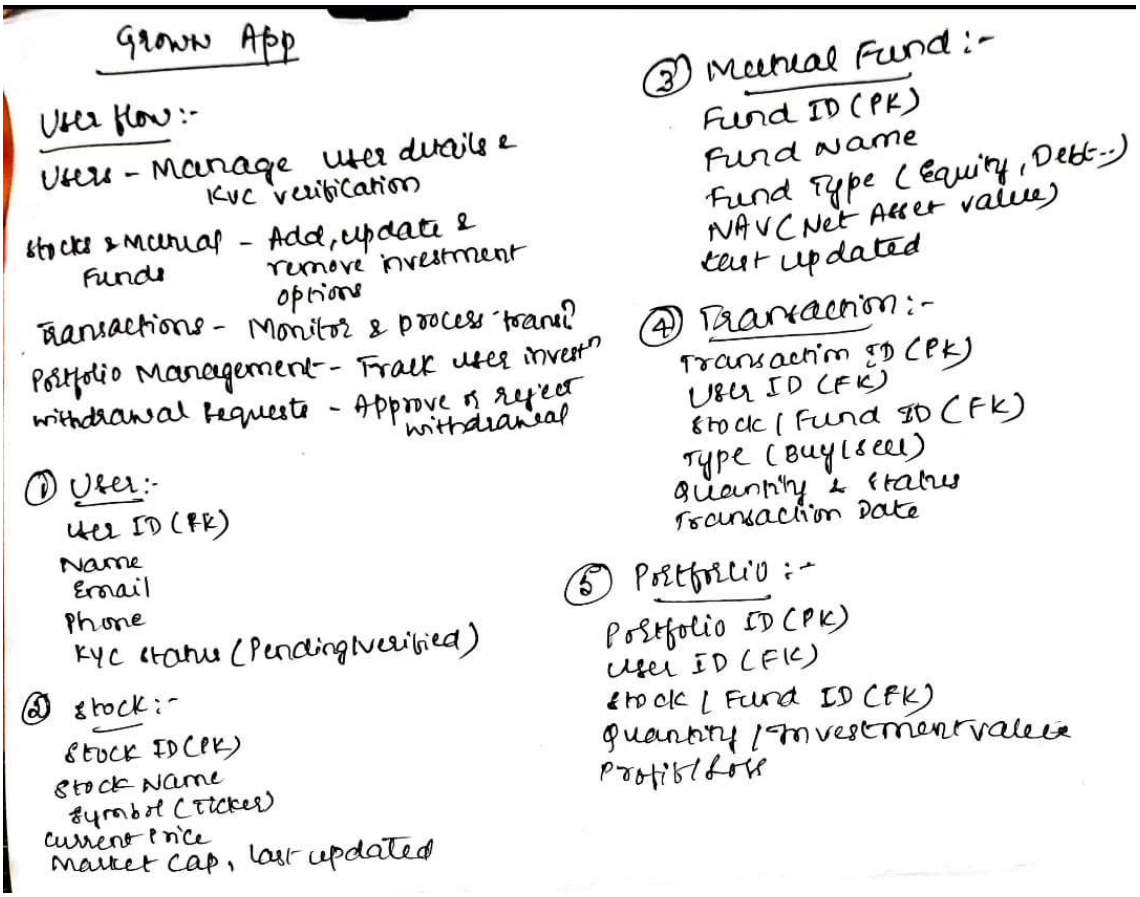
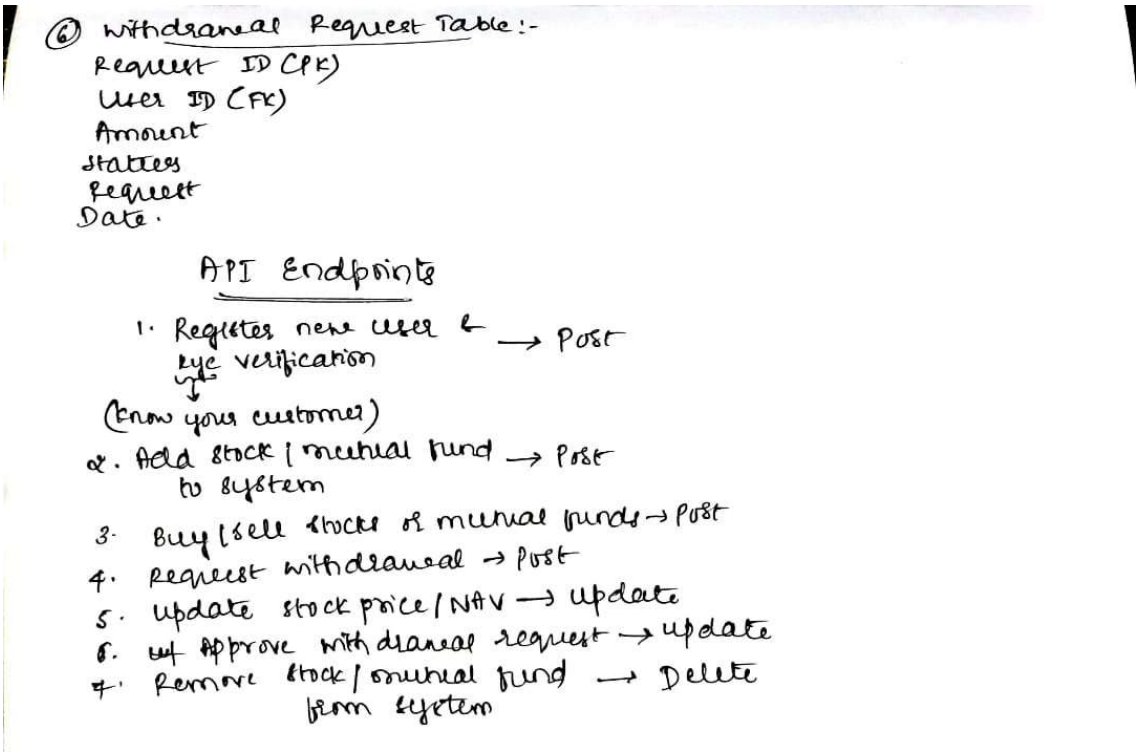
Assesment-6(Saturday)

Groww-application:

Overview:



Assesment-6(Saturday)



Scalable Architecture for Grow Application:

Overview:

The Grow application is a financial platform for stock and mutual fund investments. This architecture ensures scalability, high availability, security, and performance optimization.

Cloud-Based Infrastructure:

AWS Services: EC2 Auto Scaling, Lambda, ECS/EKS for containerization.

Load Balancing: AWS ALB for traffic distribution.

CDN: CloudFront for fast content delivery.

Microservices Architecture

API Gateway for routing requests.

Key Services: Authentication (Cognito), Trading, Portfolio Management, Notifications (SNS/SQS), Logging.

Communication: gRPC/REST APIs.

Database Design:

Storage: RDS (PostgreSQL/MySQL), DynamoDB for NoSQL data.

Caching: ElastiCache (Redis/Memcached).

Assesment-6(Saturday)

Analytics: Redshift for queries.

Security Measures

Authentication: OAuth 2.0, JWT (Cognito), IAM roles.

Encryption: AWS KMS, HTTPS/TLS.

DDoS Protection: AWS Shield & WAF.

Performance Optimization:

Auto Scaling for dynamic resource allocation.

Asynchronous Processing: AWS SQS.

Edge Caching: CloudFront.

Database Optimization: Indexing & query tuning.

Monitoring & Logging

Monitoring: CloudWatch, X-Ray for tracing.

Logging: ELK Stack, CloudTrail for audits.

CI/CD Pipeline:

Repo: CodeCommit/GitHub.

CI/CD: CodePipeline, CodeBuild, CodeDeploy.

Testing: Unit, integration, and performance testing.

Disaster Recovery & Compliance

Backup: Automated snapshots.

Failover: Multi-region deployment with Route 53.

Compliance: GDPR, PCI DSS adherence.

This architecture ensures a secure, scalable, and high-performance application for financial transactions.

Code :

```
#include <stdio.h>

#include <string.h>
```

Assesment-6(Saturday)

```
#define MAX_USERS 100
#define MAX_STOCKS 100
#define MAX_FUNDS 100
#define MAX_TRANSACTIONS 100



typedef struct {
    int userID;
    char name[50];
    char email[50];
    char phone[15];
    int kycStatus;
} User;

typedef struct {
    int stockID;
    char stockName[50];
    char symbol[10];
    float currentPrice;
    float marketCap;
} Stock;

typedef struct {
    int fundID;
    char fundName[50];
    char fundType[20];
    float nav;
} MutualFund;
```

Assesment-6(Saturday)

```
typedef struct {
    int transactionID;
    int userID;
    int stockOrFundID;
    char type[5];
    int quantity;
    char date[15];
} Transaction;



User users[MAX_USERS];
Stock stocks[MAX_STOCKS];
MutualFund funds[MAX_FUNDS];
Transaction transactions[MAX_TRANSACTIONS];

int userCount = 0, stockCount = 0, fundCount = 0, transactionCount = 0;



void registerUser();
void addStock();
void addMutualFund();
void buySellTransaction();
void requestWithdrawal();
void displayUsers();
void displayStocks();

int main() {
    int choice;
    do {
```

Assesment-6(Saturday)

```
        printf("\n1. Register User\n2. Add Stock\n3. Add Mutual Fund\n4. Buy/Sell
Transaction\n5. Request Withdrawal\n6. Display Users\n7. Display Stocks\n8. Exit\n");
        printf("Enter your choice: ");
        scanf("%d", &choice);
        switch (choice) {
            case 1: registerUser(); break;
            case 2: addStock(); break;
            case 3: addMutualFund(); break;
            case 4: buySellTransaction(); break;
            case 5: requestWithdrawal(); break;
            case 6: displayUsers(); break;
            case 7: displayStocks(); break;
            case 8: printf("Exiting...\n"); break;
            default: printf("Invalid choice!\n");
        }
    } while (choice != 8);
    return 0;
}

void registerUser() {
    if (userCount >= MAX_USERS) {
        printf("User limit reached!\n");
        return;
    }
    printf("Enter User ID: ");
    scanf("%d", &users[userCount].userID);
    printf("Enter Name: ");
    scanf("%s", users[userCount].name);
    printf("Enter Email: ");
    scanf("%s", users[userCount].email);
```

```
    printf("Enter Phone: ");
    scanf("%s", users[userCount].phone);
    printf("Enter KYC Status (1 for Verified, 0 for Pending): ");
    scanf("%d", &users[userCount].kycStatus);
    userCount++;
    printf("User registered successfully!\n");
}

void addStock() {
    if (stockCount >= MAX_STOCKS) {
        printf("Stock limit reached!\n");
        return;
    }
    printf("Enter Stock ID: ");
    scanf("%d", &stocks[stockCount].stockID);
    printf("Enter Stock Name: ");
    scanf("%s", stocks[stockCount].stockName);
    printf("Enter Symbol: ");
    scanf("%s", stocks[stockCount].symbol);
    printf("Enter Current Price: ");
    scanf("%f", &stocks[stockCount].currentPrice);
    printf("Enter Market Cap: ");
    scanf("%f", &stocks[stockCount].marketCap);
    stockCount++;
    printf("Stock added successfully!\n");
}

void addMutualFund() {
    if (fundCount >= MAX_FUNDS) {
```

```
        printf("Mutual fund limit reached!\n");
        return;
    }
    printf("Enter Fund ID: ");
    scanf("%d", &funds[fundCount].fundID);
    printf("Enter Fund Name: ");
    scanf("%s", funds[fundCount].fundName);
    printf("Enter Fund Type: ");
    scanf("%s", funds[fundCount].fundType);
    printf("Enter NAV: ");
    scanf("%f", &funds[fundCount].nav);
    fundCount++;
    printf("Mutual Fund added successfully!\n");
}

void buySellTransaction() {
    if (transactionCount >= MAX_TRANSACTIONS) {
        printf("Transaction limit reached!\n");
        return;
    }
    printf("Enter Transaction ID: ");
    scanf("%d", &transactions[transactionCount].transactionID);
    printf("Enter User ID: ");
    scanf("%d", &transactions[transactionCount].userID);
    printf("Enter Stock/Fund ID: ");
    scanf("%d", &transactions[transactionCount].stockOrFundID);
    printf("Enter Type (Buy/Sell): ");
    scanf("%s", transactions[transactionCount].type);
    printf("Enter Quantity: ");
```

```
    scanf("%d", &transactions[transactionCount].quantity);
    printf("Enter Date: ");
    scanf("%s", transactions[transactionCount].date);
    transactionCount++;
    printf("Transaction recorded successfully!\n");
}

void requestWithdrawal() {
    printf("Withdrawal feature coming soon!\n");
}

void displayUsers() {
    for (int i = 0; i < userCount; i++) {
        printf("User ID: %d, Name: %s, Email: %s, Phone: %s, KYC: %d\n",
               users[i].userID, users[i].name, users[i].email, users[i].phone, users[i].kycStatus);
    }
}

void displayStocks() {
    for (int i = 0; i < stockCount; i++) {
        printf("Stock ID: %d, Name: %s, Symbol: %s, Price: %.2f, Market Cap: %.2f\n",
               stocks[i].stockID, stocks[i].stockName, stocks[i].symbol, stocks[i].currentPrice,
stocks[i].marketCap);
    }
}
```

Output:

1. Register User

Assesment-6(Saturday)

2. Add Stock

3. Add Mutual Fund

4. Buy/Sell Transaction

5. Request Withdrawal

6. Display Users

7. Display Stocks

8. Exit

Enter your choice: 1

Enter User ID: 023

Enter Name: jyothi

Enter Email: jyo123@gmail.com

Enter Phone: 4569871230

Enter KYC Status (1 for Verified, 0 for Pending): 1

User registered successfully!

1. Register User

2. Add Stock

3. Add Mutual Fund

4. Buy/Sell Transaction

5. Request Withdrawal

6. Display Users

7. Display Stocks

8. Exit

Enter your choice: 2

Enter Stock ID: 45

Enter Stock Name: infosys

Enter Symbol: inf

Enter Current Price: 1815.15

Enter Market Cap: 13490000

Assesment-6(Saturday)

Stock added successfully!

1. Register User
2. Add Stock
3. Add Mutual Fund
4. Buy/Sell Transaction
5. Request Withdrawal
6. Display Users
7. Display Stocks
8. Exit

Enter your choice:

3

Enter Fund ID: 65

Enter Fund Name: franklin

Enter Fund Type: stc

Enter NAV: 123

Mutual Fund added successfully!

1. Register User
2. Add Stock
3. Add Mutual Fund
4. Buy/Sell Transaction
5. Request Withdrawal
6. Display Users
7. Display Stocks
8. Exit

Enter your choice:

4

Enter Transaction ID: 98

Assesment-6(Saturday)

Enter User ID: 023

Enter Stock/Fund ID: 65

Enter Type (Buy/Sell): buy

Enter Quantity: 109

Enter Date: 12-02-2025

Transaction recorded successfully!

1. Register User

2. Add Stock

3. Add Mutual Fund

4. Buy/Sell Transaction

5. Request Withdrawal

6. Display Users

7. Display Stocks

8. Exit

Enter your choice: 5

Withdrawal feature coming soon!

1. Register User

2. Add Stock

3. Add Mutual Fund

4. Buy/Sell Transaction

5. Request Withdrawal

6. Display Users

7. Display Stocks

8. Exit

Enter your choice: 6

User ID: 23, Name: jyothi, Email: jyo123@gmail.com, Phone: 4569871230, KYC: 1

Assesment-6(Saturday)

1. Register User

2. Add Stock

3. Add Mutual Fund

4. Buy/Sell Transaction

5. Request Withdrawal

6. Display Users

7. Display Stocks

8. Exit

Enter your choice: 7

Stock ID: 45, Name: infosys, Symbol: inf, Price: 15.15, Market Cap: 13490000.00

1. Register User

2. Add Stock

3. Add Mutual Fund

4. Buy/Sell Transaction

5. Request Withdrawal

6. Display Users

7. Display Stocks

8. Exit

Enter your choice: 8

Exiting...

=== Code Execution Successful ===

Mcq's ans:

1.c) To create new stock

2. B) Linked list

Assesment-6(Saturday)

3.B)To update the price of a stock

Programming ques:

1. Stock Market Simulator:

```
#include <stdio.h>
#include <string.h>

#define MAX_STOCKS 5
#define MAX_TRANSACTIONS 100

typedef struct {
    char name[20];
    float price;
    int quantity;
} Stock;

typedef struct {
    char type[5]; // Buy or Sell
    char stockName[20];
    int quantity;
    float totalPrice;
} Transaction;

Stock stocks[MAX_STOCKS] = {
    {"Apple", 150.0, 100},
    {"Google", 2800.0, 50},
```

```
    {"Amazon", 3400.0, 75},
    {"Tesla", 700.0, 80},
    {"Microsoft", 300.0, 120}
};

Transaction transactions[MAX_TRANSACTIONS];
int transactionCount = 0;

void displayStocks() {
    printf("\nAvailable Stocks:\n");
    printf("%-10s %-10s %-10s\n", "Name", "Price", "Quantity");
    for (int i = 0; i < MAX_STOCKS; i++) {
        printf("%-10s $%-9.2f %-10d\n", stocks[i].name, stocks[i].price,
stocks[i].quantity);
    }
}

void buyStock() {
    char stockName[20];
    int quantity;
    printf("\nEnter stock name to buy: ");
    scanf("%s", stockName);
    printf("Enter quantity: ");
    scanf("%d", &quantity);

    for (int i = 0; i < MAX_STOCKS; i++) {
        if (strcmp(stocks[i].name, stockName) == 0) {
```

```
            if (stocks[i].quantity >= quantity) {
                float totalPrice = stocks[i].price * quantity;
                stocks[i].quantity -= quantity;
                strcpy(transactions[transactionCount].type, "Buy");
                strcpy(transactions[transactionCount].stockName, stockName);
                transactions[transactionCount].quantity = quantity;
                transactions[transactionCount].totalPrice = totalPrice;
                transactionCount++;
                printf("Purchase successful! Total cost: $%.2f\n", totalPrice);
            } else {
                printf("Not enough stock available!\n");
            }
            return;
        }
    }
    printf("Stock not found!\n");
}

void sellStock() {
    char stockName[20];
    int quantity;
    printf("\nEnter stock name to sell: ");
    scanf("%s", stockName);
    printf("Enter quantity: ");
    scanf("%d", &quantity);
```

Assesment-6(Saturday)

```
    for (int i = 0; i < MAX_STOCKS; i++) {
        if (strcmp(stocks[i].name, stockName) == 0) {
            stocks[i].quantity += quantity;
            float totalPrice = stocks[i].price * quantity;
            strcpy(transactions[transactionCount].type, "Sell");
            strcpy(transactions[transactionCount].stockName, stockName);
            transactions[transactionCount].quantity = quantity;
            transactions[transactionCount].totalPrice = totalPrice;
            transactionCount++;
            printf("Sale successful! You earned: $%.2f\n", totalPrice);
            return;
        }
    }
    printf("Stock not found!\n");
}

void displayTransactions() {
    printf("\nTransaction History:\n");
    printf("%-5s %-10s %-10s %-10s\n", "Type", "Stock", "Quantity", "Total
Price");
    for (int i = 0; i < transactionCount; i++) {
        printf("%-5s %-10s %-10d $%-9.2f\n", transactions[i].type,
transactions[i].stockName, transactions[i].quantity, transactions[i].totalPrice);
    }
}

int main() {
```

Assesment-6(Saturday)

```
int choice;
do {
    printf("\nStock Market Simulator\n");
    printf("1. Display Stocks\n");
    printf("2. Buy Stock\n");
    printf("3. Sell Stock\n");
    printf("4. Display Transactions\n");
    printf("5. Exit\n");
    printf("Enter your choice: ");
    scanf("%d", &choice);

    switch (choice) {
        case 1:
            displayStocks();
            break;
        case 2:
            buyStock();
            break;
        case 3:
            sellStock();
            break;
        case 4:
            displayTransactions();
            break;
        case 5:
            printf("Exiting program.\n");
```

```
            break;
        default:
            printf("Invalid choice! Please try again.\n");
    }
  } while (choice != 5);

  return 0;
}
```

Output:

Stock Market Simulator

1. Display Stocks

2. Buy Stock

3. Sell Stock

4. Display Transactions

5. Exit

Enter your choice: 1

Available Stocks:

Name	Price	Quantity
Apple	$150.00	100
Google	$2800.00	50
Amazon	$3400.00	75
Tesla	$700.00	80
Microsoft	$300.00	120

Assesment-6(Saturday)

Stock Market Simulator

1. Display Stocks

2. Buy Stock

3. Sell Stock

4. Display Transactions

5. Exit

Enter your choice: 2

Enter stock name to buy: Apple

Enter quantity: 100

Purchase successful! Total cost: $15000.00

Stock Market Simulator

1. Display Stocks

2. Buy Stock

3. Sell Stock

4. Display Transactions

5. Exit

Enter your choice: 3

Enter stock name to sell: Tesla

Enter quantity: 80

Sale successful! You earned: $56000.00

Stock Market Simulator

1. Display Stocks

Assesment-6(Saturday)

2. Buy Stock

3. Sell Stock

4. Display Transactions

5. Exit

Enter your choice: 4

Transaction History:

Type Stock Quantity Total Price

Buy Apple 100 $15000.00

Sell Tesla 80 $56000.00

Stock Market Simulator

1. Display Stocks

2. Buy Stock

3. Sell Stock

4. Display Transactions

5. Exit

Enter your choice: 5

Exiting program.

=== Code Execution Successful ===

Assesment-6(Saturday)

2. search_stock function:

```
#include <stdio.h>
#include <string.h>

#define MAX_STOCKS 5

// Structure to hold stock information
typedef struct {
    char symbol[10];
    float price;
    int quantity;
} Stock;

// Function to search for a stock by symbol
void search_stock(Stock stocks[], int size, char symbol[]) {
    for (int i = 0; i < size; i++) {
        if (strcmp(stocks[i].symbol, symbol) == 0) {
            printf("Stock Found: %s\n", stocks[i].symbol);
            printf("Price: $%.2f\n", stocks[i].price);
            printf("Quantity: %d\n", stocks[i].quantity);
            return;
        }
    }
    printf("Stock %s not found.\n", symbol);
}
```

Assesment-6(Saturday)

```
int main() {
    // Sample stock data
    Stock stocks[MAX_STOCKS] = {
        {"AAPL", 175.30, 100},
        {"GOOGL", 2800.50, 50},
        {"TSLA", 850.75, 200},
        {"MSFT", 305.20, 150},
        {"AMZN", 3400.40, 80}
    };

    char searchSymbol[10];

    printf("Enter stock symbol to search: ");
    scanf("%s", searchSymbol);

    search_stock(stocks, MAX_STOCKS, searchSymbol);

    return 0;
}
```

OUTPUT:

```
Enter stock symbol to search: AAPL
Stock Found: AAPL
Price: $175.30
Quantity: 100
```

Assesment-6(Saturday)

3. Linked list to store stock information:

```
#include <stdio.h>
#include <stdlib.h>
#include <string.h>

// Structure to store stock details
typedef struct Stock {
    char symbol[10];
    float price;
    int quantity;
    struct Stock* next;
} Stock;

// Function to create a new stock node
Stock* create_stock(char symbol[], float price, int quantity) {
    Stock* newStock = (Stock*)malloc(sizeof(Stock));
    strcpy(newStock->symbol, symbol);
    newStock->price = price;
    newStock->quantity = quantity;
    newStock->next = NULL;
    return newStock;
}

// Function to add a stock to the list
void add_stock(Stock** head, char symbol[], float price, int quantity) {
    Stock* newStock = create_stock(symbol, price, quantity);
```

```
    newStock->next = *head;
    *head = newStock;
    printf("Stock %s added successfully!\n", symbol);
}

// Function to delete a stock by its symbol
void delete_stock(Stock** head, char symbol[]) {
    Stock* temp = *head, *prev = NULL;

    while (temp != NULL && strcmp(temp->symbol, symbol) == 0) {
        *head = temp->next;
        free(temp);
        printf("Stock %s deleted successfully!\n", symbol);
        return;
    }

    while (temp != NULL && strcmp(temp->symbol, symbol) != 0) {
        prev = temp;
        temp = temp->next;
    }

    if (temp == NULL) {
        printf("Stock %s not found!\n", symbol);
        return;
    }
```

```
    prev->next = temp->next;
    free(temp);
    printf("Stock %s deleted successfully!\n", symbol);
}

// Function to display all stocks
void display_stocks(Stock* head) {
    if (head == NULL) {
        printf("No stocks available.\n");
        return;
    }

    printf("Stock List:\n");
    while (head != NULL) {
        printf("Symbol: %s, Price: $%.2f, Quantity: %d\n", head->symbol, head->price, head->quantity);
        head = head->next;
    }
}

int main() {
    Stock* stockList = NULL;
    int choice;
    char symbol[10];
    float price;
    int quantity;
```

Assesment-6(Saturday)

```
do {
    printf("\nStock Management System\n");
    printf("1. Add Stock\n2. Delete Stock\n3. Display Stocks\n4. Exit\n");
    printf("Enter your choice: ");
    scanf("%d", &choice);

    switch (choice) {
        case 1:
            printf("Enter stock symbol: ");
            scanf("%s", symbol);
            printf("Enter stock price: ");
            scanf("%f", &price);
            printf("Enter quantity: ");
            scanf("%d", &quantity);
            add_stock(&stockList, symbol, price, quantity);
            break;
        case 2:
            printf("Enter stock symbol to delete: ");
            scanf("%s", symbol);
            delete_stock(&stockList, symbol);
            break;
        case 3:
            display_stocks(stockList);
            break;
        case 4:
            printf("Exiting program...\n");
```

```
            break;
        default:
            printf("Invalid choice! Try again.\n");
    }
  } while (choice != 4);

  return 0;
}
```

Output:

```
Stock Management System
1. Add Stock
2. Delete Stock
3. Display Stocks
4. Exit
Enter your choice:
1
Enter stock symbol: AAPL
Enter stock price: 150
Enter quantity: 100
Stock AAPL added successfully!

Stock Management System
1. Add Stock
2. Delete Stock
```

Assesment-6(Saturday)

3. Display Stocks

4. Exit

Enter your choice: 2

Enter stock symbol to delete: GOOGL

Stock GOOGL not found!

Stock Management System

1. Add Stock

2. Delete Stock

3. Display Stocks

4. Exit

Enter your choice:

3

Stock List:

Symbol: AAPL, Price: $150.00, Quantity: 100

Stock Management System

1. Add Stock

2. Delete Stock

3. Display Stocks

4. Exit

Enter your choice: 4

Exiting program...