인터넷 프로그래밍 기말 프로젝트 보고서

김은규 (12224414)

**- 프로그램 구조**

**메인 화면**

1. 최근 작성한 3개의 메모 확인 가능.
2. 메모가 존재하는 카테고리 목록 확인 가능.
3. 찾고 싶은 검색어를 제목이나 내용으로 포함하는 메모 리스트 확인 가능.
4. 메모 관리와 카테고리 관리 화면으로 이동 가능.

**메모 관리 화면**

1. 메모 번호 / 메모 제목 / 배경색 / 수정, 삭제 / 중요 여부 / 작성일의 순서로 테이블 형식으로 저장된 메모 리스트 확인 가능.
2. 메모 제목을 클릭할 경우 메모의 상세 정보를 확인할 수 있는 창으로 이동.
3. 수정을 클릭할 경우 메모를 수정할 수 있는 창으로 이동.
4. 삭제를 클릭할 경우 메모가 바로 삭제되며, 메모 번호는 자동으로 갱신.
5. 메인 화면을 클릭할 경우 메인 화면으로 이동되고, 새 메모 작성하기를 클릭할 경우 새로운 메모를 작성할 수 있는 창으로 이동.

**새 메모 작성 화면**

1. 제목, 카테고리, 내용을 입력할 수 있으며 배경색, 첨부 파일, 중요 여부 역시 선택이 가능하다. (이때, 제목, 카테고리, 내용은 반드시 입력되어야 하며 입력하지 않을 경우 경고창이 뜬다.)
2. 등록 버튼을 누를 경우 “등록되었습니다.”라는 창이 뜨면서 메모 관리 화면으로 이동되며, 등록한 메모를 확인할 수 있다.
3. 메인 화면을 클릭할 경우 메인 화면으로 이동.

**메모 수정 화면**

1. 수정할 메모의 기존 제목, 카테고리, 내용, 배경색, 첨부 파일, 중요 여부가 반영된 상태로 출력되며, 이때 카테고리는 수정할 수 없다.
2. 수정 버튼을 누를 경우 “수정되었습니다”라는 창이 뜨면서 메모 관리 화면으로 이동되며, 등록하 메모를 확인할 수 있다.
3. 메인 화면을 클릭할 경우 메인 화면으로 이동.

**메모 상세 정보 화면**

1. 메모 관리 화면에서 메모 제목을 클릭할 경우 이동되는 페이지이다. 메모 제목, 번호, 배경색, 중요 여부, 작성일, 내용, 첨부 이미지가 출력 된다. (이때 지정한 배경색으로 메모가 설정된다.)
2. 메모 목록을 클릭할 경우 메모 관리 화면으로 이동.

**카테고리 관리 화면**

1. 카테고리 관리를 클릭할 경우 이동되는 페이지이다. 저장된 카테고리들과 각 메모의 수가 출력.
2. 수정을 클릭할 경우 카테고리 수정 페이지로 이동.
3. 삭제를 클릭할 경우 정말 삭제할 것인지 물어보는 경고 창이 뜨며, 삭제된 것을 확인.
4. 카테고리명을 클릭할 경우 해당 카테고리에 저장된 메모들의 리스트로 이동.
5. 카테고리 등록하기를 클릭할 경우 카테고리 등록 페이지로 이동.
6. 메인 화면으로 돌아가기를 클릭할 경우 메인 화면으로 이동.

**카테고리 수정 화면**

1. 기존 카테고리 명이 출력되고, 새 카테고리 명을 입력할 수 있는 페이지.
2. 수정하기를 클릭할 경우 기존 카테고리 명이 변경.
3. 카테고리 목록으로 돌아가기를 클릭할 경우 카테고리 목록으로 이동. (수정된 카테고리 확인 가능)

**카테고리 등록 화면**

1. 새로운 카테고리를 등록할 수 있는 페이지.
2. 카테고리 명을 입력하고 등록하기를 누를 경우 자동으로 카테고리 관리 페이지로 이동되며, 등록된 카테고리를 확인할 수 있다.
3. 이미 존재하는 카테고리를 입력할 경우 이미 존재하는 카테고리 이름이라는 메시지가 뜬다.
4. 카테고리명을 빈칸으로 놔둘 경우 작성하라는 메시지가 뜬다.
5. 카테고리 목록으로 돌아가기를 클릭할 경우 카테고리 목록으로 이동.

**카테고리 메모 리스트 화면**

1. 카테고리명을 클릭할 경우 이동되는 페이지.
2. 해당 카테고리에 저장된 메모 리스트들이 출력 된다. (메모 관리 페이지와 동일)
3. 새 메모 작성하기 -> 메모 작성 페이지, 카테고리 목록 -> 카테고리 관리 페이지, 메인 화면 -> 메인 페이지로 이동.

**검색 화면**

1. 메모 제목이나 메모 내용에 검색어를 포함하고 있는 메모들을 출력 (메모 관리 페이지와 동일)
2. 메모 관리 -> 메모 관리 페이지, 메인 화면 -> 메인 페이지 이동.

**- 프로그램 파일**

**1. index.jsp (메인 화면)**

- 먼저, 중앙 부분에서 최근 작성한 3개의 메모를 간략하게 확인할 수 있다. 이 3개의 메모는 제목, 카테고리, 내용, 작성일, 메모번호, 중요 여부(별 표시)를 간략하게 확인할 수 있다.

동작 방식: 세션에 저장된 메모 배열과 저장된 메모의 개수를 가져온 후 메모 개수 – 1 (가장 최근메모)부터 3개의 메모의 제목, 카테고리, 내용, 작성일, 메모번호, 중요 여부 등을 출력한다.

- 우측 상단의 메모 관리를 클릭하게 되면, 메모를 관리할 수 있는 창으로 이동하게 된다.

- 상단 부분에서는 검색어를 입력하여 해당 검색어를 포함하고 있는 메모 제목이나 메모 내용을 가진 메모 리스트를 확인할 수 있는 페이지로 이동한다.

- 좌측 부분에서는 현재 메모가 저장된 카테고리들과 각각 저장된 메모의 수를 확인할 수 있고,

동작 방식: 카테고리를 저장할 배열을 만들고, 세션에서 가져온 메모 배열에서 카테고리가 저장된 부분을 저장한다. 이때, 변수를 선언하여 동일한 카테고리가 존재하는지 판단한다. 이미 동일한 카테고리가 존재할 경우 카테고리 수를 1 증가시켜주고, 그렇지 않은 경우 카테고리 배열에 새로운 카테고리를 추가한다.

- 카테고리 관리를 클릭하여 카테고리 관리 페이지로 이동할 수 있다.

텍스트, 스크린샷, 소프트웨어, 컴퓨터 아이콘이(가) 표시된 사진

AI 생성 콘텐츠는 정확하지 않을 수 있습니다.

**2. MemoManage.jsp (메모 관리 화면)**

- 현재 저장되어 있는 메모들의 리스트를 보여준다. 메모 번호, 제목, 배경색, 중요 여부, 작성일 등을 확인할 수 있다.

동작 방식: 세션에서 메모 배열과 개수를 가져온 후 (없는 경우 생성 후, 저장) getParamter 함수를 통해 메모의 구성 요소들을 각각 가져온다. 그리고 가져온 배열에 메모의 구성 요소들을 저장한다. (MemoCount번째에 저장) 이후, 세션에 메모 배열과 개수를 저장한다.

- 수정 및 삭제도 가능하며 메모 제목을 클릭할 경우 해당 메모의 세부 정보를 보여주는 페이지로 이동한다.

- 메인 화면으로 이동이 가능하고, 새 메모 작성을 클릭하여 새로운 메모를 작성하는 페이지로도 이동이 가능하다.

텍스트, 스크린샷, 소프트웨어, 컴퓨터 아이콘이(가) 표시된 사진

AI 생성 콘텐츠는 정확하지 않을 수 있습니다.

**3. MemoWrite.jsp (메모 작성 화면)**

- 제목, 카테고리, 내용을 입력할 수 있고, 배경색, 첨부 파일, 중요 여부를 선택할 수 있다. 이때, 제목과 카테고리, 내용은 반드시 입력되어야 한다. (경고 창 발생)

동작 방식: input을 이용하여 사용자가 작성할 수 있는 부분을 만들어 준다. 이때, 제목과 카테고리, 내용이 작성되었는지 확인하기 위한 함수를 하나 만들어 준 후, form에서 값을 전송할 때 이 함수를 실행하도록 하여 제목과 카테고리, 내용이 반드시 작성되도록 한다.

등록 버튼을 눌러서 새로운 메모를 등록할 수 있고 메인 화면으로 이동이 가능하다.

텍스트, 스크린샷, 소프트웨어, 컴퓨터 아이콘이(가) 표시된 사진

AI 생성 콘텐츠는 정확하지 않을 수 있습니다.텍스트, 스크린샷, 소프트웨어, 컴퓨터 아이콘이(가) 표시된 사진

AI 생성 콘텐츠는 정확하지 않을 수 있습니다.

텍스트, 스크린샷, 소프트웨어, 멀티미디어 소프트웨어이(가) 표시된 사진

AI 생성 콘텐츠는 정확하지 않을 수 있습니다.텍스트, 스크린샷, 소프트웨어, 멀티미디어 소프트웨어이(가) 표시된 사진

AI 생성 콘텐츠는 정확하지 않을 수 있습니다.

**4. MemoEdit.jsp (메모 수정 화면)**

메모 작성 페이지와 동일, 기존의 수정하기 전 메모 상세 정보가 그대로 나타난다. (카테고리는 수정 불가)

텍스트, 스크린샷, 소프트웨어, 컴퓨터 아이콘이(가) 표시된 사진

AI 생성 콘텐츠는 정확하지 않을 수 있습니다.텍스트, 스크린샷, 소프트웨어, 컴퓨터 아이콘이(가) 표시된 사진

AI 생성 콘텐츠는 정확하지 않을 수 있습니다.동작 방식: 세션에 저장된 메모 배열과 개수를 가져온다. (없는 경우 예외 처리) 그 후, memoId를 getParameter를 통해 가져온다. 이 memoId는 수정될 메모 번호 – 1의 값을 저장하고 있다. (수정할 메모의 index) 가져온 메모 배열에서 기존의 저장된 메모 구성요소들을 꺼내서 변수에 각각 저장 한다. (출력하기 위함) 그 후, 새로 작성된 메모의 구성요소들을 memoId 인덱스의 메모 배열에 저장한 후, 세션에 저장 한다. (이때, 카테고리는 수정되지 않기 위해 readonly도 막아둔다.)

**5. MemoView.jsp (메모 상세 정보 보기 화면)**

메모 제목을 클릭할 경우 나타나는 페이지. 해당되는 메모의 상세 정보 내용을 확인할 수 있다.

스크린샷, 텍스트, 소프트웨어, 멀티미디어 소프트웨어이(가) 표시된 사진

AI 생성 콘텐츠는 정확하지 않을 수 있습니다.동작 방식: memoId (상세 정보를 확인할 인덱스)와 메모 배열, 메모의 개수를 가져온 후, 저장된 메모의 구성요소들을 모두 출력한다. 첨부 파일은 이미지로 출력하기 위해 저장된 디렉토리에서 파일을 가져오는 식으로 구현했다.

**6. MemoDelete.jsp (메모 삭제, 별도 페이지 X)**

삭제를 클릭할 경우 해당 메모를 삭제

동작 방식: memoId, 메모 배열과 개수를 가져온다. 삭제를 위해 저장된 배열을 앞으로 한 칸씩 당겨서 memoId - 1번째에 저장된 메모를 삭제하도록 했다. 이렇게 구현하면 마지막에 저장된 메모가 2개 존재하게 되므로 오류 방지를 위해 마지막 메모를 null로 세팅하고 개수를 1 감소시켰다. 그 후, 삭제한 배열과 메모 개수를 세션에 그대로 저장해서 반영되도록 했다.

텍스트, 스크린샷, 소프트웨어, 컴퓨터 아이콘이(가) 표시된 사진

AI 생성 콘텐츠는 정확하지 않을 수 있습니다.

**7. CategoryManage.jsp (카테고리 관리 화면)**

텍스트, 스크린샷, 소프트웨어, 멀티미디어 소프트웨어이(가) 표시된 사진

AI 생성 콘텐츠는 정확하지 않을 수 있습니다.저장된 카테고리들 (메모가 없는 카테고리 포함) 모두 확인 가능. 카테고리를 등록하고, 수정하고 삭제도 할 수 있다.

동작 방식: 카테고리 배열, 개수와 메모 배열, 개수를 세션으로부터 가져온다. 카테고리가 새로 만들어진 카테고리인지 판단하기 위해 변수를 선언해 카테고리 배열을 순회했다. 이때 존재하지 않는 카테고리인 경우 카테고리 배열에 새로 추가시켜 세션에 저장하도록 했다. 그 후, 메모 배열에 저장된 요소들 중 카테고리가 저장된 부분을 모두 출력하도록 설계했다.

**8. CategoryEdit.jsp (카테고리 수정 화면)**

해당되는 카테고리의 이름을 변경할 수 있다.

스크린샷, 텍스트, 소프트웨어, 멀티미디어 소프트웨어이(가) 표시된 사진

AI 생성 콘텐츠는 정확하지 않을 수 있습니다.텍스트, 스크린샷, 소프트웨어, 멀티미디어 소프트웨어이(가) 표시된 사진

AI 생성 콘텐츠는 정확하지 않을 수 있습니다.텍스트, 스크린샷, 소프트웨어, 멀티미디어 소프트웨어이(가) 표시된 사진

AI 생성 콘텐츠는 정확하지 않을 수 있습니다.동작 방식: 카테고리, 메모 배열과 개수를 세션에서 가져온 후, 기존 카테고리와 등록할 카테고리를 getParameter를 통해 모두 가져온다. 카테고리가 존재하는 지 검사한 후, 카테고리가 존재할 경우 message 변수를 통해 등록할 수 없다고 메시지를 남긴다. 만약 존재하지 않을 경우 메모 배열에 새로운 카테고리를 저장하고 세션에 그대로 저장한다.

**9. CategoryAdd.jsp (카테고리 등록 화면)**

새로운 카테고리를 등록할 수 있다. 이미 존재하는 카테고리는 등록할 수 없다.

텍스트, 스크린샷, 소프트웨어, 멀티미디어 소프트웨어이(가) 표시된 사진

AI 생성 콘텐츠는 정확하지 않을 수 있습니다.동작 방식: 카테고리 배열과 개수를 세션에서 가져온 후, 등록할 카테고리가 이미 존재하는 지 확인하기 위해 카테고리 배열을 순회했다. 존재하는 경우 message 변수를 통해 저장할 수 없다는 것을 말해 주었고, 존재하지 않는 카테고리인 경우 카테고리 배열에 추가하고 세션에 저장하도록 했다.

스크린샷, 텍스트, 소프트웨어, 멀티미디어 소프트웨어이(가) 표시된 사진

AI 생성 콘텐츠는 정확하지 않을 수 있습니다.

텍스트, 스크린샷, 소프트웨어, 멀티미디어 소프트웨어이(가) 표시된 사진

AI 생성 콘텐츠는 정확하지 않을 수 있습니다.

**10. CategoryDelete.jsp (카테고리 삭제, 별도 페이지 X)**

존재하는 카테고리를 삭제할 수 있다.

동작 방식: 카테고리 배열과 개수를 세션에서 가져온 후, 저장된 메모도 삭제해야 하기 때문에 메모 배열과 개수도 가져온다. 삭제할 카테고리를 탐색한 후, 앞으로 한 칸씩 당기고 마지막 인덱스를 null로 세팅한다. (메모 배열 역시 동일하게 수행) 세션에 삭제된 배열들과 개수들을 저장한다.

텍스트, 스크린샷, 소프트웨어, 멀티미디어 소프트웨어이(가) 표시된 사진

AI 생성 콘텐츠는 정확하지 않을 수 있습니다.

**11. CategoryMemoList.jsp (카테고리 메모 리스트 페이지)**

텍스트, 스크린샷, 소프트웨어, 컴퓨터 아이콘이(가) 표시된 사진

AI 생성 콘텐츠는 정확하지 않을 수 있습니다.해당 카테고리로 저장된 메모 리스트들을 확인할 수 있다. (카테고리 명을 클릭할 경우 이동) 메모 관리 페이지와 유사하게 동작한다.

동작 방식: 메모 배열을 세션에서 가져온 후, 클릭된 카테고리를 변수에 저장한다. 메모 배열에 저장된 배열 중, 클릭된 카테고리와 동일한 카테고리를 가지는 메모들을 메모리스트 형식과 동일하게 출력한다.

**12. MemoSearch.jsp (검색 페이지)**

검색어를 포함하는 메모 제목이나 메모 내용을 가진 메모들을 출력한다. (이때, 대소문자 관계없이 검색할 수 있다.) 메모 관리 페이지와 유사하게 작동한다.

동작 방식: 세션에서 메모 배열과 개수를 각각 가져 온후, 검색어를 변수에 저장한다. 이때, 대소문자 구분 없이 검색할 수 있도록 하기 위해서 toLowerCase라는 함수를 사용해서 소문자로 강제 변경시켰다. 그리고 가져온 메모 배열을 순회하면서 내용과 제목 중 검색어를 포함하는 메모가 있는지 탐색하였다. 역시 대소문자 구분 없이 탐색하기 위해 toLowerCase라는 함수를 사용하였고, contains라는 함수를 통해 검색어를 포함하는지 확인하였다.

텍스트, 스크린샷, 소프트웨어, 컴퓨터 아이콘이(가) 표시된 사진

AI 생성 콘텐츠는 정확하지 않을 수 있습니다.

**- 프로젝트 수행 후기**

처음 프로젝트 작업을 시작했을 때는 머리로 어떤 식으로 설계할지에 대해서 생각해 놓은 게 있었기 때문에, 진행도중 어려운 코드만 조금 참고해서 그 계획대로만 따라가면 금방 완성할 수 있을 것이라고 생각 했다. 그러나 실제로 프로젝트 작업을 시작해보니 단순히 웹 페이지를 디자인하는 것조차 쉽지 않게 느껴졌고, 원하는 형태로 레이아웃을 구성하기 위해서 강의 노트를 뒤져보고, 관련 내용을 인터넷, AI 등을 통해서 검색한 결과를 참조해 내가 생각한 디자인에 맞게 고쳐 나아가며 하나하나 구현하는 데에 많은 시간이 걸렸다. 프론트 엔드 개발이라는 분야가 그저 버튼 몇 개 배치하고 색상만 정하면 되는 줄 알았는데, 실제로 해보니 웹 디자이너처럼 사용자 입장에서 구조를 설계하고 세부적인 요소까지 신경 써야 하기 때문에 많은 고민과 노력이 필요하다는 것을 느꼈다.

또한, 프로젝트를 처음부터 끝까지 만들어 본 경험이 없었기 때문에 여러 개의 파일을 동시에 다루면서 흐름을 파악하는 데에도 어려움이 많았다. 특히, 어떤 페이지에서 어떤 데이터를 받아오고, 어떤 페이지로 넘겨줘야 하는지 흐름을 정리하는 데 많은 혼란이 있었다. 변수나 파라미터의 전달, 서버와 클라이언트 간의 상호작용을 설계하는 부분에서 실수가 많았고, 그로 인해 의도치 않은 오류들이 자주 발생했다. 처음 만들어보는 개인 프로젝트다 보니 AI 사용을 최소한으로 하고 싶어서, 최대한 구글링이나 인터넷에서 정보를 얻어서 해결하려고 노력했다. 다행히도, 페이지를 대부분 비슷하게 설계하기도 했고, 페이지들의 동작 방식이 유사한 경우가 많아 코드를 비슷하게 작성하여 어려움을 좀 덜었다.

프로젝트 설계 도중 가장 어려웠던 점은, 스스로 코드를 잘 설계했다고 생각했음에도 불구하고 오류가 발생했을 때 그 원인을 찾고 해결하는 과정이었다. 에러 메시지를 보고도 어떤 부분이 잘못됐는지 감을 잡기 어려웠고, 디버깅에 많은 시간이 걸렸다. 디버깅을 했음에도 도무지 오류의 원인을 차지 못한 경우에는 AI나 인터넷의 도움을 받아 해결 코드를 참조하여 페이지에 스타일에 맞게 고쳐서 사용하였다. 그 과정에서 단순히 코드를 작성하는 능력 뿐만 아니라, 문제를 분석하고 해결하는 논리적인 사고력 또한 중요하다는 것을 깨 달았다.

이번 프로젝트를 통해 웹 개발의 전반적인 흐름을 이해하게 되었고, 단순히 동작하는 코드만 작성하는 것이 아니라, 유지보수와 확장성을 고려한 구조적인 설계가 얼마나 중요한지도 알게 되었다. 처음 만든 프로젝트라 구현하고 싶었지만 능력 밖의 작업인 것 같아서 구현하지 못한 부분도 많았지만, 시행착오를 겪으면서 실력이 조금씩 향상되고 있다는 것을 느꼈고, 다음 프로젝트에서는 보다 효율적으로 개발을 진행할 수 있을 것이라는 자신감도 얻게 되었다. 무엇보다 나 스스로 무언가를 처음 만들어 보아서 너무나 뿌듯한 순간이었다.