Frontend & Backend Tasks

Task 1 (Frontend – Very Easy)

Build a Personal Info Card

- Create a responsive card using **HTML, CSS, and JS**.
- Display:
  - Name
  - University
  - Favorite Programming Language
- Add **hover effects**.
- **Bonus:** Dark mode toggle.

Example Output

```
-----------------------------------------------------
|   Name: ABCDEFG                                   |
|   University: Green University of Bangladesh     |
|   Favorite Language: C++                          |
-----------------------------------------------------

(Hovering makes the card pop up slightly.
Dark mode shows the same card with dark background.)
```

—

Task 2 (Backend – Easy)

Solved Problems Tracker API

- REST API endpoints:
  - `POST /solve` → Store a solved problem.
  - `GET /solves/:user_id` → Fetch solved problems.
- **Bonus:** Return total solved count.

Example Output

POST /solve (Request):

```
{
  "user_id": 1,
  "problem_id": "CF123A",
  "time": "2025-09-21T18:00:00Z"
}
```

GET /solves/1 (Response):

```
{
  "count": 3,
  "solves": [
    {"problem_id": "CF123A", "time": "2025-09-21T18:00:00Z"},
    {"problem_id": "CF456B", "time": "2025-09-21T18:10:00Z"},
    {"problem_id": "CF789C", "time": "2025-09-21T18:30:00Z"}
  ]
}
```

—

Task 3 (Frontend – Hard)

Upcoming Contests Section (Codeforces API)

- API Documentation: `https://codeforces.com/apiHelp`
- Fetch contests from `/contest.list`.
- Show:
  - Contest Name
  - Local Start Time
  - Duration
- Add a **live countdown**.
- Add **Set Reminder** (browser notification).
- **Bonus:** Show past 5 contests with filter.

Example Output

```
Upcoming Contests
-------------------------------------------
1. Codeforces Round #999 (Div. 2)
   Start: 2025-09-25 20:35 (Local Time)
   Duration: 2h
   Countdown: 02d 03h 15m 20s
   [Set Reminder]

2. Educational CF Round #210
   Start: 2025-09-28 17:00 (Local Time)
   Duration: 2h
   Countdown: 05d 23h 40m 05s
   [Set Reminder]

(Past Contests Section)
 - CF Round #998 -> Finished on 2025-09-15
 - CF Round #997 -> Finished on 2025-09-12
```

Bonus Task (For Task 2):

Instead of querying the api every time, the server should first checks the cache (Redis). If the data is found, it is returned directly from Redis. Otherwise, it is fetched from the a and then stored in the cache for future use.

Example Scenario

Step 1: First request (data not in cache, fetched from DB).

```
GET /solves/1

{
  "source": "api",
  "count": 3,
  "solves": [
    {"problem_id": "CF123A", "time": "2025-09-21T18:00:00Z"},
    {"problem_id": "CF456B", "time": "2025-09-21T18:10:00Z"},
    {"problem_id": "CF789C", "time": "2025-09-21T18:30:00Z"}
  ]
}
```

Step 2: Second request (data served directly from Redis cache).

```
GET /solves/1

{
  "source": "cache",
  "count": 3,
  "solves": [
    {"problem_id": "CF123A", "time": "2025-09-21T18:00:00Z"},
    {"problem_id": "CF456B", "time": "2025-09-21T18:10:00Z"},
    {"problem_id": "CF789C", "time": "2025-09-21T18:30:00Z"}
  ]
}
```

Submission Process

Please follow the steps below to submit your tasks:

1. Create a **GitHub repository** and push all your code to it. Ensure that the repository is **public**.
2. Copy the repository link.
3. Submit your repository link through the following form:
   click here –> `https://forms.gle/c6P7Mb3yUnLJP2vB7`

Note: Make sure your repository is accessible and submit it before the deadline. Partial marks may be given for incomplete works, so make sure to submit whatever you have done.