Pengembangan *Front End* Sistem Informasi Akuntansi Menggunakan Kerangka Kerja Vue.js

Andhika Rizky Aryasta
Program Studi Informatika
Universitas Islam Indonesia
Yogyakarta, Indonesia
18523073@students.uii.ac.id

Andhik Budi Cahyono
Program Studi Informatika
Universitas Islam Indonesia
Yogyakarta, Indonesia
andhikbudi@uii.ac.id

***Abstrak*—Geekgarden *Software House* adalah suatu perusahaan di Yogyakarta yang berfokus dalam bidang teknologi informasi. Salah satu *software* yang dikembangkan Geekgarden adalah Sistem Informasi Akuntansi. Sistem Informasi Akuntansi dikembangkan sebagai solusi untuk meningkatkan efisiensi, pengelolaan data, dan mengubah proses bisnis pada suatu perusahaan. Sistem Informasi Akuntansi merupakan aplikasi untuk mencatat transaksi, mengelola transaksi, dan menyediakan laporan transaksi yang telah berlangsung. Pengembangan Sistem Informasi Akuntansi berbasis *website* menggunakan kerangka kerja Vue.js dengan *library* Vuetify untuk membuat tampilan antarmuka *website*. Pemilihan kerangka kerja Vue.js karena memiliki kinerja yang cepat, ringan dan mendukung penggunaan *Single Page Application* (SPA). Hasil dari pengembangan berupa tampilan *user interface* fitur transaksi jurnal pada Sistem Informasi Akuntansi, fitur transaksi yang diintegrasikan dengan API, dan pengujian fungsionalitas pada fitur transaksi. Tujuan akhir dari pengembangan *front end* pada Sistem Informasi Akuntansi menggunakan kerangka kerja Vue.js adalah memberikan kemudahan pengembang dalam membuat tampilan *user interface* dengan dukungan berbagai *library* dan komunikasi yang dilakukan antara *front end* dengan *back end* melalui integrasi API.**

Kata Kunci—Sistem Informasi Akuntansi, Vue.js, front end, Single Page Application

I. PENDAHULUAN

Geekgarden *Software House* adalah suatu perusahaan yang berlokasi di Yogyakarta dan bergerak dalam bidang teknologi informasi. Geekgarden sendiri telah menangani berbagai macam proyek pengembangan perangkat lunak berbasis *website* maupun *mobile*. Salah satu *software* yang dikembangkan Geekgarden adalah Sistem Informasi Akuntansi. Proyek tersebut dikembangkan menjadi dua bagian yaitu pengembangan *front end* dan pengembangan *back end*. Pengembangan *front end* menggunakan kerangka kerja Vue.js dengan *library* Vuetify untuk membangun tampilan antarmuka pada *website*. Pengembangan *back end* untuk mengelola penyimpanan data dan merancang *Application Programming Interface* (API). Hal tersebut digunakan untuk menjembatani proses transfer data yang dikirimkan pada *front end*.

Pengembangan *front end* dilakukan berdasarkan hasil rancangan dari *user interface/user experience designer* (UI/UX *Designer*) yang telah melalui tahap pengujian dengan pihak pelanggan. Rancangan desain kemudian dibagikan kepada pengembang *front end* yang dapat diakses melalui aplikasi Figma untuk diimplementasikan pada *website*. Penerapan *Single Page Application* (SPA) bertujuan untuk meningkatkan kinerja menjadi lebih baik, mengurangi beban pada server saat terjadi permintaan data dari pengguna, dan meminimalisir penggunaan sumber daya pada server [1].

Penggunaan kerangka kerja Vue.js cocok digunakan pada Sistem Informasi Akuntansi untuk membangun tampilan *user interface* pada aplikasi berbasis *website*. Dikarenakan Vue.js memiliki keunggulan yaitu adanya dukungan dengan berbagai *library* yang memudahkan dalam pengembangan aplikasi dan mendukung untuk membangun SPA. Selain itu, Vue.js memiliki kemudahan untuk diintegrasikan dengan bahasa program yang berbeda melalui API.

Secara umum SPA digunakan pada aplikasi *dashboard* yang menyediakan seluruh informasi mengenai segala aktivitas perusahaan. Proses perpindahan antar halaman di *handle* dengan *routing* sehingga *website* menampilkan data tanpa adanya *refresh/reload page*. Selain itu, SPA tidak membutuhkan banyak waktu untuk memproses data dan memberikan pengalaman pengguna menjadi lebih baik [2].

Pada makalah ini, dijelaskan mengenai pengembangan *front end* pada *website* menggunakan kerangka kerja Vue.js dengan berbagai *library*. Makalah ini juga membahas mengenai integrasi API dengan *back end* sehingga dapat menjalankan seluruh fungsionalitas sesuai dengan rancangan.

II. LANDASAN TEORI

A. *Sistem Informasi Akuntansi*

Sistem Informasi Akuntansi merupakan sebuah sistem berbasis *website* untuk mengumpulkan, menyimpan, dan mengolah seluruh transaksi akuntansi yang berlangsung pada sistem [3]. Dengan adanya, sistem tersebut pengguna internal maupun eksternal perusahaan yang berkepentingan dapat meningkatkan efektivitas dan kinerja kegiatan akuntansi. Selain itu, pengguna dapat menghasilkan sebuah laporan transaksi akuntansi secara *real time*.

B. *Single Page Application* (SPA)

SPA merupakan suatu aplikasi yang berjalan pada *website* menggunakan satu halaman secara dinamis untuk menangani seluruh aktivitas tanpa perlu melakukan *refresh*/*reload page*. Ketika terjadi interaksi yang dilakukan pada sistem pengguna tetap berada pada halaman yang sama setiap mengirimkan *request* ke *server* [1]. Penerapan SPA bertujuan untuk meningkatkan kinerja proses memuat data dengan waktu singkat dan meminimalisir penggunaan sumber daya *server*.

C. *Vue.js*

Vue.js merupakan sebuah kerangka kerja JavaScript progresif berbasis *open source* untuk membangun tampilan *user interface* pada aplikasi berbasis *website*. Vue.js diciptakan oleh Evan You pada tahun 2014 dengan menerapkan konsep MVC (*model-view-controller)* [4]. Vue.js dibangun menggunakan HTML, CSS, JavaScript, dan komponen yang dapat digunakan kembali.

Pada konsep MVC, Vue.js hanya merepresentasikan layer *View* untuk *front end* sedangkan untuk *Model* dan *Controller* merupakan bagian dari *back end* [5]. Selain itu, Vue.js memberikan kemudahan untuk diintegrasikan dengan *library* lain, serta mendukung untuk membangun SPA.

D. *Kajian Pustaka*

Pada beberapa penelitian yang telah dilakukan sebelumnya terkait pengembangan *front end* berbasis *website* menggunakan kerangka kerja Vue.js. Pada penelitian Belluno [1] hanya berfokus untuk membahas pengembangan SPA (*Single Page Application*) pada Sistem Informasi Akademik. Hasil dari penelitian tersebut menjelaskan SPA membantu dalam mengaplikasikan Sistem Informasi Akademik dengan tampilan *user interface* yang mudah digunakan. Tampilan dibuat agar *user friendly* sesuai dengan kebutuhan dan keinginan pengguna. Sistem tersebut menyajikan informasi berbentuk teks dan grafik sinkron, pengelolaan informasi secara *realtime* yang membuat komunikasi data lebih cepat. Selain itu, SPA memiliki keunggulan tidak memakan banyak sumber daya.

Dalam penelitian Putra, Pramana, dan Srinadi [6], membahas mengenai membangun Sistem Manajemen Arsip berbasis *website* menggunakan kerangka kerja Laravel dan kerangka kerja Vue.js. Pada penelitian ini, disebutkan bahwa penggunaan kerangka kerja Laravel untuk memanipulasi, dan mengelola basis data. Penggunaan kerangka kerja Vue.js untuk membangun tampilan antarmuka menjadi interaktif karena memiliki kemampuan tanpa *refresh page* setiap berpindah halaman.

Penelitian Chastro dan Darmawan [7] membahas mengenai perbandingan dalam pengembangan *front end* menggunakan blade template dengan kerangka kerja Laravel dan Vue.js. Dalam kasus tersebut, peneliti melakukan perbandingan kedua kerangka kerja tersebut berdasarkan performa, *routing*, tampilan, dan kecepatan *script*. Hasil dari penelitian tersebut menyebutkan bahwa kerangka kerja Vue.js memiliki performa yang stabil untuk pengembangan aplikasi kompleks. Blade template dengan kerangka kerja Laravel lebih cocok digunakan untuk pengembangan aplikasi sederhana. Pada penelitian tersebut dijelaskan karena pada aplikasi yang lebih kompleks blade template mengalami penurunan performa secara drastis.

III. METODOLOGI

Metodologi pengembangan *front end* Sistem Informasi Akuntansi terdiri dari beberapa tahapan yaitu: analisis, desain, implementasi, dan pengujian.

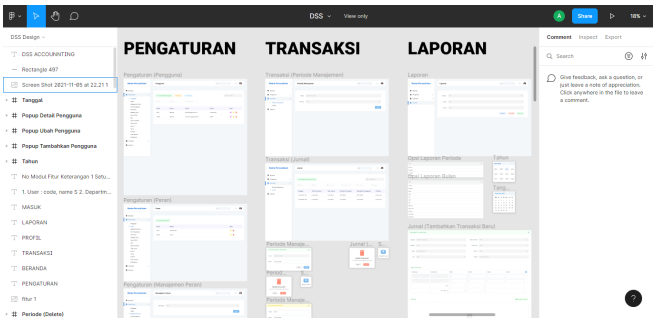
A. *Analisis*

Analisis merupakan tahap pengembangan yang dilakukan oleh *Project Manager* (PM) melalui wawancara dengan pelanggan untuk menganalisis kebutuhan yang diperlukan sistem tanpa melibatkan divisi lainnya. Seluruh data analisis dikumpulkan menjadi bentuk *document requirements specsification.* Hasil dari analisis berupa *business requirements summary*, *deliverable/scope, functional requirement, business process.* Berdasarkan kebutuhan yang didapatkan dari hasil analisis kemudian PM meminta tim UI/UX *Desainer* untuk membangun desain.

B. *Desain*

Pada tahap *desain* tim UI/UX *Designer* membangun desain berbentuk prototipe berbasis *website* menggunakan Figma sebelum diimplementasikan ke dalam sistem sesuai dengan hasil analisis. Hasil dari desain prototipe digunakan oleh pengembang untuk mengembangkan tampilan *user interface* fitur dan alur kerja pada Sistem Informasi Akuntansi.

Penggunaan Figma untuk mendukung interaksi secara *real time* antara UI/UX *Designer*, pengembang, dan pelanggan. Gambar 1 menunjukan desain prototipe pada beberapa fitur yang dikembangkan pada Sistem Informasi Akuntansi.



Gambar 1 Prototipe Sistem Informasi Akuntansi

C. *Implementasi*

Tahap implementasi digunakan oleh pengembang untuk mengimplementasikan desain prototipe ke dalam sistem atau kode program. Proses implementasi dilakukan dari sisi pengembangan *front end* menggunakan kerangka kerja Vue.js dengan beberapa *library* tambahan untuk memenuhi kebutuhan sistem.

Pada tahap ini pengembang membangun tampilan *user interface* pada *website* Sistem Informasi Akuntansi sesuai dengan desain prototipe dan dapat mengimprovisasikan. Selain itu, hasil dari tampilan *user interface* yang telah dibangun memerlukan adanya integrasi API dengan *back end* sehingga dapat dijalankan sesuai dengan proses bisnis pada sistem tersebut.

D. *Pengujian*

Pengujian dilakukan oleh tim QA/QC (*Quality Assurance/Quality Control)* setelah pengembangan telah selesai dikerjakan. Pada tahap ini tim QA/QC melakukan pengujian secara manual menggunakan metode *black box testing.* Tujuannya untuk memastikan seluruh fungsionalitas yang dikembangkan pada Sistem Informasi Akuntansi dapat berjalan sesuai dengan yang diharapkan.

Proses pengujian dilakukan dengan cara menambahkan data pada fungsional yang diuji menggunakan beberapa kasus pengujian. Berdasarkan keluaran yang ditampilkan, tim QA/QC dapat menentukan apakah hasil tersebut telah sesuai yang diharapkan. Namun, apabila keluaran yang dihasilkan memiliki *issue* maka akan dilakukan perbaikan pada proses pengembangan hingga keluaran berjalan sesuai harapan.

IV. HASIL DAN PEMBAHASAN

A. Implementasi

1) Penggunaan Library

Kerangka kerja Vue.js memiliki dukungan dalam penggunaan *library* tambahan sehingga mempercepat proses pengembangan *front end* sesuai kebutuhan sistem. Gambar 2 merupakan *library* yang di *import* secara *global* pada src/main.js dengan tujuan untuk memudahkan dalam memanggil *library* pada komponen manapun.

Library yang digunakan untuk pengembangan *front end* Sistem Informasi Akuntansi diantaranya, yaitu:

a) Library Vuetify

Penggunaan *library* Vuetify untuk membuat tampilan *user interface*. Vuetify menyediakan berbagai komponen yang dapat digunakan untuk pengembangan *front end.*

b) Library Axios

Axios merupakan http client untuk browser dan node.js. Axios digunakan untuk melakukan *request* dan melihat *response* data pada API.

c) Library vue2-datepicker

Library vue2-datepicker digunakan untuk memudahkan pengguna dalam memilih kalender secara lengkap. *library* vue2-datepicker dapat sesuaikan dengan kebutuhan sistem.

d) Library vue-numeric

Library vue-numeric digunakan untuk memasukkan suatu angka yang ditampilkan ke dalam nilai mata uang.

main.js

```
import Vue from "vue";
import App from "./App.vue";
import router from "./router";
import store from "./store";
import Axios from "axios";
import vuetify from "./plugins/vuetify";
import VueApexCharts from "vue-apexcharts";
Vue.use(VueApexCharts);

Vue.component("apexchart", VueApexCharts);

Vue.prototype.$http = Axios;

Vue.config.productionTip = false;

new Vue({
  router,
  store,
  vuetify,
  render: (h) => h(App),
}).$mount("#app");
```

Gambar 2 *Import library* secara *global*

2) Melakukan revamp menu transaksi

a) Membuat dan integrasi unggah jurnal

Unggah jurnal merupakan fitur yang digunakan untuk menambahkan transaksi berbentuk file dengan format .txt. Pembuatan fitur unggah jurnal memanfaatkan komponen *button* yang tersedia pada *library* Vuetify. Setiap terdapat aksi pada *button* unggah jurnal pengguna diarahkan untuk memilih *file* yang terdapat di perangkat.

Apabila tipe file yang diunggah selain format .txt maka sistem menampilkan sebuah *pop up error* beserta penjelasan. Jika tipe file dan isi file sesuai dengan yang diminta oleh sistem maka muncul *pop up success*. Gambar 3 merupakan sebagian kode program yang digunakan untuk membuat komponen *button* pada fitur unggah jurnal.

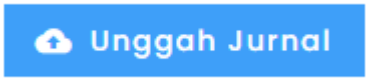
```
<v-btn
  :disabled="listPermission[1] === false"
  @click="selectFile"
  class="btn-upload-txt"
  color="#F1FFEF"
  tile
  depressed
>
  <v-icon class="mr-2" size="18">mdi-cloud-upload</v-icon>
  Unggah Jurnal
</v-btn>
<input
  id="upload"
  type="file"
  ref="file"
  accept=".txt"
  style="visibility: hidden"
  multiple
  @change="onFileChange"
/>
</v-col>
```

Gambar 3 Komponen *button* unggah jurnal

Setelah terdapat komponen *button* unggah jurnal kemudian dilakukan integrasi yang memanfaatkan *endpoint* API menggunakan *library* Axios. Ketika file dipilih Axios akan mengirimkan *request* ke server menggunakan *method* POST dengan url /upload-api/upload-file. Data yang dikirimkan melalui *front end* tersimpan ke dalam *database*. Gambar 4 merupakan kode program hasil integrasi API yang dilakukan pada fitur unggah jurnal. Hasil dari pengembangan fitur unggah jurnal dapat dilihat pada Gambar 5.

```
actionUpload() {
  this.$refs.file.value = "";
  let formData = new FormData();
  formData.append("file", this.fileUpload[0]);
  axios
    .post(
      process.env.VUE_APP_API_URL + "/upload-api/upload-file",
      formData,
      {
        headers: { "Content-Type": "multipart/form-data" },
      }
    )
    .then((resp) => {
      console.log(resp);
      //this.$store.commit("showSnackbarAdd", formData);
      this.getSnackbarAdd = true;
      this.$store.dispatch("getListTransactionData", formData);
    })
    .catch((err) => {
      console.log(err.response);
      this.fileUpload.splice(0, this.fileUpload.length);
      this.isShowError = true;
      const errorMessage = err.response.data;
```

Gambar 4 Integrasi API unggah jurnal



Gambar 5 Tampilan *button* unggah jurnal

b) Menampilkan data transaksi jurnal

Halaman transaksi jurnal merupakan fitur untuk menampilkan keseluruhan data transaksi jurnal yang ke dalam bentuk tabel. Ketika terdapat aktivitas transaksi seperti unggah jurnal dan tambah transaksi jurnal maka data transaksi terbaru ditampilkan pada tabel baris paling atas. Transaksi jurnal yang telah di posting pada sistem datanya tidak ditampilkan pada tabel.

Pada halaman tersebut terdapat tampilan antarmuka seperti tabel transaksi, *filter*, ubah transaksi, hapus transaksi,

dan lihat detail transaksi, serta posting jurnal. Pada Gambar 6 menunjukan sebagian kode untuk membuat tampilan *user interface* halaman transaksi jurnal.

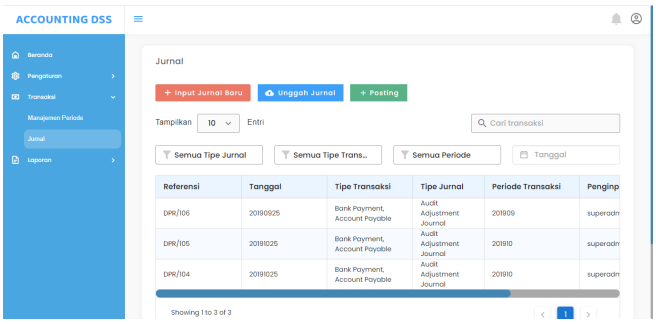
```
<v-card class="journal-table-card pb-4" elevation="0">
  <v-data-table
    v-model="selected"
    :headers="headers"
    :items="filteredTransactionLists"
    :options="{
      itemsPerPage: itemPerPage,
    }"
    :page.sync="page"
    @page-count="pageCount = $event"
    :loading="isLoading"
    :loading-text="loadingText"
    :no-data-text="noDataText"
    :footer-props="{
      'items-per-page-text': '',
      'disable-items-per-page': true,
      'items-per-page-options': [],
      'prev-icon': null,
      'next-icon': null,
    }"
    show-select
    class="journal-table"
  >
    <template v-slot:[`item.date`]="{ item }">{{
```

Gambar 6 Sebagian kode untuk membuat tampilan *user interface* transaksi jurnal

Transaksi jurnal memerlukan integrasi menggunakan *library* Axios. Nantinya, data ditampilkan pada tabel ketika melakukan *request* ke *database* menggunakan *method* GET dengan url /transaction. Setiap terdapat penambahan data melalui fitur tambah jurnal baru atau unggah jurnal maka tabel daftar transaksi melakukan *refresh* secara otomatis. Gambar 7 merupakan kode untuk melakukan pengambilan data pada *database* yang akan ditampilkan pada tabel daftar transaksi jurnal. Hasil dari pembuatan tampilan *user interface* transaksi jurnal dapat dilihat pada Gambar 8.

```
async getListTransactionData({ commit }) {
  try {
    let res = await axios.get(process.env.VUE_APP_API_URL + "/transaction");
    commit("setTransaction", res.data.data);
  } catch (error) {
    console.log(error);
  }
},
```

Gambar 7 Integrasi API tabel unggah jurnal



Gambar 8 Tampilan *user interface* transaksi jurnal

c) Melakukan revamp tambah transaksi jurnal

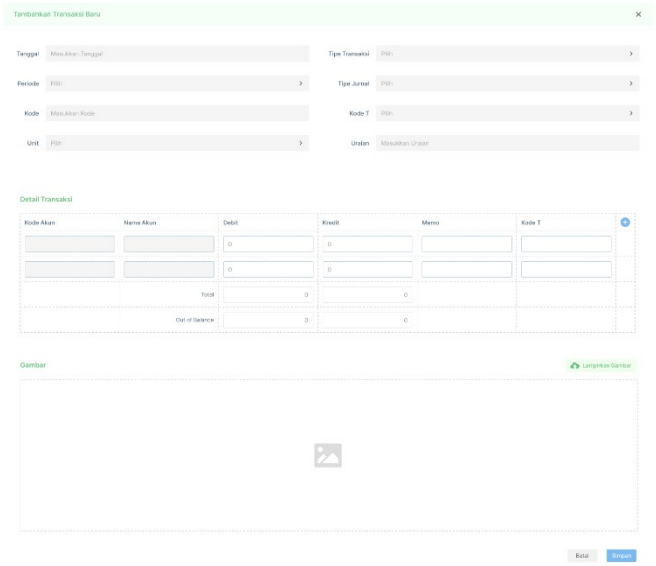
Tambah transaksi jurnal merupakan fitur untuk menambahkan transaksi jurnal yang terdapat pada menu transaksi. Sebelumnya pada fitur ini telah dikembangkan tetapi terdapat *revamp* yang dilakukan karena terdapat perubahan pada *document requirements specification*. Perubahan tersebut meliputi perubahan *input field*, pergantian *type field*, dan beberapa fitur yang perlu disesuaikan.

Pada halaman tambah transaksi jurnal memerlukan konfigurasi ulang integrasi *endpoint* API dengan *method* POST karena terdapat beberapa *revamp* pada *front end*. Konfigurasi ulang dilakukan agar data dapat dikirimkan ke *server*. Pada Gambar 9 menunjukkan kode untuk konfigurasi ulang integrasi API pada fitur tambah transaksi jurnal baru.

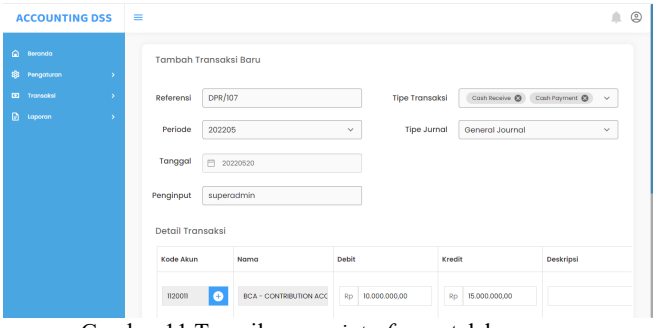
```
async addTransaction({ dispatch, commit }, payload) {
  try {
    let res = await axios.post(
      process.env.VUE_APP_API_URL + "/transaction",
      payload
    );
    if (res.data.status === 200) {
      commit("showSnackbarAdd", true);
      router.push({ name: "List Jurnal" });
      dispatch("getListTransactionData");
    } else {
      commit("showSnackbarAlert", true);
    }
  } catch (error) {
    console.log(error);
  }
},
```

Gambar 9 Kode untuk mengeksekusi integrasi API

Gambar 10 merupakan tampilan *user interface* pada halaman tambah transaksi sebelum di *revamp* masih terdapat beberapa komponen yang belum sesuai dengan harapan. Pada Gambar 11 merupakan tampilan *user interface* halaman tambah transaksi setelah di *revamp* sesuai dengan *document requirements specification versi* terbaru.



Gambar 10 Tampilan *user interface* sebelum revamp



Gambar 11 Tampilan *user interface* setelah revamp

d) Membuat posting transaksi jurnal

Posting transaksi jurnal merupakan fitur untuk memposting data jurnal yang telah ditambahkan ke dalam sistem. Ketika terdapat data transaksi jurnal yang dipilih

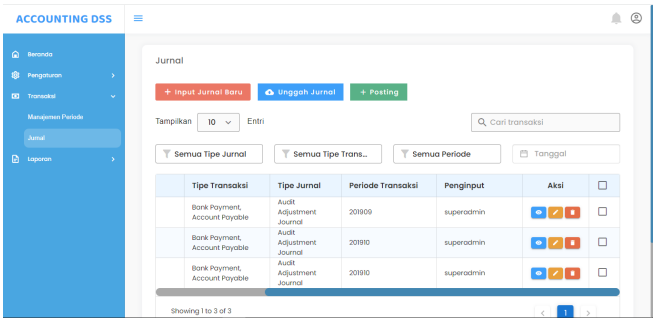
melalui *checkbox* dan melakukan aksi pada *button* posting maka akan menghasilkan sebuah laporan.

Pada proses posting jika transaksi jurnal yang dipilih merupakan periode dengan status terbuka maka posting diproses sedangkan periode dengan status tertutup posting gagal diproses. Fitur posting dibuat menggunakan komponen *button.* Selanjutnya dilakukan integrasi API yang memanfaatkan *library* Axios sehingga posting dapat berjalan sesuai dengan yang diharapkan.

Setiap kali pengguna melakukan aksi *checkbox* yang terdapat pada tabel transaksi jurnal kemudian menekan *button* posting Axios meminta *request* ke *server* menggunakan *method* POST dengan url /posting. Pada Gambar 12 merupakan hasil integrasi API yang dilakukan pada *button* untuk posting jurnal. Hasil dari pengembangan fitur posting jurnal dapat dilihat pada Gambar 13 yang terletak paling kanan pada masing-masing data berbentuk *checkbox*.

```
async postingJournal({ dispatch, commit }, payload) {
  try {
    commit("setBtnLoading", true);
    let res = await axios.post(process.env.VUE_APP_API_URL + "/posting", {
      userId: payload.userId,
      transaction: payload.transaction,
    });
    const message = res.data.data.message;
    const result = message.charAt(0).toUpperCase() + message.slice(1);
    if (res.data.status === 207) {
      commit("setErrorMessagePosting", result);
      commit("showSnackbarAlert", true);
    } else {
      commit("showSnackbarPost", true);
      dispatch("getListTransactionData");
    }
  } catch (error) {
    console.log(error);
  } finally {
    commit("setBtnLoading", false);
  }
},
```

Gambar 12 Integrasi posting jurnal



Gambar 13 Tampilan *button* dan *checkbox* posting pada tabel

B. Pengujian

Pengujian dilakukan oleh tim QA/QC (*Quality Assurance/Quality Control)* untuk memastikan seluruh hasil keluaran dapat berjalan sesuai dengan harapan. Tabel 1 merupakan tahap-tahap pengujian menggunakan metode *black box testing*.

Tabel 1 Hasil pengujian menggunakan metode *black box testing*

No	Fitur yang diuji	Kasus pengujian	Keluaran yang diharapkan	Hasil Pengujian
1	Unggah Jurnal	Pengguna melakukan unggah file berisikan data yang sesuai dengan format txt	Data ditampilkan pada tabel transaksi jurnal dan menampilkan *pop up success*	*Pass*
		Pengguna melakukan unggah file dengan format selain txt	Sistem menampilkan *pop up error* dengan keterangan "*The file must be a file of type txt*"	*Pass*
		Pengguna melakukan unggah file berisikan data yang tidak sesuai dengan format file txt	Sistem menampilkan *pop up error* dengan keterangan *error*	*Pass*
2	Halaman transaksi jurnal	Pengguna menambahkan data melalui unggah jurnal dan *input* jurnal baru	Sistem melakukan *refresh* otomatis pada tabel ketika terdapat transaksi dan data paling baru terletak di baris paling atas	*Pass*
		Pengguna melakukan *filter* tipe jurnal	Sistem hanya menampilkan data berdasarkan tipe jurnal yang dipilih	*Pass*
		Pengguna melakukan *filter* tipe transaksi	Sistem hanya menampilkan data berdasarkan tipe trasaksi yang dipilih	*Pass*
		Pengguna melakukan *filter* periode	Sistem hanya menampilkan data sesuai dengan periode yang dipilih	*Pass*
		Pengguna melakukan pencarian data transaksi	Sistem menampilkan data berdasarkan referensi yang dicari	*Pass*
3	Tambah transaksi jurnal	Pengguna menekan tombol simpan ketika data belum terisi	Sistem menampilkan peringatan pada *input field* yang harus diisi	*Pass*
		Pengguna mengisikan debit dan kredit dengan nilai 0	Sistem tidak dapat menyimpan data ketika kredit dan debit diisikan dengan nilai 0	*Pass*
4	Posting Jurnal	Pengguna memilih transaksi dengan periode yang sedang ditutup	Sistem tidak dapat memproses posting dan menampilkan *pop up* bahwa periode yang dipilih sedang ditutup	*Pass*
		Pengguna memilih transaksi dengan	Sistem memproses posting jurnal dan	*Pass*

		periode terbuka	menampilkan *pop up* bahwa jurnal berhasil diposting	

Berdasarkan pengujian yang dilakukan pada Tabel 1, menunjukkan bahwa semua fitur memberikan hasil dengan nilai "*Pass*". Artinya semua yang dilakukan pengujian dapat bekerja dengan baik sesuai dengan harapan.

V. KESIMPULAN

Berdasarkan hasil pengembangan *front end* Sistem Informasi Akuntansi, adanya dukungan *library* tambahan telah membantu dalam membangun tampilan *user interface* sehingga memudahkan proses pengembangan sesuai kebutuhan sistem. Selain itu, penerapan *library* digunakan untuk membuat fitur yang dikembangkan dapat berjalan sesuai dengan proses bisnis karena dilakukan integrasi API. Hal tersebut dibuktikan berdasarkan hasil pengujian yang telah dilakukan oleh tim QA/QC (*Quality Assurance/Quality Control)* menggunakan metode *blackbox testing* pada seluruh fitur yang dikembangan dengan nilai "*Pass*".

REFERENSI

[1] P. L. L. Belluano, "Pengembangan Single page Application Pada Sistem Informasi Akademik," *Ilk. J. Ilm.*, vol. 10, no. 1, pp. 38–43, 2018.

[2] R. Tampang, M. O. Kadang, and H. Arrang, "Pengiriman Barang Berbasis Web Pada Ekspedisi Anugrah Indah Dengan Menggunakan Teknologi Single Page Application (Spa)," *Paulus Informatics J.*, vol. 1, no. 2, pp. 21–26, 2020, [Online]. Available: www.degananda.com.

[3] S. Setiawansyah, "Monitoring Aplikasi Menggunakan Dashboard Untuk Sistem Informasi Akuntansi Pembelian Dan Penjualan (Studi Kasus : Ud Apung)," *J. Tekno Kompak*, vol. 14, no. 1, p. 47, 2020, doi: 10.33365/jtk.v14i1.503.

[4] "Introduction — Vue.js." https://v2.vuejs.org/v2/guide/ (accessed May 26, 2022).

[5] L. Gani, *Panduan Praktis Menguasai Vue.js*. Yogyakarta: Lokomedia, 2018.

[6] I. K. A. Herdinata Putra, D. Pramana, and N. L. P. Srinadi, "Sistem Manajemen Arsip Menggunakan Framework Laravel dan Vue.Js (Studi Kasus : BPKAD Provinsi Bali)," *J. Sist. Daninformatika*, vol. 13, no. 2, pp. 97–104, 2019.

[7] C. Chastro and E. Darmawan, "Perbandingan Pengembangan Front End Menggunakan Blade Template dan Vue Js," *J. Strateg. Maranatha*, vol. 2, no. 2, pp. 302–313, 2020.