Pesto

- Back End
    - config
        - db.js - Connect to mongodb
        - default.json
            - Commmon configuration File
            - secretOrKey : "Hash to be used"
        - passport.js
            - USed for uthenticating JSON web Tokens
    - models
        - Stores the schemas
        - User Schems + Pesto Schema
    - Routes
        - admin - Stores all admin endpoints
        - pestos - stores all pesto related endpoints
        - users.js - stores all users related endpoints
    - Utils
        - Login.js - Validates Login parameters
        - Register.js - Validates register parameters
    - Server.js
        - Default File to load
        - Has all middleware functions
- Front End
    - Public
        - images - Has all images
        - index.css - contains all the self defined css.
        - index.html - Contains the base html for the entire react front end.
    - Src
        - Actions
            - authActions
                - registerUser,loginuser,setcurrentuser ....
                -
        - store.js
            - createStore() creates a Redux store that holds the complete state tree of your app. There should only be a single store in your app.
            - Reducers are pure functions that specify how application state should change in response to an action. Reducers respond with the new state, which is passed to our store and, in turn, our UI.

- Reducers
    - Auth Reducer - Basically handles the cases where an error occurs in the authorization and how the state tree has to be reloaded.
    - Error Reducer : Returns the payload for an invalid response.
    - Combine oth the auth and error reducers to one using the root reducer in index.js
- Utils
    - AuthHandler.js
        - Validates the token
        - Sends the auth Header stating that the token is valid or not
        - As to whether login should be permitted or not.
- As for the exports in some modules

```
export default connect(
mapStateToProps,
{ registerUser }
)(withRouter(Register));
```

- connect : connects our React components to our Redux store provided by the Provider component
- We need to ise with router in order to redirect inside an action
- this.props.history.push('/dashboard') - This does not work in actions
- mapStateToProps allows us to get our state from Redux and map it to props which we can use inside components.

```
  export
  const mapStateToProps = state => ({
  auth: state.auth,
  errors: state.errors
  });
```

- We get our redux reducers by using mapstatetoprops
- This allows us to call this.props.auth or this.props.errors within our Register component.
- We use a proptypes constructor

```
Register.propTypes = {
registerUser: PropTypes.func.isRequired,
auth: PropTypes.object.isRequired,
```

```
errors: PropTypes.object.isRequired
};
```

- Register.proptypes : This is how we recieve without any props neing passed
- Because all that is ahndled by redux which is used for state management

- private-route
  - Created in order to make sure user is authenticated before he accesses a website.
  - Controls the passing of props

Refer These Links as Well

- https://blog.bitsrc.io/build-a-login-auth-app-with-mern-stack-part-1-c405048e3669
- https://blog.bitsrc.io/build-a-login-auth-app-with-mern-stack-part-2-frontend-6eac4e38ee82
- https://blog.bitsrc.io/build-a-login-auth-app-with-the-mern-stack-part-3-react-components-88190f8db718