MishiPay Hiring Challenge (v1.5)

Internet Usage Monitoring Service

Description

In this challenge we want you to ***plan, develop & document*** an ideal HTTP service for *Internet Usage Monitoring Service* - a reusable service which returns internet usage analytics for different users using the HTTP APIs exposed by the service.

- Choose the right techstack to build the service (e.g. frameworks or databases)
- Design & document the HTTP APIs (OpenAPI 3.0 or a Postman collection)
- Use appropriate database model (or schemas) to deliver high performance
- Cover all requirements
- Use this dataset - https://drive.google.com/file/d/14fVSrhg4ct9QWIAduvFR96zacPJdCy-_/view?usp=sharing
- Submit by pushing code to a **private** GitHub repository, then add `aishwarydhare` user to it, then notify us over email along with the repo **link**

Requirements

1. A script which ingests the provided dataset in this challenge into the service's database (where the upload and download columns represent data in Kilo*bits* unit)
2. A **paginated HTTP API** to list top users by their overall internet usage in the last 30 days, while returning also the usage in the last **1 day, 7 days & 30 days (**more details in Annexure 1 on page 2)
3. A user-details **HTTP API** to search users by their exact name and return their internet usage consumption details with respect to the provided timestamp **(**more details in Annexure 2 on page 3)
4. **100% unit-tests** coverage for both APIs and ingestion-script
5. A **readme** for the service with clear instructions for project setup, run tests & start the service etc

Non-functional Requirements

1. High performance (Fast response time, Low DB calls, Optimal space & time complexity)
2. Error handling
3. Unit-tests should cover all general cases, branches & edge cases
4. Clean code
5. Code optimised for readability
6. Feel free to get-in-touch with MishiPay team for any discussion or clarification

Annexure 1 - A ***paginated*** HTTP API to list top users by their overall internet usage in the last 30 days, while returning also the usage in the last ***1 day, 7 days & 30 days***

HTTP Method	GET
URL Path	`/analytics?date=24122022&limit=100&page=1`
Query Params	`date` - the date for which data needs to be fetched in `DDMMYYYY` format `pageSize` - number of records to fetch from the database as per pagination `page` - the page number for which the data needs to be fetched as per pagination

Success Response Example	HTTP Response Status Code - `200`
	HTTP Response Type - `application/json`
	
	Example response body-
	`{`
	`  ok: true,`
	`  data: [`
	`  {`
	`  username: “user1”,`
	`  lastDayUsage: “12h33m”,`
	`  last7DayUsage: “83h04m”,`
	`  last30DayUsage: “330h08m”,`
	`  }`
	`  ],`
	`  pageSize: 100,`
	`  page: 1,`
	`  totalPages: 17`
	`}`

Success Response Example
- when no data exists for the date
- or no data exists for the provided page
	HTTP Response Status Code - `200`
	HTTP Response Type - `application/json`
	
	Example response body-
	`{`
	`  ok: true,`
	`  data: []`
	`}`

Error Response Example
- when provided date is of future
	HTTP Response Status Code - `422`
	HTTP Response Type - `application/json`
	
	Example response body-
	`{`
	`  ok: false,`
	`  error: {`
	`  message: ‘invalid date’,`
	`  }`
	`}`

Annexure 2 - A ***user details*** HTTP API to search users by their exact name and return their internet usage consumption details with respect to the provided timestamp

HTTP Method	GET
URL Path	`/user/search?username=john&datetime=20221104T1543`
Query Params	`username` - the exact name of the user for which data needs to be fetched `datetime` - the datetime relative to which the data needs to be fetched in YYYYMMDDThhmm format

Success Response Example	HTTP Response Status Code - `200`
	HTTP Response Type - `application/json`
	
	Example response body-
	`{`
	`  ok: true,`
	`  data: {`
	`  username: “john”,`
	`  lastHourUsage: {`
	`  time: “00h33m”,`
	`  upload: “100.5MB”,`
	`  download: “30.2GB”,`
	`  },`
	`  last6HourUsage: {`
	`  time: “04h20m”,`
	`  upload: “2.7GB”,`
	`  download: “180.9GB”,`
	`  },`
	`  last24HourUsage: {`
	`  time: “19h04m”,`
	`  upload: “7.2GB”,`
	`  download: “1.4TB”,`
	`  }`
	`  }`
	`}`

Error Response Example
- When user doesn’t exist
	HTTP Response Status Code - `404`
	HTTP Response Type - `application/json`
	
	Example response body-
	`{`
	`  ok: false,`
	`  error: {`
	`  message: ‘user not found’,`
	`  }`
	`}`

NOTE - The above API path, body, response are just for example/reference. Feel free to define your own improved implementation to fulfil the same requirement.