6 Semester Training Plan(12 Weeks)

Date	**Weekly Plan**
17 Feb 23	Project Explanation
20 Feb 23- 24 Feb 23	Week 1 : Day 1-5 React introduction and project overview, React installation and Project folder structure basics of component and props
27 Feb 23 - 3 Mar 23	Week 2 : Day 6-10 Material ui introduction and css in react and registration page
6 Mar 23 - 10 Mar 23	Week 3 : Day 11-20 useState, useEffect hook and Formik form validation
13 Mar 23 - 17 Mar 23	Week 4 : Day 11-24 Fetch or axios http api call and react toast
20 Mar 23 - 24 Mar 23	Week 5 : React router dom and login page
27 Mar 23 - 31 Mar 23	Week 6 : Product list page
3 Apr 23 - 7 Apr 23	Week 7 : BookList page and add/edit/delete book
10 Apr 23 - 14 Apr 23	Week 8 : Category page and add/edit/delete category
17 Apr 23 - 21 Apr 23	Week 9 : User list page and edit/delete user
24 Apr 23 - 28 May 23	Week 10 : Update profile and search book
1 May 23 - 5 May 23	Week 11 : useContext hook, create auth context and private route
8 May 23 - 12 May 23	Week 12: Cart context, add to cart and place Order

HTML,CSS and JAVASCRIPT

- https://www.w3schools.com/html/default.asp

GitHub Tutorial:

- https://www.youtube.com/watch?v=xuB1Id2Wxak

ReactJS Training

Notes

- Programming terms you should know
- https://medium.com/codex/25-terms-all-programmers-should-know-10ec373c95f8

- We are learning to react from the below playlist and consider this document as assignment guidance
- https://www.youtube.com/watch?v=QFaFIcGhPoM&list=PLC3y8-rFHvwgg3vaYJgHGnModB54rxOk3&index=1

Day 1 : React introduction (vid. 1-3) (15min)

- Create one react project and render some HTML code to get a basic idea of react rendering

Day 2 : React introduction- Components (vid. 4-6) (20min)

Class components are stateful and functions are stateless components. So, for pages, we should use class components and for all internal components of pages we should use functional component

- Create folder name component and create a file with functional component **Title.js**
- Create folder name pages and create a file with class component **Home.js** and import the above functional component **Title.js** to **Home.js** component and render output

React introduction - Hooks /JSX/Props (vid. 7-9) (15min)

- Render **Title.js** two to three times (will give the same output three times)
- Now pass `props` to **Title.js** as title text and display different titles using the same component

Day 3 : React introduction - State (vid. 10-12) (15min)

As we know, the difference between props and state is that the props hold values for components, and the state holds values within components.

- Create state values in **Home.js** for title and description
- Pass those states as `props` to **Title.js** and display title and description using props destructuring of title and description in **Title.js**

Day 4 : React introduction - Event (vid. 13-15) (20min)

- Create one new class component **Counter.js**

- Create a number state and render that state
- Create a button and create a binding event to increase the value of that numbered state

Day 5 : React introduction - Conditional Rendering/List/List and keys(vid. 16-19)

Task-1 (30min)

- Create component **About.js** and create a page using **Title.js** as a sub-component now we have two pages home and about. display the random descriptions on every page
- Now create a button and binding event in **App.js** for both pages (home and about)
- On press, you should be able to change pages between home and about
- Write if else functions to render separate pages and also try using ternary operator within one render event

Day 6 : Task-2 (20min)

- Create a component **List.js** now create one array variable containing an object with fields title, description, and id
- List those data with Array.map function in render event using **Title.js.**
- Use Title as the value of key props to every returned item for a unique identifier

Day 7 :React introduction - Styling / Form handling (vid. 20-21)

Task-1 (15min)

- Create one CSS file and add it to the **Title.js** component and style titles and description
- Note: React uses `class` as `className`.

Task-2 (30min)

- Create **loginForm.js** component and create the registration form with username, email, password
- Style form with CSS and create submit event with alert by displaying all filled data

Day 8 :React introduction - Component Lifecycle method (vid. 22-24) (60min)

- Read the blog and perform practical within it
- https://medium.com/how-to-react/react-life-cycle-methods-with-examples-2bdb7465332b

React introduction - Fragments/Pure-Components/Memo (vid. 25-27)

Task-1 (PureComponent) (5min)

- React.PureComponent is similar to React.Component. The difference between them is that React.Component doesn't implement `shouldComponentUpdate()`, but React.PureComponent implements it with a shallow prop and state comparison.

- For class components use PureComponent. And for function components will use memo

Task-2 (memo) (20min)

- When the function component renders the same result using the same props, it can be wrapped in `React.memo()` for performance enhancement. This means that React will skip rendering the component and reuse the last rendered result.
- Open **Title.js** file and wrap export function to `React.memo()`
- Ex: `export default React.memo(Title)`
- This title component has only props value so it can be used with React memo

React introduction- Refs/REfs with class components (vid. 28-30) (30min)

- When to use Ref
    - When required to focus on the text, text selection, or playback media.
    - Triggering animations.

- Create class component **InputRef.js** and Create reference state name `inputRef` and add this ref to the input field.
- Create function name focusInput() and write `inputRef.current.focus()`
- Create a new component **FocusInput.js**
- Import **InputRef.js** and render it to the return function and also create ref state `componentRef` pass this ref to imported InputRef component
- Create a button with onClick event which will trigger reference function like `componentRef.current.focusInput()`

- On button click, the input element will get focus, defined in the child component.

React introduction-Portals/error Boundary (vid. 31-32) (25min)

- Read the documentation for `getDerivedStateFromError()` and `componrntDidCatch()`

Day 9 :React introduction - Higher Order Components (vid. 33-35) (45min)

- Create class component UserList.js with a table containing two field id and name
- Create an **HOC.js** component to render high-order components by passing data through props
- **HOC.js** will take two inputs `Hoc(HocComponent, data)` where `HocComponent` is the component to pass and `data` are the data to pass through the component as props
- At **Home.js** import both components **userList.js** and **HOC.js** and create an array of objects containing two field id and names for users

- Create custom component Users by passing **userList.js** and **userData** array to the **HOC** component and rendering Users
- For reference: [ReactJS Higher Order Components Tutorial]()

React introduction - Render Props (vid. 36-37) (15min)

- Render props: we pass a function from the parent component to the child component as a render props, and the child component calls that function instead of implementing its logic.

- Create **RenderPropsComponent.js** component with return value like
```
return (
      {this.props.render()}
)
```
- Now import **RenderPropsComponent.js** to **App.js** and render RenderPropsComponent with passing props as name render and value will be DOM returned function like

```
render={() => {
  return (
      <h3> I am coming from render props </h3>
  )
}}
```

- Now this **RenderPropsComponent** will render `h3` tag to the App component without applying extra logic

Day 10 :React introduction - Context (vid. 38-40) (45min)

- There are four steps to using React context:
  1. Create context using the `createContext` method.
  2. Take your created context and wrap the **context provider** around your component tree.
  3. Put any value you like on your context provider using the `value` prop.
  4. Read that value within any component by using the **context consumer**.
- Refer to this blog for more understanding: https://www.freecodecamp.org/news/react-context-for-beginners/

Day 11 :React introduction - HTTP Methods (vid. 41-43)

Task -1 (45min)

- For HTTP we will use https://jsonplaceholder.typicode.com/users API for development
- Create component **Users.js** and create get API call using Axios npm package
- Use `componentDidMount` for calling API so it will call only the first time when the component is getting loaded/build
- Now save response to state Users

- Create new component **UserDetail.js** in which you will print user details coming from the parent component **Users.js** as props
- In **Users.js** component import **UserDetail.js** component and render the output of the API response by using Array.map() method and pass user data to the UserDetail component as props

```
      return (
           { users.map((user) => <UserDetail user={user} key={user.id
+ user.username } />) }
      )
```

- Note: Above code, we have passed the key to the UsaerDetail component which will separate every user detail component unique and we can not use a number as a key so add any string data with id which will create a unique key

Day 12 :Task-2 (30min)

- For HTTP we will use https://jsonplaceholder.typicode.com/posts API for development
- Create **Post.js** component with a form containing three fields title, body and userId
- Now on submit form make an API call with the POST method

Day 13 :React Hooks - useState (vid. 44-46) (10min)

- Copy **Counter.js** component and rename it as **CounterHook.js** now change this component to a functional component and perform counter with useState hook

React Hooks - useState with object/array and useEffect (vid. 47-50)

Day 14 :Task-1 (45min)

- Create **loginForm.js** functional component with email and password fields and update state values with the useState hook and perform the same scenario of printing all filled data to alert prompt

Day 15 :Task-2 (20min)

- Update **Users.js** file with functional component and change `componentDidMount` to `useEfeect` hook and save data using setState

Day 16 :React Hooks - useEffect as lifecycle events (vid. 51-53) (30min)

- UseEffect returns a value when we got the exit from the component. So, when we are using some static variables/functions within components then it's used to clean up those values

```
      useEffect(() => {
        effect
        return () => {
            cleanup
        }
    }, [input])
```

Day 17 :React Hooks - state update with useEffect (vid. 54-57) (45min)

- For HTTP we will use https://jsonplaceholder.typicode.com/posts/1 API for development
- Create **PostDetail.js** functional component and Create one text input to get numbers
- Now Create a function to fetch posts API by id
- On the change event of input, we will update the state using useState and also fetch post by id.
- The above URL contains 1 as id so we will need to change the value of 1 by input field value and by default set 1 as id

Day 18 :React Hooks - useContext (vid. 58-60) (60min)

- Hint: useContext lets you use context **without** a Consumer
- You can get value from a provider by using useContext like

  ```
  const value = useContext(NumberContext);
  ```

- Update last context demo buy getting values using useContext
- To understand the difference between useContext and Consumer refer to this blog: https://daveceddia.com/usecontext-hook

React Hooks - useReducer (vid. 61-64)

- useReducer takes the same arguments and works the same way as Array.reduce()
- For keeping track of multiple pieces of state that rely on complex logic, useReducer may be useful.
- Basic flow of useReducer() is - action -> dispatch -> reducer -> state

Task-1 (15min)

- Create counter increment component using useReducer

Task-2 (45min)

- I prefer the below example: https://www.w3schools.com/REACT/react_usereducer.asp

React Hooks - useReducer with useContext (vid. 65-68)

Task-1 (20min)

- In the counter increment component used with useReducer. Now, get data by using useContext.

Task-2 (30min)

- For HTTP we will use https://jsonplaceholder.typicode.com/users/1 API for development
- Clone **Users.js** component and update API response with the useReducer dispatch method and list out with state values.

Day 19 :React Hooks - React.memo() (vid. 69-70)

- React.memo(): When a component is wrapped in React.memo(), React renders the component and memoizes the result. Before the next render, if the new props are the same, React reuses the memoized result skipping the next rendering.
- When to use memo()
    - The component should be functional and given the same props, always render the same output
    - The Component renders often
    - The same props were provided to the component during re-rendering
    - Use React.memo() when you have enough amount UI rendering like mid to large components

Task-1 (15min)

- Create Component of **RegisterUser.js** contains form data like username, email, phone number, DOB, etc.
- Get data of input fields using `useCallback` and update the state.

Task-2 (15min)

- Update **UserDetail.js** from user listing demo wrapping with `React.memo()`

React Hooks - Refs (vid. 71-72) (60min)

- Refs can be used in the following cases:
    - When we need DOM measurements such as managing focus, text selection, or media playback.
    - It is used in triggering imperative animations.
    - When integrating with third-party DOM libraries.
    - It can also use as in callbacks.
- Use this link for more understanding: https://blog.logrocket.com/complete-guide-react-refs

React Hooks - custom Hooks (vid. 73-76) (30min)

- Basically custom hooks are set of code to share with multiple components
- Use this link to know more about custom hooks: https://reactjs.org/docs/hooks-custom.html

Day 20 :React render UseState (vid. 77-79) (75min)

- Create a form with 4-5 input fields and create useStates for every input fields
- Display how many times components are being rendered while typing in input fields
- Use useEffect hook and store render count in separate useState and display it in a component.
- Add debounce effect in input fields. Ref: https://www.freecodecamp.org/news/debounce-and-throttle-in-react-with-hooks

Day 21 : React render useReducer (vid. 80 - 81) (30min)

- Create add subtract and multiply button
- Use useReducer hook to handle button click and print output
- Also, print how many times components re-render(use useEffect with empty dependency array) and observer react render behavior.

Day 22 : React render Parent and child (vid. 82 - 83) (40min)

- Create parent and child components.
- Create a useState hook in both components and display the count of render of child component and parent component
- Observe if the parent renders an update of the child component's state or visa-versa.

Day 23 : React render useMemo (vid. 84 - 89) (45min)

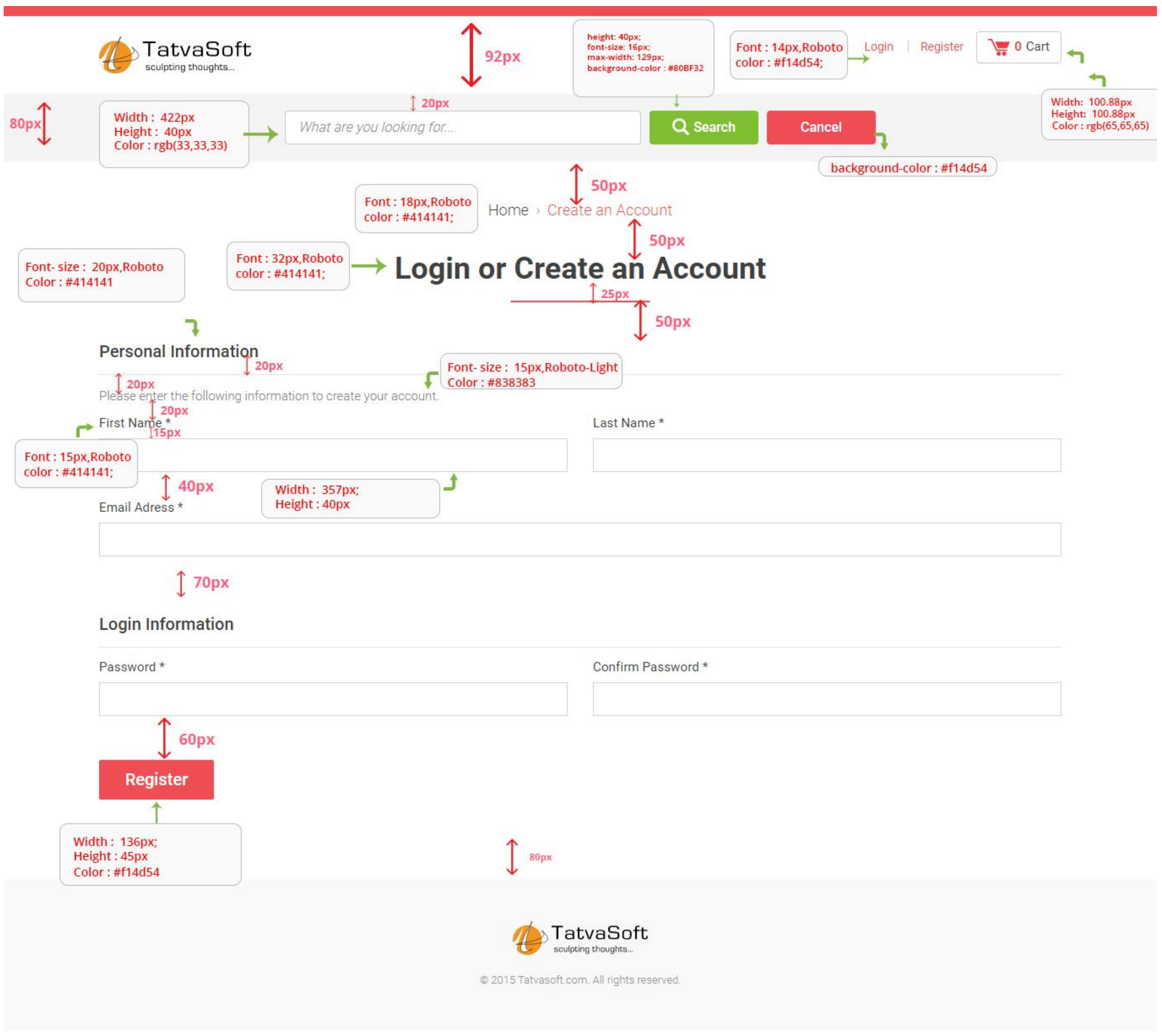
- Create parent and multiple child components
- Create separate useState in components
- Using memo prevent unwanted re-renders on change of states of other components

Day 24 : React renders and Context (vid. 90 - 92) (40min)

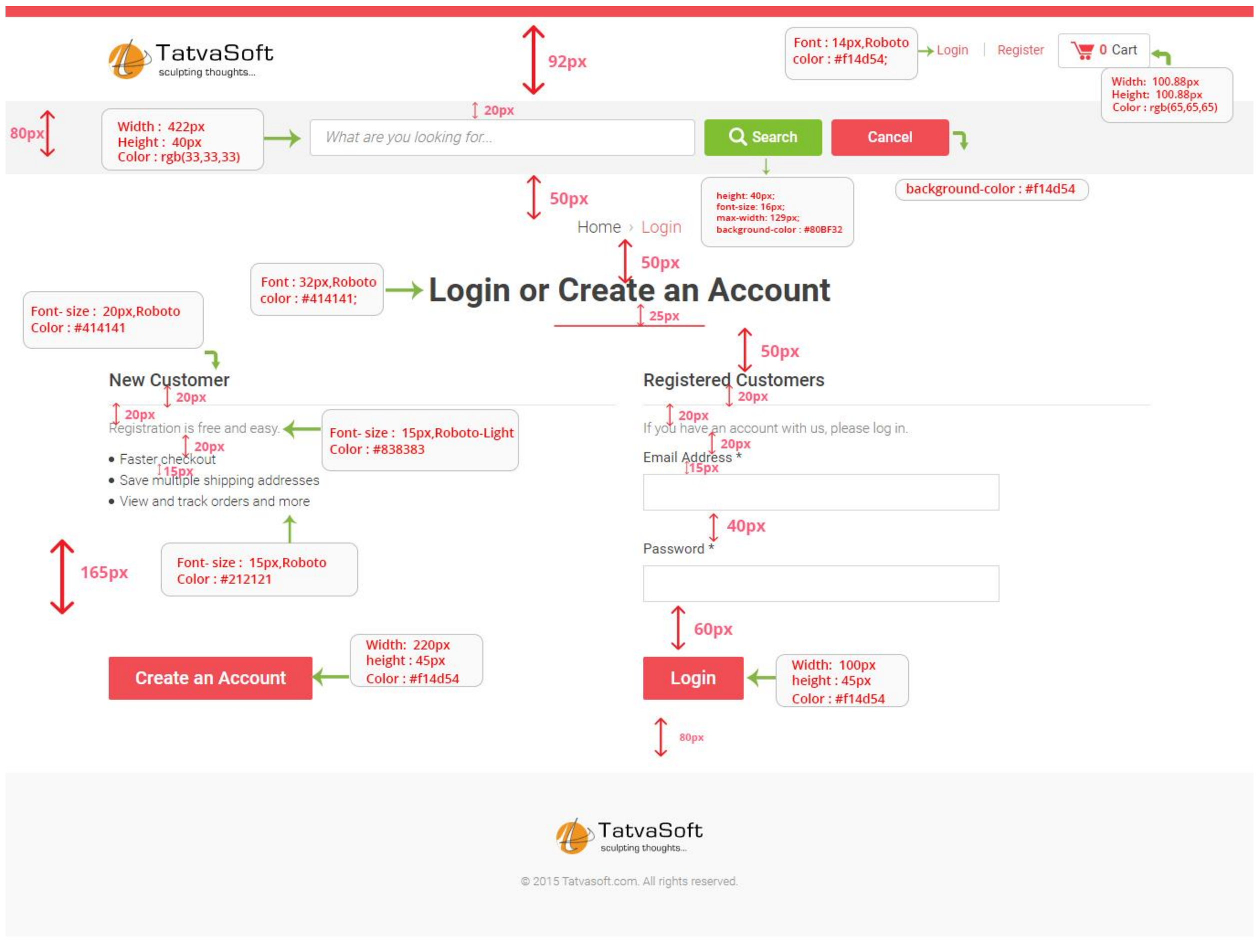
- Create multiple components and pass data of one component to other components using Context.
- All components will display the count of renders
- Observer how many times components are re-renders.

Demo Project (Online Store Application -Book Store)

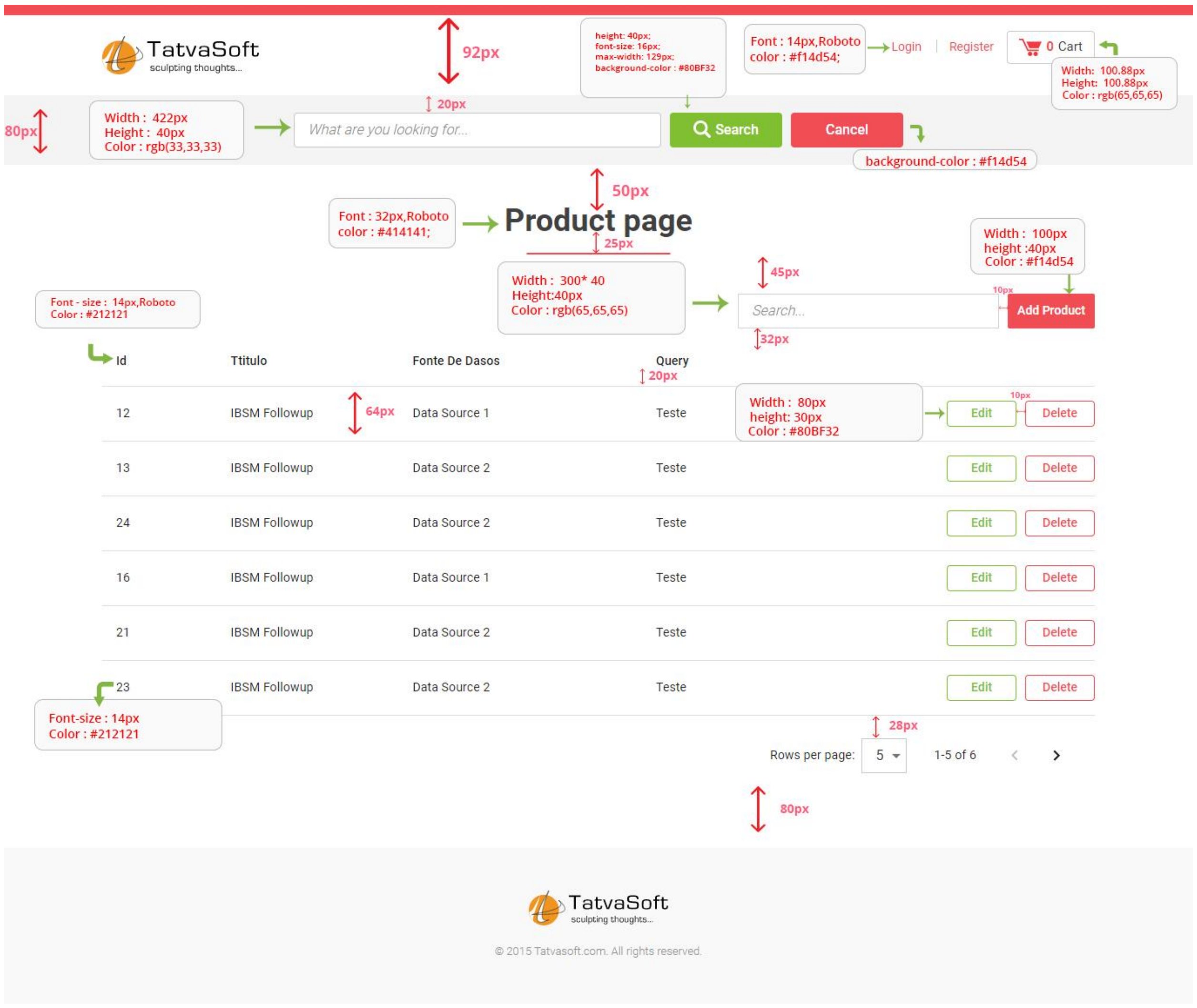
1. Registration


TatvaSoft
sculpting thoughts...
92px
height: 40px; font-size: 16px; max-width: 129px; background-color: #80BF32
Font : 14px,Roboto color : #f14d54;
Login
Register
0 Cart
Width: 100.88px Height: 100.88px Color : rgb(65,65,65)
80px
Width : 422px Height : 40px Color : rgb(33,33,33)
20px
What are you looking for...
Search
Cancel
background-color : #f14d54
50px
Font : 18px,Roboto color : #414141;
Home › Create an Account
50px
Font- size : 20px,Roboto Color : #414141
Font : 32px,Roboto color : #414141;
Login or Create an Account
25px
50px
Personal Information
20px
Font- size : 15px,Roboto-Light Color : #838383
20px
Please enter the following information to create your account.
20px
First Name *
15px
Last Name *
Font : 15px,Roboto color : #414141;
40px
Width : 357px; Height : 40px
Email Adress *
70px
Login Information
Password *
Confirm Password *
60px
Register
Width : 136px; Height : 45px Color : #f14d54
80px
TatvaSoft
sculpting thoughts...
© 2015 Tatvasoft.com. All rights reserved.

2.Login



3. Product CRUD



TatvaSoft
sculpting thoughts...

92px

height: 40px;
font-size: 16px;
max-width: 129px;
background-color : #80BF32

Font : 14px,Roboto
color : #f14d54;

Login | Register

0 Cart

Width: 100.88px
Height: 100.88px
Color : rgb(65,65,65)

80px

Width : 422px
Height : 40px
Color : rgb(33,33,33)

20px

What are you looking for...

Search

Cancel

background-color : #f14d54

50px

Font : 32px,Roboto
color : #414141;

Product page

25px

Width : 100px
height :40px
Color : #f14d54

Width : 300* 40
Height:40px
Color : rgb(65,65,65)

45px

10px

Search...

Add Product

32px

Font - size : 14px,Roboto
Color : #212121

20px

64px

Width : 80px
height: 30px
Color : #80BF32

10px

Id	Ttitulo	Fonte De Dasos	Query	
12	IBSM Followup	Data Source 1	Teste	Edit Delete
13	IBSM Followup	Data Source 2	Teste	Edit Delete
24	IBSM Followup	Data Source 2	Teste	Edit Delete
16	IBSM Followup	Data Source 1	Teste	Edit Delete
21	IBSM Followup	Data Source 2	Teste	Edit Delete
23	IBSM Followup	Data Source 2	Teste	Edit Delete

Font-size : 14px
Color : #212121

28px

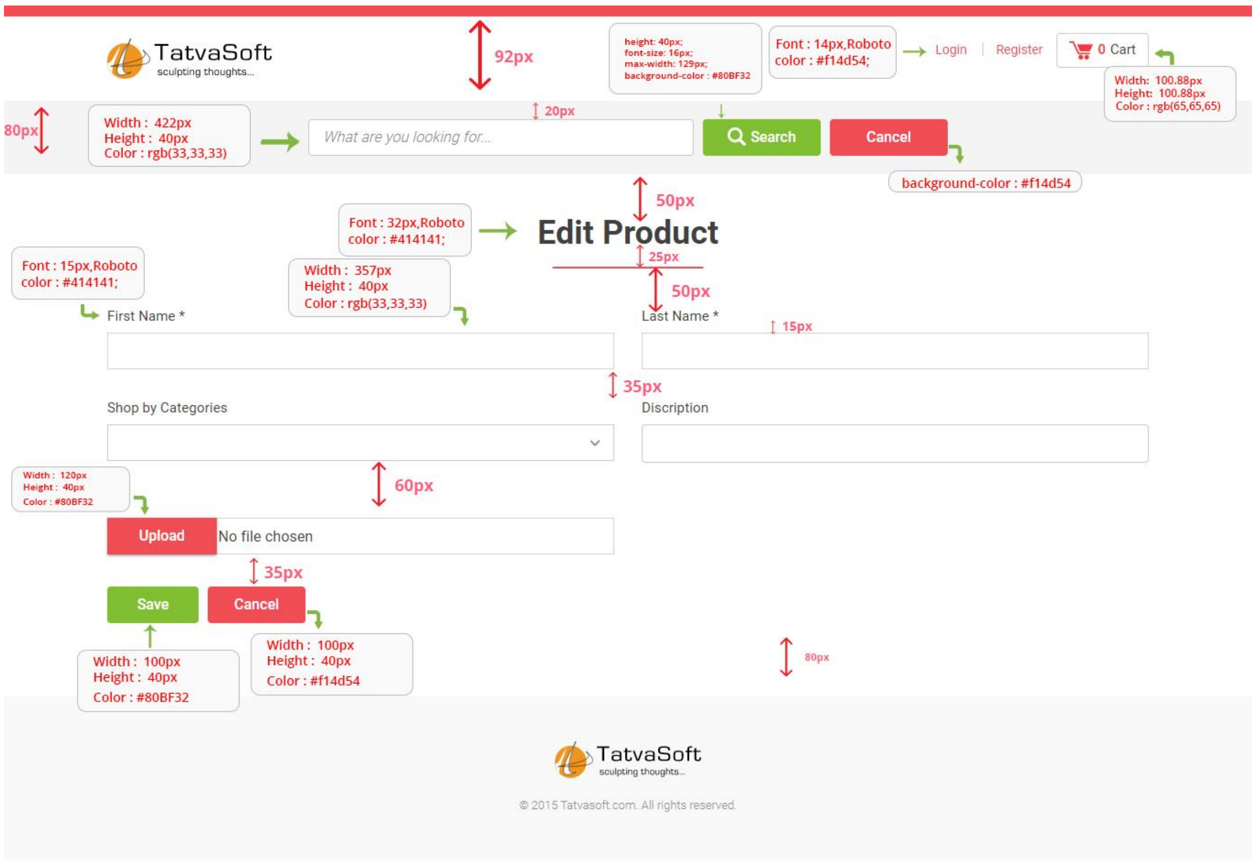
Rows per page: 5 1-5 of 6

80px

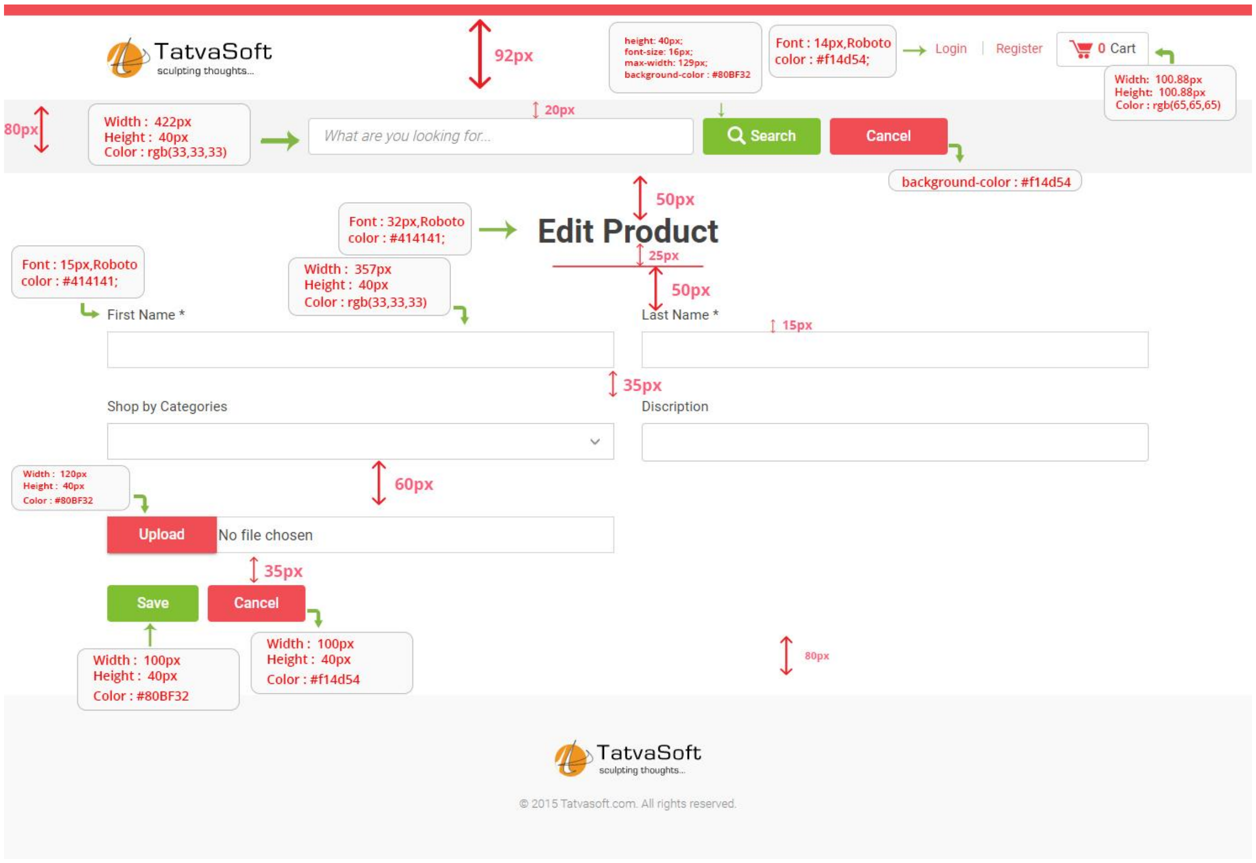
TatvaSoft
sculpting thoughts...

© 2015 Tatvasoft.com. All rights reserved.

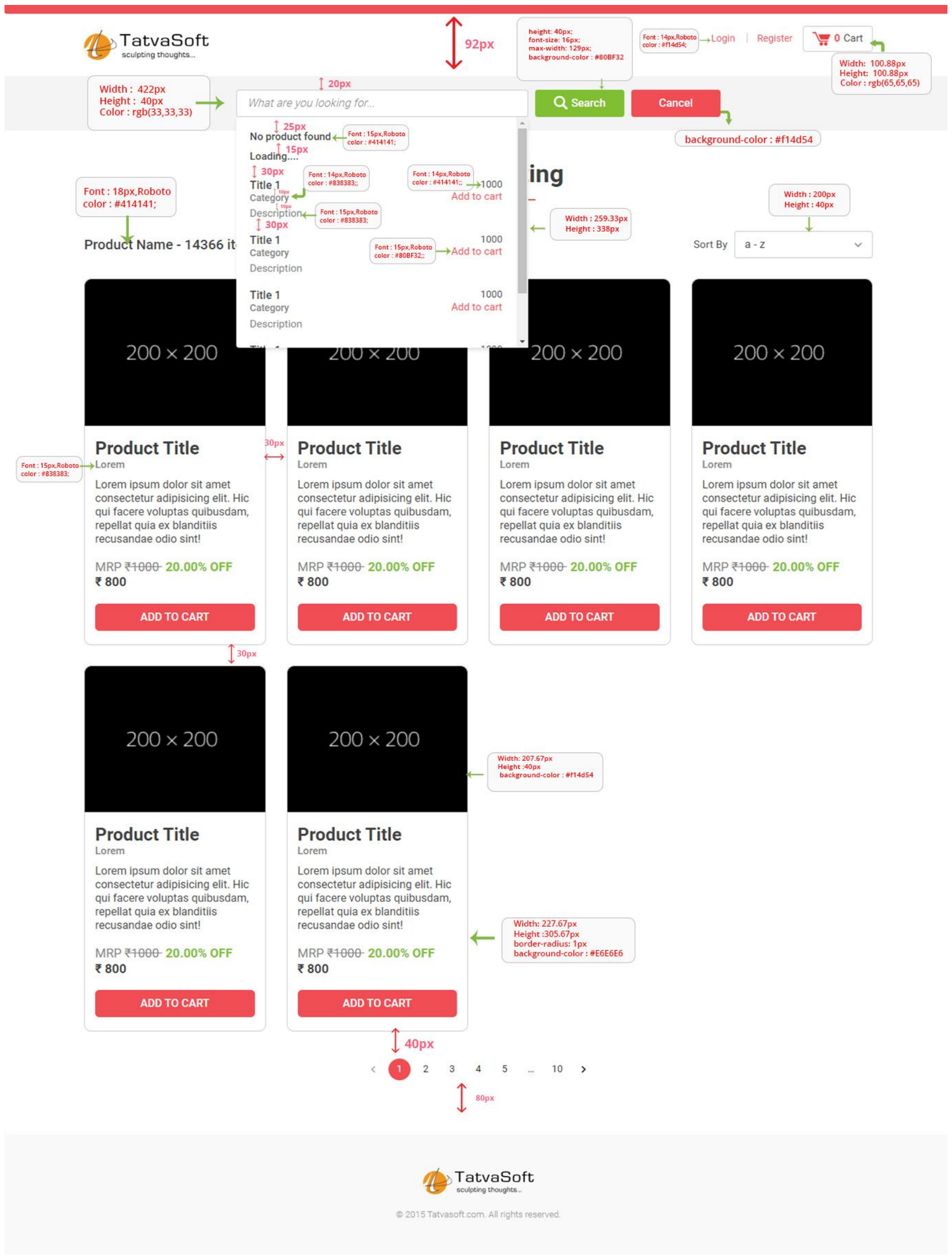
4. Edit Product Page



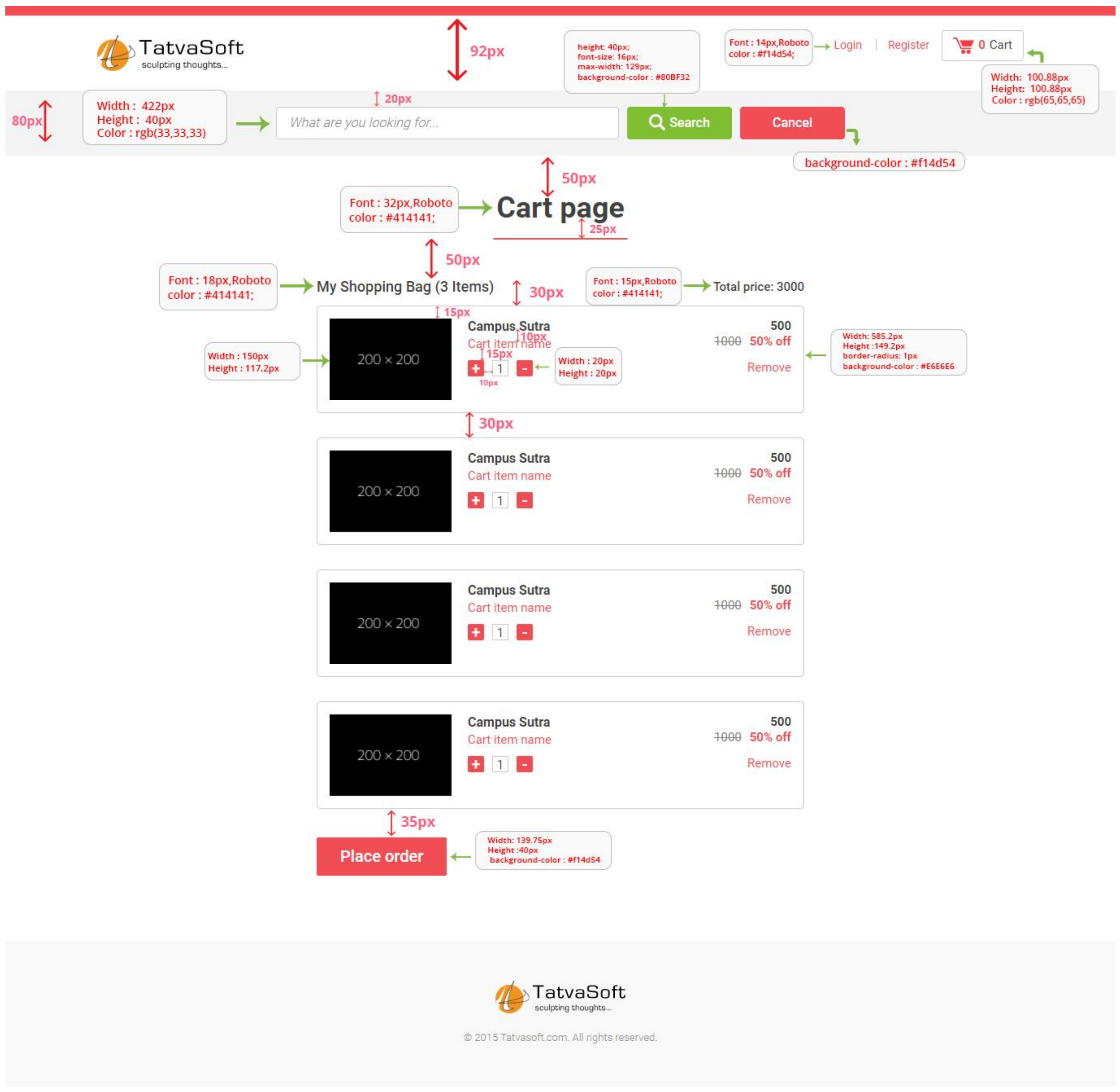
5. Product Listing



6. Product Listing Dropdown


TatvaSoft
sculpting thoughts...
92px
height: 40px;
font-size: 16px;
max-width: 129px;
background-color : #80BF32
Font : 14px,Roboto
color : #f14d54;
Login
Register
0 Cart
Width: 100.88px
Height: 100.88px
Color : rgb(65,65,65)
Width : 422px
Height : 40px
Color : rgb(33,33,33)
20px
What are you looking for...
Search
Cancel
background-color : #f14d54
25px
No product found
Font : 15px,Roboto
color : #414141;
15px
Loading....
30px
Title 1
10px
Category
10px
Description
Font : 14px,Roboto
color : #838383;;
Font : 14px,Roboto
color : #414141;;
1000
Add to cart
Font : 15px,Roboto
color : #838383;
30px
Title 1
Category
Description
1000
Font : 15px,Roboto
color : #80BF32;;
Add to cart
Title 1
Category
Description
1000
Add to cart
ing
Width : 259.33px
Height : 338px
Font : 18px,Roboto
color : #414141;
Product Name - 14366 it
Width : 200px
Height : 40px
Sort By
a - z
200 × 200
Product Title
Lorem
Font : 15px,Roboto
color : #838383;
30px
Lorem ipsum dolor sit amet consectetur adipisicing elit. Hic qui facere voluptas quibusdam, repellat quia ex blanditiis recusandae odio sint!
MRP ~~₹1000~~ 20.00% OFF
₹ 800
ADD TO CART
30px
Width: 207.67px
Height :40px
background-color : #f14d54
Width: 227.67px
Height :305.67px
border-radius: 1px
background-color : #E6E6E6
40px
1
2
3
4
5
...
10
80px
TatvaSoft
sculpting thoughts...
© 2015 Tatvasoft.com. All rights reserved.

7. Cart Page


TatvaSoft
sculpting thoughts...
92px
height: 40px;
font-size: 16px;
max-width: 129px;
background-color : #80BF32
Font : 14px,Roboto
color : #f14d54;
Login
Register
0 Cart
Width: 100.88px
Height: 100.88px
Color : rgb(65,65,65)
80px
Width : 422px
Height : 40px
Color : rgb(33,33,33)
20px
What are you looking for...
Search
Cancel
background-color : #f14d54
50px
Font : 32px,Roboto
color : #414141;
Cart page
25px
50px
Font : 18px,Roboto
color : #414141;
My Shopping Bag (3 Items)
30px
Font : 15px,Roboto
color : #414141;
Total price: 3000
15px
Width : 150px
Height : 117.2px
200 × 200
Campus Sutra
10px
Cart item name
15px
+
1
-
10px
Width : 20px
Height : 20px
500
~~1000~~ 50% off
Remove
Width: 585.2px
Height :149.2px
border-radius: 1px
background-color : #E6E6E6
30px
200 × 200
Campus Sutra
Cart item name
+
1
-
500
~~1000~~ 50% off
Remove
200 × 200
Campus Sutra
Cart item name
+
1
-
500
~~1000~~ 50% off
Remove
200 × 200
Campus Sutra
Cart item name
+
1
-
500
~~1000~~ 50% off
Remove
35px
Place order
Width: 139.75px
Height :40px
background-color : #f14d54
TatvaSoft
sculpting thoughts...
© 2015 Tatvasoft.com. All rights reserved.