Assignment 2: Personal Diary

***IMPORTANT**: This assignment must be done individually.*

Read Section A to understand the programming requirements, Section B to understand the programming tasks that you need to carry out, Section C gives you some more tips and Section D to know what you need to submit as the result.

A. Description

In this assignment, you will create a personal diary app with use of NodeJS and database mongoDB. You are free to make your own choice between: (1) Server-rendering with Handlebars, (2) Single page app, (3) Universal (Combination of 1 & 2), or even with React.

Diary behavior

The Diary app is a web site that provides a way to make a simple, **login-less** journal. Every new journal has a unique URL that is viewable and editable by anyone with the link.

Home Screen: Browsing journals

- When you go to the home page of your Diary app, you can browse all the journals **by date**

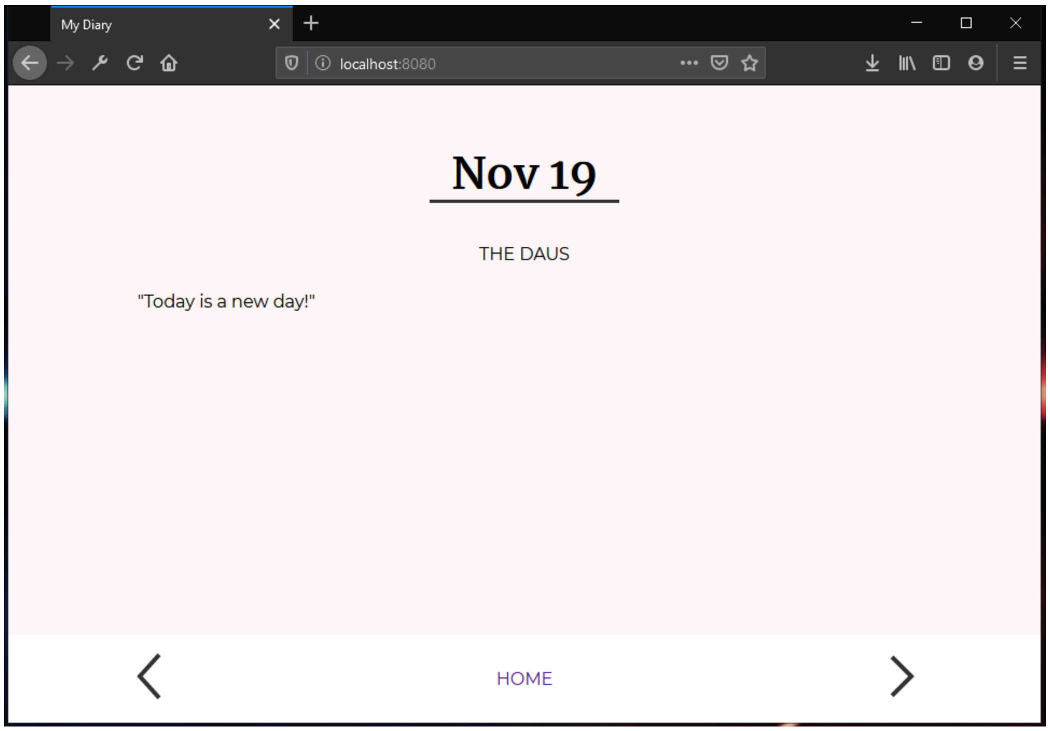


Figure 1: day with journal created

- The today's journal is shown first. If no journal for today was created, you are presented with a button to create a new journal. Similar to other days.
- Today's date is shown at the top of the screen, as is a journaling prompt for that day.
    - **Hint:** You will want to use the Date class and `toLocaleDateString`. See *Section C. Tips* for help and example usage.

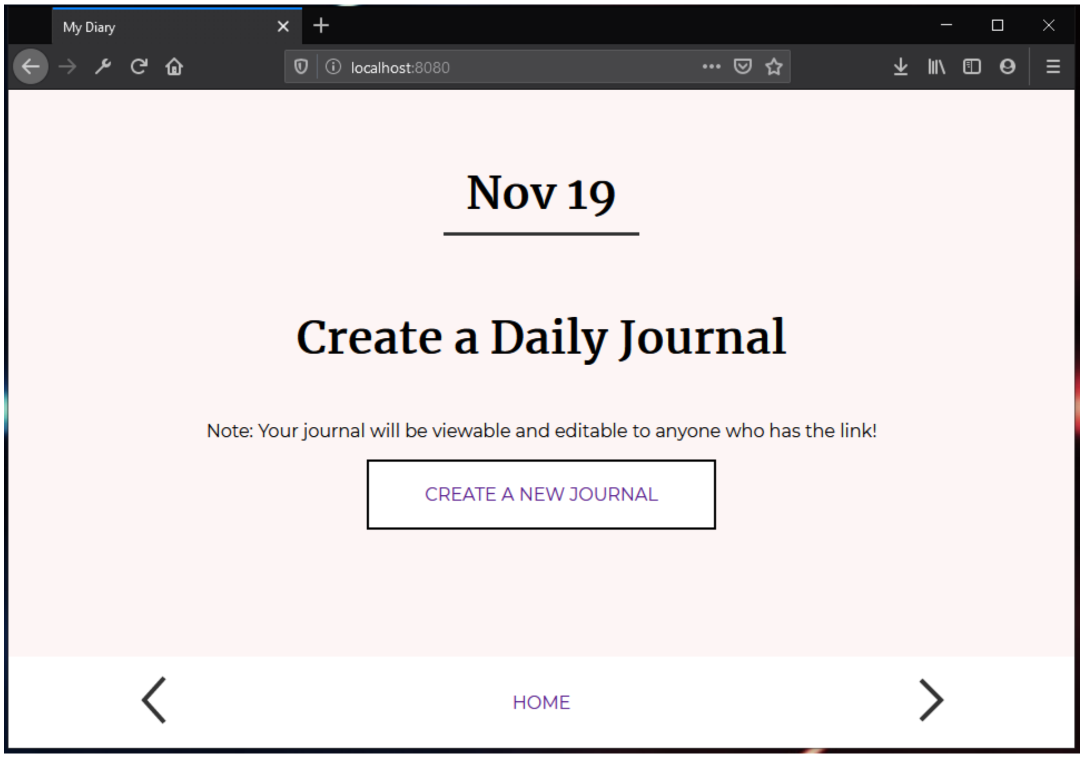

My Diary
localhost:8080
Nov 19
Create a Daily Journal
Note: Your journal will be viewable and editable to anyone who has the link!
CREATE A NEW JOURNAL
HOME

Figure 2: day with NO journal created

- Navigating between journals:
    - At the bottom of the Home Screen, there is a left and a right arrow. Click either arrow to go to the previous or the next day.
    - All days that you navigate to are editable (double click on title or content).
    - Clicking "Home" takes you back to the journal for today's date.

VIDEO: http://fit.hanu.vn/pluginfile.php/9163/mod_folder/content/0/demo-videos/browsing-journals.mp4

Editor Screen: Creating a new journal

- While browsing each day on the Home Screen, if no journal created for that day, you are presented with a button to create a new journal.
- When you press "Create a new journal":

- A form should be displayed to user for creating new journal, containing *title* & *content*.
- Your app should create a new journal on the backend.
- The browser should navigate back to Home Screen to see that created journal `/?date=<date>`, where **date** is the date this journal is created for, in format of Year-Month-Day.

  Hint: You can set `window.location.href = <URL>` to make the browser navigate to a URL.

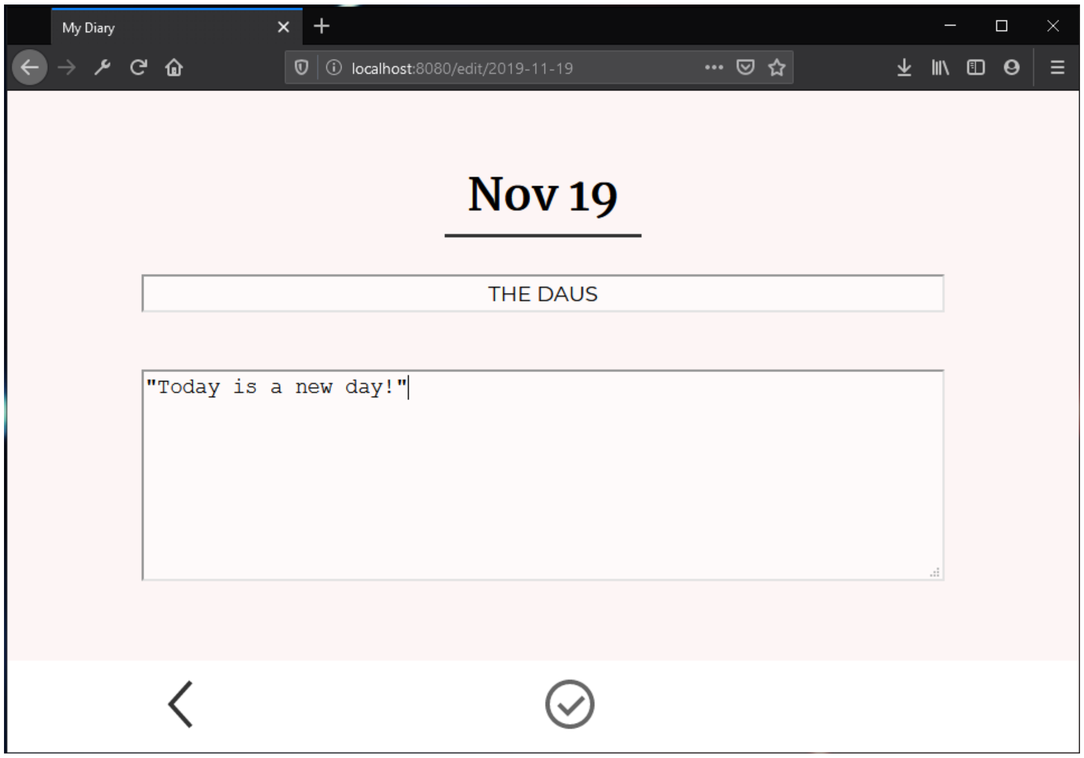


Figure 3: creating a new journal

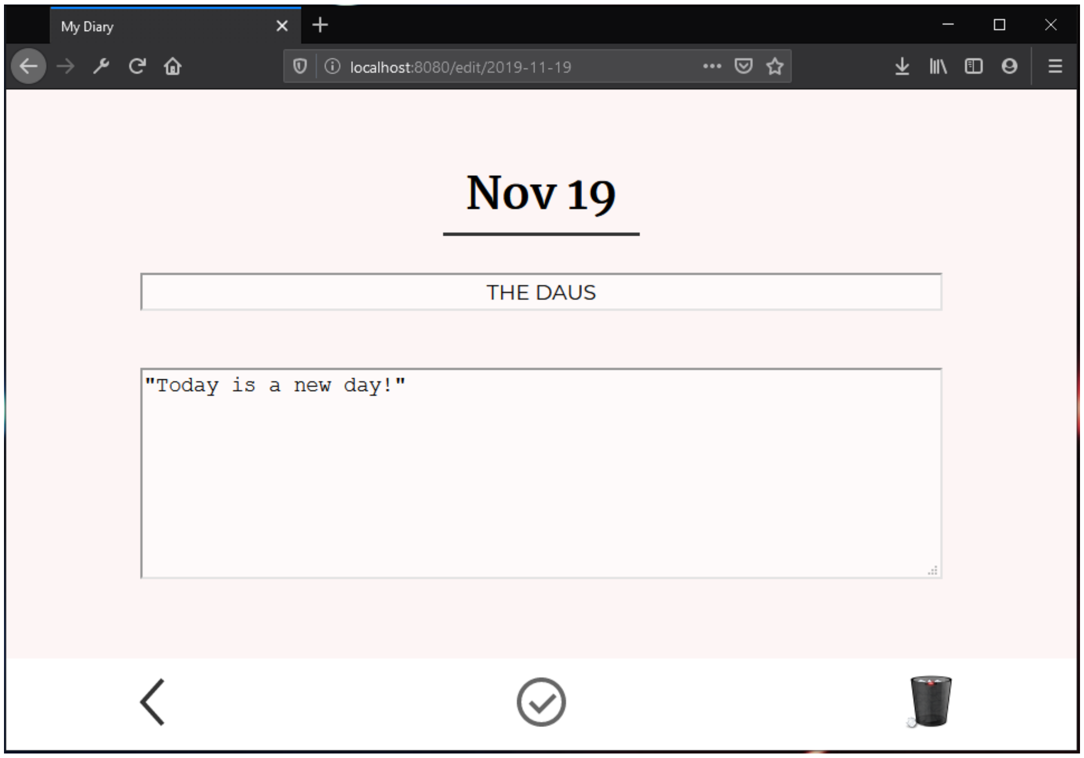
- Clicking the "Left arrow" to navigate back to the Home Screen of the same day.

VIDEO: http://fit.hanu.vn/pluginfile.php/9163/mod_folder/content/0/demo-videos/creating-journal.mp4

Editor Screen: Updating a journal entry

- When you double-click on the title or content of the journal, you should be able to edit the diary entry.
- The entry should be saved when you click the check box. This should also navigate user back to the Home Screen to see updated content.
- Clicking the "Left arrow" to navigate back to the Home Screen of the same day.

VIDEO: http://fit.hanu.vn/pluginfile.php/9163/mod_folder/content/0/demo-videos/editing-journal.mp4



Nov 19

THE DAUS

"Today is a new day!"

Figure 4: updating a journal of specified day

Editor Screen: Deleting a journal entry

- While you are editing the entry, you have another option is to delete that entry.
- Clicking the "Trash bin" icon to delete the journal
- The app prompts to ask you to confirm before deleting
- Your app should delete that journal on the backend
- Finally redirecting user back to the Home Screen of the same day.

VIDEO: http://fit.hanu.vn/pluginfile.php/9163/mod_folder/content/0/demo-videos/deleting-jounal.mp4

Persistent data

- The diary data should persist between reloads and server restarts
    - i.e. data should be stored in and loaded from a MongoDB backend.

    Hint: The *Section C. Tips* has some suggestions on how to structure your MongoDB database.

Diary style and assets

We are not going to spec out the diary app as specifically as previous assignments, since we are not grading you on how closely you followed our specification.

Text properties:

- Large heading: Date and "Create a Daily Journal" text:
    - Font face: Merriweather from Google Fonts
    - Size: 30pt
- Small text: Subtitle, Button text
    - Font face: Montserrat from Google Fonts
    - Size: 12pt
- Diary entry text
    - Font face: Merriweather from Google Fonts
    - Size: 14pt
    - Line height: 1.5em

"Create a new journal" button

- Border is 2px wide and black in color
- 20px of top-and-bottom space between the text and the border; 50px of left-and-right space between the text and the border
- Hovering over the button changes the button's font color to white and the background color of the button to black. The color transitions from white to black in 0.5 seconds.

Note: a link can be styled to look like a button, and vice versa.

Background colors

- Page background is #fdf5f5
- Input, Textarea background color is white, with 50% opacity
- Footer background is white

Assets (provided on fit.hanu.vn)

- Images
    - back.png
    - forward.png
    - checked.png

- bin.png

B. Task requirements

Again, you are **free to make your own choice** between: (1) Server-rendering with Handlebars, (2) Single page app, (3) Universal (Combination of 1 & 2), or even with React - **and to design your app**.

1. **Structure your web application**

   Note: `package.json` must be included.

2. **Frontend (HTML/CSS/JS)**
   - Styling your app's frontend as stated in *Diary style and assets (Section A)*
   - Create as many pages as you want, to serve user functions as mentioned in *Diary behavior (Section A)*

   An example is as below:

2.1. Home Screen

- `date` is optional, if not specified, `date` = today http://localhost:8080/?date=2019-11-20
- Select journal from database for specified `date`.
- If journal exists (created), show journal (see *Figure 1*)

  Or else show button "Create a Daily Journal" (see *Figure 2*)

2.2. Editor Screen

- `date` is required http://localhost:8080/edit/2019-11-20
- Select journal from database for specified `date`
- If journal exists (created), show edit form with journal info - update (see *Figure 4*)

  Or else show empty edit form – create (see *Figure 3*)

3. **Backend (NodeJS)**
   - Create server view and/or api routes to manipulate with database to serve functions as mentioned in *Diary behavior (Section A)*

   An example is as below:

3.1. Save (create/ update) journal for specified date

- `date` is required http://localhost:8080/store/2019-11-16
- Select journal from database for specified `date`.
- If journal exists → update journal with new user inputs

  Or else → create a new journal with user inputs

3.2. Delete journal

- `date` is required http://localhost:8080/delete/2019-11-16
- Delete journal form database for specified `date.`

4. Frontend – consume – Backend APIs (JS – fetch)

Update frontend to consume backend APIs created in task (3).

Again, above gives an example, you are **free to make your own choice** between: (1) Server-rendering with Handlebars, (2) Single page app, (3) Universal (Combination of 1 & 2), or even with React - **and to design your app**.

5. Weekly plan

You have to schedule yourself a weekly plan to complete your tasks, an example is given below. (for server-rendering app with APIs)

Week	Frontend	Backend	Frontend – consume – Backend APIs
1	- *Home Screen*: Browsing journals by date + HTML + CSS + JS – handle double click event to open *Editor Screen*	- Server-rendering *Home Screen* http://localhost:8080/?date=2019-11-20	
2	- *Editor Screen*: Create/ Update a journal by date + HTML + CSS	- Server-rendering *Editor Screen* http://localhost:8080/edit/2019-11-20 - Server APIs – save journal http://localhost:8080/store/2019-11-16	- JS – handle form submits to APIs then redirect user
3	- Editor Screen: Delete a journal by date + update HTML	- update Server-rendering with button Delete http://localhost:8080/delete/2019-11-16	- JS – confirm user delete → calling APIs then redirect user
	- test your application (browsing journals in 2 cases – date with & without journal, creating a new journal, updating a journal & deleting a journal)		

C. Tips

Here are some specific tips for coding the Diary app:

Date class

- You will need to use JavaScript's `Date` class to get today's date as well as other dates.
- `Date` examples: Please check this out! This shows printing a date, advancing a date, stringifying a date
- Use `toLocaleDateString` to create a string date like "June 2" (mdn)

Data Model

- You can think of your Diary app as having only one main "entities":
    - **Journals:** A diary journal contains the title, text content of the entry, and a date (Year-Month-Day, not time) belonging to the journal.
- To model this entity, consider the following structure:
    - Create an "journals" MongoDB collection, which contains one document per entry
        - Each entry has a "date" field, but instead of storing it as an actual Date type, we recommend you store the date as a string like "`2019-11-19`". See example for how to generate a string like that from a Date object
        - Each entry also contains the "title" & "content" of the entry
    - You don't have to model your data this way, but it should make querying your data simpler.

D. Submission

You must submit a single zip file containing the application to the portal by the due date. The zip file name must be of the form a2_Sid.zip, where *Sid* is your student identifier (the remaining bits of the file name must not be changed!). For example, if your student id is 1701040001 then your zip file must be named a2_1701040001.zip.

Note: Do not include `node_modules` folder into your submission

IMPORTANT: failure to name the file as shown will result in no marks being given!

NO PLAGIARISM: if plagiarism is detected, 0 mark will be given!