Joplin A note taking and to-do application with synchronisation capabilities

Project

Introduction:

The Joplin app, Joplin is an open source note taking and to-do application, organised into notebooks. The current application lacks the support for multiple profiles(#591) and Password-protected notes. The aim of this project is to introduce these essential functionalities. I aim to contribute to the mobile version of the application. Since these features in my opinion will help the end-user in entrusting the product in keeping their secrets (password functionality) and will help the end-user to share their sets of todos and notes in one go, these issues are my target through this project. My solution includes adding a new screen upon opening of the app for profile selection and adding a password option to the individual notes created by the end-user.

Goals:

Through this project, I promise to deliver the following features to the Joplin community:

- Profile Selection page on startup of the app.
- Dropdown to change the profile.
- Add password protection to individual notes.

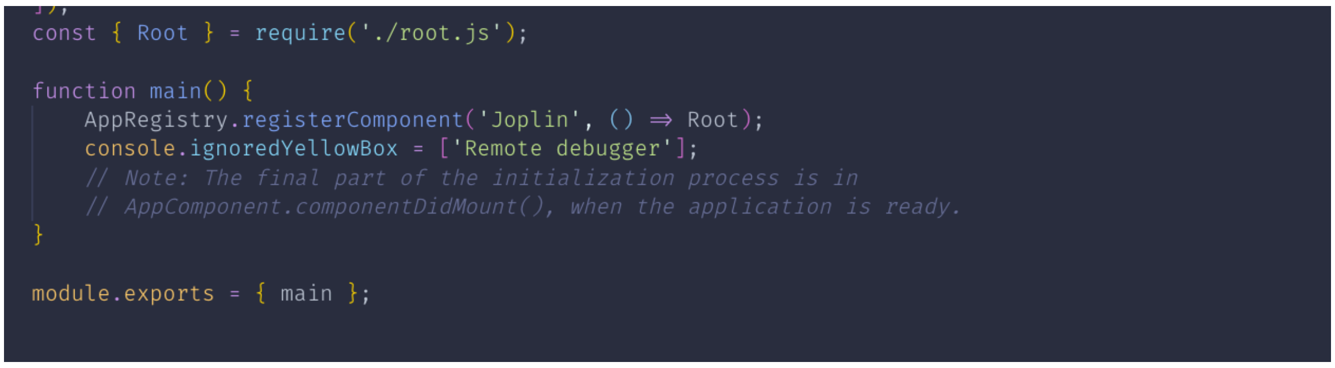
If possible, I will try to add a password feature on the profiles as well.

Implementation:

The current app is structured in such a way that making it a profile based application will ask for a lot of refactoring and restructuring the app. So to achieve my goal of introducing the concept of profiles I intend to work by utilizing the power that react offers us at every step of the development process.

Multiple Profile Problem:

The current implementation of the app includes a component called ‘root’ which gets rendered on app load.

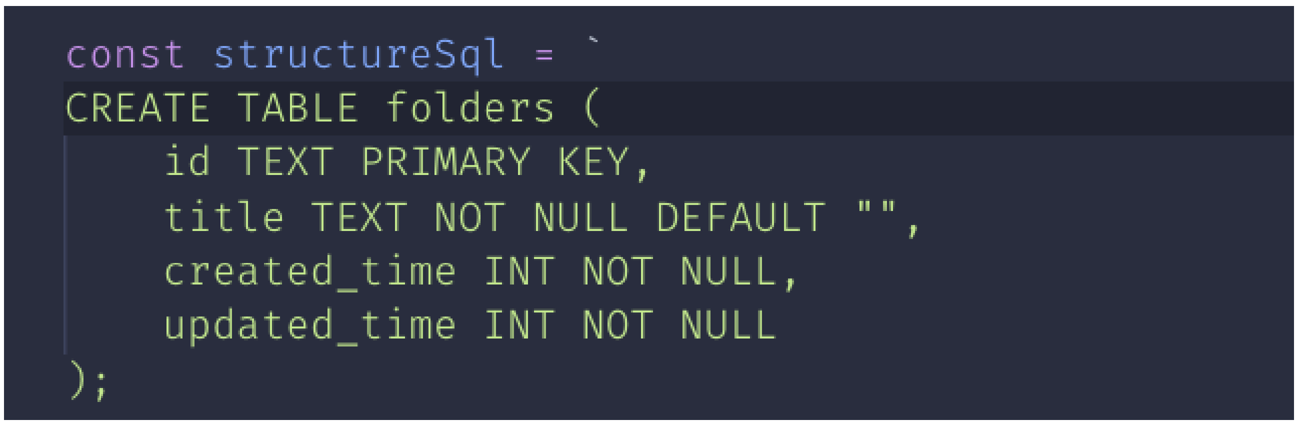


```
const { Root } = require('./root.js');

function main() {
    AppRegistry.registerComponent('Joplin', () => Root);
    console.ignoredYellowBox = ['Remote debugger'];
    // Note: The final part of the initialization process is in
    // AppComponent.componentDidMount(), when the application is ready.
}

module.exports = { main };
```

And data is stored using sqlite database. The biggest problem that we have in our current setup, in terms of setting up profiles will be about data storage. Current schema pattern include structure like:



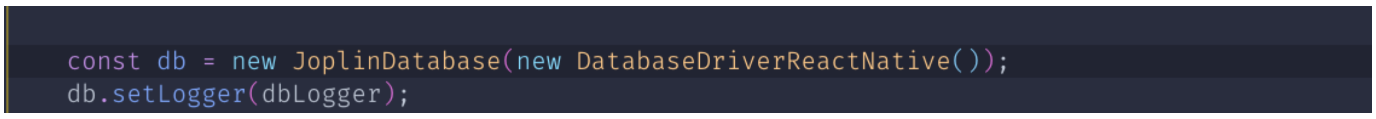
```
const structureSql = `
CREATE TABLE folders (
    id TEXT PRIMARY KEY,
    title TEXT NOT NULL DEFAULT "",
    created_time INT NOT NULL,
    updated_time INT NOT NULL
);
```

This implementation restricts our app to maintain multiple instances (or profiles) since only one table exists for a particular purpose in the current implementation.

My Solution:

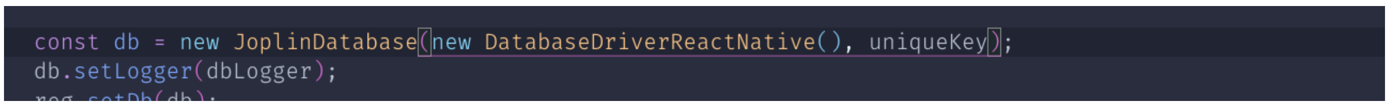
Firstly, finding a solution on how to store data profile wise takes the top priority. The application uses the predefined table names in joplin-database.js file across the entire application for accessing data. The application currently is built using react and redux combination. The entire joplin-database.js file can be refactored by wrapping the create-table snippet into a function and calling it by passing some arguments which are unique to a particular profile, and then attaching that argument to the table name before its created for unique record system across the entire application. The application also uses redux for its structuring. Creating a global state, consisting of the unique profile argument, which can therefore be used for accessing and rendering data based on profile across our application.

In our root.js file:



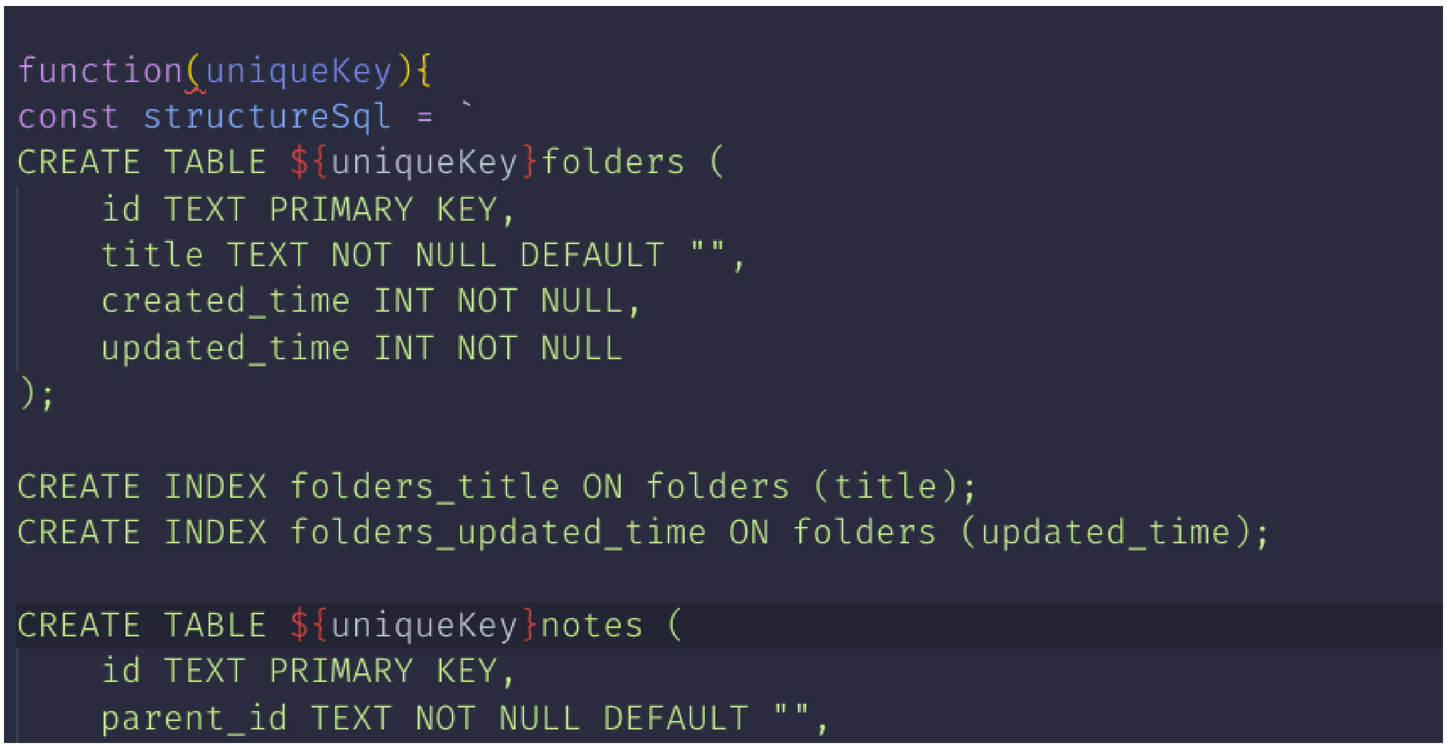
```
const db = new JoplinDatabase(new DatabaseDriverReactNative());
db.setLogger(dbLogger);
```

This is the syntax used for creating database instance. If we add an additional argument to the constructor consisting of our unique key I.e. something like this:



```
const db = new JoplinDatabase(new DatabaseDriverReactNative(), uniqueKey);
db.setLogger(dbLogger);
reg.setDb(db);
```

And then passing this "uniqueKey" to the file in which we have moved our code snippet for creation of the database, as an argument, and appending the argument to the table name making the table unique according to each profile. The queries will look something like this:

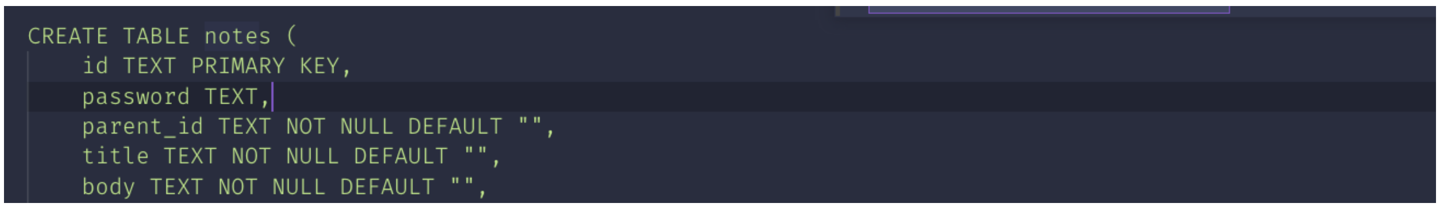


```
function(uniqueKey){
const structureSql = `
CREATE TABLE ${uniqueKey}folders (
    id TEXT PRIMARY KEY,
    title TEXT NOT NULL DEFAULT "",
    created_time INT NOT NULL,
    updated_time INT NOT NULL
);

CREATE INDEX folders_title ON folders (title);
CREATE INDEX folders_updated_time ON folders (updated_time);

CREATE TABLE ${uniqueKey}notes (
    id TEXT PRIMARY KEY,
    parent_id TEXT NOT NULL DEFAULT "",
```

Now, pushing the root.js component one step down the hierarchy and making a new profile component that wraps it, which when rendering its child component 'root.js' now, passes along a 'uniqueKey' prop using which the database gets initialized. We will maintain a separate table in our database containing information about the profiles. The new database table is created whenever the user asks for the creation of a new profile. The unique key is attached as a prop and passed down whenever a given profile gets selected. So in simple words the content of the root.js files will be rendered using another parent 'profile' component which will handle the problem of the unique key in the application. For safer database management system we will use the IF NOT EXISTS" clause of Sqlite in our query to avoid any discrepancy. So there will be a little modification in our sql queries also. The creation of tables will be done only during the profile creation.



```
CREATE TABLE notes (
    id TEXT PRIMARY KEY,
    password TEXT,
    parent_id TEXT NOT NULL DEFAULT "",
    title TEXT NOT NULL DEFAULT "",
    body TEXT NOT NULL DEFAULT "",
```

Password Protection:

Adding security will require minor changes in the note-item component. Firstly, I will change the query structure of the note adding a password field. Password field can be null. Now if the password field is not null I will ask the user for the password. The password will be stored in the database in encrypted manner. I will use some library, bcrypt for instance to achieve this. A similar implementation will be observed in case of making the profiles password protected (only if time permits me to do so).

Timeline

- Week 1-2
- Week 3-4
- Week 5-6
- Week 6-7
- Week 8-9

About Me

- Name : Rachit Anand Srivastava
- Institue : Thapar Institute of Engineering and Technology
- Email : rac.sri25@gmail.com
- Contact : +91 7071690418
- Country of Residence : India
- Timezone : IST (GMT + 05:30)

Technical Knowledge

I am a sophomore at Thapar Institute of Technology pursuing Computer Engineering as my major field. I work in the field of Web/App development and Blockchain. My specialization is in Full Stack development. I have done two internships, one in my freshman year in a Company named *Started*

India, and the other (currently ongoing) in my second year in a company named *Zuzu.AI* as an extension developer. I am working in the field of MERN Stack for almost 1.5 years now and am able to create production-ready products on my own.

Resources that I have referred to learn and grow includes various courses from Udemy, Coursera, Frontendmasters. I also read a lot of articles online and I always try to keep up with the latest technology via platforms like LinkedIn.

Some of my projects include:

- Farmer Portal: An application built to connect farmers and dealers directly and eliminate any middleman in between to maximize the farmer's earning. Goods price (which are generally decided by auction in India) was decided using an auction system using Blockchain. The entire application was made using Node, ReactJs, MongoDB.
- Funding India: An application built to manage the payments done to all the different political parties. The purpose was to eliminate the collection of black money by the different parties. Stack used was: Node, ReactJs, MongoDB, Etherium.
- A chatbot using NodeJs for Facebook messenger.
- A movie service system using ReactJs and NodeJs.
- E-Health care: System designed to ease the communication between doctors and patients for faster delivery of healthcare services.
- A video calling platform to perform court trials in case the presence of the witness etc. needed in the courtroom is difficult.
- An app to keep track of your daily exercises: using React Native.
- A Bookstore system using NodeJs.
- A Blog System implemented using NodeJs.
- A ping pong game made using vanilla javascript.
- Currently making a “Smart Suggestion Extention” using ReactJs.
- Currently making my own Instagram Clone using React Native and Redux.

Languages/Frameworks/Runtime Environment/Libraries/Tools I am proficient at:

- Javascript
- Java
- NodeJs
- ReactJs
- React Native
- Webpack

- GraphQl
- Etherium(Solidity)
- Sockets
- Redux (+ contextApi)

Reach Me:

- rac.sri25@gmail.com
- github.com/rachit2501
- linkedin.com/in/rachit-anand-srivastava-345307173/