

RANCANG BANGUN SISTEM BACKEND RESTFUL API RIDE-SHARING WINJOL UNSIKA BERBASIS GOLANG DENGAN PENERAPAN ARSITEKTUR LAYERED

Rama Diaz¹, H. Bagja Nugraha², Billy Ibrahim H³

^{1,2,3}Universitas Singaperbangsa Karawang; Jl. HS.RonggoWaluyo, Puseurjaya, Telukjambe Timur Karawang, Indonesia; (0267) 641177

Keywords:
RESTful API;
Ride sharing;
Golang;
Arsitektur Layered.

Correspondent Email:
2210631250067@student.unsika.ac.id

Abstrak. Winjol UNSIKA merupakan komunitas ride-sharing mahasiswa Universitas Singaperbangsa Karawang yang masih mengandalkan grup WhatsApp untuk pemesanan, negosiasi tarif, dan pelaporan insiden. Kondisi tersebut menyebabkan fragmentasi komunikasi, ketidakseragaman tarif, lemahnya validasi identitas, serta minimnya pencatatan layanan. Penelitian ini bertujuan merancang dan membangun backend RESTful API berbasis Golang dengan layered architecture sebagai fondasi manajemen layanan terpusat. Metode penelitian menggunakan Rapid Application Development (RAD) yang diadaptasi menjadi tujuh tahapan pengembangan. Sistem dikembangkan menggunakan Golang, Gin Framework, PostgreSQL, GORM, JWT, Casbin, Redis, dan WebSocket, dengan fitur autentikasi, verifikasi driver, forum, real-time chat, perhitungan tarif, pemesanan, pelaporan insiden, dan langganan driver. Hasil pengujian black-box menggunakan Postman menunjukkan bahwa seluruh skenario utama berjalan sesuai kontrak API. Dengan demikian, sistem yang dikembangkan mampu menjadi fondasi transformasi Winjol UNSIKA menjadi platform layanan kampus yang terstruktur, aman, terdokumentasi, dan siap diintegrasikan dengan aplikasi klien.



Copyright © 2026 The Author(s), Published by Jurnal Informatika dan Teknik Elektro Terapan. This is an open-access article distributed under the terms of the Creative Commons Attribution-NonCommercial 4.0 International License (CC BY-NC 4.0).

Abstract. Winjol UNSIKA is a student ride-sharing community at Universitas Singaperbangsa Karawang that still relies on WhatsApp groups for ride requests, fare negotiations, and incident reporting. This condition causes communication fragmentation, inconsistent fares, weak identity validation, and limited service records. This study aims to design and develop a Golang-based RESTful API backend using layered architecture as the foundation for centralized service management. The Rapid Application Development (RAD) method was adapted into seven development stages. The system was developed using Golang, Gin Framework, PostgreSQL, GORM, JWT, Casbin, Redis, and WebSocket, covering authentication, driver verification, forums, real-time chat, fare calculation, ride ordering, incident reporting, and driver subscriptions. Black-box testing using Postman showed that all major scenarios operated according to the defined API contracts. Therefore, the developed system provides a foundation for transforming Winjol UNSIKA into a structured, secure, documented campus service platform ready for integration with client applications.

1. PENDAHULUAN

Mobilitas mahasiswa di Universitas Singaperbangsa Karawang (UNSIKA) terus meningkat seiring dengan pertumbuhan aktivitas kampus, sementara ketersediaan

transportasi umum di wilayah Karawang masih terbatas. Kondisi tersebut mendorong terbentuknya Winjol, yaitu komunitas ride-sharing internal mahasiswa sebagai alternatif transportasi harian. Namun, operasional Winjol

masih bergantung pada komunikasi melalui grup WhatsApp sehingga proses pemesanan, koordinasi perjalanan, penentuan tarif, dan pelaporan insiden dilakukan secara informal tanpa dukungan sistem dan basis data terstruktur. Kondisi ini menyebabkan fragmentasi informasi, ketidakseragaman tarif, lemahnya validasi identitas, serta kesulitan dalam pencatatan dan penelusuran histori layanan.

Digitalisasi layanan ride-sharing berbasis komunitas diperlukan untuk meningkatkan keamanan, kepercayaan, dan efektivitas pengelolaan layanan. [1] menyatakan bahwa persepsi risiko keamanan memengaruhi minat pengguna dalam menggunakan transportasi daring. Sementara itu, Ahma dan [2] serta [3] menunjukkan bahwa WhatsApp sebagai media komunikasi kelompok memiliki keterbatasan dalam mendukung layanan transaksional yang membutuhkan koordinasi dan pencatatan formal. Dalam konteks Winjol UNSIKA, belum tersedianya mekanisme validasi identitas pengemudi dan penumpang memungkinkan pihak di luar mahasiswa UNSIKA mengakses layanan sehingga dapat meningkatkan risiko keamanan.

Permasalahan lainnya adalah belum adanya standar tarif dan mekanisme pelaporan insiden yang terstruktur. Kejelasan informasi dan konsistensi harga merupakan faktor yang memengaruhi willingness to pay, kepuasan, serta loyalitas pengguna layanan ride-sharing [4], [5] Selain itu, ketersediaan mekanisme pelaporan yang terdokumentasi diperlukan untuk meningkatkan akuntabilitas dan kepercayaan pengguna terhadap layanan transportasi daring [6], [7], [8] Oleh karena itu, sistem ride-sharing kampus membutuhkan mekanisme terpusat yang mampu mengelola identitas pengguna, pemesanan, tarif, serta pelaporan insiden.

Ketiadaan sistem terpusat juga menyulitkan pemantauan permintaan perjalanan secara konsisten. Efektivitas layanan ride-sharing kampus bergantung pada ketersediaan informasi yang terorganisasi, proses pemesanan yang jelas, dan kemudahan akses pengguna [9], [10] Kondisi tersebut terlihat pada operasional Winjol UNSIKA, di mana rekapitulasi tiga grup WhatsApp selama November 2025 mencatat 229 permintaan perjalanan yang tersebar dan sulit ditelusuri lintas grup. Data tersebut

menunjukkan perlunya sistem terpusat yang mampu mencatat dan mengelola aktivitas layanan secara terstruktur.

Dari sisi teknis, RESTful API dipilih sebagai pola komunikasi backend karena mendukung interoperabilitas dan kemudahan pemeliharaan sistem. [11] menunjukkan bahwa penerapan prinsip RESTful memengaruhi tingkat keterpahaman API, sedangkan [12] menekankan pentingnya konsistensi kontrak endpoint dan penanganan kesalahan. Pemilihan teknologi dalam pengembangan RESTful API juga perlu mempertimbangkan aspek kinerja karena framework backend yang berbeda dapat menghasilkan waktu respons serta penggunaan sumber daya server yang berbeda [13]. Penggunaan Golang dengan framework Gin juga menawarkan performa dan efisiensi sumber daya yang relevan untuk menangani permintaan bersamaan pada sistem layanan digital [14].

Sejumlah penelitian sebelumnya telah menerapkan Golang dalam pengembangan backend dan RESTful API. [15] mengembangkan web service untuk modul berita, [16] menerapkan REST API untuk optimalisasi server e-commerce, sedangkan [17] dan [18] menerapkan Clean Architecture dalam pengembangan backend. [19] mengembangkan REST API berbasis microservices untuk aplikasi pencarian pekerjaan sampingan. Meskipun demikian, penelitian tersebut umumnya berfokus pada domain e-commerce, Point of Sales, dan portal informasi. Penerapan Golang dan arsitektur berlapis pada sistem ride-sharing kampus dengan kebutuhan pengelolaan peran pengguna, komunikasi real-time, perhitungan tarif berbasis jarak, dan pelaporan insiden masih terbatas.

Berdasarkan permasalahan dan kesenjangan penelitian tersebut, penelitian ini bertujuan merancang dan membangun backend RESTful API Winjol UNSIKA berbasis Golang dengan penerapan layered architecture sebagai fondasi manajemen layanan terpusat. Sistem yang dikembangkan mencakup autentikasi dan verifikasi pengguna, pengelolaan pemesanan, perhitungan tarif, komunikasi real-time, serta pelaporan insiden. Kontribusi penelitian ini terletak pada penerapan RESTful API, Golang, dan layered architecture dalam sistem backend yang dirancang sesuai dengan kebutuhan ride-

sharing kampus. Hasil penelitian diharapkan dapat menjadi referensi bagi pengembangan platform layanan transportasi berbasis komunitas di lingkungan pendidikan maupun sistem transportasi dengan ruang lingkup terbatas.

2. TINJAUAN PUSTAKA

2.1 Ride-Sharing dan Winjol

Ride-sharing merupakan layanan transportasi yang mempertemukan pengemudi dan penumpang melalui platform digital untuk melakukan perjalanan secara *on-demand* [10]. Layanan ini dapat meningkatkan efisiensi mobilitas melalui peningkatan okupansi kendaraan serta memberikan alternatif transportasi yang lebih fleksibel bagi pengguna. Winjol merupakan komunitas *ride-sharing* mahasiswa Universitas Singaperbangsa Karawang yang masih menggunakan grup WhatsApp sebagai media pemesanan dan komunikasi. Mekanisme tersebut menyebabkan proses operasional belum terstruktur dan data perjalanan tidak terdokumentasi secara terpusat sehingga diperlukan sistem yang mampu mengelola proses bisnis Winjol secara terintegrasi.

2.2 RESTful API dan Arsitektur Backend

Backend bertanggung jawab dalam menjalankan logika bisnis dan mengelola data, sedangkan *Application Programming Interface* (API) menjadi penghubung antara aplikasi klien dan server. RESTful API menggunakan protokol HTTP dengan metode GET, POST, PUT atau PATCH, dan DELETE untuk melakukan operasi terhadap sumber daya sistem [7]. Pertukaran data antara klien dan server umumnya menggunakan format JSON karena ringan dan kompatibel dengan berbagai platform [16]. Pengembangan backend dapat menerapkan *layered architecture* yang terdiri atas *controller*, *service*, dan *repository* untuk memisahkan tanggung jawab setiap komponen sehingga sistem lebih modular dan mudah dipelihara [15].

2.3 Rapid Application Development (RAD)

Rapid Application Development (RAD) merupakan metode pengembangan perangkat lunak yang menekankan kecepatan pengembangan, iterasi, dan keterlibatan

pengguna selama proses pembangunan sistem [17]. RAD terdiri atas tahapan *Requirements Planning*, *User Design*, *Construction*, dan *Cutover*. Pendekatan ini memungkinkan kebutuhan sistem diidentifikasi, dirancang, diimplementasikan, dan dievaluasi secara bertahap sehingga pengembangan dapat beradaptasi terhadap perubahan kebutuhan pengguna [6].

2.4 Keamanan API dan Basis Data

Keamanan API diperlukan untuk melindungi data dan membatasi akses terhadap sumber daya sistem melalui autentikasi dan otorisasi. Salah satu mekanisme yang dapat diterapkan adalah JSON Web Token (JWT), yaitu token yang memuat informasi identitas dan hak akses pengguna untuk diverifikasi oleh server [20]. Selain itu, sistem backend memerlukan basis data relasional untuk menyimpan data secara terstruktur dan menjaga hubungan antarentitas [2]. Pengelolaan basis data pada Golang dapat dilakukan menggunakan GORM sebagai *Object-Relational Mapping* (ORM) yang mempermudah pemetaan objek dalam kode program dengan tabel basis data [12].

2.5 Golang dan Gin Framework

Golang merupakan bahasa pemrograman yang mendukung performa tinggi dan *concurrency* sehingga sesuai digunakan dalam pengembangan sistem backend dan RESTful API [16]. Pengembangan API dapat menggunakan Gin Framework yang menyediakan fitur *routing*, *middleware*, serta pengelolaan HTTP *request* dan *response* dengan struktur yang ringan [14]. Kombinasi Golang dan Gin Framework mendukung pembangunan backend yang cepat, modular, dan mampu menangani kebutuhan layanan berbasis API [20].

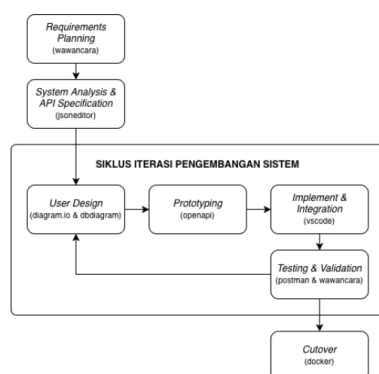
2.6 Pengujian Sistem

Pengujian sistem dilakukan untuk memastikan backend dan API berfungsi sesuai dengan kebutuhan yang telah ditentukan. *Unit testing* digunakan untuk menguji fungsi secara individual, sedangkan *integration testing* memastikan interaksi antarmodul berjalan dengan benar [12]. Selain itu, *performance testing* digunakan untuk mengevaluasi kinerja sistem berdasarkan waktu respons, *throughput*,

dan stabilitas ketika menerima beban tertentu [17].

3. METODE PENELITIAN

Penelitian ini menerapkan metode Rapid Application Development (RAD) yang diadaptasi menjadi tujuh tahapan operasional untuk mendukung perancangan dan pengembangan sistem backend RESTful API pada platform ride-sharing kampus Winjol. Pendekatan RAD dipilih karena mendukung pengembangan sistem backend secara cepat, iteratif, dan terstruktur, sekaligus memberikan fleksibilitas dalam penyempurnaan kebutuhan dan desain sistem berdasarkan hasil pengujian. Tahap user design, prototyping, implementation & integration, dan testing & validation ditempatkan dalam satu siklus iteratif sehingga ketidaksesuaian yang ditemukan pada tahap pengujian dapat langsung ditindaklanjuti melalui pembaruan rancangan tanpa mengulang keseluruhan proses.



Tahap requirements planning dilakukan dengan menelaah kondisi operasional Winjol UNSIKA yang masih bertumpu pada percakapan grup, mengelompokkan aktor sistem menjadi customer, driver, staff, dan admin, serta merumuskan kebutuhan fungsional dan non-fungsional. Tahap system analysis & API specification menerjemahkan kebutuhan tersebut menjadi kontrak teknis RESTful API yang mencakup pemetaan resource, spesifikasi endpoint, skema request-response, standar autentikasi berbasis JWT, serta kode status kesalahan yang seragam.

Tahap user design berfokus pada perancangan arsitektur backend berbasis layered architecture yang memisahkan lapisan controller, service, repository, dan model,

beserta rancangan struktur data per domain (autentikasi, forum dan pesan, pemesanan, pelaporan, serta pembayaran dan langganan). Hasil rancangan tersebut diteruskan ke tahap prototyping untuk dituangkan menjadi kontrak API menggunakan koleksi Postman, mencakup contoh request, response, dan skenario autentikasi sebelum proses pemrograman dilakukan.

Tahap implementation & integration merealisasikan kontrak API menggunakan bahasa pemrograman Golang dan framework Gin, dengan PostgreSQL dan PostGIS sebagai basis data, GORM sebagai ORM, Redis untuk manajemen sesi dan token, serta integrasi eksternal berupa Google OAuth 2.0, GraphHopper API untuk perhitungan rute, Midtrans Core API untuk pembayaran langganan, dan WebSocket untuk komunikasi real-time. Tahap testing & validation menggunakan pendekatan black-box testing melalui Postman untuk membandingkan perilaku aktual endpoint dengan kontrak yang telah ditetapkan, dilengkapi User Acceptance Testing (UAT) untuk menilai kesesuaian sistem dengan kebutuhan pengguna. Tahap akhir, cutover, mencakup pengemasan aplikasi menggunakan Docker dan Docker Compose, migrasi skema basis data, serta penyusunan dokumentasi teknis berupa referensi endpoint, spesifikasi OpenAPI, dan koleksi Postman.

4. HASIL DAN PEMBAHASAN

4.1 Kebutuhan Sistem

Tahap requirements planning menghasilkan kebutuhan fungsional yang merepresentasikan layanan inti Winjol, mulai dari registrasi pengguna hingga pengelolaan langganan driver, serta kebutuhan non-fungsional yang menjadi standar kualitas teknis sistem. Ringkasan kedua kategori kebutuhan tersebut disajikan pada Tabel 1 dan Tabel 2.

Tabel 1. Ringkasan Kebutuhan Fungsional Utama

Kode	Kebutuhan Fungsional
F-01–F-03	Registrasi pengguna dengan validasi email kampus, pendaftaran driver dengan validasi KTM/SIM/STNK, dan verifikasi dokumen oleh staff.
F-04–F-05	Pembuatan forum permintaan perjalanan sebagai media match making, serta real-

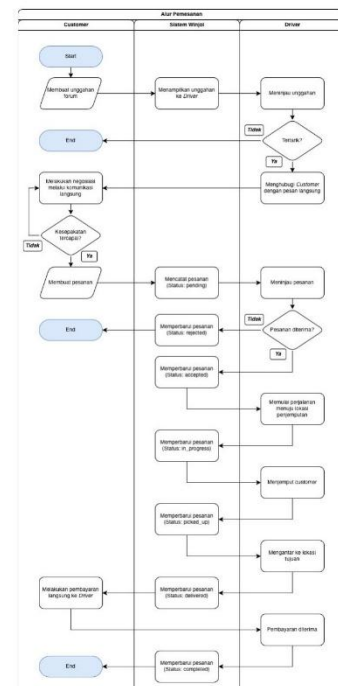
Kode	Kebutuhan Fungsional
	time chat berbasis forum untuk negosiasi customer-driver.
F-06–F-09	Estimasi tarif berdasarkan jarak rute, pembuatan order dari forum/room chat, pembaruan status order, dan penyimpanan histori order.
F-10–F-12	Penerimaan dan penyimpanan laporan insiden, pengelolaan langganan driver dan masa aktifnya, serta penyediaan rute perjalanan terpendek.

Tabel 2. Ringkasan Kebutuhan Non-Fungsional

Kode	Kebutuhan Non-Fungsional
NF-01	Sistem menggunakan arsitektur layered (controllers-services-repositories).
NF-02	Waktu respons API rata-rata < 300 ms pada beban normal.
NF-03–NF-04	Autentikasi berbasis JWT, dengan validasi input dan sanitasi parameter pada setiap endpoint.
NF-05–NF-07	Mampu menangani 100 request bersamaan, menggunakan standar RESTful berformat JSON, dan terdokumentasi melalui Postman.

4.2 Alur Proses Utama Sistem

Proses inti layanan ride-sharing Winjol dimulai ketika customer membuat unggahan forum sebagai permintaan perjalanan atau pengiriman. Driver yang tertarik dapat menghubungi customer melalui fitur real-time chat berbasis WebSocket untuk melakukan negosiasi sebelum order dibuat. Setelah kesepakatan tercapai, sistem mencatat order dengan status pending, menghitung estimasi tarif berdasarkan jarak rute melalui integrasi GraphHopper API, dan memperbarui status order secara bertahap mulai dari accepted, in_progress, picked_up, delivered, hingga completed. Alur lengkap proses pemesanan ditunjukkan pada Gambar 2.

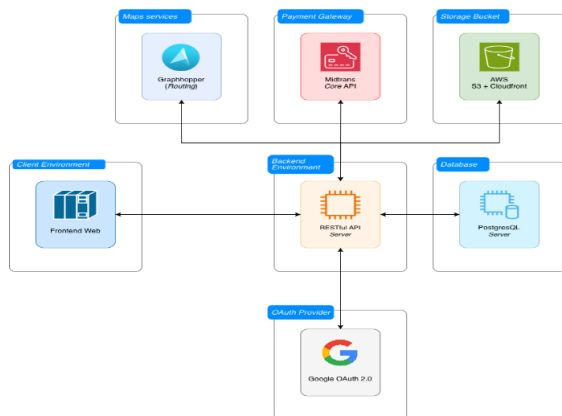


Gambar 2. Diagram Alur Pemesanan

Selain alur pemesanan, sistem juga mengatur alur autentikasi berbasis Google OAuth 2.0 yang memvalidasi domain email institusi sebelum menerbitkan token JWT, alur permohonan sebagai driver yang mensyaratkan unggahan dokumen KTM, SIM, dan STNK untuk diverifikasi staf, alur langganan driver yang terintegrasi dengan gerbang pembayaran Midtrans, serta alur pelaporan insiden yang menghubungkan pelapor, pihak terlapor, dan bukti pendukung dengan modul moderasi staf.

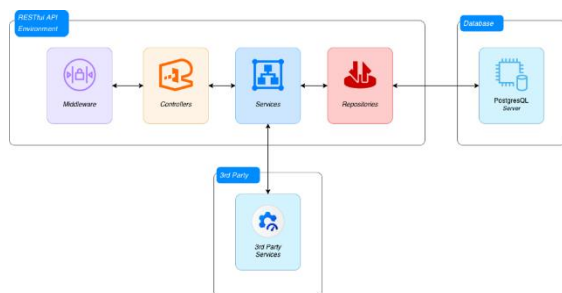
4.3 Arsitektur Sistem

Sistem Winjol dirancang menggunakan pola client-server dengan RESTful API sebagai pusat pengelolaan logika bisnis dan data, sebagaimana ditunjukkan pada Gambar 3. Backend berbasis Golang dan Gin berperan sebagai penghubung utama antara aplikasi klien, basis data PostgreSQL, dan layanan pihak ketiga seperti AWS S3 untuk penyimpanan dokumen serta Midtrans untuk manajemen langganan pengemudi.



Gambar 3. Diagram Arsitektur Sistem

Struktur internal backend menerapkan layered architecture yang memisahkan lapisan middleware, controller, service, dan repository, seperti diperlihatkan pada Gambar 4. Setiap permintaan pengguna diproses terlebih dahulu oleh lapisan middleware untuk autentikasi dan otorisasi berbasis Casbin, diteruskan ke controller untuk pemetaan data, diproses oleh service sebagai inti logika bisnis, dan akhirnya disimpan melalui repository. Pemisahan tanggung jawab ini mendukung prinsip separation of concerns yang mempermudah pemeliharaan dan pengembangan fitur baru tanpa mengganggu stabilitas komponen lain, sejalan dengan prinsip desain RESTful yang ditekankan oleh Bogner dan Kotstein (2023) serta konsistensi kontrak API yang disoroti Karlsson et al. (2024).



Gambar 4. Diagram Arsitektur Layered

4.4 Hasil Pengujian Fungsional

Pengujian fungsional dilakukan menggunakan pendekatan black-box testing melalui Postman terhadap delapan skenario utama yang mewakili domain fitur inti sistem. Rekapitulasi hasil pengujian iterasi pertama disajikan pada Tabel 3, yang menunjukkan bahwa seluruh endpoint memberikan respons kode status

HTTP yang sesuai dengan kontrak API, seperti 200 OK untuk permintaan sukses.

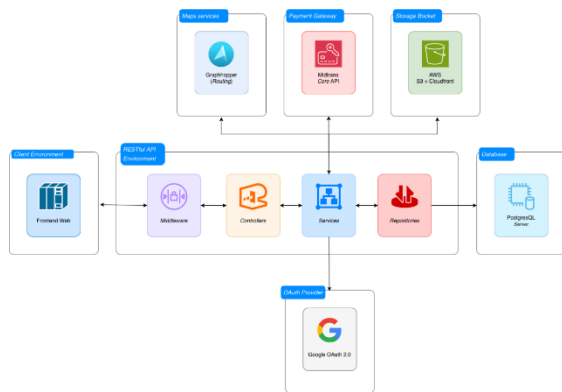
Tabel 3. Rekapitulasi Hasil Pengujian Fungsional

ID Tes	Domain Fitur	Skenario Pengujian	Status
TC-01	Autentikasi	Login menggunakan Google ID Token yang sah	OK
TC-02	Users	Memperbarui profil dan validasi nomor telepon OTP	OK
TC-03	Drivers	Mengajukan verifikasi pengemudi dengan unggah dokumen	OK
TC-04	Fares	Menghitung estimasi tarif berdasarkan koordinat GPS	OK
TC-05	Orders	Membuat pesanan perjalanan berdasarkan forum	OK
TC-06	Chats	Mengirim pesan teks dalam ruang obrolan pesanan	OK
TC-07	Subscriptions	Membuat transaksi langganan dan notifikasi Midtrans	OK
TC-08	Reports	Mengirim laporan insiden terkait perilaku pengguna	OK

Pada iterasi penyempurnaan, masukan dari User Acceptance Testing ditindaklanjuti dengan menambahkan detail alamat pemesanan yang lebih rinci serta deskripsi profil pengguna pada endpoint terkait. Pengujian ulang menunjukkan bahwa perubahan tersebut tidak menimbulkan regresi pada alur fungsional yang telah berjalan, sehingga sistem dinyatakan siap memasuki tahap cutover.

4.5 Integrasi Sistem dan Pencapaian Tujuan Penelitian

Sistem Winjol mengintegrasikan beberapa layanan eksternal sekaligus, yaitu Google OAuth 2.0 untuk autentikasi, GraphHopper untuk perutean dan estimasi jarak, Midtrans Core API untuk pembayaran langganan, serta AWS S3 dan CloudFront untuk penyimpanan dokumen, sebagaimana ditunjukkan pada Gambar 5.



Gambar 5. Diagram Integrasi Sistem

Pencapaian tujuan penelitian terhadap komponen implementasi teknis dirangkum pada Tabel 4, yang menunjukkan bahwa kelima tujuan penelitian berhasil diimplementasikan secara teknis.

Tabel 4. Matriks Pencapaian Tujuan Penelitian

No	Tujuan Penelitian	Komponen Implementasi Teknis	Status
1	RESTful API sebagai fondasi manajemen terpusat	15 modul publik dan 21 modul internal berbasis Golang Gin dengan arsitektur layered	Berhasil
2	Validasi keanggotaan pengemudi dan penumpang	Google OAuth 2.0, verifikasi nomor telepon via WhatsApp OTP, unggah dokumen ke AWS S3	Berhasil
3	Perhitungan tarif yang adil dan konsisten	FareService terintegrasi GraphHopper API dan konstanta tarif dinamis	Berhasil
4	Modul pelaporan insiden yang akuntabel	Model data Reports dan ReportEvidence terhubung modul administratif staf	Berhasil
5	Model monetisasi berlangganan pengemudi	Integrasi Midtrans Core API dan DriverSubscriptionMiddleware	Berhasil

4.6 Implementasi Algoritma dan Middleware Kunci

Sebagai ilustrasi teknis dari penerapan arsitektur layered, perhitungan tarif yang adil dan konsisten diimplementasikan melalui algoritma pada FareService yang terintegrasi dengan GraphHopper API. Logika perhitungan menggunakan konstanta FARE_INITIAL

untuk tarif dasar dan FARE_NEXT untuk biaya per kilometer, yang nilainya diambil secara dinamis dari variabel lingkungan sistem sehingga dapat disesuaikan tanpa mengubah kode program secara langsung, seperti ditunjukkan pada Gambar 6.

```

1 func CalculateFareBreakdown(distance int64, estimatedDuration *int64) FareBreakdown {
2     baseFare := config.GetFareInitial()
3     nextKilometerFare := config.GetFareNext()
4
5     var baseFareAmount int64
6     var distanceFareAmount int64
7
8     if distance <= 1000 {
9         baseFareAmount = baseFare
10        distanceFareAmount = 0
11    } else {
12        baseFareAmount = baseFare
13        additionalMeters := distance - 1000
14        perMeterFare := float64(nextKilometerFare) / 1000.0
15        additionalFare := float64(additionalMeters) * perMeterFare
16        distanceFareAmount = int64(additionalFare)
17    }
18
19    timeFareAmount := int64(0)
20    if estimatedDuration != nil {
21        timeFareAmount = calculateTimeFare(*estimatedDuration)
22    }
23
24    subtotal := baseFareAmount + distanceFareAmount + timeFareAmount
25    serviceFee := int64(0)
26    tax := int64(0)
27    discount := int64(0)
28
29    totalBeforeRounding := subtotal + serviceFee + tax - discount
30    roundedTotal := int64(math.Ceil(float64(totalBeforeRounding)/1000) * 1000)
31
32    return FareBreakdown{
33        BaseFare:    baseFareAmount,
34        DistanceFare: distanceFareAmount,
35        TimeFare:     timeFareAmount,
36        Subtotal:     subtotal,
37        ServiceFee:   serviceFee,
38        Tax:          tax,
39        Discount:    discount,
40        TotalAmount: roundedTotal,
41    }
42 }

```

Gambar 6. Cuplikan Kode Algoritma Perhitungan Tarif (FareService)

Pada sisi keamanan, setiap permintaan ke endpoint yang dilindungi diproses melalui middleware autentikasi berbasis JWT sebelum diteruskan ke lapisan controller. Middleware ini memvalidasi keberadaan dan keabsahan token, memeriksa status akun pengguna, serta menyisipkan konteks identitas pengguna ke dalam request sebelum diteruskan ke lapisan service, sebagaimana ditunjukkan pada Gambar 7. Pendekatan ini menempatkan mekanisme keamanan sebagai bagian eksplisit dari kontrak API dan bukan sekadar tambahan pada tahap implementasi, sejalan dengan penekanan Bucko et al. (2023) mengenai pentingnya penguatan autentikasi dan otorisasi berbasis JWT melalui pemeriksaan konteks perilaku pengguna.

```

1 func AuthMiddleware() gin.HandlerFunc {
2     return func(c *gin.Context) {
3         secret := config.GetJWTSecret()
4         var secretKey = []byte(secret)
5
6         authHeader := c.GetHeader("Authorization")
7         if authHeader == "" {
8             c.AbortWithStatusJSON(http.StatusForbidden,
9                 exceptions.NewException(http.StatusForbidden, exceptions.ErrForbidden))
10            return
11        }
12
13        authHeaderParts := strings.Split(authHeader, " ")
14        if len(authHeaderParts) != 2 || authHeaderParts[0] != "Bearer" {
15            c.AbortWithStatusJSON(http.StatusForbidden,
16                exceptions.NewException(http.StatusForbidden, exceptions.ErrInvalidCredentials))
17            return
18        }
19
20        tokenString := authHeaderParts[1]
21
22        isBlacklisted, := helpers.IsTokenBlacklisted(tokenString)
23        if isBlacklisted {
24            c.AbortWithStatusJSON(http.StatusUnauthorized,
25                exceptions.NewException(http.StatusUnauthorized, "Token is blacklisted"))
26            return
27        }
28
29        claims := jwt.MapClaims{}
30        token, err := jwt.ParseWithClaims(tokenString, claims, func(token *jwt.Token) (interface{}, error) {
31            if _, ok := token.Method.(*jwt.SigningMethodHMAC); !ok {
32                return nil, fmt.Errorf("unexpected signing method: %v", token.Header["alg"])
33            }
34            return secretKey, nil
35        })
36
37        if err != nil {
38            if errors.Is(err, jwt.ErrTokenExpired) {
39                c.AbortWithStatusJSON(http.StatusUnauthorized,
40                    exceptions.NewException(http.StatusUnauthorized, exceptions.ErrInvalidCredentials))
41                return
42            }
43            c.AbortWithStatusJSON(http.StatusForbidden,
44                exceptions.NewException(http.StatusForbidden, exceptions.ErrInvalidCredentials))
45            return
46        }
47
48        if !token.Valid {
49            c.AbortWithStatusJSON(http.StatusForbidden,
50                exceptions.NewException(http.StatusForbidden, exceptions.ErrInvalidCredentials))
51            return
52        }
53
54        var rules []string
55        if rulesInterface, ok := claims["rules"].([]interface{}); ok {
56            rules = make([]string, len(rulesInterface))
57            for i, r := range rulesInterface {
58                if ruleStr, ok := r.(string); ok {
59                    rules[i] = ruleStr
60                }
61            }
62        } else if ruleStr, ok := claims["rule"].(string); ok {
63            rules = []string{ruleStr}
64        } else {
65            rules = []string{"customer"}
66        }
67
68        var mid string
69        if midVal, ok := claims["uid"].(string); ok {
70            mid = midVal
71        }
72
73        var email string
74        if emailVal, ok := claims["email"].(string); ok {
75            email = emailVal
76        }
77    }
78 }

```

Gambar 7. Cuplikan Kode Middleware Autentikasi JWT

Secara keseluruhan, hasil penelitian ini menegaskan bahwa pendekatan perancangan yang terstruktur, pemisahan tanggung jawab pada setiap layer, dan penerapan standar RESTful API mampu menghasilkan sistem backend yang modular, terukur, dan siap diintegrasikan dengan aplikasi klien. Temuan ini memperluas penelitian terdahulu yang umumnya berfokus pada domain e-commerce dan Point of Sales [15], [16], [17], [18], [19] dengan menghadirkan model desain backend Golang berarsitektur layered yang teruji pada konteks ride-sharing kampus, sekaligus menjawab kebutuhan validasi identitas, kepastian tarif, akuntabilitas pelaporan, dan keberlanjutan monetisasi yang diidentifikasi pada bagian pendahuluan.

5. KESIMPULAN

5.1 Kesimpulan

Sistem backend RESTful API Winjol berhasil dikembangkan menggunakan Golang dengan penerapan arsitektur berlapis yang

terdiri atas controller, service, dan repository, sehingga mampu menyediakan pengelolaan data dan layanan komunitas ride-sharing secara terpusat, modular, dan terstruktur. Integrasi Google OAuth 2.0, AWS S3, GraphHopper API, dan Midtrans Core API mendukung proses validasi pengguna, penyimpanan dokumen, perhitungan tarif perjalanan, serta sistem langganan driver. Hasil pengujian fungsional pada delapan domain fitur utama menunjukkan bahwa seluruh fungsi sistem berjalan sesuai kontrak API setelah dilakukan penyempurnaan pada iterasi kedua. Dengan demikian, sistem backend yang dikembangkan dapat menjadi fondasi digitalisasi layanan Winjol UNSIKA yang lebih terorganisasi dan terintegrasi.

5.2 Saran

Penelitian selanjutnya dapat mengembangkan validasi identitas berbasis biometrik dan mempertimbangkan penerapan arsitektur microservices apabila jumlah pengguna dan beban sistem meningkat. Pengembangan sistem juga dapat dilakukan melalui integrasi data lalu lintas real-time dan pembayaran elektronik untuk meningkatkan akurasi tarif serta kemudahan transaksi perjalanan. Selain itu, data histori perjalanan dan laporan insiden dapat dimanfaatkan menggunakan teknik analisis data atau machine learning untuk mendukung prediksi permintaan dan menghasilkan rekomendasi operasional layanan yang lebih efektif.

UCAPAN TERIMA KASIH

Penulis mengucapkan terima kasih kepada pihak-pihak terkait yang telah memberi dukungan terhadap penelitian ini.

DAFTAR PUSTAKA

- [1] D. Aulia, N. Maswatu, D. I. Inan, R. Juita, and M. Indra, "The Role of Social Influence and Security Risk in Shaping Intention to Use Ride-Hailing in West Papua: A Theory of Planned Behavior Perspective," *J. Ref.*, vol. 14, pp. 208–215, 2025.
- [2] H. A. M. I. R. Ahma and I. Rwanyah, "Mobile Apps WhatsApp sebagai Media Komunikasi

- dan Informasi: Studi Literatur Sistematis,” *J. Ref.*, vol. 15, no. 1, 2024.
- [3] A. A. Mahatta, “Pemanfaatan WhatsApp sebagai Media Efektif dalam Komunikasi Kelompok Modern,” *J. Ref.*, vol. 2, no. 2, pp. 372–380, 2025.
- [4] K. Rofifah and A. Aprianingsih, “Understanding Nebengers’ User Willingness to Pay: The Impact of Perceived Value and Risk on Ride-Sharing,” *Int. Res. J. Econ. Manag. Stud.*, vol. 4, no. 2, pp. 170–178, 2025, doi: 10.56472/25835238/IRJEMS-V4I2P118.
- [5] S. K. Jena and A. Ghadge, “Price Competition in Ride-Sharing Platforms: A Duopoly Supply Chain Perspective,” *Comput. & Ind. Eng.*, vol. 183, p. 109507, 2023, doi: 10.1016/j.cie.2023.109507.
- [6] M. Afifudin, R. T. Rahman, and A. Bahar, “Driver Safety Protocols, Emergency Response Systems, Insurance Coverage, and Customer Feedback on Service Quality in Indonesia’s Ride-Hailing Industry,” *J. Ref.*, vol. 5, no. 2, pp. 2584–2595, 2024.
- [7] H. Wolf, D. Storch, M. Timme, and M. Schröder, “Spontaneous Symmetry Breaking in Ride-Sharing Adoption Dynamics,” *arXiv Prepr.*, pp. 1–12, 2022.
- [8] S. Hu and Y. Yang, “Safety of Female Ride-Hailing Passengers: Perception and Prevention,” *Humanit. Soc. Sci. Commun.*, vol. 11, no. 1, 2024, doi: 10.1057/s41599-024-02740-z.
- [9] T. Anagnostopoulos, “A Vehicle Ride-Sharing Algorithm Assessing Passenger Satisfaction According to Spatial, Temporal, and Social Behavior Context Based on Real Data Sources,” *J. Ref.*, pp. 1–26, 2025.
- [10] L. Mitropoulos, A. Kortsari, and G. Ayfantopoulou, “A Systematic Literature Review of Ride-Sharing Platforms, User Factors and Barriers,” *Eur. Transp. Res. Rev.*, pp. 1–22, 2021, doi: 10.1186/s12544-021-00522-1.
- [11] J. Bogner and S. Kotstein, “Do RESTful API Design Rules Have an Impact on the Understandability of Web APIs?,” 2023.
- [12] S. Karlsson, R. Jongeling, A. Čaušević, and D. Sundmark, “Exploring Behaviours of RESTful APIs in an Industrial Setting,” *Softw. Qual. J.*, vol. 32, no. 3, 2024, doi: 10.1007/s11219-024-09686-0.
- [13] W. Hadinata and L. Stianingsih, “Analisis Perbandingan Performa RESTful API antara Express.js dengan Laravel Framework dengan JMeter,” *J. Inform. dan Tek. Elektro Terap.*, vol. 12, no. 1, pp. 531–540, 2024, doi: 10.23960/jitet.v12i1.3845.
- [14] M. Avatara and R. Tan, “Implementasi Framework Gin dan gRPC pada Pengembangan Back-End Web,” *J. Ref.*, vol. 6, pp. 52–57, 2024.
- [15] S. Priadi and M. Idris, “Implementasi REST dalam Membangun Web Service Menggunakan Golang (Studi Kasus: Modul Berita Aplikasi Satudikti),” *J. Ref.*, vol. 8, pp. 349–358, 2023.
- [16] Y. Muharam, “Pengembangan Aplikasi Back-End E-Commerce Menggunakan REST API Golang untuk Optimalisasi Kinerja Server,” *J. Ref.*, vol. 11, pp. 7–13, 2024.
- [17] R. Annisa, R. A. Ananda, and W. E. Sulistiono, “Implementasi Golang Clean Architecture pada Perancangan Backend Point of Sales,” *J. Ref.*, vol. 12, no. 2, 2024.
- [18] F. Pamungkas, “Implementasi Clean Architecture pada Pembuatan API Menggunakan Golang,” *J. Ref.*, vol. 9, no. April, pp. 80–86, 2024.
- [19] A. A. Damanik and A. Voutama, “Pengembangan REST API untuk Aplikasi Pencarian Pekerjaan Sampingan dengan Arsitektur Microservices Menggunakan Metode Waterfall,” *J. Ref.*, vol. 19, no. 2, pp. 210–224, 2025.
- [20] A. P. Sitorus and D. R. Nugroho, “Efektivitas Komunikasi Grup WhatsApp Organisasi Mahasiswa di Sekolah Vokasi,” *ArtComm*, vol. 8, no. 2, pp. 266–279, 2025.