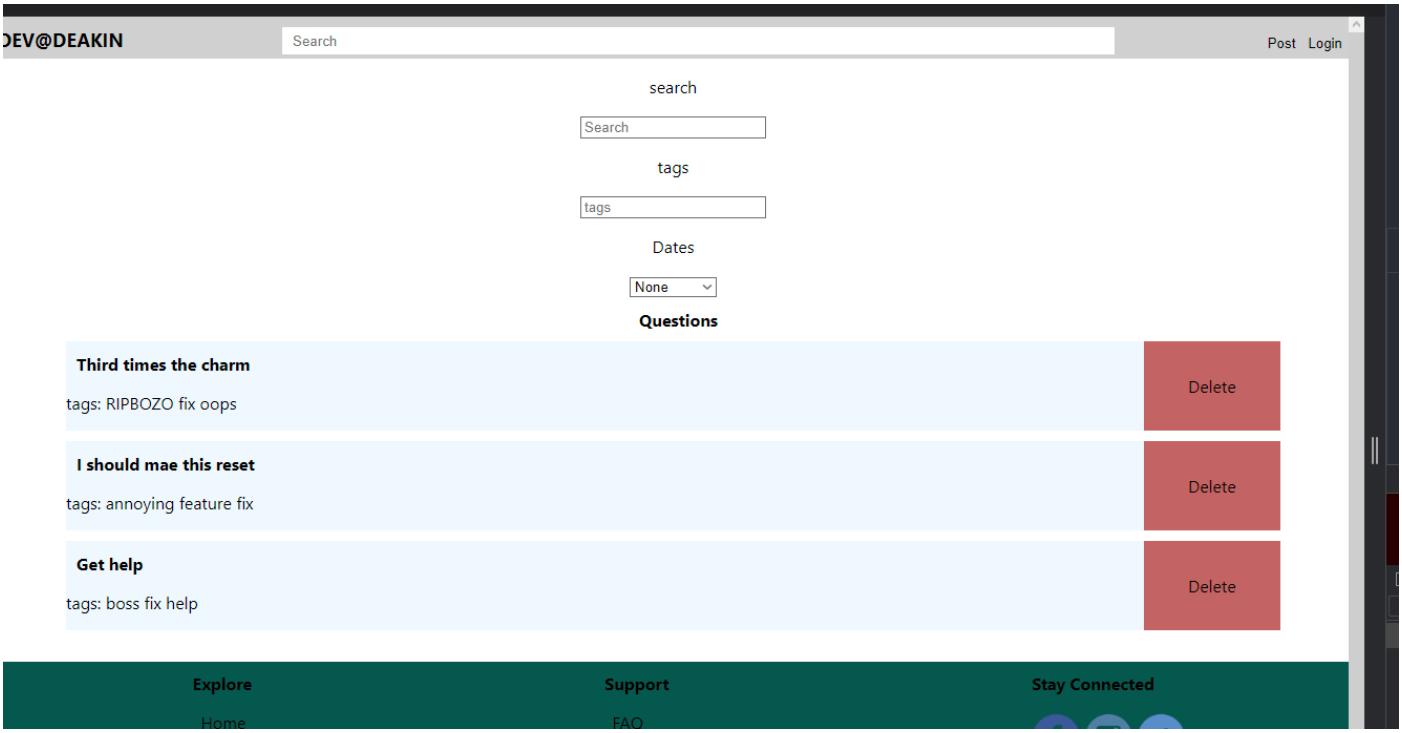
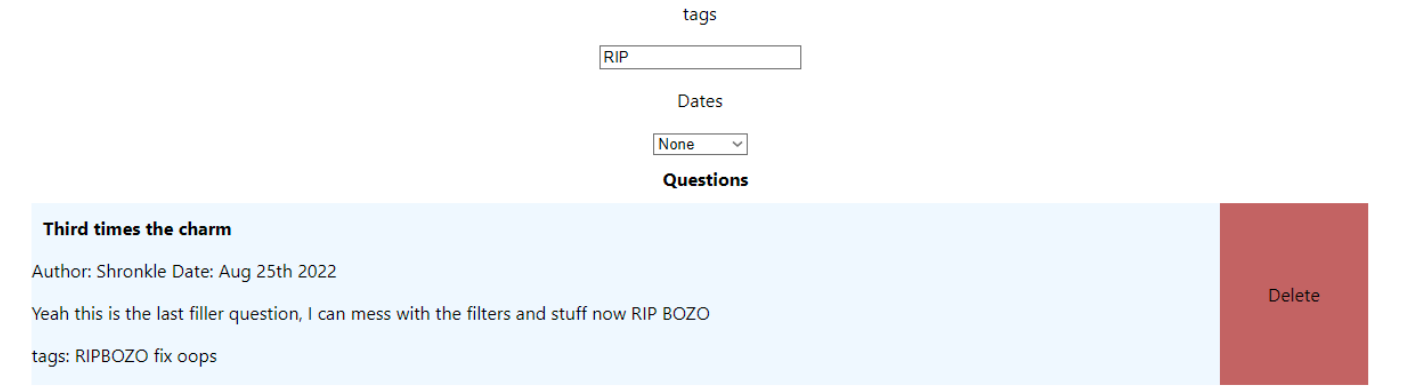
Video : https://video.deakin.edu.au/media/t/1_gtq6a5ao

Git: https://github.com/CameronHart1/SIT313-FullStack

Screenshots:





```jsx
const QuestionView = (props) => {
  const [expanded, setExpanded] = useState(false);
  const { title, rate, author, content, tags, date } = props.question;

  const ToggleExpand = () => {
    setExpanded(!expanded);
  };

  return (
    <div className="QuestionBoxDiv">
      <div className="expandDiv" onClick={ToggleExpand}>
        <h1>{title}</h1>
        {expanded && (
          <div>
            <p>
              Author: {author} Date: {moment(date).format("MMM Do YYYY")}
            </p>
            <p>{content}</p>
          </div>
        )}
        <p className="tags">tags: {tags.join(" ")}</p>
      </div>
      <div className="deleteDiv" onClick={(e) => props.removeBox(props.id)}>
        <p>Delete</p>
      </div>
    </div>
  );
};
```

```jsx
    const keys = Object.keys(questionsList).filter((i) => {
      let qi = questionsList[i];
      const df = filter.dateFilter;
      const dd = moment(qi.date);
      const id = filter.date ? moment(filter.date) : null;
      const dateTest = () => {
        if (df != "none") {
          if (df == "After") return dd.isSameOrAfter(id, "day");
          if (df == "Before") return dd.isSameOrBefore(id, "day");
          if (df == "Same") return dd.isSame(id, "day");
        }
        return true;
      };
      const concatTags = qi.tags.sort().join(" ")

      return `${qi.title} ${qi.content}`.includes(filter.text)
        ? filter.tags.length > 0
          ? filter.tags.every((e)=>concatTags.includes(e))
          : dateTest()
        : false;
    });

    return (
      <div>
        {keys.map((q) => {
          return (
            <QuestionView
              key={q}
              question={questionsList[q]}
              id={q}
              removeBox={removeBox}
            />
          );
        })}
      </div>
    );
  };
```

```jsx
return (
  <div>
    <p>search</p>
    <input
      type="text"
      name="text"
      placeholder="Search"
      onChange={handleChange}
    />
    <p>tags</p>
    <input
      type="text"
      name="tags"
      placeholder="tags"
      onChange={handleChange}
    />
    <p>Dates</p>
    <select id="dateDD" name="dateFilter" onChange={handleChange}>
      <option value="None">None</option>
      <option value="After">After</option>
      <option value="Before">Before</option>
      <option value="Same">Same day</option>
    </select>
    {filter.dateFilter != "None" && (
      <input
        type="date"
        name="date"
        placeholder={new Date()}
        onChange={handleChange}
      />
    )}
    <h1>Questions</h1>
    <QuestionList
      questionsList={questionsList}
      filter={filter}
      removeBox={removeBox}
```

```jsx
const QuestionPage = (props) => {
  const context = useContext(PostContext);

  const [questionsList, setQuestionsList] = useState(
    context.currentPosts.questions
  );
  const [filter, setFilter] = useState({
    tags: [],
    text: "",
    dateFilter: "none",
    date: new Date(),
  });

  const handleChange = (e) => {
    const { name, value } = e.target;
    setFilter((preValue) => {
      if (name == "tags")
        return {
          ...preValue,
          [name]: value.split(/(?:,| |#)+/).filter((s) => s),
        };
      return {
        ...preValue,
        [name]: value,
      };
    });
  };

  // it's wierd but mutating the array with delet doesnt live update so instead we filter out the deleted ID
  const removeBox = (id) => {
    console.log("removed " + id);
    const tmp = Object.keys(questionsList).
    filter((k) => k != id).
    reduce((cur, k) => { return Object.assign(cur, { [k]: questionsList[k] })}, {});
    setQuestionsList(tmp);
  };
```

```jsx
import { useReducer } from "react";
import { useState, useEffect } from "react";
import { createContext } from "react";
import { fetchQuestionsAndTutorials } from "../utils/firebase";
// import { addCollectionAndDocument } from "../utils/firebase";

// Basicly making it stored in session, so updating page doesnt remove everything
let reducer = (info, newInfo) => {
  if (newInfo === null) {
    sessionStorage.removeItem("Posts");
    return null;
  }
  return { ...info, ...newInfo };
};

const localState = JSON.parse(sessionStorage.getItem("Posts"));

export const PostContext = createContext({
  currentPosts: null,
  setCurrentPosts: () => null,
});

export const PostProvider = ({ children }) => {
  // useEffect(()=>{
  //     addCollectionAndDocument('posts',)
  // },[])
  const [currentPosts, setCurrentPosts] = useReducer(
    reducer,
    localState || null
  );
  const value = { currentPosts, setCurrentPosts };

  useEffect(() => {
    sessionStorage.setItem("Posts", JSON.stringify(currentPosts));
    if (currentPosts == null) {
      fetchQuestionsAndTutorials().then((data) => setCurrentPosts(data));
    }
  }, [currentPosts]);
```

```jsx
export const addCollectionAndDocument = async (collectionKey, objectsToAdd) => {
  const collectionRef = collection(db, collectionKey);
  const batch = writeBatch(db);
  objectsToAdd.forEach((obj) => {
    const docRef = doc(collectionRef, obj.id.toLowercase());
    batch.set(docRef, obj);
  });
  await batch.commit();
  console.log("succesful batch commit");
};

// ------------------------------------
export const fetchQuestionsAndTutorials = async () => {
  const articleCollectionRef = collection(db, "Articles");
  const aq = query(articleCollectionRef);
  const articleSnaphsot = await getDocs(aq);
  const ArticleMap = articleSnaphsot.docs.reduce((acc, docSnapshot) => {
    const { ...items } = docSnapshot.data();
    console.log(docSnapshot);
    acc[docSnapshot.id] = items;
    acc[docSnapshot.id].date = new Date(acc[docSnapshot.id].date.seconds * 1000);
    return acc;
  }, {});

  const questionCollectionRef = collection(db, "Questions");
  const q = query(questionCollectionRef);
  const questionSnaphsot = await getDocs(q);
  console.log(questionSnaphsot);
  const QuestionMap = questionSnaphsot.docs.reduce((acc, docSnapshot) => {
    const { ...items } = docSnapshot.data();
    acc[docSnapshot.id] = items;
    acc[docSnapshot.id].date = new Date(acc[docSnapshot.id].date.seconds * 1000);
    return acc;
  }, {});
  return { questions: QuestionMap, articles: ArticleMap };
};

export const getUserData = async (ref) => {
  const snap = await getDoc(ref);
```