GPU Finder Tool - Project Report

1. Thought Process & Approach

The goal was to create a real-time product information tool for graphics cards under ₹20,000. My approach involved:

1. **Data Collection**:
   - Validated my own created for GPU listings under ₹20,000.
   - Implemented fallback sample data in case validation fails.
2. **Backend Development**:
   - Used Flask for API endpoints (/api/products, /search, /refresh).
   - Structured data in JSON format for easy frontend consumption.
3. **Frontend Development**:
   - Designed a responsive UI with Bootstrap.
   - Added real-time search and price filtering.
   - Implemented a refresh button to update data.
4. **Error Handling & Optimization**:
   - Fallback data ensures the tool works even if scraping fails.
   - Async/Await for smooth API calls.

2. Steps Taken to Complete the Task

Step 1: Data Collection (Scraping)

- Used static JSON data .
- Extracted:
  - Product title
  - Price (filtered ≤ ₹20,000)
  - Description (auto-generated if unavailable)
  - Product link

Step 2: Backend (Flask API)

- Created endpoints:
  - /api/products → Returns all GPUs in JSON.
  - /search → Filters by keyword & max price.
  - /refresh → Simulates data reload (could trigger reloading).

Step 3: Frontend (HTML/CSS/JS)

- **Bootstrap 5** for responsive layout.
- **Dynamic filtering** via JavaScript (converted to async/await).
- **Price slider** for real-time budget adjustments.

Step 4: Deployment & Testing

- Tested locally with python app.py.
- Verified:
    - Search functionality
    - Price filtering
    - Data refresh

3. Tools, Frameworks & Libraries

Category	Tools Used
Backend	Flask (Python)
Frontend	HTML5, CSS3, Bootstrap 5, JavaScript (ES6)
Validation	JSON and file operations
Data Handling	JSON
Async Operations	Fetch API with async/await

4. Assumptions & Challenges

Assumptions

1. **Amazon as Data Source**:
    - Used mock descriptions if validation fails.
    - Data Source is dynamic.
2. **Price Limit**:
    - Hard-coded ₹20,000 as the upper limit.
3. **Manual Refresh**:
    - /refresh simulates data update (no automatic reloading).

Challenges Faced

1. **Scraping Restrictions**:

    - Amazon may block scrapers; So i have used Static JSON Data and fallback data was added.
2. **Dynamic Pricing**:
    - Prices change frequently in case of scraping; manual refresh needed.
3. **Responsive UI**:
    - Ensured Bootstrap cards adapt to mobile/desktop.

Conclusion

Future Improvements:

- Automated periodic scraping
- More retailers (Flipkart, Newegg)

Ankit Kumar