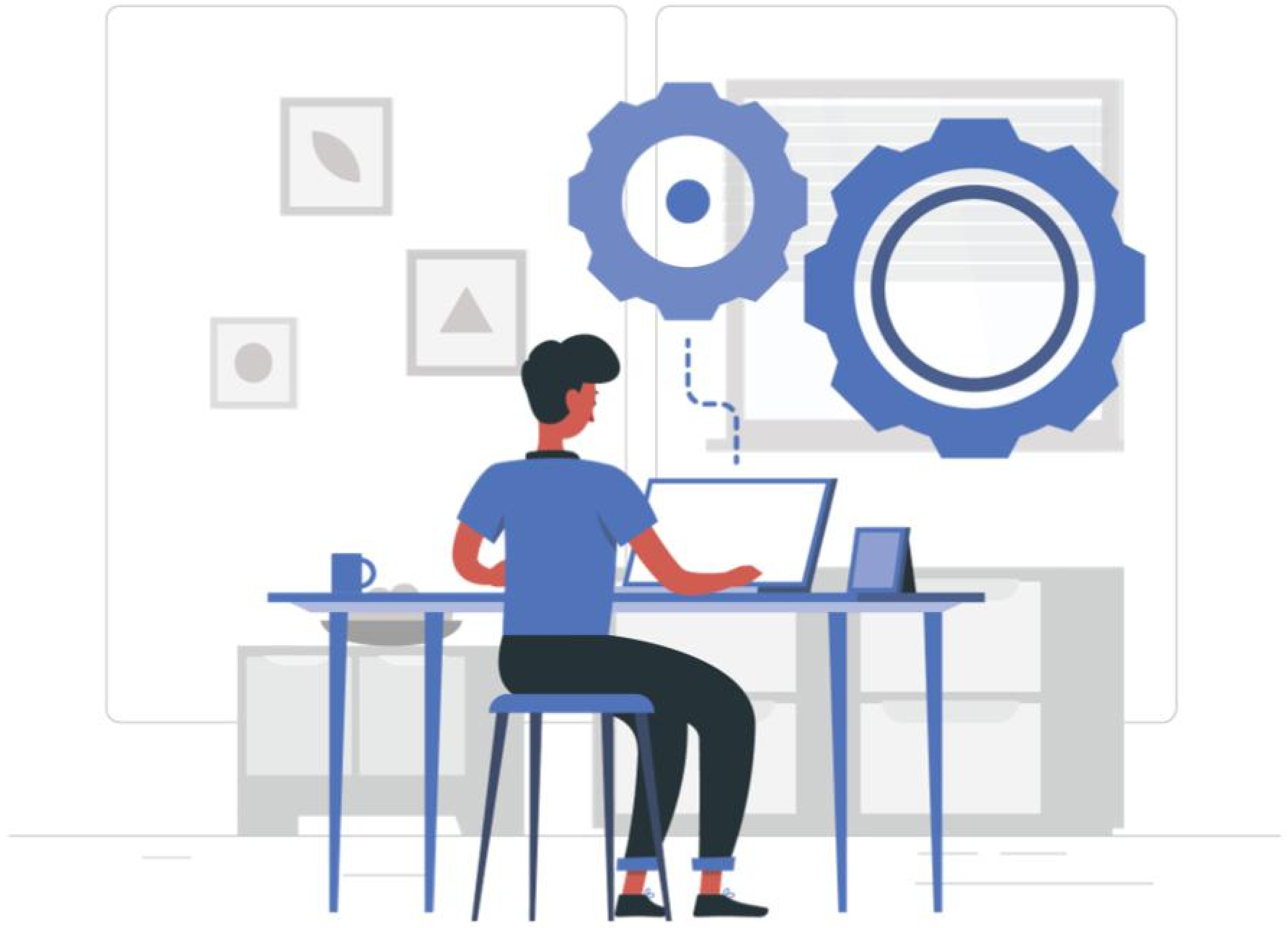


Practice

Develop Interactive Reactive Forms Inside SPA

Practice

- Practice: Develop an event schedule form using Angular reactive forms

Points to Remember

- Only Angular Material components should be used for creating the ViewTube live stream schedule event form.
- Import all the modules required to work with Angular form and material components in `app.module.ts`
- A custom validator function for validating guest emails should be defined inside the `studioLiveStream` component class.
- Text for validation error messages should be given as stated in the expected output. Failing to do so will result in failure of the test.
- Run the `json-server` to add the event data to the `events.json` file in the **`data`** folder.

Instructions for Practice

- Click here for the boilerplate.
- Please read the README.md file provided in the boilerplate for further instructions about the practice exercise.
- Fork the boilerplate into your own workspace.
- Clone the boilerplate into your local system.
- Open command terminal and set the path to the folder containing the cloned boilerplate code.
- Run the command npm install to install the dependencies.
- Open the folder(**fe-c5-s3-reactive-forms-practice**) containing the boilerplate code in VS Code.
- Create a reactive form using Angular Material components and generate the output as shown in the image files present in the boilerplate.

Notes:

1. The solution to this practice will undergo an automated evaluation on the `CodeReview` platform. (Local testing is recommended prior to `CodeReview` testing)
2. The test cases are available in the boilerplate.

Context

The application `ViewTube` is a single-page application that allows users to view a list of videos. These videos can be created and uploaded by anyone after they are approved by the organisation.

The marketing team wants to expand their market by attracting more users to their `ViewTube` application. For this, they planned to schedule live events over the OTT (Over-The-Top) platform called ViewTube Live.

`ViewTube` Live is an easy way for creators of videos to reach their respective communities in real-time. Whether streaming an event, teaching a class, or hosting a workshop, ViewTube has tools that will help manage live streams and interact with the viewers in real-time.

As a front-end developer, you have been given the responsibility of creating a reactive form to schedule live events that is cleaner, easier to read, and more maintainable.

PRACTICE

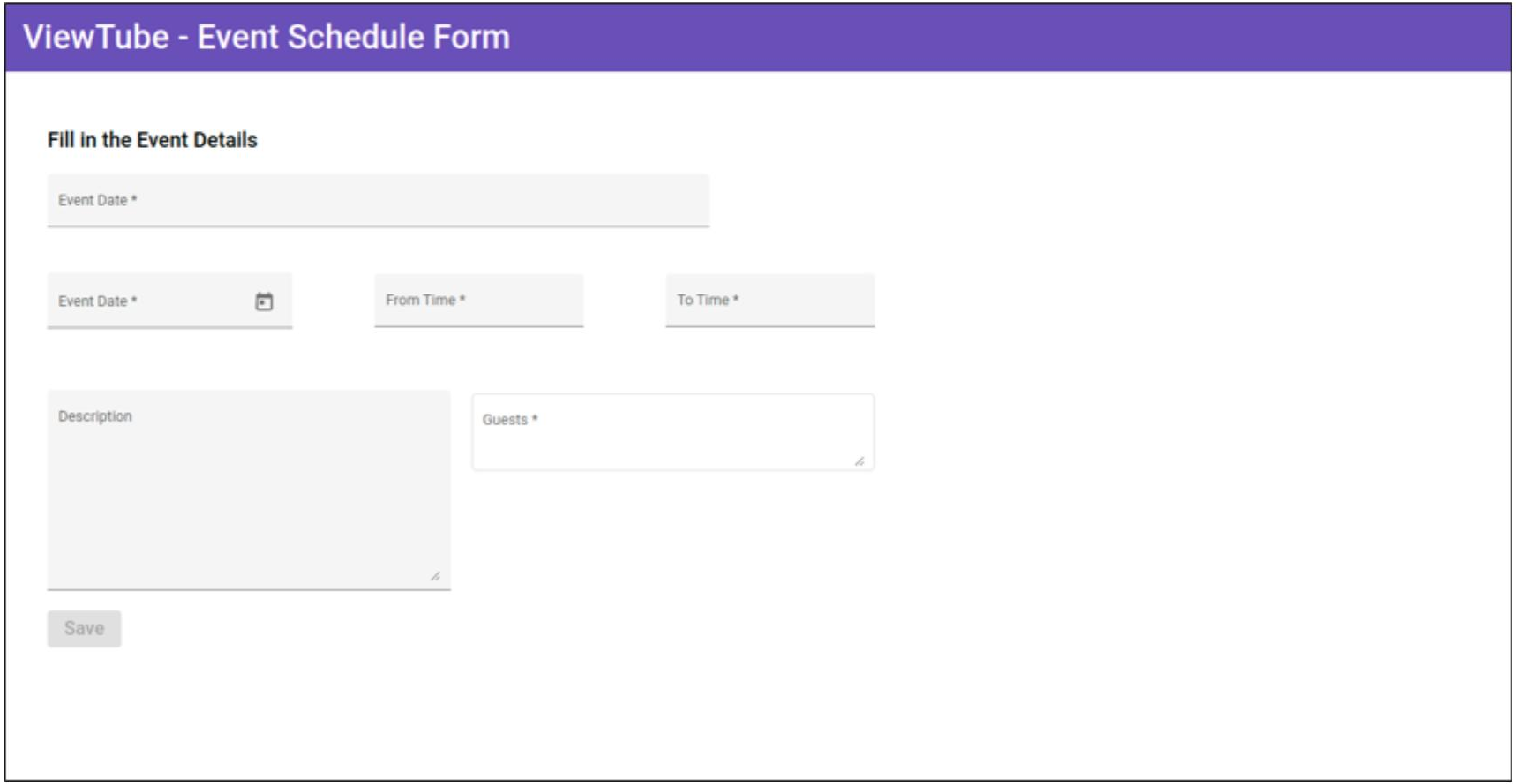
Develop an Event Schedule Form Using Angular Reactive Form

Create a reactive form using Angular Material components to schedule an event with various details: event title, event start date and time, event end date and time, event description, and guest list.

Add the specified validators for the fields to make the form more robust.

Note: The tasks to create the reactive form are provided in the upcoming slides.

Expected Output: Live Stream Event Form



ViewTube - Event Schedule Form

Fill in the Event Details

Event Date *

Event Date *

From Time *

To Time *

Description

Guests *

Task Details

The following tasks need to be completed to develop the solution for the ViewTube application:

- Task 1: Include required modules
- Task 2: Define the `StudioLivestreamComponent`
- Task 3: Add validators to the form controls
- Task 4: Create the HTML form in the template
- Task 5: Handle form validation
- Task 6: Display a notification message upon successful form submission

Note: Task details for the practice are provided in the upcoming slides.

Task 1: Include Required Modules

- Add the `ReaciveFormsModule` in the application root module to enable the reactive forms feature.
- Add the following modules in the application root module to create forms styled with Angular Material components.
  - `MatDatePickerModule`
  - `MatNativeDateModule`
  - `MatInputModule`
  - `MatAutocompleteModule`
  - `MatChipsModule`
  - `MatSnackBarModule`
  - `MatButtonModule`
  - `MatToolBarModule`

Task 2: Define the StudioLivestreamComponent

- Create the LiveStream component inside the ViewTube Angular application.

```
ng generate component studio-livestream
```

- The command creates an Angular component with the name `StudioLivestreamComponent` and updates the import statements in the ``app.module.ts`` file.
- The following are the steps to be implemented inside the `StudioLivestreamComponent`.
  - Step 1: Define the constructor to create a `FormBuilder` and `MatSnackBar` instance.

```
constructor (private fb: FormBuilder, private _snackBar: MatSnackBar){ }
```

Note: The component name should be kept as "`StudioLiveStreamComponent`" as it is used in testing.

Task 2: Define the StudioLivestreamComponent (Cont'd.)

- Step 2: Create a top-level form group instance called **`liveStreamForm`** using the form builder service.
  - The form model should have 7 form controls, each representing the properties `eventTitle`, `fromDate`, `fromTime`, `toDate`, `toTime`, `description`, and `guests` (as given in the code below).

```
liveStreamForm = this.fb.group({
  eventTitle: [''],
  fromDate: [''],
  fromTime: [''],
  toTime: [''],
  description: [''],
  guests: ['']
})
```

Task 2: Define the StudioLivestreamComponent (Cont'd.)

- Step 3: Do the following in the `StudioLivestreamComponent` (as given in the code below).
  - Define a `minDate` property, which is initialized to today's date. This date should be used as the start date for live stream form's `fromDate` and `toDate` properties.
  - Add a string array for the time value to be used for `fromTime` and `toTime` with the time interval of 15 minutes starting from "00:00" till "23:45" as mentioned in the code below.
  - Declare an empty string array called `guestList` to store the individual guest email IDs.

```
minDate: Date = new Date();
timeValues: string[] = [
"00:00", "00:15", "00:30","00:45","01:00", "01.15","01.30",
// Add till the last value "23:45"
]
guestList?: string[] = [];
```

Task 2: Define the StudioLivestreamComponent (Cont'd.)

- Step 4: Define getters for each of the `liveStreamForm` properties to access them in the template. The code below is shown for `eventTitle` property.

```
get eventTitle(){ return this.liveStream.get('eventTitle');}
```

- Step 5: Implement the `ngOnInit` method to split the email IDs from a list of comma-separated emails entered by the user and store them in the `guestList` array.

```
ngOnInit(): void {
  this.liveStream.controls['guests'].valueChanges.subscribe(
(guestEmails) => { this.guestList = guestEmails?.split(',');
});
}
```

Task 3: Add Validators to the Form Controls

- The following are the form controls with their validation criteria.

Form Control	Validation	Error Messages
Event Title	Should not be blank Should have maximum length of 100 characters	- "Event tile is required" - "Event title should not exceed 100 characters"
Event Date	Should not be blank Should not be less than today's date	- Event schedule date is required
From Time	Should not be blank	- Event schedule From Time is required
To Time	Should not be blank	- Event schedule To Time is required
Guest Email	Should not be left blank Should accept valid email values separated by comma.	- "Guest email id is required" - "One of the guests in the list has an invalid email id"
Description	No validation (Optional to type some text content)	- Nil

Notes on Guest Email:

- An email is a string separated into 2 parts by @ symbol - personal_info and domain name.
- The personal_info can contain upper- or lower-case letters, digits(0-9), characters(! # $ % & ' * + - / = ? ^ _ ` { | } ~.). The character .(dot) provided should not be the first or last character. Two or more dots cannot come together.
- The domain name part can contain letters, digits, hyphens, and dots. Some commonly used patterns are: abc@yahoo.co.in, xyz@niit.com, xyz@gmail.com etc.

Task 3: Add Validators to the Form Controls (Cont'd.)

- The custom validator function should be added to check whether each email id is valid individually. Ensure that the email ids of different guests are separated by a comma (as given in the code below).

```
checkIfGuestEmailsAreValid(c: AbstractControl) {
  if (c.value !== '') {
    const emailString = c.value;
    const emails = emailString.split(',').map((e: string) => e.trim());
    const emailRegex =
/^(([^<>()[\]\.,;:\s@\"]+(\.[^<>()[\]\.,;:\s@\"]+)*)|(\".+\"))@(([^<>()[\]\.,;:\s@\
"]+\.)+[^<>()[\]\.,;:\s@\"]{2,})$/i;
    const anyInvalidEmail = emails.every((e: string) => e.match(emailRegex) !==
null);
  if (!anyInvalidEmail) {
    return { InvalidGuestEmails: true }
  }}
return null;
}
```

Task 3: Add Validators to the Form Controls(Cont'd.)

- Modify the liveStreamForm by adding the validators to it.
  - Built-in validators such as '`required`' and '`maxLength`' should be added to the form properties.
  - Add the custom validator along with the built-in validators for the **guests** form property

```
liveStreamForm = this.fb.group({
// maxLength Validator
  eventTitle: ['', [Validators.required, Validators.maxLength(100)]],
  fromDate: ['', [Validators.required]],
  fromTime: ['', [Validators.required]],
  toTime: ['', [Validators.required]],
  description: [''],
// Every Guest needs to have a valid Email Address
  guests: ['', [Validators.required, this.checkIfGuestEmailsAreValid]]
})
```

Task 4: Create the HTML Form in the Template

- Build an HTML form using `<form>` tag and form controls using `<mat-form-field>` tag.

```
<form  [formGroup]="liveStreamForm" (ngSubmit)="onSubmit()">
```

- Use <input type="text"> for the eventTitle property.

```
<input matInput placeholder="Add Event Title" id="eventTitle"
formControlName="eventTitle" type="text">
```

Notes:

1. The form model is associated with this HTML form template using the `formGroup` directive and `formControlName` directive.
2. Style the form with custom styles to get the expected output.

Task 4: Create the HTML Form in the Template (Cont'd.)

- Use `<mat-datepicker>` for `fromDate` property.

```
<mat-form-field appearance="fill">
  <input [min]="minDate" placeholder="Event Date" matInput
[matDatepicker]="eventDatePicker" formControlName="eventDate">
  <mat-datepicker-toggle matSuffix [for]="eventDatePicker">
  </mat-datepicker-toggle>
  <mat-datepicker #eventDatePicker></mat-datepicker>
  <mat-error *ngIf="eventDate?.errors?.required">
     Event scheduled Event Date is required
  </mat-error>
</mat-form-field>
```

- Use `<textarea>` for event description and guest email properties.

Task 4: Create the HTML Form in the Template (Contd.)

- Use `<mat-autocomplete>` for `fromTime` and `toTime` form properties whose values are to be populated from the component's `timeValues` array property.

```
<mat-form-field appearance="fill">
  <input type="text" placeholder="From Time" matInput name="fromTime"
formControlName="fromTime" [matAutocomplete]="autoFromTime">
  <mat-autocomplete #autoFromTime="matAutocomplete">
    <mat-option *ngFor="let time of timeValues" [value]="time">
      {{time}}
     </mat-option>
  </mat-autocomplete>
  <mat-error *ngIf="fromTime?.errors?.required">
    Event scheduled From Time is required
  </mat-error>
</mat-form-field>
```

- Use `<mat-chip-list>` to display each guest email as a chip populated from the `guestList` array.
- Use `<button type="submit">` for saving the form details. It should be disabled for invalid values.

Task 5: Handle Form Validation

- Use the `formControlName` value to access the validation errors associated with each of the form control elements. The code below is given for `eventTitle` property.

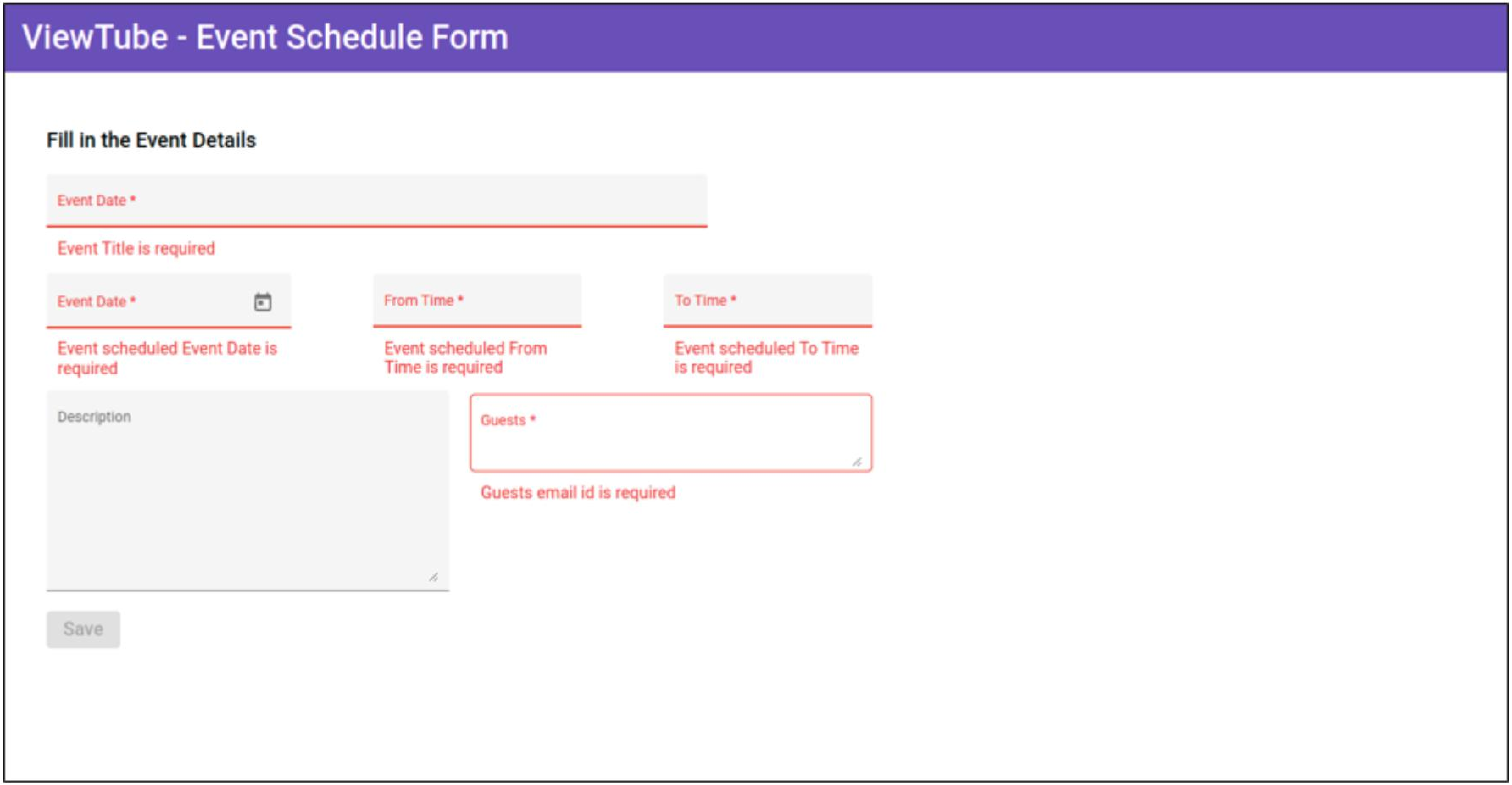
```
<input  matInput placeholder="Add Title" name="eventTitle"
id="eventTitle" formControlName="eventTitle" type="text">
```

- Use `<mat-error>` to display the validation error messages when input values are invalid. The code below displays the error message for an invalid guest email id.

```
<mat-error *ngIf="liveStreamForm.controls.guests.errors?.invalidGuestEmails" >
    One of the guests in the list has an invalid email id
/mat-error>
```

- Provide custom styles for displaying the error messages.

Expected Output: Form With Validation Errors



ViewTube - Event Schedule Form

Fill in the Event Details

Event Date *

Event Title is required

Event Date *

Event scheduled Event Date is required

From Time *

Event scheduled From Time is required

To Time *

Event scheduled To Time is required

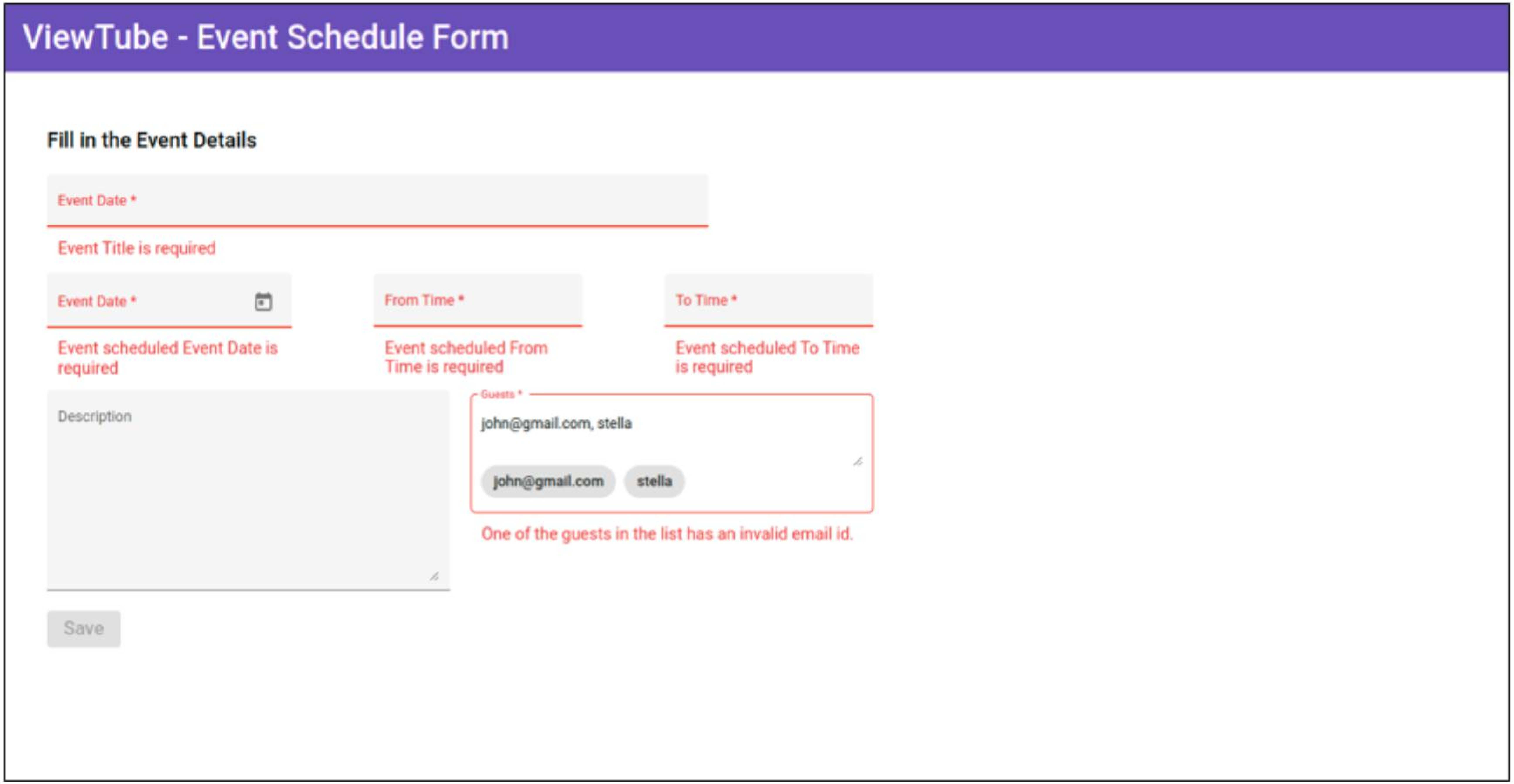
Description

Guests *

Guests email id is required

Save

Expected Output: Form With Validation Errors (Cont'd.)



ViewTube - Event Schedule Form

Fill in the Event Details

Event Date *

Event Title is required

Event Date *

Event scheduled Event Date is required

From Time *

Event scheduled From Time is required

To Time *

Event scheduled To Time is required

Description

Guests *

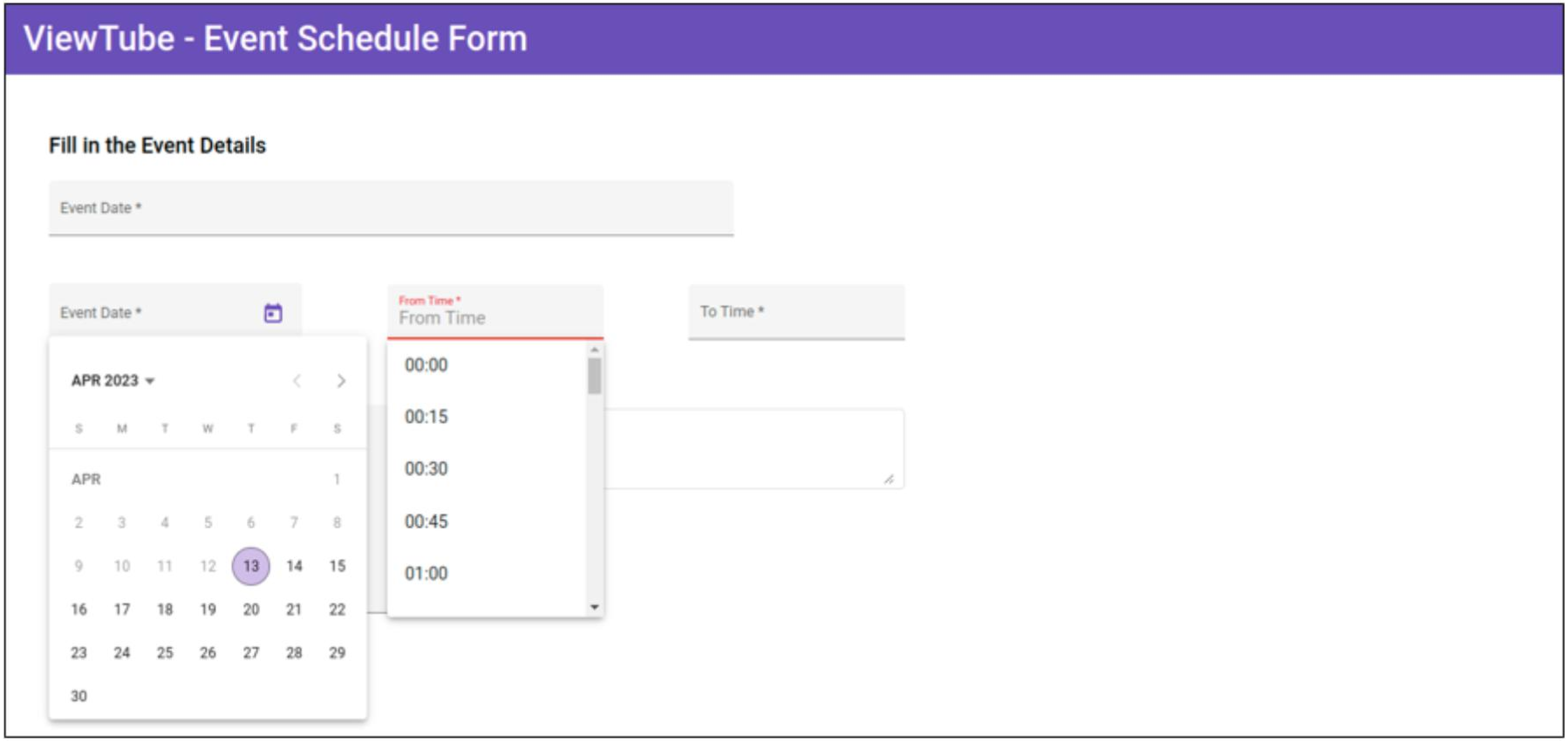
john@gmail.com, stella

john@gmail.com stella

One of the guests in the list has an invalid email id.

Save

Expected Output: Event Date and From Time



ViewTube - Event Schedule Form

Fill in the Event Details

Event Date *

Event Date *

From Time *
From Time

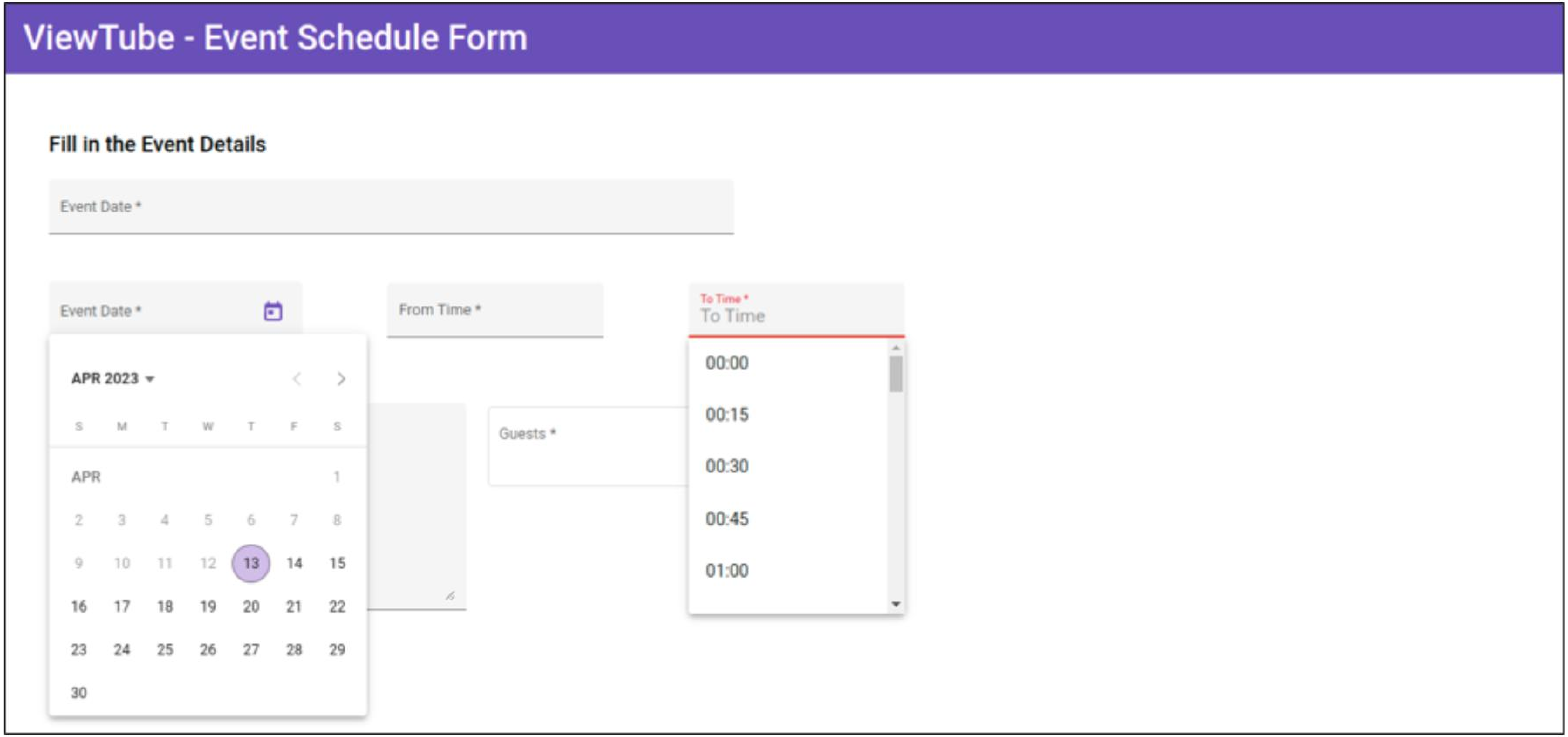
To Time *

APR 2023

S	M	T	W	T	F	S
APR						1
2	3	4	5	6	7	8
9	10	11	12	13	14	15
16	17	18	19	20	21	22
23	24	25	26	27	28	29
30						

00:00
00:15
00:30
00:45
01:00

Expected Output: Event Date and To Time



ViewTube - Event Schedule Form

Fill in the Event Details

Event Date *

Event Date *

From Time *

To Time *
To Time

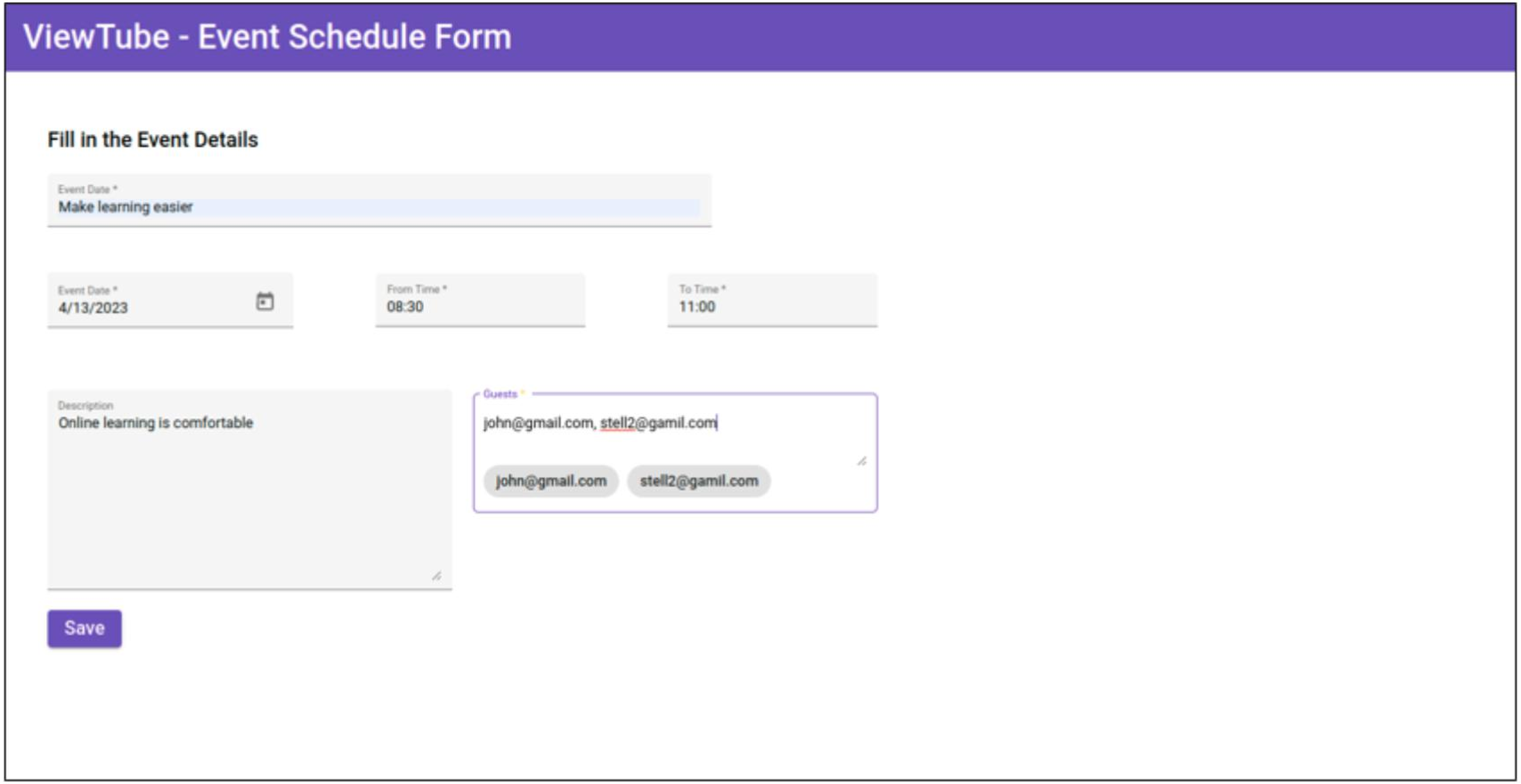
APR 2023

S	M	T	W	T	F	S
APR						1
2	3	4	5	6	7	8
9	10	11	12	13	14	15
16	17	18	19	20	21	22
23	24	25	26	27	28	29
30						

Guests *

00:00
00:15
00:30
00:45
01:00

Expected Output: Form With Valid Values



ViewTube - Event Schedule Form

Fill in the Event Details

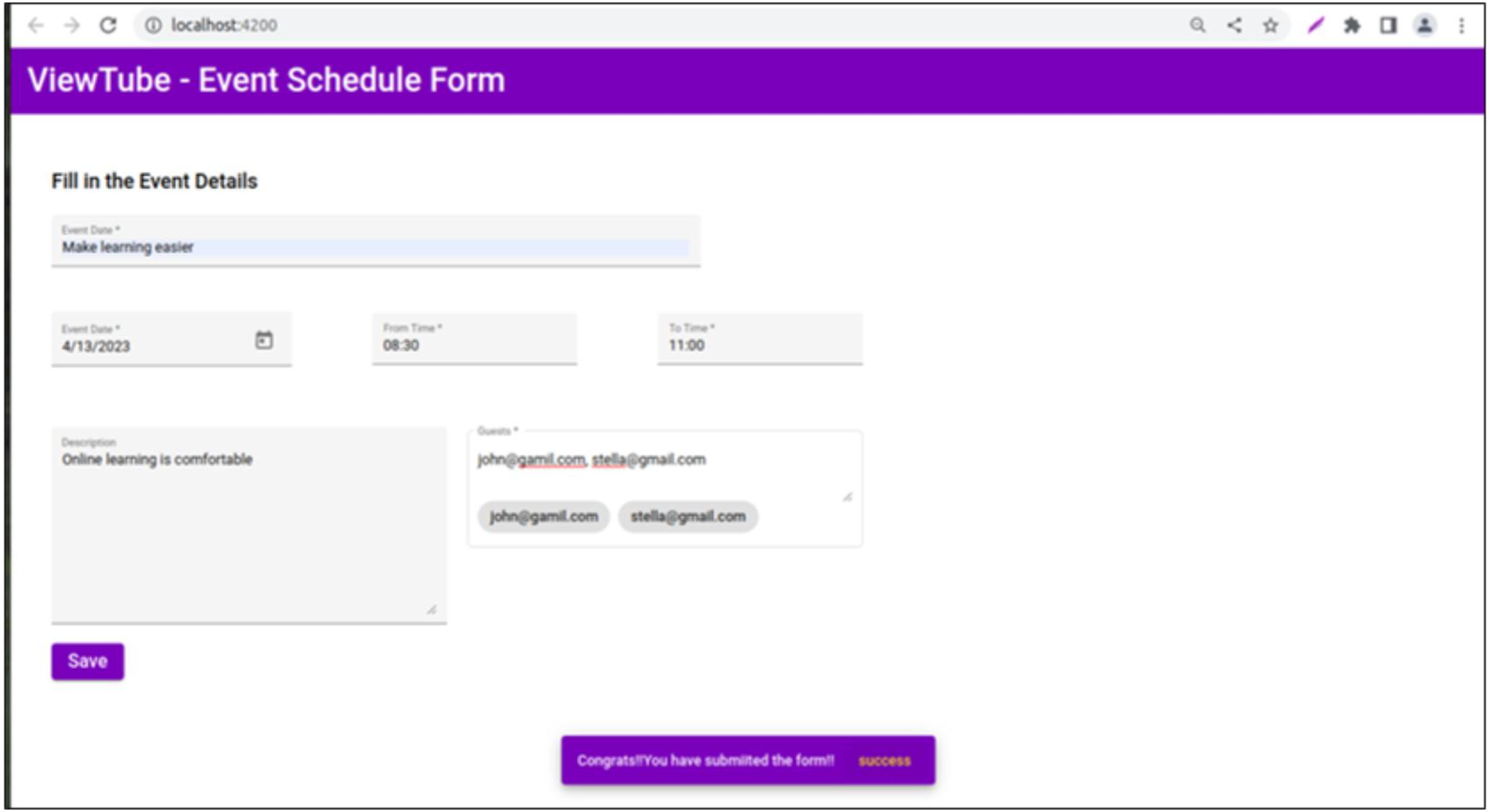
Event Date *	Make learning easier
Event Date *	4/13/2023
From Time *	08:30
To Time *	11:00
Description	Online learning is comfortable
Guests *	john@gmail.com, stell2@gamil.com

Task 6: Display Notification Message Upon Successful Form Submission

- Inside the `StudioLivestreamComponent` class, define the `onSubmit()` method, which calls the `LiveStream` service method to save the form data in the `json-server`. The application should display a notification message "Congrats, you have submitted the form!!" using a snack bar upon successful form submission.

```
let liveStream: LiveStreamEvent = this.liveStreamForm.value as LiveStreamEvent;
this.liveStreamService.addLiveStreamEvent(liveStream).subscribe({
       next: data => {
              this._snackBar.open('Congrats!!You have submiited the form!!' ,
'success', { duration: 5000, panelClass: ['mat-toolbar', 'mat-primary']}
       )},
       error: err => {
             this._snackBar.open('Failed to register user!! Please Try Again
Later', 'Failure', {duration: 3000,panelClass: ['mat-toolbar', 'mat-warn']}
         )}
  });
```

Expected Output: Successful Form Submission



localhost:4200

ViewTube - Event Schedule Form

Fill in the Event Details

Event Date *
Make learning easier

Event Date *
4/13/2023

From Time *
08:30

To Time *
11:00

Description
Online learning is comfortable

Guests *
john@gamil.com, stella@gmail.com

john@gamil.com stella@gmail.com

Save

Congrats!!You have submitted the form!! success

Test the Solution Locally

Test the solution first locally and then on the `CodeReview` platform. Steps to test the code locally are:

- From the command line terminal, set the path to the folder containing cloned boilerplate code.
- Run the command `ng test` or `npm run test` to test the solution locally and ensure all the test cases pass.
- Refactor the solution code if the test cases are failing and do a re-run.
- Finally, push the solution to git for automated testing on the `CodeReview` platform.