The Number26 ATM challenge

This challenge consists in creating a web based, running in a browser prototype of an ATM (Automated Teller Machine). At an higher level the *only imposed constraint* is to create a Single Page Application. The specific technology, framework(s) and graphic design are all left to the experience and creative process of the candidate.

The application must contain the following screens and user interactions:

1. Welcome screen, where the system waits for the customer to enter their card
    a. Provide a way to simulate the “card insert / inserted” interaction (i.e.: drag & drop of a card? Just a button?)
2. Insert the PIN and press *confirm*
    a. Present the user an error if the PIN is not correct (set the PIN always to 1234)
3. Select the amount to withdraw among 6 predefined choices. Give the user the option to manually insert an amount
    a. insert an amount manually
4. The ATM is preparing to deliver the amount requested
5. Please get your card and take your money
6. (abort screen)

From a UX point of view, the prototype must

1. feel “real”, faking with timeouts and/or animation the real hardware and network lags (the prototype doesn’t have the support of a real backend support, of course)
2. provide each screen with a way to go back to the previous one (no “forward” necessary)
3. provide each screen a way to abort and get to screen 1 (giving the card back to the user)
4. provide a way to reset the PIN or the custom amount input fields if the user makes a mistake

No design is provided for this prototype, given the relative familiarity with the subject each candidate should already have. There is no need to be “responsive” too: a fixed width for all the screens would work OK. Also, besides some elementary yet solid UX/UI interactions we won’t judge the graphic part. Feel free to be spartan and minimal as you like (but consistently so) if you need to save time on the project.

Notes about the implementation

- ➔ Compliance with a specific low end browser is not required (you can even target a specific one, if you prefer)
- ➔ You can use any JavaScript library or framework you want. Just keep in mind that the more code from you we see, the better it would be – of course
- ➔ You can use any external service you might need
- ➔ Use comments in the code to explain what you do and why you do it that way, even if they would be redundant in a real scenario. We'd like to see the "process" you used to accomplish the final result
- ➔ You can use (or not) any build system, transpiler or CSS preprocessor you want
- ➔ Please *do not use* CoffeeScript or TypeScript, but feel free to go ES2015 and CSS3 all the way

As part of the challenge, please also provide a README.md file containing the idea behind your implementation, its limitations as well as some installation instructions.