НАЦИОНАЛЬНЫЙ ИССЛЕДОВАТЕЛЬСКИЙ УНИВЕРСИТЕТ ИТМО
Факультет Программной Инженерии и Компьютерной Техники

Отчет по лабораторной работе № 5

«Внимание, память, мышление»

Выполнила группа «думаем»
Студенты группы
Белоусова Анна Константиновна P3125
Гафурова Фарангиз Фуркатовна P3120
Гольцман Глеб Михайлович P3125
Федоров Данил Максимович P3125
Садовников Олег Юрьевич P3125
Заборщиков Артем Тарасович P3124
Павловская Анна Алексеевна P3125
Михайлов Степан Сергеевич P3124

г. Санкт-Петербург
2024

Цель лабораторной работы:
Научиться разрабатывать системы в проектной деятельности, разработать систему оценки познавательных способностей (внимание, память, мышление) у программиста (например, тестировщика), как элемент батареи тестов.

Задачи:
1.Изучить материал к данному разделу, используя, в том числе, материалы лекции по данному вопросу.

2. Используя html, css, как минимум (+php, MySQL - оптимальный вариант, + JavaScript – продвинутый вариант), разработать систему оценки внимания, памяти и мышления (класс тестов в проекте – «Познавательные способности»)

3. Оценка внимание (выбрать 2–3 характеристики внимания, из представленных в этом пункте):

- переключаемость;
- объем;
- устойчивость;
- концентрация;
- распределение;
- сделать так, чтобы респондент не смог использовать какие-либо средства для помощи себе в выполнении теста (фото, видео, рисунки, мобильный телефон и т. д.)

4. Оценка памяти (выбрать 2–3 направления для оценки памяти, из представленных в этом пункте):

- зрительная;
- звуковая;
- кратковременная;
- долговременная;
- оперативная;
- сделать так, чтобы респондент не смог использовать какие-либо средства для помощи себе в выполнении теста (фото, видео, рисунки, мобильный телефон и т. д.)

5. Оценка мышления (выбрать 3–4 операции мышления, из представленных в этом пункте):

- сравнение
- анализ
- синтез
- абстракция
- конкретизация
- индукция
- дедукция
- классификация
- обобщение
- сделать так, чтобы респондент не смог использовать какие-либо средства для помощи себе в выполнении теста (фото, видео, рисунки, мобильный телефон и т.д.)

6. Система должна позволять:

- Отображать прогресс выполнения теста;

- Управлять временем выполнения теста;

- Генерировать задания в произвольном режиме;

- Генерировать задания от простого к сложному
  - очень простые 1/3
  - средней сложности 1/3
  - сложные 1/3 14

- Оценивать результаты тестирования респондента с учетом сложности заданий

- Хранить общий список респондентов;

- Хранить результаты тестирования;

- Смотреть динамику результатов одного респондента одного вида теста, который выполнялся несколько раз;

- Нормировать результаты с учетом пола, возраста +/– 5 лет по текущей выборке;

- Система отображения результатов тестирования для эксперта:
  - Результаты отдельного теста
  - Результаты динамики одного вида теста
  - Результаты всех тестов

- Система отображения результатов тестирования для респондента (подумать зачем нужен этот пункт и в выводах отразить ответ):
  - Результаты отдельного теста
  - Результаты динамики одного вида теста
  - Результаты всех тестов

- Возможность эксперту произвольно выбирать, какие тесты необходимо пройти респонденту и в каком порядке;

- Различные варианты регистрации пользователей/респондентов/экспертов – предложить и обосновать;

- Возможность проходить тестирование по ссылке-приглашению

Описание проекта:

Папки проекта:

- Корневая директория – тут лежат страницы веб-приложения.
- css – файлы стилей.
- img – картинки.
- js – js скрипты.

- vendor – js и css для внешнего вида взятой за основу админки "SB Admin 2”
- scripts – служебные бэкенд скрипты для работы с БД и сессиями.
- tests – test.php – родительский файл страницы с тестами, все остальные страницы (colors.php, light.php и т. д.) — это шаблоны, вставляющиеся в родительский файл в зависимости от номера теста, который проходит пользователь.
- В папке tests/old_tests лежат все тесты из категории "Другие тесты”.

База данных

- tests - список тестов с названиями и ссылкой на файл (например, light.php)
- users - список юзеров
- user_tests - таблица, связывающая юзеров и тесты (поля test_id и user_id вместе образуют первичный ключ, поэтому не могут дублироваться)
- user_test_results таблица, содержащая результаты прохождения теста юзером (поля test_id и user_id могут дублироваться, как угодно, так как возможно повторное прохождение теста)
- test_categories - категории тестов (внимание, память, мышление, другие тесты)

Как работает регистрация/авторизация:

Страницы login.html, register.html - интерфейсы входа и регистрации

Скрипты login.php, reg.php, logout.php реализуют авторизацию/регистрацию/выход

Файл gates.php используется почти всеми скриптами и страницами приложения, т. к. в нем прописан функционал проверки авторизации и прав доступа. Например, с помощью функции check_auth()

```
function check_auth(){
    if(!getStatus()){
        header("Location: ../login.html");
        exit();
    }
}
```

Так же с используя функцию из этого файла можно получить текущего авторизованного пользователя:

```php
function getCurrentUser(){
    $pdo = getPDO();
    $user = null;
    if(isset($_COOKIE['auth_email'])) {
        $email = $_COOKIE['auth_email'];
        $result = $pdo->query("SELECT * FROM users WHERE email = '$email'");
        $user = $result->fetch(PDO::FETCH_ASSOC);
    }
    return $user;
}
```

Управление юзерами

На странице users.php выводятся все пользователи и функции (кнопки) для работы с ними (редактировать, изменить набор тестов, посмотреть результаты). Эти функции доступны только экспертам и админам (при чем редактировать статус юзера может только админ)

Все эти кнопки ведут на другие страницы, функционал которых в целом несложный.

Стоит обратить внимание на то, как выводятся результаты тестирования юзера (код идентичен в файле main.php, и в test_results.php.

```php
<?php
if(sizeof($test['results'])){
    $first_res = 0;
    $last_res = 0;
    $max = max(array_column($test['results'], 'result'));
    if(isset($test['results'][0])){
        $first_res = $test['results'][0]['result'];
        $last_res = $test['results'][sizeof($test['results'])-1]['result'];
    }
    $dynamic = $last_res-$first_res;
    $dynamic_percent = 0;
    if($max>0)
        $dynamic_percent = round($last_res/$max*100 - $first_res/$max*100, 2);
    foreach($test['results'] as $result):
        ?>
    <h4 class="small font-weight-bold mt-3"><?= $result['date'] ?> (сложность: <?= $result['com
            class="float-right"><?= $result['result'] ?> <?= $test['measure'] ?></span></h4>
    <div class="progress mb-4">
        <div class="progress-bar bg-success" role="progressbar" style="width: <?= ($max > 0)?$r
             aria-valuenow="20" aria-valuemin="0" aria-valuemax="100"></div>
    </div>
    <?php endforeach; ?>
    <h5 class="font-weight-bold mt-4">Динамика: <?= (($dynamic>=0)?"+":"-").abs($dynamic) ?> <
```

На странице general_result.php мы можем смотреть результаты тестов и ранжировать юзеров по возрасту.

Приглашение для прохождения теста

На странице generate_invitation_link.php эксперт или админ может сгенерировать ссылку для отправки респонденту. При этом в бд “регистрируется” пользователь без пароля, который может войти на сайт только по этой ссылке. Этот пользователь появится в общем списке пользователей, где ему нужно будет назначить тесты. Иначе, перейдя по ссылке, он увидит сообщение:

```
<?php if(!$test){ ?>
                    <div class="d-sm-flex align-items-center justify-content-between mb-4">
                        <h1 class="h3 mb-0 text-gray-800">К сожалению, тесты для Вас еще не под
                    </div>
                <?php } else { ?>
```

Тесты

В папке scripts лежит очень полезный файл get_user_tests.php. С помощью функций, описанных в нем, мы можем получить список тестов для определенного юзера. Получив этот список в файле test.php, мы начинаем по очереди “подсовывать” пользователю тесты, подключая соответствующий файл

```
<?php include $test['link']; ?>
```

В файле самого теста есть форма, которая отправляется по завершении теста (отправляется в скрипт scripts/save_test_results.php), где результаты сохраняются в БД, а также вычисляется id следующего теста. Учитывается сложность пройденного теста.

```
if($complexity == 3)
    $next_test_id = getNextTestForCurrentUser($test_id)['test_id'];
```

```
if($result){
    echo "ok";
    header("Location: ../tests/test.php?id=$next_test_id&complexity=$complexity");
}
```

Пользователь снова попадает на страницу test.php, где ему “подсовывается” уже следующий тест. Когда тесты кончаются, происходит редирект на страницу main.php

Как устроены файлы самих тестов (sound.php, count_visual.php и т.д.)

Общий шаблон:

```
<form action="../scripts/save_test_results.php" method="post">
//Тут вся верстка теста
</form>

<script>
//Самое главное - js сценарий теста
/*
Он состоит из
глобальных переменных (var test_start = false; var total_score = 0;)
таймера (function countdown()),
обработчика нажатий клавищ ( $('body').keydown(function (e).... )
функций, отражающих жизеннный цикл теста (testStart(),light(),reaction(),endTest())

*/
</script>

<style>
//Стили теста
</style>
```

	id	title	link	category_id	measure
1	1	Тест на свет	old_tests/light.php	4	ms
2	2	Тест на сложение в уме (звук)	old_tests/count_sound.php	4	ms
3	3	Тест скорости реакции на разные цвета	old_tests/colors.php	4	ms
4	4	Тест на звук	old_tests/sound.php	4	ms
5	5	Тест на сложение в уме (визуально)	old_tests/count_visual.php	4	ms
6	6	Простая РДО	old_tests/simple_rdo.php	4	ms
7	7	Сложная РДО	old_tests/hard_rdo.php	4	ms
8	8	Аналоговое слежение	old_tests/analog_tracking.php	4	ms
9	9	Аналоговое преследование	old_tests/analog_pursuit.php	4	ms
10	10	Красно-черные таблицы Горбова-Шульте	red_black_tables.php	1	ms
11	11	Кольца Ландольта	rings.php	1	ms
12	12	Тест кратковременной памяти на слова	words.php	2	pc
13	13	Тест кратковременной памяти на образы	images.php	2	pc
14	14	Арифметические операции	arithmetic.php	3	pc
15	15	Кубики Косса	cubes.php	3	pc
16	16	Установление закономерностей	patterns.php	3	%

Это все тесты. Следует обратить внимание на столбец category_id. Старые тесты принадлежат 4-й категории. Соответственно добавились 7 тестов.

Тесты написаны на js, и во всех присутствуют следующие функции:

- testStart() - запускает таймер и показывает задание
- generateTable() - функция генерации блока с заданиями. Она везде реализована по-разному, так как в одном тесте мы генерируем таблицу с картинками, а в другом

таблицу с красно-черными ячейками и числами. Кроме этого, способ ввода ответа тоже везде разный.

- endTest() - функция проверяет ответы и выводит на экран результаты (в том числе время, количество ошибок и другое)
- countdown() - отвечает за обратный отсчет времени

Наиболее интересными тестами с точки зрения кода являются Закономерности (patterns.php) и Кубики Косса (cubes.php).

В кубиках Косса рисунки-задания прописаны в коде (не генерируются рандомно). При желании можно добавить или изменить их вот тут

```
var questions = [
    [
        [3, 3, 6, 6],
        [3, 6, 3, 6],
        [6, 6, 5, 6]
    ],
    [
        [4, 2, 6, 6],
        [6, 1, 2, 6],
        [4, 2, 1, 5],
        [2, 4, 5, 1]
    ],
    [
        [5, 6, 1, 3, 6, 3, 2, 6, 4],
        [2, 1, 3, 5, 4, 3, 3, 5, 4],
        [2, 1, 5, 1, 3, 2, 4, 2, 1]
    ],
]
```

В тесте “Закономерности” также есть массив question. Еще там есть 3 дополнительные функции, специфичные для данного теста:

```
function generatePattern(word){
    shuffle(pattern_set)
    let result = []
    var pattern_count = 0
    for(var i = 0; i < word.length; i++){
        for(var j = i; j < word.length; j++){
            if (word[i] == word[j] && !result[j])
                result[j] = pattern_set[pattern_count]
        }
        pattern_count++
    }
    console.log(word);
    console.log(result);
    return result.join("")
}
```

Генерирует паттерн для слова (например, “;!+%”)

```
function checkPattern(pattern, word){
    match = true
    for(var i = 0; i < word.length; i++){
        for(var j = i; j < word.length; j++){
            if(
                word[i] == word[j] &&
                pattern[i] != pattern[j] ||
                word[i] != word[j] &&
                pattern[i] == pattern[j]
            ) {
                match = false
                break
            }
        }
    }
    return match
}
```

Проверяет соответствует ли слово паттерну

```
function shuffle(array) {
    let currentIndex = array.length;
    while (currentIndex != 0) {
        let randomIndex = Math.floor(Math.random() * currentIndex);
        currentIndex--;
        [array[currentIndex], array[randomIndex]] = [
            array[randomIndex], array[currentIndex]];
    }
}
```

Перемешивает массив. Этот метод используется и в других тестах.

Выводы по проделанной работе:

В результате данной лабораторной работы была разработана система оценки когнитивных способностей, включающая тесты на внимание, память и мышление. Использовались html, css, php, MySQL и JavaScript для создания системы. Были реализованы возможности отображения прогресса выполнения теста, управление временем выполнения, генерация заданий в произвольном режиме, оценка результатов с учетом сложности заданий. Система также позволяет хранить общий список респондентов, их результаты тестирования, а также анализировать динамику результатов. Нормирование результатов учитывает возраст и пол респондентов. Для экспертов предусмотрены различные системы отображения результатов, а для респондентов - возможность просмотра результатов различных тестов. Была также предусмотрена возможность приглашения к тестированию по ссылке и выбор тестов экспертом. В целом, система представляет собой полноценный инструмент для проведения и анализа тестирования когнитивных способностей. Каждый тест сфокусирован на измерении определенных аспектов познавательной деятельности, что позволяет получить более полное представление о способностях и навыках респондента.