BAB I
PENDAHULUAN

1.1 Latar Belakang

Sebuah perangkat lunak dikatakan siap untuk dipakai yaitu apabila sudah melalui tahap pengujian. Dalam dunia pemrograman, pengujian adalah metode verifikasi perangkat lunak di mana seorang programmer menguji bahwa suatu unit atau bagian terkecil atau fitur dari program layak untuk dipakai. (Thohari, A. N. A., & Amalia, A. E., 2018). Pengujian biasanya ditulis dan dijalankan oleh seorang programmer untuk memastikan bahwa kode telah memenuhi desain dan berfungsi sebagaimana yang diinginkan. Tujuan dari pengujian adalah mengisolasi setiap bagian dari program dan menunjukkan bahwa bagian-bagian individu (unit-unit tersebut) adalah benar. Di dalam metodologi pemrograman procedural programming, sebuah unit adalah sebuah fungsi (*function*) atau sebuah prosedur (*procedure*).

Pada era pengembangan perangkat lunak dengan metodologi tradisional, pengujian dilakukan dengan cara mencoba satu persatu fitur aplikasi ketika aplikasi yang dikembangkan sudah jadi. Cara pengujian tersebut akan membutuhkan waktu yang lama apabila pengembang mengerjakan proyek perangkat lunak dalam skala besar. Masalah yang timbul yaitu jika ada beberapa fitur yang ingin ditambahkan ke sistem, dan ternyata ada beberapa fitur yang diubah karena adanya perubahan kebijakan perusahaan pemilik aplikasi, maka pengembangan sistem harus mempertimbangkan beberapa hal, misalnya: mengubah stuktur tabel yang saat ini sudah berjalan dengan aman, ada potensi bug yang muncul akibat perubahan struktur basis data, ada potensi bug yang muncul akibat penambahan atau perubahan kode fitur yang dilakukan dan proses migrasi data yang sudah ada (*existing*) ke sistem yang baru (setelah ditambah fitur). Jika terjadi kesalahan pada saat pengembangan maka akan berakibat: komplain yang tidak terduga dari pengguna, terkadang waktu yang diperlukan untuk perbaikan akibat kesalahan pada saat pengembangan penambahan fitur baru bisa lebih banyak daripada waktu pengerjaan dan fokus pekerjaan terbagi antara mengerjakan fitur baru dengan rasa takut komplain yang akan datang belakangan.

Salah satu metode pengembangan perangkat lunak yang dapat menghemat waktu pengujian, namun fungsionalitasnya tetap terjaga adalah *Test-Driven Development (TDD)*. *Test-driven development (TDD)* merupakan suatu proses pengembangan perangkat lunak dengan pengulangan siklus pengembangan yang sangat singkat: pertama pengembang menulis kode pengujian untuk pendefinisian fungsi yang diinginkan, kemudian menghasilkan kode

program untuk lulus tes dan akhirnya menghasilkan kode baru yang dapat diterima (Setiawan, A., Satoto, K.I., Isnanto, R.R., 2013). Pada metode TDD, pengembangan perangkat lunak dilakukan dengan membuat *test case* terlebih dahulu baru kemudian melakukan *producing code* (Thohari, A. N. A., & Amalia, A. E., 2018).

Pengujian yang digunakan pada metode TDD ini adalah menggunakan *Automated Testing*. *Automated Testing* adalah sebuah metode pengujian software yang menggunakan suatu *Software Testing* (terpisah dari sistem atau aplikasi yang sedang diuji), dengan tujuan untuk membandingkan antara "output yang diprediksikan" dengan "output yang dihasilkan" oleh sistem. *Automated Testing* memanfaatkan sebuah *software* atau *framework testing* sebagai pengganti manusia untuk menguji apakah sistem atau aplikasi sudah sesuai dengan spesifikasi yang ditentukan atau tidak. Dengan aplikasi yang memiliki *Automated Testing*, pengembang memiliki cadangan yang dapat membantu memastikan bahwa aplikasi sudah teruji secara otomatis dengan bantuan komputer.

Jenis *Automated Testing* yang digunakan pada penelitian ini adalah *Unit Testing*, *Unit testing* atau pengujian unit berfokus pada *verifikasi* pada unit yang terkecil pada sebuah desain perangkat lunak (komponen atau modul perangkat lunak). Setiap unit perangkat lunak diuji agar dapat diperiksa apakah masukan (*input*) dan keluaran (*output*) dari unit sudah sesuai dengan yang diinginkan. *Unit testing* biasanya dilakukan saat kode program dibuat. Unit di sini secara fisik dapat berupa *method* atau *procedure* yang terdapat pada suatu kelas dalam pemrograman berorientasi objek (Setiawan, A., Satoto, K.I., Isnanto, R.R., 2013).

Dibandingkan jika harus menguji sistem secara manual, dengan adanya *Automated Testing* yang diintegrasikan pada aplikasi, semua fitur yang sudah ditentukan diuji oleh komputer dengan waktu yang cepat, sehingga dapat menghemat waktu pengujian dan **j**ika pada aplikasi terdapat *bug* yang disebabkan karena adanya penambahan fitur, *Automated Testing* segera memberitahu dengan menampilkan *error* yang terjadi, baik karena *script* yang salah, atau struktur *database* yang ada tidak sesuai dengan *script* yang baru.

Pada penelitian ini, dikembangkan sebuah aplikasi web berupa website DIGITALOKA menggunakan metode *Test-Driven Development* dimana dilakukan *Automatic Testing* pada *source code* yang sedang dibuat. DIGITALOKA adalah sebuah startup yang bergerak di bidang *e-commerce* dan aplikasinya adalah website *marketplace* yang berfokus pada produk virtual dan mempertemukan antara penjual dan pembeli di seluruh Indonesia untuk melakukan transaksi jual beli produk virtual secara daring. Bahasa pemrograman yang dipakai adalah PHP dengan menggunakan *framework* "Laravel".

Alat yang digunakan untuk mengmplementasikan *automated testing* yaitu menggunakan *framework* PHPUnit. PHPUnit adalah framework testing berorientasi *programmer* untuk bahasa pemrograman PHP. Hasil implementasi *Automatic Testing* adalah membuktikan bahwa fungsi-fungsi dari aplikasi web yang dibangun dapat berkerja dengan baik dan meminimalisir terjadinya *bug*. Selain itu kode program yang dihasilkan juga menjadi rapi dan mudah dibaca oleh pengembang lain karena menerapkan konsep *refactoring*.

1.2 Rumusan Masalah

Rumusan masalah yang diangkat berdasarkan latar belakang masalah adalah Bagaimana implementasi *automated testing* menggunakan PHPUnit pada website marketplace berbasis Laravel?

1.3 Batasan Masalah

Batasan masalah yang diambil untuk membatasi sasaran utama adalah sebagai berikut:

a. Pengujian dilakukan pada fitur yang telah ditentukan pada website marketplace DIGITALOKA karena aplikasi web telah dikembangkan secara tim.
b. Pengujian yang dilakukan hanya pengujian unit (*Unit Testing*), tidak mencakup pengujian Model (*Model Behaviour*).

1.4 Tujuan Penelitian

Tujuan dari penelitian ini adalah Mengetahui kebutuhan, pembuatan, dan cara kerja *Automated Testing* dalam melakukan pengujian sebuah aplikasi website serta menerapkan metode-metode tersebut untuk melakukan pengujian terhadap sebuah sistem *e-commerce* yang dibangun menggunakan bahasa PHP dengan *framework* Laravel.

1.5 Manfaat Penelitian

Adapun manfaat yang ingin didapatkan pada penelitian ini adalah sebagai berikut:

a. Mengetahui implementasi metode pengembangan *Test-Driven Development* dengan *Automated Testing* menggunakan PHPUnit untuk pengembangan aplikasi website marketplace.
b. Mengetahui hasil secara menyeluruh dari perbandingan metode pengujian *Test-Driven Development* dengan *Automated Testing* menggunakan PHPUnit dan menggunakan metode pengujian tradisional.

c. Pengujian menggunakan PHPUnit dapat membantu pengembang dalam menangani kesalahan dan memudahkan dalam penambahan fitur baru dari aplikasi web.
d. Memastikan bahwa *source code* dari aplikasi website yang dikembangkan telah memenuhi desain dan berfungsi sebagaimana yang diinginkan sehingga didapatkan hasil sebuah aplikasi website yang mendekati 0% kecacatan.

1.6 Usulan Penyelesaian

Untuk membuat aplikasi website *marketplace* yang terstruktur dan minim kecacatan, dibuat sebuah sistem *Automatic Testing* pada aplikasi website marketplace DIGITALOKA, aplikasi website *marketplace* DIGITALOKA dibuat dan dikembangkan menggunakan bahasa pemrograman PHP dengan *framework* Laravel, pengujian yang dilakukan adalah pengujian unit dan pengujian fitur, pengujian diimplementasikan melalui *framework* pengujian untuk bahasa PHP yaitu PHPUnit, PHPUnit sudah terintegrasi dengan framework Laravel. PHPUnit diintegrasikan dengan *package* "PHPUnit Watcher" agar pengujian dilakukan secara otomatis saat ada perubahan pada kode program sehingga dapat menghemat waktu dalam melakukan pengujian, setelah pengujan selesai, dilakukan *commit* ke repositori *git* dan dijalankan *Continous Integration* menggunakan Bitbucket pipeline.

1.7 Sistematika Penulisan

Sistematika penulisan dibuat untuk pembaca agar lebih mudah dalam memahami laporan tugas akhir. Secara garis besar, sistematika penulisan tugas akhir dibagi menjadi 6 Bab. Berikut merupakan poin dalam sistematika penulisan pada tugas akhir ini.

a. BAB 1 PENDAHULUAN

Bab pendahuluan berisi beberapa pokok pembahasan meliputi latar belakang, rumusan masalah, batasan masalah, tujuan penelitian, manfaat penelitian, usulan penelitian, dan sistematika penulisan.

b. BAB II LANDASAN TEORI

Landasan teori berisi tinjauan pustaka dan teori dasar.Teori yang berkaitan dengan peneleitian ini yang diambil dari sumber-sumber seperti peneltian yang sudah ada, buku atau referensi lain yang berupa pengertian dan definisi. Bab ini juga menjelaskan kosep – konsep dasar sistem, konsep dasar informasi dan berbagai definisi yang bersangkutan dengan sistem yang akan dibuat.

c. BAB III ANALISIS DAN PERANCANGAN

Analisis dan perancangan mendeskipsikan bagaimana gambaran umum dari sistem, rancangan pengujian dan kerangka pengujian yang nantinya akan digunakan sebagai patokan dalam membuat sebuah sistem.

d. BAB IV IMPLEMENTASI PENGUJIAN

Bab implementasi pengujian berisi tentang pemaparan implementasi pengujian dari rancangan sistem yang telah dibuat, langkah-langkan pengujian serta hasil dari proses pengujian yang telah dilakukan.

e. BAB V KESIMPULAN

Bab kesimpulan berisi mengenai kesimpulan dan saran dari pengujian yang telah dilaksanakan, saran diperuntukkan agar menjadi perbaikan dan pertimbangan untuk penelitian tentang pengujian yang sejenis kedepannya.