Lamp with Customer Support Chatbot

Mini Project Report

Submitted in partial satisfaction of the requirements for the
Degree of Master of Science in Computer Science

Submitted by

Deepak Gupta
(Reg. No: 20MCS007)

Under the guidance of

Dr. N. A. Sheela Selvakumari, M.Sc., M.Phil., Ph.D
Associate Professor,
Department of Computer Science,
Sri Krishna Arts and Science College

Department of Computer Science,
Sri Krishna College of Arts and Science
An Autonomous College Affiliated to Bharathiar University
Re-Accredited by NAAC with 'A' Grade
Kuniyamuthur(P.O.),
Coimbatore - 641 008.

January, 2022

Certificate

This is to certify that the project entitled **"Lamp with Customer Support Chatbot"** submitted in partial satisfaction of the requirem -ents for the **Degree of Master of Science in Computer Science** to **Sri Krishna Arts and Science College**, an Autonomous College affiliated to Bharathiar University, Coimbatore, is a record of bonafide work carried out by **Deepak Gupta (Reg. No. - 20MCS007)** under my supervision and guidance, that no part of this project has been submitted for the award of any other degree or diploma and the work has not been published in popular journal or magazine.

Signature of the Guide

Signature of the HOD **Signature of the Principal**

Viva-Voce conducted on: _____

Internal Examiner **External Examiner**

Declaration

I hereby declare that the project entitled **"Lamp with Customer Support Chatbot"** submitted to **Sri Krishna Arts and Science College**, an Autonomous College affiliated to Bharathiar University, Coimbatore in partial fulfillment of the requirements for the degree of Masters of Science in Computer Science is an original work and it has not been previously formed the basis for the award of any Degree, Diploma, Associate ship, Fellowship or other similar title to any university or body during the period of my project.

Place: Coimbatore
Date: 04 Jan, 22

Signature of the Candidate

Deepak Gupta
(20MCS007)

Acknowledgement

I convey my profound gratitude to **Dr. K. Sundararaman, M.Com., M.Phil., Ph.D., CEO,** Sri Krishna Institutions for giving me this opportunity to undergo this Project.

My heartfelt thanks to **Dr. P. Baby Shakila, M.Sc., M.Phil., Ph.D., M.Ed., MBA., Principal,** Sri Krishna Arts and Science College for giving me this opportunity to undergo this Project.

It is my prime to solemnly express my sense of gratitude to **Dr. C. Sunitha, MCA., M.Phil., Ph.D., Head of Department,** Department of Computer Science, Sri Krishna Arts and Science College.

I would like to extend my thanks and unbound sense for the timely help and assistance given by **Dr. N. A. Sheela Selvakumari, M.Sc., M.Phil., Ph.D, Associate Professor,** Department of Computer Science, Sri Krishna Arts and Science College in completing the project. Her remarkable guidance at every stage of my project was coupled with suggestion and motivation.

I am earnestly thankful to teaching and non-teaching faculty members of Department of Computer Science, Sri Krishna Arts and Science College for the support and guidance provided for me to complete the project.

I take this opportunity to thank my parents and friends for their constant support and encouragement throughout this training.

To my friends

Abstract

The lamp is an all-new extension using which you can manage all your meetings being assured that now you ain't gonna miss out on any of them with the best UI/UX design, with the additional support of appearances i.e. light or dark. we are storing the necessary data of the users. so (that) we can provide the services to them. but don't worry, we are storing metadata with the encryption and it will automatically delete the users' scheduled meetings data; the customer support chatbot helps users by delivering the answer to the asked queries.

Contents

1 Introduction

During the pandemic crisis, we all were facing many challenges in attending offline/in-person classes. so, whereby everything like academic classes, official meetings, meetups and so on are undertaken remotely and on digital platforms. research suggested, google meet is a widely used and secure digital platform for meetings. most of the colleges, universities and companies are also started using google meet for academic and/or official purposes.

One of the challenges is not joining the meeting on time because of personal, technical issues. to support this, we have developed a platform for students, working professionals to schedule and attend meetings on time. so (that) the user can join the classes in every situation on time. and they will get full attendance for the same.

To make it easy to use, we have designed a very understandable UI & UX along with a customer support chatbot; and users feedbacks and bug reports to improve our system.

1.1 Objective

Lamp aim is to ensure that the students and working professionals will not miss out on any meetings and delays in joining the same. It will also provide the best user experience to make it understandable to everyone.

1.2 Problem Statement

To develop an application that automates the google meet to support students and working professionals during this pandemic crisis. so (that) they can attend meetings on time without taking care about the schedule. the application should need to be in accordance with the following points -

1. **Functionality:**
    - Users can schedule and delete meetings,
    - Allow users to automatically join/ask to join the google meet,

- Intent-based customer support chatbot,
- Send setup details and credentials to the user's provided email address, and
- Allow users to give feedback and report an issues

2. **Appearances:**
   - Light mode
   - Dark mode
3. **Security & Privacy:**
   - User authentication & authorization
   - Encryption of metadata
   - Access control
   - The microphone and camera will be turned off before getting inside the meet

1.3 Technology Stack

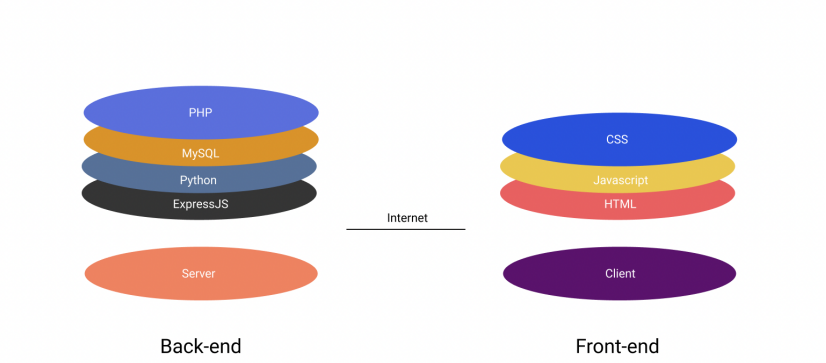


Figure 1: Technology Stack[1]

[1]These are the major technologies that have been used in this project.

2 System Analysis

Systems analysis is the process of studying a procedure or business to identify its goal and purposes and create systems and procedures that will efficiently achieve them. another view sees system analysis as a problem-solving technique that breaks down a system into its component pieces, and how well those parts work and interact to accomplish their purpose.
This involves stipulating system requirements from existing systems, potential users' inputs, and further analysis. models are developed or decided on, and the result ensures the system in question is properly understood.

2.1 Existing System

There are some browser-based extensions. they are allowing users to directly join the meetings. to use user have to just install the extension in the web browser and whenever the user will click on the meet link, then the user will automatically join/ask to join the meeting.

Drawback of the existing system -

- The user can not manage and/or schedule the classes
- Users have to manually turn off the cam and/or mic before joining the meet or else they will get into the meet without turning off the mic and/or camera

2.2 Proposed System

This application allows users to attend, schedule meetings, and attend scheduled meetings on time, it works as an assistant. So (that) users can attend meetings while sleeping, working or at a party. along with the customer support chatbot and bright/dark appearances.

Advantages of the proposed system -

- Easy to schedule and/or manage the classes
- The app will automatically turn off the cam and/or mic

- Access control and encryption of metadata
- Available in both dark and bright modes
- Customer support chatbot for the ease of the user

2.3 Module Description

The lamp contains the following modules -

1. **Web Application**

   This module contains the following components:

   - **Home [/]; Status[/status]**
     This is the landing page of the web application, where the user will be asked to authenticate first to process further to schedule the meeting and check meetings.
     The status page is used to check current meetings dynamically to schedule automatically.

   - **Authentication[/signup and /login]**
     In authentication, the user will be asked to either login or register by providing the details.

   - **Setup [/setup]**
     This page is to schedule a meeting with title, date & time and meet URL and to delete the already created meet.
     The user can directly click on the "Let's meet" button on the home page to schedule meetings.

   - **Build [/build]**
     Here in this component, we have provided the link to download the extension either using the direct link of the extension or by downloading the source code of the extension to add.

   - **Report a bug [/report]**
     This will be visible only on the home page or users can directly go by redirecting to /report or the chatbot will ask you to report a bug based on users inputs. these reports will be used to improve the quality of the lamp.

- **Feedback [feedback]**
  This will be visible only on the home page or users can directly go by redirecting to /feedback or the chatbot will ask users for feedback based on users inputs. This feedback will be used to improve the functionality and the quality of the lamp.

- **Light/Dark mode [widget]**
  On every page, users can switch to any one of dark or bright appearances.

- **Customer support chatbot [widget]**
  This chatbot will help users to provide brief information on the steps to use the lamp and some basic conversations about how is the lamp? and other queries.

2. **Web Browser Extension**

   This browser-based extension was made with the intention to make the application more accurate in terms of privacy and security. so (that) users will not have to worry about the technical glitches of the camera and/or mic, to be turned off.

3 System Design & Development

System design is the process of designing the elements of a system such as the architecture, modules and components, the different interfaces of those components and the data that goes through that system.
System development is a problem-solving process where we bring to bear appropriate elements of mankind's knowledge base to create new knowledge specific to the problem and, as a result, define a solution to the problem.

3.1 System Design

The design and architecture of this application will help the developer /designer to understand the whole system including the functionalities in this application and its user flow.

The following diagram provides an overview of the whole system

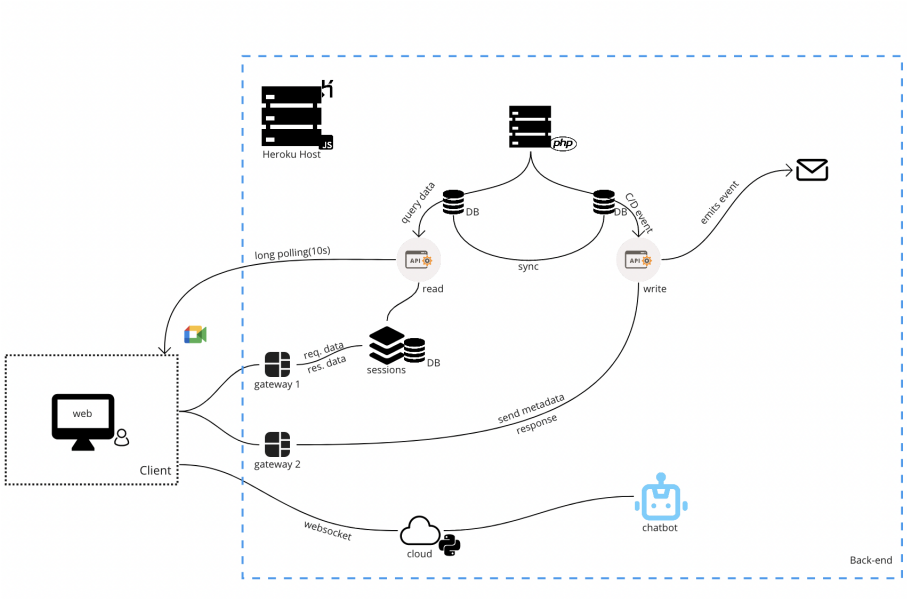


Figure 2: System Design

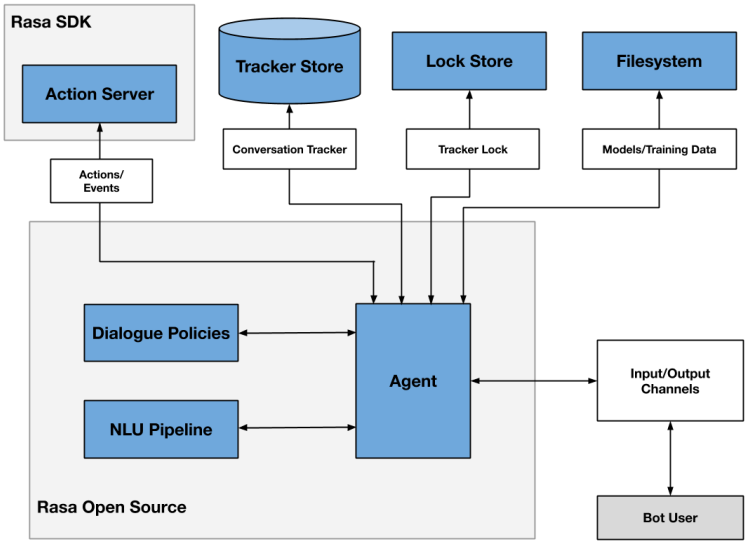


Figure 3: Customer Support Chatbot Architecture

login

#	Name	Type	Index
1	id	int(11)	primary key
2	created	varchar(15)	
3	name	varchar(150)	
4	username	varchar(150)	unique
5	email	varchar(255)	
6	password	varchar(255)	

meetings

#	Name	Type	Index
1	mid	int(11)	primary key
2	uid	int(11)	foreign key
3	mname	varchar(255)	
4	murl	varchar(500)	
5	mtime	datetime	

feedback

#	Name	Type	Index
1	fid	int(11)	primary key
2	fname	varchar(150)	
3	femail	varchar(150)	
4	comment	varchar(1000)	

report

#	Name	Type	Index
1	rid	int(11)	primary key
2	title	varchar(150)	
3	remail	varchar(250)	
4	description	varchar(250)	
5	file	varchar(150)	

Figure 4: Database Design

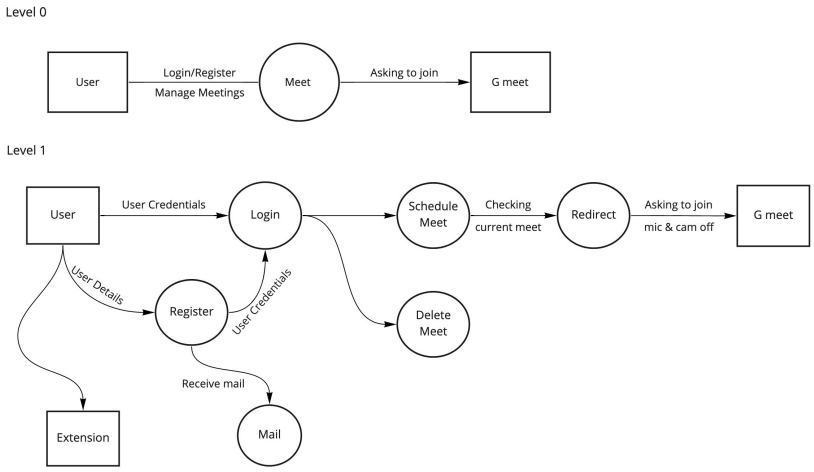


Figure 5: Data Flow Diagram

3.2 Application Development

To develop this application we will use ExpressJS, PHP, Python and some external libraries as a back-end and HTML, CSS and JS for the front-end.
We have used MySQL database for every action of this application like creation, deletion of meetings, feedback and report an issue form.

Web Application

The web application is developed using expressJS by submitting a request to an API and that API retrieves the requested data from the external server or program and delivers it back to the client (API call). for all the transactions from the signup page to the deletion of the meeting, we are using the same architecture.

Browser Extension

In it, there are three major functionality - turn off the mic, turn off the camera, detect the google meet URL.
if the extension will detect that the URL contains “meet.google.com”. It will turn off the mic, camera and ask to join/join the meeting.

Customer Support Chatbot

The chatbot is developed in python and connected to the web application using WebSocket. we have used the conversation dataset and train that dataset with NLU to understand the user's input and provide the response based on the intents. simply put, the chatbot interprets the input and process the user's words or phrases and give an instant pre-set answer based on intents.

The challenging part of this project was using the database on Heroku because that was a premium service of Heroku. so as a solution to the above problem, we designed restful APIs and hosted them over another domain and used them for every action of this application.

4 Software Testing & Deployment

Software testing is the process of evaluating and verifying that a software product or application does what it is supposed to do. the benefits of testing include preventing bugs, reducing development costs and improving performance.
Application deployment is the mechanism through which applications, modules, updates, and patches are delivered from developers to users.

4.1 Software Testing Methodologies

There are many different types of software tests, each with specific objectives and strategies:

1. **Unit testing:**
   It is the process of ensuring individual components of a piece of software at the code level are functional and work as they were designed to. we wrote some test cases and run the tests in the development phase itself before proceeding with the testing phase. for instance,

   Unit test case 1:
   To check whether the API for the login/registration is working correctly or not – Pass

   Unit test case 2:
   To check whether the API for the creation and deletion of the meeting is working correctly or not – Pass

   Unit test case 3:
   To check whether the response sent by the all meetings details is as expected or not – Pass

   Unit test case 4:
   To check whether the chatbot is giving responses correctly or not – Pass

   Unit test case 5:
   To check whether the email is sent to the user on the account creation or not – Pass

2. **Integration testing:**
   After each unit is thoroughly tested, it is integrated with other units to create modules or components that are designed to perform specific tasks or activities. these are then tested as groups through integration testing to ensure whole segments of an application behave as expected (i.e, the interactions between units are seamless). these tests are often framed by user scenarios, such as logging into an application or opening files. for instance,

   Test case 1:
   Logging into the application by providing the user credentials – Pass

   Test case 2:
   Scheduling a meeting and joining the same – Pass

3. **Security testing:**
   With the rise of cloud-based testing platforms and cyber attacks, there is a growing concern and need for the security of data being used and stored in the software. In this security testing, we are testing whether the information and data in a system are protected or not. The goal is to purposefully find loopholes and security risks in the system that could result in unauthorized access to or the loss of information by probing the application for weaknesses.

4. **System testing:**
   Since we have already checked our system up to the previous integration level of testing and it's working as expected. In this system testing, we are checking for the whole system in the context of either system requirement specifications or functional requirement specifications or in the context of both. the design and behaviour of the system and also the expectations of the customer; the system is beyond the bounds mentioned in the software requirements specification.

5. **Acceptance testing:**
   Since we are ready to deliver the product as the lamp is

meetings all the end-users requirements. so we have started the beta testing of the lamp with the QA team. it is key to getting real feedback from potential customers and can address any final usability concerns.

As a result, it's time to launch the lamp in a production environment for the intended users.

4.2 Software Deployment & Maintenance

All set, the lamp application is deployed over the Heroku [2] for the users, the web browser extension is published on firefox; for other browsers, users can download the source code of the extension [3] and the chatbot is also available to talk and is connected to the lamp application using API.
We will update the lamp to improve the performance and functionality based on the user's feedback and/or report a issue form responses. or for future enhancements, if demanded.

[2] Lamp - `https://autogmeet.herokuapp.com/`

[3] Browser extension - `https://github.com/gdeepak884/mini_project_lamp/tree/main/extensions/autogmeet`

5 Conclusion

We have developed an application that fulfilled all the requirements mentioned in the problem statement; it will help users to attend, schedule meetings or can do CD[4] operations with the additional support of appearances i.e. light or dark. we are storing the necessary data of the users. so (that) we can provide the services to them. but don't worry, we are storing metadata with the encryption and it will automatically delete the users' scheduled meetings data; this application will be used by the students and working professionals for the management of the meetings and to attend meetings on time while working or at the party without taking care of time on the clock.

[4] Create Delete

6 Scope of future enhancement

Since users requirements are dynamic as it's being used. so it is quite difficult to make it up with all. Some of the static future enhancements that can be done to the system are that

- Allow users to schedule meetings by importing the timetable of either college/school or of any kind in any format
- Allow users to create a meet URL with title and time & date and with an option to share the same.

7 Bibliography

[1] *Introduction to Rasa Open Source.* RasaDocs, 2021. `https://rasa.com/docs/rasa/`.

[2] *ExpressJS API documentation.* ExpressjsDocs. `https://expressjs.com/en/5x/api.html`.

[3] Wiyono, BambangBudi: *The Utilization of "Google Meet" and "Zoom Meetings" to Support the Lecturing Process during the Pandemic of COVID-19.* IEEE, 2021.

[4] Odhiambo, Didacus: *System Design.* Medium. `https://bit.ly/3srmRJG/`.

[5] *Agile 101.* AgileDocs. `https://www.agilealliance.org/agile101/`.

[6] *How does software testing work?* IBMDocs. `https://www.ibm.com/topics/software-testing`.

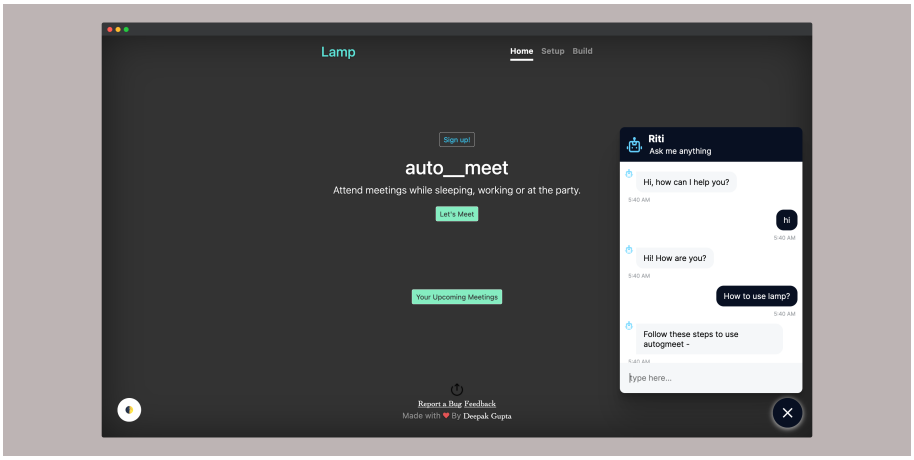
Appendices

A Application Views

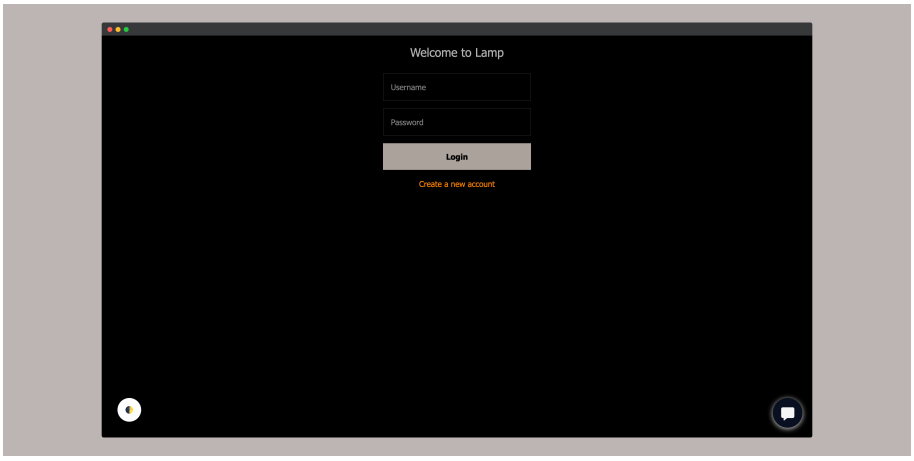
Desktop View

Main Pages

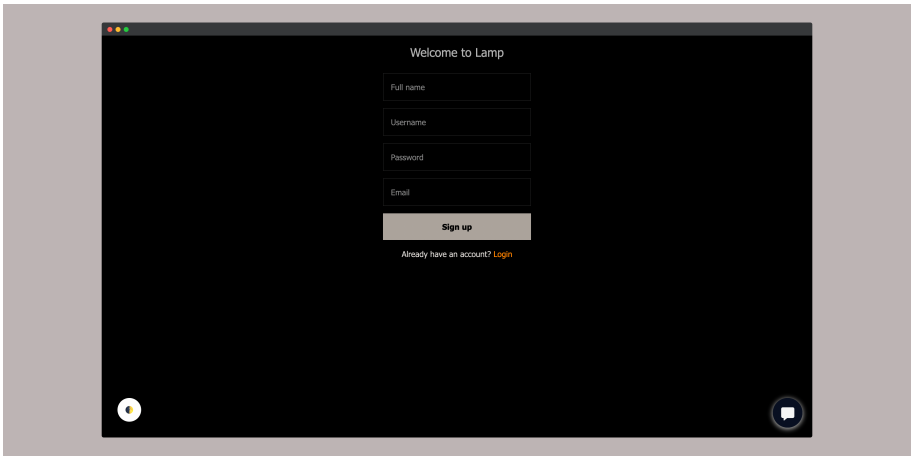
Home



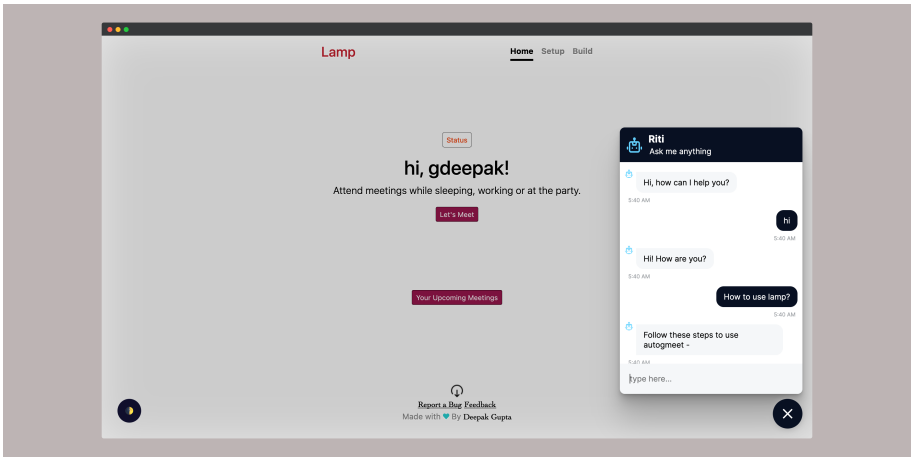
Login



Signup

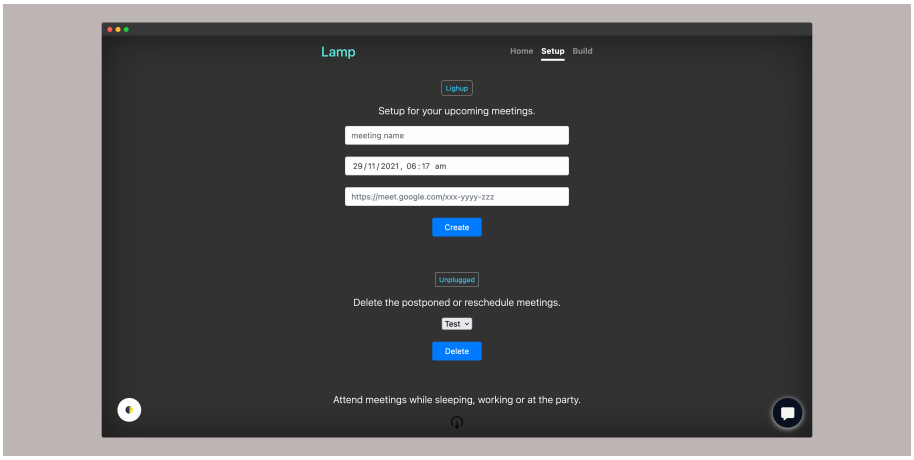


Home Page(After Login)

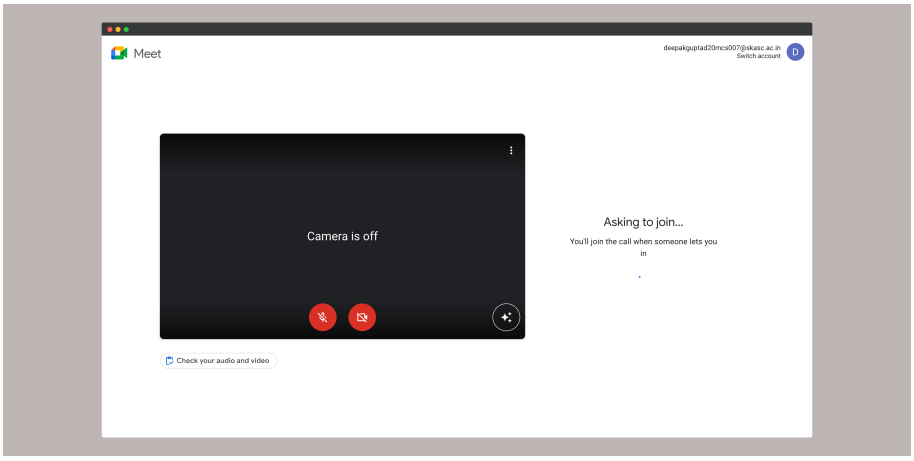


Setup Page

Managing meeting

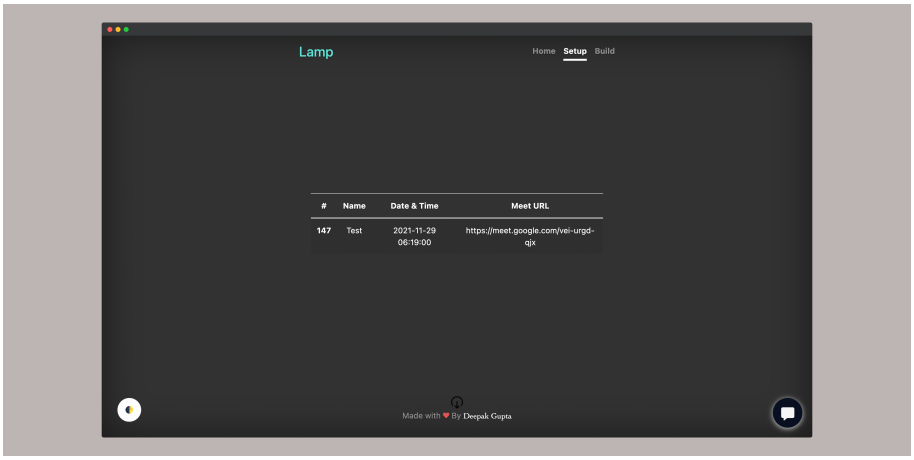


Joining meet

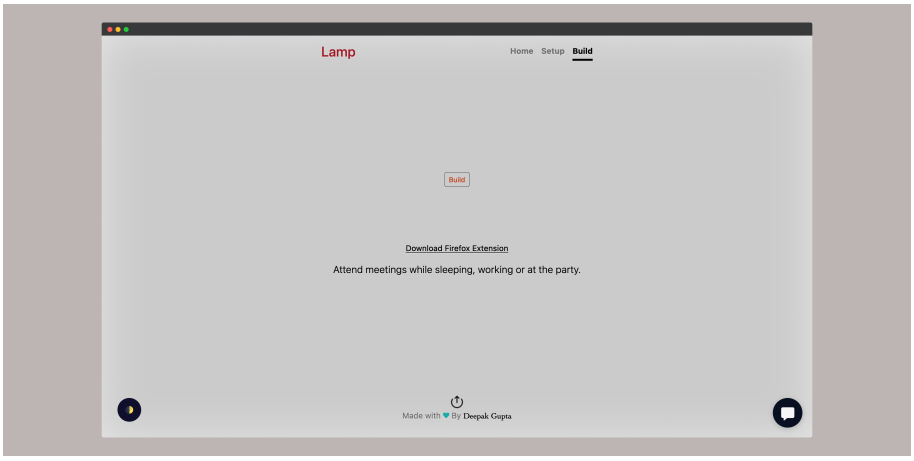


Setup Page

All meetings

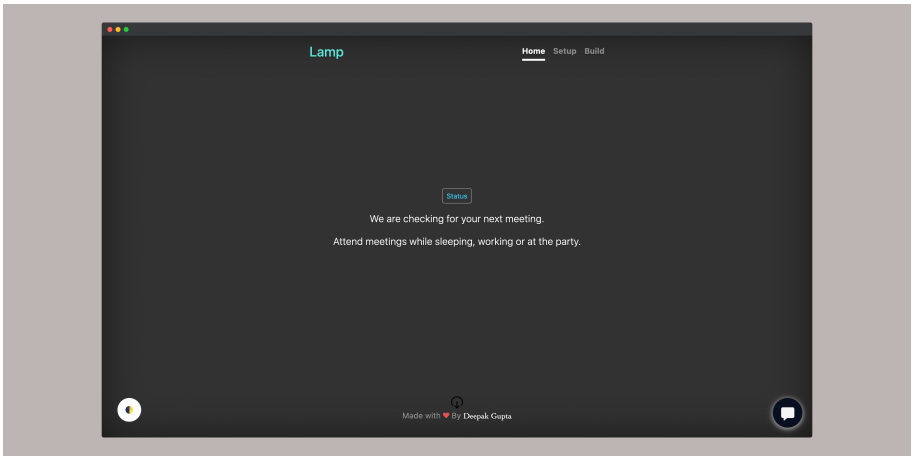


Build

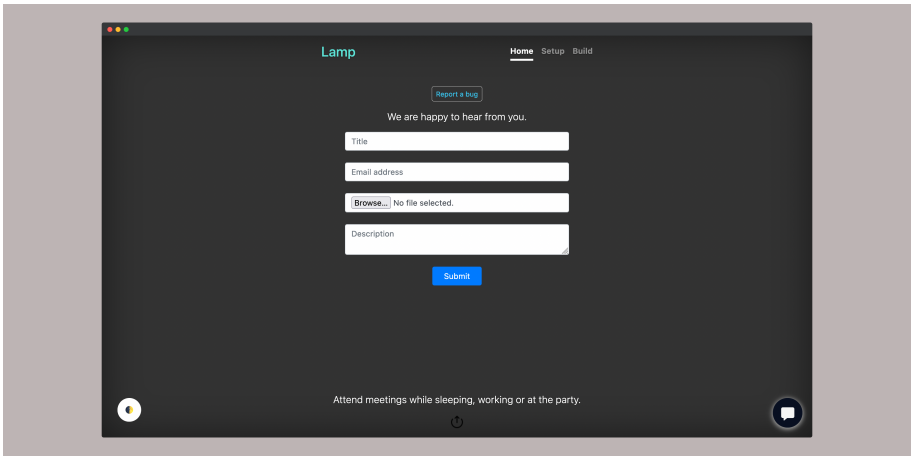


Additional Pages

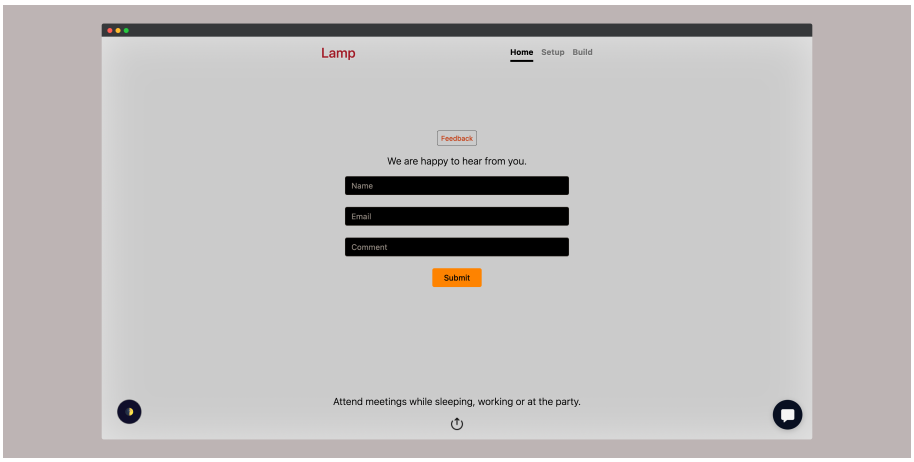
Checking for the meeting



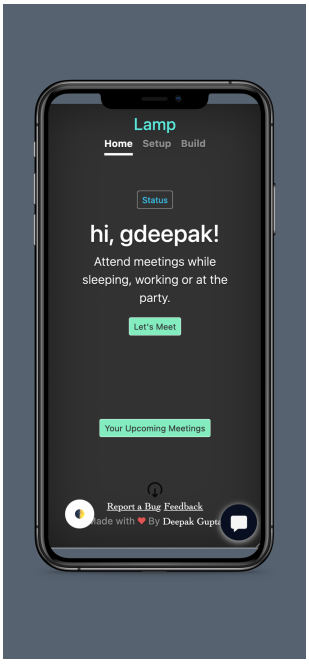
Report a bug

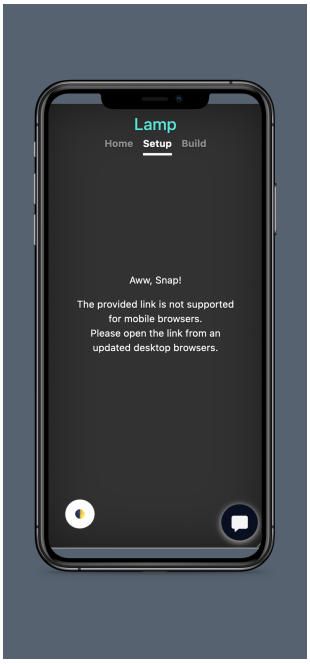


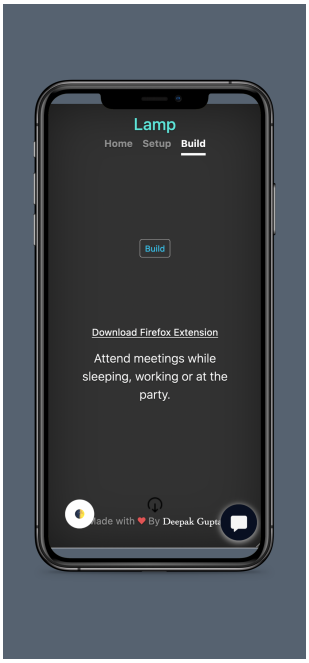
Feedback

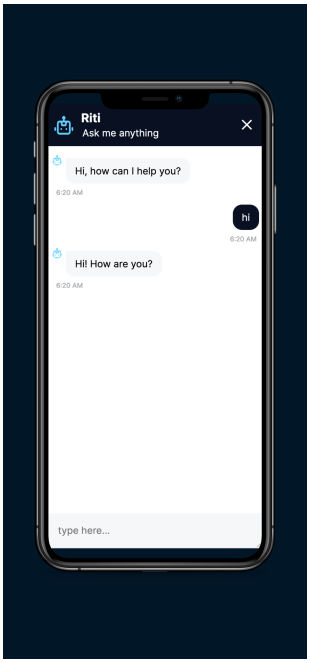

Lamp
Home
Setup
Build
Feedback
We are happy to hear from you.
Name
Email
Comment
Submit
Attend meetings while sleeping, working or at the party.

Mobile View


Lamp
Home Setup Build
Status
hi, gdeepak!
Attend meetings while sleeping, working or at the party.
Let's Meet
Your Upcoming Meetings
Report a Bug Feedback
Made with ❤ By Deepak Gupta


Lamp
Home Setup Build
Aww, Snap!
The provided link is not supported for mobile browsers.
Please open the link from an updated desktop browsers.


Lamp
Home Setup Build
Build
Download Firefox Extension
Attend meetings while sleeping, working or at the party.
Made with ❤ By Deepak Gupta


Riti
Ask me anything
Hi, how can I help you?
6:20 AM
hi
6:20 AM
Hi! How are you?
6:20 AM
type here...

B Source Code

Web App [5]

app.js

```
1
2  process.env.NODE_TLS_REJECT_UNAUTHORIZED = 0
3  const express = require('express')
4  const path = require('path');
5  const bodyParser = require('body-parser')
6  const app = express()
7  var fs = require("fs");
8  app.set('trust proxy', 1);
9  var fetch = require('cross-fetch');
10 var fetch = require('node-fetch');
11 var Darkmode = require('darkmode-js');
12 var http = require("http");
13 var session = require('express-session');
14 const e = require('express');
15 const multer  = require('multer')
16 const upload = multer({ dest: 'assets/' })
17 const { json } = require('body-parser');
18 const { exit } = require('process');
19 app.use(express.json());
20 new Darkmode().showWidget();
21
22 app.use(session({
23   secret: 'secret',
24   resave: true,
25   saveUninitialized: true
26 }));
27
28 app.use(bodyParser.json());
29 app.use(bodyParser.urlencoded({ extended: false }))
30 app.use(express.static(path.join(__dirname, 'public')));
31 app.set('view engine', 'ejs');
32
33 app.use(function(req, res, next) {
34   res.locals.username = req.session.username;
35   next();
36 });
37
38 app.get('/', (req, res) => {
39   res.render('index')
40 });
41
42
43 app.get('/signup', (req, res) => {
44   var message = req.query.message;
45   res.render('signup',{message});
46 });
47
48 app.get('/login', (req, res) => {
```

[5] Full source code - https://github.com/gdeepak884/mini_project_lamp/

```
  var message = req.query.message;
  res.render('login',{message});
});

app.get('/feedback', (req, res) => {
  var message = req.query.message;
  res.render('feedback',{message});
});

app.get('/report', (req, res) => {
  var message = req.query.message;
  res.render('report',{message});
});

app.post('/creating', function(request, response) {
  var created = Date.now();
  var name = request.body.name;
  var email = request.body.email;
  var password = request.body.password;
  var username= request.body.username;
  if (name && email && username && password) {
    fetch(`https://example.in/api/v1/signup.php?created=${created}&username=${username}&name=${name}&email=${email}&password=${password}`)
    .then(response => response.json())
    .then(data => {
      if(data.status=="200 OK") {
        if (data.message=="Account Created!") {
            fetch(`https://example.in/api/v4/send.php?username=${username}&email=${email}&password=${password}`)
            .then(response => response.json())
            .then(mail => {
              if(mail.status=="200 OK") {
                if (mail.message=="Sent") {
                }
              }
            }
              ).catch(function(err) {
               console.log(err);}
            );
          response.redirect(`/login?message=${data.message}`);
        }
      }
      else {
            response.status(200).redirect(`/signup?message=${data.message}`)
      }
    }
      ).catch(function(err) {
       console.log(err);}
      );
  }
  else {
      response.status(200).redirect(`/signup?message=Something went wrong`)
  }
});
```

```
101
102 app.post('/auth', function(request, response) {
103   var username= request.body.username;
104   var password = request.body.password;
105   if (username && password) {
106     fetch(`https://example.in/api/v1/users.php?username=${username
      }&password=${password}`)
107     .then(response => response.json())
108     .then(data => {
109       if(data.status=="200 OK") {
110         if (data.customers.uid.length > 0) {
111           var uid = data.customers.uid;
112           request.session.loggedin = true;
113           request.session.username = username;
114           request.session.uid = uid;
115           response.redirect('/');
116         }
117       }
118       else {
119             response.status(200).redirect('/login?message=Incorrect
      Username and/or Password!')
120       }
121     }
122       ).catch(function(err) {
123        console.log(err);}
124       );
125   }
126   else {
127       response.status(200).redirect('/login?message=Please enter
     Username and Password!')
128     }
129 });
130
131 app.get('/logout', (req, res) => {
132   req.session.loggedin = false;
133   req.session.destroy();
134   res.redirect('/');
135 });
136
137 app.get('/build', (req, res) => {
138   res.render('build')
139 });
140
141 app.get('/status', (req, res) => {
142   if(req.session.loggedin) {
143     res.render('status')
144   }
145   else {
146     res.redirect('/login');
147   }
148 });
149
150
151 app.get('/setup', function(request, response){
152   if(request.session.loggedin) {
153     var message = request.query.message;
154     var uid=request.session.uid
```

```
155       if (uid) {
156        fetch(`https://example.in/api/v1/view.php?uid=${uid}`)
157        .then(response => response.json())
158        .then(data => {
159       if(data!="") {
160           response.render('setup', {data,message})
161       }
162       else {
163         response.render('setup', {data,message})
164       }
165        }
166       ).catch(function(err) {
167        console.log(err);}
168          );
169       }
170   }
171   else {
172     response.redirect('/login');
173   }
174 });
175
176 app.get('/meetings', function(request, response){
177   if(request.session.loggedin) {
178     var uid=request.session.uid
179       if (uid) {
180        fetch(`https://example.in/api/v1/view.php?uid=${uid}`)
181        .then(response => response.json())
182        .then(data => {
183       if(data!="") {
184             meetings=data;
185          response.render('meetings', {data})
186       }
187       else {
188             response.render('meetings', {data})
189       }
190        }
191       ).catch(function(err) {
192        console.log(err);}
193          );
194       }
195   }
196   else {
197     response.redirect('/login');
198   }
199 });
200
201 app.post('/submitmeet', function(request, response){
202   if(request.session.loggedin) {
203   var uid= request.session.uid
204   var mname= request.body.meetName
205   var mtime= request.body.meetTime
206   var murl= request.body.meetUrl
207       if (uid && mname && mtime && murl) {
208        fetch(`https://example.in/api/v1/create.php?uid=${uid}&mname
    =${mname}&murl=${murl}&mtime=${mtime}`)
209        .then(response => response.json())
210        .then(data => {
```

```
211       if(data.status=="200 OK") {
212         if (data.message=="Inserted") {
213           response.redirect(`/setup?message=${data.message}`)
214         }
215       }
216       else {
217         response.status(200).redirect(`/setup?message=${data.
      message}`)
218       }
219        }
220       ).catch(function(err) {
221        console.log(err);}
222          );
223       }
224   }
225   else {
226     response.send('User Unauthenticated!');
227   }
228 });
229
230 app.post('/submitfeedback', function(request, response){
231   var fname= request.body.fname
232   var femail= request.body.femail
233   var comment= request.body.comment
234       if (fname && femail && comment) {
235        fetch(`https://example.in/api/v1/feedback.php?fname=${fname
      }&femail=${femail}&comment=${comment}`)
236        .then(response => response.json())
237        .then(data => {
238       if(data.status=="200 OK") {
239         if (data.message=="Submitted") {
240           response.redirect(`/feedback?message=${data.message}`)
241         }
242       }
243       else {
244         response.status(200).redirect(`/feedback?message=${data.
      message}`)
245       }
246        }
247       ).catch(function(err) {
248        console.log(err);}
249          );
250       }
251 });
252
253 app.post('/submitreport', upload.single('rfile'), function(request,
       response){
254   var title= request.body.title
255   var remail= request.body.remail
256   var description= request.body.description
257   var rfile= request.file.filename
258   var file = __dirname + '/assets/' + request.file.filename;
259   fs.rename(request.file.path,file,function(err){
260     if(err){
261       console.log(err);
262       response.send(500);
263     }
```

```
264     else {
265     if (title && remail && description && rfile) {
266        fetch(`https://example.in/api/v1/report.php?title=${title}&
    remail=${remail}&description=${description}&rfile=${rfile}`)
267        .then(response => response.json())
268        .then(data => {
269       if(data.status=="200 OK") {
270         if (data.message=="Submitted") {
271           response.redirect(`/report?message=${data.message}`)
272         }
273       }
274       else {
275         response.status(200).redirect(`/report?message=${data.
    message}`)
276       }
277        }
278       ).catch(function(err) {
279        console.log(err);}
280          );
281       }
282   }
283   });
284 });
285
286
287 app.post('/delete', function(request, response){
288   if(request.session.loggedin) {
289     var mid= request.body.delID
290       if (mid) {
291          fetch(`https://example.in/api/v1/delete.php?mid=${mid}`)
292          .then(response => response.json())
293          .then(data => {
294         if(data.status=="200 OK") {
295           if (data.message=="Deleted") {
296             response.redirect(`/setup?message=${data.message}`)
297           }
298         }
299         else {
300             response.status(200).redirect(`/setup?message=${data.
    message}`)
301         }
302          }
303         ).catch(function(err) {
304          console.log(err);}
305          );
306       }
307     else{
308       response.status(200).redirect(`/setup?message="Nothing to
    delete"`)
309     }
310    }
311     else {
312   response.send('User Unauthenticated!');
313     }
314 });
315
316 app.get('/api/v1', function(request, response){
```

```
317   if(request.session.loggedin) {
318     var uid=request.session.uid
319       if (uid) {
320        fetch(`https://example.in/api/v1/view.php?uid=${uid}`)
321        .then(response => response.json())
322        .then(data => {
323       if(data!="") {
324         response.setHeader('Content-Type', 'application/json');
325                 response.send(JSON.stringify(data));
326       }
327        }
328       ).catch(function(err) {
329        console.log(err);}
330          );
331       }
332   }
333   else {
334     response.redirect('/login');
335   }
336 });
337
338 app.get('/api/v1/delete', function(request, response){
339   if(request.session.loggedin) {
340     var mid= request.query.mid
341       if (mid) {
342          fetch(`https://example.in/api/v1/delete.php?mid=${mid}`)
343          .then(response => response.json())
344          .then(data => {
345         if(data.status=="200 OK") {
346           response.setHeader('Content-Type', 'application/json');
347                     response.send(JSON.stringify(data));
348         }
349          }
350         ).catch(function(err) {
351          console.log(err);}
352          );
353       }
354   }
355   else{
356     response.redirect('/login');
357   }
358 });
359
360 function setDaysTimeout(callback,days) {
361     var msInDay = 86400*1000;
362
363     var dayCount = 0;
364     var timer = setInterval(function() {
365         dayCount++;
366
367         if(dayCount == days) {
368            clearInterval(timer);
369            callback.apply(this,[]);
370         }
371     },msInDay);
372 }
373
```

```
const checkForMeeting = (request, response) => {
         var api_key=" "; // Enter your API key here
         fetch(`https://example.in/api/v1/show_users.php?hash=${
    api_key}&type=emails`)
         .then(response => response.json())
         .then(data => {
        if(data.status=="200 OK") {
          if (data.message=="User Details") {
            for(var i=0;i<data.customers.length;i++) {
            var emails=data.customers[i];
             fetch(`https://example.in/api/v3/build.php?content=Hey
    %20%E2%80%93%20we%20just%20wanted%20to%20make%20sure%20you%20
    saw%20our%20recent%20fundraising%20appeal.%20We%27re%20up%20
    against%20some%20very%20deep%20pockets,%20and%20we%20rely%20on
    %20your%20financial%20support%20to%20fuel%20our%20work.%20If%20
    you%27re%20able,%20can%20you%20chip%20in%20a%20donation(UPI:%20
    gdeepak884@oksbi)%20today%20to%20help%20us%20build%20and%20grow
    %20the%20movement%20for%20a%20healthier%20internet?&subject=We
    %20could%20use%20your%20help.&email=${emails}`)
                .then(response => response.json())
                .then(data => {
              if(data.status=="200 OK") {
                if (data.message=="Sent") {
                 }
              }
                }
              ).catch(function(err) {
              }
                );
            }
        }
          }
         }
        ).catch(function(err) {
        }
         );
      }

const PORT = process.env.PORT || 3000
app.listen(PORT, () => {
  setDaysTimeout(function() {
    checkForMeeting();
   }, 7);
  console.log('App listening on port 3000!');
});
```

Listing 1: app.js