BAB I PENDAHULUAN

1.1 Latar Belakang

Telkom University, sebagai institusi pendidikan yang berfokus pada teknologi informasi, memiliki Open Library sebagai unit akademik vital. Saat ini, infrastruktur digital Open Library dibangun menggunakan *framework* CodeIgniter 3. Menurut situs web resminya, *framework* ini merupakan versi lawas (*legacy*) yang hanya menerima pembaruan keamanan dan tidak lagi menjadi fokus pengembangan utama, sehingga menimbulkan risiko keamanan dan menciptakan kondisi di mana setiap perbaikan di masa depan menjadi semakin sulit dan memakan waktu[1], [2].

Seiring perkembangan teknologi, sistem Open Library yang ada menghadapi tantangan signifikan yang berdampak langsung pada pengalaman pengguna dan efisiensi pengembang. Secara nyata, salah satu masalah utama yang dirasakan adalah kinerja sistem yang lambat dan tidak responsif, terutama saat mengakses menu-menu yang membutuhkan pemrosesan data berat. Sebagai contoh, untuk mengakses halaman statistik katalog pada sistem lama, pengguna harus menunggu dengan waktu respons rata-rata 8.29 detik bahkan pada beban penggunaan yang rendah. Kondisi ini semakin buruk saat beban pengguna meningkat, di mana sistem tidak hanya menjadi lebih lambat, tetapi juga mengalami kegagalan sistemik dan tidak dapat diakses. Masalah kinerja yang krusial ini berakar pada beberapa isu teknis yang lebih dalam.

Arsitektur sistem yang mengadopsi pendekatan *microservice* dengan tujuh *website* terpisah, meskipun secara teori memiliki keunggulan, dalam praktiknya justru membawa tantangan berupa kerumitan dalam pengelolaan sistem sehari-hari. Mengelola banyak layanan terpisah memerlukan upaya lebih dalam hal *deployment* dan koordinasi, yang bisa menjadi tidak efisien untuk tim dengan sumber daya terbatas [3]. Lebih jauh, kode pada sistem lama memiliki masalah arsitektur, seperti penggunaan *Fat Controller* di mana satu kelas menangani terlalu banyak tanggung jawab. Praktik ini, yang bertentangan dengan prinsip *Single Responsibility* yang ditekankan oleh Martin (2009), secara luas diakui sebagai penghambat utama kemudahan pemeliharaan (*maintainability*) sistem [4].

Untuk mengatasi permasalahan tersebut, solusi yang diusulkan adalah migrasi layanan OLAFA dan Pengadaan ke *framework* Laravel. Laravel dipilih karena merupakan *framework* PHP modern yang secara inheren mendorong penerapan praktik terbaik untuk menghasilkan arsitektur yang bersih dan terstruktur. Salah satu praktik yang didukung adalah pemisahan tanggung jawab atau *Separation of Concerns (SoC)*, yang dapat dicapai melalui implementasi pola desain seperti

Service *Repository Pattern*. Penggunaan *Repository Pattern*, seperti yang dijelaskan oleh Kılıçdağı dan Yılmaz, adalah sebuah pendekatan populer yang digunakan pengembang Laravel untuk memisahkan logika bisnis dari urusan teknis database[5]. Pendekatan ini juga diakui sebagai *best-practice* dalam pengembangan aplikasi Laravel modern, sebagaimana dibahas dalam studi oleh Brekalo dan Sedlarević (2024) [6].

Proses migrasi ini tidak hanya melibatkan perpindahan platform, tetapi juga penulisan ulang kode (*refactoring*) untuk mengimplementasikan prinsip-prinsip kualitas kode tersebut, yang bertujuan untuk menghasilkan arsitektur yang lebih terstruktur dan mudah dikelola (*maintainable*).

Dengan migrasi ini, diharapkan sistem baru tidak hanya lebih mudah dipelihara secara arsitektural, tetapi juga memiliki kinerja yang lebih unggul. Keberhasilan migrasi kedua layanan ini diharapkan dapat menjadi studi kasus dan fondasi yang kuat untuk pembaruan layanan-layanan lain di lingkungan Open Library, sejalan dengan visi Telkom University sebagai institusi pendidikan berbasis teknologi informasi terdepan.

1.2 Rumusan Masalah dan Solusi

Berdasarkan latar belakang, sistem Open Library yang ada saat ini menghadapi permasalahan fundamental yang saling terkait, yaitu teknologi yang tertinggal, arsitektur yang tidak efisien, dan kualitas kode yang rendah. Permasalahan ini secara kolektif berdampak pada kinerja, stabilitas, dan kemudahan pemeliharaan sistem. Oleh karena itu, diajukan pertanyaan penelitian utama sebagai berikut.

“Bagaimana migrasi layanan OLAFA dan Pengadaan ke *framework* Laravel dengan implementasi prinsip kualitas kode dapat menghasilkan sistem layanan yang tidak hanya lebih mudah dipelihara secara arsitektural, tetapi juga terbukti lebih efisien stabil dan tangguh di bawah beban kerja berdasarkan pengujian kinerja kuantitatif? “

Untuk menjawab pertanyaan tersebut, solusi yang diusulkan adalah melakukan migrasi dan pembaruan arsitektur secara menyeluruh pada layanan OLAFA dan Pengadaan. Proses ini tidak hanya melibatkan perpindahan platform dari CodeIgniter 3 ke Laravel, tetapi juga penulisan ulang kode (*refactoring*) dengan menerapkan prinsip-prinsip kualitas kode modern seperti *Separation of Concerns (SoC)* dan *Clean Code*. Keberhasilan dari solusi ini akan divalidasi secara kuantitatif melalui serangkaian pengujian kinerja (*load testing* dan *stress testing*) untuk mengukur secara objektif peningkatan efisiensi dan ketahanan sistem baru. Pendekatan ini dipilih untuk memastikan bahwa solusi yang dihasilkan tidak hanya fungsional, tetapi juga andal dan mudah dikelola untuk jangka panjang.

1.3 Tujuan

Berdasarkan rumusan masalah yang telah dipaparkan, penelitian ini dilakukan dengan alasan utama untuk memodernisasi layanan OLAFA dan Pengadaan di Open Library yang menghadapi tantangan teknis. Tujuan utamanya adalah untuk menghasilkan sistem layanan baru yang secara kualitatif lebih mudah dipelihara dan secara kuantitatif terbukti lebih efisien serta tangguh. Untuk mencapai hal tersebut, berikut adalah rincian jenis pekerjaan dan sasaran terukur yang hendak dicapai sebagai solusi untuk menjawab rumusan masalah:

Tujuan Umum

Secara umum, tujuan dari Tugas Akhir ini adalah melaksanakan dan menganalisis proses migrasi layanan OLAFA dan Pengadaan dari *framework* CodeIgniter 3 ke Laravel. Proses ini berfokus pada implementasi prinsip-prinsip kualitas kode untuk membangun sistem layanan baru yang secara arsitektural lebih mudah dikelola dan secara kuantitatif terbukti lebih efisien serta tangguh di bawah beban kerja, yang divalidasi melalui pengujian kinerja.

Tujuan Khusus

Untuk mencapai tujuan umum tersebut, penelitian ini memiliki beberapa tujuan khusus yang difokuskan pada layanan OLAFA dan Pengadaan:

1. **Melakukan Migrasi Fungsional:** Memindahkan fungsionalitas inti dari layanan OLAFA dan Pengadaan dari sistem berbasis CodeIgniter 3 ke dalam lingkungan pengembangan *framework* Laravel.
2. **Mengimplementasikan Arsitektur Kode Berkualitas:** Menerapkan prinsip *Separation of Concerns (SoC)* melalui pola desain *Service Repository Pattern* serta prinsip-prinsip *Clean Code* (PSR) dalam proses penulisan ulang kode kedua layanan untuk meningkatkan kualitas internal kode.
3. **Menganalisis Peningkatan *Maintainability*:** Secara kualitatif, membandingkan arsitektur kode sistem lama dan baru untuk menunjukkan bahwa penerapan prinsip kualitas kode menghasilkan sistem yang lebih terstruktur dan mudah dipelihara.
4. **Menguji Efisiensi Sistem:** Secara kuantitatif, melakukan Uji Beban (*Load Testing*) untuk membandingkan kecepatan kedua sistem. Tujuannya adalah untuk membuktikan bahwa sistem baru lebih efisien dalam menangani permintaan data yang berat, yang diukur melalui waktu respons (*response time*) yang lebih rendah dan kapasitas pemrosesan (*throughput*) yang lebih tinggi

5. **Menguji Ketahanan Sistem:** Secara kuantitatif, melakukan Uji Ketahanan (*Stress Testing*) untuk melihat seberapa tangguh kedua sistem saat diberi beban kerja yang tinggi. Tujuannya adalah untuk membuktikan bahwa arsitektur baru lebih stabil dan tidak mudah gagal, yang diukur dengan membandingkan tingkat kestabilan dan jumlah eror (*error rate*) di bawah tekanan.

1.4 Batasan Masalah

Dalam rangka menjaga fokus penelitian dan pengembangan agar tetap terarah dan dapat diselesaikan dalam kerangka waktu yang ditentukan, beberapa batasan diterapkan:

1. **Tidak Mencakup Penambahan Fitur Baru atau Desain Ulang UI/UX Mayor:** Pembaruan ini tidak mencakup pengembangan fungsionalitas atau fitur baru yang sebelumnya tidak ada di sistem lama. Selain itu, tidak dilakukan desain ulang antarmuka pengguna (UI/UX) secara mayor, perubahan pada tampilan antarmuka bersifat konsekuensial dari penggunaan *framework* Laravel dan penerapan struktur kode yang baru.

2. **Pembatasan Lingkup pada Versi *Website*:** Proyek pembaruan sistem ini hanya dilakukan pada versi website dari layanan Open Library. Aplikasi *mobile* TelU Openlib tidak termasuk dalam cakupan proyek ini dan tetap menggunakan sistem lama yang ada. Dengan demikian, penelitian ini tidak membahas atau mengevaluasi aspek-aspek terkait aplikasi *mobile*.

3. **Lingkup Pengujian Kinerja**: Pengujian kinerja untuk membandingkan sistem lama dan sistem baru menggunakan metode *Load Testing* dengan *tool* k6 tidak dilakukan pada keseluruhan fungsionalitas sistem. Untuk memastikan analisis yang mendalam dan relevan, pengujian kinerja dibatasi secara spesifik pada dua menu representatif yang dipilih berdasarkan perbedaan fundamental beban kerja, dan kompleksitas teknis:

    - Layanan OLAFA: Menu Katalog > Data Statistik. Menu ini dipilih karena mewakili tugas yang sangat berat dalam hal membaca dan mengolah data (*read-heavy*). Untuk menampilkan satu halaman statistik, sistem lama harus membaca, menggabungkan, dan menghitung data dari berbagai tabel sekaligus, yang membuatnya menjadi sangat lambat (*bottleneck*). Pengujian pada menu ini bertujuan untuk membandingkan seberapa efisien arsitektur baru dalam menangani permintaan data yang intensif ini.

    - Layanan Pengadaan: Menu Pengajuan Pengadaan oleh Dosen/Pegawai. Skenario pengujian untuk layanan ini terbatas pada proses mengakses halaman tanpa melakukan transaksi penyimpanan data. Meskipun

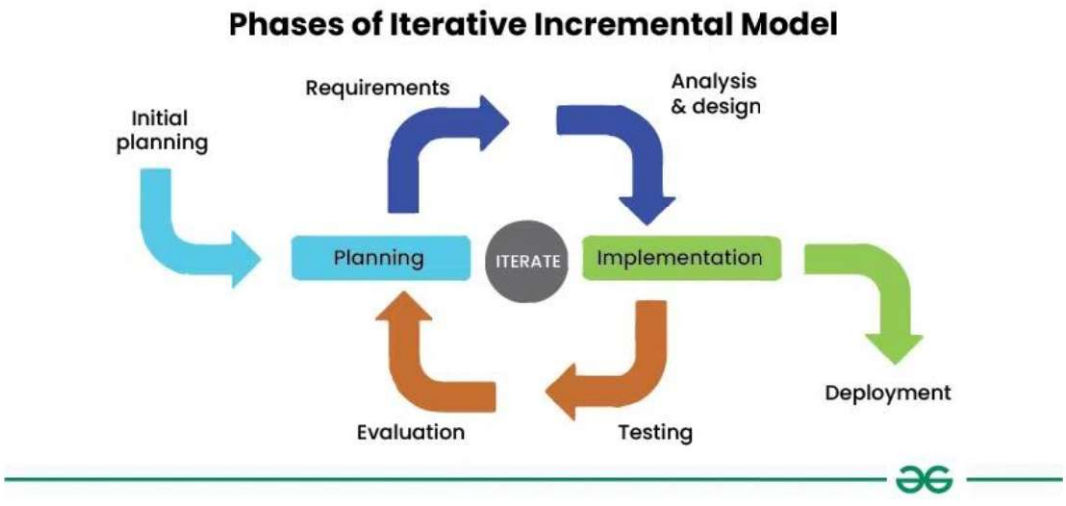
merupakan proses *read* yang ringan, pengujian dilakukan dengan skenario *Stress Testing* untuk mengukur ketahanan dan titik kegagalan masing-masing arsitektur di bawah tekanan beban pengguna yang tinggi.

Batasan-batasan ini ditetapkan untuk memastikan bahwa proyek Tugas Akhir memiliki ruang lingkup yang jelas dan terkelola, memungkinkan tercapainya tujuan migrasi pada layanan yang dipilih dan evaluasi awal terhadap efektivitas pendekatan yang diterapkan.

1.5 Metodologi

Dalam pelaksanaan proyek ini, metodologi pengembangan sistem yang diterapkan adalah Pengembangan Iteratif dan Inkremental (IID). Pendekatan ini dipilih karena paling akurat merefleksikan alur kerja nyata dalam proyek migrasi sistem warisan (*legacy system*) dan penulisan ulang kode. Berbeda dengan model *Waterfall* yang bersifat kaku dan sekuensial, menurut Larman dan Basili (2003), IID adalah praktik rekayasa perangkat lunak yang telah teruji secara historis dengan tujuan utama untuk menghindari proses satu arah yang digerakkan oleh dokumen dan tidak mampu merespons perubahan secara efisien [7].

Relevansi dan kekuatan pendekatan ini dalam menangani sistem yang kompleks juga didukung oleh penelitian terkini. Sebuah studi oleh Islam dan Ferworn (2020) menemukan bahwa metodologi *Agile*, yang berlandaskan pada prinsip IID, terbukti lebih adaptif dalam proyek-proyek yang besar dan kompleks. Hal ini sejalan dengan proses migrasi layanan OLAFA dan Pengadaan yang tercermin dalam dua aspek utama[8].



Gambar 1. 1 Alur Kerja Pengembangan Iteratif [9]

Pertama, pengembangan dilakukan secara inkremental, di mana fungsionalitas sistem dibangun bagian per bagian, seperti yang tercermin pada detail penjadwalan

kerja. Kedua, proses pengembangan bersifat iteratif, di mana setiap *increment* fungsionalitas disempurnakan melalui siklus pengembangan yang terstruktur, seperti yang divisualisasikan pada Gambar 1.1 Alur kerja pada setiap siklusnya adalah sebagai berikut:

1. **Perencanaan Awal** (*Initial Planning*): Tahap ini dilakukan di awal setiap *increment* besar untuk mendefinisikan lingkup dan tujuan utama dari fungsionalitas yang akan dikerjakan.
2. **Perencanaan dan Analisis Kebutuhan** (*Planning & Requirement*): Berdasarkan perencanaan awal, dilakukan analisis yang lebih detail terhadap kebutuhan teknis dan fungsional dari fitur yang akan dibangun pada iterasi tersebut.
3. **Analisis dan Desain** (*Analysis & Design*): Pada tahap ini, dirancang arsitektur dan komponen spesifik untuk fitur tersebut di dalam *framework* Laravel, termasuk struktur *Controller*, *Service*, dan *Repository*.
4. **Implementasi** (*Implementation*): Setelah desain disetujui, proses penulisan ulang kode dari sistem lama atau pembuatan kode baru dilakukan sesuai dengan rancangan.
5. **Pengujian** (*Testing*): Hasil implementasi langsung diuji untuk memverifikasi fungsionalitas dan kualitasnya. Dalam proyek ini, pengujian yang dilakukan adalah *white-box testing* untuk memastikan kebenaran logika internal.
6. **Evaluasi** (*Evaluation*): Hasil dari pengujian menjadi masukan untuk tahap evaluasi. Jika fitur belum memenuhi standar atau masih memerlukan perbaikan, siklus akan berputar kembali ke tahap Perencanaan dan Analisis Kebutuhan untuk memulai iterasi berikutnya.
7. **Penerapan** (*Deployment*): Jika hasil evaluasi sudah sesuai dengan yang diharapkan, maka *increment* tersebut dianggap selesai dan siap untuk diintegrasikan ke dalam sistem utama.

Penggabungan kedua strategi ini, sebagaimana dijelaskan oleh Kirpitsas dan Pachidis (2022), memberikan manfaat berupa pengiriman nilai bisnis secara berkelanjutan dan memungkinkan adanya penilaian ulang prioritas secara berkala melalui umpan balik yang cepat pada setiap siklusnya [10]. Dengan demikian, metodologi ini memberikan kerangka kerja yang terstruktur namun tetap adaptif untuk mengelola kompleksitas teknis dan memastikan kualitas dalam proyek migrasi ini.

1.6 Penjadwalan Kerja

Pelaksanaan proyek migrasi layanan OLAFA dan Pengadaan ini direncanakan berlangsung selama 39 minggu, dimulai dari 17 September 2024 hingga 17 Juni 2025. Proses pengerjaan dibagi menjadi empat fase utama yang selaras dengan Metodologi Pengembangan Iteratif dan Inkremental. Setiap fase memiliki *milestone* (titik capaian penting) yang jelas untuk memastikan proyek berjalan secara

terstruktur dan progresnya dapat dipantau dengan efektif. Rincian jadwal pengerjaan proyek dapat dilihat pada Tabel 1. 1

Tabel 1. 1 Jadwal Pengerjaan Proyek

Fase	Kegiatan Utama	Alokasi Waktu	*Milestone* (Titik Capaian)
Fase 1: Inisiasi dan Perencanaan Awal	1. Persiapan lingkungan pengembangan.	Minggu 1 - 5	Lingkungan pengembangan siap digunakan. Arsitektur dasar sistem baru di Laravel telah terbentuk dan divalidasi.
	2. Analisis arsitektur dan kode sistem lama (CodeIgniter 3) untuk layanan OLAFA & Pengadaan.		
	3. Desain arsitektur dasar dan struktur modular untuk aplikasi baru di Laravel.		
Fase 2: Iterasi Pengembangan Layanan OLAFA	1. Migrasi dan implementasi fitur-fitur inti layanan OLAFA secara inkremental.	Minggu 6 - 24	Seluruh fungsionalitas utama layanan OLAFA berhasil dimigrasi ke Laravel dan lulus pengujian internal.
	2. Siklus iteratif implementasi dan pengujian internal (*white-box*) untuk setiap fitur.		
	3. *Refactoring* dan penyempurnaan kode layanan OLAFA berdasarkan hasil pengujian.		
Fase 3: Iterasi Pengembangan Layanan Pengadaan	1. Migrasi dan implementasi fitur-fitur inti layanan Pengadaan secara inkremental.	Minggu 25 - 34	Seluruh fungsionalitas utama layanan Pengadaan berhasil dimigrasi ke Laravel dan lulus pengujian internal.
	2. Siklus iteratif implementasi dan pengujian internal untuk setiap fitur.		

Fase	Kegiatan Utama	Alokasi Waktu	*Milestone* (Titik Capaian)
	3. Finalisasi dan optimasi kode layanan Pengadaan.		
Fase 4: Pengujian Sistem	1. Pengujian integrasi antara layanan OLAFA dan Pengadaan.	Minggu 35 - 39	Hasil pengujian kinerja diperoleh dan dianalisis.
	2. Pelaksanaan pengujian kinerja sistem (*Load Testing*) menggunakan k6.		
	3. Perbaikan *bug* final berdasarkan hasil pengujian.		