|  |  |
| --- | --- |
| **Angular Practice Assignments** | |
|  |  |
|  | **Assignment-1 (Components)** |
|  | 1. Create two new components :: WarningAlert and SuccessAlert. 2. Output them each other in the AppComponent. 3. Output a warning or success message in the components. 4. Style the components appropriately.   Use Templates and styles.  Create more component, nest them into each other. |
|  |  |
|  | **Assignment-2 (Data Binding)** |
|  | 1. Add a input field which updates a property (‘username’) via two-way binding. 2. Output the username property via String Interpolation (in a paragraph below the input). 3. Add a button which may only be clicked if the username is NOT an empty string or disable it. 4. Upon clicking the button, the username should be reset to an empty string. |
|  |  |
|  | **Assignment-3 (Directives)** |
|  | 1. Add a button which says “Display Details”. 2. Add a paragraph with any content of your choice. 3. Toggle the displaying of that paragraph with the button created in the first step. 4. Log all button clicks in the Array and output that Array below the paragraph (may be log a timestamp or simply an incrementing number). 5. Starting at the 5th log item, give all future log items a blue background (via ngStyle) and white color (via ngClass). |
|  |  |
|  | **Assignment-4 (Property and event binding and View Encapsulation)** |
|  | 1. Create three new components :: GameControl, odd and even. 2. The GameComponent should have two buttons :: start and stop the game. 3. When starting the game, an event (holding an incrementing number) should get emitted each second (ref = setInterval()). 4. The event should be listenable outside the component. 5. When stopping the game, no more events should get emitted (clearInterval(ref)). 6. A new odd component should get created for every odd number emitted, the same should happen for the even component (on even numbers). 7. Simply output ODD-NUMBER or EVEN-NUMBER in the two components. 8. Style the element (paragraph) holding your output text differently in both components. |
|  |  |
|  | **Assignment-5 (About Services)** |
|  | **Active Users**  Ami | Set to Inactive  Urvish | Set to Inactive  **Inactive Users**  Denish | Set to Active  Vinit | Set to Active  Switch them from active to inactive and inactive to active by clicking on this buttons. |
|  |  |
|  | **Assignment-6 (Template Driven Forms)** |
|  | Add a form with the following inputs (validators)   1. Mail address (should not be empty and should be valid email address). 2. A dropdown which allows the user to select from three different subscriptions (“Basic”, “Advanced”, “Pro”).Set Advanced as default. 3. A password field (Should not be empty). 4. A submit button.   Display a warning message if the form is invalid. Display a warning message below each input if it’s invalid. Upon submitting the form, you should simply print the value of the form to the console. Display it in your template. |
|  |  |
|  | **Assignment-7 (Reactive Forms)** |
|  | Create a form with following controls and validators   1. Project name (Should not be empty). 2. Mail (Should not be empty and valid). 3. Project status dropdown with three values “Stable”, “Critical”, “Finished”. 4. Submit button.   Add your own validator which doesn’t allow “Test” as project name. Implememt that validator as an async validator (replace the other one). Upon submitting the form, print the value to the console. |
|  |  |
|  | **Assignment-8 (Pipes)** |
|  | Create a text box and button named “Add Server”.  Example :: Data which is already inputed are as below.  Production | Medium | Sunday, August 8,2016 | Stable  User Database | Large | Sunday, August 8,2016 | Stable  Development Server | Small | Sunday, August 8,2016 | Offline  Testing Environment | Small | Sunday, August 8,2016 | Stable  To Build Reverse pipe (reverses the string) and Sort this list by first instance (Sorting Pipes). |
|  |  |