Design and develop a custom application using Salesforce Cloud

---

🔷 Section 1: Core Salesforce Concepts

✅ 1. What is Salesforce?

Salesforce is a **cloud-based Customer Relationship Management (CRM) platform**. It helps organisations manage all kinds of relationships — not just with customers, but with students (in education), patients (in healthcare), or clients (in services).

But Salesforce is **more than a CRM** — it's also a **platform** where you can:

- Store and organise data (like student records)
- Automate processes (like sending notifications or assigning courses)
- Build **custom applications** using low-code tools or code

Why is Salesforce powerful?

- It runs entirely in the cloud — no need to install software
- It provides built-in security, scalability, and analytics
- It’s widely used in real-world companies — making it industry-relevant

In your project, Salesforce lets you build a Student Management System *without building everything from scratch*.

✅ 2. What is Salesforce Lightning?

Salesforce Lightning is the **modern user interface (UI)** and **app development framework** in Salesforce. It's the evolution of the older "Classic" interface.

It has two key meanings:

1. **Lightning Experience** (UX/UI):
    - Sleek, modern design
    - Drag-and-drop customization

    - Responsive for mobile and desktop

2. **Lightning Platform** (Development Framework):
    - Tools like **Lightning App Builder**, **Lightning Components**, and **Flows**
    - Supports low-code and custom-code (like LWC - Lightning Web Components)

Why important in your app?
You'll use Lightning to:

- Design custom record pages (e.g., Student record view)
- Create your app's homepage or dashboards
- Build flows to automate student-related actions

✅ 3. What is an App in Salesforce?

An **App** in Salesforce is a **collection of tabs, objects, reports, and dashboards** grouped to serve a specific business function or process.

There are 2 main types:

- **Standard Apps**: Pre-built by Salesforce (like Sales, Service)
- **Custom Apps**: Built by users/admins for a specific purpose — like your **Student Management System**

Think of an App like a folder — it holds everything your users need to manage something (in this case, students).
Inside your app:

- Tabs like "Students", "Courses", "Enrollments"
- Dashboards for visual data
- Reports like "Students by Department"

Apps make it easier for users to **focus on a single task flow**.

✅ 4. What is an Object in Salesforce?

An **Object** in Salesforce is like a **database table**. It stores structured data.

There are two types:

- **Standard Objects** (e.g., Account, Contact)
- **Custom Objects** (created by you — like `Student`, `Course`, `Enrollment`)

Each object contains:

- **Fields** (like columns — Name, Age, Course)
- **Records** (like rows — each student's data)

✅ 5. What is a Record in Salesforce?

A **Record** is a **single entry in an object** — similar to a row in a database.

For example:

In the `Student` object, a record could be:

Name: Ananya Sharma
Roll No: 2024ENG102
Course: B.Tech CSE

Every time a user fills out and saves a form in Salesforce, they create a new **record**.

✅ 6. What is a Field in Salesforce?

A **Field** is a **single piece of information** stored in an object — like a database column.

Types of fields include:

- **Text** (e.g., Student Name)
- **Number** (e.g., Roll Number)
- **Date** (e.g., Date of Admission)
- **Picklist** (e.g., Gender = Male/Female/Other)

In your SMS app, each object like `Student` or `Course` will have many fields to capture specific data.

✅ 7. What is the difference between Standard and Custom Objects?

Feature	Standard Object	Custom Object
Definition	Predefined by Salesforce	Created by user/admin
Examples	Account, Contact, Opportunity	Student, Course, Enrollment
Flexibility	Limited customization	Fully customizable
API name ends with	Just the name (e.g., Contact)	`__c` (e.g., `Student__c`)

You use **custom objects** when Salesforce's built-in ones don't meet your needs.

✅ 8. What is a Relationship in Salesforce?

A **relationship** connects one object to another, like a link between tables.

Types:

1. **Lookup Relationship** (like a loose link)
    - A Student "looks up" a Course
    - If the Course is deleted, Student still exists
2. **Master-Detail Relationship** (tight link)
    - If a Student has related Enrollments (child records)
    - Deleting the Student also deletes the child records

In your app:

- Use **Master-Detail** between `Student` and `Enrollment`
- Use **Lookup** between `Enrollment` and `Course`

✅ 9. What is a Tab in Salesforce?

A **Tab** is a **UI element** in the navigation bar. It lets users quickly access data from an object.

Tabs allow you to **navigate to object data** in the user interface.

Example:

- "Students" Tab → takes you to all student records
- "Courses" Tab → all course records

Tabs are part of your App's layout, making it easy for users to switch between different types of data.

✅ 10. What is the Object Manager?

Object Manager is found in **Salesforce Setup** and is used to:

- Create/edit objects and fields
- Set up relationships
- Configure page layouts, validation rules, and automation

Whenever you create or customize an object (like `Student__c`), you do it from the Object Manager.

🔷 Section 2: UI Design & App Building in Salesforce

✅ 11. What is the Lightning App Builder in Salesforce?

Lightning App Builder is a **drag-and-drop interface tool** in Salesforce used to build:

- **Custom pages** for desktop and mobile
- **Home pages**, **record pages**, and **app pages**

You can add and arrange **components** like:

- Record details
- Charts
- Lists

- Custom Lightning components

Why it's useful:
You don't need to code — just **drag components onto a canvas**, define the layout, and assign the page to users, profiles, or apps.

In your Student Management System:
Use App Builder to:

- Create a custom **Student Record Page** with summary info, courses, and related lists
- Build a **Home Page** for teachers showing reports like "Top 10 Students"

✅ 12. What is a Lightning App in Salesforce?

A **Lightning App** is a complete packaged workspace. It includes:

- Objects (like Students, Courses)
- Tabs
- Utility items (like recent items)
- Custom pages

Users can switch between apps using the App Launcher.

In your project:
You'll build a custom Lightning App called **"Student Management"** that contains:

- Tabs for each custom object (Student, Course, Enrollment)
- Dashboards for visual summaries
- Flows to automate tasks

✅ 14. What are Page Layouts in Salesforce?

A **Page Layout** controls:

- **Which fields** appear when viewing or editing a record

- **What related lists** are shown
- **Field order and grouping**

Page Layouts let you tailor the interface based on user needs.

Example:

- A Student record can be set to show basic info at the top (Name, Age)
- Below that, related lists like Courses, Attendance, Grades

You can assign different page layouts to **different profiles** using **Record Types** (explained next).

✅ 15. What is a Record Type in Salesforce?

Record Types allow you to:

- Show **different layouts and picklist values**
- Use **different business processes**

Example in your system:
You can have two record types for Students:

- **Undergraduate**
- **Postgraduate**

Each can have:

- Different fields (e.g., UG may not have Thesis Title, PG might)
- Different page layouts
- Different picklist options (e.g., "Year" options)

Record Types provide flexibility when a single object serves **different categories** of users or processes.

✅ 17. What are List Views in Salesforce?

List Views let users filter and display records based on specific criteria.

Example:

- "All Active Students"
- "Courses with fewer than 10 students"
- "Students enrolled in 2024"

You can create custom list views and even allow **inline editing** (editing fields directly from the list).

This is useful for **quick updates or analysis** without going into each record.

✅ 19. What are Apps, Objects, Fields, and Records — and how are they related?

This is often a combined conceptual question.

You can explain it like this:

Element	What It Is	Analogy
App	A container for related features and objects	A folder or workspace
Object	Like a table to store data	A spreadsheet
Field	A column in the object	Field like "Student Name" or "Course"
Record	A row of data in the object	One student's full data

They are all **connected hierarchically**. You need all four to build your app.

🔷 Section 3: Automation & Logic in Salesforce

✅ 20. What is a Validation Rule in Salesforce?

A **Validation Rule** is a formula that checks whether data entered in a record **meets certain conditions**.
If the rule evaluates to **true**, it **shows an error message** and prevents the record from being saved.

Why it's useful:
It helps maintain **clean and consistent data** in the system.

Example in your SMS app:

If a user tries to save a Student record without entering a roll number:

ISBLANK(Roll_Number__c)

- → Error Message: “Roll Number cannot be blank.”

You can also use rules like:

- Prevent students under age 16 from being admitted
- Ensure students select a course while enrolling

✅ 21. What is a Workflow Rule?

A **Workflow Rule** is an older Salesforce automation tool that performs **simple actions automatically** when a record meets certain conditions.

Actions include:

- Sending an email
- Updating a field
- Creating a task

Note: Salesforce now recommends using **Flow** instead, as it is more powerful and flexible.

Example in your SMS app:

- When a student is marked as "Graduated", send an email notification to the department head.

✅ 22. What is Salesforce Flow?

Salesforce Flow is a **powerful low-code automation tool** that allows you to:

- Automate complex logic without writing code
- Guide users through screens (like a wizard)
- Run background processes (like auto-enroll a student)

There are different types of Flows:

- **Screen Flow**: For guided user input (e.g., new student intake form)
- **Record-Triggered Flow**: Runs automatically when a record is created/updated (e.g., auto-assign a student to a default course)
- **Scheduled Flow**: Runs at specific times

Why it's powerful:
Flows can replace both **Workflow Rules** and **Process Builder**, and it supports **loops, conditions, assignments**, and even calling Apex code.

✅ 29. How do you deploy a Salesforce application?

There are multiple ways to **move your app** from a Sandbox to Production:

1. **Change Sets** (point-and-click tool for admins)
2. **Salesforce CLI (SFDX)** – for developers using code
3. **Ant Migration Tool**
4. **Workbench**

Deployment steps usually include:

- Create components (objects, fields, flows)
- Group them into a change set
- Upload from Sandbox → Deploy in Production
- Run test classes (mandatory if you use Apex)

✅ 30. What is a Profile in Salesforce?

A **Profile** controls:

- What objects a user can access
- What fields they can see/edit
- Which tabs are visible
- Which apps they can open

Every user has **exactly one Profile**.

Example:

- "Admin" profile → full access
- "Teacher" profile → read/write students, but no access to setup
- "Student" profile → can only view their records

Profiles enforce **role-based access** to keep data secure.

✅ 31. What is a Permission Set in Salesforce?

A **Permission Set** gives **extra permissions** on top of the Profile.

Used when:

- You want to **temporarily or additionally** grant permissions
- You don't want to clone/edit existing Profiles

Example:

- Profile: "Student" (read-only)
- Permission Set: "Allow Course Registration" → grants them edit access to Enrollment

You can assign **multiple permission sets** to the same user — but only **one Profile**.

✅ 32. What is Role Hierarchy in Salesforce?

Role Hierarchy defines **who can see whose data**.

It is:

- **Top-down**: Higher roles can see lower role data
- Not about features — it's about **record visibility**

Example:

- Principal (top role) can see all student data
- Department Head can see students under their team
- Teacher can see only their students
- Teacher cannot see that field

This helps enforce **data visibility based on organization structure**.

✅ 35. Explain SaaS, PaaS, and IaaS again (with better analogy)

Layer	Full Form	What it offers	Example	Analogy
SaaS	Software as a Service	Ready-to-use software	Salesforce CRM, Gmail	Renting a fully furnished flat
PaaS	Platform as a Service	A platform to build and run apps	Salesforce Platform, Heroku	Renting an empty flat to decorate
IaaS	Infrastructure as a Service	Raw computing resources	AWS EC2, Azure VMs	Renting land to build your house

- **Salesforce CRM = SaaS** → You use it out of the box
- **Salesforce Platform = PaaS** → You build apps (like SMS) on top of it

Salesforce does **not** provide IaaS — you don't deal with servers or OS.

✅ 36. Why is Salesforce considered both SaaS and PaaS?

- **SaaS**: It offers CRM applications — you can use them directly (like Sales Cloud, Service Cloud).
- **PaaS**: It offers a platform to **develop custom apps** (like your Student Management System) using tools like:
    - Apex (code)
    - Flow (no code)
    - LWC (web components)
    - App Builder (drag & drop UI)

So, Salesforce provides **ready-made functionality + customizable tools** = SaaS + PaaS.

Find a procedure to transfer two text files from one virtual machine to another virtual machine. Also, add some text in the file and display it before and after transfer on both the VMs.

✅ 1. What is a Virtual Machine (VM)?

A **Virtual Machine** is a **software-based simulation** of a physical computer. It runs an operating system and apps just like a real computer but is hosted by another machine (host system)

✅ Key Components:

- **Host OS**: The real OS (like Windows, Linux) on your computer
- **Guest OS**: The OS inside the VM (e.g., Ubuntu running in VirtualBox)
- **Hypervisor**: Software that creates and runs VMs (e.g., VirtualBox, VMware, KVM)

✅ How it works:

The hypervisor allocates:

- **RAM**

- **CPU cores**
- **Disk space**
  to the virtual machine, which then functions like a real PC.

Why we use VMs:

- Testing and development
- Running multiple OS on one system
- Isolated environments

In your task, both VM1 and VM2 are independent machines that communicate over a network.

✅ 2. What is SCP and how does it work?

SCP (Secure Copy Protocol) is a command-line utility that allows you to **copy files between hosts** over a **secure SSH connection**.

- Uses port **22** (like SSH)
- Encrypts data during transfer
- Authenticates users via passwords or SSH keys

```
scp source_file user@remote_ip:/path/to/destination
```

You can also copy files from a remote machine:

```
scp user@remote_ip:/path/to/file ./local_folder/
```

In your case, SCP sends files from VM1 to VM2 over the network securely.

Why use SCP?

- Secure alternative to `ftp`
- Easy for quick file transfers
- No need to install extra services (uses existing SSH)

✅ 3. What is SSH and how is it different from SCP?

- **SSH (Secure Shell) is a network protocol that** allows you to **remotely connect to and control another computer**.
- **SCP** is for file transfer, but it uses SSH underneath for secure authentication.

It allows you to:

- Access the remote machine's **command line**
- **Execute commands** remotely
- **Transfer files securely** (via SCP or SFTP)
- **Encrypt all communication**, making it safe from hackers

Works in **client-server** model

- Your local machine is the **client**
- The machine you connect to is the **server**
- Uses **encryption and authentication**:
    - Either with a **password**
    - Or using **SSH keys** (public/private pair)

SSH Example:

```
ssh user@192.168.1.20
```

Now you can run commands on the remote VM like it's your own terminal.

✅ 4. What command is used to display file contents in Linux?

Use:

- `cat filename.txt` – Displays the entire content
- `less filename.txt` – Scrollable view
- `tail filename.txt` – Shows last 10 lines

- `head filename.txt` – Shows first 10 lines

In your exam, use `cat` before and after transfer to verify file content.

✅ 5. How do you edit files in Linux?

There are multiple ways:

- `echo "text" >> file.txt` – Appends text
- `nano file.txt` – Opens a user-friendly terminal editor
- `vi file.txt` or `vim` – Powerful terminal editor (requires practice)

For simplicity, use `echo` for adding and `cat` for reading.

✅ 7. What is the difference between `>` and `>>` in file writing?

`>` – Overwrites the file

bash
CopyEdit
```
echo "Hello" > file.txt  # Replaces content
```

-

`>>` – Appends to the file

bash
CopyEdit
```
echo "Hello again" >> file.txt  # Adds new line
```

-

Use `>>` in your practical to **append** content instead of replacing.

✅ 8. What is the role of IP addresses in file transfer?

An **IP address** uniquely identifies a machine on a network.

In SCP or SSH:

- You use `user@IP_address` to connect to a specific VM
- Example: `scp file.txt user@192.168.1.20:/home/user/`

Without the IP, your computer wouldn't know **where to send the file**.

✅ 9. Can we use other tools to transfer files between VMs?

Yes, alternatives include:

- **rsync** – Faster, syncs only changes
- **ftp/sftp** – File transfer protocol over network
- **netcat (nc)** – Sends raw data over a socket
- **shared folders** (in VirtualBox, VMware)

However, **SCP is simplest** and secure, so preferred in most academic/practical use cases.

✅ 10. What is hostname vs IP address?

- **Hostname** is a human-readable name (e.g., `vm1.local`)
- **IP address** is a numeric label (e.g., `192.168.1.10`)

You can map hostnames to IPs using `/etc/hosts` or a DNS server.

🔷 Section 3: Deeper Networking & Linux Concepts

✅ 11. What is a network in the context of virtual machines?

A **network** allows virtual machines to **communicate with each other**.

Types of VM networking (especially in VirtualBox/VMware):

- **NAT**: VM shares host's internet, but can't be accessed directly
- **Bridged**: VM acts like a separate machine on the LAN
- **Host-only**: VM can talk to host, but not the internet
- **Internal**: VMs talk to each other but not to the host or internet

For file transfer, Bridged or Host-only is often used so VM1 and VM2 can **see each other's IP addresses**.

✅ 12. What does the `ping` command do?

`ping` checks whether another machine is **reachable over the network** and measures how long it takes.

bash
CopyEdit
```
ping 192.168.1.20
```

If it replies, it means:

- The VM is online
- There's **network connectivity**
- You can likely use SSH/SCP

If it fails, you may need to:

- Check firewall rules
- Check VM network configuration

✅ 13. What is a firewall and how can it block file transfer?

A **firewall** is a system that controls **incoming and outgoing network traffic** based on rules.

It can block:

- **SSH (port 22)** → prevents SCP
- **ICMP (ping)**
- Other protocols

In Linux, firewall tools include:

- `iptables`
- `ufw` (Uncomplicated Firewall)

To allow SCP/SSH, port 22 must be **open** between the two VMs.

✅ 14. What is port 22? Why is it important for SCP/SSH?

Port **22** is the default port for **SSH (Secure Shell)**.

- All **SCP**, **SFTP**, and **remote shell login** happen over port 22
- If this port is closed or blocked by a firewall, SCP will fail

You can verify open ports with:

```
sudo netstat -tuln
```

You can **specify a different port** with `-P`:

bash
CopyEdit

```
scp -P 2222 file.txt user@ip:/path
```

✅ 15. What is an SSH key? How is it different from a password?

SSH key authentication uses a **pair of cryptographic keys** (public + private) instead of a password.

Steps:

- Generate a key: `ssh-keygen`
- Copy public key to remote VM: `ssh-copy-id user@ip`
- Now you can SSH or SCP without entering a password

Benefits:

- More secure
- Automated scripts can run without manual login

In a secure environment, SSH key-based login is preferred over password authentication.

✅ 17. What is localhost? What is 127.0.0.1?

- **localhost** is the **loopback address** — it refers to the same machine.
- IP **127.0.0.1** is its numeric equivalent.

You use it when:

- Testing network apps on the same VM
- Hosting a local server
- Pinging your own system

You **can't use 127.0.0.1 to access another VM**, because it's self-referencing.

SCP needs absolute paths if you're copying to/from remote systems.

✅ 19. What happens if you transfer a file that already exists on the destination?

- By default, **SCP overwrites the file** without asking.

If you want to **avoid overwriting**, use checks like:

```
scp -n file.txt user@ip:/path/
```

✅ 20. How do you check your VM's IP address?

In Linux:

```
ip a     # or
ifconfig
```

Look for an address like `192.168.x.x` or `10.0.x.x` (private IP range).
Ensure both VMs are in the **same subnet** to allow communication.

NAT (Network Address Translation) is a **networking mode** that allows your **virtual machine (VM)** to **access the internet** using the **host machine's IP address**.

Your VM shares the host's network connection, but:

- It **cannot receive incoming connections** from other machines
- It is **invisible** to other devices on the local network

NAT Network in VMs

A **NAT Network** is a **virtual private network created by your virtualization software (like VirtualBox)** that:

1. Connects multiple VMs together — they can **talk to each other** as if they're on a private LAN.
2. Allows **all those VMs to access the internet** using your **host's network connection**.
3. Uses **NAT (Network Address Translation)** to map internal (private) VM IPs to the host's public IP.
4. **Is isolated from the physical network** — machines outside your computer can't see or reach the VMs.

Goal	Why we need it
Let VMs communicate with each other	So you can transfer files between them (e.g., using SSH/SCP)
Give internet access to both VMs	So they can download software, updates, packages
Keep them isolated from outside	For **security** and to prevent external access
Avoid complex configuration	NAT Network is easy to set up and needs no IP configuration

✅ How it works (Conceptually):

- The **NAT Network acts like a mini virtual router**
- Both VM1 and VM2 get private IPs like `10.0.2.4`, `10.0.2.5`
- They can **ping each other, SSH into each other, SCP files**
- When they access the internet, NAT converts their IPs to the host machine's public IP

So you get **local communication + internet access**, with **no need to manually configure routes or DHCP**.

Design and deploy a web application in a PaaS environment.

✅ 1. What is a PaaS environment?

PaaS (Platform as a Service) is a cloud computing model where you can **build, test, and deploy applications** without managing the underlying infrastructure.

In a PaaS environment:

- You write code
- The platform (like Vercel) handles:
    - Hosting
    - Server setup
    - Scaling
    - Load balancing
    - Security patches

Example PaaS platforms: Vercel, Heroku, Netlify, Google App Engine

Why use it?
You can focus entirely on your **app logic** — not server configurations.

✅ 2. How is PaaS different from SaaS and IaaS?

Model	What You Do	What Provider Manages	Example
IaaS	Build from OS up	Hardware, virtualization	AWS EC2
PaaS	Write and deploy code	OS, runtime, scaling	Vercel, Heroku
SaaS	Just use the software	Everything	Gmail, Google Docs

Analogy:

- IaaS: Renting land to build a house
- PaaS: Renting a fully equipped kitchen to cook
- SaaS: Getting home delivery

✅ 3. What are the advantages of PaaS?

- **Faster deployment**
- **No server management**
- **Automatic scaling**
- **Built-in tools for CI/CD**
- **Easier collaboration (with Git integrations)**

Especially helpful for **frontend developers** or **small teams**.

✅ 4. Why is Vercel considered a PaaS?

Because Vercel:

- Let you deploy web apps without setting up servers
- Handles build and deployment automatically
- Offers CI/CD, SSL, CDN, and domain management
- Supports full-stack apps using serverless functions

So you just **push your code**, and it:

- Builds your app
- Hosts it
- Gives you a public link instantly

You don't deal with:

- Virtual machines
- Ports
- Security patches

🔷 SECTION 2: Web Application & Deployment Concepts

✅ 5. What is a web application?

A **web application** is a software application that:

- Runs in a browser
- Is accessed via the internet
- Is built using:
    - **HTML** (structure)

    - **CSS** (design)
    - **JavaScript** (interactivity)

Modern web apps use frameworks like:

- **React**, **Next.js**, **Angular**, or **Vue**

Web apps are dynamic, unlike static websites. They respond to user input, make API calls, and often include authentication.

✅ 7. How did you deploy your app on Vercel?

1. Pushed my project to **GitHub**
2. Logged into Vercel.com
3. Clicked **"New Project"**
4. Imported my GitHub repository
5. Vercel detected the framework and built the app
6. It gave me a live URL like `https://my-app.vercel.app`

Vercel handles:

- The **build process**
- **Hosting**
- **Scaling**
- And even **automatic re-deployments** on new commits

✅ 8. What is the build process in Vercel?

The **build process** is where Vercel:

- Runs your project's build command (like `npm run build`)

- Optimises your code
- Bundles static assets (HTML, CSS, JS)
- Converts server code (like Next.js API routes) into **serverless functions**

This output is then deployed and made accessible globally.

🔷 SECTION 3: Git & CI/CD Concepts

✅ 9. What is Continuous Deployment (CI/CD)?

CI/CD means:

- **Continuous Integration**: Every code push is tested and merged automatically
- **Continuous Deployment**: Each code change is automatically built and deployed

In Vercel:

- Every push to the main branch → new build + live deployment
- Each pull request → preview URL

It reduces manual work and ensures **faster updates** and **less human error**.

🔷 SECTION 4: Hosting, Domains, Serverless, Environment

✅ 11. What is a serverless function?

In platforms like Vercel:

- You don't manage your own servers
- You write **serverless functions** (API routes) in files
- Vercel runs them on-demand

```
// pages/api/hello.js

export default function handler(req, res) {

  res.status(200).json({ message: "Hello from serverless!" });

}
```

Vercel deploys this as an endpoint:

```
https://yourapp.vercel.app/api/hello
```

Great for small APIs, forms, and backend logic.

✅ 12. What are environment variables? Why do we use them?

Environment variables store **sensitive or configurable data** like:

- API keys
- Database URLs
- Access tokens

You don't hard-code them — instead, you use:

```
process.env.API_KEY
```

In Vercel:

- You add them via the dashboard under your project → "Environment Variables"
- They are injected during **build time**

✅ 13. What is a CDN, and how does Vercel use it?

CDN (Content Delivery Network) = A globally distributed system of servers that **store and serve your app's files** from the nearest location to the user.

Benefits:

- Faster loading
- Reduced latency
- Better scalability

Vercel uses a **global CDN by default**. Your app is served from multiple edge locations.

✅ 14. What is DNS? How do you point a domain to Vercel?

DNS (Domain Name System) translates human-readable domains (like `myapp.com`) to IP addresses.

In Vercel:

- Go to “Domains”
- Add a custom domain (like from GoDaddy or Namecheap)
- Vercel gives you **DNS records** (like CNAME or A)
- Add those records to your domain registrar

Then your app becomes accessible at `https://yourdomain.com`.