Advantage Group Developer Take Home Project

Overview

Your task is to build a small backend service that supports creating and answering surveys with **role-based access control**.

The goal of this exercise is to understand how you:

- Model a real-world domain
- Enforce authorization and data access rules
- Design clean, maintainable APIs
- Make pragmatic trade-offs under time constraints

This is **not** a test of UI polish or visual design.

Time Expectations

⏱ **Estimated time:** 4–6 hours

Please do not overbuild. Completeness, clarity, and correctness matter more than additional features.

Roles

The system has two user roles:

Admin

- Create surveys
- Add questions to surveys they own
- Share survey access with other Admins
- View survey responses (individual and aggregated)

Answerer

- View available surveys
- Submit responses to surveys
- View their own past responses only

You may assume authentication is already handled (for example, a user ID and role are provided with each request).

Core Requirements

Surveys & Questions

- An Admin can create a survey
- An Admin can add questions to a survey they own
- Supported question types:
    - Rank 1-N
    - True / False
    - Text Based

Once a survey has responses, it does **not** need to be editable.

Answering Surveys

- Answerers can view surveys available to them
- Answerers can submit responses to surveys
- Answerers can view their own previous responses

Viewing Responses (Admins)

Admins can view responses for surveys they own or that have been shared with them.

They should be able to:

- View individual responses
- View simple aggregates (e.g., counts, averages, true/false totals)

Authorization Rules (Important)

Authorization must be enforced in the backend.

- Only Admins can create surveys and questions
- Only Admins can view survey responses
- Answerers must not be able to see other users' responses
- Admins must not be able to see surveys they do not own or have access to

Explicitly Out of Scope

To keep the project focused, the following are **not required**:

- A full authentication system
- UI polish or frontend (optional if you enjoy it)
- Editing or deleting surveys or users
- Pagination or advanced analytics

- Production deployment

Technical Constraints

- Language and framework are up to you
- A database should be used (any type is acceptable)
- The service must be runnable locally
- Docker is optional but appreciated

Deliverables

Please include:

1. **Runnable Code**
2. **API Documentation**
3. **Short Written Explanation**

Evaluation Criteria

Submissions will be evaluated on:

- Clarity of data modeling
- Correctness of authorization and access control
- API design and correctness
- Code quality and structure
- Documentation and communication

Optional enhancements are considered **only as positive signals** and are **not required**.

Submission Instructions

Please submit your completed take-home project as a **public GitHub repository**. The repository should include all source code, setup instructions, and any documentation you feel is relevant for reviewing your work (e.g., a README explaining design decisions, tradeoffs, and how to run the project locally).

Once your repository is ready, **email a link to the GitHub repo** to your point of contact at **Advantage Group** by the agreed-upon deadline. No additional attachments are required—reviewers will evaluate your submission directly from the repository.

If you have any questions or encounter blockers while working on the project, feel free to reach out to your Advantage Group contact for clarification.

Final Notes

This project is designed to reflect real-world backend development work.

There is no single “correct” solution — we care more about **how you think** than how much you build.

Thank you for your time and effort.