WA3257 Project for Loblaws
Full Stack Developers
Program

Web Age Solutions Inc.
USA: 1-877-517-6540
Canada: 1-877-812-8887
Web: http://www.webagesolutions.com

For customizations of this book or other sales inquiries, please contact us at:

USA: 1-877-517-6540, email: getinfousa@webagesolutions.com
Canada: 1-877-812-8887 toll free, email: getinfo@webagesolutions.com

Web Age Solutions Inc.
821A Bloor Street West
Toronto
Ontario, M6G 1M1

Table of Contents

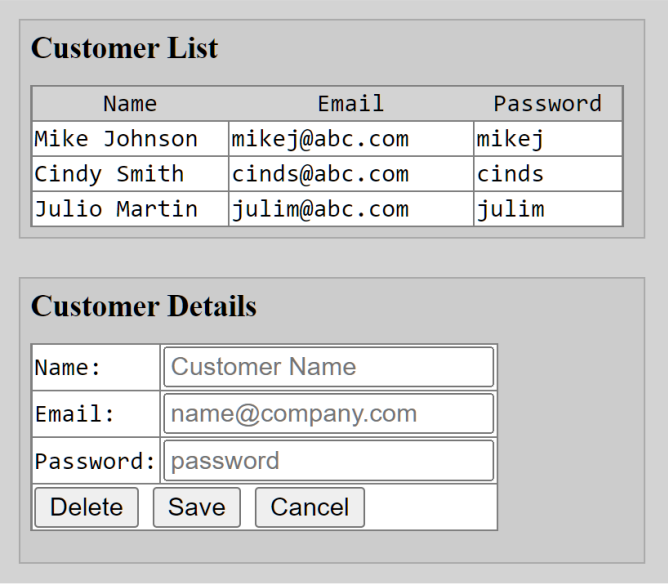
Project Phase 1 - HTTP CSS & JavaScript.....5
    Functional Requirements.....6
Project Phase 2 - React Client.....9
    Functional Requirements.....10
    Manual Test Scripts.....11
Project Phase 3 - Java Back-End.....17
Project Phase 4 - Final Assembly.....19

Project Phase 1 - HTTP CSS & JavaScript

Overview

This project uses HTML/CSS/JavaScript to build a Customer management application. The application in phase01 is a standalone application with hard coded data. In phase02 the application will be refactored to use React components for the front-end. In phase03 a RESTful back-end data source will be developed and the React app from phase02 will be refactored to exchange data through the REST back-end.

Screenshot



Project Files

Your application files should include one HTML file, one CSS file and one JavaScript file.

```
\project-phase01
   index.html
   styles.css
   app.js
```

Tooling

The development environment should include:

1. A web browser to display the application. Chrome or Firefox are preferred.
2. A web server to serve the application. Http-server, which requires nodejs, is preferred. Instructions on setting it up appear later in this document
3. A code editor. Visual Studio Code is preferred and should be available on the lab machines.

Serving the Application

The application should be made available on the URL: http://localhost:8080

This procedure can be used:

1. Open a command prompt or terminal
2. Navigate into the project directory "project-phase01"
3. Execute this node package manager command:

```
npx http-server -d .
```

4. To see your index.html page, open a browser to: http://localhost:8080

Application Requirements

You should code your application so that it fulfills these requirements.

Functional Requirements

1. The app should display a list of Customer records
2. The label "Customer List" should appear on-screen, above the list of Customer records
3. The following fields should be maintained for each Customer: name, email, password
4. Users can select a record by clicking on it in the list.
5. Selected records should appear with a **bold** font.
6. Clicking on an already selected record should remove the selection.
7. A section labeled "Customer Details" should appear below the Customer list
8. The "Customer Details" section should show the name, email and password of the record selected in the Customer list.
9. Three buttons, delete, save and cancel should appear below the Customer Details section
10. Clicking the delete button when a record is selected should delete the selected record.
11. Deleted record should no longer show in the Customer List
12. Users should be able to modify any of the data (name, email or password) in the Customer details section.
13. Clicking the save button when a selected record has been modified in the Customer Details section should replace the original record with the modified values.
14. The customer list should be updated after a modified record has been saved.
15. Clicking on the Cancel button should de-select the selected record.
16. After clicking cancel the fields in the Customer Details section should be empty.
17. When no record is selected the user can type new data into the fields in the Customer Details section.
18. Clicking Save when no record is selected and when data has been entered into the Customer Details fields should add a new record.
19. After a new record has been added the Customer List should be updated to include the new record.

Suggested Order of Work

1. Create an index.html file in the project directory. Include HTML, header and body sections.
2. Add an <h4> header labeled “Customer List” to the body of the index.html file
3. Set up the web server and try opening index.html in a browser. Index.html should be opened in the browser using a localhost URL. Do not open the file directly from the file system.
4. Start adding JavaScript in a <Script> section in index.html or in a separate app.js file. When adding code in a separate app.js file remember to connect the index.html to the app.js.
5. Add an array to your JavaScript code. You can use this to get started:

```
let customerArray = [
    { name: 'Mike Johnson', email: 'mikej@abc.com', password: 'mikej' },
    { name: 'Cindy Smith', email: "cinds@abc.com", password: 'cinds' },
    { name: 'Julio Martin', email: 'julim@abc.com', password: 'julim' }
]
```

6. Start implementing items from the requirements list. Where possible implement them in the order they are listed.
7. As you work, save your files frequently and update the browser page to check your work.
8. If you are unsure how to implement a requirement you can check with the instructor, but make sure you have though about it a bit first before asking for help. The more specific your question the better the instructor will be able to help.

Things to Consider

Keep the following in mind:

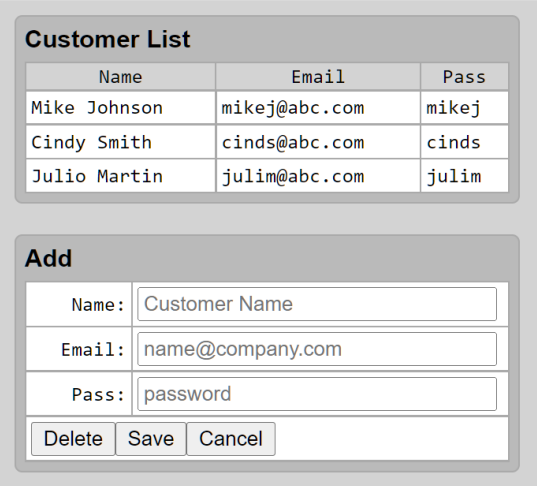
- The Customer List and Customer Details sections will look best if you format them using an HTML table.
- JavaScript arrays include methods for inserting and updating records.
- Keep the developer’s tools open in the browser for the tab where you are displaying your application. An easy way to open the developer’s tools is the F12 key.
- Use the screenshot at the beginning of this document as a guide when implementing visual features.

Project Phase 2 - React Client

Overview

In this phase of the project (phse02) React is used to build the Customer management application. The application in phase02 gets its data from a 'json-server' REST server. In phase03 students will build a RESTful back-end data source to use with the React app from phase02. The resulting solution will satisfy the overall project requirement for an end-to-end, full-stack solution.

Screenshot



Project Files

You will create the initial project using the create-react-app utility. When fully built-out the application project should include the following files:

```
wa3228-project-phase02
|    .gitignore
|    package.json
|    README.md
|
|----public
|        favicon.ico
|        index.html
|        logo192.png
|        logo512.png
|        manifest.json
|        robots.txt
|
|----src
     |    App.css
     |    App.js
     |    index.css
```

```
|   index.js
|   logo.svg
|   reportWebVitals.js
|   setupTests.js
|
└───components
        customer-add-update.js
        customer-list.js
```

Tooling

The development environment should include:

1. A web browser to display the application. Chrome or Firefox are preferred.
2. NodeJS. It's used for creating the initial project and for running your React application in development mode.
3. A code editor. Visual Studio Code is preferred and should be available on the lab machines.

Serving the Application

The application should be made available on the URL:

http://localhost:3000

For the application to function the 'json-server' REST server application must be running and available on port :4000. The files required to run 'json-server' are available in the ProjectFiles\phase02\server directory.

http://localhost:4000

Follow these instructions to start the REST server:

1. Open a command prompt or terminal
2. Navigate into the ProjectFiles\phase02\server directory
3. Execute this node package manager command:

```
npm start
```

Application Requirements

You should code your application so that it fulfills these requirements.

Functional Requirements

1 The app should display a list of Customer records
2 The label "Customer List" should appear on-screen, above the list of Customer records
3 The following fields should be maintained for each Customer: name, email, password
4 Users can select a record by clicking on it in the list.
5 Selected records should appear with a **bold** font.

6 Clicking on an already selected record should remove the selection.
7 The section below the Customer list should hold an add-update form. The form's title shoud be either "Add" or "Update" depending on the forms current mode. (Add for new records, Update when it is showing the selected record)
8 The add-update form section should show the name, email and password of the record selected in the Customer list.
9 Three buttons, delete, save and cancel should appear below the Add-update form section
10 Clicking the delete button when a record is selected should delete the selected record.
11 Deleted records should no longer show in the Customer List
12 Users should be able to modify any of the data (name, email or password) in the Add-update form section.
13 Clicking the save button when a selected record has been modified in the Add-update form section should update the original record with the modified values.
14 After a record is modified the customer list should be updated so that the modified field values are visible in the list
15 Clicking on the Cancel button should de-select the selected record.
16 After clicking cancel the fields in the Add-update form section should be empty.
17 When no record is selected the user can type new data into the fields in the Add-update form section.
18 Clicking Save when no record is selected and when data has been entered into the Add-update form fields should add a new record.
19 After a new record has been added the Customer List should be updated to include the new record.

Manual Testing

As you develop the app you should periodically run through these manual test steps to make sure you have not lost functionality. The number of the test scripts match the number of the requirement they satisfy:

Manual Test Scripts

1 The app should display a list of Customer records:
  1.1 Refresh the app in the browser
  1.2 The customer list should show all available customer records

2 The label "Customer List" should appear on-screen, above the list of Customer records
  2.1 Refresh the app in the browser
  2.2 The "Customer List" text should appear as defined in the requirements

3 The following fields should be maintained for each Customer: name, email, password
- 3.1 Refresh the app in the browser
- 3.2 The customer list should show the fields name, email and password for each record
- 3.3 The add-update form should allow entry of text for the fields; name, email and password

4 Users can select a record by clicking on it in the list.
- 4.1 See 5. Below

5 Selected records should appear with a **bold** font.
- 5.1 Click on an item in the Customer list
- 5.2 The three data fields for the item should change to bold text.
- 5.3 Field data from the selected record should appear in the add-update edit form
- 5.4 Click again on the already selected and bolded item in the customer list
- 5.5 The three data fields for the item should no longer be bolded
- 5.6 The add-update form should no longer show field data from the previously selected record, instead it should show the input field's placeholder text

6 Clicking on an already selected record should remove the selection.
- 6.1 See 5.4 above

7 The section below the Customer list should hold an add-update form. The form's title shoud be either "Add" or "Update" depending on the forms current mode. (Add for new records, Update when it is showing the selected record)
- 7.1 Refresh the app in the browser
- 7.2 With no record selected in the customer list the title of the add-update-form should be "Add"
- 7.3 Click on an item in the customer list to select it. The item should be shown in the add-update form.
- 7.4 With the selected item showing in the add-update for the form's title should be "Update"

8 The add-update form section should show the name, email and password of the record selected in the Customer list.
- 8.1 See 5.3 above.

9 Three buttons, delete, save and cancel should appear below the add-update form section
- 9.1 Refresh the app in the browser
- 9.2 Buttons should appear on the add-update form as described in the requirements.

10 Clicking the delete button when a record is selected should delete the selected record.

10.1 Refresh the app in the browser
10.2 Click on an item in the customer list to select it
10.3 Click on the [delete] button
10.3.1 The selected item's field values should no longer be shown in the add-update form
10.3.2 The selected item should no longer appear in the customer list
10.3.3 No item in the customer list should show as selected
10.3.4 Querying the REST service directly should confirm that the item has been deleted

11 Deleted records should no longer show in the Customer List
11.1 See 10.3.2 above

12 Users should be able to modify any of the data (name, email or password) in the add-update form section.
12.1 Refresh the app in the browser
12.2 Click on each of the input fields in the add-update form and try entering text
12.3 If the field was empty the input element's placeholder text should be replaced with the text being typed
12.4 If the field has data that data should be modified or appended to based on the text being typed

13 Clicking the save button when a selected record has been modified in the add-update form section should update the original record with the modified values.
13.1 Refresh the app in the browser
13.2 Select an item in the customer list, the title of the add-update section should change to 'Update'
13.3 Modify one or more fields of the seleted record's data in the add-update form
13.4 Click on the [save] button
13.4.1 The changes made in the add-update form should appear in the related item in the customer list
13.4.2 There should be no selected record after save is clicked
13.4.3 The add-update form fields should be empty except for the placeholder text
13.4.4 The title of the add-update section should revert to 'Add'

14 After a record is modified the customer list should be updated so that the modified field values are visible in the list
14.1 See 13.4.1 above

15 Clicking on the Cancel button should de-select the selected record.

15.1 Refresh the app in the browser and select a record in the customer list
15.2 Click on the [cancel] button
15.2.1 There should no longer be any selected item showing in bold
15.2.2 The add-update form fields should be empty except for the placeholder text
15.2.3 The title of the add-update section should revert to ‘Add’

16 After clicking cancel the fields in the add-update form section should be empty.
16.1 See 15.2.2 above

17 When no record is selected the user can type new data into the fields in the Add-update form section.
17.1 See 12.2 above

18 Clicking Save when no record is selected and when data has been entered into the Add-update form fields should add a new record.
18.1 Refresh the app in the browser
18.2 Check that the title of the add-update section shows as ‘Add’
18.3 Enter data into the add-update form fields
18.4 Click on the [save] button
18.4.1 The data entered in the add-update form should appear as a new item in the customer list
18.4.2 After save is clicked there should be no selected record
18.4.3 After save the add-update form fields should be empty except for the placeholder text
18.4.4 After save the title of the add-update section should be ‘Add’

19 After a new record has been added the Customer List should be updated to include the new record.
19.1 See 18.4.1 above

Suggested Order of Work

1 Create a new React project using “npx create-react-app proj-name”
2 Start the REST server by running: “npm start” in the server dir (see above.)
3 Start the React development server by running: “npm start” in the project dir. The app should automatically open in the browser as well. If it does not then go to the browser and open a tab for the app.
4 Replace the contents to the src\App.js file with a simple header like this:
<h4>Customer App</h4>

5 Create a “components” directory under the project’s \src directory.
6 Add the JavaScript files shown in bold below to the project’s \src\components directory:

```
└──src
    │   ...
    └──components
           customer-add-update.js
           customer-list.js
```

7 Assemble a static version of the application where;
The <App /> component is implemented in Src\App.js
The <CustomerList /> is implemented in src\components\customer-list.js
The <CustomerAddUpdate /> form is implemented in src\components\customer-add-update.js
HTML code from the phase01 html application can be re-used here to implement the static portion of the React app.

8 Go to the app’s tab in the browser and verify that the modified page with the “Customer App” header appears.
9 Add components with basic headers in the customer-add-update.js and customer-list.js files. Add elements to the App component’s JSX template to display the new components. Check the browser to verify that the components appear.
10 Add code to the App component to:
    10.a Retrieve customer records from the REST server using the “useEffect” hook.
    10.b Maintain the state of the retrieved records with the “useState” hook
    10.c Pass the retrieved record to the CustomerList component through the JSX template.
11 Add code to the CustomerList component in the customer-list.js file to:
    11.a Display the customer list that was passed from the App component using the JavaScript array “map” function
12 Update code in the App and CustomerList components to implement record selection:
    12.a Maintain the state of the selected record using “useState” in the App component
    12.b Pass the selected state and the “setSelection” function from App to CustomerList
    12.c Add a click handler to the table row in CustomerList that calls “setSelection”
13 Continue your implementation of items as they appear in the requirements list. Where possible implement them in the order they are listed.
14 As you work, save your files frequently and update the browser page to check your work.

15 If you are unsure how to implement a requirement you can check with the instructor, but make sure you have though about it a bit first before asking for help. The more specific your question the better the instructor will be able to help.

Things to Consider

Keep the following in mind:

- Take advantage of the React techniques you learned from the LectureBook and Labs when creating the app.
- To get/add/update and delete records you will need to use JavaScript's fetch method to make HTTP calls to the REST server. If you have any trouble take a look at the solution code.
- To verify data add, update and delete operations, use the browser to make a GET calls directly to the REST server. This way you can see how the state of the data has (or has not) changed.
- A REST client tool like Postman can be used to verify the syntax and payload(body) data required by the REST server's PUT, POST and DELETE endpoints. You can also check this out by looking through the documentation for the REST server we are using, 'json-server'. (search for it using google.)
- Visual features should match those of the application you created for phase one. Feel free to use the basic HTMl and CSS from the phase one project to get started on the React Components.

Project Phase 3 - Java Back-End

Overview

This phase of the project will concentrate on the creation of a RESTful web service which will provide data to the previously created client applications.
Complete instructions will be made available in the next release of the project instructions.

Project Phase 4 - Final Assembly

Overview

This phase of the project will concentrate on connecting the React front-end application with the RESTful web service and deploying the resulting application.
Complete instructions will be made available in the next release of the project instructions.