(Main) Backend #3

by @Haris Porobic and @Sanjin Cabaravdic

What is this?

The following instructions and requirements in this repository represent a step in the Match community onboarding process. It's a (hopefully) simple take home exercise to be later used as a conversation starter in our technical discussion. It should also help us in reducing the number of technical challenges a client wants you to go through.

Exercise brief

Design an API for a vending machine, allowing users with a “seller” role to add, update or remove products, while users with a “buyer” role can deposit coins into the machine and make purchases. Your vending machine should only accept 5, 10, 20, 50 and 100 cent coins

Tasks

- REST API should be implemented consuming and producing “application/json”
- Implement product model with amountAvailable, cost (should be in multiples of 5), productName and sellerId fields
- Implement user model with username, password, deposit and role fields
- Implement an authentication method (basic, oAuth, JWT or something else, the choice is yours)
- All of the endpoints should be authenticated unless stated otherwise
- Implement CRUD for users (POST /user should not require authentication to allow new user registration)

- Implement CRUD for a product model (GET can be called by anyone, while POST, PUT and DELETE can be called only by the seller user who created the product)
- Implement /deposit endpoint so users with a “buyer” role can deposit only 5, 10, 20, 50 and 100 cent coins into their vending machine account (one coin at the time)
- Implement /buy endpoint (accepts productId, amount of products) so users with a “buyer” role can buy a product (shouldn't be able to buy multiple different products at the same time) with the money they’ve deposited. API should return total they’ve spent, the product they’ve purchased and their change if there’s any (in an array of 5, 10, 20, 50 and 100 cent coins)
- Implement /reset endpoint so users with a “buyer” role can reset their deposit back to 0
- Take time to think about possible edge cases and access issues that should be solved

Evaluation criteria:

- Language/Framework of choice best practices
- Edge cases covered
- Write tests for /deposit, /buy and one CRUD endpoint of your choice
- Code readability and optimization

Bonus:

- If somebody is already logged in with the same credentials, the user should be given a message "There is already an active session using your account". In this case the user should be able to terminate all the active sessions on their account via an endpoint i.e. /logout/all
- Attention to security

Deliverables

A Github repository with public access. Please have the solution running on your computer so the domain expert can tell you which tests to run. Please have a Postman collection ready on your computer so the domain expert can tell you what tests to run on the API.

Miscellaneous

How long do I have to do this?

You should deliver it in 7 days at latest.

What languages should the interface be in?

English only

Who do I contact when I'm done?

The person from Match that initially gave you the exercise link or email us at techchallenge@mvpmatch.co and we will pick it up from there.

Who do I contact if I have questions?

Feel free to use Slack, Discord, Email, Linkedin or carrier pigeon to get in touch with Haris or Sanjin from Match or emails us at techchallenge@mvpmatch.co

Will this code be shown to the client?

Assume yes. Should also help us in reducing the time by clients evaluating profiles.