Saudi America
Portfolio Project

Executive Summary

The database design and implementation project has undergone a significant evolution since its inception, driven by iterative feedback and continuous improvement efforts. Initially, the project set out with a comprehensive schema aligned with the database outline and ER diagram, emphasizing consistency in naming conventions, table structures, and data types. However, as the project progressed, peer reviewers provided invaluable insights that guided several key enhancements.

One notable area of focus was the readability and organization of the schema. Peer feedback prompted a re-evaluation of naming conventions and schema layout to ensure clarity and ease of understanding. This led to refinements in the naming of entities and attributes, as well as improvements in the organization of tables within the schema. Additionally, suggestions were incorporated to enhance normalization and address potential non-normalized issues.

Throughout the implementation phase, peer feedback played a pivotal role in evaluating the syntactical correctness of SQL files and ensuring adherence to best practices. Recommendations for optimizing SQL structure, including the incorporation of CASCADE operations to maintain referential integrity, were carefully considered and implemented. These enhancements not only improved the overall integrity of the database but also facilitated smoother data management and manipulation processes.

Feedback from peer reviewers guided the implementation of CRUD functionalities for each table in the schema, ensuring comprehensive data management capabilities. Suggestions for incorporating dynamically populating dropdown lists were integrated to streamline user interactions and improve efficiency. Furthermore, recommendations for enhancing data representation through JOIN queries to display more meaningful information were taken into account. By leveraging JOINs and aliases, the UI was optimized to present relevant data in a user-friendly format, improving usability.

Overall, the project's progression from its initial proposal to its current state reflects a collaborative and iterative approach to database design and implementation. Through continuous feedback and refinement, the project has evolved into a refined and efficient database solution that meets both functional requirements and user expectations. The culmination of these efforts resulted in the successful engineering of a fully functional website equipped with CRUD functionalities and CSS styling, implemented using Express.js and Node.js. These enhancements demonstrate a commitment to delivering a high-quality database solution that is both robust and user-friendly, ultimately contributing to the project's success and effectiveness.

Project Overview

Saudi America, a burgeoning brokerage startup, is dedicated to bridging the gap between Middle Eastern and American markets, catering to the needs of everyday retail investors. With a substantial clientele of approximately 100,000 users, the platform facilitates an impressive $5,000,000,000 USD in trades each month. To efficiently manage and streamline their operations, Saudi America requires a robust database-backed system. This system would play a pivotal role in tracking and organizing a multitude of crucial data points, including but not limited to orders, securities, positions, and investor information. The database will enable seamless transaction tracking, portfolio management, and client engagement, providing the scalability needed to accommodate the dynamic nature of a rapidly growing user base and transaction volume.

Database Outline

- **Orders: Describes an order, the buying or selling, of a particular security.**
    - orderID [PK, INT, not NULL, auto increment]
    - investorID [FK, INT, not NULL]
    - securityID [FK, INT]
    - positionID [INT, not Null]
    - orderType [VARCHAR, not NULL]
    - totalUnits [decimal (65, 2)]
    - price [decimal (65, 2), not NULL]
    - tradeTotal [decimal (65, 2), not NULL]
    - date [DATE, not NULL]
    - *Relationships*:
        - **Orders -> Investors = M:1 (Optional Relationship)**
            - Many orders may be associated with a given investor, but only one investor must be associated with any given order.

          Orders -> Securities = M:1 (Optional Relationship)
            - Many orders may be associated with a given stock, but only one stock must be associated with any given order.
        - **Orders -> Positions = M:1 (Optional Relationship)**
            - Many orders may be associated with a given position, but only one position may be associated with any given order.

- **Securities: Describes a security's name, ticker, current price, and other important info.**
    - securityID [PK, INT, not NULL, auto increment]
    - name [VARCHAR, not NULL]
    - ticker [VARCHAR, not NULL]
    - currentPrice [decimal (65, 2), not NULL]
    - *Relationships*:
        - **Securities -> Orders = 1:M (Optional Relationship)**
            - Only one security may be associated with a given order, but many orders may be associated with a given security.

    - **Securities -> Positions = 1:M (Mandatory Relationship)**
        - Only one security must be associated with a given position, but many positions may be associated with a given security.
    - **Securities -> SecurityCodes = M:N (Optional Relationship)**
        - A given security may be associated with many security codes, and a given security code may be associated with many securities.

- **<u>SecurityCodes</u>: Category table used to further describe each security, such as type.**
    - securityCodeID [PK, INT, not NULL, auto increment]
    - description [VARCHAR, not NULL]
    - *Relationship*:
        - **SecurityCodes -> Securities = M:N (Optional Relationship)**
            - A given securities code may be associated with many securities and a given security may be associated with many securities codes.

- **<u>SecurityCodesSecurities</u>: Intersection table joining SecurityCodes with Securities.**
    - scsID [PK, INT, not Null, auto increment]
    - securityID [FK, INT]
    - securityCodeID [FK, INT]
    - *Relationships*:
        - **SecurityCodesSecurities -> SecurityCodes: = M:1 (Mandatory Relationship)**
            - Many SecurityCodesSecurities items may be associated with one security code, but only one security code may be associated with a given SecurityCodesSecurities code item.
        - **SecurityCodesSecurities -> Securities = M:1 (Mandatory Relationship)**
            - Many SecurityCodesSecurities items may be associated with one security, but only one security may be associated with a given SecurityCodesSecurities item.

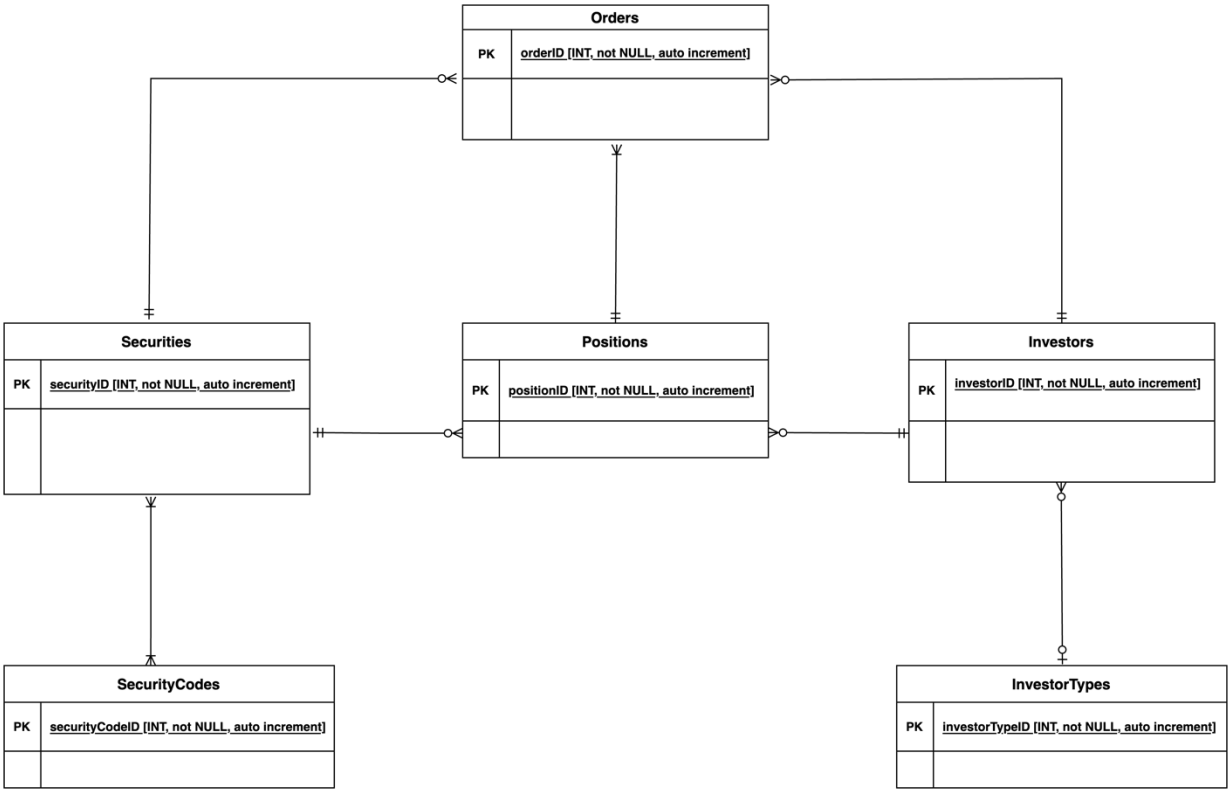
- **<u>Investors</u>: Describes an investor who trades on our platform – name, investor type, various balances, and their portfolio peformance.**
    - investorID [PK, INT, not NULL, auto increment]
    - investorType [FK, INT]
    - name [VARCHAR, not NULL]
    - cashBalance [decimal (65, 2), not NULL]
    - portfolioBalance [decimal (65, 2), not NULL]
    - portfolioPerformance [decimal (65, 2), not NULL]
    - *Relationships*:
        - **Investors -> Orders = 1:M (Mandatory Relationship)**
            - Only one investor must be associated with a given order, but many orders may be associated with a given investor.
        - **Investors -> InvestorTypes = M:1 (Optional Relationship)**
            - Many investors may be associated with a given investor type, but only one investor type must be associated with a given investor.

        - **Investors -> Positions = 1:M (Mandatory Relationship)**
            - Only one investor must be associated with a given position, but many positions may be associated with one investor.

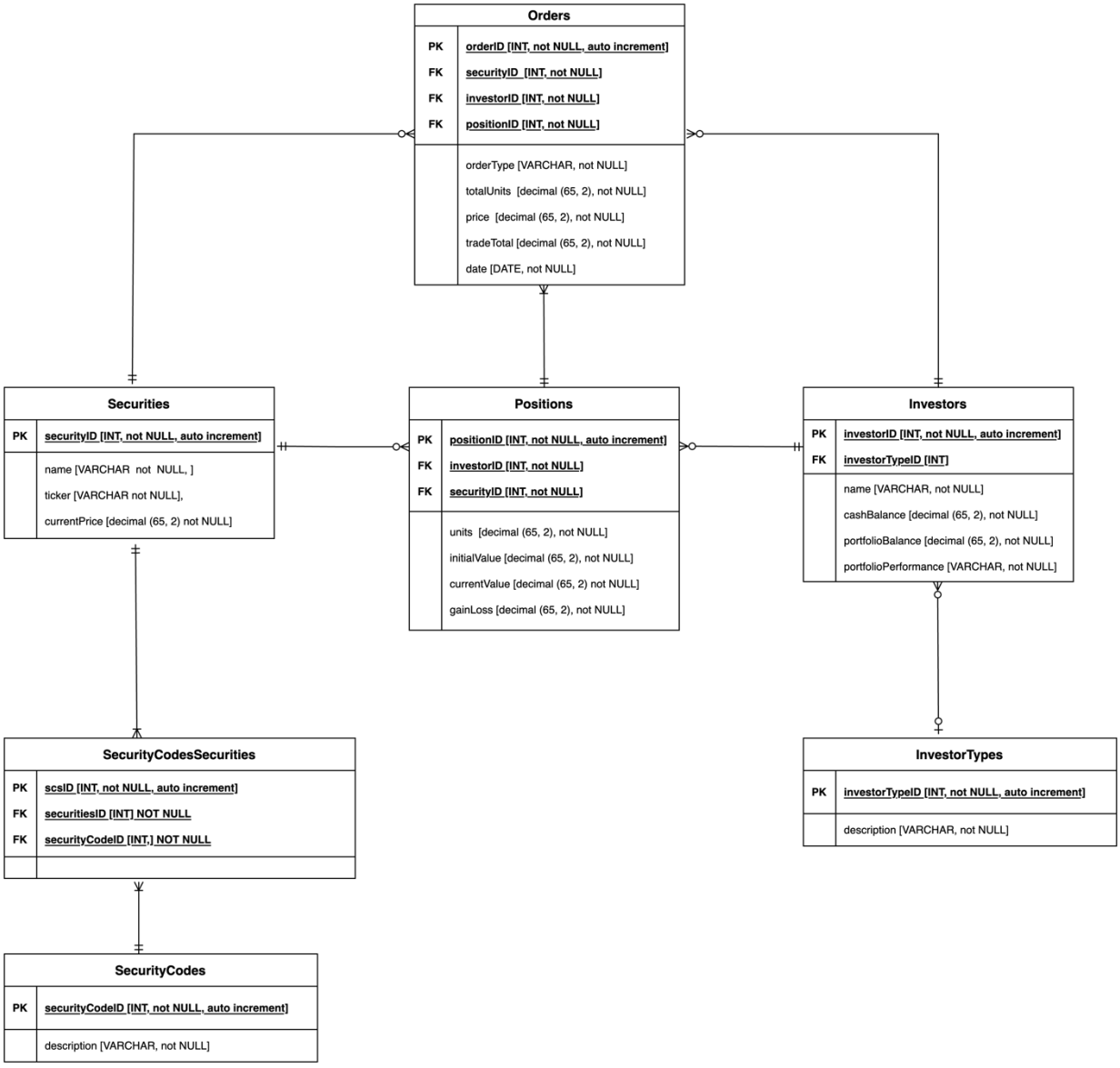
- **<u>InvestorTypes</u>: Describes the type of investor who trades on our platform – individual, bank, hedge fund, broker, or market maker.**
    - investorTypeID [PK, INT, not NULL, auto increment]
    - description [VARCHAR, not NULL]
    - *Relationship*:
        - **InvestorTypes -> Investors = 1:M (Optional Relationship)**
            - Only one investor type must be associated with a given investor, but many investors may be associated with a given investor type.

- **<u>Positions</u>: Describes an investor's position in a particular security, in terms of the number of units (meaning shares or tokens) owned and the cumulative value of those units.**
    - positionID [PK, INT, not NULL, auto increment]
    - investorID [FK, INT, not NULL]
    - securityID [FK, INT, not NULL]
    - units [decimal (65, 2), not NULL]
    - initialValue [decimal (65, 2), not NULL]
    - currentValue [decimal (65, 2), not NULL]
    - gainLoss [decimal (65, 2), not NULL]
    - *Relationships*:
        - **Positions -> Securities = M:1 (Optional Relationship)**
            - Many positions may be associated with a given security, but only one security must be associated with a given position.
        - **Positions -> Investors = M:1 (Optional Relationship)**
            - Many positions may be associated with a given investor, but only one investor must be associated with a given position.
        - **Positions -> Orders = 1:M (Mandatory Relationship)**
            - Only one position must be associated with a given order, but many orders may be associated with a given position.

Entity-Relationship Diagram



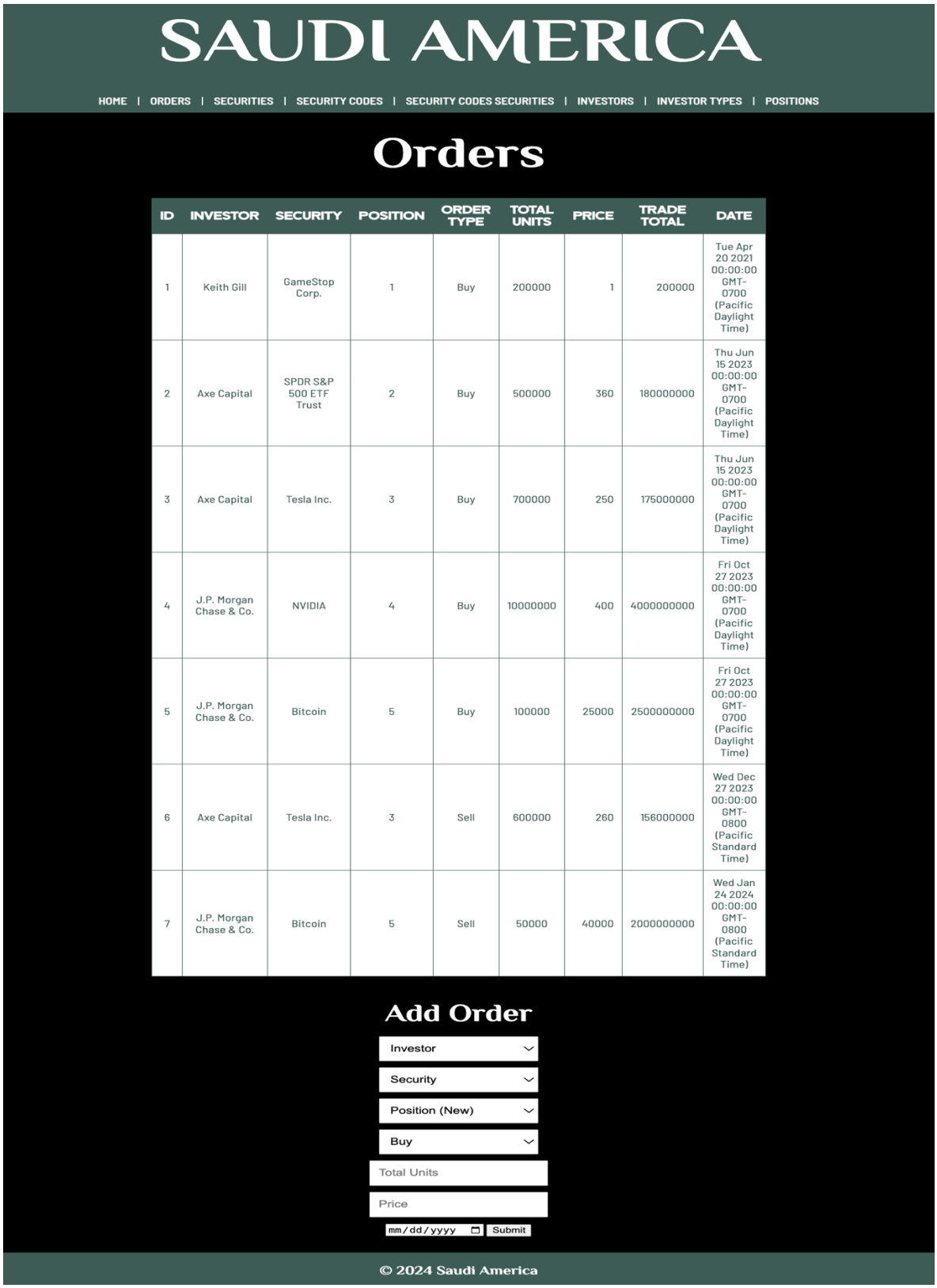
Schema


Orders
PK orderID [INT, not NULL, auto increment]
FK securityID [INT, not NULL]
FK investorID [INT, not NULL]
FK positionID [INT, not NULL]
orderType [VARCHAR, not NULL]
totalUnits [decimal (65, 2), not NULL]
price [decimal (65, 2), not NULL]
tradeTotal [decimal (65, 2), not NULL]
date [DATE, not NULL]
Securities
PK securityID [INT, not NULL, auto increment]
name [VARCHAR not NULL,]
ticker [VARCHAR not NULL],
currentPrice [decimal (65, 2) not NULL]
Positions
PK positionID [INT, not NULL, auto increment]
FK investorID [INT, not NULL]
FK securityID [INT, not NULL]
units [decimal (65, 2), not NULL]
initialValue [decimal (65, 2), not NULL]
currentValue [decimal (65, 2) not NULL]
gainLoss [decimal (65, 2), not NULL]
Investors
PK investorID [INT, not NULL, auto increment]
FK investorTypeID [INT]
name [VARCHAR, not NULL]
cashBalance [decimal (65, 2), not NULL]
portfolioBalance [decimal (65, 2), not NULL]
portfolioPerformance [VARCHAR, not NULL]
SecurityCodesSecurities
PK scsID [INT, not NULL, auto increment]
FK securitiesID [INT] NOT NULL
FK securityCodeID [INT,] NOT NULL
InvestorTypes
PK investorTypeID [INT, not NULL, auto increment]
description [VARCHAR, not NULL]
SecurityCodes
PK securityCodeID [INT, not NULL, auto increment]
description [VARCHAR, not NULL]

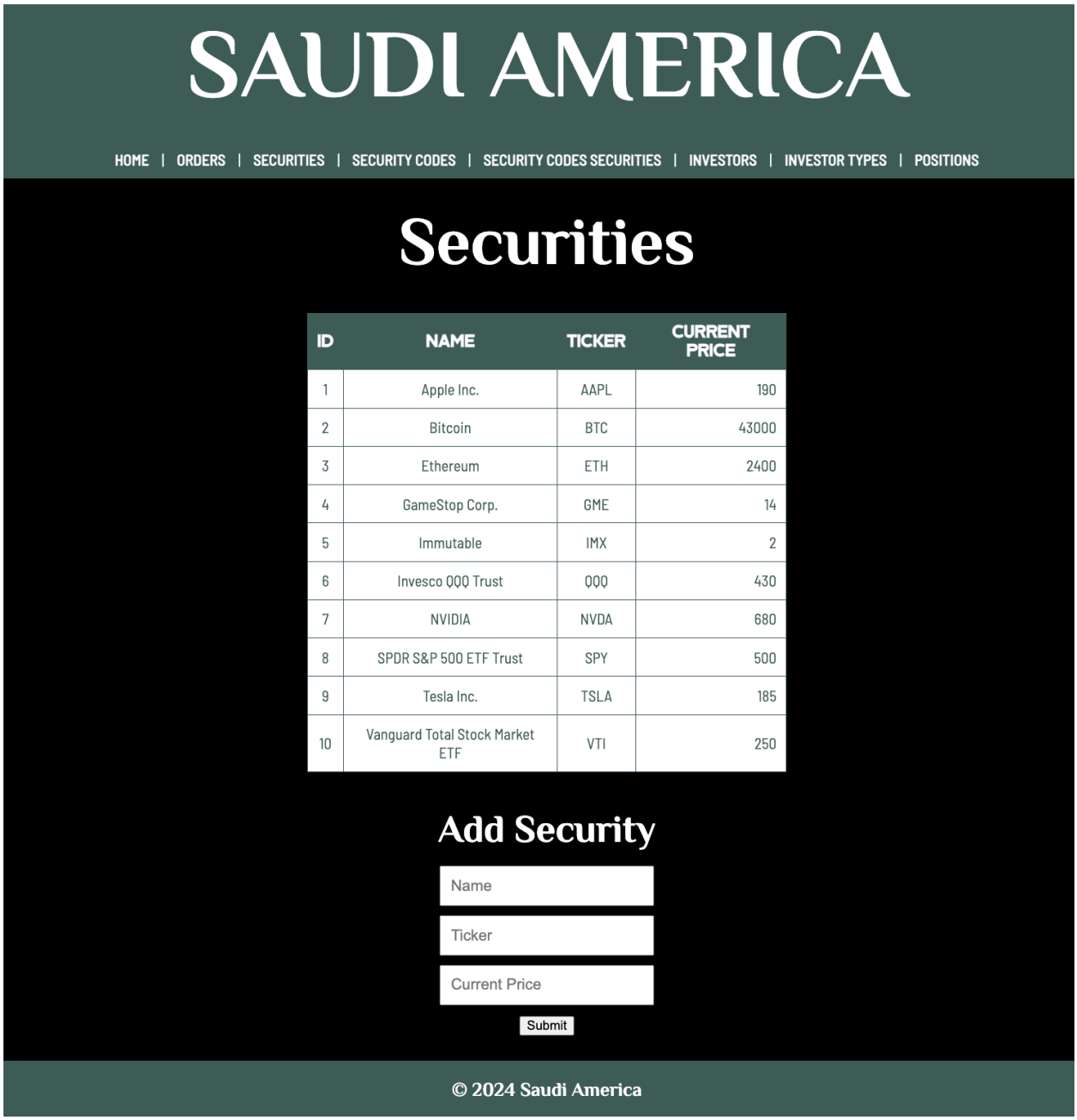
UI Screenshots

HOME – NO CRUD OPERATIONS

ORDERS – VIEW/INSERT



SECURITIES – VIEW/INSERT



SECURITY CODES – VIEW/INSERT

SECURITY CODES SECURITIES – VIEW/INSERT/DELETE (M:N)

SECURITY CODES SECURITIES – UPDATE (M:N)

INVESTORS – VIEW/INSERT (nullable FK – investorTypeID)

INVESTOR TYPES – VIEW/INSERT

POSITIONS – VIEW/INSERT

POSITIONS – UPDATE