🗓️ 8-WEEK MVP ROADMAP (QuickScholar)

Each week has:

- 🎯 Weekly Deliverable (what must exist by end of week)
- 📚 What to Study
- 🛠️ What to Build
- ⏱️ Daily 3-Hour Workflow

✅ WEEK 1 — Project Setup + Backend Foundations

🎯 Deliverable

- GitHub repo created
- FastAPI or Flask backend running
- Basic `/search` endpoint working (returns dummy data)
- Project structure ready

📚 Study

- Flask or FastAPI basics
- REST APIs (GET/POST, JSON responses)
- Python project structure

- Virtual environments & pip
- Git + GitHub basics

🛠️ Build

- Create backend folder structure
- Setup virtual environment
- Create simple API:
    - `/health`
    - `/search?q=...` (returns fake results for now)
- Test with browser / Postman

⏱️ Daily Workflow (3h)

- 45 min: Learn (tutorial/docs)
- 1h30: Code backend setup
- 45 min: Debug + commit to GitHub

✅ WEEK 2 — First Real Data Source (Semantic Scholar)

🎯 Deliverable

- `/search` endpoint fetches **real data** from Semantic Scholar API
- Returns: title, abstract, authors, year, link

📚 Study

- HTTP requests in Python (`requests` or `httpx`)
- Working with JSON APIs
- Error handling (try/except)
- Reading API docs

🛠️ Build

- Integrate Semantic Scholar API
- Parse response into your data model
- Return clean JSON to frontend
- Handle:
    - No results
    - API errors

⏱️ Daily Workflow

- 45 min: Read API docs + learn requests
- 1h30: Implement + test API integration
- 45 min: Clean code + logging + commit

✅ WEEK 3 — Add Second Source + Normalization

🎯 Deliverable

- Search pulls from **2 sources** (e.g., Semantic Scholar + CrossRef)
- Results are **merged and normalized** into one format

📚 Study

- Working with multiple APIs
- Data normalization
- Basic ranking / sorting
- Python lists, dicts, sorting

🛠️ Build

- Add CrossRef API integration

Create a common result format:

```
{
  title, summary, authors, year, source, link
}
```

-
- Merge results
- Sort by relevance or year

⏱️ Daily Workflow

- 30 min: Learn
- 2h: Code integration + merge logic
- 30 min: Test + refactor

✅ WEEK 4 — Filters + Performance Basics

🎯 Deliverable

- Filters working:
    - By source
    - By year
- Backend handles:
    - `?source=...`
    - `?year=...`

📚 Study

- Query parameters in APIs
- Async basics (`async` / `await`) OR simple optimization
- Caching basics (optional)

🛠️ Build

- Add filter logic in backend
- Update `/search` endpoint to accept filters
- Improve error handling
- Add simple in-memory cache (optional)

⏱️ Daily Workflow

- 30 min: Learn
- 2h: Implement filters + test
- 30 min: Clean + commit

✅ WEEK 5 — Save & Export (MVP Storage)

🎯 Deliverable

- User can:
    - Save results
    - Export to CSV / JSON
- SQLite database connected

📚 Study

- SQLite with Python
- Basic SQL or ORM (SQLAlchemy)
- File export in Python (csv, json)

🛠️ Build

- Create DB schema:
    - saved_results table
- Add endpoints:
    - `/save`

    - `/export`

- Implement CSV + JSON export

⏱️ Daily Workflow

- 45 min: Learn DB basics
- 1h30: Implement save/export
- 45 min: Test + fix bugs

✅ WEEK 6 — Simple Web UI (MVP Frontend)

🎯 Deliverable

- Web page with:
    - Search box
    - Results list
    - Filter options
    - Save / Export buttons

📚 Study

- HTML + basic CSS
- Fetch API (JS) OR server-side templates
- Connecting frontend to backend API

🛠️ Build

- Create simple UI page
- Connect search form to backend
- Display results dynamically
- Add filter UI

⏱️ Daily Workflow

- 45 min: Learn frontend basics
- 1h30: Build UI + connect API
- 45 min: Fix layout + bugs

✅ WEEK 7 — Testing + Stability

🎯 Deliverable

- Core flows tested:
    - Search
    - Filter
    - Save
    - Export
- No crashes in main flow
- Basic unit tests added

📚 Study

- pytest basics
- Writing simple tests
- Logging in Python

🛠️ Build

- Add tests for:
    - API calls
    - Parsing logic
    - Filters
- Manual testing:
    - Act like a real user
- Fix bugs

⏱️ Daily Workflow

- 30 min: Learn testing
- 2h: Write tests + fix bugs
- 30 min: Refactor + commit

✅ WEEK 8 — MVP Freeze + Deployment Prep

🎯 Deliverable

- MVP feature-complete:

    - Search
    - Multi-source
    - Filters
    - Save
    - Export
    - Web UI
- Deployed or ready to deploy
- README written

📚 Study

- Deployment basics (Render / Fly.io / Heroku)
- Environment variables
- Production vs dev config

🛠️ Build

- Clean project
- Add README
- Add .env config
- Deploy backend + frontend
- Final bug fixes

⏱️ Daily Workflow

- 30 min: Learn deployment
- 2h: Deploy + fix issues
- 30 min: Docs + cleanup