INTERNSHIP PROJECT PHASE (2 WEEKS)

PROJECT GUIDELINES

- **Choose any 1 or 2 projects from the given list.**
- **You are free to improvise — take the given project as a base and modify it as you like.**
- **You can use any tools, technologies, or steps you're comfortable with — there are no restrictions.**
- **Focus and work sincerely so that you have complete clarity and can explain the project confidently in interviews.**
- **Go through the Top 50 Interview Questions for your domain (attached at the end).**
- **Update your project status regularly when the Google Form is shared in group.**
- **while working on the project YOU CAN CHOOSE ANY DATASET RELAVENT TO THE PROJECT.**

.

After project completion, prepare a 1–2 page report in PDF format, containing:

- Introduction
- Abstract
- Tools Used
- Steps Involved in Building the Project
- Conclusion

◆ Note: Report must not exceed 2 pages.

DEAR INTERNS,

YOU HAVE TO UPDATE STATUS OF YOUR PROJECT EVERY 3 OR 4 DAYS ONCE WHEN THE UPDATION LINK IS SHARED IN THE GROUP.

Final Project Submission Date and Guidelines :

25 July 2025: Submission of your final project GitHub repository link with all deliverables and the project report.

If you are doing more than one project put all projects in same repository and prepare report for any one project

Final submission links will be shared later.

! READ ALL THE GUIDELINES CAREFULLY !

LIST OF PROJECTS

1. Smart Resume Parser

Objective: Extract structured info (skills, education, experience) from PDF/DOCX resumes.

Tools: Python, spaCy, PyMuPDF, docx, Streamlit

Mini Guide:

1. **Use PyMuPDF or docx to extract text.**
2. **Clean and preprocess the text.**
3. **Use spaCy + regex to extract sections like skills and experience.**
4. **Organize output into JSON or table format.**
5. **Build a UI to upload and view results.**
6. **Export results to CSV or JSON.**

Deliverables: Codebase, UI app, 5 test resumes, output files.

2. Voice-Controlled Assistant

Objective: Create an assistant that performs actions via voice commands.

Tools: speech_recognition, pyttsx3, os, webbrowser

Mini Guide:

1. **Capture voice input using speech_recognition.**
2. **Convert voice to text.**
3. **Map commands to system tasks (open app, speak time).**
4. **Add response with pyttsx3.**
5. **Implement a listening loop and exit command.**
6. **Include modules like jokes, news, or notes.**

Deliverables: Script, demo video, list of supported commands.

3. AI Code Reviewer

Objective: Analyze and improve code quality automatically.

Tools: flake8, black, radon, Streamlit

Mini Guide:

1. **Accept code input via UI or upload.**
2. **Analyze style using flake8 and black.**
3. **Measure complexity with radon.**
4. **Summarize improvements needed.**
5. **Display before/after suggestions.**
6. **Export a report.**

Deliverables: Streamlit app, sample code analysis report.

4.Crypto Price Tracker

Objective: Track live cryptocurrency prices with alerts.

Tools: requests, CoinGecko API, Streamlit, smtplib

Mini Guide:

1. Fetch prices using CoinGecko API.
2. Display prices in a table.
3. Add user-defined alert thresholds.
4. Send email when thresholds are met.
5. Auto-refresh prices periodically.
6. Visualize trends with line charts.

Deliverables: Script, price alert log, sample alert email.

5. PDF to Audiobook Converter

Objective: Convert PDF content into spoken audio.

Tools: PyMuPDF, pyttsx3/gTTS, Tkinter

Mini Guide:

1. Load and extract text from PDF.
2. Clean text and handle empty pages.
3. Convert text to speech.
4. Play or export to MP3.
5. Create file upload UI.
6. Add control options (volume, speed).

Deliverables: GUI app, sample PDFs + audio files.

6.Stock Analysis Dashboard

Objective: Analyze and visualize stock trends with indicators.

Tools: yfinance, plotly, Streamlit

Mini Guide:

1. Accept stock ticker input.
2. Fetch historical data with yfinance.
3. Calculate SMA, EMA, RSI.
4. Plot candlestick and line graphs.
5. Display performance summary.
6. Add CSV export option.

Deliverables: Dashboard, analysis reports, screenshots.

7. Python Chat App (LAN)

Objective: Create a local real-time chat app.

Tools: socket, threading, Tkinter

Mini Guide:

1. Set up socket server.
2. Allow multiple clients using threads.
3. Create chat input/output UI.
4. Handle join/leave events.
5. Save logs to file.
6. Add command support (/exit, /mute).

Deliverables: Code for server/client, 2-user test, chat logs.

8. Excel Report Generator

Objective: Generate Excel summaries from CSV inputs.

Tools: pandas, openpyxl, Tkinter

Mini Guide:

1. Load CSV into pandas.
2. Create pivot tables.
3. Generate charts with matplotlib.
4. Export styled Excel using openpyxl.
5. Add file upload and save dialogs.
6. Add summary stats.

Deliverables: Excel file, sample CSVs, screenshots.

9.. Real-Time Weather App

Objective: Display current and forecast weather for any city.

Tools: OpenWeatherMap API, requests, Streamlit

Mini Guide:

1. Create input for city names.
2. Fetch data from OpenWeatherMap.
3. Show temp, humidity, sunrise/sunset.
4. Add 5-day forecast chart.
5. Add dynamic icons (rainy, cloudy).
6. Allow unit toggle (C/F).

Deliverables: Weather app, forecast chart, sample queries.

10.Fake News Detector

Objective: Detect if a news headline is fake or real.

Tools: scikit-learn, pandas, TfidfVectorizer

Mini Guide:

1. Load and clean dataset (real vs fake).
2. Convert text to vectors using TF-IDF.
3. Train ML model (Logistic Regression or SVM).
4. Build UI to input headlines.
5. Show prediction + confidence.
6. Allow CSV upload for batch check.

Deliverables: Streamlit app, model file, test results.

11. Expense Tracker with Visuals

Objective: Track expenses and visualize insights.

Tools: pandas, matplotlib, Tkinter or Streamlit

Mini Guide:

1. Create input form or CSV upload.
2. Categorize and clean data.
3. Group by category/date.
4. Generate visuals (pie/bar).
5. Show budget alerts.
6. Export reports to Excel.

Deliverables: Expense dashboard, charts, test data.

12. Bookstore REST API

Objective: Manage inventory of books via API.

Tools: Flask/FastAPI, SQLite, Postman

Mini Guide:

1. Define database schema (books).
2. Set up API routes (CRUD).
3. Add search/filtering options.
4. Connect to SQLite with SQLAlchemy.
5. Handle errors + validations.
6. Document with Swagger/Postman.

Deliverables: API script, Postman collection, DB file.

13. LinkedIn Job Scraper

Objective: Automate job listings scraping from LinkedIn.

Tools: Selenium, BeautifulSoup, pandas

Mini Guide:

1. Set up Selenium + login.
2. Search jobs with keywords/locations.
3. Scrape job titles, company, post date.
4. Save to CSV.
5. Add duplicate removal.
6. Visualize job frequency by company.

Deliverables: Bot script, CSV data, company/job graph.

14. Product Recommendation System

Objective: Recommend products based on user input.

Tools: pandas, scikit-learn, Streamlit

Mini Guide:

1. Load product + user rating data.
2. Preprocess and vectorize data.
3. Build collaborative + content filters.
4. Input product name to get suggestions.
5. Show thumbnails/details in UI.
6. Export recommendations to CSV.

Deliverables: Recommender app, test data, report file.

15. Test Coverage Dashboard

Objective: Analyze codebase test coverage visually.

Tools: coverage.py, pytest, matplotlib

Mini Guide:

1. Clone Python repo and write tests.
2. Run coverage and generate report.
3. Parse data into usable format.
4. Plot visuals (coverage % by file).
5. Highlight uncovered files.
6. Display results in Flask/Streamlit UI.

Deliverables: Coverage reports, visual dashboard, sample repo.

TOP 50 INTERVIEW QUESTIONS FOR PYTHON

1. What are Python's key features?
2. Explain the difference between is and == operators.
3. What is the use of self in Python classes?
4. What is a Python decorator and where can it be used?
5. What are Python's data types? Give examples.
6. What is the difference between a list, tuple, and set?
7. What is list comprehension? Give an example.
8. How does Python manage memory?
9. What is the difference between shallow copy and deep copy?
10. Explain method overloading and overriding in Python.
11. What are *args and **kwargs?
12. What are lambda functions?
13. What is the difference between @staticmethod and @classmethod?
14. How does exception handling work in Python?
15. What is a generator? How is it different from a normal function?
16. How do you read and write a CSV file in Python?
17. How can you parse and manipulate Excel files using Python?
18. What is openpyxl used for?
19. How would you extract text from a PDF file?
20. How do you handle large files efficiently in Python?
21. How do you validate file types before uploading?
22. What are some common libraries for handling Excel in Python?
23. How can you merge multiple CSVs into one file?
24. How do you generate reports from data using Python?
25. How would you automate a PDF-to-audio converter in Python?
26. How do you call an API using Python?
27. What is the difference between GET, POST, PUT, and DELETE in REST APIs?
28. How do you build a REST API using Flask or FastAPI?
29. What is the role of Postman in API development?
30. How do you handle API authentication (e.g., token)?
31. What are the benefits of using FastAPI over Flask?
32. What is Selenium and how does it work?
33. How would you use BeautifulSoup for web scraping?
34. How do you automate sending an email in Python?
35. What is the difference between synchronous and asynchronous API calls?
36. How do you clean a dataset using pandas?
37. What is TF-IDF and where is it used?
38. How does a recommendation engine work?
39. How do you evaluate a classification model?
40. What is cosine similarity? Where have you used it?
41. What is the use of the TfidfVectorizer?
42. How would you classify a news headline as fake or real?
43. How do you serialize a model in Python?
44. What is the purpose of pytest and unittest?
45. How do you measure test coverage in Python?
46. What is coverage.py and how do you use it?
47. How do you use Git in a collaborative project?
48. How would you deploy a Python project on Streamlit Cloud?
49. What's the difference between a virtual environment and Docker?
50. How do you schedule a Python script to run daily?

Elevate Labs