Technical Documentation

Architecture Overview

Core Systems

The game is built with a modular architecture, separating different game systems into distinct components:

- Battle System
- Save System
- Shop System
- Healing System
- Display System
- Item Management

File Structure

```
supemon/
├── src/
│   ├── battle.c
│   ├── center.c
│   ├── display.c
│   ├── save.c
│   ├── supemon.c
│   └── utils.c
├── include/
│   ├── battle.h
│   ├── center.h
│   ├── display.h
│   ├── item.h
│   ├── move.h
│   ├── player.h
│   ├── save.h
│   ├── shop.h
│   ├── supemon.h
│   └── utils.h
└── docs/
    ├── battle.md
    ├── center.md
    ├── items.md
```

```
    ├── project.md
    └── save.md
```

Implementation Details

1. Battle System

Core Battle Functions

```
static void apply_move_effects(Supemon* attacker, Supemon* defender, Move* move) {
    float dodge_chance = (float)defender->evasion / (attacker->accuracy + defender->evasion);
    if ((float)rand() / RAND_MAX < dodge_chance) {
        printf("%s dodged the attack!\\n", defender->name);
        return;
    }

    if (move->damage > 0) {
        int base_damage = move->damage * attacker->attack;
        int damage = base_damage / (defender->defense / 2 + 1);
        if (damage < 1) damage = 1;
        defender->hp -= damage;
        if (defender->hp < 0) defender->hp = 0;
        printf("%s took %d damage!\\n", defender->name, damage);
    }

    if (move->stat_boost > 0) {
        if (strcmp(move->stat_affected, "attack") == 0) {
            attacker->attack += move->stat_boost;
        } else if (strcmp(move->stat_affected, "defense") == 0) {
            attacker->defense += move->stat_boost;
        } else if (strcmp(move->stat_affected, "evasion") == 0) {
            attacker->evasion += move->stat_boost;
        }
        printf("%s's %s increased!\\n", attacker->name, move->stat_affected);
    }
}

static int battle_capture(Battle *battle, Player *player) {
    float chance = (float)(battle->enemy_supemon->max_hp - battle->enemy_supemon->hp) /
                battle->enemy_supemon->max_hp - 0.5f;

    if ((float)rand() / RAND_MAX < chance) {
        int slot = 0;
        while (slot < MAX_SUPEMON && player->supemons[slot][0] != '\\0') {
```

```
            slot++;
        }

        if (slot < MAX_SUPEMON) {
            strncpy(player->supemons[slot],
                battle->enemy_supemon->name,
                sizeof(player->supemons[slot]) - 1);
            printf("%s was caught!\\n", battle->enemy_supemon->name);
            return 1;
        }
    }
    return 0;
}

static void calculate_rewards(Player* player, Battle* battle) {
    int supcoins = 100 + (rand() % 401);
    player->supcoins += supcoins;

    int exp_multiplier = 100 + (rand() % 401);
    int exp_gained = exp_multiplier * battle->enemy_supemon->level;
    battle->player_supemon->exp += exp_gained;
}
```

2. Save System

Save Implementation

```
void save_player(const Player *player) {
    cJSON *root = cJSON_CreateObject();
    if (!root) {
        printf("Error creating JSON object!\\n");
        return;
    }

    FILE *file = fopen("save.json", "r");
    cJSON *json = NULL;
    cJSON *old_json = NULL;

    if (file) {
        fseek(file, 0, SEEK_END);
        long length = ftell(file);
        fseek(file, 0, SEEK_SET);

        char *data = malloc(length + 1);
```

```
        fread(data, 1, length, file);
        fclose(file);
        data[length] = '\\0';

        json = cJSON_Parse(data);
        old_json = cJSON_Parse(data);
        free(data);
    }
    // ...
    // see save.c
}
```

3. Display System

Menu Display Implementation

```
void display_battle_menu(void) {
    printf("+----------------------------+\\n");
    printf("|What will you do?           |\\n");
    printf("| 1 - Move                   |\\n");
    printf("| 2 - Change Supemon         |\\n");
    printf("| 3 - Use item               |\\n");
    printf("| 4 - Capture                |\\n");
    printf("| 5 - Run away               |\\n");
    printf("+----------------------------+\\n");
    printf("1, 2, 3, 4 or 5: ");
}

void display_battle_status(Battle *battle) {
    printf("\\nYour turn...\\n\\n");
    printf("%s (enemy)\\n", battle->enemy_supemon->name);
    printf("--------------------------------\\n");
    printf("HP: %d/%d\\tLvl: %d\\n",
           battle->enemy_supemon->hp,
           battle->enemy_supemon->max_hp,
           battle->enemy_supemon->level);
    // ...
    // see display.c
}
```

4. Item System

Item Structure

```
typedef struct {
    char name[50];
    int type;
    int price;
    int quantity;
    char description[100];
    int effect_value;
} Item;
```

5. Limitations and Constraints

1. **Game Limits**:
    - Maximum 10 Supemons per player
    - Maximum 10 different items in inventory
    - 4 items per battle
    - 2 moves per Supemon
2. **Technical Limits**:
    - Save file size dependent on number of Supemons and items
    - Console display limited by terminal size
    - Integer-based calculations for performance