





STIW2044: Mobile Programming
Semester A212
School of Computing, CAS, UUM

FRONT COVER

Lab 3

Name	Juanrico Alvaro
Matric No	702301
YouTube Presentation Link	https://youtu.be/Q7rcVYwdi0w
Phone Number	0136798361
GitHub Link	https://github.com/victorico123/flutter-projct-UUM
Submission Date	23/6/2022
Acknowledgment	I hereby acknowledge that the following works are from my effort in submitting this document. If found otherwise, severe action such as marks deduction or removal from the assignment can be taken against me.
Digital Signature	
Students Picture	
Email use during book purchase	juanricoalvaro@gmail.com
Your digital key from the book purchase	5oRiMDDWKW4U5hz



STIW2044: Mobile Programming

Semester A212

School of Computing, CAS, UUM

Lab 3

Given date: 19/6/2022

Submission date: 24/6/2022

15 Marks

This assignment is a continuation of your midterm assignment **MY Tutor** app.

Answer the following questions. Don't show the entire code. Just select a segment of the codes responsible for the required questions. Make sure to retain the code format from VSC when you paste it into the Word document. Don't screenshot your code!.

1. Implement the subjects/courses search function from your subject/courses list screen. Add more subjects/course data to your database.
 - a. UI Widget tree for the search function. (Dart code)

(3 Marks)

```
Padding(  
  padding: EdgeInsets.fromLTRB(10, 10, 20, 10),  
  child: Row(  
    crossAxisAlignment: CrossAxisAlignment.center,  
    mainAxisAlignment: MainAxisAlignment.center,  
    children: [  
      Flexible(  
        flex: 9,  
        child: Padding(  
          padding: EdgeInsets.fromLTRB(0, 0, 3, 0),  
          child: TextField(  
            controller: _searchController,  
            decoration: InputDecoration(  
              border: OutlineInputBorder(),  
              hintText: 'Enter a search term',  
            ),  
          ),  
        ),  
      ),  
      Flexible(  
        flex: 1,
```

```

        child: Center(
          child: IconButton(
            icon: const Icon(Icons.search),
            onPressed: () {
              setState(() {
                searchedQuery =
_searchController.text.toString();
                loadCourse();
                print(_searchController.text.toString());
              });
            },
          )),
    ],
  ),
),
_searchController.text == ''
? Container()
: Center(
  child: Text(
    "Results for '" +
      _searchController.text.toString() +
      "'. " +
      _totalData.toString() +
      " data found.",
    style: const TextStyle(
      fontSize: 20,
      fontWeight: FontWeight.bold,
    ),
    textAlign: TextAlign.center,
  ),
),
),

```

b. Backend code section handling the search function (PHP code)

(3 Marks)

```
$limit = 5;
$page = (isset($_POST['page']) && is_numeric($_POST['page'])) ? $_POST['page'] : 1;
$pageinationStart = ($page - 1) * $limit;
if(isset($_POST['tutor_id_chosen'])){
    $chosenTutor = $_POST['tutor_id_chosen'];
    $sqlSubject = "SELECT * FROM `tbl_subjects` WHERE `tutor_id` = $chosenTutor";
    $sql = $conn->query("SELECT count(subject_id) AS id FROM tbl_subjects")->fetch_assoc();
}
else{
    if ($_POST['search'] == '') {
        $sqlSubject = "SELECT * FROM tbl_subjects LIMIT $pageinationStart, $limit";
        $sql = $conn->query("SELECT count(subject_id) AS id FROM tbl_subjects")->fetch_assoc();
    }
    else{
        $likeQuery = $_POST['search'];
        $sqlSubject = "SELECT * FROM tbl_subjects WHERE `subject_name` LIKE '%$likeQuery%' LIMIT $pageinationStart, $limit";
        $sql = $conn->query("SELECT count(subject_id) AS id FROM tbl_subjects WHERE `subject_name` LIKE '%$likeQuery%'")->fetch_assoc();
    }
}

$stmt = $conn->prepare($sqlSubject);
$stmt->execute();
$result = $stmt->get_result();

// Calculate total pages
$allRecrods = $sql['id'];
$totalPages = ceil($allRecrods / $limit);
// Prev + Next
$prev = $page - 1;
$next = $page + 1;

if ($allRecrods > 0) {
    $courses['courses'] = array();
    while ($row = $result->fetch_assoc()) {
        $clist = array();
        $clist['subject_id'] = $row['subject_id'];
        $clist['subject_name'] = $row['subject_name'];
        $clist['subject_description'] = $row['subject_description'];
        $clist['subject_sessions'] = $row['subject_sessions'];
    }
}
```

```

        $clist['subject_rating'] = $row['subject_rating'];
        $clist['subject_price'] = $row['subject_price'];
        $clist['tutor_id'] = $row['tutor_id'];
        array_push($courses['courses'], $clist);
    }
    if(isset($_POST['tutor_id_chosen'])) {
        $response = array('status' => 'success', 'data' => $courses);

    } else {
        $response = array('status' => 'success', 'page' => "$page",
'totalPages' => "$totalPages", 'data' => $courses, 'totalData' =>
'allRecrods');
    }
    sendJsonResponse($response);
} else {

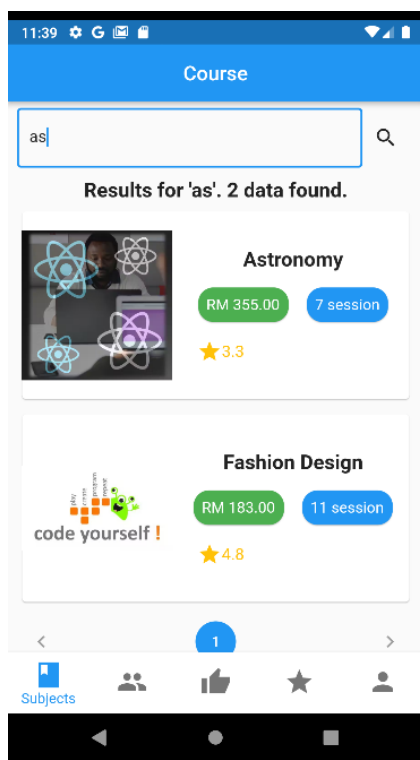
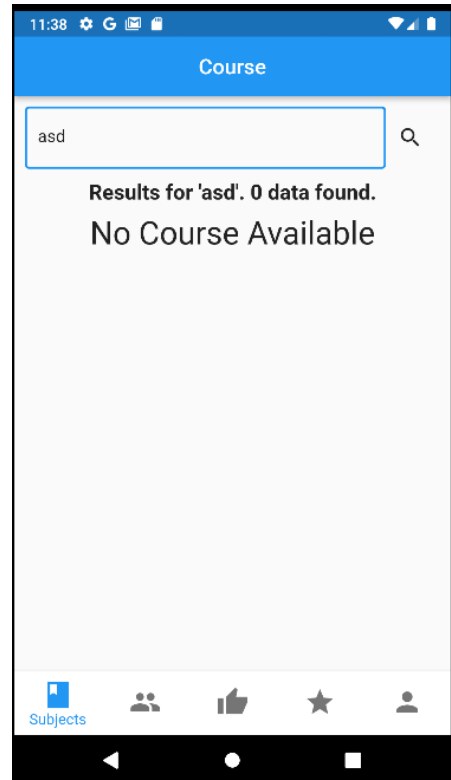
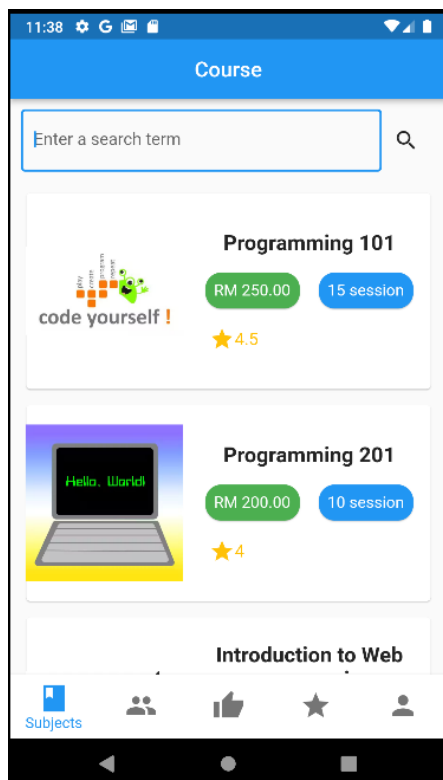
    if(isset($_POST['tutor_id_chosen'])) {
        $response = array('status' => 'failed', 'data' => null);

    } else {
        $response = array('status' => 'failed', 'page' => "$page",
'totalPages' => "$totalPages", 'data' => null, 'totalData' => "0");
    }
    sendJsonResponse($response);
}
}

```

c. UI Screenshots

(1 Marks)



2. Implement the get tutor details from the tutor list screen. The dialog window should display tutor details information and all the subject list owned by the tutor.

a. Dialog window widget.

(3 Marks)

```
Future<void> _printTutorDetails(int index) async {
  await Future.delayed(const Duration(milliseconds: 200), () {});
  showDialog(
    context: context,
    builder: (BuildContext context) {
      return AlertDialog(
        shape: const RoundedRectangleBorder(
          borderRadius: BorderRadius.all(Radius.circular(20.0))),
        title: const Text(
          "Tutor Details",
          style: TextStyle(),
        ),
        content: SingleChildScrollView(
          physics: const ScrollPhysics(),
          child: Column(
            children: [
              CachedNetworkImage(
                imageUrl:
                  "http://10.19.48.148/myTutorAPI/assets/tutors/" +
                  TutorList[index].tutor_id.toString() +
                  '.jpg',
                fit: BoxFit.cover,
                placeholder: (context, url) =>
                  const LinearProgressIndicator(),
                errorWidget: (context, url, error) =>
                  const Icon(Icons.error),
              ),
              Container(
                margin: const EdgeInsets.all(8),
                child: Text(
                  TutorList[index].tutor_name.toString(),
                  style: const TextStyle(
                    fontSize: 18, fontWeight: FontWeight.bold),
                ),
              ),
              Container(
                margin: const EdgeInsets.all(8),
                child: Text(
                  "Courses Teach List:",
                  style: const TextStyle(fontSize: 20),
                ),
              ),
            ],
          ),
        ),
      ),
    ),
  );
}
```

```

    ),
    Column(
      children: <Widget>[
        Container(
          height:
            courseDetailListHeight, // Change as per your
requirement

          width: 300.0,
          child: CurrentTutorCourseList.length == 0
            ? Text('Empty',
              style: const TextStyle(
                fontSize: 14,
              ))
            : ListView.builder(
              physics: const
NeverScrollableScrollPhysics(),

              shrinkWrap: true,
              itemCount: CurrentTutorCourseList.length,
              itemBuilder:
                (BuildContext context, int index) {
                  return Container(
                    margin: const EdgeInsets.fromLTRB(
                      0, 0, 0, 10),
                    child: Text(
                      CurrentTutorCourseList[index]
                        .subject_name
                        .toString(),
                      style: const TextStyle(
                        fontSize: 14,
                      ),
                      textAlign: TextAlign.center,
                    ),
                  ),
                );
            ],
          ),
        ],
      ),
    Column(
      crossAxisAlignment: CrossAxisAlignment.start,
      children: [
        Card(
          color: Colors.grey,
          shape: RoundedRectangleBorder(
            borderRadius: BorderRadius.circular(3.0),
          ),
          child: Container(
            margin: const EdgeInsets.all(8),

```



```

        child: Text(
          "Tutor Description: \n" +
            TutorList[index]
              .tutor_description
              .toString(),
          style: const TextStyle(
            color: Colors.white,
          )),
      ),
    ),
    Card(
      color: Colors.green,
      shape: RoundedRectangleBorder(
        borderRadius: BorderRadius.circular(15.0),
      ),
      child: Container(
        margin: const EdgeInsets.all(8),
        child: Center(
          child: Text(
            TutorList[index].tutor_phone.toString(),
            style: const TextStyle(
              color: Colors.white,
              fontSize: 15,
            ),
          ),
          textAlign: TextAlign.center,
        ),
      ),
    ),
    Card(
      color: Colors.blue,
      shape: RoundedRectangleBorder(
        borderRadius: BorderRadius.circular(15.0),
      ),
      child: Container(
        margin: const EdgeInsets.all(8),
        child: Center(
          child: Text(
            TutorList[index].tutor_email.toString(),
            style: const TextStyle(
              color: Colors.white,
              fontSize: 15,
            ),
          ),
          textAlign: TextAlign.center,
        ),
      ),
    ),
  ),
),

```

```

        Card(
          color: Colors.purple,
          shape: RoundedRectangleBorder(
            borderRadius: BorderRadius.circular(15.0),
          ),
          child: Container(
            margin: const EdgeInsets.all(8),
            child: Center(
              child: Text(
                "Join Date: " +
                  TutorList[index]
                    .tutor_datereg
                    .toString()
                    .substring(0, 11),
                style: const TextStyle(
                  color: Colors.white,
                  fontSize: 15,
                ),
                textAlign: TextAlign.center,
              ),
            ),
          ),
        ),
      ],
    ),
    actions: [
      TextButton(
        child: const Text(
          "Close",
          style: TextStyle(),
        ),
        onPressed: () {
          setState(() {});
          Navigator.of(context).pop();
        },
      ),
    ],
  );
});
}

```

b. Backend PHP file for retrieving data required for dialog widget in 2. a.

(3 Marks)

PHP

```
$limit = 5;
$page = (isset($_POST['page']) && is_numeric($_POST['page'])) ? $_POST['page'] : 1;
$pageinationStart = ($page - 1) * $limit;
if(isset($_POST['tutor_id_chosen'])){
    $chosenTutor = $_POST['tutor_id_chosen'];
    $sqlSubject = "SELECT * FROM `tbl_subjects` WHERE `tutor_id` = $chosenTutor";
    $sql = $conn->query("SELECT count(subject_id) AS id FROM tbl_subjects")->fetch_assoc();
}
else{
    if ($_POST['search'] == '') {
        $sqlSubject = "SELECT * FROM tbl_subjects LIMIT $pageinationStart, $limit";
        $sql = $conn->query("SELECT count(subject_id) AS id FROM tbl_subjects")->fetch_assoc();
    }
    else{
        $likeQuery = $_POST['search'];
        $sqlSubject = "SELECT * FROM tbl_subjects WHERE `subject_name` LIKE '%$likeQuery%' LIMIT $pageinationStart, $limit";
        $sql = $conn->query("SELECT count(subject_id) AS id FROM tbl_subjects WHERE `subject_name` LIKE '%$likeQuery%'")->fetch_assoc();
    }
}
$stmt = $conn->prepare($sqlSubject);
$stmt->execute();
$result = $stmt->get_result();

// Calculate total pages
$allRecrods = $sql['id'];
$totalPages = ceil($allRecrods / $limit);
// Prev + Next
$prev = $page - 1;
$next = $page + 1;

if ($allRecrods > 0) {
    $courses['courses'] = array();
    while ($row = $result->fetch_assoc()) {
        $clist = array();
        $clist['subject_id'] = $row['subject_id'];
        $clist['subject_name'] = $row['subject_name'];
    }
}
```

```

        $clist['subject_description'] = $row['subject_description'];
        $clist['subject_sessions'] = $row['subject_sessions'];
        $clist['subject_rating'] = $row['subject_rating'];
        $clist['subject_price'] = $row['subject_price'];
        $clist['tutor_id'] = $row['tutor_id'];
        array_push($courses['courses'], $clist);
    }
    if(isset($_POST['tutor_id_chosen']))){
        $response = array('status' => 'success', 'data' => $courses);

    }else{
        $response = array('status' => 'success', 'page' => "$page",
        'totalPages' => "$totalPages", 'data' => $courses, 'totalData' =>
        $allRecrods);
    }
    sendJsonResponse($response);
} else {

    if(isset($_POST['tutor_id_chosen']))){
        $response = array('status' => 'failed', 'data' => null);

    }else{
        $response = array('status' => 'failed', 'page' => "$page",
        'totalPages' => "$totalPages", 'data' => null, 'totalData' => "0");
    }
    sendJsonResponse($response);
}
}

```

DART

```

void getTutorSubject(String tutorId) {
    http.post(Uri.parse("http://10.19.48.148/myTutorAPI/load_subject.php"),
        body: {
            'tutor_id_chosen': tutorId,
        }).timeout(
        const Duration(seconds: 5),
        onTimeout: () {
            return http.Response(
                'Error', 408); // Request Timeout response status code
        },
    ).then((response) {
        var jsondata = jsonDecode(response.body);
        print(response.body);
        if (response.statusCode == 200 && jsondata['status'] == 'success') {
            var extractdata = jsondata['data'];

            if (extractdata['courses'] != null) {

```

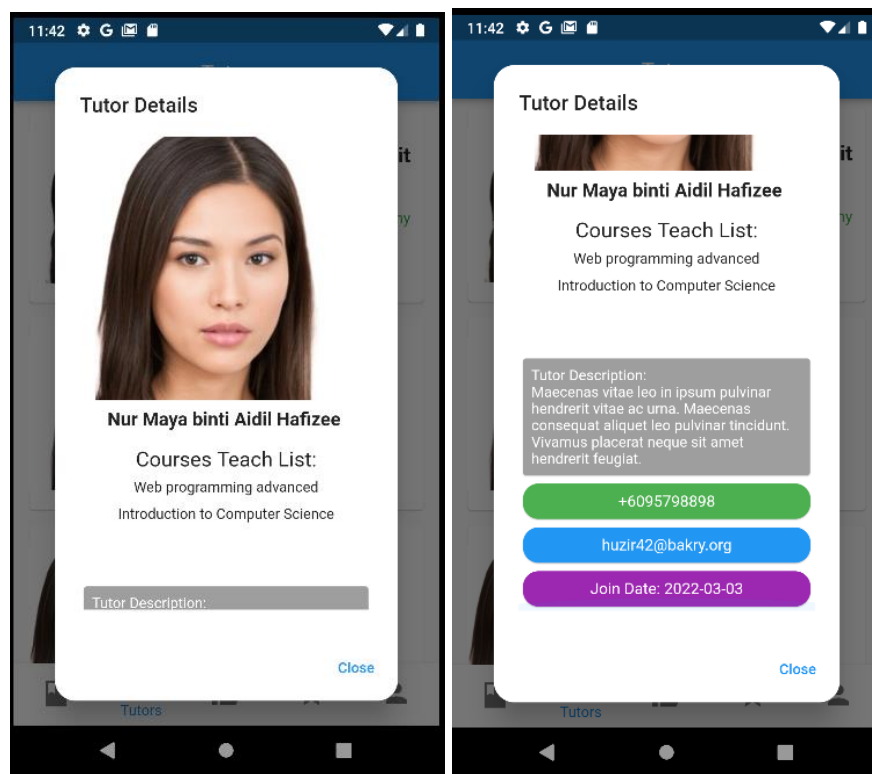
```

    CurrrentTutorCourseList = <Course>[];
    extractdata['courses'].forEach((v) {
        CurrrentTutorCourseList.add(Course.fromJson(v));
    });
    courseDetailListHeight = CurrrentTutorCourseList.length * 50;
} else {
    CurrrentTutorCourseList.clear();
}
setState(() {});
} else {
    CurrrentTutorCourseList.clear();
    setState(() {});
}
});
}
}

```

c. UI Screenshots

(1 Marks)



3. Create and publish a YouTube presentation to demonstrate both questions 1 and 2.

<https://youtu.be/Q7rcVYwdi0w>

(1 Marks)

Submission

- This document with answers and filling all the required sections. Upload to learning when the link is open for submission.
- Create a short video for your app and upload it to YouTube (provide the link on the front page).
- Upload your project to your Github Repository (provide the link on the front page).