User Tokens/Cookies

Backend:

Use package of **jsonwebtoken**

```
const jwt = require("jsonwebtoken");
```

Upon user login, if the username and password matched, the **jwt** will generate a token for us to encrypt the user's infomation and the backend will put it into response header.

```
  const token = jwt.sign(
    { _id: user._id, name: user.name, avatarUrl: user.avatarUrl },
    // eslint-disable-next-line comma-dangle
    process.env.TOKEN_SECRET
  );
  res.header("auth-token", token);
  return res
    .status(200)
    .send({ success: true, msg: "Login Successful", token });
};
```

And some of the router will use the token to verify the user

```
// Route middleware
app.use("/api/", authRoute);
app.use("/api/user", userRoute);
app.use("/api/post", verifyToken, postRoute);
app.use("/api/group", verifyToken, groupRoute);
app.use("/api/comment", verifyToken, commentRoute);
app.use("/api/chat", verifyToken, chatRoute);
app.use("/api/message", verifyToken, messageRoute);
app.use("/api/file", fileRoute);
app.use("/api/notification", verifyToken, notificationRoute);
```

```
const verifyToken = (req, res, next) => {
  const token = req.header("auth-token");
  // If no token assigned to the request, reject the access
  if (!token) {
    return res.status(401).send({ error: "Access Denied" });
  }
```

```
  try {
    const verified = jwt.verify(token, process.env.TOKEN_SECRET);
    req.user = verified;
    next();
  } catch (err) {
    res.status(400).send({ error: "Invalid Token" });
  }
};
```

Frontend:

In frontend, we will use these utils to get/store/delete the token in **localStorage** so that when the logined, they do not need to login again next time open the app.

```
export function getToken() {
  return localStorage.getItem("token");
}

export function setToken(token) {
  localStorage.setItem("token", token);
}

export function clearToken() {
  localStorage.removeItem("token");
}

export function isLogined() {
  if (localStorage.getItem("token")) {
    return true;
  }
  return false;
}
```

And, for convenience, we also store the decoded user's information from the token into our redux store upon login.

```
function App() {
  if (isLogined()) {
    // Redux
    const dispatch = useDispatch();

    // Decode token stored in local storage
    const tokenCoded = getToken();
```

```
  const tokenDecoded = jwtDecode(tokenCoded);

  // AddUser
  useEffect(() => {
    if (tokenDecoded) {
      // Set user in global state
      dispatch(setCurrentUser(tokenDecoded));
      socket.emit("addUser", tokenDecoded._id);
    }
  }, [tokenDecoded]);
}
```