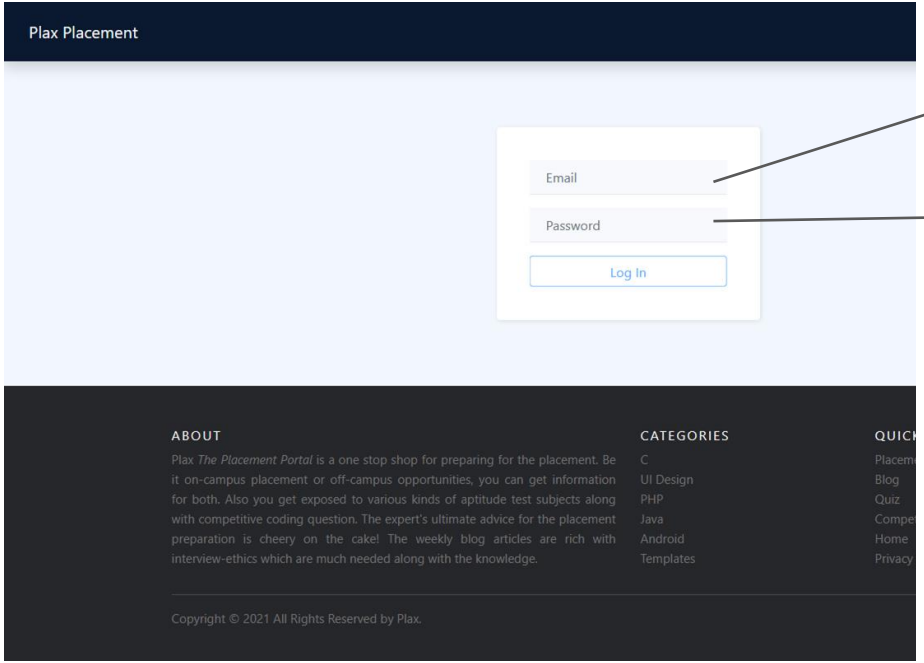
University Placement Portal

Code snippet with brief

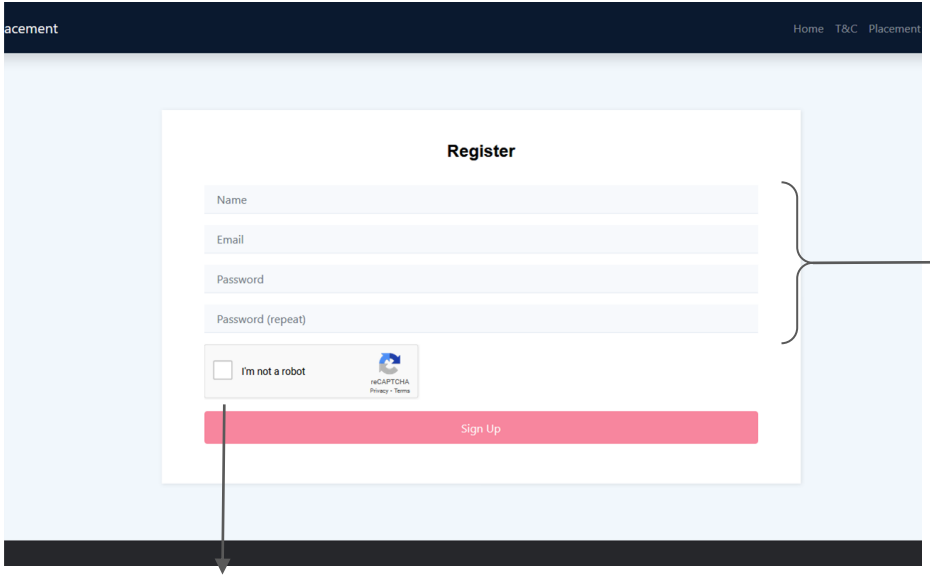
Sahil Bhuva

Devanshee Vankani




```
userRoute.post("/login", (req, res, next) => {
   User.find({email: req.body.email})
       .exec()
       .then(user => {
           if (user.length < 1) {
               return res.status(401).json({
                   message: "Auth failed"
               });
           }
           bcrypt.compare(req.body.password, user[0].password, (err, result) => {
               if (err) {
                   return res.status(401).json({
                       message: "Auth failed"
                   });
               }
               if (result) {
                   const token = jwt.sign(
                       {
                           email: user[0].email,
                           userId: user[0]._id,
                           name: user[0].name
                       },
                       secret,
                       {
                           expiresIn: "1h"
                       }
                   );
                   return res.status(200).json({
                       message: "Auth successful",
                       token: token
                   });
               }
               res.status(401).json({
                   message: "Auth failed"
               });
           });
       })
       .catch(err => {
           console.log(err);
           res.status(500).json({
               error: err
           });
       });
});
```

Email and password is sent to server as shown in code snippet and on verification either a token or message is returned to the end user.



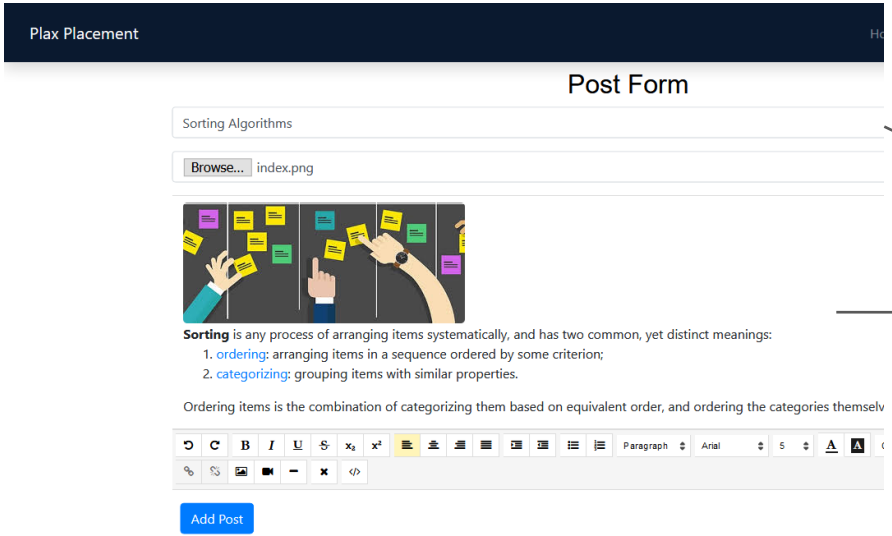

```
captcha.route('/validate').post((req, res) => {
   const token = req.body.token;
   let url = 'https://www.google.com/recaptcha/api/siteverify?secret=' +
process.env.SECRET_KEY + '&response=' + token;
   request(url, function(err, response, body){
       res.send(JSON.parse(body));
     })
})
```

Here, email has to be unique so first email is unique or not that is confirmed after that user is added to the database. reCAPTCHA is verified by fetching response from reCAPTCHA api.

```
userRoute.post("/signup", (req, res, next) => {
   User.find({email: req.body.email})
       .exec()
       .then(user => {
           if (user.length >= 1) {
               return res.status(409).json({
                   message: "Mail exists"
               });
           } else {
               bcrypt.hash(req.body.password, 10, (err, hash) => {
                   if (err) {
                       return res.status(500).json({
                           error: err
                       });
                   } else {
                       const user = new User({
                           _id: new mongoose.Types.ObjectId(),
                           name: req.body.name,
                           email: req.body.email,
                           password: hash,
                           roles: [req.body.roles]
                       });
                       user
                           .save()
                           .then(result => {
                               console.log(result);
                               res.status(201).json({
                                   message: "User created"
                               });
                           })
                           .catch(err => {
                               console.log(err);
                               res.status(500).json({
                                   error: err
                               });
                           });
                   }
               });
           }
       });
});
```

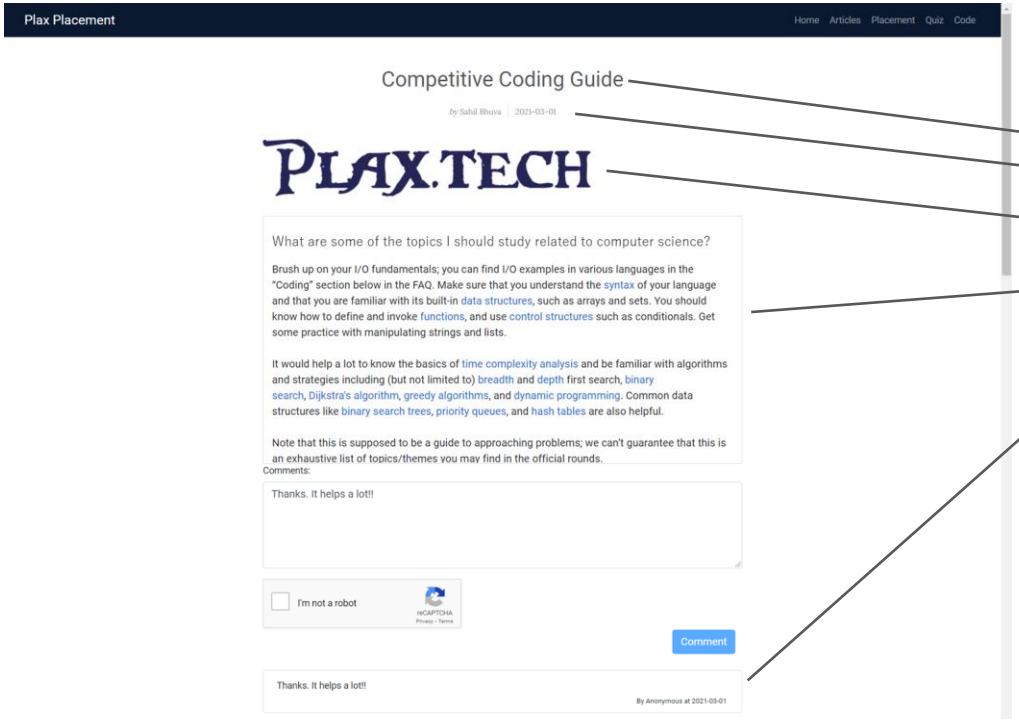
Creates new user

Middleware that confirms token




```
blogRoute.route('/add/post').post(checkAuth, async (req,
res) => {
    try {
        const image = req.file
        const _id = new mongoose.Types.ObjectId();
        const imageUrl = await uploadImage(image, _id)
        Blog.create({
            _id: _id,
            title: req.body.title,
            author: req.user.name,
            imagePath: imageUrl,
            content: req.body.content,
            comments: req.body.comments
        }, (error, data) => {
            if (error) {
                console.log(error);
                res.status(500).json({
                    error: error
                });
            } else {
                res.json(data)
            }
        })
    } catch (error) {
        console.log(error);
        res.status(500).json({
            error: error
        });
    }
});
```

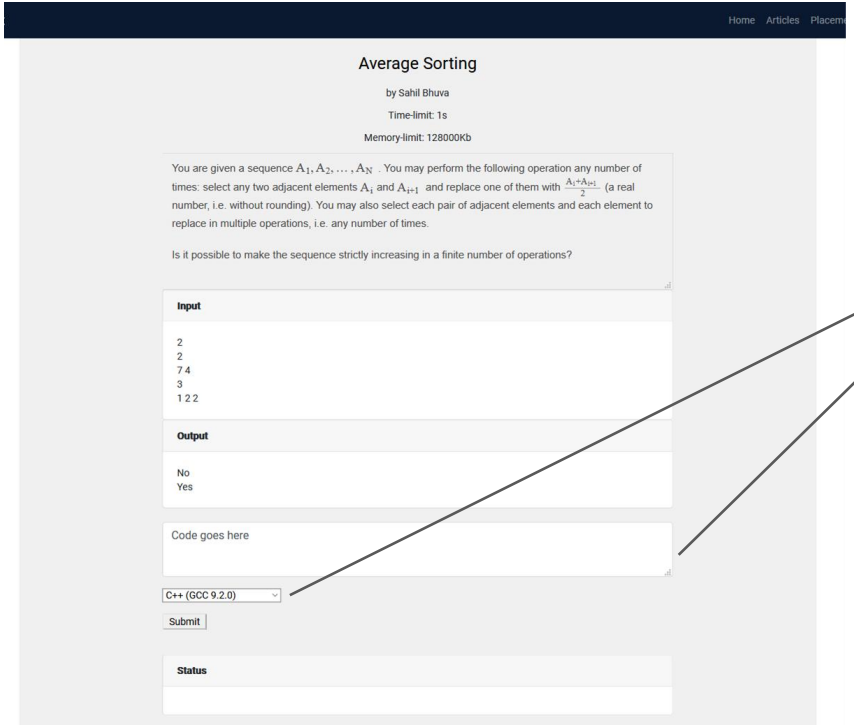
Here, first it has to pass from the middleware to confirm user is authenticate. Once verified, Users details is added to req object. After that, image will be uploaded to cloud storage that will return image URL. Next step will add data to the database and success or respective status code will be given to the end user.




```
title: String,
author: String,
date: Date,
imagePath: String,
content: String,
comments: [{
    author: String,
    content: String,
    date: Date
}]
```

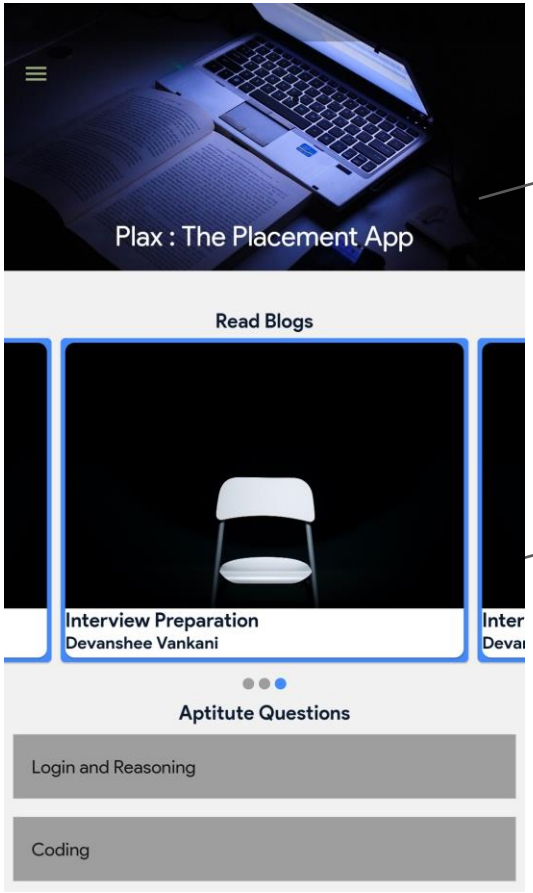
```
blogRoute.route('/get/:id').get((req, res) => {
    Blog.findById(req.params.id, (error, data) => {
        if (error) {
            console.log(error);
            res.status(500).json({
                error: error
            });
        } else {
            res.json(data)
        }
    })
})
```

So, as we can see server will fetch article by Id that is primary key for articles. And will respond with json data as above. Front-end will consume that json response and beautify the response.



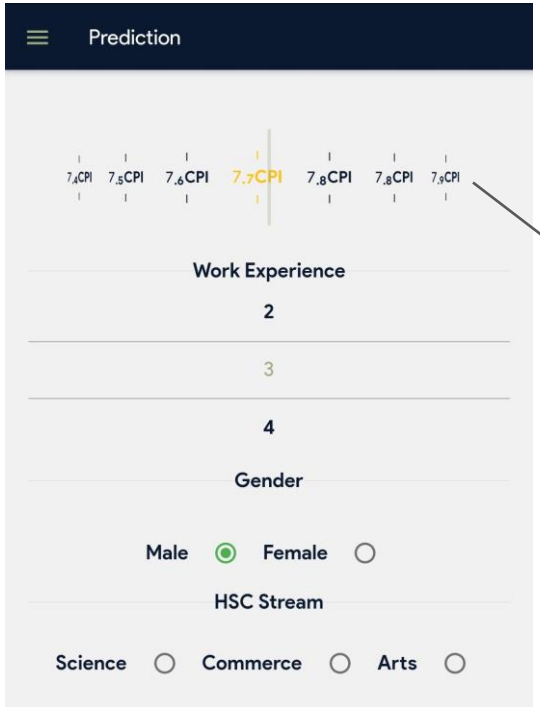

```
codingRoute.route('/create/submission/:id').post((req, res) => {
    Coding.findById(req.params.id, (error, data) => {
        if (error) {
            console.log(error);
            res.status(500).json({
                error: error
            });
        } else {
            submission = {
                "source_code": encode(req.body.source_code),
                "language_id": req.body.language_id,
                "number_of_runs": "1",
                "stdin": encode(data.TestCases[0].test),
                "expected_output": encode(data.TestCases[0].output),
                "cpu_time_limit": data.TimeLimit,
                "cpu_extra_time": "0.5",
                "wall_time_limit": "5",
                "memory_limit": data.MemoryLimit,
                "stack_limit": "64000",
                "max_processes_and_or_threads": "60",
                "enable_per_process_and_thread_time_limit": false,
                "enable_per_process_and_thread_memory_limit": false,
                "max_file_size": "1024"
            }
            console.log(submission);
            request({
                url: process.env.CODE_SERVER_URL +
'/submissions?base64_encoded=true&wait=true',
                method: "POST",
                json: true,
                body: submission
            }, function (error, response, body) {
                var status = body.status.description;
                var stderr = decode(body.stderr);
                var compile_output = decode(body.compile_output);
                r = {
                    status: status,
                    stderr: stderr,
                    compile_output: compile_output
                }
                res.status(200).json(r);
            });
        }
    })
});
```

Here, first of all test cases input, output, time limit and memory limit will be fetched from the question. Source code and language id is fetched from user input. And submission data is send to the server that compiles and runs our code. After receiving output from the server. Status, error and compilation output is given to the end user.




```
import 'package:carousel_slider/carousel_slider.dart';

SliverAppBar(
        expandedHeight: 200.0,
        floating: false,
        pinned: true,
        flexibleSpace: FlexibleSpaceBar(
            centerTitle: true,
            title: Text("Plax : The Placement App",
                style: TextStyle(
                  color: Colors.white,
                  fontSize: 16.0,
                )),
            background: Image.network(“img”,fit: BoxFit.cover)),
    ),
CarouselSlider(
        options: CarouselOptions(
          height: MediaQuery.of(context).size.height * 0.30,
          autoPlay: true,
          autoPlayInterval: Duration(seconds: 3),
          autoPlayAnimationDuration: Duration(milliseconds: 800),
          autoPlayCurve: Curves.fastOutSlowIn,
          pauseAutoPlayOnTouch: true,
          onPageChanged: (index, reason) {
            setState(() {
              _currentIndex = index;
            });
          },
        ),
        items: cardList.map((card) {
          return Builder(builder: (BuildContext context) {
            return Container(
              height: MediaQuery.of(context).size.height * 0.40,
              width: MediaQuery.of(context).size.width,
              child: Card
                child: card,
              ),
            );
          });
        }).toList();
```

The landing page of the mobile application where we can find different resources for placement preparation. It shows latest blog articles, placement opportunities and coding questions.




```python
import pandas as pd

from sklearn.model_selection import train_test_split
X_train, X_test, y_train, y_test = train_test_split(X, y,train_size=0.8,random
_state=1)
# studying and refining dataset.
# predict if a student is placed or not.
from sklearn.linear_model import LogisticRegression
logreg = LogisticRegression()
logreg.fit(X_train, y_train)

# host the model using flask.
from flask import Flask, request, jsonify


import traceback
import pandas as pd

app = Flask(__name__)

@app.route('/predict', methods=['POST'])
def predict():
    if lr:
        try:
            json_ = request.json
            print(json_)
            query = pd.get_dummies(pd.DataFrame(json_))
            query = query.reindex(columns=model_columns, fill_value=0)

            prediction = list(lr.predict(query))

            return jsonify({'prediction': str(prediction)})

        except:

            return jsonify({'trace here': traceback.format_exc()})
    else:
        print ('Train the model first')
        return ('No model here to use')
```

The user needs to enter the data using different controls. The linear regression model is trained and hosted on GCP. It is accessed as an API via Flask. The user data is send in JSON format to the model and the JSON response is received which tells the user whether he/she will be placed or not.

Plax Placement

Home Article

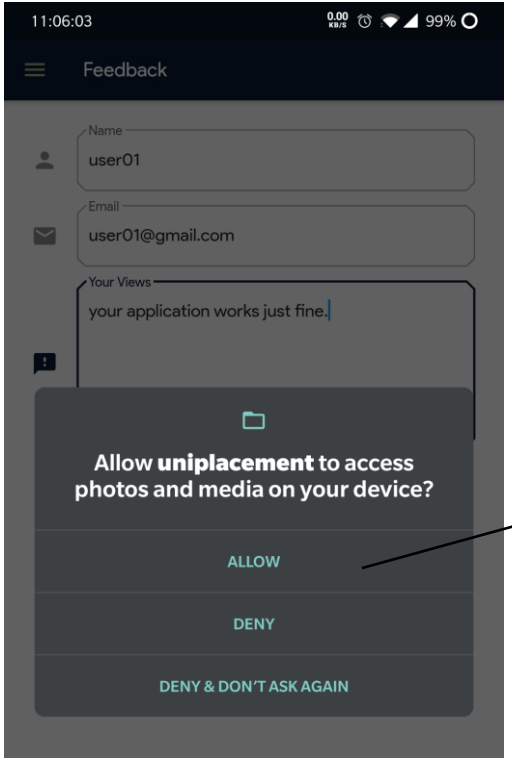
Title	Company Name	Location	History	URL	Ratings	Reviews	Experience	Salary	Qua
Opening For Software Developer @ Tech Mahindra	-	Bengaluru	12 Days Ago	Link	3.6	(10462 Reviews)	5-8 Yrs	Not disclosed	-
Senior Software Developer	-	Bengaluru	Few Hours Ago	Link	4	(8817 Reviews)	10-12 Yrs	Not disclosed	-
Senior Software Developer	-	Bengaluru	8 Days Ago	Link	4.2	(1593 Reviews)	0-8 Yrs	Not disclosed	-
Immediate Openings For 2020 Fresher's For Software developers	-	Bengaluru	11 Days Ago	Link	3.5	(137 Reviews)	0-0 Yrs	3,00,000 - 6,00,000 PA.	-
Opportunity For Software Developer - ADAS (Python + Pandas)	-	Bengaluru	2 Days Ago	Link	3.7	(9683 Reviews)	3-8 Yrs	5,00,000 - 12,00,000 PA.	-
P2- Software Engineer - Full Stack Developer	-	Bengaluru, Bangalore	7 Days Ago	Link	4.6	(5 Reviews)	0-0 Yrs	Not disclosed	-
Sr. Software Developer Engineer/ Assoc. Staff Engineer- Data Structure	-	Bengaluru	12 Days Ago	Link	4.3	(133 Reviews)	6-9 Yrs	Not disclosed	-
Senior Software Developer IBM Cloud for VMware Solutions	-	Bengaluru	4 Days Ago	Link	4	(8817 Reviews)	6-8 Yrs	Not disclosed	-
Sr . Software Developer	-	Bengaluru	4 Days Ago	Link	3.4	(370 Reviews)	5-10 Yrs	Not disclosed	-
Software Developer	-	Bengaluru	13 Days Ago	Link	3.8	(953 Reviews)	3-7 Yrs	Not disclosed	-

Items per pag 5 10 20 1 – 10 c

The web scraper, gets the data of placement opportunities from various sites and dumps it into the database. The system regularly updates the data and shows it to the user.

```
from selenium import webdriver
from bs4 import BeautifulSoup
import pandas as pd
from webdriver_manager.chrome import ChromeDriverManager
 for job_elem in job_elems:
        URL = job_elem.find('a',class_='title fw500 ellipsis').get('href')
        # Post Title
        Title = job_elem.find('a',class_='title fw500 ellipsis')

        # Company Name
        Company = job_elem.find('a',class_='subTitle ellipsis fleft')
           #   Appending data to the DataFrame
        df=df.append({'URL':URL,'Title':Title.text,'Company':Company.text,'Ratings':Ratings,'Reviews':Review
s,'Experience':Experience,'Salary':Salary,'Location':Location,'Job_Post_History':Post_History},ignore_index
= True)
```

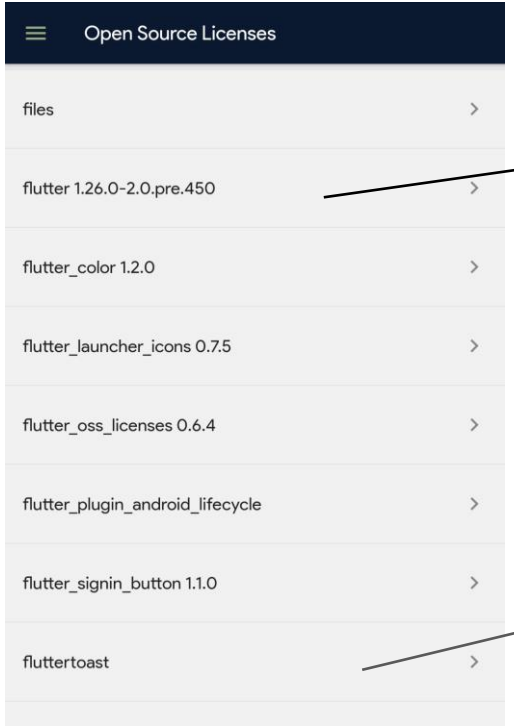
```
df.reset_index(inplace=True)
out = df.to_dict('records')

import pymongo
client = pymongo.MongoClient("db-url")
db = client['upp-db']
plc_info = db['naukri-placement']

# print(out)

insertion_result = plc_info.insert_many(out)
```




```
import 'package:image_picker/image_picker.dart';

Container(
            padding: const EdgeInsets.only(left: 130.0, top: 40.0),
            child: MaterialButton(
              color: uniFeedbackColor,
              elevation: 5,
              child: new Text('Add A Screenshot'),
              onPressed: () async {
                var file =

                  await ImagePicker.pickImage(source: ImageSource.gallery);
                var res = await uploadImage(file.path);
                setState(() {
                  state = res;
                  print(res);
                });
              },
            ),
          ),
```

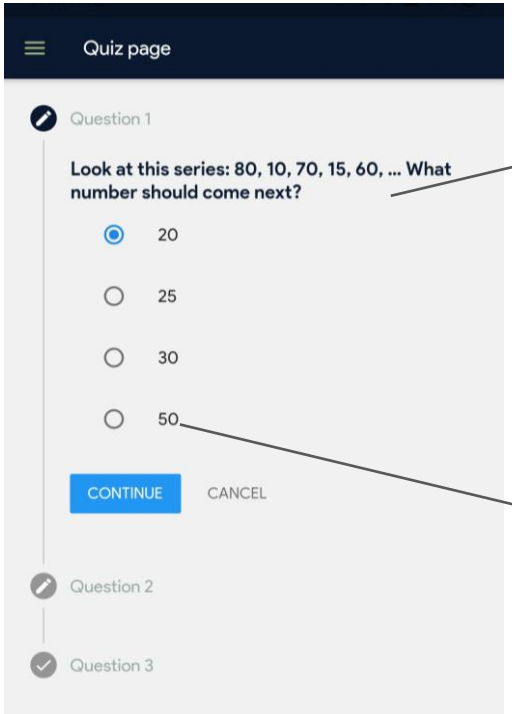
```
Future<String> uploadImage(filename) async {
  var request = http.MultipartRequest('POST', Uri.parse(url));
  request.files.add(await http.MultipartFile.fromPath('file', filename));
  var res = await request.send();
  return res.reasonPhrase;
}
```

Feedback forms helps us to know user experience. User can also report bugs and upload screenshot/file and help us improve.




```
/// This code was generated by flutter_oss_licenses
/// https://pub.dev/packages/flutter_oss_licenses

final ossLicenses = <String, dynamic>{
"flutter": {
      "name": "flutter",
      "description": "A framework for writing Flutter applications",
      "homepage": "http://flutter.dev",
      "authors": ["Flutter Authors <flutter-dev@googlegroups.com>"],
      "version": "1.26.0-2.0.pre.450",
      "license":
          "Copyright 2014 The Flutter Authors. All rights reserved.\n\nRedistr
ibution and use in source and binary forms, with or without modification,\nare
 permitted provided that the following conditions are met:\n\n ",
      "isMarkdown": false,
      "isSdk": true,
      "isDirectDependency": true
    },
}

 static Future<List<String>> loadLicenses() async {
    // merging non-dart based dependency list using LicenseRegistry.
    final ossKeys = ossLicenses.keys.toList();
    final lm = <String, List<String>>{};
    await for (var l in LicenseRegistry.licenses) {
      for (var p in l.packages) {
        if (!ossKeys.contains(p)) {
          final lp = lm.putIfAbsent(p, () => []);
          lp.addAll(l.paragraphs.map((p) => p.text));
          ossKeys.add(p);
        }
      }
    }
    for (var key in lm.keys) {
      ossLicenses[key] = {'license': lm[key].join('\n')};
    }
    return ossKeys..sort();
  }
```

Here, first of all test cases input, output, time limit and memory limit will be fetched from the question. Source code and language id is fetched from user input. And submission data is send to the server that compiles and runs our code. After receiving output from the server. Status, error and compilation output is given to the end user.




```
List<Step> steps = [
  Step(
    title: const Text('Question 1'),
    isActive: true,
    state: StepState.editing,
    content: Column(
      children: <Widget>[
        Text(quiz.question[0]),
        Column(
          children: <Widget>[
            ListTile(
              title: Text(quiz.options[0][0]),
              leading: Radio(
                value: 0,
                groupValue: answer[0],
                onChanged: (int value) {
                  setState(() {
                    answer[0] = value;
                  });},),),
```

```
class Question {
  final String question, subject, correctAnswer;
  final List<String> options;

  Question({this.question, this.subject, this.correctAnswer, this.options});

  factory Question.fromJson(Map json) {
    String option;
    List<String> temporaryList = <String>[];
    for (int i = 0; i < json['options'].length; i++) {
      option = json['options'][i];
      temporaryList.add(option);
    }
    return Question(
        question: json['question'],
        subject: json['subject'],
        correctAnswer: json['correct_answer'],
        options: temporaryList);
  }
}
```

The system provides quiz for 4 main domains (Logic and Reasoning, Language, Coding and CS Fundamentals). The multiple choice options helps the students to prepare for the aptitude test.