3. Design

https://github.com/WelecomeMe/js-Personal-Health

Personal Health Management System Based on JSP and Servlet

(Student No, Name, E-mail)

Student No: 22113877

Name: 딩카이

E-mail: q1793632722@gmail.com

Yeungnam University

[Revision history]

Revision date	Version #	Description	Author
2025/3.25	1.00	1 Draft	Author name
2025/4/25	1.10	2	
2025/5/22	1.50	3	

Yeungnam University

= Contents =

1. Introduction

1.1 Background of the Topic

Health is a timeless concern for humanity and the foundation of human existence. The World Health Organization (WHO) defines health as "a state of complete physical, mental, and social well-being, and not merely the absence of disease or infirmity." This means that a truly healthy person must be physically and mentally sound, socially adaptable, and morally upright.

With the advancement of society and the improvement of living standards, public awareness of health has steadily increased. People are paying more attention to disease prevention and personal wellness. At the same time, the rapid development of IT technologies and the widespread use of the Internet have led to the deep integration of computer applications into various aspects of social life.

As Hospital Information Systems (HIS) expand in scope and scale, data processing models centered around patient information have become a major focus in computer application research. In response to the rising demands for improved health management, how to effectively integrate computer technology into healthcare and provide support for personal diagnosis and daily health management has become a critical and ongoing area of research.

1.2 Purpose and Significance

The Personal Health Management System is a sophisticated human-computer interaction platform that integrates information science, data processing, and computer technology. It aims to help users enhance their health management by establishing personalized wellness plans, guiding and managing daily habits, and providing tools for disease prevention and monitoring. Through dynamic health tracking and early warning alerts, the system supports users in maintaining and improving their well-being.

The system provides a comprehensive platform for users to manage and access health information, including personal physical examination records, daily health behavior logs, health consultations, and educational resources. By encouraging scientific and healthy lifestyles, the system enhances users' health awareness and supports self-managed healthcare and disease prevention.

Key Design Features

- **Architecture**: Based on a B/S (Browser/Server) structure with JSP for the frontend, Servlet for the backend, and SQL Server as the database.
- **User Roles**: Supports multiple user roles including general users, doctors, and administrators, each with different permission levels.
- **Functional Modules**: Includes daily health logging, physical examination management, health information publishing and browsing, and online consultation.
- **System Considerations**: Emphasizes user-friendly interface design, data security, and system scalability.

2. Class diagram

- Draw a class diagram.
- Describe each class in detail (attributes, methods, others) (table type).

In computer information systems, the database plays a central role. With the support of a Database Management System (DBMS), it enables key operations such as information collection, organization, storage, retrieval, updating, processing, statistical analysis, and distribution. The quality of database design directly affects the overall quality and efficiency of the system.

The design of a database typically follows five stages: **planning, requirement analysis, conceptual design, logical design,** and **physical design**.

This section focuses on **conceptual design**, which builds the conceptual structure of the database from the bottom up, based on prior data analysis. From the user's perspective, views are designed first, then integrated and optimized to produce a final unified schema.

The conceptual design uses the **Entity-Relationship (E-R) model**, which consists of three core components: **entities**, **attributes**, and **relationships**. These components are expressed visually using **E-R diagrams**, where:

Entities represent real-world objects or concepts involved in the user environment,

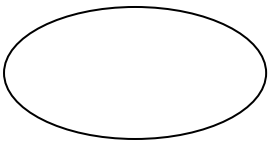
Attributes describe the characteristics of those entities,

Relationships define how entities are associated with one another.

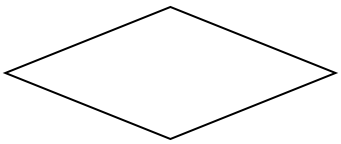
The goal of conceptual design is to produce a **conceptual schema** that accurately reflects the organization's information requirements. This schema is independent of any logical data model, DBMS, or hardware/software platform.

Based on the requirement analysis and the characteristics of the system, the following E-R diagrams were developed to illustrate the conceptual model of the database.

Entity

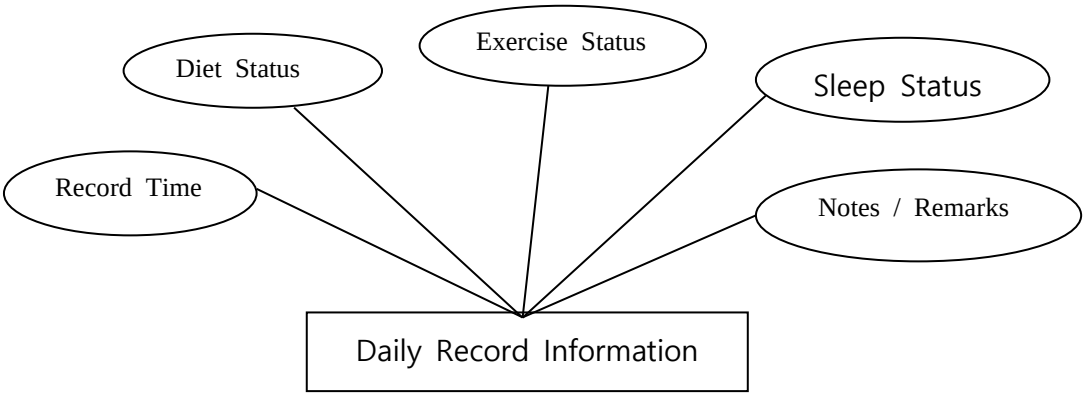


Attribute

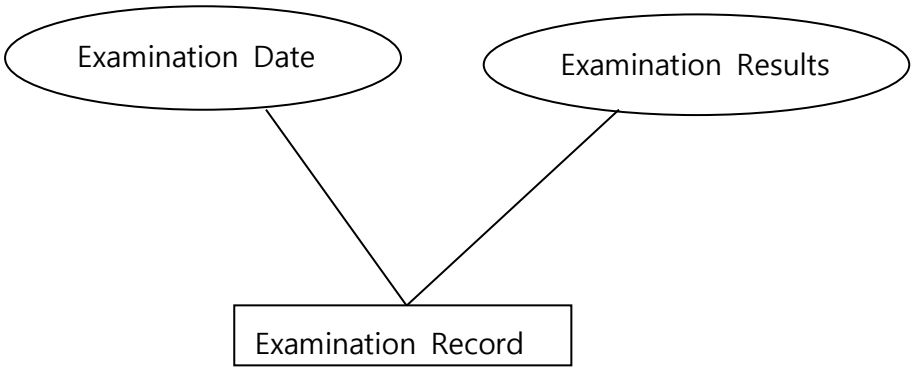


Relationship

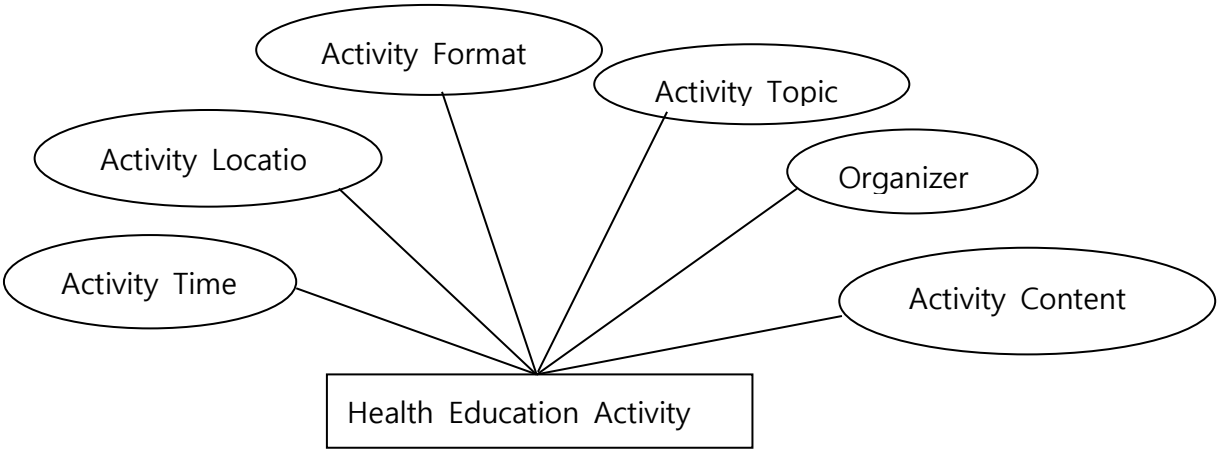
(1) Daily Record Information Entity — E-R Diagram



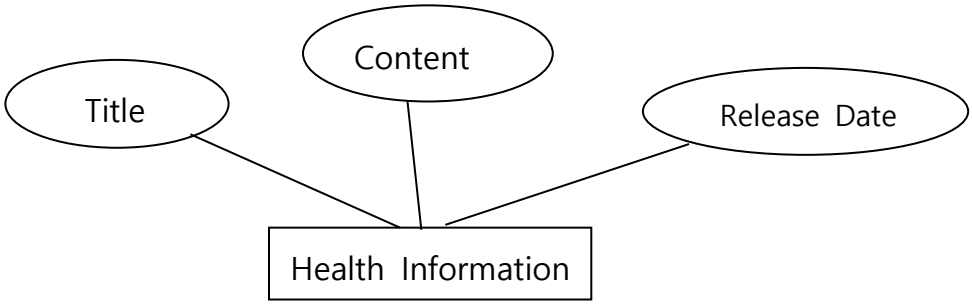
(2) E-R Diagram of the Physical Examination Information Entity



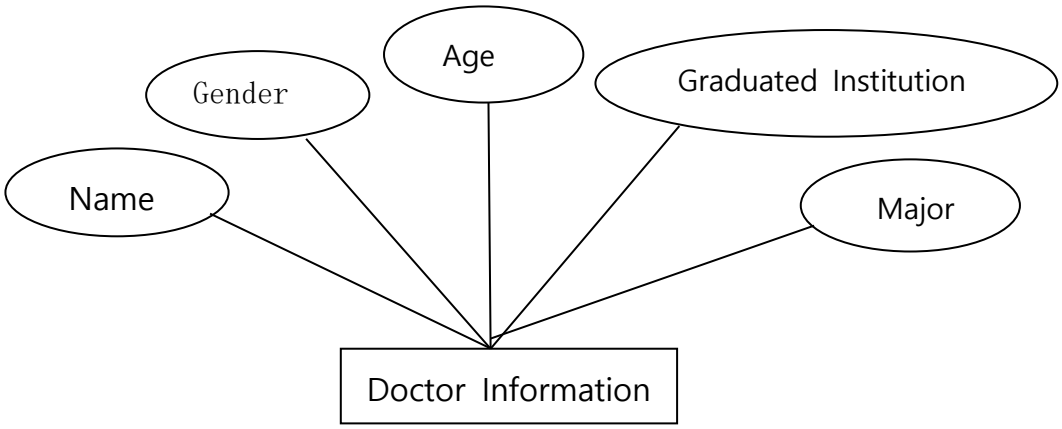
(2) E-R Diagram of Health Education Activity Information



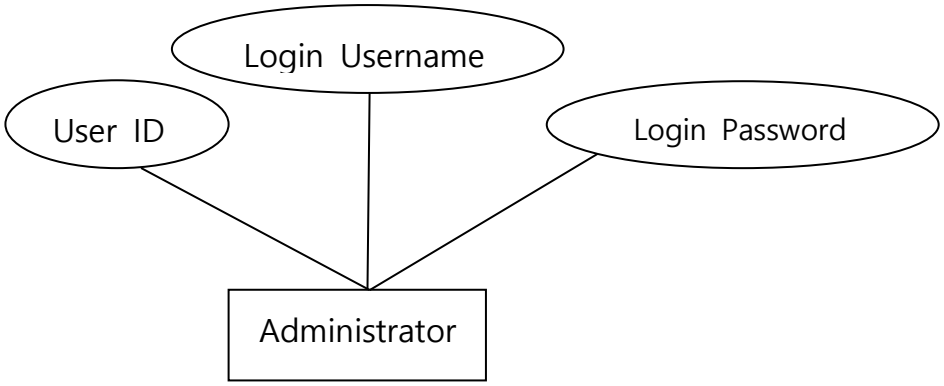
E-R Diagram of Health Information



E-R Diagram of Doctor Information



E-R Diagram of Administrator Information



Logical Database Design

Since the conceptual data model is independent of any specific DBMS, it needs to be converted into a logical relational model based on the features of the DBMS in use. The transformation from the E-R model to the relational model should follow th ese principles:

Each entity is converted into a relation (table)

All primary keys must be defined as NOT NULL

Binary relationships should define foreign keys based on their types (one-to-many, w eak to strong, one-to-one, many-to-many)

Based on the E-R model, the logical data structure for the Personal Health Manage ment System is designed as follows:

(1) General User Table

Table 3.1: General User Table (t_yonghu)

Column Name	Data Type	Length	Nullable	Primary Key	Description
id	int	4	No	Yes	ID
loginName	varchar	50	No	No	Account Name
loginPass	varchar	50	No	No	Password
xingming	varchar	50	No	No	Name
xingbie	varchar	50	No	No	Gender
shengri	varchar	50	No	No	Date of Birth

(2) Daily Record Table

Table 3.2: Daily Record Table (t_jilu)

Column Name	Data Type	Length	Nullable	Primary Key	Description
id	int	4	No	Yes	ID
yonghu_id	varchar	4	No	No	User ID
shijian	varchar	50	No	No	Record Time
yinshi	varchar	50	No	No	Diet Status
yundong	varchar	50	No	No	Exercise Status
shuimian	varchar	50	No	No	Sleep Status
beizhu	varchar	50	No	No	Remarks

(3) Physical Examination Table

Table 3.3: Physical Examination Table (t_tijian)

Column Name	Data Type	Length	Nullable	Primary Key	Description
id	int	4	No	Yes	ID
yonghu_id	int	4	No	No	User ID
shijian	varchar	50	No	No	Examination Time
jieguo	varchar	50	No	No	Examination Result

(4) Health Education Activity Table

Table 3.4: Health Education Activity Table (t_huodong)

Column Name	Data Type	Length	Nullable	Primary Key	Description
id	int	2	No	Yes	ID
yonghu_id	int	50	No	No	User ID
shijian	varchar	50	No	No	Activity Time
didian	varchar	50	No	No	Location
xingshi	varchar	50	No	No	Activity Form
zhuti	varchar	50	No	No	Activity Theme
zuzhizhe	varchar	50	No	No	Organizer
neirong	varchar	50	No	No	Activity Content

(5) Doctor Information Table

Table 3.5: Doctor Information Table (t_yisheng)

Column Name	Data Type	Length	Nullable	Primary Key	Description
id	int	4	No	Yes	ID
xingming	varchar	50	No	No	Name
xingbie	date	8	No	No	Gender
nianling	int	4	No	No	Age
biye	int	4	No	No	Graduate School
loginname	int	4	No	No	Username
loginpw	int	4	No	No	Password

(6) Consultation Table

Table 3.6: Consultation Table (t_liuyan)

Column Name	Data Type	Length	Nullable	Primary Key	Description
id	int	4	No	Yes	ID
content	varchar	5000	No	No	Content
liuyanshijian	varchar	50	No	No	Consultation Time
huifu	varchar	50	No	No	Reply Content
huifushijian	varchar	50	No	No	Reply Time
yonghu_id	int	4	No	No	User ID
yisheng_id	int	4	No	No	Doctor ID

(7) Administrator Table

Table 3.7: Administrator Table (t_admin)

Column Name	Data Type	Length	Nullable	Primary Key	Description
userId	int	4	No	Yes	ID
userName	varchar	50	No	No	Username
userPw	varchar	50	No	No	Password

Database Connectivity Principle

The system uses JDBC to connect to the database. By simply importing the corresponding database JAR package into the project, the database can be accessed easily. The Class.forName() method is used to load the driver, and the DriverManager.getConnection() method creates a connection.

The system follows the DAO (Data Access Object) pattern for database operations. DAO is a classical design pattern in Java programming, widely used and a fundamental part of the persistence layer in the J2EE architecture. DAO is based on a layered software architecture and abstracts access to data sources. This abstraction means developers don't need to know the physical location or type of the database—they can simply operate on encapsulated data objects.

DAO Pattern Class Diagram:

Figure 3.10 DAO Pattern Class Diagram

BusinessObject: The business object, which acts as the client of the DAO pattern

DataTransferObject: Transfers data between different layers of the application

DataObjectAccess: Encapsulates basic operations for the data source

DataSource: Refers to the data source itself

This structure effectively separates business logic from data logic, resulting in well-layered and maintainable code. The system uses a DBContent class to simplify database connections. Example code is as follows:

```
public DBContent(){
    String CLASSFORNAME = "com.microsoft.sqlserver.jdbc.SQLServerDriver";
    String url = "jdbc:sqlserver://localhost:1433;databaseName=db_jiankang";
    String user = "sa";
    String password = "sa";
    try {
        Class.forName(CLASSFORNAME);
        con = DriverManager.getConnection(url, user, password);
        stmt = con.createStatement(ResultSet.TYPE_SCROLL_SENSITIVE, ResultSet.
CONCUR_UPDATABLE);
    } catch (Exception ex) {
        ex.printStackTrace();
    }
}
```

Wherever a database connection is required, you can simply instantiate a DBContent object to establish and operate on the database.

3. Sequence Diagram

This section presents detailed sequence flows for the system's primary functional mo dules. Each diagram illustrates the dynamic interactions among users, web interfaces, backend servlets, and the database, reflecting how typical user actions are process ed and handled within the system. These interactions include authentication, data m anagement, role-specific operations, and feedback mechanisms designed to ensure s ystem reliability and user responsiveness.

3.1 User Login Flow

Actors: User (Admin / Regular User / Doctor), Web Interface, LoginServlet, Databas e

Description:

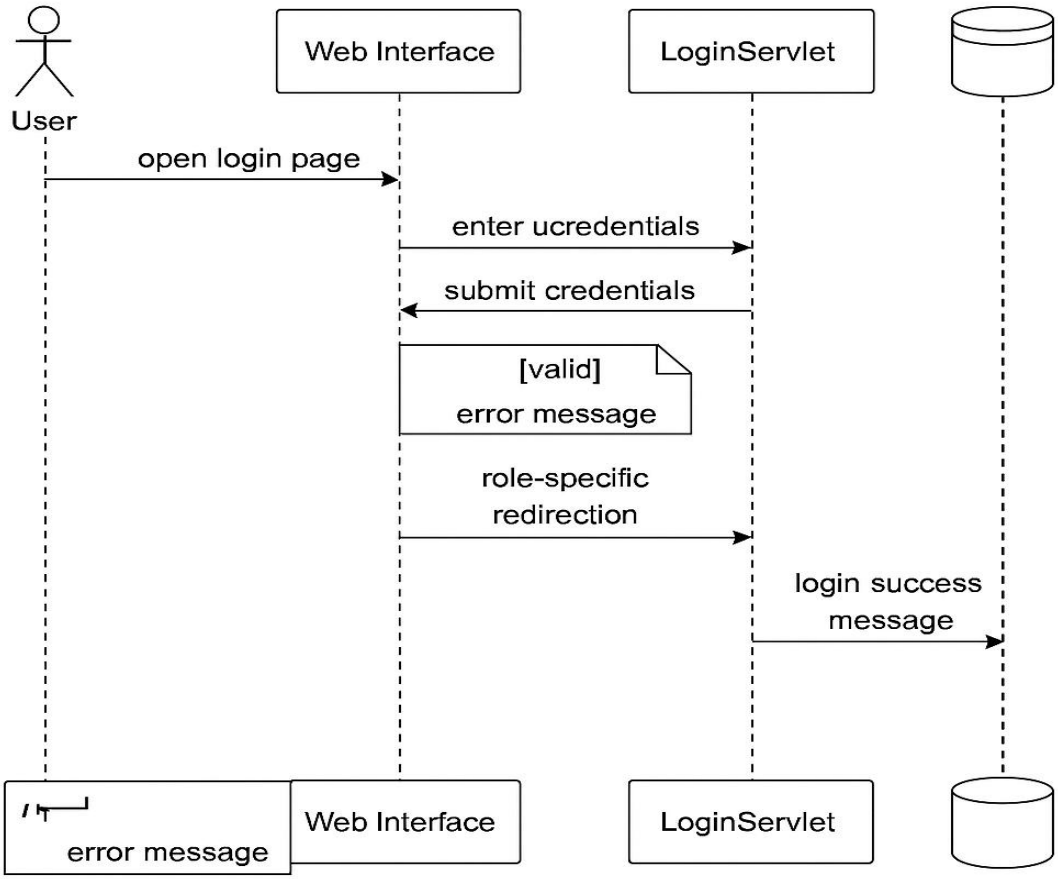
This process enables users to authenticate their identity and gain access to the syst em based on their assigned role. The login mechanism ensures role-based redirectio n and session initialization.

Sequence:

1. The user accesses the login page and inputs their **username**, **password**, and selects their **role type** (admin, user, or doctor).
2. Upon form submission, the credentials are transmitted to the LoginServlet via HTTP POST.
3. The servlet performs validation on the received input (e.g., null checks, length constraints).
4. A database query is constructed dynamically based on the selected role and executed to verify user identity.
5. If a match is found:
   A new session is created.
   Relevant user information is stored in session attributes.
   The user is redirected to the role-specific main interface:
   Admin → admin/index.jsp
   User → user/index.jsp
   Doctor → doctor/index.jsp
6. If the credentials are invalid or the user is not found:
   An error message is returned and the login page is reloaded.

Error Handling:

Empty input fields trigger client-side and server-side validation.
ncorrect role combinations or SQL exceptions are caught and logged securely.
Timeout or session expiration is handled via HTTP session lifecycle managem ent.



3.2 Health News Management (Admin Only)

Actors: Administrator, HealthNewsServlet, Web Interface, Database

Description:

Administrators can manage health-related announcements including creation, display, and deletion. These announcements are then displayed to regular users through the user interface.

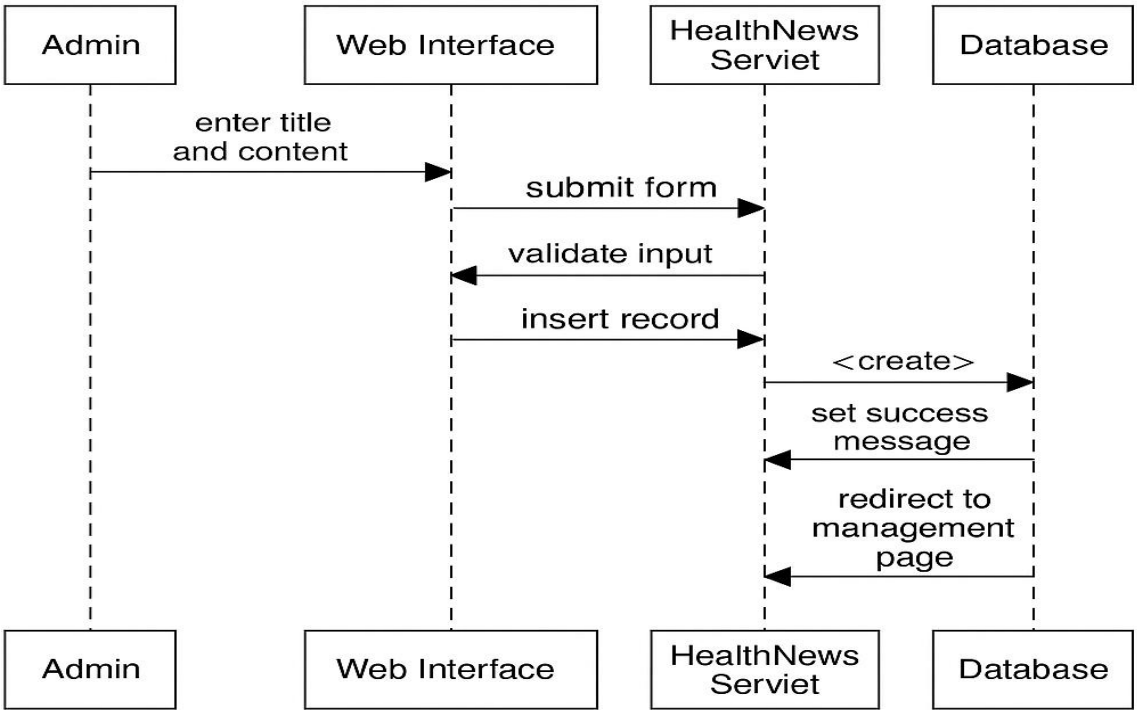
Use Cases:

Add new health news
Browse health news list
Remove outdated entries

Sequence (Add Operation):

1. The admin opens the “Add News” interface and provides the **title** and **content** of the announcement.
2. The form is submitted to HealthNewsServlet with the operation type flag.
3. The servlet performs input validation and inserts the record into the t_gonggao table along with a timestamp.
4. Upon success, the servlet returns a success notification and redirects to the management dashboard.
5. Users can view the newly published news on their respective pages.

3.2 Health News Management Sequence Diagram



3.3 **Doctor Information Management (Admin Only)**
Actors: Administrator, DoctorServlet, Web Interface, Database
Description:
This module allows administrators to manage doctor profiles including creation, logic al deletion, and listing.
Sequence (Add Operation):

1. Admin accesses the doctor information form and enters details such as name, gender, age, university, username, and password.
2. The form data is submitted to DoctorServlet.
3. The servlet verifies input validity (e.g., non-empty fields, password format).
4. A new doctor record is inserted into the t_yisheng table.
5. The system returns to the listing interface showing updated records.

Soft Delete Operation:
Upon delete request, the del flag in the database is updated to 'yes', effective ly hiding the doctor from active views without physically deleting the record.

3.4 Daily Health Record Entry (Regular User)
Actors: Regular User, Web Interface, HealthRecordServlet, Database
Description:
This process allows users to maintain daily records of personal health-related behavi ors. Each entry includes information such as diet, exercise, sleep quality, and option al remarks, contributing to long-term health data monitoring.
Sequence:

1. The user accesses the health record input page.
2. The interface prompts the user to fill out fields including:
   Record Date
   Diet Status
   Exercise Status
   Sleep Status
   Remarks / Notes
3. After completing the form, the data is submitted to HealthRecordServlet.
4. The servlet validates all input fields and constructs a prepared SQL statement to insert data into the t_jilu table.
5. Upon successful storage:
   A confirmation message is returned.
   The user is redirected to a summary view of their recent records.
6. In case of errors (e.g., missing fields, invalid date format), the system returns user-friendly feedback and retains partial form input for correction.

3.5 Physical Examination Record Entry (Regular User)

Actors: Regular User, Web Interface, PhysicalExamServlet, Database
Description:
Users may record formal physical examination data such as test dates and results, supporting periodic health assessments and consultation with doctors.
Sequence:

1. The user navigates to the “Add Physical Exam” page.
2. Required inputs include:
   Examination Date
   Result Summary
3. The user submits the form to PhysicalExamServlet.
4. The servlet inserts the data into the t_tijian table with appropriate user ID link age.
5. On successful insertion, the user is redirected to a list view of past examinati ons.

Error Handling:
Input validation includes date formatting and result field non-null checks. Duplicate date entries may be flagged depending on business rules.

3.6 Online Consultation with Doctor

Actors: Regular User, Doctor, Web Interface, ConsultationServlet, Database
Description:
The online consultation module enables asynchronous communication between users and doctors. Users may submit health-related questions, and doctors respond throu gh a secure backend system.
User Message Submission:

1. The user fills in a **consultation message** and selects a target doctor.
2. On submission, the message is sent to ConsultationServlet.
3. The servlet saves the message in the t_liuyan table with fields:
   content
   yonghu_id
   yisheng_id

liuyanshijian (timestamp)

4. The system acknowledges receipt of the message and marks it as “awaiting r esponse”.

Doctor Reply Workflow:

1. The doctor logs into their portal and reviews pending consultation messages.
2. After selecting a message, the doctor provides a **reply message**.
3. The reply is stored in the huifu field, along with huifushijian for timestamp.
4. The status is updated, and the user is able to view the response upon next l ogin.

Error Prevention:

Users can only consult with active doctors.
Replies are limited in length and encoded to prevent injection.

3.7 Password Management and Logout

Actors: All User Roles

Change Password

Description:

All user types may update their password after providing their current credentials.

Sequence:

1. The user navigates to the change password interface.
2. Inputs required:
   Old password
   New password
   Confirmation of new password
3. The servlet verifies the old password matches the current session credentials.
4. If valid, the new password is updated in the corresponding user table.
5. A success message is returned; otherwise, the user is prompted to retry.

Logout

Description:

Logging out ensures that session data is cleared and access control is enforced.

Sequence:

1. User clicks the “Logout” option.
2. A JavaScript function or a servlet invalidates the HTTP session.
3. The user is redirected to the system’s login page (index.jsp).

4. State Machine Diagram

This section describes the system's behavior in terms of its states and transitions. It illustrates how the application responds to different user actions and system events, focusing on both user-driven and background processes. The state machine model ensures that the application maintains predictable and secure flows across user role s and system operations.

4.1 Frontend (User-Oriented) State Transitions

This model reflects how the system behaves from the user's perspective, including l ogin, navigation, data entry, and role-based access.

States:

- **Initial State**: System not yet accessed
- **Login Page Displayed**
- **Authenticating**
- **Login Success**
- **Login Failure**
- **Role-Specific Main Menu**
  - Admin Dashboard
  - User Dashboard
  - Doctor Dashboard
- **Module Active States**
  - Add Health Record
  - View News
  - Submit Consultation
  - Reply to Consultation (Doctor)
  - Manage Doctor/News (Admin)
- **Logout**

Transitions:

- Initial → Login Page (when user accesses URL)
- Login Page → Authenticating (upon form submission)
- Authenticating → Login Success (if credentials valid)
- Authenticating → Login Failure (if invalid)
- Login Success → Role-Specific Main Menu (based on user role)
- Main Menu → Module State (based on selected action)
- Any → Logout (session ends)

4.2 Backend (System-Oriented) State Transitions

This model reflects the system’s background logic, including input validation, databas e operations, and error handling.

States:

- **Idle**
- **Receiving Request**
- **Validating Input**
- **Querying Database**
- **Writing to Database**
- **Responding to Client**
- **Error Handling**

Transitions:

- Idle → Receiving Request (HTTP request received)

Receiving Request → Validating Input (e.g., null check, format check)
Validating Input → Querying/Updating DB
DB Operation → Responding to Client (with data or status message)
Any State → Error Handling (on exception)
Error Handling → Idle (after log/report)

4.3 Combined Sample: Health Record Entry (User)

```
stateDiagram-v2
    [*] --> Login Page
    Login Page --> Authenticating
    Authenticating --> User Dashboard : loginSuccess
    User Dashboard --> Record Entry
    Record Entry --> Validating Input
    Validating Input --> Writing to DB : dataValid
    Writing to DB --> Record Entry : success
    Validating Input --> Record Entry : invalidInput
    Record Entry --> User Dashboard : cancel
    User Dashboard --> Logout
    Logout --> [*]
```

5. Implementation Requirements

This section outlines the software tools, technologies, and system environment required for the development and operation of the system. It also provides a brief introduction to the core technologies adopted, including their roles and benefits in the development lifecycle.

5.1 Development Tools and Technologies

5.1.1 MyEclipse

MyEclipse is an enterprise-level IDE built on the Eclipse platform, tailored for Java and J2EE development. It integrates support for a wide range of open-source technologies and provides tools for web development, database management, and server integration. Key features include:

- Complete support for HTML, CSS, JavaScript, JSP, and Struts
- Integrated debugging, testing, and deployment tools
- Enhanced productivity through database-to-code interaction and web project automation

5.1.2 Apache Tomcat

Tomcat is a lightweight Java web application server widely used for developing and testing JSP and Servlet-based applications. It functions independently as a servlet container and can be run alongside an Apache HTTP server or as a standalone process. Advantages include:

- Fast startup and response time
- Support for Java EE specifications (Servlet, JSP)
- Ideal for medium-scale deployments

5.1.3 Microsoft SQL Server

SQL Server is a high-performance relational database system supporting multiple operating systems and programming interfaces. It offers:

- High portability across platforms
- Multi-threaded performance for CPU efficiency
- Support for TCP/IP, JDBC, and ODBC
- Capabilities to handle millions of records
- Cost-effective deployment, especially for small to medium-sized systems

5.2 JSP Technology

JavaServer Pages (JSP) is a server-side scripting technology used to create dynamic, platform-independent web content. JSP files combine HTML and embedded Java code, which is processed on the server to generate dynamic output sent to the client's browser.

Key Advantages of JSP:

- *Write Once, Run Anywhere:* Platform-independent deployment
- *Component Reusability:* Supports JavaBeans and tag libraries
- *Separation of Concerns:* Presentation logic is separated from business logic
- *Robustness & Security:* Inherits Java's platform security and object-oriented features
- *Extensive Tool Support:* Compatible with a variety of powerful Java development tools

JSP Built-in Objects:

request – Client request object
response – Server response handler
session – Manages user session data
application – Global application context
pageContext – Page-level attribute management
out – Output stream for client response
config – JSP configuration information
page – Reference to the current JSP instance
exception – Captures uncaught exceptions

5.3 JavaScript

JavaScript is a client-side scripting language that enhances the interactivity and responsiveness of web applications. It is object-based and event-driven, making it suitable for:

Validating form inputs before server submission
Handling user-triggered events in real-time
Creating dynamic and interactive web interfaces

In this system, JavaScript is used extensively for front-end validation (e.g., checking for empty fields or duplicate entries) to reduce server-side load and improve user experience.

Component	Version/Specification
Operating System	Windows XP / Vista / Windows 7
Development IDE	MyEclipse 6.0.1
Web Server	Apache Tomcat 6.0
Programming Language	Java
Browser	Internet Explorer 6.0

6. Glossary

This section provides definitions of technical terms, class names, table names, and system-specific keywords used throughout the design and implementation of the system.

Term	Definition
JSP (JavaServer Pages)	A server-side technology that allows Java code to be embedded in HTML pages to generate dynamic web content.
Servlet	A Java class that handles HTTP requests and responses on the server side. Often used to control backend logic.
MyEclipse	An integrated development environment (IDE) built on Eclipse, supporting Java EE and web application development.
Apache Tomcat	A lightweight, open-source Java servlet container used to run Java web applications.
SQL Server	A relational database management system developed by Microsoft to store and retrieve data.
DAO (Data Access Object)	A design pattern used to abstract and encapsulate all access to a data source.
Session	A server-side storage mechanism used to retain user data across multiple HTTP requests.
JDBC (Java Database Connectivity)	An API that allows Java programs to interact with databases.
t_yonghu	Database table for storing regular user information.
t_jilu	Database table that stores users' daily health record data.
t_tijian	Table for physical examination records.
t_huodong	Table for storing information about health education activities.
t_yisheng	Table that holds doctor profile information.
t_liuyan	Consultation messages between users and doctors.
t_admin	Administrator credentials and access table.
HealthRecordServlet	A servlet handling insertion and retrieval of daily health records.
ConsultationServlet	A servlet that manages submission and reply to consultation messages.
HealthNewsServlet	A servlet for managing the creation, display, and deletion of health news articles.
User Roles	Distinct access levels in the system: Administrator, Regular User, Doctor.
Validation	The process of checking input data for correctness and security.
Redirect	Server response that sends the user to a different URL or page.
Response Object	Server-side object used to send data back to the client (browser).

7. References

1. Bai, Qiuchan; Gao, Aihua; Shen, Xianlai. *Physical Examination Information System Based on VFP*. Computer and Digital Engineering, 2006(1): 143–145.
2. Shu, Pan; Chen, Jingang. *Design and Implementation of Physical Examination System in Digital Campus*. Journal of Wuhan Institute of Technology, 2008(4): 108–111.
3. Xu, Chaoyi. *Object-Oriented Analysis of Personal Health Management System*. Journal of Anhui University of Science and Technology (Natural Science Edition), 2005(3): 62–64.
4. Ai, Lingxian. *Design and Implementation of Departmental Website Platform in Universities*. Science & Technology Information, 2008(16).
5. Zhang, Guoyu; Mou, Zongguo. *Improving Personal Health Management in Schools*. Journal of Sichuan Institute of Education, 2007(1): 19–20.
6. Yan, Yongjie. *How to Strengthen Health Management in Vocational Colleges*. Science & Technology Information, 2008(29): 592–593.
7. Wu, Hanlong. *Practical System, Practical Use: Design and Trial of the ZJSU Health Management System*. Journal of College Logistics, 2009(4): 94–97.
8. Xu, Yijin; Zheng, Chuhua. *Development of Medical Examination Management System*. Journal of Nanchang Aviation University, 2006(4): 94–97.
9. Hellerstein, J.M.; Stonebraker, M. *Architecture of a Database System*. Foundations and Trends in Databases, 2007, 1(2): 141–259.
10. Shou, Xiuxiang. *Design and Implementation of Health Management System Based on C/S Architecture*. Heilongjiang Science and Technology Information, 2008 (31): 55.
11. Zhang, Like. *Java Information System Development*. Beijing: Posts & Telecom Press.
12. Deng, Ziyun. *JSP Programming: From Fundamentals to Practice*. Beijing: Electronics Industry Press.
13. Zhu, Taojiang. *SQL Server Complete Reference Manual*. China Electric Power Press, 2003.
14. Hao, Yulong. *Java EE Programming Techniques*. Beijing: Beijing Jiaotong University Press.
15. Li, Qingsen; Liu, Yu; Hou, Yufeng. *On Integrated Management in College Systems*. Journal of Work Research, 2002.
16. Eckel, Bruce (Author), Chen, Haopeng (Translator). *Thinking in Java (3rd Edition)*. Beijing: Machinery Industry Press, 2007.
17. Li, Jianzhong; Wang, Shan. *Principles of Database Systems (2nd Edition)*. Electronics Industry Press, 2004.
18. Zhang, Changfu; Huang, Zhongmin. *JavaScript Dynamic Web Programming*. Beijing: Ocean Press, 2005: 196–239.
19. Ai, Lingxian. *Platform Design for Department Websites in Universities*. Science & Technology Information, 2008(16).