Schedule Creator

Team: 27

Jeet Shah - 2022201009
Mayank Gupta - 2022201012
Amitabh Paliwal - 2022201030
Gaurav Khapekar - 2022201055

Description :

The schedule generator application generates a schedule based on user input. The user only has to provide details about the event such as its name, description, number of days, number of events of a particular type and their duration, number of breaks, and their duration. Upon providing such details the user receives a downloadable schedule. The application has a variety of use cases as it can be used wherever a schedule is required.

Requirement :

A programmatically generated schedule based on user input.

Technologies used :

Frontend: Used HTML, CSS, vanilla JS, and jquery for the frontend part. The application is a single page and javascript along with CSS tags is used to manipulate what the user sees and interacts with on screen.

Backend: NodeJS with express used for the backend part. Bodyparser is used as middleware. The algorithm for generating the schedule is written in vanilla Js and is exported as a module that is used by the server.

Application overview :

1. In the first screen, the user provides the overall event name, description, and the number of days for which the event has to be scheduled.

2. Based on the number of days selected by the user they will be shown pages that ask for details about the events to be scheduled.

3. The user has to provide the date, event start time, and event end time for the day. Then the user has to provide details of particular events: event name, event description, event

venue, event duration, and the number of events of that particular type. Then the user can add the event using the add event button. This event will now appear in the added events section of the page where the user has the option to delete it if necessary.

4. The user can also add breaks if desired. We provide default options like breakfast, lunch, and dinner. The user can add additional breaks using the additional breaks option. It prompts the user for the number of breaks and the duration of each break.

5. When the user presses the submit button the user is shown the schedule screen and input data is sent to the server as a JSON file if there are no remaining days. But if there exist remaining days the button contains the text ‘next’ and clicking on it leads to another screen where the user is prompted for details of another schedule for the next day.

6. The final screen contains the schedule with the event name as the heading. The output is displayed as a card for each day which contains a table with the day’s scheduling and the day is indicated as a heading on each card with the date, start time, and end time for the day. The description contains a button called details for each entry which opens a popup containing the event's details. Users can regenerate the schedule using the regenerate schedule button provided.

Algorithm used for schedule creation :

Input to the algorithm :
JSON Object containing the main-event details with a list of JSON objects.
Each JSON object will contain the list of events and breaks details for each day of the schedule.

Steps:

1. Parse the input JSON.

2. Store the data of the list of events and breaks into a local data structure.

3. Each day will be divided into slots according to the breakfast, lunch, and dinner selected by the user.
   Eg. If lunch and dinner are selected then the day will be divided into three slots, if only lunch is selected then the day will be divided into two slots, and so on.

4. Now, two Max-heaps are created and initialized one for breaks and the other for events.

5. We extract the event one by one from the event heap and store them into slots equally. It means if the first value is stored in the first slot then the second value will be stored in the second slot and so on.

6. Whenever the number of events in any slot is equal to the break factor it will take a break by extracting the break from the break heap.

7. Here, we are using max-heap and distributing the events equally in each slot one by one. Since initially all the slots will have their maximum possible capacity, so the timing of lunch and dinner is not shifted much.

8. Whenever we are giving one event to put in any slot and if it is not having that much time remaining then first it will check the other two slots if any of them can accommodate this event, if yes then we will keep that event in that slot, if not then we will do merging of the slots and keep the event in the slot having highest remaining time from three slots.

9. After completing the assignment of all the events to slots formed, we check for the remaining breaks. If some breaks are still remaining to add then we distribute these breaks equally in all the slots. Thus one by one each slot will take a break.

10. Now, once each slot is having their events and breaks list, we will shuffle them randomly.

11. Make a list of the events and breaks again from the slot list and shuffle them individually.
    After shuffling again calculate the break factor and put them back into the slots.

12. Using shuffling and randomization for the same input may generate a new schedule.

13. The algorithm will return the JSON response, containing the list of day-wise schedules generated.

Use Case :

1. **Single-Day Event**:
   The user provides the event name, event description, and number of days (in this case 1)

The user provides details about the day: date, start time, and end time. Then user provides details about each event type that is to be scheduled. The user provides the event name, description, venue, duration, and the number of events. This is followed by the details of breaks if required. User can delete any added events.

Clicking on the submit button in this case sends data to the backend. The final schedule screen is then displayed where the user can download the schedule, regenerate it, and view details for any event.

2. **Multiple-Day Event**:
   The user provides the event name, event description, and number of days (in this case > 1)

   The user provides details about the first day: date, start time, and end time. Then user provides details about each event type that is to be scheduled. The user provides the event name, description, venue, duration, and the number of events. This is followed by the details of breaks if required. User can delete any added events.

   Clicking on the next button in this case results in a new screen prompting user for input similar to the last screen. This will contain data for the second day. Next continues to provide a new screen until there are days left to be scheduled.

   The screen for the final day has the submit button instead of the next button and clicking on it sends data to the backend. The final schedule screen is then displayed where the user can download the schedule, regenerate it, and view details for any event.