11 Express.js: Note Taker

Your Task

Your assignment is to modify starter code to create an application called Note Taker that can be used to write and save notes. This application will use an Express.js back end and will save and retrieve note data from a JSON file.

The application’s front end has already been created. It's your job to build the back end, connect the two, and then deploy the entire application to Heroku.

User Story

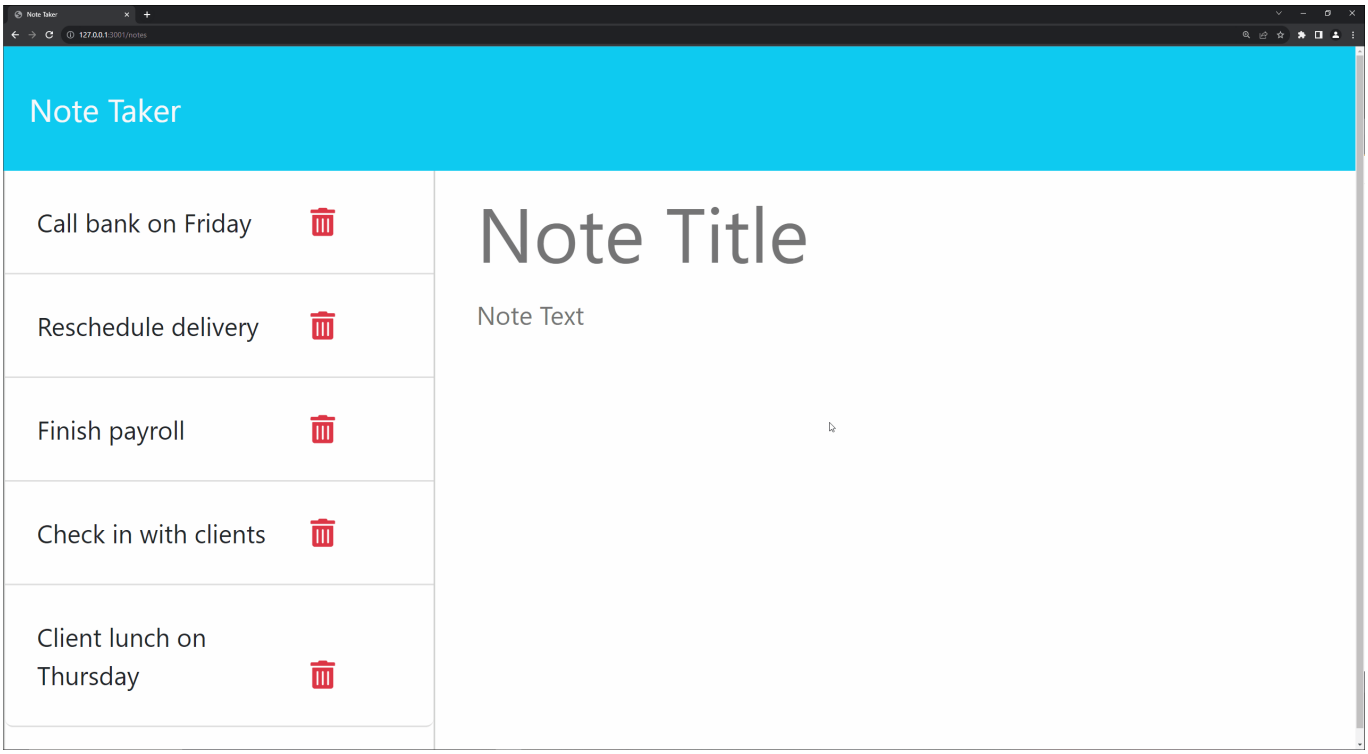
```
AS A small business owner
I WANT to be able to write and save notes
SO THAT I can organize my thoughts and keep track of tasks I need to complete
```

Acceptance Criteria

```
GIVEN a note-taking application
WHEN I open the Note Taker
THEN I am presented with a landing page with a link to a notes page
WHEN I click on the link to the notes page
THEN I am presented with a page with existing notes listed in the left-hand
column, plus empty fields to enter a new note title and the note’s text in the
right-hand column
WHEN I enter a new note title and the note’s text
THEN a "Save Note" button and a "Clear Form" button appear in the navigation at
the top of the page
WHEN I click on the Save button
THEN the new note I have entered is saved and appears in the left-hand column with
the other existing notes and the buttons in the navigation disappear
WHEN I click on an existing note in the list in the left-hand column
THEN that note appears in the right-hand column and a "New Note" button appears in
the navigation
WHEN I click on the "New Note" button in the navigation at the top of the page
THEN I am presented with empty fields to enter a new note title and the note’s
text in the right-hand column and the button disappears
```

Mock-Up

The following GIF shows the web application's appearance and functionality:



Getting Started

On the back end, the application should include a `db.json` file that will be used to store and retrieve notes using the `fs` module.

The following HTML routes should be created:

- `GET /notes` should return the `notes.html` file.
- `GET *` should return the `index.html` file.

The following API routes should be created:

- `GET /api/notes` should read the `db.json` file and return all saved notes as JSON.
- `POST /api/notes` should receive a new note to save on the request body, add it to the `db.json` file, and then return the new note to the client. You'll need to find a way to give each note a unique id when it's saved (look into npm packages that could do this for you).

Bonus

You haven't learned how to handle DELETE requests, but this application offers that functionality on the front end. As a bonus, try to add the DELETE route to the application using the following guideline:

- `DELETE /api/notes/:id` should receive a query parameter that contains the id of a note to delete. To delete a note, you'll need to read all notes from the `db.json` file, remove the note with the given `id` property, and then rewrite the notes to the `db.json` file.

Grading Requirements

> **Note**: If a Challenge assignment submission is marked as "0", it is considered incomplete and will not count towards your graduation requirements. Examples of incomplete submissions include the

> following:
>
> - A repository that has no code
> - A repository that includes a unique name but nothing else
> - A repository that includes only a README file but nothing else
> - A repository that only includes starter code

This Challenge is graded based on the following criteria:

Technical Acceptance Criteria: 40%

- Satisfies all of the preceding acceptance criteria plus the following:
    - Application front end must connect to an Express.js back end.
    - Application back end must store notes that have a unique id in a JSON file.
    - Application must be deployed to Heroku.

Deployment: 36%

- Application deployed at live URL.
- Application loads with no errors.
- Application GitHub URL submitted.
- GitHub repository contains application code.

Application Quality: 11%

- Application console is free of errors.

Repository Quality: 13%

- Repository has a unique name.
- Repository follows best practices for file structure and naming conventions.
- Repository follows best practices for class/id naming conventions, indentation, quality comments, etc.
- Repository contains multiple descriptive commit messages.
- Repository contains quality README file with description, screenshot, and link to deployed application.

Bonus: +10 Points

Fulfilling the following can add up to 10 points to your grade. Note that the highest grade you can achieve is still 100:

- Application allows users to delete notes.

Review

You are required to submit BOTH of the following for review:

- The URL of the functional, deployed application.
- The URL of the GitHub repository, with a unique name and a README describing the project.
