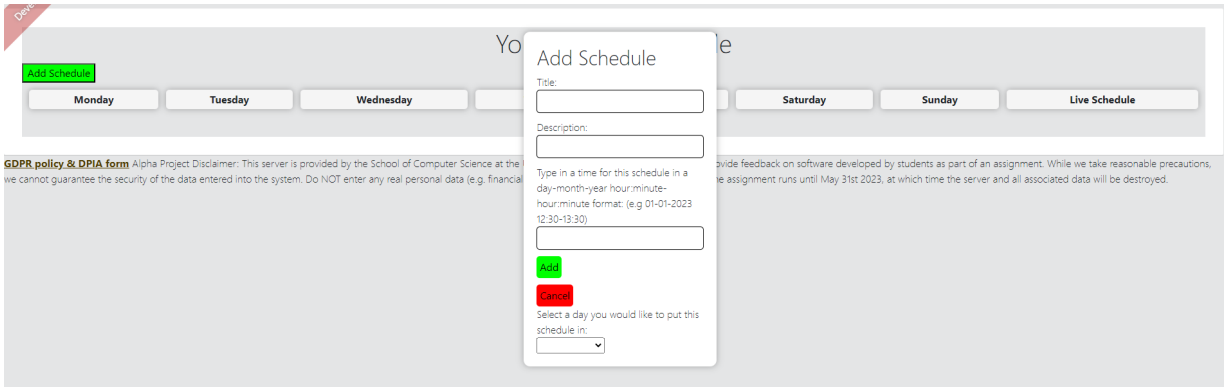
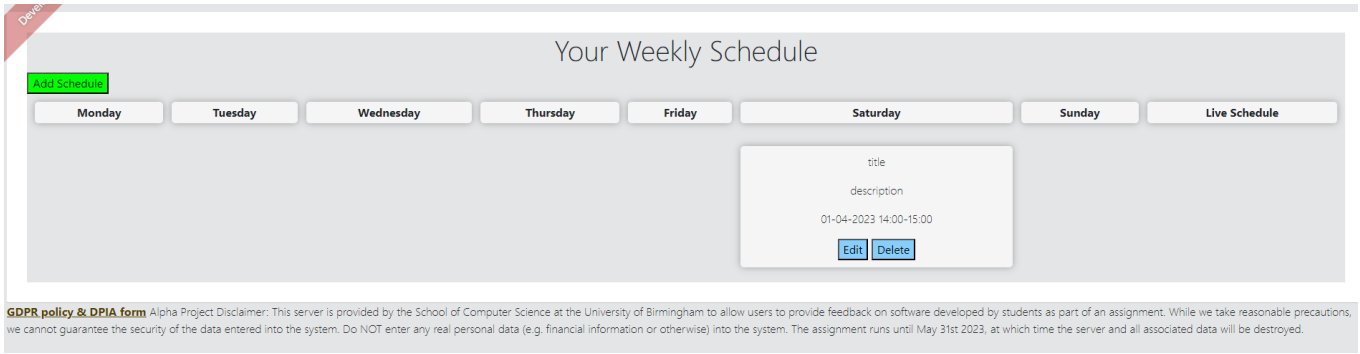
Screenshots of vertically sliced features:



When you want to add a schedule ^



When a schedule is added ^

```
1 usage
add() {
  const newschedule = {
    title: this.newschedule.title,
    description: this.newschedule.description,
    time: this.newschedule.time
  };
  const selectedDate = moment(this.newschedule.time, format: 'DD-MM-YYYY').toDate();
  const selectedDay = selectedDate.toLocaleDateString( locales: 'en-GB', options: {weekday: 'long'});
  console.log("selectedDate: ", selectedDate);
  console.log("selectedDay: ", selectedDay);
  if (selectedDay !== this.day) {
    alert('The date you added does not match that day of the week.');
    console.log(this.day)
    return;
  }
  this.Arrayforschedules[this.day].push(newschedule);
  this.newschedule.title = "";
  this.newschedule.description = "";
  this.newschedule.time = "";
  this.day = "";
  this.close();
  const td = document.getElementById(this.day);
  const div = document.createElement( tagName: "div");
  div.innerText = `${newschedule.time} ${newschedule.title} ${newschedule.description}`;
  if (td) {
    td.appendChild(div);
  }
}

1 usage
delete(day: string, index: number) {
  if (confirm('Do you want to delete this schedule?')) {
    this.Arrayforschedules[day].splice(index, deleteCount: 1);
  }
}
```

```
1 usage
edit(day: string, index: number) {
  const schedule = this.Arrayforschedules[day][index];
  const updatedTitle = prompt( message: 'Enter new title:', schedule.title);
  const updatedDescription = prompt( message: 'Enter new description:', schedule.description);
  if (updatedTitle) {
    this.Arrayforschedules[day][index].title = updatedTitle;
  }
  if(updatedDescription){
    this.Arrayforschedules[day][index].description = updatedDescription;
  }
}
```

Backend for adding, deleting and editing schedules.

Description of the development and integration with the app:

What I have developed is a scheduler feature which allows the user to make a weekly schedule for themselves that will help them to keep track of what they want to do and when exactly. At first, I mainly focused on getting the front-end interface, with just the days of the week and a table to be displayed. Once I had that, I needed to then add schedules to the table under the specified day of the week. That was when I created an “Add Schedule” button and worked on its functionality. A schedule would require the user to input a title, description and a time for the schedule, as well as choosing the day to put the schedule under, so a pop-up allows the user to do that and the schedule gets pushed to the array of schedules. However, as the user also has to input a date and the specific day to put the schedule under, I realised that it wouldn’t make sense to be able to put a schedule under “Monday” if the specified date isn’t actually on a Monday, so I made sure that it first checks if the date matches the day, and if it doesn’t, the user is alerted with the message “The date you entered does not match that day of the week”, and only when it matches can the user add the schedule to the table.

I also wanted to make sure each schedule had a “Edit” and “Delete” button so that the user could update their schedule if they wanted to, or delete it if they no longer needed it. When the user clicks these buttons, they are prompted to either edit or delete the schedule, and if they change their mind, they can just hit “Cancel”.

The live schedule is a feature that takes a schedule from the table and displays it if the date and time of that schedule matches the current date and time in real life. Seeing as the user inputs a date and time when they add a schedule, I made it so that the table is regularly checked in intervals for schedules that match the current date and time, and if such schedules are found, they are displayed in the live schedule. However, once the time for a specific schedule ends, that schedule should no longer be displayed in the live schedule, so I also made it so that the live schedule itself is regularly checked in intervals for schedules that no longer match the current date and time, so for example, if it’s currently 15:00 (3pm) and a schedule is set to go from 14:00-15:00, once the time passes 15:00, that schedule gets removed from the live schedule as it is now over. There are cases for when a schedule is currently in the live schedule, and the user makes changes to that schedule in the table, i.e they either edit it or delete it, so I had to make sure that the live schedule also displayed those changes, either by removing a schedule that was deleted by the user, or by displaying the new updated schedule if the user made an edit.

It was after all of this that I then started working on the visuals a little bit, changing colours, and changing the way schedules appeared in the table to make it a little more appealing. This feature generally synergises well with the To-Do List feature as the user may have certain things they want to complete during their schedules, and the To-Do List can help them keep track of whether they completed it or not, and if not, they can set up another schedule in the scheduler to complete it.

Link : https://git.cs.bham.ac.uk/team-projects-2022-23/team23-22/-/commits/main