

BAB I

PENDAHULUAN

A. Latar Belakang

Pada era digital saat ini, pendekatan manajemen proyek teknologi informasi (TI) semakin beragam seiring berkembangnya teknologi. Salah satu pendekatan yang banyak diadopsi dalam mengelola proyek TI adalah *Scrum*. Menurut laporan “14th Annual State of Agile Report” tahun 2020, sekitar 58% (23.200 dari 40.000) praktisi, pimpinan dan perusahaan di seluruh dunia menggunakan *Scrum* sebagai kerangka kerja[1]. Meskipun adopsinya luas, implementasi *Scrum* masih menghadapi tantangan serius di lapangan. Studi “*Issues and Challenges in Scrum Implementation*” yang dilakukan pada perusahaan Digital Prodigy Limited dan Bentley Systems menemukan permasalahan dalam praktik *Scrum* yaitu alat manajemen proyek *Scrum* yang tersedia di pasar cenderung terlalu kompleks [2].

Studi tersebut mengidentifikasi bahwa permasalahan yang diakibatkan oleh kompleksnya alat manajemen proyek *Scrum* menyebabkan praktisi *Scrum* kesulitan untuk menggunakan alat tersebut. Dari survei yang dilakukan oleh Selectifyr, sebuah perusahaan yang bergerak di bidang teknologi manajemen proyek dan penyedia ulasan alat kolaboratif, ditemukan bahwa 60% dari lebih dari 500 responden menyatakan bahwa alat manajemen proyek khususnya Jira, terasa kompleks dan membutuhkan waktu adaptasi yang cukup panjang [3]. Hal ini sejalan dengan temuan Özkan dan Mishra (2019) yang menyatakan bahwa meskipun alat manajemen proyek yang kompleks dapat memenuhi seluruh kebutuhan praktisi *Scrum*, kompleksitas tersebut justru menurunkan tingkat kemudahan dan kegunaan (*usability*) dari alat tersebut [4].

Kompleksitas ini umumnya disebabkan oleh banyaknya fitur yang terintegrasi dalam satu sistem. Fitur-fitur seperti autentikasi pengguna, manajemen proyek, pengelolaan *backlog* dan *sprint*, hingga pengiriman notifikasi *email* yang dibungkus dalam satu basis kode akan menyebabkan ketergantungan antar layanan. Ketergantungan antar layanan sering kali menimbulkan konflik sistem secara keseluruhan. Misalnya layanan autentikasi dan layanan email dibungkus dalam satu

basis kode, jika layanan email mati, maka layanan autentikasi juga tidak dapat berfungsi. Untuk mengatasi permasalahan tersebut, sistem perlu dibangun dengan pendekatan arsitektur yang memungkinkan layanan bekerja secara lebih mandiri. Studi yang dilakukan oleh Gundelsby dan Stray (2018) pada proyek *agile* dengan sembilan tim menunjukkan bahwa masalah ketergantungan ini dapat diminimalkan dengan memecah sistem menjadi layanan-layanan kecil yang terpisah satu sama lain. Pendekatan ini terbukti membuat tiap layanan mampu bekerja secara mandiri tanpa terlalu bergantung pada layanan lainnya[5].

Pendekatan tersebut sejalan dengan prinsip arsitektur *microservices*, di mana sistem dipecah menjadi layanan-layanan kecil yang independen. Dengan *microservices*, setiap layanan bertanggung jawab atas tugasnya masing-masing dan meminimalisir ketergantungan pada layanan lainnya. Collina et al. dalam penelitian yang berjudul “*Evaluating the Risk of Changes in a Microservice Architecture*” mencatat bahwa salah satu keuntungan utama *microservices* adalah kemampuannya untuk memungkinkan layanan bekerja secara mandiri, sehingga secara signifikan mengurangi ketergantungan antar layanan. Studi tersebut menemukan bahwa organisasi yang beralih ke *microservices* mampu mempercepat siklus pengembangan dan menurunkan konflik antar layanan, karena setiap layanan dikelola dalam basis kode tersendiri ketimbang berbagi satu basis kode[6].

Berdasarkan survei internal yang dilakukan terhadap praktisi Scrum di PT. XYZ, permasalahan serupa juga ditemukan. Dari 46 responden yang terdiri 12 Product Owner, 11 Scrum Master, dan 23 Development Team, 76% menyatakan bahwa alat manajemen proyek yang digunakan saat ini terlalu rumit. Temuan ini menunjukkan bahwa permasalahan yang diidentifikasi pada penelitian sebelumnya juga terjadi di PT. XYZ, sehingga diperlukan solusi yang lebih sederhana dan sesuai dengan kebutuhan PT. XYZ. Untuk mendukung layanan yang memiliki tingkat ketergantungan rendah, sistem dirancang dengan menerapkan pendekatan arsitektur *microservices*. Sistem ini diharapkan dapat menjadi alat manajemen proyek Scrum berbasis web yang mudah digunakan oleh praktisi Scrum.

B. Rumusan Masalah

Bagaimana cara merancang dan membangun sistem manajemen proyek teknologi informasi berbasis web dengan arsitektur *microservices* agar dapat digunakan dengan mudah oleh praktisi *Scrum*?

C. Batasan Masalah

Untuk menjaga fokus dan keterarahan dalam pelaksanaan pembangunan, diperlukan adanya batasan masalah yang secara jelas mendefinisikan ruang lingkup penelitian ini. Adapun batasan-batasan yang diterapkan dalam penelitian ini adalah sebagai berikut.

1. Fokus Pada Proyek Teknologi Informasi Berbasis *Scrum*

Penelitian ini dibatasi hanya untuk proyek teknologi informasi yang menggunakan kerangka kerja *Scrum* sebagai metode manajemen proyek. Semua proses pengelolaan proyek akan mengikuti praktik-praktik *Scrum*, termasuk *sprint*, penggunaan *backlog*, serta pengelolaan tugas melalui peran *product owner*, *scrum master* dan *development team*.

2. Penggunaan Arsitektur *Microservices* di *Azure Cloud Services*

Lingkup penelitian ini mencakup penerapan arsitektur *microservices* pada *Azure Cloud Services*. Fokus utama penelitian adalah bagaimana pendekatan *Microservices* memungkinkan setiap layanan berjalan secara independen dan terisolasi. Implementasi arsitektur ini terbatas pada layanan yang disediakan oleh Azure, seperti *Azure Resource Group*, *Virtual Machine* dan *Azure Database for PostgreSQL – Flexible Server*.

D. Tujuan Penelitian

Merancang dan membangun sistem manajemen proyek teknologi informasi berbasis web dengan arsitektur *microservices* agar dapat digunakan dengan mudah oleh praktisi *Scrum*.

E. Metode Penelitian

Penelitian ini menggunakan metode *waterfall* untuk mengembangkan dan merilis sistem. Setelah studi pustaka, metode ini memiliki beberapa tahapan yaitu :

1. Studi Pustaka

Studi pustaka dilakukan untuk memahami tantangan dalam praktik *Scrum*, khususnya terkait alat manajemen proyek yang dinilai terlalu kompleks. Berbagai alat, literatur dan jurnal terkait permasalahan seperti kompleksnya alat manajemen proyek *Scrum* yang ada, serta prinsip-prinsip arsitektur *microservices* yang mendukung independensi layanan dipelajari dalam tahap ini. Selain itu, kajian dilakukan terhadap teknologi pendukung seperti gRPC dan Microsoft Azure untuk memastikan sistem dapat dibangun dengan tingkat ketergantungan yang rendah dan mudah digunakan.

2. **Perancangan**

Tahap awal pembuatan sistem berupa pembuatan *use case diagram* yang bertujuan untuk mendokumentasikan dan menjelaskan alur program sistem yang dibangun. Tahap selanjutnya adalah pembuatan *Entity Relation Diagram* (ERD) yang berfungsi untuk pembuatan basis data mulai dari tabel, kolom, atribut dan relasi.

3. **Desain**

Pada tahap ini, dilakukan pembuatan desain antarmuka pengguna (*user interface*) menggunakan Figma. Desain dibuat dengan pendekatan *wireframe* untuk memvisualisasikan tata letak dari sistem manajemen proyek ini. *wireframe* dibuat menggunakan Figma untuk mempermudah proses pengerjaan.

4. **Implementasi**

Tahap implementasi melibatkan proses pengkodean untuk membangun sistem berdasarkan desain yang telah dirancang. Sistem dikembangkan dengan menggunakan arsitektur *microservices*, di mana setiap layanan seperti autentikasi, manajemen proyek, dan pengiriman email di-*deploy* secara independen menggunakan Docker pada *Azure Virtual Machine*. Komunikasi antar layanan dilakukan melalui gRPC dan REST API.

5. **Pengujian**

Pengujian dilakukan menggunakan metode *black box testing*, yaitu pengujian yang berfokus pada fungsionalitas sistem tanpa melihat struktur internal kode. Penguji hanya memberikan input dan mengamati *output* untuk

memastikan bahwa sistem memberikan respon yang sesuai. Metode ini digunakan untuk menguji fitur-fitur utama seperti *login*, manajemen proyek, *backlog*, *sprint*, dan laporan, guna memastikan semua berjalan sesuai dengan *use case* yang ada.

6. **Rilis**

Setelah seluruh pengujian dinyatakan berhasil, sistem di-*deploy* ke *Azure Cloud Services*. Proses rilis mencakup konfigurasi layanan cloud seperti *Azure Resource Group*, *Azure Virtual Machine*, *Azure PostgreSQL Flexible Server*, dan pengelolaan *container* melalui Docker.

7. **Evaluasi**

Tahap evaluasi dilakukan untuk menilai apakah sistem yang telah dibangun sesuai dengan tujuan awal, yaitu menghasilkan alat manajemen proyek berbasis *Scrum* yang mudah digunakan oleh praktisi *Scrum*. Evaluasi ini dilakukan melalui pengujian penggunaan sistem oleh pengguna secara langsung menggunakan skala Likert untuk mengukur kemudahan penggunaan sistem.

F. **Sistematika Penulisan**

Sistematika penulisan tugas akhir peneliti akan terdiri dari beberapa bab sebagai berikut:

BAB I Pendahuluan

Bab ini berisikan tentang latar belakang masalah, rumusan masalah, batasan penelitian, tujuan penelitian, metode penelitian yang digunakan, dan sistematika penulisan.

BAB II Tinjauan Pustaka

Bab ini berisikan tentang penelitian yang memiliki topik serupa dengan penelitian yang sedang dilakukan. Sumber studi kasus diperoleh melalui skripsi, jurnal, atau artikel yang dapat dipertanggung jawabkan kebenarannya.

BAB III Landasan Teori

Bab ini berisikan penjelasan tentang teori-teori atau kutipan yang diambil dari berbagai sumber, yaitu jurnal atau artikel. Tujuan dari bab ini adalah agar membantu pembaca memahami istilah-istilah yang dibahas dalam penelitian ini.

BAB IV Analisis dan Perancangan Sistem

Bab ini berisi pembahasan mengenai analisis dan perancangan sistem, meliputi *Entity Relationship Diagram* (ERD), *Use case Diagram*, desain antarmuka website, serta perancangan arsitektur *microservices* yang digunakan.

BAB V Implementasi dan Pengujian Perangkat Lunak

Bab ini berikan tentang implementasi pembangunan sistem manajemen proyek pada bagian pengkodean (*coding*). Setelah melakukan pengkodean maka dilakukan pengujian terhadap *use case* yang sudah diselesaikan.

BAB VI Penutup

Bab ini merupakan bab terakhir pada laporan tugas akhir. Bab ini berisikan tentang kesimpulan dan saran yang terkait dengan penelitian yang dilaksanakan.