



20. Emails & Advanced Auth

1. Why to send Email
2. Using SendGrid
3. Sending Email
4. Forgot Password Wireframe
5. Sending OTP
6. Reset Password





20.1 Why to send Email

1. **Welcome Emails:** Send a confirmation or greeting when a user successfully signs up.
2. **Password Reset Emails:** Provide secure links or OTPs for users to reset their passwords.
3. **Order Confirmation:** Send receipts or confirmations after a purchase is completed.
4. **Account Activity Alerts:** Notify users of important events like login from a new device.
5. **Newsletter and Promotions:** Distribute newsletters or promotional offers to subscribed users.





20.2 Using SendGrid

**twilio
SendGrid**

Products Why SendGrid Resources Developers Pricing Contact us Sign in Start for free

Twilio SendGrid

Everything you need to deliver emails at scale

- 148+ billion emails sent every month
- 82,000+ customers
- 100+ full-time email deliverability experts

[Start for free](#) [See plans & pricing →](#)

No credit card required | Get started quickly | Access to all Twilio products

Email API and SMTP service for developers
Trusted API for email delivery at scale.
[Learn more →](#)

Email campaigns for marketers
Tools to create engaging email campaigns.
[Learn more →](#)

Uber **Spotify** **airbnb** **yelp** **glassdoor** **instacart**



20.2 Using SendGrid

twilio
sendgrid

Products Why SendGrid Resources Developers Pricing Contact us Sign in Start for free

SendGrid pricing

Start for free. Then pay as you go.

- Sign up for a free trial - no credit card required
- Only pay for the volume and features you need
- Unlock volume discounts as you scale

[Start for free](#)

Email API
Integrate email into your app or website.

Marketing Campaigns
Design and send email marketing campaigns and automations.

Email API Plans

Pricing is determined by monthly email volume and features. Select your email volume below to see which plans are right for you.

How many emails would you like to send per month?

Volume Range	Plan
<3,000	Free
50,000	Starter
100,000	Starter
300,000	Starter
700,000	Pro
1,500,000	Pro
2,500,000	Pro
5,000,000+	Enterprise



20.2 Using SendGrid



API Keys

[Create API Key](#)

Get started creating API Keys

API keys help protect the sensitive areas of your SendGrid account (e.g. contacts and account settings). To control and limit access of API users, you can create multiple API keys, each with different permissions.



20.2 Using SendGrid



API Key Created

Please copy this key and save it somewhere safe.

For security reasons, we cannot show it to you again

Copied!

```
SG.0BsQdZJqSzmcAofUQqeZBQ.z-DbZaCPWs-zkAd3UfzS_YA5fqYH8UUUXg2FANf7mOyl
```

Done



20.2 Using SendGrid

Welcome

▼ How to start sending mail

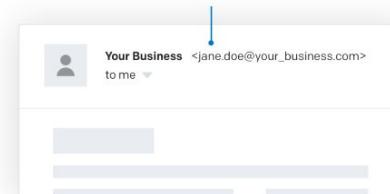
1 Send a single send

Your sender identity is the “from” email address your recipients see in their inbox. Once done, set up your integration for access to robust features.

[Create sender identity →](#)



[Learn more about sender identity](#)



2 Learn about email sending options

Complete the first step to unlock email sending options.



20.2 Using SendGrid

Sender Authentication /

Single Sender Verification (i)

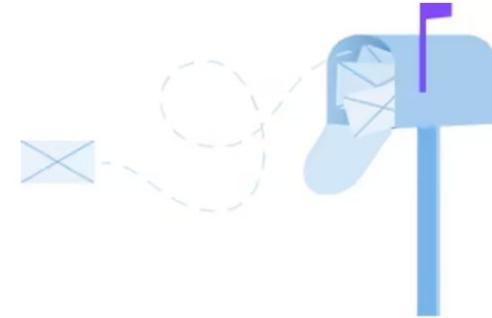
SENDERS

ADDRESS

Complete Coding

FROM contact@completecoding.in
REPLY contact@completecoding.in

Plot No 15
Sector 10A, Vasundha
Ghaziabad, 201012 IN



Sender has been created

To verify sender identity, check your inbox
at contact@completecoding.in.

Resend email

Close



20.2 Using SendGrid

Pro Teams Pricing Documentation

npm



Search packages

Search

Sign Up

Sign In

@sendgrid/mail ts

8.1.4 • Public • Published a month ago

Readme

Code

Beta

2 Dependencies

1,320 Dependents

52 Versions

build passing npm package 8.1.4

This package is part of a monorepo, please see [this README](#) for details.

Mail Service for the SendGrid v3 Web API

This is a dedicated service for interaction with the mail endpoint of the [SendGrid v3 API](#).

Installation

Install

`> npm i @sendgrid/mail`



Repository

github.com/sendgrid/sendgrid-nodejs

Homepage

sendgrid.com



20.2 Using SendGrid

```
prashantjain@Prashants-MacBook-Pro 13 mongoose % npm install @sendgrid/mail  
added 11 packages, and audited 281 packages in 3s  
  
49 packages are looking for funding  
  run `npm fund` for details  
  
1 high severity vulnerability  
  
To address all issues, run:  
  npm audit fix  
  
Run `npm audit` for details.
```



20.3 Sending Email

```
const sgMail = require('@sendgrid/mail');
const SENDGRID_API_KEY = 'SG.0BsQdZJqSzmcAofUQqeZBQ.
z-DbZaCPWs-zkAd3UfzS_YA5fqYH8UUUXg2FANf7m0yI';

bcrypt.hash(password, 12)
  .then(hashedPassword => {
    const user = new User({
      firstName: firstName,
      lastName: lastName,
      email: email,
      password: hashedPassword,
      userType: userType
    });
    return user.save();
  })
  .then(() => {
    // Send welcome email
    sgMail.setApiKey(SENDGRID_API_KEY);
    const msg = {
      to: email,
      from: 'contact@completecoding.in', // Change to your verified sender
      subject: 'Welcome to Complete Coding!!',
      html: `<h1>Welcome ${firstName}!</h1><p>Welcome to our family!We're excited to
      have you on board.</p><p>Best regards,<br>The Team</p>`
    };
    return sgMail.send(msg);
  })
  .then(() => {
    res.redirect('/login');
  })
  .catch(error => {
    console.error(error);
  })
  .finally(() => {
    res.end();
  })
}
```



20.3 Sending Email

Welcome to Complete Coding! ➤ Inbox ×



contact@completecoding.in via sendgrid.net
to me ▾

Welcome prashant!

Welcome to our family! We're excited to have you on board.

Best regards,
The Team



20.4 Forgot Password Wireframe

1. Define a new `forgot.ejs` file using the `login.ejs` file with `email` field and a `button` to Reset Password.
2. Add a link to login page for forgot password to `/forgot-password`
3. Add entry in the `auth router and controller` to show the page.
4. Submitting the form should go to `POST /forgot-password` which will find the user by `email-id` and redirect him to a new page `/reset-password` passing `email-id` as a parameter.



20.4 Forgot Password Wireframe

1.

```
<form action="/forgot-password" method="POST" class="w-full max-w-md">

  <% if (typeof errorMessage !== 'undefined' && errorMessage) { %>
    <div class="bg-red-100 border border-red-400 text-red-700 px-4 py-3 rounded relat
      mb-4" role="alert">
      <span class="block sm:inline"><%= errorMessage %></span>
    </div>
  <% } %>

  <input
    type="email"
    name="email"
    value="<%= typeof oldEmail !== 'undefined' ? oldEmail : '' %>"
    placeholder="Enter your email"
    class="w-full px-4 py-2 mb-4 border border-gray-300 rounded-lg focus:outline-none
      focus:border-blue-500"
  />

  <div class="flex justify-center">
    <input
      type="submit"
      value="Reset Password"
      class="bg-blue-500 hover:bg-blue-600 text-white font-semibold py-2 px-4 rounded
        transition duration-300 ease-in-out transform hover:scale-105 cursor-pointer"
    >
  </div>
</form>
```



20.4 Forgot Password Wireframe

2.

```
<div class="flex justify-between items-center mb-4">
  <a href="/forgot-password" class="text-blue-500 hover:text-blue-600
  text-sm">Forgot Password?</a>
</div>
```

3.

```
exports.getForgotPassword = (req, res, next) => {
  res.render("auth/forgot", { pageTitle: "Forgot Password", isLoggedIn: false,
  user: req.session.user });
};
```

```
authRouter.get("/forgot-password", authController.getForgotPassword);
authRouter.post("/forgot-password", authController.postForgotPassword);
```



20.4 Forgot Password Wireframe

4.

```
exports.postForgotPassword = async (req, res, next) => {
  const { email } = req.body;
  try {
    const user = await User.findOne({ email: email });
    if (!user) {
      return res.render('auth/forgot', {
        pageTitle: 'Forgot Password',
        isLoggedIn: false,
        user: req.session.user,
        errorMessage: 'No account found with this email'
      });
    }
    res.redirect(`/reset-password?email=${email}`);
  } catch (err) {
    console.log('Error in forgot password:', err);
    return res.render('auth/forgot', {
      pageTitle: 'Forgot Password',
      isLoggedIn: false,
      user: req.session.user,
      errorMessage: err.message
    });
  }
};
```



20.5 Sending OTP

1. Add **OTP** and **OTP-Expiry** fields to User model.
2. Generate a random 6 digit OTP, to be sent in the password reset email.
3. In the **user** object add value of both fields, save the user.
4. Send the same **OTP** to the user in an **email** along with a link to **/reset-password** page.



20.5 Sending OTP

```
1. favouriteHomes: [{}  
  |   type: mongoose.Schema.Types.ObjectId,  
  |   ref: 'Home'  
  |],  
  otp: {}  
  |   type: String,  
  |   required: false  
  |},  
  otpExpiry: {}  
  |   type: Date,  
  |   required: false  
  |}
```



20.5 Sending OTP

```
exports.postForgotPassword = async (req, res, next) => {
  const { email } = req.body;
  try {
    const user = await User.findOne({ email });
    if (!user) {
      throw new Error('No account found with this email');
    }

    const otp = Math.floor(100000 + Math.random() * 900000).toString();
    user.otp = otp;
    user.otpExpiry = Date.now() + 5 * 60 * 1000;
    await user.save();

    sgMail.setApiKey(SENDGRID_API_KEY);
    const msg = {
      to: email,
      from: 'contact@completecoding.in',
      subject: 'Reset Password OTP',
      html: `<h1>Reset Password OTP</h1><p>Your OTP is ${otp}. It is valid for 5 minutes.</p>`;
    };
    await sgMail.send(msg);

    res.redirect(`/reset-password?email=${email}`);
  } catch (err) {
    console.log('Error in forgot password:', err);
    return res.render('auth/forgot', {
      pageTitle: 'Forgot Password',
      isLoggedIn: false,
      user: req.session.user,
      oldEmail: email,
      errorMessage: err.message
    });
  }
};
```

2,3.

4.



20.6 Reset Password

1. Define a new `reset-password.ejs` file using the `signup.ejs` file that shows `email`, `otp`, `password`, `confirm_password` and a button to Reset Password.
2. Define a GET `/reset-password` route and a controller entry in auth files
3. Submitting the form should go to POST `/reset-password` which will log the request body.
4. Add same validations on `password` and `confirm_password`.
5. In the POST controller, take out the email-id and find the user. Check if user is found and the expiration date is greater than now, and the OTP matches, reset the password. And redirect to login page.



20.6 Reset Password

```
1. <input type="email" name="email" placeholder="Enter your Email" value="<%=\n  !== 'undefined' ? email : '' %>" readonly class="w-full px-4 py-2 mb-4 border\n  border-gray-300 rounded-lg bg-gray-100"/>\n\n<input type="text" name="otp" placeholder="Enter OTP" class="w-full px-4 py-2 mb-4 border\n  border-gray-300 rounded-lg focus:outline-none border-blue-500"/>\n\n<input type="password" name="password" placeholder="New Password" class="w-full px-4 py-2\n  mb-4 border border-gray-300 rounded-lg focus:outline-none border-blue-500"/>\n\n<input type="password" name="confirm_password" placeholder="Confirm New Password"\n  class="w-full px-4 py-2 mb-4 border border-gray-300 rounded-lg focus:outline-none\n  border-blue-500"/>\n\n<div class="flex justify-center">\n  <input type="submit" value="Update Password" class="bg-blue-500 hover:bg-blue-600\n    text-white font-semibold py-2 px-4 rounded-lg transition duration-300 ease-in-out\n    transform hover:scale-105 cursor-pointer">\n</div>
```



20.6 Reset Password

2.

```
authRouter.get("/reset-password", authController.getResetPassword);
authRouter.post("/reset-password", authController.postResetPassword);
```

```
exports.getResetPassword = (req, res, next) => {
  const { email } = req.query;
  res.render("auth/reset_password", { pageTitle: "Reset Password", isLoggedIn: false,
    user: req.session.user, email: email });
};
```

3.

```
exports.postResetPassword = async (req, res, next) => {
  const { email, otp, password } = req.body;
  console.log(email, otp, password);
  return res.redirect('/login');
};
```



20.6 Reset Password

4.

```
exports.postResetPassword = [
  // Password validation
  check('password')
    .isLength({ min: 8 })
    .withMessage('Password must be at least 8 characters long')
    .matches(/^[a-z]*/
    .withMessage('Password must contain at least one lowercase letter')
    .matches(/^[A-Z]*/
    .withMessage('Password must contain at least one uppercase letter')
    .matches(/[^@#$%^&*(),.?":{}|<>]*/
    .withMessage('Password must contain at least one special character')
    .trim(),

  // Confirm password validation
  check('confirm_password')
    .trim()
    .custom((value, { req }) => {
      if (value !== req.body.password) {
        throw new Error('Passwords do not match');
      }
      return true;
    }),
  async (req, res, next) => {
```



20.6 Reset Password

5.

```
async (req, res, next) => {
  const { email, otp, password } = req.body;
  try {
    const user = await User.findOne({ email: email });

    if (!user) {
      throw new Error('No user found with this email');
    } else if (user.otpExpiry < Date.now()) {
      throw new Error('OTP has expired. Please request a new one.');
    } else if (user.otp !== otp) {
      throw new Error('Invalid OTP');
    }

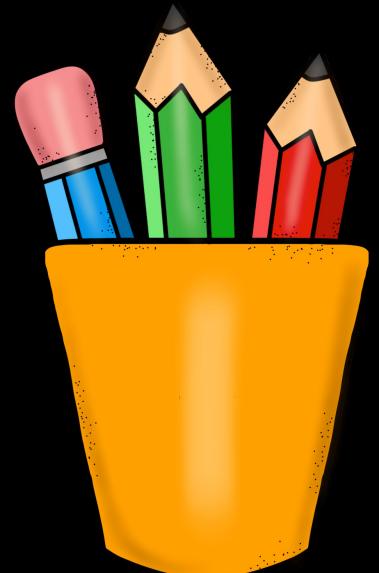
    const hashedPassword = await bcrypt.hash(password, 12);
    user.password = hashedPassword;
    user.otp = undefined;
    user.otpExpiry = undefined;
    await user.save();

    return res.redirect('/login');
  } catch (err) {
    console.log('Error in reset password:', err);
    return res.render('auth/reset_password', {
      pageTitle: 'Reset Password',
      isLoggedIn: false,
      user: req.session.user,
      email: email,
      errorMessage: err.message
    });
  }
}
```



Revision

1. Why to send Email
2. Using SendGrid
3. Sending Email
4. Forgot Password Wireframe
5. Sending OTP
6. Reset Password





Practise Milestone

Take your airbnb forward:

1. Takeout the validations in a separate file.
2. Define a partial for showing errors and add it to all files where it is used.





Practise Milestone (Solution)

1.

```
const { check } = require('express-validator');

exports.firstNameValidator = check('firstName')
    .notEmpty()
    .withMessage("First name is mandatory")
    .trim()
    .isLength({min: 2})
    .withMessage('First Name should be minium 2 chars')
    .matches(/^[a-zA-Z\s]+$/)
    .withMessage('First Name should only contain english aplhabets');

exports.lastNameValidator = check('lastName')
    .trim()
    .matches(/^[a-zA-Z\s]*$/)
    .withMessage('Last Name should only contain english aplhabets');

exports.emailValidator = check('email')
    .isEmail()
    .withMessage('Please enter a valid email')
    .normalizeEmail();

exports.passwordValidator = check('password')
    .trim()
```





Practise Milestone (Solution)

1.

```
const {
  firstNameValidator,
  lastNameValidator,
  emailValidator,
  passwordValidator,
  confirmPasswordValidator,
  userTypeValidator,
  termsAcceptedValidator
} = require('./validations');

exports.postSignup = [
  firstNameValidator,
  lastNameValidator,
  emailValidator,
  passwordValidator,
  confirmPasswordValidator,
  userTypeValidator,
  termsAcceptedValidator,

  // Final handler middleware
  (req, res, next) => {
    const { firstName, lastName, email, password, userType } = req.body;
    const errors = validationResult(req);
```

```
  exports.postResetPassword = [
    passwordValidator,
    confirmPasswordValidator,

    async (req, res, next) => {
      const { email, otp, password } = req.body;
      try {
```



Practise Milestone (Solution)

2.

```
<% if (typeof errorMessages !== 'undefined' && errorMessages && errorMessages.length > 0) { %>
  <ul class="■bg-red-100 border ■border-red-400 ■text-red-700 px-4 py-3 rounded mb-4">
    <% errorMessages.forEach(errorMessage => { %>
      <li class="flex items-center">
        <svg class="w-4 h-4 mr-2" fill="currentColor" viewBox="0 0 20 20">
          <path fill-rule="evenodd" d="M10 18a8 8 0 100-16 8 8 0 00 16zM8.707 7.293a1 1 0 00-1.414 1.414L8.586 10l-1.293 1.293a1 1 0 101.414 1.414L10 11.414L1.293 1.293a1 1 0 001.414-1.414L11.414 10l1.293-1.293a1 1 0 00-1.414-1.414L10 8.586 8.707 7.293z" clip-rule="evenodd"/>
        </svg>
        <%= errorMessage %>
      </li>
    <% }) %>
  </ul>
<% } %>
```

```
<form action="/signup" method="POST" class="w-full max-w-md">

  <%- include('../partials/error-messages') %>

  <input
    type="text"
    name="firstName"
```

