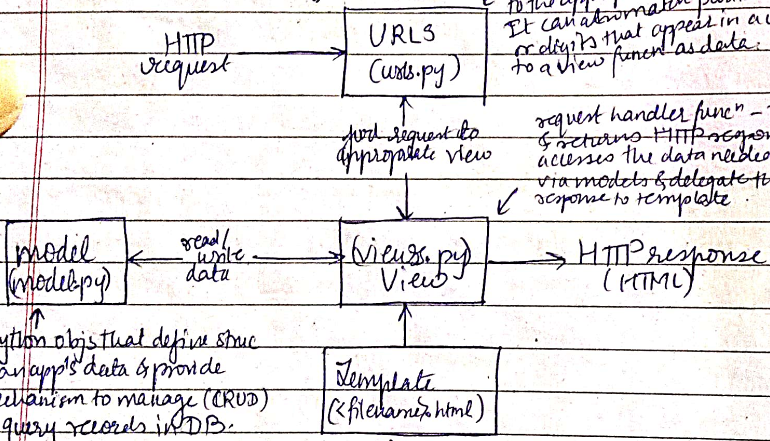
IoT - Unit 5

Q1. Demonstrate Python web applica" framework - Django with suitable example.

Ans Django :
- makes it easier to build better web apps more quickly, less code
- high level python Web framework - clean & realistic designs
- free opensource - can focus on writing app w/o needing code & operationalise framework.
- typically group code that handles each of these steps into separate files.



URL mapper is used to redirect HTTP requests to the appropriate view based on request URL. It can also match particular patterns of strin or digits that appear in a URL & pass them to a view func" as data.

request handler func" - receives HTTP req & returns HTTP responses, accesses the data needed to satisfy requ via models & delegate the formatting of response to template.

python objs that define struc of an app's data & provide mechanism to manage (CRUD) & query records in DB.

text file defining struc/ layout of file with placeholders used to display actual content

* view can dynamically create an HTML page using HTML template populating it with data from model

Q.2 Define S/w Defined Networking & Explain archi of SDN.

defn:

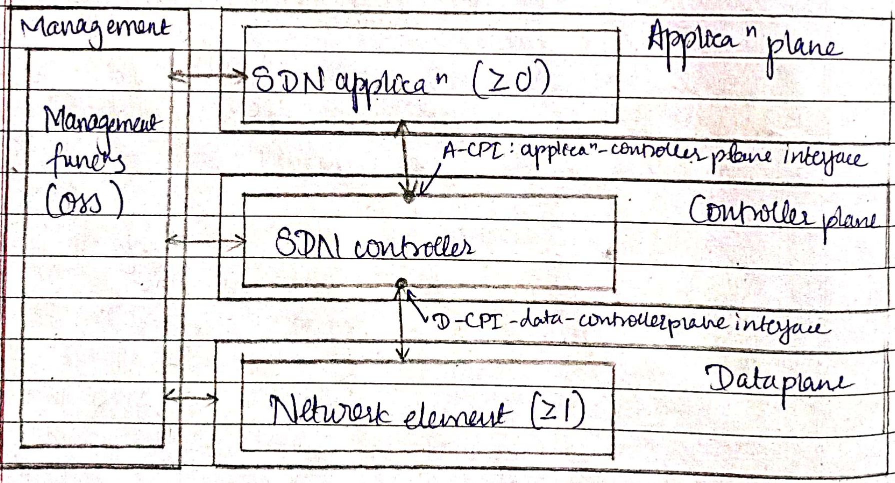
ans 1. The physical separation of network control plane from forwarding plane & where a control plane controls several devices

2. SDN is an approach to networking that uses S/w based controllers or app prog.ing interfaces (APIs) to direct traffics. on network & communicate with underlying H/w infrastruc.

Architecture of SDN:

At a high level there are 3 parts to a typical SDN archi:

1. Apps - communicate resource req. / info about network as a w
2. Controllers - use info from apps to decide how to route a data
3. Networking devices ~~Network~~ receive info from controller abt where to move data.



→ Data Plane: set of ≥1 network eles. each contains set of network forwarding / traffic processing resources. → always abstracts of underlying physical capabilities/entities.

Ex: network switch.

→ Controller plane: set of SDN controllers, each has exclusive control over a set of resources exposed by ≥1 network elements in data plane.

min functionality of SDN controller → execute req. of apps it supports while isolating each app from all others. — for this it may communicate w/ peer SDN controllers, subordinate SDN controllers / non SDN envi's ~~controllers~~

→ Application plane: comprises ≥1 app - each has exclusive control of a set of resources exposed by ~~≥~~ ≥1 SDN controllers. ~~it~~ app may invoke / collaborate with other apps.

added

→ Management: Each app, SDN controller & network element has a functional interface to a manager.

min functionality of manager - allocate resources from resource pool in lower plane to particular client entity in higher plane & to establish reachability info that permits lower & higher plane entities to mutually communicate.

Advantages of SDN:

~~1. Programmatically configured: SDN lets network operat & admins configure, manage, secure, control optimise & operate network resources very quickly via dynamic & automated SDN progs.~~

~~2. Centrally Managed: network intelligence is centralised in S/W based SDN controllers that maintain a global view of network — allows you to control various aspects of your network centrally.~~

1. Programatically configured: Simplifies network managemen through automation using programmable APIs.
2. Centrally managed: provides a single control plane for managing entire network infra struc.
Open stds -Based & vendor neutral: promotes interoperab ~~ct~~ avoiding vendor lock in & supporting diverse H/w.
4. Agile: Enables rapid dev. & scaling of network resources to meet dynamic demands.
5. Robust security: offers centralized security policies & faster threat detecn across the network.
6. Exposure to external apps: facilitates integration w extern tools & apps via APIs for enhanced funcnality.

Challenges:

1. deviation from open stds: Proprietary implementa's can limit interoperability & negate vendor neutrality benefits
2. Single point of failure: a centralized control plane can beco a bottleneck or vulnerability if it fails.
3. legacy solun's: Integerating SDN with existing legacy H/w & S/w can be complex & costly.

Date
Page

Q3. Write a short note on cloud standardization

ans: 1. Cloud standardization refers to development & adoption of common protocols, frameworks & ~~les~~ best practices to ensure interoperability, security & efficiency in cloud computing.

2. It aims to address challenges - vendor lockin, data portability & inconsistent service levels across cloud providers.

Key aspects:

1. Interoperability: Enabling seamless integration & communicaⁿ between diff cloud platforms
2. Security standards: Ensuring data protection & compliance with regulations like ISO/IEC 27017 & GDPR.
3. Data Portability: Facilitating easy migration of data & apps betⁿ cloud providers.
4. API stdizaⁿ: Providing uniform API for developers to enhance cross platform compatibility
5. Performance metrics: Establishing ~~bde~~ benchmarks for service quality & performance.

Q4 Define cloud of things & what is cloud communica^n API

ans: Cloud of things:
refers to integration of IoT with cloud computing to enable efficient data storage, processing & management.
- It allows IoT devices to connect to the cloud, facilitat
real time monitoring, analytics & decision making.

Key features:
1. scalability: handles large volumes of data from IoT devices
2. Real time analytics: Processes & analyzes IoT data for actionable insights
3. Remote accessibility: Enables users to manage IoT devices via the cloud from anywhere.

COT is used in smart homes, healthcare & industrial automation

Cloud communica^n API:
- allow ~~devs~~ devs to integrate communica^n func^nalities (ex: SMS, voice, video, chat) into apps using cloud based services.

aspects:
1- func^s: Enable messaging, VCs, video conferencing & email integeration.
2. Providers: Twilio, nexemo & AWS connect are popular cloud communica^n API providers
3. Advantages: cost effective, scalable, & easy to implement w/o maintaining physical communica^n infrastructure

These APIs are essential for building modern apps requiring seamless communica^n features like customer support & real time notifications.

Q.5. Show that cloud computing is fusion of grid computing & SOA.

ans: Gride computing:

- focuses on pooling resources (ex: computational power, storage) from multiple distributed sys to solve large scale prob's.
  - core idea: resource sharing across networks
  - relation to cloud:
    - cloud uses virtualized resources to deliver scalable computing power.
    - similar to grid, clouds allocate resources dynamically based on demand.

SOA: Service Oriented Approach

- archi. style - S/w components are provided as services, enabling interoperability & reusability
  - core idea: modular & loosely coupled services.
  - relation to cloud:
    - cloud delivers services over the internet ~~(Iaas~~ (Iaas, Paas, Saas)
    - Cloud services follow SOA principles to ensure ease of integration & flexibility.

Fusion in Cloud computing:

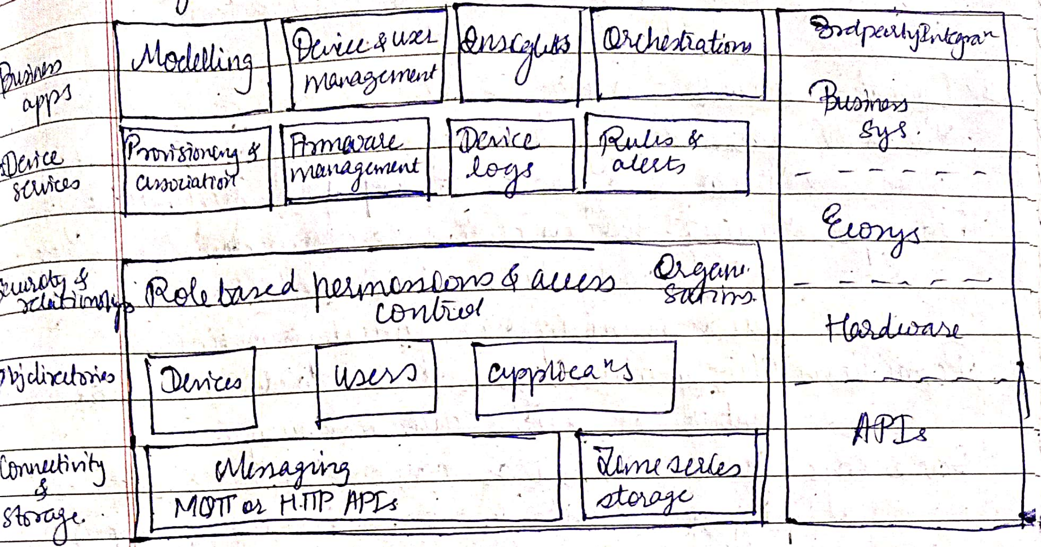
1. Resource management (from Grid)
   - clouds dynamically allocate distributed resources, like grids cloud
   - Ex: largescale data processing in cloud like AWS/Google.
2. Service delivery (from SOA):
   - Clouds provide standardized services via APIs
   - Ex: Saas apps like google workspace
3: Scalability & Efficiency:
   combines scalability of grids with modularity of SOA to provide on demand flexible services.

Q8 Use knowledge of Cloud Computing to demonstrate
i. Amazon Autoscaling
ii. Xively cloud for IoT

ans: Xively cloud for IoT :

1. Xively is an enterprise platform for building, managing & deriving business value from connected products.
2. It helps companies at any stage of IoT journey provide additional value & bring connected prods to market, quickly, reliably & securely.
3. Xively Services:
a. Real time messaging: connects prods & apps while automatically scaling as required.
b. Business logic: you can model your devices, users & grps in xively will connected Product Management
c. Security: Every message is fully encrypted & ~~[illegible]~~ devices are protected from outside attack. Xively uses your ~~user~~ device-user relationship model to authorise data access.
d. Integration: Xively connects to your backend server & also integrates with other business sys like salesforce.

High level architecture :



This architecture provides:

1. Basic infrastructure:

(CPM) Messaging: provides a message broker for secure, scalable & guaranteed message delivery betw devices, users & apps out of the box

Storage: you can store & query historical changes to a device state/commands sent over the wire from time series DB.

2. Devices:

device directory: you can define templates for your devices then store & query against the master record of their state in an indexed & accessible directory

Embedded clients: you can use firmware libs that work with any H/w you choose. You can choose your h/w from a wide range of Xively-ready H/w partners

3. Users:

User directory: you can create store & manage users & store this data in Xively's user management sys.

Template mobileapps : you can use Nirvely's end user
apps for iOS, Android & web.

4. Operational tools - manage & monitor your device fleet.

File & firmware deployments : you can deploy large files
& firmware updates to keep your fleet running sm

Device provisioning : you can choose to allow devices to wake
& securely claim their credentials & identity
autonomously w/o manual provisioning

Permissions : Nirvely allows your users to claim & share
control of devices autonomously.

Device logs : you can monitor connectivity, error states dev
- lifecycle events & diagnostics logs from your
active fleet.

Alerting & monitoring : set automated alerts on criteria that
you define & get real time updates on
health of your fleet by viewing connectivity logs.

5. Management tools

Integration : tie it with CRM sys to better serve your
customers create tickets around your prod.
activity & send data to 3D party sys. you already
use for more use case analysis / processing.

Rules & orchestrations : you can additionally build businesses
workflows that react to product &
user activity & respond to actions in the busine
sys. you have connected via integrations

Insights : gain insight into prod rollouts, geographic
distributions, feature usage patterns & more
that you define, by using our default & custom
dashboard on top of Nirvely data.

AWS Auto Scaling:

1. It monitors your apps & automatically adjusts capacity to maintain steady, predictable performance at the lowest possible cost.

2. Characteristics & features:

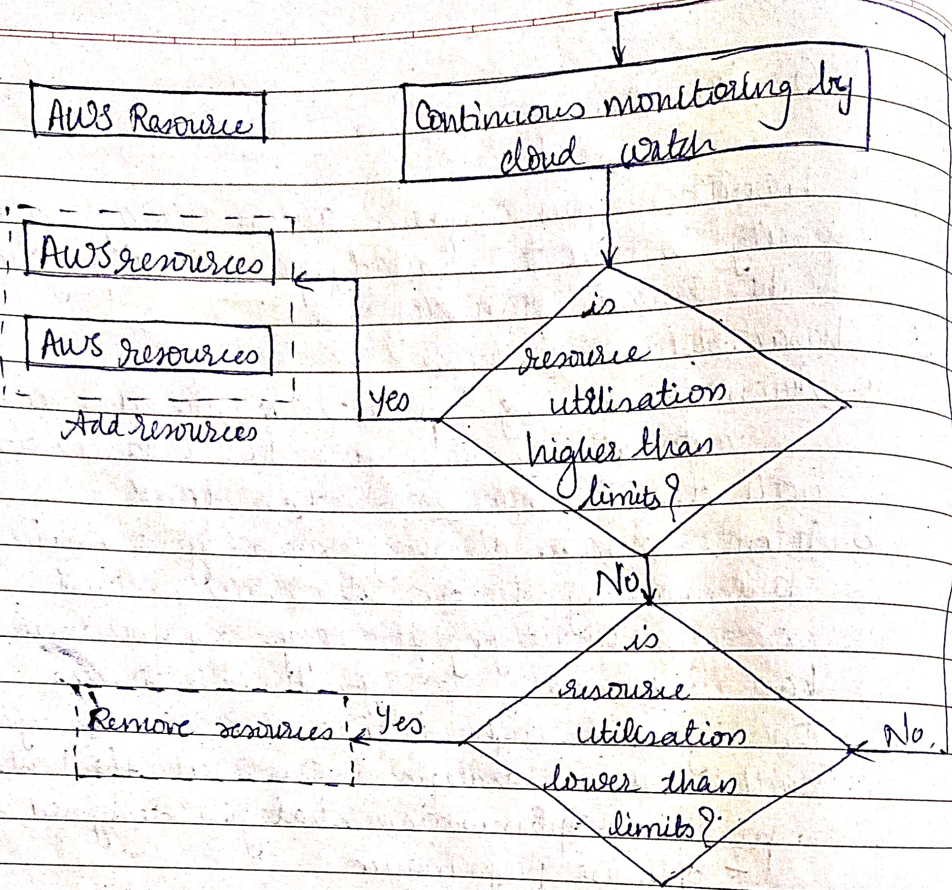
a. Automatic scaling: provides a service that can automatically scale your EC2 instances as well as DB instances based on load.

b. Automatic resource discovery: scans your envi. & automatically discovers the scalable cloud resources underlying your app so you don't have to manually identify these resources one by one through individual service interfaces.

c. Built in scaling strategies: can select ~~1/3~~ 1 out of 3 predefined optimisation strategies designed to:
- optimise performance
- optimise costs
- balance the 2.

can also set your own target ~~o~~ resource utilisation

d. Predictive scaling: predicts future traffic, including regularly occurring spikes & provisions the right no. of EC2 instances in advance predicted changes. It's ML algo detects changes in daily & weekly patterns and automatically adjusts forecasts.


AWS Rasource
Continuous monitoring by cloud watch
AWS resources
AWS resources
Add resources
Yes
is resource utilisation higher than limits?
No
is resource utilisation lower than limits?
Yes
Remove resources
No.

Amazon EC2 : Amazon Elastic Compute Cloud is a web service that provides secure, resizable compute capacity (virtual machines) in the cloud.

features:

1. Iaas (Infrastruc. as a service) : provides scalable virtual servers & computing power on demand in cloud.
2. Several Instance types: offers a wide range of instances types optimized for various use cases.
3. Start & terminate instances as per your req.: ~~offers a~~ enables launching & stopping instances dynamically based on workload needs.
4. Elastic IP address : provides a static IP address that can be remapped to diff instances for high availability
5. Autoscaling : automatically adjusts the no. of instances based on traffic on performance req.
6. Multiple OS to choose from: Allows selecting operating sys like: linux, windows / custom AMIs based on app needs.

IoT messaging mechanisms - WAMP autobahn for IoT

Q#. Show that WAMP & its key concepts are useful in cloud based IoT ~~ad~~ app dev.

ans:- WAMP = Web Applica^n^ Messaging Protocol
is an open std. web socket sub protocol that provides 2 app messaging patterns in one unified protocol.

Combining Publish & Subscribe (PubSub) & routed Remote Procedure Calls (rRPCs) in 1 web native & real time transport protocol allows WAMP to be used for entire messaging req.s of components & micro-service based apps.

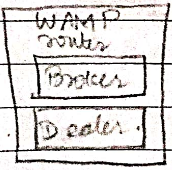
WAMP concepts

Routed RPC (roles)

- Caller — makes procedure calls to dealer through URI of the procedure to be called & any call arg.s.
- Callees — callees register a procedure at a dealer using an URI that identifies the procedure
- Dealer — routes a call originating from the caller to the callee & route back results. Both don't know of each other info is with dealer

PubSub (roles)

- Subscribers — subscribe to one of more topics that could be published by publishers via a broker
- Publishers — push events & info to various topics by sending messages to a broker. Subscribers ~~which~~ whom have subscribed to topics could then receive the messages published by the publishers through brokers
- Broker — acts as a mediator betw^n^ publishers & subscribers so as to decouple the communica^n^ betwn them



Example: Smart Agriculture sys:

sensors monitor soil moisture & publish data to the cloud. Farmer's app subscribe to updates & invoke irritation via cloud based RPC's when levels drop.

∴ Efficient resource utilization & real time updates enhance productivity.

Autobahn for IoT

1. Installing Autobahn|Python:

```
pip install autobahn
```

2. Creating WAMP applica^n components

```
from autobahn.twisted.component import component
comp = Component(...)
@comp.on_join
def joined(session, details):
    print("session ready")
```

3. Running WAMP applica^n components:

```
from autobahn.twisted.component import Component
from autobahn.twisted.component import run
comp = Component(
        transports = u"ws://localhost:8080/ws",
        realm = u"realm1",
    )

@comp.on_join
def joined(session, details):
    print("session ready")
if __name__ = "__main__":
    run([comp])
```

4. Registering Procedures:

```
from autobahn.twisted.component import
 Component, run
component= Component (...)
@component.on_join
@indline Callbacks
def joined(session, details):
    print("session ready")
    def add2(x,y):
        return x+y
    try:
        yield session.register(add2, u'com.myapp.add2')
        print ("procedure registered")
    except Exception as e:
        print("Couldn't register procedure: {0}".format(e))
```

5. Calling Procedures:

```
from Autobahn.twisted.component import
  Component, run.
from twisted.internet.defer import indlineCallbacks.
Component = Component (...)
@component.on_join
@ inline Callbacks
def joined(session, details):
    print ("session ready")
    try:
        res= yield session.call(u'com.myapp.add2', 2,3)
        print("Call result: {}".format(res))
    except Exception as e:
        print(" Call error: {0}".format(e))
```

6. Subscribing to topics

```
from autobahn.twisted.component import Component
from twisted.internet.defer import inlineCallbacks
component = Component(...)
@ component.on_join
@ inlineCallbacks
def joined(session, details):
    print("session ready")
    def oncounter(count):
        print("Even received: {0}", count)
    try:
        yield.session.subscribe(oncounter, u'com.~~ex~~myapp.
                oncounter')
        print("subscribed to topic")
    except Exception as e:
        print("could not subscribe to topic:{0}".format(e))
```

7. Publishing events:

```
from autobahn.twisted.component import Component
from autobahn.twisted.util import sleep
~~from Autobahn~~
from twisted.internet.defer import inlineCallbacks
components = Component(...)
@ component.on_join
@ inlineCallbacks
def joined(session, details):
    print ("session ready")
    counter = 0
    while True:
        #publish() only ~~t~~returns a deferred if we asked
            for an acknowledgement
        session.publish(u'com.myapp.oncounter', counter)
        counter +=1
        yield sleep(1).
```

Examine how cloud computing is an IoT enabling tech with suitable example.

Q.8. Apply concept of cloud computing to design an IoT system

ans: Cloud computing is an IoT enabling tech :: it provides scalable infrastruc, storage & processing power req. for IoT sys to func^n effectively.

1. Data storage: stores vast amts of sensor data generated by IoT devices.
2. Processing Power: analyzes real time data using cloud based tools like AWS IoT core, Azure IoT Hub
3. Scalability: supports dynamic scaling to handle increasing no.s of IoT devices.
4. Interconnectivity: connects globally distributed IoT for seamless data sharing & control
5. cost efficiency: eliminates the need for heavy on-premises infrastruc investments

Ex: Smart home automation using Cloud computing

1. IOT devices: smart bulbs, thermostats, sensors collect data
2. Cloud Platform: processes data, stores it, & manages devices communica^n via MQTT ~~& recieve alerts via cloud~~
3. ~~B~~ Mobile apps: users control devices & receive alerts via the cloud.
4. Outcome: Efficient, scalable & cost effective remote monitoring & control for a smart home.

Q.9. Design a cloud storage model for IoT based healthcare app, consider storage requirements, data security & privacy concerns associated with sensitive patient health records. Discuss pros & cons of using public, private & hybrid cloud storage option.

ans: Cloud storage model for IoT based healthcare app:

1. Storage req:
   structured data: Patient record, metadata
   Unstructured data: dignostic images, IoT sensor data
   scalability: Elastic storage for growing data
   Redundancy: Backup & disaster recovery mechanisms.

2. Data security & Privacy:
   Encryption: data encryption at rest & in transit
   Access control: Multifactor authentication & role based access.
   Compliance: Adherence to HIPAA, GDPR / local health care regulations
   Data anonymization: for non critical analyties to ensure privacy.

3. Cloud storage model:
   Hybrid Cloud:
   - private: securely store sensitive heath records
   - public: host non-sensitive anonymized data for analytics
   Edge computing: process real time IoT data locally to reduce latency.

Benefits of model:

- Balances Security & scalability
- cost effective analytics with public cloud.
- Complies with Regulatory requirements through private cloud use.

	Public cloud	Private cloud	Hybrid Cloud.
	shared infrastruc. provided by 3rd Paty (ex: AWS, Azure)	Dedicated infrastruc managed by the org.	Combines public & private clouds.
Pros:	1. cost effective (pay as you go) 2. scalable 3. Managed by provider 4. Rapid deployment	1. High security & control 2. Meets strict compliance 3. Customizable	1. Balances cost & contro 2. Flexibility 3. Enhanced ~~distance~~ disaste recovery
Cons:	1. Data privacy 2. concerns 2. limited control 3. Potential down time	1. Expensive to set up & maintain 2. limited scalability 3. Requires inhouse expertise	1. Complex integration 2. Costly initial setup 3. maintenance req.
Best use cases:	1. Noncritical data 2. Small to medium sized healthcare apps.	1. Large hospitals with strict compliance requirements	2. Sensitive data storage on private, analytics on public

Q.10. Design a home automation sys using Autobahn for IoT & Xively for IoT communican API. Discuss how these APIs can be used to enable device control, data collecn & remote monitoring of various home appliances & sensors.

ans: Home automation sys using Autobahn for IoT & Xively for IoT communican API:
supports functionalities like:
- remote control of appliances
- real time monitoring of sensors.
- Automation based on sensor inputs.

Hardware:
1. Sensors: temp., humidity, motion, light & gas
2. Actuators : smart switches, dimmers & thermostats
3. Gateway: Raspberry pi or any IoT enabled microcontroller
4. Communican: Wifi / MQTT based devices.

S/w:
Autobahn: Enables realtime bidirectional communican betwn IoT devices & backend using Websocket & WAMP protocols.
Xively: 1. device management: manages connected appliances & tracks device states.
2. Data collecn: gathers sensor data for analytics
3. remote monitoring: offers API for accessing device data & sending commands.

Work flows & features / use of these apis for following:

	Autobahn	Xively
1. Device control	real-time control commands are sent to devices over Web socket	routes commands from user's app to appropriate device.
2. Data collec^{n}: sensors continuously collect data	receives, stores & orgs. sensor data in cloud	provides live streaming of data to mobile app / dashboards
3. Remote monitoring	Enables realtime update on appliance status ?	Allows users to view historical data & trends via mobile app / web interfaces
4. Automation: rules are defined for automation	Ensures quick execu^{n} of automation logic	logs events & outcomes for analysis.