정복래 교수님

Dummy Text

# 프로그래밍 설계

S-MART

전수환
김도현
김건우

# INDEX

# INDEX

01

## 팀 소개 및 주제 재소개

# Our Project Name is

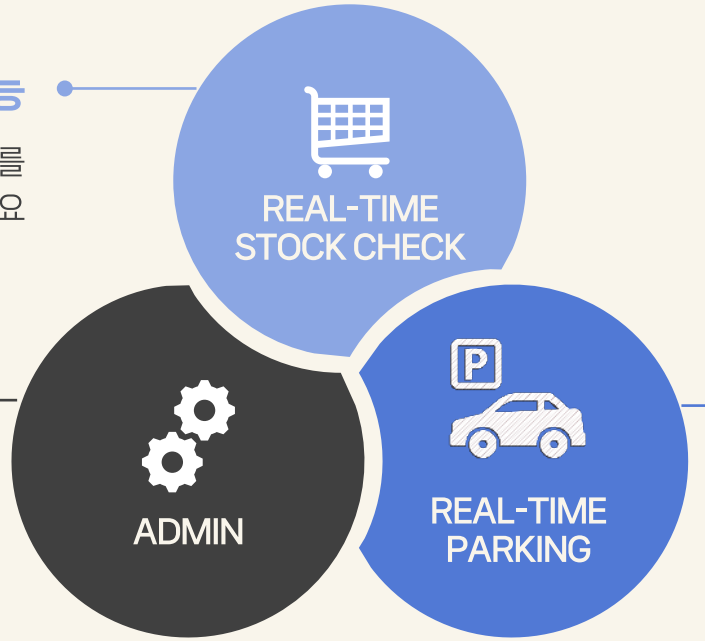
# S-MART

저희 프로젝트의 이름은 S-MART.
똑똑한 마트 어플리케이션이라는 의미와 성결 마트라는 두가지 의미를 가지고 있어요.

# S-MART
# 대표적 기능 3가지

### 실시간 재고 관리 기능

실시간으로 마트 내에 등록 되어있는 마트 재고를
알 수 있어요




### 관리자 모드

실시간 재고 및 주차 관리 정보를
관리자모드로 수정합니다.

### 실시간 주차 확인 기능

실시간으로 마트 내에 주차 되어있는
주차 현황을 알 수 있어요

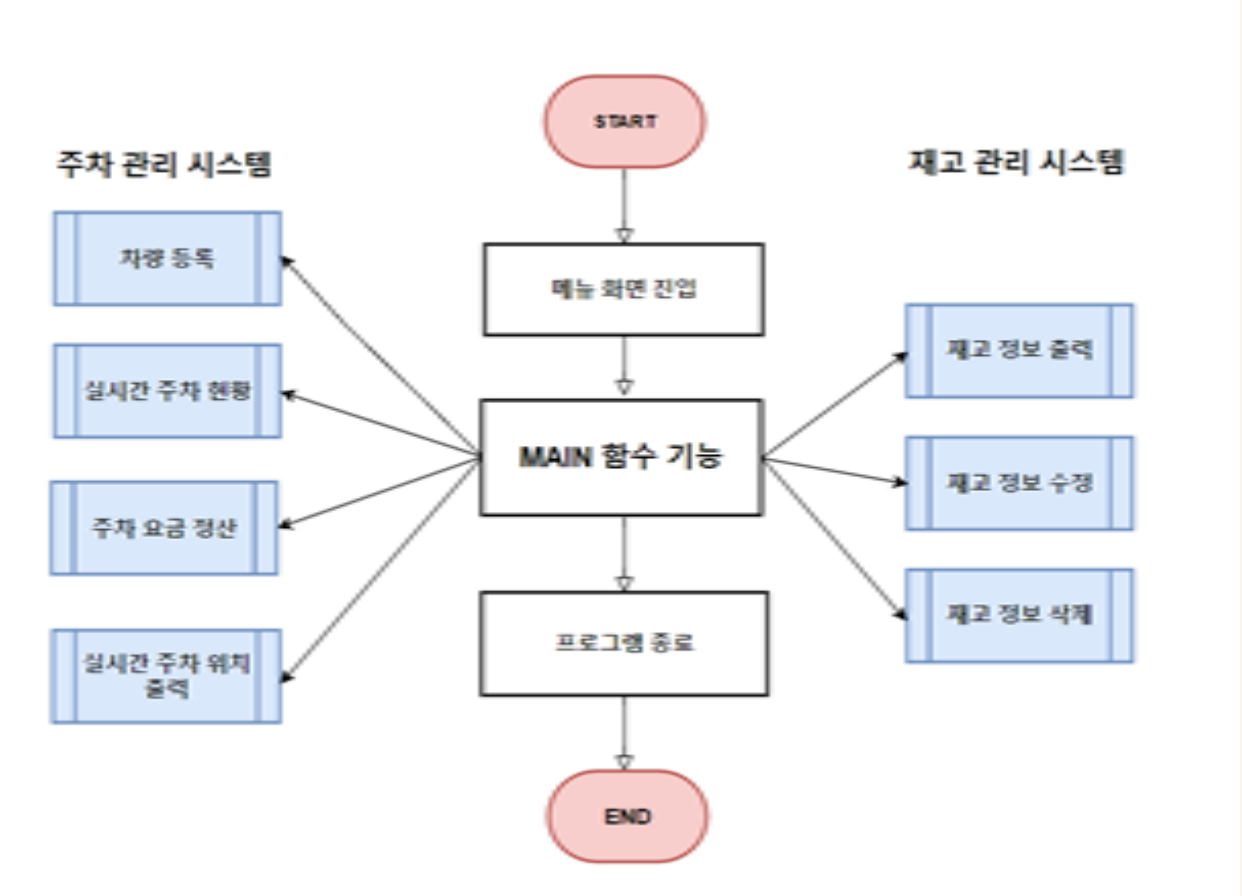
# 저희는 이 세가지를, 종합한
# S-MART 프로그램을 만드려고 합니다.

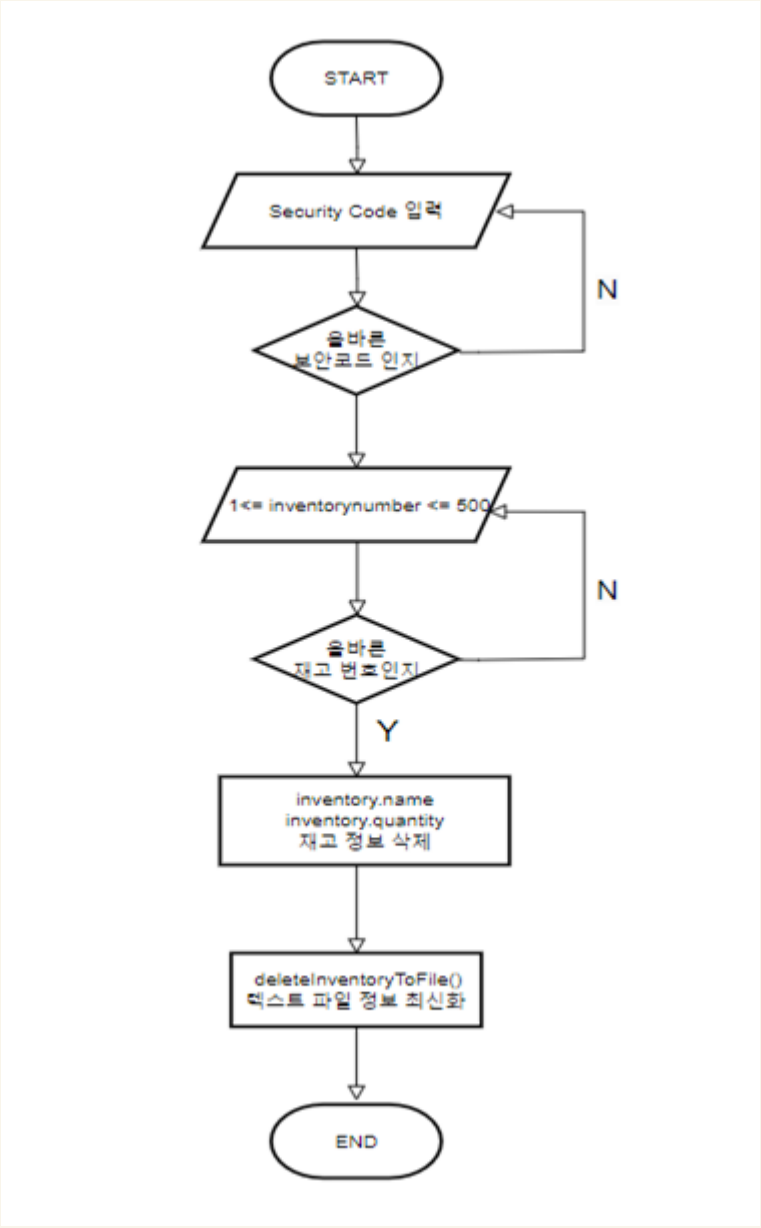

# INDEX

02

# FEEDBACK

# 재고 관리 메인 순서도 및 상세 순서도



# FEEDBACK.

## 재고 삭제 기능 추가

```
#define PASSWORD "SMART031"

void deleteInventory() {
    char password[20];
    printf("관리자 보안 코드를 입력해주세요: ");
    scanf("%s", password);


    if (strcmp(password, PASSWORD) != 0) {
        printf("보안 코드가 일치하지 않습니다 \n");
        return;다
    }

    int number;
    printf("삭제할 재고 번호를 입력하세요: ");
    if (scanf("%d", &number) != 1 || number < 1 || number > 500) {
        printf("올바르지 않은 재고 번호입니다.\n");
        return;
    }

    FILE* file = fopen("inventory.txt", "r");
    if (file == NULL) {
        printf("파일을 열 수 없습니다.\n");
        return;
    }
```

```
int number;
printf("삭제할 재고 번호를 입력하세요: ");
if (scanf("%d", &number) != 1 || number < 1 || number > 500) {
        printf("올바르지 않은 재고 번호입니다.\n");
        return;
}
```

# FEEDBACK.

## 재고 삭제 기능 추가

```
    // 복사 할 파일 생성 (삭제 후 데이터)
    FILE* tempFile = fopen("temp_inventory.txt", "w");
    if (tempFile == NULL) {
        printf("임시 파일을 열 수 없습니다.\n");
        fclose(file);
        return;
    }
성
    int found = 0;
    while (fscanf(file, "%d %s %s %d", &inventory[numInventoryItems].number,
inventory[numInventoryItems].category, inventory[numInventoryItems].name,
&inventory[numInventoryItems].quantity) != EOF) {
        if (inventory[numInventoryItems].number == number) {
            found = 1;
            continue;
        }


        fprintf(tempFile, "%d %s %s %d\n", inventory[numInventoryItems].number,
inventory[numInventoryItems].category, inventory[numInventoryItems].name,
inventory[numInventoryItems].quantity);
    }

    fclose(file);
    fclose(tempFile);


    remove("inventory.txt");
    rename("temp_inventory.txt", "inventory.txt");

}
```

```
FILE* tempFile = fopen("temp_inventory.txt",
"w");if (tempFile == NULL) {
        printf("임시 파일을 열 수 없습니다.\n");
        fclose(file);
        return;
    }


    remove("inventory.txt");
    rename("temp_inventory.txt", "inventory.txt");

}
```

# INDEX

# 03

## 코딩 진행상황

# 코딩 진행상황 및 결과

## -주차 등록

```
    //주차 등록
{
    int Floor, Space, i;
    int Result, CarNum, Num;
    time_t Time;

    Result = CarParking_Find(Car);

    if (Result == 1)
    {
        puts("주차할 공간이 없습니다.");
        getch();
        return 0;
    }

    printf("\n\n주차 층 입력: ");
    scanf("%d", &Stair);

    if (Floor > 10 || Floor == 0)
    {
        puts("층은 1층부터 10 층까지 입니다.");
        getch();
        return 0;
    }

    printf("주차 자리 입력: ");
    scanf("%d", &Space);

    if (Space > 20 || Space == 0)
    {
        puts("자리는 1부 부터 20번까지 입니다.");
        getch();
        return 0;
    }

    if (Car[Floor - 1][Space - 1] != NULL)
    {
        puts("\n주차 자리에 있습니다 다시 확인하세요!!");
        getch();
        return 0;
    }

    Car[Floor - 1][Space - 1] = (CARPARK*)malloc(sizeof(CARPARK));

    printf("\n사원 이름: ");
    scanf("%s", Car[Stair - 1][Space - 1]->CarName);

    printf("차량 번호: ");
    scanf("%d", &CarNum);

    Num = Car_Num(Car, &CarNum);

    if (Num == 1)
    {
        return 0;
    }

    Car[Floor - 1][Space - 1]->CarNum = CarNum;

    CarNum = time(NULL);

    Car[Floor - 1][Space - 1]->ParkingTime = time(&Time);

    for (i = 0; i < 200; i++)
    {
        if (Record[i] == NULL)
        {
            Record[i] = (CARRECORD*)malloc(sizeof(CARRECORD));
            strcpy(Record[i]->CarName, Car[Stair - 1][Space - 1]
            Record[i]->CarNum = Car[Floor - 1][Space - 1]->CarNum;
            Record[i]->Parking_T = Car[Floor - 1][Space - 1]->ParkingTi
            Record[i]->Floor = Floor;
            Record[i]->Space = Space;
            Record[i]->CarMoney = 0;
            puts("주차되었습니다.");
            getch();
            return Record[i];
        }
    }

    return 0;
}
```

```
Car[Floor - 1][Space - 1] = (CARPARK*)malloc(sizeof(CARPARK));

printf("\n사원 이름: ");
scanf("%s", Car[Stair - 1][Space - 1]->CarName);

printf("차량 번호: ");
scanf("%d", &CarNum);

Num = Car_Num(Car, &CarNum);

if (Num == 1)
{
    return 0;
}

Car[Floor - 1][Space - 1]->CarNum = CarNum;

CarNum = time(NULL);

Car[Floor - 1][Space - 1]->ParkingTime = time(&Time);

for (i = 0; i < 200; i++)
{
    if (Record[i] == NULL)
    {
        Record[i] = (CARRECORD*)malloc(sizeof(CARRECORD));
        strcpy(Record[i]->CarName, Car[Stair - 1][Space - 1]->CarName);
        Record[i]->CarNum = Car[Floor - 1][Space - 1]->CarNum;
        Record[i]->Parking_T = Car[Floor - 1][Space - 1]->ParkingTime;
        Record[i]->Floor = Floor;
        Record[i]->Space = Space;
        Record[i]->CarMoney = 0;
        puts("주차되었습니다.");
        getch();
        return Record[i];
    }
}

return 0;
```

## -실행 결과

```
차량 번호 입력: 1111


차 이름 : SMART031
차 번호 : 1111
주차 층 : 5
주차 자리 : 10
```

# 코딩 진행상황 및 결과

## -주차 조회

```c
// 주차 조회
{
    int CarNumber;
    int i, j;

    printf("\n차량 번호 입력: ");
    scanf("%d", &CarNumber);


    for (i = 0; i < 10; i++)
    {
        for (j = 0; j < 20; j++)
        {
            if (Car[i][j] != NULL)
            {
                if (Car[i][j]->CarNumber == CarNumber)
                {
                    printf("\n\n차 이름 : %s\n", Car[i][j]->CarName);
                    printf("차 번호 : %d\n", Car[i][j]->CarNumber);
                    printf("주차 층 : %d\n", i + 1);
                    printf("주차 자리 : %d\n", j + 1);
                    getch();
                    return Car[i][j]; // 찾은 값 배열 주소 반환
                }
            }
        }
    }

    puts("차량이 없습니다.");
    getch();

    return NULL; // 차가 없다면 널값 반환
}
```

```
차량 번호 입력: 1111


차 이름 : SMART031
차 번호 : 1111
주차 층 : 5
주차 자리 : 10
```

INDEX

?!

# 04

# 코드 작성 중 문제상황 및 해결방법

# 코드 작성 중
# 문제상황 및 해결방법

**4**

재고 삭제 부분에서 하나의 텍스트파일에서 재고를 삭제하는 것보다 임시파일을 생성 후, 기존 파일에서 삭제하고 싶은 재고의 정보를 제거 한 뒤, 임시파일에 복사하여 임시파일을 기존파일로 대체 하는 방법

| | |
|---|---|
| **성능 이점** | 더 큰 버퍼를 사용하여 I/O(input/output작업)의 성능을 향상 시켰습니다. |
| **일관성 유지** | 최종 작업 결과만 원본 파일에 반영되어 파일의 일관성이 보장했습니다. |
| **데이터 안전성** | 임시 파일을 사용하면 작업 중 오류가 발생해도 원본 파일이 안전하게 유지하도록 만들었습니다. |
| **코드 유지보수** | 파일 읽기, 처리, 쓰기를 분리하여 코드의 가독성과 유지보수성을 높혔습니다. |

# END