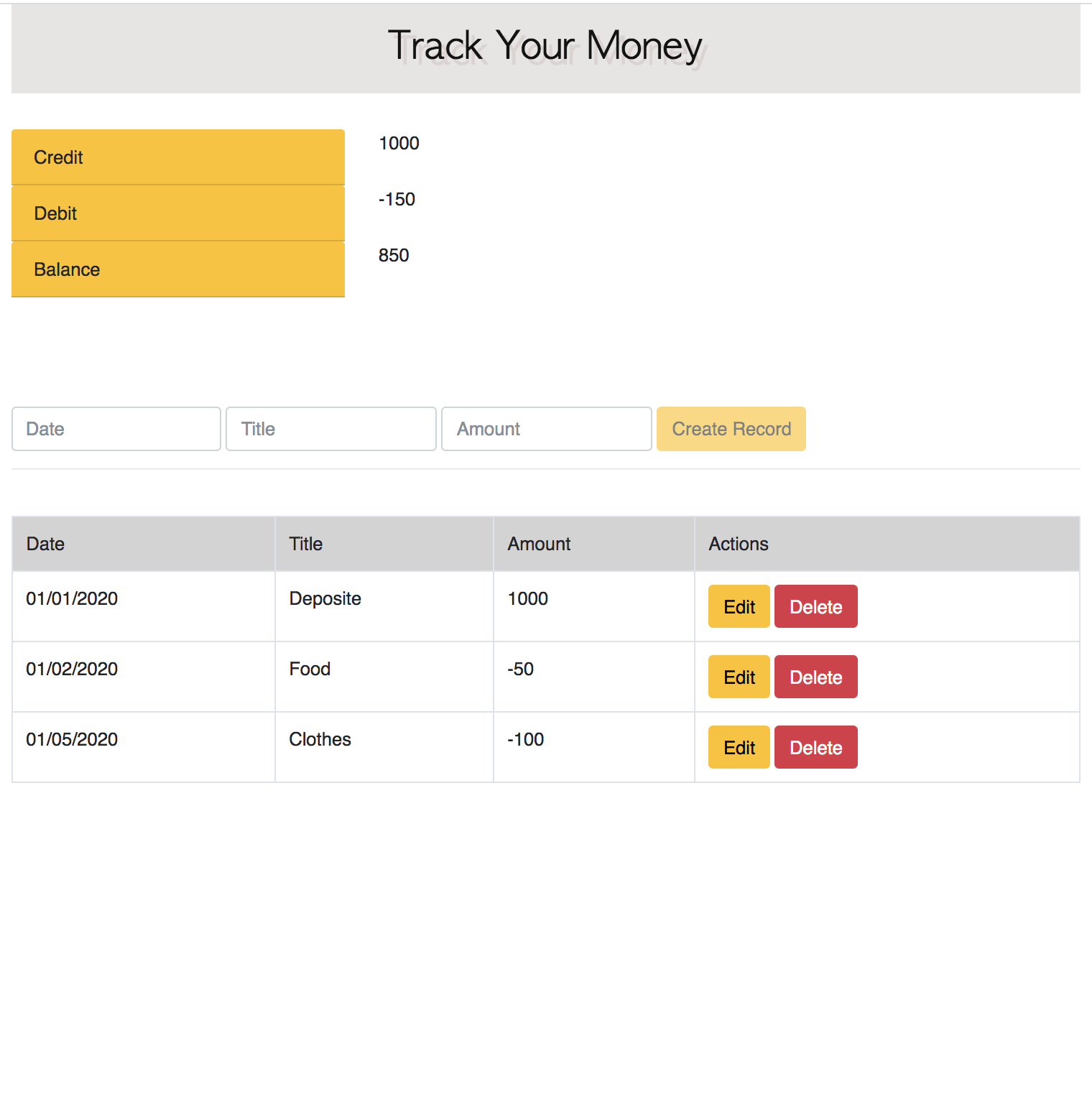
**App - *Track Your Money***

**About**

* Tracking your money through application proving CRUD using React.



**Structure**

**1. Components Structure**

App

|\_\_\_ Summary

|\_\_\_ Form \_ \_ \_ \_ RecordsAPI

| |

|\_\_\_ Table |

|\_\_\_\_\_\_\_ TableRow

|\_\_\_\_ TableDisplay

|\_\_\_\_ TableEdit

* **App Component:** is the data hub
  + state: records for entries in Table component
  + state: isLoaded show loading info, before showing Table
  + state: error
  + method: componentDidMount() get data from database
  + method: addRecord() passed to Form
  + method: deleteRecord() passed to Table and TableRow
  + method: updateRecord() passed to Table and TableRow
  + method: credits() passed to Summary
  + method: debits() passed to Summary
  + method: balance() passed to Summary
* **Summary Component:** displays negative/ positive summation for the records
* **Form Component:** add record into records, updating Table
* **Table Component:** just pass records to **TableRow**
* **TableRow Component:** display each record in records
* **RecordsAPI:** utility providing interface to make RESTful API call

**2. File Structure**

|\_\_node\_modules

|

|\_\_public

| |\_\_\_ index.html

|

|\_\_ src

|\_\_\_ utils: RecordsAPI.js

|

|\_\_\_ components:

|\_\_\_ App.js

|\_\_\_ Form.js

|\_\_\_ Summary.js

|\_\_\_ Table.js

|\_\_\_ TableRow.js

**Concept**

**1. Component data flow: Model-> View**

* Parent component M-> Child component V
  + Between parent component prop and child component prop
* Child component M-> Parent component V
  + Callback function + The parent component passes the callback function to the child component + In JS, function is a first-class citizen, so the value passed in will be saved as its own field; different from C / Java.
* sibling pass value between components M-> V
  + Must rely on the common parent component of the two to pass
  + But when the relationship between components becomes more and more complicated, this way of relying on the parent component as a middleman to pass values ​​should be a mess!
  + Redux comes into picture

**2. Two-way binding: Model <-> View**

* By binding <input>the onChange()Monitor View transformation
* Update the value of the component in onChange Handler to complete the data flow of View => Model.

**3. React life cycle**

* **Mount**
  + constructor()
  + componentWillMount()
  + render()
  + componentDidMount()
* **Update**
  + componentWillReceiveProps (): will receive new props
  + shouldComponentUpdate (): Should it be updated?
  + componentWillUpdate (): The component will be updated soon
  + render (): the component is rendered
  + componentDidUpdate (): component completes update
* **Unmount**
  + componentWillUnmount (): Do some data removal before the component is unmounted