01 Answers:

a) Requirements engineering (RE) is about finding out *what the system should do* before building it. But the process differs a little between **conventional software** and **web applications** because of their nature.

Here's a comparison in table form:

Aspect	**Conventional Software**	**Web Application**
Users	Limited, known group (e.g., company staff)	Large, diverse, often global and unknown users
Requirements Elicitation	Interviews, workshops with stakeholders	Surveys, usage analytics, online feedback, usability studies
Functionality	Task-focused, domain-specific	Mix of functionality + content delivery (interactive + multimedia)
Non-functional Requirements	Performance, reliability, maintainability	Scalability, cross-platform compatibility, security, response time
Change Frequency	Updates less frequent, planned releases	Frequent updates, continuous deployment, agile changes
Interface	Rich GUIs but platform-specific	Browser-based, responsive, device-independent
Security Concerns	Limited to organization's network	High: open access, authentication, privacy, transaction safety
Testing Focus	Correctness, performance	Correctness + usability, accessibility, compatibility (browsers/devices)

b)

Why we need Web Engineering:
Because web apps face unique challenges like **security, scalability, usability, and rapid evolution**. Web engineering ensures they are built systematically and can handle these demands.

Example:
In **online banking**, without web engineering the system could crash on peak days or be hacked. With web engineering, it stays **secure, scalable, and user-friendly**.

c)

1. What is Requirement Analysis?
It's the process of **studying and understanding what a system should do** before building it. In this step, we carefully analyze user needs, document them, and remove ambiguities or conflicts.

2. Why do we need it?

- To avoid misunderstandings between users and developers.
- To detect problems early (cheaper to fix now than later).
- To ensure the final system actually solves the user's problem.
- To provide a clear roadmap for design, coding, and testing.

<u>02 Answers:</u>

a)

Functional requirements → what the system *must do* (features, tasks). For your case:

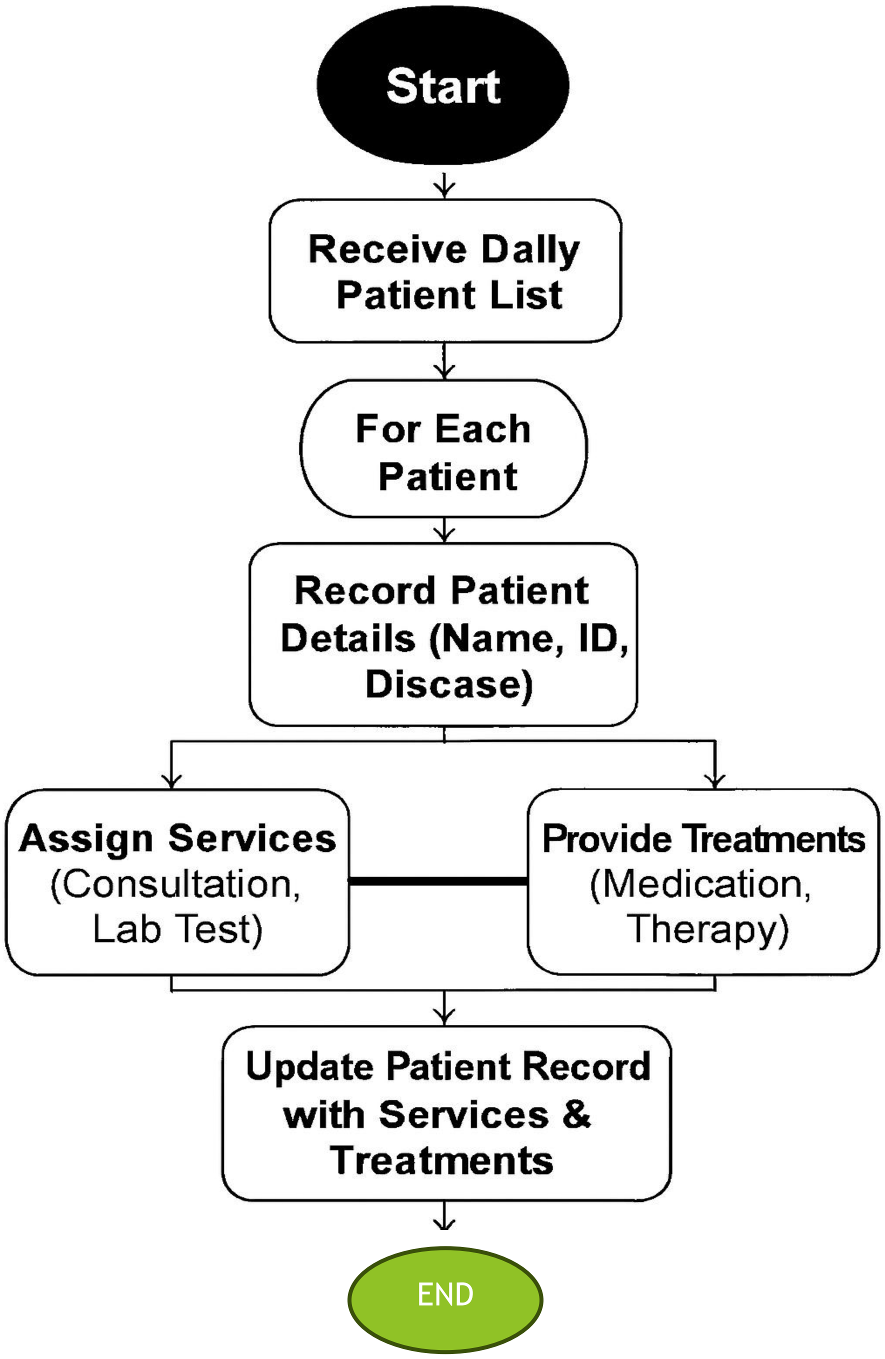
- Store patient details (name, age, contact, disease).
- Record daily services provided (e.g., check-ups, tests).
- Record treatments/medications for each patient.
- Generate daily/weekly reports of patients and treatments.
- Allow authorized staff to add/update/delete patient records.

Quality (non-functional) requirements → how the system should *perform*. For your case:

- **Reliability:** Data should not be lost or corrupted.
- **Security:** Only authorized staff can access patient info.
- **Usability:** Easy-to-use interface for hospital staff.
- **Performance:** Quick retrieval of patient records.

- **Scalability:** Should handle growing number of patients.

b)


Start
Receive Dally Patient List
For Each Patient
Record Patient Details (Name, ID, Discase)
Assign Services (Consultation, Lab Test)
Provide Treatments (Medication, Therapy)
Update Patient Record with Services & Treatments
END

c) **Web engineering borrows from software engineering but adds extra rules because the web is dynamic, network-based, and user-facing. The main principles are:**

1. **Clearly defined goals and requirements**
   Phase: **Requirement Analysis**
   → Identify what the web application must achieve, both functionally and non-functionally.
2. **Systematic development in phases**
   Phase: **Design**
   → Break the project into logical stages, ensuring each is well-structured before moving forward.
3. **Careful planning of these phases**
   Phase: **Implementation**
   → Organize resources, timelines, and tasks to execute the design effectively.
4. **Continuous audit of the entire development process**
   Phase: **Testing**
   → Regularly check for errors, performance issues, and compliance with requirements.