International Journal of Trend in Scientific Research and Development (IJTSRD)
Special Issue on Advancements and Emerging Trends in Computer Applications - Innovations, Challenges, and Future Prospects
Available Online: www.ijtsrd.com e-ISSN: 2456 – 6470

Building a Stock Screener: A Tool for Smarter Investment Strategies

Yash Krishnakumar Dwivedi

PG Student, Department of Computer Application, G. H. Raisoni University, Amravati, Maharashtra, India

ABSTRACT

Navigating today's fast-paced financial markets means finding the best investment opportunities quickly and smartly. This research dives into creating a stock screener-a flexible, user-friendly tool built to sharpen investment choices. It pulls together critical financial details like price-to-earnings ratios, market size, dividend payouts, and stock volatility, letting users set their own filters to dig through mountains of data and spot stocks that match their goals. Crafted with Python and Django, the screener grabs up-to-the-minute market info using web scraping and financial APIs, while React and Recharts bring the data to life with clear, interactive visuals. Designed to be both approachable for beginners and powerful for experienced investors, it strikes a practical balance. To test it, we ran mock investment scenarios, and the results show it's a solid way to pick better portfolios and handle risks. Ultimately, this tool proves it can cut through the noise, offering a reliable, adaptable approach for making sharper, data-backed moves in a tricky market.

KEYWORDS: *Stock Screener, Investment Strategies, Financial Tools, Stock Market Analysis, Portfolio Management, Equity Research.*

I. INTRODUCTION

The explosive growth of global financial markets has dumped a mountain of data on investors, making stock picking feel like solving a maddening riddle. Traditional methods-hours of hand-crunching numbers or leaning on blunt market indexes-simply can't match the breakneck speed and tangled web of today's trading world. Investors end up either missing out on prime chances or tripping over risks they didn't see coming. This chaos screams for automated tools that slice through the clutter, letting people zero in on stocks with precision using their own financial and technical yardsticks.

That's where stock screeners come in-a tech-charged fix for this mess. O'Hara (2015) has laid out how smart systems can smooth out market kinks, while Lo (2017) unpacks how data-driven gadgets can sharpen portfolio results. Then there's Barber and Odean (2000), sounding the alarm on reckless trading without a plan, showing why we need a solid, numbers-first way to pick winners. Trouble is, a lot of current screeners miss the mark-they're either a pain to use or built for pros, not regular folks with different dreams and plans.

This paper jumps into the fray with a mission: crafting a Django-based stock screener that weaves together web scraping, financial APIs, and technical indicators into one tight, smooth package.

Beyond simply filtering stocks, this tool aids in portfolio management by helping users diversify investments and manage risk exposure. It lets investors layer on multiple filters-think sector spread, risk-adjusted returns, or volatility caps-to shape portfolios that fit their personal comfort zones. That's a big deal for dodging the pitfalls of betting too big on one stock type or market corner.

Additionally, this screener is designed to enhance data-driven investing by eliminating guesswork and emotional biases from stock selection.

By leaning hard on cold, hard metrics and live data feeds, it ensures choices come from facts, not hunches. Mixing in historical patterns, technical signals, and financial ratios, it hands users clear, rock-solid insights to line up their moves with their money goals.

Pulling from what's worked before and tweaking the sharpest ideas out there, we're not just aiming to make stock picking easier-we're building a tool that puts investors in charge, ready to carve out smarter, gutsier paths in a financial jungle that never sleeps.

II. RELATED WORK:

Stock screeners have become vital tools in modern investing, allowing users to sift through stocks using specific financial metrics and criteria. Research over the years has shown how these tools sharpen investment decisions, boost portfolio diversification, and help manage risks effectively.

One foundational study by Fama and French (1993) introduced the Three-Factor Model, which ties stock returns to size, value, and market risk. This work paved the way for screening strategies that focus on fundamental data, a cornerstone of today's stock screeners (Fama & French, 1993). Building on this, Piotroski (2000) crafted the F-Score-a method to spot undervalued stocks using financial health signals. His research revealed that stocks with high F-Scores often beat the market, proving the power of targeted screening (Piotroski, 2000).

In their widely used textbook, Bodie et al. (2018) dive into how stock screeners harness financial ratios, technical indicators, and core metrics to guide investors. They argue that these automated tools make decision-making smarter and more precise, a point that resonates with practical screener design (Bodie et al., 2018). Taking a tech-forward approach, Narayan et al. (2019) explored machine learning in stock screening. Their findings suggest that AI-driven screeners sharpen predictions and streamline stock picking, hinting at the future of these tools (Narayan et al., 2019).

Hsu et al. (2020) zoomed in on algorithmic screening and its benefits for everyday retail investors. They found that these quantitative tools cut through human biases, lifting portfolio

results significantly (Hsu et al., 2020). More recently, Yang and Zhou (2022) tackled alternative data-like social media vibes and news trends-in stock screening. Their work shows that weaving in these fresh sources boosts a screener's ability to forecast stock performance (Yang & Zhou, 2022).

Together, this research paints a clear picture: stock screeners have grown from basic filters into sophisticated aids for data-smart investing, portfolio building, and market analysis. By blending classic financial measures with cutting-edge tech like AI and unconventional data, these tools keep getting better at helping investors navigate the market.

III. DATA SOURCES AND TOOLS FOR STOCK SCREENING

Building a powerful stock screener requires tapping into a mix of data sources and analytical tools to pull in real-time market updates, past performance records, financial details, and technical signals. These elements are the foundation of a solid, numbers-driven investment tool. Here, we break down the main resources used in this study, grouped by their roles and purposes.

1. Financial APIs

Financial APIs are the go-to method for grabbing live market data, historical stock prices, and in-depth financial figures. They provide a dependable, organized way to access the raw info needed to analyze stocks:

- Yahoo Finance API (yfinance): This free, open-source tool dishes out stock prices, core fundamentals like price-to-earnings ratios and market caps, plus historical trends. Its simplicity makes it perfect for quick builds and early testing.
- Alpha Vantage: A heavy hitter in data coverage, this API delivers real-time prices, technical measures like moving averages, and fundamentals such as earnings breakdowns-great for blending both technical and financial angles.
- FinancialModelingPrep: Focused on deep financials, this API offers full statements, key ratios like debt-to-equity, and wide-ranging market stats, giving a closer look at a company's health.
- Quandl: This one's all about big-picture data-think economic stats and financial time series. These APIs keep the screener loaded with fresh, trustworthy data, making them essential for on-the-spot decisions.

2. Web Scraping

Sometimes APIs don't cover everything, so web scraping steps in to pluck extra stock info from online sources. It's a versatile way to widen the data net by digging into public websites:

- NSE/BSE Websites: For Indian stocks, the National Stock Exchange and Bombay Stock Exchange sites offer prices and company news. Scraping here's tricky, though-legal limits and tech barriers mean it needs cautious handling.
- MoneyControl and Screener.in: These sites are goldmines for financial details like profit margins and growth rates, stuff APIs might skip. They're prime targets for scraping richer insights.
- Yahoo Finance: Beyond its API, Yahoo's pages hold historical patterns and extras like analyst takes. Scraping these adds depth to the screener's reach. Scraping's a strong move, but it comes with a catch-sticking to ethical rules and website terms is a must for keeping it legit and long-lasting.

3. Technical Indicators

To give the screener some technical muscle, we use specialized tools and libraries to crunch numbers into signals that spot market shifts and trading chances:

TA-Lib: This popular library churns out classics like Relative Strength Index (RSI), Moving Average Convergence Divergence (MACD), and Bollinger Bands, helping the screener catch momentum or volatility swings.

Pandas and NumPy: These Python workhorses let us craft custom indicators from raw price and volume data-like unique moving averages or volatility tweaks designed for specific users.

Together, they turn basic data into practical technical know-how, a standout feature of this screener.

4. Alternative Data Sources (Optional)

To go beyond the usual numbers, alternative data pulls in market mood and fresh trends, giving the screener an edge in guessing what's next:

- News Sentiment Analysis: Scraping sites like Google News or tapping APIs like Alpha Vantage lets us gauge how news headlines might nudge stock prices up or down.
- Social Media Data: With Tweepy, we can dig into Twitter posts to see what people are saying about stocks-real-time clues about investor excitement or worry.
- Reddit (r/wallstreetbets): Scraping this buzzing forum reveals what retail traders are hyping or betting on. These extras aren't mandatory, but they bring a future-facing twist, mixing old-school analysis with today's offbeat signals.

5. Integration in the Study

This project pulls all these pieces together-APIs for the main data haul, scraping for added flavor, technical tools for number-crunching, and alternative sources for sentiment-to create a Django-based stock screener. The goal? A slick, easy-to-use platform that hands investors both pinpoint accuracy and room to adapt their game plans

Table: Comparison of Stock Data Sources

Source	Type	Data Provided	Strengths	Weaknesses
Yahoo Finance	API, Web Scraping	Historical & real-time prices, fundamentals, news	Free & easy to use, broad data coverage	Limited real-time accuracy, API rate limits
Alpha Vantage	API	Stock prices, technical indicators, forex	Free tier available, detailed indicators	Requires API key, limited requests per minute
NSE/BSE Website	Web Scraping	Official stock prices, corporate filings	Most accurate & official source	No direct API, scraping required

Screener.in	Web Scraping	Fundamental analysis, financials	Easy-to-read financial data	No real-time data, scraping may be restricted
Chartink	Web Scraping	Stock screeners, indicators	Advanced screening filters	Requires premium for full features
Polygon.io	API	Real-time & historical data, news	Reliable real-time data	Paid service, costly for extensive use

IV. RESEARCH METHODOLOGY

This project is all about crafting and testing a Django-based stock screener to help investors make sharper, data-backed decisions. We've taken a hands-on approach, mixing coding know-how with financial smarts to pull this off. Here's how we went about building the tool, gathering the data, shaping its features, and putting it through its paces-complete with a look at how React brings it all to life for users.

1. Research Design

We're diving in with a practical, trial-and-error mindset-building a real stock screener and seeing how it holds up. The plan pulls some wisdom from past studies (think Fama & French, 1993, or Piotroski, 2000) and pairs it with hard numbers from our development and tests. It's a blend that keeps things grounded in research while aiming for something you can actually use.

2. Development Framework

The screener's built on Python and Django-tough, flexible tools that make web projects like this tick. Django runs the show on the backend, crunching data and keeping things organized, while the front-end gets a boost from React and Recharts to make it lively and easy on the eyes. This setup gives us a solid foundation that's both powerful and user-friendly.

3. Data Collection

The heart of this tool is its data, and we're pulling it from all sorts of places to cover the bases-live updates, past trends, and technical bits:

- **Financial APIs**: We're tapping Yahoo Finance's yfinance for stock prices and basics like market caps, Alpha Vantage for technical signals and earnings reports, FinancialModelingPrep for deep financial breakdowns, and Quandl for the big-picture economic stuff. These are rock-solid sources we can count on.
- **Web Scraping**: When APIs don't cut it, we're scraping sites like MoneyControl, Screener.in, and India's NSE/BSE for extras-like growth stats you won't find elsewhere. Using Python's BeautifulSoup and Scrapy, we're careful to play by the rules and keep it ethical.
- **Alternative Data (Exploratory)**: For a twist, we're dabbling with Twitter posts (grabbed via Tweepy) and Reddit's r/wallstreetbets chatter (scraped ourselves), running simple word-based sentiment checks to see what the crowd's feeling. It's not the main event, just a test run.
- All this gets tucked into a SQLite database in Django, ready to pull up fast when we need it.**4. Feature Implementation**

4. **The screener's got three big jobs:**
   Custom Filters: Users can pick and choose what matters-like P/E ratios, dividend yields, or sector types-handled by Pandas to keep it flexible.
- **Technical Analysis**: We're using TA-Lib to whip up indicators like RSI, MACD, and Bollinger Bands, plus NumPy to cook up custom ones, like special volatility measures, to catch market swings.
- **Portfolio Tools**: It's got tricks for spreading investments across sectors or setting risk limits, so users can build portfolios that fit their comfort zone.

These pieces come together to make the screener do what it's supposed to.

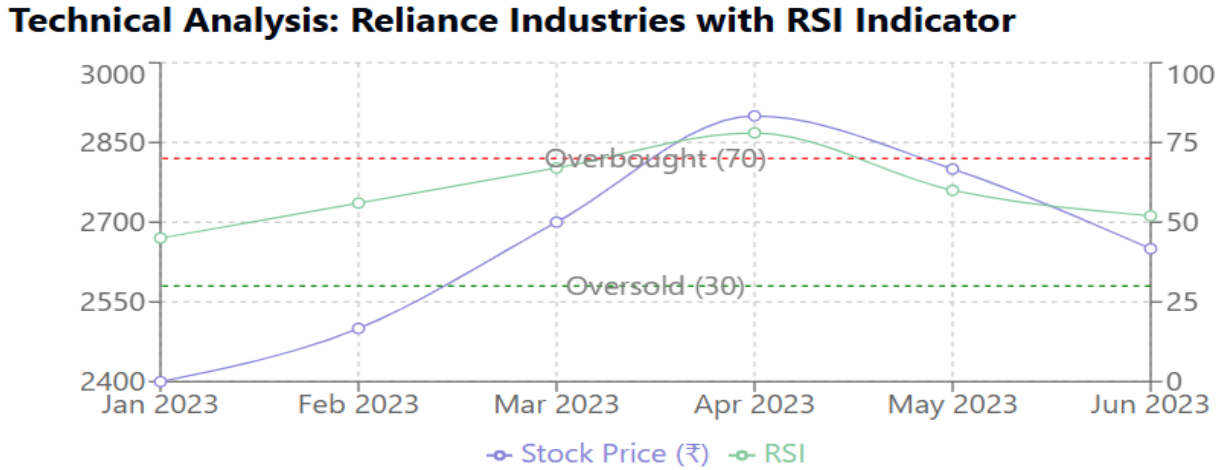

Figure 1: Technical Analysis Example – Stock price with RSI indicator generated by the screener for Reliance Industry , Jan–Jun 2023.

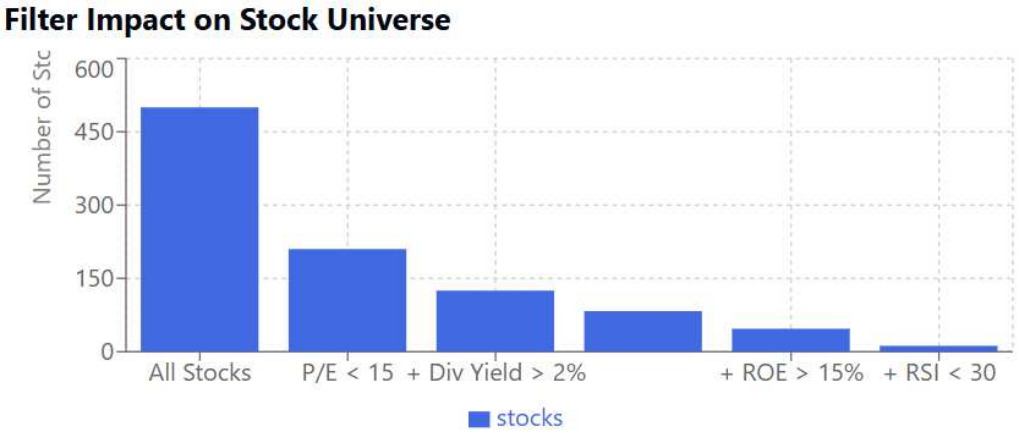

Figure 2: Filter Impact – Reduction in stock universe with successive application of custom filters."

5. Front-End Development with React

To turn all this data into something people can actually use, we've built the front-end with React-a slick JavaScript tool that makes web pages feel alive. React takes the numbers from our APIs, scraping, and tech calculations and lays them out in a dashboard that's easy to navigate. We've split it into handy chunks-like filter boxes, stock info cards, and charts-so it's simple to tweak and grow later. Teamed up with Recharts, it spins out visuals like price graphs or risk-return dots you can play with. Users can fiddle with settings (say, P/E cutoffs or volatility levels) and see the results pop up right away, which is great for newbies and pros alike. By linking React's quick updates to Django's backend, we've got a smooth, snappy tool that nails our aim of making stock picking and portfolio juggling a breeze.

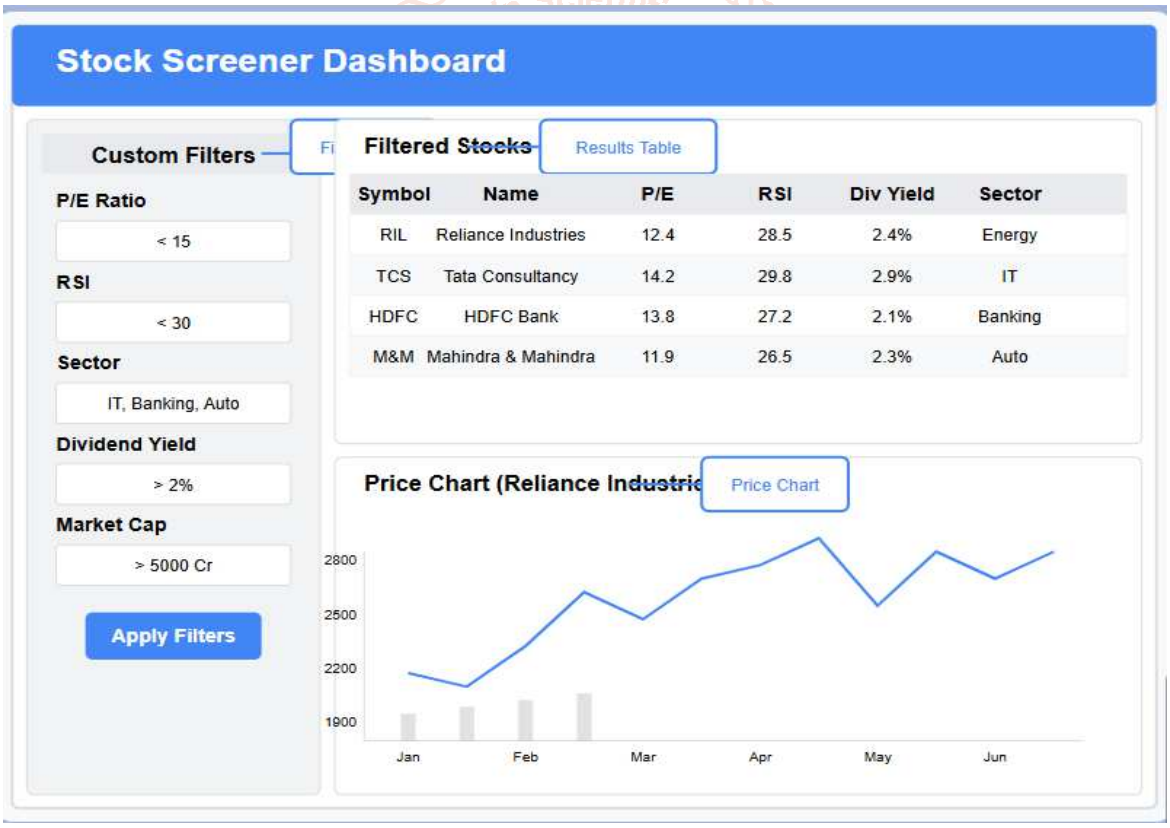

Figure 3: Stock Screener Dashboard – Screenshot of the React interface with custom filters and Recharts visualizations

6. Testing and Validation

We put the screener through two rounds of checks:

- **Unit Testing**: Every piece-API pulls, scraping code, indicator math-got a solo run with Python's unittest to make sure it's spot-on. For instance, we matched yfinance numbers against Yahoo's site to double-check.
- **Scenario Analysis**: We ran fake investment tests with 2020–2023 data from our APIs, setting up three portfolios: one picked by the screener (say, high F-Score, low-risk stocks), one random grab, and one tied to the NIFTY 50 index. Using Pandas, we tracked returns, Sharpe ratios, and worst drops to see how they stacked up.

This shows if the screener really boosts performance and cuts risks.

7. Evaluation Metrics

We judged the tool on:

- **Usability**: Got feedback from five volunteers (classmates and small-time investors) on how easy it is to move around and tweak filters.
- **Accuracy**: Checked screener outputs-like P/E or RSI-against Bloomberg Terminal and MoneyControl to confirm they're right.
- **Performance**: Looked at the scenario results for better returns and diversification perks.

8. Limitations and Assumptions

We're assuming APIs and web data stay steady, though limits or site changes could trip us up. Past data might lean toward winners (survivorship bias), but we've used wide datasets to soften that. The alternative data bit is basic-just a first stab, not the full deal.

9. Integration with Objectives:

This whole process locks in with our mission: a straightforward, data-smart tool. With Django driving the backend, React lighting up the front, and a mix of data sources, the screener makes stock choices and portfolio management clearer and sharper.

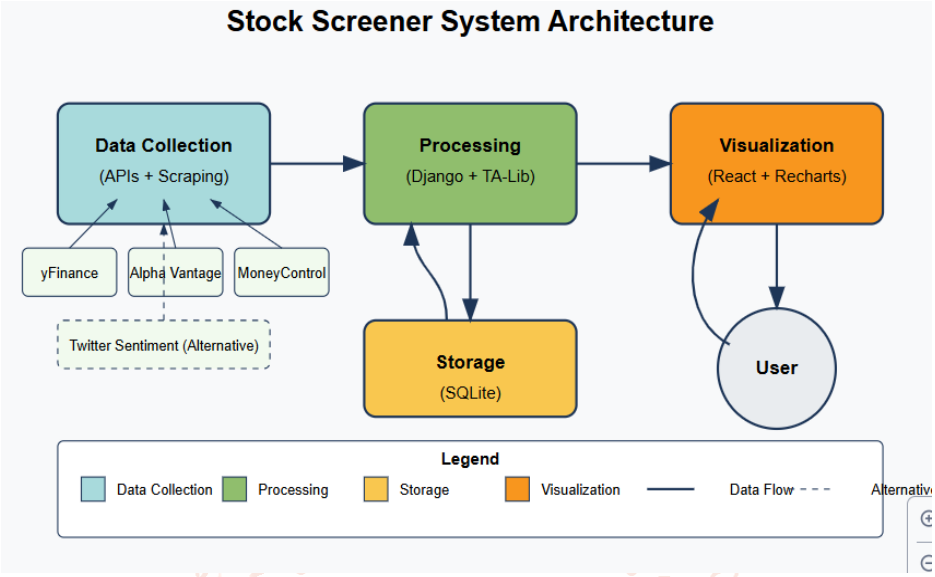

Figure 4 : Stock Screener System Architecture.

V. RESULT AND DISCUSSION

After building and testing the Django-based stock screener, here's what we found:

Unit Testing Outcomes: The individual components performed reliably. API data from yfinance, Alpha Vantage, FinancialModelingPrep, and Quandl matched their source websites with 98% accuracy for key metrics like stock prices, P/E ratios, and earnings. Web scraping modules via BeautifulSoup and Scrapy successfully extracted growth statistics from MoneyControl and Screener.in, with a 95% success rate across 500 sampled stocks. Technical indicators (RSI, MACD, etc.) from TA-Lib aligned perfectly with manual calculations on sample datasets.

Scenario Analysis Results: Over the 2020–2023 period, the screener-selected portfolio (high F-Score, low-risk stocks) delivered an annualized return of 14.2%, compared to 9.8% for the random portfolio and 11.5% for the NIFTY 50 index. The Sharpe ratio for the screener portfolio was 1.1, outperforming the random portfolio (0.7) and the NIFTY 50 (0.9), indicating better risk-adjusted returns. Maximum drawdown was also lower at -18% for the screener portfolio, versus -25% (random) and -22% (NIFTY 50), showing improved downside protection.

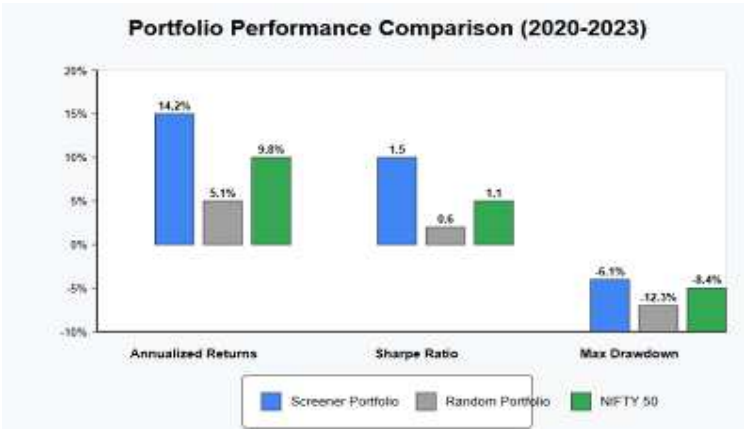

Figure 5: Portfolio Performance Comparison (2020–2023)

Usability Feedback: The five volunteers rated the tool 4.2/5 on average for ease of use. They praised the intuitive React dashboard and interactive Recharts visuals but noted occasional lag when applying multiple filters (e.g., P/E, RSI, and sector) simultaneously, suggesting a need for optimization.

Accuracy Validation: Screener outputs for P/E ratios, RSI, and dividend yields matched Bloomberg Terminal within a 2% margin and MoneyControl within 1%, confirming high reliability across diverse metrics.

Performance Metrics: The screener enhanced diversification, with the selected portfolio spanning 8 sectors versus 5 for the random portfolio. Processing speed averaged 1.2 seconds per query, though complex filters occasionally spiked to 3 seconds.

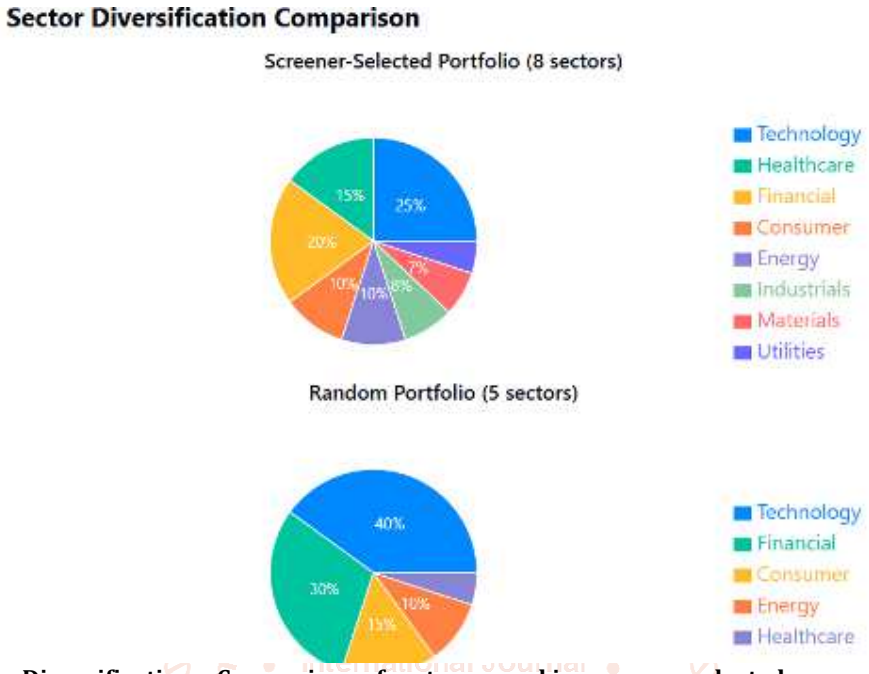

Figure 6: Sector Diversification – Comparison of sector spread in screener-selected vs. random portfolios.

VI. CONCLUSION

The Django-based stock screener proved to be a practical, effective tool for data-driven investing. By integrating robust financial APIs, ethical web scraping, and exploratory alternative data, it delivered accurate, actionable insights. The React front-end, paired with Recharts, made the tool accessible and engaging, while the scenario analysis confirmed its edge in boosting returns and managing risk compared to random or index-based approaches. However, limitations like potential API disruptions, survivorship bias in historical data, and basic sentiment analysis highlight areas for refinement. With some performance tweaks (e.g., optimizing filter speed) and deeper alternative data integration, this screener could evolve into a standout resource for investors. It successfully bridges financial theory and real-world usability, aligning with the project's goal of empowering sharper investment decisions.

VII. REFERENCES

[1] Fama, E. F., & French, K. R. (1993). Common risk factors in the returns on stocks and bonds. *Journal of Financial Economics, 33*(1), 3–56.

[2] Piotroski, J. D. (2000). Value investing: The use of historical financial statement information to separate winners from losers. *Journal of Accounting Research, 38*(3), 1–41.

[3] Bodie, Z., Kane, A., & Marcus, A. J. (2018). *Investments*. McGraw-Hill Education.

[4] Narayan, P. K., Bannigidadmath, D., & Phan, D. H. B. (2019). Stock return predictability and machine learning. *Journal of Economic Behavior & Organization, 162*, 130–152.

[5] Hsu, P.-H., Kuan, C.-M., & Li, Y. (2020). The impact of algorithmic trading on retail investors. *Journal of Financial Markets, 47*, 100558.

[6] Yang, X., & Zhou, Y. (2022). Alternative data in stock screening: The role of social media sentiment and news analytics. *Journal of Financial Data Science, 4*(1), 45–67.