JOBS PLANET

By

Anitha Gundeti
Bachelor's in Information Technology, JNTUH 2019

GRADUATE CAPSTONE SEMINAR PROJECT

Submitted in partial fulfillment of the requirements

For the Degree of Master of Science,

With a Major in Computer Science

Governors State University
University Park, IL 60484

2022

ABSTRACT

Companies need help to discover candidates who are skilled enough to get hired. Companies put in a lot of effort to locate personnel who can meet their requirements. For a Job seeker, not having proper knowledge about an organization, its work culture, and its current job opening is a big challenge.

Jobs Planet is a website that is a one-stop solution for Recruiters and Job Seekers. We reach a more comprehensive range of audiences for Recruiters who can connect with the right candidate and Job Seekers to get information about organizations and their current openings. For the Recruiters, recommendations are given based on the qualifications and skill set mentioned in the job post. For Job Seekers, our website helps the users to easily navigate among the tabs to identify the openings as per their competency effortlessly.

The website Jobs Planet is built from scratch to smoothen and shorten the hiring process, providing the ultimate search experience for Recruiters and Job Seekers and cost savings for the Recruiter. This application has a user-friendly interface with better search results due to the basic and advanced search options and various jobs available. Our website provides alerts and a tracking mechanism so that users (both the job seeker and Recruiter) know each step of progress in their application.

The application is developed and is ready for release by the end of November 2022. Post the release, we aim to start with premium services on our website and develop a mobile application.

Table of Content

1 Project Description

Opening remains vacant due to undetected candidates with required skill sets. Due to these unfilled vacancies, there is considerable loss happening to them [1]. Also, some candidates are unsatisfied with their current roles but need appropriate openings matching their skill sets. Jobs Planet is designed to fill this gap. Jobs Planet is a web application built to provide a wide range of alternatives to Job seekers and recruiters. It is not easy for a recruiter to get a candidate with desired skills, and for a job seeker, it is not easy to get an opening matching one skill set, so Jobs Planet works as a platform to connect them [2]. We have designed this web application to trim the hiring process and give a joyful experience.

1.1 Competitive Information

Many online job portals are available, but Jobs Planet's distinctive features will make it ahead and successful among its competitors. The home page gives the option for basic and advanced job search options. Also, a job seeker can rate and review the company, which helps others who view the company. The Recruiter can post for job openings and consider the interest received from job seekers [3]. The Recruiter can also look for candidates and can connect with them.

1.2 Relationship to Other Applications/Projects

Jobs Planet provides unique solutions when compared to other websites and projects. Job seekers can search and apply for appropriate openings matching their skill sets, while a recruiter can post job openings and look for candidates with skill sets as per requirement. They2 can also select or reject candidates and can connect with them.

1.3 Assumptions and Dependencies

During the making of this project, we had an assumption that job seekers and recruiters have a basic understanding of computer handling and they know how to use websites. For the Admin profile, the user must be well-versed in the workflow for Job Seeker, Recruiter, and Admin-related activities. In replacement, the new person joining the role should give a proper handover.

1.4 Future Enhancements

In future releases, we intend to introduce premium accounts for Jobs Seekers and Recruiters. In the Premium services, members from Jobs Planet will help the Jobs seeker in creating their resume and will highlight these job seekers to the recruiters based on the relevant opening. Recruiters with premium services can get their account verified, which will assure the job seeker that the Recruiter is genuine. We also have the vision to develop a mobile application for this website, adding all the current features available in the web portal [4].

1.5 Definitions and Acronyms

- HTML (Hyper Text Markup Language) –. It is a markup language that is used to make web pages. In simple terms, HTML illustrates the structure of a web page and tells the browser how to show the contents.
- CSS (Cascading Style Sheets) – It is a design language. It narrates about the documents written in markup languages like HTML.
- Bootstrap – It is an open-source CSS framework. It consists of HTML, CSS, and (optionally) JavaScript-based design templates for typography, forms, buttons, navigation, and other interface components.
- CRUD (Create, Read, Update, and Delete) – Basic CRUD activities are required in any website with a user management system. These activities are used for Database and RESTful APIs.

2 Project Technical Description

To do this project, we used the below tools:

- **Operating System** – Microsoft Windows 10
- **Database** – MYSQL Workbench as an RDMS, which is downloaded from [5].
- **UI/Front End** – HTML, CSS, Bootstrap
- **Backend** – Eclipse IDE for Java as a source code editor downloaded from [6] .
- **Local Host Server** – Apache/Tomcat Server
- **Browser** – Google Chrome, Microsoft Edge
- **Documentation** – Microsoft Word and Microsoft Excel.

2.1 *Application Architecture*

- **Admin** – This user will be from Jobs Planet and have the pinnacle of all available access. He will have all the features available to the Employer and Job Seeker and additional features. Such as Dashboard to view how many applicants are registered, how many job postings are available on the site, how many active/inactive users are, and many more.
- **Recruiter** – The users in this profile can search for resumes/CVs with the appropriate skill set needed. They will be able to post job openings and review the shapes of the candidates who have applied for their job postings. They will have the option to connect with candidates via email if required.
- **Job Seeker** – The users in this profile can post their resume/CV to be visible to employers. They can search for job openings by filtering their required skill sets and preferred location. They can connect to the employers via email if needed.
- **Architecture Diagram**

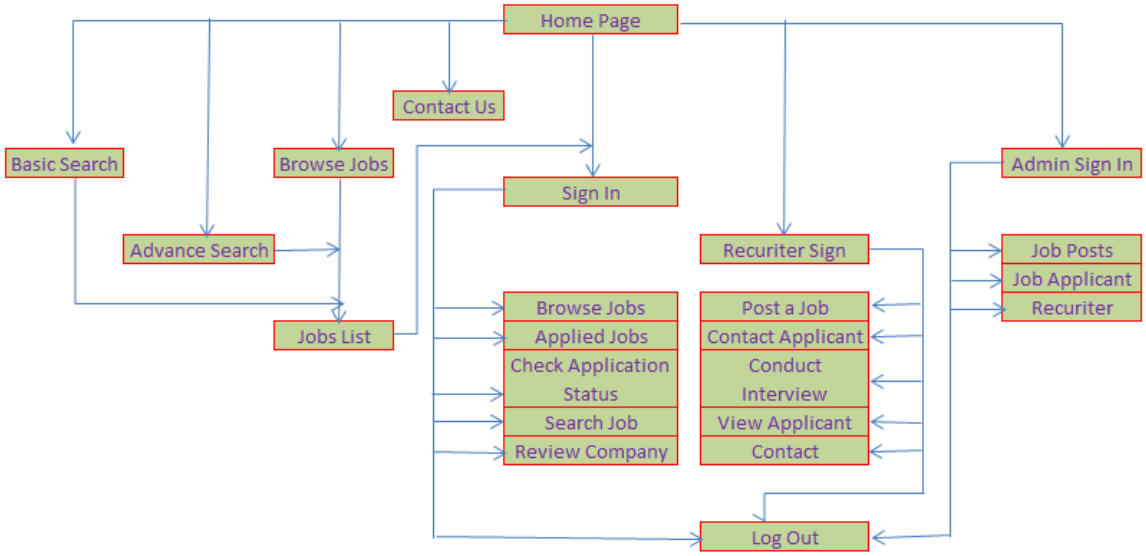

Figure 1: Application Architecture

2.2 *Application Information flows*

In this project, three roles combine to provide the solution for job search. Any visitor can search for jobs via the Basic and Advance search option available on the home page, but to apply for a job, they need to Sign In. The advanced search gives liberty to the job seeker to search for jobs with various parameters like location, job title, experience, and others. The Recruiter can only use the portal only after login. Admin is the super user among the Job Seeker and Recruiter.

2.3 *Interactions with other Applications*

Our project displays the openings, company reviews, and job seekers' details to its various viewers. We may also add content related to training for personality development which will be very useful to job seekers to break through an interview and for recruiters who will be able to find more talented resources.

2.4 *Capabilities*

- **User-Friendly Interface** – We are working to build a simple but elegant User Interface for the users.
- **Easy On boarding Process** – We will have minimalist steps and finish the users' boarding process.
- **Tracking of Applications** – We will help the users keep a watch on the application status they have applied.
- **Better Search Results** – We are working to have a diligent search that will fetch results based on the words search word suggests better profiles.
- **Alerts**– Emails will be triggered to the job seeker based on the new openings listed, or when there is a good match of the candidate profile recruiter will get such emails.
- **Variety of Jobs** – Full and part-the-time jobs can apply per their requirement.

2.5 Risk Assessment and Management

Appropriate risk assessment and management plans are in place.

- We have spent adequate time in QA testing to identify all possible issues.
- We have added all possible sample data during our testing.
- We have an Admin available 24*7 to handle any emergency.
- We have backup servers that live all the time. The backup is created at every 1-minute interval.

3 Project Requirements

3.1 Identification of Requirements

<Jobs_Planet_2022-1 Design-Capability – 000101>
The Page design should be responsive and should integrate with any devices it is being accessed (Laptop, Desktop, Mobile, or Tablet).

<Jobs_Planet_2022-1 Search-Capability – 000102>
The Job Seeker should be able to get fast search results with the advanced search based on the filters selected.

<Jobs_Planet_2022-1 Registration-Capability – 000103>
The Job Seeker and Recruiter should have a hassle-free registration experience to perform the desired action on the portal.

3.2 Operations, Administration, Maintenance and Provisioning (OAM&P)

Any visitor to the Jobs Planet website can view jobs listed on the Browser Jobs page. They can also search for jobs using the Basic and Advanced Search options. To apply for the jobs, they need to register. Similarly, for a Recruiter to post an opening, they need to register. The admin can view all this information for the admin login. All the data related to job postings and job applications are stored under the company or job seeker profiles and in the database under specified tables. Regular backups are created to safeguard the website from any data losses.

3.3 Security and Fraud Prevention

The Security and Fraud prevention planning consists of the below steps.

- We use encryption to link between a web server and a web browser.
- We have placed firewalls and Anti-spyware software on our servers.
- We are educating all our team members to monitor the website for any suspicious things regularly.
- We regularly implement the necessary patch updates.
- We save sensitive data in encrypted form.

3.4 Release and Transition Plan

The release and Transition Plan consists of the below steps.

- Before the website's release and deployment, rigorous testing is done to identify any issues or bugs that need to be fixed.
- All the codes (Frontend, Backend, and DB) are well documented.
- A project Release/Transition checklist is in place, so monitoring all the Knowledge Transfer is proper.
- Obtain feedback from users and improve the website accordingly.

4 Project Design Description

We followed the Waterfall approach during our software development life cycle. We developed this project using Eclipse IDE for backend development using Java as a coding language. We used Visual Studio Code for writing the frontend codes using HTML, CSS, and Bootstrap. For the database part, we used Workbench to develop scripts in MySQL. Our application is designed to run in all the leading browsers available in the market as of date.
A Job seeker can view the jobs listed using the Advanced and Basic search options, but to apply for them, they need to be a member of the Jobs Planet Family by registering and logging in using their credentials. They can view the history of jobs applied and their status of them. A job seeker can also review and rate the companies as well.
The Recruiter needs to be a part of the Jobs Planet group by registering using the Sign-Up page and logging in using their credentials. Upon login, they have various options available to them. They have the opportunity to post job openings. Along with

that, they can also view and connect with applicants/candidates. A Recruiter has the contact option to connect directly with the applicants. They can also conduct interviews with the selected candidates.
Admin is the supreme user among the 3. Admin also validates the information provided by the job seeker or Recruiter. The admin needs to create the categories available to the Recruiter to post for a job. They also need to monitor the job posts and applications for inappropriate content.
The contact feature of the Recruiter, along with ease of using the application, user-friendly design, and 24*7 Admin support, are some of the critical elements of Jobs Planet. The compatibility ease of Jobs Planet to open in any web browser along with mobile phones and tablets makes the user experience delighted instead of just satisfactory.

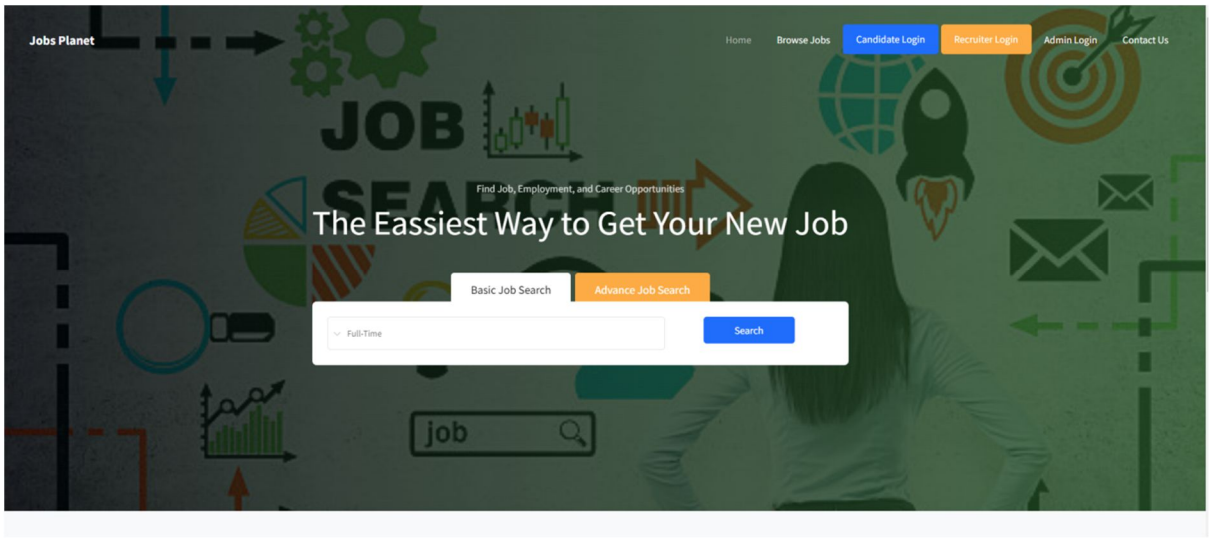

Figure 2: Home Page

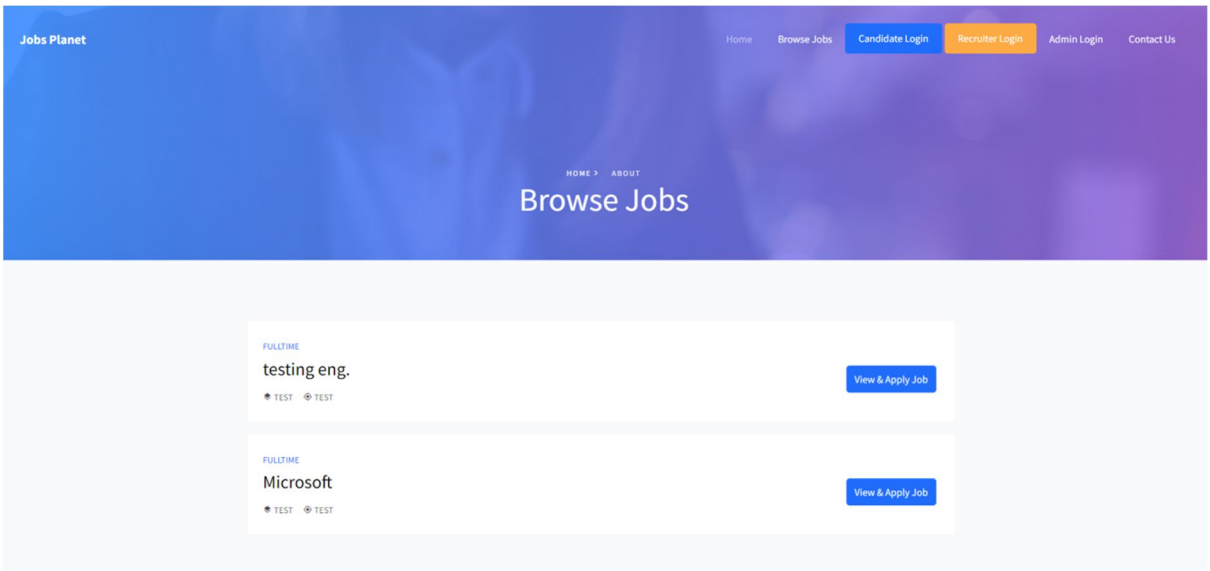

Figure 3: Browser Jobs Page

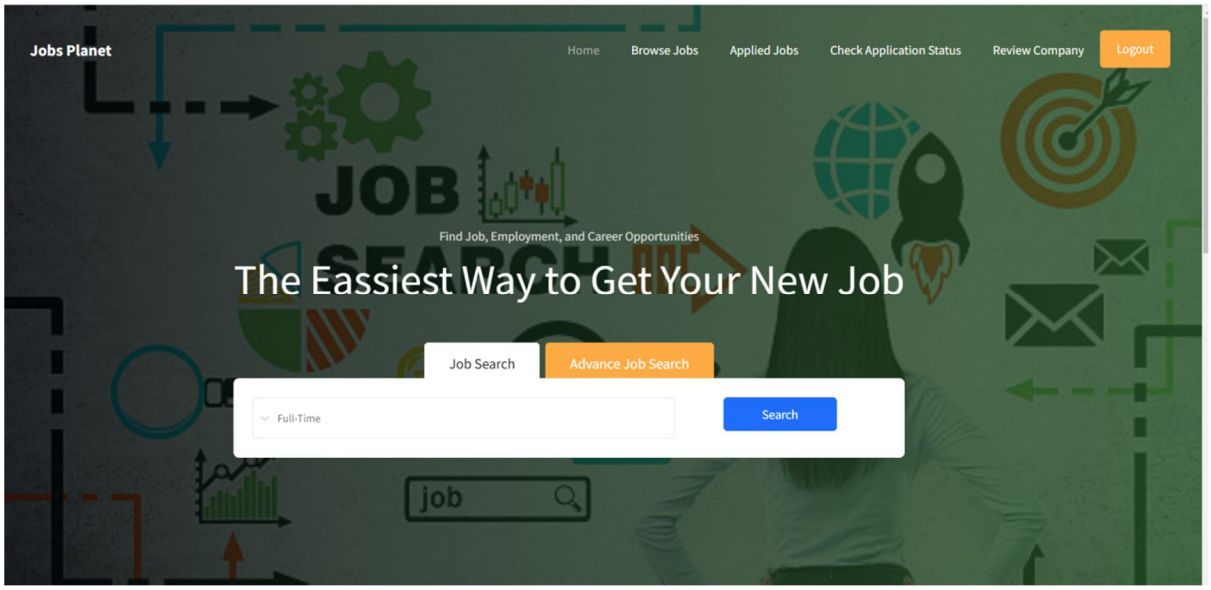

Figure 4: Job Seeker Home Page

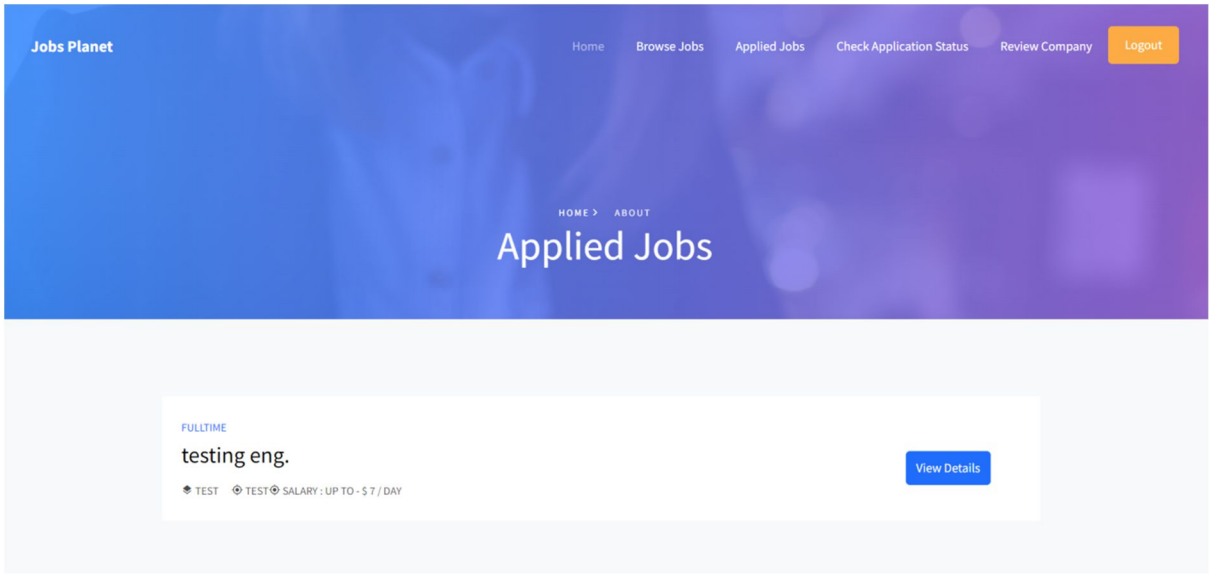

Figure 5: Applied Jobs Page

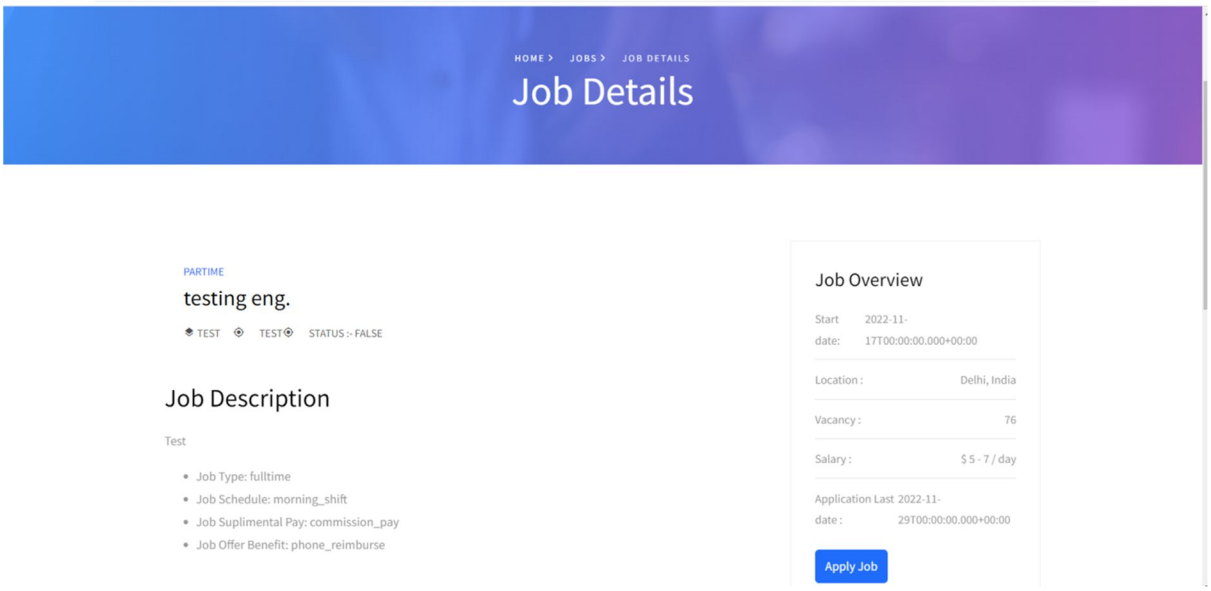

Figure 6: Jobs Details Page

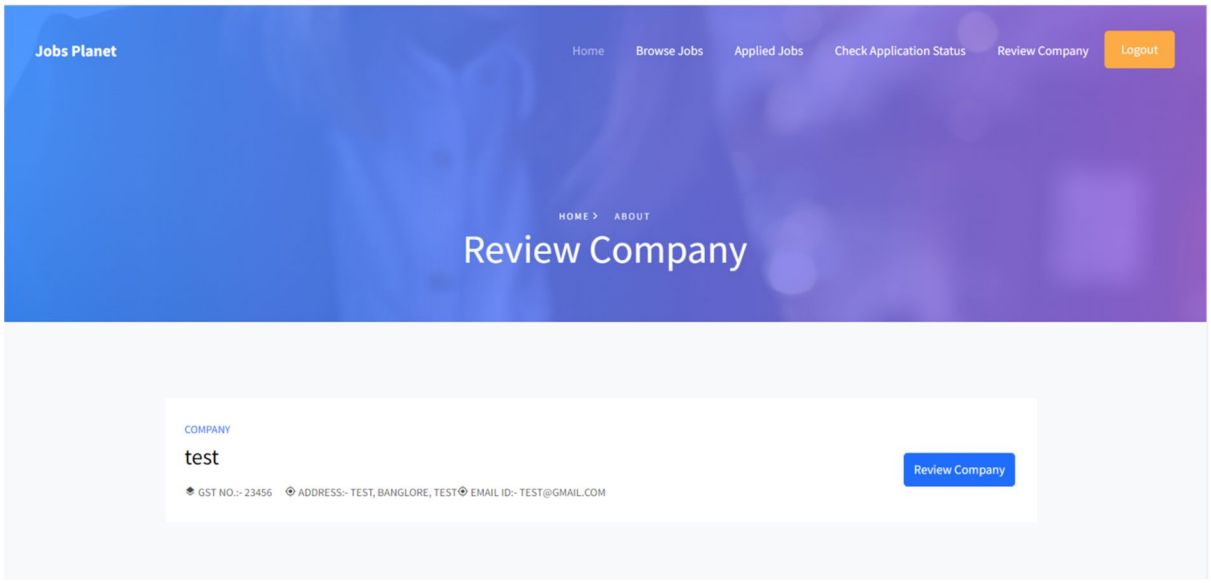

Figure 7: Company Review Page

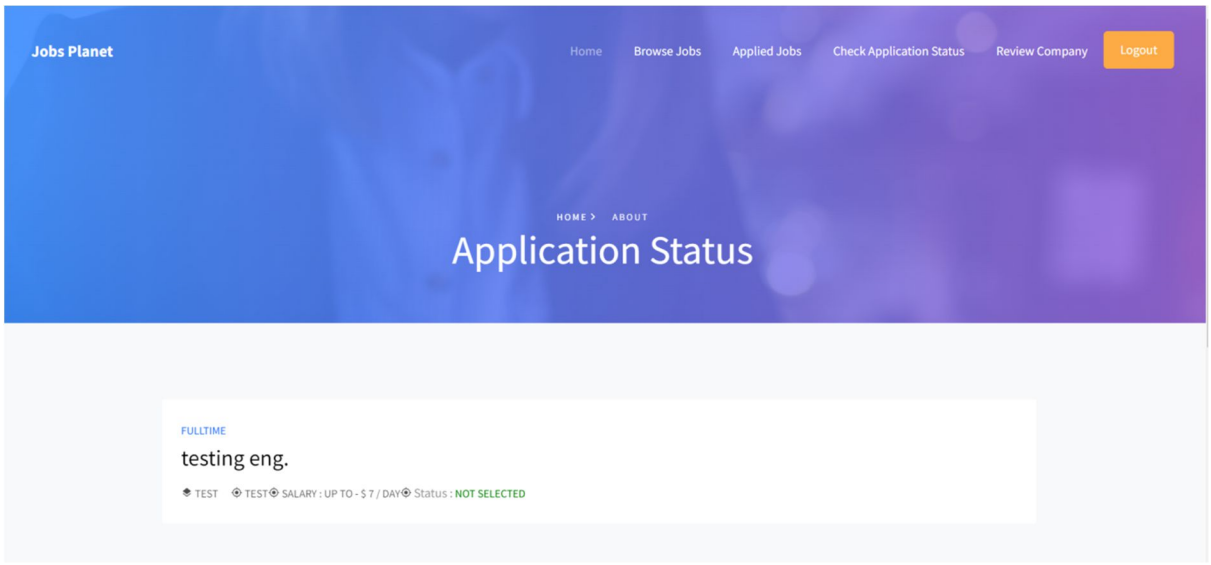

Figure 8: Application Status Page

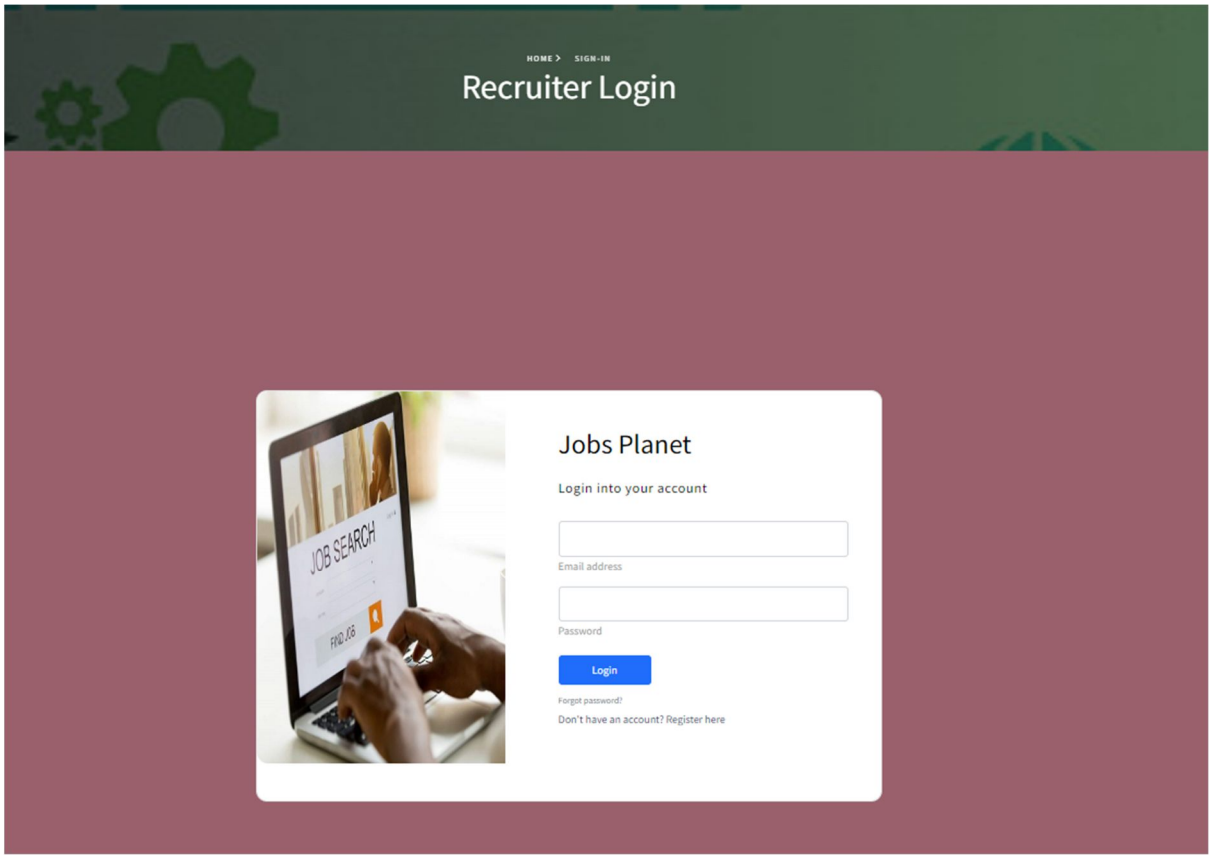

Figure 9: Recruiter Login Page

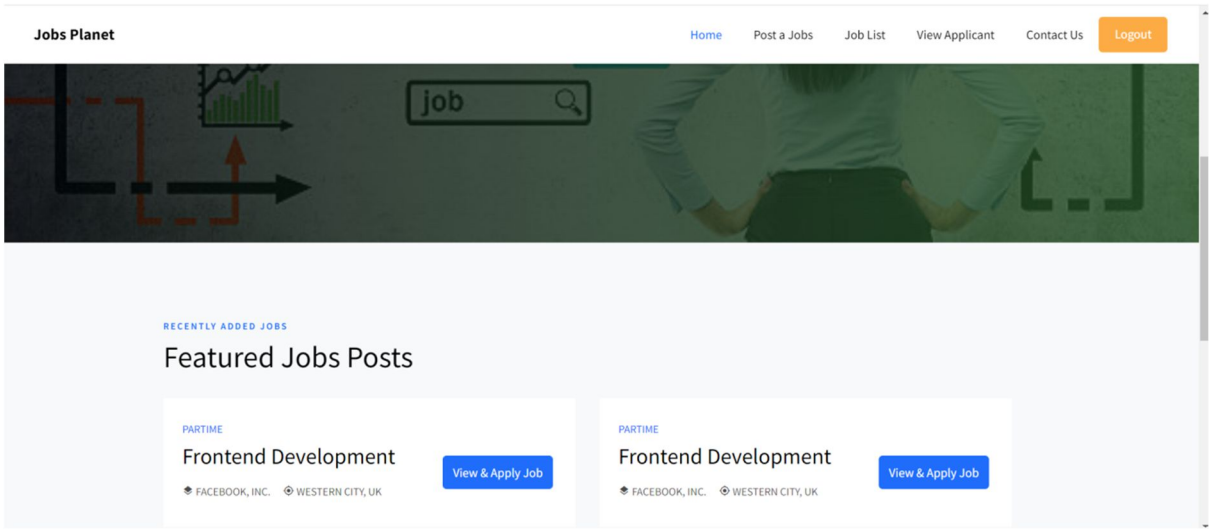

Figure 10: Recruiter Home Page

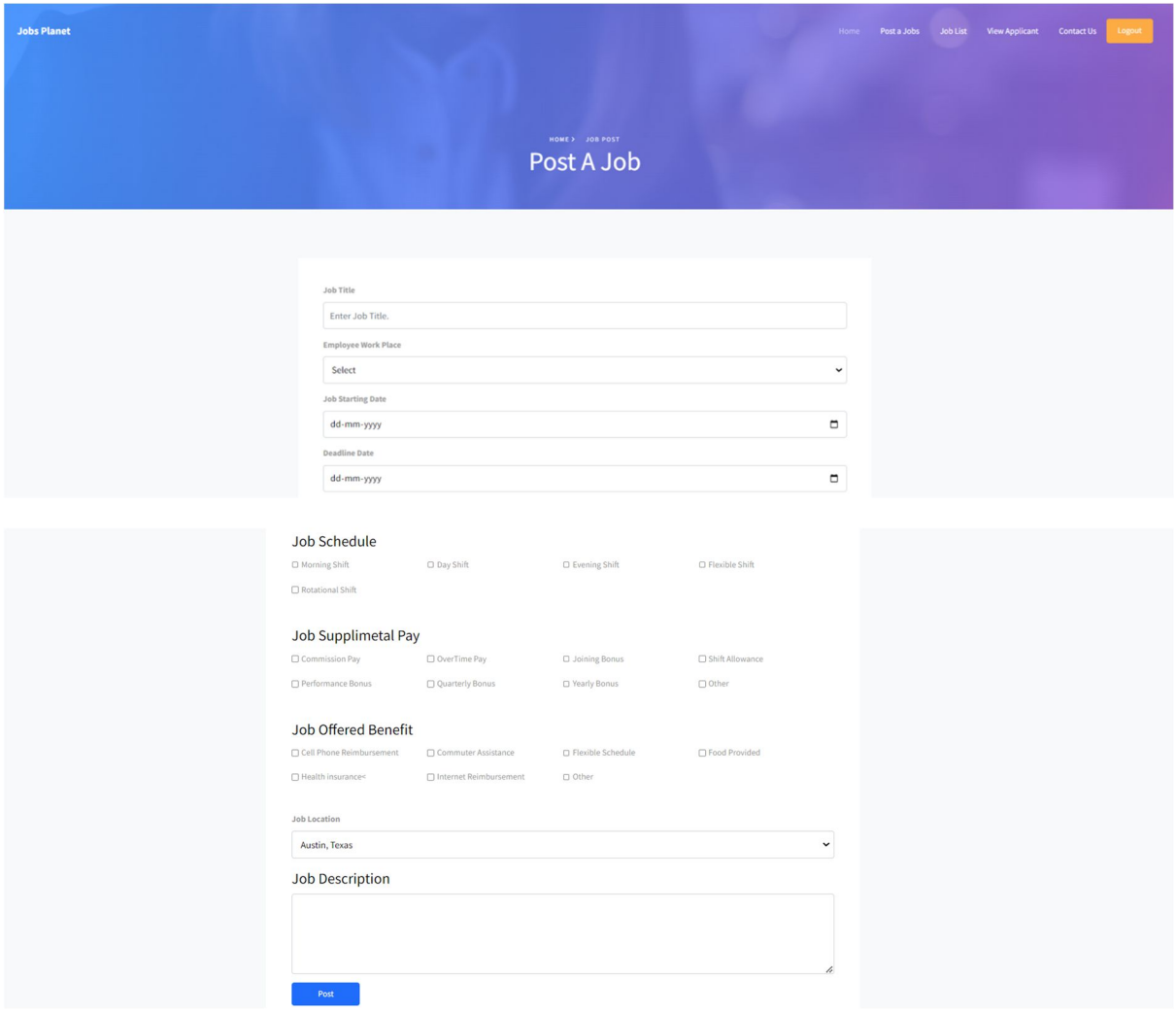

Figure 11: Post Job Page

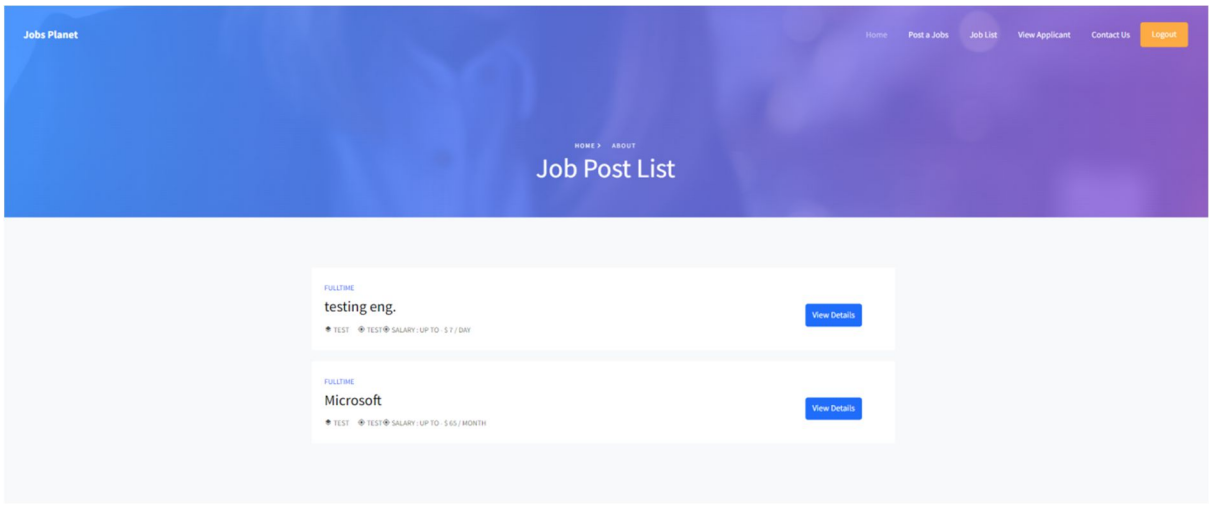

Figure 12: Recruiter Job List Page

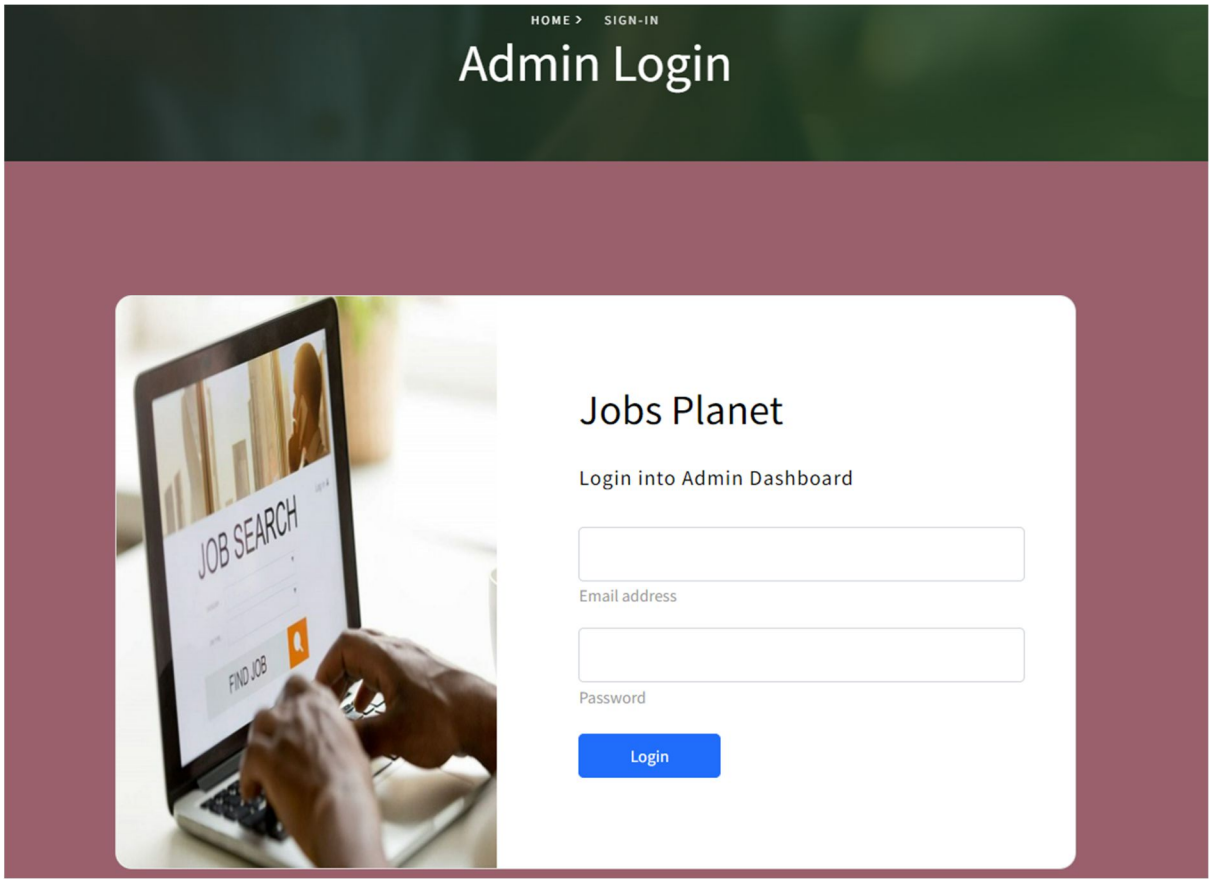

Figure 13: Admin Login Page

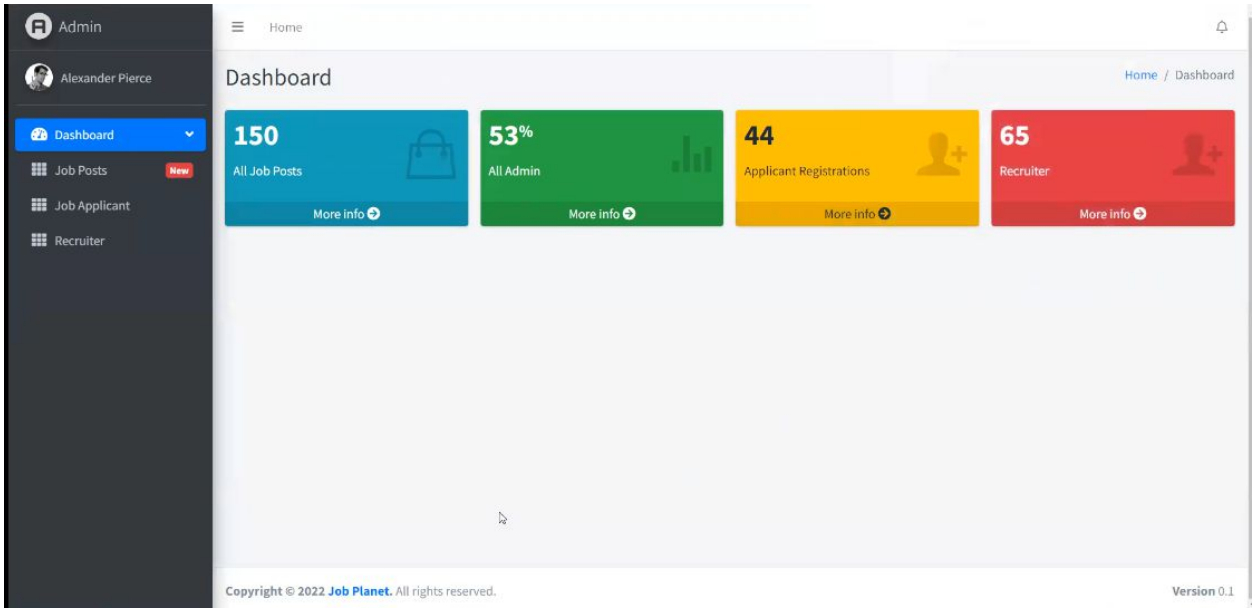

Figure 14: Admin Home Page

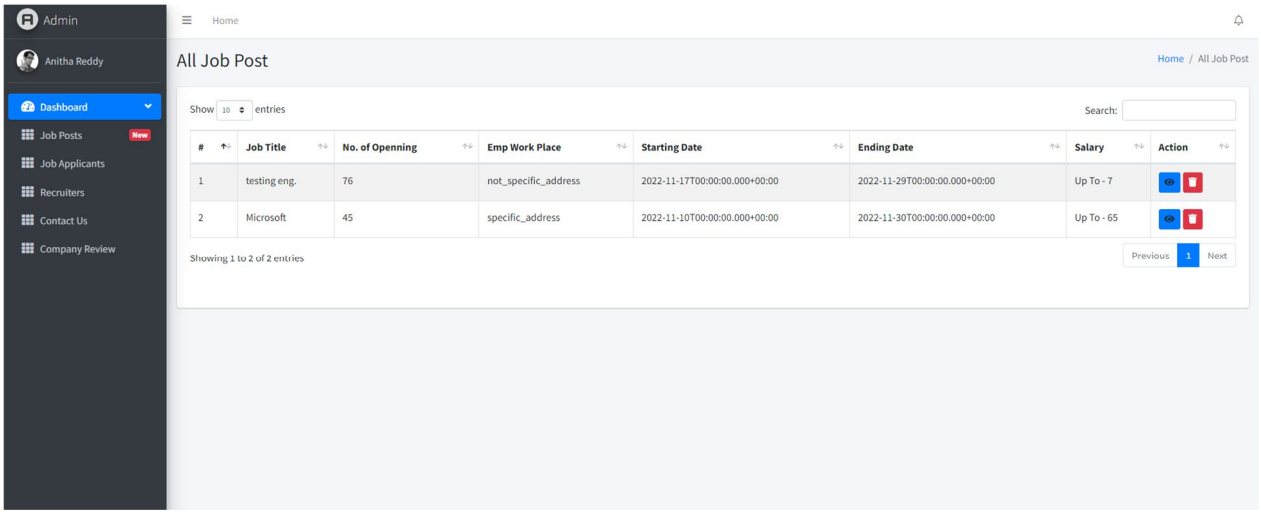

Figure 15: Admin Job Post

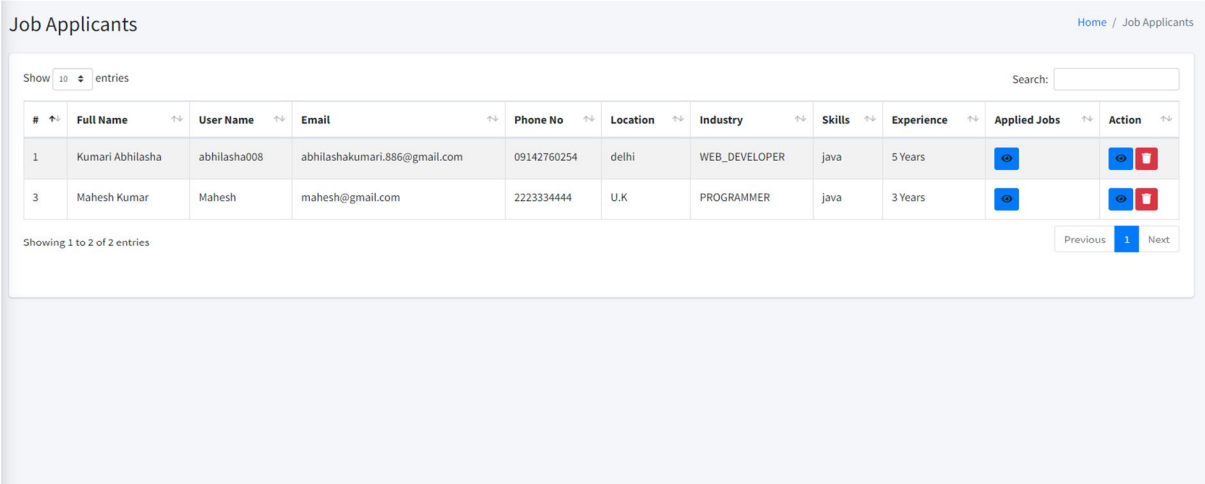

Figure 16: Admin Job Applicants

5 Internal/external Interface Impacts and Specification

Basic requirements about a system are required for the complete fledge and uninterrupted running of the portal. The application is designed and developed as per the requirement identified. The developers achieved the goal of a smooth application due to their regular interaction with the users and the continuous improvement made during the development process from the feedback obtained by them.

The system's architectural design describes the structure, behavior, additional viewpoints, and analyses. This system was developed using the three-tier architectural approach. The first tier is the interface, such as a web browser, through which the users interact with the website. The second or middle tier is the application logic that transmits data or information between the first and second levels. The third tier includes the database management system, which is responsible for storing the data and retrieving them, retrieving the data as and when needed.

6 Design Units Impacts

6.1 Functional Area A/Design Unit A

6.1.1 Functional Overview

The Project is designed to be useful for the Recruiter and Job Seeker to come on the same platform and connect.

6.1.2 Impacts

This design workflow method for the job seeker, Admin, and Recruiter is designed to impact the operational and client requirements. Authentication and data updates, user information updates, backup, and recovery (in case of conditions) are all part of the administration process.

6.1.3 Requirements

The admin verifies the information the job seeker or Recruiter provides during the authentication process. Admin should monitor the job seeker and the Recruiter for any enhancements based on feedback. Data safety is the top priority, and a data backup method should be implemented.

7 Acknowledgements

I want to express my sincere gratitude to Dr. Dae Wook (Wooky) Kim, Assistant Professor of the Division of Science, Mathematics, and Technology at Governors State University, for his contributions to the completion of this project. It was a great learning experience. I want to take this golden opportunity to convey my appreciation to all of the group members for their cooperation and input, without whom the project would not have been successful.

8 References

[1] https://www.placementindia.com, 'Why Online Job Portals Are Preferred, 2018, Online, Available: https://www.placementindia.com/blog/why-online-job-portals-are-preferred.htm#:~:text=These%20online%20job%20portals%20are,done%20by%20that%20job%20portal.

[2] https://www.perceptionsystem.com, 'Why should you create a job portal for your organization, Online, Available: https://www.perceptionsystem.com/blog/why-should-you-create-a-job-portal-for-your-organization/

[3] journal Article, E-Recruitment Transforming the Dimensions of Online Job Seeking: A Case of Pakistan, Online, Available: https://www.academia.edu/10669242/E_Recruitment_Transforming_the_Dimensions_of_Online_Job_Seeking_A_Case_of_Pakistan

[4] https://www.webdesignmuseum.org, 'Web Design History Timeline,' Online, Available: https://www.webdesignmuseum.org/web-design-history/timeline-2009-2017

[5] MySQL.2022 (Version8.0.31) Oracle, Available: https://dev.mysql.com/downloads/installer/.

[6] Eclipse IDE 2022-09 IBM, Available: https://www.eclipse.org/downloads/

9 Appendices

Backend codes

```java
package com.gsu.app.jobsplanet.controller;

import java.util.List;

import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.web.bind.annotation.CrossOrigin;
import org.springframework.web.bind.annotation.DeleteMapping;
import org.springframework.web.bind.annotation.GetMapping;
import org.springframework.web.bind.annotation.PatchMapping;
import org.springframework.web.bind.annotation.PathVariable;
import org.springframework.web.bind.annotation.PostMapping;
import org.springframework.web.bind.annotation.PutMapping;
import org.springframework.web.bind.annotation.RequestBody;
import org.springframework.web.bind.annotation.RequestMapping;
import org.springframework.web.bind.annotation.RestController;

import com.gsu.app.jobsplanet.dtos.JobDto;
import com.gsu.app.jobsplanet.entity.JobPost;
import com.gsu.app.jobsplanet.entity.Jobs;
import com.gsu.app.jobsplanet.service.JobPostService;
import com.gsu.app.jobsplanet.service.JobsService;

@CrossOrigin(origins = "http://127.0.0.1:5501", maxAge = 3600)
@RestController
@RequestMapping("/jobPost")
/*
 * this is an job post controller class
 */
public class JobPostController {

    @Autowired
    private JobPostService jobPostService;

    @PostMapping("/save")
    public JobPost createJobPost(@RequestBody JobPost jobPost) {
        return jobPostService.createJobPost(jobPost);
    }

    @PutMapping("/{id}")
    public JobPost updateJobPost(@PathVariable long id, @RequestBody JobPost jobPost) {
        return jobPostService.updateJobPost(id, jobPost);
    }

    /*
     * @PutMapping("/updateSingleColoum/{id}") public JobPost
     * updateSingleColoumJobPost(@PathVariable long id, @RequestBody JobPost
     * jobPost) { return jobPostService.updateSingleColoumJobPost(id, jobPost); }
     */

```

Figure 17: Job Post Controller

```java
package com.gsu.app.jobsplanet.entity;

import java.util.Date;
import java.util.List;

import javax.persistence.Entity;
import javax.persistence.FetchType;
import javax.persistence.GeneratedValue;
import javax.persistence.GenerationType;
import javax.persistence.Id;
import javax.persistence.JoinColumn;
import javax.persistence.ManyToOne;
import javax.persistence.Table;
import javax.validation.constraints.NotNull;
import javax.validation.constraints.Size;

import lombok.Getter;
import lombok.Setter;

@Getter
@Setter
@Entity
@Table(name = "job_post")
public class JobPost {

    @Id
    @GeneratedValue(strategy = GenerationType.IDENTITY)
    private long id;

    @Size(max = 100, message = "criteria not met")
    private String jobTitle;

    private String employeeWorkPlace;

    private Date startingDate;

    private Date deadLineDate;

    private String minimumPay;

    private String maximumPay;

    private String rate;

    private String resume;

    private String noOfOpening;

    private String jobType;
```

Figure 18: Job Post Entity

```java
package com.gsu.app.jobsplanet.repository;

import java.util.List;
import java.util.Optional;

import org.springframework.data.jpa.repository.JpaRepository;
import org.springframework.stereotype.Repository;

import com.gsu.app.jobsplanet.entity.JobPost;
import com.gsu.app.jobsplanet.entity.Jobs;

@Repository
public interface JobPostRepository extends JpaRepository<JobPost, Long> {
    Optional<JobPost> findByJobType(String jobType);
    List<JobPost> findByJobTypeAndJobLocation(String jobType, String locationId);
    //Optional<JobPost> findByTypeAndAddress(String jobType, String locationId);
    }
```

Figure 19: Job Post Repository

```java
package com.gsu.app.jobsplanet.service;

import java.util.List;

import javax.transaction.Transactional;

import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.stereotype.Service;

import com.gsu.app.jobsplanet.entity.JobPost;
import com.gsu.app.jobsplanet.entity.JobSeeker;
import com.gsu.app.jobsplanet.entity.Recruiter;
import com.gsu.app.jobsplanet.repository.JobPostRepository;
import com.gsu.app.jobsplanet.repository.JobSeekerRepository;
import com.gsu.app.jobsplanet.repository.RecruiterRepository;

@Service
public class JobPostServiceImpl implements JobPostService {

    @Autowired
    private JobPostRepository jobPostRepository;

    @Autowired
    private JobSeekerRepository jobSeekerRepository;

    @Autowired
    private RecruiterRepository recruiterRepository;

    @Override
    public List<JobPost> getAllJobPostList() {
        return jobPostRepository.findAll();
    }

    @Override
    public JobPost createJobPost(JobPost jobPost) {
        Recruiter recruiter = recruiterRepository.findById(jobPost.getRecruiter().getRecruiterId())
                .orElseThrow(() -> new RuntimeException("RecruiterId Not Found"));
        jobPost.setRecruiter(recruiter);
        jobPostRepository.save(jobPost);
        return jobPost;
    }

    @Override
    public JobPost updateJobPost(long id, JobPost jobPost) {
        JobPost updateJobPost = jobPostRepository.findById(id)
                .orElseThrow(() -> new RuntimeException("Job Id Not Found"));
        updateJobPost.setJobTitle(jobPost.getJobTitle());
        updateJobPost.setEmployeeWorkPlace(jobPost.getEmployeeWorkPlace());
        updateJobPost.setStartingDate(jobPost.getStartingDate());
```

Figure 20: Job Post Service IMPL

```java
package com.gsu.app.jobsplanet.controller;

import java.util.List;

import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.web.bind.annotation.CrossOrigin;
import org.springframework.web.bind.annotation.DeleteMapping;
import org.springframework.web.bind.annotation.GetMapping;
import org.springframework.web.bind.annotation.PatchMapping;
import org.springframework.web.bind.annotation.PathVariable;
import org.springframework.web.bind.annotation.PostMapping;
import org.springframework.web.bind.annotation.PutMapping;
import org.springframework.web.bind.annotation.RequestBody;
import org.springframework.web.bind.annotation.RequestMapping;
import org.springframework.web.bind.annotation.RestController;

import com.gsu.app.jobsplanet.dtos.JobDto;
import com.gsu.app.jobsplanet.entity.Jobs;
import com.gsu.app.jobsplanet.service.JobsService;

@CrossOrigin(origins = "http://127.0.0.1:5501", maxAge = 3600)
@RestController
@RequestMapping("/jobs")
/*
 * this is jobs controller class
 */
public class JobsController {

    @Autowired
    private JobsService jobsService;

    @PostMapping("/save")
    public Jobs createProject(@RequestBody JobDto jobs) {
        return jobsService.createJobs(jobs);
    }

    @PutMapping("/{id}")
    public Jobs updateProject(@PathVariable long id, @RequestBody Jobs jobs) {
        return jobsService.updateJobs(id, jobs);
    }

    @GetMapping("/{jobsId}")
    public Jobs getJobsById(@PathVariable long jobsId) {
        return jobsService.getJobsById(jobsId);
    }

    @GetMapping("/search/{jobType}")
    public Jobs searchJobType(@PathVariable String jobType) {
        return jobsService.searchJobType(jobType);
```

Figure 21: Jobs Controller

```java
package com.gsu.app.jobsplanet.entity;

import java.util.HashSet;
import java.util.Set;

import javax.persistence.CascadeType;
import javax.persistence.Entity;
import javax.persistence.FetchType;
import javax.persistence.GeneratedValue;
import javax.persistence.GenerationType;
import javax.persistence.Id;
import javax.persistence.JoinColumn;
import javax.persistence.ManyToOne;
import javax.persistence.OneToMany;
import javax.persistence.OneToOne;
import javax.persistence.Table;
import javax.validation.constraints.NotNull;
import javax.validation.constraints.Size;

import lombok.Getter;
import lombok.Setter;

@Getter
@Setter
@Entity
@Table(name = "jobs")
public class Jobs {

    @Id
    @GeneratedValue(strategy = GenerationType.IDENTITY)
    private long id;

    @NotNull(message = "Job Name can not be null")
    @Size(max = 100, message = "criteria not met")
    private String name;

    @NotNull(message = "Job Type can not be null")
    @Size(max = 100, message = "criteria not met")
    private String type;

    private String description;

    @NotNull(message = "Job Address can not be null")
    @Size(max = 100, message = "criteria not met")
    private String address;

    @NotNull(message = "Company Name can not be null")
    @Size(max = 100, message = "criteria not met")
    private String companyName;
```

Figure 22: Jobs Entity

```java
package com.gsu.app.jobsplanet.repository;

import java.util.List;
import java.util.Optional;

import org.springframework.data.jpa.repository.JpaRepository;
import org.springframework.stereotype.Repository;

import com.gsu.app.jobsplanet.entity.Jobs;

@Repository
public interface JobsRepository extends JpaRepository<Jobs, Long> {
    Optional<Jobs> findByType(String jobType);
    Optional<Jobs> findByTypeAndAddress(String jobType, String locationId);
    List<Jobs> findByJobSeekerJobSeekerId(Long jobSeekerId);
    //Jobs findByJobPostIdAndJobSeekerId(Long jobPostId, Long jobSeekerId);
}
```

Figure 23: Jobs Repository

```java
package com.gsu.app.jobsplanet.service;

import java.util.List;
import java.util.Optional;

import javax.transaction.Transactional;

import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.stereotype.Service;

import com.gsu.app.jobsplanet.dtos.JobDto;
import com.gsu.app.jobsplanet.entity.ContactUs;
import com.gsu.app.jobsplanet.entity.JobPost;
import com.gsu.app.jobsplanet.entity.JobSeeker;
import com.gsu.app.jobsplanet.entity.Jobs;
import com.gsu.app.jobsplanet.entity.Recruiter;
import com.gsu.app.jobsplanet.repository.ContactUsRepository;
import com.gsu.app.jobsplanet.repository.JobPostRepository;
import com.gsu.app.jobsplanet.repository.JobSeekerRepository;
import com.gsu.app.jobsplanet.repository.JobsRepository;
import com.gsu.app.jobsplanet.repository.RecruiterRepository;

@Service
public class JobsServiceImpl implements JobsService {

    @Autowired
    private JobsRepository jobsRepository;

    @Autowired
    private JobPostRepository jobPostRepository;

    @Autowired
    private JobSeekerRepository jobSeekerRepository;

    @Autowired
    private RecruiterRepository recruiterRepository;

    @Override
    public List<Jobs> getAllJobsList() {
        return jobsRepository.findAll();
    }

    @Override
    public Jobs createJobs(JobDto jobs) {
        Jobs saveJobs = new Jobs();
        Recruiter recruiter = recruiterRepository.findById(jobs.getRecruiterId())
                .orElseThrow(() -> new RuntimeException(" Id Not Found"));
        JobSeeker jobSeeker = jobSeekerRepository.findById(jobs.getJobSeekerId())
                .orElseThrow(() -> new RuntimeException(" Id Not Found"));
```

Figure 24: Jobs Service IMPL

```java
package com.gsu.app.jobsplanet.controller;

import java.util.List;

import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.http.HttpStatus;
import org.springframework.http.ResponseEntity;
import org.springframework.web.bind.annotation.CrossOrigin;
import org.springframework.web.bind.annotation.DeleteMapping;
import org.springframework.web.bind.annotation.GetMapping;
import org.springframework.web.bind.annotation.PathVariable;
import org.springframework.web.bind.annotation.PutMapping;
import org.springframework.web.bind.annotation.RequestBody;
import org.springframework.web.bind.annotation.RequestMapping;
import org.springframework.web.bind.annotation.RestController;

import com.gsu.app.jobsplanet.entity.JobSeeker;
import com.gsu.app.jobsplanet.service.JobSeekerService;

@CrossOrigin(origins = "http://127.0.0.1:5501", maxAge = 3600)
@RestController
@RequestMapping("/applicants")
/*
 * this is an applicant controller class
 */
public class ApplicantController {

    @Autowired
    private JobSeekerService jobSeekerService;

    @GetMapping
    public ResponseEntity<List<JobSeeker>> getAllJobSeekerList() {
        return ResponseEntity.ok(jobSeekerService.getAllJobSeekerList());
    }

    @GetMapping("/{id}")
    public ResponseEntity<JobSeeker> getJobSeekerById(@PathVariable Long id) {
        return ResponseEntity.ok(jobSeekerService.getJobSeekerById(id));
    }

    @PutMapping("/{id}")
    public ResponseEntity<Void> updateJobSeeker(@PathVariable Long id, @RequestBody JobSeeker jobSeeker) {
        jobSeekerService.updateJobSeeker(id, jobSeeker);
        return ResponseEntity.status(HttpStatus.NO_CONTENT).build();
    }

    @DeleteMapping("/{id}")
    public ResponseEntity<Void> deleteJobSeeker(@PathVariable Long id) {
        jobSeekerService.deleteJobSeeker(id);
```

Figure 25: Applicant Controller

```java
package com.gsu.app.jobsplanet.entity;

import javax.persistence.Entity;
import javax.persistence.EnumType;
import javax.persistence.Enumerated;
import javax.persistence.GeneratedValue;
import javax.persistence.GenerationType;
import javax.persistence.Id;
import javax.persistence.Table;
import javax.validation.constraints.NotNull;
import javax.validation.constraints.Size;

import lombok.Getter;
import lombok.Setter;

@Getter
@Setter
@Entity
@Table(name = "job_seeker")
public class JobSeeker {

    @Id
    @GeneratedValue(strategy = GenerationType.AUTO)
    private long jobSeekerId;

    @NotNull(message = "Full Name can not be null")
    private String fullName;

    @NotNull(message = "UserName can not be null")
    private String userName;

    @NotNull(message = "Email can not be null")
    private String email;

    private String mobileNumber;

    private String profilePic;

    private String resume;

    private String location;

    private String experiance;

    private String industry;

    @NotNull(message = "Company Name can not be null")
    private String userSkills;

```

Figure 26: Job Seeker Entity

```java
package com.gsu.app.jobsplanet.repository;

import org.springframework.data.jpa.repository.JpaRepository;
import org.springframework.stereotype.Repository;

import com.gsu.app.jobsplanet.entity.JobSeeker;

@Repository
public interface JobSeekerRepository extends JpaRepository<JobSeeker, Long> {

}
```

Figure 27: Job Seeker Repository

```java
package com.gsu.app.jobsplanet.service;

import java.util.List;

import javax.transaction.Transactional;

import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.stereotype.Service;

import com.gsu.app.jobsplanet.entity.JobSeeker;
import com.gsu.app.jobsplanet.entity.Users;
import com.gsu.app.jobsplanet.repository.JobSeekerRepository;
import com.gsu.app.jobsplanet.repository.UsersRepository;

@Service
public class JobSeekerServiceImpl implements JobSeekerService {

    @Autowired
    JobSeekerRepository jobSeekerRepository;

    @Autowired
    private UsersRepository usersRepository;

    @Override
    @Transactional
    public List<JobSeeker> getAllJobSeekerList() {
        return jobSeekerRepository.findAll();
    }

    @Override
    public JobSeeker getJobSeekerById(long jobSeekerId) {
        JobSeeker jobSeeker = jobSeekerRepository.findById(jobSeekerId)
                .orElseThrow(() -> new RuntimeException("JobSeekar Id Not Found"));
        return jobSeeker;
    }

    @Override
    @Transactional
    public void updateJobSeeker(long id, JobSeeker jobSeeker) {
        JobSeeker updateJobSeeker = jobSeekerRepository.findById(id)
                .orElseThrow(() -> new RuntimeException("JobSeekar Id Not Found"));
        updateJobSeeker.setFullName(jobSeeker.getFullName());
        updateJobSeeker.setUserName(jobSeeker.getUserName());
        updateJobSeeker.setEmail(jobSeeker.getEmail());
        updateJobSeeker.setMobileNumber(jobSeeker.getMobileNumber());
        updateJobSeeker.setProfilePic(jobSeeker.getProfilePic());
        updateJobSeeker.setResume(jobSeeker.getResume());
        updateJobSeeker.setLocation(jobSeeker.getLocation());
        updateJobSeeker.setExperiance(jobSeeker.getExperiance());
```

Figure 28: Job Seeker Service IMPL

```java
package com.gsu.app.jobsplanet.exception;

import javax.servlet.http.HttpServletRequest;

import org.springframework.http.HttpStatus;
import org.springframework.web.bind.annotation.ControllerAdvice;
import org.springframework.web.bind.annotation.ExceptionHandler;
import org.springframework.web.bind.annotation.ResponseBody;
import org.springframework.web.bind.annotation.ResponseStatus;

@ControllerAdvice
public class ControllerExceptionHandler {

    @ExceptionHandler(ResourceNotFoundException.class)
    @ResponseStatus(value = HttpStatus.NOT_FOUND)
    public @ResponseBody ExceptionResponse handleResourceNotFound(final ResourceNotFoundException exception,
            final HttpServletRequest request) {

        ExceptionResponse error = new ExceptionResponse();
        error.setErrorMessage(exception.getMessage());
        error.setRequestedURI(request.getRequestURI());

        return error;
    }

    @ExceptionHandler(Exception.class)
    @ResponseStatus(value = HttpStatus.INTERNAL_SERVER_ERROR)
    public @ResponseBody ExceptionResponse handleException(final Exception exception,
            final HttpServletRequest request) {

        ExceptionResponse error = new ExceptionResponse();
        error.setErrorMessage(exception.getMessage());
        error.setRequestedURI(request.getRequestURI());

        return error;
    }

}

```

Figure 29: Controller Exception Handler

```
server.port=8082
spring.datasource.url=jdbc:mysql://localhost:3306/jobsplanet
spring.datasource.username=root
spring.datasource.password=root
spring.jpa.hibernate.ddl-auto=update

#max file and request size
spring.servlet.multipart.max-file-size=300MB

spring.servlet.multipart.file-size-threshold=1KB
uploadDir=/resources

#Multipart configuration

spring.servlet.multipart.max-request-size=2MB
# Enable multipart uploads

spring.servlet.multipart.enabled=true

```

Figure 30: Resource application properties

UI codes

```
</head>
<body>
    <!-- nav section -->
        <div id="nav-section">
        </div>
        <script>
            $(function(){
                $("#nav-section").load("templates/nav.html");
            })
        </script>
    <!-- nav section -->
    <div class="hero-wrap img" style="background-image: url(images/banner.jpg);">
      <div class="overlay"></div>
      <div class="container">
        <div class="row d-md-flex no-gutters slider-text align-items-center justify-content-center">
            <div class="col-md-10 d-flex align-items-center ftco-animate">
              <div class="text text-center pt-5 mt-md-5">
                <p class="mb-4">Find Job, Employment, and Career Opportunities</p>
                <h1 class="mb-5">The Eassiest Way to Get Your New Job</h1>
                    <div class="ftco-search my-md-5">
                      <div class="row">
                        <div class="col-md-12 nav-link-wrap">
                            <div class="nav nav-pills text-center" id="v-pills-tab" role="tablist" aria-orientation="vertical">
                              <a class="nav-link active mr-md-1" id="v-pills-1-tab" data-toggle="pill" href="#v-pills-1" role="tab" aria-controls="v-pills-1" aria-selected="true">Bas

                              <a class="nav-link" id="v-pills-2-tab" data-toggle="pill" href="#v-pills-2" role="tab" aria-controls="v-pills-2" aria-selected="false">Advance Job Searc

                            </div>
                          </div>
                          <div class="col-md-12 tab-wrap">

                            <div class="tab-content p-4" id="v-pills-tabContent">

                              <div class="tab-pane fade show active" id="v-pills-1" role="tabpanel" aria-labelledby="v-pills-nextgen-tab">
                                <form id="basicSearch_form" action="#" class="search-job">
                                    <div class="row no-gutters">
                                            <div class="col-md-8">
                                                <div class="form-group">
                                                    <div class="form-field">
                                                        <div class="select-wrap">
                                                        <div class="icon"><span class="ion-ios-arrow-down"></span></div>
                                                        <select class="form-control" id="job_type" name="job_type">
                                                            <option value="fulltime">Full-Time</option>
                                                            <option value="parttime">Part-Time</option>
                                                            <option value="regular_permanent">Regular/Permanet</option>
                                                            <option value="contract_temp">Contractual/Temprorary</option>
                                                            <option value="freelance">Freelance</option>
                                                            <option value="internship">Internship</option>
                                                            <option value="volunteer">Volunteer</option>
```

Figure 31: Index.html

```
</head>
<body>
    <input type="hidden" id="applicantloginStatus" name="applicantloginStatus" value="" />

    <!-- nav section -->
        <div id="nav-jobseeker">
        </div>
        <div id="nav-section">
        </div>
        <script>
            $(function(){
                $("#nav-section").load("dynamic_nav/jobseeker_nav.html");
            })
        </script>
    <!-- nav section -->
    <div class="hero-wrap img" style="background-image: url(images/banner.jpg);">
      <div class="overlay"></div>
      <div class="container">
        <div class="row d-md-flex no-gutters slider-text align-items-center justify-content-center">
            <div class="col-md-10 d-flex align-items-center ftco-animate">
              <div class="text text-center pt-5 mt-md-5">
                <p class="mb-4">Find Job, Employment, and Career Opportunities</p>
                <h1 class="mb-5">The Eassiest Way to Get Your New Job</h1>
                    <div class="ftco-search my-md-5">
                      <div class="row">
                        <div class="col-md-12 nav-link-wrap">
                            <div class="nav nav-pills text-center" id="v-pills-tab" role="tablist" aria-orientation="vertical">
                              <a class="nav-link active mr-md-1" id="v-pills-1-tab" data-toggle="pill" href="#v-pills-1" role="tab" aria-controls="v-pills-1" aria-selected="true">Job

                              <a class="nav-link" id="v-pills-2-tab" data-toggle="pill" href="#v-pills-2" role="tab" aria-controls="v-pills-2" aria-selected="false">Advance Job Searc

                            </div>
                          </div>
                          <div class="col-md-12 tab-wrap">

                            <div class="tab-content p-4" id="v-pills-tabContent">
                                <div class="tab-pane fade show active" id="v-pills-1" role="tabpanel" aria-labelledby="v-pills-nextgen-tab">
                                    <form id="basicSearch_form" action="#" class="search-job">
                                        <div class="row no-gutters">
                                                <div class="col-md-8">
                                                    <div class="form-group">
                                                        <div class="form-field">
                                                            <div class="select-wrap">
                                                            <div class="icon"><span class="ion-ios-arrow-down"></span></div>
                                                            <select class="form-control" id="job_type" name="job_type">
                                                                <option value="fulltime">Full-Time</option>
                                                                <option value="parttime">Part-Time</option>
                                                                <option value="regular_permanent">Regular/Permanet</option>
                                                                <option value="contract_temp">Contractual/Temprorary</option>
```

Figure 32: Job Seeker.html

```
</head>
<body>
    <!-- nav section -->
        <div id="nav-jobseeker">
        </div>
        <div id="nav-section">
        </div>
        <script>
            $(function(){
                $("#nav-section").load("dynamic_nav/recruiter_nav.html");
            })
        </script>
    <!-- nav section -->
    <div class="hero-wrap img" style="background-image: url(images/banner.jpg);">
      <div class="overlay"></div>
      <div class="container">
        <div class="row d-md-flex no-gutters slider-text align-items-center justify-content-center">
            <div class="col-md-10 d-flex align-items-center ftco-animate">
              <div class="text text-center pt-5 mt-md-5">
                <p class="mb-4">Welcome to the Jobs Planet Platform</p>
                <h1 class="mb-5">You are just few Click away to find right candidates for your Organisation.</h1>
              </div>
            </div>
            </div>
      </div>
    </div>

        <section class="ftco-section bg-light">
            <div class="container">
                <div class="row">
                    <div class="col-lg-12 pr-lg-5">
                        <div class="row justify-content-center pb-3">
                            <div class="col-md-12 heading-section ftco-animate">
                                <span class="subheading">Recently Added Jobs</span>
                                <h2 class="mb-4">Featured Jobs Posts</h2>
                            </div>
                        </div>
                        <div class="row">
                            <div class="col-md-6 ftco-animate">
                                <div class="job-post-item p-4 d-block d-lg-flex align-items-center">
                                <div class="one-third mb-4 mb-md-0">
                                    <div class="job-post-item-header align-items-center">
                                        <span class="subadge">Partime</span>
                                    <h2 class="mr-3 text-black"><a href="job-details.html">Frontend Development</a></h2>
                                    </div>
                                    <div class="job-post-item-body d-block d-md-flex">
                                    <div class="mr-3"><span class="icon-layers"></span> <a href="#">Facebook, Inc.</a></div>
                                    <div><span class="icon-my_location"></span> <span>Western City, UK</span></div>
                                    </div>
```

Figure 33: Recrutier.html

```
30    <script src="plugins/jquery/jquery.min.js"></script>
31  </head>
32  <body class="hold-transition sidebar-mini layout-fixed">
33  <div class="wrapper">
34
35    <!-- header menu section -->
36    <div id="header_menu_admin-section">
37    </div>
38    <script>
39      $(function(){
40        $("#header_menu_admin-section").load("templates/header_menu.html");
41      })
42    </script>
43    <!-- header menu section -->
44
45    <!-- navigation section -->
46    <div id="navigation_admin-section">
47    </div>
48    <script>
49      $(function(){
50        $("#navigation_admin-section").load("templates/navigation.html");
51      })
52    </script>
53    <!-- navigation section -->
54
55    <!-- Content Wrapper. Contains page content -->
56    <div class="content-wrapper">
57      <!-- Content Header (Page header) -->
58      <div class="content-header">
59        <div class="container-fluid">
60          <div class="row mb-2">
61            <div class="col-sm-6">
62              <h1 class="m-0 text-dark">Dashboard</h1>
63            </div><!-- /.col -->
64            <div class="col-sm-6">
65              <ol class="breadcrumb float-sm-right">
66                <li class="breadcrumb-item"><a href="#">Home</a></li>
67                <li class="breadcrumb-item active">Dashboard</li>
68              </ol>
69            </div><!-- /.col -->
70          </div><!-- /.row -->
71        </div><!-- /.container-fluid -->
72      </div>
73      <!-- /.content-header -->
74
75      <!-- Main content -->
76      <section class="content">
77        <div class="container-fluid">
78          <!-- Small boxes (Stat box) -->
```

Figure 34: Admin Index.html

```
button.close, button.mailbox-attachment-close {
  padding: 0;
  background-color: transparent;
  border: 0;
  appearance: none;
}

a.close.disabled, a.disabled.mailbox-attachment-close {
  pointer-events: none;
}

.mailbox-messages > .table {
  margin: 0;
}

.mailbox-controls {
  padding: 5px;
}

.mailbox-controls.with-border {
  border-bottom: 1px solid rgba(0, 0, 0, 0.125);
}

.mailbox-read-info {
  border-bottom: 1px solid rgba(0, 0, 0, 0.125);
  padding: 10px;
}

.mailbox-read-info h3 {
  font-size: 20px;
  margin: 0;
}

.mailbox-read-info h5 {
  margin: 0;
  padding: 5px 0 0;
}

.mailbox-read-time {
  color: #999;
  font-size: 13px;
}

.mailbox-read-message {
  padding: 10px;
}

.mailbox-attachments {
  padding-left: 0;
```

Figure 35: Admin.page.css

```
$(document).on('click', '.contactUs_delete', function(e){
    "use strict";
    if (confirm('Are You sure ?'))
    {
        var id = $(this).attr("id");
            var myrequest = $.ajax({
                    url:'http://localhost:8082/contactus/deletebyId/'+id,
                    type: "DELETE",
                    dataType: 'json',
                    async:false,
                    processData: false,
                    headers: {
                        "Content-Type": "application/json",
                        Accept: "application/json",
                    },
            });

            alert('Thanks for Confirming......! Your Data has been Deleted successfully!');
            window.location.replace("http://127.0.0.1:5501/dashboard/admin/admin_contactUs.html");
    }
    else{
        alert('Thanks For Cancel.');
    }
});

function get_data_with_html_for_contactUs(response)
{
    var txt="";
    var pre_html=`<table id="example1" class="table table-bordered table-striped">
                    <thead>
                    <tr>
                    <th>#</th>
                    <th>Company Name</th>
                    <th>Rating</th>
                    <th>Message</th>
                    <th>Action</th>
                    </tr>
                    </thead>
                <tbody>`;

    var post_html=`</tbody>
                    </table>`;
    for(let i in response)
    {

        let j= parseInt(i)+1;

        txt+=`<tr>
            <td>1</td>
```

Figure 36: Admin_companyReview.js