KAMPUS AKADEMIK PUBLISING
Jurnal Ilmiah Ekonomi Dan Manajemen (JIEM)
Vol.4, No.7 Juli 2026
e-ISSN: 3025-7859, p-ISSN : 3025-7972, Hal 444-458
DOI : https://doi.org/10.61722/jiem.v4i7.12267

Perancangan Sistem Informasi Manajemen Keuangan pada Masjid Al Mustaqim Helvetia Tengah Berbasis Web dengan Metode *Rapid Application Development (RAD)*

Muhammad Farhan Mahendra[1], Demonius Sarumaha[2], Saiful Amir[3]
[1-3]Program Studi Ilmu Komputer, Univertas Nahdlatul Ulama Sumatera Utara, Indonesia
*Penulis Korespondensi: farhanmahendra6@gmail.com

Abstract. *Financial management in most mosques, including Al Mustaqim Mosque in Helvetia Tengah, is still carried out manually using paper-based cash books and reports, making it prone to recording errors, difficult to trace, and lacking transparency for both worshippers and mosque administrators. This research aims to design and build a web-based Financial Management Information System for Al Mustaqim Mosque in Helvetia Tengah so that the recording of income, expenses, and financial reporting can be carried out in a computerized, accurate, and transparently accessible manner for the public. The development method used is Rapid Application Development (RAD), consisting of five stages: business modeling to map the mosque's financial management business process, data modeling to design the transaction and user database structure, process modeling to translate the business process into system functions, application generation to build the application using the Flask framework. The result of this research is a web-based information system built using Flask, SQLAlchemy, and an SQLite database, equipped with a public portal for financial transparency to worshippers, an admin dashboard, CRUD (create, read, update, delete) operations for income and expense transactions, Based on the implementation and testing results, it can be concluded that the developed system is able to improve the efficiency of recording time, the accuracy of balance calculations, and the transparency of mosque financial management compared to the previously used manual recording method.*

Keywords: *Information System, Financial Management, Mosque, Rapid Application Development (RAD)*

Abstrak. Pengelolaan keuangan pada sebagian besar masjid, termasuk Masjid Al Mustaqim Helvetia Tengah, masih dilakukan secara manual menggunakan pencatatan buku kas dan laporan berbasis kertas, sehingga rawan terhadap kesalahan pencatatan, sulit ditelusuri, dan kurang transparan bagi jamaah maupun pengurus. Penelitian ini bertujuan untuk merancang dan membangun Sistem Informasi Manajemen Keuangan berbasis web pada Masjid Al Mustaqim Helvetia Tengah agar proses pencatatan pemasukan, pengeluaran, dan pelaporan keuangan dapat dilakukan secara terkomputerisasi, akurat, dan dapat diakses secara transparan oleh masyarakat. Metode pengembangan yang digunakan adalah *Rapid Application Development* (RAD), yang terdiri atas lima tahapan, yaitu *business* modeling untuk memetakan alur proses bisnis pengelolaan keuangan masjid, data modeling untuk merancang struktur basis data transaksi dan pengguna, *process* modeling untuk menerjemahkan proses bisnis ke dalam fungsi sistem, *application generation* untuk membangun aplikasi menggunakan kerangka kerja *Flask*,. Hasil penelitian berupa sebuah sistem informasi berbasis web yang dibangun menggunakan *Flask, SQLAlchemy*, dan basis data *SQLite*, dilengkapi fitur portal publik untuk transparansi informasi keuangan kepada jamaah, *dashboard* admin, operasi *CRUD (create, read, update, delete*) untuk transaksi pemasukan dan pengeluaran, modul pembuatan laporan keuangan per periode, serta pengaturan profil masjid.

Kata Kunci: Sistem Informasi, Manajemen Keuangan, Masjid, *Rapid Application Development (RAD)*

1. LATAR BELAKANG

Perkembangan teknologi infomasi berkembang pesat, dan sistem manajemen keuangan organisasi sudah beralih ke sistem digital, masih banyak pengelolaan keuangan khususnya pada Lembaga keagamaan menggunakan sistem pengelolaan manual. Contohnya pada Lembaga Badan Kemakmuran Masjid Al Mustaqim yang

Naskah Masuk: 01 Juli 2026; Revisi: 05 Juli 2026; Diterima: 06 Juli 2026; Terbit: 25 Juli 2026.

pengelolaannya masih dilakukan secara manual. Masjid tidak hanya tempat pelaksanaan ritual ibadah, karena mesjid juga pusat kegiatan sosial, pendidikan, dan pelayanan masyarakat. Seiring ber-tambahnya aktivitas di dalam mesjid, kebutuhan akan pengelolaan administrasi dan keuangan yang lebih rapi juga semakin besar. Pada banyak masjid, termasuk Masjid Al Mustaqim Helvetia Tengah, aktivitas keuangan berlangsung hampir setiap hari-mulai dari penerimaan infak, sedekah, dan donasi, hingga pengeluaran untuk operasional, kegiatan dakwah, dan pemeliharaan fasilitas. Seluruh kegiatan ini menuntut adanya sistem pencatatan yang tertib, transparan, dan mudah dipertanggungjawabkan kepada jamaah.

Pada praktiknya, sebagian besar pengelolaan keuangan masjid masih dikerjakan secara manual menggunakan buku kas atau lembar kerja sederhana. Metode ini memang mudah dilakukan, tetapi mempunyai banyak keterbatasan. Kesalahan pencatatan, data yang tercecer, atau keterlambatan penyusunan laporan merupakan masalah yang sering muncul. Tidak sedikit pengurus yang kesulitan melacak transaksi tertentu atau menyusun laporan periodik dengan cepat karena harus menelusuri catatan satu per satu. Kondisi ini bukan hanya menghambat proses administrasi, tetapi juga dapat menurunkan tingkat kepercayaan jamaah apabila laporan keuangan tidak tersusun dengan baik (Walingkas & Saian, 2023).

Perkembangan teknologi informasi sebenarnya membuka peluang besar bagi masjid untuk meningkatkan tata kelola administrasi melalui pemanfaatan sistem informasi (Pradhana et al., 2022). Sistem informasi berbasis web, misalnya, memungkinkan pengurus mengelola data secara terpusat, mengakses informasi kapan saja, serta menghasilkan laporan secara otomatis (Purnasari et al., 2022). Dibandingkan pencatatan manual, sistem berbasis web jauh lebih fleksibel, mudah dioperasikan, dan minim risiko kehilangan data. Selain itu, aspek keamanan seperti pengaturan hak akses dapat diterapkan sehingga data keuangan lebih terlindungi (Wahyudi et al., 2025).

Namun, sebuah sistem tidak hanya harus berfungsi, tetapi juga mudah digunakan oleh pengurus dengan latar belakang teknologi yang berbeda-beda. Karena itu, desain antarmuka (UI) dan pengalaman pengguna (UX) menjadi aspek penting dalam proses perancangan. Sistem yang sederhana, informatif, dan tidak membingungkan akan membantu pengurus menjalankan tugas tanpa hambatan. Dalam tahap pengembangan, pemilihan metode yang tepat juga sangat berpengaruh. Metode *Rapid Application Development (RAD)* menjadi salah satu pendekatan yang sesuai untuk proyek seperti ini karena memungkinkan proses pembuatan sistem dilakukan secara bertahap dan cepat, dengan adanya umpan balik langsung dari pengguna di setiap tahapan (Hanum et al., 2026); (Hidayat & Hati, 2021). Dengan demikian, sistem yang dihasilkan lebih sesuai dengan kebutuhan lapangan.

Masjid Al Mustaqim Helvetia Tengah saat ini menghadapi tantangan tersebut. Dengan jumlah jamaah yang cukup besar dan kegiatan rutin yang berjalan terus-menerus, proses pencatatan dan pelaporan keuangan sering kali memakan waktu. Transaksi yang jumlahnya banyak baik harian maupun bulanan membuat pengurus perlu bekerja ekstra dalam menyusun laporan yang akurat. Sistem manual yang digunakan saat ini belum mampu memberikan efisiensi dan transparansi seperti yang diharapkan. Selain itu, keterbatasan perangkat dan pengalaman teknis pengurus juga menjadi pertimbangan dalam memilih solusi yang tepat.

Melihat kondisi tersebut, pengembangan sistem informasi manajemen keuangan berbasis web menjadi sebuah kebutuhan yang mendesak. Sistem ini diharapkan dapat membantu pengurus melakukan pencatatan transaksi secara lebih terstruktur, menyusun

laporan keuangan secara otomatis, dan meningkatkan akuntabilitas pengelolaan dana di hadapan jamaah. Dengan menerapkan metode *Rapid Application Development (RAD)*, proses pembangunan sistem dapat dilakukan lebih adaptif dan cepat, sehingga hasil akhirnya benar-benar mencerminkan kebutuhan pengguna.

2. METODE PENELITIAN

Metode penelitian yang digunakan dalam penelitian ini adalah pendekatan kualitatif deskriptif dengan model pengembangan sistem *Rapid Application Development* (RAD). Pendekatan kualitatif digunakan untuk mengidentifikasi kebutuhan pengguna melalui observasi, dokumentasi, dan wawancara dengan pengurus Masjid Al-Mustaqim Helvetia Tengah, khususnya bendahara dan ketua takmir, sehingga sistem yang dikembangkan sesuai dengan kondisi nyata di lapangan. Selanjutnya, pengembangan sistem dilakukan menggunakan metode RAD yang meliputi tahapan business modeling, data modeling, process modeling, application generation, dan testing secara iteratif dengan melibatkan pengguna dalam setiap tahap penyempurnaan. Sistem dibangun menggunakan framework Flask, basis data SQLite, dan SQLAlchemy sebagai ORM, sedangkan pengujian dilakukan secara fungsional menggunakan black box testing untuk memastikan seluruh fitur berjalan sesuai kebutuhan. Melalui metode ini dihasilkan sistem manajemen keuangan masjid berbasis web yang mampu mendukung pencatatan transaksi, penyusunan laporan keuangan, serta meningkatkan transparansi dan efisiensi pengelolaan keuangan masjid.

3. HASIL DAN PEMBAHASAN

Implementasi Metode *Rapid Application Development* (RAD)

1. *Business Modeling*

Hasil *business modeling* mengidentifikasi bahwa aktivitas keuangan masjid bersifat linear: pemasukan dari jamaah (infak, sedekah, wakaf) dicatat oleh bendahara, kemudian dana tersebut dikeluarkan untuk kebutuhan operasional (listrik, air, honor penceramah, santunan) dan kegiatan sosial. Pemodelan bisnis ini diterjemahkan langsung ke dalam satu entitas utama pada sistem, yaitu *Transaction*, yang direpresentasikan sebagai satu baris data dengan atribut tipe bernilai "Pemasukan" atau "Pengeluaran".

Aktor tunggal yang teridentifikasi pada *business modeling*, yaitu Admin/Pengurus Masjid, direalisasikan menjadi entitas *User* pada sistem. Karena ruang lingkup penelitian membatasi sistem hanya digunakan oleh pengurus (bukan publik), maka pada inisialisasi awal aplikasi, sistem secara otomatis membuat satu akun admin bawaan (*username admin*, *password* 1234) apabila belum ada akun yang terdaftar. Pendekatan *default account* ini adalah wujud *application generation* yang cepat, sesuai prinsip *RAD* dimana pengguna dapat langsung mencoba sistem tanpa proses instalasi akun yang rumit terlebih dahulu, sekaligus tetap dapat diubah kemudian melalui mekanisme keamanan yang lebih lengkap.

2. *Data Modeling*

Pemodelan data yang telah dirancang diimplementasikan secara konkret pada berkas *models.py*. Terdapat dua kelas model utama, yaitu User dan Transaction, keduanya diturunkan dari db.Model milik *SQLAlchemy*.

```
class User(db.Model): __tablename__ = 'users' id = db.Column(db.Integer, primary_key=True) username = db.Column(db.String(80), unique=True, nullable=False) password_hash = db.Column(db.String(255), nullable=False) class
```

```
Transaction(db.Model):  __tablename__ = 'transactions'  id = db.Column(db.Integer, primary_key=True)  dt = db.Column(db.DateTime, default=datetime.utcnow, nullable=False)  tipe = db.Column(db.String(20), nullable=False)  kategori = db.Column(db.String(80), nullable=True)  keterangan = db.Column(db.String(255), nullable=True)  jumlah = db.Column(db.Integer, nullable=False)  created_at = db.Column(db.DateTime, default=datetime.utcnow
```

Model User menyimpan kredensial dalam bentuk *password_hash*, bukan teks polos, memanfaatkan fungsi *generate_password_hash* dan *check_password_hash* dari *Werkzeug*. Keputusan desain ini penting karena meskipun sistem berskala kecil, prinsip keamanan dasar tetap harus dijaga mengingat data ini menyangkut hak akses terhadap keuangan masjid. Model *Transaction* dirancang dengan pendekatan tanda (*signed value*) pada kolom jumlah: nilai positif merepresentasikan pemasukan, sedangkan nilai negatif merepresentasikan pengeluaran. Pendekatan ini dipilih karena sangat menyederhanakan perhitungan saldo, saldo kas cukup dihitung dengan menjumlahkan seluruh nilai jumlah tanpa perlu logika percabangan tambahan pada level *query*. Kolom dt menyimpan waktu transaksi aktual (misalnya waktu infak Jumat diterima), sedangkan *created_at* menyimpan waktu pencatatan ke sistem, sehingga keduanya dapat berbeda apabila bendahara mencatat transaksi secara mundur (*backdate*).

Setiap objek *Transaction* juga memiliki *method as_dict()* yang mengonversi baris data menjadi struktur dictionary *Python*, kemudian secara otomatis diserialisasi menjadi *JSON* oleh *Flask* ketika dikembalikan melalui *endpoint* API. Method ini menjadi jembatan penting antara *data modeling* dan *process modeling*, karena struktur *JSON* yang dihasilkan persis menjadi bentuk data yang dikonsumsi oleh *JavaScript* pada sisi antarmuka *dashboard* dan portal. Pada tingkat konfigurasi basis data, *config.py* menetapkan lokasi penyimpanan berkas *SQLite* pada instance/finance.db, dan seluruh pembuatan tabel dilakukan otomatis melalui *db.create_all()* saat aplikasi pertama kali dijalankan. Pendekatan ini konsisten dengan pembahasan BAB III bahwa *data modeling* pada metode *RAD* tidak memerlukan proses migrasi skema yang berat di awal, karena *Flask-Migrate* tetap disediakan untuk menangani perubahan skema pada iterasi berikutnya tanpa kehilangan data yang sudah tersimpan.

3. *Process Modeling*

Pemodelan proses direalisasikan melalui kombinasi antara *forms*.py sebagai lapisan validasi *input* dan *app.py* sebagai lapisan logika alur. *Form TransactionForm*, misalnya, membatasi *field* tipe hanya boleh bernilai Pemasukan atau Pengeluaran melalui *SelectField*, serta mewajibkan *field* jumlah diisi melalui validator *DataRequired*. Pembatasan pada level *form* ini mencegah data yang tidak valid sampai ke proses penyimpanan basis data.

Alur proses pencatatan transaksi paling menonjol terlihat pada logika penyesuaian tanda nominal, yang diterapkan secara konsisten baik pada *route* berbasis *form* (*edit_transaction*) maupun pada *endpoint* API (*api_add_transaction dan api_update_transaction*):

```
if tipe == 'Pengeluaran' and jumlah > 0:  jumlah = -abs(jumlah)
```

Baris kode di atas memastikan bahwa berapa pun angka yang diketik bendahara pada kolom nominal, sistem akan otomatis mengubahnya menjadi nilai negatif apabila tipe transaksinya adalah pengeluaran. Dengan demikian bendahara tidak perlu mengingat aturan tanda minus secara manual seperti pada pencatatan buku kas konvensional, cukup

memilih tipe transaksi yang benar, dan sistem yang menyesuaikan nilainya. Ini adalah salah satu bentuk nyata bagaimana proses bisnis manual (mencatat pemasukan dan pengeluaran secara terpisah) diterjemahkan menjadi aturan otomatis pada level aplikasi.

Alur autentikasi diproses pada *route* /*login*, yang menerima metode *GET* dan *POST*. Ketika *form* divalidasi *(form.validate_on_submit()),* sistem mencari *user* berdasarkan *username*, kemudian memverifikasi *password* menggunakan *method check_password()* milik model *User*. Apabila kredensial valid, sistem menyimpan *user_id* dan *username* ke dalam *session* sehingga permintaan-permintaan berikutnya dapat dikenali sebagai sesi yang sama. Selanjutnya, seluruh proses pengambilan data ringkasan diproses melalui *endpoint* /api/summary, yang menghitung total pemasukan dan pengeluaran bulan berjalan menggunakan agregasi *SQL (db.func.sum)* yang difilter berdasarkan tanggal awal bulan dan tanda nilai jumlah:

```
pemasukan =
db.session.query(db.func.coalesce(db.func.sum(Transaction.jumlah), 0)).filter(
Transaction.jumlah > 0, Transaction.dt >= start ).scalar() or 0 pengeluaran =
db.session.query(db.func.coalesce(db.func.sum(Transaction.jumlah), 0)).filter(
Transaction.jumlah < 0, Transaction.dt >= start ).scalar() or 0 saldo = pemasukan +
pengeluaran
```

Proses agregasi ini berjalan sepenuhnya di sisi server sehingga perhitungan saldo selalu konsisten dan tidak bergantung pada kalkulasi manual di sisi *client*. *Endpoint* yang sama juga mengembalikan lima transaksi terbaru (*latest*) yang diurutkan berdasarkan *created_at*, sehingga baik portal publik maupun *dashboard* admin dapat menggunakan satu sumber data yang seragam. Pendekatan satu *endpoint* untuk banyak konsumen (*reusable endpoint*) ini juga sejalan dengan prinsip modular dan *reusable* components pada *RAD*.

Selain */api/summary*, sistem juga menyediakan */api/latest* untuk daftar transaksi ringkas, /api/transactions dengan dukungan filter bulan (parameter month=YYYY-MM) untuk kebutuhan laporan periode tertentu, serta empat *endpoint CRUD* penuh pada */api/transaction (POST* untuk tambah, *PUT* untuk ubah, *DELETE* untuk hapus) yang seluruhnya dilindungi oleh login_required. Struktur ini menunjukkan penerapan pola *REST API*, di mana setiap resource memiliki representasi URI yang konsisten dan dapat diakses menggunakan metode *HTTP* standar.

4. *Application Generation*

Tahap pembuatan aplikasi (*application generation*) merupakan tahap di mana seluruh hasil *business modeling*, *data modeling*, dan *process modeling* disatukan menjadi aplikasi yang dapat dijalankan dan diuji langsung oleh pengguna. Pada penelitian ini, proses tersebut menghasilkan tujuh halaman utama yang saling terhubung melalui routing *Flask*: portal publik, *login*, *dashboard*, form kelola transaksi (pemasukan/pengeluaran dalam satu antarmuka), halaman edit transaksi, laporan, dan pengaturan.

Application generation juga mencakup penyediaan skrip pendukung, yaitu *import_csv_to_db.py*, yang memungkinkan bendahara memindahkan riwayat transaksi lama dari format CSV (misalnya hasil ekspor sistem pencatatan berbasis *spreadsheet* sebelumnya) ke dalam database *SQLite* yang baru. Skrip ini membaca berkas transaksi.csv baris demi baris, mencoba beberapa format tanggal yang umum digunakan, kemudian menyimpan setiap baris sebagai objek *Transaction* baru. Adanya skrip migrasi ini penting agar transisi dari sistem manual ke sistem digital tidak menghapus jejak

historis keuangan masjid, sebuah kekhawatiran yang secara eksplisit menjadi bagian dari tahap *cutover*.

Keseluruhan proses *application generation* dilakukan secara bertahap, dimulai dari fungsi inti *CRUD* transaksi, kemudian dilanjutkan dengan fitur pendukung seperti tombol nominal cepat (*quick amount*), penyaringan laporan per bulan, dan halaman pengaturan profil masjid. Pola pembangunan bertahap seperti ini merupakan ciri khas *RAD* yang membedakannya dari model *waterfall*, karena setiap modul dapat langsung diuji dan digunakan begitu selesai dibangun, tanpa harus menunggu keseluruhan sistem rampung terlebih dahulu.

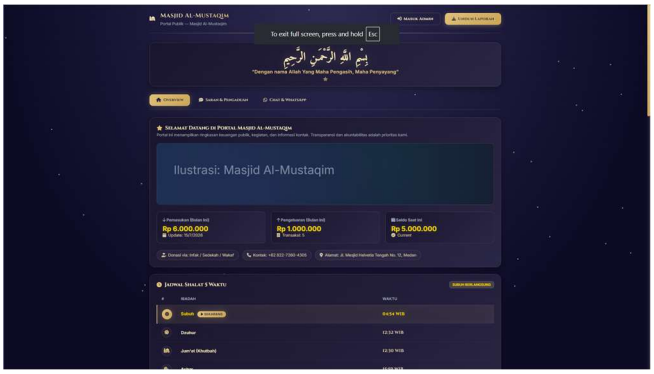
5. *Testing*

Pengujian dilakukan secara iteratif setiap kali satu modul fungsional selesai dibangun, sejalan dengan sifat *RAD* yang tidak menunda pengujian hingga akhir proyek. Pengujian awal difokuskan pada jalur autentikasi (*login* gagal dan berhasil), dilanjutkan dengan pengujian form transaksi (validasi nominal, penyesuaian tanda otomatis), kemudian pengujian terhadap *endpoint* API menggunakan *browser developer tools* untuk memastikan format *JSON* yang dikembalikan sesuai dengan yang dibutuhkan oleh antarmuka.

Implementasi Antarmuka Sistem

Hasil implementasi antarmuka sistem secara rinci berdasarkan tangkapan layar (*screenshot*) aplikasi yang telah selesai dibangun. Setiap halaman dijelaskan mulai dari elemen visual yang tampil, fungsi tiap komponen, alur proses di baliknya, hubungan dengan basis data dan API, keterkaitan dengan tahapan *RAD*, hingga manfaat langsung yang dirasakan oleh bendahara maupun jamaah masjid.

1. Halaman Portal Publik



Gambar 1. Halaman Portal Publik

Halaman portal publik adalah halaman pertama yang ditemui oleh siapa pun yang mengakses alamat root aplikasi, dilayani oleh *route* '/' pada *app.py* melalui fungsi portal() yang me-render *template portal.html* tanpa memerlukan proses *login*. Pada bagian atas halaman ditampilkan identitas masjid (Masjid Al-Mustaqim) beserta kalimat basmalah dan terjemahannya, yang berfungsi memberikan nuansa keislaman sekaligus menegaskan bahwa portal ini memang milik lembaga keagamaan, bukan aplikasi keuangan umum.

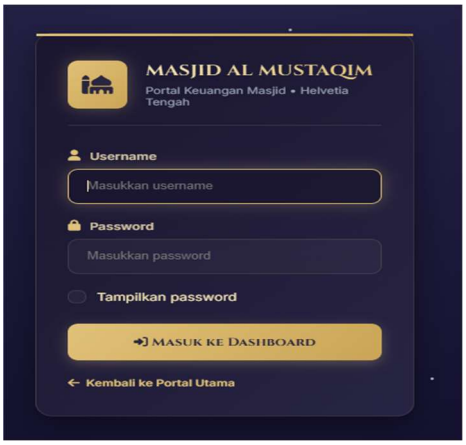
Di bawah identitas masjid terdapat tiga menu navigasi, yaitu *Overview*, Saran & Pengaduan, dan *Chat* & *WhatsApp*, yang memberi ruang bagi jamaah tidak hanya untuk melihat kondisi keuangan tetapi juga menyampaikan masukan atau menghubungi pengurus secara langsung. Bagian paling penting dari halaman ini adalah tiga kartu ringkasan: Pemasukan (Bulan Ini), Pengeluaran (Bulan Ini), dan Saldo Saat Ini. Ketiga angka tersebut bukan data statis yang ditulis manual oleh pengurus, melainkan diambil

secara langsung dari *endpoint* /*api/summary* setiap kali halaman dimuat, sehingga apa yang dilihat jamaah selalu mencerminkan kondisi kas terkini tanpa jeda waktu publikasi seperti pada laporan bulanan konvensional.

Fungsi *api_summary()* pada *backend* menghitung total pemasukan dan pengeluaran bulan berjalan menggunakan agregasi terhadap tabel *Transaction*, kemudian mengembalikan hasilnya dalam format *JSON* yang diambil oleh kode *JavaScript* pada portal.html dan langsung disisipkan ke dalam elemen kartu ringkasan. Dengan mekanisme ini, transparansi keuangan yang menjadi salah satu manfaat penelitian benar-benar terwujud secara teknis, bukan sekadar janji konsep. Selain ringkasan keuangan, portal juga menampilkan jadwal shalat lima waktu lengkap dengan penanda waktu yang sedang berlangsung (misalnya status "Subuh Berlangsung"), informasi kontak masjid, serta alamat lengkap. Kehadiran informasi ini menunjukkan bahwa portal tidak dirancang semata sebagai halaman laporan keuangan kaku, melainkan sebagai pusat informasi masjid secara umum yang kebetulan salah satu fungsi utamanya adalah transparansi dana. Dari sisi pengguna, jamaah cukup membuka alamat portal tanpa perlu mengunduh aplikasi atau membuat akun apa pun, sehingga hambatan akses menjadi seminimal mungkin ,sebuah pertimbangan penting mengingat variasi kemampuan teknologi jamaah yang sangat beragam.

Tombol Masuk Admin pada pojok kanan atas mengarahkan pengurus menuju halaman *login*, sedangkan tombol Unduh Laporan memberi akses cepat bagi siapa pun yang ingin mengunduh ringkasan keuangan tanpa harus *login* sebagai admin terlebih dahulu, memperkuat prinsip akuntabilitas publik yang menjadi salah satu tujuan penelitian ini.

2. Halaman Login



Gambar 2. Halaman *Login* Admin

Halaman *login* menyajikan formulir sederhana yang hanya berisi dua kolom input, yaitu *Username* dan *Password*, ditambah satu *checkbox* Tampilkan *Password* yang memungkinkan pengurus memeriksa kembali ketikannya sebelum menekan tombol Masuk ke *Dashboard*. Kesederhanaan ini disengaja: karena pengguna sistem adalah pengurus masjid yang variasi kemampuan teknologinya berbeda-beda, maka halaman *login* tidak diberi elemen tambahan yang berpotensi membingungkan, sejalan dengan prinsip desain minim distraksi.

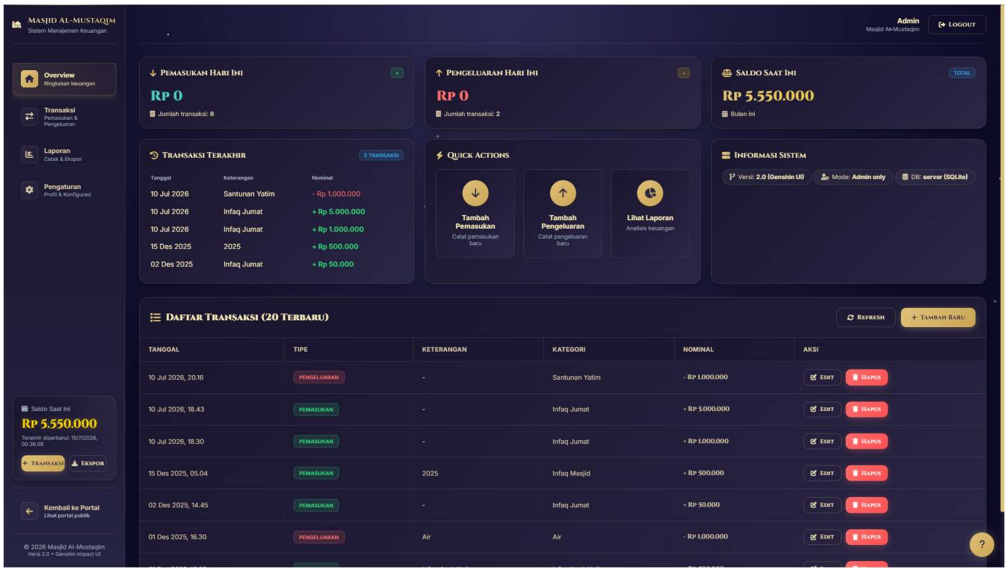
Secara teknis, halaman ini dilayani oleh *route* /login yang menerima dua metode *HTTP*, *GET* untuk menampilkan *form* dan *POST* untuk memproses percobaan *login*. *Form* itu sendiri dibangun menggunakan kelas *LoginForm* pada *forms.py*, yang

mewajibkan kedua *field* diisi melalui validator *DataRequired* sekaligus secara otomatis menyertakan token *CSRF* berkat integrasi *Flask-WTF*, sehingga formulir terlindung dari serangan lintas situs.

Ketika *form* disubmit dan lolos validasi, *backend* mencari data *user* berdasarkan username menggunakan *User.query.filter_by(username=...).first(),* kemudian memanggil method *check_password()* pada objek yang ditemukan. Method ini pada gilirannya memanggil *check_password_hash()* dari *Werkzeug* untuk membandingkan *password* yang dimasukkan dengan *hash* yang tersimpan di kolom *password_hash*, tanpa pernah menyimpan atau membandingkan *password* dalam bentuk teks asli. Jika kredensial valid, sistem menuliskan *user_id* dan username ke dalam *session Flask*, kemudian mengarahkan pengguna ke halaman *dashboard* sekaligus *menampilkan* pesan flash "*Login* berhasil". Jika tidak valid, pesan "*Username* atau *password* salah" ditampilkan tanpa memberi petunjuk yang membocorkan sisi mana yang keliru, sebagai bentuk praktik keamanan dasar.

Session yang terbentuk inilah yang kemudian diperiksa oleh *decorator login_required* pada setiap *route* yang bersifat administratif. Selama *session* belum dihapus (baik melalui *logout* maupun kedaluwarsa), *pengguna* dapat berpindah antar halaman admin tanpa perlu *login* ulang. Bagi bendahara, hal ini berarti mereka cukup *login* satu kali di awal sesi kerja, misalnya setelah shalat Jumat ketika infak mulai dihitung, dan dapat langsung mencatat beberapa transaksi berturut-turut tanpa hambatan otentikasi berulang.

3. Halaman Dashboard



Gambar 3. Halaman *Dashboard Overview*

Halaman *dashboard* merupakan pusat kendali bagi pengurus setelah berhasil *login*, dilayani oleh *route /dashboard* yang diproteksi *login_required* dan *me-render dashboard.html*. Sesuai komentar pada kode sumbernya, halaman ini hanya menyediakan kerangka tampilan, sedangkan seluruh data diisi secara dinamis melalui pemanggilan *API* dari sisi client, sehingga *dashboard* selalu menampilkan angka terbaru setiap kali dibuka atau *di-refresh* tanpa perlu logika render ulang di sisi server.

Pada bagian atas *dashboard* terdapat tiga kartu utama: Pemasukan Hari Ini, Pengeluaran Hari Ini, dan Saldo Saat Ini. Kartu saldo bersifat total kumulatif (bertanda *TOTAL*), berbeda dengan dua kartu lainnya yang menghitung transaksi hari berjalan saja,

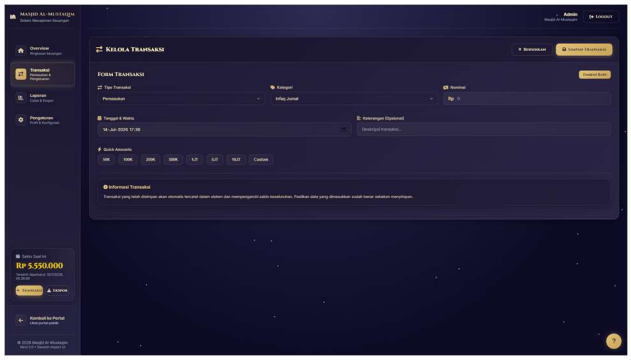
memberi bendahara gambaran cepat mengenai aktivitas harian sekaligus kondisi kas keseluruhan dalam satu pandangan. Di bawahnya terdapat panel Transaksi Terakhir yang menampilkan lima transaksi paling baru lengkap dengan tanggal, keterangan, dan nominal yang diwarnai berbeda (hijau untuk pemasukan, merah untuk pengeluaran), memudahkan bendahara mengenali jenis transaksi tanpa perlu membaca angka secara detail.

Panel *Quick Actions* menyediakan tiga pintasan, yaitu Tambah Pemasukan, Tambah Pengeluaran, dan Lihat Laporan, yang langsung mengarahkan pengurus ke form transaksi dengan tipe yang telah terisi otomatis, memangkas langkah yang harus dilakukan bendahara dibandingkan harus memilih tipe secara manual setiap saat. Di sampingnya, panel Informasi Sistem menampilkan versi aplikasi, mode akses (*Admin only*), dan jenis database yang digunakan (server *SQLite*), memberi konteks teknis singkat bagi pengurus yang ingin memahami *platform* yang mereka gunakan.

Bagian paling substansial dari *dashboard* adalah tabel Daftar Transaksi (20 Terbaru), yang menyajikan seluruh kolom penting: tanggal, tipe (dengan *badge* berwarna), keterangan, kategori, nominal bertanda, serta dua tombol aksi, *Edit* dan Hapus. Tabel ini diisi melalui pemanggilan *endpoint /api/transactions*, sedangkan tombol *Edit* mengarahkan pengurus ke *route /transaction/edit/<id>* yang menyiapkan form *TransactionForm* terisi data lama (*form = TransactionForm(obj=tx)*), dan tombol Hapus memanggil *endpoint DELETE /api/transaction/<id>* yang menjalankan *db.session.delete(tx)* sebelum melakukan *commit*.

Dari perspektif bendahara, *dashboard* ini menggantikan kebiasaan membuka lembar demi lembar buku kas untuk mengetahui saldo terkini atau mencari transaksi tertentu. Segala sesuatu yang sebelumnya membutuhkan waktu penelusuran manual kini tersaji dalam satu halaman yang termuat dalam hitungan detik, sekaligus mengurangi risiko human error yang lazim terjadi pada pencatatan manual.

4. Halaman Manajemen Pemasukan



Gambar 4. *Form* Kelola Transaksi (Tipe Pemasukan)

Untuk mencatat transaksi baru, baik pemasukan maupun pengeluaran, sistem menyediakan satu antarmuka terpadu bernama Kelola Transaksi, dan subbab ini secara khusus membahas alur ketika tipe transaksi yang dipilih adalah Pemasukan. Form ini terdiri atas lima elemen utama: *dropdown* Tipe Transaksi, *dropdown* Kategori (misalnya Infaq Jumat, Infaq Masjid, Donasi), input Nominal berformat Rupiah, input Tanggal & Waktu, serta input Keterangan yang sifatnya opsional.

Salah satu fitur yang memudahkan bendahara adalah baris *Quick Amounts*, yaitu sederet tombol nominal siap pakai (50K, 100K, 250K, 500K, 1JT, 5JT, 10JT, serta *Custom* untuk nilai bebas). Fitur ini lahir dari proses uji coba bersama pengurus, di mana

salah satu masukan pengguna adalah "kolom input nilai rupiah perlu pemformatan otomatis". Dengan tombol nominal cepat, bendahara yang mencatat infak Jumat dalam pecahan umum tidak perlu mengetik angka nol berulang kali, sehingga mempercepat proses input sekaligus mengurangi risiko salah ketik jumlah digit.

Ketika form ini disubmit dengan tipe Pemasukan, data dikirim ke *endpoint POST /api/transaction* dalam format *JSON*. Pada sisi *backend*, fungsi *api_add_transaction()* membaca *field* tipe, kategori, keterangan, dan jumlah dari *body* permintaan, kemudian mengonversi nilai jumlah menjadi tipe *integer*. Karena tipe transaksinya Pemasukan, kondisi percabangan penyesuaian tanda tidak dieksekusi, sehingga nilai nominal tersimpan apa adanya sebagai angka positif, konsisten dengan konvensi. *Field* Tanggal & Waktu memungkinkan bendahara mencatat transaksi dengan waktu aktual kejadian, bukan sekadar waktu saat data dimasukkan ke sistem, sehingga pencatatan tetap akurat meski *input* dilakukan belakangan, misalnya mencatat infak Jumat pada Senin berikutnya karena kesibukan. Setelah tersimpan, panel Informasi Transaksi mengingatkan bahwa data yang disimpan akan langsung memengaruhi saldo keseluruhan, mendorong bendahara memeriksa kembali data sebelum menekan tombol Simpan Transaksi, sebuah bentuk penerapan prinsip umpan balik UX, di mana sistem memberi peringatan yang jelas alih-alih membiarkan kesalahan tersimpan tanpa konfirmasi.

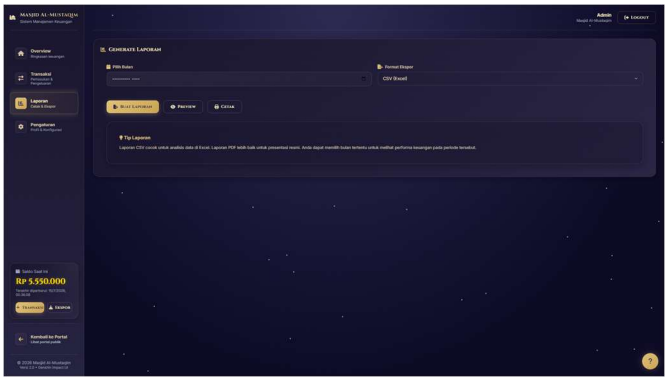
5. Halaman Manajemen Pengeluaran

Halaman yang digunakan untuk mencatat pengeluaran secara fisik adalah antarmuka, karena sistem dirancang menggunakan satu komponen form yang *reusable* untuk kedua tipe transaksi, sejalan dengan prinsip *modular* dan *reusable components* pada *RAD*. Perbedaan hanya terletak pada pilihan dropdown Tipe Transaksi yang diubah menjadi Pengeluaran, sehingga kategori yang tersedia pun menyesuaikan, misalnya Air, Listrik, Santunan Yatim, atau Honor Penceramah, sebagaimana terlihat pada data yang tercatat di tabel *dashboard*.

Perbedaan paling penting justru terjadi di balik layar, yaitu pada logika penyesuaian tanda nilai. Ketika tipe yang dipilih adalah Pengeluaran, baik *route* berbasis form (*edit_transaction*) maupun *endpoint API* (*api_add_transaction* dan *api_update_transaction*) akan mengeksekusi baris kondisi *if tipe == 'Pengeluaran' and jumlah > 0: jumlah = -abs(jumlah)*, yang secara otomatis membalik nilai nominal menjadi negatif sebelum disimpan ke *database*. Bendahara tetap mengetik angka positif sebagaimana biasa (misalnya mengetik 1000000 untuk santunan yatim sebesar satu juta rupiah), namun sistem yang menerjemahkannya menjadi representasi akuntansi yang benar.

Keputusan menyembunyikan logika tanda minus dari pengguna ini memiliki manfaat besar bagi kemudahan penggunaan. Pada pencatatan manual menggunakan buku kas, kesalahan umum yang terjadi adalah kekeliruan menuliskan pengeluaran pada kolom pemasukan atau lupa mengurangkan nilai dari total saldo. Dengan sistem ini, satu-satunya hal yang perlu diperhatikan bendahara adalah memilih tipe transaksi dengan benar; perhitungan tanda dan pembaruan saldo sepenuhnya ditangani oleh aplikasi. Setelah tersimpan, transaksi pengeluaran akan langsung tercermin pada kartu Pengeluaran Hari Ini di *dashboard* maupun pada perhitungan saldo di portal publik, karena keduanya bersumber dari agregasi yang sama pada tabel *Transaction*. Konsistensi satu sumber data ini memastikan tidak ada perbedaan angka antara apa yang dilihat bendahara di *dashboard* dan apa yang dilihat jamaah di portal, memperkuat akuntabilitas pengelolaan dana masjid.

6. Halaman Laporan

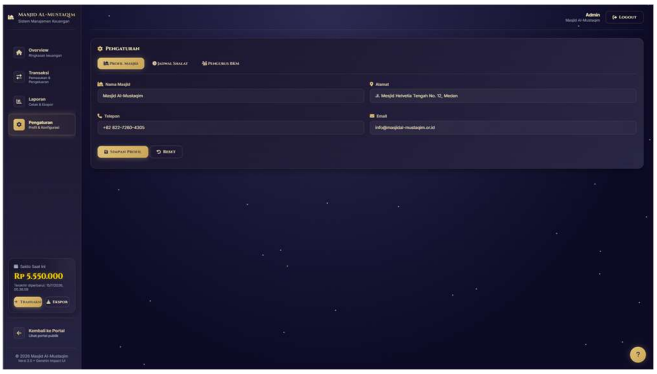


Gambar 5. Halaman *Generate* Laporan

Halaman Laporan menyediakan sarana bagi pengurus untuk menyusun rekap keuangan berdasarkan periode tertentu, sesuai dengan modul laporan dan ekspor yang direncanakan. Antarmuka ini terdiri atas input Pilih Bulan berupa *month-picker*, *dropdown* Format Ekspor (*CSV Excel* atau *PDF*), serta tiga tombol aksi: Buat Laporan, *Preview*, dan Cetak. Di bawahnya terdapat kotak Tip Laporan yang memberi saran kontekstual, misalnya format *CSV* lebih cocok untuk analisis lanjutan di *Excel* sedangkan format *PDF* lebih sesuai untuk keperluan presentasi resmi kepada jamaah atau pihak eksternal. Secara teknis, ketika pengguna memilih suatu bulan, permintaan data dikirim ke *endpoint /api/transactions* dengan parameter month berformat *YYYY-MM*. Pada sisi *backend*, fungsi *api_transactions()* memproses parameter tersebut dengan memecah nilai tahun dan bulan, kemudian membentuk rentang tanggal awal dan akhir bulan yang bersangkutan (termasuk penanganan khusus pergantian tahun ketika bulan yang dipilih adalah Desember), sebelum menjalankan *filter query Transaction.dt* antara rentang tersebut.

Pendekatan penyaringan berbasis rentang tanggal ini memungkinkan bendahara menyusun laporan bulanan tanpa perlu menandai atau memisahkan data secara manual seperti pada metode buku kas, yang mengharuskan penjumlahan ulang setiap kali laporan hendak disusun. Cukup dengan memilih bulan yang diinginkan, sistem akan mengambil seluruh transaksi pada rentang tersebut dan menyusunnya menjadi bentuk laporan yang siap diekspor. Ketersediaan dua pilihan format ekspor mencerminkan kebutuhan ganda pengguna: format *CSV* memudahkan bendahara yang ingin melakukan pengolahan lanjutan (misalnya membuat grafik tambahan di aplikasi *spreadsheet*), sedangkan format *PDF* lebih sesuai untuk diumumkan atau ditempel di papan informasi masjid maupun dibagikan melalui grup komunikasi jamaah. Fitur laporan inilah yang secara langsung menjawab tujuan penelitian kedua pada BAB I, yaitu menghasilkan fitur pelaporan keuangan otomatis yang akurat, transparan, dan mudah dipahami oleh pengurus maupun jamaah.

7. Halaman Pengaturan



Gambar 6. Halaman Pengaturan Profil Masjid

Halaman Pengaturan menyediakan tiga tab navigasi, yaitu Profil Masjid, Jadwal Shalat, dan Pengurus BKM, yang seluruhnya berfungsi menyimpan data pendukung yang ditampilkan pada berbagai bagian sistem, terutama portal publik dan kop laporan. Pada tab Profil Masjid yang aktif pada tangkapan layar, pengurus dapat mengubah Nama Masjid, Alamat, Telepon, dan Email, kemudian menyimpannya melalui tombol Simpan Profil atau mengembalikan ke nilai semula melalui tombol *Reset*.

Keberadaan halaman pengaturan ini merealisasikan modul pengaturan profil yang direncanakan, yaitu menyimpan data identitas masjid agar tidak perlu dituliskan ulang secara manual setiap kali laporan atau tampilan portal diperbarui. Sebagai contoh, alamat dan nomor telepon yang diisi pada halaman ini adalah data yang sama persis dengan yang tampil pada bagian kontak di portal publik, menunjukkan bahwa data pengaturan menjadi satu sumber kebenaran (*single source of truth*) yang dipakai ulang di berbagai bagian antarmuka. Pada tahap implementasi saat ini, data pengaturan profil dikelola pada level antarmuka dan konfigurasi awal aplikasi, sedangkan tab Jadwal Shalat dan Pengurus BKM disediakan sebagai kerangka pengembangan lanjutan yang dapat dihubungkan ke tabel basis data baru pada iterasi *RAD* berikutnya, sejalan dengan sifat *RAD* yang memang memungkinkan penambahan modul secara bertahap tanpa mengganggu fitur inti yang sudah berjalan (pencatatan transaksi, *dashboard*, dan laporan). Poin ini turut dibahas lebih lanjut sebagai saran pengembangan pada BAB V.

Dari sisi manfaat, halaman pengaturan memberi kendali penuh kepada pengurus untuk memperbarui informasi identitas masjid secara mandiri tanpa perlu meminta bantuan pengembang setiap kali terjadi perubahan alamat, nomor kontak, atau kepengurusan, sehingga sistem tetap relevan dan *up-to-date* dalam jangka panjang tanpa ketergantungan teknis yang berlebihan terhadap pihak ketiga.

Pengujian Sistem

1. Metode Pengujian

Pengujian sistem dilakukan menggunakan metode *black box testing*, yaitu pengujian yang berfokus pada hasil keluaran (*output*) sistem terhadap sejumlah masukan (*input*) tertentu, tanpa perlu menelusuri detail logika internal kode

No	Skenario Pengujian	Data / Aksi	Hasil yang Diharapkan	Hasil Aktual	Status
1	*Login* dengan kredensial benar	*username*: admin, *password*: 1234	Sistem masuk ke halaman *dashboard*	Berhasil masuk ke *dashboard*, muncul pesan "*Login* berhasil"	Berhasil
2	*Login* dengan kredensial salah	*username: admin, password:* salah	Sistem menolak dan menampilkan pesan *error*	Muncul pesan "*Username* atau *password* salah", tetap di halaman login	Berhasil
3	Akses /*dashboard* tanpa *login*	Membuka URL *dashboard* langsung	Sistem mengarahkan ke halaman *login*	*Redirect* ke *login* dengan pesan "Silakan *login* terlebih dahulu"	Berhasil
4	Tambah transaksi Pemasukan	Tipe: Pemasukan, Nominal: 500000	Data tersimpan dengan nilai +500000	Data tersimpan positif, saldo bertambah 500000	Berhasil
5	Tambah transaksi Pengeluaran	Tipe: Pengeluaran, Nominal: 200000	Data tersimpan otomatis menjadi -200000	Data tersimpan negatif, saldo berkurang 200000	Berhasil
6	*Input* nominal bukan angka	Nominal: "abc"	Sistem menolak dengan pesan *error*	Muncul pesan "Nominal harus angka", data tidak tersimpan	Berhasil
7	Edit transaksi	Ubah kategori & nominal transaksi lama	Data pada baris tersebut diperbarui	Data berubah sesuai *input* baru, saldo ikut menyesuaikan	Berhasil
8	Hapus transaksi	Klik tombol Hapus pada salah satu baris	Data terhapus dan tidak tampil lagi di tabel	Baris hilang dari daftar, saldo diperbarui otomatis	Berhasil
9	Filter laporan per bulan	Pilih bulan tertentu pada halaman Laporan	Hanya transaksi bulan tersebut yang ditampilkan	Data sesuai rentang bulan yang dipilih	Berhasil
10	Ringkasan portal publik	Membuka portal tanpa *login*	Kartu pemasukan, pengeluaran, saldo tampil sesuai data terbaru	Angka sesuai dengan data pada *dashboard* admin	Berhasil

program. Metode ini dipilih karena tujuan utama pengujian adalah memastikan setiap fungsi yang dibutuhkan pengurus masjid, yaitu *login*, pencatatan transaksi, pengeditan, penghapusan, dan penyusunan laporan, bekerja sesuai harapan dari sudut pandang pengguna akhir, konsisten dengan pendekatan pengujian pada tahap *testing RAD*. Pengujian dilakukan secara manual oleh peneliti bersama satu orang bendahara masjid sebagai perwakilan pengguna, dengan skenario yang mencakup jalur normal (data valid) maupun jalur pengecualian (data tidak valid atau akses tanpa izin), sehingga dapat

diketahui apakah sistem menangani kesalahan pengguna dengan baik, bukan hanya berhasil pada kondisi ideal.

2. Analisis Pengujian

Berdasarkan sepuluh skenario pengujian yang dilaksanakan, seluruh fungsi inti sistem, yaitu autentikasi, proteksi akses halaman *admin*, pencatatan dan penyesuaian tanda nominal otomatis, validasi *input* non-angka, operasi ubah dan hapus data, penyaringan laporan per bulan, hingga konsistensi data antara *dashboard* dan portal publik, berjalan sesuai dengan hasil yang diharapkan tanpa ditemukan kegagalan fungsional. Pengujian juga mengonfirmasi bahwa perbaikan-perbaikan yang dilakukan pada tahap *cutover RAD*, seperti kebutuhan pemformatan nominal otomatis dan kemudahan menemukan transaksi lama, telah terealisasi secara nyata melalui fitur *quick amount* pada *form* transaksi dan tabel Daftar Transaksi yang menampilkan dua puluh data terbaru secara terurut. Dengan kata lain, umpan balik yang diperoleh selama iterasi pengembangan bukan hanya dicatat sebagai catatan evaluasi, tetapi benar-benar ditindaklanjuti menjadi fitur yang teruji.

4. KESIMPULAN DAN SARAN

Berdasarkan hasil perancangan, implementasi, dan pengujian, Sistem Manajemen Keuangan Masjid Al-Mustaqim Helvetia Tengah berbasis web yang dikembangkan menggunakan metode Rapid Application Development (RAD) berhasil memenuhi tujuan penelitian. Sistem mampu memfasilitasi pencatatan transaksi pemasukan dan pengeluaran secara terstruktur, menyajikan laporan keuangan yang cepat, akurat, dan transparan, serta menyediakan portal publik yang dapat diakses jamaah secara real-time. Penerapan metode RAD terbukti mempercepat proses pengembangan melalui tahapan yang iteratif dan berorientasi pada kebutuhan pengguna. Selain itu, hasil pengujian black box menunjukkan seluruh fungsi utama sistem berjalan sesuai dengan yang diharapkan, sehingga sistem layak digunakan sebagai pengganti pencatatan keuangan manual di Masjid Al-Mustaqim Helvetia Tengah.

Sistem yang dikembangkan masih dapat disempurnakan melalui penambahan fitur ekspor laporan ke format PDF dan CSV, penyimpanan profil masjid pada basis data, pengelolaan pengguna dengan hak akses berbeda dan audit trail, visualisasi grafik keuangan, serta peningkatan keamanan melalui HTTPS, backup otomatis, dan pembatasan percobaan login. Selain itu, perlu dilakukan pengujian dengan volume data yang lebih besar untuk memastikan stabilitas sistem, serta pengembangan modul tambahan seperti manajemen jamaah, inventaris masjid, dan optimalisasi tampilan responsif agar penggunaan melalui perangkat seluler menjadi lebih nyaman.

DAFTAR REFERENSI

Hanum, R., Fachrudin, D. H., & Rohyana, C. (2026). Design and Development of a Web-Based Information System for MSME Digital Promotion Using the Rapid Application Development (RAD) Method. *MALCOM: Indonesian Journal of Machine Learning and Computer Science*, *6*(January), 314–323.

Hidayat, N., & Hati, K. (2021). Penerapan Metode Rapid Application Development (RAD) dalam Rancang Bangun Sistem Informasi Rapor Online (SIRALINE). *Jurnal Sistem Informasi*. https://doi.org/10.51998/jsi.v10i1.352

Pradhana, D. R. A., Saputro, D. K., & Maulindar, J. (2022). Analisa dan Perancangan

Sistem Informasi dan Aplikasi Manajemen Keuangan dan Infaq Masjid Berbasis Web. *Prosiding Seminar Nasional Teknologi Informasi Dan Bisnis*.

Purnasari, M., Hartiwi, Y., & Nurhayati. (2022). Perancangan Sistem Informasi Pengelolaan Dana Masjid Berbasis Web Menggunakan Unified Modeling Language (UML). *Resolusi: Rekayasa Teknik Informatika Dan Informasi*.

Wahyudi, E., Hanif, A., Adianto, H., & Baidawi, T. (2025). Implementasi Sistem Informasi Manajemen Keuangan Masjid Menggunakan Metode Prototype Base on Cloud Computing. *Reputasi: Jurnal Rekayasa Perangkat Lunak Dan Sistem Informasi*.

Walingkas, H. L., & Saian, P. O. N. (2023). Penerapan Framework Flask pada Pembangunan Sistem Informasi Pemasok Barang. *Jurnal JTIK (Jurnal Teknologi Informasi Dan Komunikasi)*.