BAB II
TINJAUAN PUSTAKA

2.1 Penelitian Terdahulu

Penelitian terdahulu yang memiliki keterkaitan dengan penelitian ini dicantumkan pada tabel 2.1 mengenai penelitian terdahulu yang terkait. Terdapat 5 jurnal referensi yang sesuai dengan topik penelitian. Adapaun penjelasan dari penelitian yang terkait sebagai berikut:

Penelitian pertama [2], tujuan dari penelitian ini yaitu untuk mengembangkan pendekatan baru dalam memprediksi cacat pada kode program dengan menggunakan teknik *anomaly detection*. Prediksi cacat sebagai masalah deteksi anomali dan mengusulkan model REPD (*Reconstruction Error Probability Distribution*). Penelitian ini juga bertujuan untuk mengevaluasi kinerja model REPD pada dataset fitur kode tradisional dan dataset fitur kode semantik, serta menganalisis keberlanjutan model terhadap ketidakseimbangan dataset. Penelitian ini memberikan kontribusi dalam bidang prediksi cacat dalam kode program perangkat lunak dengan memperkenalkan pendekatan Teknik *anomaly detection* menggunakan model REPD. Hasil penelitian ini dapat digunakan sebagai dasar untuk pengembangan lebih lanjut dalam deteksi cacat dan peningkatan kualitas perangkat lunak [2].

Penelitian kedua [8], yang bertujuan untuk mengembangkan metodologi keamanan baru yang efektif dalam mendeteksi serangan berorientasi data pada sistem cyber-fisik. Penelitian ini bertujuan untuk mengatasi kelemahan mekanisme deteksi anomali berbasis program yang ada dengan menggabungkannya dengan pemahaman tentang perilaku sistem cyber-fisik. Dalam penelitian ini, *Orpheus* diperkenalkan sebagai metodologi keamanan baru yang menggabungkan deteksi anomali berbasis program dengan pemahaman tentang perilaku sistem cyber-fisik untuk mengidentifikasi serangan yang tidak terdeteksi oleh mekanisme deteksi yang ada. Hasil evaluasi eksperimental besar-besaran menunjukkan bahwa

Orpheus dapat mengidentifikasi serangan berorientasi data pada sistem cyber-fisik dengan tingkat akurasi yang tinggi. Penelitian ini juga menunjukkan bahwa integrasi model berbasis fisika ke dalam pendekatan *Orpheus* dapat meningkatkan efektivitas deteksi serangan pada sistem cyber-fisik. Tujuan akhir dari penelitian ini adalah untuk meningkatkan keamanan sistem cyber-fisik dan melindungi mereka dari serangan berorientasi data [8].

Penelitian ketiga [9] pada tahun 2022 yang bertujuan mengembangkan alat ConCAD yang mendukung deteksi interaktif dari anomali kode dalam pengembangan perangkat lunak. Alat ini dirancang untuk memberikan pengembang kemampuan untuk mendeteksi anomali saat mereka masih mengedit kode, sehingga memungkinkan deteksi lebih awal. Selain itu, alat ini memungkinkan pengguna untuk mengidentifikasi dan menganalisis kemungkinan anomali kode dalam potongan kode yang diberikan. Alat ini menyediakan mode deteksi interaktif dan non-interaktif. Dalam mode interaktif, pengguna dapat mengaktifkan ConCAD dan fokus pada aktivitas pemrograman dalam fragmen kode tertentu, sementara alat ini melakukan analisis rinci terhadap metrik yang terkait dengan kode tersebut. Anomali kemudian diberitahukan kepada pengembang melalui penanda dalam kode. Dalam mode non-interaktif, alat ini menganalisis metrik yang terkait dengan semua elemen kode dalam proyek. Hasil deteksi anomali tersebut terdaftar dan dalam Tampilan ConCAD. Alat ini juga memungkinkan konfigurasi strategi deteksi dan prioritas anomali kode. Penelitian ini bertujuan untuk menyediakan alat yang fleksibel dan dapat disesuaikan, dengan fitur-fitur seperti pemetaan informasi, deteksi anomali, prioritas anomali, pemilihan pendekatan deteksi, dan visualisasi hasil. Penelitian ini juga bertujuan untuk menguji validitas dan efektivitas alat ConCAD dalam mendeteksi anomali kode [9].

Penelitian keempat [10], penelitian ini bertujuan untuk menerapkan teknik deteksi anomali pada kode sumber dan *bytecode* dalam rangka meningkatkan pengembangan bahasa pemrograman kotlin dan kompilernya. Penelitian ini bertujuan untuk mengidentifikasi dan memahami anomali dalam bahasa pemrograman kotlin, serta mengevaluasi kegunaan anomali tersebut dalam konteks

pengembangan bahasa pemrograman. Penelitian ini mengusulkan metode untuk mendeteksi anomali dalam program Kotlin menggunakan pohon sintaksis dan instruksi *bytecode*, lalu mengumpulkan dataset besar dari repositori GitHub dan menggunakan algoritma deteksi *outlier* dan *autoencoder* untuk mendeteksi anomali. Penelitian ini menunjukkan bahwa metode deteksi anomali, seperti menggunakan pohon sintaksis dan instruksi *bytecode*, dapat mengidentifikasi anomali dalam kode Kotlin Penelitian ini juga menemukan bahwa beberapa anomali yang terdeteksi dianggap bermanfaat oleh ahli bahasa Kotlin. Anomali-anomali ini dapat digunakan untuk meningkatkan infrastruktur pengujian kompiler atau menjadi bahan diskusi dalam perancangan bahasa. Penelitian ini memberikan wawasan berharga tentang penggunaan teknik deteksi anomali dalam pengembangan bahasa pemrograman dan menunjukkan efektivitasnya dalam rekayasa perangkat lunak [10].

Penelitian terakhir [11] yang bertujuan untuk mengembangkan sistem deteksi anomali yang dapat melindungi sistem basis data dari serangan yang dilakukan oleh program aplikasi. Sistem yang diusulkan, yaitu AD-PROM (*Anomaly Detection System for the Protection of Relational Database Systems against Data Leakage by Application Programs*), yang bertujuan untuk memonitor perilaku program aplikasi dan mendeteksi anomali yang dapat menjadi penyebab dari kemungkinan terjadinya kebocoran data. Sistem ini dapat melacak panggilan yang dilakukan oleh program aplikasi pada data yang diekstraksi dari basis data dan menganalisis aplikasi tersebut. Kemudian, sistem ini memantau eksekusi program untuk mendeteksi anomali yang mungkin merupakan upaya kebocoran data. Hasil eksperimen menunjukkan bahwa sistem ini sangat akurat dalam mendeteksi perubahan perilaku program dengan tingkat positif palsu yang rendah. Sistem AD-PROM menggunakan analisis statis untuk membangun grafik ketergantungan data dan grafik aliran kontrol program. Kemudian, sistem ini menggunakan analisis dinamis untuk mengumpulkan jejak program dan melatih Hidden Markov Model (HMM) untuk membuat profil aplikasi. Sistem ini dapat mendeteksi berbagai jenis serangan, termasuk serangan injeksi SQL dan serangan man-in-the-middle. AD-PROM juga menggunakan kombinasi teknik analisis statis dan dinamis untuk

membangun matriks transisi panggilan program (CTM) dan menginisialisasi parameter Hidden Markov Model (HMM). CTM mewakili probabilitas transisi antara panggilan sistem yang berbeda dalam program, sedangkan HMM digunakan untuk deteksi anomali. Alur kerja AD-PROM terdiri dari empat langkah utama: analisis statis, analisis dinamis, agregasi, dan inisialisasi dan pelatihan HMM. Pada langkah analisis statis, AD-PROM membangun grafik aliran kontrol (CFG) program dan mengidentifikasi panggilan sistem dalam kode. Pada langkah analisis dinamis, AD-PROM menjalankan program dengan input yang berbeda dan mencatat urutan panggilan sistem yang dieksekusi [11].

Tabel 2. 1 Tabel Penelitian Sebelumnya

No	Judul	Tahun	Penulis	Isi Penelitian
1	REPD: *Source code defect prediction as anomaly detection*	2020	Petar Afric, Lucija Sikic, Adrian Satja Kurdija, Marin Silic	Penelitian ini membahas tentang penggunaan model REPD (*Reconstruction Error Probability Distribution*) untuk prediksi cacat dalam kode perangkat lunak. Penggunaan Teknik *anomaly detection* sebagai pendekatan alternatif dalam memprediksi cacat pada source code. Model baru yang berbasis *anomaly detection* untuk memprediksi cacat pada *source code* dan menguji model tersebut pada dataset yang menggunakan fitur tradisional dan semantik.
2	*Checking is Believing: Event-Aware Program*	2018	Long Cheng, Ke Tian, Danfeng Daphne Yao, Lui Sha, Raheem A. Beyah	Penelitian ini membahas tentang kebutuhan akan mekanisme deteksi yang efektif untuk melindungi sistem

No	Judul	Tahun	Penulis	Isi Penelitian
	Anomaly Detection in Cyber-Physical Systems			cyber-fisik (CPS) dari serangan berorientasi data. Para penulis mengusulkan metodologi keamanan baru yang disebut *Orpheus*, yang menerapkan semantik eksekusi cyber-fisik untuk mendeteksi anomali dalam perilaku program kontrol. *Orpheus* menggabungkan deteksi anomali berbasis program dengan pemahaman tentang perilaku sistem cyber fisik untuk melihat serangan yang tidak terdeteksi oleh mekanisme deteksi yang ada.
3	ConCAD: *A Tool for Interactive Detection of Code Anomalies*	2022	Danyllo Albuquerque, Everton Guimaraes, Mirko Perkusich, Hyggo Almeida	Jurnal ini membahas tentang ConCAD, yaitu sebuah alat yang mendukung deteksi interaktif dari anomali kode dalam pengembangan perangkat

No	Judul	Tahun	Penulis	Isi Penelitian
				lunak. Alat ini memungkinkan pengembang untuk mendeteksi anomali saat mereka masih mengedit kode, melakukan deteksi yang dini dan berkelanjutan. Alat ini menyediakan berbagai fitur seperti pemetaan informasi, deteksi anomali, prioritas anomali, pemilihan pendekatan deteksi, dan visualisasi hasil.
4	*Using Large-Scale Anomaly Detection on Code to Improve Kotlin Compiler*	2020	Timofey Bryksin, Victor Petukhov, Ilya Alexin, Stanislav Prikhodko, Alexey Shpilman, Vladimir Kovalenko, Nikita Povarov	Jurnal ini membahas penerapan teknik deteksi anomali pada kode yang bersumber dan *bytecode* untuk meningkatkan pengembangan bahasa pemrograman dan kompilernya. Penelitian ini mendefinisikan anomali sebagai fragmen kode yang berbeda dalam bahasa

No	Judul	Tahun	Penulis	Isi Penelitian
				pemrograman tertentu.
5	*An Anomaly Detection System for the Protection of Relational*	2020	Daren Fadolalkarim, Elisa Bertino, Asmaa Sallam	Jurnal ini membahas tentang sistem deteksi anomali bernama AD-PROM (*Anomaly Detection System for the Protection of Relational Database Systems against Data Leakage by Application Programs*) yang bertujuan untuk melindungi sistem basis data relasional dari serangan yang dilakukan oleh program aplikasi. Sistem ini melacak panggilan yang dilakukan oleh program aplikasi pada data yang diekstraksi dari basis data dan menganalisis perilaku aplikasi untuk membangun profil yang mewakili perilaku normalnya. Kemudian, sistem ini memantau eksekusi

No	Judul	Tahun	Penulis	Isi Penelitian
				program untuk mendeteksi anomali yang mungkin merupakan upaya kebocoran data.

2.2 Dasar Teori

2.2.1 Anomali Kode Program

Secara umum anomali merupakan suatu kejadian yang tidak sesuai dengan pola atau standar yang diharapkan [3]. Anomali kode program adalah ketidaknormalan atau masalah dalam kode program yang terdapat pada perangkat lunak yang dapat mempengaruhi kualitas dan pemeliharaan perangkat lunak. Anomali ini dapat berupa pelanggaran aturan pemrograman, ketidaksesuaian desain, kompleksitas yang berlebihan, atau masalah lain yang dapat menyebabkan kesulitan dalam memahami atau mengembangkan perangkat lunak [9]. Berikut adalah beberapa contoh fungsi kode anomali yang terdapat pada aplikasi android:

1. *READ_PHONE_STATE*

"*READ_PHONE_STATE*" adalah sebuah kode yang digunakan dalam file AndroidManifest.xml dalam proyek Android untuk menyatakan bahwa aplikasi membutuhkan izin untuk membaca informasi tentang status dan identitas yang berada pada perangkat telepon. Izin ini biasanya diperlukan jika aplikasi Anda ingin mengakses informasi seperti nomor telepon perangkat, identifikasi unik perangkat (IMEI), status panggilan, dan daftar akun telepon apa saja yang terdapat pada perangkat [4].

"*READ_PHONE_STATE*" adalah sebuah kode untuk meminta izin yang berbahaya dan mengancam pengguna ponsel ke ancaman malware setelah izin tersebut diaktifkan. Penyerang dapat memperoleh informasi penting pengguna dan perangkat identifikasi untuk melakukan berbagai bentuk serangan, seperti

pembajakan akun media sosial, phishing, dan pencurian data untuk mendapatkan keuntungan. Pada website developer.android.com, google sudah mendapat peringatan bahwa protection level pada kode *permission* ini berbahaya, jadi ketika digunakan maka akan bersifat sensitif [4].

2. *WRITE_EXTERNAL_STORAGE*

"*WRITE_EXTERNAL_STORAGE*" digunakan dalam pengembangan aplikasi Android untuk mengakses izin tertentu pada perangkat. Izin ini memungkinkan aplikasi untuk menulis atau menyimpan data ke penyimpanan eksternal perangkat, seperti kartu SD atau penyimpanan internal. WRITE_EXTERNAL_STORAGE merupakan kode izin tingkat berbahaya yang memungkinkan penyimpanan eksternal ditulis eksternal oleh aplikasi apa pun [12].

3. *GET_ACCOUNTS*

"*GET_ACCOUNTS*" digunakan dalam mengakses data atau informasi dari akun atau kontak yang terdapat pada perangkat pengguna, jadi dengan adanya kode *permission* ini semua isi dari kontak seperti nomor telepon dan email akan dapat dengan mudah untuk diakses. Pada website developer.android.com, google juga mendapat peringatan bahwa kode permission ini berbahaya [4].

4. *SEND_SMS, RECEIVE_SMS dan READ_SMS*

SEND_SMS, RECEIVE_SMS, dan READ_SMS berada dalam kelompok *permission* yang sama dengan tingkat level yang berbahaya dan dapat mengekspos pengguna perangkat Android ke serangan malware seperti *Short Message Service* (SMS) apabila izin diberikan. *Fake Player* adalah salah satu malware Android yang menghasilkan panggilan dan SMS [12].

2.2.2 Aplikasi

Aplikasi adalah suatu perangkat lunak yang memiliki perintah agar dapat mengolah setiap data sehingga menghasilkan yang namanya *input* dan *output*. Aplikasi memiliki berbagai macam komponen yang terdiri dari beberapa kolom form yang dirancang dengan baik dan sederhana sehingga menghasilkan suatu tampilan

yang menarik agar dapat membuat pengguna lebih mudah dalam menjalankannya. Aplikasi juga dapat melakukan segala bentuk pekerjaan sesuai dengan perintah yang dilakukan oleh klien aplikasi [13]. Kesimpulannya yaitu aplikasi merupakan suatu program perangkat lunak yang dioperasikan pada sistem tertentu yang berguna bagi manusia [14].

2.2.3 Metode *Prototyping*

Prototyping merupakan suatu metodologi dalam perencanan perangkat lunak yang dimana sebelum ke tahap pengembangan dilaksanakan, suatu program dapat dicoba penggunaannya terlebih dahulu. Metode *prototyping* terdiri dari beberapa rincian yaitu komunikasi, perencanaan dan perancangan, implementasi, evaluasi dan terakhir pengujian. Pada tahap komunikasi terdiri dari pengumpulan dan identifikasi kebutuhan dan garis besar mengenai program yang dibuat. *Prototyping* menerapkan sistem yang secara konsisten ditingkatkan ke tingkat berikutnya. Adapun tahap perancangan yang telah dibuat masih bersifat sementara. Pada tahap implementasi *prototyping,* sering kali dilakukan evaluasi. Jika pengguna merasa prototypingnya sesuai dengan keinginan dan kebutuhan maka akan melanjutkan ke tahap selanjutnya, jika masih belum sesuai maka kembali lagi pada tahap satu dan dua. Pada tahap evaluasi dan pengujian sistem, pengguna melakukan evaluasi dan pengujian sistem tersebut apakah sudah sesuai, dengan asumsi demikian, maka sistem akan siap untuk digunakan [15]. Tahapan dalam metode prototyping terdiri dari:

1. Analisa Kebutuhan

   Analisa kebutuhan merupakan tahap yang pertama dilakukan untuk menganalisis dan mengidentifikasi kebutuhan dari sistem yang akan dibuat. Pada tahap ini akan didefinisikan yang akan menggunakan sistem seperti admin dan user yang akan terlibat dalam sistem.
2. Desain *Prototype*

   Tahap ini merupakan tahap yang berfokus pada tampilan suatu sistem. Pada tahap ini, perancang sistem membuat perancangan sementara

yang meliputi fitur menu yang cepat dan praktis, dan juga meliputi tampilan input dan output dari sistem yang akan dibuat.

3. Evaluasi desain

   Pada tahapan dilakukan pengecekan kembali terhadap desain *prototype* yang telah dibuat dengan bertujuan untuk memastikan sistem yang dirancang sudah sesuai dengan yang diinginkan. Apabila *prototype* yang dibangun belum sesuai dengan keinginan, maka akan dilakukan koreksi serta perbaikan kembali. Tahap ini akan memperbaiki tampilan input dan output yang belum sempurna ataupun penambahan fitur baru.

4. Pengkodean Sistem

   Prototype yang telah dievaluasi dan disetujui, maka akan dilakukan pengkodean sistem dengan menterjemahkannya ke dalam bahasa pemrograman yang akan digunakan lalu diimplementasikan dalam bentuk kode program. Sebelum mengimplementasikan pengkodean sistem pembuat sistem harus memahami terlebih dahulu bahasa pemrograman yang akan digunakan.

5. Pengujian Sistem

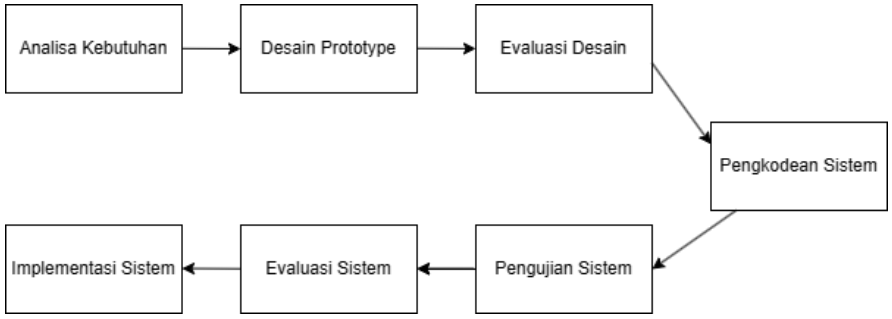
   Langkah selanjutnya yaitu pengujian sistem atau testing program. Sistem yang telah diubah ke dalam bahasa pemrograman dan apabila telah menjadi sebuah perangkat lunak, maka akan diuji terlebih dahulu untuk menentukan apakah sistem tesebut telah layak digunakan atau tidak. Pada penelitian ini menggunakan *black box testing* yaitu untuk menguji fungsionalitas dari tampilan suatu sistem apakah sudah berjalan dengan baik atau tidak.

6. Evaluasi sistem

   Pada tahap pengevaluasian ini dilakukan evaluasi untuk memastikan apakah program atau sistem yang sudah dibuat sudah sesuai dengan keinginan atau belum. Apabila telah sesuai maka sistem sudah dapat diimplementasikan. Namun apabila belum sesuai maka pengembang harus kembali ke tahap sebelumnya untuk memperbaiki ketidakseuaian itu hingga mencapai hasil yang diharapkan.

7. Implementasi Sistem

   Implementasi sistem merupakan tahap terakhir untuk *maintenance system* agar sistem terjaga dan berfungsi dengan baik. Sistem yang telah dibuat dan berhasil melewati tahapan evaluasi sistem dengan baik maka sistem tersebut sudah dapat digunakan [16]. Pada gambar dibawah ini merupakan alur atau tahapan metode *prototyping* dalam pengembangan sistem.



Gambar 2.1 Tahapan Metode *Prototyping*

2.2.4 JavaScript

Javascript pertama kali diperkenalkan oleh Netscape di tahun 1995. Pada awalnya browser Netscape Navigator 2 mendukung bahasa pemrograman JavaScript yang disebut sebagai "LiveScript". Javascript sendiri merupakan bahasa pemrograman yang terdiri dari berbagai macam skrip yang fungsinya dijalankan pada suatu dokumen HTML, bahasa pemrograman ini adalah bahasa pengaturan pertama yang digunakan dalam pembuatan web. Bahasa ini memberikan kemampuan pada HTML dengan mengizinkan eksekusi perintah di sisi user, dan itu berarti dari sisi browser bukan dari sisi server web [17]. Java Script juga dapat disematkan di dalam dokumen HTML atau dibuat menjadi dokumen sendiri kemudian dihubungkan dengan dokumen yang dituju [18].

Javasript adalah bahasa pemrograman untuk sisi klien. Bahasa pemrograman ini mendekati bahasa manusia atau bisa dikatakan bahasa tingkat tinggi, sehingga javascript tidak sulit untuk di pelajari. Javascript sendiri dibuat untuk menyempurnakan elemen-elemen pada website agar lebih dinamis, misalnya untuk

menampilkan dan menghilangkan objek-objek pada website kemudian memanfaatkan kemampuan JavaScript untuk memanggil kembali objek yang dihilangkan [19].

2.2.5 HTML (*Hypertext Markup Language*)

HyperText Markup Language (HTML) adalah bahasa pemrograman yang sudah sering digunakan untuk membantu membuat halaman pada web, dan kemudian diakses untuk menampilkan sebuah informasi di *browser*. HTML dapat digunakan sebagai penghubung antar file pada suatu halaman web menggunakan *localhost*, atau bisa juga disebut sebagai penghubung yang menghubungkan suatu website dalam dunia internet. Kemampuan HTML adalah untuk mengelola informasi dan data sehingga suatu dokumen dapat diakses oleh klien dan muncul di Internet melalui layanan web [20]. HTML juga merupakan bahasa sisi klien yang dapat menampilkan informasi-informasi dalam bentuk grafik, teks, serta multimedia serta dapat menampilkan tampilan halaman antarmuka (*hyperlink*) [14].

2.2.6 CSS (*Cascading Style Sheets*)

CSS adalah singkatan dari *Cascading Style Sheets* yaitu merupakan bahasa *web design* yang menyusun tampilan halaman web dengan menggunakan *markup language*. Mayoritas CSS digunakan untuk membuat tampilan dari HTML, namun kini CSS juga dapat digunakan untuk semuaa dokumen XML, termasuk diantaranya XUL dan SVG. CSS dapat memisahkan konten utama yang kita buat dengan tampilan yang meliputi desain, gaya teks dan warna. Hal ini dapat memperluas keterbukaan konten di web, sehingga memungkinkan untuk membagi sebuah halaman untuk formatting dan dapat mengurangi masalah dalam penulisan kode dan strukturnya. CSS membuat sebuah halaman yang dapat ditampilkan dalam berbagai gaya, dengan berbagai sorotan seperti *in-print*, *on-screen*, *by voice*, dan sebagainya. Selain itu, admin web dapat menentukan link yang akan menghubungkan antara konten dengan CSS [21]. Alasan mendasar CSS adalah untuk dapat membedakan konten dari dokumen dan dari tampilan sebuah dokumen, sehingga pembuatan ulang web akan lebih mudah untuk dibuat lagi. Pemformatan, warna, dan ukuran adalah bagian dari desain web. Oleh karena itu dengan adanya CSS, konten dan

desain web akan mudah untuk dibedakan, jadi sangat memungkinkan untuk melakukan pengulangan pada tampilan-tampilan web[20].

2.2.7 PHP (*Hypertext Preprocessor*)

PHP merupakan singkatan dari (Hypertext Preprocessor) adalah suatu bahasa pemrograman yang biasa digunakan dalam pembuatan website yang dapat mengolah data secara dinamis. PHP bersifat *server-side embedded script language* yang artinya, semua program yang telah dituliskan itu semuanya akan dijalankan oleh server, namun bisa dimasukkan ke dalam halaman HTML yang biasa digunakan. MySQL adalah salah satu sistem manajemen database yang umumnya dipakai bersama bahasa pemrograman PHP [22].

2.2.8 Visual Studio Code

Visual Studio Code atau biasa disebut VSCode adalah perangkat lunak kode program editor yang dikembangkan oleh Microsoft diantaranya itu untuk windows, linux, dan MacOS. VSCode mendukung berbagai macam Bahasa pemrograman seperti javascript, java, C++, nodejs. Fitur yang ada di VSCode ada banyak, diantaranya itu intellisense, debugging, git integration dan fitur axtensi yang sangat banyak sehingga dapat menambah kemampuan dan mempermudah dalam teks editor. Fitur-fitur tersebut akan bertambah terus-menerus seiring dengan bertambahnya versi yang ada pada Visual Studio Code yang dilakukan secara berkala setiap bulannya. Visual Studio Code ini juga bersifat *open source*, yang kode sumbernya dapat dilihat dan dapat berkontribusi dalam pengembanyannya. Kode sumber yang ada di VSCode dapat dilihat melalui link Github. Inilah mengapa VSCode memiliki banyak peminat dan menjadi favorit bagi para pengembang aplikasi [23].

2.2.9 Database

Database adalah suatu sistem yang di buat untuk mengolah, menyimpan dan menarik data dengan mudah. Database sendiri terdiri dari beberapa data yang terorganisir untuk 1 atau lebih penggunaan, dan dalam bentuk digital. Database digital diatur menggunakan Database Management System (DBMS), yang dapat menyimpan isi database, mengizinkan pembuatan dan mengolah data dan pencarian

data. Adapun beberapa Database yang sudah ada pada saat ini yaitu : Mysql, Sql Server, PostgreSql, Ms.Access, dan Oracle [24]. Berikut merupakan fungsi dari database antara lain:

1. Memudahkan pengelompokkan data, contohnya dengan membuat table yang berbeda-beda.
2. Memudahkan pengguna dalam menginput data yang baru.
3. Mengurangi risiko terjadinya suatu data ganda.
4. Mempermudah dalam menyimpan data secara digital

2.2.10 MySQL

MySQL atau biasa disebut juga dengan SQL yaitu singkatan dari *Structured Query Language* yang merupakan database server yang terkenal. MySQL dikhususkan untuk mengolah database mulai dari kecil sampai besar yang bersifat *open source*. SQL dapat diartikan sebagai antar muka standar untuk sistem manajemen, termasuk sistem yang berjalan pada komputer pribadi. SQL juga memungkinkan pengguna untuk mengetahui dimana lokasinya, atau bagaimana informasi tersebut. SQL lebih mudah dioperasikan jika dibandingkan dengan bahasa pemrograman, akan tetapi rumit digunakan apabila dibandingkan dengan *software* lembar kerja dan pengolah data lainnya. Sebuah SQL yang sederhana dapat menghasilka suatu permintaan untuk informasi yang tersimpan di komputer yang berbeda diberbagai lokasi, sehingga memerlukankan waktu dan sumber daya komputasi yang cukup banyak. SQL dirancang untuk mengirimankan perintah *query* (pengaksesan data berdasarkan pengalamatan tertentu) terhadap suatu database. SQL juga didukung oleh SMBD, seperti MySQL , MySQL Server, PostgreSQL, Oracle, dan Interbase [25].

2.2.11 Tailwinds CSS

Tailwind CSS merupakan bagian dari *framework* CSS yang digunakan oleh pengembang web untuk membangun tampilan web dengan cepat, seperti warna, *margin*, ukuran objek, dan dapat membuat situs website menjadi responsif dengan mudah [26].

2.2.12 *BlackBox Testing*

BlackBox testing adalah salah satu metode pengujian perangkat lunak yang terfokus pada spesifikasi fungsi-fungsi yang dikembangakan, yang terdiri dari struktur data, kesalahan inputan data dalam database, kesalahan GUI, dan beberapa kesalahan yang muncul nantinya. Penggunaan *BlackBox testing* memiliki beberapa keuntungan yaitu, pengguna tidak harus memahami bahasa pemrograman yang ada di dalam sistem, dan pengujian dilakukan berdsarkan sudut pandang dari pengguna. Jadi, *BlackBox testing* ini dapat memudahkan untuk melihat suatu perangkat lunak apakah dapat berjalan dengan baik [27]. Dalam pengujian *BlackBox testing* itu terdapat beberapa cara salah satu caranya yaitu dengan mengunakan metode *Equivalence Partitions*. *Equivalence Partitions* adalah sebuah pengujian yang berdasar pada inputan data di setiap form yang terdapat pada aplikasi deteksi anomali kode program terbaik, di setiap menu inputan akan dilakukan pengujian [28]. Adapun tahapan pada *blackbox testing* antara lain:

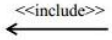
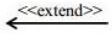
1. Membuat skenario pengujian sistem atau *test case* untuk menguji fungsionalitas dari sistem.
2. Membuat skenario pengujian sistem atau *test case* untuk menguji ketetapan alur kerja suatu fungsi pada sistem apakah sesuai denga napa yang dibutuhkan atau tidak.
3. Menemukan *error* atau *bug* yang terdapat pada tampilan aplikasi.

Black Box testing biasanya berfokus terhadap pada 5 hal, yaitu menemukan fungsi yang tidak benar, kesalahan pada tampilan antarmuka *(interface)*, menemukan kesalahan pada basis data, menemukan kesalahan pada *performance* sistem, dan terakhir menemukan kesalahan pada *inisialisasi* [29].

2.2.13 *Use Case* Diagram

Use case adalah langkah pertama yang dilakukan dalam memodelkan sebuah sistem. *use case* juga merupakan pemodelan kebutuhan sebuah sistem fungsional, setiap *use case* digambarkan sebagai kunci dari suatu skenario yang dilakukan oleh aktor dan dibuat ringkas dalam sebuah batas sistem, setiap *use case*

dihubungkan dengan sebuah garis notasi. *Use case diagram* membantu dalam pemodelan sistem dengan mengidentifikasi actor apa saja yang terlibat, tindakan atau proses yang dilakukan oleh aktor, serta hubungan antara aktor. *Use case* diagram memberikan gambaran yang jelas tentang kebutuhan *user*, fungsi yang diharapkan, dan batasan sistem yang sedang dikembangkan, sehingga meudahkan pemahaman dan komunikasi antara pemangku kepentingan dalam pengembangan perangkat lunak [30].

Simbol	Keterangan
	Aktor : Mewakili peran orang, sistem yang lain, atau alat ketika berkomunikasi dengan *use case*
	*Use case* : Abstraksi dan interaksi antara sistem dan aktor
	Association : Abstraksi dari penghubung antara aktor dengan use case
	Generalisasi : Menunjukkan spesialisasi aktor untuk dapat berpartisipasi dengan use case
	Menunjukkan bahwa suatu use case seluruhnya merupakan fungsionalitas dari use case lainnya
	Menunjukkan bahwa suatu use case merupakan tambahan fungsional dari use case lainnya jika suatu kondisi terpenuhi

Gambar 2.2 Simbol-simbol *Use case* Diagram

2.2.14 *Activity Diagram*

Activity diagram adalah diagram yang dapat menggambarkan proses-proses yang terjadi pada sebuah sistem. Runtutan proses dari suatu sistem digambarkan secara vertikal. *Activity diagram* merupakan pengembangan dari *Use Case* diagram yang memiliki alur aktivitas. Tahapan atau aktivitas dapat berupa runtutan fitur-fitur atau proses bisnis yang terdapat di dalam sistem tersebut. Kesimpulannya yaitu pembuatan *Activity diagram* hanya dapat dipakai untuk menggambarkan atau memodelkan alur atau aktivitas dari sistem [30].