CLOUD CHARGING ENGINE

Real Work Assignment

About this Assignment

In this assessment, you will be presented with a realistic task that closely resembles the challenges you may face on the job.

Our primary focus is to evaluate your ability to analyze technical alternatives thoroughly, offer valuable insights, determine the optimal solution, make necessary code modifications to fulfill the requirements, and ensure the effectiveness of your changes through testing.

Business Background

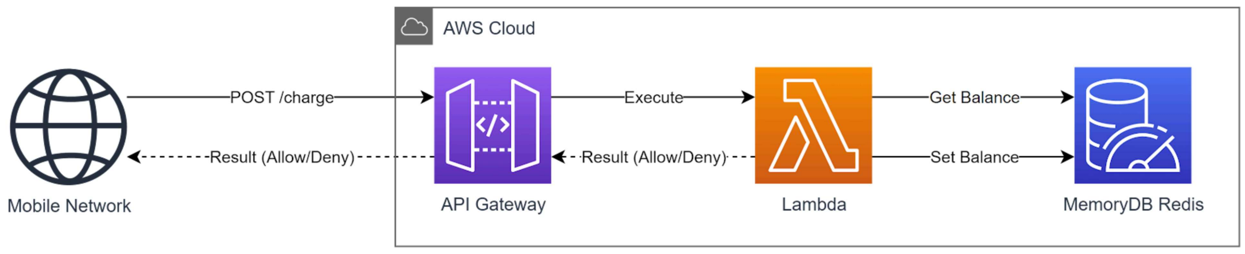
In telecommunications, billing is based on consumption (e.g., data transferred, text messages sent). Service providers rely on a charging engine to manage the funds ("balance") of subscriber accounts.

- *Requirement*: Accounts lacking funds must be prevented from accessing services (e.g., text messaging).
- *Requirement*: Low latency must be maintained for a seamless user experience.

The charging engine performs two primary operations before granting access to a service: (1) verify adequate account funds and (2) deduct the necessary amount from the account balance.

- *Requirement*: Consistency is crucial to avoid unauthorized service usage, especially during high load.

Technical Background



- AWS API Gateway: Exposes the REST API for charging operations.
- AWS Lambda: Hosts the charging engine logic.
- AWS MemoryDB (Redis): Stores the data consistently and with low latency.

Problem Statement

There are discrepancies between the anticipated charges based on subscriber consumption and the actual billed amounts. Our charging engine occasionally undercharges accounts and grants access to services to users with insufficient account balances. A small subset of accounts even have negative balances.

- *Requirement*: This must be corrected **without increasing the latency**.

Your Work

1. Make a copy of this template and set the sharing permissions to "anyone with the link can view".
2. Download the code from here and follow the README to get it running locally.
3. Run an initial set of tests to determine the baseline latency on your machine.
    - Include the results (screenshots) in your submission.
4. **Identify the root cause of the issue** mentioned above.

5. **Research the potential options** for solving it without significantly changing the architecture, and create a hypothesis as to which one will be the best fit for the requirements.
    - Your hypothesis must select the option that will both solve the issue and maintain low latency.
    - Note that the code contains a middleware that simulates AWS networking conditions while running locally. Removing it **is NOT a valid solution**.
    - Keep in mind that the code runs on AWS Lambda in production. Make sure to account for this when selecting the solution to the issue.
    - Document the reasoning behind your root causes analysis and hypothesis in your submission.
6. **Adjust the code to fulfill the requirements** based on your hypothesis.
    - Link your code in the submission and highlight important changes.
7. **Test the implementation** to ensure that it fulfills the requirements.
    - Write test code to call the API to prove that the issue is solved.
    - Check that the overall latency has not increased.
    - Include the test results (screenshots) in your submission.
8. **Propose changes to the architecture and requirements** to simplify the solution.

Grading
Your submission will be graded using the following criteria:

- **Analysis & Reasoning** - The ability to analyze an existing code base, detect issues, make technical decisions between alternative solutions, and document the criteria that influenced these decisions. High-quality submissions will include a clear choice based on a logical, detailed, and well-structured rationale underpinned by the requirements.
- **Implementation** - The ability to update code to implement a solution, based on requirements extracted from a given input. High-quality submissions will include clean and simple code changes that fully address the requirements.
- **Testing** - The ability to verify the code changes performed to ensure the requirements are met and the solution works as designed. High-quality submissions will thoroughly test the code changes against the requirements and provide extensive evidence of the results.
- **Simplification** - The ability to identify the core business value of a system and propose options to simplify the system (technically and functionally) while retaining the core business value. High-quality submissions will include feasible, detailed, and well-reasoned proposals to simplify the system.