**Project Plan: Funda Price Range Prediction App**

**Phase 1: Define & Prep (1 week)**

**Goals:**

* Clarify project scope and requirements **✓**
* Gather initial data (scraping plan or static dataset) **✓**
* Set up environment and tools **✓**

**Tasks:**

* Finalize which Funda URLs you will scrape (e.g., /detail/ pages) **✓**
* Inspect Funda page structure, identify data fields to extract **✓**
* Decide tech stack: Python (requests + BeautifulSoup or Scrapy), FastAPI, Streamlit/React **✓**
* Set up version control (GitHub repo), environment (virtualenv/conda), and CI basics **✓**

**Phase 2: Data Collection & Cleaning (1–2 weeks)**

**Goals:**

* Build a robust scraper that respects robots.txt and rate limits **✓**
* Collect a representative dataset of listings (target 1000+ samples)
* Clean and preprocess the data for modelling

**Tasks:**

* Develop scraper with error handling, logging, retries, user-agent rotation
* Save scraped data in CSV or database (SQLite/Postgres)
* Explore and clean data: handle missing values, normalize formats, create features
* Conduct exploratory data analysis (EDA) to understand distributions and correlations

**Phase 3: Modeling & Evaluation (1–2 weeks)**

**Goals:**

* Train and validate predictive models to estimate asking price range
* Experiment with features, try different regression algorithms
* Evaluate model performance and select best approach

**Tasks:**

* Feature engineering (size, location encoding, rooms, energy labels, etc.)
* Try models: Linear Regression, Random Forest, Gradient Boosting, maybe LightGBM/XGBoost
* Use cross-validation, tune hyperparameters
* Define price range prediction (e.g., predict mean ± confidence interval)
* Document model results and interpretation

**Phase 4: API & Backend (1 week)**

**Goals:**

* Build a REST API backend to serve predictions
* Connect scraper and model inference code to API

**Tasks:**

* Develop FastAPI backend with endpoints:
  + /predict — input listing URL or features, output price range
  + /scrape — optionally scrape on-demand or pre-scrape in batch
* Implement input validation and error handling
* Containerize backend with Docker for easy deployment

**Phase 5: Frontend & User Interface (1 week)**

**Goals:**

* Build an easy-to-use frontend for users to input URLs or property info
* Display predictions and comparison with actual asking price

**Tasks:**

* Create frontend with Streamlit or React
* Connect frontend to backend API
* Add UI elements: input box, buttons, result display, loading indicators
* Improve UX: error messages, guidance, data visualizations (price distributions, etc.)

**Phase 6: Testing, Logging & Deployment (1 week)**

**Goals:**

* Make the app production-ready and reliable
* Deploy app on cloud platform or VPS

**Tasks:**

* Write unit and integration tests for scraper, model, API
* Add logging to monitor scraping and prediction errors
* Deploy backend and frontend (e.g., Heroku, AWS, Google Cloud, or DigitalOcean)
* Set up CI/CD pipeline if desired
* Configure domain, SSL, and basic monitoring

**Phase 7: Documentation & Portfolio Prep (1 week)**

**Goals:**

* Document your work clearly and professionally
* Prepare materials for job hunting and interviews

**Tasks:**

* Write README with project overview, setup, usage, results
* Prepare a blog post or slide deck explaining approach and learnings
* Create a short demo video or live demo link
* Polish GitHub repo (code comments, notebooks, data samples)

**Adjusted Timeline at 2 Hours/Day (10 Hours/Week)**

| **Phase** | **Hours Estimated** | **Weeks Needed (10 hrs/wk)** |
| --- | --- | --- |
| **Define & Prep** | **10–15** | **1–1.5** |
| **Data Collection & Cleaning** | **25–40** | **2.5–4** |
| **Modeling & Evaluation** | **25–40** | **2.5–4** |
| **API & Backend** | **15–20** | **1.5–2** |
| **Frontend & UI** | **15–20** | **1.5–2** |
| **Testing & Deployment** | **15–20** | **1.5–2** |
| **Documentation & Portfolio** | **10–15** | **1–1.5** |
| **Total** | **115–170** | **11.5–17** |

**Bonus Tips**

* **Start small and iterate**: build a minimal viable version quickly, then improve.
* **Use version control religiously** to track changes.
* **Automate scraping schedules** to refresh data regularly.
* **Focus on explaining your trade-offs** (e.g., why you chose a model or tech).