Reslaj: Religion Education Social Laa Roiba Journal
Volume 6 Nomor 8 (2024) 4066 – 4078 P-ISSN 2656-274x E-ISSN 2656-4691
DOI: 10.47476/reslaj.v6i8.3388

Teknik *Refactoring* untuk Kualitas *Usability* Sistem Informasi SPP dan Tunggakan SDIT Nuralima

Wildah Fatma Lestari, Yulison Herry Chrisnanto, Rezki Yuniarti
Jurusan Informatika, Fakultas Sains dan Informatika, Universitas Jenderal Achmad Yani
wildahfatmal20@if.unjani.ac.id, yhc@if.unjani.ac.id,
rezki.yuniarti@lecture.unjani.ac.id

ABSTRACT

Software development in a system is important to support the efficiency and effectiveness of the system. However, codes and documentation related to software systems are always changing due to problems or the need for improvement. Therefore, software maintenance becomes an important part of software development and management. Refactoring techniques are considered a standard solution that involves improving the design structure of software while maintaining its functionality. The methodology used in this research includes system analysis, code smells, refactoring and system testing. The results of refactoring the internal code of the system make the internal code structure simpler and easier to read so that it will facilitate future development and updates to the system.

Keywords: *refactoring, code smells*

ABSTRAK

Pengembangan perangkat lunak dalam sebuah sistem merupakan hal yang penting untuk mendukung efisiensi dan efektivitas sistem tersebut. Namun, kode-kode dan dokumentasi terkait sistem perangkat lunak selalu mengalami perubahan karena adanya masalah atau kebutuhan untuk dilakukan perbaikan. Karena itu, pemeliharaan perangkat lunak menjadi bagian penting dalam pengembangan dan manajemen perangkat lunak. Teknik refaktoring dianggap sebagai solusi standar yang melibatkan peningkatan struktur desain perangkat lunak sambil mempertahankan fungsionalitasnya. Metodologi yang digunakan dalam penelitian ini meliputi analisis sistem, *code smells*, *refactoring* dan pengujian sistem. Hasil dari *refactoring* pada kode internal sistem membuat struktur kode internal lebih sederhana dan mudah untuk dibaca sehingga akan memudahkan dalam pengembangan dan pembaharuan pada sistem dimasa yang akan datang.

Kata kunci: refaktoring*, code smell*

PENDAHULUAN

Pengembangan perangkat lunak dalam sebuah sistem merupakan hal yang penting untuk mendukung efisiensi dan efektivitas sistem tersebut. Namun, kode-kode dan dokumentasi terkait sistem perangkat lunak selalu mengalami perubahan karena adanya masalah atau kebutuhan untuk dilakukan perbaikan[1]. Karena itu, pemeliharaan perangkat lunak menjadi bagian penting dalam pengembangan dan manajemen perangkat lunak[2]. Perbaikan ini bersifat inkremental yang bertujuan

Reslaj: Religion Education Social Laa Roiba Journal
Volume 6 Nomor 8 (2024) 4066 – 4078 P-ISSN 2656-274x E-ISSN 2656-4691
DOI: 10.47476/reslaj.v6i8.3388

untuk memperbarui beberapa fungsi, memperbaiki beberapa kekurangan desain, atau memperbaiki beberapa *bug*[1]-[2].

Refaktoring merupakan kegiatan yang mengambil tata letak internal perangkat lunak tetapi perilaku eksternal masih tetap sama[3].Teknik refaktoring dianggap sebagai solusi standar yang melibatkan peningkatan struktur desain perangkat lunak sambil mempertahankan fungsionalitasnya[4]. Meskipun teknik refaktoring dapat digunakan untuk menguji kualitas eksternal sebuah sistem, namun studi yang mengevaluasi secara empiris manfaat teknik refaktoring masih terbilang sedikit[5]. Oleh karena itu, diperlukan lebih banyak penelitian untuk mengatasi kurangnya studi empiris mengenai teknik refaktoring. Studi yang dilakukan di lingkungan akademis menemukan dampak positif yang lebih besar dari refaktoring pada kualitas perangkat lunak daripada studi yang dilakukan di industri[6]. Teknik refaktoring akan digunakan untuk meningkatkan kualitas *usability* dari sistem informasi SPP dan tunggakan berbasis web yang digunakan di lingkungan akademis SD Islam Terpadu Nuralima Karawang.

Sistem informasi SPP dan tunggakan berbasis web digunakan untuk menyimpan data pembayaran SPP bulanan, tunggakan bagi yang belum melakukan pembayaran, pendapatan harian dari SPP bulanan dan pelaporan bulanan pada SD Islam Terpadu Nuralima Karawang. Pengukuran mutu sistem informasi SPP dan tunggakan berbasis web dapat membantu untuk memastikan bahwa sistem tersebut dapat memenuhi kebutuhan pengguna dan dapat digunakan secara efektif dan efisien. Sistem ini digunakan oleh kepala sekolah dan staf tata usaha. Oleh karena itu, penting untuk memastikan bahwa sistem tersebut dapat memenuhi kebutuhan pengguna dan dapat digunakan secara efektif dan efisiensi.

Pengukuran kualitas pada sistem informasi SPP dan tunggakan berbasis web juga dapat membantu untuk mengidentifikasi kekurangan atau kelemahan sistem, sehingga dapat dilakukan perbaikan untuk meningkatkan mutu sistem. Sistem informasi ini merupakan sistem yang kompleks dan terus berkembang. Oleh karena itu, penting untuk melakukan pengukuran kualitas sistem untuk mengidentifikasi kekurangan atau kelemahan sistem. Dengan demikian, perbaikan dapat dilakukan untuk meningkatkan kualitas sistem[8]-[9].

Penelitian ini akan melakukan penerapan teknik refaktoring terhadap sistem informasi SPP dan Tunggakan SDIT Nuralima untuk meningkatkan kualitas pada bagian *usability* sistem.

Pentingnya penelitian ini selain untuk meningkatkan kualitas kode, juga dapat membuatnya lebih efisien, mudah dibaca dan dipelihara serta mengurangi kompleksitas pada kode sistem.

METODE PENELITIAN

Metodologi yang digunakan dalam penelitian ini meliputi analisis sistem, *code smells*, *refactoring* dan pengujian sistem. Penjelasan setiap tahapannya adalah sebagai berikut:

Reslaj: Religion Education Social Laa Roiba Journal
Volume 6 Nomor 8 (2024) 4066 – 4078 P-ISSN 2656-274x E-ISSN 2656-4691
DOI: 10.47476/reslaj.v6i8.3388

Analisis sistem

Analisis sistem dilakukan untuk memahami dan mengidentifikasi kebutuhan serta masalah dalam suatu sistem yang ada. Analisis sistem terbagi kedalam tiga tahap yaitu, analisis kebutuhan, analisis sistem dan analisis kode. Tahap ini dilakukan untuk mengetahui letak kekurangan kualitas sistem dan ditemukan kekurangan kualitas sistem terletak pada kualitas *usability*.

Code smells

Code smells dilakukan untuk mengidentifikasi perbaikan pada kode sistem. Dengan mengidentifikasi jenis *code smells* yang ada dalam kode dengan memperhatikan apakah ada *large class*, *long method*, *duplicate code* atau *lazy class*. Lalu memprioritaskan perbaikan berdasarkan dampaknya pada kualitas dan pemeliharaan kode yaitu pada aspek *usability*. Selanjutnya refaktor kode dengan mengikuti prinsip *clean code*. Memisahkan fungsi yang terlalu panjang, mengurangi kompleksitas dan menghapus duplikasi kode. Sistem dibuat menggunakan *framework* Laravel, analisis *code smell* hanya akan dilakukan pada bagian *view* dan *controller* karena *view* berhubungan langsung dengan *usabillity* dan *controller* tidak berhubungan langsung dengan *usability* namun *controller* mengendalikan bagaimana data diproses dan disiapkan untuk ditampilkan kepada pengguna. Sedangkan model berfokus pada interaksi dengan *database* dan data bisnis dan tidak langsung berinteraksi langsung dengan pengguna.

Refactoring

Refactoring adalah proses memperbaiki struktur internal program tanpa mengubah struktur eksternalnya. Tujuannya adalah untuk meningkatkan kualitas kode, membuatnya lebih efisien, mudah dibaca dan dipelihara serta mengurangi kompleksitas. Langkah yang pertama sebelum melakukan *refactoring* adalah dengan identifikasi kode yang perlu diperbaiki dengan berfokus pada bagian yang mengandung *code smells*. Lalu memilih teknik *refactoring* yang sesuai seperti menggabungkan metode, mengubah nama *variable*, memecah metode panjang atau menghapus kode yang tidak perlu. Langkah terakhir setelah melakukan *refactoring* adalah dengan melakukan pengujian pada sistem untuk memastikan sistem berfungsi dengan baik dan memastikan tidak ada efek samping pada sistem.

Pengujian

Tahap terakhir adalah tahap pengujian terhadap sistem yang dilakukan *refactoring*. Pengujian dilakukan untuk memastikan bahwa perubahan yang dilakukan terhadap kode tidak mengganggu fungsionalitas yang ada.

Reslaj: Religion Education Social Laa Roiba Journal
Volume 6 Nomor 8 (2024) 4066 – 4078 P-ISSN 2656-274x E-ISSN 2656-4691
DOI: 10.47476/reslaj.v6i8.3388

HASIL DAN PEMBAHASAN

Untuk melakukan *feractoring*, diperlukan analisis *code* sistem dan analisis kode untuk menemukan *code smells*, sehingga mempermudah dalam proses *refactoring*. Adapun fitur-fitur dari sistem informasi SPP dan Tunggakan adalah sebagai berikut:

a. Halaman login

Halaman login ini memungkinkan pengguna untuk masuk sebagai admin, kepala sekolah, dan bendahara dengan *username* dan *password* yang sesuai dengan yang tertera di dalam *database*. Pendaftaran anggota baru hanya dapat dilakukan oleh admin jika diperlukan.

Gambar 1. Tampilan Halaman Login

b. Halaman beranda

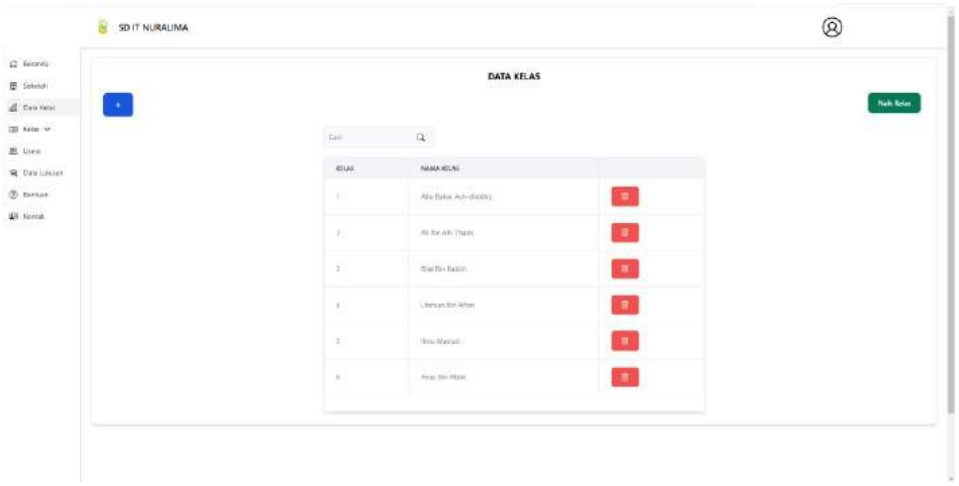
Halaman beranda akan menampilkan jumlah siswa yang terdaftar dalam sistem, jumlah *user* dan menampilkan profil sekolah seperti nama sekolah, NPSN, lokasi sekolah, dan kepala sekolah.

Gambar 2. Tampilan Halaman Beranda

Reslaj: Religion Education Social Laa Roiba Journal
Volume 6 Nomor 8 (2024) 4066 – 4078 P-ISSN 2656-274x E-ISSN 2656-4691
DOI: 10.47476/reslaj.v6i8.3388

c. Halaman data kelas

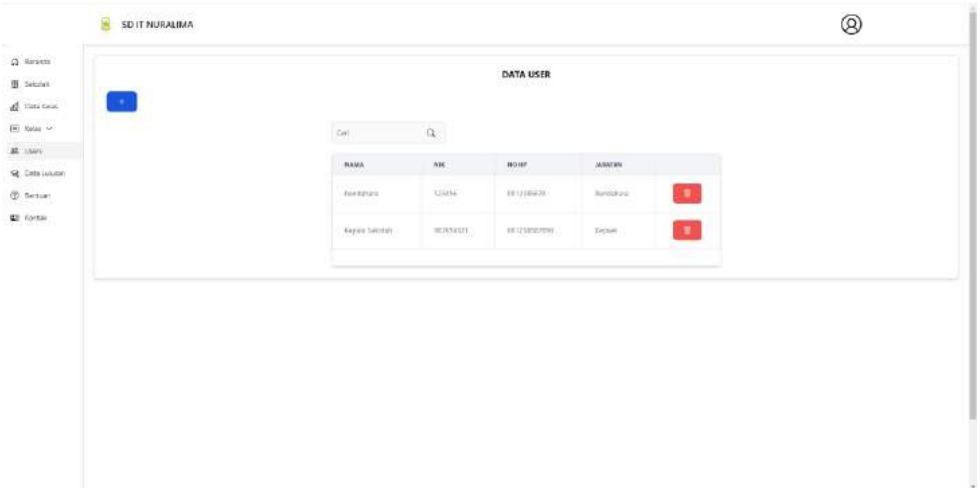
Halaman data kelas akan menampilkan nama-nama kelas yang ada di sekolah. Pada halaman ini admin dapat menaikkan kelas jika tahun ajaran baru telah tiba dan dapat menambahkan kelas baru.



Gambar 3. Tampilan Halaman Data Kelas

d. Halaman data *user*

Halaman data *user* ini akan menampilkan data *user* yang terdaftar dan hanya admin yang dapat *menambahkan* user baru.

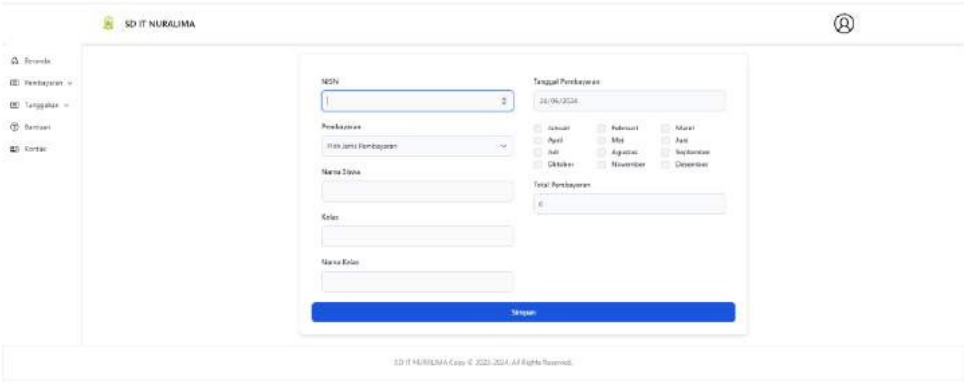


Gambar 4. Tampilan Halaman Data User

e. Halaman pembayaran

Halaman pembayaran terdapat pada halaman bendahara untuk mencatat pembayaran yang terjadi dan setelah data tersimpan akan mengeluarkan bukti pembayaran yang dapat dicetak untuk diberikan kepada orang tua siswa.

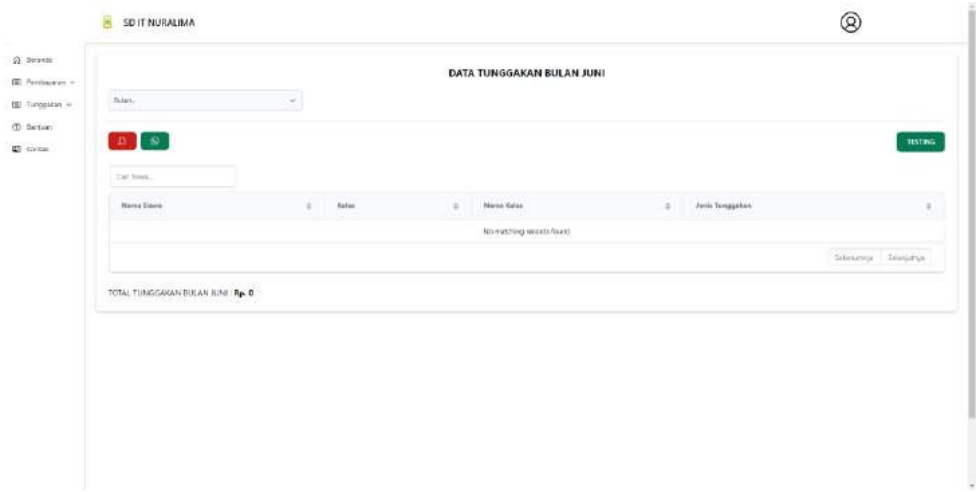
Reslaj: Religion Education Social Laa Roiba Journal
Volume 6 Nomor 8 (2024) 4066 – 4078 P-ISSN 2656-274x E-ISSN 2656-4691
DOI: 10.47476/reslaj.v6i8.3388



Gambar 5. Tampilan Halaman Pembayaran

f. Halaman data tunggakan

Halaman data tunggakan menampilkan data tunggakan siswa per bulannya.

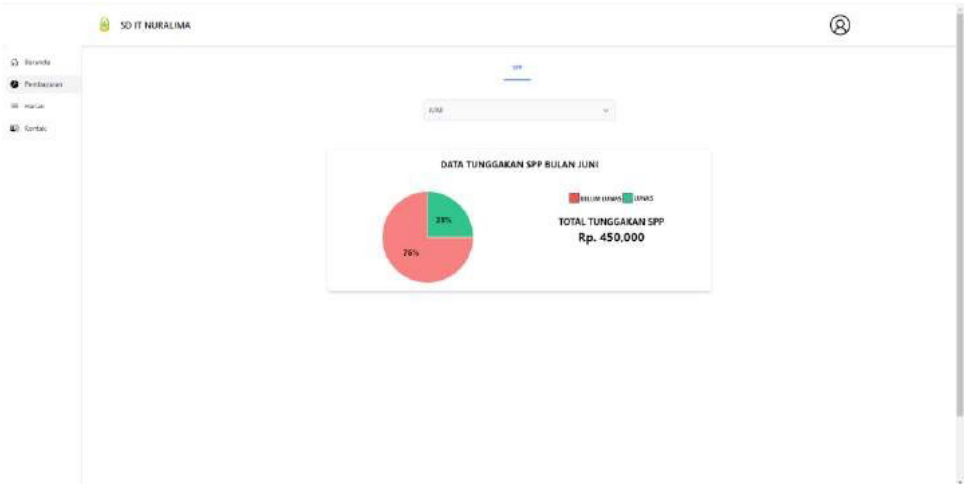


Gambar 6. Tampilan Halaman Data Tunggakan

g. Halaman data persentase pembayaran

Halaman data persentase akan menampilkan grafik persentase tunggakan yang dapat dilihat oleh kepala sekolah.

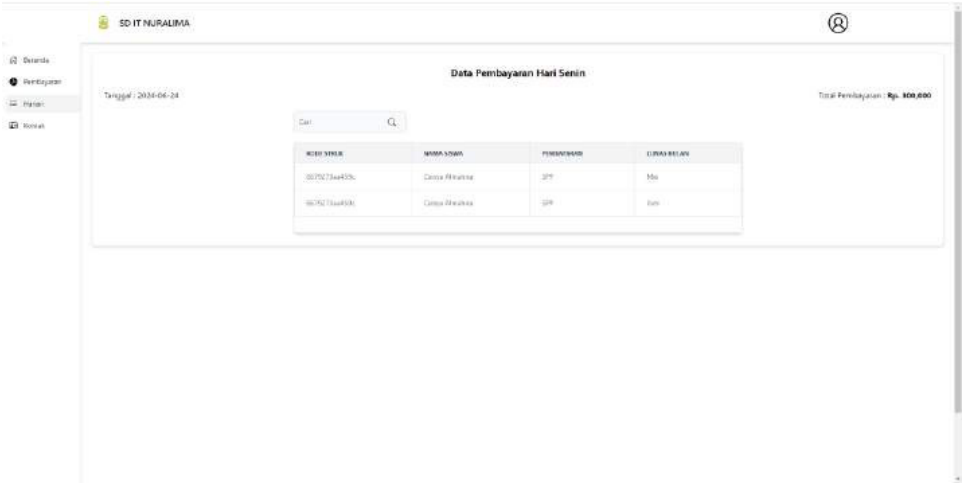
Reslaj: Religion Education Social Laa Roiba Journal
Volume 6 Nomor 8 (2024) 4066 – 4078 P-ISSN 2656-274x E-ISSN 2656-4691
DOI: 10.47476/reslaj.v6i8.3388



Gambar 7. Tampilan Halaman Data Persentase Pembayaran

h. Halaman data harian

Halaman data harian akan menampilkan data pembayaran harian yang telah terjadi. Halaman ini hanya dapat dilihat oleh kepala sekolah sebagai laporan harian.



Gambar 8. Tampilan Halaman Data Harian

Code smells

Setelah dilakukan analisis *code smells*, ditemukan beberapa jenis *code smell* pada kode sistem yaitu:

1. Duplikasi kode

   Ditemukan duplikasi kode yang dapat mengakibatkan berbagai masalah seperti pemeliharaannya yang sulit, kode yang lebih panjang dan kompleks, meningkatnya kemungkinan *bug*, kinerja yang buruk, dan desain yang buruk.

2. *Long Method*

   Ditemukan *script* Javascript pada *markup* HTML yang dapat membuat pemeliharaan sistem lebih sulit jika logika JavaScript lebih kompleks.

Reslaj: Religion Education Social Laa Roiba Journal
Volume 6 Nomor 8 (2024) 4066 – 4078 P-ISSN 2656-274x E-ISSN 2656-4691
DOI: 10.47476/reslaj.v6i8.3388

3. *Large Class*

   Terdapat inisialisasi yang panjang dan kompleks di dalam satu blok kode JavaScript yang membuat *class*-nya memiliki terlalu banyak tanggung jawab atau *variable*. *Large class* dapat mengakibatkan kompleksitas yang berlebihan dan mengurangi keterbacaan kode.

4. *Inconsistent Naming*

   Ditemukan ketidakkonsistenan penamaan id dan *variable* pada kode sistem seperti 'jenisPembayaranYangTersedia', 'bulanDipilih', 'PilihanBulan' yang dapat menyebabkan kesulitan untuk membaca kode, kesulitan untuk melakukan pencarian dan memungkinkan untuk terjadinya kesalahan logika.

Refactoring

Setelah ditemukan masalah pada kode, dilakukan *refactoring* pada kode untuk menghilangkan *code smell* guna membuat kode lebih mudah dipahami oleh developer lain dan memfasilitasi pemeliharaan kode di masa depan. Berikut adalah teknik yang digunakan untuk menghilangkan *code smells*:

1. Duplikasi kode

   Untuk menghilangkan duplikasi kode dapat dengan cara melakukan ekstrak komponen dengan membuat komponen yang nantinya akan dipanggil pada kode utama. Dengan melakukan ekstrak komponen, duplikasi kode akan berkurang dan meningkatkan keterbacaan serta kemudahan perawatan kode. Gambar 1 merupakan contoh untuk penerapan ekstrak komponen.

Reslaj: Religion Education Social Laa Roiba Journal
Volume 6 Nomor 8 (2024) 4066 – 4078 P-ISSN 2656-274x E-ISSN 2656-4691
DOI: 10.47476/reslaj.v6i8.3388

resources > views > components > data-table.blade.php

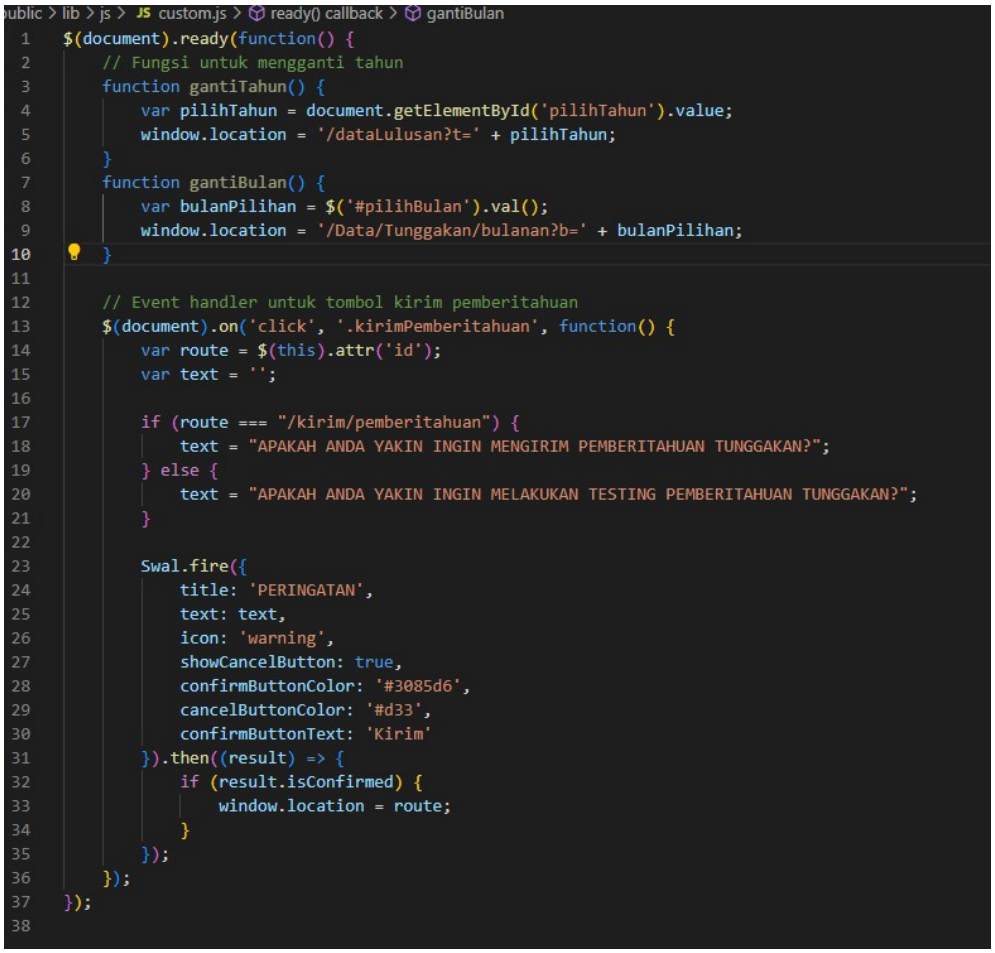
```
<div class="max-w-6xl block m-auto rounded-lg border shadow-md relative overflow-x-auto">
    <table class="w-full text-sm text-left text-gray-500 border-gray-500">
        <thead class="text-xs text-gray-700 uppercase bg-gray-100">
            <tr>
                <th scope="col" class="px-6 py-3 border">
                    Nama Siswa
                </th>
                <th scope="col" class="px-6 py-3 border">
                    Kelas
                </th>
                <th scope="col" class="px-6 py-3 border">
                    Jenis Pembayaran
                </th>
                <th scope="col" class="px-6 py-3 border">
                    Tunggakan
                </th>
            </tr>
        </thead>
        <tbody>
            @forelse ($data as $item)
                <tr class="bg-white border-b">
                    <td class="px-6 py-4 border">
                        {{ $item->namaSiswa }}
                    </td>
                    <td class="px-6 py-4 border">
                        {{ $item->kelas }}
                    </td>
                    <td class="px-6 py-4 border">
                        {{ $item->jenisPembayaran }}
                    </td>
                    <td class="px-6 py-4 border">
                        {!! $item->tunggakan !!}
                    </td>
                </tr>
            @empty
                <tr>
                    <td class="px-6 py-4 border text-center" colspan="4">DATA KOSONG</td>
                </tr>
            @endforelse
        </tbody>
        <tfoot>
            <tr class="bg-white border-b">
                <td class="px-6 py-4" colspan="4">
                    {{ $data->onEachSide(1)->links() }}
                </td>
            </tr>
        </tfoot>
    </table>
</div>
```

Gambar 9. Penerapan Ekstrak Komponen

2. *Long Method*

Solusi untuk jenis *code smell long method* karena terdapat kode JavaScript pada *markup* HTML adalah dengan cara melakukan eksternalisasi kode yaitu dengan memisahkan logika JavaScript yang ada pada *markup* HTML ke dalam *file* JavaScript dan memanggil *file* JavaScript eksternal pada *markup* HTML. Gambar 2 merupakan penerapan eksternalisasi JavaScript .

Reslaj: Religion Education Social Laa Roiba Journal
Volume 6 Nomor 8 (2024) 4066 – 4078 P-ISSN 2656-274x E-ISSN 2656-4691
DOI: 10.47476/reslaj.v6i8.3388



Gambar 10. Penerapan Eksternalisasi JavaScript

3. *Large Class*

Solusi untuk menghilangkan *large class* dalam kode adalah dapat dengan cara memisahkan fungsi yang memiliki tanggung jawab yang berbeda ke dalam *class* lain atau membuat *class* baru. Gambar 3 merupakan penerapan untuk solusi *large class*.

Reslaj: Religion Education Social Laa Roiba Journal
Volume 6 Nomor 8 (2024) 4066 – 4078 P-ISSN 2656-274x E-ISSN 2656-4691
DOI: 10.47476/reslaj.v6i8.3388

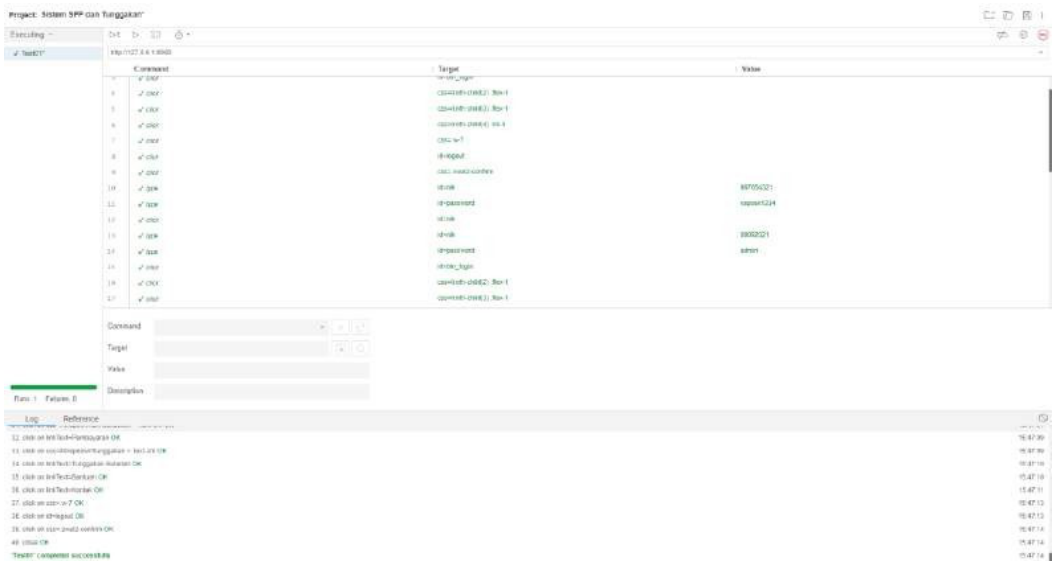
Gambar 11. Penerapan Memisahkan Fungsi

4. *Inconsistent Naming*

   Solusi untuk ketidakkonsistenan dalam melakukan penamaan id dan *variable* dapat dengan cara konvensi penamaan yang konsisten dan deskriptif.

Pengujian

Pengujian ini dilakukan dengan alat pengujian otomatis Selenium IDE yang akan memungkinkan pengujian web secara otomatis. Dengan hasil seperti pada gambar di bawah ini:



Gambar 12. Hasil Pengujian dengan Selenium IDE

Pengujian dilakukan pada sebagian besar fitur dan fungsionalitas yang penting dari sistem. Hasil pengujian menunjukkan bahwa sistem telah memenuhi kriteria kualitas dan fungsionalitas yang diharapkan berdasarkan hasil pengujian dan *refactoring* yang dilakukan pada kode sistem tidak mengubah fungsionalitas sistem.

Volume 6 Nomor 8 (2024) 4066 – 4078 P-ISSN 2656-274x E-ISSN 2656-4691
DOI: 10.47476/reslaj.v6i8.3388

KESIMPULAN DAN SARAN

Refactoring yang dilakukan pada kode sistem hanya akan mengubah struktur kode saja dan tidak akan mempengaruhi fungsionalitas sistem. Dengan melakukan analisis *code smell* dapat mempermudah untuk menentukan teknik refaktoring yang akan diterapkan ke dalam sistem. Hasil dari *refactoring* pada kode internal sistem membuat struktur kode internal lebih sederhana dan mudah untuk dibaca sehingga akan memudahkan dalam pengembangan dan pembaharuan pada sistem dimasa yang akan datang. Fungsionalitas sistem berjalan dengan baik dan dapat memenuhi tujuan awal pengembangan sistem, yaitu untuk membantu kinerja kepala sekolah dan bendahara dalam mencatat pembayaran SPP dan tunggakan yang terjadi di SDIT Nuralima.

Dalam pelaksanaan penelitian ini tentu tidak terlepas dari berbagai kesulitan, maka saran untuk pembangunan sistem selanjutnya adalah dengan menggunakan *framework* agar kode lebih terstruktur, juga menerapkan *clean code* untuk mengurangi duplikasi kode dan meningkatkan kualitas kode.

DAFTAR PUSTAKA

Alizadeh, V., Kessentini, M., Mkaouer, M. W., Ocinneide, M., Ouni, A., & Cai, Y. (2020). An interactive and dynamic search-based approach to software refactoring recommendations. *IEEE Transactions on Software Engineering*, 46(9), 932–961.

Almogahed, A., Omar, M., & Zakaria, N. H. (2022). Recent studies on the effects of refactoring in software quality: Challenges and open issues. In *2022 2nd International Conference on Emerging Smart Technologies and Applications (eSmarTA 2022)*.

Fowler, M. (2018). *Refactoring: Improving the design of existing code* (2nd ed.). Addison-Wesley.

Ghannem, A., Kessentini, M., Hamdi, M. S., & El Boussaidi, G. (2018). Model refactoring by example: A multi-objective search based software engineering approach. *Journal of Software Evolution and Process*, 30(4), 1–20.

Kaur, S., & Singh, P. (2019). How does object-oriented code refactoring influence software quality? Research landscape and challenges. *Journal of Systems and Software*, 157.

L'Erario, A., Thomazinho, H. C. S., & Fabri, J. A. (2020). An approach to software maintenance: A case study in small and medium-sized businesses IT organizations. *International Journal of Software Engineering and Knowledge Engineering*, 30(5), 603–630.

Lenstra, A. K. (1994). Factoring. *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, 857 LNCS, 28–38.

Volume 6 Nomor 8 (2024) 4066 – 4078 P-ISSN 2656-274x E-ISSN 2656-4691
DOI: 10.47476/reslaj.v6i8.3388

Pressman, R. S. (2001). *Software engineering: A practitioner's approach* (5th ed.). McGraw-Hill Book Company.

Sommerville, I. (2011). *Software engineering* (9th ed.). Pearson Education.