CS3249 User Interface Development

Project: React, Meteor and Unit Testing

Due date:

This project submission is due **April 20, 2016, 23:59** (Wednesday of reading week). There will be a **group presentation** on Thursday or Friday, **April 21, 22, 2016**, during reading week by individual appointments.

The weightage of this project is **25%**.

Grading:

- Documentation: 5%
- System Design and Implementation: 15%
- Testing: 5%

Description:

This project is a **pair** assignment. There are two students per group.

The purpose is to apply the lessons you have learned on React JS, Meteor and unit testing to recreate certain parts of IVLE Events.

1. Repository Setup

1. Create a private BitBucket repository. Name your repository using this convention:

   <Matric No 1>_<Matric No 2>_CS3249_Project

   Where *<Matric No>* are the Matric numbers of both team members.
   Improper naming will result in deduction of marks.
2. Invite cs3249uidevelopment@gmail.com as a collaborator.
3. Commit your code regularly to this repository.

2. Documentation

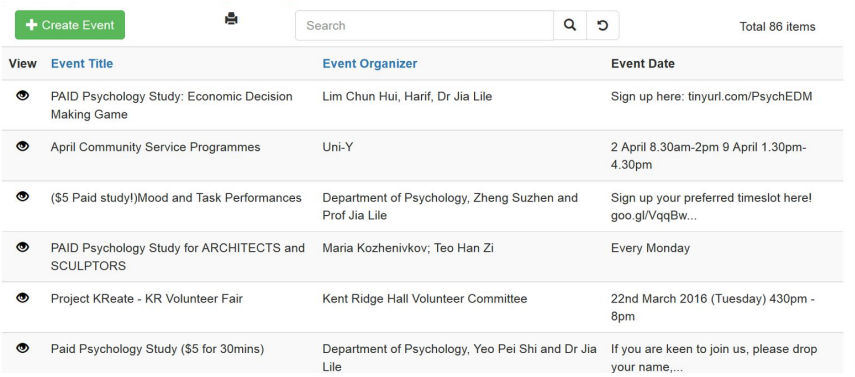
For this project, you will submit documentation as a .pdf. Include:

- Documentation on system planning, design, implementation and testing
    - Include helpful diagrams to explain your system
- Structure of your project folder
    - Include brief descriptions of main folders and files
- Instructions for setting up the project
- Team Contribution - list what each project member did for the project. This will be cross-checked with your commit history.

3. Developing the application

About the web application

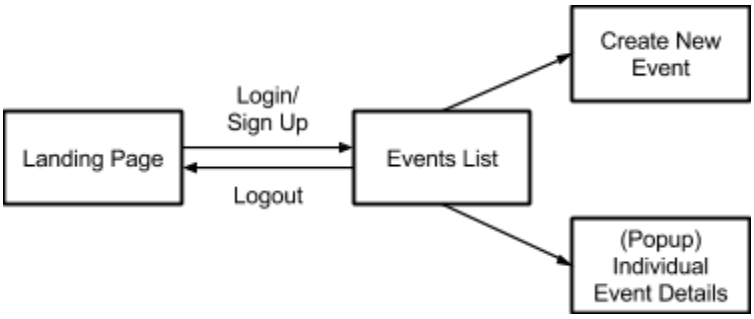
Your application will be a remake of some parts of IVLE Events (new IVLE version).



Screenshot of IVLE Events - Events List page

You are required to use **React.js** for your frontend and integrate it with **Meteor** to create a fully functional web application. The layout and design of the pages should follow the original IVLE pages as much as possible. You will probably achieve this fastest by using **Bootstrap**. You do not need to implement the top nav bar of IVLE.

In this application, you will allow user signup and login, viewing of all events in a sortable table, creation of new events and viewing individual event details in a popup. Feel free to explore it in IVLE. Here is a diagram of the views to illustrate what you will make:



Screen flow diagram

- Landing page
  - Allows users to access Sign Up and Login functions.
- Sign Up
  - Allows users to sign up with their Name, Username and Password
- Login
  - Allows users to log in to the events system with their Username and Password
- Events List containing events table (link to original page, under “Current Events” tab)
  - Allows viewing of all events in a **sortable table** - *follow original page*
  - Display a **count** of items at the top of the table - *follow original page*
  - Ability to **search** for events from the search bar - *follow original page*
    - Search by Event Title and Event Organizer
    - No need to implement advanced search
  - Each row contains a link to **view more details** in a popup - *follow original page*
  - No need to implement Print.
- View Individual Event Details popup (Click inside events table for popup)
  - Opens a popup that displays more **details** of one event - *follow original page*
  - No need to implement Email me on this event
- Create New Event (link to original page)
  - A form that allows the user to **create** a new event - *follow original page*
    - Implement appropriate field **validations**

Designing and building the application

When you start to build your application, Think in React. We want you to follow the steps inside https://facebook.github.io/react/docs/thinking-in-react.html. Your documentation and explanations ideally resemble those in the page.

Follow the steps and document your process for each step.

Design your mocks and screenshot the original IVLE pages, then you can start from Step 1.

Step 1: break the UI into a component hierarchy

- Just as "Think in React" describes, draw boxes around every component and subcomponent in the mock, and give them names (label the boxes). We want to see this for every view in your documentation.
- Explain your data models and corresponding UI components.
- Illustrate the component hierarchy that you arranged.

Step 2: Build a static version in React

- Implement a library of reusable components that render your data model.

Step 3: Identify the minimal (but complete) representation of UI state

- Look at your data and figure out which one is state. Document your process.

Step 4: Identify where your state should live and integrate with Meteor

- Identify which component mutates, or owns, the state. Document your process.

Step 5: Add inverse data flow and additional interaction

- Components update their own state.

Step 6: Add additional Meteor features, such as login management, security, etc.

4. Testing the application

You need to write unit tests for the search component of the website. The search component includes all the interactive features related to the search-bar as well as generating search result list based on the keywords you enter. Please briefly discuss the rationale behind your unit tests and instructions on running them in your documentation. You can use either Jasmine or Jest to write the unit tests.

5. Presentation

On **April 21,22**, you will give a 10 minute presentation @ COM1-01-22, Meeting Room 4 where you will

- Demo your application (in less than 4 minutes). Make sure you have the demo setup and ready to go during your presentation on your laptop computer. You only have 4 minutes to demo all the features. Wasted time on setting things up will not be compensated during presentation.
- Answer questions on your system design, implementation and testing (in 6 minutes)

Note that you team only need to come for your own session. The schedule can be arranged in the following google spreadsheet. Please pick a slot in the sheet and enter the names of all members in your group there.
https://docs.google.com/spreadsheets/d/1YDzBLUlEGa_w8FajSN1VF--XLG-1R30t5l7alJIMprM/edit?usp=sharing

Submitting Project

One team member may upload the finished product to IVLE.
All deliverables should be archived into a .zip file with the following naming convention:

MATRICNUMBER1_MATRICNUMBER2_Project.zip

submitted to the IVLE Assignment 1 workbin **before Apr 20, 2016, 23:59**.

In the .zip, you will include

- Documentation.pdf
- Your project folder