

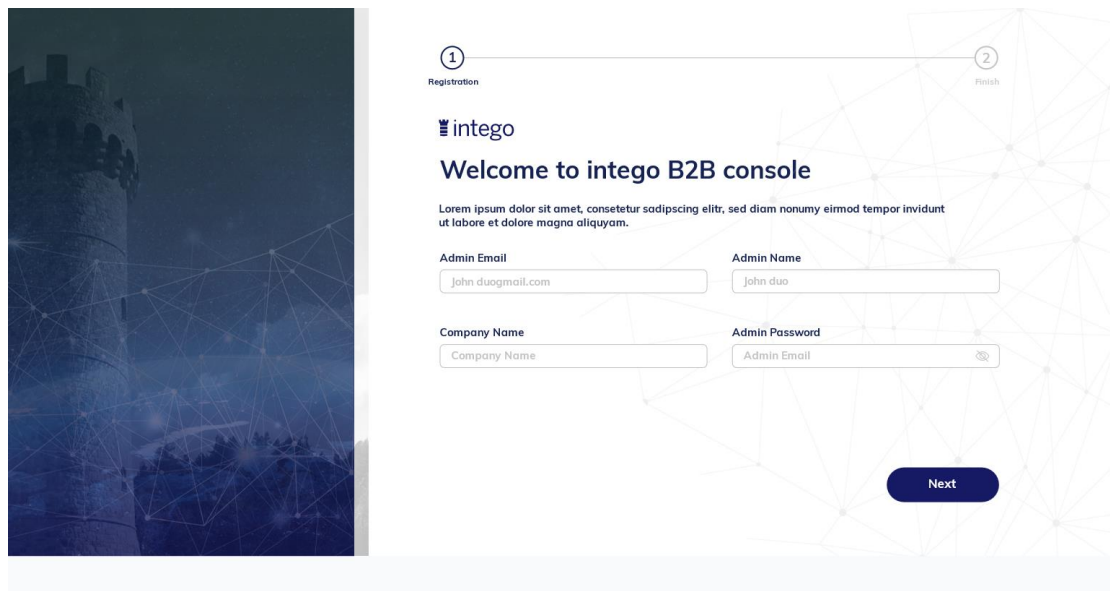
Full-stack skills assessment

Part 1 – full stack with nodejs

Using HTML, CSS, javascript and nodejs backend:

1. create the page in the “Full-stack Practical Skills assessment.xd”
2. validate input (all fields are mandatory, email)
3. Submit form to a nodejs backend
4. Store name, email, company, and hash for password in mysql table using nodejs backend

Preview:



The image shows a preview of a web form titled "Welcome to intego B2B console". The form is part of a registration process, indicated by a progress bar at the top with two steps: "1 Registration" and "2 Finish". The form includes the "intego" logo and a welcome message. Below the message, there are four input fields: "Admin Email" (containing "john.duo@gmail.com"), "Admin Name" (containing "John duo"), "Company Name" (containing "Company Name"), and "Admin Password" (containing "Admin Email" and a password icon). A "Next" button is located at the bottom right of the form. The background of the form is a dark blue image of a castle tower with a network overlay.

Part 2 – python

1. Implement a class `PriceConverter` – the class should provide a method to convert given price in USD to a requested currency.
The price should include the currency symbol at the correct position (to the left or right to the price – i.e. \$40.95 and not 40.95\$ for USD).
You should translate only supported currencies – from this list:
USD, JPY, GBP, EUR, CAD, AUD, SEK, SGD, MXN, NZD, DKK, BRL, NOK, HKD, CLP, THB, ZAR, INR, COP
Any other – return USD price
You can use this API to convert the currency: <https://exchangeratesapi.io/>
The prices were converted from USD under these guidelines:
 - a. Always round up
 - b. For currency with 2-3 numbers always add decimal 2 with .99.
 - c. For currencies with 4 numbers - unit's place=0
 - d. For currencies with 5+ - tens place=00
 - e. Examples:
 - i. 18.77 -> 18.99
 - ii. 254.01 -> 254.99
 - iii. 5,456 -> 5,460
 - iv. 11,527 -> 11,600
 - v. 111,510 -> 111,600

2. Letter Combinations of a Phone Number

Given a string containing digits from 2–9 inclusive, return all possible letter combinations that the number could represent. Return the answer in **any order**.

A mapping of digit to letters (just like on the telephone buttons) is given below. Note that 1 does not map to any letters.



Example 1:

Input: `digits = "23"`

Output: `["ad", "ae", "af", "bd", "be", "bf", "cd", "ce", "cf"]`

Example 2:

Input: `digits = ""`

Output: `[]`

Example 3:

Input: `digits = "2"`

Output: `["a", "b", "c"]`

Constraints:

- `0 <= digits.length <= 4`
- `digits[i]` is a digit in the range `['2', '9']`.

Part 3 – php

1. Implement class `ScanResults`, the class should get XML for scan results. There are 2 types of scan results – `privacy` and `junk`, the XML can contain one or another. The class should be able to output the results as a json that includes in a tree the dir, file and size, and can be ordered by alpha-numeric (dir & file) or ordered by size (of dir & file).
Do not assume the XML is ordered.

XML definition:

```
<ScanResults type="<str>">
  <dir name="<str>" items="<int>" sizeBytes="<int>">
    <file path="<str>" sizeBytes="<int>" />
  </dir>
</ScanResults>
```

Example for privacy:

```
<ScanResults type="privacy">
  <dir name="chrome" items="3" sizeBytes="32324">
    <file path="c:\chrome\1.file" sizeBytes="23" />
    <file path="c:\chrome\2.file" sizeBytes="25858" />
    <file path="c:\chrome\3.file" sizeBytes="6443" />
  </dir>
  <dir name="firefox" items="2" sizeBytes="2343">
    <file path="c:\firefox\1.file" sizeBytes="43" />
    <file path="c:\firefox\2.file" sizeBytes="2300" />
  </dir>
</ScanResults>
```

Example for junk:

```
<ScanResults type="junk">
  <dir name="c:\user\temp" items="3" sizeBytes="32324">
    <file path="c:\user\temp\1.file" sizeBytes="23"/>
    <file path="c:\user\temp\2.file" sizeBytes="25858"/>
    <file path="c:\user\temp\3.file" sizeBytes="6443"/>
  </dir>
  <dir name="c:\windows\temp" items="2" sizeBytes="2343">
    <file path="c:\windows\temp\1.file" sizeBytes="43"/>
    <file path="c:\windows\temp\2.file" sizeBytes="2300"/>
  </dir>
</ScanResults>
```

json output definition

```
{
  "type": "string",
  "totalSize": "int",
  "totalItems": "int",
  "results": {
    "folders": "array",
    "items": {
      "location": "string",
      "totalSize": "int",
      "totalItems": "int",
      "files": "array",
      "items in files array": {
        "file": "string",
        "size": "int"
      }
    }
  }
}
```

Example:

```
{
  "type": "junk",
  "totalSize": 34667,
  "totalItems": 5,
  "results": {
    "folders": [
      {
        "location": "c:\\user\\temp",
        "totalSize": 32324,
        "totalItems": 3,
        "files": [
          {
            "file": "c:\\user\\temp\\1.file",
            "size": 23
          },
          {
            "file": "c:\\user\\temp\\2.file",
            "size": 25858
          },
          {
            "file": "c:\\user\\temp\\3.file",
            "size": 6443
          }
        ]
      }
    ]
  },
}
```

```
{
  "location": "c:\\windows\\temp",
  "totalSize": 2343,
  "totalItems": 2,
  "files": [
    {
      "file": "c:\\windows\\temp\\1.file",
      "size": 43
    },
    {
      "file": "c:\\windows\\temp\\2.file",
      "size": 2300
    }
  ]
}
```

