Assignment 8

This project is continuation of project 7 where lay the groundwork for a new revolutionary application where we tend to digitize punch cards (the once you get when you buy coffee and you get this punch card which gives you the tenth cup for free).

In this project we continue building our API and here we include a data-store, namely MongoDB. We also do minor changes to the API from project 7.

You can either communicate directly with MongoDB with the NodeJS MongoDB driver or use the Mongoose ORM library.

The code should be separated into the following files:

- index.js - starts the web server, connects to the database, but does very little work otherwise
- api.js - defines the routes for the application, and implements them
- entities.js - defines the database entities (see below)

Authentication

Some of the endpoints that we implement need to be authenticated. The authentication method that we use is a simple token based authentication. For example in the `GET /api/user` route (described below), clients need to add the Authorization header to the request with a token value known only by the user. This token is stored within the user document in MongoDB. On the server-side we identify the user by this token.

There should also be a token for administrators. The value of that token can be hard-coded in the code. Endpoints, such as `post /api/company`, need to be authenticated using this token.

MongoDB

In this assignment we should have a single database named `app` with three collections, users, companies and punches.

Users

In the `app.users` collection we store documents which describe users. User document should have the following fields.

- _id: Unique id for the user.
- name: String which represents the name of the user.
- token: The token value for this user. Used by the user-client to endpoints that need authorization and authentication of users. This can be a randomly generated value (The uuid node module can be handy here).
- gender: String with a single character `m`, `f` or `o`. These character stand for male, female or other respectively.

Companies

In the `app.companies` collection we store documents representing companies that have been added to the system and can give out punch cards. Company document should have the following fields.

- _id: Unique id for the company
- name: String for the company name
- punchCount: The number of punches a user needs to obtain in order to be given a discount (default: 10)

Punches

In the `app.punches` we store documents which represents punches. A given punch should have the following fields.

- _id: representing the punch id
- company_id: company id
- user_id: user id
- created: time stamp when this punch was created
- used: this value indicates if the user has used up the discount given to him/her after reaching the punchCount for the given company. Initial value should be false.

Routes

The following routes should be implemented in this assignment.

GET /api/companies - 10%

Fetches a list of companies that have been added to MongoDB. This endpoint should not use any authentication. If no company has been added this endpoint should return an empty list.

GET /api/companies/:id - 10%

Fetches a given company that has been added to MongoDB by id. This endpoints should return a single JSON document if found.

If no company is found by the id then this endpoint should return response with status code 404. No authentication is needed for this endpoint.

POST /api/companies - 15%

Allows administrators to add new companies to MongoDB. The company is posted with a POST method and the data sent as a JSON object within the request body. This endpoint should be authenticated using the Authorization header, using the hardcoded admin token.

The following requirements should hold for this endpoint.

- If admin token is missing or is incorrect (the token value can be hard-coded in the server) the server should respond with status code 401.
- If the payload is not valid (fields missing or have invalid types) the service should not add the document to MongoDB and respond with 412 (Precondition failed).

- If company was successfully added, this endpoint answers with status code 201 and returns a JSON document with the company id, `{company_id: ...}`.

POST /api/users - 15%

Allows administrators to add a new user. The client must provide the name and gender properties. Otherwise similar to the method which adds a company, except that the response contains the token of the newly created user.

GET /api/users - 20%

Returns a list of all users that are in the MongoDB. This endpoint is not authenticated and the token value within the user document **must be removed** from the document before it is written to the response.

POST /api/my/punches - 30%

Creates a new punch for the "current user" for a given company, the company id should be passed in via the request body.

This endpoint is authenticated using the user token. This endpoint should work as follows:

Clients sends a request with the Authorization header value. That value is used to authenticate the user. A new document is created in the `app.punches` collection with the user_id which owns the token value found in the header.

The following requirements should hold for this endpoint:

- If no user is found by the token, or the token value is missing, then the server should respond with status code 401.
- If no company is found by the `company_id` within the body, the server should respond with 404.
- If the user has collected as many punches as the punchCount value states is required to be given a discount, the method should return that information in the response:
  ```
  {
     discount: true
  }
  ```
  as well as marking the given punches as "used" in the database (In later implementations, we will probably save this to a separate table and allow the user to collect the discount at any time, i.e. not necessarily immediately, but this is not required at the time).

Otherwise the server should return with a status code 201 and return the newly created punch id back.

Hand-in

You only need to hand-in the JavaScript code and a `package.json` which describes the project (in a single archive file). If there is anything that you want to describe in your solution -- then please provide a read-me file with your hand-in.