TripGain - Python & AI Developer Assessment (Round 2)

Duration: 2 Hours
Total Marks: 100

Instructions

The assessment consists of **three compulsory sections**:

Section	Topic	Marks
1	Pandas Analysis	**25**
2	Playwright Web Automation	**45**
3	Gemini Integration	**30**

Each section must be completed within the total duration.
Partial credit will be awarded for correctly implemented sub-tasks.

Evaluation Criteria

Performance will be evaluated based on the following parameters:

- **Code functionality and correctness**
- **Logic and problem-solving approach**
- **Code structure, readability, and appropriate commenting**
- **Prompt design and reasoning quality** *(for Section 3 – Gemini Integration)*

Candidates are expected to work **independently** and maintain **professional coding standards** throughout the assessment.
Final selection will be based on **overall performance and total marks**.

Submission Instructions

All submissions must be made via the official GitHub repository:

🔗 **Repository Link:** https://github.com/nikhil-swamix/tripgain-python-interview

Follow the steps below carefully:

1. **Fork** the repository to your personal GitHub account.
2. **Create a new branch** using your **email prefix** (e.g., stevejobs72@gmail.com , branch name stevejobs72).
3. Add your **solution files** for each section inside the appropriate folder.
4. Ensure that each section of code runs without errors before submission.
5. **Commit** and **push** your changes to your branch.
6. Create a **Pull Request (PR)** from your branch to the **main branch** of the original repository

A – Pandas Analysis

(Total: 25 Marks)

Q1. Load the dataset into a pandas DataFrame and show:

- Total number of matches
- Column names
- First 5 rows of data
- Describe the data

Q2. Which player has won the most *"Player of the Match"* awards in games decided on the final ball? *(i.e., matches won by just 1 run or 1 wicket).*

Q3. At Wankhede Stadium, is it more common to win by batting first (runs) or by batting second (wickets)?

Q4. Which team has the highest number of wins where the victory margin was greater than 50 runs?

Q5. How many times has the team that won the toss also set a target and won the match?

Q6. Which of the two umpires (**umpire1** or **umpire2**) has officiated more matches involving the **Kolkata Knight Riders**?

B – Playwright Web Automation

(Total: 45 Marks)

Q1. Flight Search Automation (Screen Scraping)
Use **Playwright (Python)** to open https://www.budgetticket.in and perform the following steps:

- Open the flight search page.
- Enter:
  Origin – Bangalore
  Destination – Delhi
  Journey Date – (any valid future date)
- Click **Search Flights**.
- Wait until all results are fully loaded.

Q2. Extract the following details for each visible flight:

- Airline name
- Flight number
- Departure time
- Arrival time
- Price

Store all extracted flight results in a **list of dictionaries and** save them to a file named **flight_results.json**.

Q3. Add the following fields automatically:

- searchdatetime → current UTC timestamp
- origin and destination → searched cities

Q4. Save your output file as **flight_results.json** and print the **total number of flights extracted**.

Q5. FastAPI Integration
Wrap your Playwright automation inside a **FastAPI endpoint** that runs the scraping logic and returns the scraped JSON response directly to the client.

The API should:

- Accept query parameters: origin, destination, and journey_date.
- Execute the Playwright scraper with those inputs.
- Return the list of flight details (as JSON) in the response body.

Example Endpoint:

GET /flight-search?origin=Bangalore&destination=Delhi&journey_date=2025-10-18

Expected Behavior:

- When called, the API runs the Playwright automation.
- Returns the scraped flight details as a JSON object.
- **No database connection is required.** Simply return the scraped data directly from the endpoint.

Response Format (Example Output for Reference)


```
[
 {
  "airline": "IndiGo",
  "flight_number": "6E-123",
  "departure": "06:30",
  "arrival": "09:10",
  "price": "₹5,450",
  "origin": "Bangalore",
  "destination": "Delhi",
  "searchdatetime": "2025-10-17T09:10:00Z"
 },
 {
  "airline": "Air India",
  "flight_number": "AI-504",
  "departure": "07:15",
  "arrival": "09:55",
  "price": "₹6,120",
  "origin": "Bangalore",
  "destination": "Delhi",
  "searchdatetime": "2025-10-17T09:10:00Z"
 }
]
```


Total Flights Extracted: 25

Deliverables:

- flight_search_automation.py → Playwright script
- flight_results.json → Output file
- flight_search_api.py → FastAPI wrapper that returns the JSON response

C – Gemini Integration (Applied Intelligence Task)

(Total Marks: 30)

Q1. Gemini Integration and Intelligent Summarization

Use Google Gemini 2.5 Flash API in Python to analyze and summarize information from a live webpage.

Perform the following steps:

- Connect to **Gemini 2.5 Flash** using the official SDK or REST API.
- Automatically fetch webpage data (no manual copy-paste) from **one** of the following sources:
  https://en.wikipedia.org/wiki/Artificial_intelligence
  https://www.bbc.com/news/technology
  https://edition.cnn.com/business
- Clean the HTML — remove scripts, navigation, and irrelevant text.
- Send the cleaned text to Gemini using a **custom prompt** that you create.
- The prompt must instruct Gemini to summarize and provide a short analytical insight (not just paraphrase).
- Print the result in the required console format.

Q2. What Your Script Must Do

Your script must:

- Fetch and clean webpage content automatically.
- Pass the cleaned content to Gemini 2.5 Flash with your custom prompt.
- Ask Gemini to:
  - Summarize the content in **3–5 bullet points**.
  - Add **one short insight** that interprets the overall theme or trend.
- Display output exactly in the following structure:

Summary:
• <point 1>
• <point 2>
• <point 3>
• <point 4>
• <point 5>

Insight:
<single-line insight>

- Save both parts (summary + insight) into a file named **summary_output.txt**.

Q3. Prompt Design and Reasoning Quality

- Write a **creative, clear, and well-structured prompt** that tells Gemini exactly what to produce.
- The prompt should specify the **tone**, **structure**, and **focus area** (e.g., technology, ethics, or business impact).
- Avoid generic instructions like *"Summarize this text."*
- Include an explicit request for an **insight line** after the summary.

Example concept (do not copy):

"Analyze the following webpage content and summarize it into 4 concise bullet points focusing on emerging AI trends, followed by one line that explains what these trends suggest about the future of technology."

Response Format (Example Output for Reference)

Summary:
• AI is increasingly used across healthcare, finance, and education.
• Ethical and policy concerns around AI governance are growing.
• Major tech companies are investing in responsible AI development.
• Transparency and public trust remain critical focus areas.

Insight:
The article indicates that AI is shifting from rapid innovation to responsible regulation and long-term governance.

Deliverables:

- **tripgain_gemini_analysis.py** → Python script implementing Gemini integration
- **summary_output.txt** → Saved summary and insight output
- Console output must match the exact response format shown above