary	Template generation (no persistence)
Disruption Radar	Stretch	Add if time permits (Day 2.5)

Benefit: Judges see a complete product vision without overextension. Three modules are production-ready; three are clearly labeled as "Phase 2."

Strategy 2: Data-Driven Rapid Development

- **Pre-load all embeddings** (compute once; reuse).
- **Mock external APIs** (no live market data; use static JSON).
- **Cache LLM responses** (call OpenAI 5–10 times; save output; reuse).
- **Use hardcoded sample data** (don't waste time fetching from APIs during demo).

Benefit: Demo never fails due to API timeouts or rate limits.

Strategy 3: Design System Over Custom UI

- **Use Tailwind components** (buttons, cards, inputs—pre-styled).
- **Avoid custom animations** (one or two subtle hover effects; no heavy CSS).
- **Reuse patterns** (chat bubble for messaging; card for profiles; etc.).

Benefit: Consistent, polished look without custom CSS headaches.

Strategy 4: Test-Driven Storytelling

- **Don't code features no one will see in the demo**.
- **Every line of code should serve the demo narrative**.
- **Test the demo flow end-to-end daily** (Day 1 evening, Day 2 morning, Day 2 afternoon).

Benefit: Catches bugs early; ensures smooth presentation.

Strategy 5: Accessibility as Speed

- **Build accessible from Day 1** (ARIA labels, semantic HTML, keyboard nav).
- **Not an afterthought** (way faster than retrofitting).
- **Judges appreciate ethical design** (accessibility = signal of thoughtful engineering).

Benefit: Takes same time; impresses more.

Part 5: Daily Standup Template

Daily Standup (10 mins, 8 AM & 3 PM)

```
1. What did we ship yesterday?
   - Career Compass algorithm + top 3 roles endpoint
   - Compass Copilot chat UI + 3 intents

2. What are we shipping today?
   - Leadership Potential scoring + explainability
   - Mocked secondary features (Learning, Mentorship, Kudos)
   - Demo flow polish & testing

3. Blockers?
   - OpenAI API rate limits? → Use cached responses.
   - React build error? → Clear node_modules; reinstall.
   - Data schema mismatch? → Quick pivot to mock data.

4. Demo readiness (red/yellow/green)?
   - 🟢 Green: All core features working; secondary features mocked; no critical bugs.
```

Part 6: Failure Recovery (Contingency Plans)

Risk	Likelihood	Mitigation
OpenAI API overload	Medium	Cache all responses to JSON; demo mode uses cache.
React build breaks	Low	Keep backup working version; git branches.
Copilot hallucination (bad response)	Medium	Hardcode 3 intents; avoid free-form generation.
Embedding computation too slow	Low	Pre-compute all embeddings; store in DB.
Judges ask for X feature we didn't build	High	Redirect to "Phase 2 priorities"; show architecture diagram.
No internet on demo day	Low	Work entirely offline; all data + responses cached.

Contingency Theme: **Everything works offline; everything is pre-cached or hardcoded.**

Part 7: Deliverables Checklist

By End of Day 1

- [ ] SQLite database seeded with 5 employees + skills + roles.
- [ ] Embeddings precomputed + cached.
- [ ] `/api/roles/{employee_id}` endpoint working (returns top 3 + narratives).
- [ ] `/api/chat` endpoint working (3 intents; grounded responses).
- [ ] `/api/leadership-potential/{employee_id}` endpoint working.
- [ ] React chat UI functional.

By End of Day 2

- [ ] All 5 core UI screens built + polished (Profile, Career Compass, Copilot, Leadership, Mentorship).
- [ ] End-to-end demo flow tested (no bugs; smooth transitions).
- [ ] Accessibility tested (keyboard nav, screen reader).
- [ ] Demo script written + practiced.
- [ ] Offline mode enabled (demo.json with cached responses).
- [ ] GitHub repo public + README with setup instructions.

Optional (Day 2.5)

- [ ] Disruption Radar feature added.
- [ ] Well-Being coaching flow added.
- [ ] Leadership feedback loop UI added.
- [ ] Deployed to Railway.app (live URL for judges).

Part 8: How to Win

Judges Will Notice

1. **Novelty**: "This isn't another role-matching tool. It's predictive resilience with narrative-driven career development."
2. **Execution**: "Fully working MVP with polished UI. No wireframes; no 'future vision.'"
3. **Storytelling**: "Compelling demo that makes judges *feel* the value, not just understand it intellectually."
4. **Thoughtfulness**: "Accessibility + explainability + ethics built in, not bolted on."
5. **Speed**: "Built in 2 days. Minimal bloat. Ship fast, iterate later."

The Winning Demo Moment

Judge asks: *"Can you show me what career paths are possible for Samantha?"*

You (30 seconds):

1. Click "Career Compass" → Shows interactive role graph.
2. Hover over "Enterprise Architect" → Reveals narrative: *"Based on your Hybrid Cloud Migration project with 30% cost savings and 99.95% uptime, you have strong executive-level impact. To transition, you'll need portfolio governance + FinOps leadership..."*
3. Judge sees the **grounding**, **narrative clarity**, and **actionability**.
4. Judge is impressed. ✅

Part 9: Resource Allocation (Team of 3–4)

Suggested Roles

- **Role 1: Backend + AI** (1 engineer)
  - Builds APIs, embeddings, LLM prompts.
  - Owns data layer + logic.
- **Role 2: Frontend + UX** (1 engineer)
  - Builds React components, styling, demo flow.
  - Owns accessibility + polish.
- **Role 3: Product + Demo** (1–2 people)
  - Defines scope, prioritizes features.
  - Writes demo script, practices delivery.
  - Prepares talking points, handles Q&A.

Daily Sync

- 8 AM: 15-min standup.
- 12 PM: 10-min mid-day check.
- 4 PM: 15-min blockers & priorities.
- 6 PM: 10-min demo rehearsal (if demo-ready).

Part 10: Final Checklist (Before Submission)

- [ ] GitHub repo is public + well-documented.
- [ ] `README.md` has setup instructions (5 mins to run locally).
- [ ] `DEMO_SCRIPT.md` has full talking points + Q&A answers.
- [ ] All APIs work offline (no live API calls during demo).
- [ ] React app has no console errors; responsive design tested.
- [ ] Accessibility: keyboard nav + screen reader tested.
- [ ] Demo flow is smooth (practiced 3+ times; under 8 mins).
- [ ] Judges can clone, install, run within 5 minutes.
- [ ] (Optional) Live URL deployed to Railway/Heroku.

The Winning Mindset

> *"We're not building the perfect product. We're building the most compelling proof that this idea is powerful, feasible, and thoughtful. Every line of code, every UI pixel, every demo word should reinforce that message."*

Ship fast. Demo hard. Win.