Backend Coding Challenge

INSTRUCTIONS

At PLAYSTUDIOS player engagement is an important factor within our games. One feature we utilize heavily is quests. A quest is a basic progression mechanic in which you earn points that contribute toward an end goal. Along the way certain milestones are hit in which case you may be given an award. As part of your challenge we want you to design and develop a basic questing engine. Assume that we already have an existing game and you're being asked to create a separate internal API service that will be called by the game to progress a players quest. As such this should be a stand-alone API project. Your solution must meet the following requirements:

- Create a progress quest endpoint with the following signature:
  - Url: /api/progress
  - Method: POST
  - URL Parameters: None
  - Data Parameters:

    ```
    {
        "PlayerId": [string],
        "PlayerLevel": [number],
        "ChipAmountBet": [number]
    }
    ```
  - Success Response:
    - Code: 200
    - Content:

      ```
      {
          "QuestPointsEarned": [number],
          "TotalQuestPercentCompleted": [number],
          "MilestonesCompleted": [{ "MilestoneIndex": [number], "ChipsAwarded": [number] }]
      }
      ```
- Create a quest state endpoint with the following signature:
  - Url: /api/state
  - Method: GET
  - URL Parameters: PlayerId

- Data Parameters: None
- Success Response:
  - Code: 200
  - Content:

```
{
   "TotalQuestPercentCompleted": [number],
   "LastMilestoneIndexCompleted": [number]
}
```

- Quest point accumulation is derived from the following formula:
  (ChipAmountBet * RateFromBet) + (PlayerLevel * LevelBonusRate)
  - ChipAmountBet is the amount of chips the player bet. This will be passed into your API.
  - RateFromBet is a value derived from your configuration.
  - PlayerLevel is the player's current level which will be passed into your API.
  - LevelBonusRate is a value derived from your configuration.
- The total quest points needed to complete a quest should be configurable.
- Quests can have n amount of milestones. These should be configurable.
- Milestone completion can award the player chips. This should also be configurable.
- A player cannot complete a milestone more than once.
- Return the following data from your endpoint (these map to the success response above):
  - Quest points earned
  - Total quest percent complete
  - Milestones completed as a result of the current progression with the amount of chips awarded at each completion
- Store all your quest configuration in a single JSON file. This should be the single source of truth.
- Player quest state must be persisted.

Your solution can make the following assumptions:

- Assume there's only ever one quest active at any given time.
- The caller of your API doesn't know what quests are active.
- Only concern yourself with the player quest state and not with the overall player state. The caller of your API will be responsible for incrementing the amount of chips received from your response.

RESTRICTIONS

Use your favorite coding language for this challenge, we prefer C#, Java, or Node. You may use any data store of your choosing. You may also use any outside libraries or templates in support of your solution, but the quest logic and model structure must be all your own. You can use any method you would like to test your solution such as slapping on a simple frontend or creating a suite of unit tests (or both).

DELIVERABLES

- Please zip up your solution or give us access to your repository.
- Step-by-step instructions detailing how to get your solution up and running.
- Provide a sequence diagram of a milestone completion from request to response.
- Documentation that lists out what each property does in your quest configuration JSON.
- The schema of your data model that stores your players quest progress.

EVALUATION

We will evaluate your submission based on the following:

- Correctness
- Code quality, readability, testability, and extensibility
- Time is not a major factor but please let us know how long you took
- Be prepared to participate in a code review to discuss your solution
- Be prepared to also go through a brainstorming session that extends upon your solution

HAPPY CODING!