DHIRUBHAI AMBANI INSTITUTE OF INFORMATION AND COMMUNICATION TECHNOLOGY

IT314 – SOFTWARE ENGINEERING

GROUP - 13

JOB MATCHING PLATFORM
- CAREERXPERT -

PROFESSOR – SAURABH TIWARI

MENTOR – SARTHAK

GROUP MEMBERS :

202201231 – HARMIT KHIMANI
202201192 – SMIT SHAH
202201193 – RUDRA PATEL
202201197 – VIVEK VAGHELA
202201198 – DEVANSH MODI
202201203 – AJAY CHOVATIYA
202201205 – DARSHAN GAMI
202201234 – ANSHU DHANKECHA
202201252 – NISHANT ITALIYA
202201258 – MARMIK VASAVA

Unit Testing with Node.js

In our project, we focused on ensuring that the Node.js server was robust and reliable through comprehensive unit testing. To achieve this, we leveraged several testing libraries and frameworks, including **Mocha**, **Chai**, **Supertest**, and **nyc** for coverage. Below is a breakdown of the libraries and tools we used in our testing approach:

a) Mocha (Test Framework)

Mocha is a flexible and feature-rich JavaScript test framework that allows us to structure our tests efficiently. It supports asynchronous testing, which is crucial for testing Node.js applications where many operations (such as database queries or HTTP requests) are asynchronous. Mocha also provides a clear test lifecycle, including hooks such as **before()**, **beforeEach()**, **after()**, and **afterEach()**, which makes setting up and tearing down tests easy and organized.

Key Features of Mocha:

- ➢ Asynchronous test support: Mocha handles asynchronous operations gracefully, making it an ideal choice for Node.js testing.
- ➢ Clear test lifecycle: Hooks for setup and teardown (**before()**, **after()**, etc.).
- ➢ Extensibility: Can be integrated with assertion libraries (like Chai) and other tools for enhanced functionality.

b) Chai (Assertion Library)

Chai is an expressive assertion library that works seamlessly with Mocha. It allows us to perform more readable and declarative tests. Chai supports different assertion styles, such as **expect()**, **should()**, and **assert()**, which help make the tests more intuitive and clear in their intent.

Key Features of Chai:

- ➢ Multiple assertion styles: Chai supports **expect()**, **should()**, and **assert()**, allowing us to choose the syntax that best fits the style of testing.
- ➢ Readable and intuitive syntax: Chai helps make tests easy to read and understand, reducing the cognitive load for developers.
- ➢ Integration with Mocha: Chai integrates seamlessly with Mocha to provide a complete testing environment.

c) Supertest (HTTP Request Testing)

Supertest is a tool that helps us simulate HTTP requests and assert the responses. This is particularly useful for testing the endpoints of our Node.js server. Supertest allows us to send **GET, POST, PUT, DELETE**, etc., requests to the server and assert that the correct responses are returned based on the expected behavior.

Key Features of Supertest:

- ➢ Efficient HTTP request simulation: Supertest makes it easy to simulate a wide variety of HTTP requests to test REST APIs.
- ➢ Response assertion: We can assert that the server's responses match expectations, including status codes, headers, and body content.
- ➢ Seamless integration with Mocha: Supertest can be used in conjunction with Mocha to test the functionality of endpoints, ensuring our server handles requests as expected.

d) nyc (Code Coverage Tool)

While Jest provides built-in coverage analysis, we used **nyc** (a command-line tool for Istanbul) for generating code coverage reports. **nyc** works by instrumenting the code to track how much of it is covered during testing, providing insights into untested portions of the application. It generates detailed coverage reports, which help us ensure that our unit tests are comprehensive and that critical parts of the code are thoroughly tested.

Key Features of nyc:

- ➢ Code coverage tracking: nyc provides an easy way to track code coverage during test execution.
- ➢ Reports in various formats: It generates reports in formats like text, HTML, helping us analyze the results in different ways.
- ➢ Works well with Mocha: We integrated nyc with Mocha to generate coverage reports while running tests.

Testing Workflow

1. **Setup**: We use Mocha as the test framework and Chai for assertions. Tests are written to cover both unit and integration scenarios, ensuring the server's logic and HTTP endpoints are thoroughly tested.

2. **HTTP Testing**: With Supertest, we simulate HTTP requests to our Node.js server to verify that it responds as expected. This includes checking status codes, response body, and headers.

3. **Code Coverage**: nyc is used to track and report code coverage during the testing process. By running the tests with nyc, we get detailed reports on which parts of the code were executed during the tests and which parts were not.

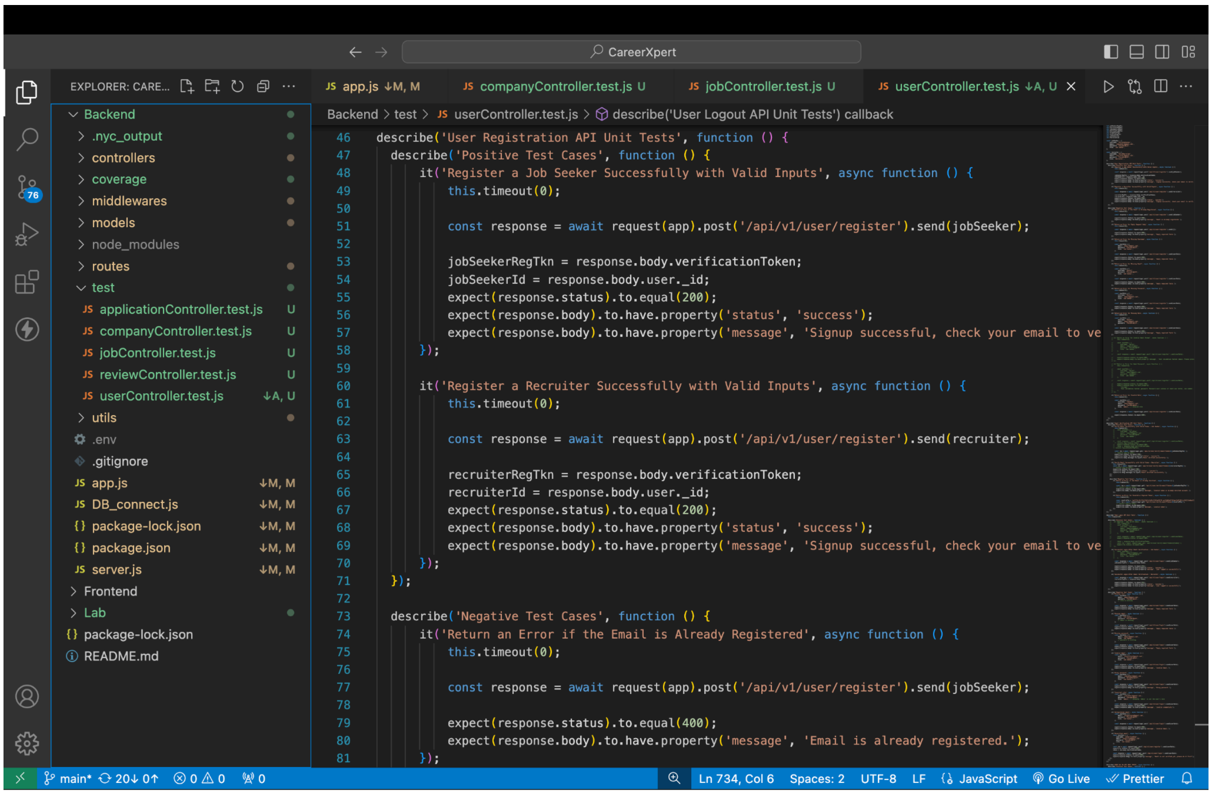
4. **Execution**: Mocha runs the tests, Supertest verifies the correctness of HTTP responses, and nyc generates the coverage report, all working together to provide a robust testing environment.

5. **Review and Refactor**: After the tests have run, we review the coverage reports to identify any gaps in test coverage and ensure that all critical paths in the application are tested. If necessary, we add additional tests to improve coverage and refactor the code as needed.

Conclusion

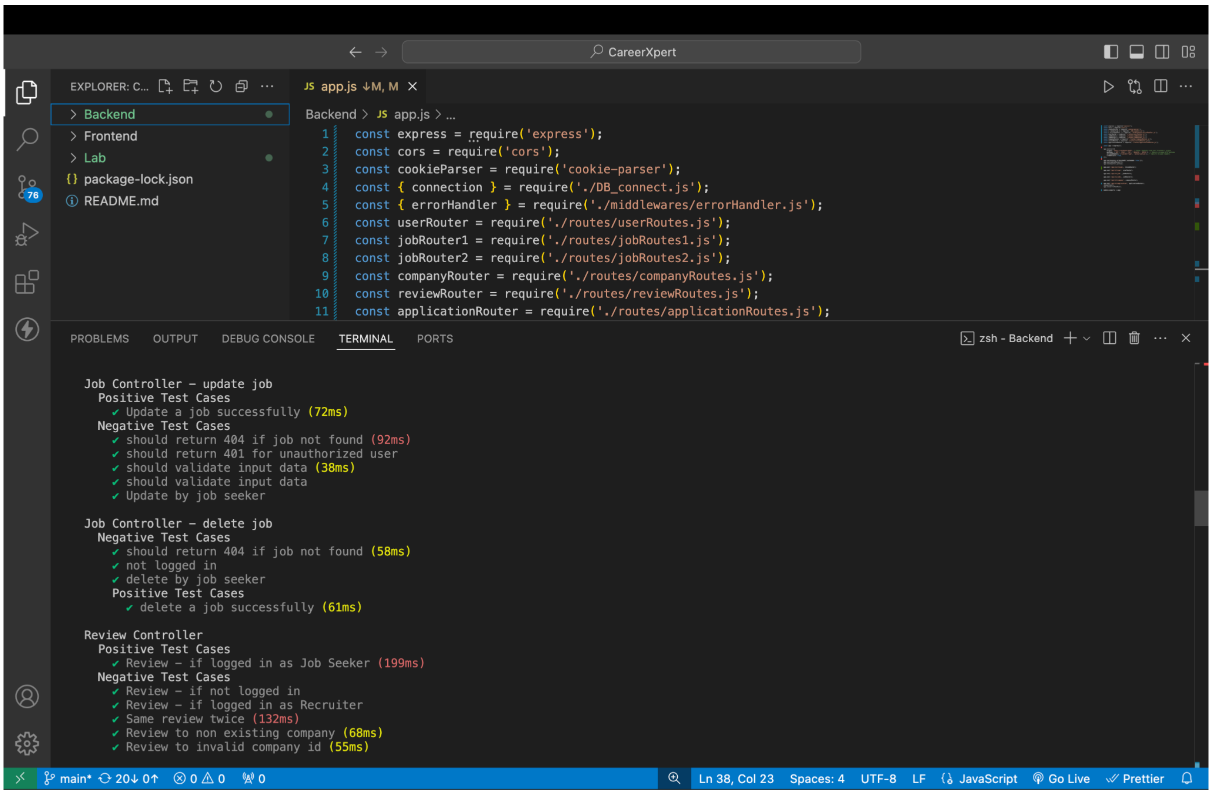
Using Mocha, Chai, Supertest, and nyc, we have created a powerful testing environment for our Node.js application. Mocha provides a flexible test framework, Chai enables readable assertions, Supertest allows us to test HTTP requests, and nyc ensures we maintain comprehensive test coverage. This combination of tools helps us ensure that our application is robust, reliable, and maintainable.

➢ TestCases :



```
describe('User Registration API Unit Tests', function () {
  describe('Positive Test Cases', function () {
    it('Register a Job Seeker Successfully with Valid Inputs', async function () {
      this.timeout(0);

      const response = await request(app).post('/api/v1/user/register').send(jobSeeker);

      jobSeekerRegTkn = response.body.verificationToken;
      jobSeekerId = response.body.user._id;
      expect(response.status).to.equal(200);
      expect(response.body).to.have.property('status', 'success');
      expect(response.body).to.have.property('message', 'Signup successful, check your email to ve
    });

    it('Register a Recruiter Successfully with Valid Inputs', async function () {
      this.timeout(0);

      const response = await request(app).post('/api/v1/user/register').send(recruiter);

      recruiterRegTkn = response.body.verificationToken;
      recruiterId = response.body.user._id;
      expect(response.status).to.equal(200);
      expect(response.body).to.have.property('status', 'success');
      expect(response.body).to.have.property('message', 'Signup successful, check your email to ve
    });
  });

  describe('Negative Test Cases', function () {
    it('Return an Error if the Email is Already Registered', async function () {
      this.timeout(0);

      const response = await request(app).post('/api/v1/user/register').send(jobSeeker);

      expect(response.status).to.equal(400);
      expect(response.body).to.have.property('message', 'Email is already registered.');
    });
```

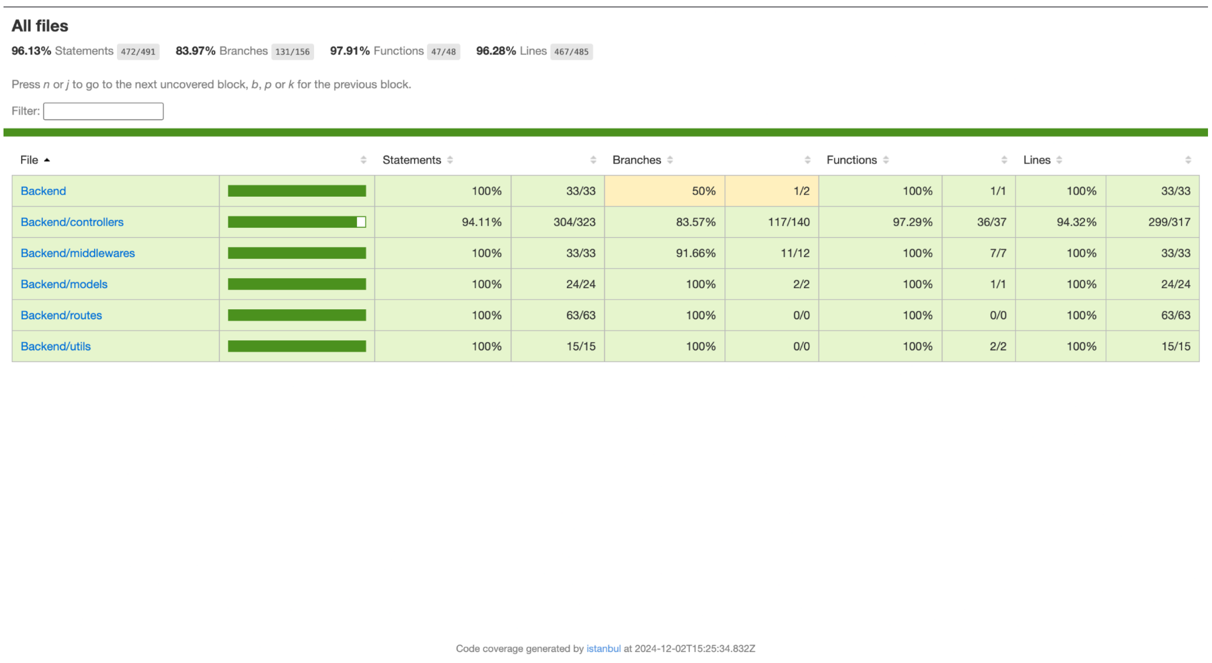
➢ Accepted TestCases :



```
const express = require('express');
const cors = require('cors');
const cookieParser = require('cookie-parser');
const { connection } = require('./DB_connect.js');
const { errorHandler } = require('./middlewares/errorHandler.js');
const userRouter = require('./routes/userRoutes.js');
const jobRouter1 = require('./routes/jobRoutes1.js');
const jobRouter2 = require('./routes/jobRoutes2.js');
const companyRouter = require('./routes/companyRoutes.js');
const reviewRouter = require('./routes/reviewRoutes.js');
const applicationRouter = require('./routes/applicationRoutes.js');
```

```
  Job Controller - update job
    Positive Test Cases
      ✔ Update a job successfully (72ms)
    Negative Test Cases
      ✔ should return 404 if job not found (92ms)
      ✔ should return 401 for unauthorized user
      ✔ should validate input data (38ms)
      ✔ should validate input data
      ✔ Update by job seeker

  Job Controller - delete job
    Negative Test Cases
      ✔ should return 404 if job not found (58ms)
      ✔ not logged in
      ✔ delete by job seeker
      Positive Test Cases
        ✔ delete a job successfully (61ms)

  Review Controller
    Positive Test Cases
      ✔ Review - if logged in as Job Seeker (199ms)
    Negative Test Cases
      ✔ Review - if not logged in
      ✔ Review - if logged in as Recruiter
      ✔ Same review twice (132ms)
      ✔ Review to non existing company (68ms)
      ✔ Review to invalid company id (55ms)
```

➢ All Files :



All files

96.13% Statements 472/491 **83.97%** Branches 131/156 **97.91%** Functions 47/48 **96.28%** Lines 467/485

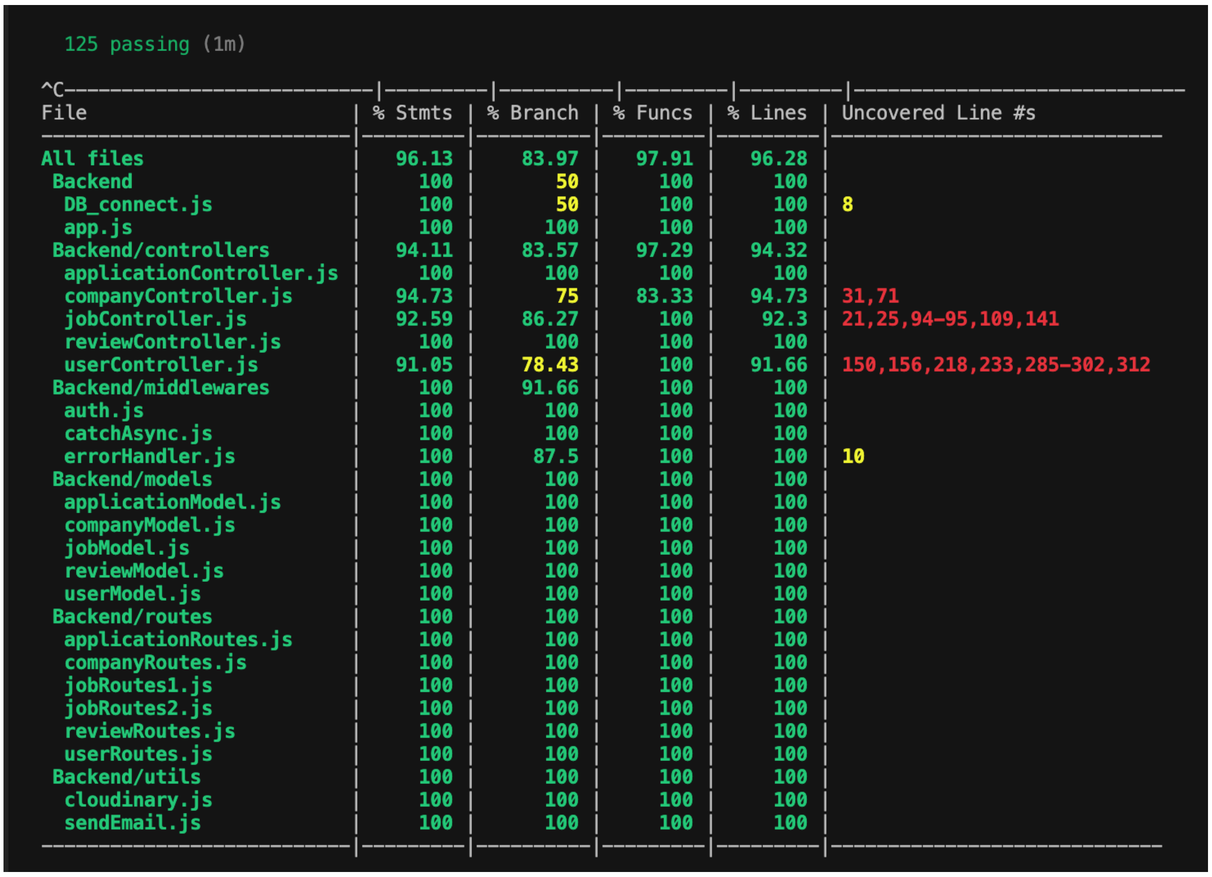
Press *n* or *j* to go to the next uncovered block, *b*, *p* or *k* for the previous block.

Filter:

File ▲		Statements		Branches		Functions		Lines	
Backend		100%	33/33	50%	1/2	100%	1/1	100%	33/33
Backend/controllers		94.11%	304/323	83.57%	117/140	97.29%	36/37	94.32%	299/317
Backend/middlewares		100%	33/33	91.66%	11/12	100%	7/7	100%	33/33
Backend/models		100%	24/24	100%	2/2	100%	1/1	100%	24/24
Backend/routes		100%	63/63	100%	0/0	100%	0/0	100%	63/63
Backend/utils		100%	15/15	100%	0/0	100%	2/2	100%	15/15

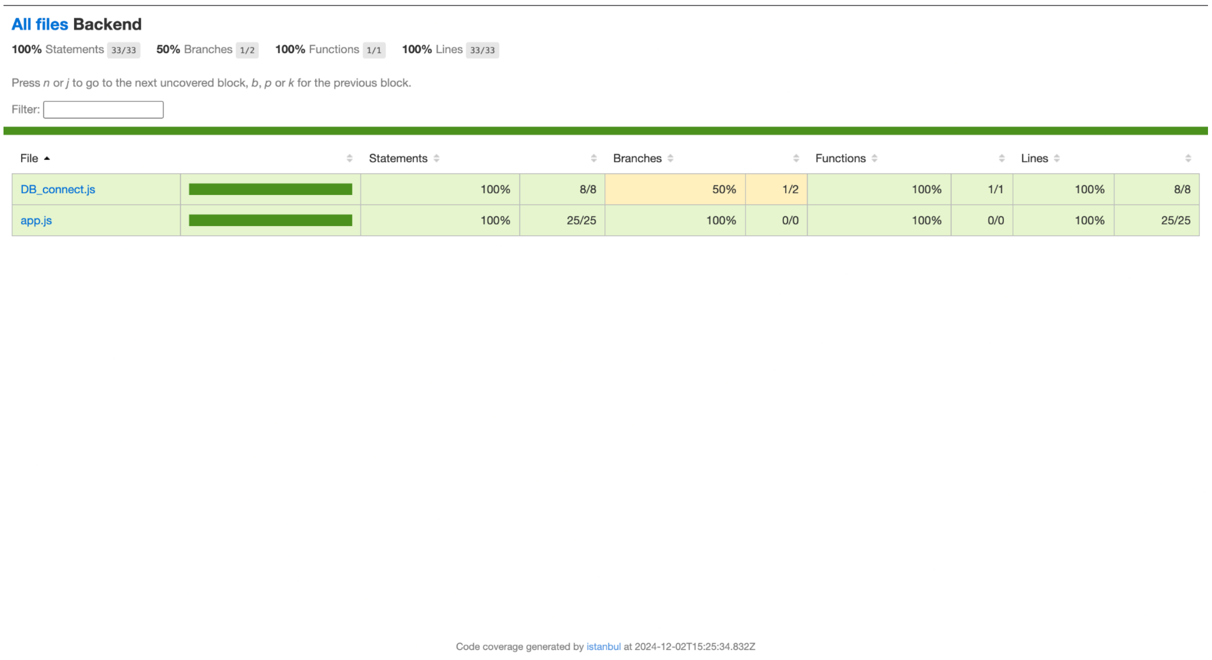
Code coverage generated by istanbul at 2024-12-02T15:25:34.832Z

➢ All Files TxT :



```
  125 passing (1m)

^C----------------------------|----------|----------|----------|----------|-----------------------------
File                          | % Stmts  | % Branch | % Funcs  | % Lines  | Uncovered Line #s
------------------------------|----------|----------|----------|----------|-----------------------------
All files                     |   96.13  |   83.97  |   97.91  |   96.28  |
 Backend                      |     100  |      50  |     100  |     100  |
  DB_connect.js               |     100  |      50  |     100  |     100  | 8
  app.js                      |     100  |     100  |     100  |     100  |
 Backend/controllers          |   94.11  |   83.57  |   97.29  |   94.32  |
  applicationController.js    |     100  |     100  |     100  |     100  |
  companyController.js        |   94.73  |      75  |   83.33  |   94.73  | 31,71
  jobController.js            |   92.59  |   86.27  |     100  |    92.3  | 21,25,94-95,109,141
  reviewController.js         |     100  |     100  |     100  |     100  |
  userController.js           |   91.05  |   78.43  |     100  |   91.66  | 150,156,218,233,285-302,312
 Backend/middlewares          |     100  |   91.66  |     100  |     100  |
  auth.js                     |     100  |     100  |     100  |     100  |
  catchAsync.js               |     100  |     100  |     100  |     100  |
  errorHandler.js             |     100  |    87.5  |     100  |     100  | 10
 Backend/models               |     100  |     100  |     100  |     100  |
  applicationModel.js         |     100  |     100  |     100  |     100  |
  companyModel.js             |     100  |     100  |     100  |     100  |
  jobModel.js                 |     100  |     100  |     100  |     100  |
  reviewModel.js              |     100  |     100  |     100  |     100  |
  userModel.js                |     100  |     100  |     100  |     100  |
 Backend/routes               |     100  |     100  |     100  |     100  |
  applicationRoutes.js        |     100  |     100  |     100  |     100  |
  companyRoutes.js            |     100  |     100  |     100  |     100  |
  jobRoutes1.js               |     100  |     100  |     100  |     100  |
  jobRoutes2.js               |     100  |     100  |     100  |     100  |
  reviewRoutes.js             |     100  |     100  |     100  |     100  |
  userRoutes.js               |     100  |     100  |     100  |     100  |
 Backend/utils                |     100  |     100  |     100  |     100  |
  cloudinary.js               |     100  |     100  |     100  |     100  |
  sendEmail.js                |     100  |     100  |     100  |     100  |
------------------------------|----------|----------|----------|----------|-----------------------------
```

➢ All Files - Backend :



All files **Backend**

100% Statements 33/33 **50%** Branches 1/2 **100%** Functions 1/1 **100%** Lines 33/33

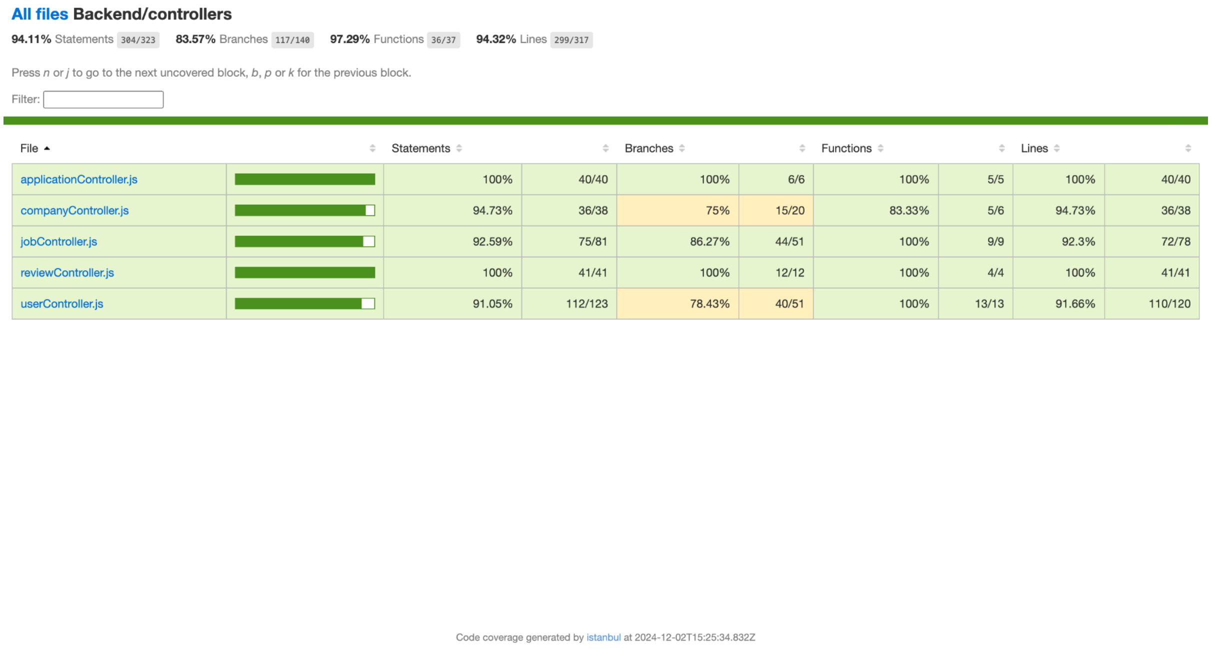
Press *n* or *j* to go to the next uncovered block, *b*, *p* or *k* for the previous block.

Filter:

File ▲		Statements		Branches		Functions		Lines	
DB_connect.js		100%	8/8	50%	1/2	100%	1/1	100%	8/8
app.js		100%	25/25	100%	0/0	100%	0/0	100%	25/25

Code coverage generated by istanbul at 2024-12-02T15:25:34.832Z

➢ All Files - Backend - Controllers :



All files **Backend/controllers**

94.11% Statements 304/323 **83.57%** Branches 117/140 **97.29%** Functions 36/37 **94.32%** Lines 299/317

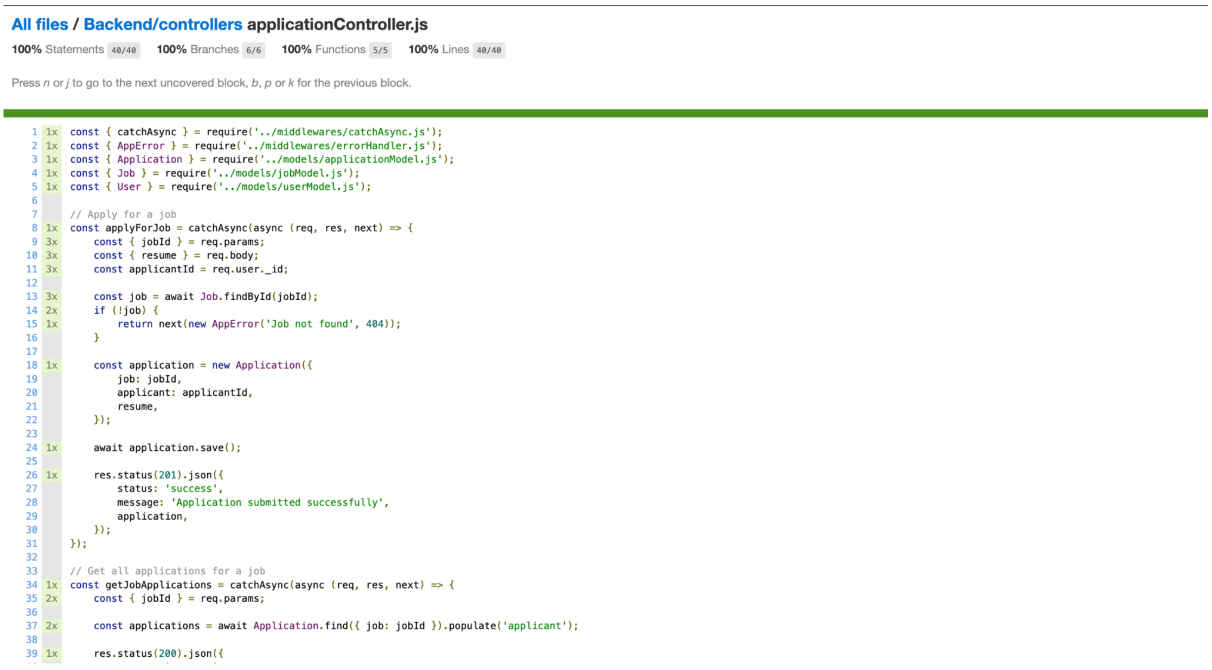
Press *n* or *j* to go to the next uncovered block, *b*, *p* or *k* for the previous block.

Filter:

File ▲		Statements		Branches		Functions		Lines	
applicationController.js		100%	40/40	100%	6/6	100%	5/5	100%	40/40
companyController.js		94.73%	36/38	75%	15/20	83.33%	5/6	94.73%	36/38
jobController.js		92.59%	75/81	86.27%	44/51	100%	9/9	92.3%	72/78
reviewController.js		100%	41/41	100%	12/12	100%	4/4	100%	41/41
userController.js		91.05%	112/123	78.43%	40/51	100%	13/13	91.66%	110/120

Code coverage generated by istanbul at 2024-12-02T15:25:34.832Z

➢ All Files - Backend - controllers - applicationController :



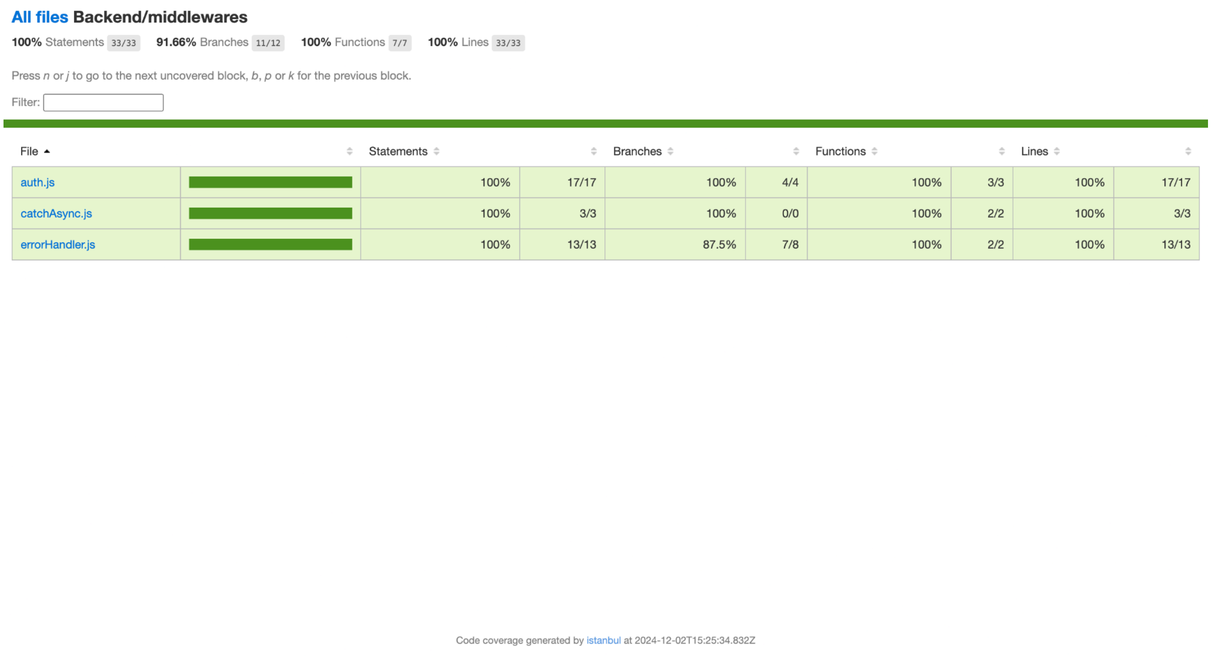
All files / Backend/controllers applicationController.js

100% Statements 40/40 **100%** Branches 6/6 **100%** Functions 5/5 **100%** Lines 40/40

Press *n* or *j* to go to the next uncovered block, *b*, *p* or *k* for the previous block.

```
1x  const { catchAsync } = require('../middlewares/catchAsync.js');
1x  const { AppError } = require('../middlewares/errorHandler.js');
1x  const { Application } = require('../models/applicationModel.js');
1x  const { Job } = require('../models/jobModel.js');
1x  const { User } = require('../models/userModel.js');

    // Apply for a job
1x  const applyForJob = catchAsync(async (req, res, next) => {
3x      const { jobId } = req.params;
3x      const { resume } = req.body;
3x      const applicantId = req.user._id;

3x      const job = await Job.findById(jobId);
2x      if (!job) {
1x          return next(new AppError('Job not found', 404));
        }

1x      const application = new Application({
            job: jobId,
            applicant: applicantId,
            resume,
        });

1x      await application.save();

1x      res.status(201).json({
            status: 'success',
            message: 'Application submitted successfully',
            application,
        });
    });

    // Get all applications for a job
1x  const getJobApplications = catchAsync(async (req, res, next) => {
2x      const { jobId } = req.params;

2x      const applications = await Application.find({ job: jobId }).populate('applicant');

1x      res.status(200).json({
```

➢ All Files - Backend - middlewares :



All files Backend/middlewares

100% Statements 33/33 **91.66%** Branches 11/12 **100%** Functions 7/7 **100%** Lines 33/33

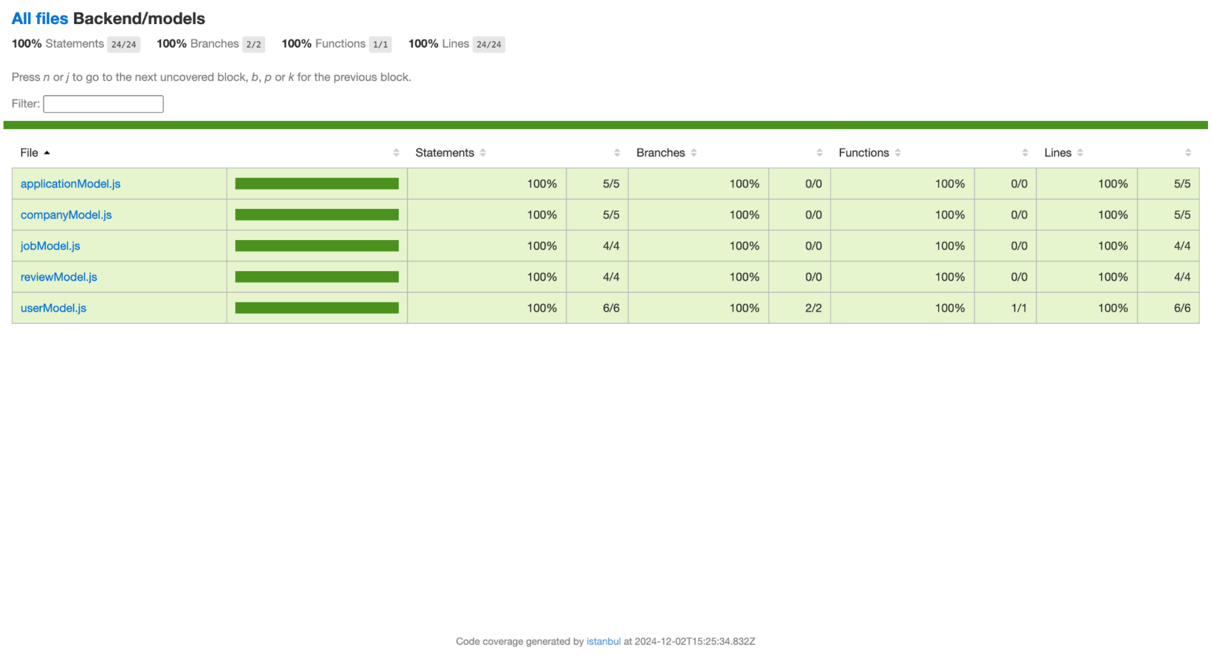
Press *n* or *j* to go to the next uncovered block, *b*, *p* or *k* for the previous block.

Filter:

File		Statements		Branches		Functions		Lines	
auth.js	🟩	100%	17/17	100%	4/4	100%	3/3	100%	17/17
catchAsync.js	🟩	100%	3/3	100%	0/0	100%	2/2	100%	3/3
errorHandler.js	🟩	100%	13/13	87.5%	7/8	100%	2/2	100%	13/13

Code coverage generated by istanbul at 2024-12-02T15:25:34.832Z

➢ All Files - Backend - models :



All files **Backend/models**

100% Statements 24/24 **100%** Branches 2/2 **100%** Functions 1/1 **100%** Lines 24/24

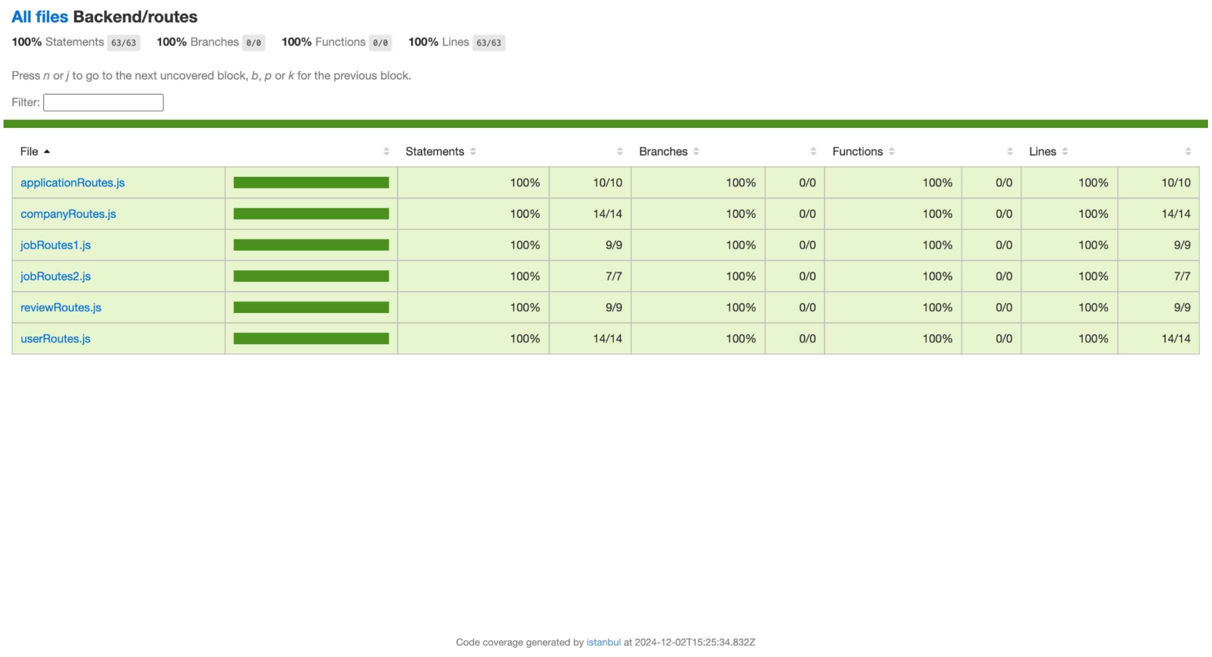
Press *n* or *j* to go to the next uncovered block, *b*, *p* or *k* for the previous block.

Filter:

File ▲		Statements		Branches		Functions		Lines	
applicationModel.js	🟩	100%	5/5	100%	0/0	100%	0/0	100%	5/5
companyModel.js	🟩	100%	5/5	100%	0/0	100%	0/0	100%	5/5
jobModel.js	🟩	100%	4/4	100%	0/0	100%	0/0	100%	4/4
reviewModel.js	🟩	100%	4/4	100%	0/0	100%	0/0	100%	4/4
userModel.js	🟩	100%	6/6	100%	2/2	100%	1/1	100%	6/6

Code coverage generated by istanbul at 2024-12-02T15:25:34.832Z

➢ All Files – Backend – Routes :



All files **Backend/routes**

100% Statements 63/63 **100%** Branches 0/0 **100%** Functions 0/0 **100%** Lines 63/63

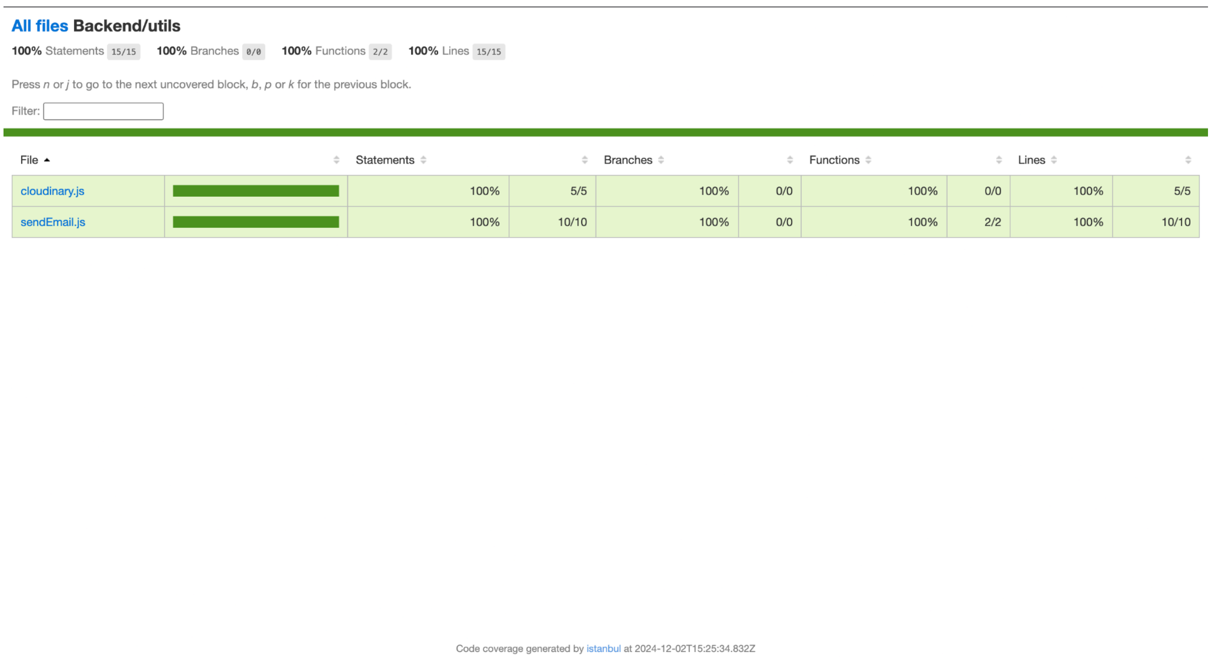
Press *n* or *j* to go to the next uncovered block, *b*, *p* or *k* for the previous block.

Filter:

File ▲		Statements		Branches		Functions		Lines	
applicationRoutes.js	🟩	100%	10/10	100%	0/0	100%	0/0	100%	10/10
companyRoutes.js	🟩	100%	14/14	100%	0/0	100%	0/0	100%	14/14
jobRoutes1.js	🟩	100%	9/9	100%	0/0	100%	0/0	100%	9/9
jobRoutes2.js	🟩	100%	7/7	100%	0/0	100%	0/0	100%	7/7
reviewRoutes.js	🟩	100%	9/9	100%	0/0	100%	0/0	100%	9/9
userRoutes.js	🟩	100%	14/14	100%	0/0	100%	0/0	100%	14/14

Code coverage generated by istanbul at 2024-12-02T15:25:34.832Z

➢ All Files – Backend – utils :



All files **Backend/utils**

100% Statements 15/15 **100%** Branches 0/0 **100%** Functions 2/2 **100%** Lines 15/15

Press *n* or *j* to go to the next uncovered block, *b*, *p* or *k* for the previous block.

Filter:

File		Statements		Branches		Functions		Lines	
cloudinary.js	🟩	100%	5/5	100%	0/0	100%	0/0	100%	5/5
sendEmail.js	🟩	100%	10/10	100%	0/0	100%	2/2	100%	10/10

Code coverage generated by istanbul at 2024-12-02T15:25:34.832Z