BAB II
TINJAUAN PUSTAKA

2.1 Penelitian Terdahulu

Rujukan penelitian pertama yaitu penelitian oleh Khonsa Salsabila, Fetty Tri Anggraeny, Agung Mustika Rizki (2022), diketahui bahwa pengujian menggunakan metode *Black Box Testing Equivalence Partitions,* berfokus pada masukan data pada setiap form dalam sistem. Setiap form masukan diuji dan dikelompokkan berdasarkan fungsinya, baik yang sesuai maupun tidak, sehingga pengujian ini dapat membantu pembuatan test case, memastikan kualitas perangkat lunak, serta dapat menemukan kesalahan yang sebelumnya tidak terdeteksi [8].

Rujukan penelitian kedua yaitu penelitian oleh Andrea Arcuri (2020), menyimpulkan bahwa dengan melakukan pengujian menggunakan metode *Black Box* pada RESTful API telah terbukti dapat dengan mudah menemukan kesalahan pada RESTful API dan sebagian besar ditemukan karena validasi input yang kurang baik [9].

Rujukan penelitian ketiga yaitu penelitian oleh Anshu Soni dan Virender Ranga (2019), penelitian ini dilakukan percobaan pengujian *Black Box* menggunakan Postman untuk membandingkan arsitektur REST dan SOAP dan mengevaluasi kinerja API. Diketahui bahwa pengujian API dengan menggunakan Postman dapat memastikan bahwa API berfungsi dengan benar dan memberikan *response* yang sesuai ketika *request* API dilakukan dan dapat melakukan *request* HTTP seperti GET, POST, PUT, dan Patch pada API [10].

Tabel 2. 1 Penelitian Terdahulu

No	Peneliti	Judul Penelitian	Metode	Pembahasan	Hasil
1	Khonsa Salsabila, Fetty Tri Anggraeny, Agung Mustika Rizki	Pengujian Sistem Pendukung Keputusan Penentuan Jurusan Pada Siswa SMA	*Black Box Testing Equivalence Partitions*	Penelitian ini menggunakan metode Black Box Testing Equivalence Partitions untuk menguji sistem	Pengujian menggunakan metode Black Box Testing Equivalence Partitions dapat

		Dengan Menggunakan Metode Black Box Berbasis Equivalence Partitions		pendukung keputusan penentuan jurusan siswa SMA. Prosesnya mencakup pemilihan fungsionalitas yang diuji, perancangan skenario, pemilihan data uji, menentukan input dari struktur basis data, melaksanakan pengujian, dan menyimpulkan hasil.	membantu pembuatan test case, memastikan kualitas perangkat lunak, serta dapat menemukan kesalahan yang sebelumnya tidak terdeteksi
2	Andrea Arcuri	Automated Black- And White-Box Testing of RESTful APIs with EvoMaster	*Automated Black Box Testing dan White Box Testing*	Penelitian ini membahas tentang pengujian otomatis pada RESTful API dengan menggunakan metode black box dan white box. Penelitian ini menunjukkan bagaimana pengujian otomatis dapat menemukan beberapa bug dalam API RESTful.	Pengujian otomatis menggunakan metode Black Box pada RESTful API telah terbukti efektif dalam mendeteksi kesalahan pada RESTful API dengan biaya yang terjangkau, terutama dalam hal validasi input yang kurang memadai.
3	Anshu Soni dan Virender Ranga	API features individualizing of web services: REST and SOAP	*Black Box Testing*	Pada penelitian ini membahas tentang pengujian API pada arsitektur REST dan SOAP.	Diketahui bahwa pengujian API menggunakan Postman

				Penelitian ini akan melakukan pengujian dengan metode Black Box dengan menggunakan *tools* Postman pada kedua arsitektur API tersebut.	dapat memverifikasi bahwa API beroperasi dengan tepat dan memberikan response yang sesuai saat request API dilakukan.

2.2 Pengujian Perangkat Lunak

Pengujian perangkat lunak merupakan proses yang dilakukan secara sistematis dalam pengembangan perangkat lunak untuk memastikan keabsahan dan validitas hasil perangkat lunak yang telah dikembangkan. Tahapan pengujian merupakan metode untuk menemukan kekurangan dan kesalahan yang terjadi selama tahap pengembangan [11]. Pengujian perangkat lunak adalah tahapan paling penting untuk memastikan kualitas perangkat lunak sesuai dengan kebutuhan. Dikarenakan apabila terdapat kesalahan atau kekurangan yang ditemukan selama proses pengujian, maka dapat dilakukan perbaikan dan pengembangan perangkat lunak lebih lanjut.

Dasar pengujian (*test basis*) memiliki peran penting sebagai indikator utama dalam mencapai tujuan pengujian perangkat lunak. Dasar pengujian yang sesuai dengan tingkat tes yang dapat dipertimbangkan melibatkan analisis terhadap beberapa faktor, seperti spesifikasi persyaratan, informasi desain dan implementasi, implementasi komponen atau sistem, serta laporan analisis risiko. Spesifikasi persyaratan mencakup persyaratan bisnis, fungsional, sistem, cerita pengguna, kasus penggunaan, atau produk sejenis yang menentukan komponen fungsional dan non-fungsional serta perilaku sistem. Informasi desain dan implementasi mencakup diagram, dokumen arsitektur sistem atau perangkat lunak, spesifikasi desain, grafik aliran panggilan, diagram pemodelan (seperti UML atau ERD), spesifikasi antarmuka, atau produk sejenis yang menentukan struktur komponen atau sistem. Implementasi komponen atau sistem melibatkan kode, metadata database, *query*, dan antarmuka.

Laporan analisis risiko mempertimbangkan aspek fungsional, non-fungsional, dan struktural dari komponen atau sistem [12].

Salah satu dari beberapa metode pengujian perangkat lunak yang dapat digunakan untuk menguji perangkat lunak atau *software* adalah metode pengujian *Black Box Testing*. Metode *Black Box Testing* difokuskan pada pengujian fungsionalitas perangkat lunak, yaitu perilaku perangkat lunak terhadap input yang dimasukkan oleh pengguna sehingga dapat memeriksa apakah hasil yang dikeluarkan sesuai dengan yang diharapkan [5].

Black Box Testing adalah metode pengujian perangkat lunak yang hanya bertujuan untuk mengevaluasi apakah program sesuai dengan fungsi yang diinginkan tanpa memperhatikan kode program yang digunakan [13]. Metode pengujian *Black Box Testing* memiliki kemampuan untuk menemukan beberapa jenis kesalahan seperti fungsionalitas yang tidak benar atau tidak ada, kesalahan pada struktur data, kesalahan akses basis data, kesalahan antarmuka, serta kesalahan pada kinerja perangkat lunak [14].

2.3 Equivalence Partitions

Dalam melakukan pengujian perangkat lunak, penting untuk memilih teknik yang sesuai dan mampu menemukan kesalahan yang belum terdeteksi untuk meningkatkan kualitas perangkat lunak. Salah satu teknik pengujian dalam metode *Black Box Testing* yang dapat digunakan adalah *Equivalence Partitions*. Metode pengujian *Black Box Testing Equivalence Partitions* adalah sebuah teknik pengujian perangkat lunak di mana input dari setiap form akan dikelompokkan berdasarkan fungsinya dan diuji baik yang valid maupun tidak valid [15].

Dengan menggunakan metode *Black Box Testing Equivalence Partitions*, proses pembuatan test case pengujian dapat dimudahkan dan kualitas pengujian dapat ditingkatkan, sekaligus memungkinkan untuk menemukan kesalahan yang tidak terdeteksi sebelumnya yang mungkin disebabkan oleh kesalahan pengetikan. [16]. Ada beberapa tahapan utama dalam teknik ini, yaitu menentukan *use case* yang akan diuji, menentukan kriteria, mendefinisikan partisi, membuat data uji, membuat kasus uji, melakukan pengujian, dan melakukan evaluasi [17]. Dengan beberapa tahapan tersebut

, dapat memperoleh hasil yang sangat terperinci, sehingga pada tahapan evaluasi dapat dijalankan dengan efektif untuk pengembangan perangkat lunak. Penjelasan dari urutan tersebut dijelaskan pada poin-poin berikut:

2.3.1 Menentukan Use Case

Pada tahapan ini, akan dilakukan identifikasi dan pembuatan use case untuk fitur-fitur yang akan diuji.

2.3.2 Menentukan Kriteria

Setelah use case ditentukan, langkah selanjutnya adalah menentukan kriteria yang berdasarkan pada use case yang telah dipilih pada tahap pertama.

2.3.3 Mendefinisikan Partisi

Setelah menentukan kriteria, tahapan selanjutnya adalah mendefinisikan partisi. Hal ini dilakukan dengan membagi setiap kriteria menjadi beberapa bagian berdasarkan karakteristiknya.

2.3.4 Membuat Data Uji

Setelah melakukan definisi partisi, langkah berikutnya adalah menyusun tabel data uji. Pada tahap pembuatan data uji, dibuat tabel yang berisi data yang akan dimasukkan ke dalam *field* saat pengujian dilakukan.

2.3.5 Membuat Kasus Uji

Pada tahap kelima, dilakukan pembuatan kasus uji berdasarkan daftar data uji yang telah dibuat sebelumnya. Untuk memudahkan pengujian pada tahap selanjutnya, kasus uji dapat dibuat dalam bentuk tabel. Tahap ini mencakup pembuatan scenario pengujian yang memanfaatkan data uji dengan berbagai jenis uji yang berbeda dan disesuaikan pada hasil pengujian yang diharapkan.

2.3.6 Pengujian

Pada tahapan selanjutnya adalah pengujian, akan dilakukan dengan mengacu pada skenario uji yang terdapat pada tabel kasus uji. Pada tahapan ini, akan terdapat tabel hasil pengujian untuk masing-masing *end point*, yang berisi daftar kasus uji dan hasil uji yang telah dilakukan.

2.3.7 Evaluasi

Pada tahapan terakhir yaitu evaluasi, akan dilakukan perhitungan untuk menghitung nilai efektivitas setiap tabel *test case* dan nilai efektivitas keseluruhan pengujian. Selain itu, untuk mengetahui tingkat pencapaian hasil pengujian, akan menggunakan kategori kelayakan yang telah ditentukan (Arikunto, 2009:44). Hasil evaluasi tersebut akan digunakan untuk menyimpulkan apakah pengujian telah mencapai nilai efektivitas yang diharapkan oleh penguji berdasarkan kategori kelayakan yang ditetapkan.

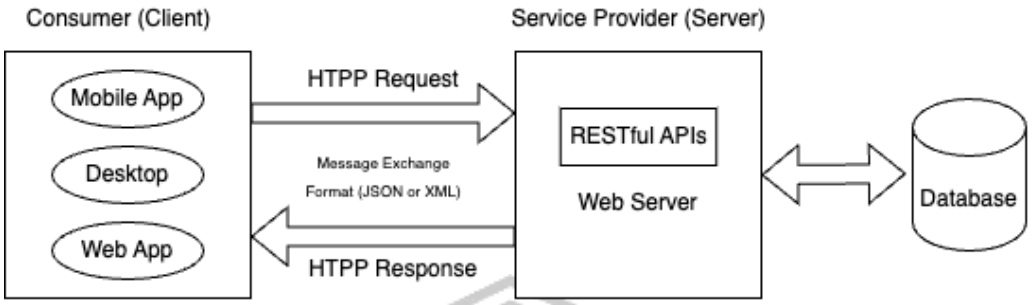
2.4 Postman

Postman adalah sebuah *tools* kolaborasi untuk pengembangan API. Beberapa fitur yang dapat dilakukan oleh Postman antara lain sebagai client yang dapat mengakses REST secara langsung, pengujian otomatis, simulasi endpoint langsung, dan dokumentasi API [18]. Postman digunakan oleh lebih dari tiga juta insinyur dan pengembang perangkat lunak di seluruh dunia untuk membangun perangkat lunak yang terhubung melalui API. [19].

Pada pengujian kali ini digunakan alat yang disebut Postman. Alat ini dipilih sebagai alat pengujian karena mudah digunakan dan memiliki fitur pengujian otomatis dalam menguji rancangan API yang dibuat. Dengan menggunakan alat ini, dapat mempercepat waktu dan efisiensi dalam mengetahui hasil dari pengujian API pada website Monitoring Kartu Santri.

2.5 RESTful API

REST atau RESTful (*Representationl Satate Transfer*) adalah gaya arsitektur yang umum digunakan dalam pengembangan layanan web, di mana sistem dapat mengakses dan mengelola representasi teks dari sumber daya dengan menggunakan HTTP (*Hypertext Transfer Protocol*) sebagai protokol untuk komunikasi data [20]. RESTful API mengimplementasikan satu atau lebih operasi *create*, *read*, *update*, atau *delete* (CRUD) terhadap *resource* (sumber daya /data) tertentu. Operasi-operasi ini umumnya dihubungkan dengan HTTP methods seperti POST, GET, PUT, dan DELETE [21].



Gamber 2. 1 Arsitektur RESTful API

Dalam arsitektur REST, REST *server* menyediakan *resource* (sumber daya /data) dan REST *client* mengakses serta menampilkan *resource* (sumber daya /data) tersebut untuk penggunaan lebih lanjut. *Resource* tersebut direpresentasikan dalam bentuk format teks, JSON atau XML. Secara umum, format yang paling umum digunakan adalah JSON dan XML, namun pada penelitian ini peneliti menggunakan JSON sebagai formatnya. Setiap *resource* (sumber daya /data) diidentifikasi menggunakan URI (Universal Resource Identifier) atau global id [22].