Carnegie Mellon

School of Computer Science

15-415/615 Spring 2013
Homework 7

Building A Web Application

03/27/2012

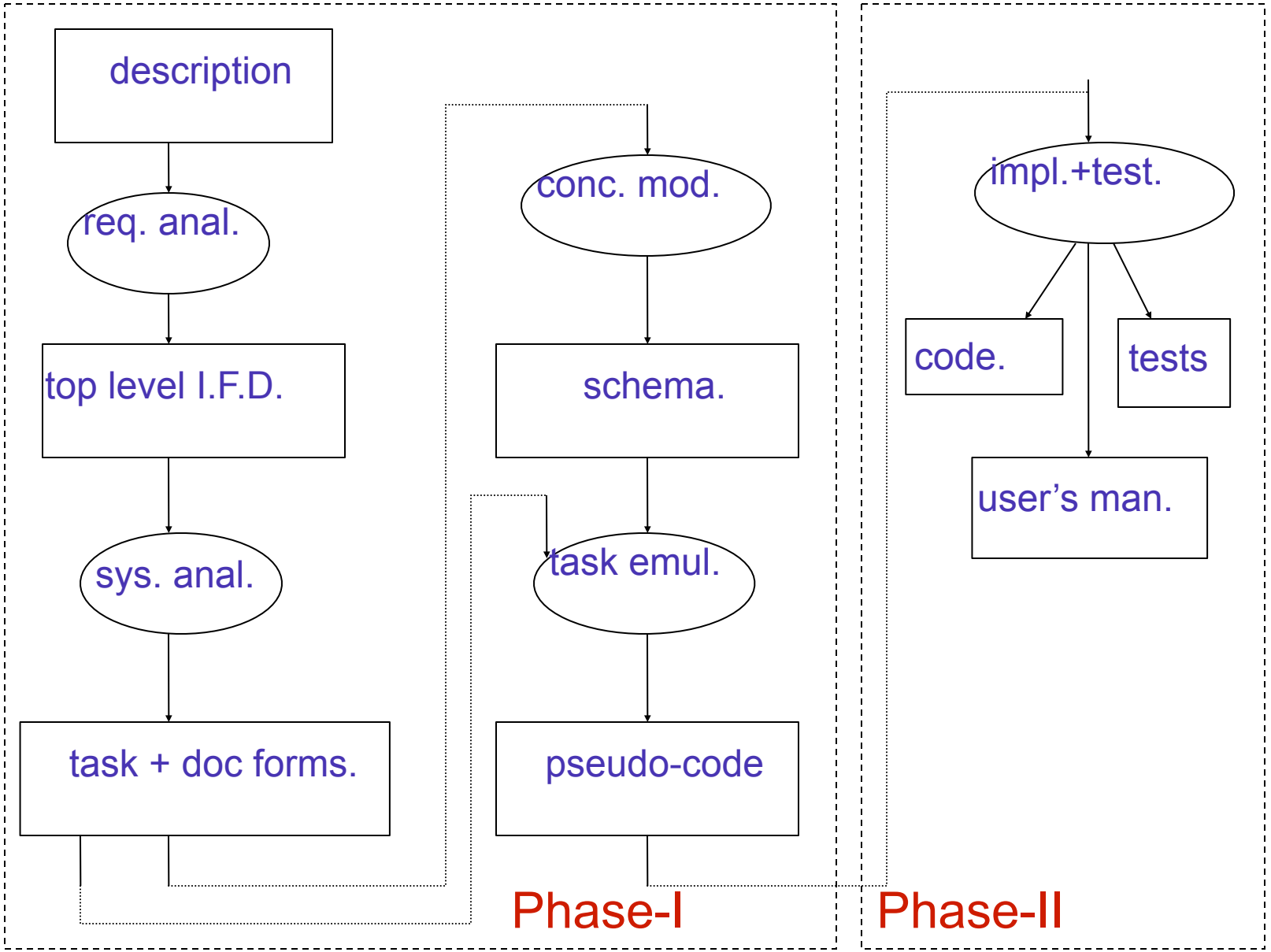
HW7 Outline

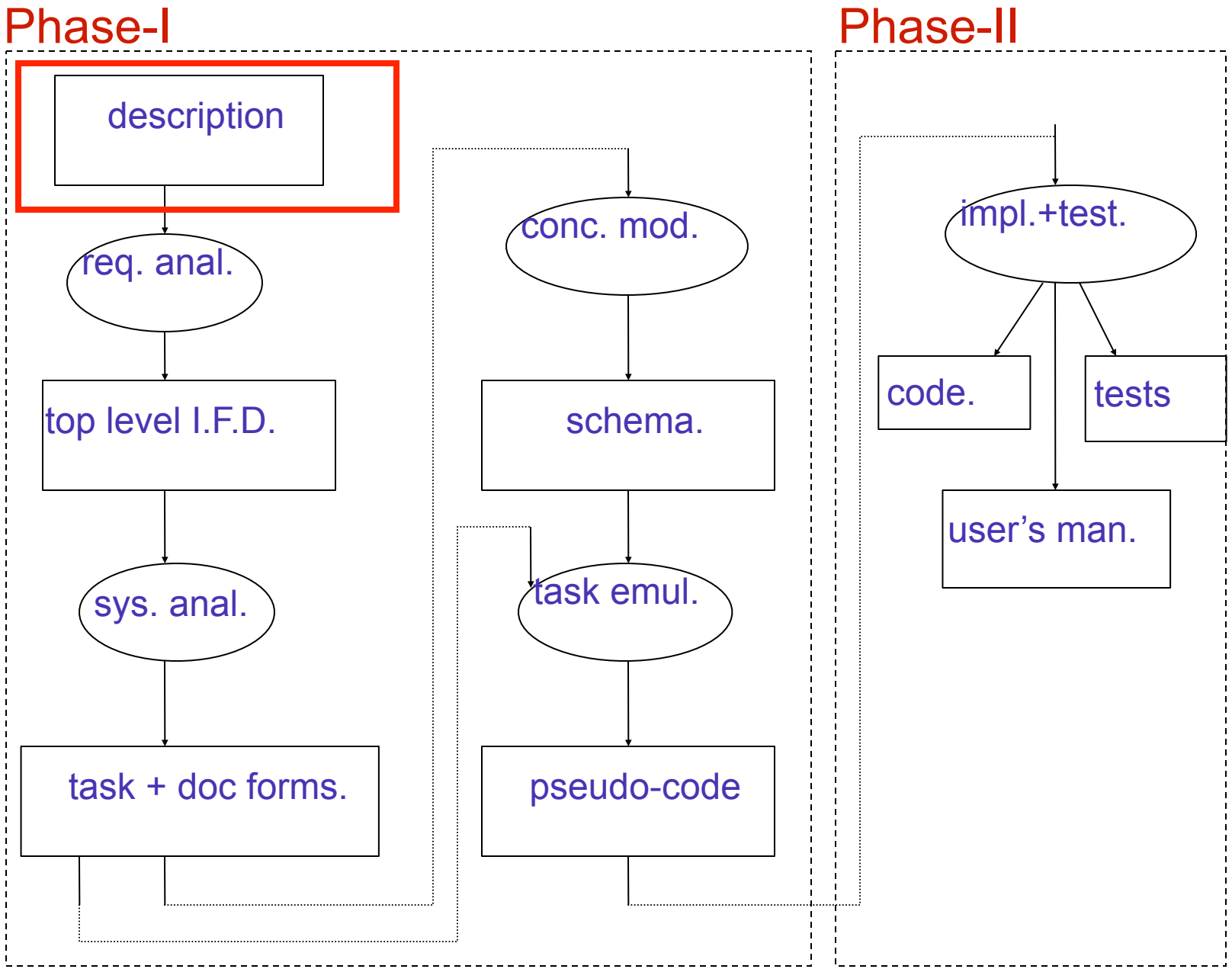
- Building a simple Web application ("CMUFlix") using JSP
- 2 phases
- Phase I : design and documentation
  - due 4/2
  - hard copy in class
- Phase II : implementation
  - due 4/11
  - both hard copy in class and electronically.

  - **Start early!**

Database Design Methodology

[N. Roussopoulos and R.T. Yeh]



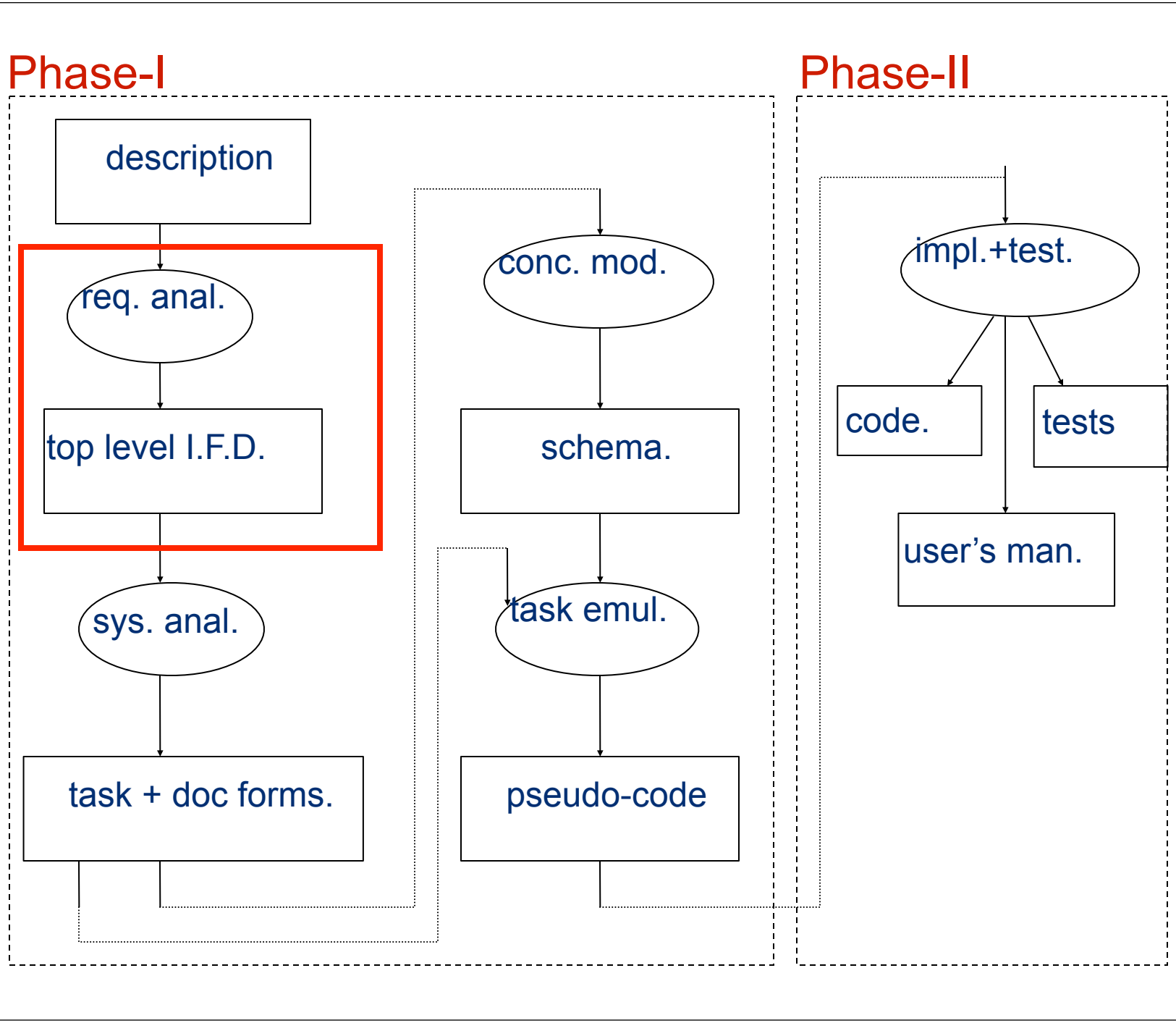

Phase-I
Phase-II
description
req. anal.
top level I.F.D.
sys. anal.
task + doc forms.
conc. mod.
schema.
task emul.
pseudo-code
impl.+test.
code.
tests
user's man.

Homework 7

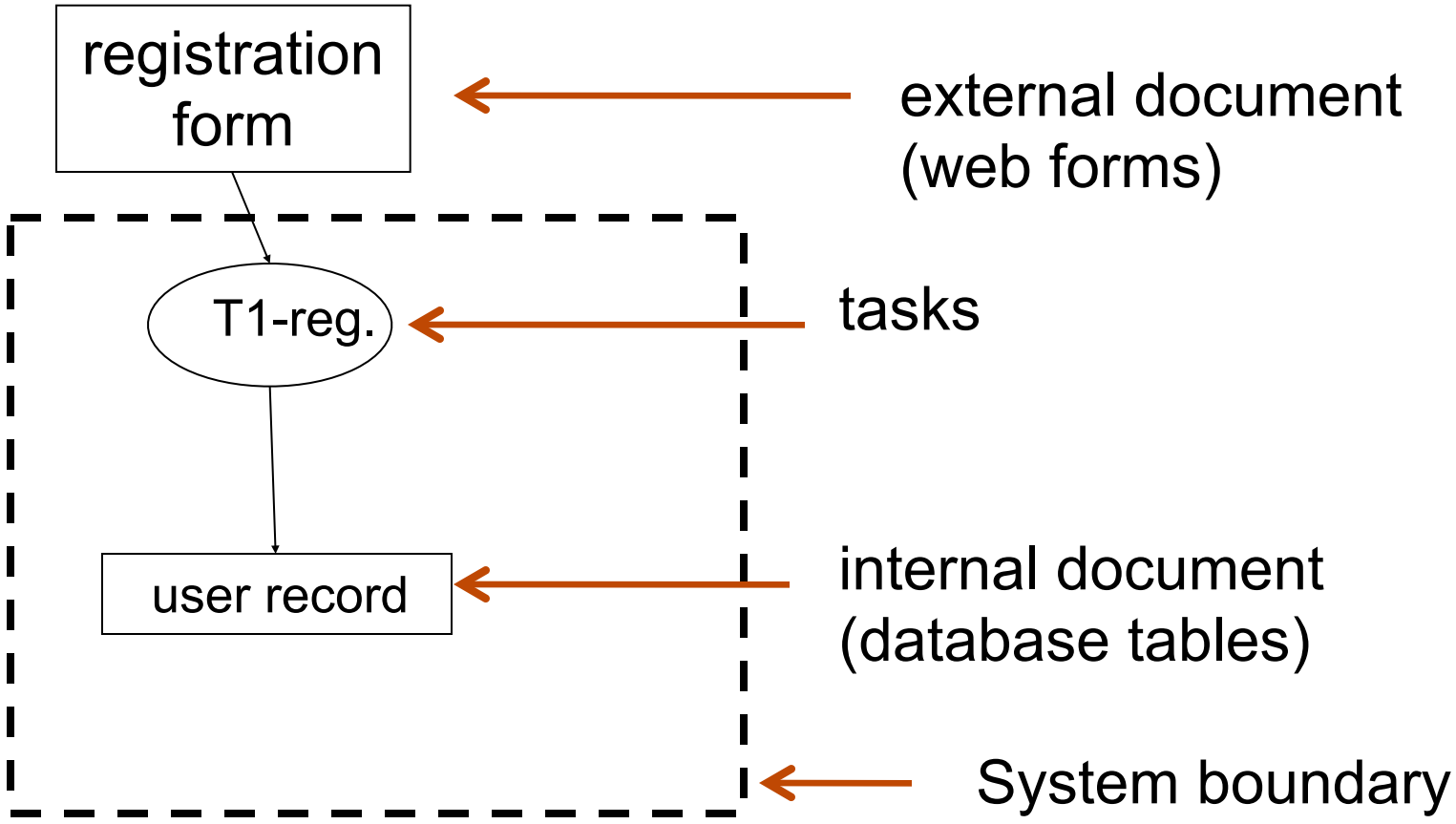
- A recommendation web application for movies, like NetFlix.
- Users can register, like a movie, get recommendations, etc.

Tasks to implement

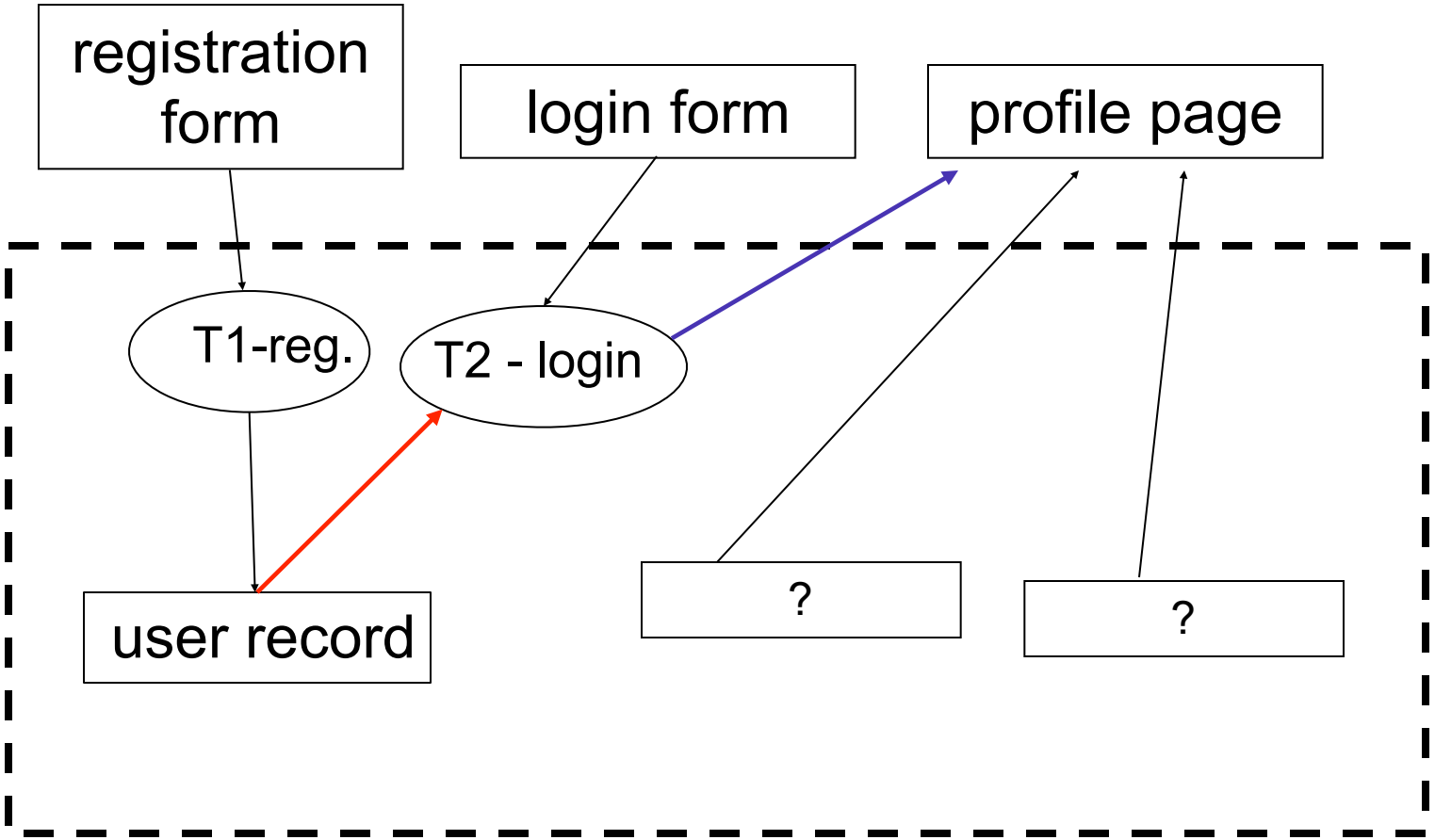
- Registration (a simple example is already provided)
- Login/Logout
- Profile Page
- “Like” a movie
- Ask for a movie recommendation
- Reporting (website statistics)

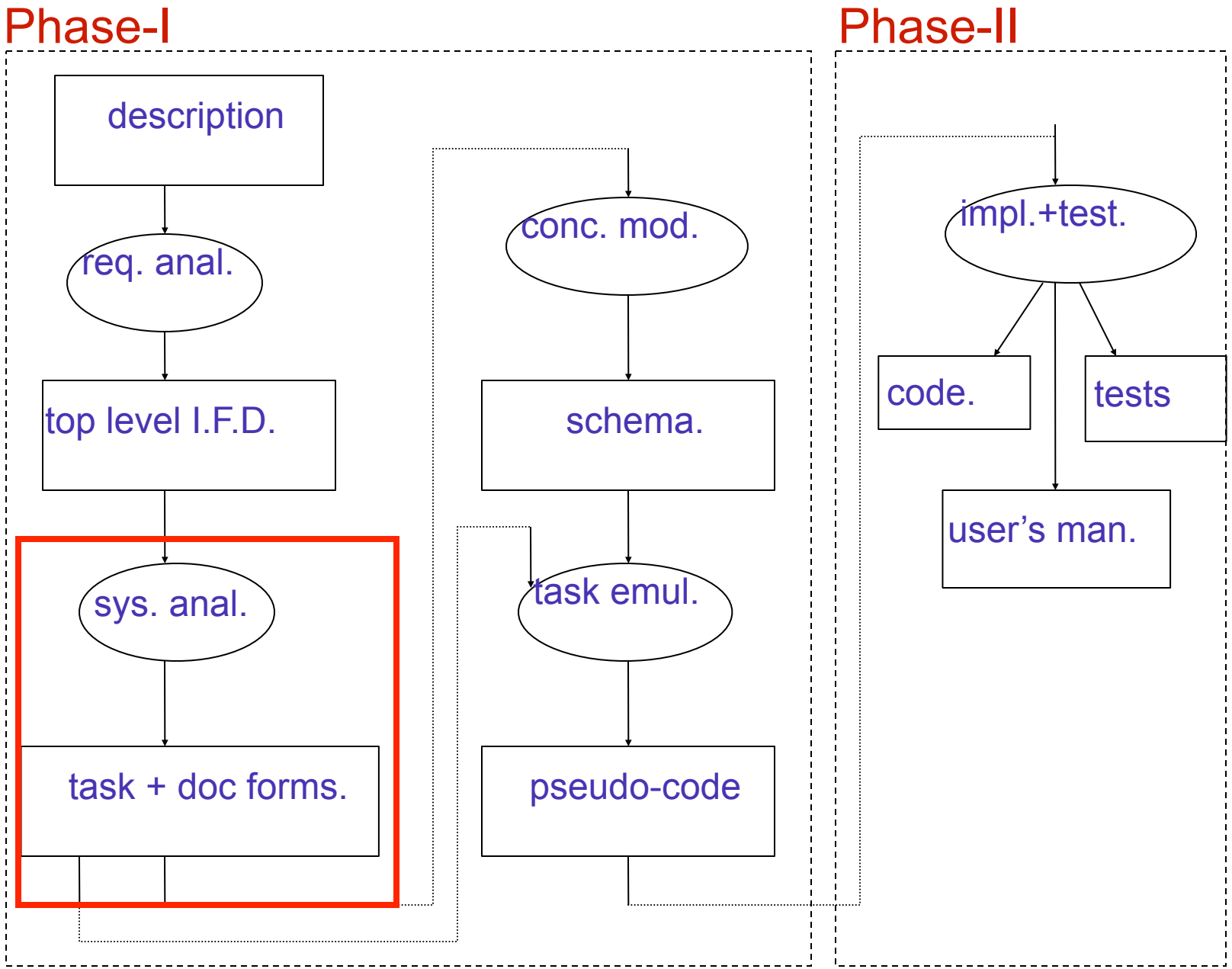

Phase-I
Phase-II
description
req. anal.
top level I.F.D.
sys. anal.
task + doc forms.
conc. mod.
schema.
task emul.
pseudo-code
impl.+test.
code.
tests
user's man.

Top level information flow diagram (Homework I)



More on top level diagram




Phase-I
Phase-II
description
req. anal.
top level I.F.D.
sys. anal.
task + doc forms.
conc. mod.
schema.
task emul.
pseudo-code
impl.+test.
code.
tests
user's man.

Document + Task forms

- Task forms and task list
  - not required for this homework

- Document forms and document list

  - D1: registration form
  - D2: login form
  - D3: profile form
  - …
  - Dx: user record
  - …

external

internal

Document forms (Homework I)

D1: registration form

- login name
- password
- email

D3: profile form

- login name?
- …

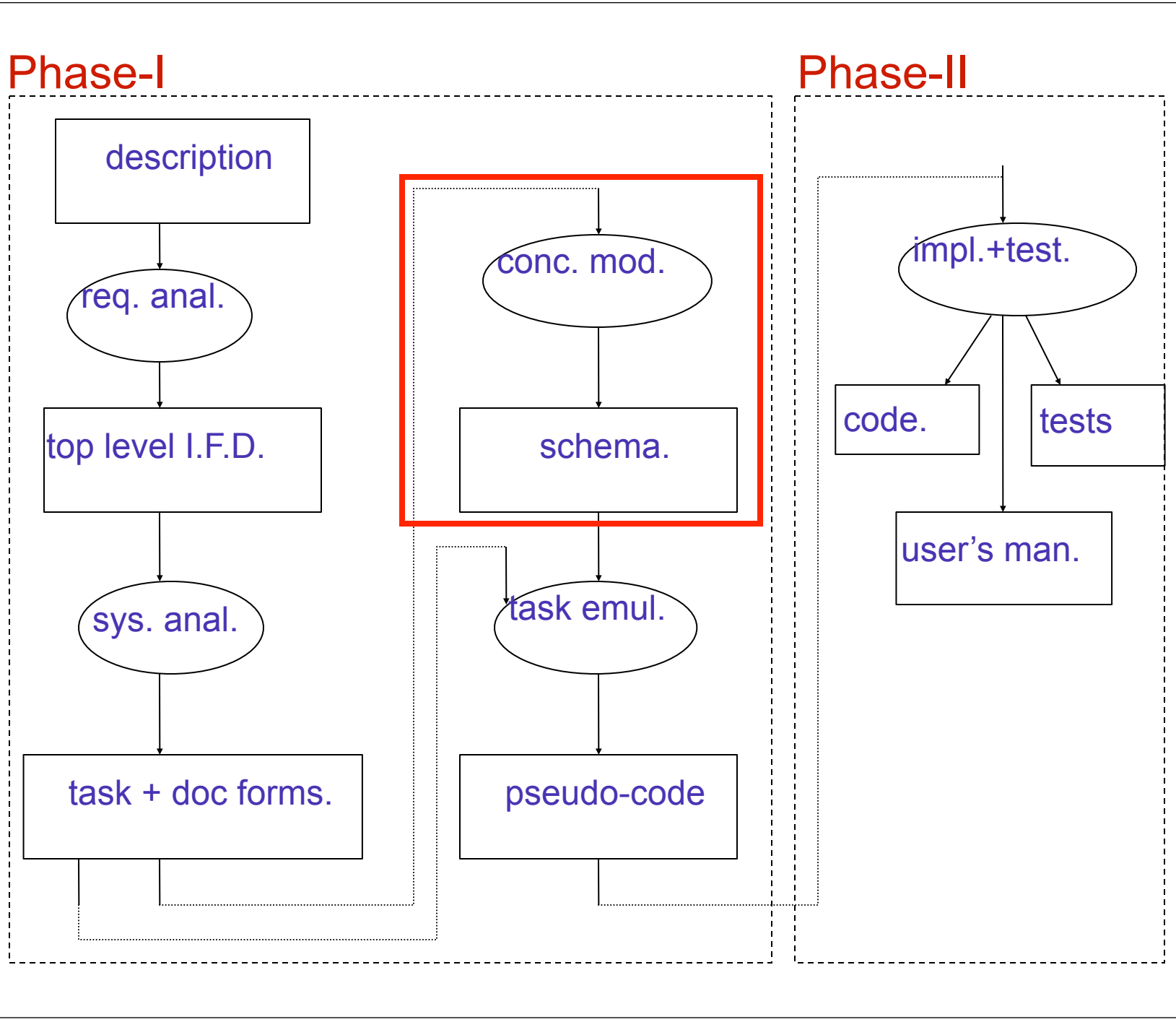
List-of:

Movies liked

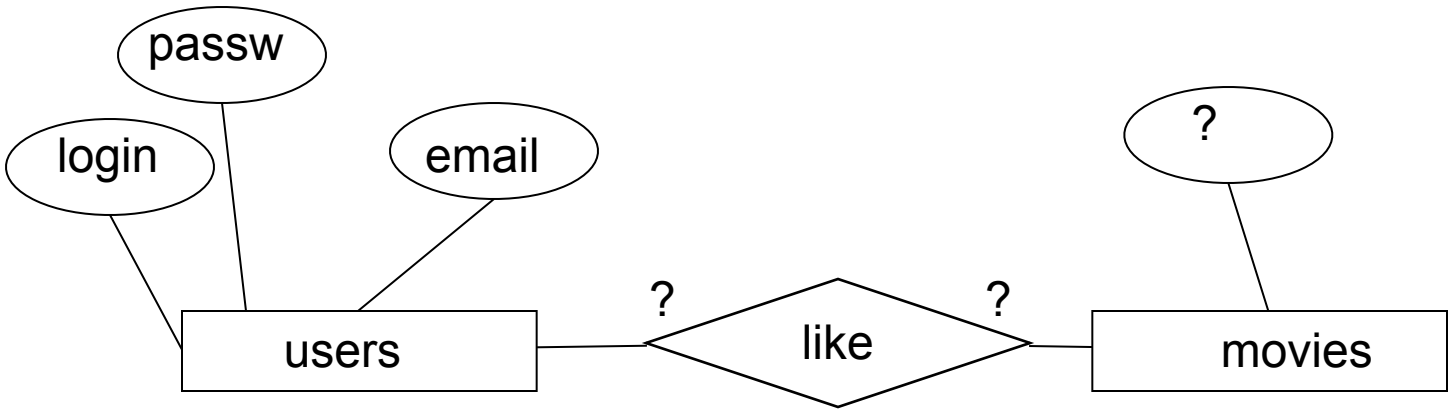
…

Dx: user record

- login name
- password
- email


Phase-I
Phase-II
description
req. anal.
top level I.F.D.
sys. anal.
task + doc forms.
conc. mod.
schema.
task emul.
pseudo-code
impl.+test.
code.
tests
user's man.

E-R diagram (Homework I)



...

- Specify cardinalities
- Think about weak/strong entities

Relational schema (Homework I)

users(login, passw, email …)

movies(…) ?

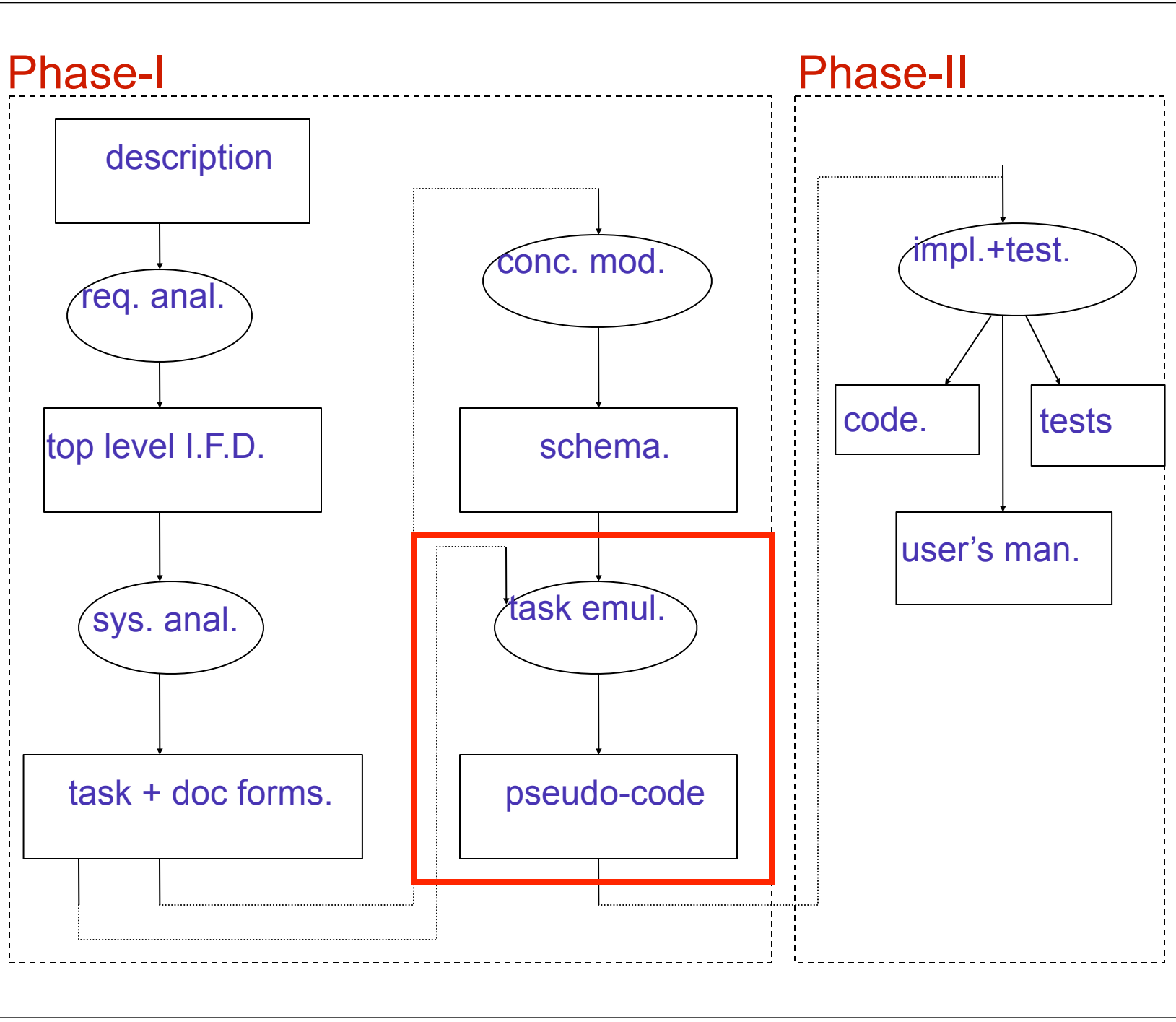
….

SQL DDL statements (Homework I)

create table users (login char(20), passwd char (20) NOT NULL (?), …);

create table ? (…);

…

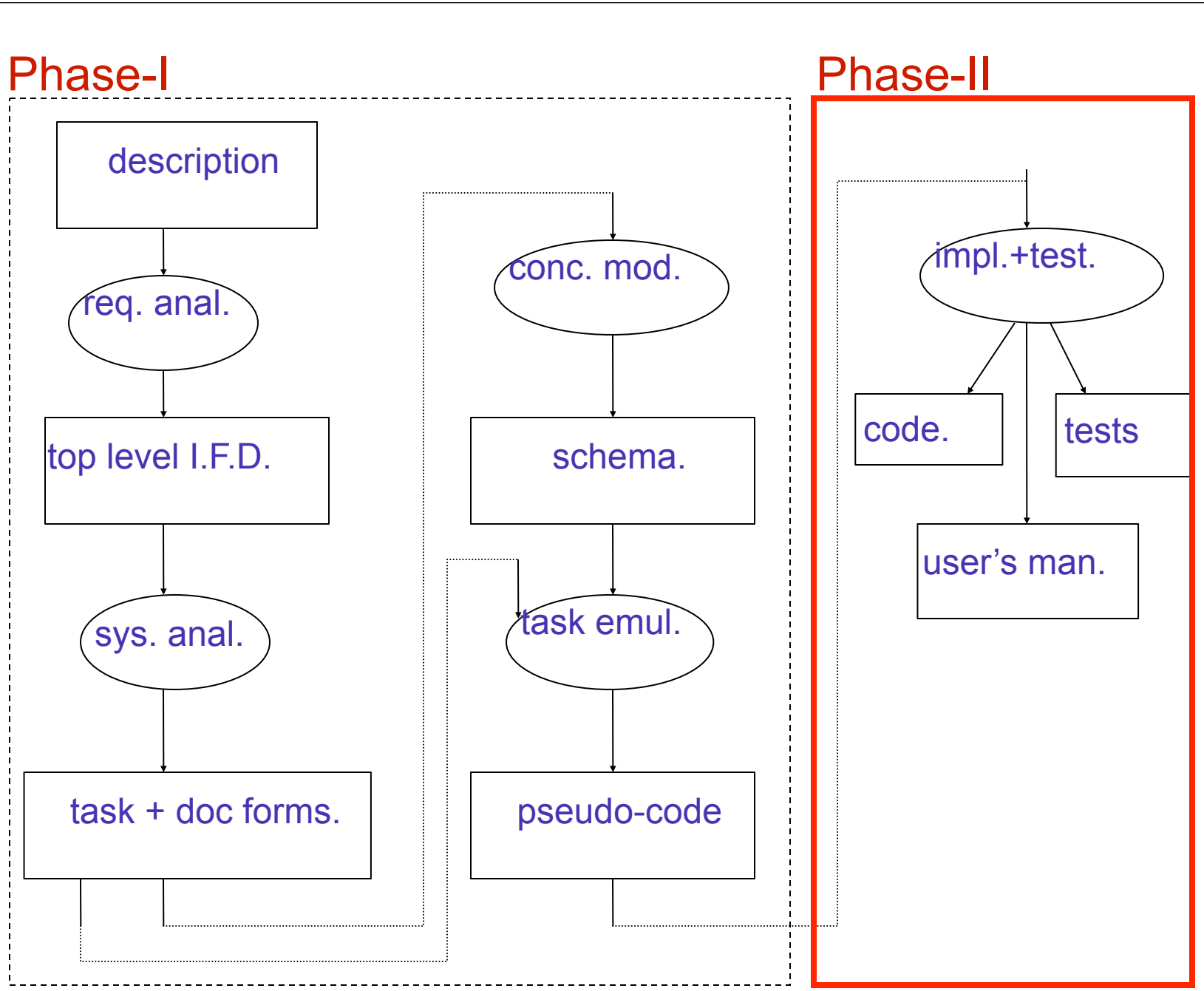

Phase-I
Phase-II
description
req. anal.
top level I.F.D.
sys. anal.
task + doc forms.
conc. mod.
schema.
task emul.
pseudo-code
impl.+test.
code.
tests
user's man.

Task emulation/pseudo-code (Homework I)

Task1: Registration

read login, password and email
if (login does not exist in 'users'){
insert into users values (login_id, password, email);
} else{
print "error: login exists, choose another login name"
}

should be valid SQL queries

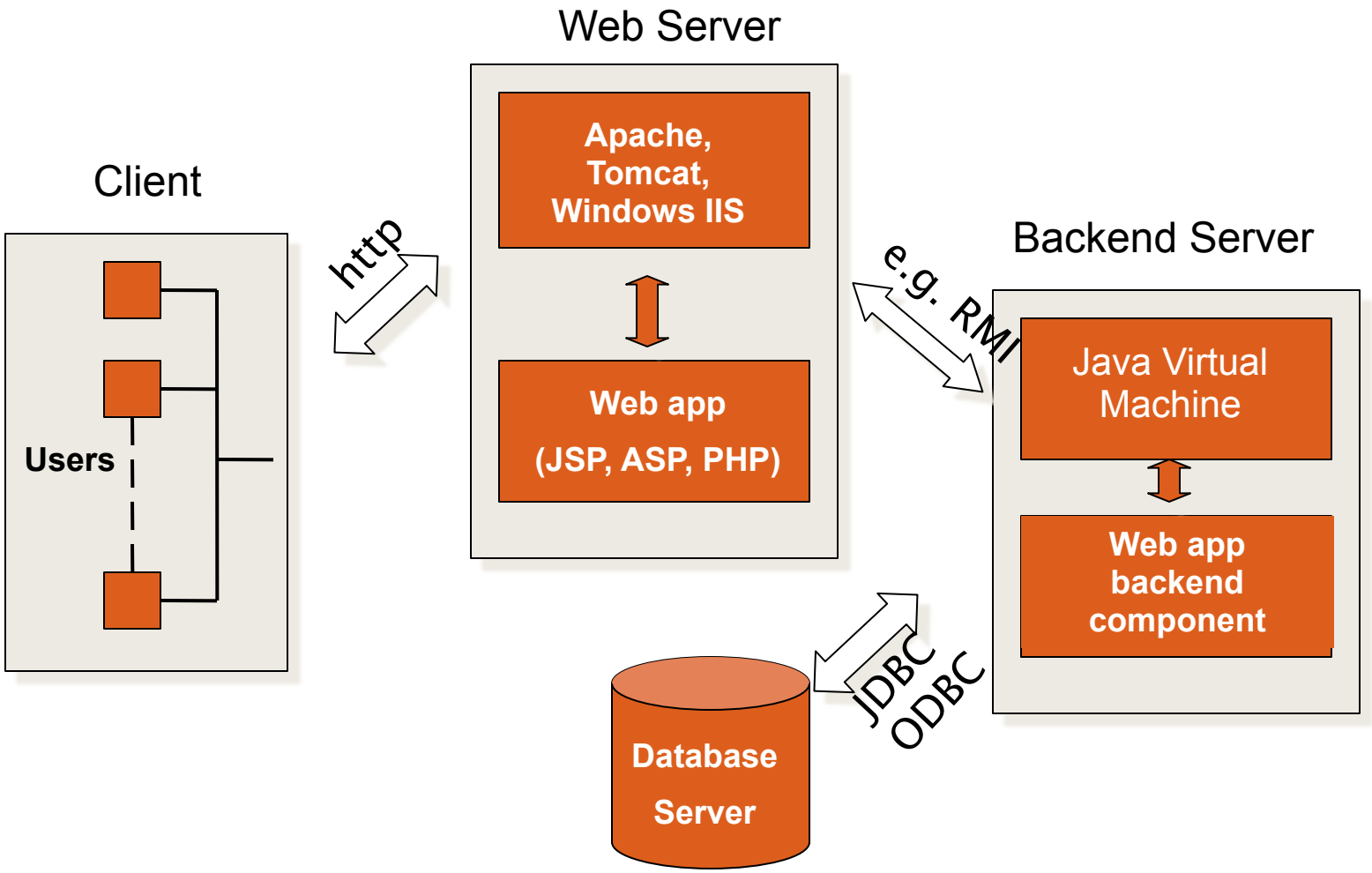

Phase-I
Phase-II
description
req. anal.
top level I.F.D.
sys. anal.
task + doc forms.
conc. mod.
schema.
task emul.
pseudo-code
impl.+test.
code.
tests
user's man.

Phase II

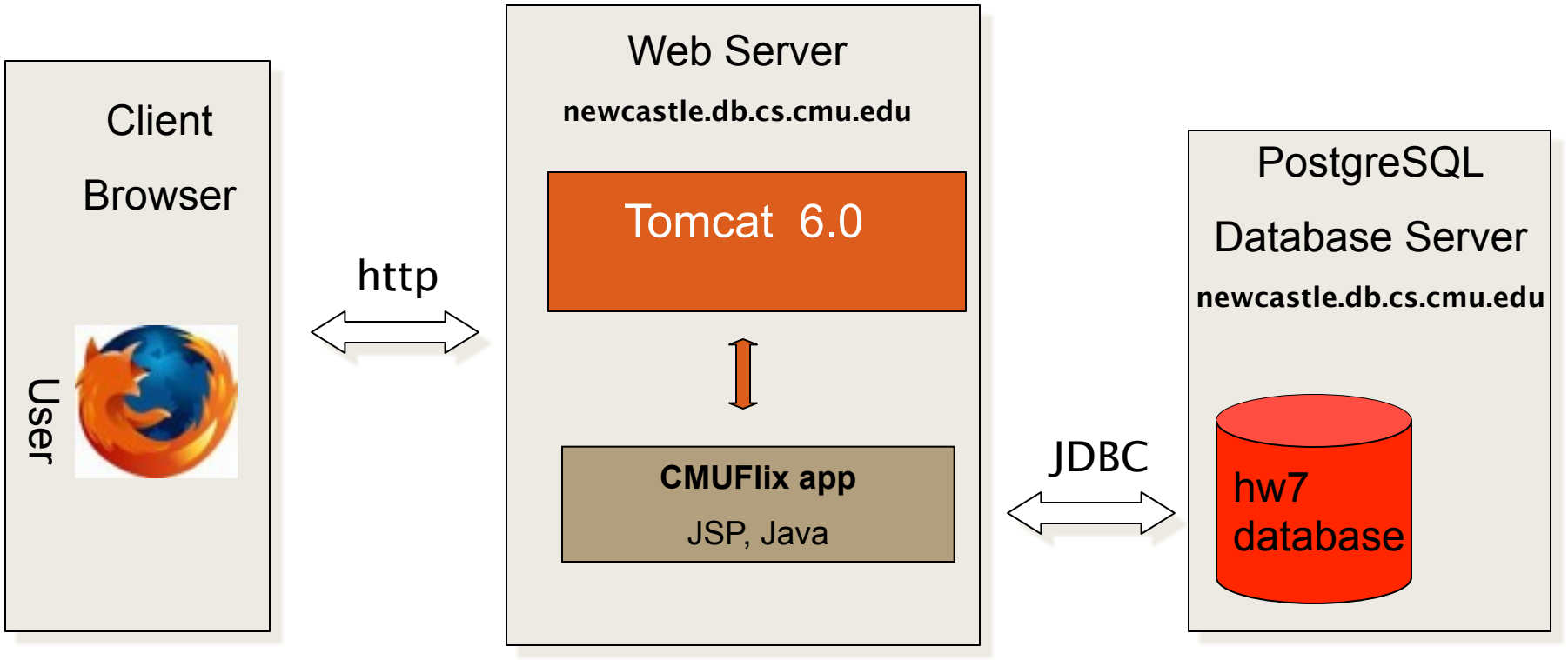
You will develop

- JSP pages that handle user interactions
- Processing logic written in Java class
- Manipulating data in database, connect from JSP to database using JDBC

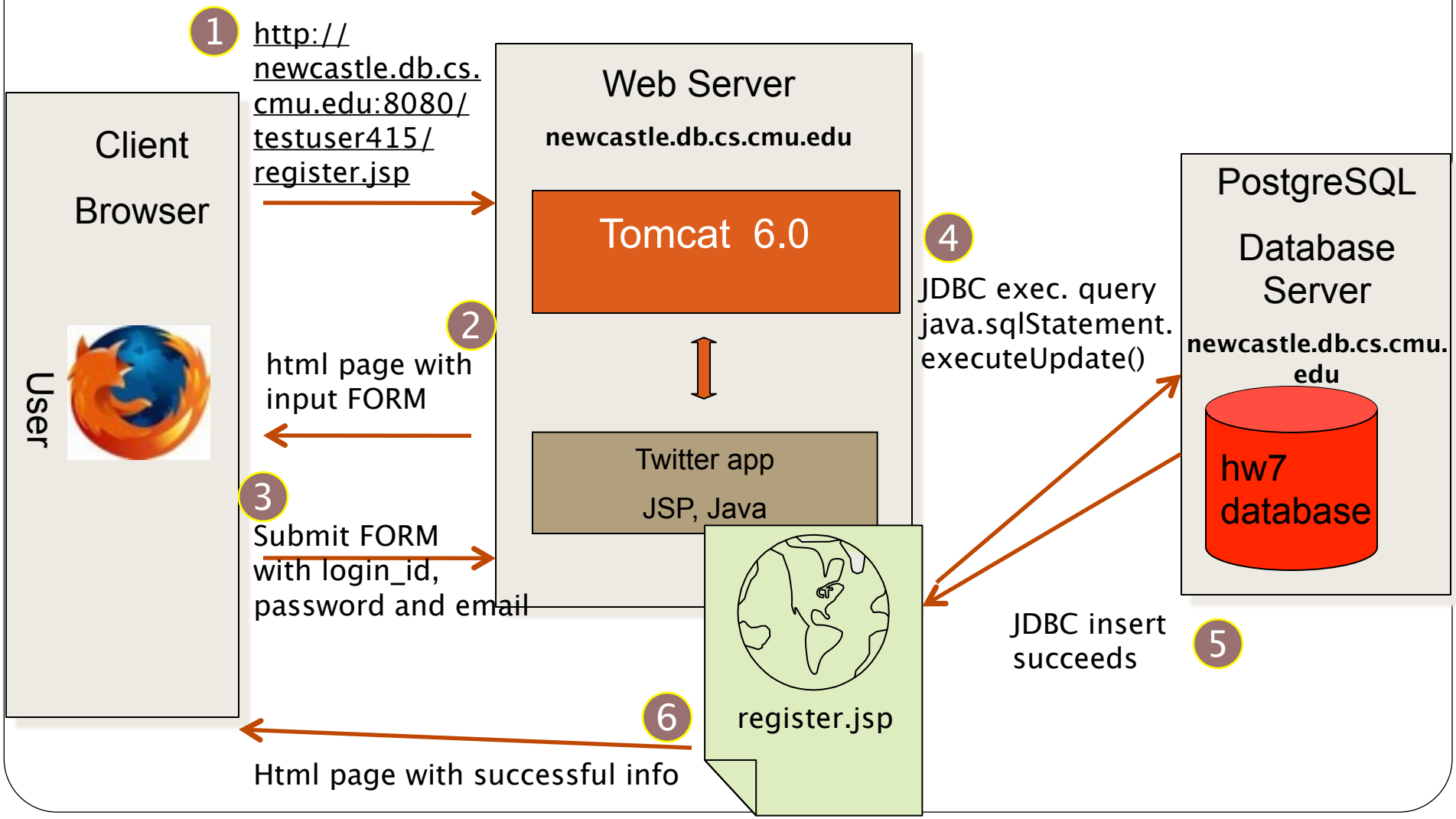
Web Application Architecture
Multi-tier architecture



Homework 7: Architecture



Registration example – register.jsp



Registration example – register.jsp

register.jsp

```
1  <html>
2  <%@ page import="java.sql.*"%>
3  <%
4    String fullname = request.getParameter("fullname");
5    String email = request.getParameter("email");
6    String login = request.getParameter("login");
7    String passwd = request.getParameter("passwd");
8    String submit = request.getParameter("submit");
9
10   if (submit==null) {
11 %>
12 <body>
13 <h1>
14 CMUBook Registration
15 </h1>
16 <form method="post" action="register.jsp">
17   Login Name: <input name="login" type="text" /> <br />
18   Full Name: <input name="fullname" type="text" /> <br />
19   Password: <input name="passwd" type="password" /> <br />
20   Email: <input name="email" type="text" /> <br />
21   <input name="submit" type="submit" value="Submit" />
22 </form>
23 </body>
24 <%
```

Registration example – register.jsp (continued)

```
25    ) else {
26  %>
27  <body>
28  <%
29    Connection conn = null;
30    Statement stmt = null;
31    try {
32      Class.forName("org.postgresql.Driver");
33      conn = DriverManager.getConnection("jdbc:postgresql://localhost:40123/hw9?user=www&password=lakoglu415");
34      stmt = conn.createStatement();
35      int r = stmt.executeUpdate("INSERT INTO users(login, fullname, passwd, email)
36              VALUES ('" + login + "','" + fullname + "','" + passwd + "','" + email + "')");
37      if (r==1) {
38        out.println("Register successfuly");
39      } else {
40        out.println("Register failed!");
41      }
42    } catch (Exception ex) {
43      ex.printStackTrace();
44    } finally {
45     //this is important: free up resources. Always. In a finally block.
46      stmt.close();
47      conn.close();
48    }
49  %>
50  </body>
51  <%
52    }
53  %>
54  </html>
```

JSP Basics

- JSP is a technology that helps software developers serve dynamically generated web pages
- Similar to PHP and ASP (Microsoft), but using Java: It simply “put Java inside HTML”.

JSP Basics

- Three primitives
  - – expressions
  - – directives
  - – declarations

JSP Basics – expressions

- JSP is being turned into a Java file, compiled and loaded

```
<HTML> <BODY>
Hello!  The time is now <%= new java.util.Date() %>
</BODY> </HTML>
```

- <%= and %> enclose Java expressions, which are evaluated at run time
- **Scriptlets:** blocks of Java code (<% and %>)

JSP Basics – directives

- JSP "directives" starts with <%@ characters.
  - "page directive": can import external java (can be user-defined) classes.

    <%@ page import="java.util.*, MyPackage.MyClass" %>
  - “include directive”: can include other jsp/html files.

    <%@ include file="hello.jsp" %>

JSP Basics – declarations

- The JSP code turns into a class definition. All the scriptlets are placed in a single method of this class.
- Can add variable and method declarations to this class. These variables and methods can later be "called" from your scriptlets and expressions.
- **<%!** and **%>** sequences enclose your declarations

```
<%@ page import="java.util.*" %>
<HTML> <BODY>
<%!      Date theDate = new Date();
         Date getDate()     {
               System.out.println( "In getDate() method" );
                return theDate;
          }
%>

Hello!  The time is now <%= getDate() %>
</BODY> </HTML>
```

JSP Basics – communication w/ server

- A "request" in server-side processing refers to the transaction between a browser and the server.
  - request.getParameter(“login”); // Common use: get the query string values
- A "response" is used to affect the response being sent to the browser.
  - response.addCookie(cookie); // See later slide for Cookies
  - response.sendRedirect(anotherUrl); // Tell browser to jump to another URL

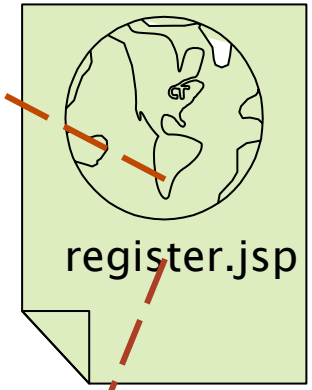
Connect JSP to database

```
<%
            …
Connection conn = null;
Statement stmt = null;
ResultSet r = null;

Class.forName("org.postgresql.Driver");
conn = DriverManager.getConnection
("jdbc:postgresql://localhost:40032/hw7?
user=www&password=415pass");

stmt = conn.createStatement();
r = stmt.executeQuery(your-SQL-query);
    if (r.next()) {
      session.setAttribute("login", r.getString(1));

            …
%>
```



JSP Basics – Session management

- On a typical web site, a visitor might visit several pages and perform several interactions. If you are programming the site, it is very helpful to be able to associate some data with each visitor.
- For example, after a user logs in, you might want to keep the login name and/or other information of the user to maintain his/her credential.
- Two ways to implement in JSP: session, and cookies (optional).

JSP Basics – sessions (method#1)

- A session is an object associated with a visitor.
- If you open different browsers, or use different IPs, you are essentially use multiple sessions.
- Data can be put in the session and retrieved from it, much like a Hashtable.
  - session.setAttribute("theName", name);
  - session.getAttribute("theName")
- Default expiration time: 30 minutes. You don't need to change it in the homework.

(Optional)

JSP Basics – cookies (method#2)

- Cookies are commonly used for session management.
- short pieces of data sent by web servers to the client browser
- saved to clients hard disk in the form of a small text file
- helps the web servers to identify web users, by this way server tracks the user.

Cookie example

```
String username=request.getParameter("username");
Cookie cookie = new Cookie ("username",username);
cookie.setMaxAge(365 * 24 * 60 * 60);
response.addCookie(cookie);
```

```
<%
String cookieName = "username";
Cookie cookies [] = request.getCookies ();
Cookie myCookie = null;
if (cookies != null){
        for (int i = 0; i < cookies.length; i++) {
                if (cookies [i].getName().equals (cookieName)){
                myCookie = cookies[i];
                break;
                }
        }
}
%>
<p>Welcome: <%=myCookie.getValue()%>.
<%
```

JSP Basics – Exception Handling

```
try {

        ….

 }
catch (Exception ex) {
        ex.printStackTrace();
        out.println("Login failed!");
        out.println("<a href=login.jsp>Go back to login!</a>");

 }
```

// ***out***: Output stream for page context, i.e., the content to show in the HTML.

(Optional)

Better Software Design?

- Design Pattern
  - Design Patterns: Elements of Reusable Object-Oriented Software, Gamma, Erich; Richard Helm, Ralph Johnson, and John Vlissides
  - Thinking in Patterns (http://www.mindviewinc.com/downloads/TIPatterns-0.9.zip)
- Design Pattern for Web App:
  - MVC pattern (Model – View – Controller)
  - Basic idea: break software functionalities and interfaces
  - Tool: struts