Prosiding Seminar Nasional Sains dan Teknologi Seri 02
Fakultas Sains dan Teknologi, Universitas Terbuka
Vol. 1 No. 2 (2024)
e-ISSN: 3047-6569

VISUALISASI ALGORITMA *SORTING* MENGGUNAKAN DJANGO PADA DATA ASET PERUSAHAAN BERDASARKAN KAPITALISASI PASAR

Surya Triyono[1*], Mas Febri Adithya[2], Afriza Fradifta Arya Putra[3], Muhammad Adde Dian Husri[4], Suharsono[5]
[1,2,3,4,5]Program Studi Teknik Informatika, Politeknik Negeri Pontianak, Pontianak
[5]Program Studi Teknologi Informasi, Politeknik Negeri Pontianak

**Penulis korespondensi: suryatriyonoxy@gmail.com*

ABSTRAK

Visualisasi algoritma *sorting* digunakan dalam menganalisa data yang bertujuan untuk mengetahui kinerja dan kompleksitas dari sebuah algoritma dalam bentuk visual. pada data aset perusahaan berdasarkan kapitalisasi pasar memiliki data yang sangat banyak dan kompleks untuk itu perlu sebuah algoritma untuk mengurutkan data *asset* berdasarkan kebutuhan yang diinginkan. Pada penelitian ini data yang digunakan adalah data *asset* perusahaan yang diambil dari *website* CompaniesMarketCap menggunakan teknik *web scraping* dengan *library Requests* dan *BeautifulSoup*. Tujuan penelitian ini adalah mengembangkan aplikasi yang menyediakan *platform* interaktif dimana pengguna dapat memahami data asset hasil hasil perbandingan visualisasi algoritma *sorting*. Algoritma yang digunakan dalam memvisualisasikan data asset tersebut adalah *Merge Sort, Bubble Sort,* dan *Shell Sort.* Pengembangan aplikasi menggunakan Bahasa Python dengan *Framework* Django, Tailwind, Chart.js untuk visualisasi grafis, dan SweetAlert untuk notifikasi. Hasil dari penelitian ini menunjukkan bahwa aplikai yang dikembangkan berhasil memvisualisasikan proses seorting secara interaktif dan informatif. Visualisasi yang ditampilkan memperlihatkan perbedaan kinerja dari algoritma *Merge Sort, Bubble Sort,* dan *Shell Sort* secara *realtime* pada data *asset* perusahan. Sehingga dapat disimpulkan bahwa aplikasi ini dapat membantu dalam memberikan informasi tentang proses dan kinerja dari algoritma *sorting* dalam mengurutkan data.

Kata kunci: Visualisasi, *Sorting*, Django, *Web scraping*, Algoritma.

1 PENDAHULUAN

Penggunaan algoritma *sorting* merupakan salah satu konsep fundamental dalam ilmu komputer yang digunakan untuk mengatur elemen-elemen dalam urutan tertentu berdasarkan kriteria yang telah ditentukan. Berbagai algoritma *sorting* seperti *Merge Sort*, *Bubble Sort*, dan *Shell Sort* memiliki karakteristik dan performa yang berbeda dalam menangani berbagai jenis data dan kasus. Penelitian mengenai algoritma *sorting* telah banyak dilakukan untuk memahami dan meningkatkan efisiensi proses *sorting* (Cormen et al., 2009). Studi meta oleh Hundhausen et al. (2002) menunjukkan bahwa visualisasi algoritma dapat membantu dalam memahami proses dan kinerja algoritma tersebut.

Kemajuan dalam teknologi informasi telah membuka peluang baru dalam pengembangan alat bantu pembelajaran yang interaktif dan efektif. Visualisasi algoritma *sorting* memberikan gambaran yang lebih jelas mengenai cara kerja algoritma dan perbandingan kinerja antar algoritma. *Framework* seperti Django, Tailwind, Chart.js, dan SweetAlert memungkinkan pengembangan aplikasi visualisasi yang interaktif dan mudah digunakan. Saraiya et al. (2004)

mengidentifikasi fitur-fitur visualisasi algoritma yang efektif dalam meningkatkan pemahaman pengguna tentang proses dan kinerja algoritma.
Penelitian ini berfokus pada pembuatan aplikasi visualisasi *sorting* menggunakan tiga algoritma *sorting* populer yaitu *Merge Sort*, *Bubble Sort*, dan *Shell Sort*. Data yang digunakan untuk pengujian aplikasi ini diambil dari situs companiesmarketcap.com menggunakan teknik *web scraping* dengan pustaka requests dan BeautifulSoup. Visualisasi algoritma *sorting* bertujuan untuk mempermudah pemahaman terhadap proses kerja algoritma tersebut dan perbandingan kinerja antara satu algoritma dengan yang lain.

Tujuan dari penelitian ini adalah untuk mengembangkan aplikasi visualisasi *sorting* yang interaktif dan informatif, yang dapat digunakan sebagai alat bantu pembelajaran dalam memahami dan menganalisis performa berbagai algoritma *sorting*. Aplikasi ini diharapkan dapat menjadi alat bantu yang efektif dalam pendidikan komputer serta meningkatkan pemahaman terhadap berbagai algoritma *sorting*.

2 METODE

2.1 Alat Penelitian

Dalam penelitian ini, terdapat beberapa alat dan bahan yang digunakan untuk mengembangkan dan mengimplementasikan aplikasi visualisasi *sorting* menggunakan *Framework* Django, Tailwind, Chart.js, dan SweetAlert serta JavaScript untuk membuat halaman web yang interaktif. Berikut daftar alat dan bahan yang digunakan.

a) *Hardware*: Laptop dengan spesifikasi minimum prosesor Intel i5, RAM 8GB.
b) Bahasa Pemrograman: Python, JavaScript, HTML dan CSS
c) *Sofware*: Visual Studio Code, Google Chrome
d) *Framework*: Django, Tailwind CSS, Chart.js dan SweetAlert
e) *Liblary*: Requests dan BeautifulSoup.
f) Data: Data diambil dari situs companiesmarketcap.com. Koneksi *Internet*: Diperlukan untuk proses *web scraping* dan pengembangan aplikasi.

2.2 Metode Penelitian

2.2.1 Metode Melalui Studi Literatur

Dalam metode ini, penulis mengumpulkan informasi dari jurnal, buku, artikel, dan sumber *online* yang relevan untuk memahami dasar-dasar, prinsip-prinsip, dan tren terbaru dalam pengembangan aplikasi visualisasi algoritma *sorting*. Tujuannya adalah untuk memperoleh pemahaman yang mendalam sebelum memulai pengembangan aplikasi.

2.2.2 Metode Eksperimen

Metode ini melibatkan serangkaian tahapan konkret yang dilakukan secara praktis untuk mengembangkan aplikasi visualisasi. Langkah-langkahnya yaitu sebagai berikut.

2.2.2.1 Pengumpulan Data

Pengumpulan data dalam penelitian ini dilakukan melalui teknik *web scraping*, dengan target data berupa informasi aset perusahaan berdasarkan kapitalisasi pasar dari situs CompaniesMarketCap. Proses *scraping* dilakukan menggunakan pustaka *Requests* dan BeautifulSoup. Hasil dari pengumpulan data adalah diperolehnya data asset Perusahaan berdasarkan kapitalisasi pasar, yang selanjutnya akan diurutkan menggunakan algoritma *Merge Sort*, *Bubble Sort*, dan *Shell Sort.*

2.2.2.2 Implementasi Django

Pengembangan aplikasi visualisasi *sorting* dilakukan menggunakan *Framework* Django. Aplikasi ini mencakup beberapa komponen utama seperti model untuk menyimpan data, *view* untuk menangani logika aplikasi, dan *template* untuk antarmuka pengguna. *Framework* Django dipilih karena memiliki tampilan berbasis *website* yang tergolong menarik dan mudah untuk memvisualisasikan langkah pengurutan dan visualisasi algoritma.

2.2.2.3 Analisa Hasil

Setelah implementasi, aplikasi diuji dengan melakukan beberapa pengurutan data menggunakan algoritma *Merge Sort*, *Bubble Sort*, dan *Shell Sort*. Hasil pengujian meliputi analisis kinerja yaitu waktu yang dibutuhkan untuk mengurutkan data dan kompleksitas dari setiap algoritma. Adapun tahapan metode eksperimen pada penelitian ini terdiri dari pengumpulan data, implementasi Django, dan analisis hasil yang diperoleh seperti pada **Gambar 1**.

Gambar 1 Tahapan Metode Eksperimen

3 HASIL DAN PEMBAHASAN

3.1 Hasil Data Aset Perusahan melalui Teknik Scraping

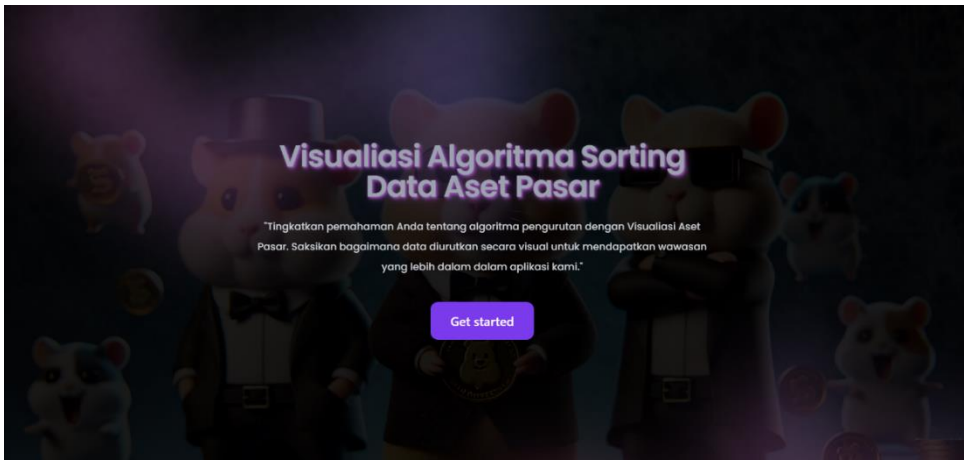
Hasil pengumpulan data yang dilakukan melalui teknik *web scraping* pada situs CompaniesMarketCap. Data yang dihasilkan mencakup informasi penting seperti peringkat perusahaan, nama, kapitalisasi pasar, harga saham, perubahan harga harian, dan negara asal. Seluruh data ini kemudian disimpan dalam *database* SQLite menggunakan model Django, memastikan integritas dan kemudahan akses data untuk proses selanjutnya. Hasil data aset dapat dilihat pada **Gambar 2**.

▸ **0**: {rank: '1', name: 'Gold\nGOLD', img_src: '/img/company-logos/64/GOLD.XM.png', market_cap: '$15.580 T', price: '$2,320', …}
▸ **1**: {rank: '2', name: 'Microsoft\nMSFT', img_src: '/img/company-logos/64/MSFT.png', market_cap: '$3.180 T', price: '$427.87', …}
▸ **2**: {rank: '3', name: 'NVIDIA\nNVDA', img_src: '/img/company-logos/64/NVDA.png', market_cap: '$2.995 T', price: '$121.79', …}
▸ **3**: {rank: '4', name: 'Apple\nAAPL', img_src: '/img/company-logos/64/AAPL.png', market_cap: '$2.961 T', price: '$193.12', …}
▸ **4**: {rank: '5', name: 'Alphabet (Google)\nGOOG', img_src: '/img/company-logos/64/GOOG.png', market_cap: '$2.171 T', price: '$176.63', …}
▸ **5**: {rank: '6', name: 'Amazon\nAMZN', img_src: '/img/company-logos/64/AMZN.png', market_cap: '$1.946 T', price: '$187.06', …}
▸ **6**: {rank: '7', name: 'Saudi Aramco\n2222.SR', img_src: '/img/company-logos/64/2222.SR.png', market_cap: '$1.830 T', price: '$7.57', …}
▸ **7**: {rank: '8', name: 'Silver\nSILVER', img_src: '/img/company-logos/64/SILVER.XM.png', market_cap: '$1.649 T', price: '$29.30', …}
▸ **8**: {rank: '9', name: 'Bitcoin\nBTC', img_src: '/img/company-logos/64/BTC.X.png', market_cap: '$1.342 T', price: '$67,935', …}
▸ **9**: {rank: '10', name: 'Meta Platforms (Facebook)\nMETA', img_src: '/img/company-logos/64/META.png', market_cap: '$1.274 T', price: '$502.60', …}
▸ **10**: {rank: '11', name: 'Berkshire Hathaway \nBRK-B', img_src: '/img/company-logos/64/BRK-B.png', market_cap: '$886.21 B', price: '$410.81', …}
▸ **11**: {rank: '12', name: 'TSMC\nTSM', img_src: '/img/company-logos/64/TSM.png', market_cap: '$872.20 B', price: '$168.16', …}
▸ **12**: {rank: '13', name: 'Eli Lilly\nLLY', img_src: '/img/company-logos/64/LLY.png', market_cap: '$822.10 B', price: '$865.00', …}
▸ **13**: {rank: '14', name: 'Broadcom\nAVGO', img_src: '/img/company-logos/64/AVGO.png', market_cap: '$667.54 B', price: '$1,440', …}
▸ **14**: {rank: '15', name: 'Novo Nordisk\nNVO', img_src: '/img/company-logos/64/NVO.png', market_cap: '$639.76 B', price: '$143.63', …}
▸ **15**: {rank: '16', name: 'JPMorgan Chase\nJPM', img_src: '/img/company-logos/64/JPM.png', market_cap: '$573.21 B', price: '$199.61', …}
▸ **16**: {rank: '17', name: 'Visa\nV', img_src: '/img/company-logos/64/V.png', market_cap: '$562.75 B', price: '$275.04', …}
▸ **17**: {rank: '18', name: 'Tesla\nTSLA', img_src: '/img/company-logos/64/TSLA.png', market_cap: '$554.25 B', price: '$173.79', …}
▸ **18**: {rank: '19', name: 'Walmart\nWMT', img_src: '/img/company-logos/64/WMT.png', market_cap: '$539.68 B', price: '$66.96', …}
▸ **19**: {rank: '20', name: 'Exxon Mobil\nXOM', img_src: '/img/company-logos/64/XOM.png', market_cap: '$507.26 B', price: '$113.08', …}
▸ **20**: {rank: '21', name: 'SPDR S&P 500 ETF Trust\nSPY', img_src: '/img/company-logos/64/SPY.png', market_cap: '$491.61 B', price: '$535.66', …}
▸ **21**: {rank: '22', name: 'UnitedHealth\nUNH', img_src: '/img/company-logos/64/UNH.png', market_cap: '$455.59 B', price: '$495.00', …}
▸ **22**: {rank: '23', name: 'Tencent\nTCEHY', img_src: '/img/company-logos/64/TCEHY.png', market_cap: '$450.44 B', price: '$47.63', …}
▸ **23**: {rank: '24', name: 'Ethereum\nETH', img_src: '/img/company-logos/64/ETH.X.png', market_cap: '$430.03 B', price: '$3,562', …}
▸ **24**: {rank: '25', name: 'Mastercard\nMA', img_src: '/img/company-logos/64/MA.png', market_cap: '$417.62 B', price: '$449.25', …}
▸ **25**: {rank: '26', name: 'ASML\nASML', img_src: '/img/company-logos/64/ASML.png', market_cap: '$416.15 B', price: '$1,042', …}

Gambar 2 Data-Data Aset Perusahan melalui Teknik Scraping

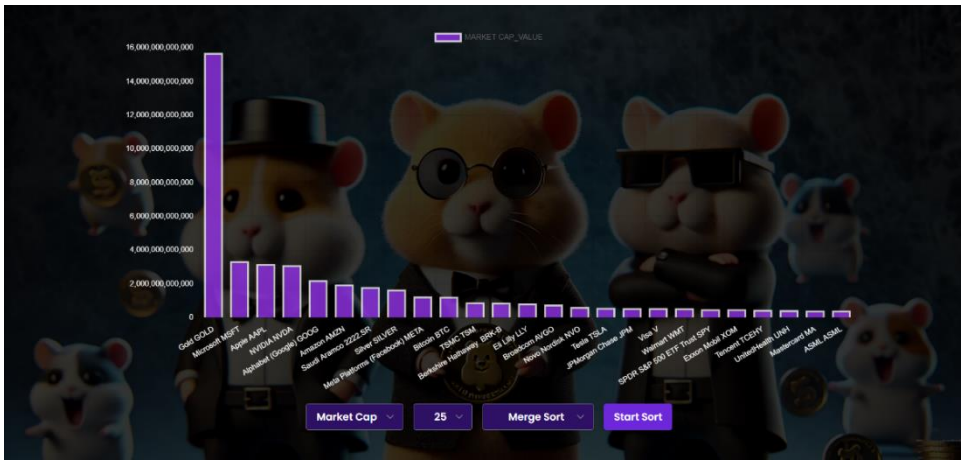
3.2 Implementasi Algoritma Sorting pada Django

Penerapan sorting untuk memvisualisasikan proses *sorting* yang diimplementasikan menggunakan *Framework* Django. Algoritma yang digunakan untuk pengurutan adalah algoritma *Merge Sort*, *Bubble Sort*, dan *Shell Sort.* Aplikasi *sorting* yang dibangun untuk mengurutkan data berdasarkan kapitalisasi pasar, harga saham, atau perubahan harga hari ini. Visualisasi proses *sorting* ditampilkan secara *real-time* dengan bantuan Chart.js, dengan demikian pengguna dapat mengamati perubahan urutan data secara langsung ketika proses pengurutan sedang terjadi. Tampilan antarmuka pengguna yang responsif dan modern dibangun menggunakan Tailwind CSS, dengan *dropdown* menu yang memungkinkan pengguna untuk memilih metode *sorting*, kriteria pengurutan, dan jumlah data yang ingin ditampilkan. Tampilan awal aplikasi pengurutan data asset Perusahaan dapat dilihat pada **Gambar 3**.



Gambar 3 Halaman *Home*

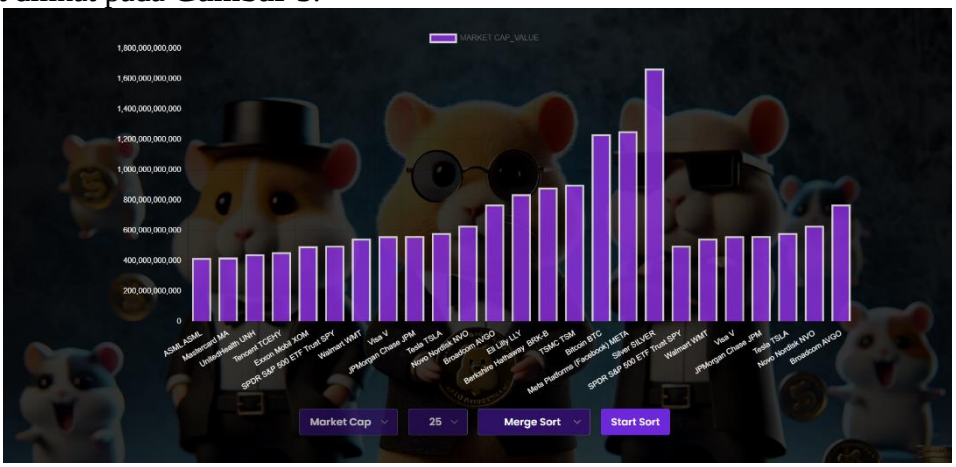
Pada **Gambar 3** pengguna dapat melihat proses pengurutan dengan menekan tombol *Get started*. Maka pengguna akan mendapatkan tampilan halaman *sorting* seperti pada **Gambar 4**.



Gambar 4 Halaman *Sorting*

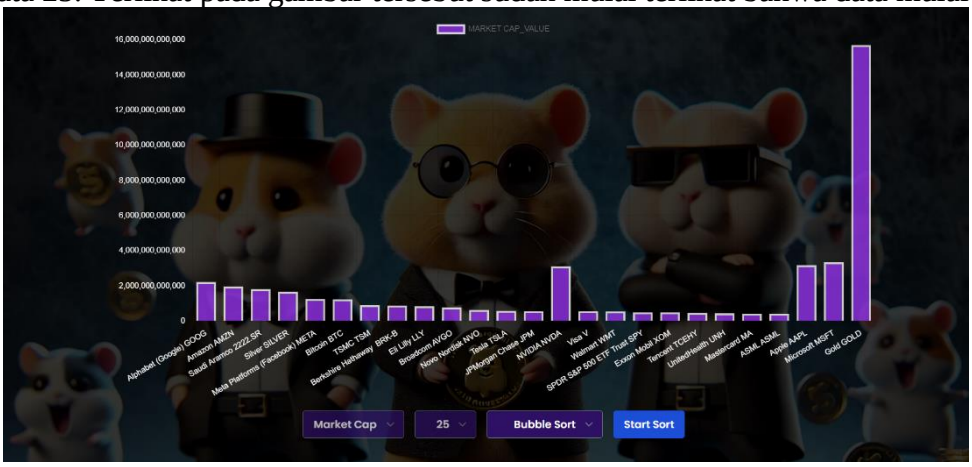
Halaman *sorting* seperti pada **Gambar 4** menunjukan data *asset* perusahaan yang terurut secara *descending* atau dari besar ke kecil. Pada halaman tersebut terdapat beberapa *menu* yaitu klasifikasi *asset*, jumlah data, pilihan algoritma, dan tombol *start sort* untuk memulai pengurutan. Ketika pengguna menekan tombol *Start Sort* maka proses pengurutan akan

dilakukan secara *real-time*. Proses pengurutan menggunakan salah satu algoritma yaitu *Merge Sort* dapat dilihat pada **Gambar 5**.



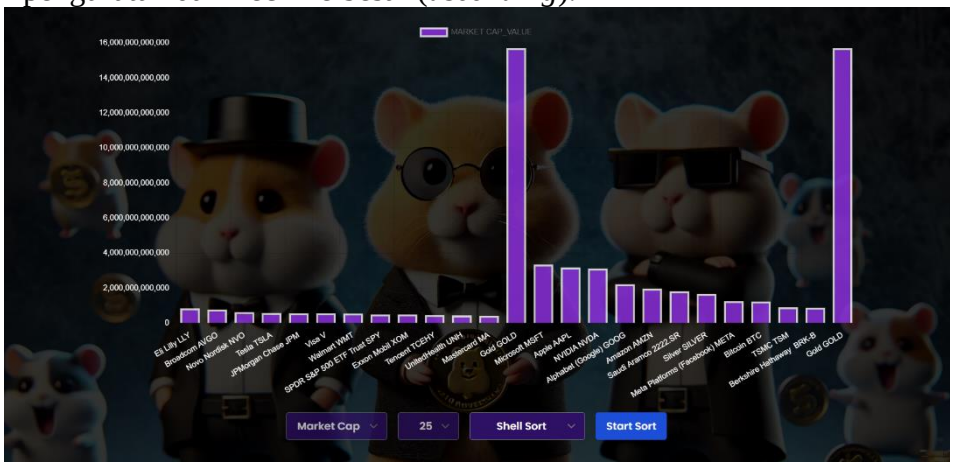
Gambar 5 Visualisasi *Merge Sort*

Pada **Gambar 5** memperlihatkan visualisasi algoritma yang pertama yaitu *Merge Sort* dengan jumlah data 25. Terlihat pada gambar tersebut sudah mulai terlihat bahwa data mulai terurut.



Gambar 6 Visualisasi *Bubble Sort*

Pada **Gambar 6** menunjukkan visualisasi algoritma yang kedua yaitu *Bubble Sort* dalam melakukan pengurutan dari kecil ke besar (*ascending*).

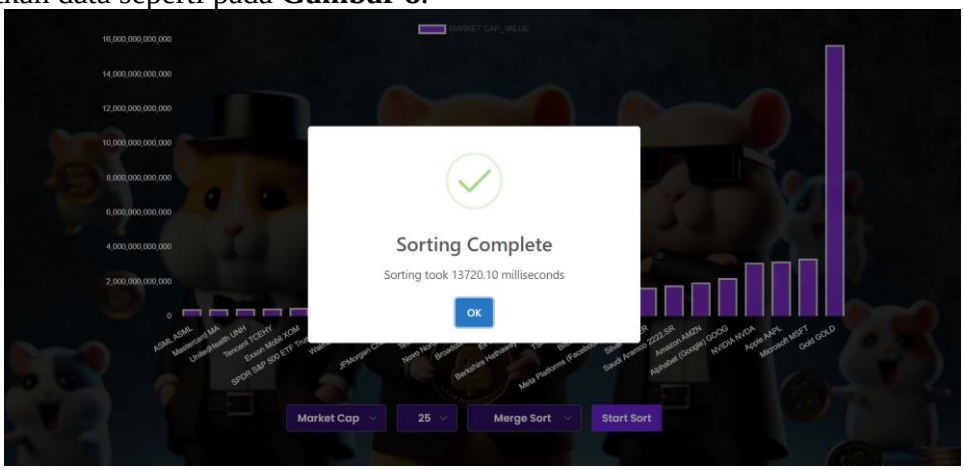


Gambar 7 Visualisasi *Shell Sort*

Pada **Gambar 7** ditampilkan visualisasi algoritma yang ketiga yaitu *Shell Sort*. Dimana jumlah sampel data yang digunakan sama dengan algoritma *Merge Sort*, *Bubble Sort.*

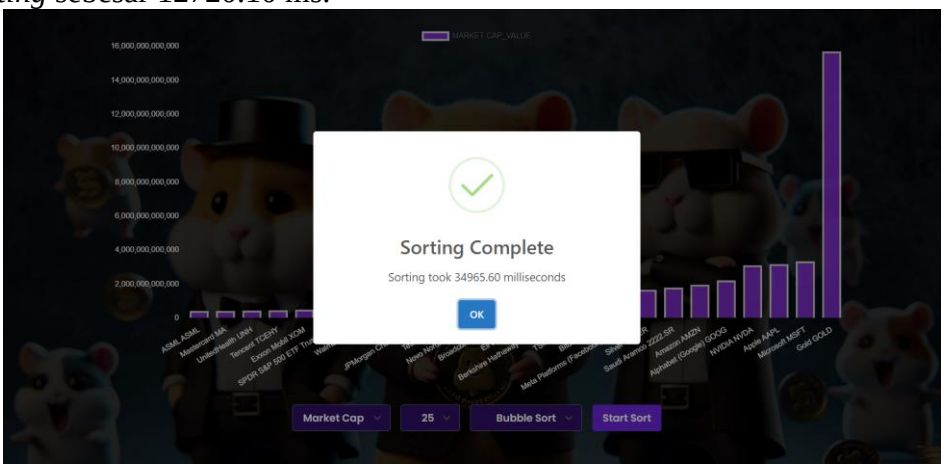
3.3 Analisis dan Hasil

Pengujian dilakukan untuk mengetahui kinerja dari tiga algoritma yang digunakan yaitu algoritma *Merge Sort*, algoritma *Bubble Sort*, dan algoritma *Shell Sort*, dengan menggunakan data aset perusahaan berdasarkan kapitalisasi pasar yang diperoleh melalui teknik *web scraping* dari situs CompaniesMarketCap. Analisis dilakukan dengan membandingkan waktu eksekusi dan kompleksitas algoritma dalam mengurutkan data berdasarkan kriteria kapitalisasi pasar. Selanjutnya setelah proses pengurutan akan dihasilkan waktu yang dibutuhkan dalam mengurutkan data seperti pada **Gambar 8**.



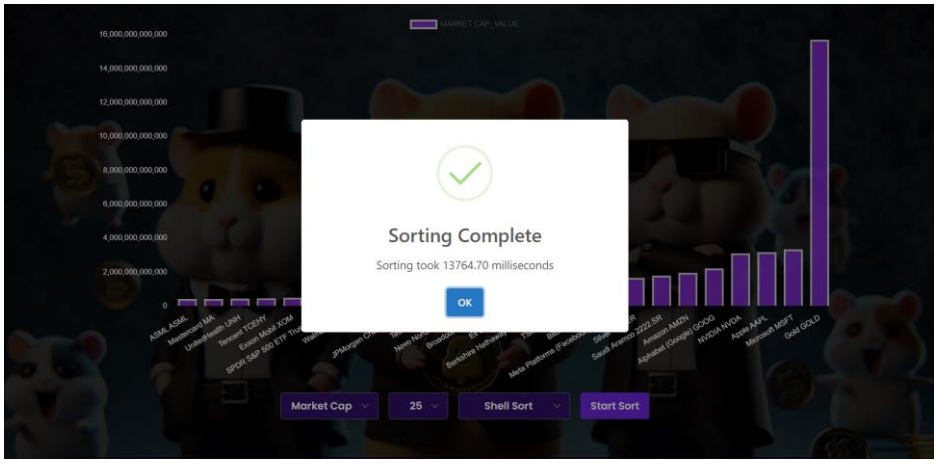
Gambar 8 Waktu Eksekusi *Merge Sort*

Pada **Gambar 8** waktu eksekusi algoritma *Merge Sort* dalam mengurutkan 25 data diperoleh waktu *sorting* sebesar 12720.10 ms.



Gambar 9 Waktu Eksekusi *Bubble Sort*

Pada **Gambar 9** menunjukkan waktu eksekusi algoritma *Bubble Sort* dalam mengurutkan 25 data diperoleh waktu *sorting* sebesar 34965.60 ms. Waktu yang di butuhkan oleh algoritma *Bubble Sort* jauh lebih besar dari algoritma *Merge Sort.*



Gambar 10 Waktu Eksekusi *Shell Sort*

Sedangkan **Gambar 10** ditampilkan waktu eksekusi algoritma *Shell Sort* dalam mengurutkan 25 data membutuhkan waktu sebesar 13764.70 ms. Algoritma *Shell Sort* memiliki waktu yang lebih sedikit dari algoritma *Merge Sort* dan *Bubble Sort*. Adapun hasil pengujian Algoritma *Merge Sort, Bubble Sort, dan Shell Sort* dapat dilihat pada **Tabel 1**.

Tabel 1 Hasil Pengujian Algoritma *Merge Sort, Bubble Sort, dan Shell Sort*

Algoritma	**Waktu Visualisasi (ms)**
Merge Sort	13099.40
Bubble Sort	46748.60
Shell Sort	12932.00

Pada **Tabel 1** dapat dijelaskan bahwa:
1. *Merge Sort*: Waktu eksekusi sebesar 13099.40 ms menunjukkan bahwa *Merge Sort* bekerja efisien dengan waktu eksekusi yang relatif cepat untuk dataset besar.
2. *Bubble Sort*: Waktu eksekusi sebesar 46748.60 ms menunjukkan bahwa *Bubble Sort* kurang efisien dengan waktu eksekusi yang jauh lebih lama dibandingkan dengan *Merge Sort* dan *Shell Sort*.
3. *Shell Sort*: Waktu eksekusi sebesar 12932.00 ms menunjukkan bahwa *Shell Sort* bekerja hampir seefisien *Merge Sort*, dengan waktu eksekusi yang sedikit lebih cepat.

Berdasarkan pengujian yang dilakukan terhadap tiga algoritma dengan menggunakan data yang sama bahwa *Merge Sort* dan *Shell Sort* lebih efisien dalam menangani *dataset* besar dibandingkan *Bubble Sort.* Algoritma *Shell Sort* jauh lebih efisien dibandingkan denagn algoritma *Merge Sort* dan *Bubble Sort* karena memiliki waktu eksekusi yang lebih sedikit.

4 KESIMPULAN

Penelitian ini berhasil membangun aplikasi visualisasi *sorting* yang interaktif dan informatif menggunakan *Framework* Django, Tailwind, Chart.js, dan SweetAlert. Aplikasi ini memungkinkan pengguna untuk mengamati proses dan kinerja algoritma *sorting* seperti *Merge Sort*, *Bubble Sort*, dan *Shell Sort* secara *real-time* pada data aset perusahaan yang diambil dari situs CompaniesMarketCap. Bahwa *Merge Sort* dan *Shell Sort* menunjukkan efisiensi waktu yang lebih baik dibandingkan dengan *Bubble Sort. Bubble Sort* memiliki waktu eksekusi yang jauh lebih lama dalam mengurutkan data aset perusahaan yang besar, sehingga kurang efisien untuk *data set* besar. Aplikasi ini memberikan gambaran yang jelas mengenai perbedaan

kinerja dan kompleksitas dari masing-masing algoritma *sorting*, sehingga dapat menjadi alat bantu yang efektif dalam pembelajaran dan analisis algoritma *sorting*.

UCAPAN TERIMAKASIH

Ucapan terima kasih disampaikan kepada Politeknik Negeri Pontianak, khususnya Program Studi Teknik Informatika, yang telah memberikan fasilitas dan dukungan dalam pelaksanaan penelitian ini. Terima kasih juga kepada rekan-rekan dan semua pihak yang telah membantu dalam pengumpulan data dan pengembangan aplikasi ini. Dukungan dan kontribusi dari berbagai pihak telah sangat berharga dalam menyelesaikan penelitian ini.

DAFTAR PUSTAKA

Shankar, M. P., Joshi, A., Sawant, N., & Jagdale, A. (2022). Web-based Visualization Tools to Demonstrate the Working of *Sorting* and Pathfinding Algorithms. International Journal of Computer Applications, 184(12), 13-20. https://doi.org/10.5120/ijca2022922096

Battistella, P. E., Amaral, E. H. D., & Santana, E. P. (2020). A visual tool to study *sorting* algorithms and their complexity. Journal of Computer Science Education, 32(4), 345-356. https://doi.org/10.1080/08993408.2020.1769845

Daher, R. A., & Sleem, H. K. (2021). Visualization of *sorting* algorithms in the virtual reality environment. Frontiers in Education, 6, Article 689216. https://doi.org/10.3389/feduc.2021.689216

Hundhausen, C. D., & Naps, T. L. (2020). An experimental study of the effectiveness of algorithm visualization in teaching. Journal of Visual Languages & Computing, 40, 50-63. https://doi.org/10.1016/j.jvlc.2020.100991

Li, L., Qiao, X., Lu, Q., Ren, P., & Lin, R. (2020). Rendering Optimization for Mobile Web 3D Based on Animation Data Separation and On-Demand Loading. IEEE Access, 8, 88474-88486. https://doi.org/10.1109/ACCESS.2020.2993613

Silva, D. B., Aguiar, R. D. L., Dvconlo, D. S., & Silla, C. N. (2019). Recent Studies About Teaching Algorithms (CS1) and Data Structures (CS2) for Computer Science Students. IEEE Frontiers in Education Conference (FIE), 2019, 1-8. https://doi.org/10.1109/FIE.2019.8857616

Paredes-Velasco, M., & Amores-Jiménez, E. (2023). Interactive Visualization of *Sorting* Algorithms in Real-Time. Journal of Educational Technology Systems, 51(2), 130-145. https://doi.org/10.1177/00472395231115888

Naps, T. L., & Rolfes, M. C. (2020). Visualization techniques for explaining the temporal and spatial complexity of *sorting* algorithms. Computers & Education, 148, Article 103811. https://doi.org/10.1016/j.compedu.2020.103811

Abdul Hanid, M. F., Bobrovnikov, I., & Sleem, H. (2022). Virtual Reality-based *Sorting* Algorithm Learning Tools. Journal of Computer Assisted Learning, 38(3), 621-634. https://doi.org/10.1111/jcal.12656

Qiu, G., & Chen, J. (2018). Web-based 3D map visualization using WebGL. 13th IEEE Conference on Industrial Electronics and Applications (ICIEA), 2018, 759-763. https://doi.org/10.1109/ICIEA.2018.8397815