1. store==➔index.js

```
import { legacy_createStore as createStore } from 'redux';
import reducer from './reducers';

// ==============================|| REDUX - MAIN STORE
||============================== //

const store = createStore(reducer);
const persister = 'App';

export { store, persister };
```

2. reducers====>actionTypes.js

```
//Courses
export const ADD_COURSE = "ADD_COURSE";
export const GET_ALL_COURSES = "GET_ALL_COURSES";
export const EDIT_COURSES = "EDIT_COURSES";
export const DELETE_COURSE = "DELETE_COURSE";
//Lessons
export const ADD_LESSON = "ADD_LESSON";
export const GET_ALL_LESSONS = "GET_ALL_LESSONS";
export const EDIT_LESSONS = "EDIT_LESSONS";
export const DELETE_LESSON = "DELETE_LESSON";
//The underprivileged
export const GET_ALL_USERS = "GET_ALL_USERS";
export const ADD_USER = "ADD_USER";
export const EDIT_USER = "EDIT_USER";
export const DELETE_USER = "DELETE_USER";
//Trainings
export const GET_ALL_TRAININGS = "GET_ALL_TRAININGS";
export const ADD_TRAINING = "ADD_TRAINING";
export const EDIT_TRAINING = "EDIT_TRAINING";
export const DELETE_TRAINING = "DELETE_TRAINING";
//Assignments
export const GET_ALL_ASSIGNMENTS = "GET_ALL_ASSIGNMENTS";
export const EDIT_ASSIGNMENT = "EDIT_ASSIGNMENT";
export const DELETE_ASSIGNMENT = "DELETE_ASSIGNMENT";
//Certificate
export const GET_ALL_CERTIFICATES = "GET_ALL_CERTIFICATES";
export const EDIT_CERTIFICATE = "EDIT_CERTIFICATE";
export const DELETE_CERTIFICATE = "DELETE_CERTIFICATE";
```

3. reducers====>index.js

```
import { combineReducers } from 'redux';
// reducer import
import userReducer from './userReducer';
```

```
import courseReducer from './courseReducer';
import trainingReducer from './trainingReducer';
import assignmentReducer from './assignmentReducer';
import certificateReducer from './certificateReducer';
import lessonReducer from './lessonReducer';
// ============================|| COMBINE REDUCER
||============================ //
const reducer = combineReducers({
    user: userReducer,
    course: courseReducer,
    training: trainingReducer,
    assignment: assignmentReducer,
    certificate: certificateReducer,
    lesson:lessonReducer
});
export default reducer;
```

4. reducers====>lessonReducer.js

```
import { ADD_LESSON, DELETE_LESSON, EDIT_LESSONS,
GET_ALL_LESSONS } from 'store/actionTypes';

const initlessonState = {
    lessons: []
};

const lessonReducer = (state = initlessonState, action) => {
    switch (action.type) {
        case GET_ALL_LESSONS:
            return {
                ...state,
                lessons: [...action.payload.lessons]
            };
        case ADD_LESSON:
            return {
                ...state,
                lessons: [action.payload.lesson, ...state.lessons]
            };
        case EDIT_LESSONS:
            return {
                ...state,
                lessons: state.lessons.map((lesson) =>
                    lesson.id !== action.payload.lesson.id ? lesson :
action.payload.lesson
                )           };
        case DELETE_LESSON:
            return {
                ...state,
```

```
            lessons: state.lessons.filter((lesson) => lesson.id !==
action.payload.id)
          };
      default:
          return state;
  }    };
export default lessonReducer;
```

5. ***actions===>lessons.js***

```
import * as actions from '../actionTypes';

export const getAllLessons = ({ lessons = [] }) => {
  return {
      type: actions.GET_ALL_LESSONS,
      payload: {
          lessons
      }   };     };
export const addLesson= (lesson) => ({
  type: actions.ADD_LESSON,
  payload: {
      lesson
  }    });

export const editLesson = (lesson) => ({
  type: actions.EDIT_LESSONS,
  payload: {
      lesson
  }      });

export const deleteLesson = (id) => ({
  type: actions.DELETE_LESSON,
  payload: {
      id
  }   });
```

6. ***lessons===>index.js***

```
import Grid from "@mui/material/Grid";
import Card from "@mui/material/Card";
import SoftBox from "components/SoftBox";
import SoftTypography from "components/SoftTypography";
import DashboardLayout from
"examples/LayoutContainers/DashboardLayout";
import LessonCard from "examples/Cards/lessonCard";
import PlaceholderCard from "examples/Cards/PlaceholderCard";
import DashboardNavbar from "examples/Navbars/DashboardNavbar";
import homeDecor1 from "assets/images/home-decor-1.jpg";
import homeDecor2 from "assets/images/home-decor-2.jpg";
```

```
import homeDecor3 from "assets/images/home-decor-3.jpg";
import team1 from "assets/images/team-1.jpg";
import team2 from "assets/images/team-2.jpg";
import team3 from "assets/images/team-3.jpg";
import team4 from "assets/images/team-4.jpg";
import DataWidget from "components/Global/DataWidget";
import { getAllLessons, addLesson, editLesson, deleteLesson } from
"store/actions/lessons";
import { connect } from 'react-redux';
import { useFetcher, API } from "apiFetch";
import Sidebar from 'components/Global/Sidebar';
import { toast } from 'react-hot-toast';
import { compareObj } from "config/constant";
import SoftInput from "components/SoftInput";
import SelectInput from "components/SoftInput/SelectInput";
import ModalDialog from "components/Global/ModalDialog";
import { useState, useEffect } from "react";
import {
  FormControl
} from "@mui/material"
import ChooseFileImage from "components/Global/ChooseFileImage";
import ChooseFileField from "components/Global/ChooseFileField";
import MultipleSelectorInput from
"components/SoftInput/MultipleSelectorInput";
import { uploadSingle } from "config/cloudinary";
import AddMultipleFields from "components/Global/AddMultipleFields";
import { useLocation } from "react-router-dom";

const initFormData = {
  title: '',
  description: '',
  image: '',
  assignmentLink: '',
  trainingId: '',
  content: '',
  materials: [],
  presentations: [],
  videos: [],
  references: [],
};
const initState = { loading: false, error: null };

function Lessons({ lessons, getLessons, addLesson, editLesson, deleteLesson }) {
  const route = useLocation().pathname.split("/").slice(1);
  const [openSidebar, setOpenSidebar] = useState(false);
  const [formData, setFormData] = useState(initFormData);
```

```
  const [state, setState] = useState(initState);
  const [currentCourse, setCurrentCourse] = useState(null);
  const [isUploadingImage, setIsUploadingImage] = useState(false)
  const { data, isError, isLoading } =
useFetcher(`/courses/${route[1]}/viewLessons`);
  const { data: availableTrainings } = useFetcher('/trainings');
  console.log(formData);
  useEffect(() => {
     if (data?.length) {
       getLessons({ lessons: data });
     }
  }, [data?.length]);

  console.log("data",data);

  useEffect(() => {
     if (currentCourse) {
        setFormData({
          title: currentCourse.title,
          description: currentCourse.description,
          image: currentCourse.image,
          assignmentLink: currentCourse.assignmentLink,
          trainingId: currentCourse.trainingId,
          content: currentCourse.content,
          materials: currentCourse.materials,
          presentations: currentCourse.presentations,
          videos: currentCourse.videos,
          references: currentCourse.references,
        });
     } else {
        setFormData(initFormData);
     }
  }, [currentCourse]);

  const handleChange = (name, value) => {
     setFormData((prev) => ({ ...prev, [name]: value }));
  };

  const handleSubmit = async () => {
     setState(initState);
     try {
        setState((prev) => ({ ...prev, loading: true }));
        if (currentCourse) {
           const newObj = compareObj(currentCourse, formData);
           if (!Object.keys(newObj).length) {
              toast.error('No changes made', { position: 'top-center' });
```

```
            return;
          }
          const result = await toast.promise(
            API.patch(`/courses/${currentCourse.id}`, newObj),
            {
              loading: `Updating course, please wait...`,
              success: `Course updated successfully!`,
              error: `Something went wrong while updating this course`
            },
            { position: 'top-center' }
          );
          editLesson({...result.data.data, training: { title:
currentCourse?.training?.title }});
          setCurrentCourse(null);
        } else {
          const result = await toast.promise(
            API.post(`/courses`, formData),
            {
              loading: `Adding course, please wait...`,
              success: `Course added successfully!`,
              error: `Something went wrong while adding this course`
            },
            { position: 'top-center' }
          );
          addLesson(result.data.data);
        }
        setFormData(initFormData);
        setOpenSidebar(false);
    } catch (error) {
        setState((prev) => ({
          ...prev,
          error: error.response?.data?.message || error.message || 'Unknown
error occured, please try again.'
        }));
    } finally {
        setState((prev) => ({ ...prev, loading: false }));
    }     };

  const handleOpenSidebar = () => {
    setOpenSidebar(true);
  };
  const handleCloseSidebar = () => {
    if (state.loading) return;
    setOpenSidebar(false);
    setCurrentCourse(null);
    setState(initState);
```

```
  };

  const [openModal, setOpenModal] = useState(false);
  const handleOpenModal = () => {
     setOpenModal(true);
  };
  const handleCloseModal = () => {
     setOpenModal(false);
     setCurrentCourse(null);
  };

  return (
    <DashboardLayout>
      <DashboardNavbar />
      <SoftBox mb={3} mt={4}>
        <Card sx={{ p: 1 }}>
          <SoftBox pt={2} px={2}>
            <SoftBox pt={2} px={2} mb={3} display="flex" justifyContent="space-
between" alignItems="center">
              <SoftTypography variant="h6" fontWeight="medium">
                All Lessons
              </SoftTypography>
          <Sidebar
          title={currentCourse ? 'Update Lesson' : 'Add New Lesson'}
          openSidebar={openSidebar}
          onOpenSidebar={() => {
             setCurrentCourse(null);
             handleOpenSidebar();
          }}

          onCloseSidebar={handleCloseSidebar}
          handleSubmit={handleSubmit}
          state={state}
        >
          <SoftBox display="flex" flexDirection="column">
            <SoftTypography variant="h6" fontWeight="medium"
color="secondary">
              Select Course
            </SoftTypography>
            <FormControl fullWidth>
              <SelectInput
                value={formData.trainingId}
                onChange={(e) => handleChange('trainingId', e.target.value)}
                placeholder="Choose the course"
                options={
                  availableTrainings?.data?.map((training) => ({
```

```
                value: training.id,
                label: training.title
              }))              }
            />
          </FormControl>
        </SoftBox>
        <SoftBox display="flex" flexDirection="column">
          <SoftTypography variant="h6" fontWeight="medium"
color="secondary">
            Lesson Title
          </SoftTypography>
          <SoftInput
            placeholder="Lesson title"
            type="text"
            value={formData.title}
            onChange={(e) => handleChange('title', e.target.value)}
          />
        </SoftBox>
        <SoftBox display="flex" flexDirection="column">
          <SoftTypography variant="h6" fontWeight="medium"
color="secondary">
            Lesson Description
          </SoftTypography>
          <SoftInput
            placeholder="Lesson description"
            type="text"
            multiline
            rows={4}
            value={formData.description}
            onChange={(e) => handleChange('description', e.target.value)}
          />
        </SoftBox>
        <ChooseFileImage
          selected={formData.image}
          title="Course Image"
          uploading={isUploadingImage}
          onSelect={async(selected) => {
            setIsUploadingImage(true)
            const img = await uploadSingle(selected);
            handleChange('image', img)
            setIsUploadingImage(false)
          }}
        />
        <SoftBox display="flex" flexDirection="column">
          <SoftTypography variant="h6" fontWeight="medium"
color="secondary">
```

```
            Lesson Content
          </SoftTypography>
          <ChooseFileField
            label="Lesson Content"
            error={state.error}
            currentItem={currentCourse}
            currentItemUrl={formData?.content}
            chipLabel="Lesson Content"
            isMultiple={false}
            onChange={(selectedFiles) => handleChange('content', selectedFiles)}
          />
        </SoftBox>
        <SoftBox display="flex" flexDirection="column">
          <SoftTypography variant="h6" fontWeight="medium"
color="secondary">
            Lesson Materials
          </SoftTypography>
          <ChooseFileField
            label="Lesson Materials"
            currentItem={currentCourse}
            currentItemUrl={formData?.materials}
            chipLabel="Material"
            error={state.error}
            isMultiple={true}
            onChange={(selectedFiles) => handleChange('materials', selectedFiles)}
          />
        </SoftBox>
        <SoftBox display="flex" flexDirection="column">
          <SoftTypography variant="h6" fontWeight="medium"
color="secondary">
            Presentations
          </SoftTypography>
          <ChooseFileField
            label="Presentations"
            currentItem={currentCourse}
            currentItemUrl={formData?.presentations}
            chipLabel="Presentation"
            error={state.error}
            isMultiple={true}
            onChange={(selectedFiles) => handleChange('presentations',
selectedFiles)}
          />
        </SoftBox>
        <SoftBox display="flex" flexDirection="column">
          <SoftTypography variant="h6" fontWeight="medium"
color="secondary">
```

```
            Upload Assignment
          </SoftTypography>
          <ChooseFileField
            label="Upload Assignment"
            currentItem={currentCourse}
            currentItemUrl={formData?.assignmentLink}
            chipLabel="Assignment Link"
            error={state.error}
            isMultiple={false}
            onChange={(selectedFiles) => handleChange('assignmentLink',
selectedFiles)}
          />
        </SoftBox>
        <SoftBox display="flex" flexDirection="column">
          <SoftTypography variant="h6" fontWeight="medium"
color="secondary">
            Add Videos
          </SoftTypography>
          <MultipleSelectorInput
            options={[]}
            placeholder='Paste video links here'
            selectedPartners={formData.videos}
            setSelectedPartners={(value) => handleChange('videos', value)}
          />
        </SoftBox>
        <SoftBox display="flex" flexDirection="column">
          <SoftTypography variant="h6" fontWeight="medium"
color="secondary">
            References
          </SoftTypography>
          <AddMultipleFields
            references={formData.references}
            onChange={(value) => handleChange('references', value)}
          />
        </SoftBox>
      </Sidebar>
      </SoftBox>
      </SoftBox>
        <SoftBox p={2}>
        <DataWidget
          title="Lessons"
          isLoading={isLoading && !lessons.length}
          isError={isError && !lessons.length}
          isEmpty={!lessons.length}
        >
          <Grid container spacing={3}>
```

```
        {
          lessons?.map((lesson, index) => (
            <Grid item xs={12} md={6} xl={3} key={index}>
              <LessonCard
                image={lesson.image}
                label={`#${lesson?.training?.title}`}
                title={lesson.title}
                maxTitleCharacters={30}
                maxBodyCharacters={70}
                materials={lesson.materials}
                presentations={lesson.presentations}
                videos={lesson.videos}
                references={lesson.references}
                description={lesson.description}
                onEdit={() => {
                  setCurrentCourse(lesson);
                  handleOpenSidebar();
                }}
                onDelete={() => {
                  setCurrentCourse(course);
                  handleOpenModal();
                }}
                action={{
                  type: "internal",
                  route: "/pages/profile/profile-overview",
                  color: "info",
                  label: "view learners",
                }}
                authors={[
                  { image: team1, name: "Elena Morison" },
                  { image: team2, name: "Ryan Milly" },
                  { image: team3, name: "Nick Daniel" },
                  { image: team4, name: "Peterson" },
                ]}
                  />
                </Grid>
                ))              }
              <Grid item xs={12} md={6} xl={3} sx={{ cursor: "pointer" }}
onClick={handleOpenSidebar}>
                  <PlaceholderCard title={{ variant: "h5", text: "New course" }}
outlined />
              </Grid>
          </Grid>
          </DataWidget>
          <ModalDialog
          title="Delete Course?"
```

```
          subTitle={`Are you sure do you want to delete this course? `}
          item={currentCourse?.title}
          open={openModal}
          handleClose={handleCloseModal}
          cancelText="Cancel"
          showOkText
          handleClickOk={async () => {
            const id = currentCourse?.id;
            const title = currentCourse?.title;
            setOpenModal(false);
            try {
              await toast.promise(API.delete(`/courses/${id}`), {
                loading: `Hold on, we are deleting ${title} from our system.`,
                success: `Course has been deleted successfully`,
                error: (error) => {
                  if (error.response) {
                    return `Error: ${error.response?.data?.message ||
error.message || 'Unknown error occured'}`;
                  } else {
                    return 'Something went wrong while deleting this course,
please try again';
                  }                  }                  });
              deleteLesson(id);
            } catch (error) {
            } finally {
              handleCloseModal();
            }           }}
        />
        </SoftBox>
       </Card>
     </SoftBox>
   </DashboardLayout>
  );   }

const mapStateToProps = (state) => ({
  lessons: state.lesson.lessons
});
const mapDispatchToProps = (dispatch) => {
  return {
    getLessons: (data) => dispatch(getAllLessons(data)),
    addLesson: (data) => dispatch(addLesson(data)),
    deleteLesson: (id) => dispatch(deleteLesson(id)),
    editLesson: (data) => dispatch(editLesson(data))
  };   };

export default connect(mapStateToProps, mapDispatchToProps)(Lessons);
```