Project - JOB PORTAL

Company Name: Coincent

Website Name: Coincent.ai

Name: MA Shaik Shoyeb

TABLE OF CONTENTS

1. ABSTRACT

The purpose of Online Job Portal is to automate the existing manual system by the help of computerized equipments and full-fledged computer software, fulfilling their requirements, so that their valuable data/information can be stored for a longer period with easy accessing and manipulation of the same. The required software and hardware are easily available and easy to work with.

Online Job Portal, as described above, can lead to error free, secure, reliable and fast management system. It can assist the user to concentrate on their other activities rather to concentrate on the record keeping. Thus it will help organization in better utilization of resources. The organization can maintain computerized records without redundant entries. That means that one need not be distracted by information that is not relevant, while being able to reach the information.

The aim is to automate its existing manual system by the help of computerized equipments and full-fledged computer software, fulfilling their requirements, so that their valuable data/information can be stored for a longer period with easy accessing and manipulation of the same. Basically the project describes how to manage for good performance and better services for the clients.

2. OBJECTIVE

The main objective of the Project or Online Job Portal is to manage the details of Job, Vacancy, Resume, Jobseeker, Interview. It manages all the information about Job, Call Later, Interview, Job. The project is totally built at administrative end and thus only the administrator is guaranteed the access. The purpose of the project is to build an application program to reduce the manual work for managing the Job, Vacancy, Call Later, Resume. It tracks all the details about the Resume, Jobseeker, Interview.

Functionalities provided by on online portal are as follow:

- Provides the searching facilities based on various factors. Such as Job, Resume, Jobseeker, Interview
- Online Job Portal also manage the Call Later details online for Jobsekeer details Interview details, Job
- It tracks all the information of Vacancy, Call Later, Jobsekeer etc
- Manage the information of Vacancy
- Shows the information and description of the Job, Resume.
- To increase efficiency of managing the job, vacancy
- It deals with monitoring the information and transaction of jobseeker.
- Manage the information of job.
- Editing, Adding and Updating of records is improved which results in proper resource management of job data.
- Manage the information of jobseeker.
- Integration of all records of interview.

3. INTRODUCTION

The present system requires applicants to search through print and visual media for job opportunities. Applicants need to apply for jobs using conventional methods and appear for interview on a specified date at a specified location. Employers need to advertise the vacancies and sort all applicant details, conduct selection procedures and complete the formalities. This approach is tedious and requires much effort and resources..

Proposed System of Online Job Portal:

The aim of proposed system is to develop a system of improved facilities. The proposed system can overcome all the limitations of the existing system. The system provides proper security and reduces the manual work.

- Security of data.
- Ensure data accuracy's
- Proper control of the higher oficials.
- Minimize manual data entry
- Minimum time needed for the various processing
- Greater efficiency.
- Better service.
- User friendliness and interactive.
- Minimum time required.

Existing System of Online Job Portal:

In the existing system the exams are done only manually but in proposed system we must computerize the exams using this application.

- Lack of security of data
- More man power.
- Time consuming.
- Consumes large volume of pare work.
- Needs manual calculations.
- No direct role for the higher officials

4. METHODOLOGY

Implementation Methodology:

Model View Controller or MVC as it is popularly called, is a software design pattern for developing web applications. A Model View Controller pattern is made up of the following three parts:

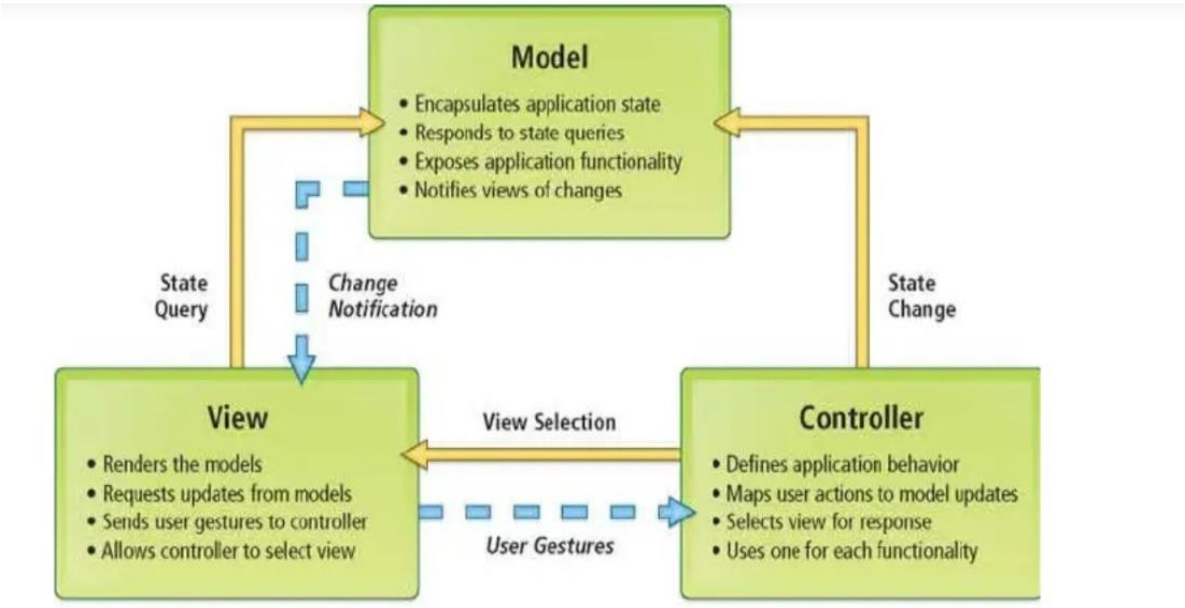
Model - The lowest level of the pattern which is responsible for maintaining data

View - This is responsible for displaying all or a portion of the data to the user

Controller - Software Code that controls the interactions between the Model and View.

MVC is popular as it isolates the application logic from the user interface layer and supports separation of concerns. Here the Controller receives all requests for the application and then works with the Model to prepare any data needed by the View. The View then uses the data prepared by th e Controller to generate a final presentable response. The MVC abstraction can be graphically represented as follows.

MVC (Model View Controller Flow) Diagram



Modules:

The application comprises the following major modules:

- **Job Seeker module**
- **Employer module**
- **Administration module**

Job Seeker module: vacancies available. Job seekers can also get mail alerts when their resumes are selected by employers vacancies available. Job seekers can also get mail alerts when their resumes are selected by employers.

Employer module: This module provides functionalities related to employers. Employers can post vacancy details and update the details as and when necessary. Employers can search through applicant resumes based on different criteria.

Administration module: This module provides administrator related functionalities. Administrator manages entire application and maintains the profiles of applicants and employers.

Technical Specifications:

This section explains the main features, technologies, tools and the process involved in building the online job portal web application

- PHP
- CSS
- MYSQL

PHP (Hypertext Pre-processor)

PHP (PHP, Hypertext Pre-processor) is a scripting language originated in 1995 by a software development contractor named Rasmus Lerdorf. It has developed as one of the best larger scripting languages around the globe . PHP is a server-side scripting language that was introduced for developing dynamic web applications. PHP code is embedded into HTML source file with PHP tags and interpreted by web server. lts syntax is a mixture of C, Java and Perl languages. PHP became very famous among developers because of its easy connectivity to MySQL databases that leads to creating dynamic web sites. PHP Can be used from command line interface or in standalone graphical applications. It can be deployed on any server and any operating system easily. Figure shows the syntax of PHP code in an HTML file. PHP starting and ending tags makes it available anywhere in the page to use it as scripting language.

CSS (Cascading Style Sheets)

CSS (Cascading Style Sheet) is a style sheet language use for styling HTML elements on a webpage. It defines the presentation of HTML elements, how they appear on web page,including designs, layouts, fonts and tailor pages on different environment. Figure explains the syntax of a CSS file with the properties of HTML tags.

MYSQL (My Structured Query Language)

A database is a collection of useful information that can be easily accessed, modified, managed, and updated. It is use to provide efficient retrieval. The data, which is stored in a database, can be in any format. MySQL is an open source. Relational Database Management System (RDBMS) that uses Structured Query Language (SQL). The main features of using MySQL that it is very user friendly, easy to use, quickly processing and flexible Because it is a relational database, it stores data in different tables making it possible to use same data flexibly in different ways. MySQL server controls the access to project data and makes sure that multiple users can work parallel with the same data with fast and easy access. It also restricts to the authorized users who can have access to the data.

Software & Hardware Requirements:

Software Requirements:

Name of the Component	Specification
Operating System	Windows 98 Windows7, Linux
Language	Java 2 Runtime Environment
Database	MySQL Server
Browser	Any of Mozilla, Opera, Chrome..
Web Server	Tomcat 7
Software Development Kit	Java JDK 1.7 or Above
Scripting Language Enable	JSP (Java Server Pages)
Database JDBC Driver	MySQL Jconnector

Hardware Requirements:

Name of the Component	Specification
Processor	Pentium Ill 630MHZ
RAM	128 MB
Hard disk	20 GB
Monitor	15" color monitor
Keyboard	122 keys

5.CODE

```
<?php
@file
* Enables multi-user blogs.
*/
*
/**
* Implements hook_node_info().
*/
function blog_node_info() {
return array(
'blog' => array(
)
);
}
```

Source Code of the Project

```
'name' => t('Blog entry'),
'base' => 'blog',
'description' => t('Use for multi-user blogs. Every user gets a personal blog.'),
/**
* Implements hook_user_view().
*/
function blog_user_view($account) {
if (user_access('create blog content', $account)) {
$account->content['summary']['blog'] = array(
'#type' => 'user_profile_item',
'#title' => t('Blog'),
// 1() escapes the attributes, so we should not escape lusername here.
'#markup' => I(t('View recent blog entries"), "blog/$account->uid", array('attributes' => array('title' =>
```

```
t("Read !username's latest blog entries.", array('username' => format_username($account)))),
'#attributes' => array('class' => array('blog')),
);
}
}
Implements hook_help().
*/
function blog_help($path, $arg) {
switch ($path) {
case 'admin/help#blog':
$output= '<h3>'.t('About').'</h3>';
$output= '<p>' .t("The Blog module allows registered users to maintain an online journal, or
<em>blog</em>. Blogs are made up of individual <em>blog entries</em>. By default, the blog entries
are displayed by creation time in descending order, with comments enabled, and are promoted to the
site's front page. For more information, see the online handbook entry for <a href='@blog'>Blog
module</a>.", array('@blog' => 'http://drupal.org/documentation/modules/blog/')).'</p>';
$output.= '<h3>'.t('Uses') . '</h3>';
$output .= '<dl>';
$output.= '<dt>'. t('Single-user blogs'). '</dt>';
$output.= '<dd>'. t("Each user's blog entries are automatically displayed with a link to the user's
main blog page. You can create as many single-user blogs as you have site users with permission to
create blog content.").'</dd>';
$output.= '<dt>'.t('Multi-user blogs'). '</dt>';
$output= '<dd>' .t("Blog entries from each single-user blog are also aggregated into one central
multi-user blog, which displays the blog content of all users in a single listing."). '</dd>';
$output.= '<dt>' . t('Navigation'). '</dt>';
$output= '<dd>' .t("There is an optional <em>Blogs</em> menu item added to the Navigation
```

menu, which displays all blogs available on your site, and a My blog item displaying the

current user's blog entries."). '</dd>';

$output.= '<dt>'. t('Blocks') . '</dt>';

$output= '<dd>' .t('The Blog module also creates a default Recent blog posts block that

may be enabled at the blocks administration page.', array('@blocks' =>

url('admin/structure/block'))) . '</dd>';

$output.= '</dl>';

return $output;

}

}

/**

*

* Implements hook_form().

*/

function blog_form($node, $form_state) {

return node_content_form($node, $form_state);

}

/**

* Implements hook_view().

*/

function blog_view($node, $view_mode) {

if ($view_mode == 'full' && node_is_page($node)) {

// Breadcrumb navigation. I() escapes title, so we should not escape !name.

drupal_set_breadcrumb(array(l(t('Home'), NULL), I(t('Blogs'), 'blog'), I(t("Iname's blog", array('!name"

=> format_username($node))), 'blog/' . $node->uid)));

}

return $node;

}

```
* Implements hook_node_view().
*/
function blog_node_view($node, $view_mode) {
if ($view_mode != 'rss') {
if ($node->type == 'blog' && (arg(0) != 'blog' || arg(1) != $node->uid)) {
// This goes to I(), which escapes lusername in both title and attributes.
$links['blog_usernames_blog'] = array(
'title' => t("!username's blog", array('username' => format_username($node))),
'href' => "blog/$node->uid",
'attributes' => array('title' => t("Read lusername's latest blog entries.", array('lusername' =>
format_username($node)))),
);
$node->content["links']['blog'] = array(
'#theme' => 'links_node_blog',
}
}
}
);
'#links' => $links,
'#attributes' => array('class' => array('links', 'inline')),
/**
*
*/
Implements hook_menu().
function blog_menu(){
$items['blog'] = array(
);
'title' => 'Blogs',
'page callback' => 'blog_page_last',
'access arguments' => array('access content'),
```

```
'type' => MENU_SUGGESTED_ITEM,
'file' => 'blog.pages.inc',
$items['blog/%user_uid_optional'] = array(
'title' => 'My blog',
'page callback' => 'blog_page_user',
'page arguments' => array(1),
'access callback' => 'blog_page_user_access',
'access arguments' => array(1),
'file' => 'blog.pages.inc',
}
);
$items['blog/%user/feed'] = array(
'title' => 'Blogs',
'page callback' => 'blog_feed_user',
'page arguments' => array(1),
'access callback' => 'blog_page_user_access',
'access arguments' => array(1),
'type' => MENU_CALLBACK,
'file' => 'blog.pages.inc',
);
$items['blog/feed'] = array(
'title' => 'Blogs',
'page callback' => 'blog_feed_last',
'access arguments' => array('access content'),
'type' => MENU_CALLBACK,
'file' => 'blog.pages.inc',
);
return $items;
/**
* Implements hook_menu_local_tasks_alter().
```

```
*/
function blog_menu_local_tasks_alter(&$data, $router_item, $root_path) {
global $user;
// Add action link to 'node/add/blog' on 'blog' page.
if ($root_path == 'blog') {
}
$item = menu_get_item('node/add/blog');
if ($item['access']) {
}
$item['title'] = t('Create new blog entry');
$data['actions']['output'][] = array(
);
'#theme' => 'menu_local_action',
'#link' => $item,
// Provide a helper action link to the author on the 'blog/%' page.
elseif ($root_path == 'blog/%' && $router_item['page_arguments'][0]->uid == $user->uid) {
$data['actions']['output']['blog'] = array(
);
'#theme' => 'menu_local_action',
if (user_access('create blog content')){
}
$data['actions']['output']['blog']['#link']['title'] = t('Post new blog entry.');
$data['actions']['output']['blog']['#link']['href'] = 'node/add/blog';
else {
$data['actions']['output']['blog']['#link']['title'] = t('You are not allowed to post a new blog entry.');
}
}
}
*
*/
```

```
Access callback for user blog pages.
function blog_page_user_access($account) {
// The visitor must be able to access the site's content.
// For a blog to 'exist' the user must either be able to
// create new blog entries, or it must have existing posts.
return $account->uid && user_access('access content') && (user_access('create blog content',
$account) || _blog_post_exists($account));
}
/**
* Helper function to determine if a user has blog posts already.
*/
function_blog_post_exists($account) {
return (bool)db_select('node', 'n')
->fields('n', array('nid'))
->condition('type', 'blog')
->condition('uid', $account->uid)
->condition('status', 1)
->range(0, 1)
->addTag('node_access')
->execute()
}
->fetchField();
/**
* Implements hook_block_info().
*/
function blog_block_info(){
}
$block['recent']['info'] = t('Recent blog posts');
$block['recent']['properties']['administrative'] = TRUE;
return $block;
* Implements hook_block_configure().
```

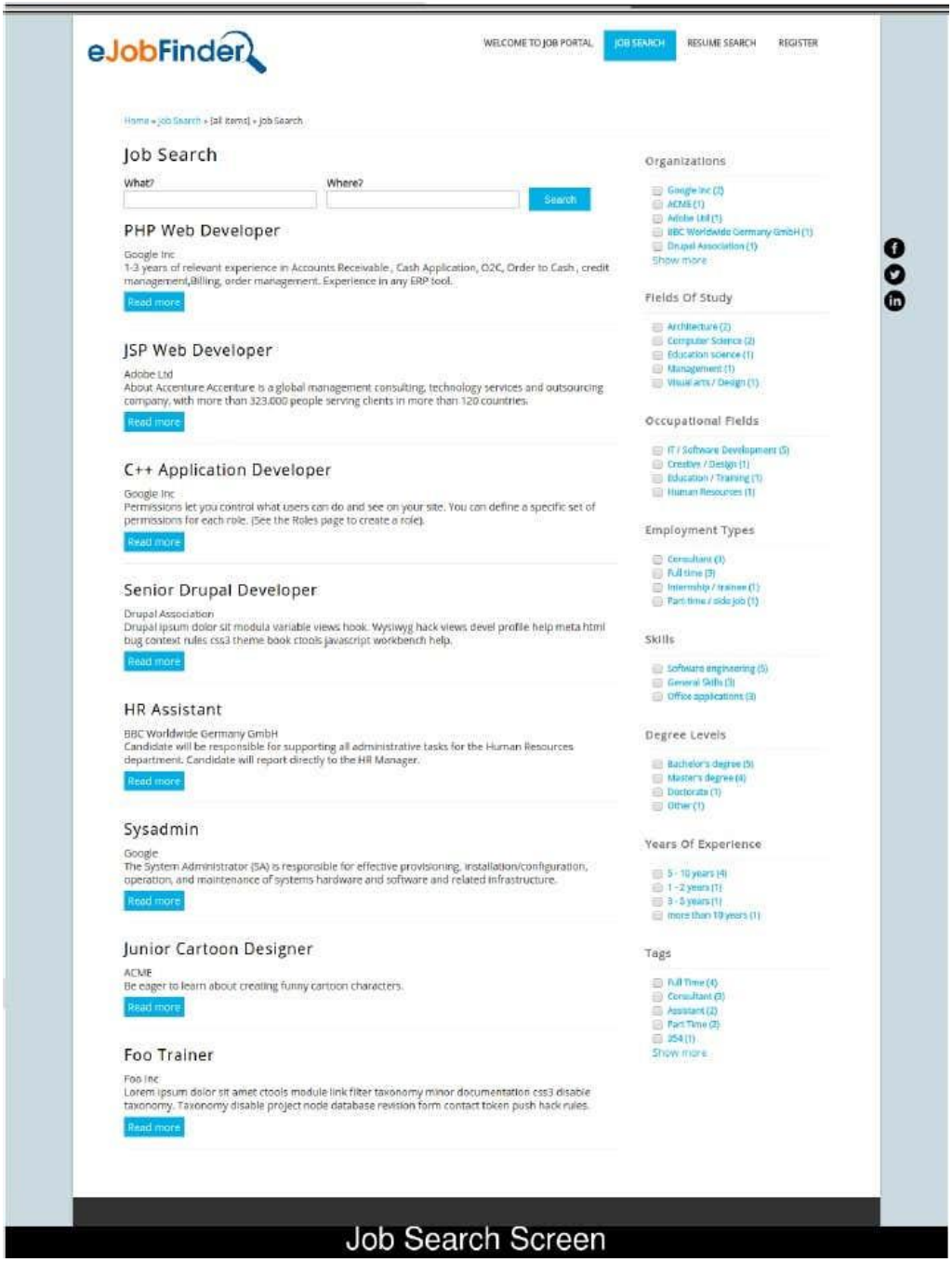
```
*/
function blog_block_configure($delta = "") {
if ($delta == 'recent'){
$form['blog_block_count'] = array(
'#type' => 'select',
'#title' => t('Number of recent blog posts to display'),
'#default_value' => variable_get('blog_block_count', 10),
'#options' => drupal_map_assoc(array(2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15, 16, 17, 18,
19, 20, 25, 30)),
);
return $form;
}
}
/**
*
*/
Implements hook_block_save().
function blog_block_save($delta = '', $edit = array()) {
if ($delta == 'recent'){
variable_set('blog_block_count', $edit['blog_block_count']);
}
}
/**
*Implements hook_block_view().
* Displays the most recent 10 blog titles.
*/
function blog_block_view($delta = '') {
global $user;
if (user_access('access content')){
$result = db_select('node', 'n')
->fields('n', array('nid', 'title', 'created'))
```

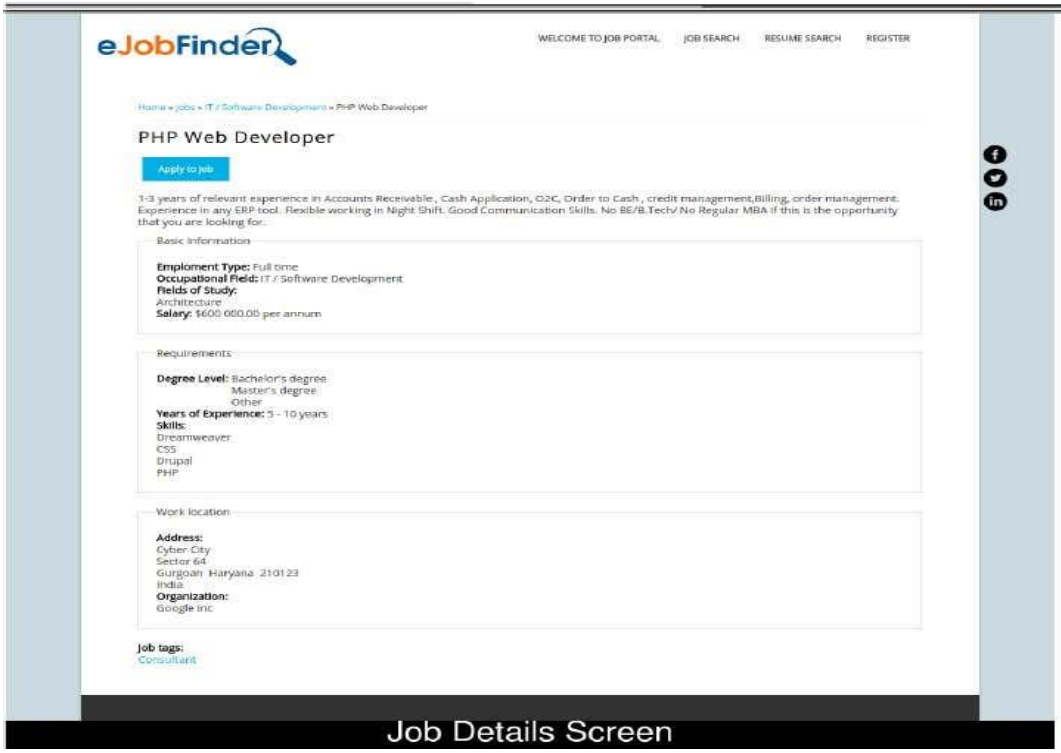
```
->condition('type', 'blog')
->condition('status', 1)
->orderBy('created', 'DESC')
}
}
>>range(0, variable_get('blog_block_count', 10))
->addTag('node_access')
->execute();
if ($node_title_list = node_title_list($result)) {
$block['subject'] = t('Recent blog posts');
$block['content']['blog_list'] = $node_title_list;
$block['content']['blog_more'] = array(
);
'#theme' => 'more_link',
'#url' => 'blog',
'#title' => t('Read the latest blog entries.'),
}
return $block;
/**
* Implements hook_rdf_mapping().
*/
function blog_rdf_mapping(){
return array(
array(
'type' => 'node',
),
);
}
'bundle' => 'blog',
'mapping' => array(
```

```
),
'rdftype' => array('sioc:Post', 'sioct:BlogPost'),
```

6.CONCLUSION

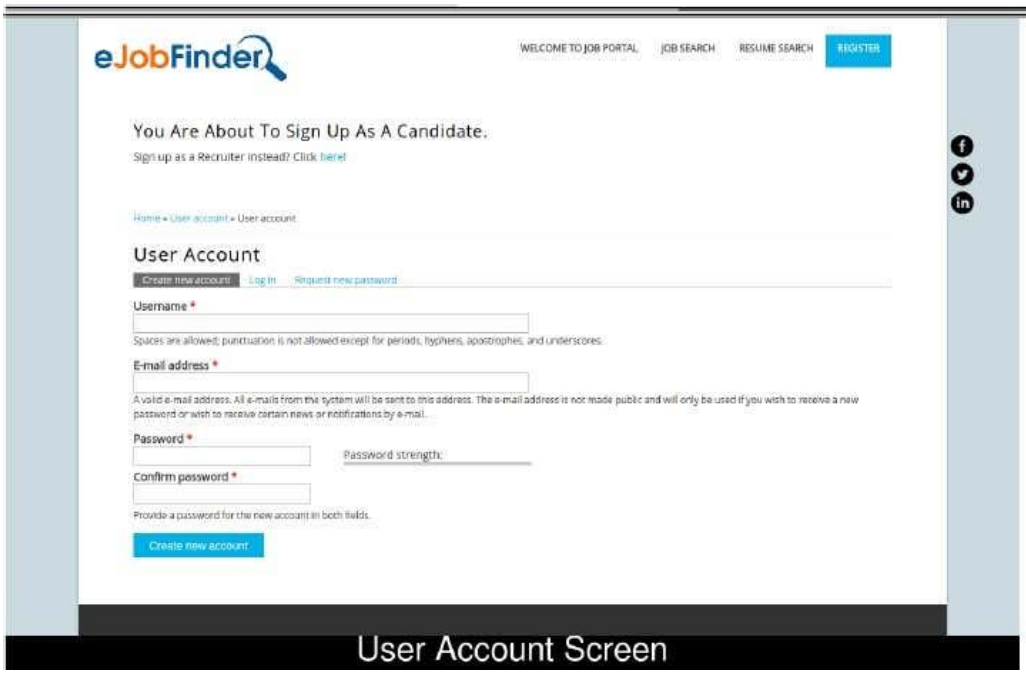
(SRCEEN SHOT OF RESULT AND OUTPUT)



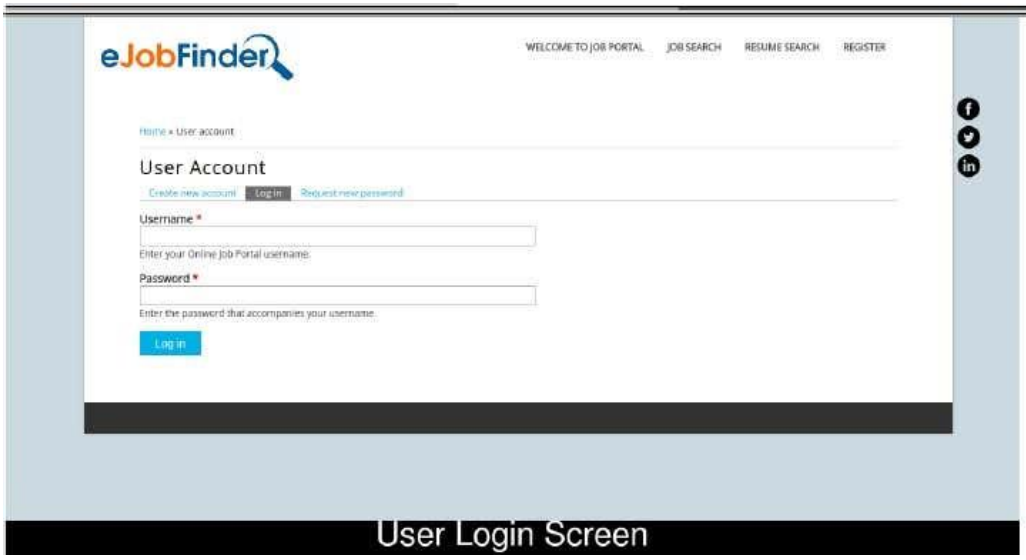


Job Details Screen

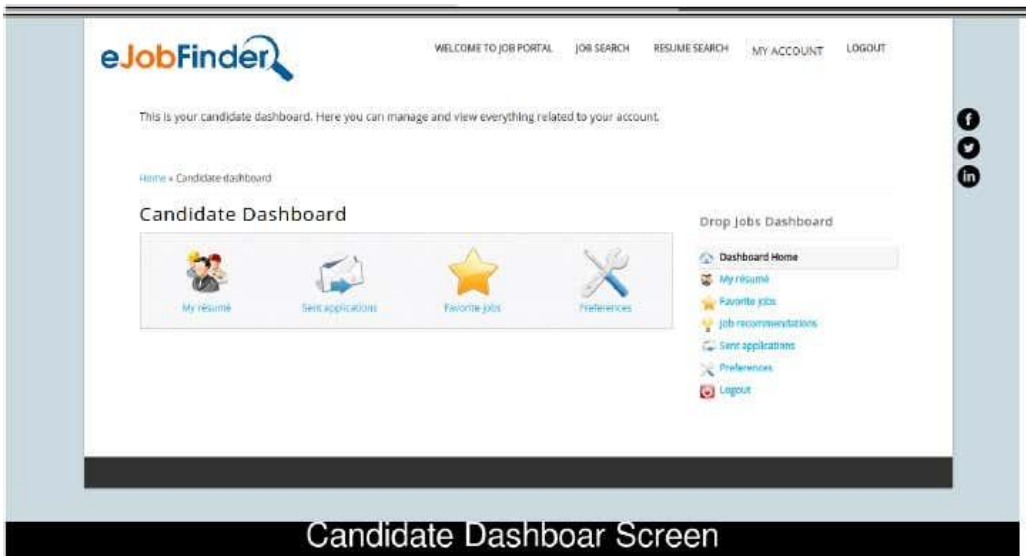
Apply to Job Screen



User Account Screen

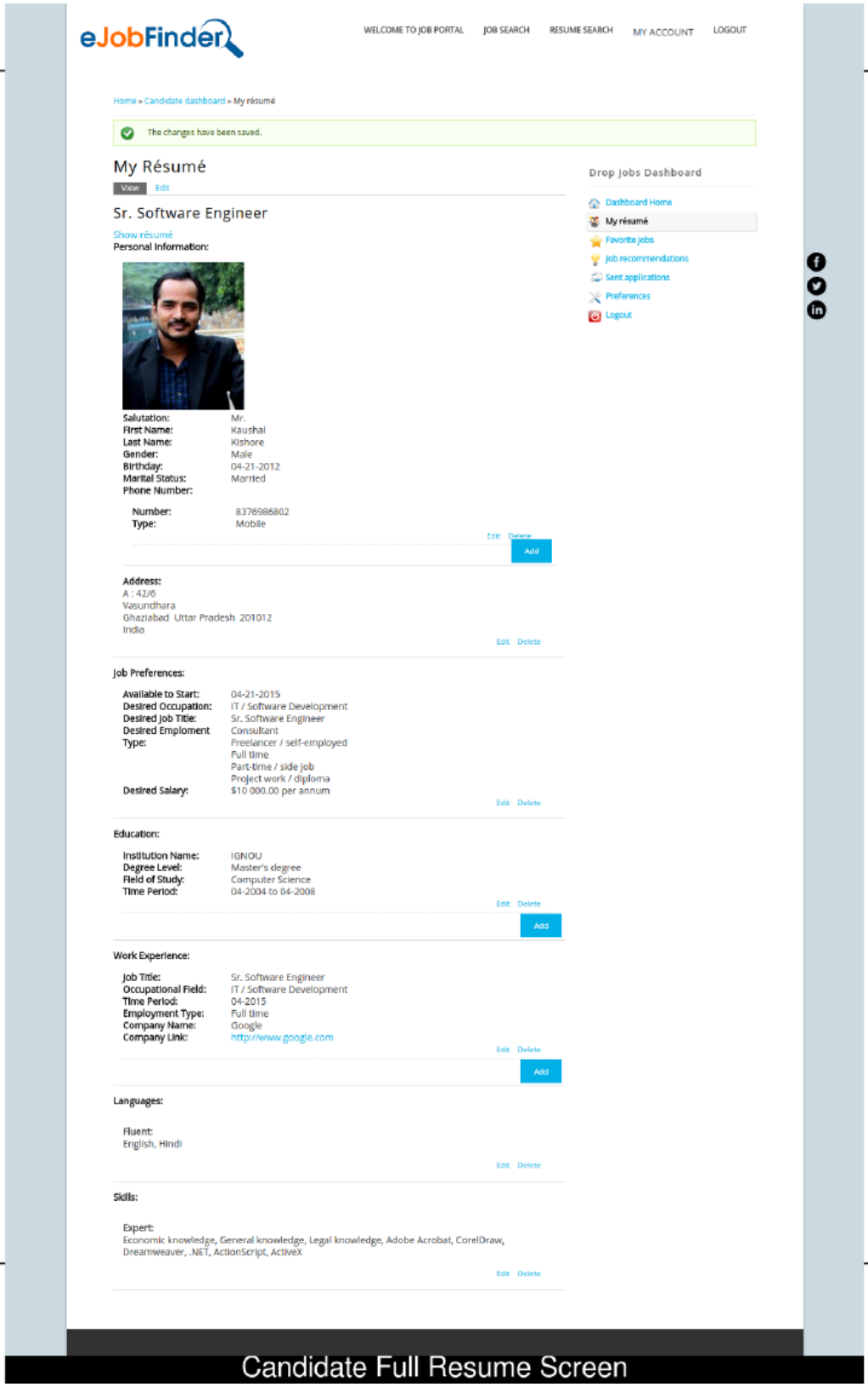


User Login Screen

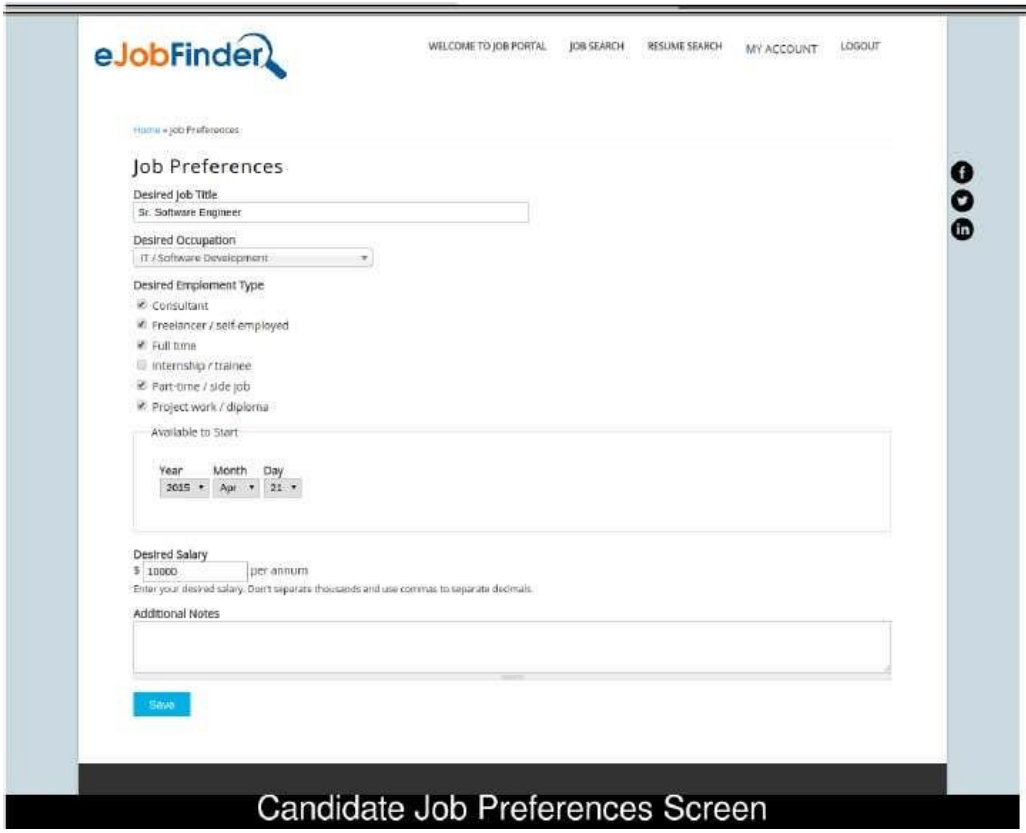


Candidate Dashboar Screen

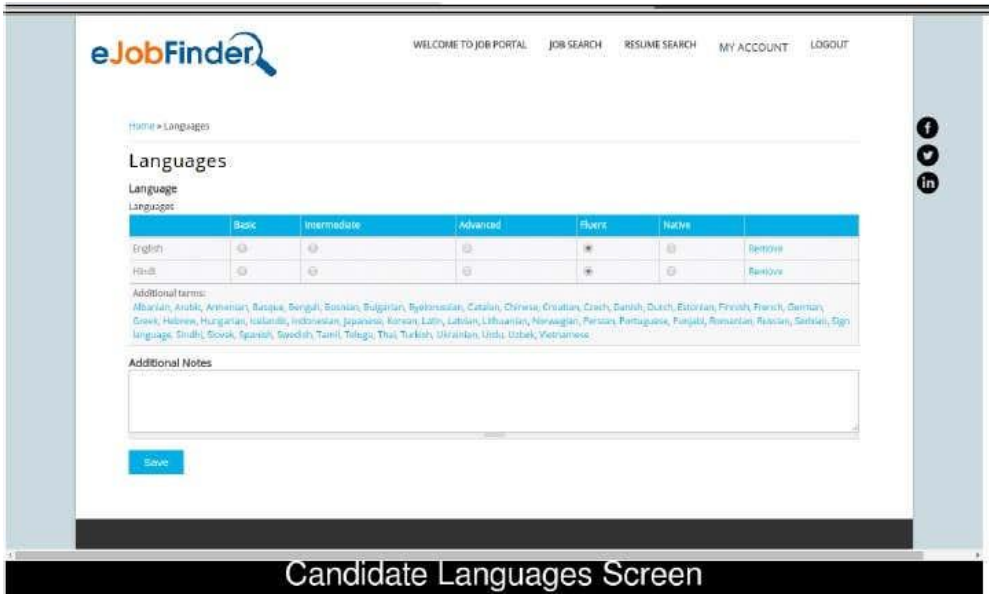
Candidate Favorite jobs Screen



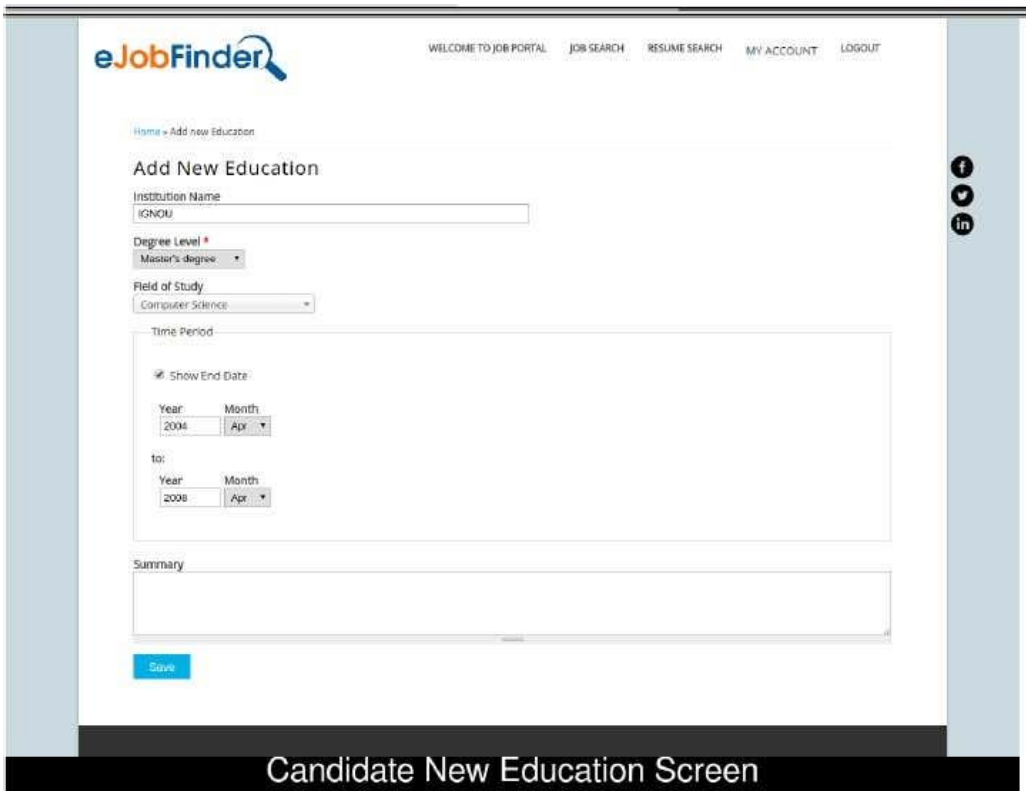
Candidate Full Resume Screen



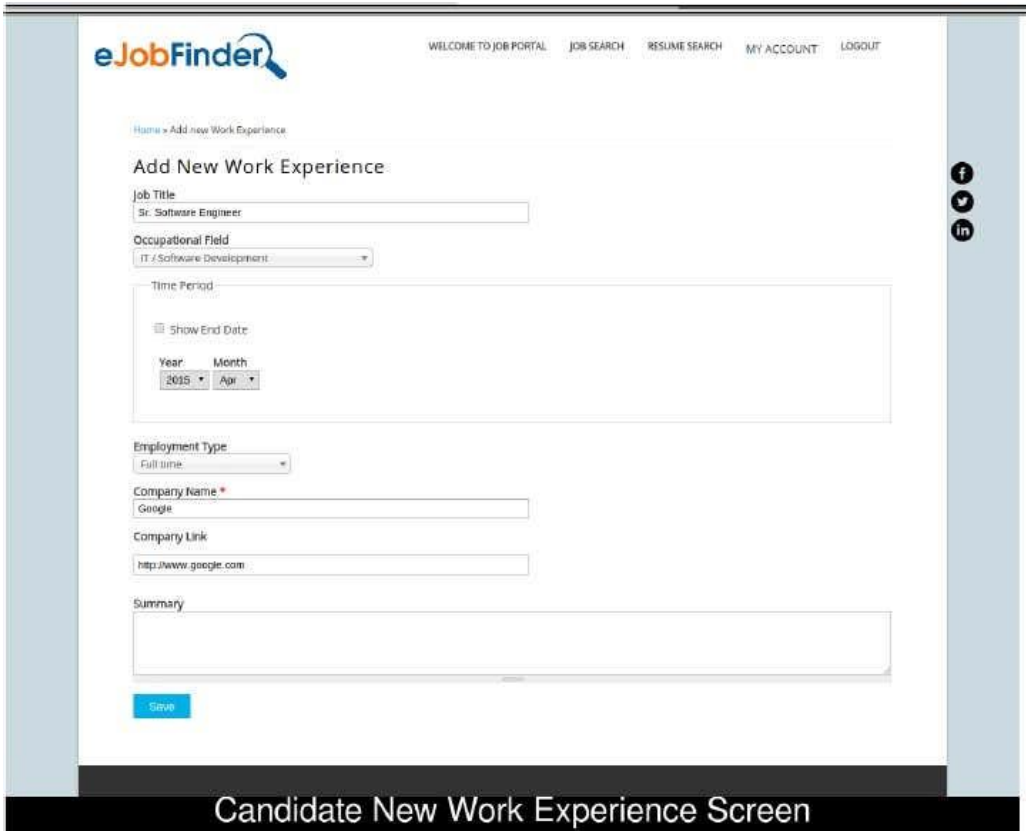
Candidate Job Preferences Screen



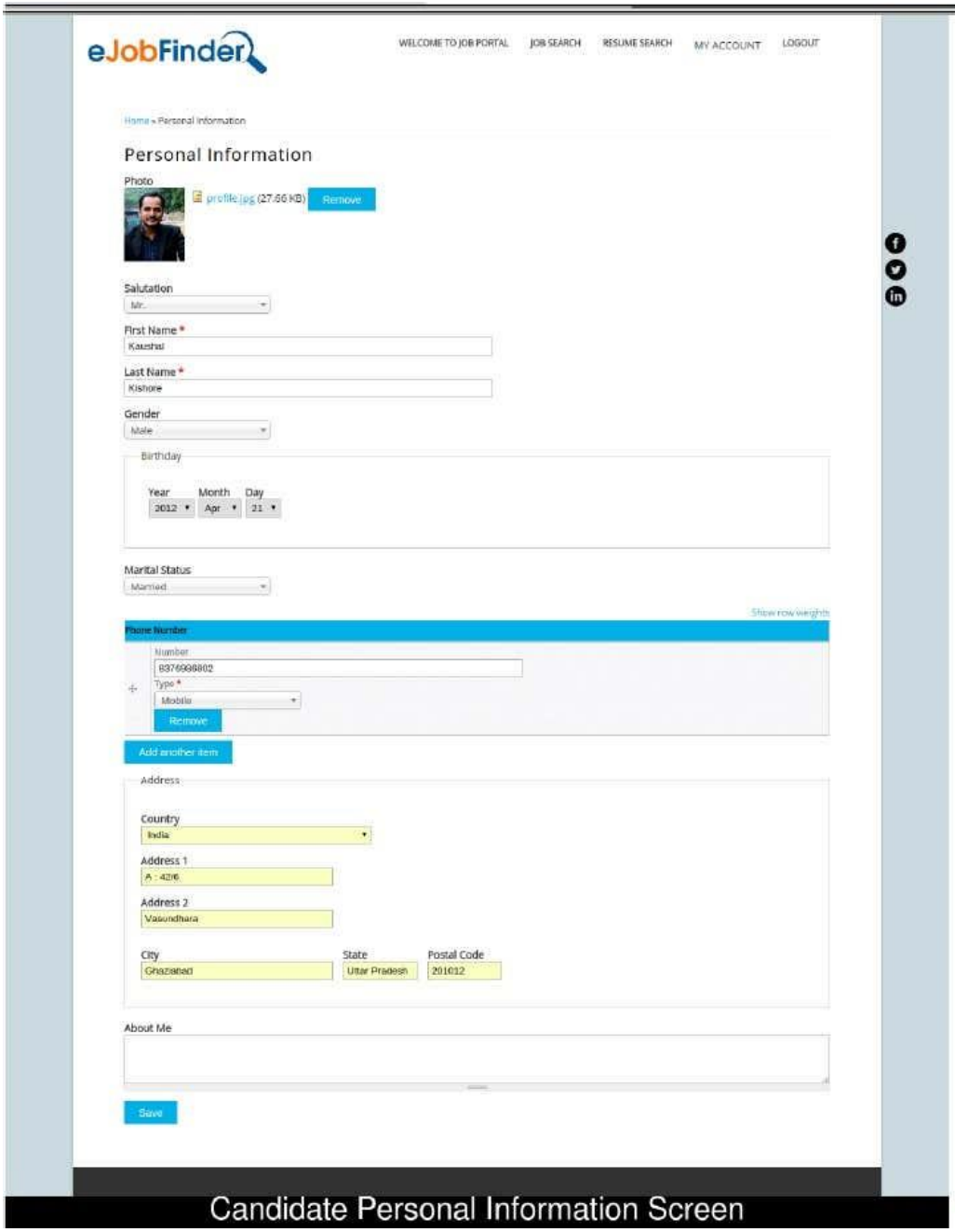
Candidate Languages Screen



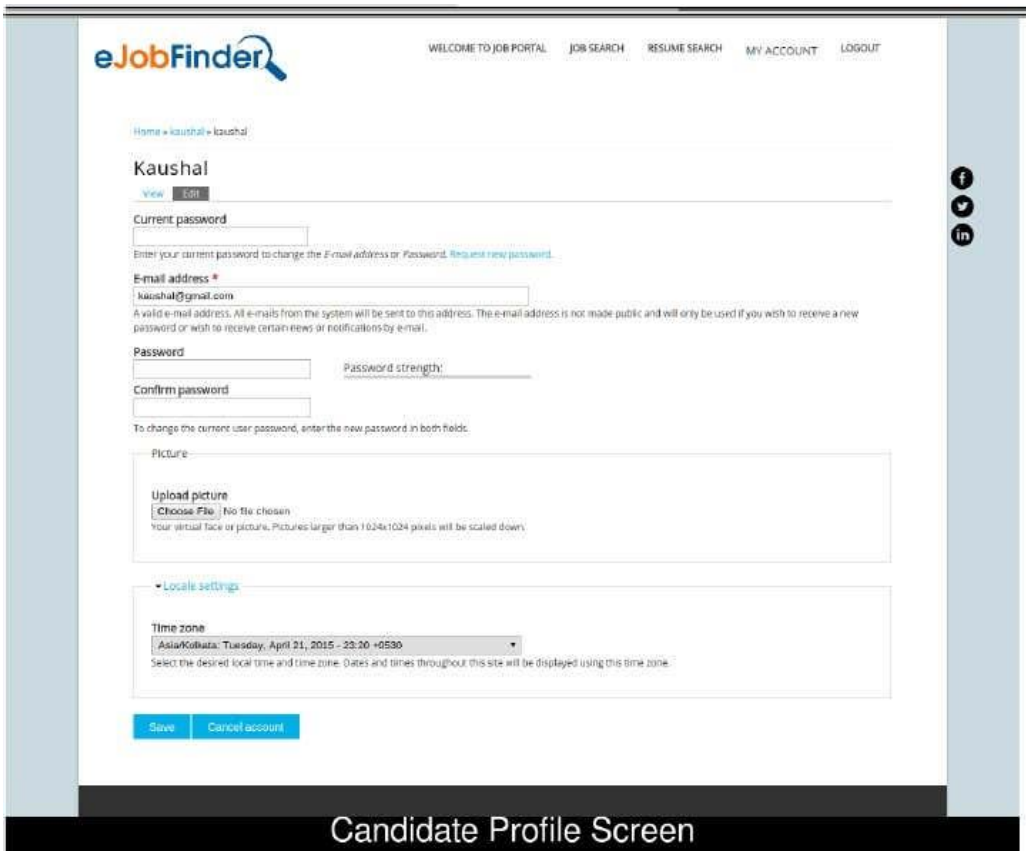
Candidate New Education Screen



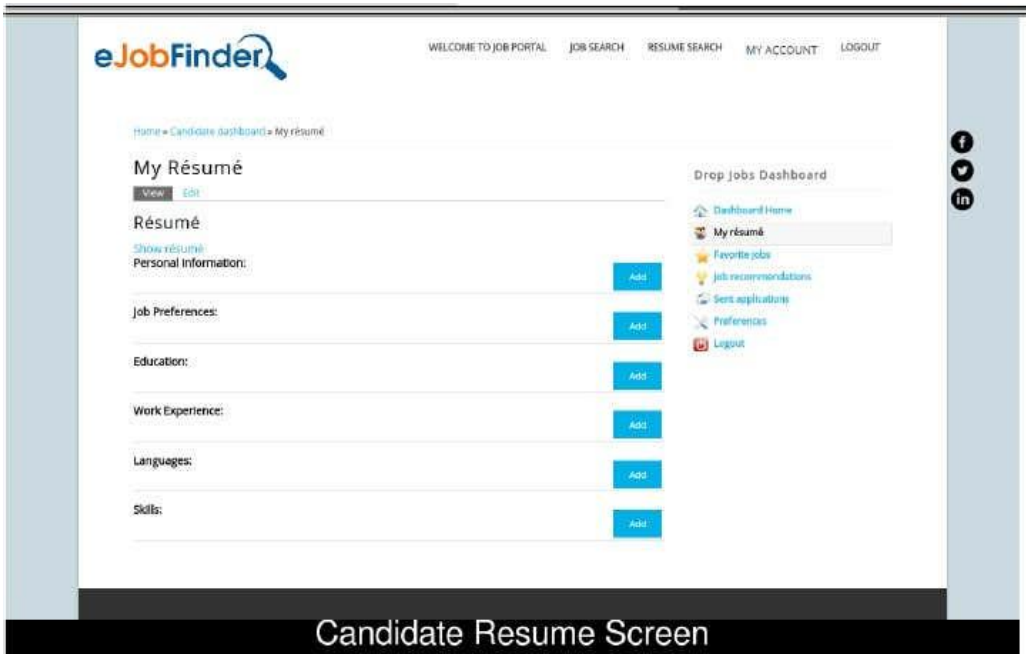
Candidate New Work Experience Screen



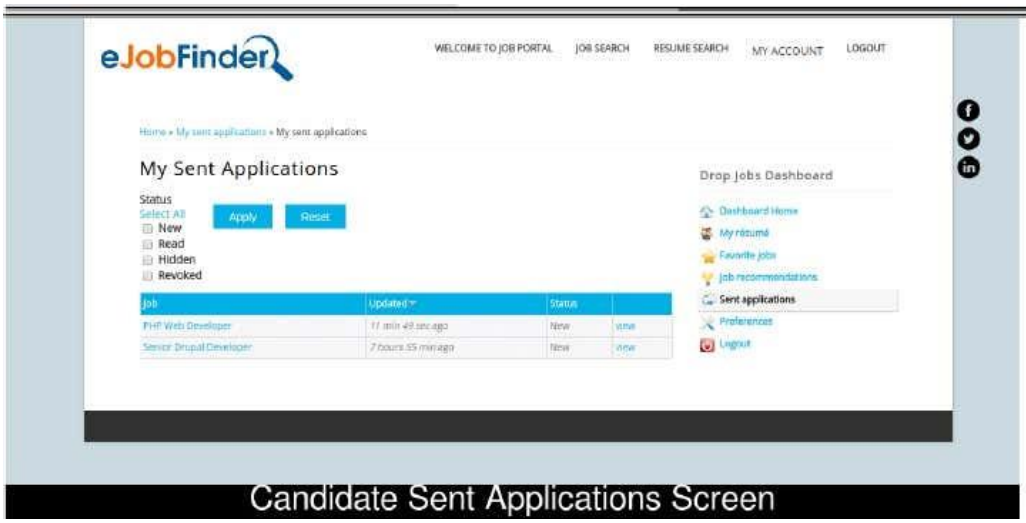
Candidate Personal Information Screen



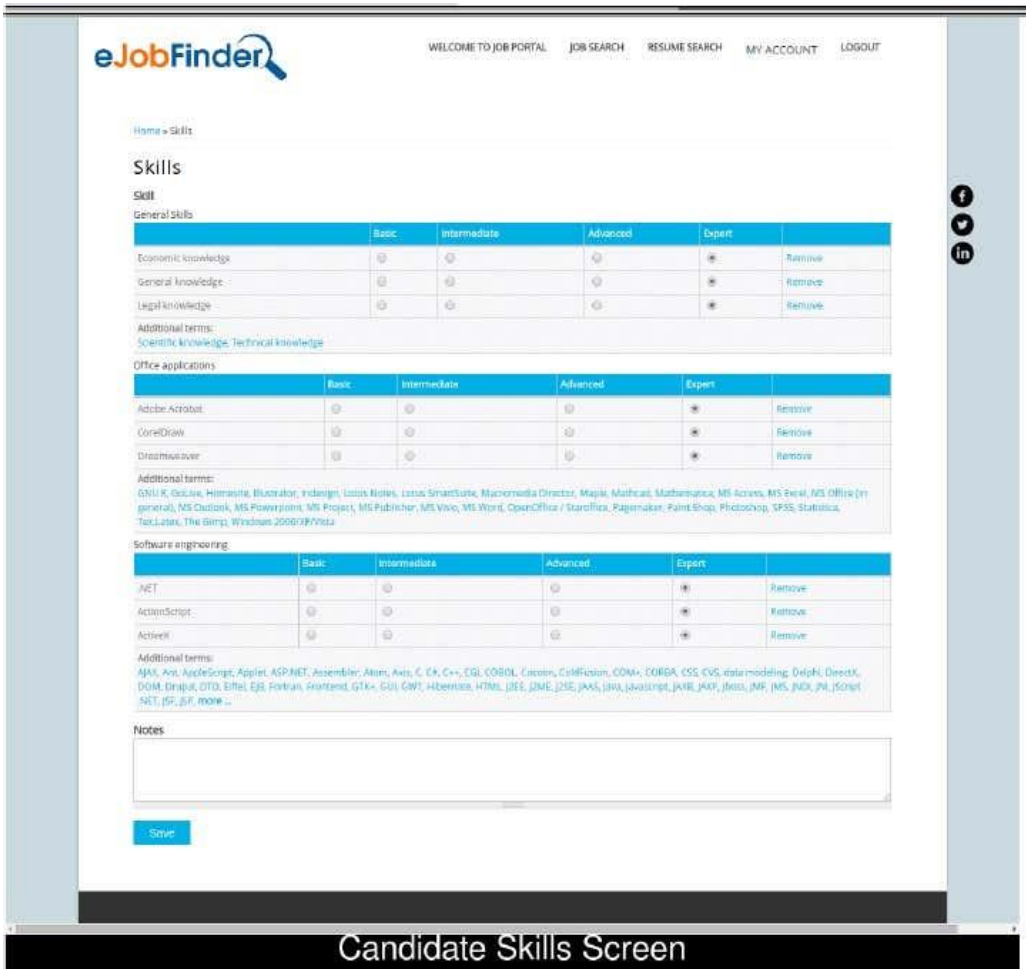
Candidate Profile Screen



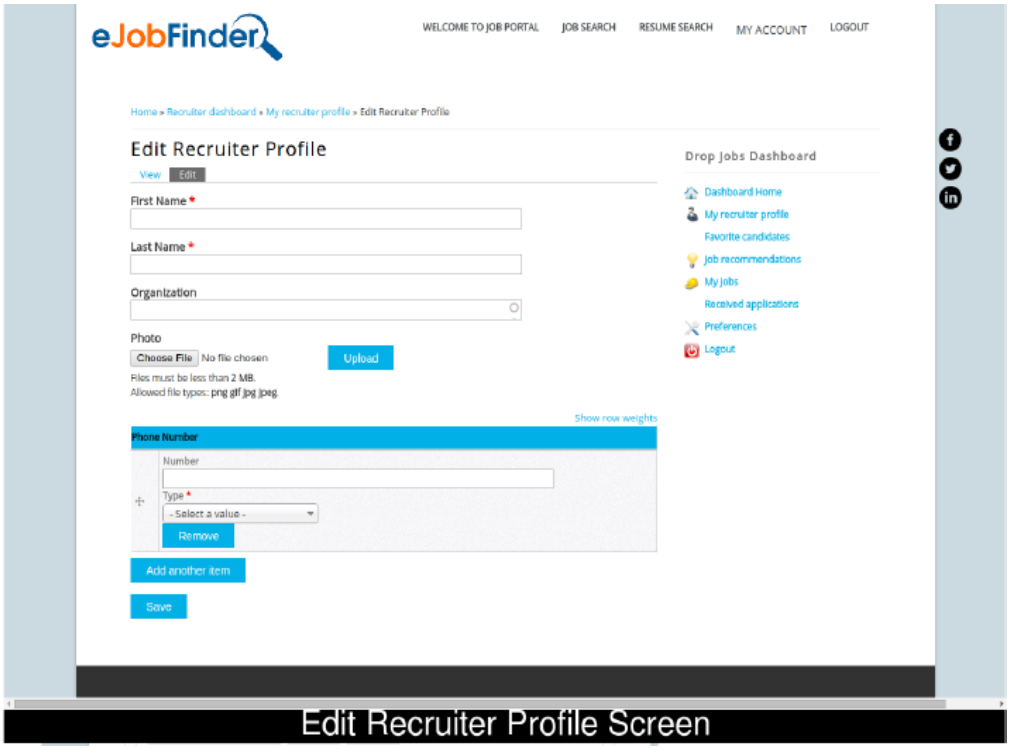
Candidate Resume Screen



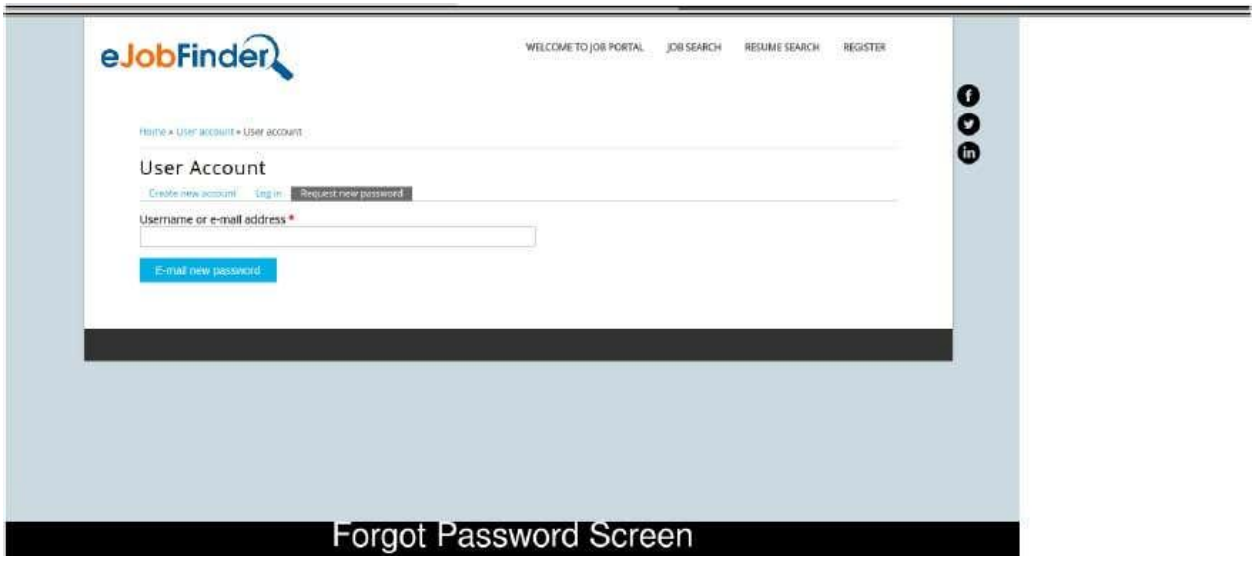
Candidate Sent Applications Screen



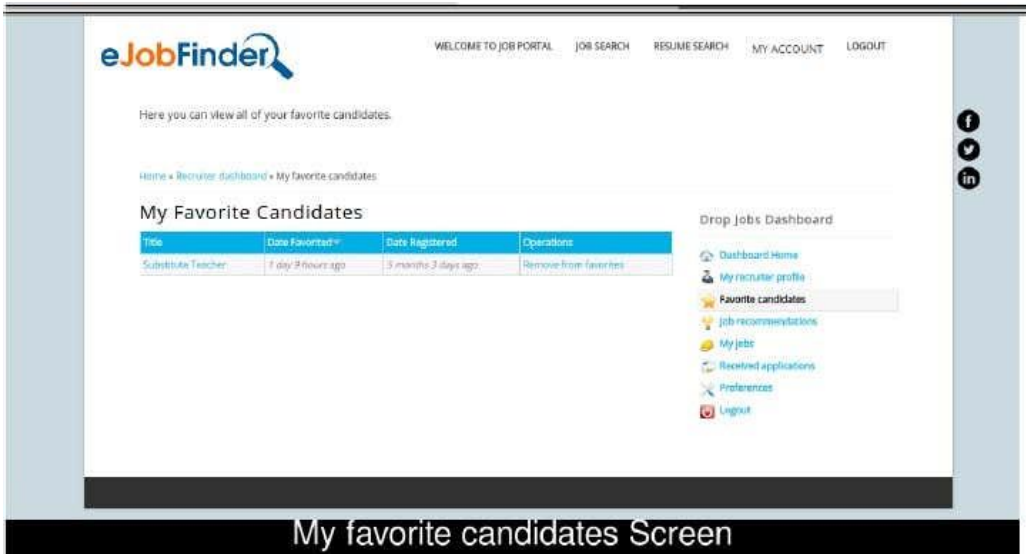
Candidate Skills Screen



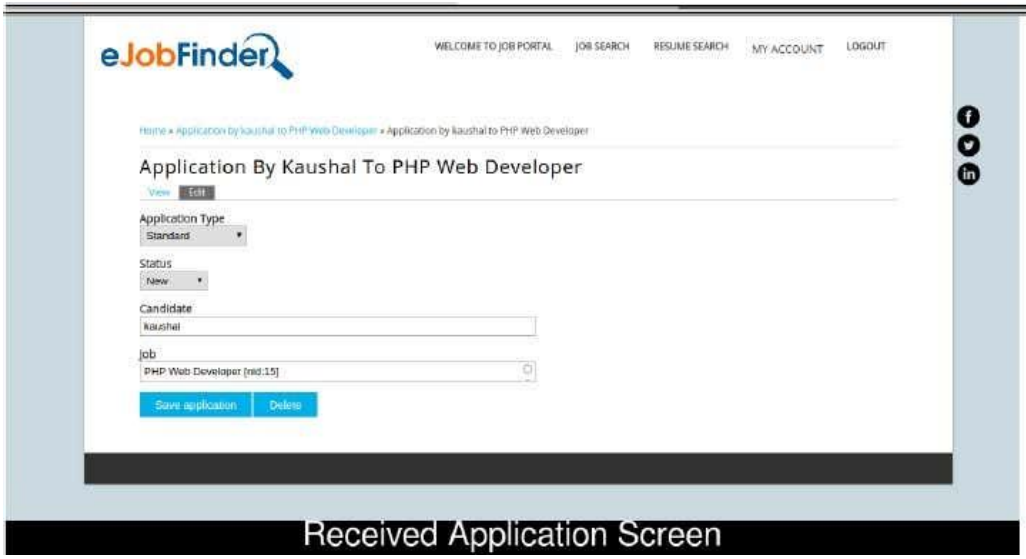
Edit Recruiter Profile Screen



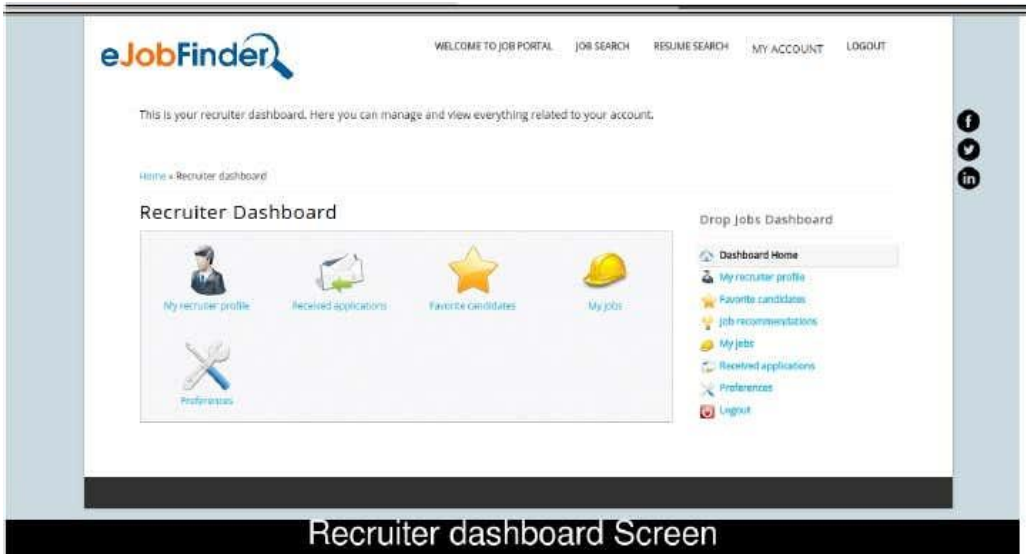
Forgot Password Screen



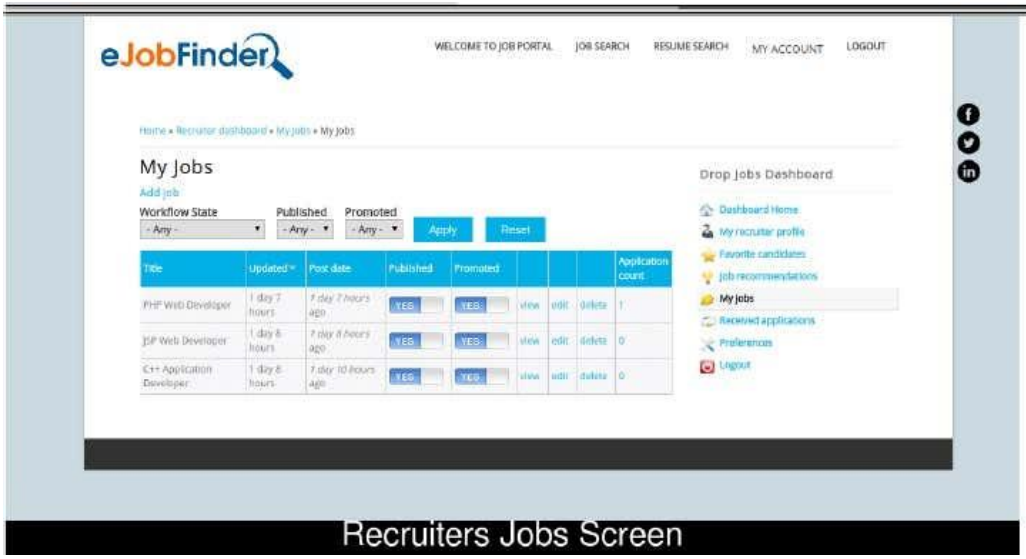
My favorite candidates Screen



Received Application Screen



Recruiter dashboard Screen



Recruiters Jobs Screen

Conclusion of the Project Online Job Portal:

Our project is only a humble venture to satisfy the needs to manage their project work. Several user friendly coding have also adopted. This package shall prove to be a powerful package in satisfying all the requirements of the school. The objective of software planning is to provide a frame work that enables the manger to make reasonable estimates made within a limited time frame at the beginning of the software project and should be updated regularly as the project progresses.

At the end it is concluded that I have made effort on following points...

- A description of the background and context of the project and its relation to work already done in the area.
- Made statement of the aims and objectives of the project.
- The description of Purpose, Scope, and applicability.
- We define the problem on which we are working in the project.
- We describe the requirement Specifications of the system and the actions that can be done on these things.
- We understand the problem domain and produce a model of the system, which describes operations that can be performed on the system.
- We included features and operations in detail, including screen layouts.
- We designed user interface and security issues related to system.
- Finally the system is implemented and tested according to test cases.