IT-EXPLORE
Jurnal Penerapan Teknologi Informasi dan Komunikasi
ISSN 2828-7940, e-ISSN 2829-1727, DOI: 10.24246/itexplore.v5i1.2026.pp50-64
Volume 05 Nomor 01 Tahun 2026
Journal homepage: https://ejournal.uksw.edu/itexplore

Perancangan sistem manajemen *session real-time* menggunakan firestore, python, dan javascript

Richard Alvin Munandar [1], Pratyaksa Ocsa Nugraha Saian [2]
[1,2] Program Studi Teknik Informatika, Fakultas Teknologi Informasi, Universitas Kristen Satya Wacana, Salatiga, Jawa Tengah, Indonesia

Informasi Artikel

Riwayat Artikel:

Dikirim:
16 Mei 2025
Direvisi:
20 Juni 2025
Diterima:
20 September 2025

Kata Kunci:

Manajemen *Session*
Real-Time,
Firestore,
Python,
Javascript,
Black Box Testing

Keywords:

Real-Time Session Management,
Firestore,
Python,
Javascript,
Black Box Testing

This is an open access article under the CC BY-SA license

ABSTRAK

PT. XYZ merupakan salah satu perusahaan retail terbesar di Indonesia. Perusahaan ini memiliki aplikasi berbasis *website* yang berfungsi untuk membantu karyawan-karyawan yang bekerja dalam bidang IT Support untuk menangani masalah-masalah operasional. Akan tetapi, *website* ini belum menerapkan sistem manajemen *session* yang di mana dapat mempengaruhi tingkat keamanan. Maka dari itu, dirancanglah suatu sistem untuk manajemen *session* secara *real-time* menggunakan Firestore, Python, dan Javascript untuk memenuhi kebutuhan dari perusahaan. Firestore digunakan sebagai *database* untuk menyimpan *session* serta Firestore memiliki fitur yang bernama Firestore *snapshot listener* yang berfungsi untuk mendeteksi perubahan secara *real-time*. Metode penelitian mencakup identifikasi kebutuhan, perancangan awal *prototype*, pembuatan *prototype,* evaluasi *prototype,* penyempurnaan *prototype,* dan implementasi sistem. *Black Box Testing* digunakan sebagai metode untuk pengujian sistem ini, yang di mana hasilnya sudah sesuai dengan kebutuhan serta dilakukan wawancara dengan *admin* dari *website* ini untuk menguji sistem manajemen *session* dan mengetahui respon dari *admin* terkait sistem manajemen *session* yang telah dirancang.

ABSTRACT

PT. XYZ is one of the largest retail companies in Indonesia. This company has a website-based application that functions to help employees who work in IT Support to handle operational problems. However, this website has not implemented a session management system which can affect the level of security. Therefore, a system design was created for real-time session management using Firestore, Python, and Javascript. Firestore is used as a database to store sessions and Firestore has a feature called the Firestore snapshot listener which functions to detect changes in real-time. The research method includes identifying needs, initial prototype design, prototype creation, prototype evaluation, prototype refinement, and system implementation. Black Box Testing is used as a method for testing this system, where the results are in accordance with the needs and interviews are conducted with the admin of this website to test the session management system and find out the admin's response regarding the designed session management system.

Corresponding Author:

Richard Alvin Munandar
Fakultas Teknologi Informasi, Universitas Kristen Satya Wacana
Jl. Diponegoro No.52-60, Sidorejo, Salatiga, Jawa Tengah, Indonesia
Email: 672021057@student.uksw.edu

1. PENDAHULUAN

PT. XYZ merupakan salah satu perusahaan retail terbesar di Indonesia yang sudah berdiri kurang lebih 30 tahun ini. Perusahaan ini memiliki cabang toko yang tersebar luas pada berbagai daerah, bahkan hingga ke luar negeri. PT. XYZ memiliki aplikasi berbasis *website* yang disebut Dashboard IT Support. Aplikasi ini bertujuan untuk membantu karyawan-karyawan yang bekerja dalam bidang IT Support dalam menangani masalah-masalah yang terjadi pada operasional.

Akan tetapi, *website* perusahaan ini belum menerapkan sistem untuk manajemen *session*. Belum adanya sistem manajemen session ini dapat mempengaruhi tingkat keamanan, terutama dalam hal validasi perubahan hak akses secara *real-time*, yang dimana perubahan hak akses terjadi ketika pengguna sedang login tidak akan langsung berubah dikarenakan session hanya akan berubah saat *session* yang lama *expired*. Selain itu, pengguna masih bisa login ke *website* ini dengan lebih dari satu perangkat secara bersamaan. Hal tersebut menjadi celah keamanan apabila satu akun dipakai oleh beberapa orang selain pemilik akun tersebut.

Dengan demikian, PT. XYZ ini ingin membuat sistem manajemen *session* yang bersifat *real-time* untuk mengatasi permasalahan tersebut. Sistem ini memiliki fitur untuk mendeteksi *session* dari pengguna yang lebih dari satu. Ketika ditemukan jumlah *session* lebih dari satu dari pengguna yang sama, maka akan dihapus secara langsung sehingga pengguna hanya dapat melakukan login di satu perangkat saja.

Supaya sistem manajemen *session* dapat melakukan hal tersebut, maka digunakanlah basis data Firestore. Firestore merupakan salah satu servis basis data bagian dari Firebase. Basis data Firestore memiliki latensi yang rendah dan efisien sehingga sinkronisasi data dapat dilakukan secara *real-time* [1]. Basis data Firestore ini memiliki fitur *listener* yang dapat langsung terhubung dengan *client* [2]. Dengan demikian, tidak perlu menggunakan Javascript *timer* untuk terus mengirim permintaan mengenai pengecekan *session* ke database, tetapi bisa langsung memanfaatkan fitur *listener* milik Firestore [3].

Dalam perancangan sistem manajemen *session* ini, digunakan dua bahasa pemrograman, yaitu Python dan Javascript. Kedua bahasa tersebut telah didukung oleh pustaka resmi (*library*) yang terintegrasi dengan Firestore, sehingga memudahkan pengembangan dan pemanfaatan berbagai fitur yang disediakan oleh Firestore, termasuk fitur *real-time* dan manajemen dokumen. Python berperan sebagai bahasa pemrograman di sisi *backend*, yang bertugas untuk melakukan validasi terhadap data dari *frontend*, khususnya dalam proses autentikasi akun pengguna, serta menangani pengelolaan dan manipulasi data pada Firestore. Lalu Javascript digunakan di sisi *frontend* untuk mengirim data dari pengguna ke *backend* serta melakukan validasi terhadap berbagai *event* yang terjadi di sisi pengguna.

Setiap perubahan data yang dilakukan oleh *backend* (Python) akan langsung terdeteksi oleh *frontend* secara *real-time* melalui Firestore, dengan memanfaatkan fitur *real-time listener* yang diimplementasikan menggunakan Javascript. Hal tersebut memungkinkan antarmuka pengguna menampilkan perubahan data secara langsung tanpa perlu melakukan penyegaran halaman, sehingga meningkatkan responsivitas dan pengalaman pengguna secara keseluruhan. Selain itu, menggunakan bahasa pemrograman Python karena PT. XYZ menggunakan Python untuk seluruh proses bisnisnya, serta menggunakan Javascript karena *website* Dashboard IT Support ini dirancang menggunakan Javascript.

2. KAJIAN PUSTAKA

2.1. State of The Art

Menurut hasil penelitian dari Riswanda dan Priandika [4], didapatkan hasil bahwa dengan dibuatnya aplikasi sistem manajemen pemesanan barang dapat mempermudah dalam memberikan informasi serta pengaturan stok barang. Kemudian hasil penelitian menurut Puspitasari, et al. [5], yang merancang aplikasi sistem informasi manajemen perpustakaan, didapatkan hasil bahwa aplikasi tersebut dapat membantu pihak perpustakaan dalam hal pengelolaan data buku serta transaksi. Dengan demikian, dapat disimpulkan bahwa perancangan sistem manajemen *session* ini dapat membantu pekerja di PT. XYZ dalam pengelolaan *session* dari pengguna yang sedang mengakses *website.*

Menurut hasil penelitian dari Maulana [1] dalam perancangan aplikasi E-Tilang berbasis *mobile*, didapatkan hasil bahwa dengan memanfaatkan salah satu komponen dari Firebase yang bernama Firestore, aplikasi dapat bekerja secara *real-time* karena aplikasi tersebut membutuhkan kecepatan dan keakuratan untuk mengirim data penilangan. Kemudian, menurut hasil penelitian dari Napitupulu dan Nugraha [6] yang

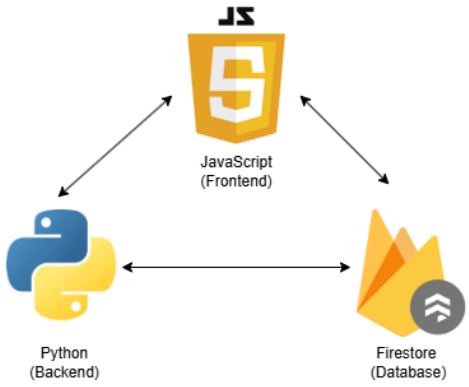
merancang aplikasi sistem parkir pintar menggunakan Firebase, didapatkan hasil bahwa dengan memanfaatkan Firebase Firestore dapat meningkatkan keamanan privasi, serta memberikan tingkat kecepatan dalam mengelola data yang tinggi karena langsung terhubung dengan jaringan *internet*. Dengan demikian, penggunaan Firestore dalam perancangan sistem manajemen *session* merupakan keputusan yang tepat karena Firestore memiliki tingkat kecepatan yang tinggi serta dapat melakukan sinkronisasi data secara *real-time.*

Menurut Sinlae, et al. [7], dikatakan bahwa Javascript pada umumnya diimplementasikan dalam pembuatan animasi dan berbagai elemen interaktif pada *website*, serta terdapat banyak *library* yang dapat digunakan dalam membuat *website* supaya menjadi lebih interaktif. Pernyataan tersebut sejalan dengan hasil penelitian [8] yang merancang *game* Pacman menggunakan Javascript, bahwa Javascript dapat memberikan interaktivitas dan responsivitas yang baik. Oleh karena itu, Javascript digunakan sebagai *frontend* yang dinamis dan interaktif untuk sistem manajemen *session*.

Menurut hasil penelitian dari Thangarajah [9], didapatkan hasil bahwa bahasa pemrograman Python merupakan bahasa pemrograman yang sangat populer di dunia dengan fitur-fiturnya seperti cepat, mudah, kuat, sederhana, dan *open source*, serta banyaknya *library* yang dapat digunakan dalam pemrograman *website*. Penelitian tersebut sejalan dengan penelitian dari Patkar, et al. [10], yang dimana bahasa pemrograman Python menyediakan berbagai *library* yang dapat mempermudah perancangan aplikasi *website*, serta Python juga dapat berperan sebagai *web server* dengan bantuan *library*. Maka dari itu, Python akan digunakan dalam perancangan sistem manajemen *session* ini sebagai *backend.*

Terdapat penelitian yang dilakukan oleh Ulah dan Suartana [11] yang mengimplementasikan manajemen *session* pada *website* magang menggunakan teknologi MERN dengan pendekatan *express-session* dan penyimpanan *session* di MongoDB. Sistem tersebut berhasil menangani autentikasi dan otorisasi pengguna dengan baik melalui penggunaan *token* JWT dan *session id* yang disimpan di sisi *server*, serta *cookie* di sisi *client*. Namun pendekatan ini masih bersifat tidak *real-time*, karena status dari *session* hanya diperiksa ketika permintaan (*request*) dikirim ke *server*. Sebagai perbandingan, penelitian ini menggunakan Firestore sebagai media penyimpanan *session* karena Firestore menyediakan fitur sinkronisasi data secara *real-time*. Dengan demikian, sistem dapat langsung mendeteksi perubahan *session* tanpa memerlukan konfigurasi *WebSocket* tambahan, yang menjadikan penggunaan Firestore ini lebih efisien untuk kebutuhan sistem dengan pembaruan status *session* secara *real-time*.

Oleh karena itu, berdasarkan studi literatur yang telah dibahas, diagram arsitektur dari sistem manajemen session dapat digambarkan sebagai berikut:



Gambar 1: Diagram arsitektur sistem manajemen *session*

Penjelasan pada gambar 1 adalah Firestore, Python, dan Javascript saling terkait satu dengan yang lainnya. Firestore sebagai *database* serta penyedia fitur *real-time*, Python sebagai *backend,* dan Javascript sebagai *frontend.* Hubungan antara *backend* dengan Firestore adalah *backend* untuk membuat, menyimpan, dan memperbarui *session* di Firestore. Hubungan antara *backend* dengan *frontend* yaitu sebagai sistem untuk berkomunikasi terkait adanya pembuatan *session* (ketika *user login*) atau perubahan pada *session* (ketika *user logout* atau *double login* atau terjadi perubahan hak akses). Hubungan antara *frontend* dengan Firestore yaitu sebagai pemberi notifikasi kepada *user* bila terjadi perubahan pada *session* (*Double login* atau perubahan hak akses).

Firestore sebagai database serta penyedia fitur real-time, Python sebagai backend, dan Javascript sebagai frontend. Hubungan antara backend dengan Firestore adalah backend untuk membuat, menyimpan, dan memperbarui session di Firestore. Hubungan antara backend dengan frontend yaitu sebagai sistem untuk berkomunikasi terkait adanya pembuatan session (ketika user login) atau perubahan pada session (ketika user logout atau double login atau terjadi perubahan hak akses). Hubungan antara frontend dengan Firestore yaitu sebagai pemberi notifikasi kepada user bila terjadi perubahan pada session (Double login atau perubahan hak akses).

2.2. Landasan Teori

Session merupakan suatu metode untuk menyimpan informasi pada komputer di *server* komputer yang dimana informasi tersebut dapat digunakan di berbagai halaman *website,* termasuk halaman yang sedang aktif [12]. Kemudian sistem informasi manajemen merupakan sistem terintegrasi yang melibatkan manusia maupun mesin, yang dirancang untuk menyediakan informasi yang dapat membantu dalam kegiatan operasional, manajemen, dan pengambilan keputusan [13]. Dengan demikian, sistem manajemen *session* merupakan suatu sistem yang dirancang untuk membantu manusia dalam memanajemen *session*, baik dalam hal membuat dan menghapus.

Firestore merupakan sebuah basis data *NoSQL* berbasis *cloud* tanpa *server* dengan kemampuan *real-time.* Firestore merupakan salah satu servis dari Firebase. Fitur-fitur dari Firestore diantara lain penyimpanan, sinkronisasi *real-time*, penskalaan secara otomatis, mendukung mode *offline*, serta dapat mengambil, menyortir, dan menyaring data [14]. Firestore memiliki fitur *listener* yang dapat terhubung langsung dengan *Client* menggunakan Javascript, yaitu dengan menggunakan *function* dari Firestore yang bernama ‘onSnapshot’ [3].

Python merupakan bahasa pemrograman tingkat tinggi yang bersifat umum dan dijalankan melalui proses interpretasi. Identasi menjadi cara pada Python untuk membaca kode. Python memiliki sifat dinamis dan dapat mengelola memori secara otomatis. Struktur bahasa dan pendekatan berbasis objek dari Python dirancang untuk mempermudah *programmer* dalam menulis kode yang mudah dipahami, baik untuk proyek berskala kecil maupun besar. Python juga memiliki banyak *library* yang siap digunakan sesuai kebutuhannya [10].

Javascript merupakan bahasa pemrograman untuk membuat *website*. JavaScript bersifat *client-side programming language* yang berarti pemrosesan dari bahasa pemrograman ini dilakukan oleh *client* melalui aplikasi *web browser*. JavaScript digunakan untuk mengendalikan interaksi antara halaman *web* dengan *client* [15].

3. METODE PENELITIAN

3.1. Tahapan Penelitian

Metode yang digunakan dalam tahapan penelitian ini yaitu metode *Research and Development* (R&D). Metode R&D merupakan metode untuk merancang produk yang layak untuk digunakan melalui proses pengujian telebih dahulu [16]. Terdapat lima tahapan, yaitu *analysis*, *design*, *development*, *implementation*, dan *evaluation* [17]. Pada tahap pertama (*Analysis*), penganalisaan terkait masalah yang ingin dipecahkan dilakukan. Selain itu, dilakukan wawancara secara langsung atau tatap muka dengan pekerja di PT. XYZ. Pertanyaan-pertanyaannya mencakup masalah mengenai kebutuhan dari sistem manajemen *session*, serta pertanyaan-pertanyaan yang akan diajukan bersifat terbuka. Selain itu, pencarian artikel jurnal terkait topik penelitian akan dilakukan dengan tujuan untuk memperkuat landasan teori dalam penelitian.

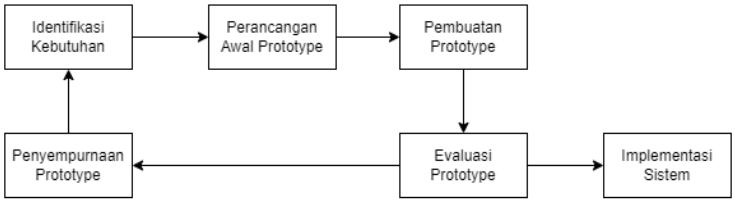
Selanjutnya pada tahap kedua (*Design*), akan dilakukan proses pengembangan awal sistem, yaitu membuat diagram alur proses sistem dan menentukan teknologi-teknologi yang akan digunakan dalam perancangan sistem manajemen session. Kemudian pada tahap ketiga (*Development*), perancangan sistem pun dimulai, yaitu membuat *prototype* dari sistem serta pengujian sistem akan dilakukan untuk memastikan bahwa sistem dapat bekerja dengan tepat.

Lalu pada tahap keempat (*Implementation*), proses *deploy* akan dilakukan terhadap sistem manajemen *session* di *staging website* Dashboard IT Support. Sistem akan mulai digunakan oleh seluruh pengguna di lingkungan *staging* dengan tujuan untuk uji coba secara keseluruhan. Pada tahap terakhir (*Evaluation*), evaluasi terkait sistem manajemen *session* akan dilakukan. Kemudian, kesimpulan dan saran pun akan diberikan berdasarkan hasil dari evaluasi.

3.2. Metode Pengembangan Sistem

Dalam penelitian ini, metode pengembangan sistem yang akan digunakan adalah metode *prototype.* Metode *prototype* merupakan metode yang melibatkan pengembang dan pengguna untuk komunikasi dalam membuat aplikasi [18][19]. Dalam metode *prototype*, pengembang dan pengguna akan terus berkomunikasi selama proses pembuatan program berlangsung, yang di mana pengembang mendapat masukkan dari pengguna untuk memperbaiki program *prototype* [19].

Pada metode *Prototype* ini, terdapat beberapa tahapan seperti pada Gambar 2:



Gambar 2: Tahapan Metode *Prototype* [18]

Tahap pertama merupakan identifikasi kebutuhan. Tujuan dari tahap ini adalah mengumpulkan kebutuhan dari pengguna, kemudian menganalisa sistem sesuai kebutuhan. Tahap kedua merupakan perancangan awal *prototype*. Dalam tahap ini, pengembang memberikan gambaran umum mengenai sistem kepada pengguna dengan menampilkan desain antarmuka yang sederhana dan alur dari penggunaan sistem.

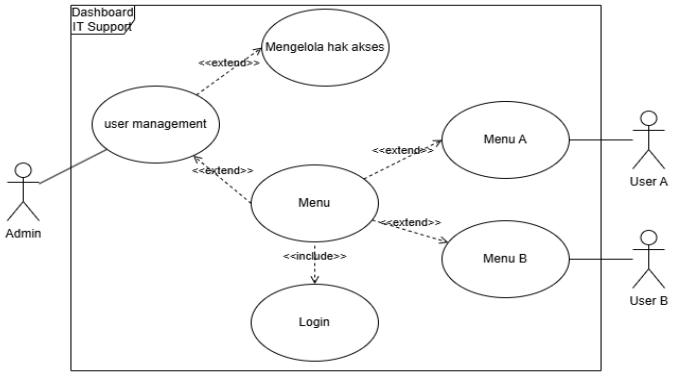
Tahap ketiga merupakan pembuatan *prototype.* Tahap ini merupakan pengembangan awal sistem berdasarkan desain awal dari tahap kedua. Fokus pada tahap ini yaitu menyelesaikan antarmuka pengguna dan fungsi dasar dari sistem. Tahap keempat adalah evaluasi *prototype*. Dalam tahap ini, pengguna akan memberikan umpan balik terkait *prototype* dan pengembang akan mencatat hasil umpan baliknya. Pemeriksaan dan iterasi akan dilakukan pada tahap ini. Bila terdapat evaluasi, maka masuk ke tahap penyempurnaan *prototype* dan bila tidak, maka masuk ke tahap berikutnya yaitu implementasi sistem.

Tahap kelima yaitu penyempurnaan *prototype.* Tujuan pada tahap ini adalah untuk memperbaiki *prototype* berdasarkan umpan balik dari pengguna. Tahap terakhir yaitu implementasi sistem. Pada tahap ini, *prototype* dinyatakan sudah sesuai dengan kebutuhan pengguna. Dengan demikian, *prototype* dapat digunakan sebagai landasan untuk pengembangan sistem yang lebih lanjut.

4. HASIL DAN PEMBAHASAN

4.1. Desain Sistem

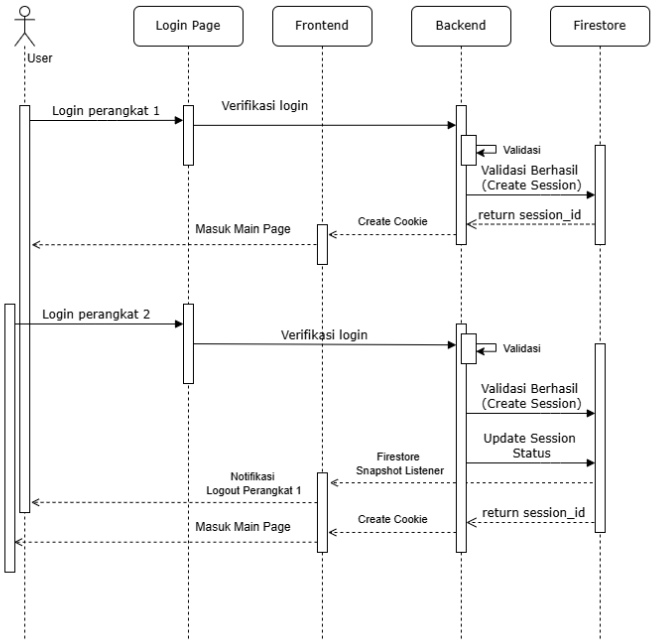
Desain sistem akan dibuat menggunakan *Unified Modelling Language* (UML), yaitu *use case diagram*, *sequence diagram, activity diagram*. *Use case diagram* digunakan untuk menggambarkan interaksi antara pengguna dengan sistem secara keseluruhan. *Sequence diagram* digunakan untuk memberikan gambaran mengenai alur dari pengguna ketika melakukan *login* di dua perangkat yang berbeda dan ketika *admin* merubah hak akses. *Activity diagram* digunakan untuk menggambarkan alur *login* disatu perangkat, alur *login* di perangkat lainnya, dan alur perubahan hak akses oleh *admin*.



Gambar 3: Use Case Diagram dari sistem Dashboard IT Support

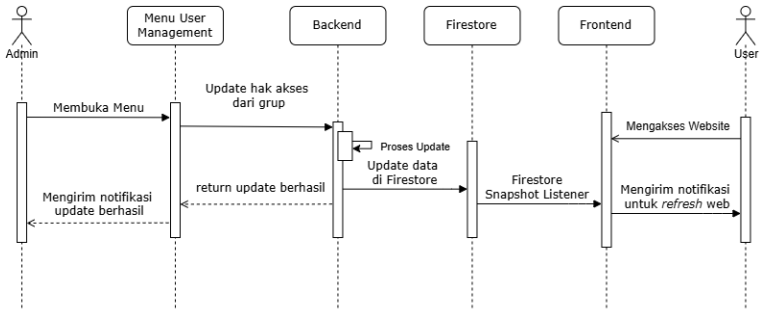
Gambar 3 menjelaskan mengenai *use case diagram* dari sistem Dashboard IT Support. Sistem ini memiliki dua puluh lima fitur serta enam belas *group,* di mana setiap *group* berisi lebih dari satu kode jabatan. Tujuan pembentukkan *group* ini adalah untuk mempermudah *admin* dalam mengelola hak akses setiap *user*. Setiap *group* tentunya memiliki akses menu yang berbeda dan sistem ini bersifat dinamis, yang berarti setiap *group* (jabatan) dapat mengalami perubahan akses, seperti menu A dapat diakses oleh aktor A, tetapi aktor B tidak bisa mengaksesnya karena tidak memiliki akses. Lalu bila *admin* memberikan akses untuk *group* B supaya dapat mengakses menu A, maka *user* B pun dapat mengakses menu A. Oleh karena itu, supaya alur sistem dapat digambarkan dengan rapi, digunakanlah nama alias sebagai permisalan untuk beberapa aktor, seperti aktor A dan aktor B yang menunjukkan bahwa aktor A berasal dari *group A* dan aktor B dari *group B.*

Setiap *user* yang ingin mengakses *website* ini harus melakukan *login* terlebih dahulu. Kemudian terdapat aktor *admin* yang memiliki akses ke menu *user management*. Menu ini berfungsi untuk mengelola *users,* seperti mendaftarkan *user* baru, *update user*, dan lainnya. Selain itu, menu ini juga digunakan oleh *admin* untuk mengelola hak akses menu dari setiap *group.*



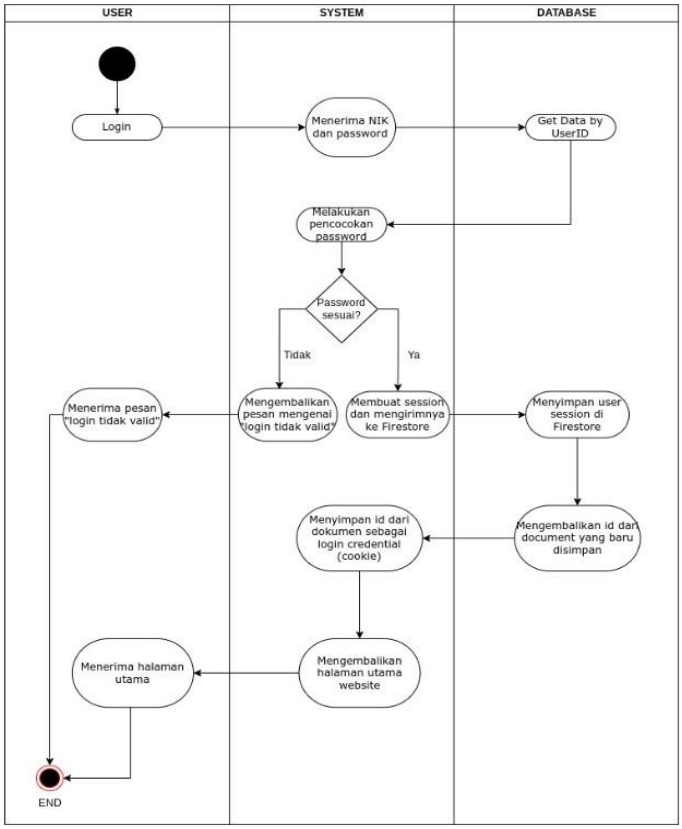
Gambar 4: Sequence Diagram dari double login dengan perangkat yang berbeda

Penjelasan dari Gambar 4 adalah pertama *user* menggunakan perangkat pertama untuk *login*. Setelah itu, dari *login page* akan mengirimkan *payload* ke *backend* untuk melakukan proses validasi NIK dan kata sandi*,* dan proses validasi *session* di Firestore untuk mengecek apakah *session* dengan NIK tersebut sudah ada di database atau belum. Setelah validasi berhasil, maka selanjutnya *backend* akan memgirim data ke Firestore untuk disimpan, dan dari Firestore mengembalikan sebuah *id* dari dokumen yang baru saja disimpan. *Id* dari dokumen tersebut digunakan untuk membuat *cookie* dan dikirim ke *frontend.* Setelah *cookie* terdeteksi, maka *frontend* mengembalikan halaman utama (*main page*) kepada *user*. Lalu *user* yang sama, melakukan *login* di perangkat kedua. Proses *login* sama dengan proses *login* perangkat pertama, yang menjadi pembeda adalah pada bagian validasi *session* di Firestore. *User* belum *logout* pada perangkat pertama sehingga *session* dari perangkat pertama masih ada di Firestore. Oleh karena itu, telah ditemukan *session* dengan NIK yang sama, sehingga terdapat tambahan proses untuk melakukan *update* terhadap data "status" dari *session* perangkat satu. Adanya perubahan "status" ini akan langsung terdeteksi oleh Firestore *snapshot Listener* pada *frontend*. Frontend pun akan memberikan notifikasi kepada *user* bahwa telah terdeteksi *login* di perangkat yang berbeda, serta perangkat dua pun dapat masuk ke halaman utama (*main page*).



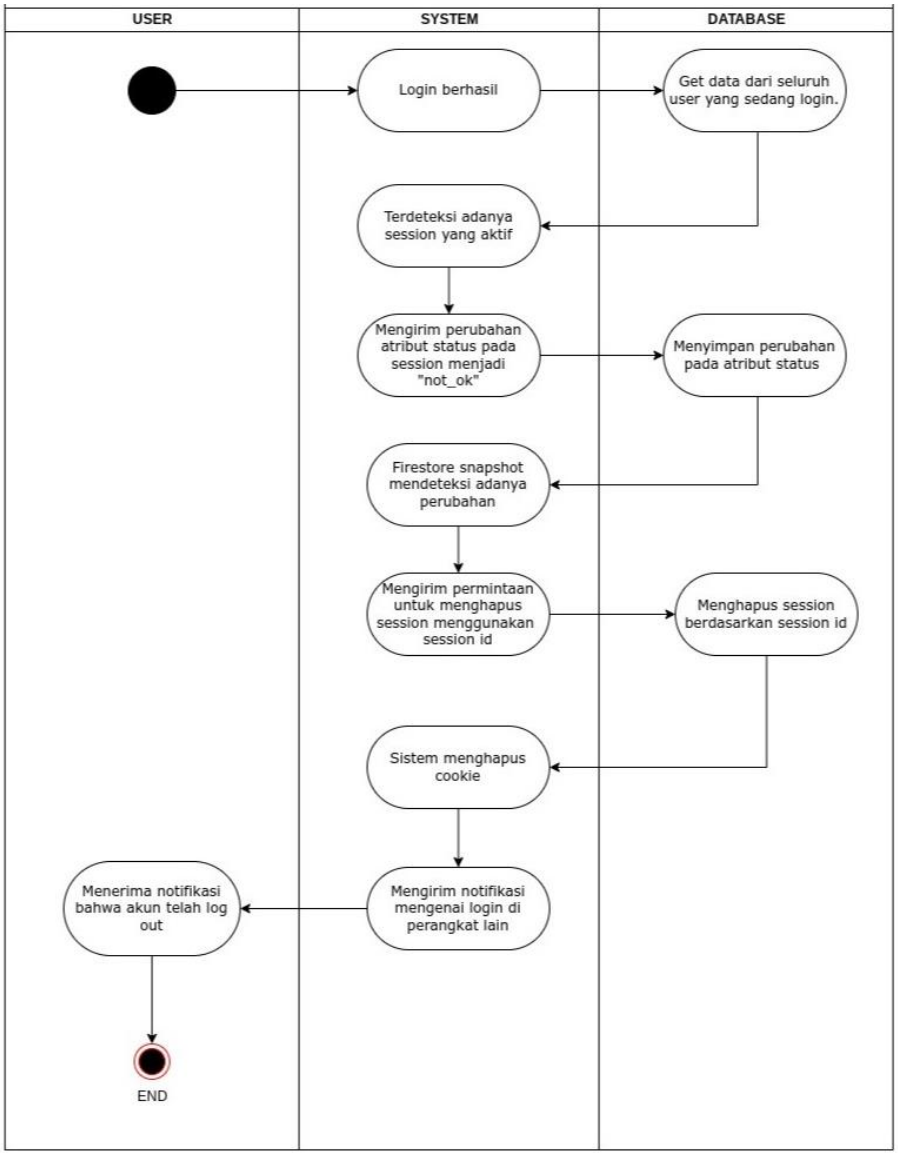
Gambar 5: Sequence Diagram dari perubahan hak akses oleh admin

Penjelasan dari Gambar 4 adalah pertama *user* menggunakan perangkat pertama untuk *login*. Setelah itu, dari *login page* akan mengirimkan *payload* ke *backend* untuk melakukan proses validasi NIK dan kata sandi, dan proses validasi *session* di Firestore untuk mengecek apakah *session* dengan NIK tersebut sudah ada di database atau belum. Setelah validasi berhasil, maka selanjutnya *backend* akan memgirim data ke Firestore untuk disimpan, dan dari Firestore mengembalikan sebuah *id* dari dokumen yang baru saja disimpan. *Id* dari dokumen tersebut digunakan untuk membuat *cookie* dan dikirim ke *frontend.* Setelah *cookie* terdeteksi, maka *frontend* mengembalikan halaman utama (*main page*) kepada *user*. Lalu *user* yang sama, melakukan *login* di perangkat kedua. Proses *login* sama dengan proses *login* perangkat pertama, yang menjadi pembeda adalah pada bagian validasi *session* di Firestore. *User* belum *logout* pada perangkat pertama sehingga *session* dari perangkat pertama masih ada di Firestore. Oleh karena itu, telah ditemukan *session* dengan NIK yang sama, sehingga terdapat tambahan proses untuk melakukan *update* terhadap data "status" dari *session* perangkat satu. Adanya perubahan "status" ini akan langsung terdeteksi oleh Firestore *snapshot Listener* pada *frontend*. Frontend pun akan memberikan notifikasi kepada *user* bahwa telah terdeteksi *login* di perangkat yang berbeda, serta perangkat dua pun dapat masuk ke halaman utama (*main page*).



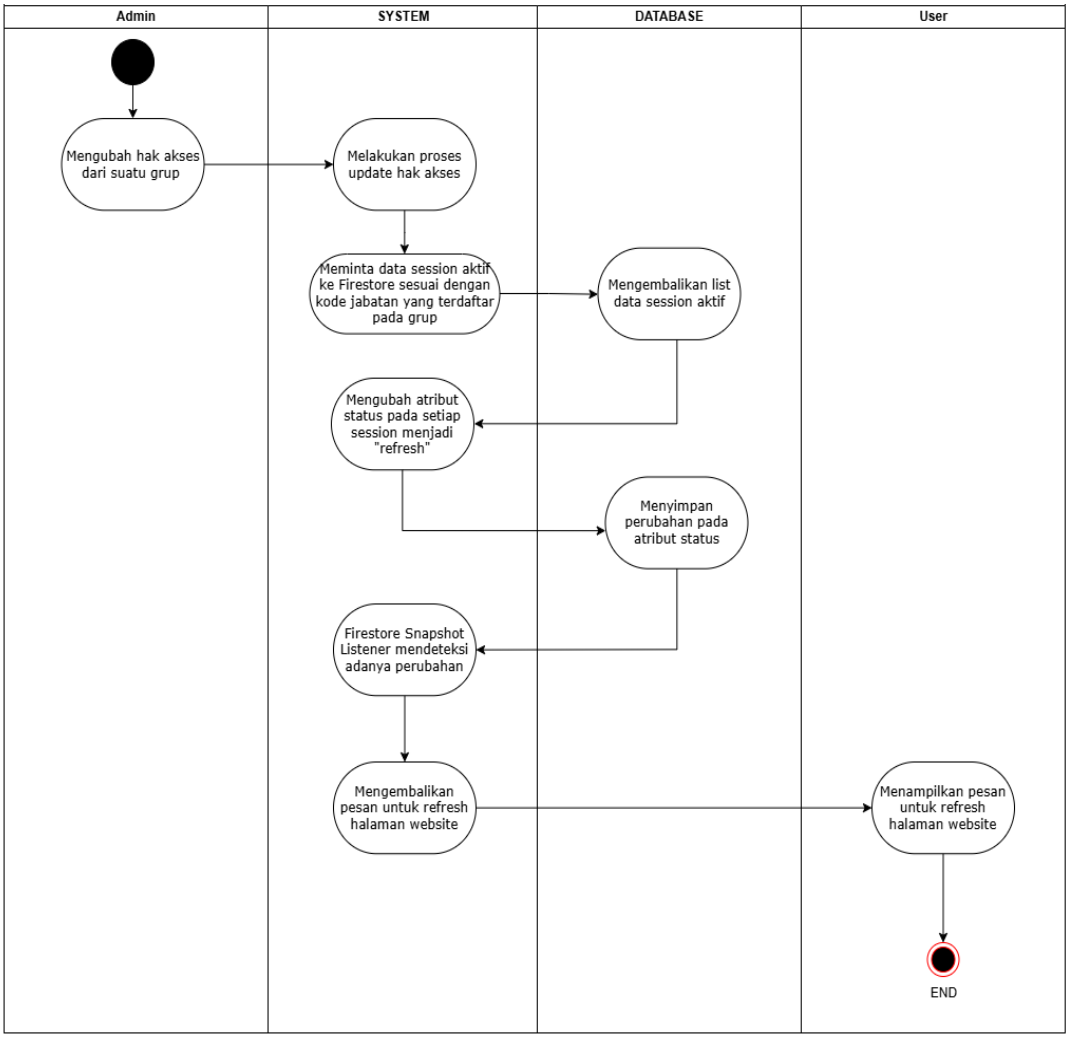
Gambar 6: Activity Diagram dari proses login (pembuatan session)

Gambar 6 merupakan proses ketika *user* melakukan *login*. Proses berawal dari *user* yang memasukkan NIK dan *password*, lalu sistem akan meminta data *password* ke *database* berdasarkan NIK dari *user*. Setelah itu, sistem melakukan proses validasi *password.* Bila *password* tidak sesuai, maka akan sistem akan mengembalikan notifikasi kepada *user* bahwa *login* tidak valid dan aktivitas pun selesai. Bila *password* sesuai, maka selanjutnya terdapat proses pencarian *session* aktif yang akan dijelaskan pada Gambar 6. Setelah itu, sistem akan membuat *session* dan mengirimnya ke *database* untuk disimpan. Kemudian *database* mengembalikan *id* dari *session* yang baru saja disimpan, yang kemudian *id* tersebut digunakan oleh sistem untuk membuat *login credential* (*cookie).* Kemudian sistem mengembalikan halaman utama *website* kepada *user* dan aktivitas *login* selesai.



Gambar 7: Activity Diagram dari proses deteksi double login

Gambar 7 merupakan proses ketika *user* melakukan *login* tetapi telah ditemukan *session* yang aktif di perangkat lain milik *user*. Proses berawal dari *user* yang berhasil melakukan *login* di pernagkat yang berbeda, lalu sistem akan mencari *data session* di *database* berdasarkan NIK dari *user* yang baru saja *login*. Kemudian telah ditemukan *session* yang masih aktif dari akun *user* yang baru saja *login,* sehingga sistem melakukan proses *update value* pada atribut "status" dari *session* yang lama dan perubahan tersebut disimpan di *database*. Sistem mendeteksi adanya perubahan secara *real-time* menggunakan Firestore *snapshot listener*, dan kemudian sistem mengirim permintaan ke *database* untuk menghapus *session* yang lama tersebut berdasarkan *session id*. Kemudian sistem menghapus *cookie* dan mengirim notifikasi ke perangkat yang lama bahwa *login* telah *expired* dan aktivitas deteksi *double login* telah selesai.



Gambar 8: Activity Diagram dari proses Firestore Snapshot Listener mendeteksi perubahan hak akses
Listener mendeteksi perubahan hak akses

Gambar 8 merupakan proses ketika *admin* mengubah hak akses dari suatu *group*. Proses berawal dari *admin* yang sudah melakukan *login* dan berada di *menu user management*, kemudian mengubah hak akses dari suatu *group*. Lalu sistem melakukan proses *update* hak akses. Setelah itu, sistem akan meminta data-data *session* yang sedang aktif ke Firestore sesuai dengan kode jabatan yang terkait dengan *group.* Firestore mengembalikan *list* yang berisi data-data *session* yang sedang aktif, kemudian sistem melakukan proses *update* pada atribut "status" dari setiap data-data *session* yang diperoleh. Setelah itu, hasil perubahan disimpan di *database* dan sistem dengan menggunakan Firestore *snapshot listener* telah mendeteksi adanya perubahan secara *real-time* dan mengirimkan notifikasi kepada *session* yang terkait untuk melakukan *refresh* pada *website*.

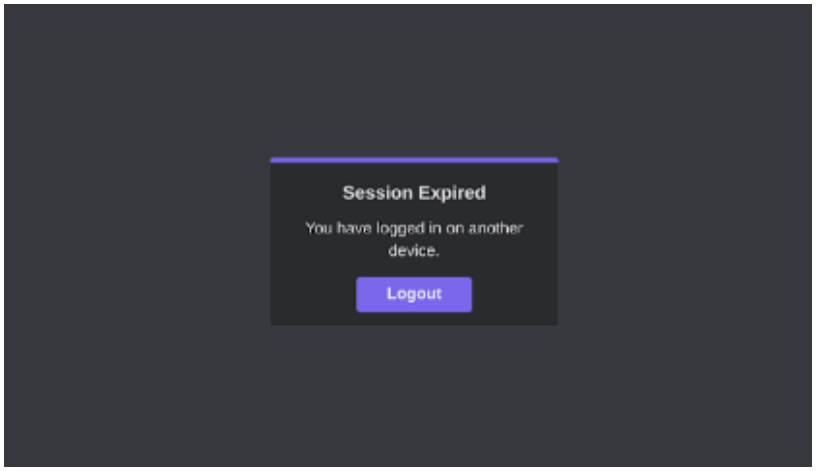
4.2. Implementasi Sistem

Implementasi sistem dilakukan dengan cara *deploy* sistem manajemen *session* ke lingkungan *staging*. Lingkungan *staging* dipilih sebagai tahap antara pengembangan dan produksi karena memungkinkan pengujian dilakukan secara lebih aman sebelum sistem diterapkan di lingkungan *production*. Tujuan utama dari implementasi ini adalah untuk melakukan uji coba sistem secara menyeluruh oleh beberapa pengguna yang telah ditentukan, agar dapat mengidentifikasi dan memperbaiki potensi permasalahan sebelum sistem diluncurkan secara resmi ke tahap produksi.

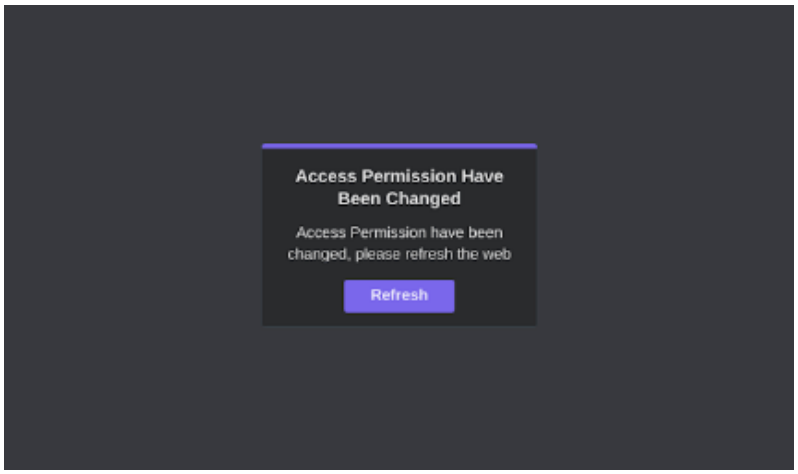
Sistem manajemen *session* yang diimplementasikan memiliki fitur untuk mendeteksi adanya aktivitas *double login*, yaitu ketika pengguna pengguna mencoba *login* dengan akun yang sama secara bersamaan

dari dua perangkat yang berbeda secara *real-time*. Fitur ini bertujuan untuk meningkatkan keamanan guna mencegah penggunaan akun yang tidak sah secara bersamaan.

Selain itu, sistem ini dapat mendeteksi adanya perubahan terhadap hak akses pengguna secara *real-time*. Bila terdapat perubahan hak akses, maka sistem akan memberikan notiikasi kepada pengguna yang bersangkutan untuk merefresh halaman *website*. Fitur ini bertujuan untuk mempermudah pengguna supaya tidak perlu melakukan proses *relogin* dan meningkatkan keamanan sistem dalam mengelola otorisasi pengguna secara dinamis.



Gambar 9: Notifikasi *Session Expired* karena terdeteksi login di perangkat lain



Gambar 10: Notifikasi ketika hak akses telah diubah ketika *user* sedang *login*

4.3. Pemanfaatan Teknologi Firestore *Snapshot Listener*

Adanya teknologi Firestore *Snapshot Listener* ini dapat mempermudah komunikasi antara *backend* dengan *frontend*, yang di mana *backend* akan memperbarui data *session* dan *frontend* dapat langsung mendeteksi adanya perubahan data *session* terkait berkat teknologi Firestore *Snapshot Listener* ini.

4.3.1. Fungsi untuk *Update Session Status*

Kode program 1: ***Update session status***

```
collection = self.firestore.collection("nama_koleksi")
documents = collection.stream()
for doc in documents:
    data = doc.to_dict()
    if data.get("nik") == nik:
        data_update = {
            "status": "not_ok"
        }
        collection.document(doc.id).update(data_update)
        break
```

Fungsi untuk *Update Session Status* merupakan fungsi untuk memperbarui atribut dari dokumen yang terdapat pada *Collection* di basis data Firestore dengan menggunakan bahasa pemrograman Python. "self.firestore" merupakan suatu *instance object* yang membawa konfigurasi koneksi ke database Firestorenya. Lalu "self.firestore.collection("nama_koleksi")" berfungsi untuk membuat koneksi yang mengambil *collection* dari Firestore. Lalu "collection.stream()" berfungsi untuk mengambil seluruh dokumen yang ada pada *Collection*. Setelah itu menggunakan perulangan untuk mencari data *session* menggunakan "nik". Bila data ditemukan, berarti akun dengan nik tersebut sudah *login* diperangkat lainnya.

Variabel "data_update" merupakan variabel yang menampung data yang ingin diubah pada dokumen, yang dimana *value* dari atribut "status" yang akan diubah.

Sebelum itu, terdapat atribut "status" pada *session* yang akan menjadi kunci utama dalam komunikasi antara *backend* dengan *frontend*. Terdapat tiga jenis "status", diantaranya "ok" untuk *user* yang baru melakukan *login* dan *session* masih valid, "not_ok" untuk *user* yang melakukan *login* dua kali di perangkat yang berbeda, dan "refresh" yang berarti terjadi perubahan hak akses ketika *session* sedang aktif atau *user* sedang mengakses *website*.

Fungsi di atas merupakan contoh fungsi untuk mengecek *double session*, sehingga bila ditemukan, "status" nya akan berubah menjadi "not_ok". Lalu yang mengubah value dari "status"nya menjadi "refresh", kodenya sama persis dengan kode yang di atas, hanya berbeda pada pergantian isi *value* untuk "status"nya.

Kemudian "collection.document(doc.id).update(data_update)" merupakan kode untuk memperbarui data dari dokumen berdasarkan id dari dokumen tersebut dan variabel "data_update" yang berisi data yang sudah diperbarui dikirim ke parameter "update". Ketika proses *update* sudah berhasil, maka *frontend* dapat mendeteksi secara langsung terkait perubahan data tersebut.

4.3.2. Fungsi onSnapshot dari Firestore

Kode program 2: Firestore *snapshot listener* di Javascript

```
webix.firestore.collection("nama_koleksi").doc(isLogin).onSnapshot((doc) => {
    if (doc.data().status == "not_ok") {
        ...
    }
    else if (doc.data().status == "refresh") {
        var user = webix.firestore.collection("nama_koleksi").doc(isLogin)
        user.update({
            status: "ok"
        })
        ...
    }
})
```

Fungsi onSnapshot dari Firestore merupakan fungsi yang terletak pada *frontend,* menggunakan bahasa pemrograman Javascript dan memanfaatkan Webix sebagai *framework* untuk membuat *frontend.* Fungsi tersebut berfungsi untuk pendeteksian *double login* maupun perubahan hak akses secara *real-time*. "webix.firestore" merupakan sebuah variabel yang menampung konfigurasi koneksi *database* Firestore. Lalu terdapat ".doc(isLogin)" yang merupakan fungsi untuk mendapatkan dokumen dari *Collection* berdasarkan *session id* dari *user* yang sedang *login. Session id* dari user ini disimpan pada variabel "isLogin". Kemudian terdapat fungsi *listener* dari Firestore yang bernama "onSnapshot()". Penggunaan

onSnapshot ini memanggil *callback function* setiap kali terjadi perubahan pada dokumen. Isi dari *snapshot* tersebut akan disimpan pada variabel "doc". *Callback function* berisi mengenai pemeriksaan atribut "status" dari dokumen tersebut. Bila statusnya "not_ok", maka akan menembak *endpoint logout* menggunakan *ajax*, untuk menghapus *cookie* dan *session* di Firestore. Bila statusnya "refresh", maka sistem akan memperbarui "status" dari dokumennya menjadi "ok" supaya tidak terjadi *infinite loop* akibat "status" dari dokumen yang tidak berubah, dan mengirim notifikasi ke *user* untuk *refresh* halaman *web.*

4.3. Pengujian Sistem

Setelah *deploy* sistem manajemen session di lingkungan *staging*, maka selanjutnya dilakukan pengujian sistem untuk mengevaluasi kemampuan sistem dan memastikan bahwa sistem yang dibuat sudah sesuai dan siap digunakan [20].

Pengujian sistem akan dilakukan menggunakan *Black box Testing* sebagai metode pengujiannya. *Black box Testing* merupakan sebuah metode untuk menguji perangkat lunak dari segi spesifikasi fungsional tanpa menguji desain dan kode program. *Black box Testing* bertujuan untuk mengetahui apakah fungsi, masukan, dan keluaran dari perangkat lunak sesuai dengan hasil yang diharapkan [21]. *Black box Testing* dilakukan pada fitur login dan perubahan hak akses, serta dilakukan secara mandiri oleh *developer*.

Tabel 1. *Black box Testing*

No	Fungsi	Kondisi	Hasil yang diharapkan	Hasil Pengujian	Kesimpulan
1.	Login	NIK dan *password* benar	*Login* berhasil	*Login* berhasil	Sesuai
		NIK atau *password* salah	*Login* gagal	*Login* gagal	Sesuai
		Akun sudah *login* di perangkat lain	Perangkat lain menerima notifikasi "*Session expired*"dan *logout* secara *real-time*	Perangkat lain menerima notifikasi "*Session expired*"dan *logout* secara *real-time*	Sesuai
		Terjadi *login* di perangkat lain	Menerima notifikasi "*Session expired*" dan *logout* secara *real-time*	Menerima notifikasi "*Session expired*" dan *logout* secara *real-time*	Sesuai
2.	Perubahan Hak Akses	Mengakses menu User Management	Menu dapat diakses	Menu dapat diakses	Sesuai
		Dapat mengubah hak akses	Berhasil mengubah hak akses	Berhasil mengubah hak akses	Sesuai
		Terjadi perubahan hak akses pada suatu *group*	Muncul notifikasi secara *real-time* terkait perubahan hak akses kepada pengguna-pengguna yang bersangkutan dengan *group*	Notifikasi berhasil muncul secara *real-time* kepada pengguna-pengguna yang bersangkutan dengan *group*	Sesuai

Selain melakukan pengujian menggunakan metode Black Box, penelitian ini juga melibatkan wawancara dengan admin Dashboard IT Support sebagai narasumber. Sebelum wawancara dilaksanakan, sistem terlebih dahulu dipresentasikan kepada admin untuk menjelaskan alur dan mekanisme kerja sistem secara menyeluruh. Dalam wawancara tersebut, terdapat dua topik utama yang dibahas, yaitu deteksi double login secara real-time dan deteksi perubahan hak akses secara real-time.

Terkait fitur deteksi double login, narasumber menyampaikan bahwa fitur tersebut merupakan salah satu peningkatan keamanan yang baik karena mampu bekerja secara real-time. Menurutnya, fitur yang ada saat

ini sudah sesuai dan telah memenuhi kebutuhan, terutama karena adanya pembatasan session yang dapat mencegah penumpukan session yang tidak diperlukan. Ia juga menegaskan bahwa keamanan sistem dapat ditingkatkan dengan adanya fitur ini. Dari sisi notifikasi, pesan yang ditampilkan dinilai sudah cukup jelas dan mudah dipahami. Meskipun demikian, terdapat kemungkinan pada awalnya fitur ini dapat sedikit mengganggu pengguna yang terbiasa membuka website pada dua perangkat berbeda. Namun, seiring waktu, pengguna diyakini akan terbiasa sehingga tidak lagi merasa terganggu.

Pada topik deteksi perubahan hak akses, narasumber menilai bahwa fitur yang dikembangkan sudah baik. Akan tetapi, ia memberikan saran agar perubahan hak akses tidak hanya dilakukan berdasarkan group, melainkan dapat dipersempit cakupannya, misalnya berdasarkan kode cabang. Hal ini dikarenakan terdapat beberapa cabang yang kerap menimbulkan permasalahan, seperti penggunaan fitur "kirim data" secara berulang-ulang yang menyebabkan request duplikat, sehingga perlu adanya pembatasan yang lebih spesifik. Adapun notifikasi yang diberikan pada fitur ini juga dinilai sudah jelas dan dapat dimengerti.

V. SIMPULAN DAN SARAN

Selain memiliki peran sebagai database, Firestore memiliki fitur yang dapat digunakan untuk mendeteksi adanya perubahan secara real-time, yaitu Firestore snapshot listener. Firestore snapshot listener ini dapat memberikan performa yang baik, yaitu dengan memberikan respon yang sangat cepat ketika terjadi adanya suatu perubahan data. Oleh karena itu, Firestore merupakan pilihan yang tepat untuk digunakan dalam perancangan sistem manajemen session, yaitu dengan memanfaatkan fitur dari Firestore yaitu Firestore snapshot listener.

Selain itu, sistem ini dibuat menggunakan Python, Javascript, dan Firestore. Firestore berperan sebagai teknologi yang menyediakan fitur deteksi perubahan *data real-time*. Python berperan sebagai *backend* dan Javascript sebagai *frontend* yang berinteraksi langsung dengan *user*, serta kedua bahasa ini sudah mendukung integrasi dengan Firestore melalui library, sehingga komunikasi data dapat dilakukan dengan mudah. Oleh karena itu, kombinasi dari ketiga teknologi ini dapat menghasilkan sistem manajemen *session* yang *real-time*.

Penerapan sistem manajemen *session* ini juga meningkatkan pengalaman pengguna, seperti dari segi keamanan yang di mana sistem ini dapat mencegah penyalahgunaan akun. Selain itu, pengguna dapat lebih dimudahkan dengan adanya notifikasi untuk *refresh* halaman *website* ketika terjadi perubahan hak akses. Menurut hasil wawancara dari kepala TI, dikatakan bahwa sistem ini dapat sangat membantu dalam mencegah tindakan yang tidak diinginkan. Dengan demikian, sistem manajemen *session* ini dapat meningkatkan pengalaman pengguna secara efektif.

Terdapat beberapa saran yang diberikan mengenai sistem manajemen *session* ini. Pertama, akan lebih baik bila menerapkan fitur untuk melihat *history* dari perangkat yang pernah *login* dari akun untuk meningkatkan keamanan lebih tinggi supaya bila terjadi hal yang tidak diinginkan, maka bisa dianalisa melalui fitur ini. Lalu, menurut hasil dari wawancara dengan *admin* dari Dashboard IT Support, perubahan hak akses dapat diterapkan di level cabang supaya pembatasan hak akses dapat dilakukan dengan jangkauan yang lebih kecil. Hal ini bertujuan disaat hanya ingin membatasi hak akses pada satu atau beberapa cabang saja, tidak seluruh semua cabang dibatasi.

DAFTAR PUSTAKA

[1] I. F. Maulana, "Penerapan Firebase Realtime Database pada Aplikasi E-Tilang Smartphone berbasis Mobile Android," *J. RESTI (Rekayasa Sist. dan Teknol. Informasi)*, vol. 4, no. 5, pp. 854–863, 2020, doi: 10.29207/resti.v4i5.2232.

[2] Google, "Get realtime updates with Cloud Firestore." [Online]. Available: https://firebase.google.com/docs/firestore/query-data/listen

[3] M. N. Abidi, "Kitchin: An Employment Oriented Networking Application for Restaurant Industry," vol. 9, no. December, pp. 356–363, 2022.

[4] D. Riswanda and A. T. Priandika, "Analisis Dan Perancangan Sistem Informasi Manajemen Bengkel Sepeda Motor Berbasis Online," *J. Inform. dan Rekayasa Perangkat Lunak*, vol. 2, no. 1, pp. 94–101, 2022.

[5] M. Puspitasari, Setiawansyah, and A. Budiman, "Perancangan Sistem Informasi Manajemen Perpustakaan Menggunakan Metode Fast (Framework for the Application System Thinking)(Studi Kasus: Sman 1 Negeri Katon)," *J. Teknol. Dan Sist. Inf.*, vol. 2, no. 2, pp. 69–77, 2021, [Online]. Available: https://jim.teknokrat.ac.id/index.php/sisteminformasi/article/view/1136%0Ahttps://jim.teknokrat.ac.id/index.php/sisteminformasi/article/viewFile/1136/361

[6] H. Y. P. Napitupulu and I. G. D. Nugraha, "Sistem Berbasis Komputasi Kabut Untuk Sistem Parkir Pintar Terdesentralisasi Menggunakan Firebase," vol. 13, pp. 44–52, 2024.

[7] F. Sinlae, I. Maulana, F. Setiyansyah, and M. Ihsan, "Pengenalan Pemrograman Web: Pembuatan Aplikasi Web Sederhana Dengan PHP dan MYSQL," *J. Siber Multi Disiplin*, vol. 2, no. 2, pp. 68–82, 2024, [Online]. Available: https://creativecommons.org/licenses/by/4.0/

[8] R. E. Pongoh, H. E. Wowor, and K. Santa, "Pengembangan Game Pacman Interaktif dengan Pemanfaatan Teknologi JavaScript," *TEKTONIK J. Ilmu Tek.*, vol. 1, no. 2, pp. 188–196, 2023, [Online]. Available: https://doi.org/10.62017/tektonik.v1i2.478

[9] V. Thangarajah, "International Journal of Advance Engineering and Research PYTHON CURRENT TREND APPLICATIONS- AN OVERVIEW," *ResearchGate*, no. October 2019, pp. 6–12, 2019.

[10] U. Patkar, P. Singh, H. Panse, S. Bhavsar, and C. Pandey, "Python for Web Development," *Int. J. Comput. Sci. Mob. Comput.*, vol. 11, no. 4, pp. 36–48, 2022, doi: 10.47760/ijcsmc.2022.v11i04.006.

[11] R. M. Ulah and I. M. Suartana, "Implementasi Session Management Pada Website Magang Menggunakan Teknologi MERN," *JINACS (Journal Informatics Comput. Sci.*, vol. 06, no. 3, pp. 765–777, 2024, doi: 10.26740/jinacs.v6n03.p765-777.

[12] R. Moch, M. Puja Pradana, M. Data, and D. P. Kartikasari, "Implementasi Shared Session Dalam Klaster Server Web Menggunakan PHP dan MySQL," *J. Pengemb. Teknol. Inf. dan Ilmu Komput.*, vol. 3, no. 5, pp. 4856–4864, 2019, [Online]. Available: http://j-ptiik.ub.ac.id

[13] K. P. Zuhroh and R. Darni, "Perancangan Sistem Informasi Manajemen dan Ekstrakurikuler SMAN 1 Kota Solok," *Pendidik. Tambusai*, vol. 6, no. 2, pp. 12769–12779, 2022, [Online]. Available: https://www.jptam.org/index.php/jptam/article/view/4138

[14] K. Milojković, M. Živković, and N. Bačanin Džakula, "Agile Multi-user Android Application Development With Firebase: Authentication, Authorization, and Profile Management," pp. 405–412, 2024, doi: 10.15308/sinteza-2024-405-412.

[15] I. P. Sari, A. Jannah, A. M. Meuraxa, A. Syahfitri, and R. Omar, "Perancangan Sistem Informasi Penginputan Database Mahasiswa Berbasis Web," *Hello World J. Ilmu Komput.*, vol. 1, no. 2, pp. 106–110, 2022, doi: 10.56211/helloworld.v1i2.57.

[16] S. Maskar and P. S. Dewi, "Praktikalitas dan Efektifitas Bahan Ajar Kalkulus Berbasis Daring Berbantuan Geogebra," *J. Cendekia J. Pendidik. Mat.*, vol. 4, no. 2, pp. 888–899, 2020, doi: 10.31004/cendekia.v4i2.326.

[17] F. F. Dewi and S. L. Handayani, "Pengembangan Media Pembelajaran Video Animasi En-Alter Sources Berbasis Aplikasi Powtoon Materi Sumber Energi Alternatif Sekolah Dasar," *J. Basicedu*, vol. 5, no. 4, pp. 2530–2540, 2021, doi: 10.31004/basicedu.v5i4.1229.

[18] D. B. Pradifta and I. S. Widiati, "Perancangan Sistem Informasi Pemesanan dan Penjualan Berbasis Web Pada Burjo Ben Klaten," *JEKIN (Jurnal Tek. Inform.*, vol. 4, no. 2, 2024, doi: 10.58794/jekin.v4i2.712.

[19] A. Ichwani, N. Anwar, K. Karsono, and M. Alrifqi, "Sistem Informasi Penjualan Berbasis Website dengan Pendekatan Metode Prototype," *Pros. SISFOTEK*, vol. 5, no. 1, pp. 1–6, 2021, [Online]. Available: https://scholar.google.com/citations?view_op=view_citation&hl=en&user=FOwZ8hUAAAAJ&pagesize=100&citation_for_view=FOwZ8hUAAAAJ:F9fV5C73w3QC

[20] N. M. D. Febriyanti, A. Sudana, and I. N. Piarsa, "Implementasi Black Box Testing pada Sistem Informasi Manajemen Dosen," *Jurnal Ilmiah Teknologi Dan Komputer*, vol. 2, no. 3. 2021. [Online]. Available: http://download.garuda.kemdikbud.go.id/article.php? article=3457876& val=30165&title=Implementasi Black Box Testing pada Sistem Informasi Manajemen Dosen

[21] M. Syarif and E. B. Pratama, "Analisis Metode Pengujian Perangkat Lunak Blackbox Testing Dan Pemodelan Diagram Uml Pada Aplikasi Veterinary Services Yang Dikembangkan Dengan Model Waterfall," *J. Tek. Inform. Kaputama*, vol. 5, no. 2, pp. 253–258, 2021.