Journal of Information Technology Ampera
Vol. 3, No. 2, August 2022 e-ISSN: 2774-2121
https://journal-computing.org/index.php/journal-ita/index

Pengembangan Sistem Informasi Perantara Bisnis Menggunakan *Framework* Flask

Yuny Tamariska Bota[1], Nina Setiyawati[2]

[1,2]Program Studi Teknik Informatika, Universitas Kristen Satya Wacana, Salatiga, Indonesia
Email: [1]672018269@student.uksw.edu, [2]nina.setiyawati@uksw.edu

Abstrak

Ketersediaan informasi secara cepat dan tepat diperlukan oleh *supplier* dalam menjalankan bisnis retail saat ini. Pengelolaan data transaksi yang dilakukan secara manual tentu tidak relevan lagi mengingat jumlah data, risiko kesalahan dalam perekapan serta waktu pemrosesan yang cukup lama. Masalah-masalah tersebut dapat diatasi dengan menggunakan sebuah sistem yang dapat melakukan perekapan data secara otomatis. Berdasarkan masalah tersebut, peneliti membangun Sistem Informasi Perantara Bisnis berbasis web yang dapat menyediakan informasi transaksi berupa laporan pembelian dan tagihan bagi *supplier*. Pembangunan sistem informasi tersebut menggunakan metode *waterfall model* dengan memanfaatkan *framework* Flask yang dimiliki Python. Hasil dari penelitian yang dilakukan menunjukkan bahwa aplikasi yang dibuat berjalan sesuai yang diharapkan. Hal ini dapat dilihat dari pengujian yang menunjukkan hasil yang valid, dimana program yang dibangun dapat berjalan sebagaimana mestinya. Dengan adanya aplikasi ini, diharapkan dapat membantu *supplier* dalam melihat ataupun mengunduh laporan pembelian serta melakukan perubahan atas tagihan yang diberikan. Dengan demikian kesalahan-kesalahan dapat lebih diminimalisir dan lebih efisien dalam penggunaan waktu.

Keywords: Sistem Informasi, *Waterfall*, Flask

1. PENDAHULUAN

Informasi telah menjadi salah satu kebutuhan yang sangat penting bagi kelangsungan berjalannya proses bisnis. Informasi digunakan untuk menilai kinerja dari suatu badan usaha, mengawasi dan mengendalikan jalannya proses bisnis [1]. Kecepatan pengolahan dan penyampaian informasi dibutuhkan setiap perusahaan, terutama pada perusahaan-perusahaan yang memiliki tingkat rutinitas yang tinggi dan memiliki data yang harus diolah [2]. Adanya kepentingan tersebut memaksa perusahaan menggunakan teknologi yang mampu mendukung kegiatan aktivitas kerja menjadi lebih efektif dan efisien dimanapun dan kapanpun

terlebih lagi dengan adanya dampak pandemik yang mengharuskan sebagian besar aktivitas kerja tidak dilakukan secara tatap muka.

Dalam dunia bisnis, *supplier* memegang peranan sangat penting dalam menjamin ketersediaan barang pasokan yang dibutuhkan oleh perusahaan [3]. Informasi mengenai transaksi perlu dicatat untuk nantinya diolah maupun disimpan sebagai dokumentasi. Hasil dari pengolahan tersebut akan digunakan sebagai pengambil keputusan untuk transaksi-transaksi yang akan dilakukan berikutnya. Akan tetapi, pengolahan data secara manual sudah tidak relevan lagi dalam menangani kebutuhan bisnis saat ini. Proses perekapan data secara manual hingga menjadi sebuah laporan tentu memakan waktu yang tidak sedikit [4] dan kemungkinan terjadinya kesalahan seperti ketidaklengkapan data, kesalahan dalam perhitungan atau hilangnya catatan-catatan penting menjadi lebih besar bila dibandingkan dengan perekapan secara otomatis [5]. Kejadian tersebut tentu memiliki pengaruh pada bisnis karena informasi yang dihasilkan kurang akurat. Berjalannya suatu transaksi juga tidak terlepas dari tagihan-tagihan yang perlu dibayarkan. Tagihan tersebut perlu persetujuan apakah *supplier* keberatan dengan nominal yang diberikan atau setuju dengan nominal tersebut. Jika proses tersebut dilakukan secara manual waktu yang dibutuhkan tidak sedikit. Baik perusahaan maupun *supplier* pasti menginginkan proses tersebut berjalan cepat sehingga tidak menggangu proses bisnis yang lain.

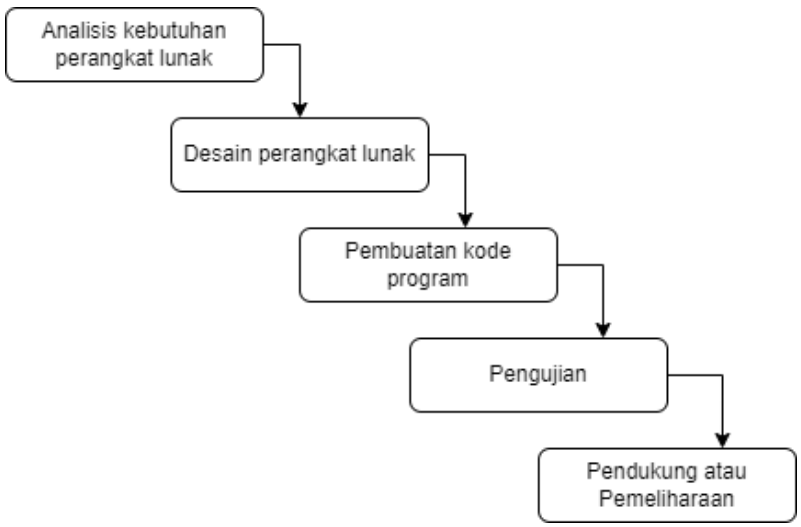
Pada penelitian ini dilakukan pembangunan Sistem Informasi Perantara Bisnis. Melalui aplikasi tersebut *supplier* dimudahkan dalam hal memperoleh informasi berupa laporan pembelian yang lebih cepat dan akurat. Selain itu, *supplier* juga dimudahkan dalam hal melakukan persetujuan atas tagihan-tagihan yang diberikan secara efektif dan efisien. Pembangunan aplikasi tersebut menggunakan metode *waterfall model*. *Waterfall model* melakukan pendekatan secara sistematis dan berurutan. Setiap tahapan yang dilalui harus menunggu tahap sebelumnya selesai sebelum lanjut ke tahap berikutnya [6]. Hal ini membuat prosesnya mudah dipahami oleh setiap pengembang. Selain itu, seluruh *requirement* telah diidentifikasi dan didokumentasikan sehingga mudah dalam pengelolaan [7]. *Tools* yang digunakan dalam membangun aplikasi tersebut adalah bahasa pemrograman Python dan *framework* Flask. Python merupakan bahasa pemrograman tingkat tinggi yang berfokus pada keterbacaan kode dengan penulisan baris kode yang lebih sedikit daripada bahasa pemrograman lainnya [8]. Hal ini membuat sintaks Python lebih sederhana dan mudah dibaca.

Flask merupakan *micro-framework* Python. Secara *default*, Flask tidak menyertakan komponen umum, seperti *form validation*, *database* atau lainnya. Namun, Flask mendukung ekstensi untuk menambahkan fungsionalitas tersebut

ke aplikasi yang akan dibuat [9]. Hal ini membuat fungsionalitas inti sederhana tetapi dapat diperluas dalam hal pengembangan [10]. Aplikasi yang dibangun pada penelitian ini diharapkan dapat mempermudah proses bisnis antara perusahaan dan *supplier* dalam hal *approve* atau persetujuan transaksi [11] serta menjamin ketersediaan informasi mengenai pembelian yang telah dilakukan.

2. METODE PENELITIAN

Pembangunan suatu aplikasi memerlukan sebuah metode dalam pembuatannya. Metode diperlukan agar aplikasi yang dibuat sesuai dengan tujuan yang ingin dicapai atau terhindar dari kegagalan proses pembangunan perangkat lunak (*software crisis*). Hal tersebut dapat dilakukan dengan menerapkan metode *System Development Life Cycle* (SLDC). SLDC merupakan metodologi untuk mengembangkan perangkat lunak [12]. Salah satu model SLDC yang sering digunakan adalah *waterfall model*. *Waterfall model* melakukan pendekatan secara sistematis dan berurutan. Setiap tahapan yang dilalui harus menunggu tahap sebelumnya selesai sebelum lanjut ke tahap berikutnya [6]. Pembangunan Sistem Informasi Perantara Bisnis menggunakan *waterfall model* dalam pembangunannya. *Waterfall model* memiliki sejumlah tahapan, di antaranya [13] analisis kebutuhan perangkat lunak, desain perangkat lunak, pembuatan kode program, pengujian, dan pendukung atau pemeliharaan (*maintenance*).



Gambar 1. *Waterfall model*

2.1. Analisis Kebutuhan Perangkat Lunak

Tahapan pertama penelitian pada Gambar 1 adalah analisis kebutuhan perangkat lunak dimana peneliti mengumpulkan informasi-informasi dengan cara melakukan wawancara dengan karyawan perusahaan mengenai kebutuhan sistem yang akan dipakai untuk membangun aplikasi. Melalui wawancara tersebut diproleh kebutuhan untuk pembangunan aplikasi, antara lain Python 3.8.x, Flask 1.1.2, Bootstrap, PostgreSQL, *library* pandas dan *library* apexcharts.

2.2. Desain perangkat lunak

Setelah memperoleh informasi tersebut selanjutnya masuk pada tahap kedua, yakni desain perangkat lunak. Tahap ini merupakan tahap perancangan sistem menggunakan *Unified Modeling Language* (UML). UML merupakan salah satu model yang digunakan dalam pembuatan perangkat lunak berbasis *object-oriented*. Diagram UML yang digunakan adalah *use case diagram* dan *activity diagram*. *Use case diagram* merupakan diagram yang merepresentasikan sebuah hubungan antara aktor dengan sistem sedangkan *activity diagram* merupakan gambaran *workflow* dari aktivitas yang berjalan di dalam suatu sistem [14].

2.3. Pembuatan kode program

Tahap ketiga merupakan tahap pembuatan kode program sesuai dengan desain yang telah dibuat pada tahap sebelumnya. Bahasa pemrograman yang digunakan untuk membangun aplikasi tersebut adalah Python dan *framework* Flask.

2.4. Pengujian

Tahap berikutnya dilakukan pengujian terhadap aplikasi yang telah selesai dibangun untuk memastikan keluaran yang dihasilkan sesuai dengan yang diinginkan menggunakan pengujian *Black Box Testing*. *Black Box Testing* merupakan pengujian perangkat lunak dari segi spesifikasi fungsional tanpa menguji desain dan kode program untuk mengetahui apakah fungsi, masukan dan keluaran dari perangkat lunak sesuai dengan spesifikasi yang dibutuhkan [15].

2.5. Pendukung atau pemeliharaan

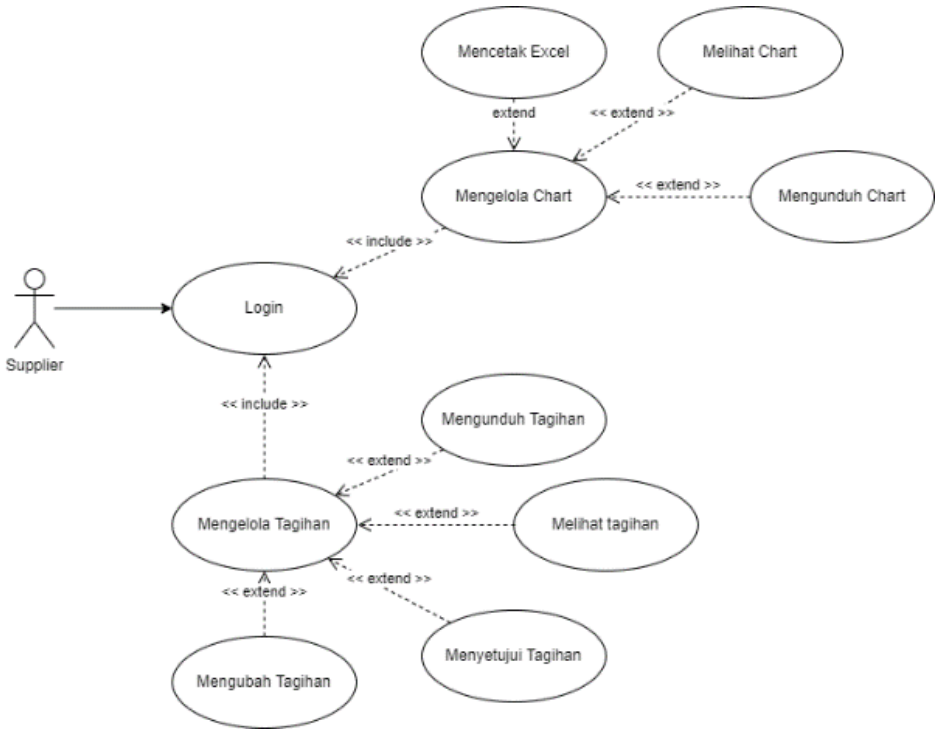
Tidak menutup kemungkinan sebuah aplikasi mengalami perubahan karena adanya adaptasi lingkungan baru atau adanya perubahan sesuai dengan permintaan pengguna, seperti penambahan fungsi. Oleh karena itu, tahap pendukung atau pemeliharaan diperlukan untuk menangani kasus tersebut.

3. HASIL DAN PEMBAHASAN

3.1. Desain Perangkat Lunak

3.1.1 *Use Case Diagram*

Use case diagram merupakan diagram yang menggambarkan hubungan antara aktor dengan sistem [16]. *Use case* digunakan untuk mengetahui fungsi apa saja yang ada di dalam sistem informasi dan siapa saja yang berhak menggunakan fungsi-fungsi tersebut [17].

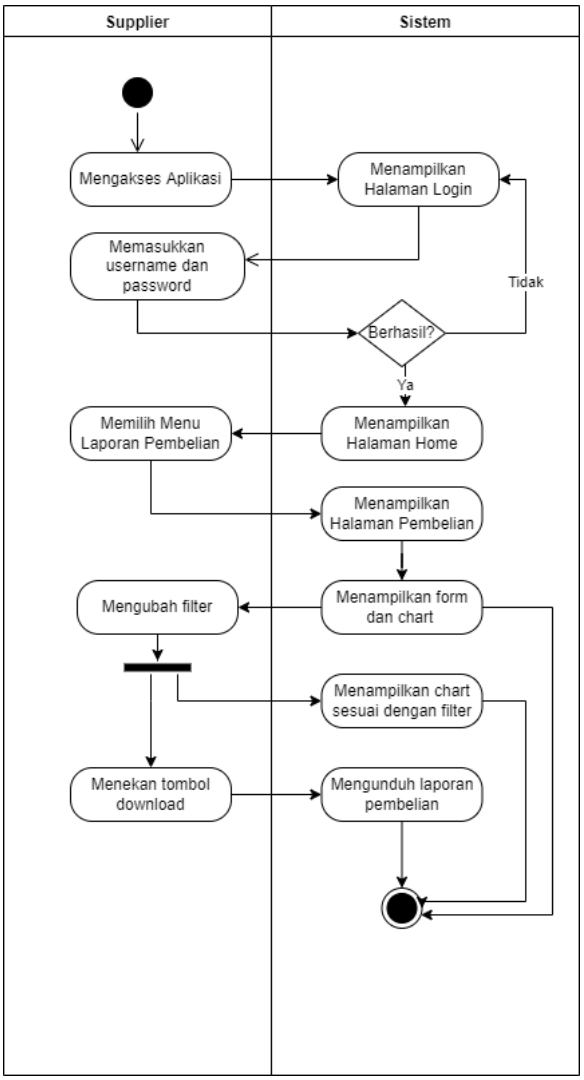


Gambar 2. *Use case diagram*

Gambar 2 merupakan *use case diagram* yang memperlihatkan *supplier* sebagai aktor yang mengakses aplikasi. Pada diagram tersebut *supplier* berperan untuk mengelola laporan pembelian baik melihat, mencetak maupun mengunduh laporan. Selain itu, *supplier* juga berperan dalam mengelola tagihan, seperti melihat, mengunduh, menyetujui dan melakukan perubahan pada nominal tagihan yang diberikan.

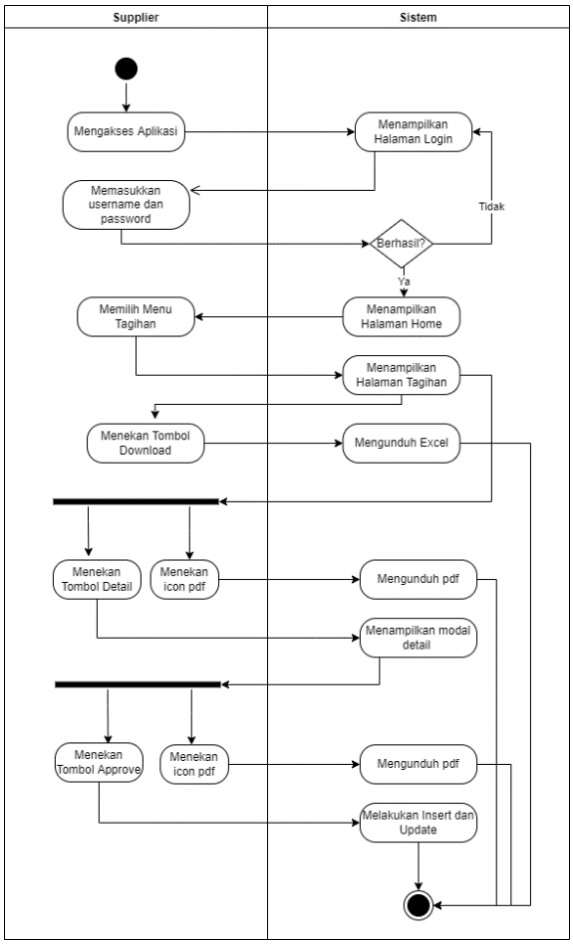
3.1.2 *Activity Diagram*

Activity diagram menggambarkan *workflow* (aliran kerja) atau aktivitas dari sebuah sistem atau proses bisnis [17]. *Activity diagram* menggambarkan aliran fungsionalitas dalam suatu sistem informasi. Secara lengkap, *activity diagram* mendefinisikan dimana *workflow* dimulai, dimana berhentinya, aktivitas apa yang terjadi selama *workflow*, dan bagaimana urutan kejadian aktivitas tersebut [18].



Gambar 3. *Activity diagram* laporan pembelian

Gambar 3 merupakan *activity diagram* laporan pembelian yang menjelaskan bagaimana *supplier* dapat mengelola laporan dengan fitur-fitur yang disediakan baik itu menampilkan *chart* maupun mengunduh laporan. Proses dimulai saat *supplier* mengakses halama tersebut. Sesaat setelah berada di halaman laporan, *chart* kemudian ditampilkan sesuai dengan data yang ada pada *form*. *Supplier* dapat mengganti pilihan periode untuk melihat data pembelian yang diinginkan. Tombol download yang disediakan berguna untuk mengunduh laporan dalam bentuk *excel*. Berbeda dengan *chart*, data yang disajikan dengan *excel* disajikan dengan lebih lengkap dan rinci sehingga data tersebut dapat digunakan untuk kepentingan bisnis yang lain.

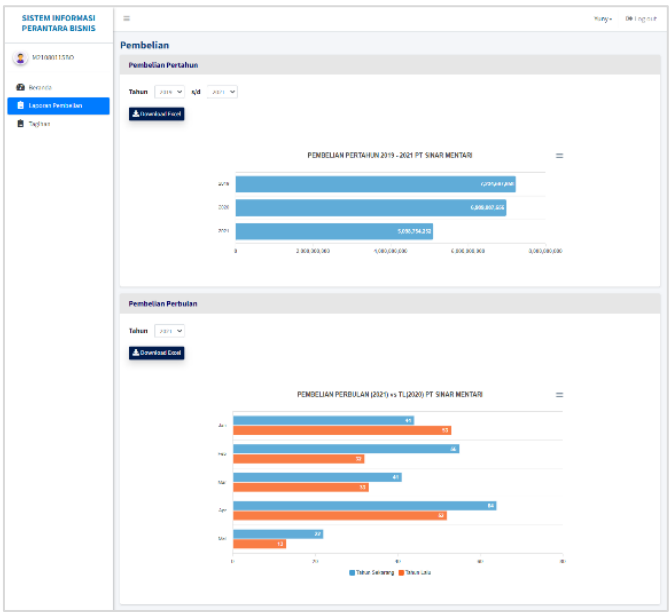


Gambar 4. *Activity diagram* tagihan

Gambar 4 merupakan *activity diagram* tagihan yang memperlihatkan bagaimana *supplier* dapat mengelola tagihan mulai dari melihat tagihan, mengunduh, menyetujui dan melakukan perubahan atas tagihan yang diberikan. Prosesnya dimulai ketika *supplier* berada di halaman tagihan. Di halaman tersebut, *supplier* dapat mengunduh *file* tagihan dalam bentuk *excel* dengan menekan tombol Download. Data tagihan yang diunduh disesuaikan dengan data pilihan pada *form*. Daftar tagihan yang disajikan dalam tabel juga dapat diunduh dalam bentuk pdf dengan format yang tentu berbeda dengan *excel*. Pernyataan setuju dengan tagihan yang tertera dapat dilakukan dengan menekan tombol Approve. Di samping itu, apabila *supplier* merasa perlu untuk mengganti nominal tagihan yang diberikan, hal tersebut dapat dilakukan dengan cara mengganti nominal pada *form* yang tersedia di modal detail.

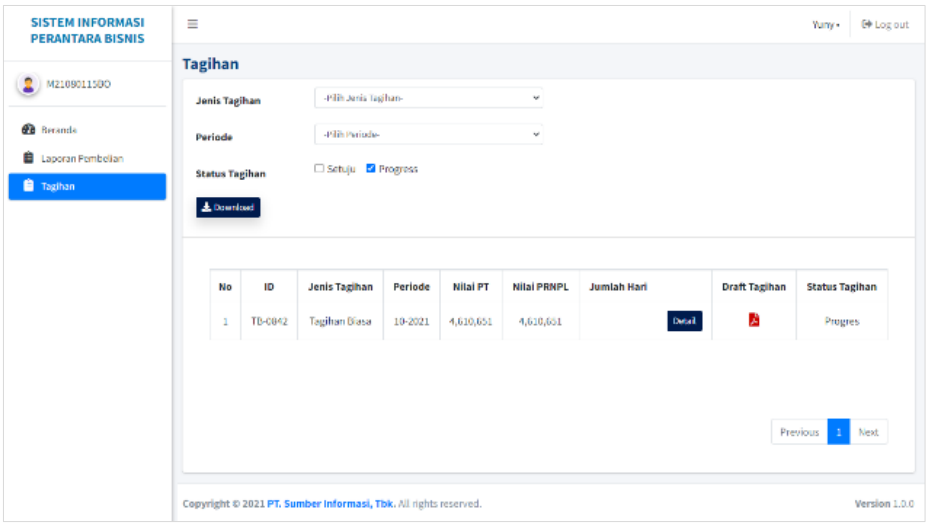
3.2. Pembuatan Kode Program

Berdasarkan rancangan sistem yang telah dibuat, berikut adalah hasil dari pembuatan aplikasi mulai dari tampilan hingga penulisan kode program menggunakan bahasa pemrograman Python dan *framework* Flask. Gambar 5 berikut ini merupakan tampilan dari Menu Laporan Pembelian. Halaman ini menyajikan laporan dalam bentuk *chart* dan *excel*.



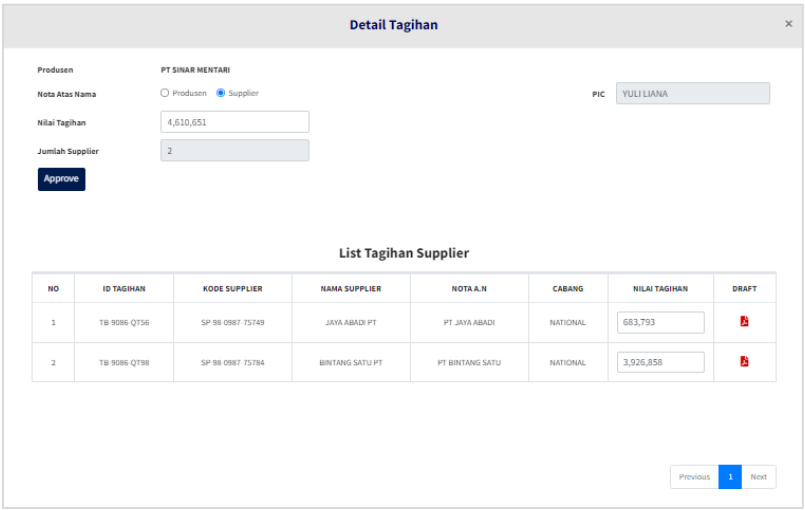
Gambar 5. Tampilan menu laporan pembelian

Data yang tampil di *chart* menyesuaikan dengan filter, sehingga apabila *supplier* ingin melihat data pembelian berdasarkan periode atau kategori yang lain, maka hal itu dapat dilakukan dengan memilih periode atau kategori yang sesuai. Selain itu, laporan tersebut juga dapat diunduh dalam .png, .svg, dan .csv. Laporan tersedia dalam banyak pilihan sehingga hal ini membantu dalam penyesuaian kebutuhan *supplier*.



Gambar 6. Tampilan menu tagihan

Gambar 6 merupakan tampilan utama dari Menu Tagihan. Jika *supplier* memiliki tagihan yang perlu dibayarkan, tagihan tersebut ditampilkan pada halaman ini. *Supplier* dapat mengunduh tagihan dengan format *excel* maupun pdf. Di samping itu, *supplier* dapat melakukan persetujuan terhadap tagihan yang diberikan dengan menekan tombol Approve. Saat tombol Approve ditekan, sistem melakukan perintah *insert* dan *update* sehingga perlu dipastikan terlebih dahulu apakah tagihan sudah sesuai atau belum. Oleh karena itu, disedikan data tagihan secara lebih rinci melalui modul detail. Tampilan modul detail tagihan dapat dilihat pada gambar 7.



Gambar 7. Tampilan modal detail tagihan

Modul tersebut dapat diakses dengan menekan tombol Detail pada halaman utama. Melalui modal ini, *supplier* dapat melakukan perubahan pada tagihan apabia keberatan dengan nominal yang diberikan. Hal tersebut dapat dilakukan dengan mengganti nominal yang terdapat pada tabel kolom nilai tagihan ataupun pada *form* nilai tagihan. Perubahan ini menjalankan proses *insert* dan *update* pada *database*. Kode program merupakan istiah dalam *programming* untuk mendeskripsikan instruksi tertulis. Penulisan kode pada aplikasi ini menggunakan bahasa pemrograman Python dengan penggunaan *framework* Flask. Gambar 8 berikut merupakan contoh kode program *controller* untuk menampilkan halaman.

```
1  @app.route("/laporan/pembelian", methods=['GET'])
2  @login_required
3  def pembelianView():
4      '''Controller Pembelian '''
5      print('======================')
6      print('In Controller Pembelian')
7      try:
8          now = datetime.now()
9          v_last_year = now.year - 3
10         v_periode = getPeriode(v_last_year)
11         v_region = getRegion()
           v_branch = getBranch()
```

```
12
13          print('End Controller Pembelian')
14          return
15  render_template('laporan/pembelian.html',
    title='Pembelian', periode=v_periode, region =
    v_region, branch = v_branch)
        except Exception as error:
            print('error :
16  {error}'.format(error=str(error)))
17          flash(str(error))
            return
    render_template('error/error_app.html')
```

Gambar 8. Kode program *controller* halaman pembelian

Pada baris 1 terdapat dekorator route() yang mengacu pada suatu URL untuk menjalankan suatu fungsi. Dalam kode program tersebut fungsi yang dijalankan adalah pembelianView(). Fungsi pembelianView() merupakan fungsi yang menampilkan halaman Laporan Pembelian. Selanjutnya, pada baris 2 terdapat dekorator *login_required* yang mengecek apakah *user* yang mengakses url tersebut telah *login* terlebih dahulu atau belum. Apabila ada user yang mengakses url tersebut, maka dekorator *login_required* dijalankan. Jika kondisinya benar atau sudah login, fungsi pembelianView() dieksekusi dan halaman pembelian ditampilkan. Selajutnya fungsi *render_template*() pada baris 15 berfungsi untuk me-*render template* html dengan pemberian argumen berupa nama *file* html dan variabel. Nilai dari variabel tersebut diterima oleh *template* Jinja yang disematkan pada html. Penggunaan *template* Jinja dapat dilihat pada Gambar 9 berikut.

```
1   <select class="form-control form-control-sm"
        name='inp_periode' id='inp_periode'>
2       {% if periode %}
3           {% for dt in periode %}
4               <option value="{{ dt }}">{{ dt
    }}</option>
5           {% endfor %}
6       {% endif %}
7   </select>
8
```

Gambar 9. Kode program jinja

Dapat dilihat bahwa penulisan sintaks menggunakan *template* Jinja mirip dengan sintaks Python. Sintaks pada kode program tersebut mengambil data yang diberikan *controller* pada fungsi *render_template*() untuk ditampilkan pada halaman html.

3.3. Pengujian

Pengujian hasil penelitian ini menggunakan *Black Box Testing*. *Black Box Testing* merupakan pengujian perangkat lunak dari segi spesifikasi fungsional tanpa menguji desain dan kode program untuk mengetahui apakah fungsi, masukan dan keluaran dari perangkat lunak sesuai dengan spesifikasi yang dibutuhkan [15]. Tabel 1 dan Tabel 2 berikut ini merupakan hasil pengujian Sistem Informasi Perantara Bisnis.

Tabel 1. Hasil Pengujian Menu Laporan pembelian

No	Skenario pengujian	Hasil yang diharapkan	Hasil pengujian	Hasil
1.	Memilih periode untuk menampilkan *chart*	Menampilkan *chart* sesuai dengan filter yang dipilih	Berhasil menampilkan *chart* sesuai dengan periode yang dipilih	Valid
2.	Memilih format unduh *chart*, antara lain .svg, .png atau .csv	Mengunduh *chart* sesuai format yang dipilih	Berhasil mengunduh *chart* sesuai format yang dipilih	Valid
3.	Menekan tombol Download untuk mengunduh *file excel*	Menampilkan Mencetak *excel* sesuai dengan periode yang terpilih	Berhasil mencetak *excel* sesuai dengan periode yang terpilih	Valid

Tabel 1 memperlihatkan hasil pengujian menu laporan pembelian menggunakan *Black Box Testing*. Skenario-skenario pengujian yang telah dilakukan menghasilkan kesimpulan yang valid, artinya program berjalan sesuai dengan hasil yang diharapkan. Begitu pula dengan hasil pengujian menu tagihan pada Tabel 2. Hasil pengujiannya juga valid, artinya program berjalan sesuai dengan hasil yang diharapkan.

Tabel 2. Hasil Pengujian Menu Tagihan

No	Skenario pengujian	Hasil yang diharapkan	Hasil pengujian	Hasil
1.	Memilih menu tagihan	Menampilkan halaman beserta tabel tagihan	Berhasil menampilkan halaman tagihan beserta tabel tagihan	Valid

2.	Memilih periode atau status tagihan dan menekan Download	Mengunduh *file excel* berdaasarkan pilihan periode atau status	Berhasil mengunduh *file* tagihan dalam *excel* berdasarkan periode atau status dipilih	Valid
3.	Menekan tombol detail untuk melihat data tagihan secara lebih detail	Menampilkan modal detail tagihan	Berhasil mengunduh *file* tagihan dalam *excel* berdasarkan periode atau status yang diberikan	Valid
4.	Menekan tombol icon pdf pada tabel untuk mengunduh *file* tagihan	Mengunduh *file* tagihan dalam bentuk pdf	*File* tagihan dalam bentuk pdf berhasil diunduh	Valid
5.	Menekan tombol Approve untuk menyetujui tagihan	Menampilkan pesan "Tagihan berhasil di-*approve*" dan *redirect* ke halaman utama	Menampilkan "Tagihan berhasil di-*approve*" dan *redirect* ke halaman utama	Valid

4. KESIMPULAN

Berdasarkan penelitian yang telah dilakukan, dihasilkan sebuah Sistem Informasi Perantara Bisnis yang dapat membantu *supplier* dalam mengelola laporan pembelian serta tagihan berbasis web. Penggunaan bahasa pemrograman Python dan *framework* Flask dalam pembangunan aplikasi membantu pengembang dalam membuat sebuah web terstruktur yang mempunyai sifat *simplicity*, yang mana memudahkan pengembangan aplikasi dengan hasil yang optimal dan sesuai dengan yang kebutuhan. Hasil penelitian menunjukkan hasil yang valid, dimana fitur-fitur yang dibangun dapat berjalan sebagaimana mestinya. Sistem Informasi Perantara Bisnis nantinya membantu *supplier* dalam melihat, mengunduh laporan pembelian maupun tagihan serta melakukan perubahan atas tagihan yang diberikan. Dengan demikian, kesalahan-kesalahan dapat lebih diminimalisir dan lebih efisien dalam penggunaan waktu.

DAFTAR PUSTAKA

[1] D. Luhukay, Y. Kurniawan, and T. Titan, "Analisis dan Perancangan Sistem Informasi Penjualan dan Persediaan pada PT. Xyz," *ComTech Comput. Math. Eng. Appl.*, vol. 4, no. 1, p. 162, 2013, doi: 10.21512/comtech.v4i1.2696.

[2] 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.

[3] R. Cocroach, "Aplikasi Metode Analytical Hierarchy Process Dalam Menentukan Kriteria."

[4] Fatkhudin, "Toko Elektronik Lubada Jaya Kajen Dengan," vol. 6, no. 1, pp. 23–36, 2016.

[5] M. Abdurahman, "Sistem Informasi Pengolahan Data Pembelian Dan Penjualan Pada Toko Koloncucu Ternate," *IJIS - Indones. J. Inf. Syst.*, vol. 2, no. 1, 2017, doi: 10.36549/ijis.v2i1.22.

[6] A. A. Wahid, "Analisis Metode Waterfall Untuk Pengembangan Sistem Informasi," *J. Ilmu-ilmu Inform. dan Manaj. STMIK*, no. November, pp. 1–5, 2020.

[7] D. S. Budi, T. A. Y. Siswa, and H. Abijono, "Analisis Pemilihan Penerapan Proyek Metodologi Pengembangan Rekayasa Perangkat Lunak," *Teknika*, vol. 5, no. 1, pp. 24–31, 2017, doi: 10.34148/teknika.v5i1.48.

[8] F. Armash Aslam, H. Nabeel Mohammed Jummal Musab Mohd Munir Murade Aaraf Gulamgaus, and P. S. Lokhande Assistant Professor, "Efficient Way Of Web Development Using Python And Flask," *Int. J. Adv. Res. Comput. Sci.*, vol. 6, no. 2, pp. 54–57, 2015, [Online]. Available: www.ijarcs.info.

[9] "What does 'micro' mean?" https://flask.palletsprojects.com/en/1.0.x/foreword/ (accessed Jun. 21, 2022).

[10] F. A. Aslam, H. N. Mohammed, J. Musab, and M. Munir, "International Journal of Advanced Research in Computer Science Available Online at www.ijarcs.info Efficient Way Of Web Development Using Python And Flask," vol. 6, no. 2, pp. 54–57, 2015.

[11] S. Monalisa and D. Apsyarin, "Rancang Bangun Sistem Informasi Supply Chain Management Distribusi Barang Dan Jasa Berbasis Web," *J. Ilm. Rekayasa dan Manaj. Sist. Inf.*, vol. 7, no. 2, pp. 138–144, 2021, [Online]. Available: http://ejournal.uin-suska.ac.id/index.php/RMSI/article/view/13143.

[12] M. M. Lucini, P. J. Van Leeuwen, and M. Pulido, "IMPLEMENTASI SOFTWARE DEVELOPMENT LIFE CYCLE (SDLC) DALAM PENERAPAN PEMBANGUNAN APLIKASI PERANGKAT LUNAK," *J. Teknol. Inf.*, vol. 9, no. 2, pp. 681–707, 2021, doi: 10.1137/19M1297300.

[13] R. A. S and M. Shalahudin, *Rekayasa Perangkat Lunak Terstruktur dan Berorientasi Objek*. Bandung: Informatika Bandung, 2013.

[14] M Teguh Prihandoyo, "Unified Modeling Language (UML) Model Untuk Pengembangan Sistem Informasi Akademik Berbasis Web," *J. Inform. J.*

Pengemb. IT, vol. 3, no. 1, pp. 126–129, 2018.

[15] W. N. Cholifah, Y. Yulianingsih, and S. M. Sagita, "Pengujian Black Box Testing pada Aplikasi Action & Strategy Berbasis Android dengan Teknologi Phonegap," *STRING (Satuan Tulisan Ris. dan Inov. Teknol.*, vol. 3, no. 2, p. 206, 2018, doi: 10.30998/string.v3i2.3048.

[16] R. K. Ngantung and M. A. I. Pakereng, "Model Pengembangan Sistem Informasi Akademik Berbasis User Centered Design Menerapkan Framework Flask Python," *J. Media Inform. Budidarma*, vol. 5, no. 3, p. 1052, 2021, doi: 10.30865/mib.v5i3.3054.

[17] J. J. Robinson, "DIAGRAM: A Grammar for Dialogues," *Commun. ACM*, vol. 25, no. 1, pp. 27–47, 1982, doi: 10.1145/358315.358387.

[18] L. P. Dewi, U. Indahyanti, and Y. H. S, "Pemodelan Proses Bisnis Menggunakan Activity Diagram Uml Dan Bpmn (Studi Kasus Frs Online)," *Informatika*, pp. 1–9, 2017.