IMPLEMENTASI *JSON PARSING* UNTUK PERTUKARAN DATA PADA APLIKASI VPN *CLIENT* BERBASIS *MOBILE*

IMPLEMENTATION OF JSON PARSING FOR DATA EXCHANGE IN MOBILE-BASED VPN CLIENT APPLICATIONS

Nandana Surya Guna[1)], **Alam Rahmatulloh**[2)], **Andi Nur Rachman**[3)]

[1), 2), 3)] *Program Studi Informatika, Fakultas Teknik, Universitas Siliwangi*
Jl. Siliwangi No. 24 Kahuripan, Kota Tasikmalaya
Email : nsuryaguna@gmail.com[1)], alam@unsil.ac.id[2)], andy.rachman@unsil.ac.id[3)]

Abstrak

Meningkatnya permintaan akan transmisi data yang aman pada perangkat mobile, penting untuk mengembangkan metode yang efisien untuk pertukaran data sambil menjaga integritas dan kerahasiaan informasi. Penelitian ini bertujuan untuk mengatasi masalah implementasi JSON Parsing dalam pertukaran data pada aplikasi klien VPN berbasis mobile menggunakan layanan web. Salah satu tantangan yang dihadapi selama proses pengembangan adalah pembatasan alamat IP web service. Untuk mengatasi pembatasan ini, pengguna harus menghubungkan perangkat ke jaringan VPN, yang memungkinkan untuk mengubah kata sandi dan membuat catatan kehadiran dengan aman. Penelitian ini juga mencakup pengujian penerimaan untuk mengevaluasi kinerja aplikasi. Hasil pengujian menunjukkan hasil yang memuaskan, menandakan keberhasilan implementasi JSON Parsing dan fungsi layanan web. Selain itu, pengujian kinerja jaringan mengungkapkan bahwa waktu respons dipengaruhi oleh metode dan lokasi jaringan yang dipilih. Waktu respons untuk pengambilan data kehadiran pada jaringan default adalah 1187 ms, namun respons untuk pengiriman data kehadiran dan pengubahan kata sandi adalah 403 karena web service membatasi akses dengan alamat IP. Sementara itu, jaringan SgVPN untuk permintaan pengubahan password memiliki waktu respons sekitar 1088 ms, pengambilan data kehadiran sekitar 1727 ms, dan pengiriman data kehadirans sekitar 955 ms. Sedangkan pada jaringan DeVPN, permintaan pengubahan password memiliki waktu respons sekitar 1264 ms, pengambilan data kehadiran sekitar 1793 ms, dan pengiriman data kehadirans sekitar 1351 ms. Temuan ini menekankan pentingnya mempertimbangkan lokasi jaringan dalam upaya mengoptimalkan waktu respons aplikasi.

Kata kunci: *Aplikasi Klien VPN, Jaringan VPN, JSON Parsing, Layanan Web, Waktu Respons*

Abstract

The increasing demand for secure data transmission on mobile devices, it is important to develop efficient methods for exchanging data while maintaining the integrity and confidentiality of information. This study aims to address the problem of implementing JSON Parsing in exchanging data on mobile-based VPN client applications using web services. One of the challenges faced during the development process was limiting the IP address of the web service. To overcome this limitation, users must connect device to a VPN network, which allows to change passwords and create attendance records securely. This research includes acceptance testing to evaluate application performance. The test results show satisfactory results, indicating the successful implementation of JSON Parsing and web service functions. In addition, network performance tests reveal that response times are affected by the selected network method and location. Response time for retrieving attendance data on the default network is 1187 ms, but the response for sending attendance data and changing passwords is 403 because the web service limits access by IP address. Meanwhile, the SgVPN network for password change requests has a response time of around 1088 ms, taking attendance data is around 1727 ms, and sending attendance data is around 955 ms. Whereas on the DeVPN network, requests to change passwords have a response time of around 1264 ms, attendance data retrieval is around 1793 ms, and attendance data transmission is around 1351 ms. These findings emphasize the importance of considering network location in an effort to optimize application response time.

Keywords: *JSON Parsing, Response Time, VPN Client, VPN Network, Web Service*

1. PENDAHULUAN

Penggunaan jaringan dalam perusahaan semakin meningkat seiring dengan meningkatnya hubungan dengan jaringan luar. Namun, potensi ancaman keamanan data juga semakin besar [1]. Untuk menjaga keamanan data saat terhubung dengan jaringan, penggunaan

Virtual Private Network (VPN) semakin populer sebagai solusi yang efektif [2]. Namun, dalam pengembangan aplikasi *mobile multiplatform*, pertukaran data melalui jaringan menjadi hal penting. Koneksi internet menggunakan protokol HTTP atau HTTPS diperlukan untuk memastikan pertukaran data berjalan dengan baik [3]. Dalam konteks ini, *middleware* menjadi solusi dengan mengombinasikan logika pemrograman dan basis data terdistribusi [3].

Penelitian ini berfokus pada pengembangan aplikasi presensi mahasiswa dengan menggunakan metode *personal extreme programming* dan metode MoSCoW. Penelitian ini mengimplementasikan JSON *parsing* untuk pertukaran data pada aplikasi VPN *client* berbasis *mobile* menggunakan *web service*. *JavaScript Object Notation* (JSON) dipilih sebagai format pertukaran data karena keunggulannya yang ringan, mudah dibaca dan ditulis oleh manusia, serta mudah diterjemahkan dan dibuat oleh komputer. JSON telah digunakan dalam beberapa penelitian sebelumnya [4]–[6] dan terbukti efektif.

Metode *personal extreme programming* dan metode MoSCoW digunakan dalam pengembangan aplikasi presensi mahasiswa untuk memastikan kualitas aplikasi yang dihasilkan. Contoh penggunaan metode *personal extreme programming* dapat ditemukan dalam pengembangan aplikasi kamus bahasa Jepang berbasis Android yang dilakukan oleh Rustam [7]. Dalam penelitian tersebut, aplikasi berhasil membantu pengguna dalam menerjemahkan bahasa Indonesia ke bahasa Jepang dan sebaliknya. Selain itu, metode MoSCoW juga telah digunakan dalam berbagai konteks, seperti dalam industri pelayanan katering maskapai penerbangan selama pandemi COVID-19 [8] dan dalam meningkatkan efektivitas rantai pasokan makanan melalui platform informasi [9].

Web service memungkinkan berbagai aplikasi komputer berkomunikasi melalui jaringan internet menggunakan protokol dan standar seperti HTTP dan XML. Penggunaan web service telah terbukti efektif dalam berbagai konteks, seperti dalam sistem informasi jual beli barang bekas [10] dan sistem pendukung penyediaan informasi lowongan kerja [11]. Integrasi *web service* juga diterapkan dalam lingkup perpustakaan universitas [12].

Mengacu pada penelitian Bhakti Destian Wijaya, Fenty E.M.A., dan Andrew Fiade [13], implementasi JSON *parsing* berhasil dilakukan untuk mengambil data dari *website* dan ditampilkan ke dalam sebuah aplikasi *mobile e-commerce*. Penelitian ini menggunakan metode HTTP *connection* dan JSON *parsing* untuk menghubungkan aplikasi klien dengan *database* yang ada di dalam server [13]. Selanjutnya penelitian yang dilakukan oleh Ester Lumba [14], membahas tentang pertukaran data pada aplikasi Android menggunakan JSON dan REST API. Penelitian ini menggunakan *library* Retrofit 2 untuk mempermudah implementasi JSON *parsing* [14].

Tujuan dari penelitian ini adalah untuk mengimplementasikan JSON *parsing* dalam pertukaran data pada aplikasi VPN *client* berbasis *mobile* menggunakan *web service*. Penelitian ini juga menerapkan metode *personal extreme programming* dan metode MoSCoW dalam pengembangan aplikasi presensi mahasiswa. Dengan demikian, penelitian ini diharapkan dapat memberikan kontribusi dalam pengembangan aplikasi mobile yang aman, efektif, dan sesuai dengan kebutuhan pengguna.

2. DASAR TEORI

2.1. Jaringan *Virtual Private Network* (VPN)

VPN berfungsi sebagai sebuah tunnel di dalam jaringan internet yang menghubungkan antar jaringan, serta dilakukan enkripsi pada data tersebut [2]. Dengan menggunakan VPN, perusahaan dapat membentuk saluran komunikasi yang aman dan terenkripsi antara pengguna dan jaringan internal perusahaan [1]. Hal ini memungkinkan data yang dikirim melalui jaringan publik tetap terlindungi dari potensi serangan dan penyadapan oleh pihak yang tidak berwenang.

2.2. *JavaScript Object Notation* (JSON)

JavaScript Object Notation (JSON) merupakan sebuah format pertukaran data yang memiliki karakteristik ringan, mudah dibaca dan ditulis oleh manusia, serta mudah diterjemahkan dan dibaca oleh komputer. Selain itu, JSON juga digunakan dalam *web service* yang merupakan standar untuk implementasi sistem berbasis *web* saat ini. Dalam sebuah penelitian, Tiwary [15] mengajukan teknik kompresi baru untuk dokumen XML yang dapat meningkatkan efisiensi komunikasi dan kinerja sistem berorientasi layanan.

2.3. Pertukaran Data Melalui Jaringan

Pertukaran data melalui jaringan merupakan salah satu hal yang penting dalam pengembangan aplikasi *mobile* yang berbasis multiplatform. Purwoko [3] menyebutkan bahwa dalam kondisi multiplatform, aplikasi memerlukan koneksi internet agar dapat terhubung dengan pengguna. Hal ini terjadi karena aplikasi menggunakan protokol HTTP atau HTTPS untuk pertukaran data, yang memungkinkan aliran data yang lancar antara aplikasi dan pengguna.

2.4. Aplikasi *Mobile* Berbasis *Android*

Aplikasi *mobile* memberikan kemudahan akses informasi dan penggunaannya lebih fleksibel dibandingkan dengan aplikasi *web*. Beberapa penelitian telah dilakukan dalam pengembangan aplikasi mobile berbasis Android untuk berbagai kebutuhan. Rustam [7] mengembangkan aplikasi kamus bahasa Jepang berbasis mobile android menggunakan Applybuilder/MIT Inventor dengan metode *extreme programming*. Hasil penelitian menunjukkan bahwa aplikasi tersebut dapat membantu pengguna dalam menerjemahkan bahasa Indonesia ke bahasa Jepang dan sebaliknya. Aplikasi Android dapat dibagun dengan beberapa *framework* dan bahasa pemrograman seperti Android Studio dengan Java dan Kotlin, serta Flutter dengan bahasa pemrograman Dart. Flutter memungkinkan pengembang membangun aplikasi untuk *platform* Android dan iOS dalam satu *source code* [16].

2.5. *Web Service*

Web service adalah teknologi yang memungkinkan berbagai aplikasi komputer berkomunikasi dan berinteraksi melalui jaringan internet. *Web service* memanfaatkan protokol dan standar tertentu, seperti HTTP (*Hypertext Transfer Protocol*) dan XML (*eXtensible Markup Language*), untuk pertukaran data antara aplikasi yang berbeda. Dalam konteks penelitian yang dilakukan oleh Maisaroh [11], *web service* digunakan sebagai sistem pendukung dalam menyajikan informasi lowongan kerja kepada para pencari kerja. Dengan menggunakan *web service*, informasi tersebut dapat diakses melalui *website* dan aplikasi berbasis Android.

2.6. Metode MoSCoW (*Must have, should have, could have, and won't have*)

Metode MoSCoW adalah teknik prioritisasi kebutuhan yang dapat digunakan untuk mengelompokkan kebutuhan ke dalam 4 (empat) kelas berdasarkan tingkat prioritasnya, yaitu *Must have*, lalu *Should have*, selanjutnya *Could have*, dan *Won't have*. Metode ini digunakan dalam penelitian oleh Rajaratnam [8] untuk memprioritaskan *performance metrics* yang dibutuhkan dalam industri pelayanan katering maskapai penerbangan selama pandemi COVID-19. Sedangkan, Marthasari [17] mengimplementasikan teknik ini pada pengukuran usabilitas *website* dengan modifikasi terhadap *framework* Sirius. Hasil penelitian yang dilakukan oleh Marthasari menunjukkan bahwa penggunaan teknik MoSCoW memberikan prioritas yang lebih baik dalam menentukan kriteria *usability* berdasarkan nilai bisnis, resiko, dan biaya pembangunan sebuah *website*.

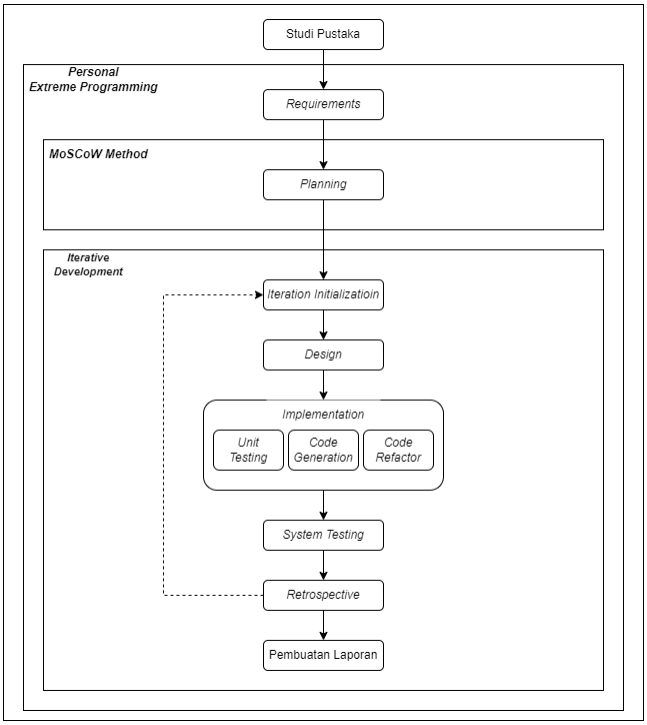
2.7. Metode *Personal Extreme Programming*

Personal *Extreme Programming* (PXP) adalah metode pengembangan perangkat lunak yang merupakan pengembangan dari *extreme programming* yang dapat digunakan oleh pengembang tunggal. PXP terdiri dari tujuh tahap yang meliputi *requirement*, *planning*, *iteration initialization*, *design*, *implementation*, *system testing*, dan *retrospective*. Salah satu penelitian yang menerapkan *Personal Extreme Programming* adalah penelitian yang dilakukan oleh Hasan [18]. Dalam penelitian ini, metode PXP digunakan untuk mengembangkan sistem pelayanan sistem terpadu desa. Hasil dari penelitian ini menunjukkan bahwa metode PXP dapat

diimplementasikan dalam studi kasus pengembangan sistem pelayanan terpadu desa, dengan waktu pengembangan yang lebih cepat dan efisien.

3. METODOLOGI PENELITIAN

Metode penelitian diawali dengan melakukan studi pustaka menggunakan sumber pengetahuan yang terkait dengan penelitian ini dan observasi langsung terhadap objek penelitian. Metode pengembangan perangkat lunak menggunakan *Personal Extreme Programming* (PXP). Sedangkan untuk menentukan prioritas pengerjaan digunakan metode MoSCoW. Studi kasus yang digunakan adalah aplikasi presensi mahasiswa untuk mengirim maupun menerima catatan kehadiran.



Gambar 1. *Tahapan Penelitian*

Gambar 1 menunjukkan tahapan penelitian yang dilakukan pada penelitian ini mulai dari pelaksanaan studi pustaka hingga pembuatan laporan.

3.1. Studi Pustaka

Penelitian ini menggunakan studi pustaka untuk memperdalam pemahaman tentang *Personal Extreme Programming* (PXP), *Virtual Private Network* (VPN), dan pengembangan aplikasi Android. Sumber studi pustaka yang digunakan termasuk buku, jurnal, internet, dan sumber pustaka lainnya. Dengan memanfaatkan berbagai sumber ini, penelitian ini dapat menyajikan informasi yang komprehensif dan mendukung analisis mendalam tentang pengembangan sistem, metode PXP, dan VPN.

3.2. Pengembangan Sistem

Metode *Personal Extreme Programming* (PXP) adalah pendekatan pengembangan yang iteratif dan fleksibel dalam menangani perubahan kebutuhan sistem. PXP memecah setiap user stories menjadi task-task yang lebih kecil. Tahapan implementasi PXP meliputi *requirements*, *design*, *coding*, *testing*, dan *retrospective*.

a. *Requirements*

Pada tahap Requirements, dilakukan proses pembuatan dokumen kebutuhan fungsional dan nonfungsional secara opsional. Dokumen ini dapat disusun melalui pertemuan klien dan pengembang, atau dengan memanfaatkan dokumen kebutuhan yang sudah ada. Diskusi

melibatkan mahasiswa program studi Informatika angkatan 2018 di Universitas Siliwangi untuk memahami kebutuhan sistem.

b. *Planning*

Pada tahap *Planning* (Perencanaan), tugas-tugas diorganisasikan berdasarkan dokumen kebutuhan yang telah disusun sebelumnya. Tugas-tugas ini dibagi menjadi tugas-tugas kecil yang dikategorikan sesuai kebutuhan. Perkiraan waktu penyelesaian tugas-tugas kecil didasarkan pada data perencanaan dari proyek serupa atau asumsi terbaik jika data tidak tersedia. Estimasi waktu ini digunakan untuk tugas utama. Selanjutnya, *user stories* dibuat dari tugas-tugas yang telah diorganisasikan. Prioritas kebutuhan ditentukan menggunakan metode MoSCoW, yang mengklasifikasikan kebutuhan. Setiap *user story* diberi estimasi waktu untuk menentukan prioritas, estimasi waktu, dan iterasi dalam pengembangan sistem.

c. *Iteration Initialization*

Pada tahap *Iteration Initialization*, merupakan tahap awal setiap iterasi dalam pengembangan sistem. Durasi iterasi dapat berbeda tergantung pada kompleksitas proyek. Proses iterasi dimulai dengan pemilihan tugas yang telah diprioritaskan sebelumnya berdasarkan daftar prioritas. Hal ini memastikan pekerjaan yang paling penting atau mendesak dilakukan terlebih dahulu. Dengan memulai iterasi dengan tugas yang sudah diprioritaskan, pengembangan sistem difokuskan pada pekerjaan yang paling relevan dan memberikan dampak signifikan.

d. *Design*

Pada tahap *Design* (Perancangan), pengembang merancang *class diagram*, *diagram use case*, dan antarmuka sistem sesuai dengan daftar iterasi yang sedang dikerjakan. Rancangan harus memenuhi kebutuhan klien yang ditetapkan pada tahap *Requirements* sebelumnya. Desain *database* mencakup struktur entitas, hubungan, dan atribut yang digunakan. Diagram *use case* menggambarkan interaksi antara pengguna dan sistem serta fungsi sistem. Antarmuka sistem dirancang agar mudah digunakan dan sesuai kebutuhan pengguna. Tahap perancangan ini memastikan sistem memenuhi kebutuhan klien dan berfungsi sesuai harapan.

e. *Implementation*

Pada tahap *Implementation*, dilakukan pembuatan objek yang telah dirancang sebelumnya. Tahap ini terdiri dari *unit testing*, *code generation*, dan *code refactoring*. *Unit testing* dilakukan untuk menguji bagian-bagian kode atau komponen program seperti fungsi atau objek. Tujuannya adalah memastikan setiap komponen berfungsi dengan benar. Jika ada kesalahan, dilakukan perbaikan sebelum melanjutkan. Setelah lulus *unit testing*, *Code generation* dilakukan untuk setiap fitur dalam *user stories*, mengimplementasikan fungsi yang diperlukan. Jika diperlukan, dilakukan *code refactoring* untuk mengoptimalkan kode yang telah ditulis sebelumnya.

f. *System Testing*

Pada tahap System Testing, dilakukan pengujian keseluruhan sistem aplikasi yang telah dibangun. Pengujian ini meliputi fungsionalitas, keamanan, dan kinerja sistem. Terdapat dua jenis pengujian yang dilakukan, yaitu *acceptance test* dan *network performance test*. *Acceptance test* bertujuan memastikan bahwa sistem aplikasi sesuai dengan persyaratan fungsional dan non-fungsional yang telah ditetapkan pada tahap perancangan. *Network performance test* bertujuan menguji kinerja jaringan yang digunakan oleh sistem aplikasi dalam transfer data ke server melalui HTTP *connection*.

g. *Retrospective*

Tahap *Retrospective* merupakan penutup dari satu iterasi dalam pengembangan sistem. Data yang telah dikumpulkan sebelumnya dianalisis. Pada tahap ini, semua persyaratan klien terkait sistem perangkat lunak harus terpenuhi dan tidak ada cacat yang tersisa, menandakan keberhasilan proyek.

h. Pembuatan Laporan

Tahap ini merupakan tahap penutup dari penelitian, di mana laporan final disusun untuk menggambarkan hasil dari seluruh kegiatan yang dilakukan dalam proses pengembangan sistem. Laporan ini mencakup analisis, pengujian, dan kesimpulan dari penelitian yang telah dilakukan.

4. PENGUJIAN DAN PEMBAHASAN

4.1. Requirements

Berdasarkan pengumpulan data yang telah dilakukan, sistem yang akan dibuat adalah aplikasi berbasis Android. Sistem ini memungkinkan mahasiswa melakukan presensi di gawainya dengan keamanan jaringan melalui penggunaan VPN. Pengguna sistem termasuk mahasiswa yang dapat melihat daftar kehadiran, mengubah kata sandi akun, dan membuat presensi.

4.2. Planning

Pada tahap ini, terjadi kolaborasi antara klien dan pengembang untuk menghasilkan *user stories*. Metode MoSCoW digunakan untuk menentukan prioritas kebutuhan pengembangan. *User stories* lalu dikelompokkan menjadi empat kelompok berdasarkan tingkat kepentingannya.

Tabel 1. *Daftar Iterasi*

Kode *User Stories*	Deskripsi	Prioritas	*Story Points*
Iterasi 1			
US01	Melakukan *login* ke sistem	*Must have*	2
US03	Melihat profil	*Should have*	1
US04	Menyunting kata sandi	*Must have*	2
US05	Melihat daftar kehadiran	*Could have*	1
Velocity			6
Iterasi 2			
US06	Menghubungkan ke jaringan VPN	*Must have*	2
US07	Memilih server VPN	*Must have*	2
US08	Pembatasan akses halaman presensi	*Must have*	1
US09	Pembatasan akses halaman ubah kata sandi	*Must have*	1
Velocity			6
Iterasi 3			
US10	Memilih matakuliah	*Must have*	1
US11	Nama dosen otomatis terpilih	*Must have*	1
US12	Menambahkan bukti kehadiran	*Must have*	1
US13	Memotong ukuran gambar bukti kehadiran	*Must have*	1
US14	Nama dan nomor induk mahasiswa otomatis dimasukkan	*Must have*	1
US15	Memilih status kehadiran	*Must have*	1
Velocity			6

Tabel 1 menunjukkan bahwa *velocity* yang digunakan dalam setiap iterasi adalah 6, yang berarti setiap satu iterasi akan dikerjakan 6 *story points* dan membutuhkan waktu enam hari untuk menyelesaikan cerita pengguna tersebut. Dengan demikian, untuk menyelesaikan 18 cerita pengguna diperlukan pada tiga iterasi adalah 18 hari.

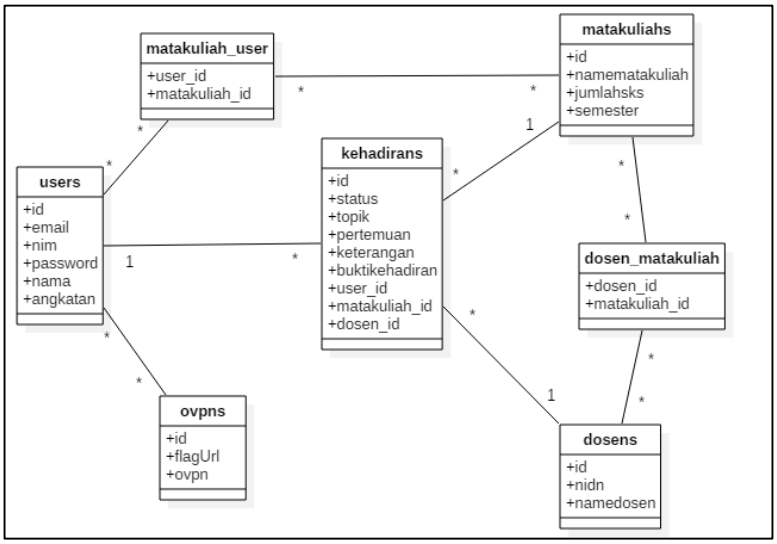
4.3. Iteration Initialization

Berdasarkan Tabel 2, inisialisasi iterasi dimulai dari iterasi pertama hingga iterasi ketiga dengan memfokuskan masing-masing *user stories* yang telah ditetapkan sebelumnya. Setiap

iterasi dikerjakan secara berurutan dan tidak boleh melompat ke iterasi lain jika belum selesai. Hal ini bertujuan agar pengembangan sistem dapat dilaksanakan secara teratur.

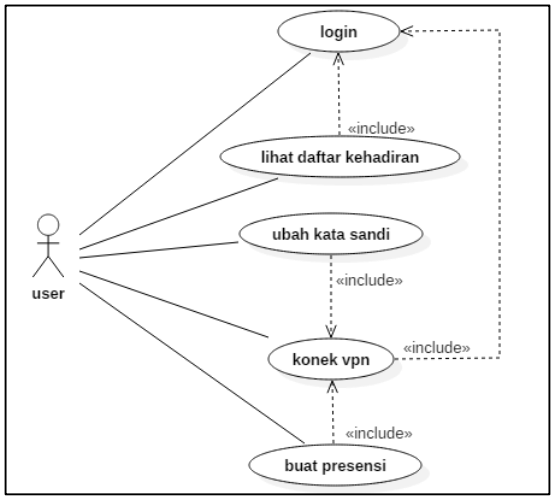
4.4. Design

Selama fase ini, dibuat *class diagram*, diagram *use case*, dan antarmuka sistem yang akan dikembangkan, dengan mempertimbangkan daftar iterasi yang sedang dikerjakan. Rancangan sistem ini harus sesuai dengan kebutuhan klien yang telah ditetapkan pada tahap *requirements*.



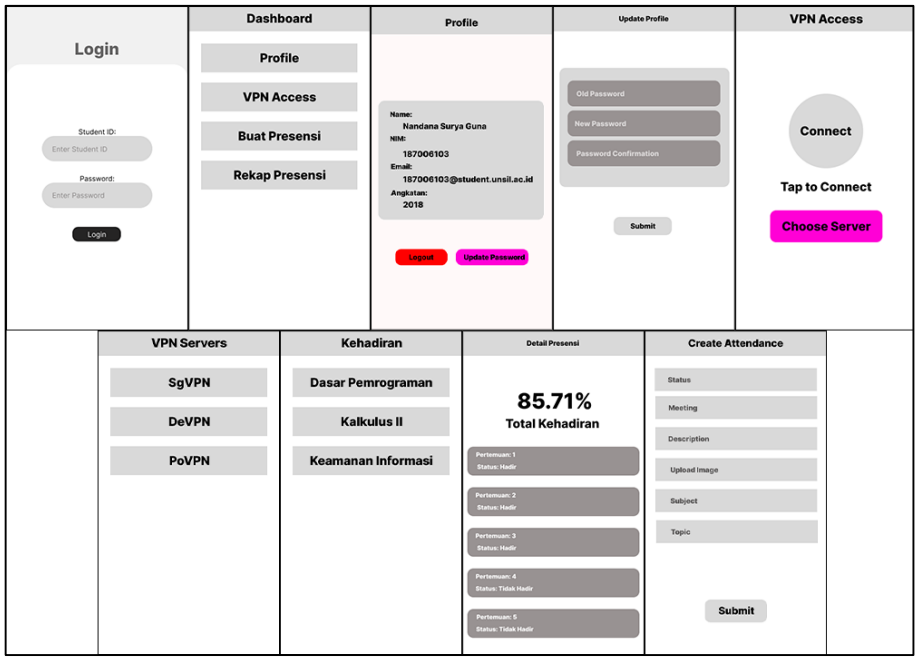
Gambar 2. *Class Diagram*

Gambar 2 menampilkan *class diagram* untuk iterasi 1 yang telah dibuat. *Class diagram* pada iterasi pertama terdiri dari tabel *ovpns*, *users*, *matakuliahs*, *dosens*, *matakuliah_user*, dan *dosen_matakuliah*. *Class diagram* ini menjadi representasi database yang digunakan pada penelitian ini.



Gambar 3. *Use Case Diagram*

Gambar 3 menunjukkan *use case diagram* bahwa seorang pengguna harus melakukan login terlebih dahulu untuk melakukan ubah kata sandi dan melihat daftar kehadiran, pengguna dapat menghubungkan perangkat mereka ke jaringan VPN setelah melakukan *login*.



Gambar 4. *Rancangan Antarmuka Aplikasi*

Gambar 4 menunjukkan rancangan antarmuka untuk tampilan aplikasi yang meliputi halaman *login*, *dashboard*, profil pengguna, halaman ubah kata sandi, halaman VPN *access* untuk menghubungkan perangkat ke jaringan VPN, halaman VPN *servers* untuk memilih server VPN, halaman kehadiran yang berisi matakuliah yang dikontrak mahasiswa, halaman detail kehadiran, dan halaman untuk membuat presensi.

4.5. Implementation

Pembuatan kode program dilakukan menggunakan bahasa pemrograman Java pada Android Studio untuk implementasi yang sesuai dengan cerita pengguna termasuk *library* OpenVPN dan Retrofit 2. Setelah pembuatan kode program selesai, dilakukan pengujian unit menggunakan JUnit 4 untuk mengetahui fungsionalitas kode program yang telah dibuat.

Tabel 2. *Hasil Unit Testing*

Units	***Status***
LoginActivityTest	*passed*
ProfileActivityTest	*passed*
UpdateProfileActivityTest	*passed*
DashboardActivityTest	*passed*
UpdateProfileActivityTest2	*passed*
PresenceActivityTest	*passed*
VPNActivityTest	*passed*
ServersActivityTest	*passed*
PresenceActivityTest2	*passed*
KehadiranActivityTest	*passed*
DetailKehadiranActivityTest	*passed*

Tabel 2 menunjukkan hasil dari *Unit Test* yang meliputi sebelas unit pengujian. Berdasarkan Tabel 2, semua *unit testing* yang dilakukan pada aplikasi telah berhasil dan dinyatakan lulus. Semua aktivitas dan fungsi pada aplikasi telah diuji dan dinyatakan berfungsi dengan baik.

4.6. System Testing

Acceptance test dilakukan untuk memastikan bahwa sistem aplikasi sesuai dengan persyaratan fungsional dan non-fungsional yang telah ditetapkan pada tahap perancangan.

Tabel 3. *Hasil Acceptance Test*

Kode *User Stories*	**Kriteria *Acceptance test***	**Hasil yang Diharapkan**	**Hasil**
US01	Pengguna dapat membuka halaman *login* dan melihat opsi untuk melakukan *login* tanpa registrasi.	Opsi untuk login tanpa registrasi ditampilkan pada halaman *login*.	Sesuai
US03	Pengguna dapat membuka halaman profil dan melihat informasi profil yang sesuai dengan akun pengguna.	Informasi profil pengguna yang aktif ditampilkan di halaman profil.	Sesuai
US04	Pengguna dapat memasukkan kata sandi lama dan kata sandi baru ke dalam formulir pengaturan kata sandi.	Pengguna dapat mengganti kata sandi dengan memasukkan kata sandi lama dan baru.	Sesuai
US05	Pengguna dapat membuka halaman daftar kehadiran berdasarkan matakuliah.	Daftar kehadiran yang tersedia ditampilkan di halaman daftar kehadiran.	Sesuai
US06	Pengguna dapat membuka halaman pengaturan VPN dan menemukan opsi untuk menghubungkan perangkat ke jaringan VPN.	Pengguna dapat dengan mudah menemukan opsi untuk menghubungkan perangkat ke jaringan VPN pada halaman pengaturan.	Sesuai
US07	Pengguna dapat membuka halaman pengaturan server VPN dan melihat daftar server yang tersedia.	Pengguna dapat dengan mudah melihat daftar server yang tersedia pada halaman pengaturan server VPN.	Sesuai
US08	Pengguna mencoba mengakses halaman presensi tanpa terhubung ke jaringan VPN.	Sistem harus menolak akses dan menampilkan pesan yang jelas kepada pengguna bahwa mereka harus terhubung ke jaringan VPN.	Sesuai
US09	Pengguna mencoba mengakses halaman ubah kata sandi tanpa terhubung ke jaringan VPN.	Sistem harus menolak akses dan menampilkan pesan yang jelas kepada pengguna.	Sesuai
US10	Pengguna dapat membuka halaman membuat presensi dan menemukan opsi untuk memilih mata kuliah.	Pengguna dapat dengan mudah menemukan opsi untuk memilih mata kuliah pada halaman membuat presensi.	Sesuai
US11	Pengguna tidak perlu memilih nama dosen pengampu karena sudah otomatis.	Nama dosen pengampu matakuliah otomatis ditambahkan pada variabel dan siap disubmit.	Sesuai
US12	Pengguna dapat membuka halaman membuat presensi dan menemukan opsi untuk menambahkan bukti kehadiran.	Pengguna dapat dengan mudah menemukan opsi untuk menambahkan bukti kehadiran pada halaman membuat presensi.	Sesuai
US13	Pengguna dapat membuka halaman membuat presensi dan menambahkan bukti kehadiran.	Pengguna dapat dengan mudah menambahkan bukti kehadiran pada halaman membuat presensi.	Sesuai
US14	Pengguna dapat membuka halaman membuat presensi dan membuatnya tanpa memasukkan nama dan nomor induk mahasiswa.	Sistem secara otomatis akan mengirimkan nama dan nomor induk mahasiswa saat pengguna melakukan *submit* data.	Sesuai
US15	Pengguna dapat membuka halaman membuat presensi dan menemukan opsi untuk memilih status kehadiran.	Pengguna dapat dengan mudah menemukan opsi untuk memilih status kehadiran pada halaman membuat presensi.	Sesuai

Tabel 3 menunjukkan hasil dari *acceptance test* implementasi pada *user stories* selesai dan telah sesuai dengan kebutuhan dari klien. Selanjutnya, pengujian performa jaringan dilakukan pada beberapa jaringan yaitu *Default Network* dari penyedia layanan internet Indonesia dan tiga server VPN. *Network performance test* dilakukan sebanyak seratus kali di setiap jaringan untuk menguji performa pertukaran data oleh aplikasi ke server.

Tabel 3. *Hasil Acceptance Test*

Kode Server VPN	*vCores*	*Memory*	*Storage*	*Bandwidth*
PoVPN	1	2 GB	20 GB	250 Mbps
DeVPN	1	2 GB	20 GB	250 Mbps
SgVPN	2	4 GB	40 GB	250 Mbps

Tabel 3 menunjukkan spesifikasi dari tiga server VPN yang akan dipakai untuk pengujian performa jaringan. Lokasi dari setiap server VPN terletak di negara yang berbeda. PoVPN berlokasi di Polandia, DeVPN berlokasi di Jerman, dan SgVPN berlokasi di Singapura.

Tabel 4. *Hasil Network Performance Test*

No	*Network*	*Requests*	*Size (Byte)*	*Time (ms)*	Status	Method
1	*Default*	*login*	393	862	200	POST
		ovpns	15700	415	403	GET
		dosens	1500	531	200	GET
		matakuliahs	5800	484	200	GET
		kehadirans	6500	559	403	POST
		kehadirans	1300	1187	200	GET
		change-password	6500	497	403	POST
		logout	0	342	204	POST
		matakuliahuser	178	1006	200	GET
		matakuliahdosen	183	376	200	GET
2	SgVPN	*ovpns*	15700	895	200	GET
		dosens	1500	1273	200	GET
		matakuliahs	5800	923	200	GET
		kehadirans	6500	955	200	POST
		kehadirans	1300	1727	200	GET
		change-password	6500	1088	200	POST
		logout	0	556	204	POST
		matakuliahuser	178	1304	200	GET
		matakuliahdosen	183	1288	200	GET
3	PoVPN	*ovpns*	15700	1075	200	GET
		dosens	1500	1187	200	GET
		matakuliahs	5800	1324	200	GET
		kehadirans	6500	1223	200	POST
		kehadirans	1300	1826	200	GET
		change-password	6500	1172	200	POST
		logout	0	672	204	POST
		matakuliahuser	178	1421	200	GET
		matakuliahdosen	183	1321	200	GET
4	DeVPN	*ovpns*	15700	1107	200	GET
		dosens	1500	1231	200	GET
		matakuliahs	5800	1314	200	GET
		kehadirans	6500	1351	200	POST
		kehadirans	1300	1793	200	GET
		change-password	6500	1264	200	POST
		logout	0	663	204	POST
		matakuliahuser	178	1324	200	GET
		matakuliahdosen	183	1372	200	GET

Tabel 4 menunjukkan hasil dari pengujian performa jaringan untuk *Default Network*, PoVPN, DeVPN, dan SgVPN. Sistem aplikasi telah mampu melakukan transfer data ke server melalui HTTP *connection* dengan baik pada jaringan *default* dari penyedia layanan internet dan beberapa server VPN. Semua method yang diuji pada jaringan *default* dan VPN menghasilkan status yang diharapkan (200 atau 204) kecuali pada request *kehadirans* POST dan *change-password* POST yang menghasilkan status 403 pada jaringan default karena *web service* membatasi alamat IP.

4.7. Retrospective

Setelah *system testing* selesai dan sesuai, langkah selanjutnya adalah tahap retrospektif. Pada fase ini, analisis kecocokan estimasi waktu dilakukan untuk mengimplementasikan cerita pengguna pada setiap iterasi.

Tabel 5. *Verifikasi Estimasi Waktu Pengerjaan*

Kode *User Stories*	**Prioritas**	***Story Points***	**Estimasi**	**Realisasi**
US01	*Must have*	1	1	2
US03	*Should have*	1	1	1
US04	*Must have*	2	2	1
US05	*Could have*	1	1	1
US06	*Must have*	2	2	2
US07	*Must have*	2	2	2
US08	*Must have*	1	1	1
US09	*Must have*	1	1	1
US10	*Must have*	1	1	1
US11	*Must have*	1	1	1
US12	*Must have*	1	1	1
US13	*Must have*	1	1	1
US14	*Must have*	1	1	1
US15	*Must have*	1	1	1
Velocity		18	18	18

Tabel 5 menujukkan bahwa estimasi waktu yang diberikan sebagian besar cukup akurat dengan realisasi waktu yang tidak jauh berbeda. Selain itu, dapat dilihat bahwa total *story points* yang direalisasikan adalah 18, yang sesuai dengan *velocity* (kecepatan) pengembangan pada periode tersebut. Ini menunjukkan bahwa pengembangan telah berhasil menyelesaikan semua *User Stories* dengan Prioritas *Must have* dan sebagian besar Prioritas *Should have* dan *Could have* pada periode tersebut.

5. KESIMPULAN

Dalam penelitian ini, berhasil dikembangkan aplikasi VPN *client* berbasis *mobile* dengan teknik JSON *parsing* untuk mengirim dan menerima data melalui jaringan VPN. Pengujian performa jaringan menunjukkan koneksi yang berhasil dengan berbagai jaringan VPN. Hasil pengujian menunjukkan jaringan *Default* memiliki waktu *respons* terbaik, sementara jaringan lain menunjukkan waktu *respons* yang sedikit lebih lambat. Aplikasi ini juga sesuai dengan persyaratan *user stories* dan memberikan manfaat dalam kemudahan akses dan keamanan. Penelitian ini berpotensi memberikan kontribusi dalam pengembangan aplikasi VPN *client mobile* yang efektif dan efisien melalui teknik JSON *parsing*. Penelitian selanjutnya dapat menambah fitur untuk meningkatkan fungsionalitas dan kinerja aplikasi.

DAFTAR PUSTAKA

[1] A. A. Setya and A. Sudaryanto, ‘Analisa Konektivitas Jaringan IPSEC Dan OpenVPN Pada Jaringan Berbasis IP Dinamis’, *Informatics, Electrical and Electronics Engineering (Infotron)*, vol. 1, no. 2, 2021, doi: 10.33474/infotron.v1i2.11422.

[2] P. E. Kristianto and A. T. Putra, 'Comparative Analysis of IPv4 and IPv6 OpenVPN Protocol Performance Based on QoS Parameters', *Journal of Advances in Information Systems and Technology*, vol. 3, no. 1, 2021, doi: 10.15294/jaist.v3i1.49095.

[3] H. Purwoko, 'Pemanfaatan Middleware Layer untuk Multi-Platform pada Mobile dan Desktop', *STRING (Satuan Tulisan Riset dan Inovasi Teknologi)*, vol. 4, no. 2, 2019, doi: 10.30998/string.v4i2.4260.

[4] A. Triawan and M. A. Prasetiyo, 'Penerapan Web Service (XML dan JSON) Untuk Meningkatkan Performance Pada Informasi Pembayaran Uang Kuliah', *Teknois : Jurnal Ilmiah Teknologi Informasi dan Sains*, vol. 8, no. 1, 2019, doi: 10.36350/jbs.v8i1.22.

[5] D. Sahlinal, R. Maulini, and D. K. Widyawati, 'Sistem Informasi Pemasaran Hasil Pertanian Polinela Berbasis JSON', *SIMADA (Jurnal Sistem Informasi & Manajemen Basis Data)*, vol. 1, no. 1, 2018, doi: 10.30873/simada.v1i1.1109.

[6] R. Sahrial, D. F. Fauzi, and E. Susilawati, 'Pemanfaatan Json Untuk Menampilkan Data Realtime Covid-19 Dengan Model View Presenter', *Jurnal Teknoinfo*, vol. 16, no. 1, 2022, doi: 10.33365/jti.v16i1.780.

[7] R. Rustam and A. Y. F. Pangestu, 'Aplikasi Kamus Bahasa Jepang Berbasis Mobile Android', *Jurnal Informasi dan Komputer*, vol. 8, no. 1, 2020, doi: 10.35959/jik.v8i1.178.

[8] D. Rajaratnam and F. Sunmola, 'Adaptations in scor based performance metrics of airline catering supply chain during covid-19 pandemic', *Journal of Industrial Engineering and Management*, vol. 14, no. 4, 2021, doi: 10.3926/jiem.3592.

[9] P. R. Burgess and F. T. Sunmola, 'Prioritising Requirements of Informational Short Food Supply Chain Platforms Using A Fuzzy Approach', in *Procedia Computer Science*, 2021. doi: 10.1016/j.procs.2021.01.335.

[10] F. Mahmud, L. N. Amali, Moh. R. A. Kaluku, and M. Latief, 'Pengembangan Sistem Informasi Penjualan Berbasis Android Memanfaatkan Layanan Web Service', *Jambura Journal of Informatics*, vol. 4, no. 1, 2022, doi: 10.37905/jji.v4i1.12367.

[11] S. Maisaroh, O. Fajarianto, and M. Nasir, 'Sistem Informasi Lowongan Kerja Kota Tangerang Berbasis Android dan Web Service', *JURNAL SISFOTEK GLOBAL*, vol. 9, no. 1, 2019, doi: 10.38101/sisfotek.v9i1.222.

[12] A. T. S. Christanto and R. Kurniawati, 'Penerapan Service Oriented Architecture Menggunakan Web Service Pada Aplikasi Perpustakaan Berbasis Android', *Jurnal Buana Informatika*, vol. 7, no. 1, 2016, doi: 10.24002/jbi.v7i1.486.

[13] B. D. Wijaya, F. E. M. A., and A. Fiade, 'Implementasi JSON Parsing Pada Aplikasi Mobile E-commerce Studi Kasus : CV V3 Tekno Indonesia', *Jurnal Pseudocode*, vol. 2, no. 1, pp. 1–9, Jun. 2015.

[14] Ester Lumba, 'Pertukaran Data Pada Aplikasi Android Menggunakan Java Script Object Notation (JSON) Dan Rest Api Dengan Retrofit 2', in *Prosiding Seminar Nasional Aplikasi Sains & Teknologi (SNAST)*, Yogyakarta: INSTITUT SAINS & TEKNOLOGI AKPRIND YOGYAKARTA, Mar. 2021, pp. 118–127.

[15] G. P. Tiwary, E. Stroulia, and A. Srivastava, 'Compression of XML and JSON API Responses', *IEEE Access*, vol. 9, 2021, doi: 10.1109/ACCESS.2021.3073041.

[16] A. P. Pratama and M. Kamisutara, 'Pengembangan Sistem Informasi Akademik Berbasis Mobile Menggunakan Flutter Di Universitas Narotama Surabaya', *Network Engineering Research Operation*, vol. 6, no. 2, 2021, doi: 10.21107/nero.v6i2.238.

[17] G. I. Marthasari and N. Hayatin, 'Evaluasi Heuristik Website berbasis Framework Sirius dengan Pengaturan Prioritas menggunakan Teknik Moscow', *Jurnal Teknologi Informasi dan Ilmu Komputer*, vol. 7, no. 2, 2020, doi: 10.25126/jtiik.2020701662.

[18] H. I. Hasan, 'Implementasi Metode Personal Extreme Programming Dalam Pengembangan Sistem Administrasi Pelayanan Desa (Studi Kasus: Desa Bulangan Barat Kec. Pegantenan Kab. Pamekasan)', *Jurnal Repositor*, vol. 3, no. 1, 2021, doi: 10.22219/repositor.v3i1.1224.