

A mini project report submitted on

LOCAL JOBS

For the partial fulfillment of the requirements for the award of the degree of

BACHELOR OF TECHNOLOGY

in

COMPUTER SCIENCE & ENGINEERING

Submitted by

P.MONIKA	319129510035
T.SWATHI	319129510046
D.VINEETHA	319129510013
P.HEMANTH	319129510033
A.PAUL SUDARSHAN	319129510002



Under the Esteemed Guidance of

Mrs. K.V.LAKSHMI M.Tech

Assistant Professor

Department of Computer Science & Engineering

WELFARE INSTITUTE OF SCIENCE TECHNOLOGY AND MANAGEMENT

(Approved by AICTE, New Delhi and Affiliated to Andhra University)Pinagadi, Pendurthi,

Visakhapatnam-531173, Andhra Pradesh, India.

WELFARE INSTITUTE OF SCIENCE TECHNOLOGY AND MANAGEMENT

(Approved by AICTE, New Delhi and Affiliated to Andhra University) Pinagadi, Pendurthi,
Visakhapatnam-531173, Andhra Pradesh, India.



CERTIFICATE

This is to certify that the mini project report entitled "LOCAL JOBS" being submitted by

P.MONIKA	319129510035
T.SWATHI	319129510046
D.VINEETHA	319129510013
P.HEMANTH	319129510033
A.PAUL SUDARSHAN	319129510002

In partial fulfillment for the award of the degree Bachelor of Technology in Computer Science and Engineering to the Andhra Pradesh is a record of bonafide project work carried out under my guidance and supervision .The results embodied in this project report have not been submitted to any University for the award of any degree.

INTERNAL GUIDE

Mrs. K.V.LAKSHMI M.Tech
Assistant Professor,
Department of CSE ,
WISTM, Engg.College

HEAD OF THE DEPARTMENT

Mrs.K.V.LAKSHMI M.Tech
Assistant Professor,
Department of CSE ,
WISTM, Engg.College

EXTERNAL EXAMINER

ACKNOWLEDGEMENT

The satisfaction that accompanies the successful completion of any task would be incomplete without the mention of people who made it possible and whose constant guidance and encouragement crown all the effort with success. I would like to thank **Dr. M. JAMES STEPHEN**, Principal, welfare Institute of Science Technology and Management, for this kind of cooperation. It gives me boundless pleasure to avail this opportunity to express my deep sense of gratitude and wholehearted thanks to Head of Department **Mrs. K.V.LAKSHMI**, **Assistant Professor**, Department of CSE and all the faculty of computer science department for their valuable suggestions and co-operation during the development of this project.

I am really grateful to all my friends for their constructive suggestions to do this project in effective manner and I would like to thank my parents and other family members for their continuous support and encouragement in completing this project.

P.MONIKA	319129510035
T.SWATHI	319129510046
D.VINEETHA	319129510013
P.HEMANTH	319129510033
A.PAUL SUDARSHAN	319129510002

DECLARATION

I here declare that the project entitled “**Local Jobs**” has been done under the guidance of **Mrs.K.V.LAKSHMI Assistant Professor, Department of CSE** and is dissertation of my own work except where specifically ask to the contrary and is submitted to the department of computer science and Engineering, Wellfare Institute of science technology and management for the partial fulfilment of the requirement for the award of B.Tech degree.

P.MONIKA	319129510035
T.SWATHI	319129510046
D.VINEETHA	319129510013
P.HEMANTH	319129510033
A.PAUL SUDARSHAN	319129510002

ABSTRACT

In this project, you will be able to create a better platform for small businesses to post job openings and for many job searchers, both educated and unskilled. Our portal informs users about job opportunities in their immediate vicinity. The goal of this initiative is to boost small business/enterprise growth while also assisting job seekers in finding employment that meet their needs and are close to home. Many job searchers look for work by visiting every location to check if there are any openings; however, visiting many locations takes more time. As a result, our website prioritises employment based on their qualifications and area, making it simple for them to find work.

TABLE OF CONTENTS

1. INTRODUCTION	2
2. LITERATURE SURVEY	3
3. SYSTEM ANALYSIS	4
3.1 Existing System	4
3.2 Proposed System	4
4. SYSTEM REQUIREMENTS	5
4.1 Software Requirements	5
4.2 Hardware Requirements	5
5. SYSTEM ARCHITECTURE	6
6. TECHNOLOGIES	7-29
7. UML DIAGRAMS	30-32
8. CODING	33-51
9. SCREENS	52-62
10. CONCLUSION	63

INTRODUCTION

Our Internet Site Local Jobs is a platform that connects job seekers and employers. Many job seekers are looking for work depending on their qualifications, location, and criteria. It is difficult to visit every location to determine whether or not a job is available. Job publishers use pamphlets and newspapers to distribute their work. However, no outcome is necessary. On our website, you can both submit a job and seek for one. Our website has an administrator who will review and verify if the job advertised by the job publisher is present or not. If it exists, he either posts it on the website or notifies the job publisher. Our website is simple to navigate and comprehend. It is beneficial to both educated and illiterate individuals .It has a simple Interface.

LITERATURE SURVEY

There are several avenues for finding career information, including newspapers, friends' recommendations, and numerous internet services. However, we are unsure if the position exists or not. It is tough to get a job near their home by visiting every little business, and so much time is lost by visiting every area. Some online job posting platforms charge a fee to submit a position, and you will receive emails from several job postings. Our platform informs users about available employment, and posting a job on our website is free. Our website has an administrator that examines and reviews the jobs published by the publisher before publishing them.

SYSTEM ANALYSIS

3.1 EXISTING SYSTEM

Many prominent online websites exist, such as Naukeri.com and LinkedIn.com. However, while posting a job, one must pay for it. As a result, small businesses cannot afford to pay for job postings. We are unsure about the job on certain internet platforms, whether the work is available at that place or not. Many small businesses do not have access to a better platform for posting job openings. As a result, one of the ways they choose to publicise their jobs is by distributing flyers at various public locations such as bus stops and railway stations. Some job searchers are illiterate and want positions such as sales boy, hotel server, and so on. They should inquire at all nearby businesses to see if there are any openings. It takes time and effort to visit several businesses.

3.2 PROPOSED SYSTEM

To address the existing system issues, we provide a platform for small businesses to advertise jobs and fill vacancies. Job searchers may search for employment in their area and nearby areas depending on their qualifications and needs. This platform includes contact information for both the publisher and the job seeker, allowing them to speak directly and discuss the position. They may post and apply for employment from home using their mobile phones or computers, and they can apply for positions that match their qualifications. The proposed system saves time. It is simple to learn about the job information and to contact each other directly. Job fulfillment and free job posting.

SYSTEM REQUIREMENTS

4.1 SOFTWARE REQUIREMENTS

Front End:

- HTML
- CSS
- JAVA SCRIPT
- BOOTSTRAP

Back End:

- PYTHON
- DJANGO

4.2 HARDWARE REQUIREMENTS

- Windows 10
- Intel i3 Processor
- RAM 1GB or more
- External 10GB

SYSTEM ARCHITECTURE

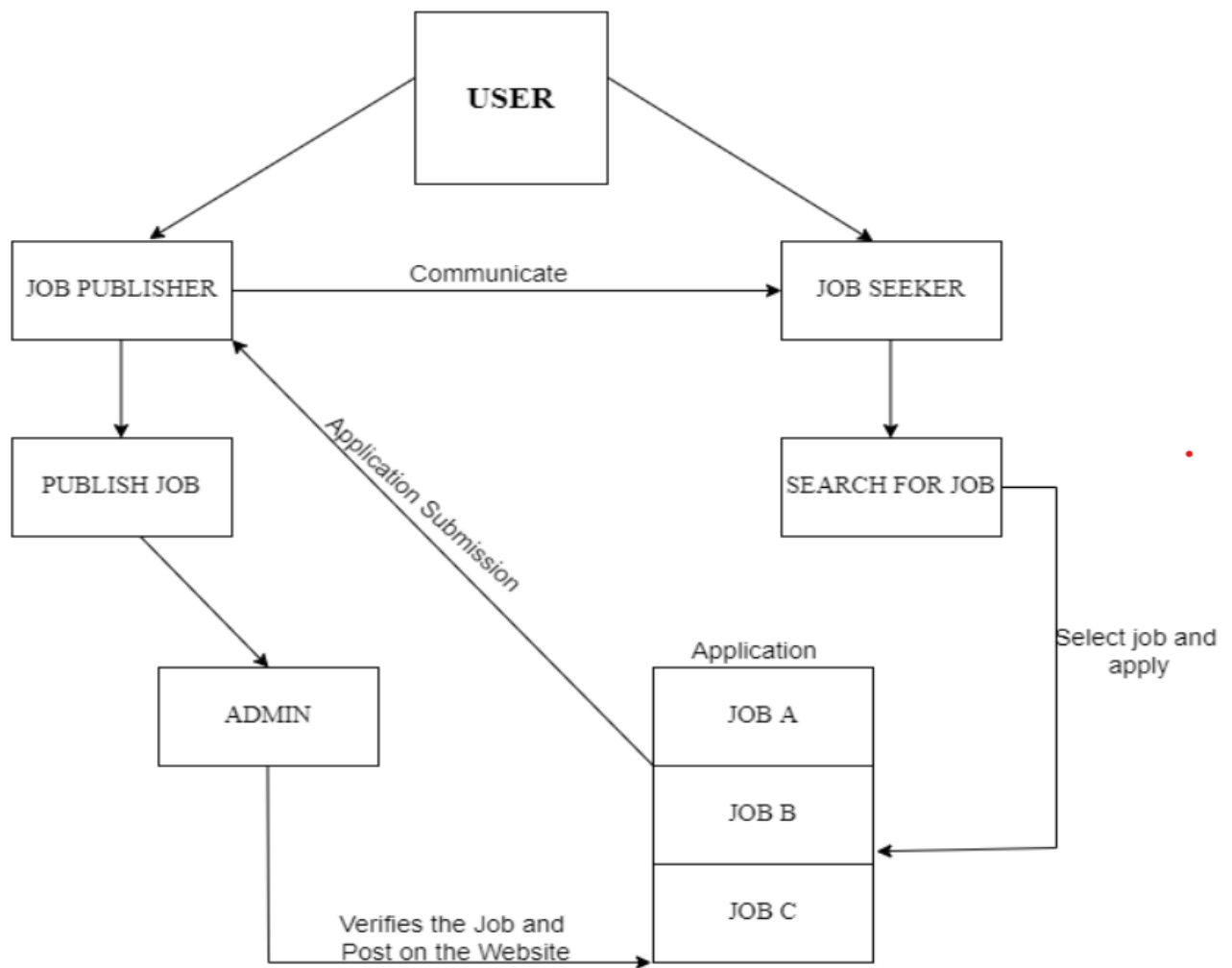


Fig 1: System Architecture

There are three types of users in this System Architecture: Job Seeker, Job Publisher, and Admin. Here, the job publisher publishes the job, and the admin verifies it before posting it on the website. The job seeker searches for and applies for jobs. The job seeker's information will be sent to the job publisher. When a candidate is chosen for a job, the publisher contacts the candidate directly.

TECHNOLOGIES STACK

- HTML
- CSS
- JAVA SCRIPT
- BOOTSTRAP
- PYTHON
- DJANGO

HTML [Hyper Text Markup Language]

- HTML stands for Hyper Text Markup Language.
- HTML describes the structure of web pages using markup.
- HTML elements are the building blocks of HTML pages.
- HTML elements are represented by tags.
- HTML tags label pieces of content such as heading, paragraph, table and so on.
- Browsers do not display the HTML tags, but use them to render the content of the page.

<!doctype html>: The <!doctype> is not an HTML tag. It is an instruction to the web browser about what version of html the pages is written in. This page is written in HTML5 as opposed to say HTML 4.01.

<html>: The <html> element is the root element of an HTML page.

<head>: The <head> element is a container for meta information about the document and it is placed between the <html> and <body> tags.

<title>: The <title> tag is required in all HTML documents and it specifies a title for the document.

<body>: The <body> element contains the visible page content.

<h1>: The <h1> element defines a large heading.

<h6>: The <h6> element defines a small heading.

<p>: The <p> element defines a paragraph.

<a>: The <a> element defines the HTML links. The link's destination is specified in the href attribute.

: The element defines the HTML images. The source file (src), alternative text (alt), width, and height are provided as attributes.

<button>: The <button> element defines a clickable button, inside a button you can put content.

**
:** The
 element defines the line break. This tag is an empty tag, which means that it has no end tag.

<!-- -->: The <!-- --> element defines the comment tag.

<table>: The <table> element defines the HTML tables.

<tr>: The <tr> element defines the table row.

<th>: The <th> element defines the table header. By default, table headings are bold and centered.

<td>: The <td> element defines the table data/cell.

<div>: The <div> element defines the block level element. A block-level element always starts on a new line and takes up the full width available.

<form>: It defines a form that is used to collect user input.

<label>: The <label> element represents a caption for an item in a user interface.

<input>: The <input> element is the most important form element. The <input> element can be displayed in several ways, depending on the type attribute.

: The element to define an unordered list.

: The element to define an ordered list.

: The element to define a list item.

CSS [Cascading Style Sheet]

Cascading Style Sheets, fondly referred to as CSS, is a simple design language intended to simplify the process of making web pages presentable.

CSS handles the look and feel part of a web page. Using CSS, you can control the colour of the text, the style of fonts, the spacing between paragraphs, how columns are sized and laid out, what background images or colours are used, layout designs, variations in display for different devices and screen sizes as well as a variety of other effects.

CSS can be added to HTML elements in 3 ways:

- **Inline** - by using the style attribute in HTML elements.
- **Internal** - by using a <style> element in the <head> section.
- **External** - by using an external CSS file.

Css Colors: Colors are specified using predefined color names, or RGB, HEX, HSL, RGBA, HSLA values.

- **Css Names:** In HTML, a color can be specified by using a color name.
- **Background Color:** You can set the background color for HTML elements.
- **Text Color:** You can set the color of text.
- **Border Color:** You can set the color of borders.
- **Color Values:** In HTML, colors can also be specified using RGB values, HEX values, HSL values, RGBA values, and HSLA values.

Css Backgrounds: The CSS background properties are used to define the background effects for elements.

- **Background color:** The background-color property specifies the background color of an element.
- **Background Image:** The background-image property specifies an image to use as the background of an element. By default, the image is repeated so it covers the entire element.
- **Background Repeat:** The background-repeat property specifies a background image will be repeated.

- **Background Attachment:** The background-attachment property specifies a background image is fixed or scrolls with the rest of the page.
- **Background Position:** The background-position property specifies a starting position of a background image.

Css Borders:

- **Css Border Properties:** The CSS border properties allow you to specify the style, width, and color of an element's border.
- **Border Style:** The Border-style property specifies what kind of border to display.
- **Border Width:** The border-width property specifies the width of the four borders. The width can be set as a specific size (in px, pt, cm, em, etc) or by using one of the three pre-defined values: thin, medium, or thick.
- **Border Color:** The border-color property is used to set the color of the four borders.

Css Margins: The CSS margin properties are used to create space around elements, outside of any defined borders.

- **Margin-Individual Sides:** The CSS margin properties are used to create space around elements, outside of any defined borders.
- **Margin-Shorthand Property:** To shorten the code, it is possible to specify all the margin properties in one property.
- **Margin Collapse:** Top and bottom margins of elements are sometimes collapsed into a single margin that is equal to the largest of the two margins. This does not happen on left and right margins! Only top and bottom margins.

Css Padding: The CSS padding properties are used to generate space around an element's content, inside of any defined borders.

- **Padding-Individual sides:** CSS has properties for specifying the padding for each side of an elements are padding-top, padding-bottom, padding-left, padding-right.
- **Padding - Shorthand Property:** To shorten the code, it is possible to specify all the padding properties in one property.

CSS Height and Width:

- **Setting max-width:** The max-width property is used to set the maximum width of an element. The max-width can be specified in length values, like px, cm, etc., or in percent (%) of the containing block, or set to none (this is default. Means that there is no maximum width).

CSS Box Model: All HTML elements can be considered as boxes. In CSS, the term "box model" is used when talking about design and layout.

The CSS box model is essentially a box that wraps around every HTML element. It consists of margins, borders, padding, and the actual content.

- **Content** - The content of the box, where text and images appear
- **Padding** - Clears an area around the content. The padding is transparent
- **Border** - A border that goes around the padding and content
- **Margin** - Clears an area outside the border. The margin is transparent

The box model allows us to add a border around elements, and to define space between elements.

CSS Outline: An outline is a line that is drawn around elements, OUTSIDE the borders, to make the element "stand out".

- **Outline Style:** The outline-style property specifies the style of the outline, and can have one of the following values like dotted, dashed, solid, double etc.
- **Outline Color:** The outline-color property is used to set the color of the outline.
- **Outline Width:** The outline-width property specifies the width of the outline, and can have one of the following values:
 - thin (typically 1px)
 - medium (typically 3px)
 - thick (typically 5px)
 - A specific size (in px, pt, cm, em, etc)
- **Outline Offset:** The outline-offset property adds space between an outline and the edge/border of an element. The space between an element and its outline is transparent.

CSS Text:

- **Text Color:** The color property is used to set the color of the text.
- **Text Alignment:** The text-align property is used to set the horizontal alignment of a text. A text can be left or right aligned, centered, or justified.
- **Text Decoration:** The text-decoration property is used to set or remove decorations from text. The value text-decoration: none; is often used to remove underlines from link.
- **Text Transformation:** The text-transform property is used to specify uppercase and lowercase letters in a text. It can be used to turn everything into uppercase or lowercase letters, or capitalize the first letter of each word.
- **Text Indentation:** The text-indent property is used to specify the indentation of the first line of a text.
- **Letter Spacing:** The letter-spacing property is used to specify the space between the characters in a text.
- **Line Height:** The line-height property is used to specify the space between lines.
- **Text Direction:** The direction property is used to change the text direction of an element.
- **Word Spacing:** The word-spacing property is used to specify the space between the words in a text.

CSS Font Families:

In CSS, there are two types of font family names:

- **generic family** - a group of font families with a similar look (like "Serif" or "Monospace")
- **font family** - a specific font family (like "Times New Roman" or "Arial"). The font family of a text is set with the font-family property. The font-family property should hold several font names as a "fallback" system. If the browser does not support the first font, it tries the next font, and so on.
- **Font Style:** The font-style property is mostly used to specify italic text. This property has three values:
 - **normal** - The text is shown normally.
 - **italic** - The text is shown in italics.

- **oblique** - The text is "leaning" (oblique is very similar to italic, but less supported)
- **Font Size:** The font-size property sets the size of the text. Being able to manage the text size is important in web design. However, you should not use font size adjustments to make paragraphs look like headings, or headings look like paragraphs.

Css Links: Links can be styled differently depending on what **state** they are in.

The four links states are:

- a:link - a normal, unvisited link
- a:visited - a link the user has visited
- a:hover - a link when the user mouses over it
- a:active - a link the moment it is clicked

JAVA SCRIPT

JavaScript is a dynamic computer programming language. It is lightweight and most commonly used as a part of web pages, whose implementations allow client-side script to interact with the user and make dynamic pages. It is an interpreted programming language with object-oriented capabilities.

- JavaScript is a lightweight, interpreted programming language.
- Designed for creating network-centric applications.
- Complementary to and integrated with Java.
- Complementary to and integrated with HTML.
- Open and cross-platform.

JavaScript Can Change HTML Content: One of many JavaScript HTML methods is getElementById().

<script>Tag: In HTML, JavaScript code must be inserted between <script> and </script> tags.

JavaScript in <head> or <body>: You can place any number of scripts in an HTML document. Scripts can be placed in the <body>, or in the <head> section of an HTML page, or in both.

External JavaScript:

External scripts are practical when the same code is used in many different web pages.

JavaScript files have the file extension **.js**.

To use an external script, put the name of the script file in the src (source) attribute of a <script> tag:

JavaScript Display Possibilities:

JavaScript can "display" data in different ways:

- Writing into an HTML element, using innerHTML.
- Writing into the HTML output using document.write().
- Writing into an alert box, using window.alert().
- Writing into the browser console, using console.log().

Using inner HTML

To access an HTML element, JavaScript can use the document.getElementById(id) method. The id attribute defines the HTML element. The innerHTML property defines the HTML content.

Using window.alert(): You can use an alert box to display data.

Using console.log(): For debugging purposes, you can use the console.log() method to display data.

JavaScript White Space: JavaScript ignores multiple spaces. You can add white space to your script to make it more readable.

JavaScript Code Blocks: JavaScript statements can be grouped together in code blocks, inside curly brackets {...}. The purpose of code blocks is to define statements to be executed together.

JavaScript Values: The JavaScript syntax defines two types of values: Fixed values and variable values. Fixed values are called **literals**. Variable values are called **variables**.

JavaScript Variables: In a programming language, **variables** are used to **store** data values. JavaScript uses the var keyword to **declare** variables. An **equal sign** is used to **assign values** to variables.

JavaScript Identifiers:

All JavaScript **variables** must be **identified** with **unique names**.

These unique names are called **identifiers**.

Identifiers can be short names (like x and y) or more descriptive names (age, sum, totalVolume).

The general rules for constructing names for variables (unique identifiers) are:

- Names can contain letters, digits, underscores, and dollar signs.
- Names must begin with a letter
- Names can also begin with \$ and _ (but we will not use it in this tutorial)
- Names are case sensitive (y and Y are different variables)
- Reserved words (like JavaScript keywords) cannot be used as names

JavaScript Functions:

A JavaScript function is a block of code designed to perform a particular task.

A JavaScript function is executed when "something" invokes it.

Function Invocation:

The code inside the function will execute when "something" **invokes** (calls) the function:

- When an event occurs (when a user clicks a button)
- When it is invoked (called) from JavaScript code
- Automatically (self invoked)

Function Return:

When JavaScript reaches a return statement, the function will stop executing.

If the function was invoked from a statement, JavaScript will "return" to execute the code after the invoking statement.

JavaScript Events:

HTML events are "**things**" that happen to HTML elements.

When JavaScript is used in HTML pages, JavaScript can "**react**" on these events.

HTML Events:

An HTML event can be something the browser does, or something a user does.

Here are some examples of HTML events:

- An HTML web page has finished loading
- An HTML input field was changed
- An HTML button was clicked

Often, when events happen, you may want to do something. JavaScript lets you execute code when events are detected.

HTML allows event handler attributes, **with JavaScript code**, to be added to HTML elements.

- **Onchange-** An HTML element has been changed.
- **Onclick-** The user clicks an HTML element.
- **Onmouseover-** The user moves the mouse over an HTML element.
- **Onmouseout-** The user moves the mouse away from an HTML element.
- **Onkeydown-** The user pushes a keyboard key.
- **Onload-** The browser has finished loading the page.

BOOTSTRAP

Bootstrap is a free and open source front end development framework for the creation of websites and web apps. The Bootstrap framework is built on HTML, CSS and JavaScript (JS) to facilitate the development of responsive.

.align : A set of utility classes that are equivalent to writing the css property. You can use this on inline and table cell elements.

.align-content : Added to the parent flexbox container to determining how the elements are aligned horizontally.

.align-items : Class added to flexbox child items to specify if it should align towards the top or bottom of the container (start, end).

.bg : Background color utility classes.

.border : A versatile border utility class that lets you add/remove borders on a side or change a border color.

.btn-outline: A button variation to have outlined buttons instead of a solid background.

.carousel-control : When you have an image carousel with pagination you will use this class on the previous and next anchor links.

.carousel-fade : Animates the slide transition with a crossfade instead of a slide.

.form-control-plaintext : Use the class to remove the default form field styling and preserve the correct margin and padding.

.form-row : Works similar to a grid. but is more compact to make the form look more uniform.

.h-* : Height utility class that makes the element a percentage height of its parent element.

.justify-content-*-* : Class specifies where the flex items will be positioned inside the container.

.nav-fill : Makes all nav items use all available horizontal space. Nav items are different widths based on their content.

.nav-justified : Makes all nav items equal width and use all available horizontal space.

.navbar-collapse : The nav links that are collapsed and shown when toggled on mobile widths.

.navbar-expand-* : Since the navbar is displayed collapse on mobile first, this class specifies what breakpoint you want the navbar to not be collapsed.

.navbar-text : Vertically centres text inside a navbar.

.navbar-toggler-icon : The cheeseburger navigation icon is set using an svg background image of three horizontal lines.

.btn-group-lg : Increases the default button group size.

.btn-group-sm : Decreases the default button group size.

.btn-group-toggle : This class replaces an input checkbox with a custom style that is toggable on click.

.btn-outline-* : Transparent background with colored text and boarder.In bootstrap there is some button styles.There are

.btn,.btn-default,.btn-primary,.btn-success,.btn-info,.btn-warning,.btn-danger,.btn-link.

.carousel-item :The wrapper class applied to each individual carousel item.

.col-form-label : Class added to form labels to apply consistent padding and margins

.display-* : This set of classes increases the font size of headings in 4 stages. These classes are used for headings outside of the main content of the page like jumbotrons and page headers. Append (1-4) to the end to adjust size.

.dropdown-item : This class is added to each link item shown in a dropdown menu.

.dropdown-toggle-split : Removes the interactivity from a dropdown so it does not appear clickable.

.font-* : italic, weight-bold, weight-light, weight-normal, monospace.

.form-inline : Use this class to have a series of labels and form elements on a single horizontal row.

.form-text : This class is used for help text alongside form elements. You can add .text-muted to make the text lighter in color.

.input-group-text: This class adds the background color and text styles to the text inside an input group.

.nav-item : If your nav uses a list add this class to each list item for the proper spacing.

.nav-link : Each anchor link inside your nav is given this class in order to have the proper styling.

.nav-pills : Use this class along with .nav to make each nav link into a button.

.navbar-brand : Most navbars contain a logo or brand. This class is added to the anchor.

.navbar : Navigation header class.

.navbar-nav : The wrapper class of the navigation elements excluding the brand.

.navbar-light : Add this class to your navbar if you would like it to have a light background and dark text.

.navbar-toggler : The infamous cheeseburger icon to signify a navigation menu on mobile.

.position-* : Not responsive, but a group of utility classes to add common position values. Position may be absolute, fixed, relative, static, sticky.

.container : Fixed width container with widths determined by screen sizes. Equal margin on the left and right

.container-fluid : Spans the full width of the screen.

.media : Media components are image heading and description text items. Blog comments, portfolio projects, album covers, etc.

.modal : The parent wrapper class of modal content.

.modal-body : The modal body content classes : Header - Body – Footer.

.modal-content : modal-content contains modal-body, modal-header, and modal-footer.

.modal-dialog : The secondary wrapper class of the entire modal content.

.modal-header : The header section of the modal that contains the title and close button.

.modal-footer : The footer of the modal that contains action buttons or help text.

.tooltip : The Tooltip plugin is small pop-up box that appears when the user moves the mouse pointer over an element.

.popover: It is a pop-up box that appears when the user clicks on an element. The difference is that the popover can contain much more content.

.panel: A panel in bootstrap is a bordered box with some padding around its content. To color the panel, use contextual classes `.panel-default`, `.panel-primary`, `.panel-success`, `.panel-info`, `.panel-warning`, or `.panel-danger`.

. pagination: If you have a web site with lots of pages, you may wish to add some sort of pagination to each page.

. Jumbotron: A jumbotron indicates a big box for calling extra attention to some special content or information. A jumbotron is displayed as a grey box with rounded corners. It also enlarges the font sizes of the text inside it.

.pager: Pager is also a form of pagination, Pager provides previous and next buttons (links).

PYTHON

Python is an interpreted, object-oriented, high-level programming language with dynamic semantics. Its high-level built in data structures, combined with dynamic typing and dynamic binding, make it very attractive for Rapid Application Development, as well as for use as a scripting or glue language to connect existing components together. Python's simple, easy to learn syntax emphasizes readability and therefore reduces the cost of program maintenance. Python supports modules and packages, which encourages program modularity and code reuse. The Python interpreter and the extensive standard library are available in source or binary form without charge for all major platforms, and can be freely distributed.

IN BUILT FUNCTIONS IN PYTHON

abs(x): Return the absolute value of a number. The argument may be an integer, a floating point number, or an object implementing `__abs__()`. If the argument is a complex number, its magnitude is returned.

Aiter(async_iterable) : Return an asynchronous iterator for an asynchronous iterable. Equivalent to calling `x.__aiter__()`.

all(iterable) : Return `True` if all elements of the *iterable* are true (or if the iterable is empty).

any(*iterable*) : Return `True` if any element of the *iterable* is true. If the iterable is empty, return `False`.

bin(*x*) : Convert an integer number to a binary string prefixed with “0b”. The result is a valid Python expression. If *x* is not a Python int object, it has to define an `__index__()` method that returns an integer.

breakpoint(args*, ***kws*)** : This function drops you into the debugger at the call site.

callable(*object*) : Return `True` if the *object* argument appears callable, `False` if not. If this returns `True`, it is still possible that a call fails, but if it is `False`, calling *object* will never succeed.

chr(*i*) : Return the string representing a character whose Unicode code point is the integer *i*.

dir([*object*]) : Without arguments, return the list of names in the current local scope. With an argument, attempt to return a list of valid attributes for that object.

divmod(*a*, *b*) : Take two (non-complex) numbers as arguments and return a pair of numbers consisting of their quotient and remainder when using integer division.

enumerate(*iterable*, *start*=0) : Return an enumerate object. *iterable* must be a sequence, an iterator, or some other object which supports iteration.

eval(*expression*[, *globals*[, *locals*]]) : The arguments are a string and optional globals and locals. If provided, *globals* must be a dictionary. If provided, *locals* can be any mapping object.

exec(*object*[, *globals*[, *locals*]]) : This function supports dynamic execution of Python code. *object* must be either a string or a code object.

filter(*function*, *iterable*) : Construct an iterator from those elements of *iterable* for which *function* returns true.

float([*x*]) : Return a floating point number constructed from a number or string *x*.

format(*value*[, *format_spec*]) : Convert a *value* to a “formatted” representation, as controlled by *format_spec*. The interpretation of *format_spec* will depend on the type of the *value* argument.

getattr(object, name[, default]) : Return the value of the named attribute of *object*. *name* must be a string. If the string is the name of one of the object's attributes, the result is the value of that attribute.

globals() : Return the dictionary implementing the current module namespace. For code within functions, this is set when the function is defined and remains the same regardless of where the function is called.

hasattr(object, name) : The arguments are an object and a string. The result is `True` if the string is the name of one of the object's attributes, `False` if not.

hash(object) : Return the hash value of the object (if it has one). Hash values are integers.

hex(x) : Convert an integer number to a lowercase hexadecimal string prefixed with "0x".

id(object) : Return the "identity" of an object. This is an integer which is guaranteed to be unique and constant for this object during its lifetime.

input([prompt]) : If the *prompt* argument is present, it is written to standard output without a trailing newline.

isinstance(object, classinfo) : Return `True` if the *object* argument is an instance of the *classinfo* argument.

issubclass(class, classinfo) : Return `True` if *class* is a subclass (direct, indirect, or virtual) of *classinfo*. A class is considered a subclass of itself.

iter(object[, sentinel]) : Return an iterator object. The first argument is interpreted very differently depending on the presence of the second argument.

len(s) : Return the length.

locals() : Update and return a dictionary representing the current local symbol table.

map(function, iterable, ...) : Return an iterator that applies *function* to every item of *iterable*, yielding the results.

max(iterable, *[, key, default]) : Return the largest item in an iterable or the largest of two or more arguments.

min(*iterable*, *[, *key*, *default*]) : Return the smallest item in an iterable or the smallest of two or more arguments.

next(*iterator*[, *default*]) : Retrieve the next item from the iterator by calling its `__next__()` method.

oct(*x*) : Convert an integer number to an octal string prefixed with “0o”.

open(*file*, *mode*='r', *buffering*=1, *encoding*=None, *errors*=None, *newline*=None, *closefd*=True, *opener*=None) : Open *file* and return a corresponding file object.

ord(*c*) : Given a string representing one Unicode character, return an integer representing the Unicode code point of that character.

pow(*base*, *exp*[, *mod*]) : Return *base* to the power *exp*.

print(**objects*, *sep*=' ', *end*='\n', *file*=sys.stdout, *flush*=False) : Print *objects* to the text stream *file*, separated by *sep* and followed by *end*. *sep*, *end*, *file*, and *flush*, if present, must be given as keyword arguments.

range(*start*, *stop*[, *step*]) : Rather than being a function, range is actually an immutable sequence type.

repr(*object*) : Return a string containing a printable representation of an object.

reversed(*seq*) : Return a reverse iterator.

round(*number*[, *ndigits*]) : Return *number* rounded to *ndigits* precision after the decimal point.

setattr(*object*, *name*, *value*) : This is the counterpart of `getattr()`.

sorted(*iterable*, /, *, *key*=None, *reverse*=False) : Return a new sorted list from the items in *iterable*.

str(*object*=") : Return a str version of *object*.

sum(*iterable*, /, *start*=0) : Sums *start* and the items of an *iterable* from left to right and returns the total. The *iterable*'s items are normally numbers, and the start value is not allowed to be a string.

super([*type*[, *object-or-type*]]) : Return a proxy object that delegates method calls to a parent or sibling class of *type*.

tuple([*iterable*]) : Rather than being a function, tuple is actually an immutable sequence type, as documented in Tuples .

type(*object*) : With one argument, return the type of an *object*. The return value is a type object and generally the same object.

vars([*object*]) : Return the `__dict__` attribute for a module.

zip(**iterables*, *strict=False*) : Iterate over several iterables in parallel, producing tuples with an item from each one.

PYTHON INSTALLATION

Python is a widely used high-level programming language. To write and execute code in python, we first need to install Python on our system.

Installing Python on Windows takes a series of few easy steps.

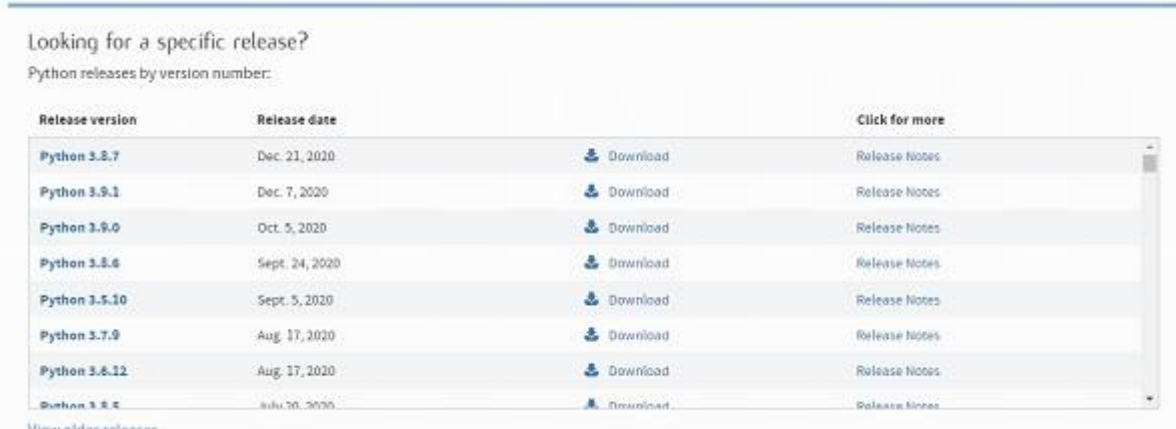
Step 1 – Select Version of Python to Install

Python has various versions available with differences between the syntax and working of different versions of the language. We need to choose the version which we want to use or need. There are different versions of Python 2 and Python 3 available.

Step 2 – Download Python Executable Installer

On the web browser, in the official site of python (www.python.org), move to the Download for Windows section.

All the available versions of Python will be listed. Select the version required by you and click on Download. Let suppose, we chose the Python 3.9.1 version.



Looking for a specific release?
Python releases by version number:

Release version	Release date	Click for more	
Python 3.8.7	Dec. 21, 2020	Download	Release Notes
Python 3.9.1	Dec. 7, 2020	Download	Release Notes
Python 3.9.0	Oct. 5, 2020	Download	Release Notes
Python 3.8.6	Sept. 24, 2020	Download	Release Notes
Python 3.8.5	Sept. 5, 2020	Download	Release Notes
Python 3.7.9	Aug. 17, 2020	Download	Release Notes
Python 3.6.12	Aug. 17, 2020	Download	Release Notes
Python 3.5.2	July 30, 2020	Download	Release Notes

[View older releases](#)

Fig 2: Different versions of Python

On clicking download, various available executable installers shall be visible with different operating system specifications. Choose the installer which suits your system operating system and download the installer. Let suppose, we select the Windows installer(64 bits).

The download size is less than 30MB.

Version	Operating System	Description	MDS Sum	File Size	GPG
Gzipped source tarball	Source release		429ae95d24227f8fa1560684fad6fca7	25372998	SIG
XZ compressed source tarball	Source release		61981498e75ac8f00adcb908281fad66	18897104	SIG
macOS 64-bit intel installer	Mac OS X	for macOS 10.9 and later	74f5cc5b5783ce8fb2ca55f11f3f0699	29795899	SIG
macOS 64-bit universal2 installer	Mac OS X	for macOS 10.9 and later, including macOS 11 Big Sur on Apple Silicon (experimental)	8b19748473609241e60aa3618bbaf3ed	37451735	SIG
Windows embeddable package (32-bit)	Windows		96c6fa81fe8b650e68c3dd41258ae317	7571141	SIG
Windows embeddable package (64-bit)	Windows		e70e5c22432d8f57a497cde5ec2e5ce2	8402333	SIG
Windows help file	Windows		c49d9b6ef88c0831ed0e2d39bc42b316	8787443	SIG
Windows installer (32-bit)	Windows		dde210ea04a31c27488605a9e7cd297a	27126136	SIG
Windows installer (64-bit)	Windows	Recommended	b3fce2ed8bc315ad2bc49eae48a94487	28204528	SIG

Fig 3: Specifications of the Version

Step 3 – Run Executable Installer

We downloaded the Python 3.9.1 Windows 64 bit installer.

Run the installer. Make sure to select both the checkboxes at the bottom and then click Install New.



Fig 4: Installation of Python

On clicking the Install Now, The installation process starts.

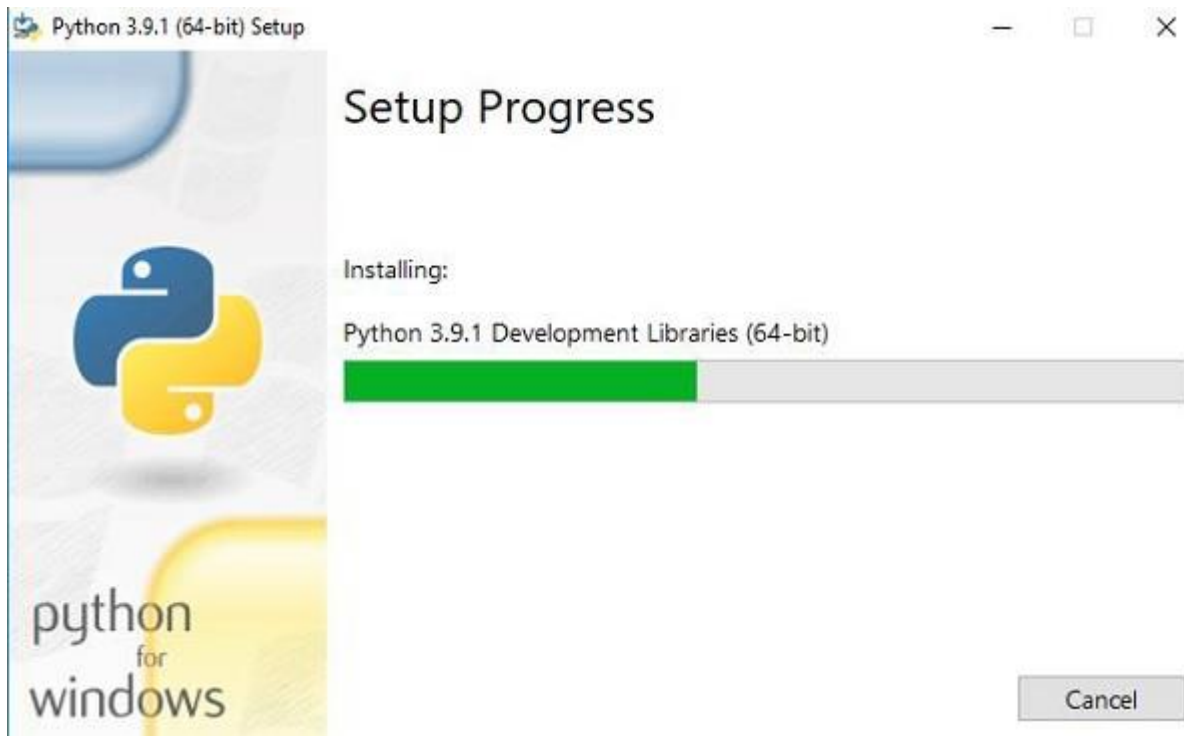


Fig 5: Installation Setup Progress

The installation process will take few minutes to complete and once the installation is successful, the following screen is displayed.

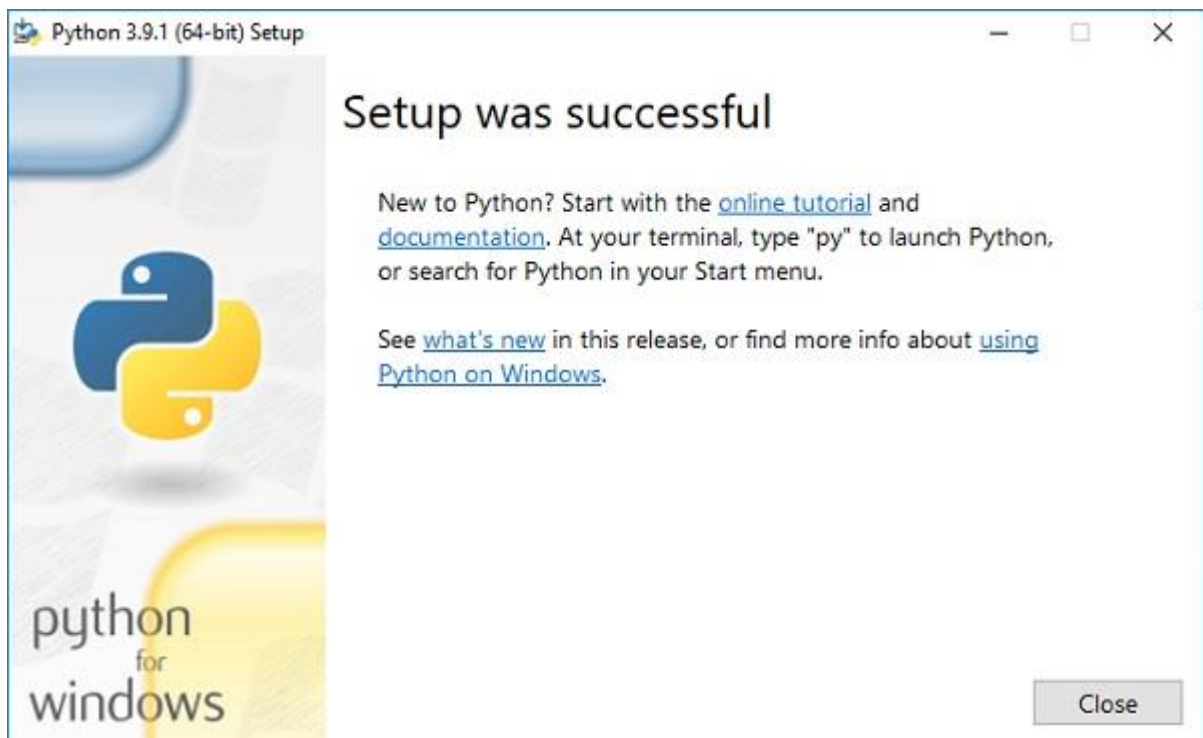
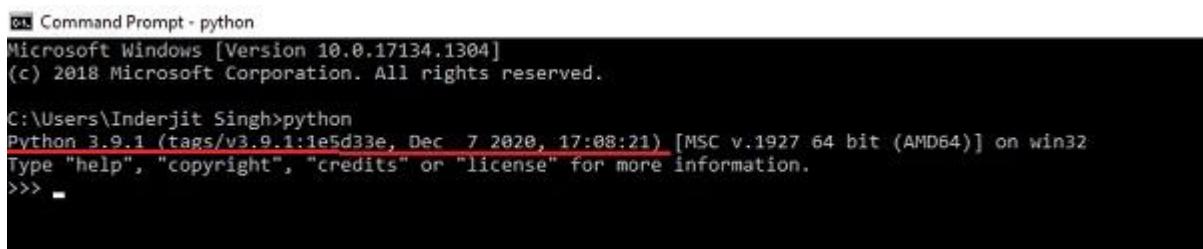


Fig 6: Installation setup was Successful

Step 4 – Verify Python is installed on Windows

To ensure if Python is successfully installed on your system. Follow the given steps –

- Open the command prompt.
- Type 'python' and press enter.
- The version of the python which you have installed will be displayed if the python is successfully installed on your windows.



```
Command Prompt - python
Microsoft Windows [Version 10.0.17134.1304]
(c) 2018 Microsoft Corporation. All rights reserved.

C:\Users\Inderjit Singh>python
Python 3.9.1 (tags/v3.9.1:1e5d33e, Dec 7 2020, 17:08:21) [MSC v.1927 64 bit (AMD64)] on win32
Type "help", "copyright", "credits" or "license()" for more information.
>>> _
```

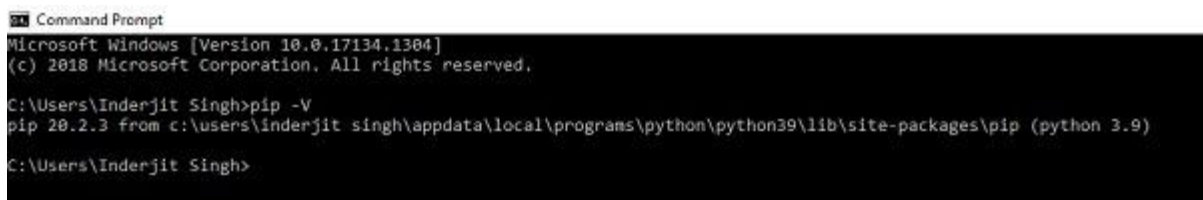
Fig 7: Check whether Python was installed

Step 5 – Verify Pip was installed

Pip is a powerful package management system for Python software packages. Thus, make sure that you have it installed.

To verify if pip was installed, follow the given steps –

- Open the command prompt.
- Enter pip -V to check if pip was installed.
- The following output appears if pip is installed successfully



```
Command Prompt
Microsoft Windows [Version 10.0.17134.1304]
(c) 2018 Microsoft Corporation. All rights reserved.

C:\Users\Inderjit Singh>pip -V
pip 20.2.3 from c:\users\inderjit singh\appdata\local\programs\python\python39\lib\site-packages\pip (python 3.9)

C:\Users\Inderjit Singh>
```

Fig 8: Check whether Pip was installed

We have successfully installed python and pip on our Windows system.

DJANGO

Django is a high-level Python web framework that encourages rapid development and clean, pragmatic design. Built by experienced developers, it takes care of much of the hassle of web development, so you can focus on writing your app without needing to reinvent the wheel. It's free and open source.

UML DIAGRAMS

SEQUENCE DIAGRAM

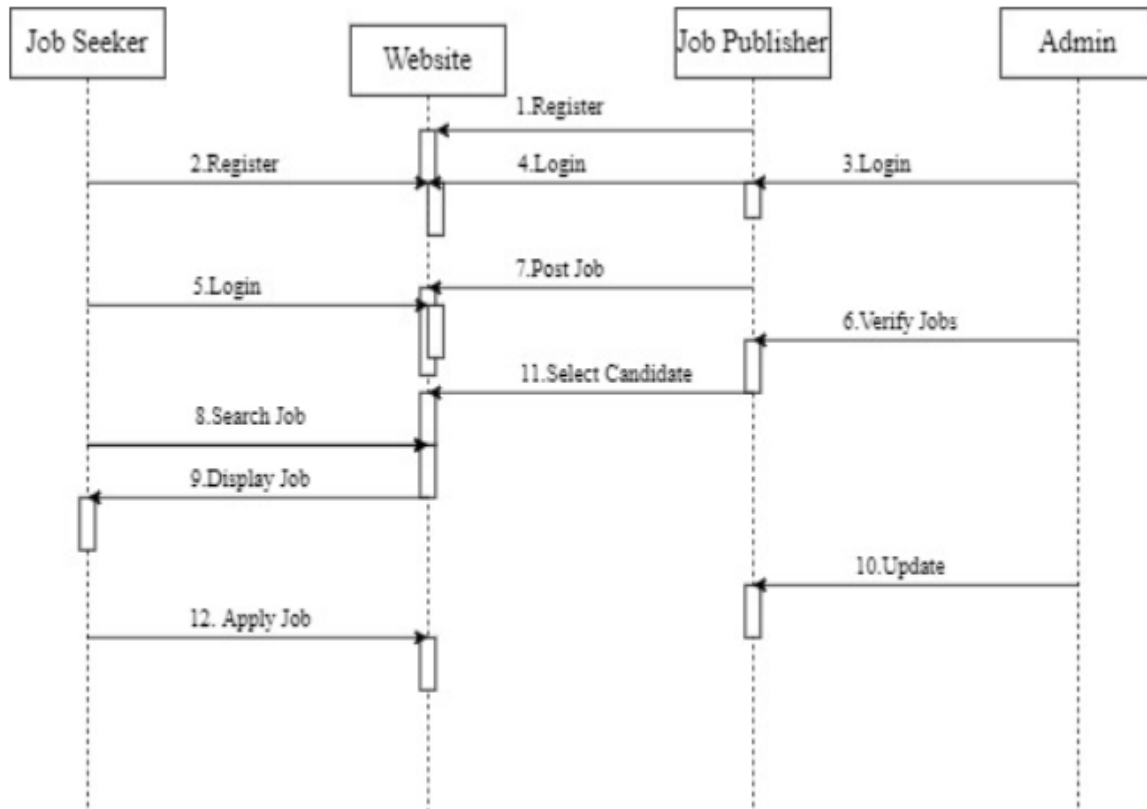


Fig 1: SEQUENCE DIAGRAM

A sequence diagram depicts the sequence of actions that occur in a system. It is a very useful tool to easily represent dynamic behaviour of the system. The sequence diagram can show the user, about the availability of jobs by Register, Login, Post Job, Verify Job, Apply Job, Update, Select Candidate.

USECASE DIAGRAM

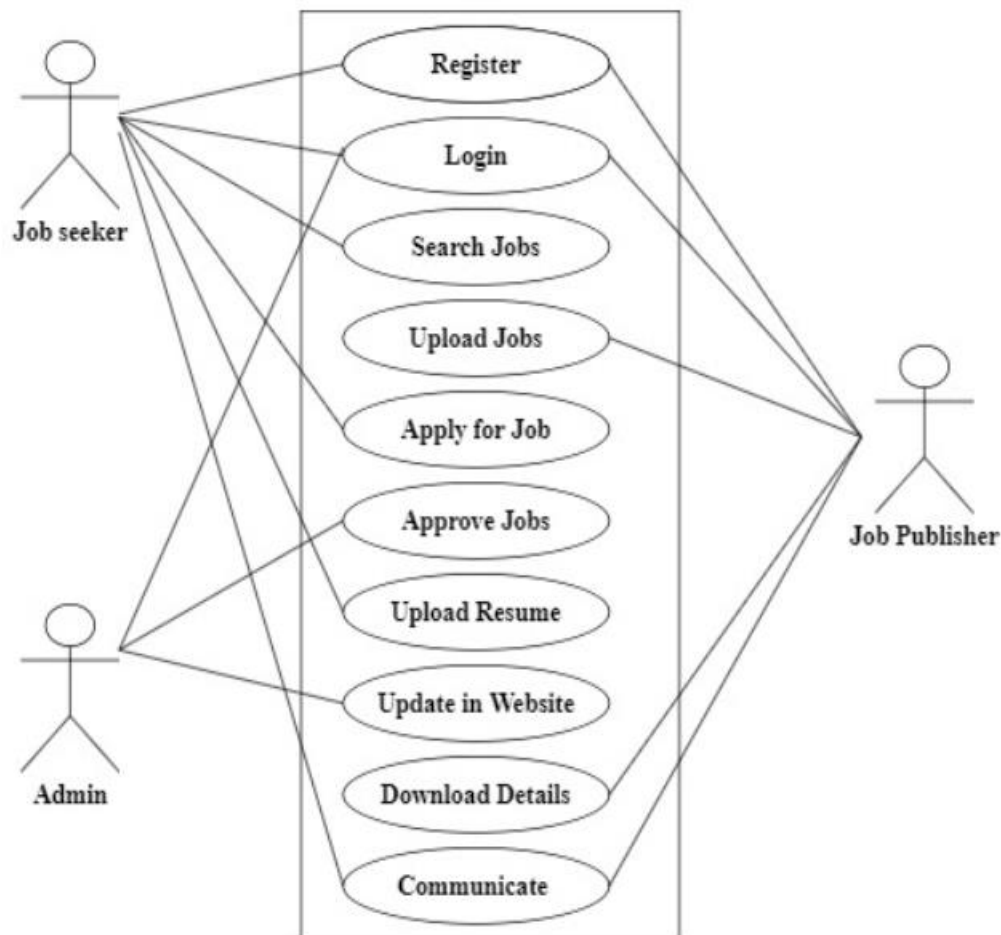


Fig 2: USECASE DIAGRAM

Use Case diagram give a graphic overview of the actors involved in a system, different functions needed by those actors are how different functions interact. We have three actors that is Admin, Job Publisher and Job Seeker. The job is verified by the administrator and published on the website. Job Publishers post jobs, find individuals, and communicate with them. Job seekers look for open positions and apply for them.

ACTIVITY DIAGRAM

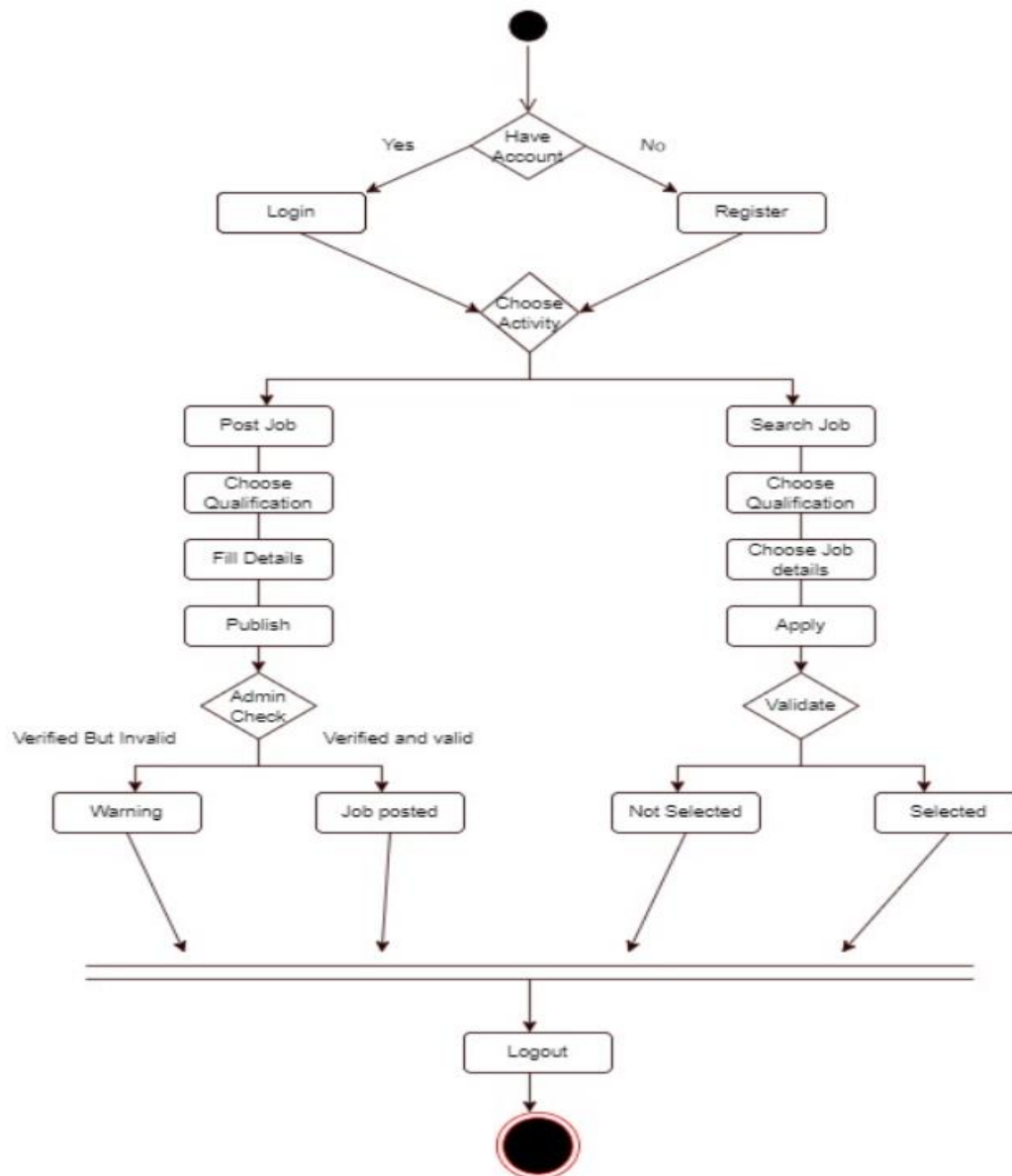


Fig 3: ACTIVITY DIAGRAM

If the user already has an account, he will log in; otherwise, he will create one. There are two types of jobs: post jobs and search jobs. The Publisher selects the qualification and fills in the information required for the position. The admin checks the information, verifies it, and publishes it on the website. If the publisher's information is false, a notice is sent to the publisher, and the information is also shown on the website. The job seeker selects the qualifications and specifics of the position for which he wishes to apply. If he is chosen, he will be notified and given the option to apply for additional positions.

CODING

Login.html

```
<!DOCTYPE html>

<html>

<head>

<link href="basic.css" rel="stylesheet"/>

</head>

<body>

<div class="main_bg">

<div class="logo">

<h2>LOCAL JOBS</h2>

</div>

<input type="checkbox" id="login"/>

<input type="checkbox" id="register"/>

<div class="login">

<div class="section1">

<h1>Sign-In</h1>

<label for="phone">Phone Number</label>

<input type="text" id="userPhone"/>

<label for="phone">Password</label>

<input type="password" id="userPassword"/>

<button id="loginButton">Continue</button>
```

<p>By continuing, you need to agree LOCAL JOBS Conditions for to Use and Privacy Notice.</p>

</div>

<div class="section2">

<hr/>

<p> New to LOCAL JOBS?</p>

<hr/>

</div>

<label class="button" for="register">Create your account</label>

</div>

<div class="register">

<div class="section1">

<h1>Create Account</h1>

<label for="phone">Your Name</label>

<input type="text" id="name"/>

<label for="phone">Phone Number</label>

<input type="text" id="number"/>

<label for="phone">Password</label>

<input type="password" id="password"/>

<button id="registerButton">Continue</button>

<p>By continuing, you agree to LOCAL JOB's Conditions of Use and Privacy Notice.</p>

</div>

<div class="section2">

```
<hr/>

<p> Already have an Account</p>

<hr/>

</div>

<label for="login" class="button">Sign-Up</label>

</div>

</div>

<script src="login.js"></script>

</body>

</html>
```

Login.js

```
let named =document.getElementById("name")

let number =document.getElementById("number")

let password = document.getElementById("password")

let registerButton = document.getElementById("registerButton");

let loginButton = document.getElementById("loginButton");

let userPhone= document.getElementById("userPhone")

let userPassword = document.getElementById("userPassword")

let obj = [{

name:"swathi",

phone:7794826922,

passw:"swathi"

}
```



```

]

localStorage.setItem("name","Swathi")

localStorage.setItem("obj",JSON.stringify(obj))

registerButton.addEventListener("click",function(){

localStorage.setItem("name",named.value)

obj.push({ name:named.value,phone:number.value,passw:password.value})

localStorage.setItem("obj", JSON.stringify(obj))

window.location.href="trail.html"

})

loginButton.addEventListener("click",function(){

let obj1=localStorage.getItem("obj")

for (let i of JSON.parse(obj1)){

if((i.phone==userPhone.value) & (i.passw==userPassword.value)){

window.location.href="trail.html"

}

else{

alert("You don't have any account, please register")

}

}

})

```

Home.html

```

<!DOCTYPE html>

<html lang="en">

```

```

<head>

<meta charset="UTF-8">

<meta http-equiv="X-UA-Compatible" content="IE=edge">

<meta name="viewport" content="width=device-width, initial-scale=1.0">

<link rel="stylesheet" href="local.css">

<title>job-details</title>

</head>

<body>

<div class="container2">

<div class="card">

<h1 style="color: rgb(39, 6, 116);">

..local jobs..  
</h1>

<p>enter all the details:</p>

<p>enter your name:    <input type="text"></p><br>

<p>phone number:    <input type="number"></p><br>

<p>Email-address:    <input type="email"></p><br>

<div class="mn">

<div class="mm"><h3 style="color: rgb(39, 6, 116);">choose action:</h3>

<button>post a job</button>

<button>search for a job</button>

</div>

<div class="mm"><h3 style="color: rgb(39, 6, 116);">choose locality:</h3>

```

```
<button>Anakapalli </button>

<button>Pendurti</button>

<button>Chodavaram</button>

<button>Rambilli</button></div>

</div>

</div>

</div>

</div>

</body>

</html>
```

Jobdetails.html

```
<!DOCTYPE html>

<html lang="en">

<head>

<meta charset="UTF-8">

<meta http-equiv="X-UA-Compatible" content="IE=edge">

<meta name="viewport" content="width=device-width, initial-scale=1.0">

<link rel="stylesheet" href="jobdetails.css">

<title>job-details</title>

</head>

<body>

<div class="container2">

<div class="card">
```

<h1 style="color: rgb(39, 6, 116);">Job details</h1> <p>Enter all the details:</p> <p>Jobrolename<input type="text"/></p> <p>Enterprisesname<input type="text"/></p> <p>Address<input type="text"/></p> <p>Gender<input type="text"/></p> <p>Age-category<input type="text"/></p> <p>Timings<input type="text"/></p> <p>No.ofvaccancy<input type="text"/></p> <p>Salary<input type="text"/></p> <p><button onclick="window.location.href='posttq.html';">Publish</button></p>

Jobdetasils.css

$$*\{$$

```
margin: 0;

padding: 0;

}

body{

background-color: white;

}

h1{

text-align: center;

font-size: 25px;

margin-top: 20px;

}

p{

line-height: 40px;

text-align: center;

font-size: 15px;

font-style: Playbill;

}

/* table{

margin: 90px;

border: thick;

color: aqua;

} */

input{
```

```
height: 22px;

width: 180px;

flex: 0 0 300px;

margin-left: 20px;

}

.card{

display: flex;

flex-direction: column;

align-items: center;

background-color: rgba(129, 209, 247, 0.425);

width: 450px;

height: 680px;

box-shadow: 8px 8px 10px rgba(0, 0, 0, 0.534);

border-radius: 20px;

}

.container2{

display: flex;

justify-content: center;

align-items: center;

background-color: aliceblue;

width: 100%;

height: 100vh;

}
```

```
button{

background-color: rgb(39, 6, 116);

height: 36px;

width: 180px;

color: white;

border-radius: 25px;

border-width: 0px;

margin-top: 30px;

padding:8px;

}

.g{

background-color: transparent;

color: rgb(39, 6, 116);

font-size: 20px;

margin-left: -180px;

border-style: none;

}
```

qualification.html

```
<!DOCTYPE html>

<html lang="en">

<head>

<meta charset="UTF-8">

<meta http-equiv="X-UA-Compatible" content="IE=edge">
```

```

<meta name="viewport" content="width=device-width, initial-scale=1.0">

<link rel="stylesheet" href="qualification.css">

<title>job-details</title>

</head>

<body>

<div class="container2">

<div class="card">

<table>
  <tr>
    <th style="color: rgb(39, 6, 116);">
      <button class="g"
        onclick="window.location.href='trail.html';"></button>
    </th>
  </tr>
</table>

Choose Qualification:<br>

</h1>

<p>
  <button onclick="window.location.href='jobdetails.html';">10th qualification
</button><br>

  <button onclick="window.location.href='jobdetails.html';">12th qualification
</button><br>

  <button onclick="window.location.href='jobdetails.html';">Diploma qualification
</button><br>

  <button onclick="window.location.href='jobdetails.html';">ITI qualification </button><br>

  <button onclick="window.location.href='jobdetails.html';">Degree
  qualification</button><br>

  <button onclick="window.location.href='jobdetails.html';">Engineering
  qualification</button><br>

  <button onclick="window.location.href='jobdetails.html';">PG/UG qualification
</button><br>

  <button onclick="window.location.href='jobdetails.html';">No qualification</button>

```


</p>

</table>

</div>

</div>

</body>

</html>

joblocker.css

{

margin: 0;

padding: 0;

}

body{

background-color: white;

}

h1{

text-align: center;

font-size: 25px;

margin-top: 20px;

}

p{

line-height: 40px;

text-align: center;

font-size: 15px;

```
font-style: italic;

}

/* table{

margin: 90px;

border: thick;

color: aqua;

} */

input{

height: 22px;

width: 180px;

flex: 0 0 300px;

margin-left: 20px;

}

.card{

display: flex;

flex-direction: column;

align-items: center;

background-color: rgba(129, 209, 247, 0.425);

width: 450px;

height: 680px;

box-shadow: 8px 8px 10px rgba(0, 0, 0, 0.534);

border-radius: 20px;

}
```

```
.container2{  
  
display: flex;  
  
justify-content: center;  
  
align-items: center;  
  
background-color:aliceblue;  
  
width: 100%;  
  
height: 100vh;  
  
}  
  
.g{  
  
background-color: transparent;  
  
color: rgb(39, 6, 116);  
  
font-size: 20px;  
  
margin-left: -200px;  
  
border-style: none;  
  
}  
  
.b{  
  
background-color: rgb(39, 6, 116);  
  
height: 36px;  
  
width: 180px;  
  
color: white;  
  
border-radius: 25px;  
  
border-width: 0px;  
  
margin-top: 30px;
```

```
padding:8px;

}

button{

background-color:transparent;

height: 36px;

width: 180px;

color:rgb(39, 6, 116);

border-radius: 25px;

border-width: 1px;

margin-top: 30px;

padding:8px;

}
```

apply .html

```
<!DOCTYPE html>

<html lang="en">

<head>

<meta charset="UTF-8">

<meta http-equiv="X-UA-Compatible" content="IE=edge">

<meta name="viewport" content="width=device-width, initial-scale=1.0">

<link rel="stylesheet" href="apply.css">

<title>job-details</title>

</head>

<body>
```

[illegible]

</html>

trail.html

<!DOCTYPE html>

<html lang="en">

<head>

<meta charset="UTF-8">

<meta http-equiv="X-UA-Compatible" content="IE=edge">

<meta name="viewport" content="width=device-width, initial-scale=1.0">

<link rel="stylesheet" href="style.css">

<title>job-details</title>

</head>

<body>

<div class="container2">

<div class="card">

<table>

<h1 style="color: rgb(39, 6, 116);">LOCAL JOBS
</h1>

<p>Enter all the details:</p>

<p>Enter your name: <input type="text"></p>

<p>Phone number: <input type="number"></p>

<p>Email-address: <input type="email"></p>

</table>

<div class="mm">

<h3 style="color: rgb(39, 6, 116);">*choose locality:</h3>

<select id="choose here" name="branch">

<option value="akp">Anakapalli</option>

<option value="akp">Allikhanudupalem</option>

<option value="akp">Bhatlapudi</option>

<option value="akp">chintha nippula agraharam</option>

<option value="akp">Golagam</option>

<option value="akp">Gopalapuram</option>

<option value="akp">Jaganathapuram</option>

<option value="akp">Koduru</option>

<option value="akp">Kondupalem</option>

<option value="akp">Koppaka</option>

<option value="akp">Kunchangi</option>

<option value="akp">Kundram</option>

<option value="akp">Makavaram</option>

<option value="akp">Mamidipalem</option>

<option value="akp">Maredupudi</option>

<option value="akp">Maredupudi Agraharam</option>

<option value="akp">Marturu</option>

<option value="akp">Mettapalem</option>

<option value="akp">Papayya Palem</option>

<option value="akp">Papayya Santha Palem</option>

<option value="akp">Pisinikada</option>

```

<option value="akp">Rajupalem</option>

<option value="akp">Rebaka</option>

<option value="akp">Sampathipuram</option>

<option value="akp">Sankaram</option>

<option value="akp">Seethanagaram</option>

<option value="akp">Tagarampudi</option>

<option value="akp">Thummapala</option>

<option value="akp">Valluru</option>

<option value="akp">Venkupalem</option>

<option value="akp">Vetajangalapalem</option>

<option value="akp">Vooderu</option>

</select>

</div>

<div class="mm">

<h3 style="color: rgb(39, 6, 116);">*choose action:</h3><br>

<button onclick="window.location.href='qualification.html';">Post a job</button>

<button onclick="window.location.href='qualifications.html';">Search for a
job</button>

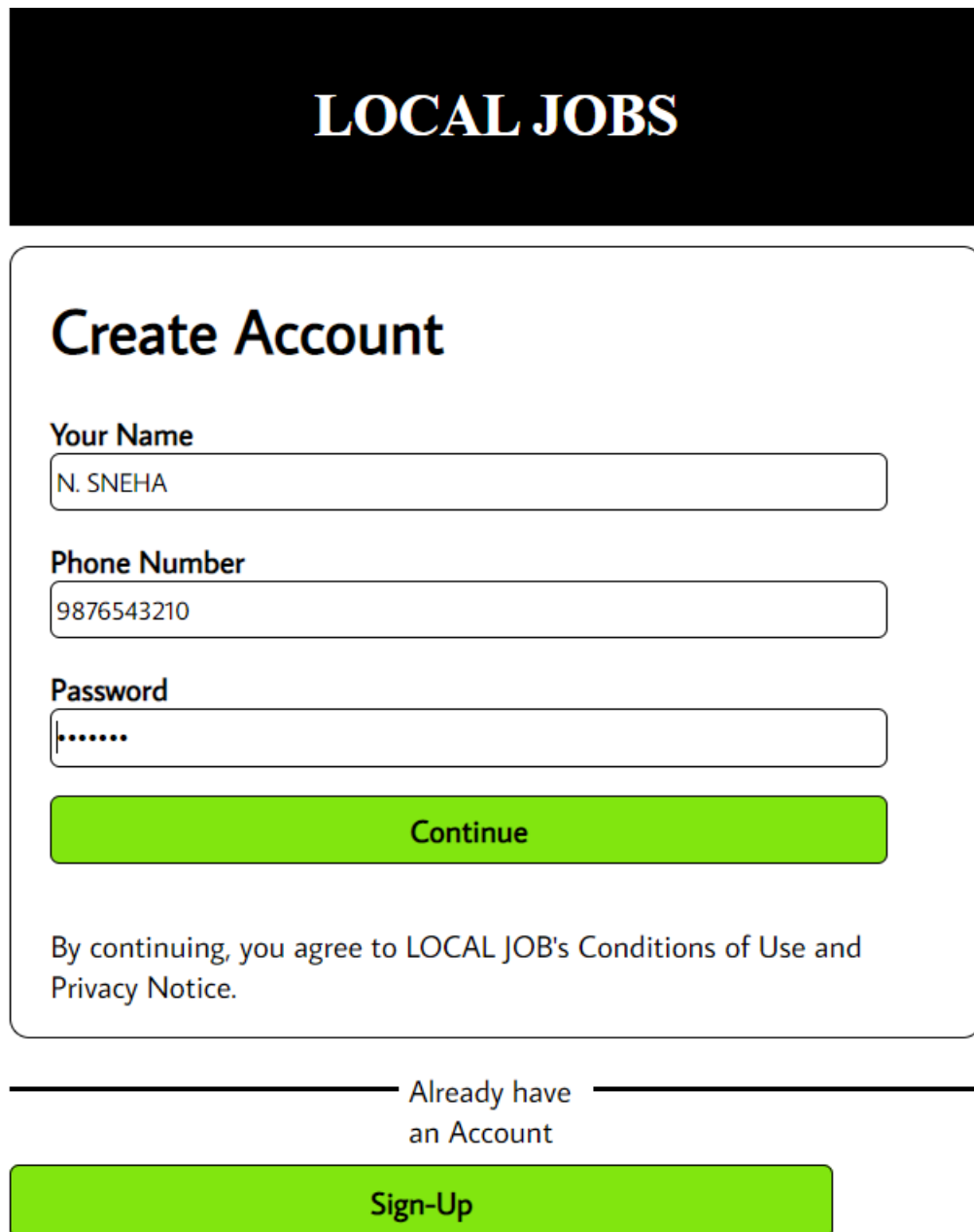
</div>

</body>

```


OUTPUT SCREENS

SIGN-UP SCREEN



The image shows a mobile application sign-up screen for 'LOCAL JOBS'. At the top, a black header bar contains the text 'LOCAL JOBS' in white, bold, sans-serif font. Below the header is a white rounded rectangle containing the sign-up form. The form has a title 'Create Account' in bold black font. It includes three input fields: 'Your Name' with the text 'N. SNEHA', 'Phone Number' with the text '9876543210', and 'Password' with masked characters '.....'. Each input field has a thin grey border. Below these fields is a large orange button with the text 'Continue' in white. Under the button, there is a line of text: 'By continuing, you agree to LOCAL JOB's Conditions of Use and Privacy Notice.' At the bottom of the screen, there is a horizontal line. Below this line, the text 'Already have an Account' is centered. Below that is another orange button with the text 'Sign-Up' in white.

LOCAL JOBS

Create Account

Your Name
N. SNEHA

Phone Number
9876543210

Password
.....

Continue

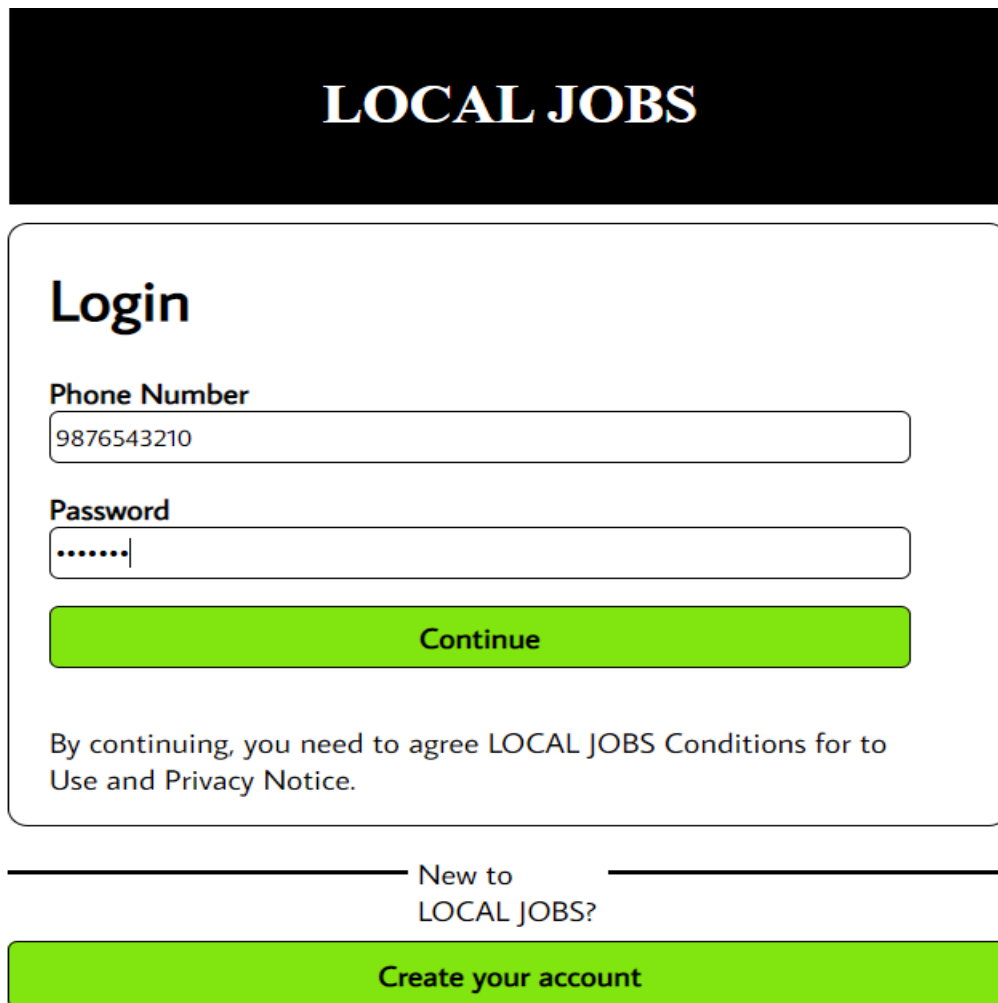
By continuing, you agree to LOCAL JOB's Conditions of Use and Privacy Notice.

Already have an Account

Sign-Up

Fig 1: Login Screen

LOGIN SCREEN



The image shows a mobile application interface for 'LOCAL JOBS'. At the top, a black header bar contains the text 'LOCAL JOBS' in white, bold, serif font. Below this is a white rounded rectangle containing the 'Login' section. The 'Login' section has a title 'Login' in bold black font. It includes two input fields: 'Phone Number' with the value '9876543210' and 'Password' with masked characters '.....'. Below these fields is a green button labeled 'Continue'. Under the button, a line of text states: 'By continuing, you need to agree LOCAL JOBS Conditions for to Use and Privacy Notice.' Below the login section, a horizontal line separates it from the sign-up section. The sign-up section features the text 'New to LOCAL JOBS?' centered above a green button labeled 'Create your account'.

LOCAL JOBS

Login

Phone Number

9876543210

Password

.....

Continue

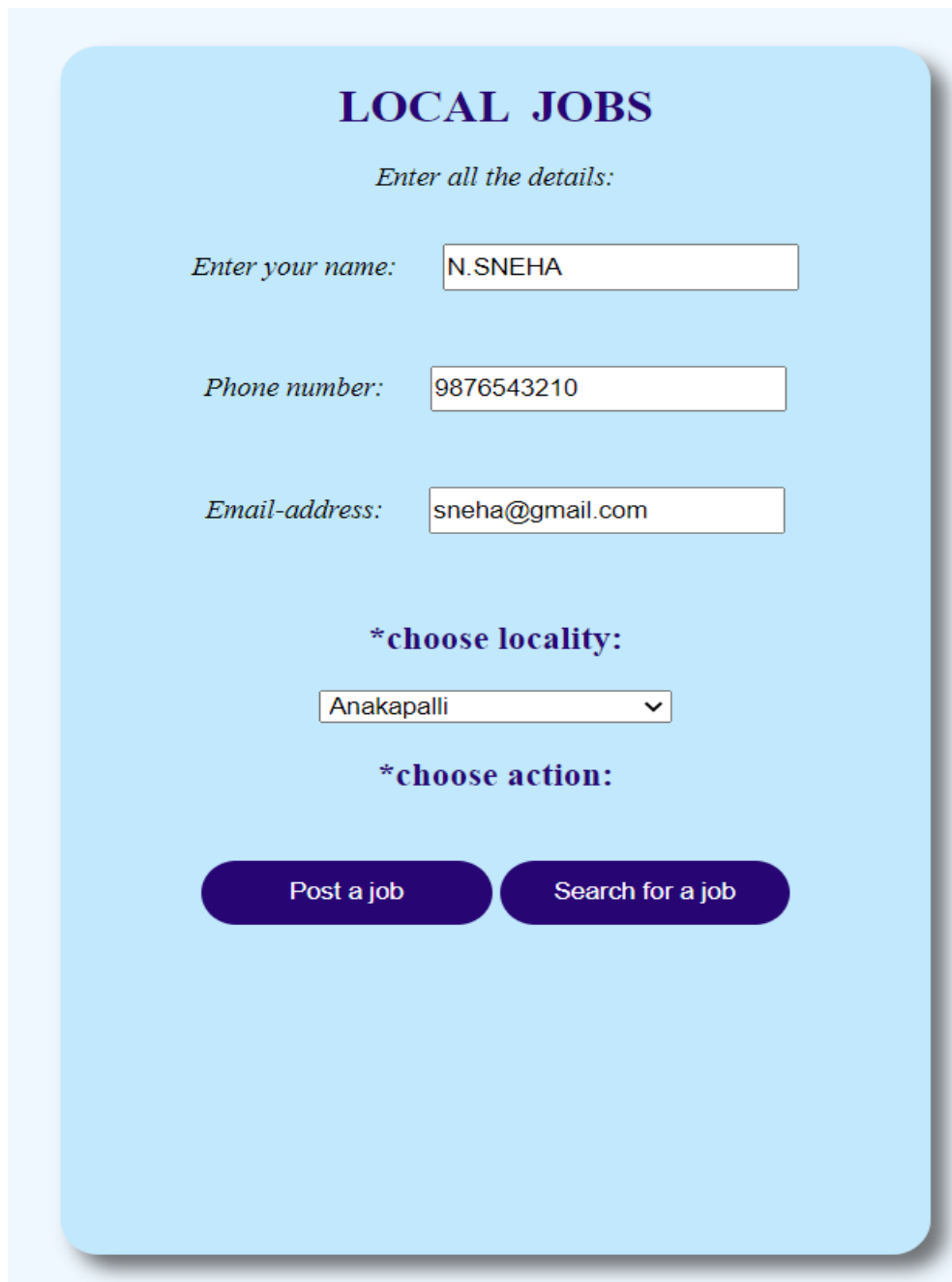
By continuing, you need to agree LOCAL JOBS Conditions for to Use and Privacy Notice.

New to
LOCAL JOBS?

Create your account

Fig 2:Sign-Up Screen

HOME SCREEN



The image shows a mobile application home screen for 'LOCAL JOBS'. The screen has a light blue background with a darker blue rounded rectangle in the center containing the form. The form is titled 'LOCAL JOBS' in bold purple text. Below the title is the instruction 'Enter all the details:'. There are three input fields: 'Enter your name:' with the value 'N.SNEHA', 'Phone number:' with the value '9876543210', and 'Email-address:' with the value 'sneha@gmail.com'. Below these is a dropdown menu labeled '*choose locality:' with 'Anakapalli' selected. Another dropdown menu labeled '*choose action:' is below that. At the bottom are two dark blue buttons: 'Post a job' and 'Search for a job'.

LOCAL JOBS

Enter all the details:

Enter your name:

Phone number:

Email-address:

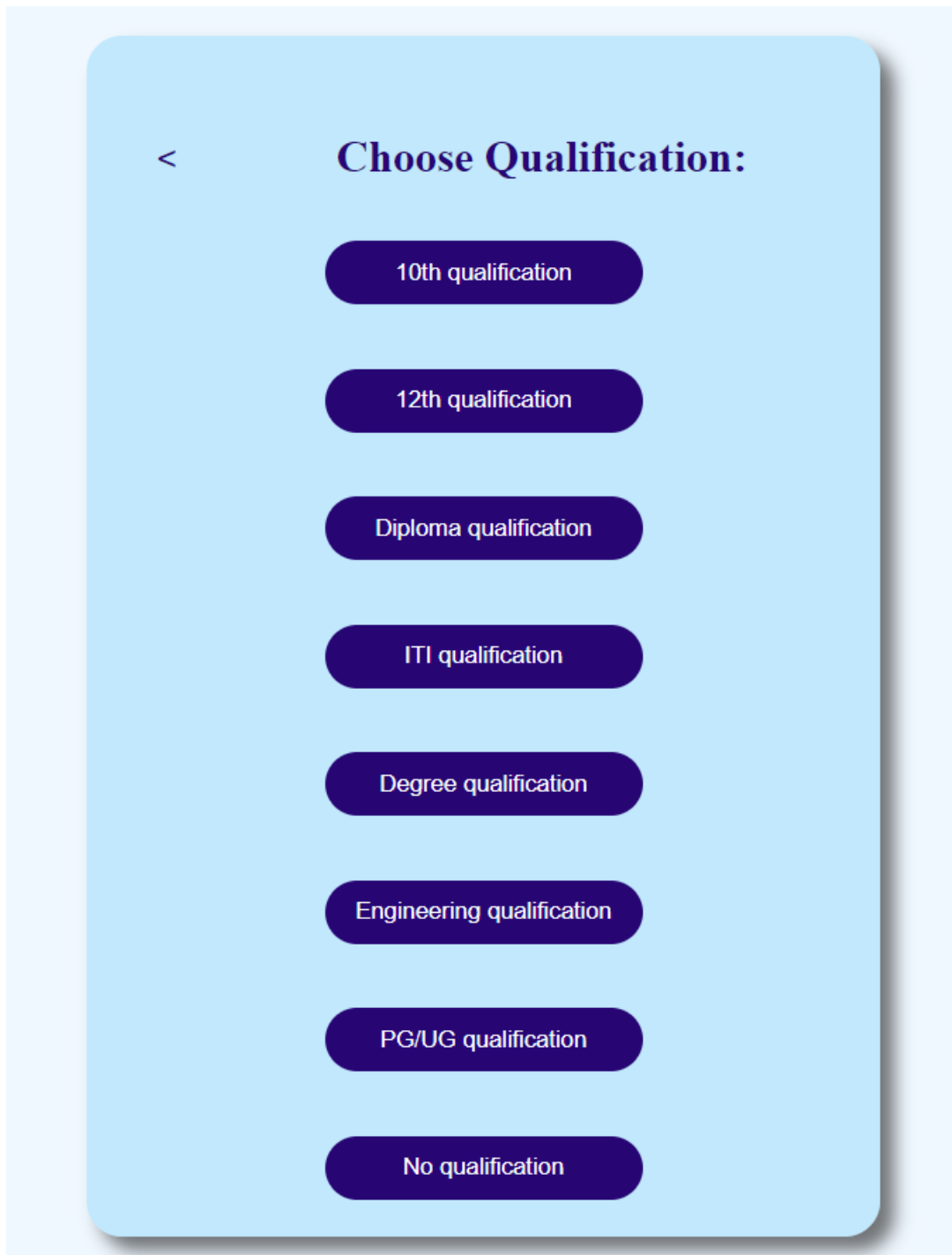
***choose locality:**

▼

***choose action:**

Fig 3: Home Screen

JOB PUBLISHER SCREENS



The image shows a mobile application screen for job publishers. It features a light blue background with a darker blue rounded rectangle in the center. At the top left of this rectangle is a back arrow icon. To its right is the title 'Choose Qualification:'. Below the title, there is a vertical list of eight rounded rectangular buttons, each containing a qualification level. The buttons are stacked vertically with equal spacing between them.

< **Choose Qualification:**

- 10th qualification
- 12th qualification
- Diploma qualification
- ITI qualification
- Degree qualification
- Engineering qualification
- PG/UG qualification
- No qualification

Fig 4: Job Publisher Screen

<

Job details

Enter all the deatils:

Job role name :

Delivery Boy

Enterprises name :

Workex Solutions

Address :

Anakapalli

Gender :

Male

Age-category :

17 above

Timings :

9:00 am to 7:00pm

No.of vaccancy :

5

Salary :

10k-15k per month

Publish

Fig 4: Job Details Screen

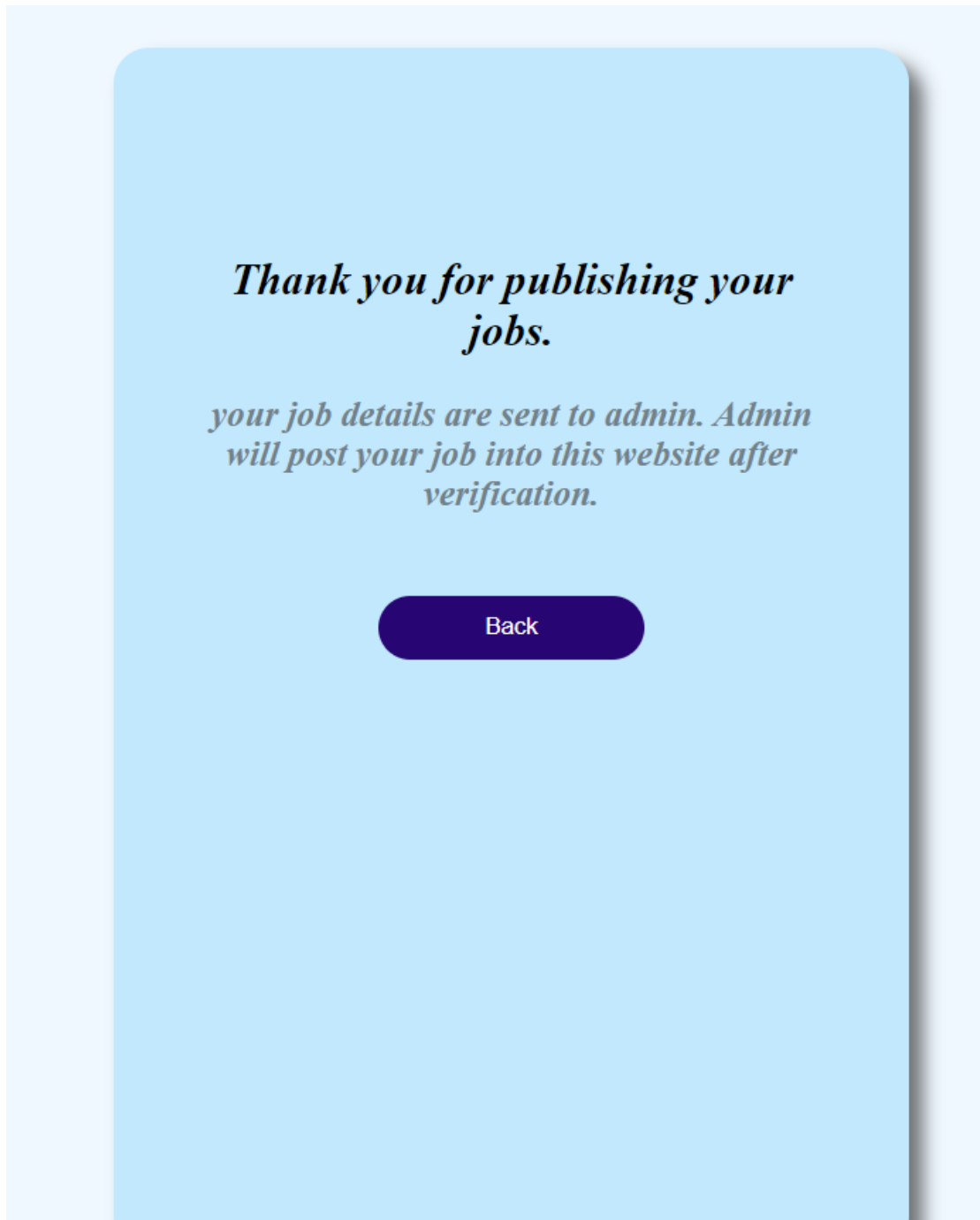


Fig 6: Thank you Screen

ADMIN SCREEN

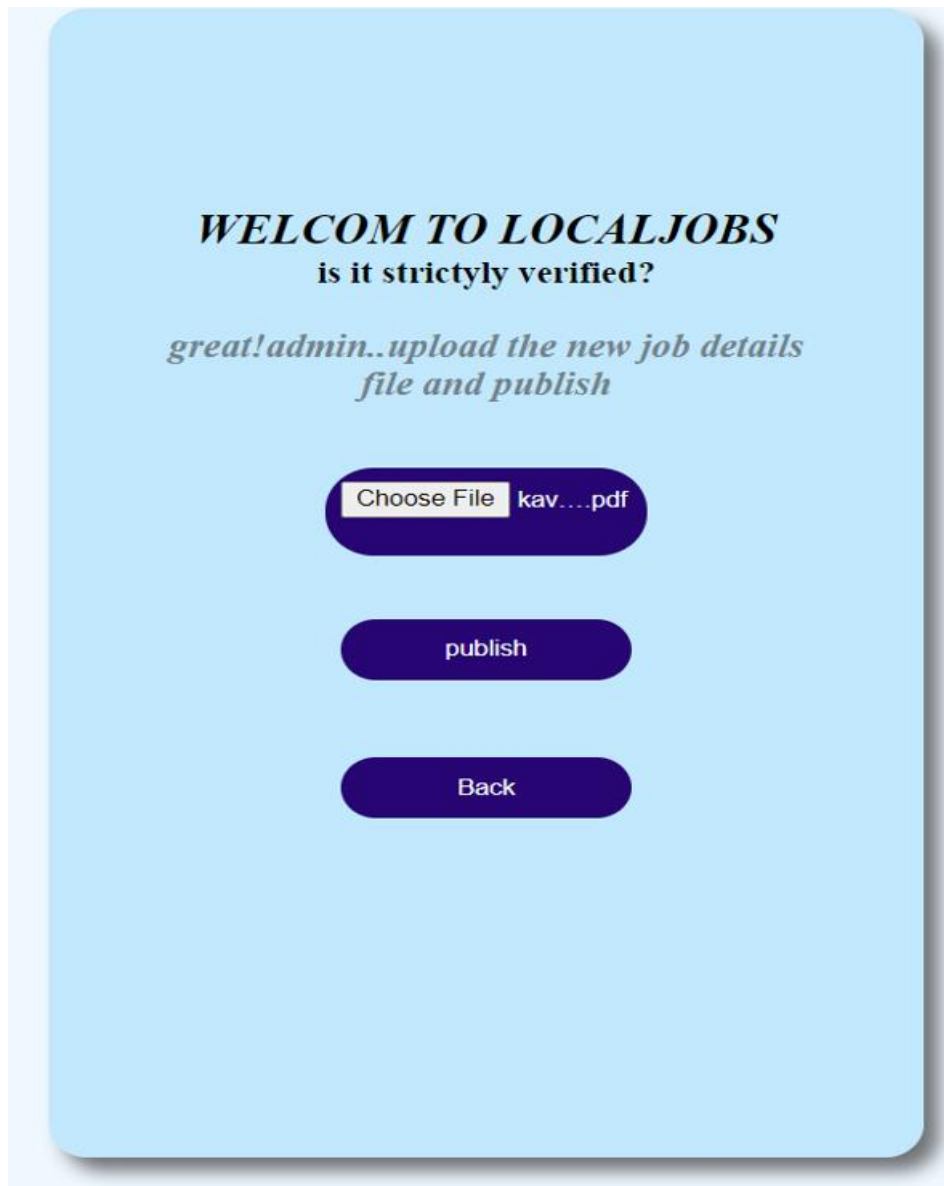


Fig 7 : Admin Screen

new post

Job deatils are posted like this:

Job role name :

Delivery Boy

Enterprises name :

Workex Solutions

Address :

Anakapalli

Gender :

Male

Age-category :

17 above

Timings :

9.00am to 7.00pm

No.of vaccancy :

5

Salary :

10k-15k

back

Fig 8: New Post Screen

JOB SEEKER SCREENS

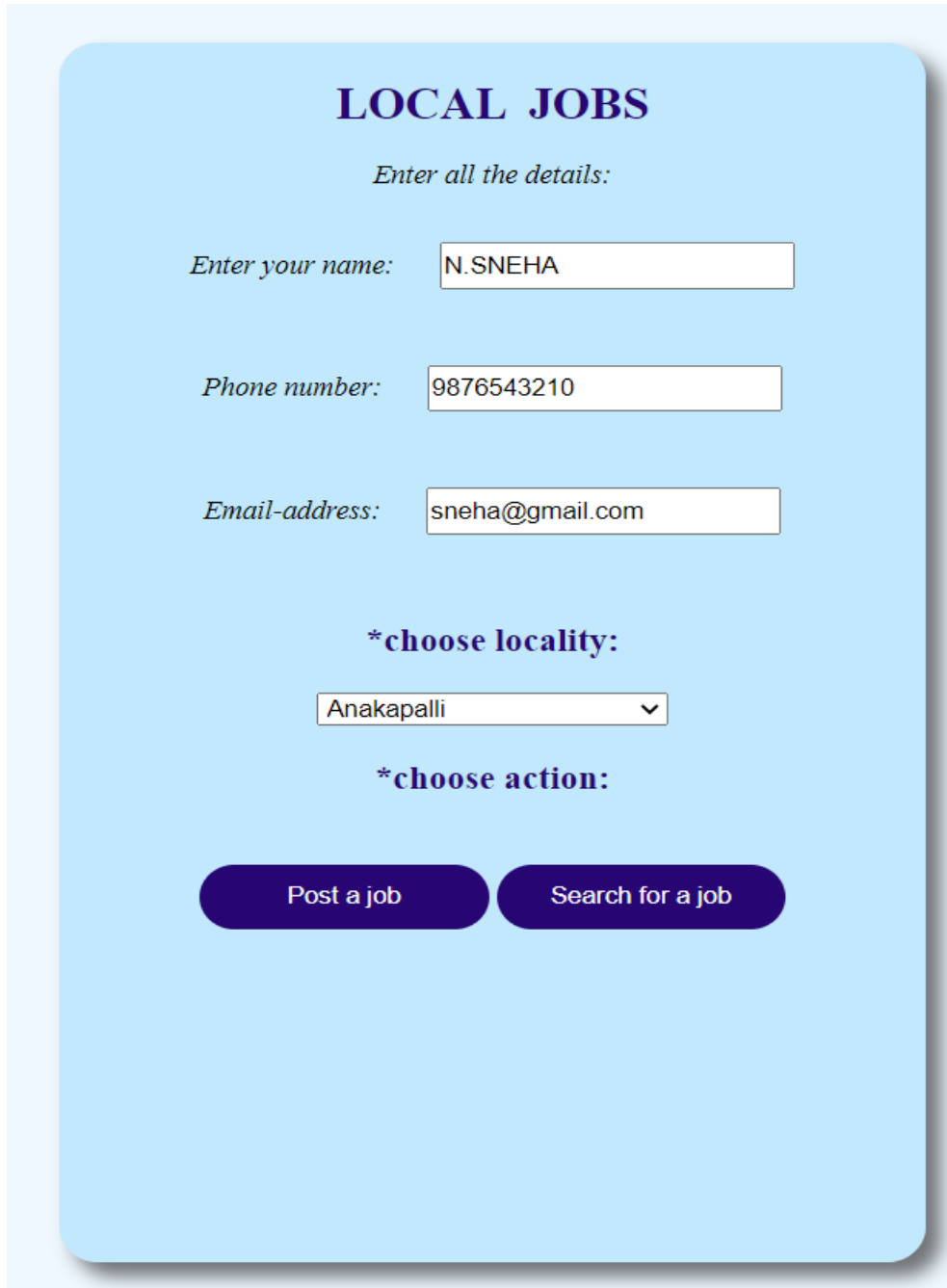
The image shows a user interface for a job seeker screen. It features a light blue background with a central white rounded rectangle containing the form. The title 'LOCAL JOBS' is at the top in bold purple. Below it is the instruction 'Enter all the details:'. There are three input fields: 'Enter your name:' with 'N.SNEHA', 'Phone number:' with '9876543210', and 'Email-address:' with 'sneha@gmail.com'. Below these is a dropdown menu for '*choose locality:' with 'Anakapalli' selected. Then another dropdown for '*choose action:'. At the bottom are two dark blue buttons: 'Post a job' and 'Search for a job'.

Fig 9: Job Seeker Screen



Fig 10: Qualification Screen

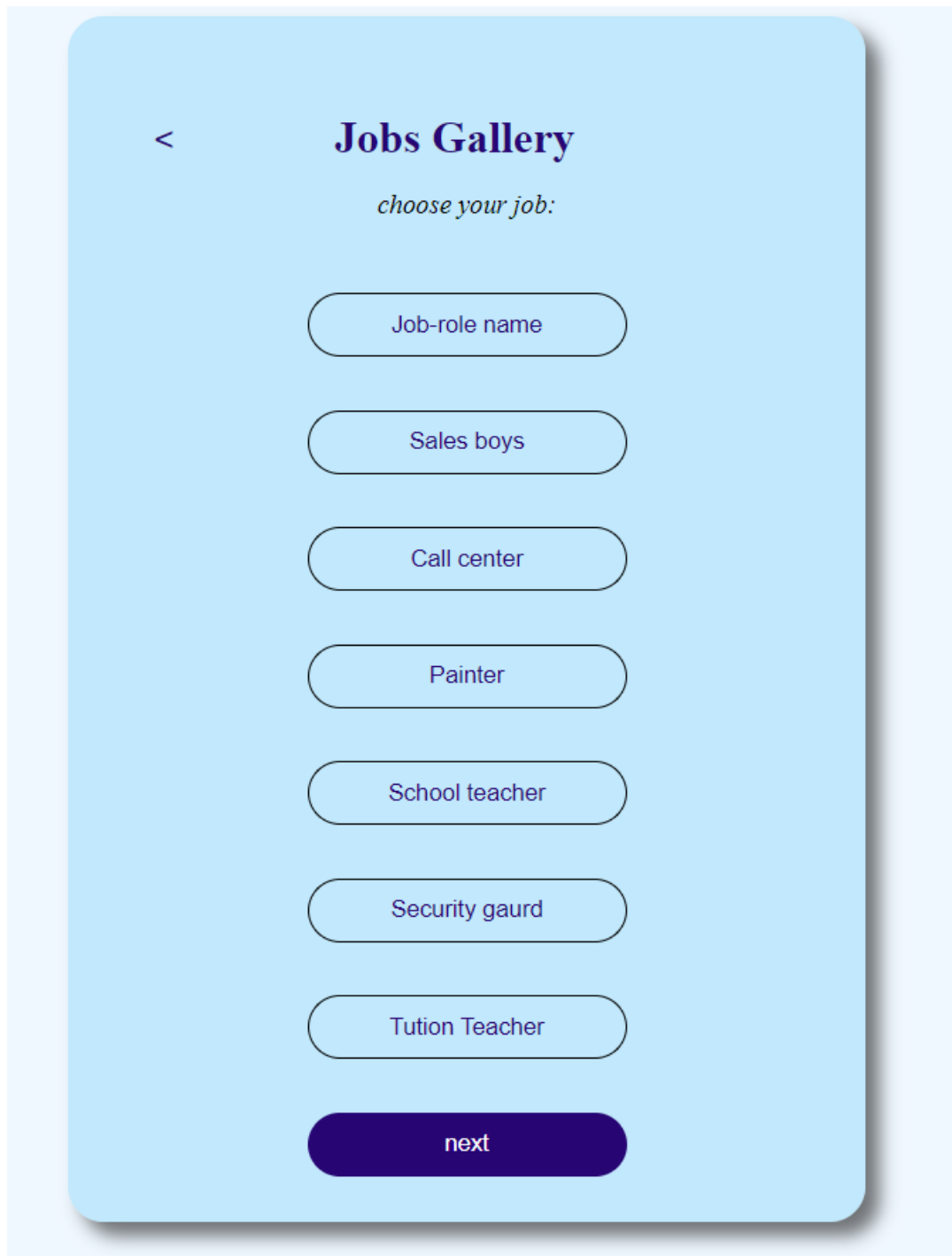


Fig 11: Job Gallery Screen

<

Apply Here

Job deatils:

Job role name :

Delivery Boy

Enterprises name :

Workex Solutions

Address :

Anakapalli

Gender :

Male

Age-category :

17 above

Timings :

9:00am to 7:00pm

No.of vaccancy :

5

Salary :

10k-15k per month

upload file

apply now

Fig 12: Fill Details Screen

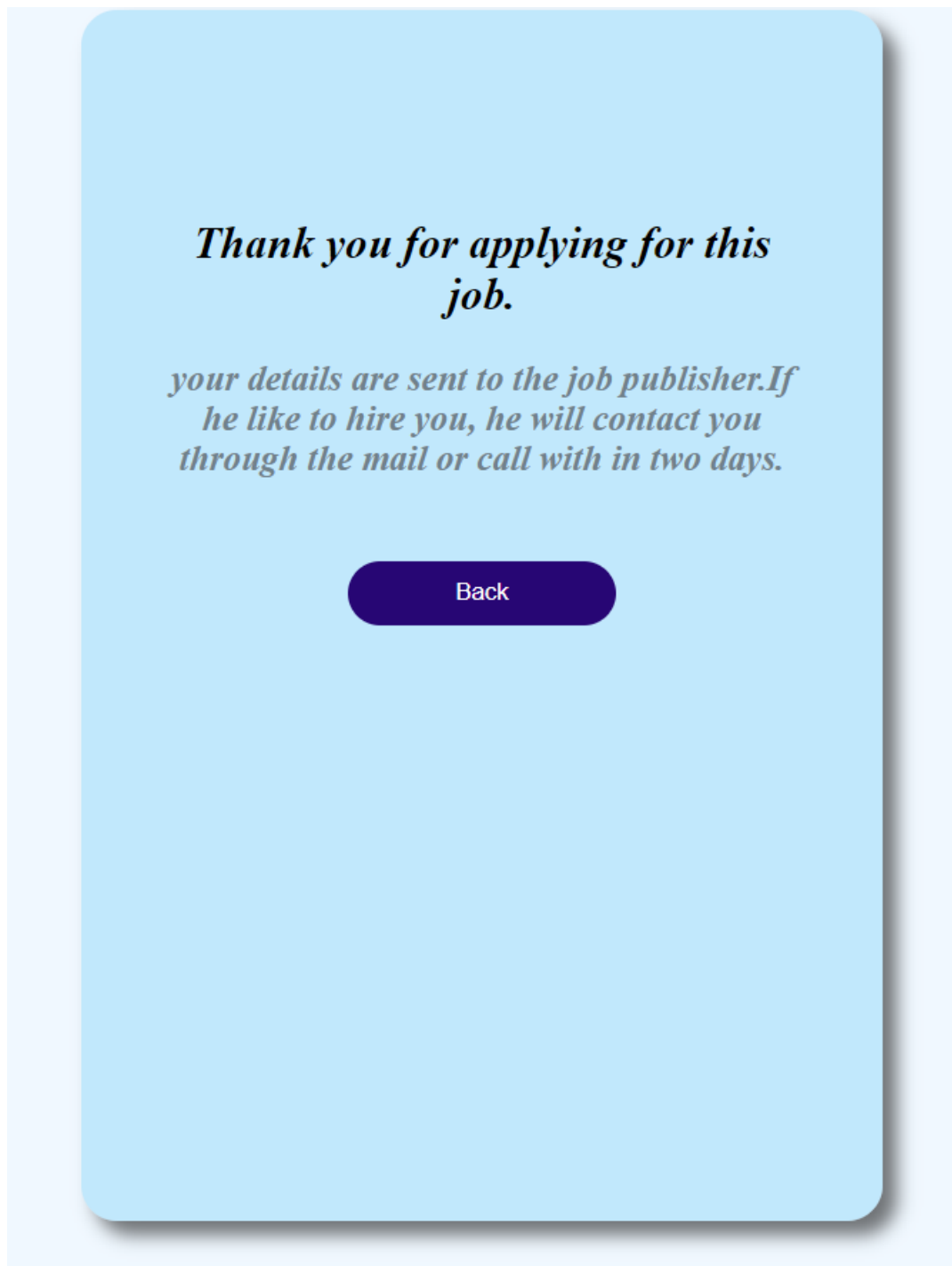


Fig 13: Thanks Screen

CONCLUSION

Our project's main goal is to create a communication link between job publishers and job seekers across the area. So that they may contact with one another and discuss any job-related issues on both ends. The main duty will be difficult for most job searchers since they lack information of available jobs and employment in their immediate vicinity. The major goal of our initiative is to produce beneficial outcomes for both job publishers and job seekers.