Journal of Information System Research (JOSH)
Volume 4, No. 4, Juli 2023, pp 1455−1463
ISSN 2686-228X (media online)
https://ejurnal.seminar-id.com/index.php/josh/
DOI 10.47065/josh.v4i4.3948

Perancangan Aplikasi Naviku untuk Memberikan Informasi Navigasi Kepada Tunanetra Menggunakan Metode Test Driven Development

Muhammad Rafiadly, Rahmat Fauzi*, Ahmad Musnansyah

Fakultas Rekayasa Industri, Program Studi Sistem Informasi, Universitas Telkom, Bandung
Jl. Telekomunikasi. 1, Terusan Buahbatu - Bojongsoang, Telkom University, Sukapura, Kec. Dayeuhkolot, Kabupaten Bandung, Jawa Bara, Indonesia
Email: [1]muhammadrafiadly@student.telkomuniversity.ac.id, [2,*]rahmatfauzi@telkomuniversity.ac.id, [3]ahmadanc@telkomuniversity.ac.id
Email Penulis Korespondensi: rahmatfauzi@telkomuniversity.ac.id
Submitted: **27/07/2023;** Accepted: **31/07/2023;** Published: **31/07/2023**

Abstrak−Difabel netra adalah sebutan untuk seseorang yang telah kehilangan atau berkurangnya fungsi penglihatan. Penyandang difabel netra sering mengalami kesulitan dalam beraktivitas sehari-hari, contohnya saat berjalan atau bernavigasi. Mereka memerlukan bantuan orang lain agar dapat membantu mereka untuk pergi ke suatu ruangan maupun berpindah dari satu tempat ke tempat lain. Saat ini, alat bantu navigasi yang ada belum sepenuhnya memenuhi kebutuhan para difabel netra. Mereka masih memerlukan bantuan dari orang lain untuk mendapatkan informasi detail tentang objek-objek visual seperti tata letak ruangan dalam gedung, rambu lalu lintas, dan kendaraan. Berdasarkan permasalahan tersebut, diperlukan sebuah aplikasi navigasi berbasis Android yang memanfaatkan teknologi Text-to-Speech (TTS) untuk membantu penyandang difabel netra melakukan navigasi secara mandiri dengan memanfaatkan informasi navigasi yang diberikan melalui suara. Agar proses pengembangan terbantu, aplikasi dikembangkan dengan mengadopsi prinsip Agile Software Development dengan metode Test Driven Development (TDD). Fitur yang dikembangkan diawali dengan fase test fails dimana developer membuat kode unit test sebelum kode produksi diimplementasi, lalu dilakukan refactoring pada kode, dan melakukan pengujian setelahnya (test passes). Status keberhasilan pada fitur yang diuji mendapatkan status success dengan rate 100% di semua fitur. Lalu dalam uji TTS dilakukan pengujian dengan 2 mesin TTS, yaitu Google TTS dan Samsung TTS didapatkan hasil bahwa tingkat Word Error Rate (WRE) dari Google TTS yaitu 2,86, lebih kecil dari Samsung TTS yang mendapatkan angka 4,06.

Kata Kunci: Difabel Netra; Navigasi; Text-to-Speech; Pengembangan Aplikasi Android; Unit Test; Test Driven Development

Abstract−Visually impaired is a term for someone who has lost or reduced the function of vision. People with visual disabilities often experience difficulties in daily activities, for example when walking or navigating. They need the help of others to help them go to a room or move from one place to another. Currently, existing navigation aids do not fully meet the needs of the blind. They still need the help of others to get detailed information about visual objects such as the layout of rooms in buildings, traffic signs, and vehicles. Based on these problems, an Android-based navigation application needed that utilizes Text-to-Speech (TTS) technology to help blind people navigate independently by utilizing navigation information provided by voice. In order to help the development process, applications are developed by adopting Agile Software Development principles using the Test Driven Development (TDD) method. The features developed begin with a test fail phase where the developer creates unit test code before the production code is implemented, then refactor the code, and tests afterward (test passes). The success status of the features being tested gets success status with a rate of 100% for all features. Then in the TTS test, testing was carried out with 2 TTS engines, namely Google TTS and Samsung TTS, the results showed that the Word Error Rate (WRE) of Google TTS was 2,86, lower than Samsung TTS which got 4,06.

Keywords: Visually Impaired; Navigation; Text-to-Speech; Android Application Development; Unit Test; Test Driven Development

1. PENDAHULUAN

Perlu diketahui bahwa dari berbagai jenis disabilitas, populasi penyandang difabel netra cukup signifikan. Saat ini, sekitar 2,2 miliar orang di seluruh dunia mengalami gangguan penglihatan, dimana sekitar 1 miliar di antaranya menderita gangguan penglihatan yang sebenarnya dapat dicegah atau belum diatasi dengan baik [1]. Berdasarkan survei yang dilakukan oleh Badan Pusat Statistik (BPS), didapatkan data bahwa jumlah penyandang difabel netra mendominasi di antara jenis disabilitas lainnya, yakni mencapai 63,7% [2]. Penyandang difabel netra menghadapi banyak kesulitan, terutama dalam mengakses informasi. Hal tersebut membatasi kemampuan mereka untuk beraktivitas mandiri karena mereka tidak dapat melakukan beberapa tugas seperti orang normal. Beberapa dari mereka juga mengandalkan indera pendengaran maupun alat bantu seperti tongkat ketika mereka berjalan untuk membuat navigasi sendiri dan itu merupakan tugas yang sulit bagi mereka [3]. Maka dari itu, melalui penggunaan teknologi, kemampuan penyandang difabel netra dapat ditingkatkan dengan memperluas pengalaman mereka dengan memberikan deskripsi lingkungan yang serupa [4]. Terdapat alat bantu modern seperti kacamata pintar yang dapat mendeteksi objek dalam bentuk 3D dan memiliki bantuan suara, tongkat pintar yang memberikan getaran dan suara ketika mendeteksi objek, serta beberapa aplikasi khusus pada smartphone untuk mendeteksi objek yang dirancang khusus untuk kebutuhan difabel netra [5]. Dalam penelitian ini akan dirancang sebuah aplikasi smartphone yang memiliki fitur navigasi. Navigasi merupakan tugas kompleks yang menggabungkan strategi dan isyarat lingkungan guna mencapai tujuan. Penyandang difabel netra menghadapi tantangan navigasi seperti menghindari rintangan di jalan, penentuan arah di luar ruangan, serta di dalam ruangan karena banyak informasi yang bersifat visual [6]. Pada penelitian ini terdapat literatur yang mendukung, seperti penelitian yang

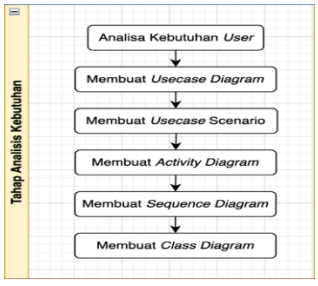
Journal of Information System Research (JOSH)
Volume 4, No. 4, Juli 2023, pp 1455−1463
ISSN 2686-228X (media online)
https://ejurnal.seminar-id.com/index.php/josh/
DOI 10.47065/josh.v4i4.3948

dilakukan oleh Dong, Schafer, dan lainnya pada penelitian yang berjudul "PERCEPT-V: Integrated Indoor Navigation System for the Visually Impaired using Vision-Based Localization and Waypoint-Based Instructions" (2020), mereka melakukan penelitian terkait sistem navigasi dalam ruangan bernama "PERCEPT-V" untuk difabel netra berbasis visi dan memberikan intruksi navigasi secara real-time menuju tujuan yang dipilih, termasuk pemandu-pemandu penting di sepanjang jalur [7]. Lalu ada penelitian yang dilakukan oleh Real dan Araujo yang berjudul "Navigation Systems for the Blind and Visually Impaired: Past Work, Challenges, and Open Problems", penelitian yang menyajikan pandangan terkini serta komprehensif terkait pengembangan perangkat navigasi untuk difabel netra dalam berbagai lingkungan, baik dalam ruangan maupun luar ruangan. Penelitian tersebut juga membahas solusi-solusi sebelumnya dari sudut pandang sejarah, mulai dari "Electronic Travel Aids" pertama hingga sistem terbaru berbasis visi buatan. Prinsip-prinsip desain pencahayaan pada pengguna juga sama dengan pelaporan teknologi terbaru. Ponsel pintar dan perangkat wearable dengan kamera bawaan dianggap sebagai pilihan potensial untuk mendukung solusi komputer visi terkini, yang memungkinkan perawatan posisi dan pemantauan area sekitar pengguna [8]. Agar dapat mengetahui langsung permasalahan yang dialami oleh penyandang difabel netra, penulis melakukan wawancara langsung dengan penyandang difabel netra. Wawancara dilaksanakan di Persatuan Tunanetra Indonesia (Pertuni) yang terletak di Cimahi, Jawa barat. Pada wawancara tersebut didapatkan responden sebanyak 21 orang yang terdiri dari 12 orang difabel netra dengan kategori totally blind dan 9 orang difabel netra dengan kategori low vision. Terdapat pertanyaan pada wawancara tersebut untuk mengetahui permasalahan apa saja yang dialami oleh difabel netra saat bernavigasi di dalam maupun di luar ruangan. Meskipun mayoritas responden difabel netra telah menggunakan tongkat dan smartphone sebagai alat bantu navigasi, tetapi diperlukan pengembangan teknologi yang lebih canggih dan inovatif untuk meningkatkan aksesibilitas dan kemandirian mereka. Kesimpulan dari hasil wawancara tersebut yaitu difabel netra dengan kategori low vision maupun totally blind menghadapi berbagai tantangan dalam navigasi luar ruangan, termasuk permukaan jalan yang tidak rata, minim pencahayaan di malam hari, dan adanya objek penghalang. Penggunaan alat bantu seperti tongkat dan smartphone membantu, tetapi belum sepenuhnya memenuhi kebutuhan mereka, terutama ketika berada di tempat yang baru atau dalam ruangan gelap. Dengan mempertimbangkan hasil kesimpulan wawancara, maka dirancanglah aplikasi bernama "Naviku" yang dapat diakses melalui smartphone yang dimiliki oleh responden difabel netra. Aplikasi ini bertujuan untuk menjadi alat bantu dalam navigasi baik dalam ruangan maupun luar ruangan. Sebagai alat bantu navigasi, Naviku mengimplementasikan sistem Text-to-Speech sebagai teknologi yang digunakan untuk memberikan informasi kepada difabel netra. TTS merupakan sebuah sistem yang secara otomatis menghasilkan suara dari semua karakter teks melalui transkip grafem ke fenom dengan menggunakan sampel suara, warna yang sama, dan suara asli sebagai suara sintetik serta penggunaan bisa secara online maupun installer [9]. Dengan teknologi TTS, perangkat dapat berkomunikasi serta berinteraksi dengan manusia tidak hanya secara tertulis, namun juga secara lisan menggunakan bahasa yang digunakan sehari-hari [10]. Pada pengembangan Naviku akan dilakukan pengujian TTS menggunakan metode Word Error Rate (WER) yang bertujuan untuk mengukur akurasi pengucapan dari mesin TTS yang digunakan. Dalam pengembangan aplikasi Naviku, digunakan pendekatan Agile Software Development untuk mempermudah proses pengembangan serta menggunakan metode Test Driven Development (TDD). TDD merupakan metode yang mengedepankan pembuatan tes dan pengujian sebelum menulis kode implementasi. Developer harus menulis kode hingga tes berhasil dan tidak menghasilkan kesalahan atau error [11]. Dengan demikian developer dapat menentukan alur tes yang akan diujikan sebelum mengimplementasikan kode produksi dan mendapatkan kualitas kode yang baik.

2. METODOLOGI PENELITIAN

2.1 Tahap Analisis Perancangan Sistem Informasi

Tahap ini bertujuan untuk merancang kebutuhan dalam membangun sistem informasi pada aplikasi yang akan dirancang. Fokus analisis perancangan ini adalah perancangan sistem informasi pada aplikasi mobile berbasis Android. Pada Gambar 1 dijelaskan tahapan analisis yang dimulai dari analisa kebutuhan user, membuat usecase diagram, membuat usecase scenario, membuat activity diagram, membuat sequence diagram, dan membuat class diagram.



Gambar 1. Tahap Analisis

Journal of Information System Research (JOSH)
Volume 4, No. 4, Juli 2023, pp 1455−1463
ISSN 2686-228X (media online)
https://ejurnal.seminar-id.com/index.php/josh/
DOI 10.47065/josh.v4i4.3948

2.2 Tahap Perancangan

2.2.1 Agile Software Development

Berdasarkan kutipan dari website Agile Manifesto, agile merupakan pendekatan yang dipaparkan oleh Kent Beck bersama 16 rekan lainnya pada tahun 2001, dikenal dengan sebutan Agile Alliance. Kata "Agile" berasal dari bahasa Inggris yang berarti lincah, tangkas, atau cepat bergerak. Dalam konteks pengembangan software, agile merujuk pada sebuah konseptual yang menekankan untuk menciptakan serta merespon perubahan dengan cepat [12].

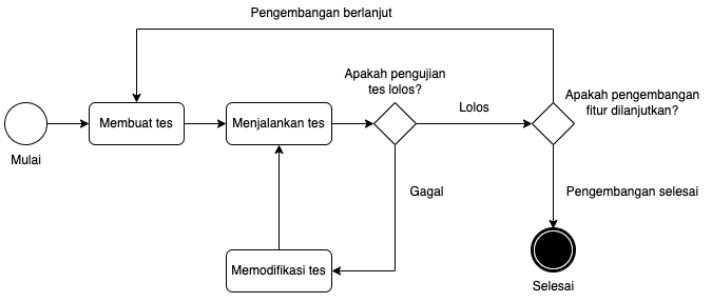
Pendekatan agile berfokus pada pengembangan software dengan menekankan pada iterasi dan inkrementalisme, yang memungkinkan tim developer dapat mengembangkan software secara bertahap dengan cara yang terorganisir dan terukur. Selain itu, pendekatan agile mengutamakan untuk bergerak cepat serta lincah dalam mengantisipasi perubahan yang mungkin terjadi, dan semua aktor yang terlibat di dalam proyek harus saling bekerja sama. Kelebihan utama menggunakan pendekatan agile dalam pengembangan software yaitu kualitas software yang dikembangkan bisa lebih baik, user dapat merasa lebih puas serta bisa ikut berkontribusi dengan aktif pada saat pengembangan, sangat fleksibel, dan dapat meningkatkan komunikasi serta kooridnasi antar tim developer [13].

2.2.2 Test Driven Development

Test Driven Development (TDD) merupakan metode yang menerapkan pendekatan agile, diperkenalkan oleh Kent Beck sekaligus muncul bersamaan dengan model proses agile, dengan menekankan proses yang bersifat iteratif, inkremental, dan evolusioner yang digunakan sejak tahun 1950-an [14]. Metode TDD berfokus pada pembuatan tujuan dari program yang dibuat, lalu membuat beberapa skenario ke dalam unit test yang berperan dalam menguji beberapa kasus yang mungkin dapat menghasilkan output error pada program [11].

Pada Gambar 2 [11], dijelaskan alur dalam pengembangan sistem menggunakan metode TTD. Metode TDD dimulai dari tahap membuat tes. Pada tahap ini, developer menetapkan tujuan mengenai fitur dan mengimplementasikannya ke dalam test function. Setelah itu dilakukan running pada tes dan dilihat hasil output-nya, apakah tes lolos atau gagal. Apabila hasil dari tes menunjukkan output gagal / error, maka developer akan memodifikasi atau melakukan refactor pada kode sampai hasil dari testing tidak menunjukkan output error. Sedangkan kalau hasil dari testing berhasil / success, maka fitur siap diimplementasikan. Metode TTD ini akan diulangi prosesnya terus menerus hingga semua fitur pada software sudah diujikan dengan nilai output berhasil.

Sebelum peneliti menggunakan metode TDD, terdapat beberapa referensi dan bukti yang sudah dilakukan oleh peneliti lain terkait pengembangan software menggunakan metode TDD.



Gambar 2. Alur Metode Test Driven Development (TDD)

Pada studi empiris perbandingan metode dengan pendekatan agile yang dilakukan oleh Soobia, Jhanjhi, Naqvi, dan Humayun (2019), ditunjukkan perbandingan antara metode Test Driven Development (TDD), Feature Driven Development (TFD), Crystal, Extreme Programming (XP), Dynamic System Development Method (DSDM), dan Lean Software Development (LSD). TDD dapat diimplementasikan pada ukuran proyek yang sedang hingga besar, lalu karakteristik yang menjadi unggulan TDD yaitu debugging yang cepat, kualitas yang lebih tinggi, serta sistem yang lebih teruji [15].

Penelitian oleh Jonathan dan Pakereng (2020) mengembangkan aplikasi Android untuk memantau COVID-19 dengan menggunakan Flutter dan menerapkan metode Test-Driven Development (TDD). Hasilnya menunjukkan fungsi dan fitur aplikasi terintegrasi dengan baik dan kode menjadi rapi berkat proses refactoring dan debugging [16].

Rizkyana (2021) melakukan penelitian tentang pengembangan software penyedia data dengan menggunakan arsitektur REST API dan menerapkan metode Test-First Development dari TDD. Hasilnya menunjukkan bahwa pembuatan test case sangat membantu dalam mengantisipasi kesalahan [17].

Moe (2019) melakukan perbandingan antara metode Test-Driven Development (TDD), Behavior-Driven Development (BDD), dan Acceptance Test-Driven Development (ATDD) dalam pengembangan software.

Journal of Information System Research (JOSH)
Volume 4, No. 4, Juli 2023, pp 1455−1463
ISSN 2686-228X (media online)
https://ejurnal.seminar-id.com/index.php/josh/
DOI 10.47065/josh.v4i4.3948

Hasilnya menunjukkan perbedaan fokus antara ketiga metode tersebut dalam aspek implementasi, perilaku sistem, target pengguna, dan skala proyek yang dihadapi [18].

2.2.3 Word Error Rate

Word Error Rate (WER) merupakan suatu ukuran evaluasi yang diterapkan dalam bidang pengenalan ucapan (speech recognition) dan pemrosesan bahasa alami. Tujuan dari metrik ini adalah untuk menilai seberapa tepat sistem pengenalan ucapan atau sistem pengenalan teks dalam mentransformasi ucapan menjadi teks tertulis.

Metode evaluasi WER telah banyak digunakan pada penelitian speech recognition, contohnya pada penelitian yang dilakukan oleh Hou dan lainnya (2020) dalam penelitian mereka yang berjudul "Large-Scale End-to-End Multilingual Speech Recognition and Language Identification with Multi-Task Learning", mereka menggunakan metode evaluasi WER untuk menghitung model multibahasa yang mereka buat dan angka WER yang didapatkan sebesar 52,8 [19].

Lalu ada juga penelitian yang dilakukan oleh Blackley dan lainnya (2019) dalam penelitian mereka yang berjudul "Speech recognition for clinical documentation from 1990 to 2018: a systematic review". Tujuan dari penelitian tersebut yaitu meninjau literatur terkini mengenai penggunaan teknologi speech recognition untuk dokumentasi klini serta untuk memahami dampak speech recognition pada akurasi dokumen, efisiensi penyedia layanan, biaya institusi, dan lainnya. Mereka melakukan pengujian menggunakan WER pada data yang telah dikumpulkan dan mendapatkan tingkat WER berkisar antara 7,4% hingga 38,7% [20].

WER dihitung dengan mengevaluasi seberapa baik sistem pengenalan ucapan atau teks dengan mengukur persentase kesalahan yang terjadi dalam hasil keluaran sistem dibandingkan dengan teks referensi yang benar. Semakin rendah nilai WER, semakin akurat sistem pengenalan tersebut [21].

$$\text{WER} = \frac{\text{S+I+D}}{\text{N}} \quad (1)$$

Pada rumus (1) dijelaskan terkait formula atau rumus menghitung nilai WER untuk mengukur persentase kata yang salah (Substitutions (S), Insertions (I), Deletions (D), Len Text (N)) [22].

3. HASIL DAN PEMBAHASAN

3.1 Functional Requirements

Functional requirements adalah persyaratan atau kebutuhan dalam pengembangan software yang berkaitan dengan fungsi-fungsi yang harus dilakukan oleh software tersebut. Fungsi-fungsi ini menjadi dasar bagi developer dalam merancang dan mengembangkan fitur-fitur dalam software. Tabel 1 menampilkan gambaran functional requirements pada aplikasi yang akan dibangun, termasuk deskripsi aktor sebagai calon pengguna aplikasi. Deskripsi aktor menjelaskan fungsi pengguna terhadap sistem, sementara fungsionalitas sistem berisi kemampuan yang ditawarkan kepada aktor.

Tabel 1. Functional Requirements

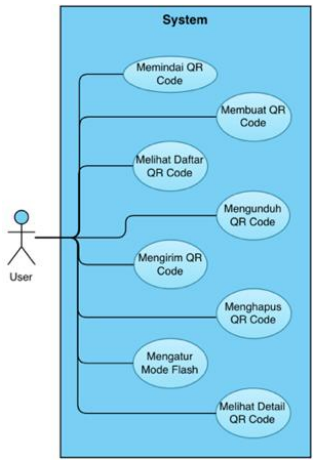
Aktor	Deskripsi Aktor	Fungsionalitas Sistem
User	User ini berperan sebagai pengguna dengan penyandang difabel netra yang melakukan navigasi dalam ruangan menggunakan software yang dirancang.	Sistem dapat menampilkan halaman utama Sistem dapat menampilkan menu navigasi bawah pada halaman utama Sistem dapat memindai QR Code Sistem dapat menampilkan hasil pindai berupa informasi teks Sistem dapat membaca hasil teks dalam bentuk suara Sistem dapat menampilkan QR Code yang telah disediakan untuk diunduh Sistem dapat membuat QR Code baru Sistem dapat menghapus QR Code yang ada Sistem dapat mengirim QR Code Sistem dapat mengatur mode flash kamera

3.2 Use Case Diagram

Use case diagram merupakan salah satu jenis Unified Modeling Language (UML) yang digunakan untuk menggambarkan atau memodelkan interaksi antara user dan sistem dalam sebuah software. Use case diagram berfokus pada system functional dari perspektif user serta digunakan untuk memvisualisasikan hubungan antara user dan system functional dalam suatu use case scenario. Pada Gambar 3, software yang dikembangkan terdapat

Journal of Information System Research (JOSH)
Volume 4, No. 4, Juli 2023, pp 1455−1463
ISSN 2686-228X (media online)
https://ejurnal.seminar-id.com/index.php/josh/
DOI 10.47065/josh.v4i4.3948

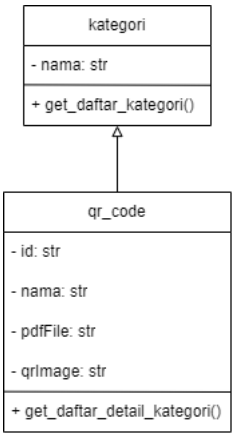
beberapa use case yang berinteraksi dengan user, yaitu memindai QR code, membuat QR code, melihat daftar QR code, mengunduh QR code, mengirim QR code, menghapus QR code, mengatur mode flash, dan melihat detail QR code.



Gambar 3. Use Case Diagram

3.3 Class Diagram

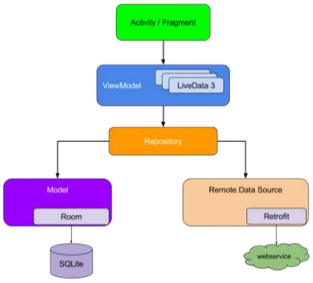
Class diagram merupakan salah satu jenis diagram yang digunakan dalam pemodelan berorientasi objek untuk menggambarkan struktur kelas, atribut, metode, dan hubungan dari setiap objek. Pada Gambar 4 merupakan class diagram dari aplikasi Naviku.



Gambar 4. Class Diagram

3.4 Model-View-ViewModel (MVVM) Architecture

Model-View-ViewModel (MVVM) merupakan sebuah architecture pattern yang digunakan dalam pengembangan aplikasi Android yang dirancang untuk memisahkan business logic dari user interface, sehingga memungkinkan pengembangan aplikasi yang lebih terstruktur, mudah diuji, serta mudah dipelihara.



Gambar 5. MVVM Architecture

Pada Gambar 5 merepresentasikan MVVM architecture yang diterapkan pada pengembangan aplikasi Naviku. Alur interaksi dalam MVVM architecture dimulai dari tindakan user di Activity / Fragment (View), kemudian tindakan tersebut dikirim ke ViewModel. ViewModel mengolah tindakan tersebut serta memperbarui

Journal of Information System Research (JOSH)
Volume 4, No. 4, Juli 2023, pp 1455−1463
ISSN 2686-228X (media online)
https://ejurnal.seminar-id.com/index.php/josh/
DOI 10.47065/josh.v4i4.3948

data yang didapatkan dari Repository (Model) jika diperlukan. Setelah data diperbarui, ViewModel akan memberi tahu View tentang perubahan yang terjadi, sehingga View secara otomatis memperbarui tampilan sesuai data yang baru. Repository dapat mengambil data dari dua sumber, yaitu dari local storage (room database) dan remote storage yang menggunakan tools Retrofit untuk pemanggilan API (Application Programming Interface).

3.5 Test Driven Development

Test driven development (TDD) diawali dengan membuat kode testing untuk fitur yang akan dibuat. Pada pengembangan aplikasi Naviku, testing akan dibuat untuk menguji fungsi yang ada di beberapa objek. Hasil testing akan gagal saat pertama kali dijalankan karena belum ada implementasi kode produksi. Unit test akan digunakan pada pengujian aplikasi Naviku. Unit test merupakan pengujian yang termasuk dalam kategori small test, yaitu spesifik pada suatu individual unit kode secara terisolasi dan berfokus pada pengujian fungsi, metode, maupun kelas tunggal.

3.1.1 Test Fails

Tahap pertama TDD yaitu test fails. Pada tahap ini ViewModel yang ada akan dibuat unit test-nya sebelum kode produksi diimplementasikan. Lalu pada Gambar 6 terdapat contoh salah satu unit test yang dibuat sebelum melakukan implementasi kode produksi. Apabila unit test dijalankan maka hasilnya akan error dikarenakan belum adanya kode produksi yang diimplementasikan. Hasil dari semua pengujian unit test ViewModel pada fase fails dapat dilihat pada Tabel 2.

Gambar 6. Contoh Unit Test Sebelum Implementasi Kode Produksi

Tabel 2. Hasil Pengujian Unit Test Pada Fase Test Fails

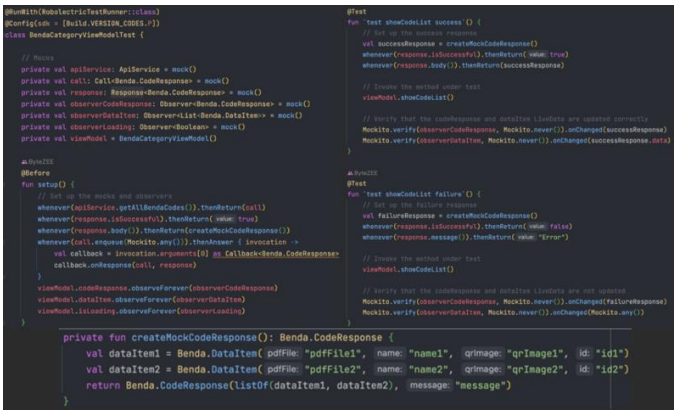
Object	Function	Duration	Result	Rate
PersonalCodeListViewModel	getAllPersonalCodes	3.048s	Failed	0%
AddCodeViewModel	insertCodeIntoDatabase	2.901s	Failed	0%
PersonalCodeDetailViewModel	deleteCode	2.959s	Failed	0%
RuanganCategoryViewModel	showCodeList success	1.445s	Failed	0%
	showCodeList failure	0.001s	Failed	0%
RuanganCodeDetailViewModel	showCodeDetail success	1.716s	Failed	0%
	showCodeDetail failure	0.001s	Failed	0%
BangunanCategoryViewModel	showCodeList success	1.406s	Failed	0%
	showCodeList failure	0s	Failed	0%
BangunanCodeDetailViewModel	showCodeDetail success	1.717s	Failed	0%
	showCodeDetail failure	0.002s	Failed	0%
BendaCategoryViewModel	showCodeList success	1.350s	Failed	0%
	showCodeList failure	0.001s	Failed	0%
BendaCodeDetailViewModel	showCodeDetail success	1.375s	Failed	0%
	showCodeDetail failure	0.001s	Failed	0%
RambuCategoryViewModel	showCodeList success	1.471s	Failed	0%
	showCodeList failure	0s	Failed	0%
RambuCodeDetailViewModel	showCodeDetail success	1.379s	Failed	0%
	showCodeDetail failure	0.001s	Failed	0%
AngkaCategoryViewModel	showCodeList success	1.523s	Failed	0%
	showCodeList failure	0.001s	Failed	0%
AngkaCodeDetailViewModel	showCodeDetail success	1.416s	Failed	0%
	showCodeDetail failure	0s	Failed	0%

Journal of Information System Research (JOSH)
Volume 4, No. 4, Juli 2023, pp 1455−1463
ISSN 2686-228X (media online)
https://ejurnal.seminar-id.com/index.php/josh/
DOI 10.47065/josh.v4i4.3948

3.1.2 Refactor

Setelah melakukan tahap test fails, pengujian berlanjut pada tahap refactor. Pada tahap ini dilakukan implementasi pada kode produksi, membuat fungsi / metode yang disesuaikan dengan fitur yang akan dibuat. Lalu melakukan refactor pada unit test untuk menyesuaikan agar hasil pengujian dapat berhasil. Pada Gambar 7 merupakan salah satu implementasi kode produksi pada ViewModel. Serta pada Gambar 8 merupakan salah satu refactoring pada kode unit test.

Gambar 7. Contoh Implementasi Kode Produksi Pada ViewModel



Gambar 8. Contoh Refactoring Kode Unit Test

3.1.3 Test Passes

Setelah fase refactor telah dilakukan, pengujian masuk ke tahap akhir yaitu test passes. Pada tahap ini akan dilakukan pengujian terhadap unit test yang telah di-refactor. Pada tahap ini juga memastikan bahwa hasil pengujian unit test mendapatkan status berhasil. Apabila hasil pengujian tetap gagal, maka pengujian akan mengulang pada fase refactor untuk memperbaiki baris kode yang menyebabkan kegagalan pada pengujian. Jika hasil pengujian berhasil atau sukses, maka proses pengujian dapat diakhiri atau melakukan iterasi kembali dari fase fails untuk menguji object yang lain. Pada Tabel 3 dipaparkan hasil dari semua pengujian unit test pada fase test passes.

Tabel 3. Hasil Pengujian Unit Test Pada Fase Test Passes

Object	Function	Duration	Result	Rate
PersonalCodeListViewModel	getAllPersonalCodes	3.172s	Passed	100%
AddCodeViewModel	insertCodeIntoDatabase	3.023s	Passed	100%
PersonalCodeDetailViewModel	deleteCode	3.313s	Passed	100%
RuanganCategoryViewModel	showCodeList success	5.263s	Passed	100%
	showCodeList failure	0.024s	Passed	100%
RuanganCodeDetailViewModel	showCodeDetail success	4.753s	Passed	100%
	showCodeDetail failure	0.018s	Passed	100%
BangunanCategoryViewModel	showCodeList success	4.887s	Passed	100%
	showCodeList failure	0.019s	Passed	100%
BangunanCodeDetailViewModel	showCodeDetail success	4.694s	Passed	100%
	showCodeDetail failure	0.018s	Passed	100%
BendaCategoryViewModel	showCodeList success	4.676s	Passed	100%
	showCodeList failure	0.021s	Passed	100%

Journal of Information System Research (JOSH)
Volume 4, No. 4, Juli 2023, pp 1455−1463
ISSN 2686-228X (media online)
https://ejurnal.seminar-id.com/index.php/josh/
DOI 10.47065/josh.v4i4.3948

Object	Function	Duration	Result	Rate
BendaCodeDetailViewModel	showCodeDetail success	5.049s	Passed	100%
	showCodeDetail failure	0.019s	Passed	100%
RambuCategoryViewModel	showCodeList success	5.060s	Passed	100%
	showCodeList failure	0.022s	Passed	100%
RambuCodeDetailViewModel	showCodeDetail success	5.476s	Passed	100%
	showCodeDetail failure	0.019s	Passed	100%
AngkaCategoryViewModel	showCodeList success	4.895s	Passed	100%
	showCodeList failure	0.020s	Passed	100%
AngkaCodeDetailViewModel	showCodeDetail success	4.931s	Passed	100%
	showCodeDetail failure	0.022s	Passed	100%

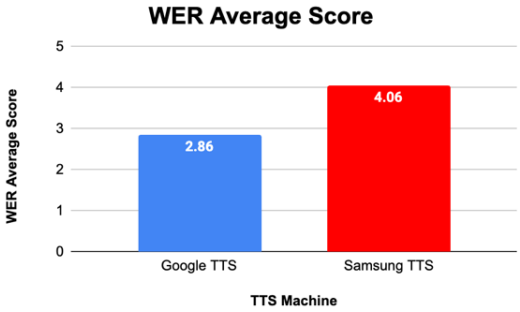
3.6 Word Error Rate

Pada pengujian Text-to-Speech (TTS) menggunakan metode evaluasi Word Error Rate (WER) dimulai dari pembuatan reference text (teks referensi) yang akan menjadi patokan penilaian dari teks yang dihasilkan berupa generated text (teks hasil). Generated text didapatkan dari proses pengujian menggunakan aplikasi TTS dan Speech-to-Text (STT), output suara yang dihasilkan dari aplikasi TTS akan ditangkap dan dikonversi kembali menjadi teks oleh aplikasi STT agar hasilnya dapat dibandingkan dengan reference text. Pada pengujian ini menggunakan 50 sampel reference text bahasa Indonesia, lalu pengujian sistem TTS menggunakan perangkat smartphone Samsung A14 dengan dua jenis mesin TTS, yaitu Google TTS dan Samsung TTS. Mengenai aplikasi STT dijalankan pada Android Studio langsung. Kode pengujian WER dapat dilihat pada Gambar 9, serta hasil dari pengujian WER dipaparkan pada Gambar 10.

```
import nltk
nltk.download('punkt')
from nltk import edit_distance

def tokenize_text(text):
  return nltk.word_tokenize(text.lower())

def calculate_wer(reference_text, generated_text):
    reference_tokens = tokenize_text(reference_text)
    generated_tokens = tokenize_text(generated_text)

    distance = edit_distance(reference_tokens, generated_tokens)
    wer = distance / len(reference_tokens) * 100
    return wer

reference_text = "Di taman ada berbagai macam bunga berwarna-warni"
generated_text = "Di taman ada ada berbagai macam bunga berwarna-warni"

wer_score = calculate_wer(reference_text, generated_text)
print(f"Word Error Rate (WER): {wer_score:.2f}%")
```

```
Word Error Rate (WER): 14.29%
[nltk_data] Downloading package punkt to /root/nltk_data...
[nltk_data]   Package punkt is already up-to-date!
```

Gambar 9. Kode Pengujian Word Error Rate



Gambar 10. Hasil Akhir Pengujian Word Error Rate

Dari hasil pengujian Word Error Rate (WER) yang telah dipaparkan pada Gambar 10, didapatkan WER average score (hasil rata-rata) yaitu untuk Google TTS mencapai angka 2,86 dan untuk Samsung TTS mencapai angka 4,06. Nilai tersebut didapatkan dengan menjumlah nilai WER dan totalnya dibagi dengan 50, karena sampel kalimat yang diujikan berjumlah 50 sampel. Dapat disimpulkan bahwa mesin Google TTS memiliki tingkat WER yang kecil dibandingkan dengan mesin Samsung TTS.

4. KESIMPULAN

Aplikasi Naviku telah berhasil dirancang dan dikembangkan untuk penyandang difabel netra dengan menggunakan metode Test Driven Development (TDD). Beberapa fitur dari segi fitur penyimpanan yang dikembangkan berdasarkan kebutuhan difabel netra, yaitu fitur mengakses, membuat, serta menghapus kode QR pribadi, dan melihat daftar serta detail dari kode QR yang ada pada aplikasi Naviku. Fitur yang disebutkan sebelumnya dibuat melalui 3 tahapan, yaitu test fails, refactor, dan test passes. Pada tahap terakhir yaitu test passes, unit test yang diujikan mendapatkan result success dan rate mencapai angka 100% untuk semua fitur berdasarkan data yang

Journal of Information System Research (JOSH)
Volume 4, No. 4, Juli 2023, pp 1455−1463
ISSN 2686-228X (media online)
https://ejurnal.seminar-id.com/index.php/josh/
DOI 10.47065/josh.v4i4.3948

dipaparkan pada Tabel 3. Lalu Text-to-Speech (TTS) telah berhasil diimplementasikan ke dalam aplikasi Naviku dengan menggunakan library TTS dari Kotlin. Lalu developer melakukan pengujian terhadap TTS menggunakan smartphone dan mesin TTS dari Google TTS dan Samsung TTS, serta mengguanakan 50 sampel reference text bahasa Indonesia. Pengujian tersebut menggunakan metode evaluasi Word Error Rate (WER) dan hasilnya mesin TTS dari Google TTS memiliki angka WER yaitu 2,86 dan lebih kecil dari angka WER dari mesin TTS dari Samsung TTS yang mendapatkan angka WER 4,06.

REFERENCES

[1] World Health Organization, “Blindness and vision impairment,” who.int, 2022. https://www.who.int/news-room/fact-sheets/detail/blindness-and-visual-impairment (diakses 5 Januari 2023).
[2] V. Yulaswati, F. Nursyamsi, M. N. Ramadhan, H. Palani, dan E. Kurnia Yazid, “PENYANDANG DISABILITAS INDONESIA : ASPEK SOSIOEKONOMI DAN YURIDIS,” Jakarta Pusat, 2021.
[3] Z. A. Nor Hisham, M. A. Faudzi, A. A. Ghapar, dan F. A. Rahim, “A Systematic Literature Review of the Mobile Application for Object Recognition for Visually Impaired People,” dalam 2020 8th International Conference on Information Technology and Multimedia, ICIMU 2020, Institute of Electrical and Electronics Engineers Inc., Agu 2020, hlm. 316–322. doi: 10.1109/ICIMU49871.2020.9243523.
[4] Z. Bauer, A. Dominguez, E. Cruz, F. Gomez-Donoso, S. Orts-Escolano, dan M. Cazorla, “Enhancing perception for the visually impaired with deep learning techniques and low-cost wearable sensors,” Pattern Recognit Lett, vol. 137, hlm. 27–36, Sep 2020, doi: 10.1016/j.patrec.2019.03.008.
[5] B. Kuriakose, R. Shrestha, dan F. E. Sandnes, “Tools and Technologies for Blind and Visually Impaired Navigation Support: A Review,” IETE Technical Review, vol. 39, no. 1, hlm. 3–18, Jan 2022, doi: 10.1080/02564602.2020.1819893.
[6] Z. Başgöze, J. Gualtieri, M. T. Sachs, dan E. A. Cooper, “Navigational Aid Use by Individuals with Visual Impairments,” 2020.
[7] H. Dong, J. Schafer, Y. Tao, dan A. Ganz, “PERCEPT-V: Integrated Indoor Navigation System for the Visually Impaired using Vision-Based Localization and Waypoint-Based Instructions,” 2020.
[8] S. Real dan A. Araujo, “Navigation systems for the blind and visually impaired: Past work, challenges, and open problems,” Sensors (Switzerland), vol. 19, no. 15. MDPI AG, 1 Agustus 2019. doi: 10.3390/s19153404.
[9] G. A. Manu dan P. L. Masan, “APLIKASI TEXT TO SPEECH UNTUK MENINGKATKAN PEMBELAJARAN BAHASA INGGRIS BAGI SISWA DISABILITAS,” 2020.
[10] R. Adriati, H. Tolle, dan O. Setyawati, “Pengembangan Aplikasi Text-to-Speech Bahasa Indonesia Menggunakan Metode Finite State Automata Berbasis Android,” 2016.
[11] I. B. Kerthyayana Manuaba, “Combination of Test-Driven Development and Behavior-Driven Development for Improving Backend Testing Performance,” Procedia Comput Sci, vol. 157, hlm. 79–86, 2019, doi: 10.1016/j.procs.2019.08.144.
[12] Agile Manifesto, “Manifesto for Agile Software Development,” agilemanifesto.org. https://agilemanifesto.org/ (diakses 26 Desember 2022).
[13] S. Alsaqqa, S. Sawalha, dan H. Abdel-Nabi, “Agile Software Development: Methodologies and Trends,” International Journal of Interactive Mobile Technologies (iJIM), vol. 14, no. 11, hlm. 246, Jul 2020, doi: 10.3991/ijim.v14i11.13269.
[14] D. Janzen dan H. Saiedian, “Test-driven development concepts, taxonomy, and future direction,” Computer (Long Beach Calif), vol. 38, no. 9, hlm. 43–50, Sep 2005, doi: 10.1109/MC.2005.314.
[15] S. Soobia.et.al., “Analysis of Software Development Methodologies,” International Journal of Computing and Digital Systems, vol. 8, no. 5, hlm. 445–460, Jan 2019, doi: 10.12785/ijcds/080502.
[16] F. Jonathan dan M. A. I. Pakereng, “Test-Driven Development pada Pengembangan Aplikasi Android untuk Memantau COVID-19,” IJCIT (Indonesian Journal on Computer and Information Technology), vol. 6, no. 1, 2020, Diakses: 27 Desember 2022. [Daring]. Tersedia pada: https://scholar.archive.org/work/xa7yr6ys6zdfhkvfjsnjxqusn4/access/wayback/https://ejournal.bsi.ac.id/ejurnal/index.php/ijcit/article/download/9502/pdf
[17] M. A. Rizkyana, “Implementasi Unit Testing menggunakan metode Test-First Development,” MULTINETICS, vol. 7, no. 1, hlm. 37–47, Mei 2021, doi: 10.32722/multinetics.v7i1.3525.
[18] M. M. Moe, “Comparative Study of Test-Driven Development (TDD), Behavior-Driven Development (BDD), and Acceptance Test–Driven Development (ATDD),” hlm. 231–234, 2019, [Daring]. Tersedia pada: http://creativecommons.org/licenses/by/4.0
[19] W. Hou, Y. Dong, B. Zhuang, L. Yang, J. Shi, dan T. Shinozaki, “Large-Scale End-to-End Multilingual Speech Recognition and Language Identification with Multi-Task Learning,” dalam Interspeech 2020, ISCA: ISCA, Okt 2020, hlm. 1037–1041. doi: 10.21437/Interspeech.2020-2164.
[20] S. V Blackley, J. Huynh, L. Wang, Z. Korach, dan L. Zhou, “Speech recognition for clinical documentation from 1990 to 2018: a systematic review,” Journal of the American Medical Informatics Association, vol. 26, no. 4, hlm. 324–338, Apr 2019, doi: 10.1093/jamia/ocy179.
[21] V. E. Budiman dan A. Widjaja, “Building Acoustic and Language Model for Continuous Speech Recognition in Bahasa Indonesia,” Jurnal Teknik Informatika dan Sistem Informasi, vol. 6, no. 2, Agu 2020, doi: 10.28932/jutisi.v6i2.2684.
[22] R. Errattahi, A. El Hannani, dan H. Ouahmane, “Automatic Speech Recognition Errors Detection and Correction: A Review,” Procedia Comput Sci, vol. 128, hlm. 32–37, 2018, doi: 10.1016/j.procs.2018.03.005.