BAB II

TINJAUAN PUSTAKA

2.1. Penelitian Sebelumnya

Tabel 2.1 Tabel Penelitian Sebelumnya

No	Penulis(Tahun)	Judul	Hasil/Kesimpulan
1	Sonam Khedkar dan Swapnil Thube (2017)	Real Time Database for Application, International Research Journal of Engineering and Technology (IRJET)	Penelitian ini membandingkan antara Firebase, Mongo DB, dan Rethink. Ditemukan kesimpulan bahwa *realtime database* yang menggunakan ekstensi lebih memberikan performa untuk menghasilkan respon yang dipercaya, karena data yang disimpan dalam *cloud* lebih bisa diakses dimana saja. Dengan menggunakan Firebase, Mongo DB, atau Rethink tidak ada keperluan untuk membuat database atau API sendiri dan biasa akan terbuat *backend* dari aplikasi kita jika menggunakan jenis *real time database*.
2	Navdeep Singh (2016)	Study of Google Firebase API for Android, International Journal of Innovative Research in Computer and Communication Engineering	Penelitian ini untuk mempelajari lebih dalam mengenai API Firebase dengan fitur"nya. Mempelajari pemakaian Firebase pada *android project* dan cara menggunakan fitur sesuai yang kita perlukan.
3	Vikhyat Kumar, Tushar Thussu, Vinod Kumar (2016)	Developing Hangman Game in Android using Android Studio, International Journal of	Kesimpulan penelitian ini adalah implementasi pembuatan *game* dengan penerapan *framework* pada Android Studio.

		Scientific & Engineering Research	
4	Abdullah Alsalemi, Yahya Alhomsi, Mohammed Al Disi, Ibrahim Ahmed, Faycal Bensaali, dan Abbes Amira (2017)	Real-Time Communication Network using Firebase Cloud IoT Platform for ECMO Simulation, IEEE International Conference on Internet of Things (iThings) and IEEE Green Computing and Communications (GreenCom) and IEEE Cyber, Physical and Social Computing (CPSCom) and IEEE Smart Data (SmartData)	Penelitian ini membahas tentang prosedur penyelamatan nyawa yaitu *Extracorporeal Membrane Oxygenation* atau disingkat ECMO dimana dalam pelaksanaannya harus cepat dan teratur untuk menghindari kematian. Maka dari itu untuk mengimplementasi hal tersebut pada suatu sistem diperlukan sesuatu yang kuat, dan arsitektur yang sangat terencana. Penggunaan IoT dengan berbasis Firebase digunakan untuk membuat sistem ECMO. Hasilnya adalah Firebase cocok untuk pengembangan sistem ECMO dengan menyebutkan kelebihan-kelebihannya dengan kriteria yang diharuskan pada ECMO.

2.2. Pengertian *Game* Gomoku

Gomoku, atau yang biasa disebut Gobang atau juga *Five in a Row*, merupakan *game* strategi papan sederhana yang bersifat abstrak. Dimainkan oleh 2 pemain, hitam (jalan duluan) dan putih. Pemenang dari *game* ini adalah siapakah pemain yang berhasil membuat 5 pion warnanya berturut-turut dengan metode bisa

horizontal, vertikal, maupun diagonal. *Game* ini dimainkan pada papan berukuran 15 x 15 (Gomoku World, 2018).

Gomoku ini berasal dari Japan pada jaman Heian. Nama dari Gomoku ini berasal dari Bahasa Jepang, yang berarti *gomokunarabe*. *Go* berarti lima, *moku* berarti pion dan *narabe* yang berarti sebaris. *Game* ini juga populer di Korea dengan nama *omok*.

Setelah beberapa tahun *game* ini dibuat, muncul beberapa peraturan karena diketahui pion hitam yang jalan duluan selalu mempunyai kesempatan yang lebih besar, dan telah dibuktikan oleh L.Victor Allis bahwa pemain pion hitam dapat melakukan menang paksa. Maka dari itu banyak variasi dari *game* ini menambah aturan tambahan dengan tujuan mengurangi kesempatan pion hitam. Berikut adalah aturan-aturan tambahan yang dipakai:

1. Aturan *three and three* yaitu melarang pion hitam untuk membuat 2 baris dengan panjang tiap baris merupakan 3 pion hitam dan 3 pion hitam dalam 1 langkah.
2. Aturan *four and four* yang melarang pion hitam untuk membuat 2 baris dengan panjang tiap baris merupakan 4 pion hitam dan 4 pion hitam dalam 1 langkah.
3. Aturan *overlines* yang melarang pion hitam untuk membuat 1 baris dengan panjang 6 pion hitam dalam 1 langkah.

Hitam akan menang apabila melakukan penempatan 5 pion hitam berturut-turut baik secara vertikal, horizontal, maupun diagonal. Putih dapat menang apabila

menempatkan 5 pion putih berturut-turut baik secara vertikal, horizontal, maupun diagonal, dan memaksa pion hitam untuk melakukan pelanggaran maka putih akan langsung menang (Gomoku World, 2018).

2.3. Fitur Firebase

Dalam Firebase, terdapat 18 produk yang dapat digunakan untuk mengembangkan aplikasi Android yang akan dibuat:

1. Realtime database.

   Merupakan produk Firebase yang menggunakan *cloud-hosted* NoSQL *database* yang dapat digunakan untuk menyimpan dan menyinkronkan data diantara *users* dan juga *realtime*.

2. Cloud Firestore.

   Cloud Firestore merupakan NoSQL *document database* yang digunakan untuk memudahkan penyimpanan, sinkronisasi, dan juga melakukan *query* pada data untuk aplikasi *mobile* dalam skala *global*.

3. Cloud Functions.

   Membuat fungsi yang akan mengaktifkan produk Firebase, seperti perubahan data pada Realtime Database, *user* baru *sign-up* dengan Authentication, dan pengukuran kejadian pada Analytics.

4. Hosting.

   Dapat membuat *single-page web app*, halaman utama aplikasi *mobile*, atau aplikasi *website* yang progresif tanpa susah payah.

5. Performance Monitoring.

Dapat melihat bagaimana performa aplikasi kita dari pandangan *user*, dengan dilengkapi laporan performa tiap *user* tersebut secara otomatis.

6. Crashlytics.

   Membantu untuk melacak, memprioritaskan, dan memperbaiki masalah stabilitas yang merusak kualitas aplikasi, secara *realtime*. Dapat mengurangi waktu untuk *troubleshoot* masalah *crash* dan tentunya jadi lebih banyak waktu untuk membangun aplikasi lagi.

7. Authentication.

   Digunakan untuk membuat sistem *login* secara mudah, yang dapat meningkatkan kualitas pengalaman untuk *end users*. Produk ini menyediakan *end-to-end identity solution*, mendukung *login* via *email* dan *password*, otentifikasi dari *smartphone*, Google, Twitter, Facebook, dan masih banyak lagi metode *login* lainnya.

8. Cloud Storage.

   Dirancang untuk membantu dengan mudah melakukan penyimpanan dan untuk menyediakan konten untuk *user* lihat, seperti foto dan video.

9. Test Lab for Android.

   Untuk memastikan kualitas aplikasi, produk ini menyediakan *virtual devices* yang mengizinkan untuk melakukan tes percobaan untuk mensimulasi bagaimana lingkungan pemakaian yang akan terjadi.

10. Google Analytics.

SDK secara otomatis menangkap gerakan pemakaian suatu *key event* dan *user*, yang dapat digunakan sebagai laporan untuk mengukur sesuatu yang unik yang dapat digunakan untuk strategi bisnis.

11. Predictions.

    Menggunakan konsep dari mesin pembelajaran yang dimiliki Google untuk menciptakan sekelompok *users* yang dikategorikan berdasarkan kebiasaan yang dilakukan *user* tersebut. Dengan produk ini, dapat membuat keputusan untuk pengembangan aplikasi selanjutnya tanpa harus membuat *team* pengembangan lagi.

12. Remote Config.

    Dapat menganti tampilan dari aplikasi apabila ada iklan langsung dari Firebase, lalu secara otomatis juga menggunakan produk Analytics. Dengan demikian lebih mudah untuk menganti konten, mengatur *audience segment*, tanpa harus menunggu pembaruan dari *app store*.

13. App Indexing.

    Sistem *search engine* Google dapat dimasukkan pada aplikasi *mobile* apabila diperlukan beserta hasil carian halamannya.

14. AdWords.

    Menghubungkan AdWords *campaign* sehingga dapat dihubungkan dengan Firebase untuk dapat melakukan *conversion tracking* dengan mudah melewati Firebase sendiri. *Conversion tracking* yaitu untuk mengamati apakah iklan yang dipasang benar-benar dilihat dan diklik oleh *users*.

15. Cloud Messaging.

Digunakan untuk memberikan notifikasi dari *server* ke tiap perangkat yang telah melakukan instalasi dengan tujuan mengirim, dan menerima pesan notifikasi dari iOS, Android, dan website tanpa dipungut biaya apapun.

16. Dynamic Links.

    Merupakan *smart URL* karena dapat membantu untuk mengirim pada *user* atau yang berpotensi menjadi *user* pada lokasi apapun melewati aplikasi *mobile*.

17. Invites.

    Digunakan apakah aplikasi memerlukan sistem *referral* atau *sharing via email* untuk menarik pengguna lebih banyak.

18. AdMob.

    Membuat iklan pada aplikasi *mobile* yang dapat digunakan untuk mendapatkan keuntungan dari aplikasi yang telah *publish* (Firebase, 2018).

2.4. Metode Pengembangan Sistem SDLC Model *Waterfall*

Dapat diartikan secara literatur yang berarti air terjun. Namun bagi ilmu komputer dan juga teknologi informasi, *waterfall* merupakan salah satu jenis metode yang digunakan dalam melakuan pengembangan sistem. Pengembangan sistem dan juga perangkat lunak dari sebuah *software* komputer dilakukan secara sekuensial dan juga saling berurutan. Pada model pengembangan ini, sebuah pengembangan sistem dilakukan berdasarkan urutan perencanaan & analisis, desain, pengkodean, pengujian dan implementasi, dan berakhir pada tahap *maintenance* (Bassil, 2012).

Penjelasan 5 tahapan penting dalam model *waterfall*:

1. Tahapan perencanaan & analisis: mengacu pada fenomena dan juga permasalahan yang terjadi, dan mengapa sebuah aplikasi sangat penting untuk dibuat dalam mengatasi masalah atau fenomena tersebut. Kemampuan analisis tidak hanya dibebankan pada *programmer* saja, namun bisa juga dibebankan pada ahli ekonomi dan juga sosial politik.
2. Tahapan desain: pembuatan desain dari sebuah sistem. Dalam tahapan ini, tidak hanya desain interface sistemnya saja yang dikembangkan, namun juga dikembangkan desain dari alur sistem tersebut, hingga bagaimana satu sistem tersebut bisa bekerja, mulai dari tampilan awal, fungsi-fungsi tombol, hingga *input output* yang akan dihasilkan nantinya.
3. Tahapan pengkodean: merupakan tahapan yang wajib dilakukan oleh mereka yang mengerti bahasa pemrograman. Untuk menjalankan desain sistem yang sudah dibuat, maka kemudian kode dan juga script akan dimasukkan ke dalam desain sistem tersebut, sehingga nantinya desain dari sistem tersebut bisa berjalan dengan lancar dan juga baik.
4. Tahapan pengujian dan implementasi: Dalam pengujian dan implementasi akan dilihat apakah sistem dapat bekerja dengan baik, tampilan *interface* sesuai harapan, dan semua fungsinya bisa digunakan dengan baik dan lancar.
5. Tahapan *maintenance*: mengacu pada perbaikan sebuah sistem yang mengalami kerusakan, penambahan fitur-fitur baru pada sistem. Tahap ini ditentukan oleh kebutuhan dari *user*, dan apabila sebuah sistem memiliki

tahap *maintenance* yang baik, maka sistem tersebut akan berkembang dengan sangat baik.

Karena berbagai kerugian pada model *waterfall* ini maka muncul banyak versi modifikasi lain seperti:

1. Royce Model

   Model Royce menekankan pada kenyataan bahwa model *waterfall* menderita cacat serius yaitu tidak dapat kembali ke tahap sebelumnya untuk memperbaiki sesuatu yang tidak beres.

2. *Sashimi* Model

   Sashimi adalah versi modifikasi sejati dari model *waterfall*. Fase-fase yang agak sama seperti dalam model *waterfall*; yang membedakan tiap fase saling tumpang tindih satu sama lain yang bisa menjadi nilai positif. Misalnya waktu tidak akan terbuang karena sebelu tahap pertama selesai, tahap kedua akan berlangsung. Selain itu, karena mereka tumpang tindih, seseorang dapat kembali ke langkah sebelumnya jika diinginkan.

3. Model Siklus Hidup *Aorta*

   Perbedaan dalam model ini adalah banyak mengandalkan pada umpan balik yang berasal dari fase-fase lain sebelum maju ke berikutnya.

4. Model *Waterfall* V

   Bergantung pada sebuah program perangkat lunak perkembangan *linear* yang menekankan pada pengembangan seimbang lebih dari apa pun.

Kelebihan dari metode ini adalah memiliki proses yang urut, mulai dari analisa hingga *maintenance*. Setiap proses memiliki spesifikasinya sendiri,

sehingga sebuah sistem dapat dikembangkan sesuai dengan apa yang dikehendaki (tepat sasar). Lalu setiap proses tidak dapat saling tumpang tindih.

Kekurangan dari metode ini adalah proses yang dilakukan cenderung panjang dan juga lama, biaya penggunaan metode yang cenderung mahal, dan membutuhkan banyak riset dan juga penelitian pendukung untuk mengembangkan sistem (Arora & Arora, 2016).

2.5. Metode Uji Coba *Black Box*

Black-box testing adalah metode uji coba yang memfokuskan pada keperluan *software*. Karena itu uji coba *black-box* memungkinkan pengembang *software* untuk membuat himpunan kondisi *input* yang akan melatih seluruh syarat-syarat fungsional suatu *program*. Metode pengujian *black-box* berusaha untuk menemukan kesalahan dalam beberapa kategori, diantaranya: fungsi-fungsi yang salah atau hilang, kesalahan *interface*, kesalahan dalam struktur data atau akses *database* eksternal, kesalahan performa, kesalahan inisialisasi, dan terminasi.