# Manifesto Full-Stack Technical Challenge

## Candidate briefing

Below, on the next page, are 3 tasks we'd like to see you work on.

We're not looking for a 'right' or 'wrong' answer. We want to see how you solve problems and write code.

You should submit your solution in a way that makes it easy for us to use, we would prefer it in a git repository, GitHub or Bitbucket, so we can see some commit messages.

It's up to you how long to spend on this. We'd recommend as long as you need to produce code you feel gives a good account of yourself, your abilities and your idea of clean, well structured code.

### Hints for task 2:

- If you're using Java, we'd expect to see an Ant or Maven build script, if PHP a composer.json file, if JavaScript a package.json.
- We expect to be able to build and run the solution with no changes.
- Some documentation, like a well formed README file, telling us how to do this would be helpful. We should be able to see easily that the test input produces the correct output.
- We want to see you can create production-quality code.
- If you feel you can give your work to a brand new colleague with a minimum of hand-over, you've probably got it right.
- Including some unit tests is also a good idea.
- We don't expect you to build a user interface - a command line application or working test cases are fine. Just indicate how to demonstrate it works in the README.

# Task 1 - HTML and CSS

In the zip file linked below there are a collection of posters.

Please pick one and reproduce it using just HTML and CSS (no images), it does not need to be responsive, but it should look correct in Chrome.

https://drive.google.com/file/d/1UeLfxBdXoNyPvPiayEjUxPMLZT91nHh9/view?usp=sharing

# Task 2 - ATM CLI application

You are tasked with developing software for an ATM machine.

The software is responsible for validating customer account details and performing basic operations including balance inquiries and cash withdrawals.

A third party is developing the UI and will provide data to the application in an agreed format defined below.

The application should receive the data, process the operations and then output the results in the required format also defined below.

For the purposes of the test you are free to implement any mechanism for feeding input into your solution. You should provide sufficient evidence that your solution is complete by, as a minimum, indicating that it works correctly against the supplied test data.

The solution should meet the following business requirements:

- The ATM cannot dispense more money than it holds.
- The customer cannot withdraw more funds then they have access to.
- The ATM should not dispense funds if the pin is incorrect.
- The ATM should not expose the customer balance if the pin is incorrect.
- The ATM should only dispense the exact amounts requested.

## Input

The first line is the total cash held in the ATM followed by a blank line. The remaining input represents zero or more user sessions. Each session consists of:

- The user's account number, correct PIN and the PIN they actually entered. These are separated by spaces.
- Then, on a new line, the customer's current balance and overdraft facility.
- Then, one or more transactions, each on a separate line.
- These can be one of the following types:
    - Balance inquiries, represented by the operation code B.
    - Cash withdrawals, represented by the operation code W followed by an amount.
    - A blank line marks the end of a user session.

### *Example test Input*

| |
|---|
| 8000 |
| |
| 12345678 1234 1234 |

| |
|---|
| 500 100 |
| B |
| W 100 |
| |
| 87654321 4321 4321 |
| 100 0 |
| W 10 |
| |
| 87654321 4321 4321 |
| 0 0 |
| W 10 |
| B |

In the test data above the ATM is initialized with £8000 cash. The first customer has an actual account number 12345678 and pin number 1234. The customer entered the correct pin number 1234.

The customer has a balance of £500 and overdraft facility of £100 (allowing them to withdraw £600 in total). The customer performed 2 operations, including a balance inquiry and cash withdrawal of £100.

### Output

For each customer operation the solution should respond with the remaining customer balance or ACCOUNT_ERR if the account details could not be validated.

If funds aren't available for cash withdrawal the required response is FUNDS_ERR.

If the ATM is out of cash the required response is ATM_ERR.

The response to the test data above would be:

#### *Test Output*

| |
|---|
| 500 |
| 400 |
| 90 |
| FUNDS_ERR |
| 0 |

## Task 3 - Writing a user story

Using one of the core business requirements in the ATM project brief from task 2 above, create a user story as you would like it to be given to you.

Think a Jira ticket which we can discuss further.

We would expect to see things like ‘As a user’ and/or ‘Given, when, then.’

This can be committed to your submission repo in a /jira directory text/markdown format.