Penerapan *Continuous Integration* Dalam Pengembangan Aplikasi Manajemen Rumah Kost Menggunakan Metode *Rapid Application Development* (Studi Kasus: Rumah Kost Rahmatika)

1st Ginza Maulana Putra
Sistem Informasi
Telkom University
Surabaya, Indonesia
ginzamaulanaputra@student.telkomuniversity.ac.id

2nd Mochamad Nizar Palefi Ma'ady
Sistem Informasi
Telkom University
Surabaya, Indonesia
mnizarpm@telkomuniversity.ac.id

3rd Purnama Anaking
Sistem Informasi
Telkom University
Surabaya, Indonesia
purnama@telkomuniversity.ac.id

***Abstrak* — Tingginya angka urbanisasi di Surabaya dan Sidoarjo menyebabkan peningkatan kebutuhan terhadap hunian sementara seperti rumah kost. Salah satunya adalah rumah kost Rahmatika yang tersebar di tiga lokasi berbeda, yakni Gunung Anyar, Rungkut Menanggal, dan Berbek. Pengelolaan manual yang dilakukan oleh pemilik kost menghadapi berbagai kendala, seperti pencatatan pembayaran secara konvensional, keterlambatan pembayaran sewa, serta kesulitan dalam pemantauan langsung karena jarak antar lokasi. Berdasarkan permasalahan tersebut, penelitian ini bertujuan untuk membangun sistem informasi manajemen kost berbasis website dengan menggunakan framework *Laravel* dan pendekatan *Rapid Application Development* (RAD). Pengembangan dilakukan secara iteratif dan melibatkan pengguna pada setiap tahap perancangan sistem. Sistem ini mendukung fitur utama seperti manajemen data penghuni, pelaporan keluhan, dan pembayaran sewa menggunakan *midtrans*. Pengujian sistem dilakukan melalui *blackbox testing* serta *User Acceptance Testing* (UAT) dengan melibatkan 23 responden, yang menghasilkan nilai rata-rata kepuasan sebesar 89%, menunjukkan sistem telah sesuai dengan kebutuhan pengguna. Selain itu, sistem juga diintegrasikan dengan *continuous integration* (CI) menggunakan *jenkins* untuk mempercepat otomatisasi pengujian dan pembaruan sistem. Hasil pengujian menunjukkan bahwa CI dapat membantu menjaga stabilitas dan kualitas integrasi fitur dalam sistem secara berkelanjutan. Dengan demikian, sistem ini mampu meningkatkan efisiensi pengelolaan kost secara digital dan mendukung operasional pemilik kost tanpa harus melakukan kunjungan langsung ke lokasi.**

Kata kunci* — Sistem Informasi Kost, *Rapid Application Development*, *User Acceptance Testing*, *Continuous Integration*, *Web Application

I. PENDAHULUAN

Urbanisasi pesat di Kota Surabaya telah meningkatkan permintaan hunian, terutama rumah kost, sebagai akibat dari migrasi penduduk yang mencari peluang ekonomi dan fasilitas lebih baik [1]. Surabaya, sebagai pusat pendidikan dan perekonomian, menarik banyak mahasiswa dan pekerja dari berbagai daerah. Berdasarkan data Badan Pusat Statistik Provinsi Jawa Timur, terdapat sekitar 270 ribu mahasiswa dan 1,5 juta pekerja di Surabaya, mendorong tingginya kebutuhan akan hunian sementara yang terjangkau dan strategis[2]. Selain itu, sekitar 35,37 persen rumah tangga di Surabaya menempati bangunan bukan milik sendiri, menunjukkan tingginya kebutuhan akan rumah kost [3]. Pertumbuhan pesat ini berbanding lurus dengan terbatasnya lahan dan harga properti yang tinggi, membuat rumah kost menjadi pilihan utama, khususnya bagi mahasiswa dan pekerja perantauan. Rumah Kost Rahmatika, yang terdiri dari tiga lokasi di Gunung Anyar, Rungkut Menanggal, dan Berbek, menghadapi tantangan pengelolaan operasional, terutama dengan meningkatnya jumlah penghuni dan kompleksitas kebutuhan mereka.

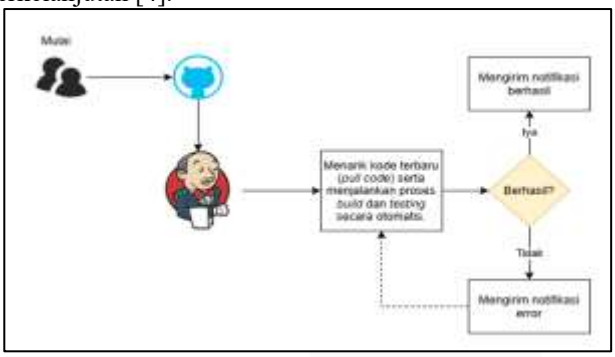
Sistem manajemen kost yang masih menggunakan metode manual saat ini memunculkan permasalahan seperti kesalahan pencatatan, keterlambatan informasi, serta kesulitan dalam pembayaran dan pengelolaan data penghuni. Hal ini semakin sulit mengingat jarak antar lokasi rumah kost yang berjauhan. Untuk mengatasi hal tersebut, diperlukan digitalisasi sistem manajemen kost yang dapat mempermudah pencatatan dan pengelolaan data secara terpusat. Dalam pengembangannya, digunakan metode *rapid application development* (RAD) dan *Continuous Integration* (CI) untuk menciptakan sistem berbasis website yang fleksibel dan dapat disesuaikan dengan kebutuhan pengguna secara berkelanjutan. RAD memungkinkan pengembangan sistem secara cepat dengan prototipe yang terus diuji dan disesuaikan berdasarkan umpan balik pengguna, sementara CI memfasilitasi pembaruan sistem secara otomatis, menjaga kualitas kode, dan mempercepat pengembangan.

II. KAJIAN TEORI

A. *Continuous Integration* (CI)

GAMBAR 1
(CONTINUOUS INTEGRATION (CI))

Continuous integration (CI) merupakan salah satu metode dalam praktik DevOps (*Development and Operations*) yang berfungsi untuk mengotomatisasi proses integrasi kode. Metode ini memungkinkan anggota tim pengembang untuk menggabungkan perubahan kode secara rutin ke dalam repositori bersama. CI juga mendukung otomatisasi proses *build*, *testing*, dan *reporting* selama pengembangan perangkat lunak. Dengan adanya CI, proses integrasi kode menjadi lebih terstruktur dan terkelola dengan baik. Secara umum, CI bertujuan untuk menjaga kualitas perangkat lunak melalui pengujian dan integrasi yang dilakukan secara berkelanjutan [4].



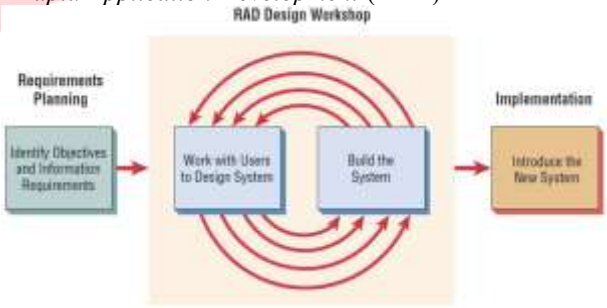
GAMBAR 2
(ARSITEKTUR *CONTINUOUS INTEGRATION* (CI))

Gambar 2 merupakan alur arsitektur *Continuous Integration* (CI). Pada tahap awal, pengembang akan melakukan *push* kode ke dalam *GitHub* melalui repositori. Selanjutnya, *GitHub* akan melakukan sinkronisasi dengan Jenkins. Agar proses integrasi antara *GitHub* dan *Jenkins* dapat berjalan, diperlukan pembuatan *token* autentikasi dari *GitHub*. Setelah itu, dilakukan konfigurasi pada *Jenkins* melalui *project pipeline* yang dikustomisasi sesuai dengan kebutuhan proyek. Tahap akhir meliputi proses *build* dan *testing*, di mana *Jenkins* akan melakukan *build* serta pengujian kode. Apabila proses tersebut berhasil, *Jenkins* akan mengirimkan notifikasi keberhasilan. Namun, jika terjadi kegagalan, proses *build* dan *testing* akan diulang hingga berhasil. Proses ini berjalan secara otomatis setiap kali terjadi perubahan pada repositori, sehingga integrasi kode menjadi lebih cepat dan efisien. Notifikasi keberhasilan maupun error akan dikirimkan ke tim pengembang untuk memastikan tindak lanjut dapat segera dilakukan. Dengan mekanisme ini, potensi bug dapat terdeteksi lebih awal sebelum masuk ke tahap produksi. Arsitektur Continuous Integration seperti ini sangat membantu dalam menjaga kualitas perangkat lunak secara konsisten selama siklus pengembangan berlangsung.

B. *Jenkins*

Jenkins merupakan *open source automation server* yang berfungsi untuk mengotomatisasi tugas-tugas dalam proses *continuous integration* dan *continuous delivery* pada pengembangan sistem [5]. *Jenkins* banyak digunakan oleh para developer karena memiliki beragam fitur yang mendukung dan kompatibel dengan berbagai repositori populer, seperti *GitHub*, *GitLab*, *Bitbucket*, dan lainnya. Selain itu, *jenkins* menyediakan banyak plugin yang dapat disesuaikan dengan kebutuhan pengembangan sistem, sehingga mempermudah proses otomatisasi. Penggunaan *jenkins* juga memungkinkan pengembang untuk melakukan integrasi dan pengujian kode secara berkala guna menjaga kualitas perangkat lunak. Hal ini membuat jenkins menjadi pilihan favorit bagi banyak pengembang karena dapat dengan mudah diintegrasikan ke dalam alur kerja mereka tanpa hambatan yang signifikan [6].

C. *Rapid Application Development* (RAD)



GAMBAR 3
(*RAPID APPLICATION DEVELOPMENT* (RAD))

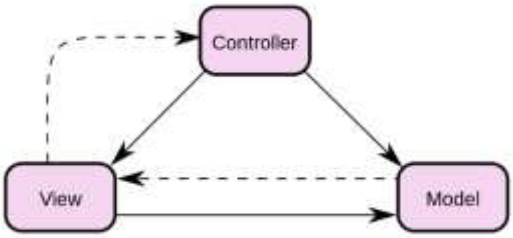
Rapid Application Development (RAD) adalah metode pengembangan perangkat lunak yang menekankan percepatan siklus pengembangan melalui pembuatan prototipe dan umpan balik berkelanjutan dari pengguna. Metode ini dapat mempersingkat waktu pengembangan dari rata-rata 180 hari menjadi hanya 30–90 hari [7][8]. RAD cocok untuk proyek yang memerlukan kecepatan serta fleksibilitas terhadap perubahan kebutuhan selama pengembangan. Tahapan Rapid Application Development (RAD) terdiri dari tiga fase utama. Fase Perencanaan Kebutuhan (*Requirement Planning*) dilakukan untuk mengidentifikasi dan menganalisis kebutuhan perangkat lunak, termasuk menentukan peran aktor dan fitur yang diperlukan. Selanjutnya, pada Fase Desain Workshop (*Workshop Design* RAD), dilakukan kolaborasi dengan pengguna untuk merancang sistem sesuai kebutuhan, kemudian dilanjutkan dengan pembangunan sistem berdasarkan desain yang telah disepakati. Terakhir, Fase Implementasi (*Implementation*) mencakup pengembangan sistem sesuai persetujuan pengguna, pengujian, serta evaluasi hasil oleh pengguna.

D. *Unified Model Language* (UML)

Unified Modeling Language (UML) adalah alat pemodelan visual yang berfungsi sebagai blueprint dalam pengembangan perangkat lunak. UML memberikan representasi grafis yang memudahkan pemahaman,

perancangan sistem, serta komunikasi antar tim pengembang. Penggunaan UML membantu mengurangi kesalahan desain dan meningkatkan efisiensi pengembangan melalui panduan yang jelas dan terstruktur [9].

E. *Framework Laravel*



GAMBAR 3
(ARSITEKTUR *MODEL-VIEW-CONTROLLER LARAVEL)*

Laravel adalah *framework* web open-source berbasis PHP yang dikembangkan oleh Taylor Otwell dengan menerapkan arsitektur *Model-View-Controller* (MVC). Dalam MVC, *model* mengelola logika data, *view* menampilkan antarmuka pengguna, dan *controller* menghubungkan keduanya untuk mengatur alur data dan proses aplikasi [10]. Laravel menyederhanakan proses pengembangan, menekan biaya pemeliharaan, serta menyediakan sintaks rapi dan efisien [11]. Dilengkapi fitur seperti routing, middleware, autentikasi, serta dukungan integrasi layanan pihak ketiga, Laravel mendukung pengembangan aplikasi berskala besar secara terstruktur.

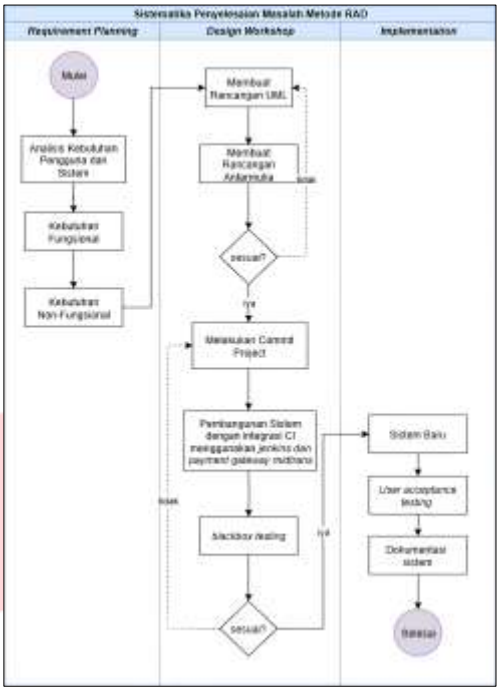
F. *Blackbox Testing*

Blackbox testing adalah metode pengujian perangkat lunak yang mengevaluasi fungsionalitas sistem tanpa memperhatikan kode program. Pengujian ini berfokus pada hasil berdasarkan input yang diberikan untuk memastikan perangkat lunak sesuai dengan spesifikasi dan kebutuhan [12]. Teknik ini mencakup pengujian fungsionalitas, validasi input, antarmuka pengguna, serta membantu mengidentifikasi kesalahan logika bisnis, proses input-output, dan integrasi sistem eksternal.

G. *User Acceptance Testing* (UAT)

User acceptance testing (UAT) merupakan salah satu jenis pengujian perangkat lunak yang bertujuan untuk memastikan bahwa sistem yang dikembangkan telah sesuai dengan kebutuhan dan harapan pengguna akhir. Pengujian ini dilakukan pada tahap akhir sebelum sistem resmi diimplementasikan, guna memastikan bahwa seluruh fungsi berjalan sebagaimana mestinya dalam konteks penggunaan nyata. UAT berfokus pada tiga variabel utama dalam mengevaluasi kelayakan sistem, yaitu: (1) fungsionalitas sistem, (2) efisiensi sistem, dan (3) pengalaman dan tampilan antarmuka pengguna [13].

III. METODE



GAMBAR 4
(SISTEMATIKA METODE PENYELESAIAN)

A. *Requirement Planning*

Tahap *requirement planning* merupakan langkah yang digunakan untuk merencanakan seluruh kebutuhan yang diperlukan oleh rumah kost Rahmatika di Surabaya dan Sidoarjo. Proses ini dilakukan melalui hasil wawancara dengan pemilik kost ibu Novelia Karlinda Tristiyanti, untuk menganalisis proses yang ada kemudian menyiapkan solusi dan tujuan sistem, serta kebutuhan informasi (*fungsional* dan *non-fungsional*) yang dibutuhkan oleh rumah kost Rahmatika di Surabaya dan Sidoarjo.

B. *Design Workshop*

Pada tahap ini, peneliti merancang desain sistem yang mencakup *use case diagram*, *activity diagram*, dan antarmuka menggunakan *wireframe* yang diperoleh melalui proses kolaborasi dengan pemilik kost. Proses kolaborasi ini bertujuan untuk memahami secara mendalam permasalahan yang terdapat pada rumah kost Rahmatika di Surabaya dan Sidoarjo. Selanjutnya, melakukan *commit project* sebagai langkah awal dalam proses pengkodean sistem. Proses ini dilanjutkan dengan kolaborasi dengan pengguna untuk membangun sistem manajemen rumah kost berbasis *website* dengan menggunakan *framework laravel* sebagai *tools build*, kemudian *jenkins* sebagai *tools continuous integration* untuk mengintegrasikan berbagai fitur pada *website*. Selain itu, *midtrans payment gateway* dimanfaatkan sebagai solusi sistem pembayaran. Kemudian dilakukan pengujian *blackbox testing* untuk menunjukkan apakah sistem sesuai dengan yang telah ditetapkan.

C. *Implementation*

Setelah melakukan *design workshop*, selanjutnya melakukan *implementation* berupa pengenalan sistem baru kepada pemilik kost yang sudah siap untuk di implementasikan. Pengujian *user acceptance testing* (UAT)

dilakukan untuk memvalidasi apakah sistem telah sesuai dengan tujuan dan kebutuhan yang telah ditetapkan. Selain itu, dokumentasi sistem akan disediakan

IV. HASIL DAN PEMBAHASAN

A. *Requirement Planning*

1. Kebutuhan Pengguna

Kebutuhan pengguna digunakan untuk mendefinisikan preferensi dan ekspektasi yang diinginkan oleh pengguna, sehingga sistem yang dikembangkan dapat sesuai dengan persyaratan dan ketentuan yang telah ditetapkan. Kebutuhan pengguna ini diperoleh melalui wawancara mendalam dengan pemilik rumah kost sebagai informan utama, serta didukung oleh masukan dari beberapa penghuni kost. Gambaran kebutuhan pengguna disajikan dalam bentuk tabel 4.1 sebagai berikut:

TABEL 1
(KEBUTUHAN PENGGUNA)

ID	Aktor	Deskripsi Kebutuhan
U1	Penghuni	Ingin dapat melakukan registrasi untuk mendapatkan akun sistem.
U2		Ingin dapat melakukan login ke dalam sistem untuk mengakses sistem.
U3		Ingin melihat tampilan dashboard berisi informasi terkait kost.
U4		Ingin dapat melakukan pembayaran sewa kost secara online.
U5		Ingin melihat riwayat pembayaran kost sebelumnya.
U6		Ingin dapat mengirimkan laporan keluhan.
A1	Admin	Ingin melihat tampilan dashboard yang berisi ringkasan data penghuni dan transaksi.
A2		Ingin mengakses data penghuni kost secara lengkap.
A3		Ingin melihat daftar laporan keluhan yang dikirimkan oleh penghuni kost.
A4		Ingin melihat riwayat transaksi pembayaran dari masing-masing penghuni.

2. Kebutuhan Fungsional

Kebutuhan fungsional pada sistem ini dibagi menjadi dua bagian, *user* dan *admin*. kebutuhan fungsional merupakan fitur-fitur atau fungsi utama yang harus dimiliki sistem agar dapat berjalan sesuai tujuan yang telah ditetapkan. Tabel 2 merupakan kebutuhan fungsional Penghuni dan Tabel 3 kebutuhan fungsinal Admin.

TABEL 1
(KEBUTUHAN FUNGSIONAL "PENGHUNI")

ID	Aktor	Kebutuhan fungsional
U1	Penghuni	Sistem harus menyediakan fitur registrasi akun, agar pengguna baru dapat melakukan pendaftaran untuk mendapatkan akses ke dalam sistem.
U2		Sistem harus menyediakan fitur login yang memungkinkan pengguna untuk masuk ke dalam sistem menggunakan kredensial yang valid.
U3		Sistem harus dapat menampilkan dashboard user, yang memuat informasi status sewa, tagihan dan pembayaran, serta daftar laporan yang telah dibuat oleh user.
U4		Sistem harus dapat melakukan pembayaran online yang terintegrasi dengan pihak ketiga, yaitu m*idtrans*, untuk memfasilitasi proses pembayaran sewa kost.
U5		Sistem harus menyediakan informasi riwayat pembayaran, yang mencakup kode transaksi, tanggal pembayaran, total pembayaran, dan status pembayaran.
U6		Sistem harus menyediakan fitur pengiriman laporan keluhan, sehingga pengguna dapat menyampaikan permasalahan terkait fasilitas atau layanan kost kepada pemilik.

TABEL 2
(KEBUTUHAN FUNGSIONAL "ADMIN")

ID	Aktor	Kebutuhan fungsional
A1	Admin	Sistem harus mampu menampilkan dashboard khusus admin yang menyajikan ringkasan informasi jumlah penghuni kost dan data pendapatan, yang divisualisasikan dalam bentuk grafik.
A2		Sistem harus mampu menampilkan data penghuni kost dalam bentuk tabel, serta menyediakan fitur untuk melakukan operasi CRUD (Create, Read, Update, Delete) terhadap data penghuni.
A3		Sistem harus mampu menampilkan daftar laporan keluhan dari pengguna, serta menyediakan fitur bagi admin untuk menambahkan laporan baru, menghapus laporan yang tidak valid, dan mengubah status laporan menjadi “Selesai” setelah ditindaklanjuti.
A4		Sistem harus mampu menampilkan riwayat transaksi pembayaran dari seluruh penghuni kost, yang mencakup informasi berupa ID transaksi, kode transaksi, nama penyewa, lokasi kost, nomor kamar, jumlah pembayaran (dalam rupiah), status pembayaran, tanggal masuk, dan tanggal keluar.

3. Kebutuhan Non-Fungsional

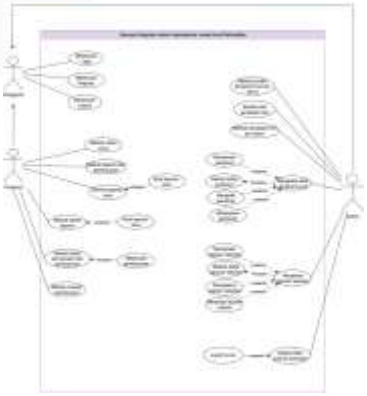
Kebutuhan non-fungsional merupakan kebutuhan pendukung yang tidak secara langsung memengaruhi jalannya fungsi utama sistem, namun berperan penting dalam meningkatkan kualitas, kinerja, dan kenyamanan penggunaan sistem secara keseluruhan. Adapun kebutuhan non-fungsional dari sistem ini ditunjukkan pada tabel 4 sebagai berikut:

TABEL 4
(KEBUTUHAN NON-FUNGSIONAL)

No	Kebutuhan non-fungsional	Penjelasan
1	Ketersediaan sistem	Sistem harus tersedia dan dapat diakses oleh pengguna selama 24 jam sehari.
2	Kinerja sistem	Sistem harus dapat memproses permintaan dengan cepat.
3	Responsivitas tampilan	Tampilan sistem harus menyesuaikan secara otomatis dengan ukuran layar pengguna (responsive).

B. *Design Workshop*

1. *Diagram UML*



GAMBAR 5
(USECASE DIAGRAM)

Pada Gambar 5 merupakan *use case diagram* keseluruhan dari sistem manajemen rumah kost Rahmatika. Diagram ini

menggambarkan dua aktor utama yang terlibat dalam sistem, yaitu penghuni dan *admin*. Aktor pengguna merupakan generalisasi dari kedua aktor tersebut, yang merepresentasikan entitas yang berinteraksi langsung dengan sistem, baik sebagai penghuni maupun sebagai *admin*.

2. *Rancangan Antarmuka*

Tahap ini dilakukan untuk merancang desain sederhana yang memberikan gambaran umum mengenai sistem yang akan dikembangkan. desain antarmuka ini disusun berdasarkan hasil identifikasi kebutuhan sistem yang telah ditetapkan sebelumnya. Desain tersebut diwujudkan dalam bentuk *wireframe* berbentuk low fidelity yang berfungsi sebagai representasi awal dari struktur dan alur sistem. *Wireframe* ini digunakan sebagai acuan awal dalam proses pengembangan lebih lanjut serta memudahkan komunikasi antara pengembang dan pihak terkait.

GAMBAR 6
(DESAIN *LOW-FIDELITY* SYSTEM)

3. Pembangunan Sistem

1) Register

GAMBAR 7
(HALAMAN WEBSITE "*REGISTER*")

Gambar 7 menunjukkan halaman website "*Register*" yang digunakan pengguna untuk mendaftar ke dalam sistem.

2) Login

GAMBAR 8
(HALAMAN WEBSITE "*LOGIN*")

Gambar 8 menunjukkan halaman website "Login" yang digunakan oleh pengguna maupun admin untuk mengakses sistem dengan mengisi email dan kata sandi.

3) Melihat Status Penghuni

GAMBAR 9
(HALAMAN WEBSITE "MELIHAT STATUS PENGHUNI")

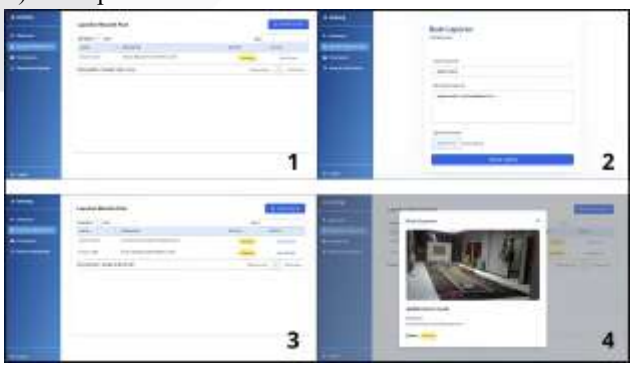
Gambar 9 menunjukkan halaman website "Melihat Status Penghuni" yang menampilkan ringkasan informasi terkait rumah kost yang ditempati.

4) Melihat Informasi Kost Admin

GAMBAR 10
(HALAMAN WEBSITE "MELIHAT INFORMASI KOST ADMIN")

Gambar 10 menunjukkan halaman website "Melihat Informasi Kost Admin" yang menyajikan data dalam bentuk card dan tiga diagram. Card pertama menampilkan jumlah penghuni aktif, card kedua memuat total pendapatan kost, dan card ketiga menunjukkan total laporan yang masuk. Diagram di bawahnya menyajikan rincian lebih lanjut dari data yang ditampilkan pada card-card tersebut.

5) Melaporkan Masalah Kost



GAMBAR 11
(HALAMAN WEBSITE "MELAPORKAN MASALAH KOST")

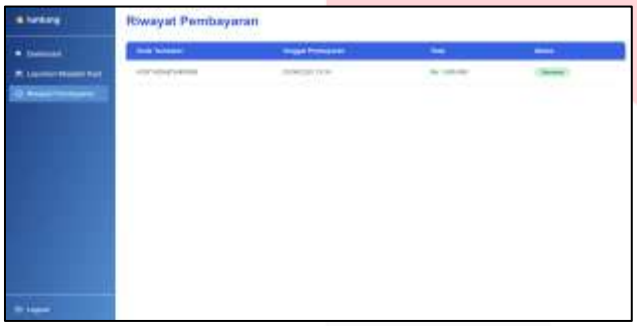
Gambar 4.7 menunjukkan halaman website "Melaporkan Masalah Kost" yang digunakan penghuni untuk melaporkan masalah terkait rumah kost melalui sistem.

6) Pembayaran

GAMBAR 12
(HALAMAN WEBSITE "PEMBAYARAN")

Gambar 12 menunjukkan halaman website "Pembayaran" yang digunakan penghuni untuk membayar tagihan kamar kost.

7) Riwayat Pembayaran



GAMBAR 13
(HALAMAN WEBSITE "RIWAYAT PEMBAYARAN")

Gambar 13 menunjukkan halaman website "Riwayat Pembayaran" yang memudahkan penghuni dalam memantau transaksi yang telah dilakukan.

8) Manajemen Data Penghuni

GAMBAR 14
(HALAMAN WEBSITE "MANAJEMEN DATA PENGHUNI")

Gambar 14 menunjukkan halaman website "Manajemen Data Penghuni". Pada halaman ini *admin* dapat memanajemen penghuni rumah kost. Dari mengubah, menambah, sampai menghapus data penghuni rumah kost.

9) Manajemen Data Laporan Keluhan

GAMBAR 15
(HALAMAN WEBSITE "MANAJEMEN DATA LAPORAN KELUHAN")

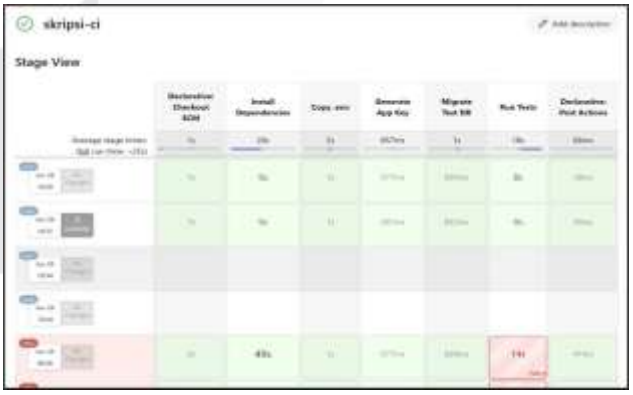
Gambar 15 menunjukkan halaman website "Manajemen Data Laporan Keluhan" yang digunakan *admin* untuk mengelola laporan penghuni.

10) Melihat Data Seluruh Transaksi

GAMBAR 16
(HALAMAN WEBSITE "MELIHAT DATA SELURUH TRANSAKSI")

Gambar 16 menunjukkan halaman website "Melihat Data Seluruh Transaksi" yang digunakan *admin* untuk memantau seluruh riwayat pembayaran dari penyewa kost.

4. Hasil CI



GAMBAR 17
(HASIL CONTINUOUS INTEGRATION)

Gambar 17 menunjukkan hasil Jenkins build yang terdiri dari tujuh tahapan, yaitu: (1) *Checkout SCM*, (2) *Install Dependencies*, (3) *Copy .env*, (4) *Generate App Key*, (5) *Migrate Test DB*, (6) *Run Tests*, dan (7) *Post Actions*. Seluruh tahapan berhasil dijalankan, mulai dari proses *checkout repository*, instalasi *dependencies*, penyalinan *file .env*, pembuatan *application key*, migrasi *test database*, hingga menjalankan *automated tests* dan pengiriman *success*

notification, sehingga pipeline CI berjalan dengan baik dan berhasil.

5. *Blackbox Testing*

Berikut hasil dari pengujian *blackbox* yang telah dilakukan:

TABEL 3
(HASIL PENGUJIAN *BLACKBOX TESTING 1)*

No	Fitur	*Test Case*	Hasil yang Diharapkan	Kesimpulan
1	Melaporkan Masalah Kost	Mengisi Judul Laporan, Deskripsi Laporan dan Gambar	Sistem dapat menyimpan data Laporan	Valid
2	Pembayaran	Klik “Bayar Sekarang” dan snap token midtrans muncul.	Pembayaran berhasil, dan data tersimpan	Valid
3	Riwayat Pembayaran	Akses fitur Riwayat Pembayaran	Data Riwayat Pembayaran akan ditampilkan	Valid

TABEL 4
(HASIL PENGUJIAN *BLACKBOX TESTING* 2)

No	Fitur	*Test Case*	Hasil yang Diharapkan	Kesimpulan
1	Manajemen Data Penghuni	Isi form dengan data penghuni baru dan klik tombol "Simpan"	Data penghuni baru berhasil ditambahkan dan disimpan dalam sistem	Valid
		Pilih penghuni yang sudah ada, ubah data penghuni, dan klik tombol "Simpan"	Data penghuni berhasil diubah dan disimpan dengan benar	Valid
		Klik tombol "Detail" pada salah satu penghuni dan lihat informasi lengkap yang ditampilkan	Menampilkan informasi lengkap penghuni seperti Nama, Email, No. telepon, Lokasi Kost, Status, dan Gambar.	Valid
		Pilih penghuni yang ingin dinonaktifkan, klik tombol "Hapus", dan periksa status penghuni tersebut.	Status penghuni berubah dari "Aktif" menjadi "Nonaktif".	Valid
2	Manajemen Laporan Keluhan	Isi form dengan data laporan baru (judul, deskripsi, gambar, nomor kamar) dan klik tombol "Simpan".	Laporan baru berhasil ditambahkan dan disimpan dalam sistem.	Valid
		Pilih laporan yang ingin dihapus dan klik tombol "Hapus".	Laporan berhasil dihapus dan tidak ada lagi di daftar laporan.	Valid
		Klik tombol "Detail" pada salah satu laporan dan periksa apakah informasi lengkap (gambar, judul, deskripsi, nomor kamar, status) ditampilkan.	Menampilkan informasi lengkap laporan sesuai yang diminta.	Valid
		Pilih laporan yang belum selesai, klik tombol "Tandai Selesai".	Laporan berhasil diubah statusnya menjadi "Selesai" dan informasi diperbarui.	Valid
3	Melihat Seluruh Data Transaksi	Klik fitur “Data Seluruh Transaksi”	Menampilkan keseluruhan data pembayaran yang telah dilakukan pengguna/pengghuni.	Valid

6. *User Acceptance Testing (UAT)*

Sebelumnya telah dilakukan pengujian *black-box*, namun untuk melanjutkan proses evaluasi kelayakan sistem, dilakukan pengujian menggunakan *user acceptance testing (UAT)*. Pengujian ini melibatkan 24 responden yang diperoleh, di mana masing-masing responden diminta menjawab pertanyaan yang tercantum pada Tabel V.4 dengan memberikan penilaian berdasarkan skala Likert 1–5.

TABEL 5
BOBOT PENILAIAN *SKALA LIKERT*

Bobot	Keterangan
1	Tidak Setuju (ST)
2	Kurang Setuju (KS)
3	Cukup Setuju (CS)
4	Setuju (S)
5	Sangat Setuju (SS)

Pertanyaan-pertanyaan pada evaluasi kuesioner disusun berdasarkan tiga variabel pengujian, yaitu: (1) Fungsionalitas sistem, (2) Efisiensi sistem, dan (3) Evaluasi antarmuka pengguna (User Interface/UI).

TABEL 6
(DAFTAR PERTANYAAN KUESIONER)

ID	Variabel	Pertanyaan
A1	Evaluasi fungsionalitas sistem	Apakah sistem ini memungkinkan penghuni untuk melakukan pembayaran dengan mudah dan aman melalui Midtrans?
A2		Apakah penghuni dapat melaporkan masalah atau keluhan dengan mudah menggunakan fitur laporan keluhan pada sistem?
A3		Apakah sistem ini dapat menampilkan riwayat pembayaran penghuni dengan akurat dan mudah diakses?
B1	Evaluasi efisiensi sistem	Apakah sistem ini membantu mengurangi waktu yang dibutuhkan untuk menyelesaikan tugas-tugas penghuni, seperti melakukan pembayaran dan melaporkan keluhan?

B2		Apakah sistem ini dapat mengurangi beban pekerjaan manual dalam mengelola data penghuni dan pembayaran?
B3		Apakah penggunaan sistem ini mengurangi kesalahan atau masalah yang sering terjadi dalam proses pengelolaan rumah kost?
C1	Evaluasi antarmuka (UI)	Apakah tampilan sistem pembayaran terlihat jelas dan mudah digunakan oleh penghuni untuk menyelesaikan transaksi dengan Midtrans?
C2		Apakah tampilan antarmuka laporan keluhan memudahkan penghuni untuk mengisi dan mengirimkan keluhan secara praktis?
C3		Apakah antarmuka manajemen penghuni pada sistem mudah dinavigasi oleh admin untuk mengelola data penghuni, pembayaran, dan laporan keluhan?
C4		Apakah tata letak dan desain tombol pada sistem intuitif, membantu pengguna dalam mengakses berbagai fitur seperti pembayaran, laporan keluhan, dan manajemen penghuni?

Setelah dilakukan perhitungan hasil kuesioner UAT yang telah dilakukan, berikut ringkasan dari penilaian tersebut yang disajikan pada tabel .

TABEL 7
(HASIL RINGKASAN AKHIR PERHITUNGAN UAT)

No	Variabel	Nilai bobot %	Keterangan
1	Fungsionalitas sistem	89%	Sangat baik
2	Efisiensi sistem	89%	Sangat baik
3	Antarmuka (UI)	89%	Sangat baik

Tabel 9 menyajikan hasil akhir berupa rata-rata dari masing-masing variabel evaluasi. Berdasarkan perhitungan, dapat disimpulkan bahwa hasil evaluasi kuesioner terhadap website manajemen rumah kost rahmatika tergolong dalam kategori sangat baik.

V. KESIMPULAN

Berdasarkan hasil penelitian "Penerapan *Continuous Integration* Dalam Pengembangan Aplikasi Manajemen Rumah Kost Menggunakan Metode *Rapid Application Development* (Studi Kasus: Rumah Kost Rahmatika)", dapat disimpulkan bahwa:

1. Website manajemen rumah kost Rahmatika berhasil dibangun menggunakan *Laravel* dengan pendekatan *Rapid Application Development (RAD)* serta *Continuous Integration (CI)* melalui *Jenkins*. Sistem menerapkan arsitektur *Model-View-Controller (MVC)* yang memudahkan implementasi dan pemeliharaan, serta terintegrasi dengan *Midtrans* untuk mendukung pembayaran digital. *Jenkins* berperan dalam mengotomatisasi proses *build* dan *testing*, sehingga mendukung pengembangan berkelanjutan.
2. Pengujian dilakukan menggunakan *blackbox testing* dan *user acceptance testing (UAT)*. Hasil *blackbox testing* menunjukkan seluruh fitur berfungsi sesuai skenario, sedangkan *UAT* yang melibatkan 23 responden menghasilkan tingkat kepuasan pengguna sebesar 89%, menandakan sistem telah memenuhi ekspektasi pengguna.

REFERENSI

[1] F. Asha Sabitha, "Analisis Pengaruh Tingkat Urbanisasi Terhadap Ketersediaan Lahan Permukiman Perumahan Di Kota Surabaya," *Jurnal Lembaga Ketahanan Nasional Republik Indonesia*, vol. 10, no. 1, p. 19, Mar. 2022.

[2] Badan Pusat Statistik, "Jumlah Mahasiswa (Negeri dan Swasta) di Bawah Kementerian Pendidikan dan Kebudayaan Menurut Kabupaten/Kota, 2021 dan 2022." Accessed: Jan. 13, 2025. [Online]. Available: https://jatim.bps.go.id/id/statistics-table/1/MjkzOCMx/jumlah-mahasiswa-negeri-dan-swasta-di-bawah-kementerian-pendidikan-dan-kebudayaan-menurut-kabupaten-kota-2021-dan-2022.html

[3] Badan Pusat Statisik Kota Surabaya, "Keadaan Ketenagakerjaan Kota Surabaya Agustus 2024," Nov. 2024.

[4] M. Pratama and K. Dana, "Implementasi Continuous Integration dan Continuous Delivery (CI/CD) Pada Automatic Performance Testing," Dec. 2020.

[5] P. Restu, S. Hudan, and A. Rizky, "Implementasi Continuous Integration dan Continuous Delivery Pada Aplikasi myITS Single Sign On," *TEKNIK ITS*, vol. 11, no. 3, 2022.

[6] A. Farid and I. Gita Anugrah, "Implementasi CI/CD Pipeline Pada Framework Androbase Menggunakan Jenkins (Studi Kasus: PT. Andromedia)," *Jurnal Nasional Komputasi dan Teknologi Informasi*, vol. 4, no. 6, 2021.

[7] M. Ardhiansyah, "Penerapan Model Rapid Application Development pada Aplikasi Helpdesk Trouble Ticket PT. Satkomindo Mediyasa," *Jurnal Teknologi Sistem Informasi dan Aplikasi*, vol. 2, no. 2, pp. 2654–4229, Apr. 2019, [Online]. Available: http://openjournal.unpam.ac.id/index.php/JTSI43

[8] E. Sutinah, I. Alfarobi, and A. Setiawan, "Metode Rapid Application Development dalam Pembuatan Sistem Informasi Pemenuhan SDM pada Perusahaan Outsourcing," *InfoTekJar : Jurnal Nasional Informatika dan Teknologi Jaringan*, vol. 5, no. 2, 2020, doi: 10.30743/infotekjar.v5i2.3528.

[9] A. Voutama and E. Novalia, "Perancangan Sistem Informasi Plakat Wisuda Berbasis Web Menggunakan UML dan Model Waterfall," *Syntax: Jurnal Informatika*, vol. 11, no. 01, 2022.

[10] L. Rahmawati and S. Sumarsono, "Desain Pengembangan Website dengan Arsitektur Model View Controller pada Framework Laravel," *Jurnal Teknologi Dan Sistem Informasi Bisnis*, vol. 6, no. 4, pp. 785–790, Oct. 2024, doi: 10.47233/jteksis.v6i4.1497.

[11] R. Yuniarti, I. H. Santi, and W. D. Puspitasari, "Perancangan Aplikasi Point of Sale untuk Manajemen Pemesanan Bahan Pangan Berbasis Framework Laravel," *Jati (Jurnal Mahasiswa Teknik Informatika)*, vol. 6, no. 1, Feb. 2022.

[12] S. Endah, S. Effendy, M. Dani, and B. Aji, "Pengujian Menggunakan Black Box Testing dengan Teknik State Transition Testing Pada Perpustakaan Yayasan Pendidikan Islam Pakualam Berbasis Web,"

JATIMIKA (Jurnal Kreativitas Mahasiswa Informatika), vol. 2, no. 1, Jan. 2022.

[13] Aliyah, Nahrun Hartono, and Asrul Azhari Muin, "Penggunaan User Acceptance Testing (UAT) Pada Pengujian Sistem Informasi Pengelolaan Keuangan Dan Inventaris Barang," *Switch : Jurnal Sains dan Teknologi Informasi*, vol. 3, no. 1, pp. 84–100, Dec. 2024, doi: 10.62951/switch.v3i1.330.