**Expense manager**

**Description:**

This is a simple expense manager application. In this you can add the type of payment, friends involved in the expense, date, currency type and the amount. There is an option to add a friend by clicking on the plus button on the top right.

All the details of the expenses are maintained in the below table. The expenses should always be in descending order of date. The expenses should be filterable according to individual name selected. All the expense amounts in the table should be in only one currency.

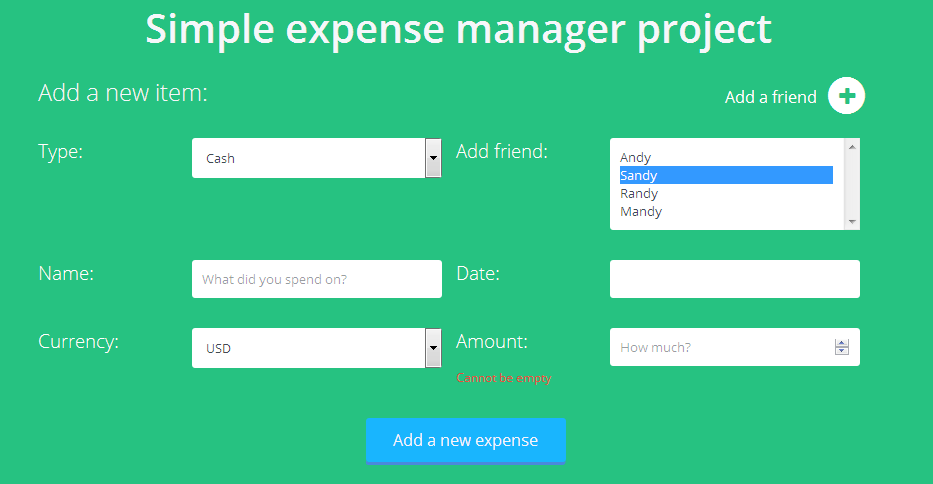
The html and css part of the application is complete. Jquery is used to validate the form and dom manipulation. Remove jquery and add Angular.js to complete the application. Any modification to the look of the page is acceptable.

**Tasks:**

* Validate the form using Angular.js.
* Add error messages under each input with the class name ‘error-msg’.
* After adding a friend, the value should be inserted into the ‘Add friend’ select options. Trigger the add action on the ‘enter’ key press event.
* Use any angular date picker library for date field.
* Append the expense details into the table after validating the form and set the input fields to default values.
* The expenses should be sorted always in the descending order of date(latest date first).
* All the expense amounts should be converted to a single currency which is selected in the filters bar.
* After selecting edit option, the details should be filled into the form above and ‘Add a new expense’ header and ‘Add a new item' button text should be changed to ‘Edit expense’ and ‘Done’ respectively.
* After editing the form and on click of the ‘Done’ button, the details of that row should be updated.
* Delete the expense row after clicking on the delete option.
* Add a friend, Edit, Delete and Filter functionalities are not implemented in jquery, you have to implement it with Angular.

**Note:**

* The whole code should be in Angular.js. **Jquery shouldn't be used anywhere.**
* Bonus points for code modularization.



====================

1. Make a Single Page Application for login. Hash the password and store it in database. Once the username and password matches let the user logs in. (Encrypt the password and then match it.)

2. Modify above application and set the login page as the “App Entry Point”.

3. Apply unit testing in the above application.

================

Examples

1. Make an angular application in which the user data (username/email) is stored in and displayed on HTML view. The application should have a simple interface of a textbox and an add button. When a user writes anything in the textbox and clicks on the add button the newly added username/email should be displayed in the list of usernames present in web page.

2. Modify the above example make one base controller (Employee) and 2 child controllers (admin and development.). Make 2 separate pages for Admin department and Development department. The user should be able to add employees in respective departments.

3. In the above application add one more field for adding the contact number of the users. Also add the editing functionality to the user data by maintaining separate states for adding and editing.

Assignments

1. Make a bookmarks listing application. In this application create 4 categories. List a menu on the left side of the page. In this menu create a navigation menu that highlights the selected entry. Fetch the category and the bookmark data from a JSON file. No Need to apply the filtering logic for filtering the bookmarks as per the selected link.

**Online Test App**

Create an Online Test App, Through this app trainees should be able to appear for the test created by the trainer. There will be one entry point (login) for all the users. Whether trainee, trainer or administrator.

There are three modules in this app.

1. Trainee
2. Trainer
3. Administrator

Task of each user

Trainee:

* Appear for the test

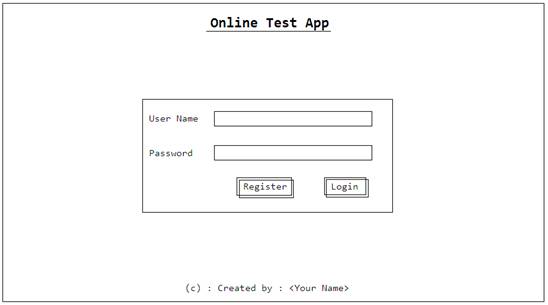
Trainer:

* + Create Test
  + Create Question Bank
  + Do Test Setting

Administrator

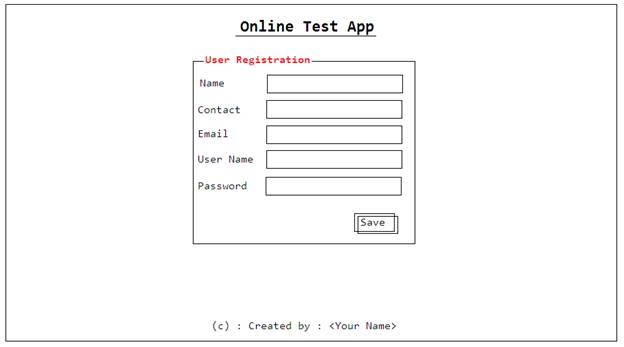
* + He will give access to guest user and will give him rights of trainer or trainee.
  + Update the trainer or trainee information
  + Delete the trainer or trainee

Wireframe is provided for your reference:



Login view:

* This must be single entry point for all types of user. Registered user should be able to login from this view.
* Proper form validation should be provided.
* If user credentials match properly then based on user type app should redirect user on accepted view.
* If user credentials do not match then user should be redirected to login view itself with error message on top of login view in red color and both the text boxes should be blank and focus should be on first text box.

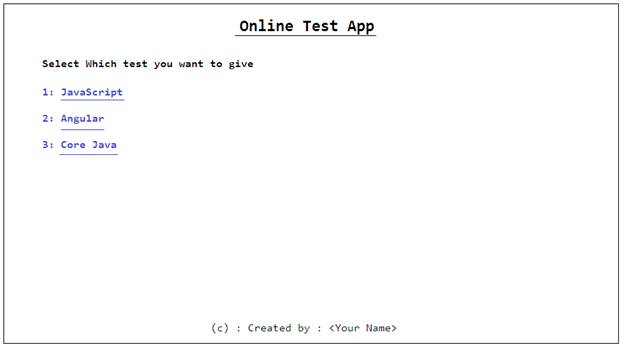


Registration:

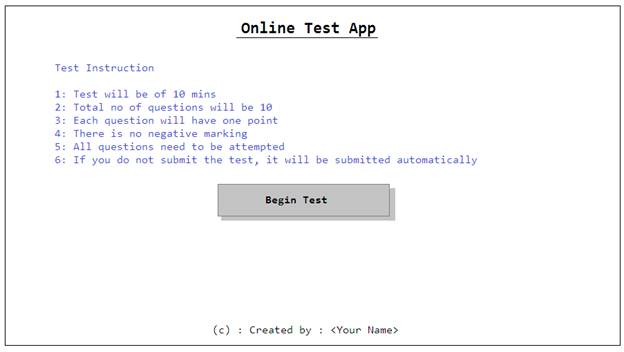
* User should be able to register himself as a guest user.
* All fields should have proper user validation.
* If user have been registered, and if he login with the same registered username and password, then user should be redirected to guest view, saying you need administrator approval. Once administrator will approve user then according to the usertype user will be redirected to expected view based on type.

Screen for Trainee:

1st View: Test Selection

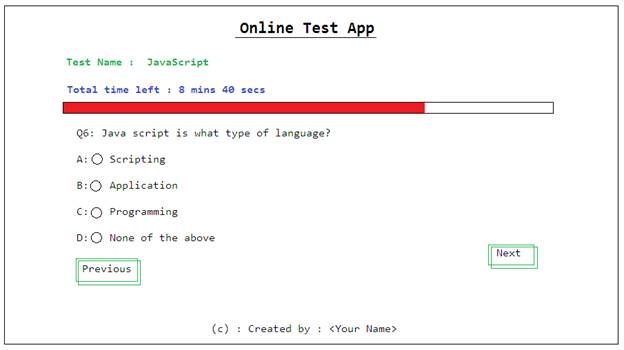


2nd view: instruction view



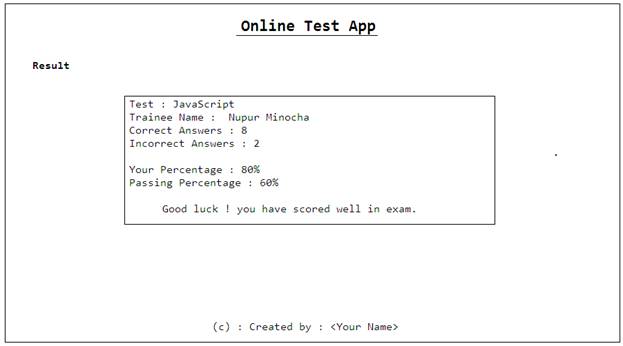
Note: Instructions will be displayed based on setting the trainer will be doing.

3rd View: StartTest view



Trainee should be able to move to next or previous question. Once he will reach to the end of the test he will get only submit button. Once he clicks on submit button. He should get the result immediately.

5th view: Result view

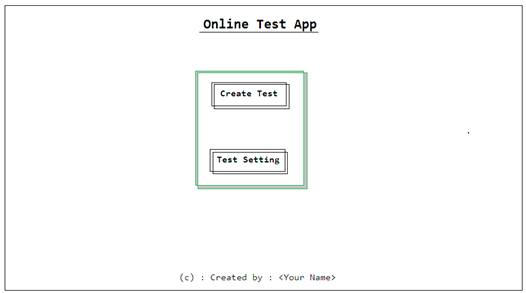


**Trainer module**

In this module trainer will create the test & question bank, which administrator will check and allow to be displayed for trainees to attempt test.

Once trainer login from the single login point he will get the first view as

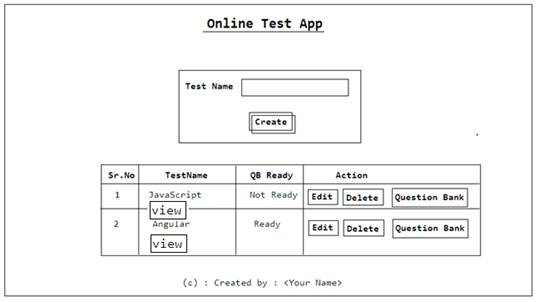
1st view: testcreation\_setting



Trainer cannot do test setting without creating test, so first he will have to create test and then he can do the test setting.

Let’s assume trainer choose the **Create Test** option.

2nd view: Test Creation



**Edit:** you can edit the question or answer

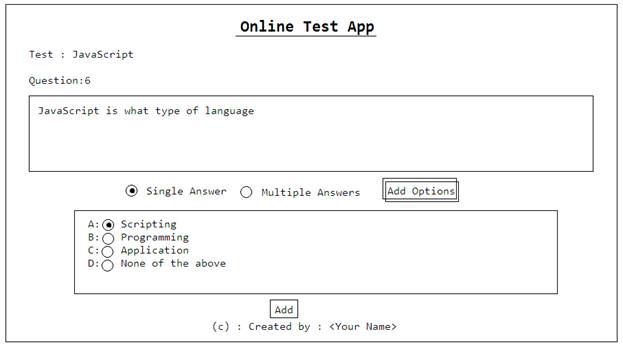
**Delete:** this option will delete the test

**Question Bank:** Initially Test will be created with the Not Ready (QB Status), By clicking on this option trainer will get the option to create the test.

**View:** by clicking on this button trainer should be able to see the tabular view of all the questions, options and current answers.

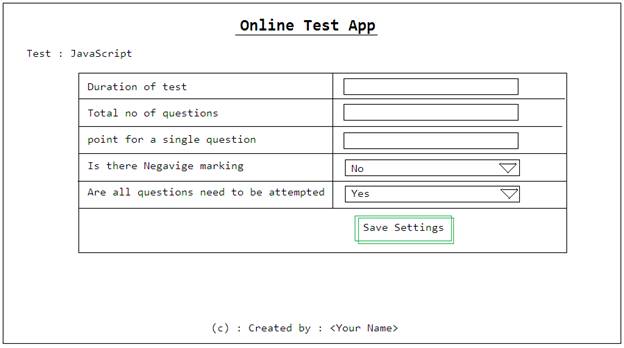
If trainer clicks on Question Bank button the QB view will be appeared.

3rd View: QuestionBank Creating view



If once test is ready, trainer can do the setting of test. For this you can create a test setting button on the test creating page itself in tabular display of tests.

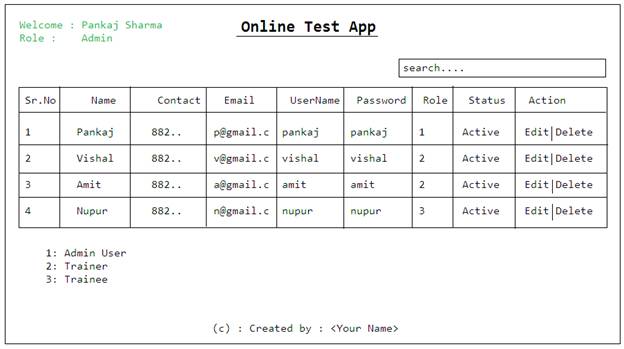
4th View: TestSetting view



**Administrator Module**

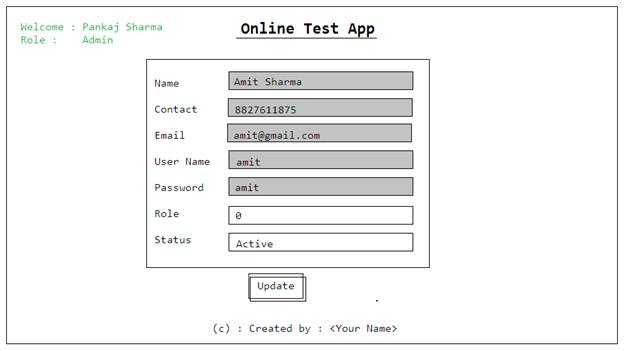
Administrator should see the entire user list once he login to the system.  He will be editing the user, he should be able to assign the rights to user, and other details should not be changeable.

1st View: usersview



Once administrator clicks on Edit, he should be able to change user role.

By default all the users should have role value as 0 (zero) [Guest User]: which has to be later changed by the administrator based on request.



**Note: to save the data of user, test use text file and store data in JSON format. On startup of the application data has to come from those text files.**