

3) 기능 목록

핵심 기능 (Core Features)

- 회원 관리:
 - 로그인: 세션(session)을 이용한 사용자 인증 및 비로그인 접근 차단.
 - 회원가입: 아이디 중복 체크 및 신규 사용자 등록.
 - 로그아웃: 세션 만료 및 로그인 페이지로 리다이렉트.
- 할 일(Todo) 관리:
 - 생성 (Create): 새로운 할 일 추가.
 - 조회 (Read): 로그인한 사용자의 할 일 목록을 순서대로(seq 기준) 표시.
 - 수정 (Update): 할 일 내용 수정 및 진행 상태(TODO/DONE) 토글.
 - 삭제 (Delete): 특정 할 일 삭제.

부가 기능 (Additional Features)

- 드래그 앤 드롭 정렬 (Reorder): 마우스 드래그로 할 일의 순서를 변경하고 DB에 저장 (seq 업데이트).
- 상세 메모 (Memo): 각 할 일에 대한 추가 메모 작성 및 저장 (토글 방식 UI).
- 다크 모드 (Dark Mode): localStorage를 활용하여 브라우저 종료 후에도 유지되는 테마 설정.
- 검색 및 필터링: 키보드 입력 시 실시간으로 할 일 목록 필터링 (JS DOM 조작).
- 완료 항목 일괄 삭제: 완료된(DONE) 상태의 할 일을 한 번에 삭제.
- Toast 알림: 작업 성공/실패 시 화면 하단에 부드럽게 나타나는 알림 메시지 제공.

페이지 구성도

- 진입: login.jsp
- 회원가입: register.jsp
- 메인: index.jsp (할 일 목록 관리)
- 로직 처리 (Backend):
 - loginAction.jsp, registerAction.jsp, logoutAction.jsp (회원 관련)
 - todoAction.jsp (할 일 추가, 수정, 삭제, 정렬 등 처리)

4) 전체 구조도 (System Architecture)

HTML/CSS – JSP – DB 데이터 흐름

데이터는 Client(브라우저) ↔ Server(Tomcat/JSP) ↔ Database(MySQL) 순서로 흐릅니다.

1. Client (View & Event):

- HTML5/CSS3로 렌더링되며 script.js가 사용자 이벤트(클릭, 드래그, 입력)를 감지합니다.

- 데이터 전송 방식: Form Submit (로그인, 추가) 및 AJAX/Fetch API (순서 변경, 메모 저장 등 화면 깜빡임 없는 처리).

2. Server (Controller & Logic):

- **Session Management:** session.getAttribute("userID")로 로그인 여부를 확인합니다.
- **Action Handling:** todoAction.jsp에서 action 파라미터(insert, delete, reorder 등)에 따라 로직을 분기합니다.
- **JDBC:** DBConnection.java를 통해 MySQL과 연결하고 PreparedStatement로 쿼리를 실행합니다.

3. Database (Storage):

- MySQL에 회원 정보와 투두 데이터를 영구 저장하며 ON DELETE CASCADE로 무결성을 유지합니다.

페이지 연결 흐름도 (Sitemap)

- login.jsp (진입) → index.jsp (성공 시) 또는 register.jsp (회원가입)
- register.jsp → registerAction.jsp → login.jsp
- index.jsp ↔ todoAction.jsp (AJAX 통신 또는 Form 제출 후 리다이렉트)
- index.jsp → logoutAction.jsp → login.jsp.

메인 UI 구조

사용자 경험(UX)을 고려하여 다음과 같이 배치되어 있습니다.

- **Header:** 사용자 환영 문구, 다크 모드 토글 스위치, 로그아웃 버튼.
- **Input:** 할 일 입력창(autofocus 적용) 및 추가 버튼.
- **Toolbar:** 실시간 검색창, 완료 항목 일괄 삭제 버튼.
- **List:** 할 일 목록(), 드래그 핸들러, 메모(토글), 수정/완료/삭제 버튼 그룹.

5) DB 설계 (Database Design)

회원별로 독립적인 관리가 가능하도록 **1:N** 관계로 설계되었으며, utf8mb4 문자셋을 사용하여 이모지를 지원합니다.

테이블 구조

| 테이블명 | 컬럼명 | 데이터 타입 | Key | Null | 설명 |
|--------------|----------|-------------|-----------|------|------------------|
| users | username | VARCHAR(50) | PK | NN | 사용자 아이디 (고유 식별자) |

| | | | | | |
|--------------|------------|--------------|-----------|-------------|------------------------|
| | password | VARCHAR(50) | | NN | 사용자 비밀번호 |
| todos | id | INT | PK | Auto Inc | 할 일 고유 번호 (자동 증가) |
| | username | VARCHAR(50) | FK | NN | 작성자 ID (users 테이블 참조) |
| | content | VARCHAR(200) | | NN | 할 일 내용 |
| | status | ENUM(...) | | | 진행 상태 ('TODO', 'DONE') |
| | seq | INT | | Default 0 | 드래그 정렬 순서 |
| | memo | TEXT | | Null | 상세 메모 |
| | created_at | TIMESTAMP | | Default Now | 생성 일시 |

주요 컬럼 설명

- **seq (순서):** 사용자가 드래그 앤 드롭으로 재정렬한 순서를 저장합니다. 조회 시 ORDER BY seq ASC, id ASC로 정렬하여 출력합니다.
- **status (상태):** ENUM('TODO', 'DONE') 타입을 사용하여 지정된 값 외의 데이터 입력을 차단, 데이터 무결성을 보장합니다.
- **memo (메모):** TEXT 타입을 사용하여 길이 제한 없이 긴 내용의 메모를 저장할 수 있습니다.

제약조건 (Constraints)

- **Foreign Key & Cascade:**
 - FOREIGN KEY (username) REFERENCES users(username) ON DELETE CASCADE
 - 회원이 탈퇴(삭제)할 경우 해당 회원이 작성한 모든 할 일 데이터도 자동으로

삭제되도록 하여 고아 레코드(Orphan Data) 발생을 방지합니다.

1. 핵심 기능 (Core Features)

① 회원 관리 시스템

- 회원가입: `register.jsp`와 `registerAction.jsp`를 통해 신규 회원을 등록합니다. 아이디 중복 확인 로직이 포함되어 있으며, 비밀번호 확인 절차를 거칩니다.
- 로그인/로그아웃: `login.jsp`와 `loginAction.jsp`에서 DB의 회원 정보와 대조하여 인증을 처리합니다. 인증 성공 시 `session`에 `userID`를 저장하여 로그인 상태를 유지하며, `logoutAction.jsp`를 통해 세션을 만료시킵니다.
- 접근 제어: 메인 페이지(`index.jsp`) 진입 시 세션을 확인하여 비로그인 사용자의 접근을 차단하고 로그인 페이지로 리다이렉트합니다.

② 할 일(Todo) 기본 관리 (CRUD)

- 할 일 추가 (Create): 메인 화면 상단의 입력창을 통해 새로운 할 일을 추가합니다. `todoAction.jsp`의 `action=insert` 로직을 통해 DB에 저장되며, 이때 정렬 순서(`seq`)는 현재 최대값 + 1로 자동 설정됩니다.
- 목록 조회 (Read): 로그인한 사용자의 할 일 목록을 DB에서 가져와 화면에 표시합니다. 이때 `seq` 컬럼을 기준으로 오름차순 정렬하여 사용자가 설정한 순서를 유지합니다.
- 수정 (Update):
 - 내용 수정: 할 일 내용을 수정할 수 있으며, `todoAction.jsp`의 `action=edit`에서 처리됩니다.
 - 상태 변경: 완료 버튼 클릭 시 TODO ↔ DONE 상태가 토글됩니다.
- 삭제 (Delete): 특정 할 일을 삭제할 수 있으며, 삭제 시 애니메이션 효과가 적용됩니다.

2. 부가 및 고급 기능 (Advanced Features)

① 드래그 앤 드롭 정렬 (Drag & Drop Reordering)

- HTML5의 Drag and Drop API를 사용하여 할 일의 순서를 직관적으로 변경할 수 있습니다.
- 순서 변경 시 `script.js`의 `saveOrder` 함수가 호출되어 변경된 순서(id 배열)를 서버로 전송하고, `todoAction.jsp`에서 일괄 업데이트(Batch Update)를 수행하여 DB에 영구 저장합니다.

② 상세 메모 (Memo)

- 각 할 일마다 별도의 메모를 작성하고 저장할 수 있는 기능을 제공합니다.
- 토글 방식의 UI로 메모장을 열고 닫을 수 있으며, TEXT 타입 컬럼을 사용하여 긴 내용도 저장 가능합니다.

③ 실시간 검색 (Real-time Filtering)

- 상단의 검색창에 입력 시 서버 요청 없이 자바스크립트로 DOM을 조작하여 실시간으로 일치하는 할 일만 필터링해 보여줍니다.

④ 완료 항목 일괄 삭제

4. 완료된(DONE) 상태의 할 일들을 한 번의 클릭으로 모두 삭제하여 목록을 정리할 수 있습니다.

⑤ 다크 모드 (Dark Mode)

2. CSS 변수(var(--bg-gradient) 등)를 활용하여 테마를 구현했습니다.
3. localStorage를 사용하여 사용자의 테마 설정(다크/라이트)을 브라우저 종료 후에도 기억합니다.

⑥ 토스트 알림 (Toast Notification)

2. 사용자의 작업(추가, 삭제, 저장 등) 결과에 대해 화면 하단에 부드럽게 나타났다가 사라지는 토스트 메시지를 띄워 피드백을 제공합니다.

3. 데이터베이스 설계 특징

2. **1:N 관계:** 하나의 회원이 여러 개의 할 일을 가지는 구조로 설계되었습니다.
3. **참조 무결성 유지:** FOREIGN KEY에 ON DELETE CASCADE 옵션을 설정하여, 회원이 탈퇴하면 해당 회원의 모든 할 일 데이터가 자동으로 삭제되도록 구현했습니다.
4. **이모지 지원:** 데이터베이스 문자셋을 utf8mb4로 설정하여 할 일 내용에 이모지를 사용할 수 있습니다.

9) 문제 해결 과정

① 구현하면서 발생한 문제 (Problems) & ② 해결 방법 (Solutions)

1. 데이터 무결성 및 고아 레코드(Orphan Data) 발생 문제

- 문제: 회원이 탈퇴(삭제)할 경우, 해당 회원이 작성했던 할 일(Todo) 데이터가 삭제되지 않고 데이터베이스에 남아있는 '고아 레코드'가 발생하여 데이터 공간을 낭비하고 무결성을 해치는 문제가 있었습니다.
- 해결: MySQL 데이터베이스 설계 시 todos 테이블의 username 외래키(Foreign Key)에 **ON DELETE CASCADE** 제약조건을 설정했습니다. 이를 통해 users 테이블에서 특정 회원이 삭제되면, 이를 참조하고 있는 todos 테이블의 모든 관련 데이터도 자동으로 연쇄 삭제되도록 구현하여 데이터 무결성을 확보했습니다.
- 배운 점: RDBMS의 참조 무결성 제약조건을 활용하면 애플리케이션 코드의 개입 없이도 데이터의 일관성을 효율적으로 관리할 수 있음을 배웠습니다.

2. 잦은 새로고침으로 인한 화면 깜빡임 및 사용자 경험 저하

- 문제: 할 일의 순서를 변경(Reorder)하거나 메모를 저장할 때마다 Form Submit 방식을 사용하여 페이지 전체를 새로고침(Reload)하니, 화면이 깜빡이고 사용 흐름이 끊기는 불편함이 있었습니다.
- 해결: 자바스크립트의 **Fetch API**를 활용한 **AJAX** 비동기 통신을 도입했습니다. 순서 변경이나 메모 저장과 같은 작업은 백그라운드에서 서버(todoAction.jsp)로 요청을 보내 처리하고, UI는 변경된 상태를 유지하도록 하여 화면 깜빡임 없는 부드러운 사용자 경험(UX)을 구현했습니다.
- 배운 점: 동적인 웹 애플리케이션에서 비동기 통신이 UX 향상에 필수적임을 이해하고, 클라이언트와 서버 간의 데이터 교환 흐름을 익혔습니다.

3. 한글 및 이모지(Emoji) 데이터 저장 시 깨짐 현상

- 문제: 투두 리스트의 특성상 사용자가 이모지(예: 🍌, 📝)를 자주 사용하는데, 일반적인 utf8 문자셋 설정으로는 4바이트 문자인 이모지가 제대로 저장되지 않고 물음표(?) 등으로 깨지는 현상이 발생했습니다.
- 해결: 데이터베이스 생성 시 문자셋(Character Set)을 ****utf8mb4****로, 콜레이션(Collation)을 **utf8mb4_general_ci**로 명시적으로 설정하여 한글뿐만 아니라 다양한 이모지까지 완벽하게 지원하도록 데이터베이스 환경을 구성했습니다.
- 배운 점: 데이터베이스 설계 단계에서 서비스의 요구사항(특수문자, 다국어 등)을 고려한 적절한 문자셋 선택의 중요성을 배웠습니다.