MANIPAL INSTITUTE OF TECHNOLOGY
MANIPAL
(A constituent unit of MAHE, Manipal)

A REPORT ON

PROJECT

In

Mobile Application Development

22-April-2025

On the Topic

"Stock Market Tracker App"

Group Member Details:

Sl. No.	Name of the Student	Reg. No.	Roll No.	Brief description of student's contribution
1	Prajwal Kini	220953005	2	Api , Database and Holdings .
2	Pranav Sai	220953342	29	Settings ,News and Report.
3	Anirudha L S	220953663	61	Watchlist, Search ,Database.

1. ABSTRACT

The Stock Market Tracker app provides users with a comprehensive platform to monitor stock prices, manage personal holdings, and track market news. It addresses the challenge of fragmented financial information by consolidating real-time stock data, portfolio management tools, and financial news in one application. Key features include stock searching with detailed information, personal holdings management with profit/loss tracking, customizable watchlists, and financial news integration, all within an intuitive interface designed for both novice and experienced investors.

2. INTRODUCTION

Individual investors often struggle to efficiently track their investments across multiple platforms. Traditional methods require toggling between various websites and applications to view stock prices, manage portfolios, and stay updated with financial news. This fragmentation creates inefficiency and potential information gaps that can impact investment decisions

As someone interested in both financial markets and mobile application development, I wanted to create a solution that simplifies the investment tracking process. The project offered an opportunity to work with real-world APIs, implement complex data visualization, and design an intuitive user interface for financial information.

With the increasing number of retail investors entering the market, there's a growing need for accessible tools that provide comprehensive market information. This app serves as a one-stop solution for investors to make informed decisions by consolidating essential financial data and tools in a single, user-friendly platform

3. PROJECT OBJECTIVE

The app aims to:

- Create a comprehensive stock tracking application that provides real-time market data
- Implement portfolio management functionality to track personal holdings and performance
- Develop a customizable watchlist for monitoring selected stocks
- Integrate financial news to keep users informed about market developments
- Design an intuitive, responsive interface that presents complex financial data clearly
- Ensure offline functionality for basic features when internet connectivity is unavailable.

4. TOOLS AND TECHNOLOGIES USED

Technology	Purpose
Android Studio (Kotlin DSL)	IDE for development
Java	Primary programming language
Database	Room Persistence Library for local storage
APIs	Alpha Vantage API for stock quotes and historical data EODHD API for financial news
RecyclerView	To display dynamic lists
Fragments & ViewPager	Tab navigation
Intents	Data transfer between activities

5. SYSTEM REQUIREMENTS

- **Minimum Android Version**: Android 8.0 (API level 26)
- **Target Android Version**: Android 13 (API level 33)
- **IDE Version**: Android Studio Flamingo | 2022.2.1
- **SDK Version**: Android SDK 33
- **Hardware Requirements**:
    - Minimum 2GB RAM
    - 100MB available storage

- Internet connectivity for real-time data

- **Permissions Required**:
  - Internet access
  - Network state acces

6.PROJECT MODULES/FEATURES

Search Module

The Search module allows users to look up stock information by entering a company's ticker symbol. It displays comprehensive details including current price, price change, previous close, P/E ratio, and market capitalization.

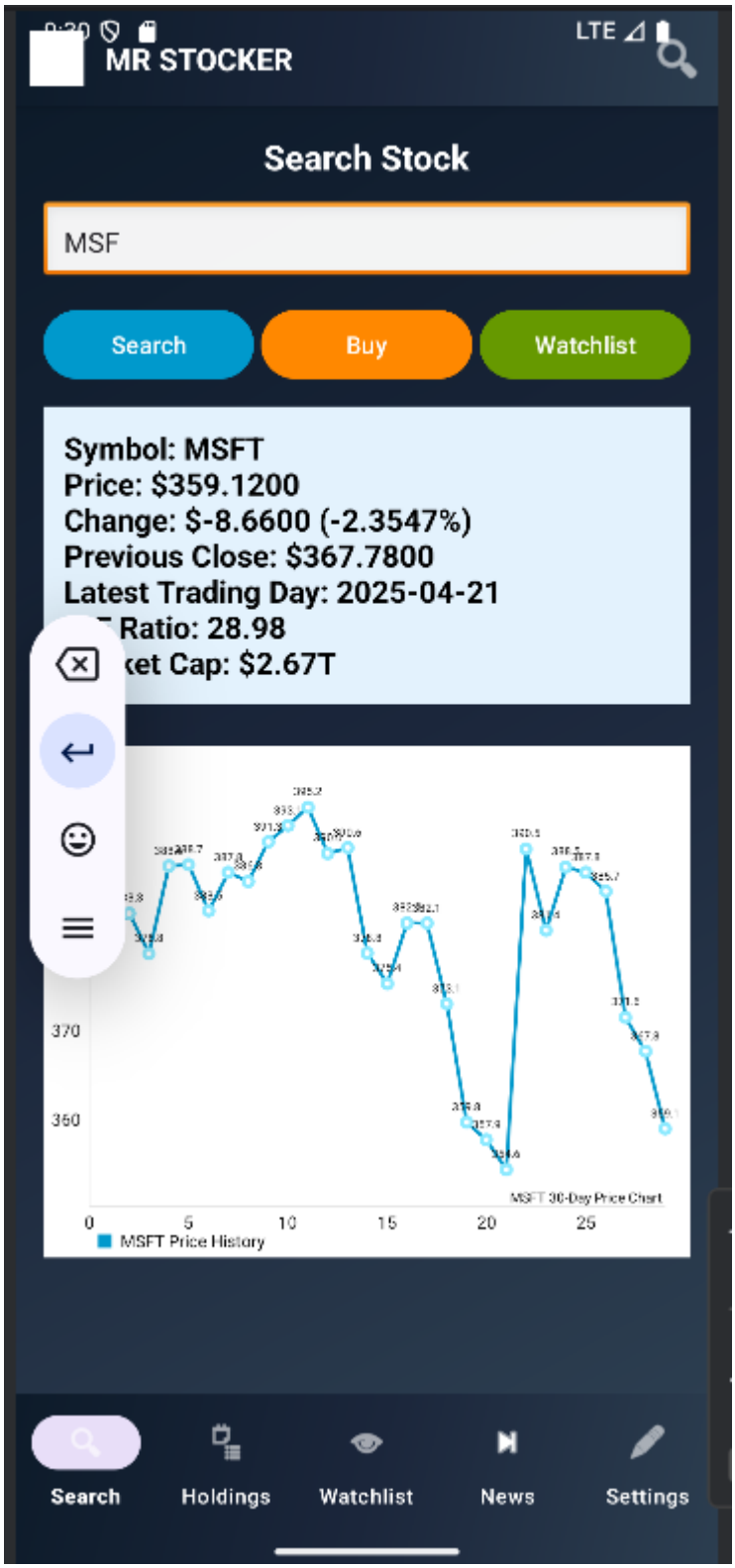
![Search Screen](https://example.com/search- Real-time stock quote retrieval

- Historical price data visualization with line charts
- Buy functionality to add stocks to holdings
- Add to watchlist option

Important code snippet:

```
Java private void fetchStockData(String symbol) { tvResult.setText("Loading...");
Call<GlobalQuoteResponse> quoteCall = apiService.getStockQuote("GLOBAL_QUOTE", symbol,
API_KEY);
quoteCall.enqueue(new Callback<GlobalQuoteResponse>() {
    @Override
    public void onResponse(Call<GlobalQuoteResponse> call, Response<GlobalQuoteResponse>
response) {
        if (response.isSuccessful() && response.body() != null) {
            StockQuote quote = response.body().getGlobalQuote();
            fetchCompanyOverview(symbol, quote);
        } else {
            tvResult.setText("Error fetching quote data");
```

```
        }
      }
    });
}
```



Holdings Module

The Holdings module enables users to track their stock purchases and monitor portfolio performance. It displays a list of owned stocks with quantity, purchase price, and current value.

![Holdings Screen](https://example.com/holdings

- Portfolio summary with total investment value
- Profit/loss calculation with percentage change
- Sell functionality to remove stocks from holdings
- Refresh button to update current prices

The portfolio summary section calculates and displays performance metrics:

java

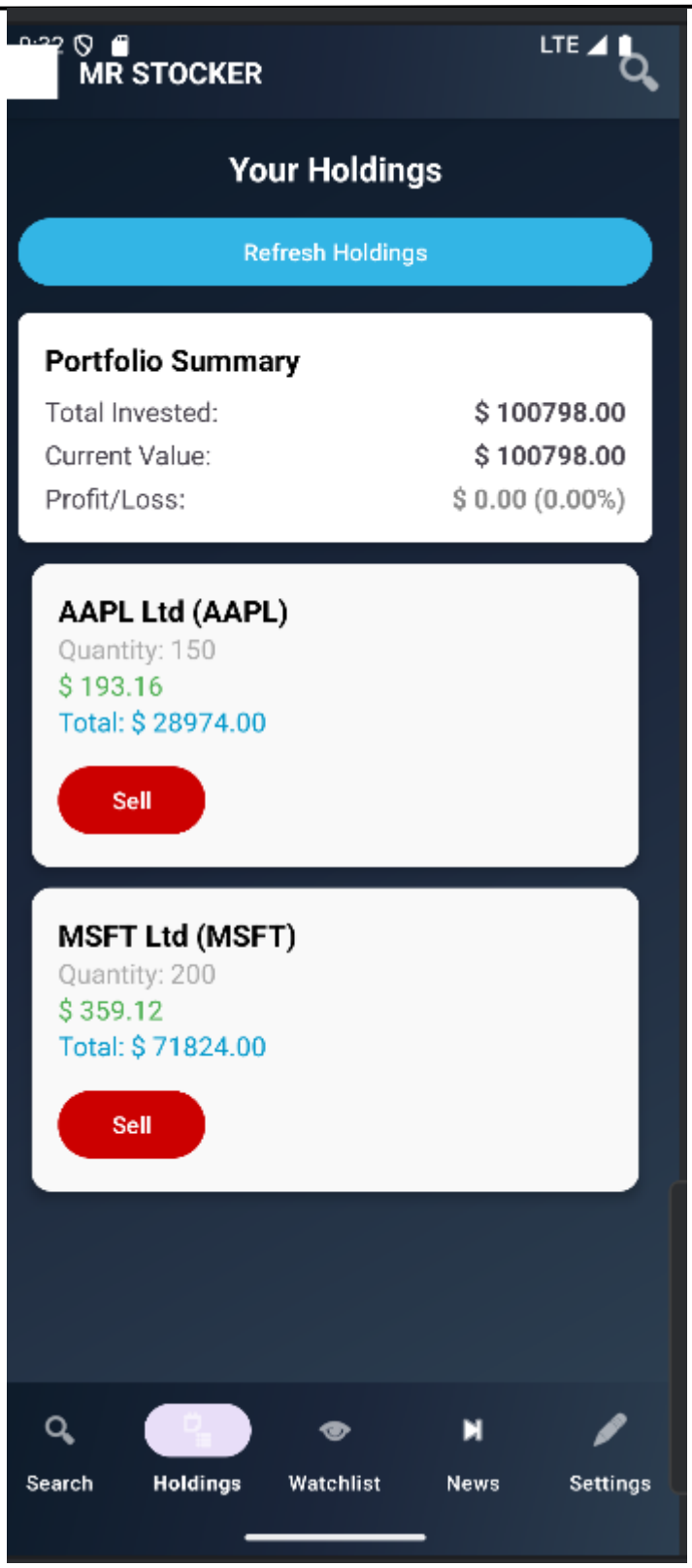
```
private void updatePortfolioSummary(List<HoldingStock> holdings) {
    double totalInvested = 0.0;
    double currentValue = 0.0;

    for (HoldingStock stock : holdings) {
        totalInvested += stock.getPricePerStock() * stock.getQuantity();
        currentValue += stock.getTotalValue();
    }

    double profitLoss = currentValue - totalInvested;
    double profitLossPercentage = (totalInvested > 0) ?
        (profitLoss / totalInvested) * 100 : 0;

    tvTotalInvested.setText(String.format("$ %.2f", totalInvested));
    tvCurrentValue.setText(String.format("$ %.2f", currentValue));
    tvProfitLoss.setText(String.format("$ %.2f (%.2f%%)",
        profitLoss, profitLossPercentage));

    if (profitLoss > 0) {
        tvProfitLoss.setTextColor(Color.parseColor("#4CAF50"));
    } else if (profitLoss < 0) {
        tvProfitLoss.setTextColor(Color.parseColor("#F44336"));
    }
}
```



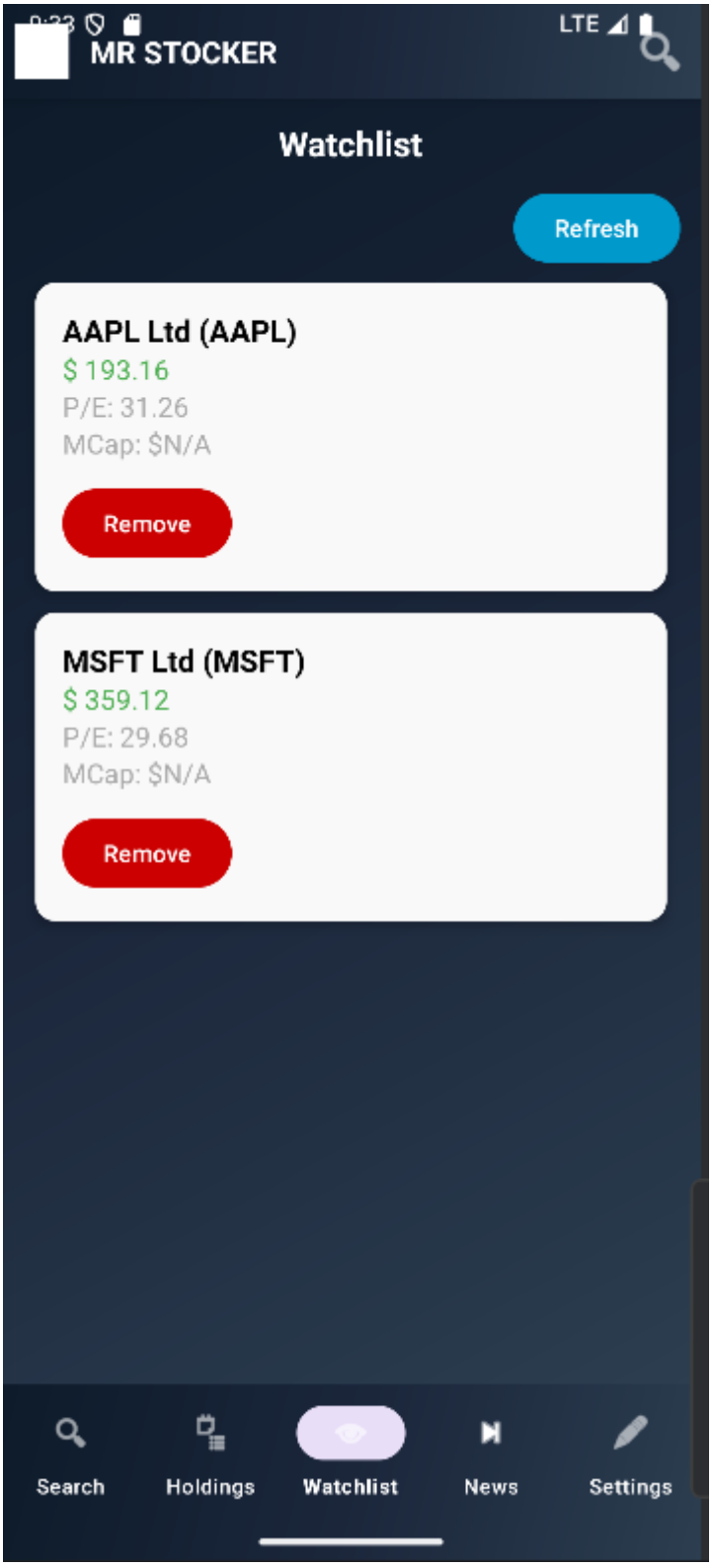
Watchlist Module

The Watchlist module allows users to save and monitor stocks of interest without purchasing them. It displays a list of tracked stocks with current prices and key metrics.

![Watchlist Screen](https://example.com/ includes:

- Quick view of stock prices and P/E ratios

- One-tap removal of stocks from watchlist
- Automatic refresh when the tab is selected



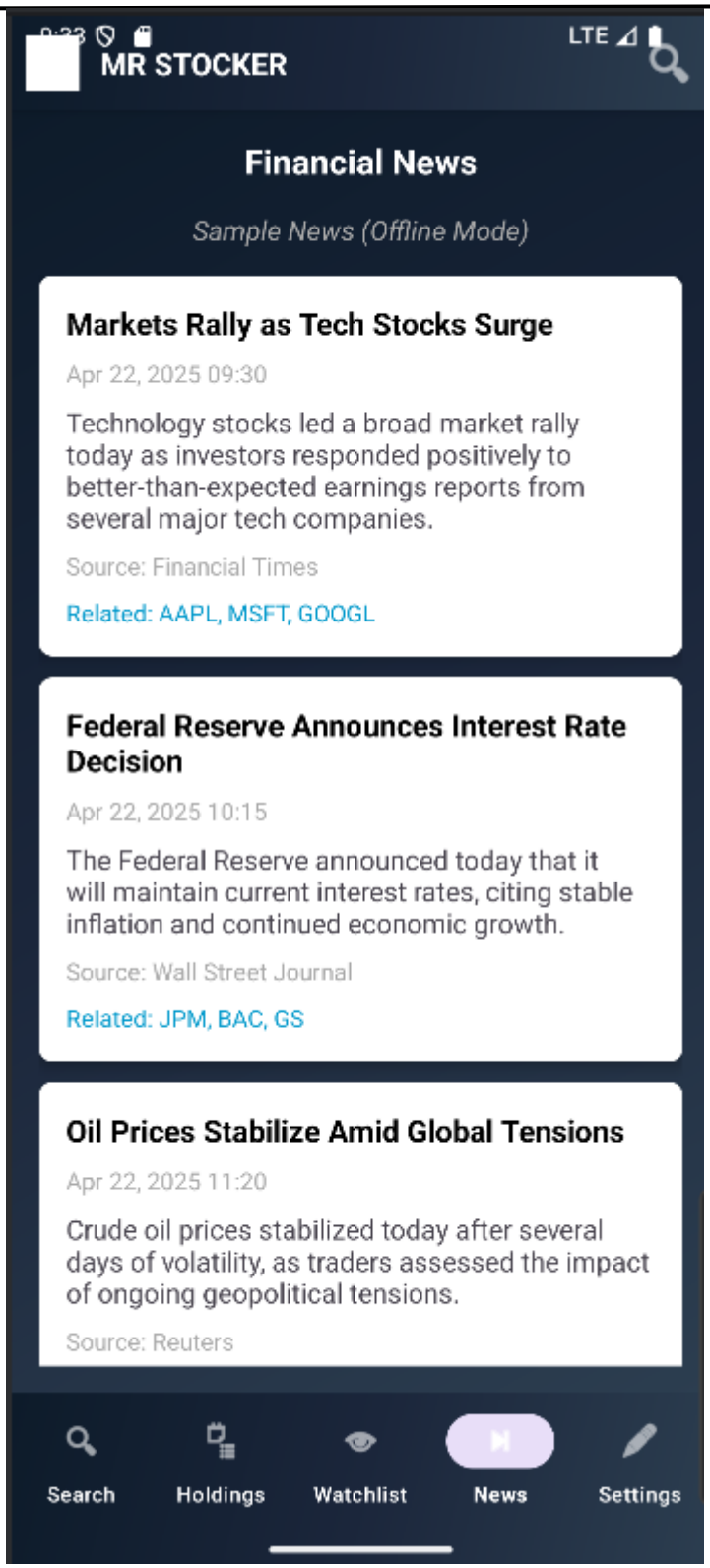
News Module

The News module provides financial news relevant to markets and specific stocks. It displays headlines,

publication dates, and sources with the ability to filter by stock symbol.

![News Screen](https://example.includes:

- General market news display
- Stock-specific news filtering
- Pull-to-refresh for latest updates
- Related symbols for each news item



7.DATABASE DESIGN

The application uses Room Persistence Library with two main entities:

WatchlistStock Table

- **symbol** (TEXT, Primary Key): Stock ticker symbol
- **name** (TEXT): Company name
- **price** (REAL): Current stock price
- **peRatio** (REAL): Price-to-earnings ratio
- **lastUpdated** (TEXT): Timestamp of last update
- **marketCap** (TEXT): Market capitalization

Sample data:

symbol	name	price	peRatio	lastUpdated	marketCap
AAPL	Apple Inc.	175.34	28.76	2025-04-22T14:30:00	2.74T
MSFT	Microsoft Corporation	412.65	35.21	2025-04-22T14:30:00	3.12T

HoldingStock Table

- **symbol** (TEXT, Primary Key): Stock ticker symbol
- **name** (TEXT): Company name
- **pricePerStock** (REAL): Purchase price per share
- **quantity** (INTEGER): Number of shares owned
- **purchaseDate** (TEXT): Date of purchase

Sample data:

symbol	name	pricePerStock	quantity	purchaseDate
GOOGL	Alphabet Inc.	142.56	10	2025-03-15T10:30:00
AMZN	Amazon.com Inc.	178.25	5	2025-04-01T09:45:00

8. Testing and Output

Testing Methodology

- **Unit Testing**: Tested individual components like API calls and database operations
- **Integration Testing**: Verified interactions between modules
- **UI Testing**: Manually tested user interface on multiple devices

- **Performance Testing**: Monitored app performance with different data loads

Sample Outputs

The application successfully retrieves and displays stock information, manages user holdings, and shows relevant financial news. Screenshots in the Modules section demonstrate the expected outputs.

Known Issues

- API rate limiting can occasionally cause delays in data retrieval
- News module requires internet connectivity with no offline caching
- Limited historical data available in the free API tier

8. CONCLUSION

Learning Outcomes

Developing this stock market tracker app provided valuable experience in working with financial APIs, implementing complex UI components, and managing local databases. I gained insights into effective data visualization techniques for financial information and learned to handle real-time data updates efficiently.

Challenges Faced

- Managing API rate limits while maintaining a responsive user experience
- Implementing accurate profit/loss calculations with changing market prices
- Creating an intuitive UI that presents complex financial data clearly
- Handling network connectivity issues gracefully

Future Scope

The application could be enhanced with several additional features:

- Advanced portfolio analytics with diversification metrics
- Stock alerts and price notifications
- Integration with brokerage APIs for actual trading
- Expanded historical data analysis with technical indicators
- Social features to share investment ideas
- Cryptocurrency tracking alongside traditional stocks

9.REFERENCE

- Android Developer Documentation: https://developer.android.com/docs
- Alpha Vantage API Documentation: https://www.alphavantage.co/documentation/
- EODHD API Documentation: https://eodhd.com/financial-apis/
- MPAndroidChart Library: https://github.com/PhilJay/MPAndroidChart
- Room Persistence Library: https://developer.android.com/training/data-storage/room
- Material Design Guidelines: https://material.io/design
- StackOverflow: Various solutions for Android development challenge.