BAB 6

KESIMPULAN DAN SARAN

6.1 Kesimpulan

Perangkat lunak sistem pembangkit kode uji berbasis *behavior specification* berhasil dibangun. Berdasarkan hasil analisis dan perancangan perangkat lunak, cara kerja sistem pembangkit kode uji berbasis *behavior specification* berhasil dipelajari. Cara kerja perangkat lunak dimulai dari membaca *file* skenario Gherkin, setelah itu kalimat dari skenario Gherkin di*parsing* untuk mendapatkan *keyword* yang dibutuhkan, setelah mendapatkan semua *keyword*, *file testing* sudah dapat dibuat dan menghasilkan *file testing* berformat php. *File testing* yang dibangkitkan oleh sistem pembangkit kode uji yaitu *file* PHPUnit dan Dusk. Terdapat sedikit perbedaan antara skenario Gherkin PHPUnit dengan Dusk dikarenakan alur pengujiannya berbeda. Pada pengujian PHPUnit, hal yang dilakukan ialah pengujian *unit testing*, sedangkan pada Dusk dilakukan pengujian *integration testing*. Hal yang diuji pada *unit testing* ialah fungsi setiap fiturnya, sedangkan pada *integration testing*, komponen lain seperti elemen pada antarmuka *website* PERPUSKU ikut serta dalam pengujian. Pada skenario Dusk terdapat baris yang berisi perintah "menekan suatu link", namun pada skenario Gherkin PHPUnit tidak ada perintah tersebut. Oleh karena itu, terdapat perbedaan dari segi skenario maupun *file testing* yang dihasilkan.

Pengujian PHPUnit dapat dilihat dari tiap hasil pengujian selalu lebih cepat waktunya dibandingkan pengujian Dusk, dikarenakan pengujian PHPUnit hanya menguji suatu *method*, sedangkan untuk pengujian Dusk, antarmuka untuk fiturnya ikut untuk diuji. Berdasarkan pengujian eksperimental, *file* PHPUnit dan Dusk bergantung pada skenario Gherkin yang dibuat, jika skenario Gherkin yang dibuat benar, maka *file* PHPUnit dan Dusk juga benar serta ketika pengujian terhadap kedua *file* tersebut dijalankan akan berhasil, tetapi jika skenario Gherkin salah, maka *file* PHPUnit dan Dusk juga akan salah, serta pengujian terhadap kedua *file* tersebut juga akan gagal. Jika kode pada website *PERPUSKU* salah, hal tersebut tidak mempengaruhi pada pembangkitan kode uji oleh sistem pembangkit kode uji berbasis *behavior specification*, *file* PHPUnit dan Dusk tetap berhasil dibuat dan benar isinya, tetapi ketika pengujian dilakukan, hasil yang dikeluarkan akan gagal karena kode yang salah tersebut.

6.2 Saran

Berdasarkan hasil skripsi yang dilakukan, penulis dapat memberikan beberapa saran sebagai berikut :

1. Menambahkan *keyword* baru jika ada kondisi yang belum ditangani pada skripsi ini.
2. Pada skripsi ini belum ditangani kondisi pemeriksaan otentikasi sehingga hal ini dapat ditambahkan jika skripsi ini dikembangkan lebih lanjut.

DAFTAR REFERENSI

[1] Balaji, S. dan Murugaiyan, M. (2012) Waterfall vs v-model vs agile: A comparative study on sdlc. *International Journal of Information Technology and Business Management*, **2**, 26–30.

[2] Al-Fedaghi, S. (2016) International journal of advanced computer science and applications. *International Journal of Advanced Computer Science and Applications*, **7**, 1–6.

[3] Lewis, W. dan Veerapillai, G. (2008) *Software Testing and Continuous Quality Improvement*, 3rd edition. Auerbach Publications, United States of America.

[4] Sale, D. (2014) *Testing Python*, 1st edition. Wiley, New Jersey.

[5] Smart, J. (2014) *BDD in Action: Behavior-driven development for the whole software lifecycle*, 1st edition. Manning Publications, United States of America.

[6] Bergmann, S. (2005) *PHPUnit Pocket Guide: Test-Driven Development in PHP*, 1st edition. O'Reilly Media, California.

[7] Stauffer, M. (2019) *Laravel: Up and Running: A Framework for Building Modern PHP Apps*, 2nd edition. O'Reilly Media, California.

[8] Kumar, S. dan Dubey, P. (2013) Software development life cycle (SDLC) analytical comparison and survey on traditional and agile methodology. *Journal of Research in Science and Technology*, **2**, 22–30.

[9] Wibisono, W. dan Baskoro, F. (2002) Pengujian perangkat lunak dengan menggunakan model behaviour UML. *JUTI: Jurnal Ilmiah Teknologi Informasi*, **1**, 43.

[10] Freeman, H. (2002) Software testing. *IEEE instrumentation measurement magazine*, **5**, 48–50.

[11] Gulati, S. dan Sharma, R. (2017) *Java Unit Testing with JUnit 5: Test Driven Development with JUnit 5*, 1st edition. Apress, New York City.

[12] Lazar, I., Motogna, S., dan Pârv, B. (2010) Behaviour-driven development of foundational uml components. *Electr. Notes Theor. Comput. Sci.*, **264**, 91–105.

[13] Herrington, J. (2003) *Code Generation in Action*, 1st edition. Manning Publications, United States of America.

[14] Jorgensen, P. (2018) *Software Testing: A Craftsman's Approach*, 4th edition. Auerbach Publications, United States of America.

[15] Kaner, C. (2003) An introduction to scenario testing. *Lecture Notes, Center for Software Testing Education and Research, Florida Institute of Technology*, **11**, 10.

[16] Everett, G. (2007) *Software Testing : Testing Across the Entire Software Development Life Cycle*, 1st edition. Wiley, New Jersey.

[17] Bourque, P., Dupuis, R., Abran, A., Moore, J., dan Tripp, L. (1999) The guide to the software engineering body of knowledge. *IEEE Software*, **16**, 35–44.

[18] Mathur, S. dan Malik, S. (2010) Advancements in the v-model. *International Journal of Computer Applications*, **1**, 30–35.

[19] Stellman, A. dan Greene, J. (2017) *Head First Agile: A Brain-Friendly Guide to Agile and the PMI-ACP Certification*, 1st edition. O'Reilly Media, Inc., California.

[20] McCool, S. (2012) *Laravel Starter*, 1st edition. Packt Pub., Birmingham.