# BAB I
# PENDAHULUAN

## 1.1. Latar Belakang

*Low-code development platforms* (LCDP) merupakan sebuah platform pengembangan perangkat lunak berbasis *cloud* dengan menyediakan Platform-as-a-Service (PaaS) model yang memungkinkan pengguna untuk mengembangkan perangkat lunak dengan berinteraksi melalui antarmuka pengguna dengan grafis dinamis,diagram visual, dan bahasa deklaratif [1]. Hingga saat ini, sudah ada banyak sekali LCDP yang dapat digunakan. Namun, untuk melakukan pengujian pada aplikasi yang dibangun dengan LCDP belum ada *framework* yang dapat digunakan secara umum. Low Code Testing Framework (LCTF) yang ada pada saat ini hanya dapat digunakan oleh platform masing-masing. Selain itu, LCTF dari masing-masing LCDP ini juga tidak dapat diakses secara publik, hingga membatasi penggunaannya [2]. Kemudian selain itu, hingga saat ini riset terkait penelitian LCDP juga masih sangatlah sedikit dan pengujian pada LCDP merupakan salah satu yang masih sedikit penelitiannya [3]. Dikarenakan tidak ada LCTF yang dapat digunakan secara umum, hal ini memicu beberapa tantangan, salah satunya yaitu kebutuhan atas *high-level automation*. Otomatisasi pengujian dibutuhkan karena salah satu kelebihan LCDP, yaitu dapat membangun aplikasi yang kompleks, sehingga semakin kompleks aplikasi maka dibutuhkan otomatisasi pengujian sehingga tidak menghabiskan waktu dan biaya [2].

Penelitian ini menggunakan platform Mendix sebagai LCDP yang akan digunakan, karena dapat diintegrasikan dengan aplikasi pihak ketiga [2]. Mendix menggunakan pendekatan Model-Based untuk proses pengujiannya, dimana menggunakan Microflow sebagai modelnya. Selain sebagai model untuk melakukan pengujian, microflow juga digunakan untuk membuat logika untuk aplikasi yang dibangun [4]. Microflow diadaptasi dari model Business Process Model Notation (BPMN) [4] . Meskipun Mendix sudah dapat melakukan pengujian otomatis menggunakan Microflow, namun pengujian ini hanya dapat dilakukan untuk *unit testing* saja [5].

Data-Driven, Model-Based dan Record and Replay merupakan teknik pengujian yang paling sering digunakan karena kemudahan penggunaannya serta

otomatiosasinya, namun Model-Based Testing (MBT) merupakan teknik yang paling komprehensif dikarenakan dapat melakukan automatisasi pengujian di skala manapun, dari pengujian *unit* hingga pengujian *end-to-end* [2]. Hal ini membuat Model-Based Testing dapat memastikan kualitas sistem secara menyeluruh, tidak hanya pada beberapa skala saja. Kemudian berdasarkan penelitian yang dilakukan oleh Tosi et al. (2025), riset saat ini dianjurkan untuk menggunakan metode pemodelan untuk menanggulangi permsalahan yanga ada pada LCDP [6]. Selain itu, hal ini juga dapat mengatasi permasalahan *vendor lock-in* pada LCDP [7]. Maka dari itu, penelitian ini akan dilakukan dengan menggunakan Model-Based Testing untuk melakukan pengujian otomatis. Selain itu, seperti yang sudah disebutkan sebelumnya, dikarenakan Mendix menggunakan pendekatan MBT, maka teknik MBT ini cocok digunakan karena dapat meningkatkan skala pengujian MBT di Mendix dari *unit testing* menjadi *integration testing* [5].

Model-Based Testing memiliki banyak model, antara lain Finite State Machine (FSM), Extended Finite State Machine (EFSM), Activity Diagram, dan masih banyak lagi. EFSM merupakan model yang dikembangkan dari model FSM. EFSM memiliki karakteristik antara lain, yaitu dapat meletakkan suatu kondisi di suatu transisi [8]. Hal ini membuat EFSM dapat beradaptasi dengan seiringnya kompleksitas dari suatu aplikasi.

Untuk melakukan otomatisasi pengujian dengan MBT-EFSM pada Mendix, maka perlu dilakukan transformasi dari model Microflows ke model EFSM. Model EFSM ini nantinya akan dibuat menggunakan TestOptimal. TestOptimal merupakan *tools* yang digunakan untuk menghasilkan *test script* secara automatisasi dengan menggunakan model FSM atau EFSM [9] . Maka dari itu, dikarenakan penelitian ini dilakukan menggunakan model EFSM, maka *tools* TestOptimal merupakan *tools* yang cocok untuk digunakan pada penelitian ini. Selain itu, terdapat penelitian serupa oleh Wibowo et al. (2021) yang melakukan pengujian pada aplikasi web SiPProp menggunakan model Extended Finite State Machine (EFSM). Aplikasi SiPProp memiliki fitur utama berupa fungsi CRUD (Create, Read, Update, dan Delete), yang memiliki kemiripan dengan sistem manajemen training event yang akan dibangun menggunakan platform Mendix [9]

Penelitian ini akan dilakukan dengan mengadopsi metode STLC (Software

Testing Life Cycle). Di akhir penelitian, hasil pengujian pada MBT-EFSM di TestOptimal akan dibandingkan dengan hasil pengujian di menggunakan metode Record and Replay di Selenium IDE. Selain itu juga *state* dan *transition* yang ada pada EFSM akan dipetakan dengan aktivitas-akivitas yang ada di Microflow. Penelitian oleh Yazdani Seqerloo et al. (2021) menunjukkan pendekatan serupa, di mana mereka melakukan transformasi model dari BPMN ke state diagram menggunakan aturan transformasi. Untuk memverifikasi hasil transformasi, mereka menghitung nilai *precision* dan *recall* sebagai bagian dari proses analisis validasi [10]. Untuk memastikan bahwa transformasi model dari Microflow ke EFSM dilakukan dengan baik, dimana semua proses yang ada di Microflow terkonversi dengan baik tana menyalahi alur logika yang ada pada sebelumnya, maka akan digunakan *traceability* untuk memetakan proses-proses yang ada di MBT ke *state* dan *transition* yang ada pada EFSM, hal ini sesuai dengan penetilitian dari Matragkas et al. (2015) [11].

## 1.2. Rumusan Masalah

*Low Code Development Platform* (LCDP) seperti Mendix memberikan kemudahan dalam membangun aplikasi melalui pendekatan visual. Pada proses pengujian, Mendix menggunakan Microflow sebagai model untuk membuat *test case* pengujian. Meskipun dengan menggunakan Microflow pengujian dapat dilakukan secara otomatis, namun pengujian ini hanya terbatas pada *unit testing* saja. Untuk meningkatkan skala pengujian, maka diperlukannya otomatisasi pengujian di luar Mendix menggunakan aplikasi pihak ketiga. Dikarenakan Mendix menggunakan pendekatan *Model-Based Testing* (MBT), maka pengujian akan dilakukan dengan menggunakan MBT. Model yang digunakan untuk melakukan *Model-Based Testing* adalah *Extended Finite State Machine* (EFSM), hal ini dikarenakan EFSM dapat meletakkan suatu kondisi di suatu transisi, sehingga dapat menyesuaikan dengan model Microflow yang ada pada Mendix. Untuk membuat model EFSM, maka diperlukan untuk melakukan konversi terlebih dahulu dari Microflow menjadi EFSM, kemudian model dari EFSM ini akan digunakan untuk melakukan pengujian dengan bantuan aplikasi otomatisasi pengujian pihak ketiga, yaitu TestOptimal. Berdasarkan hal ini, penelitian ini difokuskan untuk menjawab permasalahan utama berikut:

1. Bagaimana cara mentransformasi model Microflow pada Mendix menjadi model *Extended Finite State Machine* (EFSM) ?
2. Bagaimana cara memastikan aturan-aturan yang diterapkan untuk melakukan transformasi ke EFSM sudah sesuai dengan logika aplikasi yang ada pada Microflow?
3. Bagaimana pengujian dengan menggunakan MBT-EFSM dapat meningkatkan skala pengujian dari *unit testing* menjadi *integration testing*?

## 1.3. Tujuan

Penelitian ini bertujuan untuk mengembangkan pendekatan pengujian *Model-Based Testing* (MBT) pada platform *Low-Code Development Platform* (LCDP) Mendix, dengan mentransformasi model Microflow menjadi model *Extended Finite State Machine* (EFSM) yang akan diuji menggunakan aplikasi pengujian pihak ketiga, yaitu TestOptimal. Transformasi yang dilakukan diharapkan dapat merepresentasikan logika aplikasi yang ada pada Mendix secara akurat dan meningkatkan skala pengujian dari *unit testing* menjadi *integration testing*.

## 1.4. Manfaat

Manfaat yang diharapkan dari penelitian ini adalah:

1. Menyediakan pedoman transformasi model Microflow ke EFSM yang dapat digunakan kembali untuk pengujian aplikasi berbasis Mendix.
2. Memberikan hasil dari metode verifikasi model transformasi melalui precision, recall, dan traceability sehingga analisis pengujian transformasi lebih akurat.
3. Memperluas cakupan dan efektivitas pengujian otomatis pada platform *low-code*, khususnya Mendix, dari tingkat unit ke tingkat integrasi.

## 1.5. Batasan Masalah

Batasan masalah pada penelitian ini adalah sebagai berikut :

1. Aplikasi akan dibangun menggunakan platform Mendix.
2. Aplikasi yang dibangun pada Mendix adalah aplikasi sistem manajemen sederhana.

3. Pengujian yang akan dilakukan pada Mendix, menggunakan modul dari yang tersedia di Mendix, yaitu Unit Testing.
4. Pengujian yang akan dilakukan pada TestOptimal, menggunakan Model Based Testing dengan model Extended Finite-State Machine
5. Transformasi model akan dianalisis berdasarkan *precision* dan *recall* dan *traceability*

## 1.6. Metode Penelitian

Penelitian ini akan mengadaptasi Software Testing Life Cycle (STLC) sebagai metode penelitian. Dengan mengadaptasi metode STLC, penelitian akan dilakukan dalam 6 tahapan, yaitu *AUT Preparation, Test Planning, Test Case Development, Environment Setup, Test Execution, dan Test Result Analysis*. *Proses Model-Based testing mencakup Test Planning, Test Case Development, Environment Setup*, dan *Test Execution*. Proses MBT ini akan dilakukan secara iteratif hingga menemukan aturan-aturan konversi yang sudah sesuai.

## 1.7. Jadwal Pelaksanaan

Berikut Perencanaa jadwal kegiatan untuk riset yang akan dilakukan:

**Tabel 1. Jadwal Pelaksanaan Tugas Akhir**

| No. | Deskripsi Tahapan | Bulan 1 | Bulan 2 | Bulan 3 | Bulan 4 | Bulan 5 | Bulan 6 |
|---|---|---|---|---|---|---|---|
| 1 | Studi Literatur | | | | | | |
| 2 | Analisi Kebutuhan | | | | | | |
| 3 | Perancangan Pengujian | | | | | | |
| 4 | Pengembangan Uji Kasus | | | | | | |
| 5 | Perancangan Lingkungan Uji | | | | | | |
| 6 | Eksekusi Pengujian | | | | | | |
| 7 | Analisis Hasil Pengujian | | | | | | |
| 8 | Penyusunan Laporan/Buku TA | | | | | | |