Description

The EDETU Student app, was created with the intention of helping students find a better way of increasing their productivity in their studies, by creating a place, where of their studies is combined into one. Being a student myself, I understand the stress of schooling, which was what inspired me to create an application that will help students better control their studies. EDETU aims to provide students with an “all in one service”, that is their one stop for their studies, giving the students power to create notes and create a calendar of when events/homework is due.

The system was built using HTML, CSS and VUE framework, using NodeJS. This allowed the file structure, allowing the ability of creating different components into different files. This allowed for scalability and the ability to add or remove components when needed without effecting the greater program.

Deviations from proposal

the following features were either removed/pushed to the next phase:

- **Calendar** – I initially thought the implementation of a calendar would have easy, but I discovered too late into the creation of my project. Due to the calendar not being implemented, this obviously didn’t allow me to create the event handling system, as it was based around the calendar. This had to be pushed back to the next phase
- **login/registration –** Although creating a login and registration page with working elements, a fully functioning login/registration system would require a backend, which was out of the scope of this task.
- **edit notes –** the feature where students can review and edit notes must be pushed back to the next phase, due to unforeseen circumstances and problems with the creation of the project.

Single file components

Since I had created lots of my project in the earlier weeks of the start of the assignment, it would have been very difficult to convert into single file components. This would have been preferred, as it makes the overall program much more scalable. Single file components will be introduced and made as standard in the next implementations of EDETU student.

Coding concepts

Event handling

The use of event handling was used on all my buttons throughout the project. I used event handling on the buttons, so once the button is pressed, it will store locally everything the user has input into any of the fields. This would allow for backend integration, where the event handling will have greater purpose, as the information is only being stored locally.

HTML

```
<div class= container >
  <p>
  <label for="title"><b>title</b></label>
  <!--this line usses v-model, to store locally the input of the user-->
  <input v-model="title" type="text" placeholder="title" required>

  <label for="note">Write your notes here:</label>
  <!--this line usses v-model, to store locally the content of what is written in the note-->
  <br><textarea v-model="note" id="note" name="note"rows="20" cols="165"></textarea>
  </p>
  <p>
    <!--onclick to save locally (can be show in console)-->
    <button @click="persist">Save</button>
  </p>

</div>
```

VUE

```
methods: {
  persist() { //function calls once button is pressed
    localStorage.uname = this.uname; //saves username locally
    localStorage.psw = this.psw; ////saves password locally
    console.log('user: ' + this.uname + ' saved'); //logs to t
  }                                                //prints t
}
```

Form input bindings

Due to the use of the v-model, when a user enters a value, it is assigned to the data object, due to two-way binding. Form input bindings were used in text, where a user types a response into a box and for a select, where the user will be told to select from two the different options of remembering or don't remember when the user logs in next.

HTML

```
<!--creates textbox for a users response of each category (first name, last name, username and password)-->
<label for="First_name"><b>First name</b></label>
<input type="text" placeholder="First name" name="fname" required>

<label for="Last_name"><b>Last name</b></label>
<input type="text" placeholder="Last name" name="Lname" required>

<label for="uname"><b>Username</b></label>
<input type="text" placeholder="Username" name="uname" required>

<label for="psw"><b>Password</b></label>
<input type="password" placeholder="E Password" name="psw" required>

<button type="submit">Register<a href="index.html"></a></button>
```

Text/multiple line text

```
<p>{{ message }}</p> <!--displays the message the user has chosen (save or dont save password)-->
<select v-model="selected">
    <option disabled value="">Please select one</option>
    <option>Remember me</option> <!--option 1, which will save password for when they log in next-->
    <option>Don't Remember me</option> <!--option 2, which wont save password for when they log in next-->
  </select>
  <span>Selected: {{ selected }}</span> <!--dispalys users optin in plain text-->
```

Selector (dropdown menu)

VUE

```
var app = new Vue({
  el: '#app',
  data: {
    message: 'Stay Logged in?',
    selected: '',
    uname: '',
    psw: '',
  },
```

Client-side storage

Client-side storage is used to store data on the browser itself. In theory with the implementation of a backend system, this would greatly increase loading times as it doesn't have to fetch data from a server every time the user logs in. I've implemented client-side storage with my notepad, login and registration, so that in a future phase when backend is implemented, it will allow for quick fetching of information

HTML

```
<p>
<label for="title"><b>title</b></label>
<!--this line usses v-model, to store locally the input of the user-->
<input v-model="title" type="text" placeholder="title" required>

<label for="note">Write your notes here:</label>
<!--this line usses v-model, to store locally the content of what is written in the note-->
<br><textarea v-model="note" id="note" name="note"rows="20" cols="165"></textarea>
</p>
```

Vue

```
const app = new Vue({
  el: '#app',
  data: {
    title: '',
    note: '',
  },
  mounted() {
    if (localStorage.title) {
      this.title = localStorage.title;
    }
    if (localStorage.note) {
      this.note = localStorage.note;
    }
  },
  methods: {
    persist() {
      localStorage.title = this.title;
      localStorage.note = this.note;
      console.log('Title: ' + this.title + ' saved');
    }
  }
})
```

Output

```
2 Title: hello saved                    notes.js:19
  Title: new note saved                 notes.js:19
  Title: second note saved              notes.js:19
>
```