LAPORAN
ALGORITMA DAN PEMROGRAMAN 2
“APLIKASI SIMULASI PASAR SAHAM VIRTUAL”

DISUSUN OLEH:

Raja Muhammad Lufhti 103112400027

M. Davi Ilyas Renaldo 103112400062

Muhammad Shabrian Fadly 103112400087

S1 IF-12-01

DOSEN:

Dimas Fanny Hebrisianto Permadi S.ST, M.Kom

PROGRAM STUDI S1 TEKNIK INFORMATIKA

FAKULTAS INFORMATIKA

TELKOM UNIVERSITY PURWOKERTO

2024/2025

Aplikasi Simulasi Pasar Saham Virtual

Raja Muhammad Lufhti[1], M. Davi Ilyas Renaldo[2], MuhammadShabrian Fadly[3]

Program Studi Teknik Informatika, Telkom University Purwokerto Jl. DI Panjaitan No.128, Karangreja, Purwokerto Kidul, Kec. Purwokerto Sel., Kabupaten Banyumas, Jawa Tengah 53147

rajamuhammadlufhti@student.telkomuniversity.ac.id[1], mdaviilyasrenaldo@student.telkomuniversity.ac.id[2], muhammadshabrian@student.telkomuniversity.ac.id[3]

Abstrak

Studi ini berfokus pada perancangan dan implementasi aplikasi simulasi pasar saham virtual yang bertujuan untuk menyediakan platform pembelajaran yang aman dan komprehensif bagi calon investor. Mengingat volatilitas dan kerumitan pasar saham, seringkali terdapat hambatan bagi individu untuk memulai aktivitas investasi. Aplikasi ini hadir sebagai solusi untuk mengatasi tantangan tersebut, dengan menciptakan lingkungan simulasi yang mereplikasi mekanisme pasar saham riil, termasuk pergerakan harga, informasi pasar, dan ragam instrumen investasi yang tersedia.

Proses pengembangan aplikasi melibatkan desain antarmuka pengguna yang mudah dipahami, integrasi data pasar historis dan, bila memungkinkan, data waktu nyata, serta penerapan algoritma untuk mensimulasikan order buku, eksekusi transaksi, dan fluktuasi nilai portofolio. Fitur-fitur utama yang ditawarkan meliputi dasbor portofolio pribadi, catatan transaksi terperinci, dan grafik harga yang interaktif. Pengguna dapat melakukan aktivitas jual-beli saham menggunakan dana virtual, yang memungkinkan mereka untuk menguji berbagai strategi investasi, memahami risiko pasar, dan menganalisis kinerja portofolio tanpa eksposur terhadap kerugian finansial aktual.

Temuan dari penelitian ini menunjukkan bahwa aplikasi simulasi pasar saham virtual berpotensi menjadi instrumen edukasi yang efektif. Aplikasi ini dapat berkontribusi dalam meningkatkan literasi keuangan serta kepercayaan diri pengguna dalam berinteraksi dengan pasar saham. Dengan demikian, diharapkan aplikasi ini dapat mendorong partisipasi investasi yang lebih bertanggung jawab, didukung oleh pemahaman yang lebih mendalam mengenai cara kerja pasar serta potensi risiko dan keuntungan yang menyertainya.

PENDAHULUAN

Di era digital saat ini, akses terhadap informasi dan teknologi semakin terbuka lebar, termasuk di sektor keuangan. Pasar saham, sebagai salah satu instrumen investasi paling dinamis, menawarkan potensi keuntungan yang menarik namun juga menyimpan risiko yang signifikan. Banyak individu, terutama generasi muda dan mereka yang baru tertarik pada investasi, seringkali merasa enggan atau takut untuk terjun langsung karena kurangnya pemahaman mendalam tentang mekanisme pasar, volatilitas harga, dan kompleksitas instrumen investasi. Kesalahan investasi yang diakibatkan oleh minimnya pengetahuan dapat berujung pada kerugian finansial yang substansial, sehingga menciptakan persepsi bahwa investasi saham adalah aktivitas yang berisiko tinggi dan hanya cocok untuk kalangan tertentu.

Literasi keuangan yang rendah menjadi salah satu akar permasalahan. Mayoritas masyarakat belum sepenuhnya memahami konsep dasar investasi, manajemen risiko, atau cara menganalisis pergerakan pasar. Hal ini didukung oleh berbagai penelitian yang menunjukkan tingkat literasi keuangan yang masih perlu ditingkatkan di berbagai negara, termasuk di Indonesia**.** Misalnya, berdasarkan Survei Nasional Literasi dan Inklusi Keuangan (SNLIK) oleh Otoritas Jasa Keuangan (OJK) tahun 2022, tingkat literasi keuangan masyarakat Indonesia baru mencapai 49,68% (OJK, 2022). Sumber belajar yang ada seringkali bersifat teoritis dan kurang memberikan pengalaman praktis yang dibutuhkan untuk membangun kepercayaan diri. Di sinilah peran simulasi menjadi krusial**.** Simulasi menawarkan lingkungan bebas risiko di mana pengguna dapat bereksperimen, belajar dari kesalahan, dan mengembangkan strategi investasi tanpa konsekuensi finansial, sebagaimana ditekankan dalam literatur pendidikan finansial. Aplikasi simulasi telah terbukti efektif dalam meningkatkan pemahaman dan keterampilan praktis di berbagai bidang kompleks . Dengan demikian, pengembangan aplikasi simulasi pasar saham virtual menjadi sangat relevan sebagai jembatan antara teori dan praktik, memungkinkan individu untuk merasakan dinamika pasar saham secara langsung dan membangun fondasi pengetahuan yang kuat sebelum berinvestasi di dunia nyata.

Tujuan

Aplikasi simulasi pasar saham virtual yang kami kembangkan ini punya beberapa misi utama:

1. Sarana Belajar yang Aman: Kami ingin menciptakan "arena latihan" yang mirip dengan pasar saham sungguhan. Di sini, kamu bisa berlatih jual-beli saham, mengelola portofolio, dan memahami bagaimana harga bergerak naik-turun, tanpa perlu cemas uangmu akan hilang beneran.
2. Meningkatkan Pemahaman Keuangan: Aplikasi ini dirancang untuk membantumu mengerti dasar-dasar investasi saham, seperti pentingnya tidak menaruh semua telur dalam satu keranjang (diversifikasi), cara menganalisis saham sederhana, dan yang paling penting, bagaimana mengelola risiko, semua itu lewat pengalaman yang praktis dan interaktif.
3. Membangun Kepercayaan Diri: Dengan aplikasi ini, kamu punya kebebasan untuk mencoba berbagai strategi investasi, melihat hasilnya, dan belajar dari setiap kesalahan di lingkungan yang aman. Harapannya, ini akan meningkatkan rasa percaya dirimu sebelum kamu benar-benar terjun ke investasi yang sesungguhnya.
4. Menghubungkan Teori dan Praktik: Kami ingin aplikasi ini jadi jembatan yang sempurna antara pengetahuan teoritis tentang pasar saham dan pengalaman praktis. Jadi, saat nanti kamu sudah siap, kamu akan lebih matang menghadapi tantangan dan peluang di pasar investasi sesungguhnya.

SOURCE CODE

package main

```
package main

import (
    "bufio"
    "encoding/json"
    "fmt"
    "io/ioutil" //buat baca/nulis file
    "os"
    "sort"
    "strconv"
    "strings"
)

// (huruf pertama kapital) aiar bisa diakses oleh paket encoding/json
type Stock struct {
    Name   string  `json:"name"` // nambahin tag JSON untuk penamaan field di JSON
    Code   string  `json:"code"`
    Price  float64 `json:"price"`
    Volume int     `json:"volume"`
}

type User struct {
    Name      string         `json:"name"`
    Balance   float64        `json:"balance"`
    Portfolio map[string]int `json:"portfolio"`
}

// struct pembantu
type AppData struct {
    User          User     `json:"user"`
    MarketStocks   []Stock `json:"market_stocks"`
    InitialBalance float64 `json:"initial_balance"`
}

// buat sv data
func saveData(user User, stocks []Stock, initialBalance float64, filename string) error {
    data := AppData{
        User:          user,
        MarketStocks:   stocks,
        InitialBalance: initialBalance,
    }

    jsonData, err := json.MarshalIndent(data, "", "  ")
    if err != nil {
        return fmt.Errorf("gagal meng-marshal data: %w", err)
    }
```

```
    err = ioutil.WriteFile(filename, jsonData, 0644) //permission file
    if err != nil {
        return fmt.Errorf("gagal menulis data ke file %s: %w", filename, err)
    }

    fmt.Printf("Data berhasil disimpan ke %s\n", filename)
    return nil
}

// buat load data
func loadData(filename string) (*User, []Stock, float64, error) {
    _, err := os.Stat(filename)
    if os.IsNotExist(err) {
        // file gak ada, anggap ini startup pertama
        return nil, nil, 0, nil
    } else if err != nil {
        return nil, nil, 0, fmt.Errorf("gagal memeriksa status file %s: %w", filename, err)
    }

    jsonData, err := ioutil.ReadFile(filename)
    if err != nil {
        return nil, nil, 0, fmt.Errorf("gagal membaca file %s: %w", filename, err)
    }

    var data AppData
    err = json.Unmarshal(jsonData, &data)
    if err != nil {
        return nil, nil, 0, fmt.Errorf("gagal meng-unmarshal data dari file %s: %w",
filename, err) // gagal ngubah data
    }

    fmt.Printf("Data berhasil dimuat dari %s\n", filename)
    return &data.User, data.MarketStocks, data.InitialBalance, nil
}

func (u *User) BuyStock(stock Stock, quantity int) {
    if quantity <= 0 {
        fmt.Println("Kuantitas untuk membeli harus positif.")
        return
    }
    totalCost := stock.Price * float64(quantity)
    if u.Balance >= totalCost {
        u.Balance -= totalCost
        u.Portfolio[stock.Code] += quantity
        fmt.Printf("Berhasil membeli %d lembar saham %s (%s) seharga $%.2f per lembar.
Saldo baru: $%.2f\n", quantity, stock.Name, stock.Code, stock.Price, u.Balance)
    } else {
```

```
        fmt.Printf("Saldo tidak mencukupi untuk membeli %d lembar saham %s.
Dibutuhkan: $%.2f, Tersedia: $%.2f\n", quantity, stock.Name, totalCost, u.Balance)
    }
}

func (u *User) SellStock(stock Stock, quantity int) {
    if quantity <= 0 {
        fmt.Println("Kuantitas untuk menjual harus positif.")
        return
    }
    if currentQuantity, ok := u.Portfolio[stock.Code]; ok && currentQuantity >=
quantity {
        totalSale := stock.Price * float64(quantity)
        u.Balance += totalSale
        u.Portfolio[stock.Code] -= quantity
        if u.Portfolio[stock.Code] == 0 {
            delete(u.Portfolio, stock.Code)
        }
        fmt.Printf("Berhasil menjual %d lembar saham %s (%s) seharga $%.2f per lembar.
Saldo baru: $%.2f\n", quantity, stock.Name, stock.Code, stock.Price, u.Balance)
    } else {
        fmt.Printf("Tidak cukup lembar saham %s untuk dijual. Anda memiliki %d,
mencoba menjual %d.\n", stock.Name, u.Portfolio[stock.Code], quantity)
    }
}

func (u *User) DisplayPortfolio(marketStocks []Stock) {
    fmt.Printf("\nPortofolio untuk %s:\n", u.Name)
    if len(u.Portfolio) == 0 {
        fmt.Println("  Portofolio Anda kosong.")
    } else {
        fmt.Println("-------------------------------------------------------------------------")
        fmt.Printf("  %-20s | %-10s | %-10s | %-15s | %-15s\n", "Nama Saham", "Kode",
"Kuantitas", "Harga Saat Ini", "Nilai Saat Ini")
        fmt.Println("-------------------------------------------------------------------------")
        var totalPortfolioValue float64
        for code, quantity := range u.Portfolio {
            if quantity == 0 {
                continue
            }
            var currentStockPrice float64 = -1
            var stockName string = "N/A"
            foundInMarket := false
            for _, marketStock := range marketStocks {
                if strings.EqualFold(marketStock.Code, code) {
                    currentStockPrice = marketStock.Price
                    stockName = marketStock.Name
                    foundInMarket = true
                    break
```

```
            }
        }

        if foundInMarket {
            currentValue := currentStockPrice * float64(quantity)
            totalPortfolioValue += currentValue
            fmt.Printf("  %-20s | %-10s | %-10d | $%14.2f | $%14.2f\n", stockName, code,
quantity, currentStockPrice, currentValue)
        } else {
            fmt.Printf("  %-20s | %-10s | %-10d | %-15s | %-15s\n", "Tidak
Dikenal/Delisting", code, quantity, "Harga T/A", "Nilai T/A")
        }
    }
    fmt.Println("-----------------------------------------------------------------------")
    fmt.Printf("  Total Nilai Saham: $%.2f\n", totalPortfolioValue)
  }
  fmt.Printf("  Saldo Tunai: $%.2f\n", u.Balance)
  fmt.Println("==========================================
==============================")
}

func SequentialSearch(stocks []Stock, searchTerm string) *Stock {
  searchTermLower := strings.ToLower(searchTerm)
  for i, stock := range stocks {
    if strings.ToLower(stock.Code) == searchTermLower ||
strings.ToLower(stock.Name) == searchTermLower {
      return &stocks[i]
    }
  }
  return nil
}

func SelectionSortByPrice(stocks []Stock) {
  n := len(stocks)
  for i := 0; i < n-1; i++ {
    maxIdx := i
    for j := i + 1; j < n; j++ {
      if stocks[j].Price > stocks[maxIdx].Price {
        maxIdx = j
      }
    }
    stocks[i], stocks[maxIdx] = stocks[maxIdx], stocks[i]
  }
}

func SelectionSortByVolume(stocks []Stock) {
  n := len(stocks)
  for i := 0; i < n-1; i++ {
    maxIdx := i
```

```
        for j := i + 1; j < n; j++ {
            if stocks[j].Volume > stocks[maxIdx].Volume {
                maxIdx = j
            }
        }
        stocks[i], stocks[maxIdx] = stocks[maxIdx], stocks[i]
    }
}

func readString(prompt string) string {
    reader := bufio.NewReader(os.Stdin)
    fmt.Print(prompt)
    input, _ := reader.ReadString('\n')
    return strings.TrimSpace(input)
}

func readFloat(prompt string) float64 {
    for {
        s := readString(prompt)
        val, err := strconv.ParseFloat(s, 64)
        if err == nil && val >= 0 {
            return val
        }
        fmt.Println("Input tidak valid. Silakan masukkan angka non-negatif.")
    }
}

func readInt(prompt string) int {
    for {
        s := readString(prompt)
        val, err := strconv.Atoi(s)
        if err == nil && val >= 0 {
            return val
        }
        fmt.Println("Input tidak valid. Silakan masukkan bilangan bulat non-negatif.")
    }
}

func printStocks(stockList []Stock, title string) {
    if title != "" {
        fmt.Println(title)
    }
    if len(stockList) == 0 {
        fmt.Println("Tidak ada saham untuk ditampilkan.")
        return
    }
    fmt.Println("-----------------------------------------------------------------")
    fmt.Printf("%-20s | %-10s | %-10s | %-10s\n", "Nama", "Kode", "Harga", "Volume")
    fmt.Println("-----------------------------------------------------------------")
```

```
    for _, stock := range stockList {
        fmt.Printf("%-20s | %-10s | $%8.2f | %-10d\n", stock.Name, stock.Code,
stock.Price, stock.Volume)
    }
    fmt.Println("----------------------------------------------------------------")
}

func printStocksVolume(stockList []Stock, title string) {
    if title != "" {
        fmt.Println(title)
    }
    if len(stockList) == 0 {
        fmt.Println("Tidak ada saham untuk ditampilkan.")
        return
    }
    fmt.Println("----------------------------------------------------------------")
    fmt.Printf("%-20s | %-10s | %-10s | %-10s\n", "Nama", "Kode", "Volume", "Harga")
    fmt.Println("----------------------------------------------------------------")
    for _, stock := range stockList {
        fmt.Printf("%-20s | %-10s | %-10d | $%8.2f\n", stock.Name, stock.Code,
stock.Volume, stock.Price)
    }
    fmt.Println("----------------------------------------------------------------")
}

func main() {
    fmt.Println("Selamat Datang di Simulator Perdagangan Saham!")

    const dataFile = "simulasi_saham.json" // nanti jadi nama file yg dibuat

    var user *User
    var stocks []Stock
    var initialBalance float64

    //buat load data dari file
    loadedUser, loadedStocks, loadedInitialBalance, err := loadData(dataFile)
    if err != nil {
        fmt.Println("Error saat memuat data:", err)
        //semisal error fatal ada inisialisasi baru saat load
        user = nil
        stocks = nil
        initialBalance = 0
    } else if loadedUser != nil {
        //kalo data berhasil diload
        user = loadedUser
        stocks = loadedStocks
        initialBalance = loadedInitialBalance
        fmt.Printf("Melanjutkan sesi untuk pengguna %s dengan saldo $%.2f\n",
user.Name, user.Balance)
```

```
    }

    // kalo data tidak dimuat (misalnya file tidak ada atau error), inisialisasi baru
    if user == nil {
        numStocks := readInt("Masukkan jumlah saham yang tersedia di pasar: ")
        stocks = make([]Stock, numStocks)
        for i := 0; i < numStocks; i++ {
            fmt.Printf("\nMasukkan detail untuk Saham #%d:\n", i+1)
            stocks[i].Name = readString("  Nama: ")

            for {
                codeCandidate := readString("  Kode (mis., AAPL, GOOG - harus unik): ")
                isUnique := true
                for j := 0; j < i; j++ {
                    if strings.EqualFold(stocks[j].Code, codeCandidate) {
                        isUnique = false
                        fmt.Println("  Kode saham sudah ada. Silakan masukkan kode yang
unik.")
                        break
                    }
                }
                if isUnique {
                    stocks[i].Code = codeCandidate
                    break
                }
            }
            stocks[i].Price = readFloat("  Harga Saat Ini: $")
            stocks[i].Volume = readInt("  Volume: ")
        }
        fmt.Println("\nSaham pasar berhasil diinisialisasi.")
        printStocks(stocks, "Saham Pasar Saat Ini:")

        userName := readString("\nMasukkan nama Anda: ")
        initialBalance = readFloat("Masukkan saldo awal perdagangan Anda: $")
        user = &User{Name: userName, Balance: initialBalance, Portfolio:
make(map[string]int)}
        fmt.Printf("\nPengguna %s berhasil dibuat dengan saldo $%.2f\n", user.Name,
user.Balance)
    }

    for {
        fmt.Println("\n-------------------------------------")
        fmt.Println("Pilih tindakan:")
        fmt.Println("1. Beli Saham")
        fmt.Println("2. Jual Saham")
        fmt.Println("3. Tampilkan Portofolio & Saldo")
        fmt.Println("4. Lihat Saham Pasar (Urut berdasarkan Harga)")
        fmt.Println("5. Lihat Saham Pasar (Urut berdasarkan Volume)")
        fmt.Println("6. Cari Saham (berdasarkan Nama atau Kode - Sekuensial)")
```

```
        fmt.Println("7. Hitung Total Untung/Rugi (berdasarkan saldo awal Anda)")
        fmt.Println("8. Keluar dan Simpan")
        fmt.Println("-----------------------------------")

        choiceStr := readString("Masukkan pilihan Anda (1-8): ")
        choice, err := strconv.Atoi(choiceStr)
        if err != nil {
            fmt.Println("Pilihan tidak valid. Silakan masukkan angka antara 1 dan 8.")
            continue
        }

        switch choice {
        case 1:
            fmt.Println("\n--- Beli Saham ---")
            if len(stocks) == 0 {
                fmt.Println("Tidak ada saham yang tersedia di pasar untuk dibeli.")
                continue
            }
            printStocks(stocks, "Saham Tersedia untuk Dibeli:")
            stockCode := readString("Masukkan Kode saham yang akan dibeli: ")
            stockToBuy := SequentialSearch(stocks, stockCode)
            if stockToBuy == nil {
                fmt.Println("Saham tidak ditemukan di pasar.")
                continue
            }
            quantity := readInt(fmt.Sprintf("Masukkan kuantitas %s yang akan dibeli: ",
stockToBuy.Name))
            user.BuyStock(*stockToBuy, quantity)

        case 2:
            fmt.Println("\n--- Jual Saham ---")
            if len(user.Portfolio) == 0 {
                fmt.Println("Portofolio Anda kosong. Tidak ada yang bisa dijual.")
                continue
            }
            user.DisplayPortfolio(stocks)
            stockCode := readString("Masukkan Kode saham yang akan dijual: ")

            ownedQuantity, ok := user.Portfolio[strings.ToUpper(stockCode)]
            if !ok || ownedQuantity == 0 {
                fmt.Printf("Anda tidak memiliki saham %s atau kodenya salah.\n", stockCode)
                continue
            }

            stockToSell := SequentialSearch(stocks, stockCode)
            if stockToSell == nil {
                fmt.Println("Error: Detail saham tidak ditemukan di pasar untuk saham dalam
portofolio Anda. Ini mungkin menunjukkan saham tersebut telah delisting.")
                continue
```

```
        }
        quantity := readInt(fmt.Sprintf("Masukkan kuantitas %s yang akan dijual (Anda memiliki %d): ", stockToSell.Name, ownedQuantity))
        user.SellStock(*stockToSell, quantity)

    case 3:
        fmt.Println("\n--- Portofolio Anda ---")
        user.DisplayPortfolio(stocks)

    case 4:
        fmt.Println("\n--- Saham Pasar (Urut berdasarkan Harga: Tertinggi ke Terendah) ---")
        if len(stocks) == 0 {
            fmt.Println("Tidak ada saham di pasar untuk ditampilkan.")
            continue
        }
        stocksToSort := make([]Stock, len(stocks))
        copy(stocksToSort, stocks)
        SelectionSortByPrice(stocksToSort)
        printStocks(stocksToSort, "")

    case 5:
        fmt.Println("\n--- Saham Pasar (Urut berdasarkan Volume: Tertinggi ke Terendah) ---")
        if len(stocks) == 0 {
            fmt.Println("Tidak ada saham di pasar untuk ditampilkan.")
            continue
        }
        stocksToSort := make([]Stock, len(stocks))
        copy(stocksToSort, stocks)
        SelectionSortByVolume(stocksToSort)
        printStocksVolume(stocksToSort, "")

    case 6:
        fmt.Println("\n--- Pencarian Saham Sekuensial ---")
        if len(stocks) == 0 {
            fmt.Println("Tidak ada saham di pasar untuk dicari.")
            continue
        }
        searchTerm := readString("Masukkan Nama atau Kode saham untuk dicari: ")
        foundStock := SequentialSearch(stocks, searchTerm)
        if foundStock != nil {
            fmt.Printf("Ditemukan: %s (%s), Harga: $%.2f, Volume: %d\n", foundStock.Name, foundStock.Code, foundStock.Price, foundStock.Volume)
        } else {
            fmt.Println("Saham tidak ditemukan menggunakan pencarian sekuensial.")
        }

    case 7:
```

```
        fmt.Println("\n--- Perhitungan Untung/Rugi ---")
        currentTotalValue := user.Balance
        for code, quantity := range user.Portfolio {
            stockFound := false
            for _, marketStock := range stocks {
                if strings.EqualFold(marketStock.Code, code) {
                    currentTotalValue += marketStock.Price * float64(quantity)
                    stockFound = true
                    break
                }
            }
            if !stockFound {
                fmt.Printf("Peringatan: Saham %s dalam portofolio tidak ditemukan di
pasar saat ini untuk perhitungan U/R.\n", code)
            }
        }
        profitLoss := currentTotalValue - initialBalance
        fmt.Printf("Saldo Awal: $%.2f\n", initialBalance)
        fmt.Printf("Total Nilai Portofolio Saat Ini (Tunai + Saham): $%.2f\n",
currentTotalValue)
        if profitLoss >= 0 {
            fmt.Printf("Total Untung: $%.2f\n", profitLoss)
        } else {
            fmt.Printf("Total Rugi: $%.2f\n", -profitLoss)
        }

    case 8:
        fmt.Println("\nMenyimpan data dan keluar...")
        if err := saveData(*user, stocks, initialBalance, dataFile); err != nil {
            fmt.Println("Gagal menyimpan data:", err)
        }
        fmt.Println("Terima kasih telah menggunakan Simulator Perdagangan Saham.
Sampai jumpa!")
        return

    default:
        fmt.Println("Pilihan tidak valid. Silakan masukkan angka antara 1 dan 8.")
    }
  }
}
```

DESKRIPSI

Program memperbolehkan user untuk melakukan beberapa kegiatan pasar saham seperti beli, jual saham, lihat portofolio, hitung keuntungan atau kerugian. Struktur data utama berupa dua struct, yaitu Stock yang digunakan untuk menggambarkan data saham (nama, kode, harga, volume) dan User yang digunakan untuk melewati data pengguna (nama, saldo, dan portofolio saham). Interaksi dengan pengguna dilakukan melalui menu teks berbasis, di mana pemilihan beberapa opsi seperti membeli saham, menjual saham, mengurutkan daftar saham berdasarkan harga atau volume, serta mencari saham dilakukan dengan menggunakan metode pencarian sekuensial atau biner. Proses input-output dikendalikan menggunakan beberapa fungsi pembantu, pada sementara fungsi-fungsi utama mengatur logika perdagangan, validasi saldo, serta pengelolaan data saham. Semua fitur dirancang untuk memberikan simulasi sederhana namun fungsional tentang mekanisme dasar pasar saham.

Deskripsi Program

Berikut adalah penjelasan mengenai setiap fungsi yang digunakan dalam aplikasi simulator perdagangan saham:

- *func saveData(user User, stocks []Stock, initialBalance float64, filename string) error*

  Fungsi ini menyimpan seluruh data aplikasi (informasi pengguna, daftar saham pasar, dan saldo awal) ke dalam file JSON, memungkinkan pemuatan kembali di kemudian hari.

- *func loadData(filename string) (*User, []Stock, float64, error)*

  Fungsi ini memuat data aplikasi yang tersimpan dari file JSON. Jika file tidak ditemukan atau ada error, ini menandakan inisialisasi sesi baru.

- *func (u *User) BuyStock(stock Stock, quantity int)*

  Metode User ini memungkinkan pengguna untuk membeli sejumlah lembar saham. Ini mengurangi saldo pengguna dan menambahkan saham ke portofolio jika dana mencukupi.

- *func (u *User) SellStock(stock Stock, quantity int)*

  Metode User ini memungkinkan pengguna untuk menjual sejumlah lembar saham dari portofolionya. Ini menambah saldo pengguna dan mengurangi saham di portofolio jika saham yang dimiliki mencukupi.

- *func (u *User) DisplayPortfolio(marketStocks []Stock)*

  Metode User ini menampilkan detail portofolio pengguna, termasuk saham yang dimiliki, kuantitas, harga terkini, nilai saat ini, dan saldo tunai.

- *func SequentialSearch(stocks []Stock, searchTerm string) *Stock*

  Fungsi ini melakukan pencarian linear pada daftar saham untuk menemukan saham berdasarkan nama atau kodenya (case-insensitive).

- *func SelectionSortByPrice(stocks []Stock)*

  Fungsi ini mengurutkan daftar saham berdasarkan harga dari tertinggi ke terendah menggunakan algoritma Selection Sort. Pengurutan dilakukan secara in-place.

- *func SelectionSortByVolume(stocks []Stock)*

  Fungsi ini mengurutkan daftar saham berdasarkan volume dari tertinggi ke terendah menggunakan algoritma Selection Sort. Pengurutan dilakukan secara in-place.

- *func readString(prompt string) string*

  Fungsi untuk membaca input teks dari konsol setelah menampilkan prompt.

- *func readFloat(prompt string) float64*

  Fungsi untuk membaca input angka desimal non-negatif dari konsol, dengan validasi input.

- *func readInt(prompt string) int*

  Fungsi untuk membaca input bilangan bulat non-negatif dari konsol, dengan validasi input.

- *func printStocks(stockList []Stock, title string)*

  Fungsi ini menampilkan daftar saham dalam format tabel standar (Nama, Kode, Harga, Volume).

- *func printStocksVolume(stockList []Stock, title string)*

  Fungsi ini menampilkan daftar saham dalam format tabel dengan urutan kolom berbeda (Nama, Kode, Volume, Harga).

- *func main()*

  Fungsi ini adalah titik awal program. Ia mengelola inisialisasi data (memuat atau membuat baru), menampilkan menu interaktif, dan memproses pilihan pengguna untuk menjalankan berbagai fitur simulasi.

TANTANGAN DAN SOLUSI

Tantangan: Duplikasi Kode untuk Pencarian Saham dalam Fungsi DisplayPortfolio dan Perhitungan Untung/Rugi. Dalam DisplayPortfolio dan bagian perhitungan untung/rugi, ada *loop* yang mencari saham berdasarkan kode di marketStocks atau stocks. Ini pada dasarnya adalah *sequential search* yang diulang-ulang.

- Solusi: Manfaatkan fungsi SequentialSearch yang sudah ada. Alih-alih melakukan *loop* pencarian manual di dalam DisplayPortfolio dan logika untung/rugi, panggil stockFound := SequentialSearch(marketStocks, code) atau stockFound := BinarySearch(marketStocks, code). Ini akan mengurangi duplikasi kode dan membuat pemeliharaan lebih mudah.

Tantangan**:** Bagaimana cara mengubah struktur data Go (struct User dan []Stock) menjadi format yang dapat disimpan ke file dan dibaca kembali? Data ini mengandung map (Portfolio) yang mungkin memerlukan penanganan khusus saat serialisasi.

- Solusi: Gunakan paket encoding/json dari Go. Paket ini menyediakan fungsi json.Marshal atau json.MarshalIndent untuk mengkonversi *struct* Go menjadi *byte* JSON, dan json.Unmarshal untuk mengkonversi *byte* JSON kembali ke *struct* Go. Untuk memastikan semua *field* di-*encode* dan di-*decode* dengan benar, pastikan semua *field* struct yang ingin disimpan memiliki nama yang diawali huruf kapital (diexport) dan berikan json tag (misalnya, json:"name") jika diinginkan penamaan kunci yang berbeda di JSON. Buatlah *struct* pembantu (AppData) untuk mengelompokkan semua data yang ingin disimpan (User, []Stock, initialBalance) ke dalam satu objek JSON.

KESIMPULAN DAN REKOMENDASI

Kesimpulan

Simulasi perdagangan saham ini berhasil menggambarkan mekanisme dasar jual beli saham secara interaktif menggunakan bahasa pemrograman Go. Melalui fitur-fitur seperti pembelian, penjualan, pencarian saham, pengurutan berdasarkan harga dan volume, serta perhitungan untung-rugi, pengguna dapat merasakan pengalaman sederhana namun realistis dalam mengelola portofolio saham. Penggunaan struct, fungsi modular, dan logika kontrol dalam bentuk menu terminal menjadikan program ini tidak hanya fungsional tetapi juga mudah dipahami dan dikembangkan lebih lanjut. Program ini dapat menjadi dasar yang baik untuk pengembangan sistem perdagangan saham yang lebih kompleks dan terhubung dengan data pasar nyata di masa depan.

Rekomendasi:

Program ini dapat menjadi dasar yang baik untuk pengembangan sistem perdagangan saham yang lebih kompleks. Rekomendasi untuk pengembangan selanjutnya meliputi:

- Integrasi data pasar historis dan, bila memungkinkan, data waktu nyata untuk meningkatkan realisme simulasi.
- Pengembangan lebih lanjut fitur-fitur yang ditawarkan, seperti dasbor portofolio pribadi yang lebih interaktif, catatan transaksi terperinci, dan grafik harga yang lebih dinamis.
- Peningkatan algoritma untuk mensimulasikan order buku dan eksekusi transaksi yang lebih kompleks, serta fluktuasi nilai portofolio yang lebih mendekati kondisi pasar nyata.
- Menambahkan fitur edukasi interaktif, seperti panduan investasi dasar, analisis teknikal dan fundamental sederhana, atau skenario pasar yang berbeda untuk melatih pengambilan keputusan.

TAUTAN GITHUB

https://github.com/Kelompok-G/Tubes

REFERENSI

GeeksforGeeks. (n.d.). *Binary Search in Golang*. Diakses dari https://www.geeksforge.org/binary-search/

Go by Example. (n.d.). *Reading Files*. Diakses dari https://medium.com/@smart_byte_labs/efficient-file-reading-in-go-examples-and-benchmark-comparisons-2335b097431a

Go by Example. (n.d.). *Sorting*. Diakses dari https://edwinsiby.medium.com/common-sorting-methods-in-data-structure-with-golang-d02ec6b965c8

Go by Example. (n.d.). *String Conversions*. Diakses dari https://pkg.go.dev/strings

Go by Example. (n.d.). *Structs*. Diakses dari https://gobyexample.com/structs

Go Programming Language. (n.d.). *Package encoding/json*. Diakses dari http://pkg.go.dev/encoding/json

Otoritas Jasa Keuangan (OJK). (2022). *Siaran Pers: Indeks Literasi dan Inklusi Keuangan Nasional Meningkat*. Diakses dari https://www.ojk.go.id/id/berita-dan-kegiatan/siaran-pers/Pages/Siaran-Pers-Indeks-Literasi-dan-Inklusi-Keuangan-Nasional-Meningkat.aspx

Prayogo, N.A. (n.d.). *JSON*. Dasar Pemrograman Golang. Diakses dari https://dasarpemrogramangolang.novalagung.com/A-json.html

TutorialsPoint. (n.d.). *Sequential Search Algorithm*. Diakses dari https://www.tutorialspoint.com/introduction-to-searching-algorithms