w7-starter

Navbar.js

```
const NavBar = (props) => {
  const [loggedIn, setLoggedIn] = useState(false);

  const handleLogin = (res) => {
    // 'res' contains the response from Google's authentication servers
    console.log(res);
    setLoggedIn(true);
  };

  const handleLogout = () => {
    console.log("Logged out successfully!");
    setLoggedIn(false);
  };
```

```
{loggedIn ? (
  <GoogleLogout
    clientId={GOOGLE_CLIENT_ID}
    buttonText="Logout"
    onLogoutSuccess={handleLogout}
    onFailure={(err) => console.log(err)}
    className="NavBar-link NavBar-login"
  />
) : (
  <GoogleLogin
    clientId={GOOGLE_CLIENT_ID}
    buttonText="Login"
    onSuccess={handleLogin}
    onFailure={(err) => console.log(err)}
    className="NavBar-link NavBar-login"
  />
)}
```

w7-step1

User.js

```
Cory Lynch, 4 years ago | 1 author (Cory Lynch)
1  const mongoose = require("mongoose");
2
3  const UserSchema = new mongoose.Schema({
4    name: String,
5    googleid: String,
6  });
7
8  // compile model from schema
9  module.exports = mongoose.model("user", UserSchema);
10
```

w7-step2

Part A: api.js

```
router.post("/login", auth.login);
router.post("/logout", auth.logout);
```

Part B: navbar.js

```
const handleLogin = (res) => {
  // 'res' contains the response from Google's authentication servers
  console.log(res);

  setLoggedIn(true);
  const userToken = res.tokenObj.id_token;
  post("/api/login", { token: userToken }).then((user) => {
    // the server knows we're logged in now
    console.log(user);
  });
};

const handleLogout = () => {
  console.log("Logged out successfully!");
  setLoggedIn(false);
  post("/api/logout");
};
```

w7-step3-v2

navbar.js

```
useEffect(() => {
  get("/api/whoami").then((user) => {
    if (user._id) {
      // they are registed in the database, and currently logged in.
      setUserId(user._id);
    }
  });
}, []);
```

```
62   router.get("/whoami", (req, res) => {
63     if (req.user) {
64       res.send(req.user);
65     } else {
66       // user is not logged in
67       res.send({});
68     }
69   });
```

w7-step3

Part A: story.js, comment.js

```
//define a comment schema for the database
const CommentSchema = new mongoose.Schema({
  creator_id: String,
  creator_name: String,
  parent: String, // links to the _id of a parent
  content: String,
});
```

```
3   //define a story schema for the database
4   const StorySchema = new mongoose.Schema({
5     creator_id: String,
6     creator_name: String,
7     content: String,
8   });
```

w7-step3

Part B: api.js

```
router.post("/story", (req, res) => {
  // TODO: Introduce creator_id
  // TODO: Use the real creator name
  const newStory = new Story({
    creator_id: req.user._id,
    creator_name: req.user.name,
    content: req.body.content,
  });

  newStory.save().then((story) => res.send(story));
});
```

```
router.post("/comment", (req, res) => {
  // TODO: Introduce creator_id
  // TODO: Use the real creator name
  const newComment = new Comment({
    creator_id: req.user._id,
    creator_name: req.user.name,
    parent: req.body.parent,
    content: req.body.content,
  });

  newComment.save().then((comment) => res.send(comment));
});
```

w7-step4

app.js

```
<Router>
  <Feed path="/" />
  <Profile path="/profile/:userId" />
  <NotFound default />
</Router>
```

feed.js

```
<Card
  key={`Card_${storyObj._id}`}
  _id={storyObj._id}
  creator_name={storyObj.creator_name}
  creator_id={storyObj.creator_id}
  content={storyObj.content}
/>
```

w7-step4

card.js

```
<SingleStory
  _id={props._id}
  creator_id={props.creator_id}
  creator_name={props.creator_name}
  content={props.content}
/>
```

singlestory.js

```
<div className="Card-story">
  <Link to={`/profile/${props.creator_id}`} className="u-link u-bold">
    {props.creator_name}
  </Link>
  <p className="Card-storyContent">{props.content}</p>
</div>
```

w7-step5

api.js — add user route

```
42  router.get("/user", (req, res) => {
43    User.findById(req.query.userid).then((user) => {
44      res.send(user);
45    });
46  });
47
```

w7-step5

profile.js — initialize user state with new endpoint, and update name

```
useEffect(() => {
  get("/api/user", { userid: props.userId }).then((user) => {
    setUser(user);
  });
}, []);
```

```
return (
  <>
    {!user ? (
      <div> Loading! </div>
    ) : (
      <>
        <div
          className="Profile-avatarContainer"
          onClick={() => {
            incrementCatHappiness();
          }}
        >
          <div className="Profile-avatar" />
        </div>
        <h1 className="Profile-name u-textCenter">{user.name}</h1>
```

w7-step6

navbar.js

```
const [userId, setUserId] = useState(null);

const handleLogin = (res) => {
  // 'res' contains the response from Google's authentication servers
  console.log(res);

  const userToken = res.tokenObj.id_token;
  post("/api/login", { token: userToken }).then((user) => {
    // the server knows we're logged in now
    setUserId(user._id);
    console.log(user);
  });
};

const handleLogout = () => {
  console.log("Logged out successfully!");
  post("/api/logout");
  setUserId(null);
};

return (
  <nav className="NavBar-container">
    <div className="NavBar-title u-inlineBlock">Catbook</div>
    <div className="NavBar-linkContainer u-inlineBlock">
      <Link to="/" className="NavBar-link">
        Home
      </Link>
      <Link to={`/profile/${userId}`} className="NavBar-link">
        Profile
      </Link>
      {userId ? (
```

w7-step7

api.js — set up whoami route

```
62    router.get("/whoami", (req, res) => {
63      if (req.user) {
64        res.send(req.user);
65      } else {
66        // user is not logged in
67        res.send({});
68      }
69    });
```

w7-step9

app.js

```
<Router>
  <Feed path="/" userId={userId} />
  <Profile path="/profile/:userId" />
  <NotFound default />
</Router>
```

feed.js

```
return (
  <>
    {props.userId && <NewStory addNewStory={addNewStory} />}
    {storiesList}
  </>
);
```