BAB II
TINJAUAN PUSTAKA

2.1 Teori Umum

2.1.1 Pengertian Rancang Bangun

Rancang adalah proses merancang sistem baru yang dapat mengatasi masalah perusahaan dengan memilih alternatif sistem terbaik. Bangun adalah proses membangun sistem baru atau memperbaiki secara keseluruhan sistem yang sudah ada (Pressman 2019). Penggambaran, perencanaan, dan pembuatan sketsa serta pengaturan dari beberapa komponen terpisah menjadi satu kesatuan yang utuh dan berfungsi adalah semua bagian dari rancang bangun. Dapat disimpulkan, Rancang bangun adalah proses menerjemahkan hasil analisis ke dalam perangkat lunak, serta melakukan perbaikan dan pembangunan sistem yang sudah ada.

2.1.2 Pengertian Sistem Informasi

Sistem adalah gabungan komponen yang saling berinteraksi untuk memproses masukan dan mengelola informasi tersebut hingga menghasilkan keluaran. Informasi mengacu pada data yang telah diubah menjadi bentuk yang lebih relevan dan bermanfaat bagi penerimanya (Kristanto, 2018). Sistem informasi terdiri dari sejumlah prosedur, seperti pengumpulan data, pengelolaan data yang tersimpan, dan penyebaran data. Dapat disimpulkan, sistem informasi merupakan kombinasi orang, data, proses, dan teknologi, seperti komputer, yang mendukung proses pengambilan keputusan dan membantu perusahaan mencapai tujuannya.

2.1.3 Pengertian *Monitoring*

Monitoring merupakan proses mengumpulkan informasi dan data secara periodik untuk melacak kemajuan suatu program atau kegiatan, mengukur kinerja, dan memastikan bahwa program atau kegiatan tersebut berjalan sesuai dengan rencana dan mencapai tujuan yang telah ditetapkan (Anang, 2020). Tujuannya adalah untuk memantau dan memeriksa suatu kondisi atau aktivitas secara terus-menerus guna memastikan bahwa segala hal berjalan sesuai dengan yang diharapkan. *Monitoring* dilakukan dengan mengaplikasikan perangkat lunak khusus yang dapat memantau dan merekam data, serta memberikan notifikasi atau tindakan yang diperlukan jika terjadi masalah atau ketidaksesuaian dengan standar yang telah ditetapkan.

2.2 Teori Khusus

2.2.1 *System Development Life Cycle* (SDLC)

SDLC adalah proses interpretasi sistem informasi yang memiliki kemampuan untuk membantu operasi bisnis selama proses perancangan sistem. Pembuatan, perancangan, dan pengiriman sistem kepada pengguna untuk memenuhi kebutuhan bisnis merupakan bagian dari proses penerapan SDLC. SDLC terdiri dari empat tahap utama yang digunakan untuk mengembangkan sebuah sistem, yaitu perencanaan, analisis, desain, dan implementasi. Penjabaran dari keempat tahap SDLC adalah sebagai berikut (Dennis et al., 2020):

a. Perencanaan (*Planning*)

Tahap pertama dalam pengembangan dan pembuatan sistem adalah perencanaan. Tujuan dari tahap ini adalah untuk memahami alasan perusahaan untuk membangun sistem informasi. Analisis kelayakan dan pengendalian arah proyek melalui fase SDLC adalah langkah-langkah yang harus dilakukan pada tahap perencanaan. Pada tahap perencanaan, analisis kelayakan mencakup kelayakan ekonomi, ide teknis, dan organisasi (Dennis et al., 2020).

b. Analisis (*Analysis*)

Tahap analisis dilakukan dengan tujuan untuk mengevaluasi sistem yang sedang digunakan. Hasil analisis dapat digunakan sebagai dasar untuk perencanaan dan pengambilan keputusan dalam pengembangan sistem. Untuk mendukung tujuan tahap analisis, beberapa kegiatan dilakukan dalam tahap ini. Kegiatan ini meliputi evaluasi sistem yang sedang berjalan, pengumpulan kebutuhan sistem baru, dan perancangan sistem baru. Evaluasi sistem yang sedang berjalan dilakukan untuk mendapatkan pemahaman tentang kondisi sistem saat ini dan sebagai referensi untuk proses pembuatan sistem baru. Survei atau kuesioner digunakan untuk mengumpulkan kebutuhan sistem baru untuk mengetahui operasi bisnis yang terkait dengan sistem yang akan dibangun. Perancangan sistem baru juga memerlukan persetujuan pengguna dan persyaratan bisnis yang harus dipenuhi oleh sistem baru. Persyaratan ini kemudian diuraikan dalam proposal sistem (Dennis et al., 2020).

c. Desain (*Design*)

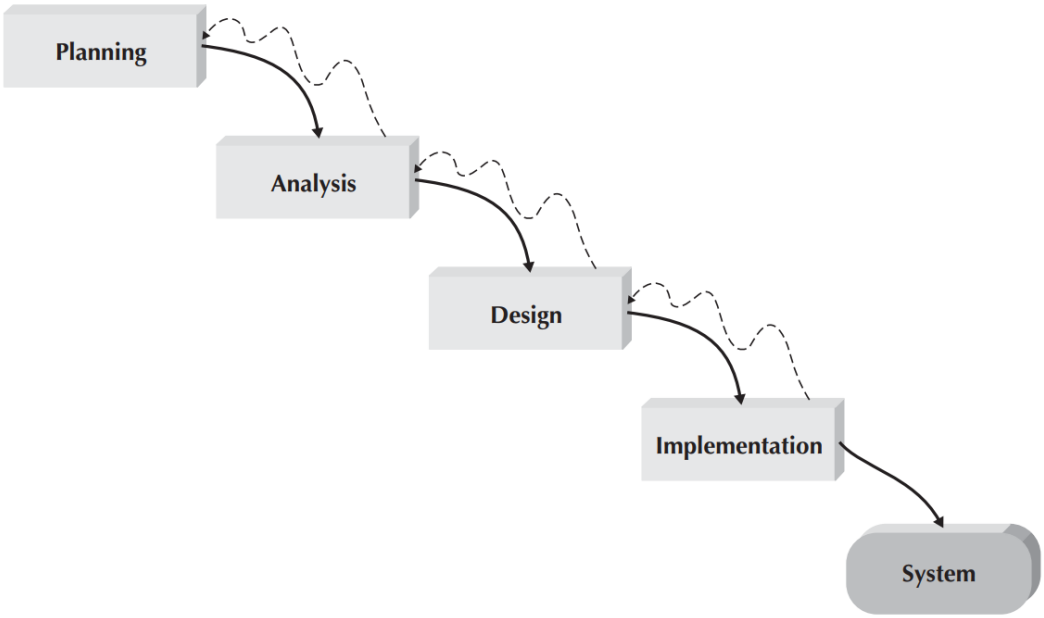
Tahap desain adalah tahap perancangan sistem yang berfokus pada spesifikasi rancangan. Hasil akhir dari tahap ini adalah rancangan kontrol, sistem, basis data, masukan, dan rancangan keluaran. Verifikasi sistem yang akan dibangun adalah tahap pertama dalam proses desain. Tahap kedua melibatkan pembuatan desain arsitektur dasar, yang menunjukkan infrastruktur dan perangkat yang akan digunakan, menentukan spesifikasi sistem yang akan dibangun, dan menentukan lokasi basis data. Pada tahap terakhir, pengembangan sistem dilakukan dengan penentuan program yang akan digunakan selama tahap implementasi (Dennis et al., 2020).

d. Implementasi (*Implementation*)

Tahap implementasi dimulai dengan pengkodean sistem informasi yang dapat dioperasikan oleh komputer menggunakan bahasa pemrograman. Tahap ini terdiri tiga langkah, yaitu pembangunan sistem melalui bahasa pemrograman dan pengujian sistem yang dilakukan secara bersamaan untuk mencegah kesalahan atau bug. Langkah kedua melibatkan konversi dan pelatihan pengguna, dimana konversi dapat dilakukan dengan menggunakan salah satu dari tiga strategi, yaitu konversi bertahap, konversi paralel, dan *cutover* langsung. Langkah terakhir adalah membuat penetapan rencana dukungan sistem, yang mencakup tujuan pasca-implementasi dan memberikan buku panduan pengguna untuk penggunaan sistem (Dennis et al., 2020).

2.2.2 Metode *Waterfall*

Metode pengembangan *waterfall* adalah sebuah metode pengembangan yang mengikuti struktur desain dengan melakukan setiap fase secara bertahap, dari satu fase ke fase berikutnya (Dennis et al., 2015). Metode *waterfall* tidak memiliki fase yang tumpang tindih, setiap fase harus diselesaikan sebelum fase berikutnya dapat dimulai. Metode ini sering digambarkan seperti air terjun. Salah satu kelebihan dari model *waterfall* adalah kemampuannya untuk meminimalkan kesalahan, karena kesalahan dapat teridentifikasi pada setiap tahap dan dapat dikoreksi pada tahap sebelumnya jika terdapat perubahan. Metode *waterfall* terdiri dari lima tahap, yaitu perencanaan, analisis, desain, implementasi, dan sistem (Dennis et al., 2015). Pada **Gambar 2.1** dapat dilihat lima tahapan *waterfall*.



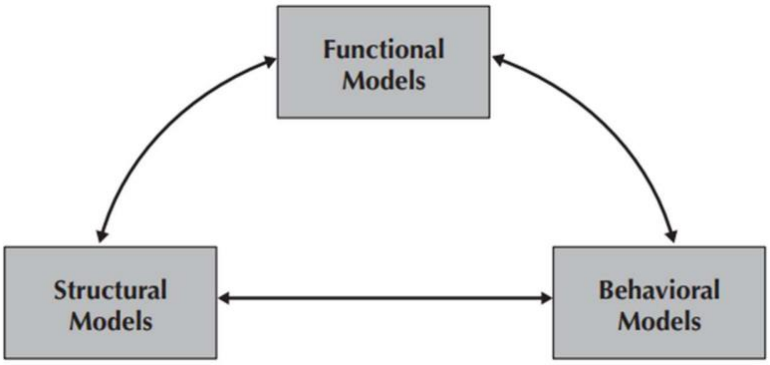
Gambar 2.1 Metode Pengembangan *Waterfall* (Dennis, Wixom, & Tegarden, 2015)

2.2.3 *Object Oriented Analysis Design* (OOAD)

OOAD adalah metode analisis yang menilai kebutuhan dari sudut pandang kelas dan objek dalam konteks masalah sistem dan subsistem (Nugroho, 2017). Metode ini digunakan dalam pengembangan dan analisis perangkat lunak dengan pendekatan berorientasi objek. Pendekatan ini merupakan paradigma baru dalam pengembangan sistem, di mana sistem dianggap sebagai kumpulan objek yang berinteraksi dan dikendalikan oleh perilakunya (Purwaningtias, 2018).

2.2.4 *Unified Modelling Language* (UML)

Unified modeling language adalah bahasa yang sering digunakan di industri untuk menjelaskan persyaratan, merancang arsitektur, dan melakukan analisis dalam konteks pemrograman berorientasi objek. UML juga dianggap sebagai bahasa yang fleksibel dalam melakukan analisis dan pengembangan karena memiliki notasi yang konsisten secara keseluruhan. UML membuat representasi berorientasi objek dengan berbagai jenis diagram untuk memodelkan setiap aspek proyek pengembangan sistem, dari analisis hingga implementasi (Dennis et al., 2015). Berikut pada **Gambar 2.2** *Unified modeling language* (UML) memiliki tiga jenis model diagram.



Gambar 2.2 Jenis *Unified Modeling Language* (Dennis, Wixom, & Tegarden, 2015)

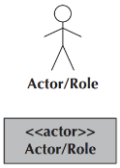
A. *Functional Model*

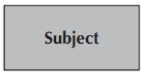
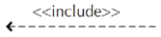
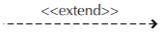
Functional model dalam UML memiliki peran dalam mengilustrasikan proses bisnis dan bagaimana sistem informasi berinteraksi dengan lingkungannya. Dalam pendekatan OOAD, model fungsional direpresentasikan melalui *use case diagram*, *use case description*, dan *activity diagram*.

- *Use Case Diagram*

  Use case diagram adalah representasi visual yang menyajikan cara sistem berinteraksi dengan aktornya secara dasar. Interaksi antara aktor dan sistem direpresentasikan dalam bentuk kegiatan. Aktivitas yang dilakukan oleh aktor dengan sistem yang dijelaskan melalui *use case* dan direpresentasikan dengan notasi atau simbol tertentu. Seluruh *use case* akan dijelaskan secara rinci dalam dokumen terpisah yang disebut spesifikasi *use case*, yang memuat informasi detail dari setiap *use case* yang terdapat dalam diagram *use case* (Dennis et al., 2015). Notasi yang digambarkan dalam *use case diagram* sebagai berikut.

Tabel 2.1 Notasi *Use Case Diagram*

Simbol	Notasi	Keterangan
	*Actor*	Notasi yang menggambarkan orang atau sistem yang memperoleh manfaat dan berada di luar subjek.
	*Use Case*	Notasi yang menggambarkan bagian utama dari fungsionalitas sistem.

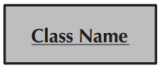
	*Subject Boundary*	Notasi yang mewakili ruang lingkup subjek, misalnya sistem atau proses bisnis individual.
	*Assossiation Relationship*	Notasi yang menghubungkan aktor dengan use case yang berinteraksi dengannya.
	*Include Relationship*	Notasi yang mewakili masuknya fungsionalitas dari satu use case ke use case lainnya.
	*Extend Relationship*	Notasi yang mewakili perluasan use case untuk menyertakan perilaku opsional.
	*Generalization Relationship*	Notasi yang mewakili kasus penggunaan khusus ke kasus penggunaan yang lebih umum.

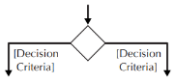
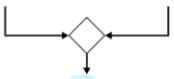
Sumber: (Dennis, Wixom, & Tegarden, 2015)

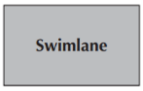
- *Activity Diagram*

Activity diagram sering digunakan untuk mensimulasikan perilaku dalam suatu proses bisnis, dan menggambarkan hubungan antara aktivitas dalam proses tersebut (Dennis et al., 2015). Saat membuat *activity diagram*, penting untuk merujuk kembali pada kebutuhan sistem yang akan dirancang dan dokumentasi yang telah dibuat, seperti *use case diagram* dan lainnya. Selain itu, diperlukan juga penentuan aktivitas yang diperlukan untuk mendukung suatu proses bisnis. Notasi yang digambarkan dalam *activity diagram* sebagai berikut.

Tabel 2.2 Notasi *Activity Diagram*

Simbol	Notasi	Keterangan
	*Activity*	Notasi yang digunakan untuk mewakili serangkaian tindakan.
	*Object*	Notasi yang digunakan untuk mewakili suatu objek yang terhubung ke sekumpulan aliran objek

	Control Flow	Notasi yang menunjukkan urutan eksekusi.
	Object Flow	Notasi yang menunjukkan aliran suatu objek dari satu aktivitas ke aktivitas lainnya.
	*Initial Node*	Notasi yang menggambarkan awal serangkaian tindakan atau aktivitas.
	*Final activity Node*	Notasi yang digunakan untuk menghentikan semua aliran kontrol dan aliran objek dalam suatu aktivitas.
	*Final-flow Node*	Notasi yang digunakan untuk menghentikan aliran kontrol atau aliran objek tertentu.
	*Decision Node*	Notasi yang digunakan untuk mewakili kondisi pengujian untuk memastikan bahwa aliran kontrol atau aliran objek hanya menempuh satu jalur.
	*Merge Node*	Notasi yang digunakan untuk menyatukan kembali jalur keputusan berbeda yang dibuat menggunakan simpul keputusan.
	*Fork Node*	Notasi yang digunakan untuk membagi perilaku menjadi serangkaian aliran aktivitas yang paralel atau bersamaan.
	*Join Node*	Notasi yang digunakan untuk menyatukan kembali serangkaian aliran aktivitas yang paralel atau bersamaan.

	*Swimlane*	Notasi yang digunakan untuk memecah diagram aktivitas menjadi baris dan kolom untuk ditetapkan aktivitas individu kepada individu atau objek yang bertanggung jawab untuk melaksanakan aktivitas.

Sumber: (Dennis, Wixom, & Tegarden, 2015)

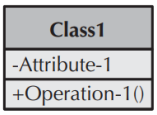
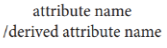
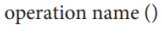
B. *Structural Model*

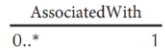
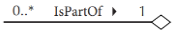
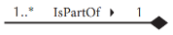
Structural model menggambarkan bagaimana objek-objek yang ada mendukung proses bisnis dalam perusahaan. Sementara *functional model* merepresentasikan perilaku sistem, model struktur fokus pada pemahaman tentang informasi yang digunakan dan dihasilkan oleh sistem, seperti informasi pengguna, transaksi, dan sebagainya. Dalam pendekatan OOAD, model struktur direpresentasikan melalui *class diagram.*

- *Class Diagram*

  Class diagram adalah representasi visual yang menampilkan klasifikasi dan hubungan antar kelas yang konsisten didalam sistem. *Class diagram* menggambarkan kelas beserta perilaku dari kelas tersebut, seperti metode atau operasi (Dennis et al., 2015). Notasi yang digambarkan dalam *class diagram* sebagai berikut.

Tabel 2.3 Notasi *Class Diagram*

Simbol	Notasi	Keterangan
	*Class*	Notasi yang mewakili jenis orang, tempat, atau sesuatu yang perlu ditangkap dan disimpan oleh sistem informasi
	*Attribute*	Notasi yang merupakan properti yang menggambarkan keadaan suatu obyek.
	*Operation*	Notasi yang mewakili tindakan atau fungsi yang dimiliki suatu kelas dapat tampil.

	*Association*	Notasi yang merupakan hubungan antara banyak kelas atau kelas itu sendiri.
	*Generalization*	Notasi yang merupakan semacam hubungan antara beberapa kelas.
	*Aggregation*	Notasi yang merupakan bagian dari hubungan yang logis antara beberapa kelas atau kelas itu sendiri.
	*Composition*	Notasi yang merupakan bagian fisik dari hubungan antara beberapa kelas atau kelas itu sendiri.

Sumber: (Dennis, Wixom, & Tegarden, 2015)

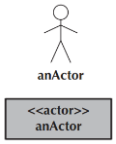
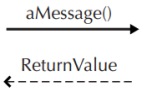
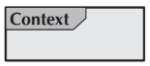
C. *Behavioral Model*

Behavioral model dalam UML dirancang untuk menjelaskan komponen dinamis internal sistem yang mendukung proses bisnis didalam organisasi. Dalam pemodelan perilaku, *use case diagram* pada pemodelan fungsional digunakan sebagai referensi, karena tujuan dari pemodelan perilaku adalah untuk menunjukkan bagaimana sistem dan pengguna sistem informasi terkait berinteraksi satu sama lain. Pemodelan perilaku menggunakan diagram interaksi seperti *sequence diagram.*

- *Sequence Diagram*

Sequence diagram memberikan gambaran objek yang terlibat dalam suatu *use case* dan pesan yang dikirim antara mereka selama proses use case tersebut berlangsung. Oleh karena itu, *sequence diagram* adalah model dinamis yang menggambarkan urutan pesan-pesan yang dikirim oleh objek dalam suatu interaksi tertentu. *Sequence diagram* juga memberikan urutan umum yang menggambarkan semua kemungkinan skenario untuk suatu *use case* (Dennis et al., 2015). Dalam pembuatan *sequence diagram*, dapat merujuk pada *use case diagram* yang telah dibuat. Notasi yang digambarkan dalam *sequence diagram* sebagai berikut.

Tabel 2.4 Notasi *Sequence Diagram*

Simbol	Notasi	Keterangan
	*Actor*	Notasi seseorang atau sistem yang memperoleh manfaat dari dan berada di luar sistem.
	*Object*	Notasi yang berpartisipasi dalam suatu urutan dengan mengirim dan/atau menerima pesan.
	Lifeline	Notasi yang menunjukkan kehidupan suatu objek selama suatu urutan.
	Execution Occurrence	Notasi yang merupakan persegi panjang dan sempit yang ditempatkan di atas garis hidup.
	*Message*	Notasi yang menyampaikan informasi dari satu objek ke objek lainnya.
	Guard Condition	Notasi yang merupakan tes yang harus dipenuhi agar pesan dapat dikirim.
	Object Destruction	Notasi yang menunjukkan bahwa objek tersebut akan padam keberadaan yang ditempatkan di akhir garis hidup suatu objek.
	*Frame*	Notasi yang menunjukkan konteks diagram urutan.

Sumber: (Dennis, Wixom, & Tegarden, 2015)

2.2.5 Basis Data

Basis data adalah sarana penyimpanan yang memungkinkan akses data dengan lebih mudah. Basis data memegang peranan penting dalam pengelolaan data karena merupakan sumber informasi utama dan penentu keakuratan informasi (Rusmawan, 2019). Basis data dapat dijelaskan sebagai kumpulan informasi yang disimpan. Penggunaan basis data memberikan manfaat dalam hal kecepatan proses, kemampuan penggunaan bersama, pemeliharaan data, penghematan ruang penyimpanan, keakuratan, kelengkapan, dan keamanan informasi.

Pembuatan basis data melibatkan entitas dan atribut. Entitas merujuk pada objek atau hal yang diamati dalam basis data. Atribut merupakan nilai yang terkait dengan suatu entitas (Conger, 2015). Manajemen basis data dilakukan menggunakan sistem dengan sebutan *Database Management System* (DBMS). DBMS adalah teknik yang biasa digunakan untuk menyimpan, mengatur, dan mengakses data. DBMS sangat penting untuk memudahkan penggunaannya, seperti mengatur akses dan pemindahan data, memelihara dan mengelola data, mengelola transaksi data, mencegah konflik pembaruan data, mengawasi proses pencadangan data, mendukung bahasa kueri, dan menjaga keamanan data (Sukamto & Shalahuddin, 2018).

2.2.6 *Black Box Testing*

Black box testing atau pengujian fungsional adalah metode testing yang didasarkan pada spesifikasi program tanpa memerlukan pengetahuan tentang implementasi internal program. Dengan demikian, *black box testing* bertujuan untuk memastikan bahwa program memenuhi spesifikasi kebutuhan tanpa memperhatikan detail implementasi program (Bierig et al., 2021). *Black box testing* memiliki keuntungan dalam memisahkan antara pengembang dan pengujian, fokus pada kebutuhan pengguna, memungkinkan pengujian menyeluruh tanpa memperhatikan detail implementasi internal, mengurangi kompleksitas pengujian, dan membantu meningkatkan keamanan sistem dengan mengidentifikasi celah keamanan yang mungkin terlewatkan dalam pengujian *white box.*

2.3 Tinjauan Studi

Bagian ini berisi rangkuman dari penelitian yang telah dilakukan sebelumnya yang dipilih berdasarkan topik dan konten yang relevan dengan penelitian ini, antara lain:

1. Pada jurnal berjudul ***"Rancang Bangun Aplikasi Monitoring Mahasiswa Penerima Bantuan Beasiswa Bidikmisi pada Universitas Teknokrat Indonesia Berbasis Web"*** ditulis oleh Indah Septia Wulandari dan dipublikasikan oleh *Computer Science* pada tahun 2019. Jurnal ini membahas tentang rancang bangun aplikasi *monitoring* mahasiswa penerima bantuan beasiswa Bidikmisi pada Universitas Teknokrat Indonesia berbasis web. Aplikasi ini memudahkan operator Bidikmisi untuk melakukan monitoring mahasiswa Bidikmisi pada Universitas Teknokrat Indonesia dengan maksimal. Aplikasi ini mencakup monitoring terhadap aspek persyaratan seperti IPK,

masa studi, semester, dan keaktifan dalam mengikuti kegiatan kampus seperti organisasi atau UKM. Sistem ini dikembangkan menggunakan metode *Waterfall*.

2. Pada jurnal berjudul ***"Rancang bangun sistem informasi monitoring dan evaluasi hafalan al-Qur'an Program Beasiswa Santri Berprestasi (PBSB) berbasis web pada Universitas Islam Negeri (UIN) Maulana Malik Ibrahim Malang dengan metode Extreme Programming (XP)"*** ditulis oleh Retno Wulandari dan dipublikasikan oleh *Computer Science* pada tahun 2018. Jurnal ini membahas tentang rancang bangun sistem informasi *monitoring* dan evaluasi hafalan al-Qur'an Program Beasiswa Santri Berprestasi (PBSB) berbasis web pada Universitas Islam Negeri (UIN) Maulana Malik Ibrahim Malang. Sistem ini dirancang dengan menggunakan metode *Extreme Programming* (XP). Sistem ini bertujuan untuk memudahkan para pihak yang terkait dalam sistem dalam akses kegiatan *monitoring* tanpa melalui tatap muka, serta dalam hal pengelolaan data hafalan Al-Qur'an yang selama ini masih dilakukan secara manual.
3. Pada jurnal berjudul ***"Rancang Bangun Aplikasi Monitoring Realisasi Program Kerja pada Dinas Olahraga dan Kepemudaan Provinsi Jawa Timur"*** ditulis oleh Farhannur Priambodo dan dipublikasikan pada tahun 2021. Jurnal ini membahas tentang rancang bangun aplikasi *monitoring* realisasi program kerja pada Dinas Olahraga dan Kepemudaan Provinsi Jawa Timur. Aplikasi ini dirancang untuk memudahkan pengelola dalam melakukan *monitoring* terhadap realisasi program kerja yang telah ditetapkan. Aplikasi ini mencakup *monitoring* terhadap aspek-aspek seperti target, capaian, dan kendala yang dihadapi dalam pelaksanaan program kerja. Sistem ini dikembangkan dengan menggunakan bahasa pemrograman PHP dan *database* MySQL.
4. Pada jurnal berjudul ***"Rancang Bangun Aplikasi Monitoring Alat Kalibrasi Berbasis Website pada PT. Sucofindo Cabang Utama Surabaya"*** ditulis oleh Rizaldi Nugraha dan dipublikasikan pada tahun 2020. Jurnal ini membahas tentang rancang bangun aplikasi *monitoring* alat kalibrasi berbasis website pada PT. Sucofindo Cabang Utama Surabaya. Aplikasi ini dirancang untuk memudahkan *monitoring* terhadap alat kalibrasi yang dimiliki oleh PT. Sucofindo Cabang Utama Surabaya. Aplikasi ini mencakup *monitoring* terhadap aspek-aspek seperti jadwal kalibrasi, hasil kalibrasi, dan status alat kalibrasi. Sistem ini dikembangkan menggunakan metode *Waterfall*.