

レポート提出票

科目名: 情報工学実験3

実験課題名: 教育システム開発-e-Testing システムの
実装

実施日: 2023年 5月 24日

学籍番号: 4619027

氏名: キムドヒョン

共同実験者:

_____	_____
_____	_____
_____	_____
_____	_____

1 実験の要旨

第2回目の実験では、実在する学生を対象に実験を行う。そのため、理論だけではなくテストシステムのインタフェースも重要になる。ということでアプリケーションを作成することを主とし、古典的な方法で実装する。また、そのシステムを利用して簡易的な実験を行い主観評価やインタフェースの重要性について理解する。

2 実験の目的

HTML と JavaScript の JSON ファイルを用いて、古典的なテストアプリケーションを作成する。

3 実験の理論

古典テスト理論は一般的に行われているテストの採点方法である。受験者の能力値 θ は以下の式で定式化される。

$$x = \theta + se \tag{1}$$

観測反応 x には真の能力 θ と偶然誤差 se が含まれるという意味である。この観測を多数行い、平均をとれば x は θ に一致する。

$$\theta = E(x) \tag{2}$$

ただし、 se は単なる偶然誤差であり、多要因とは無相関である。

この計算は x の取りうる範囲に依存し、これまで通り $x \in 0, 1$ の時は加点方式となる。これを、正答時に $x = 1$ 、無回答時に $x = 0$ 、誤答時に $x = -1$ とすると誤謬率となり、回答が多くても間違いが多いと能力値は下がる。SPI で「わからない問題は回答しないでください」という指示があった場合は誤謬率を計測している可能性が高い。

古典理論は問題の性質を考察していないという点が限界があり、さまざまなテスト理論が提案されている。一般的にテストは、その目的別に表1のように分類される。

表 1: テストの分類

種類	規模	例	理論
測定	広	センター試験	IRT
選抜	広	入試	IRT
診断	中	期末テスト	ネットワーク型 IRT
形式	狭	授業中の課題	SP 表

能力測定型テストは受験者の能力値を計算することが目的であり、大学入試センター試験や TOEFL、TOEIC などが代表的な例として挙げられる。選抜型テストは、定数内の受験者の選抜やある能力基準以上の受験者の選抜が目的であり、入社・入学試験や資格試験などが代表例として挙げられる。診断型テストは学習者の学習行き詰まりの原因を調査するためのテストで、学期末などに行われる定期試験などで実施されることが多く、直接的には能力測定を目的とし

ていない。ネットワーク型 IRT とは局所独立仮定を仮定しない IRT である。形式的テストは、教育現場で学習者・教師の日々の改善のために実施されるテストで、授業で配布される演習問題プリントなどを指す。テストの結果より、教師は個々の学習者の習熟度を確認したり、授業ペースや授業難易度の適切性を確認したりする。SP 表とは Student-Problem 表の略であり、授業改善を目的とした順次統計的な分析法の一種である。

4 課題

- 演習 2-1- `getItem(0).then(item => console.log(item));` で問題が表示されるか表示されるかかくにんしなさい。
- 演習 2-2- ブラウザに問題文とラジオボタンと選択肢が表示されること、および能力値（正答率）が正しく計算されることを確認しなさい。
- 演習 2-3- 与えられた英語の問題セット 5 問を解き、自身の解答を記録しなさい。その後、古典テスト理論による能力値を算出するテストアプリケーションを完成させ、自身の能力値を求めなさい。
- 課題 2-1- 現在何問目かを表示するようにシステムを改良しなさい。また、結果のページで解いた問題番号と選択した解答、およびその正誤を表示になさい。表示方法については、実験中に指示を出す。考察は特に必要としない。
- 課題 2-2- テストの最後に、テスト全体に対する自信度を 5 件法で問う設問を追加しなさい。また、結果のページで自信度を表示しなさい。考察は特に必要としない。
- 課題 2-3- 以下の中で一つ以上実装、期待できることについて考察
 1. 誤謬率を計算し、結果のページで表示
 2. 各問題の解答時間を記録し、結果のページで表示
 3. 「英語」と「国語」に対する自信度をテスト冒頭で取得し、結果のページで表示

4.1 演習 2-1



図 1: 演習 2-1 の実装

4.2 演習 2-2

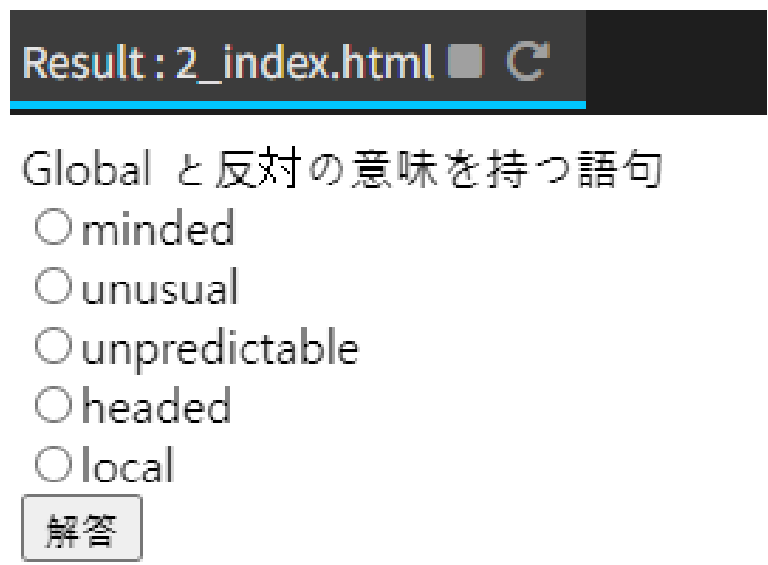


図 2: 演習 2-2 の実装結果

4.3 演習 2-3

表 2: 自分の解答

問	正誤
0.	○
11.	○
13.	○
18.	×
26.	○

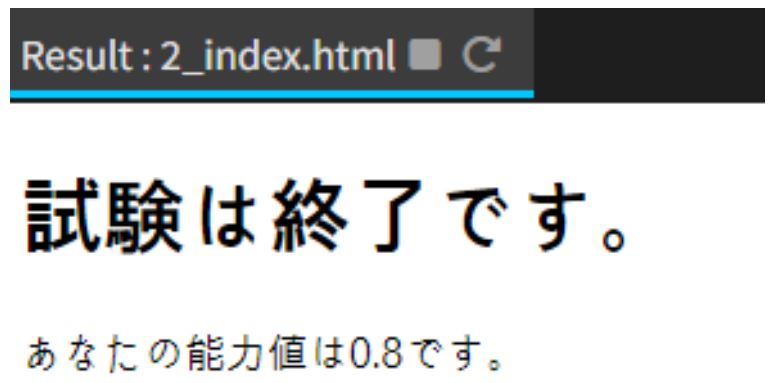


図 3: 演習 2-3 の実装結果

4.4 課題 2-1

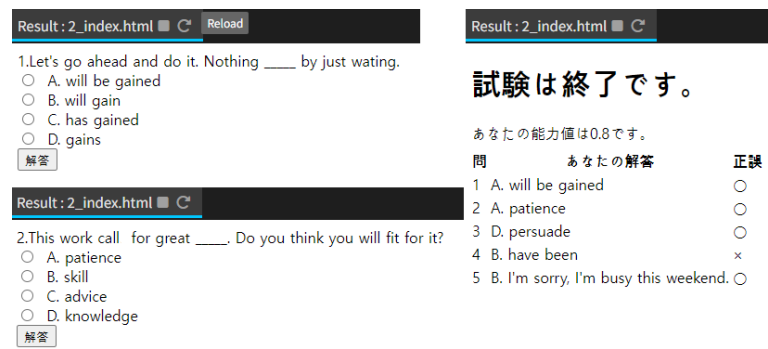


図 4: 課題 2-1 の実装

4.5 課題 2-2

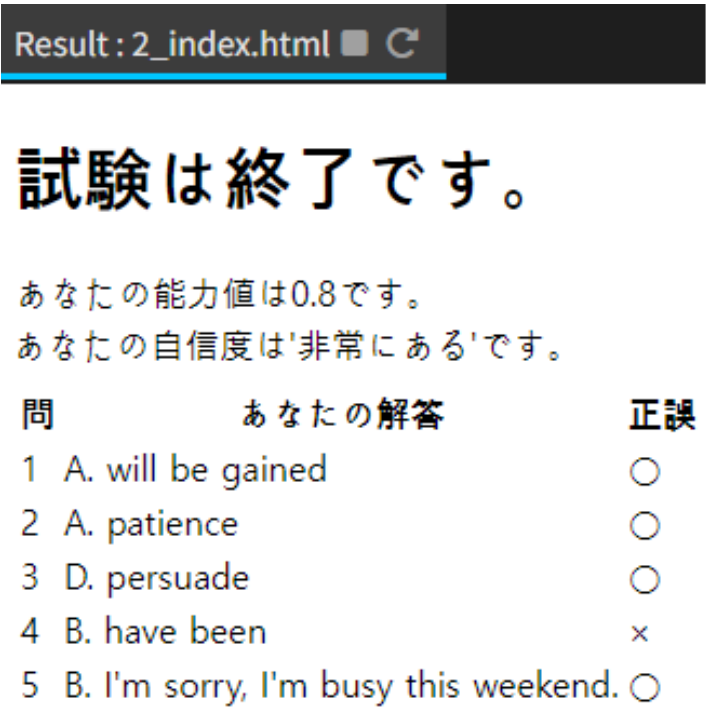
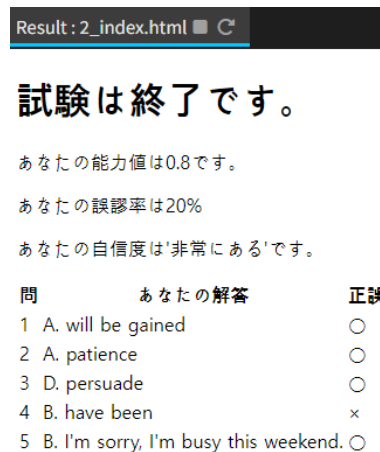


図 5: 課題 2-2 の実装

4.6 課題 2-3

4.6.1 誤謬率



問	あなたの解答	正誤
1	A. will be gained	○
2	A. patience	○
3	D. persuade	○
4	B. have been	×
5	B. I'm sorry, I'm busy this weekend.	○

図 6: 誤謬率の表示

誤謬率の計算は $1 - \text{正答率 (能力値)}$ として設定した。ソースコードは以下のようにした

```
1 errorRate = ((1-exam.theta) * 100).toFixed(0);
```

受験者はテストが終わってから最終ページで自分の誤謬率が見れるようにコードを作成した。これを通じて期待できるのは受験者は自分の正答率をわかることができる点である。

4.6.2 解答時間



問	あなたの解答	正誤	所要時間
1	A. will be gained	○	8s
2	A. patience	○	11s
3	D. persuade	○	1s
4	B. have been	×	2s
5	B. I'm sorry, I'm busy this weekend.	○	1s

図 7: 解答時間の表示

この機能は受験者が各問いについてどれくらい時間が所要されるかを計算し結果のページで表示したものである。ソースコードは以下のようにした

```

1 const questionTimes = []; //解答にかかった時間の配列
2 ...
3 const createExam = (item) => { /* 問題文・選択肢の生成 */
4   startTime = Math.floor(new Date().getTime() / 1000); //時間測定の始まり
5   ...
6   const submitButton = document.getElementById('submit-button');
7   submitButton.addEventListener('click', () => { //解答ボタンが押されると実行され
  関数
8     const endTime = Math.floor(new Date().getTime() / 1000); //解答が終わる時間
9     elapsedTime = endTime - startTime; //解答所要時間の計算
10  }
11  });
12  questionTimes.push(elapsedTime);
13 }
14
15 ...
16
17 for (let i = 1; i <= 5; i++) {
18   const t = document.getElementById(i + '_3'); //時間
19   t.innerText = questionTimes[i + 1] + 's';
20 }

```

まず、questionTimes というリストを作り、そこに所要時間が記録されるように push 関数を使用した。時間の計算は、createExam が始めて作動するときの時間を startTime にセーブし、解答ボタンが押されるときを endTime に入れて、その差を求めて秒単位で表現した。また、各問別の時間を表現するため、結果のページに for 文を利用して HTML に記録した。

4.6.3 自信度

The screenshot shows two side-by-side browser windows. The left window, titled 'Result: 2_index.html', displays a form for selecting confidence levels. The right window, also titled 'Result: 2_index.html', displays the results of the test.

Left Window (Form):

1. あなたの英語の自信度を選択してください。

- ☐ 全くない
- ☐ あまり無い
- ☐ 普通
- ☐ ややある
- ☐ 非常にある

Right Window (Results):

試験は終了です。

あなたの能力値は0です。

あなたの誤謬率は100%

自信度

英語： 非常にある
 国語： 非常にある
 テスト： 非常にある

図 8: 英語と国語の自信度の表示

冒頭に英語と国語の自信度を調査する質問を JSON ファイルに入れて、結果のページに自分の選択した自信度が表示されるようにコートを作成した。

5 まとめ

古典テストアプリケーションを JavaScript で実装し、様々な機能を追加してユーザーインターフェースを改善する作業を行いました。以下は、今回の作業で行った主な作業および改善事項の要約である。

問題および選択肢動的生成、問題別所要時間記録、問題別受験者の解答の記録、結果出力、UI の改善などがある。

上記の作業により、古典テスト アプリケーションはユーザーにより直感的で便利なインターフェースを提供するようになった。ユーザーはテストをより効果的に行うことができ、結果を通じて自分の能力値と自信度に関する情報を得ることができる。

最後に、今回の作業を通じて JavaScript を活用したアプリケーション開発に対する理解度を高め、問題解決とプログラミング能力を発展させることができた。今後の実験でも学んだ内容を積極的に活用し、より豊かで革新的な結果を生み出したい。

A 付録

Listing 1: 2_index.html

```
<!DOCTYPE html>
<html>

<head>
  <meta charset="UTF-8">
</head>

<body>
  <form id="exam-box">
    <div id="question-area"></div>
    <div id="choice-area"></div>
    <input type="submit" value="解答" id="submit-button">
  </form>
  <p id="confidence-level" style="display: none;"></p>
  <table id="result-table" style="display: none;">
    <thead>
      <tr>
        <th class="tg-fymr">問</th>
        <th class="tg-7btt">あなたの解答</th>
        <th class="tg-7btt">正誤</th>
        <th class="tg-1wig">所要時間</th>
      </tr>
    </thead>
    <tbody>
      <tr>
        <td class="tg-fymr">1</td>
        <td class="tg-dvpl" id="1_1"></td>
        <td class="tg-dvpl" id="1_2"></td>
        <td class="tg-dvpl" id="1_3"></td>
      </tr>
      <tr>
        <td class="tg-fymr">2</td>
        <td class="tg-dvpl" id="2_1"></td>
        <td class="tg-dvpl" id="2_2"></td>
        <td class="tg-dvpl" id="2_3"></td>
      </tr>
      <tr>
        <td class="tg-1wig">3</td>
        <td class="tg-0lax" id="3_1"></td>
        <td class="tg-0lax" id="3_2"></td>
        <td class="tg-0lax" id="3_3"></td>
      </tr>
      <tr>
        <td class="tg-1wig">4</td>
        <td class="tg-0lax" id="4_1"></td>
        <td class="tg-0lax" id="4_2"></td>
        <td class="tg-0lax" id="4_3"></td>
      </tr>
    </tbody>
  </table>

```

```

    <tr>
      <td class="tg-1wig">5</td>
      <td class="tg-0lax" id="5_1"></td>
      <td class="tg-0lax" id="5_2"></td>
      <td class="tg-0lax" id="5_3"></td>
    </tr>
  </tbody>
</table>
<script src="2_functions.js"></script>
</body>

</html>

```

```

1 const getItem = async (i) => {
2   const response = await fetch('./2_itemBank.json?v=' + Date.now());
3   const data = await response.json();
4   return data[i];
5 }
6 //practice2-1
7 // const practice1 = () => {
8 //   getItem(5).then(item => console.log(item.choices[0]));
9 // }
10 // practice1();
11 const exam = {};
12 let questionNum = 1; //問題番号
13 const myChoice = []; //選択した解答
14 const ox = []; //正誤○×,
15 const questionTimes = []; //解答にかかった時間の配列
16
17 const createExam = (item) => { /* 問題文・選択肢の生成 */
18   startTime = Math.floor(new Date().getTime() / 1000); //時間測定の始まり
19   //問題文領域
20   const questionArea = document.getElementById('question-area');
21   questionArea.innerHTML = questionNum + '.' + item.question;
22
23   //選択肢領域
24   const choices = item.choices;
25   const choiceArea = document.getElementById('choice-area');
26   choiceArea.innerHTML = '';
27   choices.forEach((eachChoice, index) => {
28     //ラジオボタン
29     const input = document.createElement('input');
30     input.type = 'radio';
31     input.name = 'choices';
32     input.value = index;

```

```

33     input.required = true; //ラジオボタンが押されないとき解答ボタンが押されない機能
34
35     //選択肢ラベル
36     const label = document.createElement('label');
37     label.innerHTML = '&nbsp;&nbsp;&nbsp;' + eachChoice;
38     //選択肢領域への追加
39     const div = document.createElement('div');
40     div.appendChild(input);
41     div.appendChild(label);
42     choiceArea.appendChild(div);
43 });
44 questionNum++;
45 const submitButton = document.getElementById('submit-button');
46 submitButton.addEventListener('click', () => { //解答ボタンが押されたとき実行さ
れる関数
47     const endTime = Math.floor(new Date().getTime() / 1000); //解答が終わる時間
48     elapsedTime = endTime - startTime; //解答所要時間の計算
49 }
50 );
51 questionTimes.push(elapsedTime);
52 }
53 const startTesting = async () => { /* テスト開始時の処理 */
54     exam.n = 0; //解答数
55     exam.x = []; //正誤
56     exam.theta = 0; //能力値
57     errorRate = 0; //誤謬率
58     createExam(await getItem(0));
59 }
60 startTesting();
61
62 // フォームが送信されたとき解答ボタンが押されたときの処理 ()
63 document.getElementById('exam-box').onsubmit = (e) => {
64     e.preventDefault(); //フォームの送信ページ更新の停止 ()
65     continueTesting(); //テスト継続処理の呼び出し
66 }
67
68 const continueTesting = async () => { /*テスト継続時の処理*/
69
70     //選択されているラジオボタンの値を取得
71     const choice = parseInt(document.getElementById('exam-box').choices.value);
72     //現在の問題項目を取得
73     let item = await getItem(exam.n);
74     //選択した解答の記録

```

```

75 myChoice.push(item.choices[choice]);
76 console.log(myChoice);
77 //正答なら、誤答なら1 0 を記録
78 exam.x.push(choice == item.correct ? 1 : 0);
79 //配列に正答なら oX ○、誤答なら × を記録
80 ox.push(choice == item.correct ? '○' : '×');
81 //解答数を繰り上げ
82 exam.n++;
83 //能力値正答率を計算()
84 exam.theta = exam.x.reduce((a, b) => a + b) / (exam.n - 3);
85 //誤謬率の計算
86 errorRate = ((1 - exam.theta) * 100).toFixed(0);
87 //次の問題項目を取得
88 item = await getItem(exam.n);
89 //次の問題があれば出力、なければテスト終了
90 if (typeof item !== 'undefined')
91     createExam(item);
92 else {
93     finishTesting();
94     const confidenceLevel = document.getElementById('confidence-level');
95     const englishConf = myChoice[0];
96     const japaneseConf = myChoice[1];
97     const testConf = myChoice[exam.n - 1];
98     confidenceLevel.style.display = 'block';
99     confidenceLevel.innerHTML = '<h3自信度></h3>' + '英語： ' + englishConf + '<br>' + '
100 国語： ' + japaneseConf + '<br>' + 'テスト：' + testConf;
101     for (let i = 1; i <= 5; i++) {
102         const seigo = document.getElementById(i + '_2');
103         seigo.innerText = ox[i + 1];
104         const t = document.getElementById(i + '_3');
105         t.innerText = questionTimes[i + 1] + 's';
106     }
107 }
108 };
109
110 const finishTesting = async () => { /*テスト終了時の処理 */
111     const result = '<h1試験は終了です。></h1>'
112         + '<pあなたの能力値は>' + exam.theta + 'です。</p>'
113         + '<pあなたの誤謬率は>' + errorRate + '%</p>';
114     document.getElementById('exam-box').innerHTML = result;
115     const resultTable = document.getElementById('result-table');
116     resultTable.style.display = 'table';
117     for (let i = 1; i <= 5; i++) {

```

```
117     const choice = document.getElementById(i + '_1');
118     choice.innerText = myChoice[i + 1];
119 }
120 }
121
```

Listing 2: 2_functions.js