Emerging Technologies (Full Stack)

Unit 1 Foundations of Full Stack Development

Q. Give any two difference between application and websites.

→ **Website** – It has static content, with the main focus of informing the user about the business vision/ products/ benefits. The website doesn't require user authentication. A websites is a series of web pages that are typically accessed through a web browser and are hosted on a web server.

Web Application – It has dynamic content, designed specifically for information with the end-users. Most web applications require user authentication. A web application is a software program that is accessed through a web browser and are hosted on a web server.

Q. Why is UX Design Important?

→ It tries to fulfil the user's needs. UX measures how a consumer feels when interacting with a system.

Q. What is a UX Design Process?

→ User experience design is a process of enhancing user satisfaction with a product by improving the usability, accessibility, and pleasure provided in the interaction with the product.

A UX design process is a process that starts with research and ends with a solution. Continuous process - research, prototyping, testing, and refining. The first step in the design process is to conduct user research to identify the problem or opportunity to be addressed by the product or service.

Q. List the different ways which is used for component interaction.

→ Asking for data, taking an action synchronously, taking an action asynchronously

Q. What is goal of modularity?

→ The goal of modularity is to partition the application in such a way that it remains flexible, maintainable, and stable even as features and technologies are added and removed.

Q. Explain system architecture in detail.

→ Two common types of system architectures : "monolithic" and "microservices".

Monoliths can be easier to build initially, but have the downside of becoming large and difficult to change if multiple teams work on the same codebase. Monolith is a single codebase with a single application that performs all the functions of that system.

Microservices require a strong platform to grow from, but can be easier to adapt and change in the future.

A microservice is a small service. It can be deployed and scaled independently of any other part of the system, and communicates through well-defined APIs.

• To designing a system at the application level is to make each component as small as it needs to be.

Q. Explain the different communication type between component in a system.
→ **1. Requests using HTTP**: For requesting data or making synchronous changes, HTTP is a natural fit, and a
particular type of HTTP interaction known as REST. HTTP GET can be used for requesting data, and HTTP POST or PUT should be used for making changes.
2. Message queues: For asynchronous actions, where it doesn't matter to the user whether it succeeds or not. Message queues are a good fit. However, if the code is front-end code, it is often easy to make a fire-and-forget XHR call (XMLHttpRequest, a browser API for making HTTP requests, and an important part of the asynchronous JavaScript and XML-AJAX technique). RESTful queries can be used to interact with jobs on a message queue.

Q. Write short note on component interactions?
→ Relationships between the components. How they communicate. The different scenarios through which a user may come to use the system, and the paths and workflows a user takes to accomplish their goals. These are called user journeys, and it's useful to identify which front-end component is used. Determine the nature of that interaction.

Q. Explain in detail three different patterns used for communication in between modules.
→ Modules should have low coupling between each other, it is common for modules to communicate with each other. There are several loosely coupled communication patterns, each with their own strengths.
The following are some of these patterns:
Loosely coupled events - A module can broadcast that a certain event has occurred. Other modules can subscribe to those events so they will be notified when the event occurs. A design that relies too heavily on events can become hard to maintain, especially if many events have to be orchestrated together to fulfill a single task. In that case, it might be better to consider a shared service.
Shared services - A shared service is a class that can be accessed through a common interface. shared services are found in a shared assembly and provide system-wide services, such as authentication, logging, or configuration.
Shared resources - If you do not want modules to directly communicate with each other, you can also have them communicate indirectly through a shared resource, such as a database or a set of web services.

Q. Differentiate between application and module.
→ An **application** is a software program that's designed to perform a specific function directly for the user or, in some cases, for another application program. Which means that an app can contain many modules.
A **module** is a software component, and an API is instructions, and possibly some tools, for using and communicating with a software component.

Unit 2 Front End

Q. Give the advantages of Bootstrap.

→ Easy to use, Responsive features, Mobile-first approach, Browser compatibility.

Q. Why do we need to use Bootstrap?

→ Free front-end framework for faster and easier web development.

Q. List some features of Bootstrap.

→ Grid system, Browser compatibility, Responsive features, Topography, Customizable, jQuery support

Q. What is a Bootstrap Container? & Compare and contrast container and container-fluid.

→ A Bootstrap container is a basic layout element that's used to contain, pad, and align content within it. Containers are required when using Bootstrap's default grid system.

The **.container** class provides a responsive fixed width container

The **.container-fluid** class provides a full width container, spanning the entire width of the viewport

Q. What is Bootstrap Grid System?

→ 1. Rows must be placed within a .container (fixed-width) or .container-fluid (full-width) for proper alignment and padding

2. Use rows to create horizontal groups of columns

3. Content should be placed within columns, and only columns may be immediate children of rows

4. Predefined classes like .row and .col-sm-4 are available for quickly making grid layouts

5. Grid columns are created by specifying the number of 12 available columns you wish to span. For example, three equal columns would use three .col-sm-4

Q. Discuss Bootstrap table and various classes that can change the appearance of the table.

→ The .table class adds basic styling to a table.

• **Striped Rows** : The .table-striped class adds zebra-stripes to a table. <table class="table table-striped">

• **Bordered Table :** The .table-bordered class adds borders on all sides of the table and cells. <table class="table table-bordered">

• **Hover Rows** : The .table-hover class adds a hover effect (grey background color) on table rows <table class="table table-hover">

• **Contextual Classes**: Contextual classes can be used to color table rows or table cells. The classes that can be used are: .active, .success, .info, .warning, and .danger.

Q. What is Button group and which class is used for basic button group?

→ Bootstrap allows you to group a series of buttons together (on a single line) in a button group. Use a <div> element with class .btn-group to create a button group.

Vertical Button Groups - Use the class .btn-group-vertical to create a vertical button group

Justified Button Groups - To span the entire width of the screen, use the .btn-group-justified class <div class="btn-group btn-group-justified">

Q. What are bootstrap alerts and how will you create them?
→ Bootstrap alerts are contextual feedback messages that provide information for user actions. Bootstrap provides an easy way to create predefined alert messages. Alerts are created with the .alert class, followed by one of the four contextual classes .alert-success, .alert-info, .alert-warning or .alert-danger
<div class="alert alert-success alert-dismissible">
Q. Explain the uses of carousel plugin in Bootstrap.
→ Carousel allows to navigate through a collection of images in a sequential fashion, moving to the previous/next one through the arrows on the sides. The carousel is a slideshow for cycling through a series of content, built with CSS 3D transforms and a bit of JavaScript. It works with a series of images, text, or custom markup.
Q. How can you create Nav elements in Bootstrap?
→ A standard navigation bar is created with <nav class="navbar navbar-default">.
Q. What are grid classes in Bootstrap?
→ 1 .col- (extra small devices - screen width less than 576px)
2. .col-sm- (small devices - screen width equal to or greater than 576px)
3. .col-md- (medium devices - screen width equal to or greater than 768px)
4. .col-lg- (large devices - screen width equal to or greater than 992px)
5. .col-xl- (xlarge devices - screen width equal to or greater than 1200px)
Q. Why accessibility is important in web design?
→ Accessibility is often used to talk about assistive technologies (AT), like screen readers; or larger font size; or considerations specifically for people who have disabilities; If the text is too small, or if there is poor color contrast (for example, a light gray on white), then it can be hard
Q. Discuss accessibility in UI.
→ Avoiding Common Mistakes in UI : 1. Hover and Focus Styling, 2. The Order of Headings, 3. Multiple h1s, 4. Skip Links, 5. Buttons vs Anchors, 6. The Correct Use of an alt Attribute.
Q. Explain AngularJS MVC architecture in detail.
→ MVC stands for Model View Controller. It is a software design pattern for developing web applications. It is very popular because it isolates the application logic from the user interface layer and supports separation of concerns. The MVC pattern is made up of the following three parts: **1. Model**: It is responsible for managing application data. It responds to the requests from view and to the instructions from controller to update itself.
2. View: It is responsible for displaying all data or only a portion of data to the users. It also specifies the data in a particular format triggered by the controller's decision to present the data. They are script-based template systems such as JSP, ASP, PHP and very easy to integrate with AJAX technology. **3. Controller**: It is responsible to control the relation between models and views. It responds to user input and performs interactions on the data model objects. The controller receives input, validates it, and then performs business operations that modify the state of the data model.

Unit 3 Middleware

Q. What is Middleware?

→ A middleware component serves as a bridge between frontend and backend.

Middleware is software that provides a bridge between operating systems and the applications that run on them, behaving as a discrete transition layer.

middleware acts as the "middleman," facilitating communication, bridging the gaps between tools, databases, and applications, and ultimately providing unified services to end-users.

Middleware is software that different applications use to communicate with each other.

Q. Why middleware is important?

→ Developers use middleware to support application development and simplify design processes.

This leaves them free to focus on business logic and features instead of connectivity between different software components.

Without middleware, developers would have to build a data exchange module for each software component that connects to the application.

Q. Discuss the different factors consider in middleware platform.

→ 1. Company support, 2. Connectors, 3. Costs, 4. Expansion ability, 5. Flexibility, 6. Functionality, 7. Usability

Q. How middleware works?

→ • Android users depends on middleware every time they use any phone application. The Android operating system is built on a modified Linux kernel, so developers need to build the application with the need to communicate with Linux in mind.

• Linux needs to communicate with an app, but it needs to use the Android OS as middleware in order to successfully do so. see, the request from the application needs to communicate with the back end.

• In order to do that, the Android operating system will: 1. Send the request back, 2. Receive the data in response, 3. Transform the data for the application.

Q. What is Object Request Broker middleware?

→ In distributed computing, an object request broker (ORB) is a middleware, which allows program calls to be made from one computer to another via a computer network, providing location transparency through remote procedure c

Q. Explain the types of middleware in details.

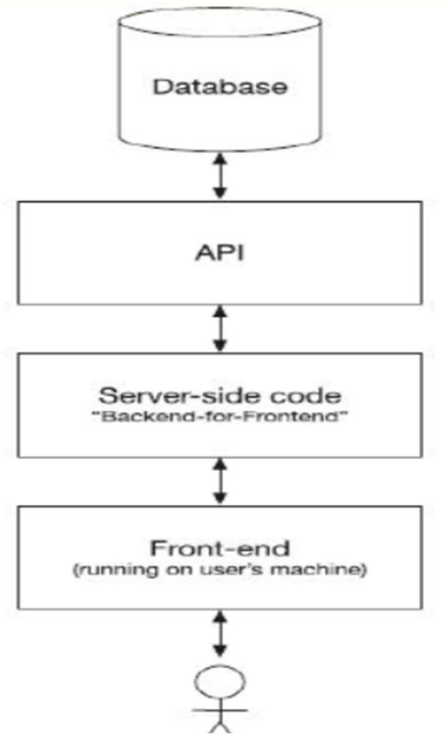
→ **1. Message-Oriented Middleware** - This middleware infrastructure supports message receiving and sending over distributed applications. It translates and transforms messages between applications and routs them, so they arrive at the correct components and in the proper order. Which is an infrastructure that allows communication and exchanges the data (messages). It involves the passing of data between applications using a communication channel that carries self-contained units of information (messages).In a MOM-based communication environment, messages are sent and received asynchronously.

2. Object Middleware - Object middleware, also called an object request broker, gives applications the ability to send objects and request services via an object oriented system. In short, it manages the communication between objects. It is runtime software that enables objects (components) to work cooperatively with a container program or another object, even if the software is distributed across multiple computers.

3. Remote Procedure Call (RPC) Middleware - It calls procedures on remote systems and is used to perform synchronous or asynchronous interactions between applications or systems. It is usually utilized within a software application. RPC is used to call other processes on the remote systems like a local system.

4. API Middleware - An API is middleware software that communicates information between a front end and a back end. A common area that people will find APIs —the back-end only communicates with the APIs, which then run the data to the front-end. This allows web services to be completely customizable, instead
of being locked into a rigid, monolithic provision. API middleware enables developers to create and manage their application's APIs.

Fig. Backend for front-end



Unit 4 APIs

Q. What is API? And responsibilities of API

→ APIs operate through requests and responses with endpoints. When an API requests data from an app or server, an endpoint sends back a response. This helps the API get the resources it needs from a server to perform a task.

APIs exist for different parts of your system, and parts outside of it, to communicate.

APIs can be thought of as the user experience for other applications, in that they must be designed to be usable and to ease integrations.

Create and maintain API client libraries to provide an easy-to-use interface for third-party developers

Track API performance metrics and work to improve the stability, scalability, and availability of the APIs.

If API is handling both server-to-server authentication and this typical user authentication, it can cause complexity, and complexity leads to bugs.

Q. Explain the concept of "Backend for frontend" pattern.

→ Instead, what we could do is introduce layering to APIs. One such set of layers is a rich backend API, and an API that allows front-end UIs to communicate back to a server.This layer is a concept called "backends for frontends." In this, our product API is locked, so it's only available for server-to-server communication (this can be done by requiring an API key or some other secret that isn't exposed to users, or by implementing a firewall at a network level). We then have a "backend-for-frontend" API layer, which is often the same application that serves the original HTML for the page. This API simply exposes a set of AJAX endpoints that the front end can use to query the back end.

Q. What is an API key?

→ An API key is a unique identifier used to authenticate calls to an API. The key is made up of a string of letters and numbers that identify the client. (Remember, this is the application or site making the request.) The key can grant or deny that request based on the client's access permissions. The key also tracks the number of requests made for usage and billing purposes.

Q. Give any 5 API best practices for API security.

→1. Implement authentication and authorization, 2. Use SSL/TLS encryption, 3. Use auditing and logging.

4. Restrict access to sensitive data, 5. Monitor and alert on anomalous activity, 6. Update regularly and patch vulnerabilities quickly, 7. Use API gateways, 8. Secure storage and encryption of data at rest, 9. API Security Testing

Q. What is an event based API?

→ RESTful APIs are useful when user want to discover or change the state of an external system, but sometimes application needs to know when a change has occurred and behave appropriately. Event-driven APIs enable real-time data processing, enabling systems to respond to events instantly.

Q. Explain the benefits of Event based API.

→ **Scalability**: Event-driven APIs allow systems to scale horizontally, as events can be processed by multiple subscribers in parallel. This makes it easier to handle high volumes of requests.

Real-time data: Event-driven APIs enable real-time data processing, enabling systems to respond to events instantly, instead of having to repeatedly poll for updates

Decoupled systems: Event-driven APIs promote decoupled systems, as they allow for loose coupling between systems. This means that systems can evolve independently, and changes in one system don't necessarily impact the other.

Flexibility: Event-driven APIs allow for more flexible and dynamic data flow between systems, as events can be generated and processed by different systems, at different times, in different ways.

Q. What is the purpose of API discovery?

→ API discovery refers to defining innovations and capabilities when it's used in internal programming. It plays a key role in ensuring effective API development done for applications granting access to third-party resources and applications.

Q. Why API discovery is important?

→ API discovery is important because it helps developers to quickly find their APIs, especially those best suited for use in their apps or websites.

It also helps them to mitigate risks by uncovering hidden vulnerabilities, like shadow APIs that are utilizing sensitive data like credit card info, social security numbers, and other personally identifiable information.

API discovery allows developers to find pre-existing and pre-built functionalities and services to easily integrate into their applications. This allows developers to save time and effort and accelerate the development cycle, thereby reducing the product's time to market.

Q. What is cURL?

→ The manual approach is the most common way of discovering APIs. The manual approach involves searching for APIs on the internet and then using tools like cURL to make API calls. This process has been around for a long time and can take a lot of time and effort.

Q. Explain working of API.

→ Third-party payment processing. When a user purchases a product on an ecommerce site, they may be prompted to "Pay with phonepay" or another type of third-party system. This function relies on APIs to make the connection.

• When the buyer clicks the payment button, an API calls to retrieve information—also known as a request. This request is processed from an application to the web server via the API's Uniform Resource Identifier (URI) and includes a request body.

• After receiving a valid request from the product webpage, the API makes a call to the external program or web server, in this case, the third-party payment system.

• The server sends a response to the API with the requested information.

• The API transfers the data to the initial requesting application, here the product website.

Unit 5 Security

Q. What is Phreaking?

→ Phreaking is a form of hacking that targets the telephone system. It is the practice of manipulating or hacking into the telephone system to make free or unauthorized calls. As with validation, sanitization does not only apply when dealing with displaying information to the user. Any data that leaves the system should be sanitized.

Q. Explain any four Golden Rules of API Security?

→ **1. Robust Authorization Mechanism** - The authorization mechanism controls the resources and provides access only to users who have the right to access them. Some of the mitigation techniques for preventing authorization vulnerabilities : define user policies to implement authorization techniques, use random values for GUIDs

2. Never Trust Client Input (Input Validation) - Implementing proper input validation ensures the mitigation of Injection vulnerabilities. Always sanitize the input whenever an application requires input from a user. Also, ensure no input is directly fed into queries like SQL etc.

3. Logging Each Security Event - If logs are analyzed properly in real-time, 99 percent of security breaches may be avoided. Some of the mitigation techniques for preventing big security incidents: It is highly recommended to integrate Security Information and Event Management (SIEM) system to analyze logs produce from the API system. Define rules and configure alerts, so that any anomaly or suspicious activities can be identified
early.

4. Use SSL/TLS - Always use encrypted layer transport while communicating data. Use the latest SSL/TLS protocol to secure in-transit data.

5. Authentication Mechanism - Authentication verifies the credentials of a user and checks whether the user has an authentication right.
Authentication endpoints are the favorite spots of malicious actors as it is easily available and most rewarded if breached.

Q. What is Microsoft STRIDE? & Explain STRIDE model.

→ STRIDE is a model for identifying computer security threats developed by Praerit Garg and Loren Kohnfelder at Microsoft. It provides a mnemonic for security threats in different categories like Spoofing, Tampering, Repudiation, Information Disclosure, Denial of Service (DoS) etc.

1. Spoofing identity - where a user can somehow impersonate another user or log in as another user to take on their characteristics.In other words,Identity spoofing occurs when a scammer assumes the identity of another person/entity and uses that identity to commit fraud.

2. Tampering with data – Where a user can manipulate data that they should not have access to (often caused by validation failures).

3. Repudiation – where an action cannot be reliably traced back to a user. In which an attacker attempts to deny or repudiate actions that they have taken, such as making a transaction or sending a message. These attacks can be a serious problem because they can make it difficult to track down the source of the attack or
determine who is responsible for a particular action.

4. Information disclosure – It means releasing or making the information available to another person or organization. when information that should be private is revealed to someone who should not have access to that data.
5. Denial of service – A Denial-of-Service (DoS) attack is a malicious, targeted attack that floods a network with false requests in order to disrupt business operations.
6. Elevation of privilege – if different users have different layers of privilege (for example, administrator and non-administrator roles),then the application should ensure that a logged-in user cannot get access to any more functionality than they should have access to, by doing authorization checks.

Unit 7 Deployment

Q. Explain the terms: Build,Release and Run

→ **1. Build** : Convert your source code into something that can actually run. This could involve installing any dependencies, or converting any front-end SASS into the final CSS.
2. Release : Which gives us a distinct version number and ID for the build. It is common for configuration to be added here as well, so changing the configuration involves re-releasing the build (but not rebuilding it).
3. Run : Put a release (build + config) in an environment so that it actually starts and does whatever it needs to do.

Q. Explain Continuous delivery and Continuous deployment.

→ **Continuous delivery** is a set of engineering principles that aims to reduce the time an organization takes to make a change to software.
Continuous deployment is a technique where every change is automatically deployed from development into live.

Q. Give the steps for Moving Code into Production.

→ 1. Choosing the right tools
2. Defining a release strategy
3. Integration with CI/CD pipelines
4. Implementing automation for deployment

Q. What is immutable infrastructure?

→ Immutable infrastructure is a model in which no updates, security patches, or configuration changes happen in-place on production systems. If any change is needed, a new version of the architecture is built and deployed into production. Immutable infrastructure refers to servers (or VMs) that are never modified after deployment. With an immutable infrastructure paradigm, servers work differently.
Popular immutable infrastructure tools – Docker, Kubernetes, Spinnaker, AWS, GCP

Unit 6 Python Frameworks

Q. Give advantages of Python frameworks.

→ Easier implementation, Good documentation, Efficient operations, Secure framework, Open-source, Code reusability, Easy integration.

Q. List any three frameworks in python.

→ Django, Flask, Streamlit, Web2Py, CherryPy

Q. What is the meaning of if __ name __ == __ main __ in Python?

→ We use the if-statement to run blocks of code only if our program is the main program executed.

Q. List any 4 four benefits of URL building.

→ 1. It avoids hard coding of the URLs.

2. We can change the URLs dynamically instead of remembering the manually changed hard-coded URLs.

3. URL building handles the escaping of special characters and Unicode data transparently.

4. The generated paths are always absolute, avoiding unexpected behavior of relative paths in browsers.

Q. Give and explain attributes of request object in flask.

→ **1. Form** - It is the dictionary object which contains the key-value pair of form parameters and their values.

2. args - It is parsed from the URL. It is the part of the URL which is specified in the URL after question mark (?).

3. Cookies - It is the dictionary object containing cookie names and the values. It is saved at the client-side to track the user session.

4. Files - It contains the data related to the uploaded file.

5. Method - It is the current request method (get or post).

Q. Explain different HTTP methods in flask.

→ **GET**: to request data from the server.

POST: to submit data to be processed to the server.

PUT: A PUT request is used to modify the data on the server. It replaces the entire content at a particular location with data that is passed in the body payload. If there are no resources that match the request, it will generate one.

PATCH: PATCH is similar to a PUT request, but the only difference is, it modifies a part of the data. It will only replace the content that you want to update.

DELETE: A DELETE request is used to delete the data on the server at a specified location. 20

Q. Describe abort() function with different status code in flask.

→ Flask class has abort() function with an error code. The abort() function is used to handle the cases where the errors are involved in the requests from the client side, such as bad requests, unauthorized access and many more. However, the error code is to be mentioned due to which the error occurred.

The syntax to use the abort() function is given below - Flask.abort(code)

400 : for bad requests, 401: for unauthorized access, 403: for forbidden, 404: for not found, 406: for not acceptable, 415: for unsupported media types, 429: for too many request

Q. Write short note on cookies and session in flask.

→ **Cookies** - The cookies are stored in the form of text files on the client's machine. Cookies are used to track the user's activities on the web and reflect some suggestions according to the user's choices to enhance the
user's experience. In flask, the cookies are associated with the Request object as the dictionary object of all the cookie variables and their values transmitted by the client. Flask facilitates us to specify the expiry time, path, and the domain name of the website. response.setCookie(<title>, <content>, <expiry time>)

Sessions - The session can be defined as the duration for which a user logs into the server and logs out. The data which is used to track this session is stored into the temporary directory on the server. In the flask, a session object is used to track the session data which is a dictionary object that contains a key-value pair of the session variables and their associated values. The following syntax is used to set the session variable to a specific value on the server. session[<variable-name>] = <value>

Q. Give any four standard HTTP codes.

→ 1. HTTP_300_MULTIPLE_CHOICES, 2.HTTP_301_MOVED_PERMANENTLY, 3. HTTP_302_FOUND
4.HTTP_303_SEE_OTHER, 5. HTTP_304_NOT_MODIFIED

Q. Write a program accept the input in URL and concatenate with return statement using flask.

→

```
from flask import Flask, request
app = Flask(__name__)
@app.route('/concatenate', methods=['GET'])
def concatenate():
   # Get the input parameter from the URL
   user_input = request.args.get('input', default='', type=str)
   # Concatenate the input with the return statement
   response = f"Received input: {user_input}. Thank you for using our service!"
   return response
if __name__ == '__main__':
   app.run(debug=True)
```

Q. Write a program using flask to display message "Welcome to School of Computer Science".

→

```
from flask import Flask
app = Flask(__name__)
@app.route('/')
def welcome():
   return "Welcome to School of Computer Science"
if __name__ == '__main__':
   app.run(debug=True)
```

Q. Write a program to implement app routing, page redirection and URL building using flask.

→

```
from flask import Flask, redirect, url_for
app = Flask(__name__)
# Home route
@app.route('/')
def home():
   return "Welcome to the Home Page!"
# About route
@app.route('/about')
def about():
   return "Welcome to the About Page!"
# Redirect route
@app.route('/redirect_to_about')
def redirect_to_about():
   # Redirects to the 'about' route
   return redirect(url_for('about'))
# Profile route with dynamic URL building
@app.route('/profile/<username>')
def profile(username):
   return f"Hello, {username}! Welcome to your profile."
@app.route('/go_to_profile/<username>')
def go_to_profile(username):
   # Builds the URL for the profile route dynamically
   profile_url = url_for('profile', username=username)
   return redirect(profile_url)
if __name__ == '__main__':
   app.run(debug=True)
```

Q. Write a program using flask to display message "Welcome to School of Computer Science".(use template)

→

```
from flask import *
app = Flask(__name__)
@app.route('/')
def customer():
      return
      render_template('customer.html')
@app.route('/success',methods =
['POST', 'GET'])
def print_data():
        if request.method == 'POST':
```

```
        Result = request.form
        return
render_template("result_data.html",result=result)
if __name__=='__main__':
        app.run()
```