Module 1 – Introduction to Spring: Lab

# Lab Objectives

High-Level overview of the Project.

Business Logic

Take a look at the Wireframes.

Talk through the Architecture.

Create Project Backlog

Section 1 – High-Level Overview of the Project

Over the course of his week you will be working, in groups, to create an application that will:

Allow a user to sign up for a bank account using their first and last name.

Create user accounts.

Allow a user to enter into a prize draw.

Calculate what prize, if any, the user has won based on their account number.

Communicate this information back to the user.

Section 2 – Business Logic

We want 3 ways to generate the number:

One will create a 6 figure number.

One will create an 8 figure number.

One will create a 10 figure number.

Depending on the length of the number and the first Character (‘a’, ’b’ or ‘c’) the prize that the person wins will change.

6 digits and starts with an a you get Nothing

6 digits and starts with an b you get $50

6 digits and starts with an c you get $100

8 digits and starts with an a you get Nothing

8 digits and starts with an b you get $500

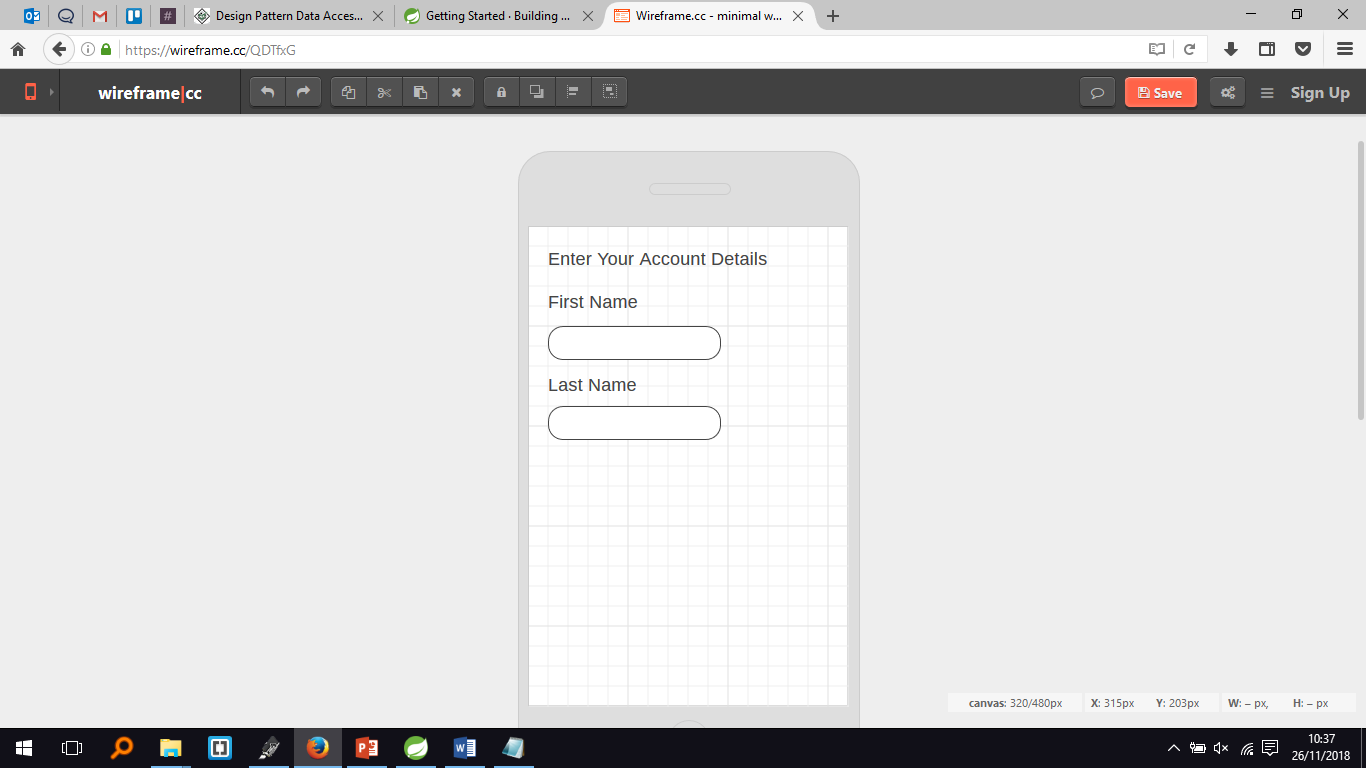
8 digits and starts with an c you get $750

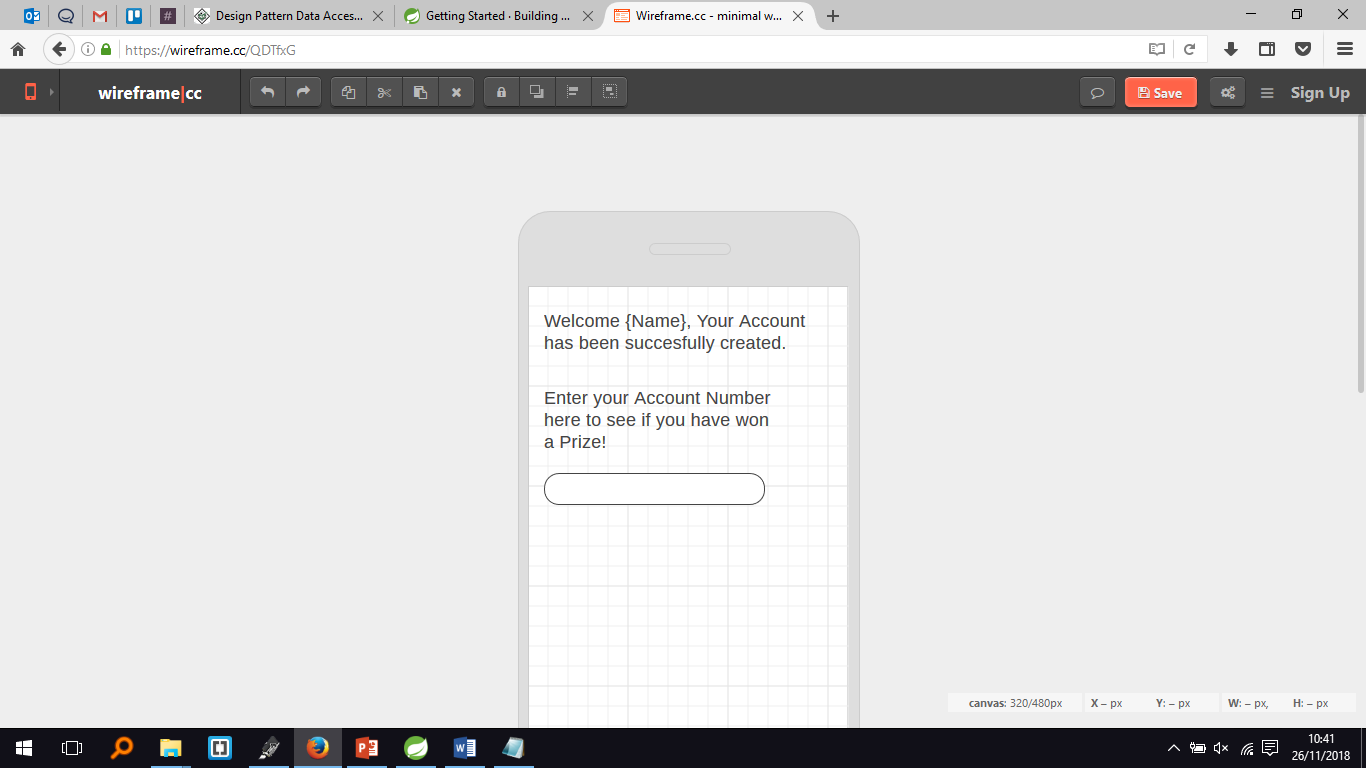
10 digits and starts with an a you get Nothing

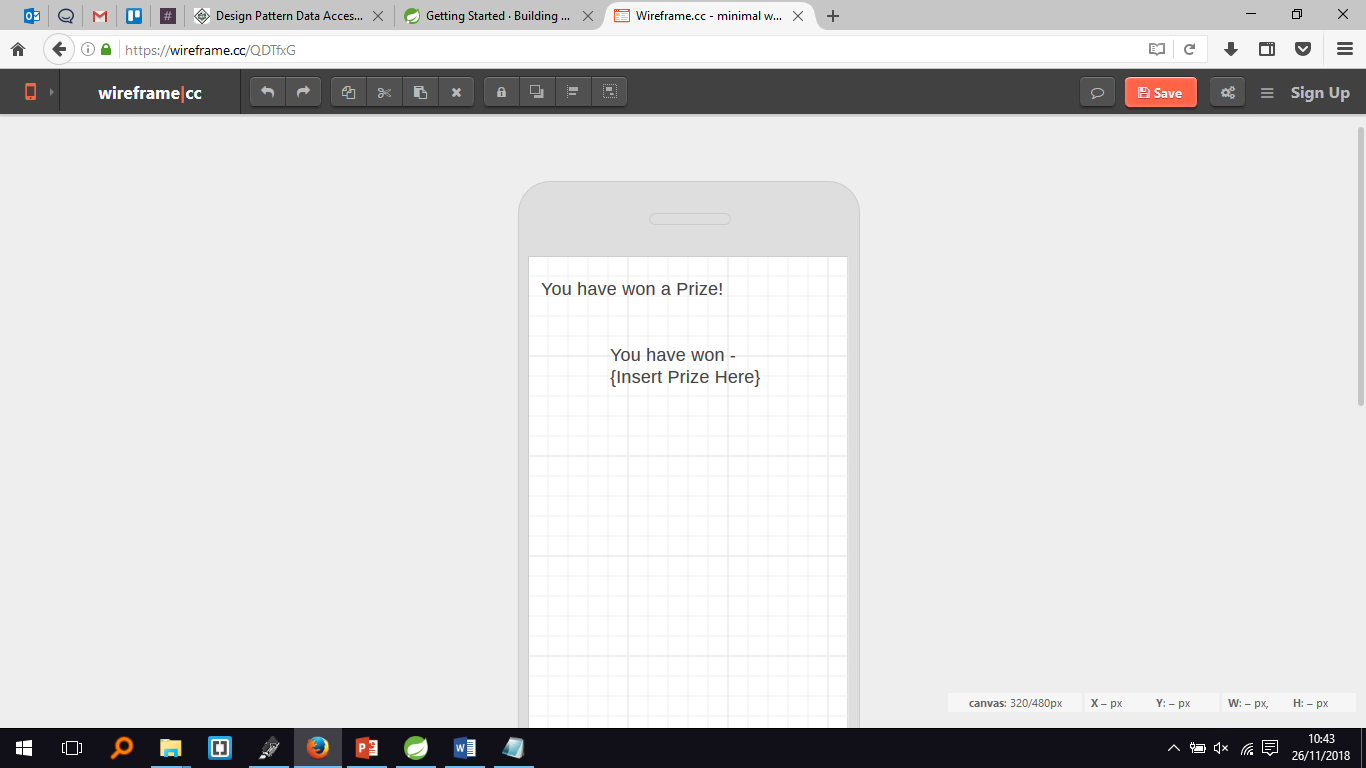
10 digits and starts with an b you get $5000

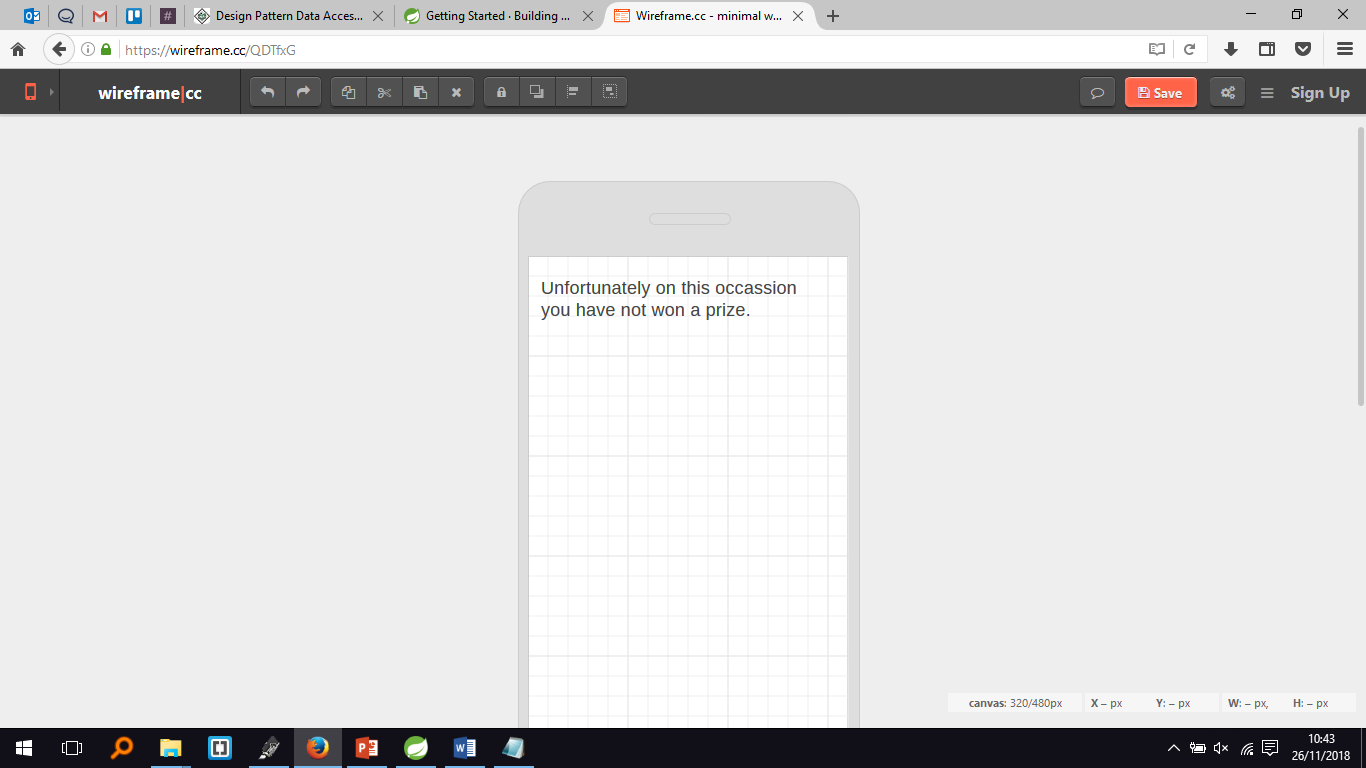
10 digits and starts with an c you get $10000

Section 3 – Take a look at the Wireframes

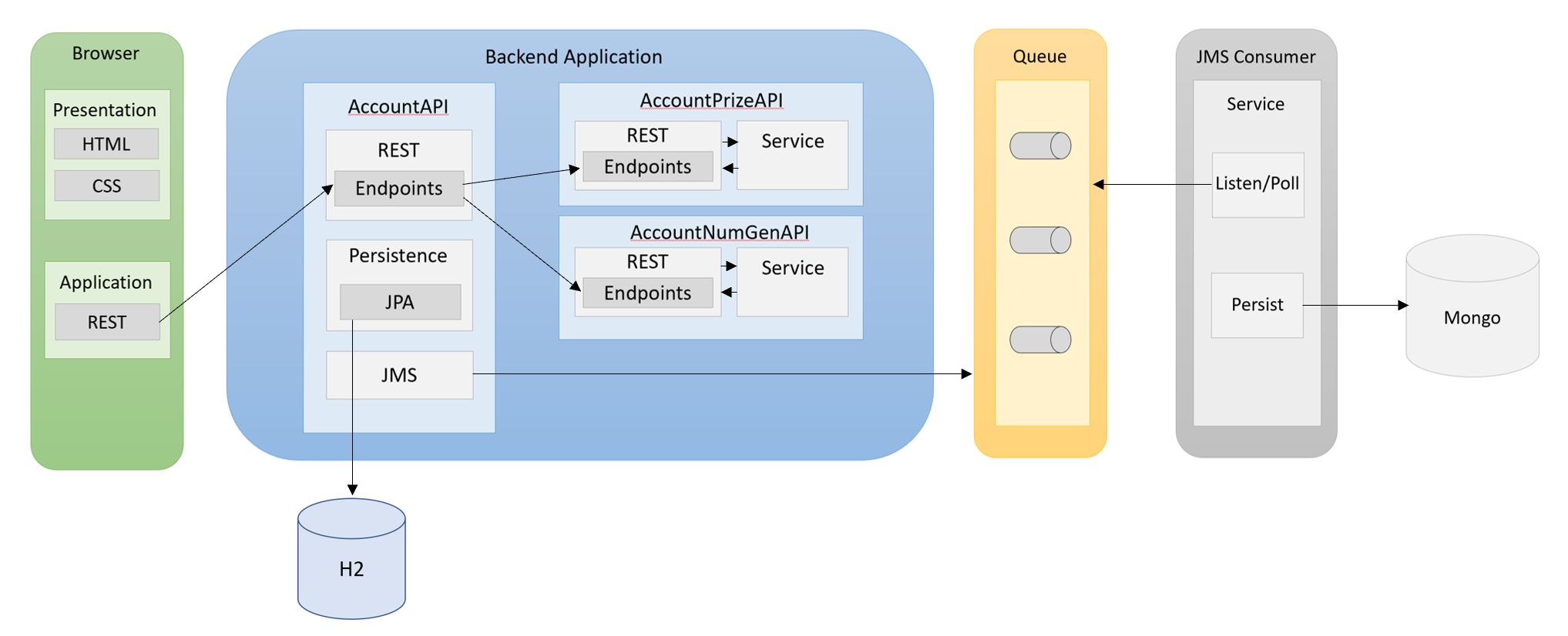
1. ****Home Page (Account Creation):
2. Account created:

****

****3. Successfully won a prize:

****4. Failed to win a prize:

Section 4 – Talk through the architecture

****

From the diagram we can see four major components of the entire application.

**Browser**:

* The user of the application will only see the front end, depicted in the diagram as Browser. More specifically they will be exposed to Presentation. This of course is the HTML rendered by the browser, styled up with CSS.
* The user will perform operations in the browser that will make REST calls to endpoints in AccountAPI.

**Backend Application**:

* This is comprised of three APIs.
* As described in the [project definition](https://github.com/JHarry444/RealAccountApi#0-Project-Definition), the user should be assigned an account number. The AccountNumGenAPI will take care of generating this number.
* The user must also know how much cash they are entitled to; this service will be taken care of by the AccountPrizeAPI.
* We do not want the Browser to make calls to multiple APIs. Instead we want it to make calls to a main API which will then make the appropriate calls to AccountNumGenAPI and AccountPrizeAPI. This main API will be AccountAPI.

**The process:**

The user will click register account and a call will be made to AccountAPI. AccountAPI must return an account number and so makes a call to AccountNumGenAPI which generates the number and sends it back.

The AccountAPI must now check if the user is entitled to a cash prize and so makes a call to the AccountPrizeAPI which runs a check and returns a cash value.

The AccountAPI now has all the information it needs and returns the data (account number and prize) to the browser while also saving this information to the local database.

But we also want to store the data to a long-term database. So we send off the data to the Queue.

Consider each API as a different person with different specialties. One API may know about prices and another may know about phones.

As a customer, if you wanted advice on phones you would ask the phone person, but the phone person doesn't know about prices!

So they would ask the prices person for help. The phone person can then tell the customer what they know about phones. In this case, the customer acts as the browser, making a request to an external API for phone advice.

**Queue**:

* The queue is used for communication between two applications. In this case it, the AccountAPI wants to talk to the JMS Consumer, since it wants to store data in the long-term database. The queue picks up data from the AccountAPI and waits for the JMS Consumer to take the data.

Imagine this as a factory conveyor belt. A worker places a chocolate bar on the conveyor belt.

Another worker is waiting at the conveyor belt for chocolate bars because it is their job to store chocolate bars away.

The worker is constantly checking the conveyor belt for the chocolate until they see one. Once they do, they grab it, store it away and go back to the conveyor belt.

In this sense, the AccountAPI is the worker placing chocolate on the belt, the Queue is the conveyor belt, and the JMS Consumer is the worker taking chocolate off the belt.

**JMS Consumer**:

* This API constantly checks if the Queue has any data that it is allowed to receive. If it does, it will take the data off of the queue.
* Once off the queue, this API will persist the data to the database.

Section 5 – Create the Product Backlog

Start by getting into groups, once you have done this your teams will have 45 minutes to create a Product Backlog on Trello for the Account application we have just covered in this lab.

Each element of your Product Backlog should take the form of a User Story.