

**PENGEMBANGAN *BACKEND APPLICATION PROGRAMMING INTERFACE*
(*API*) PADA APLIKASI *MOBILE MEDIA PEMBELAJARAN* BERBASIS
GAMIFIKASI DENGAN METODE *SCRUM* UNTUK MAHASISWA
KEDOKTERAN**

(Skripsi)

Oleh

CHARLES GUNAWAN

NPM 2015061044



**FAKULTAS TEKNIK
UNIVERSITAS LAMPUNG
BANDAR LAMPUNG**

2024

**PENGEMBANGAN *BACKEND APPLICATION PROGRAMMING INTERFACE*
(API) PADA APLIKASI *MOBILE MEDIA PEMBELAJARAN* BERBASIS
GAMIFIKASI DENGAN METODE *SCRUM* UNTUK MAHASISWA
KEDOKTERAN**

Oleh

CHARLES GUNAWAN

Skripsi

**Sebagai Salah Satu Syarat untuk Mencapai Gelar
SARJANA TEKNIK**

Pada

**Jurusan Teknik Elektro
Fakultas Teknik Universitas Lampung**



**FAKULTAS TEKNIK
UNIVERSITAS LAMPUNG
BANDAR LAMPUNG**

2024

ABSTRAK

PENGEMBANGAN *BACKEND APPLICATION PROGRAMMING INTERFACE* (API) PADA APLIKASI *MOBILE MEDIA PEMBELAJARAN* BERBASIS GAMIFIKASI DENGAN METODE *SCRUM* UNTUK MAHASISWA KEDOKTERAN

Oleh

CHARLES GUNAWAN

Mahasiswa kedokteran sering mengalami kesulitan saat transisi dari pendidikan akademik ke pendidikan profesi. Penelitian menunjukkan bahwa mereka merasakan kelelahan emosional dan kekhawatiran terkait proses pendidikan selama fase klinik. Hal ini disebabkan oleh berbagai faktor, seperti lingkungan baru, beban kerja yang lebih berat, kesulitan mengatur waktu, dan kendala dalam menerapkan teori yang telah dipelajari di fase akademik. Solusi yang diusulkan dalam penelitian ini adalah pengembangan QuestMD yaitu aplikasi *mobile media pembelajaran* berbasis gamifikasi. Pengembangan aplikasi ini membutuhkan sistem *backend* untuk mendukung fungsionalitas, skalabilitas, penyimpanan file, otentikasi, dan pengelolaan pengguna. Sistem *backend* dibangun menggunakan *framework* ExpressJS dengan metode siklus pengembangan *software Scrum*. Pengujian fungsionalitas sistem dilakukan menggunakan metode *blackbox testing* dengan bantuan *tools* Postman. Hasil dari pengembangan sistem *backend Application Programming Interface (API)* ini adalah *server* dan *API* yang dapat menangani *request* dari *frontend*, mengelola data, mampu berinteraksi dengan *database* untuk menambah, membaca, mengubah, maupun menghapus data sesuai permintaan, memberikan *response* kepada *frontend*, dan menyimpan *file* berupa gambar pada *server*. Sistem *backend API* berhasil dikembangkan dalam 17 iterasi *sprint*, yang terdiri dari 10 iterasi pengembangan produk awal, 2 iterasi pengembangan fitur tambahan, 3 iterasi pengembangan fitur admin, dan 2 iterasi penambahan data konten. Secara total, 72 *endpoints* telah berhasil dibuat dan persentase pengujian fungsional *endpoint* pada setiap iterasi mencapai 100%.

Kata kunci: mahasiswa kedokteran, gamifikasi, *backend API*, *expressjs*, *scrum*

ABSTRACT

DEVELOPMENT OF BACKEND APPLICATION PROGRAMMING INTERFACE (API) IN GAMIFICATION-BASED MOBILE LEARNING MEDIA APPLICATION WITH SCRUM METHOD FOR MEDICAL STUDENTS

By

CHARLES GUNAWAN

Medical students often experience difficulties during the transition from academic to professional education. Research shows that they experience emotional exhaustion and concerns related to the educational process during the clinical phase. This is caused by various factors such as a new environment, heavier workloads, difficulty managing time, and challenges in applying the theories learned in the academic phase. The solution proposed in this research is the development of QuestMD, a gamified mobile learning media application. The development of this application requires a backend system to support functionality, scalability, file storage, authentication, and user management. The backend system is built using the ExpressJS framework with the Scrum software development cycle method. The system's functionality testing is done using the blackbox testing method with the help of the Postman tool. The result of developing this Application Programming Interface (API) backend system is a server and API that can handle frontend requests, manage data, interact with the database to add, read, modify, or delete data as requested, provide responses to the frontend, and store image files on the server. The backend API system was successfully developed in 17 sprint iterations, consisting of 10 iterations of initial product development, 2 iterations of additional feature development, 3 iterations of admin feature development, and 2 iterations of adding content data. In total, 72 endpoints were successfully created and the percentage of functional endpoint testing in each iteration reached 100%.

Keywords: *medical student, gamification, backend API, expressjs, scrum*

Judul Skripsi : PENGEMBANGAN BACKEND
APPLICATION PROGRAMMING INTERFACE
(API) PADA APLIKASI MOBILE MEDIA
PEMBELAJARAN BERBASIS GAMIFIKASI
DENGAN METODE SCRUM UNTUK
MAHASISWA KEDOKTERAN

Nama Mahasiswa : Charles Gunawan

Nomor Pokok Mahasiswa : 2015061044

Program Studi : Teknik Informatika

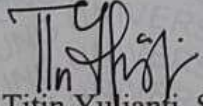
Jurusan : Teknik Elektro

Fakultas : Teknik



1. Komisi Pembimbing


Ir. Ing. Hery Dian Septama, S.T., IPM
NIP 198509152008121001

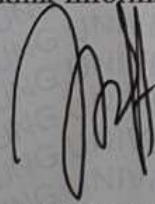

Ir. Titin Yulianti, S.T., M.Eng.
NIP 198807092019032015

2. Mengetahui

Ketua Jurusan
Teknik Elektro


Herlinawati, S.T., M.T.
NIP 197103141999032001

Ketua Program Studi
Teknik Informatika


Yessi Mulyani, S.T., M.T.
NIP 197312262000122001

MENGESAHKAN

1. Tim Penguji

Ketua : Ir. Ing. Hery Dian Septama, S.T., IPM.

Sekretaris : Ir. Titin Yulianti, S.T., M.Eng.

Penguji : Dr. Eng. Ir. Mardiana, S.T., M.T., IPM

2. Dekan Fakultas Teknik



Dr. Eng. Ir. Helmy Fitriawan, S.T., M.Sc.)

NIP 197509282001121002

Tanggal Lulus Ujian Skripsi : 25 Juli 2024

SURAT PERNYATAAN

Saya yang bertanda tangan di bawah ini, menyatakan bahwa skripsi saya yang berjudul “Pengembangan *Backend API* pada Aplikasi *Mobile Media Pembelajaran Berbasis Gamifikasi* dengan Metode *SCRUM* untuk Mahasiswa Kedokteran” merupakan hasil kerja saya sendiri. Apabila di kemudian hari terbukti bahwa pernyataan saya tidak benar, maka saya bersedia dikenai sanksi sesuai dengan hukum yang berlaku.

Bandar Lampung, 12 Agustus 2024

Penulis,



Charles Gunawan

2015061044

RIWAYAT HIDUP



Penulis dilahirkan di Jakarta, pada tanggal 14 Desember 2001, sebagai anak kedua dari lima bersaudara, dari Bapak Chandra Gunawan dan Ibu Herlina. Penulis menyelesaikan pendidikan di SD Xaverius Kotabumi pada tahun 2014, SMP Xaverius Kotabumi pada tahun 2017, SMA Negeri 2 Kotabumi pada tahun 2020. Pada tahun 2020 penulis terdaftar sebagai mahasiswa Program Studi Teknik Informatika, Jurusan Teknik Elektro, Fakultas Teknik Universitas Lampung melalui jalur SBMPTN (Seleksi Bersama Masuk Perguruan Tinggi Negeri). Selama menjadi mahasiswa, penulis melakukan beberapa kegiatan, antara lain:

1. Mengikuti Program Kredensial Mikro Mahasiswa Indonesia (KMMI) dengan mengambil kursus Teknologi Multimedia pada tahun 2021.
2. Mengikuti Program Beasiswa Pendidikan dari Feedloop untuk belajar mengenai Teknologi *Low Code Development Platform* pada tahun 2022.
3. Mengikuti Studi Independen Bersertifikat Kampus Merdeka dari Kementerian Pendidikan dan Budaya di Binar Academy mengambil kursus Backend Javascript pada tahun 2022.
4. Menjadi Asisten Laboratorium Komputer Jurusan Teknik Elektro Universitas Lampung pada tahun 2022 hingga 2024.
5. Menjadi Ketua Asisten Laboratorium Komputer Jurusan Teknik Elektro Universitas Lampung pada tahun 2023 hingga 2024.
6. Melaksanakan Kuliah kerja Nyata di Desa Bumi Mulya, Kecamatan Pakuan Ratu, Kabupaten Way Kanan, Lampung pada tahun 2023.
7. Mengikuti kegiatan magang di PT. Queen Network Nusantara sebagai Backend Developer pada tahun 2023.

MOTTO

“Terlalu sombong jika manusia memaknai takdir dari sudut pandang Tuhan hanya untuk membenarkan perbuatannya.”

(Charles Gunawan)

"If you're not a good shot today, don't worry. There are other ways to be useful."

(Sova – Valorant)

“But you'll never know unless you walk in my shoes.”

(Blackpink)

PERSEMBAHAN

Bismillahirrahmanirrahim

Dengan ini kupersembahkan karya ini kepada :

Kedua orang tua yang selalu memberikan dukungan, nasihat, dan doa;

Saudara-saudara ku yang selalu menjadi penyemangat;

Keluarga dan sahabat yang senantiasa memberikan dukungan dan mewarnai

hidup;

Orang-orang yang membantu ku;

dan

Tempat ku menuntut ilmu

Universitas Lampung.

Terima kasih telah menjadi bagian dan mewarnai cerita hidup ku.

SANWACANA

Alhamdulillahirabbil'alamin. Puji dan syukur penulis hanturkan ke hadirat Tuhan Yang Maha Esa, karena atas berkat dan rahmat-Nya, penulis dapat menyelesaikan skripsi yang berjudul “Pengembangan *Backend API* pada Aplikasi *Mobile Media Pembelajaran Berbasis Gamifikasi* dengan Metode *SCRUM* untuk Mahasiswa Kedokteran” sebagai salah satu syarat untuk memperoleh gelar sarjana di Fakultas Teknik Universitas Lampung. Dalam proses penyelesaian skripsi ini tidak lepas dari bantuan, bimbingan, kerjasama, dan doa dari berbagai pihak, baik secara langsung maupun tidak langsung.

Dalam kesempatan ini penulis mengucapkan banyak terima kasih kepada:

1. Bapak Dr. Eng. Helmy Fitriawan, S.T., M.Sc., selaku Dekan Fakultas Teknik Universitas Lampung;
2. Ibu Herlinawati, S.T., M.T. selaku Ketua Jurusan Teknik Elektro Universitas Lampung;
3. Ibu Yessi Mulyani, S.T., M.T. selaku Ketua Program Studi Teknik Informatika Universitas Lampung;
4. Bapak Ir. Ing. Hery Dian Septama, S.T., IPM. selaku Pembimbing Utama yang bersedia memberikan waktu serta pikiran dalam membimbing penulis selama proses penyelesaian penelitian dan skripsi;
5. Ibu Ir. Titin Yulianti, S.T., M.Eng. selaku Pembimbing Pendamping yang selalu memberikan saran yang membangun dan selalu sabar dalam membimbing penulis selama proses penyelesaian skripsi;
6. Ibu Dr. Eng. Ir. Mardiana, S.T., M.T., IPM selaku Penguji yang telah memberikan kritik, saran, serta arahan agar skripsi ini menjadi lebih baik;
7. Bapak Ir. Gigih Forda Nama, S.T., M.T.I. selaku Pembimbing Akademik yang selalu memberikan dukungan serta bimbingan;

8. Seluruh Dosen dan Staff Jurusan Teknik Informatika Unila yang telah memberikan ilmu dan teladan.
9. Mba Rika selaku Admin Program Studi Teknik Informatika yang telah banyak membantu penulis dalam segala urusan administrasi selama perkuliahan.
10. Kedua orang tua, Chandra Gunawan dan Herlina, yang telah berusaha sebaik mungkin untuk membesarkan dan mendidik penulis dengan penuh kesabaran dan kasih sayang.
11. Popo Mifa, yang telah menjadi nenek yang selalu mendukung segala cita-cita penulis hingga saat ini dan memenuhi kebutuhan asupan gizi penulis dengan masakannya yang enak.
12. Koko Charlie Alhady dan adik-adik yang penulis sayangi, Charlos Algun, Charl Julius Gunawan, dan Ridwansyah Alhady, yang selalu menjadi penyemangat penulis dalam menjalani hidup.
13. Kepada Khiukung dan Almarhumah Khiupo, Lunto Hasan dan Vincentia Lenny, yang telah membantu penulis dan keluarga dalam segala macam hal dan aspek kehidupan.
14. Kepada Muhammad Wafa Al Ausath dan Muhamad Satrio sebagai tim peneliti yang membantu dalam pengembangan aplikasi.

Penulis sadar skripsi ini masih jauh dari kata sempurna dan masih banyak kesalahan, sehingga kritik dan saran sangat terbuka bagi penulis. Penulis berharap skripsi ini dapat menjadi referensi untuk penelitian lain dan bermanfaat bagi siapapun yang membacanya. Sekali lagi penulis ucapkan terima kasih atas seluruh bantuan dan maaf atas seluruh kesalahan. Semoga Tuhan selalu memberikan keberkahan dalam kehidupan kita semua.

Bandar Lampung, 12 Agustus 2024

Penulis,

Charles Gunawan

DAFTAR ISI

Judul	Halaman
ABSTRAK	iii
DAFTAR ISI.....	xiii
DAFTAR GAMBAR.....	xviii
DAFTAR TABEL	xxi
I. PENDAHULUAN.....	1
1.1. Latar Belakang	1
1.2. Perumusan Masalah.....	4
1.3. Tujuan Penelitian.....	4
1.4. Manfaat Penelitian.....	4
1.5. Batasan Masalah.....	4
1.6. Sistematika Penulisan Skripsi / Tugas Akhir	5
II. TINJAUAN PUSTAKA	7
2.1. Pendidikan Profesi Kedokteran	7
2.2. Gamifikasi	8
2.3. <i>Server Side Application (Backend)</i>	9
2.4. API.....	9
2.5. <i>Metode SDLC Scrum</i>	11
2.6. <i>ExpressJS</i>	12
2.7. Format JSON	13
2.8. Postgresql	14
2.9. <i>Blackbox Testing</i>	14
2.10. <i>Postman</i>	15

2.11. Penelitian Terkait	15
III. METODOLOGI PENELITIAN	19
3.1. Waktu dan Tempat	19
3.2. Tim Penelitian	20
3.3. Alat Penelitian	21
3.4. Metode Penelitian.....	22
3.4.1. <i>Product backlog</i>	23
3.4.2. <i>Sprint Planning</i>	23
3.4.3. <i>Sprint (Execution)</i>	29
3.4.4. <i>Sprint Review</i>	33
3.4.5. <i>Sprint Retrospective</i>	34
IV. HASIL DAN PEMBAHASAN	35
4.1. Iterasi 1	35
4.1.1. <i>Product backlog</i>	35
4.1.2. <i>Sprint Planning</i>	36
4.1.3. <i>Sprint (Coding dan Testing)</i>	36
4.1.4. <i>Sprint Review</i>	47
4.1.5. <i>Sprint Retrospective</i>	47
4.2. Iterasi 2	47
4.2.1. <i>Sprint Planning</i>	47
4.2.2. <i>Sprint (Coding dan Testing)</i>	47
4.2.3. <i>Sprint Review</i>	59
4.2.4. <i>Sprint Retrospective</i>	60
4.3. Iterasi 3	60
4.3.1. <i>Sprint Planning</i>	60
4.3.2. <i>Sprint (Coding dan Testing)</i>	60
4.3.3. <i>Sprint Review</i>	69
4.3.4. <i>Sprint Retrospective</i>	69
4.4. Iterasi 4	70
4.4.1. <i>Sprint Planning</i>	70
4.4.2. <i>Sprint (Coding dan Testing)</i>	70
4.4.3. <i>Sprint Review</i>	73

4.4.4.	<i>Sprint Retrospective</i>	74
4.5.	Iterasi 5	74
4.5.1.	<i>Sprint Planning</i>	74
4.5.2.	<i>Sprint (Coding dan Testing)</i>	74
4.5.3.	<i>Sprint Review</i>	77
4.5.4.	<i>Sprint Retrospective</i>	77
4.6.	Iterasi 6	78
4.6.1.	<i>Sprint Planning</i>	78
4.6.2.	<i>Sprint (Coding dan Testing)</i>	78
4.6.3.	<i>Sprint Review</i>	82
4.6.4.	<i>Sprint Retrospective</i>	82
4.7.	Iterasi 7	82
4.7.1.	<i>Sprint Planning</i>	82
4.7.2.	<i>Sprint (Coding, Deploy, dan Testing)</i>	83
4.7.3.	<i>Sprint Review</i>	86
4.7.4.	<i>Sprint Retrospective</i>	86
4.8.	Iterasi 8	87
4.8.1.	<i>Sprint Planning</i>	87
4.8.2.	<i>Sprint (Coding dan Testing)</i>	87
4.8.3.	<i>Sprint Review</i>	92
4.8.4.	<i>Sprint Retrospective</i>	93
4.9.	Iterasi 9	93
4.9.1.	<i>Sprint Planning</i>	93
4.9.2.	<i>Sprint (Coding, Deploy, dan Testing)</i>	93
4.9.3.	<i>Sprint Review</i>	95
4.9.4.	<i>Sprint Retrospective</i>	95
4.10.	Iterasi 10	96
4.10.1.	<i>Sprint Planning</i>	96
4.10.2.	<i>Sprint (Coding dan Testing)</i>	96
4.10.3.	<i>Sprint Review</i>	101
4.10.4.	<i>Sprint Retrospective</i>	102
4.11.	Iterasi 11	102

4.11.1. <i>Sprint Planning</i>	102
4.11.2. <i>Sprint (Coding)</i>	102
4.11.3. <i>Sprint Review</i>	104
4.11.4. <i>Sprint Retrospective</i>	104
4.12. Pengujian	104
4.12.1. Metode Pengujian.....	104
4.12.2. Profil Responden Pengujian.....	104
4.12.3. <i>Feedback</i>	105
4.13. Iterasi 12	106
4.13.1. <i>Product backlog</i>	106
4.13.2. <i>Sprint Planning</i>	108
4.13.3. <i>Sprint (Coding dan Testing)</i>	108
4.13.4. <i>Sprint Review</i>	111
4.13.5. <i>Sprint Retrospective</i>	111
4.14. Iterasi 13	111
4.14.1. <i>Sprint Planning</i>	111
4.14.2. <i>Sprint (Coding dan Testing)</i>	112
4.14.3. <i>Sprint Review</i>	115
4.14.4. <i>Sprint Retrospective</i>	115
4.15. Iterasi 14	115
4.15.1. <i>Sprint Planning</i>	115
4.15.2. <i>Sprint (Coding dan Testing)</i>	116
4.15.3. <i>Sprint Review</i>	121
4.15.4. <i>Sprint Retrospective</i>	121
4.16. Iterasi 15	121
4.16.1. <i>Sprint Planning</i>	121
4.16.2. <i>Sprint (Coding dan Testing)</i>	122
4.16.3. <i>Sprint Review</i>	125
4.16.4. <i>Sprint Retrospective</i>	125
4.17. Iterasi 16	126
4.17.1. <i>Sprint Planning</i>	126
4.17.2. <i>Sprint (Coding dan Testing)</i>	126

4.17.3. <i>Sprint Review</i>	129
4.17.4. <i>Sprint Retrospective</i>	130
4.18. Iterasi 17	130
4.18.1. <i>Sprint Planning</i>	130
4.18.2. <i>Sprint</i>	130
4.18.3. <i>Sprint Review</i>	130
4.18.4. <i>Sprint Retrospective</i>	131
4.19. Pembahasan	131
V. KESIMPULAN DAN SARAN	138
5.1. Kesimpulan	138
5.2. Saran	138
DAFTAR PUSTAKA	140

DAFTAR GAMBAR

Gambar	Halaman
Gambar 1. Desain Arsitektur Sistem Website.....	10
Gambar 2. Penggunaan API dengan Metode HTTP GET.....	10
Gambar 3. Gambaran Metode <i>Scrum</i> [20]	11
Gambar 4. Contoh Data JSON	14
Gambar 5. Peran Tim Pengembang	20
Gambar 6. Tahapan Penelitian	22
Gambar 7. Usecase Diagram.....	28
Gambar 8. ERD QuestMD	29
Gambar 9. Tampilan Postman.....	33
Gambar 10. Tampilan Trello.....	34
Gambar 11. Hasil Menjalankan <code>npx express-generator</code>	36
Gambar 12. Struktur Direktori Awal	37
Gambar 13. Baris Kode untuk Menjalankan Aplikasi pada Port Tertentu	37
Gambar 14. Hasil Menjalankan command <code>npx sequelize init</code>	38
Gambar 15. Modifikasi File <code>index.js</code> pada Direktori Models	38
Gambar 16. Kode <code>config.js</code>	38
Gambar 17. Variabel dalam File <code>.env</code>	39
Gambar 18. Source Code Model User	40
Gambar 19. Source Code Register	41
Gambar 20. Source Code Login.....	42
Gambar 21. Source Code Router <code>user.js</code>	43
Gambar 22. Source Code File <code>index.js</code> pada Direktori Routes	43
Gambar 23. Penambahan Fungsi untuk Router pada <code>app.js</code>	44

Gambar 24. Scripts Menjalankan nodemon	44
Gambar 25. Hasil Menjalankan Perintah “npm run dev”	45
Gambar 26. Testing Fitur Register	45
Gambar 27. Testing Fitur Login	46
Gambar 28. Source Code Fitur Register Setelah Penambahan Token	48
Gambar 29. <i>Source Code Create User</i>	50
Gambar 30. Source Code Read All Users	51
Gambar 31. <i>Source Code Read User by id</i>	51
Gambar 32. Source Code Update User by id	52
Gambar 33. Source Code Delete User by id	52
Gambar 34. Source Code Middleware Auth isLogin	53
Gambar 35. Source Code Middleware Auth isAdmin	54
Gambar 36. Penggunaan Fungsi Middleware	55
Gambar 37. Output Register Setelah Diperbarui	56
Gambar 38. Penggunaan Token dalam Postman	56
Gambar 39. Authorization Error Output	57
Gambar 40. Output Fungsi Read All Users	57
Gambar 41. Source Code Model Avatar	61
Gambar 42. Relasi Tabel User, My_Avatar, dan Avatar	61
Gambar 43. Fungsi Asosiasi pada Model My_avatar	62
Gambar 44. Source Code Create Avatar	62
Gambar 45. Source Code Buy Avatar	64
Gambar 46. Source Code Get All Avatar with Mine	65
Gambar 47. Source Code Media Handling	66
Gambar 48. Penggunaan Media Handling di Router	66
Gambar 49. Source Code Model User Setelah Penambahan OTP	70
Gambar 50. Source Code Send OTP	71
Gambar 51. Source Code Change Password	72
Gambar 52. Source Code Fitur Create New Leaderboard	83
Gambar 53. Hasil Testing Fungsi Create New Leaderboard	85
Gambar 54. Penambahan Logika Achievement “Si Kolektor” pada Fungsi Buy Avatar	88

Gambar 55. Source Code Model User Setelah Penambahan loginCount	96
Gambar 56. Source Code Logger	97
Gambar 57. Fungsi Pendefinisian Error	98
Gambar 58. Contoh Penggunaan Fungsi Pendefinisi Error	99
Gambar 59. Fungsi Error Handler.....	100
Gambar 60. Hasil Pencatatan Error.....	101
Gambar 61. Source Code Seeder Stase	103
Gambar 62. Hasil Menjalankan Perintah npx sequelize db:seed:all	103

DAFTAR TABEL

Tabel	Halaman
Tabel 1. Penelitian Terkait	17
Tabel 2. Waktu Penelitian	19
Tabel 3. Alat Penelitian.....	21
Tabel 4. Kebutuhan Fungsional	24
Tabel 5. Tabel Testing.....	33
Tabel 6. Tabel Sprint Backlog untuk Iterasi Pertama.	36
Tabel 7. Tabel Endpoints Fitur Register dan Login	44
Tabel 8. Tabel Testing Fitur Iterasi 1	46
Tabel 9. Tabel Sprint Backlog untuk Iterasi Kedua	47
Tabel 10. Tabel Endpoints Fitur Iterasi Kedua	55
Tabel 11. Tabel Testing Fitur Iterasi 2	58
Tabel 12. Tabel Sprint Backlog untuk Iterasi Ketiga.....	60
Tabel 13. Tabel Endpoints Fitur Iterasi Ketiga	67
Tabel 14. Tabel Hasil Testing Fitur Iterasi 3	67
Tabel 15. Tabel Sprint Backlog untuk Iterasi Keempat	70
Tabel 16. Tabel Endpoints Fitur Iterasi Keempat	73
Tabel 17. Tabel Hasil Testing Fitur Iterasi 4	73
Tabel 18. Tabel Sprint Backlog untuk Iterasi Kelima.....	74
Tabel 19. Tabel Endpoints Fitur Iterasi Kelima.....	75
Tabel 20. Tabel Hasil Testing Fitur Iterasi 5	76
Tabel 21. Tabel Sprint Backlog untuk Iterasi Keenam	78
Tabel 22. Tabel Endpoints Fitur Iterasi Keenam	80
Tabel 23. Tabel Hasil Testing Fitur Iterasi 6	80

Tabel 24. Tabel Sprint Backlog untuk Iterasi Ketujuh	82
Tabel 25. Tabel Endpoints Fitur Iterasi Ketujuh.....	84
Tabel 26. Tabel Hasil Testing Fitur Iterasi 7	84
Tabel 27. Daftar Masalah Setelah Melakukan Deploy	86
Tabel 28. Tabel Sprint Backlog untuk Iterasi Kedelapan	87
Tabel 29. Daftar Achievement	89
Tabel 30. Tabel Endpoints Fitur Iterasi Kedelapan	90
Tabel 31. Tabel Hasil Testing Fitur Iterasi 8	90
Tabel 32. Tabel Sprint Backlog untuk Iterasi Kesembilan	93
Tabel 33. Tabel Endpoints Fitur Iterasi Kesembilan	94
Tabel 34. Tabel Hasil Testing Fitur Iterasi 9	94
Tabel 35. Tabel Sprint Backlog untuk Iterasi Kesepuluh	96
Tabel 36. Tabel Endpoints Fitur Iterasi Kesepuluh	100
Tabel 37. Tabel Hasil Testing Fitur Iterasi 10	100
Tabel 38. Tabel Sprint Backlog untuk Iterasi Kesebelas	102
Tabel 39. Profil Responden.....	105
Tabel 40. Penambahan Kebutuhan Fungsional.....	107
Tabel 41. Tabel Sprint Backlog untuk Iterasi Kedua belas.....	108
Tabel 42. Tabel Endpoints Fitur Iterasi Kedua belas.....	109
Tabel 43. Tabel Hasil Testing Fitur Iterasi 12	110
Tabel 44. Tabel Sprint Backlog untuk Iterasi Ketiga belas	112
Tabel 45. Tabel Endpoints Fitur Iterasi Ketiga belas.....	113
Tabel 46. Tabel Hasil Testing Fitur Iterasi 13	113
Tabel 47. Tabel Sprint Backlog untuk Iterasi Keempat belas.....	115
Tabel 48. Tabel Endpoints Fitur Iterasi Keempat belas.....	117
Tabel 49. Tabel Hasil Testing Fitur Iterasi 14	118
Tabel 50. Tabel Sprint Backlog untuk Iterasi Kelima belas	121
Tabel 51. Tabel Endpoints Fitur Iterasi Kelima belas.....	123
Tabel 52. Tabel Hasil Testing Fitur Iterasi 15	123
Tabel 53. Tabel Sprint Backlog untuk Iterasi Keenam belas.....	126
Tabel 54. Tabel Endpoints Fitur Iterasi Keenam belas	127
Tabel 55. Tabel Hasil Testing Fitur Iterasi 16	128

Tabel 56. Tabel Sprint Backlog untuk Iterasi Ketujuh belas	130
Tabel 57. Rekapitulasi Setiap Iterasi.....	131

I. PENDAHULUAN

1.1. Latar Belakang

Dokter merupakan profesi yang mulia, yang berkepentingan dalam kesejahteraan umat manusia. Menjadi seorang dokter harus senantiasa melaksanakan perintah moral dan intelektual, serta melayani masyarakat yang sakit maupun sehat agar mendapatkan kesejahteraan [1]. Untuk menjadi seorang dokter, seseorang harus memiliki kompetensi dalam bidang kedokteran dan memiliki gelar dokter (dr). Seseorang yang ingin menjadi seorang dokter harus melewati seluruh tahap pendidikan kedokteran.

Menurut Undang-Undang Republik Indonesia nomor 20 tahun 2013 tentang pendidikan kedokteran, pendidikan kedokteran merupakan usaha sadar dan terencana dalam pendidikan formal yang terdiri atas pendidikan akademik dan pendidikan profesi pada jenjang pendidikan tinggi yang program studinya terakreditasi untuk menghasilkan lulusan yang memiliki kompetensi di bidang kedokteran atau kedokteran gigi. Pendidikan akademik merupakan pendidikan tinggi program sarjana dan/atau program pascasarjana kedokteran dan kedokteran gigi terutama pada penguasaan ilmu kedokteran dan ilmu kedokteran gigi. Sedangkan pendidikan profesi merupakan pendidikan kedokteran yang dilaksanakan dalam bentuk pembelajaran klinik dan pembelajaran komunitas yang menggunakan berbagai bentuk dan tingkat pelayanan kesehatan nyata yang memenuhi persyaratan sebagai tempat praktik kedokteran [2].

Dalam menempuh pendidikan kedokteran, banyak mahasiswa yang merasa cemas saat masa transisi dari pendidikan akademik ke pendidikan profesi. Dalam penelitian yang dilakukan di Universitas Andalas terdapat 80,4% mahasiswa klinik

baru dari 46 responden mencemaskan bagaimana proses pendidikan selama fase klinik [3]. Kemudian berdasarkan penelitian pada 164 mahasiswa tahun pertama klinik Fakultas Kedokteran Universitas Indonesia diperoleh setengah dari mahasiswa pada masa transisi mengalami kelelahan emosional dan *personal accomplishment* yang rendah [4]. Hal tersebut terjadi karena mahasiswa baru fase klinik dihadapkan dengan masalah baru seperti lingkungan baru, beban kerja yang lebih berat, kesulitan mengatur waktu, dan sulit dalam mengaplikasikan teori yang telah dipelajari pada fase akademik.

Berdasarkan permasalahan tersebut diperlukan suatu inovasi pemecahan masalah agar mahasiswa dapat beradaptasi lebih mudah dalam masa transisi dari pendidikan akademik ke pendidikan profesi. Maka dari itu, salah satu solusi yang dapat dilakukan adalah mengembangkan QuestMD, sebuah aplikasi *mobile* media pembelajaran berbasis gamifikasi. Berdasarkan hasil sebuah studi terkait efektivitas gamifikasi terhadap *engagement* pengguna, didapatkan hasil bahwa gamifikasi memiliki efek positif terhadap *engagement* pengguna yaitu pelajar [5] dan kesimpulan dari penelitian mengenai penggunaan gamifikasi dalam proses belajar menunjukkan bahwa menggunakan pendekatan gamifikasi dapat membuat proses belajar menjadi lebih menyenangkan, menarik, dan efektif [6].

Pengembangan aplikasi QuestMD dibagi menjadi 3 bagian pekerjaan berbeda yaitu *mobile android (frontend)*, *server & API (backend)*, dan *database (backend)* yang dibahas dalam 3 penelitian berbeda sesuai dengan bagian pekerjaannya. Bagian *frontend* berfokus pada pengembangan tampilan dan interaksi *user* berbasis *mobile* agar *user* mendapatkan pengalaman yang baik saat menggunakan aplikasi. Bagian *backend* terbagi menjadi 2 yaitu pengembangan *server* dan *API* dan pengembangan *database*. *Backend server* dan *API* berfokus pada pengembangan logika aplikasi, pemrosesan data, dan antarmuka virtual untuk interaksi dengan *frontend* dan *database*. Sedangkan *database* berfokus pada pemodelan dan pengembangan *database* sebagai media penyimpanan data pada aplikasi agar dapat digunakan secara optimal dalam sistem.

Terdapat beberapa alasan mengapa pengerjaan frontend dan *backend* dipisahkan, seperti untuk kepentingan skalabilitas, mempermudah untuk mengganti *framework* apabila ingin migrasi menggunakan *tech stack* lain, mempercepat pengerjaan karena bekerja secara independen antara *frontend* dan *backend*, mempermudah ketika perlu meng-*upgrade* kode tanpa mengganggu kode satu sama lain [7].

Penelitian ini fokus membahas lebih *detail* mengenai pengembangan dan pengujian fungsional *backend server* dan *API* yang memainkan peran penting dalam mendukung fungsionalitas, skalabilitas, penyimpanan file, otentikasi, dan pengelolaan pengguna pada sistem. Dalam pengembangan *server* dan *API* media pembelajaran ini menggunakan *framework* ExpressJS dengan mengimplementasikan metode siklus pengembangan *software scrum*.

Server dan *API* pada aplikasi media pembelajaran ini dikembangkan menggunakan *framework* ExpressJS karena *framework* ExpressJS dikenal sebagai *unopinionated framework* yang memberikan kenyamanan untuk mengembangkan aplikasi sesuai kebutuhannya dengan sedikit batasan tentang cara optimal penggunaan komponennya (fleksibel) [8]. Kemudian berdasarkan hasil penelitian yang menguji *API* antara *framework* Codeigniter dan ExpressJS dengan 100-1000 *user request*, menunjukkan bahwa *response time API* dari ExpressJS lebih cepat sehingga cocok untuk diimplementasikan pada sistem yang diakses oleh banyak *user* [9]. *Output* yang dihasilkan dari *API* berupa *response* dari *server* berformat JSON. Metode siklus pengembangan *software* yang digunakan dalam pengembangan aplikasi ini adalah *scrum*. *Scrum* merupakan sebuah kerangka kerja yang adaptif dan dapat menangani masalah-masalah yang dinamis (cenderung berubah seiring waktu) [10]. Dalam implementasinya digunakan alat bantu berupa aplikasi berbasis web bernama Trello untuk mendokumentasikan proses pada setiap tahapan *scrum*.

Pengujian fungsional sistem dilakukan untuk memastikan sistem berjalan sesuai kebutuhan baik dari *input* maupun *output*. Pengujian pada sistem ini menggunakan metode *blackbox testing*. Metode ini dipilih karena memiliki fokus utama pada pengujian fungsionalitas perangkat lunak tanpa melibatkan desain dan kode program secara spesifik, serta penggunaannya yang mudah karena hanya

memerlukan batas atas dan batas bawah dari nilai yang diharapkan [11]. Alat bantu yang digunakan dalam pengujian adalah software *Postman*, yang memudahkan untuk mengirim permintaan HTTP ke *web server* dan menerima *response* dari *web server*.

1.2. Perumusan Masalah

Berdasarkan latar belakang, kajian masalah yang mendasari penelitian ini adalah:

1. Bagaimana cara mengembangkan sistem *backend* menggunakan *framework* ExpressJS untuk menghasilkan *API* yang dapat digunakan pada aplikasi pembelajaran berbasis gamifikasi menggunakan metode *scrum*?
2. Bagaimana melakukan uji coba terhadap *API* yang dikembangkan untuk memastikan fungsionalitasnya berjalan sesuai kebutuhan sistem dan *frontend*?

1.3. Tujuan Penelitian

Tujuan penelitian ini adalah:

1. Melakukan pengembangan *backend API* untuk aplikasi pembelajaran berbasis gamifikasi dengan *framework* ExpressJS menggunakan metode *scrum*.
2. Melakukan pengujian fungsionalitas *API* menggunakan metode *blackbox testing*.

1.4. Manfaat Penelitian

Manfaat dari hasil penelitian ini adalah:

1. Tersedia aplikasi QuestMD, media pembelajaran berbasis gamifikasi bagi mahasiswa kedokteran.
2. Pengalaman praktis dalam mengembangkan sistem bagi peneliti.
3. Sebagai referensi dan inspirasi bagi penelitian selanjutnya.

1.5. Batasan Masalah

Dalam penelitian ini, pembatasan masalah meliputi hal-hal sebagai berikut:

1. *Output* dari *API* yang dihasilkan berupa *response* dari *server* berformat JSON.

2. Penelitian ini tidak membahas mengenai tampilan sistem dan pengembangan sistem bagian *frontend* maupun *database*.

1.6. Sistematika Penulisan Skripsi / Tugas Akhir

Dalam pembahasan laporan penelitian ini, sistematika penulisan yang digunakan adalah sebagai berikut:

BAB I: PENDAHULUAN

Pada bab satu menjelaskan tentang latar belakang mengenai permasalahan yang dihadapi mahasiswa kedokteran hingga perlu dikembangkannya *backend API*, merinci fokus penelitian dalam rumusan masalah yaitu pengembangan *API* yang berfungsi dengan baik dan pengujiannya menggunakan *blackbox testing*, mendefinisikan tujuan penelitian yaitu mengembangkan dan menguji *API*, menetapkan ruang lingkup penelitian dalam batasan masalah agar fokus pada pembahasan yang spesifik sehingga hasil penelitian lebih jelas dan terarah, dan sistematika penulisan laporan.

BAB II: TINJAUAN PUSTAKA

Pada bab dua membahas tentang dasar-dasar teori dari penelitian. Pada penelitian ini menggunakan teori dasar mengenai Pendidikan Profesi Kedokteran, Gamifikasi, *Server Side Application (Backend)*, *Application Programming Interface (API)*, Format JSON, Metode *Scrum*, *ExpressJS*, *Postgresql*, *Blackbox Testing*, dan *Postman*. Pada bab ini juga membahas penelitian terkait yang menjadi referensi dalam pelaksanaan penelitian ini.

BAB III: METODE PENELITIAN

Pada bab tiga menjelaskan waktu penelitian, tim proyek dalam penelitian, tempat pelaksanaan penelitian, alat yang digunakan dalam penelitian, metode penelitian berupa *Scrum* dan tahapannya, serta *testing* yang dilakukan. Tahapan yang ada pada metode *Scrum* adalah *Product*

backlog, Sprint Planning, Sprint, Daily Scrum, Sprint Review, dan Sprint Retrospective.

BAB IV: HASIL DAN PEMBAHASAN

Pada bab empat memaparkan hasil pengembangan *backend* berupa *API* dan pengujian fungsionalitas *API* dengan metode *blackbox testing* menggunakan *postman*.

BAB V: KESIMPULAN DAN SARAN

Pada bab lima emuat tentang kesimpulan dan saran dari hasil penelitian untuk pengembangan penelitian selanjutnya.

DAFTAR PUSTAKA

Pada bagian daftar pustaka memuat sitasi jurnal, buku, *website*, atau tulisan yang menjadi referensi pada pengerjaan proyek dan laporan.

LAMPIRAN

Pada bagian lampiran memuat gambar atau dokumen pendukung penelitian .

II. TINJAUAN PUSTAKA

2.1. Pendidikan Profesi Kedokteran

Menurut Undang-Undang Republik Indonesia nomor 20 tahun 2013 tentang pendidikan kedokteran, pendidikan kedokteran merupakan usaha sadar dan terencana dalam pendidikan formal yang terdiri atas pendidikan akademik dan pendidikan profesi pada jenjang pendidikan tinggi yang program studinya terakreditasi untuk menghasilkan lulusan yang memiliki kompetensi di bidang kedokteran atau kedokteran gigi. Pendidikan akademik merupakan pendidikan tinggi program sarjana dan/atau program pascasarjana kedokteran dan kedokteran gigi terutama pada penguasaan ilmu kedokteran dan ilmu kedokteran gigi. Sedangkan pendidikan profesi merupakan pendidikan kedokteran yang dilaksanakan dalam bentuk pembelajaran klinik dan pembelajaran komunitas yang menggunakan berbagai bentuk dan tingkat pelayanan kesehatan nyata yang memenuhi persyaratan sebagai tempat praktik kedokteran [2].

Pendidikan profesi merupakan tahapan pendidikan yang harus ditempuh oleh dokter muda setelah menyelesaikan pendidikan akademik yang ditandai dengan memperoleh gelar Sarjana Kedokteran (S.Ked). Pendidikan profesi ditujukan pada lulusan pendidikan akademik agar dapat memperoleh kompetensi dan gelar seorang dokter (dr) [12].

Selain pendidikan profesi, mahasiswa kedokteran wajib lulus Uji Kompetensi Mahasiswa Program Profesi Dokter (UKMPPD) untuk mendapatkan gelar dokter. Uji kompetensi ini memastikan lulusan memenuhi standar nasional dan memperoleh sertifikat profesi. [13].

2.2. Gamifikasi

Gamifikasi merupakan strategi yang mengintegrasikan komponen-komponen permainan ke dalam kegiatan yang bukan permainan. Seiring berjalannya waktu, konsep gamifikasi telah diadopsi secara luas dalam berbagai sektor, termasuk bisnis dan pendidikan [14]. Pengertian gamifikasi pada penelitian Ariani adalah proses penggunaan elemen *game* di dalam kondisi *non-game* pada aspek pembelajaran bertujuan untuk meningkatkan perilaku belajar yang positif, dengan elemen dasar game yang meliputi lencana, papan, peringkat, poin, *level*, dan *avatar* [15].

Dalam implementasi gamifikasi digunakan beberapa elemen game, sebagai berikut:

1. *Point*

Point merupakan penggambaran keberhasilan dari pengerjaan tugas oleh pengguna yang digambarkan dalam numerik.

2. *Achievement/Badge*

Achievement/Badge merupakan penggambaran pencapaian dari penyelesaian suatu aktivitas tertentu yang dilakukan pengguna.

3. *Customization*

Customization merupakan kemampuan pengguna untuk bebas menyunting avatar pada profil dan informasi pengguna sesuai kebutuhan atau keinginan pengguna.

4. *Leaderboard*

Leaderboard merupakan papan yang berisi informasi mengenai posisi atau peringkat dari pengguna-pengguna di dalam aplikasi yang diurutkan berdasarkan poin.

5. *Level (Experience Level)*

Level merupakan penggambaran tingkatan pengguna berdasarkan poin *experience* yang didapatkan setelah menyelesaikan *challenge*.

6. *Challenges*

Challenges merupakan kegiatan untuk menguji tingkat pengetahuan atau kemahiran yang harus diselesaikan oleh pengguna.

7. *Reward/Feedback*

Reward/Feedback merupakan informasi yang didapatkan *user* sebagai respon karena telah melakukan suatu aksi, dan dapat menjadi acuan untuk meningkatkan motivasi pengguna dalam menyelesaikan aktivitas lainnya.

Aplikasi QuestMD merupakan aplikasi pembelajaran berbasis *mobile* untuk mahasiswa kedokteran yang dikembangkan dengan mengimplementasikan konsep gamifikasi. Dalam pengembangannya QuestMD terdiri dari pengembangan frontend, backend, dan database.

2.3. *Server Side Application (Backend)*

Backend atau *server side* adalah tempat di mana proses aplikasi berlangsung. Di sini, data diproses dan dikelola, termasuk penambahan, perubahan, atau penghapusan data. *Backend* mengurus aspek-aspek yang tidak terlihat atau berinteraksi langsung dengan pengguna, seperti *database* dan *server* [16]. Pada pengembangan aplikasi ini digunakan model arsitektur MVC (*Model, View, Controller*). Model mengandung fungsi kode yang membantu dalam pemrosesan yang berhubungan dengan *database*. *View* merupakan tampilan yang berinteraksi langsung dengan *user*. *Controller* merupakan perantara yang memproses permintaan HTTP yang menghubungkan antara *Model* dan *View* [17].

Dalam penerapannya seorang pengembang *backend* (*backend developer*) akan membuat *Model* dan *Controller*, sedangkan *View* akan dikembangkan oleh pengembang *frontend* (*frontend developer*) [18].

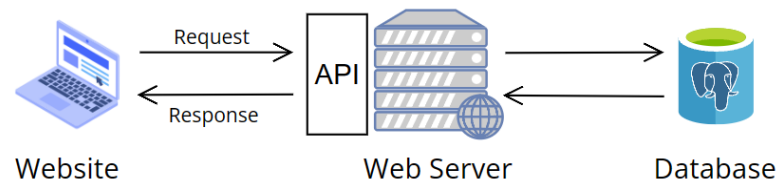
2.4. **API**

API merujuk pada seperangkat pedoman atau aturan yang memfasilitasi komunikasi antara dua atau lebih program. *API* dibangun pada *server* dengan tujuan memungkinkan klien untuk berinteraksi dengan *server* tersebut [19].

Seiring berkembangnya internet, pembicaraan mengenai *API* hampir selalu membahas tentang *API* web yang mengambil konsep antarmuka antara dua entitas dan diaplikasikan dalam hubungan klien – *server*. Dalam *API* web, di satu sisi

terdapat klien yang mengirimkan permintaan, sementara di sisi lain server memberikan respon dari permintaan klien. Struktur untuk *API request* terdiri dari *endpoints*, *actions*, *parameters*, *header*, dan *body*. Dari semua struktur *API request* tersebut terdapat dua elemen yang wajib dimiliki pada setiap penggunaan *API* yaitu *endpoints* dan *actions*. *Endpoints* berupa URL (*Uniform Resource Locator*) yang mengarahkan klien ke sumber daya atau layanan tertentu. Sedangkan *actions* berupa metode HTTP (*GET*, *POST*, *PUT*, *DELETE*) yang digunakan sebagai perintah dari *request* yang diminta oleh klien [20].

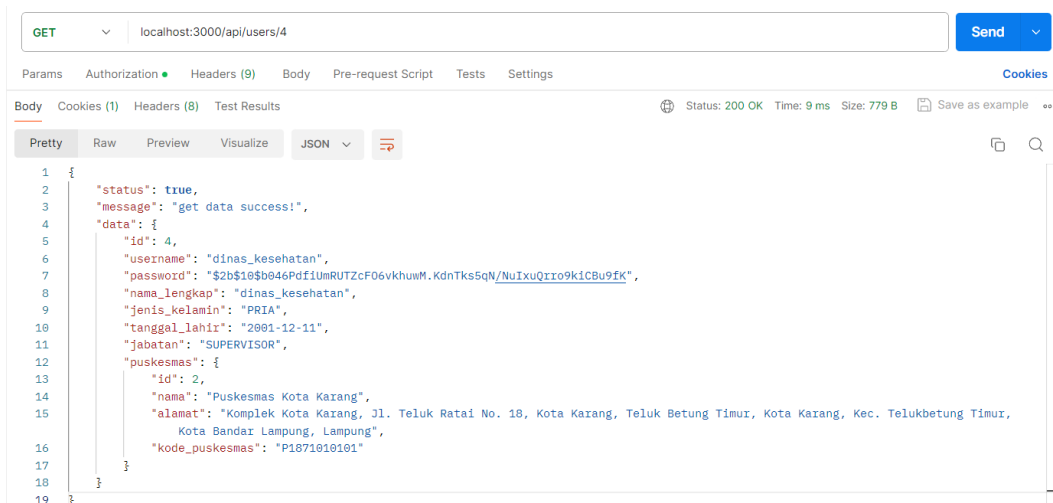
Berikut merupakan desain arsitektur dari sebuah sistem yang terdiri dari *frontend* (*website*) dan *backend* (*server* dan *database*) yang saling terhubung melalui *API*.



Gambar 1. Desain Arsitektur Sistem Website

Pada Gambar 1 penggunaan *API* adalah sebagai antarmuka untuk *client-side application* (*frontend*) agar dapat berkomunikasi dengan *server-side application* (*backend*).

Berikut merupakan contoh penggunaan *API* pada *website* untuk mendapatkan data *detail* pengguna menggunakan metode HTTP *GET* melalui *Postman*.

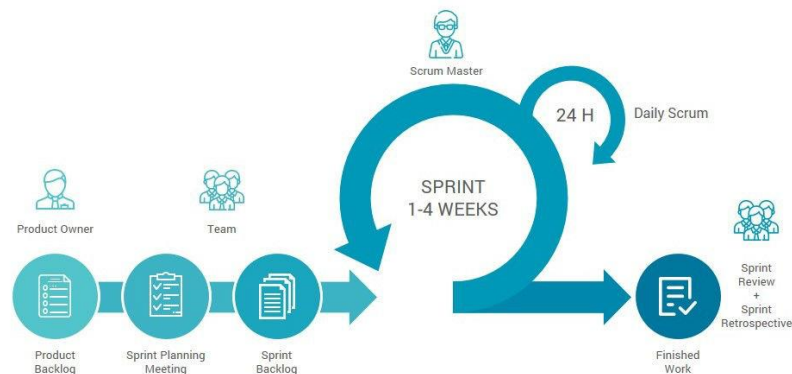


Gambar 2. Penggunaan API dengan Metode HTTP GET

Pada Gambar 2 dilakukan *request* terhadap *API* dengan URL `localhost:3000/api/users/4` menggunakan metode HTTP *GET* untuk membaca data dari sumber yang dituju yaitu *users* dengan *parameter* id yaitu empat. Sehingga didapatkan *response* berupa *status code* 200 yang menandakan *request* berhasil dan *response body* yang menampilkan data *user* dengan id bernilai empat.

2.5. Metode Software Development Life Cycle (SDLC) Scrum

Berdasarkan *website* panduan resmi *scrumguides.org* yang merupakan bagian dari *Scrum.org, the Home of Scrum*, yang dibentuk oleh Ken Schwaber seorang *co-creator* dari *Scrum*, menjelaskan bahwa *Scrum* adalah sebuah kerangka kerja yang dapat menangani masalah-masalah kompleks adaptif (cenderung berubah dan membutuhkan adaptasi seiring waktu), di waktu yang sama dapat secara produktif dan kreatif menghasilkan produk dengan nilai setinggi mungkin [10].



Gambar 3. Gambaran Metode Scrum [21]

Metode *Scrum* memiliki beberapa tahapan, yaitu:

1. *Product backlog*

Product backlog merupakan semua *list* kebutuhan dari aplikasi yang akan dikembangkan. Pada tahap ini pemilik dari aplikasi bertanggung jawab atas konten, ketersediaan, dan susunannya. *Product backlog* bersifat dinamis, dapat berubah atau bertambah seiring waktu untuk menggambarkan kebutuhan aplikasi agar lebih baik.

2. *Sprint Planning*

Sprint Planning merupakan tahapan untuk mengumpulkan tugas yang harus dikerjakan pada saat *sprint execution* berikutnya. Hasil dari *sprint planning* adalah *sprint backlog* yang merupakan bagian dari *product backlog* yang sudah dipilih oleh tim untuk dikerjakan pada *sprint* berikutnya dan sudah lebih rinci lagi.

3. *Sprint*

Sprint merupakan tahapan pengerjaan dari *sprint backlog* yang sudah dipilih. Pada tahap ini tim berfokus untuk mencapai tujuan dari *Sprint* yang telah ditetapkan.

4. *Daily Scrum*

Daily Scrum merupakan pertemuan singkat yang dilakukan setiap hari kerja untuk membagikan progress, identifikasi masalah, dan merencanakan tindakan selanjutnya.

5. *Sprint Review*

Sprint Review merupakan tahapan untuk memeriksa hasil dari *Sprint Execution* dan mendemonstrasikan hasil *sprint*. Pada tahap ini dilakukan juga diskusi mengenai rencana untuk *sprint* berikutnya.

6. *Sprint Retrospective*

Sprint Retrospective merupakan tahapan untuk mengevaluasi kinerja tim selama *sprint* yang sudah berlangsung.

2.6. *ExpressJS*

Express adalah sebuah *framework* yang berdiri di atas *web server* NodeJS untuk menyederhanakan API. *Framework* ini memudahkan untuk mengatur fungsi pada aplikasi dengan *middleware* dan *routing* [22]. *ExpressJS* adalah *framework* yang dikenal sebagai *unopinionated*, artinya *framework* ini memberikan kebebasan dan memiliki sedikit batasan tentang cara optimal penggunaan komponennya. Fleksibilitas yang dimilikinya membuat *ExpressJS* menjadi pilihan dalam penelitian ini [8].

Dalam penelitian lain hasil perbandingan pengujian *API* dengan 100 – 1000 *user request* antara *framework* Codeigniter dan ExpressJS menunjukkan bahwa *response time API* dari *framework* ExpressJS lebih cepat dengan nilai rata-rata 438,64 ms dari Codeigniter dengan nilai rata-rata 550,72 ms. Sehingga *framework* ExpressJS cocok untuk diimplementasikan pada sistem yang diakses oleh banyak *user* [9].

2.7. Format JSON

JSON (JavaScript Object Notation) merupakan format pertukaran data yang ringkas, mudah dipahami dan ditulis oleh manusia, serta mudah diproses oleh komputer. Format ini terinspirasi dari struktur objek JavaScript, berdasarkan standar ECMA-262 Edisi ke-3 yang dirilis pada Desember 1999. JSON menggunakan format teks yang bebas dari ketergantungan bahasa pemrograman, karena gaya bahasanya umum digunakan oleh programmer berbagai bahasa, termasuk C, C++, C#, Java, JavaScript, Perl, Python, dan lain sebagainya. Keunggulan ini menjadikan JSON sebagai pilihan ideal untuk pertukaran data lintas platform [23].

JSON dibangun dari dua struktur dasar:

1. Koleksi pasangan nama/nilai. Dalam berbagai bahasa pemrograman, ini dikenal sebagai objek, *record*, *struct*, *dictionary*, *hash table*, *keyed list*, atau *associative array*.
2. Daftar nilai yang berurutan. Di sebagian besar bahasa pemrograman, ini dikenal sebagai *array*, *vector*, *list*, atau *sequence*.

Kedua struktur ini merupakan struktur data universal. Hampir semua bahasa pemrograman modern mendukungnya dalam satu bentuk atau lainnya. Oleh karena itu, masuk akal jika format data yang dapat dipertukarkan dengan bahasa pemrograman juga didasarkan pada struktur ini. Berikut merupakan contoh data berformat JSON:

```

1  [{
2      "id": 1,
3      "nama": "Budi",
4      "kelas": "XII IPA 1"
5  },
6  {
7      "id": 2,
8      "nama": "Ani",
9      "kelas": "XII IPA 2"
10 }
11 ]

```

Gambar 4. Contoh Data JSON

2.8. Postgresql

PostgreSQL adalah sistem *database open-source* yang dikenal memiliki kemampuan yang sangat baik jika dibandingkan dengan *database* lainnya. Dikenal dengan fitur-fitur lengkapnya, PostgreSQL sangat cocok digunakan untuk aplikasi *database*, baik yang berskala menengah maupun besar [24].

PostgreSQL, yang dieja sebagai *Post-Gres-Q-L*, atau lebih dikenal sebagai *postgres*, adalah Sistem Manajemen *Database* Relasional Objek yang *bersifat open source*. PostgreSQL menonjolkan ekstensibilitas, kreativitas, dan kompatibilitas dalam pengembangannya. Bersaing dengan penyedia sistem basis data relasional besar seperti Oracle, MySQL, SQL Server, dan lainnya, PostgreSQL memiliki penggunaan yang luas di berbagai sektor, termasuk lembaga pemerintah, sektor publik, dan sektor swasta. PostgreSQL merupakan DBMS *cross-platform* dan dapat dijalankan pada berbagai sistem operasi [25].

2.9. Blackbox Testing

Blackbox testing adalah metode pengujian fungsional perangkat lunak yang tidak melibatkan pengujian terhadap desain dan kode program secara spesifik. Pendekatan pengujian *Blackbox testing* ini lebih fokus pada pemahaman terhadap fungsionalitas perangkat lunak. Keunggulan dari metode ini terletak pada kemudahan penggunaannya, karena hanya memerlukan batas bawah dan batas atas dari nilai yang diharapkan [26].

Metode *Blackbox testing* adalah pendekatan pengujian yang memungkinkan pengujian terhadap setiap fitur aplikasi dalam pengujian perangkat lunak. Metode ini terutama fokus pada pengujian fungsionalitas perangkat lunak, memungkinkan penguji untuk menguji berbagai fitur melalui berbagai masukan yang berbeda pada setiap formulir atau komponen untuk memastikan bahwa program yang dibuat sesuai dengan spesifikasi yang telah direncanakan [27].

2.10. Postman

Postman merupakan sebuah *platform* yang diciptakan untuk merancang dan menguji *API* yang telah dibuat sebelumnya. Fungsionalitasnya mirip dengan peran klien yang dapat mengakses antarmuka *REST* secara langsung. *Postman* memungkinkan pengujian otomatis, simulasi *endpoints* secara langsung, pembuatan dokumentasi *API*, pemantauan kinerja, pengukuran waktu *response* dari *API*, dan juga penggunaan *API* secara *real-time* [28].

2.11. Penelitian Terkait

Pada sebuah penelitian yang membahas mengenai perancangan *user interface* dan *user experience* aplikasi *mobile e-learning* yang berbasis gamifikasi dengan menggunakan metode *design thinking* menjelaskan mengenai perancangan tampilan dan pengalaman *user* dalam menggunakan aplikasi sebagai langkah untuk memotivasi mahasiswa kedokteran program profesi dengan menyediakan media pembelajaran yang menyenangkan. Hasil dari penelitian tersebut berupa prototype aplikasi *mobile e-learning* dengan grade B pada pengukuran SUS dan masuk ke dalam kategori baik dalam hasil pengujian UEQ [29]. Pada penelitian tersebut dikembangkan suatu fitur bernama “*Journey*” yang menjadi referensi utama pada penelitian ini. Kemudian pada penelitian lain yang bertujuan untuk menghasilkan gamifikasi pada mata kuliah Pengembangan Media Sederhada di Program Studi Teknologi Pendidikan Universitas Negeri Jakarta dengan menggunakan metode pengembangan *Model Rapid Prototyping* didapatkan hasil bahwa mahasiswa tertarik dengan gamifikasi pada kegiatan pembelajaran UTS mata kulliah Pengembangan Media Sederhana [30].

Penelitian lain membahas mengenai pengembangan aplikasi untuk mengelola data siswa, orang tua, guru, dan karyawan pada SMAN 4 Malang dengan mengimplementasikan metode SDLC (*Software Development Life Cycle*) *scrum*. Metode *scrum* dipilih karena selalu melibatkan pengguna dalam proses pengembangan sistem agar hasil sesuai dengan permintaan pengguna [11]. Pada penelitian lain yang bertujuan untuk mengimplementasikan *scrum* dalam merancang sistem informasi untuk tata usaha sekolah berbasis web di MTs Sirojul Athfal dan disimpulkan bahwa metode *scrum* merupakan metode yang baik dan cepat untuk diimplementasikan dalam proses pembuatan sistem informasi dan dapat mengatasi kebutuhan *user* yang dinamis pada saat pengembangan sistem [31].

Lalu terdapat penelitian yang membahas mengenai pengembangan REST API menggunakan NodeJS untuk mengembangkan aplikasi transaksi jasa elektronik berbasis *android*. Penelitian tersebut menjelaskan bagaimana cara mengembangkan REST API menggunakan NodeJS dengan menginstall modul-modul yang terdapat di dalamnya seperti JSON Web Token (JWT), Sequelize, dan dotenv [32]. Dalam penelitian lain membahas perbandingan *response time* dari API yang dikembangkan menggunakan *framework* ExpressJS dan Codeigniter. Hasil yang didapat adalah *response time* dari API ExpressJS lebih cepat dengan nilai rata-rata 438,64 ms, sedangkan API Codeigniter lebih lambat dengan nilai rata-rata 550,72 ms. Sehingga disimpulkan bahwa *framework* ExpressJS cocok untuk diimplementasikan pada sistem yang diakses oleh banyak *user* [9].

Terdapat penelitian lain yang membahas mengenai perancangan dan implementasi restful API Sistem Informasi Manajemen Dosen pada Universitas Udayana. Penelitian tersebut menerapkan metode *blackbox testing* untuk menguji API menggunakan aplikasi *Postman*. Berdasarkan pengujian tersebut didapatkan bahwa seluruh API berfungsi dengan baik [33]. Terdapat penelitian lain yang juga menerapkan *blackbox testing* untuk pengujian sistem dengan menggunakan aplikasi *Postman* pada API *microservices* yang dibangun menggunakan *framework* ExpressJS dan Flask. Pengujian dilakukan dengan mendefinisikan skenario, prosedur pengujian, masukan (*input*), ekspektasi keluaran (*output*), dan hasil pengujian [34].

Tabel 1. Penelitian Terkait

No.	Peneliti	Objek	Metode/Tools	Hasil
1.	Azzahra, A. K. (2023) [29]	UI/UX Gamifikasi Aplikasi <i>Mobile E-Learning</i> (Studi Kasus: Program Profesi Dokter Universitas Lampung).	<i>Design Thinking</i> , SUS, UEQ	<i>Prototype</i> aplikasi <i>mobile e-learning</i> dengan <i>grade B</i> pada pengukuran SUS dan masuk ke dalam kategori baik dalam hasil pengujian UEQ.
2.	Ariani, D., Fatharani, W., & Wargahadibrata, R. H. (2022) [30]	Gamifikasi pada pembelajaran mata kuliah Pengembangan Media Sederhana.	<i>Model Rapid Prototyping</i> , survey pendapat menggunakan Google Form	Gamifikasi dengan elemen skor, <i>level</i> , tantangan, <i>leaderboard</i> , dan <i>lencana</i> .
3.	Rifai, M., Wibawa, A., Zaeni, I., & Setiawan, D. (2021) [11]	Pengembangan aplikasi bidang kesiswaan SMAN 4 Malang.	<i>Scrum</i>	Aplikasi yang terimplementasikan dalam siklus 5 <i>sprint</i> .
4.	Mardika, P. D., & Fauzi, A. (2022) [31]	Implementasi metode <i>Scrum</i> dalam merancangan sistem informasi untuk tata usaha sekolah berbasis web di MTs Sirojul Athfal.	<i>Scrum</i>	Hasil penelitian menunjukkan bahwa metode <i>Scrum</i> sangat baik untuk digunakan dalam pembuatan sistem informasi yang membutuhkan waktu yang cepat dan kebutuhan sistem yang dinamis.

Tabel 1. Penelitian Terkait (Lanjutan)

No.	Peneliti	Objek	Metode/Tools	Hasil
5.	Kurniawan, I., & Rozi, F. (2020) [32]	Pengembangan REST API Aplikasi Transaksi Jasa Elektronik Berbasis <i>Android</i>	NodeJS	<i>Server API</i> yang terdiri dari 54 <i>API</i> dan memiliki format JSON.
6.	Mulana, L., Prihandani, K., & Rizal, A. (2022) [9]	Perbandingan <i>API</i> ExpressJS dan <i>Codeigniter</i>	<i>Performance</i> <i>testing</i> dengan JMeter.	<i>Response time</i> dari <i>API</i> ExpressJS lebih cepat dibandingkan dengan <i>API</i> <i>Codeigniter</i> .
7.	Paramitha, I. A. K. P., Wiharta, D. M., & Arsa, I. M. (2022) [33]	Pengujian fungsionalitas <i>API</i> Sistem Informasi Manajemen Dosen.	<i>Blackbox testing</i> , <i>postman</i>	Seluruh fungsi dari <i>API</i> berhasil berjalan dengan baik.
8.	Royyan, F. F. (2023) [34]	Pengujian <i>API</i> <i>framework</i> NodesJS dan Flask pada <i>microservices</i> .	<i>Blackbox testing</i> , <i>postman</i>	Pengujian pada 23 <i>API</i> yang terdiri dari 18 <i>API</i> NodeJS dan 5 <i>API</i> Flask mendapatkan persentase keberhasilan sebesar 100%.

Penelitian ini bertujuan untuk mengembangkan *backend API* pada aplikasi pembelajaran berbasis gamifikasi dengan menggunakan *framework ExpressJS*, metode pengembangan perangkat lunak *Scrum*, dan metode pengujian *Blackbox testing* dengan bantuan *tools* perangkat lunak *postman* untuk melakukan pengujian fungsional pada *API* yang telah dikembangkan.

III. METODOLOGI PENELITIAN

3.1. Waktu dan Tempat

Penelitian dan pembuatan Tugas Akhir ini adalah dilakukan pada:

Waktu : Desember 2023 – Mei 2024

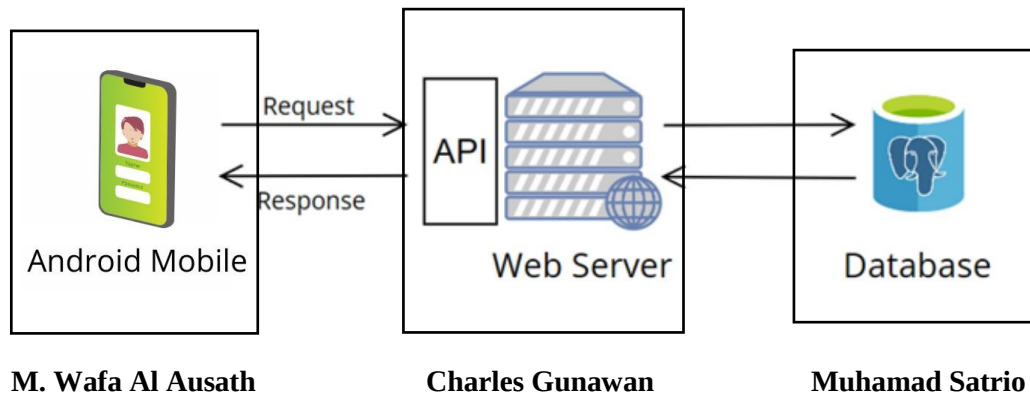
Tempat : Laboratorium Komputer, Teknik Elektro Universitas Lampung

Jadwal kegiatan penelitian yang akan dilakukan adalah sebagai berikut:

Tabel 2. Waktu Penelitian

No	Aktivitas	2023				2024																			
		Desember				Januari				Februari				Maret				April				Mei			
		1	2	3	4	1	2	3	4	1	2	3	4	1	2	3	4	1	2	3	4	1	2	3	4
1	Studi Literatur																								
2	Product backlog																								
3	Sprint Planning																								
4	Sprint																								
5	Sprint Review																								
6	Sprint Retrospective																								
7	Analisis Hasil																								

3.2. Tim Penelitian



Gambar 5. Peran Tim Pengembang

Pengembangan proyek penelitian ini dilakukan bersama oleh tim yang terdiri dari dua orang pengembang *backend* dan satu orang pengembang *frontend*. Pengembang *backend* yaitu Charles Gunawan dan Muhamad Satrio yang fokus membangun *server* untuk mengelola data dan menyimpan gambar yang dibutuhkan, menyediakan *API* yang dapat digunakan *frontend* untuk menghubungkan aplikasi ke *server*, dan membuat *database* untuk menyimpan data yang dibutuhkan aplikasi. Teknologi yang digunakan dalam pengembangan *backend* adalah *framework* ExpressJS untuk membangun *server* dan *API* dengan menggunakan bahasa JavaScript, serta PostgreSQL sebagai *database* nya.

Pengembang *frontend* pada penelitian ini adalah Muhammad Wafa Al Ausath yang fokus merancang dan mengembangkan *frontend* atau tampilan yang berinteraksi langsung dengan *user* berbasis aplikasi *mobile* (*Android*). Bahasa yang digunakan dalam pengembangan aplikasi *mobile* berbasis *Android* ini adalah Kotlin yang merupakan bahasa natif yang digunakan untuk mengembangkan aplikasi *mobile* berbasis *Android*.

3.3. Alat Penelitian

Adapun alat dan bahan yang dibutuhkan dalam penelitian ini adalah sebagai berikut:

Tabel 3. Alat Penelitian

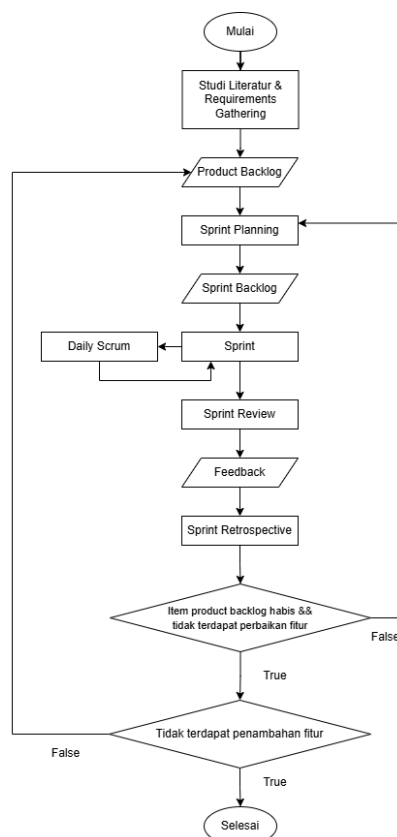
No	Perangkat	Spesifikasi	Kegunaan
1	Laptop	Ryzen 5 4600H	Perangkat pembuat.
2	Visual Studio <i>Code</i>	<i>Version 1.83.1</i>	Perangkat lunak sebagai <i>text editor</i> untuk mengembangkan API.
3	ExpressJS	<i>Version 4.16.1</i>	<i>Framework</i> berbasis bahasa Javascript yang digunakan dalam pengembangan aplikasi.
4	NodeJS	<i>Version 18.15.0</i>	<i>Runtime javascript</i> yang digunakan untuk menjalankan aplikasi.
5	<i>Postman</i>	<i>Version 10.18</i>	Perangkat lunak untuk melakukan <i>testing</i> pada API yang dihasilkan.
6	Umlet	<i>Version 14.3.0</i>	Perangkat lunak untuk melakukan desain sistem
7	<i>Deployment (Development)</i>	Cyclic.sh (app), vercel (database)	<i>Deployment service</i> yang digunakan agar dapat mengakses <i>database</i> bersama dengan tim, dan aplikasi agar dapat digunakan secara <i>online</i> .
8	Github	Git <i>version 2.37.1</i>	Platform untuk melakukan kontrol versi dan kolaborasi dalam pengembangan perangkat lunak.
9	Trello	-	Platform untuk berkolaborasi dan mengatur jadwal atau agenda <i>sprint</i> .

Tabel 3. Alat Penelitian (Lanjutan)

No	Perangkat	Spesifikasi	Kegunaan
10	<i>Deployment</i> (<i>Production</i>)	Idwebhost	Melakukan hosting agar aplikasi dapat digunakan secara <i>online</i> dan tanpa kendala.

3.4. Metode Penelitian

Pada pengembangan sistem ini digunakan metode *Software Development Life Cycle* (SDLC) yaitu *Scrum* dimana terdapat beberapa tahapan yaitu, *product backlog*, *sprint planning*, *sprint & testing*, *sprint review*, dan *sprint retrospective*. Metode *scrum* merupakan metode yang iteratif, dimana di setiap iterasinya terdapat penambahan atau peningkatan fitur atau progres dari iterasi *sprint* sebelumnya. Berikut merupakan diagram *flowchart* penelitian yang menjelaskan alur tahapan penelitian.



Gambar 6. Tahapan Penelitian

Detail dari setiap tahapan pada Gambar 6 dapat dilihat pada penjelasan berikut ini.

3.4.1. *Product backlog*

Product backlog adalah hasil dari kegiatan *requirements gathering* atau pengumpulan kebutuhan dari aplikasi yang akan dikembangkan berupa daftar pekerjaan yang perlu dilakukan untuk mengembangkan dan memelihara produk. Berdasarkan hasil studi literatur pada penelitian sebelumnya yang membahas mengenai perancangan *user interface* dan *user experience* pada aplikasi *mobile e-learning* berbasis gamifikasi dengan metode *design thinking* dengan studi kasus pada program profesi dokter Universitas Lampung [29] dan pembahasan bersama *product owner*, diidentifikasi terdapat beberapa kebutuhan pada aplikasi ini, seperti:

1. Terdapat sistem *login* dengan 2 *role user*, yaitu admin dan *player*.
2. Terdapat elemen game seperti *challenge*, *avatar*, *points*, *coins*, *experience points (exp)*, *leaderboard*, *history*, dan *achievement*.
3. Terdapat beberapa stase, di setiap stase terdapat beberapa bab, dan di setiap bab memiliki beberapa *challenge*.
4. *Challenge* merupakan skenario simulatif dari pasien yang datang berobat ke rumah sakit dengan keluhan tertentu, lalu *player* dapat melihat hasil investigasi yang dilakukan terhadap pasien, kemudian *player* sebagai mahasiswa koas yang bertugas membantu dokter harus mencoba mendiagnosa penyakit pasien dan memberikan penanganan yang tepat dengan memilih pilihan yang tepat.
5. Dalam aplikasi *user* dengan *role player* dapat mendaftarkan akun, *login*, mengerjakan *challenge*, membeli *avatar*, mengubah *avatar* yang digunakan, mengedit profil, melihat *history*, membuat topik diskusi, menambahkan komentar dalam diskusi, menggunakan fitur lupa *password*, melihat *leaderboard*, dan mengambil *achievement*.
6. *User* dengan *role admin* dapat melakukan *login* dan mengelola *user*, *avatar*, stase, bab, *challenge*, serta mengelola diskusi dan komentar *user*.
7. Terdapat *leaderboard* yang akan diperbaharui setiap bulannya.

3.4.2. *Sprint Planning*

Membagi tugas yang akan dikerjakan dalam *sprint*. Kebutuhan yang ada pada *product backlog* dirincikan menjadi beberapa kebutuhan fungsional yang dapat

dilihat pada Tabel 4. Kebutuhan Fungsional. Kebutuhan fungsional tersebut akan menjadi tugas anggota tim untuk mengimplementasikannya. Hasil dari *sprint planning* adalah *sprint backlog* yang berisi detail pekerjaan, tanggung jawab individu atau tim, dan estimasi waktu selesai untuk menyelesaikan tugas. *Sprint backlog* bersifat dinamis sehingga dapat berubah seiring berjalannya *sprint*.

Tabel 4. Kebutuhan Fungsional

No.	Fitur	Kebutuhan Fungsional
1.	Register	Sistem harus dapat menyimpan data <i>user</i> seperti email, password, nama, dan <i>role</i> .
2.	Login	Sistem harus dapat membedakan <i>user</i> berdasarkan <i>role</i> . Sistem harus dapat memeriksa apakah email dan password yang <i>diinput user</i> cocok dan ada di <i>database</i> .
3.	Challenge	Sistem harus dapat menampilkan <i>list</i> dari stase, bab, dan <i>challenge</i> . Sistem harus dapat menampilkan hasil investigasi, pilihan jawaban diagnosa, dan pilihan jawaban penanganan. Sistem harus dapat memeriksa hasil pengerjaan untuk dapat melanjutkan ke <i>challenge</i> selanjutnya. Sistem harus dapat menampilkan nilai dari pengerjaan <i>challenge</i> .
4.	Diskusi dan Komentar.	Sistem harus dapat memastikan kondisi <i>challenge</i> telah diselesaikan oleh <i>user</i> . Sistem harus dapat menyimpan topik diskusi yang diberikan <i>user</i> . Sistem harus dapat menyimpan komentar yang diberikan <i>user</i> dalam suatu topik diskusi. Sistem harus dapat menampilkan diskusi dan komentar yang <i>user</i> tambahkan. Sistem harus dapat mengubah data diskusi. Sistem harus dapat mengubah data komentar. Sistem harus dapat menghapus data diskusi. Sistem harus dapat menghapus data komentar.

Tabel 4. Kebutuhan Fungsional (Lanjutan)

5.	Melihat <i>history</i>	Sistem harus dapat menyimpan hasil pengerjaan <i>challenge</i> .
		Sistem harus dapat menampilkan <i>history</i> pengerjaan <i>challenge</i> oleh <i>user</i> .
6.	Membeli avatar	Sistem harus dapat menyimpan <i>coins</i> yang didapat <i>user</i> .
		Sistem harus dapat memastikan <i>coins</i> yang dimiliki <i>user</i> cukup untuk membeli avatar.
		Sistem harus dapat mengurangi <i>coins</i> ketika <i>user</i> membeli avatar.
7.	Mengubah avatar	Sistem harus dapat menyimpan avatar yang digunakan <i>user</i> saat ini.
8.	Edit profil	Sistem harus dapat menampilkan data <i>user</i> yang login.
		Sistem harus dapat menyimpan perubahan pada data <i>user</i> yang diubah.
9.	Lupa password	Sistem harus dapat memeriksa ketersediaan email yang <i>user inputkan</i> di dalam <i>database</i> .
		Sistem harus dapat mengirimkan OTP ke email <i>user</i> .
		Sistem harus dapat menyocokkan OTP yang <i>diinput user</i> dengan OTP yang telah diberikan kepada <i>user</i> .
10.	Melihat leaderboard	Sistem harus dapat menyimpan points yang didapat <i>user</i> ketika mengerjakan <i>challenge</i> .
		Sistem harus dapat mengakumulasi points yang <i>user</i> dapatkan.
		Sistem harus dapat menampilkan <i>user</i> pada leaderboard secara berurutan berdasarkan points yang didapatkan.

Tabel 4. Kebutuhan Fungsional (Lanjutan)

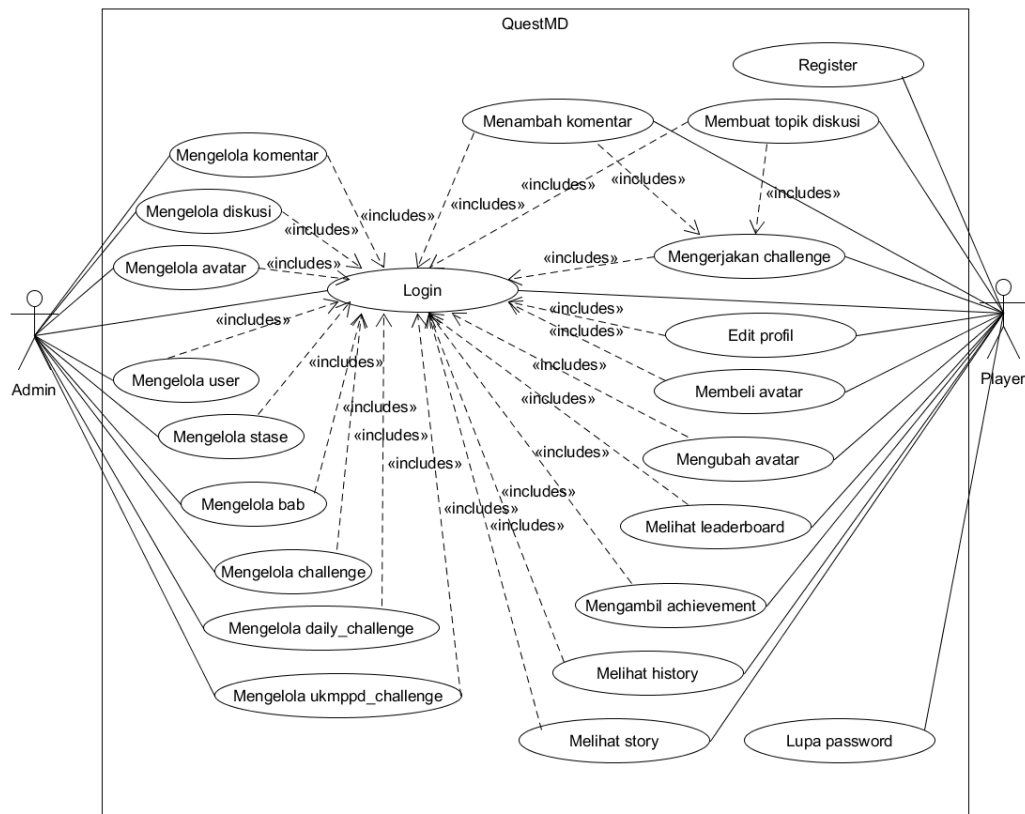
11.	Mengambil achievement	Sistem harus dapat memeriksa keadaan <i>user</i> dengan kondisi achievement.
12.	Mengelola <i>user</i>	Sistem harus dapat menampilkan <i>list</i> data <i>user</i> .
		Sistem harus dapat menambahkan data pada tabel <i>user</i> .
		Sistem harus dapat mengubah data pada tabel <i>user</i> .
		Sistem harus dapat menghapus data pada tabel <i>user</i> .
13.	Mengelola avatar	Sistem harus dapat menampilkan <i>list</i> data avatar.
		Sistem harus dapat menambahkan data pada tabel avatar.
		Sistem harus dapat menyimpan gambar yang diinputkan.
		Sistem harus dapat mengubah data pada tabel avatar.
		Sistem harus dapat menghapus data pada tabel avatar.
		Sistem harus dapat menghapus gambar apabila terjadi perubahan atau penghapusan gambar.
14.	Mengelola stase	Sistem harus dapat menampilkan <i>list</i> data stase.
		Sistem harus dapat menambahkan data pada tabel stase.
		Sistem harus dapat mengubah data pada tabel stase.
		Sistem harus dapat menghapus data pada tabel stase.
15.	Mengelola bab	Sistem harus dapat menampilkan <i>list</i> data bab.
		Sistem harus dapat menambahkan data pada tabel bab.
		Sistem harus dapat menyimpan gambar yang diinputkan sebagai <i>icon</i> dari bab.
		Sistem harus dapat mengubah data pada tabel bab.
		Sistem harus dapat menghapus data pada tabel bab.
		Sistem harus dapat menghapus gambar apabila terjadi perubahan atau penghapusan gambar.

Tabel 4. Kebutuhan Fungsional (Lanjutan)

16.	Mengelola <i>challenge</i>	Sistem harus dapat menampilkan <i>list</i> data <i>challenge</i> .
		Sistem harus dapat melakukan penambahan data, menampilkan <i>list</i> data, mengubah data <i>challenge</i> , mengubah opsi pilihan <i>challenge</i> , dan menghapus data <i>challenge</i> oleh admin.
		Sistem harus dapat menambahkan data <i>challenge</i> , beserta investigasi, opsi penyakit, opsi penanganan, kunci penyakit dan kunci penanganannya.
		Sistem harus dapat menampilkan <i>list</i> investigasi dari <i>challenge</i> .
		Sistem harus dapat mengubah investigasi dari <i>challenge</i> .
		Sistem harus dapat menampilkan <i>list</i> opsi penyakit dari <i>challenge</i> .
		Sistem harus dapat mengubah opsi penyakit dari <i>challenge</i> .
		Sistem harus dapat mengubah kunci penyakit dari <i>challenge</i> .
		Sistem harus dapat menampilkan <i>list</i> opsi penanganan dari <i>challenge</i> .
		Sistem harus dapat mengubah opsi penanganan dari <i>challenge</i> .
		Sistem harus dapat mengubah kunci penanganan dari <i>challenge</i> .
		Sistem harus dapat menghapus data <i>challenge</i> beserta data investigasi, opsi penyakit, opsi penanganan, kunci penyakit, dan kunci penanganan.
17.	Mengelola diskusi dan komentar	Sistem dapat menghapus topik diskusi.
		Sistem dapat menghapus topik komentar.

Pada tahap *sprint planning* juga dilakukan penggambaran desain sistem yang sesuai dengan kebutuhan sistem. Penggambaran desain sistem dilakukan dengan menggunakan usecase diagram untuk merepresentasikan aktor dan hak aksesnya pada sistem, dan menggunakan ERD (*Entity Relationship Diagram*) untuk merepresentasikan struktur data dalam *database*.

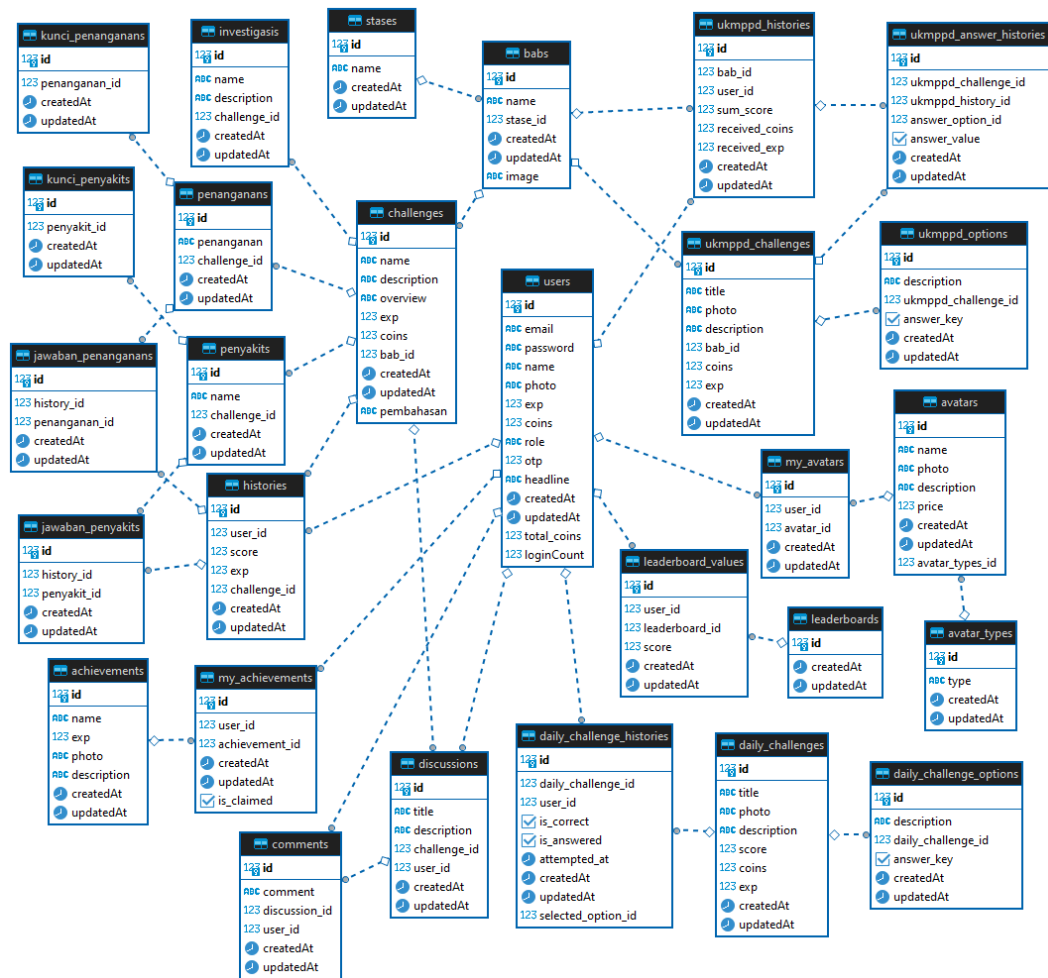
3.4.2.1. Use Case Diagram



Gambar 7. Usecase Diagram

Gambar 7 menggambarkan interaksi aktor atau *user* dengan sistem. Diagram tersebut juga menggambarkan hak akses *user* di dalam sistem. *User* dengan *role player* dapat melakukan *register*, *login*, mengerjakan *challenge*, edit profil, membeli *avatar*, mengubah *avatar*, melihat *leaderboard*, mengambil *achievement*, melihat *history*, dan mengganti *password* apabila *user* lupa *password*. Sedangkan *user* dengan *role admin* dapat melakukan *login* dan mengelola *user*, *avatar*, *stase*, *bab*, *challenge*, *daily challenge*, *ukmppd challenge*, serta mengelola diskusi dan komentar *user*.

3.4.2.2. Entity Relationship Diagram



Gambar 8. ERD QuestMD

Berdasarkan Gambar 8 terdapat beberapa entitas berhubungan yang menggambarkan hubungan antar tabel di *database*. ERD dibuat dengan mengikuti kebutuhan aplikasi. Pada proses pembuatannya, diagram ini telah melalui beberapa kali perubahan pada setiap iterasi *sprint* yang berlangsung.

3.4.3. Sprint (Execution)

Pada tahap ini, anggota tim mengerjakan tugas yang ada di *sprint backlog* dan melakukan *testing* secara pribadi pada lingkungan lokal terhadap pekerjaan tersebut, sehingga *code* yang nantinya digabungkan (*merge*) sudah sesuai dengan kebutuhan dan tidak ada konflik. Berdasarkan kesepakatan tim, setiap *sprint* berdurasi 1 minggu (7 hari). Pada masa *sprint* setiap hari nya terdapat *daily scrum*

atau *daily meet* yang berlangsung sekitar 15 menit melalui software *Discord*, untuk membahas progress pengerjaan, kendala yang dialami, dan apa yang akan dilakukan kedepannya.

3.4.3.1. Pengembangan Server dan API

Server dikembangkan menggunakan *expressjs* yang merupakan *webserver framework* dari runtime *nodejs* yang menggunakan basis bahasa JavaScript. Ketika menjalankan *server expressjs*, secara *default server* akan berjalan di port 3000. Pengembangan *API* dilakukan di dalam *server expressjs*. Ketika membicarakan mengenai *API* pada *server expressjs*, hal tersebut sama dengan *routes* atau *endpoints* di dalamnya yang digunakan sebagai penghubung antara pengguna (*client*) dengan *server*. *Detail* pengembangan *server* dapat dilihat pada poin-poin berikut:

a. Pembuatan Server

Pembuatan aplikasi dilakukan dengan menjalankan *express-generator* untuk membuat struktur awal *server expressjs* secara otomatis. Cara ini dapat mempercepat proses pembuatan aplikasi *server* yang dapat langsung dijalankan. Contoh *source code* dapat dilihat pada halaman Lampiran A1.

Berdasarkan Lampiran A1 terlihat bahwa terdapat beberapa pemanggilan module seperti *cors*, *express*, dan *router*. *Express* merupakan inti dari *framework* untuk membangun aplikasi di dalam *nodejs*. *Cors* digunakan untuk mengontrol akses ke sumber daya pada sistem. *Router* merupakan penunjuk yang digunakan untuk mengarahkan client ke fungsi tertentu melalui suatu jalur yang telah didefinisikan.

b. Pembuatan Model

Pembuatan *model* pada *server* menggunakan ORM (*Object Relational Mapping*) bernama *sequelize*. *Sequelize* digunakan agar *server* dapat berinteraksi dengan *database* menggunakan fungsi yang disediakan tanpa harus menuliskan *query*. Contoh *source code* dapat dilihat pada halaman Lampiran A2.

Berdasarkan Lampiran A2 terlihat bahwa terdapat pendefinisian untuk hubungan antara model *users* dengan model lain. Hubungan tersebut mengikuti desain *ERD* yang dapat dilihat pada Gambar 3. *ERD QuestMD*. Kemudian terdapat pendefinisian atribut dari model *users* dengan tipe data yang dimilikinya.

c. Pembuatan *Controller*

Controller merupakan fungsi yang terdapat dalam *server* untuk menjalankan suatu perintah dan mengelola data pada *model* dengan melakukan *create*, *read*, *update*, dan *delete*. *Controller* mengatur kebutuhan *input* dan *output* yang akan dikeluarkan. Contoh *source code* dapat dilihat pada halaman Lampiran A3.

Berdasarkan Lampiran A3 terdapat fungsi-fungsi yang digunakan untuk mengelola data dari model tertentu. Fungsi-fungsi yang digunakan adalah *create*, *findOne*, *findAll*, *update*, dan *destroy*. Fungsi *create* digunakan untuk melakukan penambahan data pada model, lalu fungsi *findOne* digunakan untuk mengambil satu data dari model, *findAll* digunakan untuk mengambil semua data dari model, *update* digunakan untuk melakukan *edit* terhadap satu data pada model, dan *destroy* digunakan untuk menghapus satu data pada model.

d. Pembuatan *Routes*

Routes atau *endpoints* merupakan penghubung antara pengguna dan *server*. *Routes* digunakan sebagai pengarah yang mengatur rute atau tujuan ke suatu fungsi tertentu di dalam *server*. Rincian *endpoints* yang akan dibuat dapat dilihat pada halaman Lampiran B. Daftar *Endpoints*.

Kemudian untuk *source code route* dapat dilihat pada halaman Lampiran A4. Berdasarkan Lampiran A4, *router* dibuat dengan mendefinisikan *method* yang digunakan, *path* atau jalur yang digunakan, *middleware* yang digunakan (opsional), dan fungsi yang akan digunakan apabila mengakses jalur tersebut yang diambil dari *file controller*. Pada *stase.js* digunakan *middleware auth handling* untuk membatasi hak akses pada *router* tersebut.

e. Pembuatan *Middleware*

Middleware pada konteks penelitian ini merupakan potongan kode yang memiliki tugas khusus untuk membantu dalam proses *input* dan *output* dalam sistem. *Middleware* yang terdapat pada aplikasi ini adalah *auth handler* yang akan berisi logika untuk menangani proses otorisasi yang memeriksa hak akses pengguna untuk mengakses *router* dan fungsi tertentu. Lalu terdapat *logger* untuk melakukan penulisan riwayat *error* yang terjadi dalam aplikasi, kemudian *file handling* untuk mengatur *input file* dan memvalidasi *file* yang digunakan. Contoh *source code* dapat dilihat pada halaman Lampiran A5.

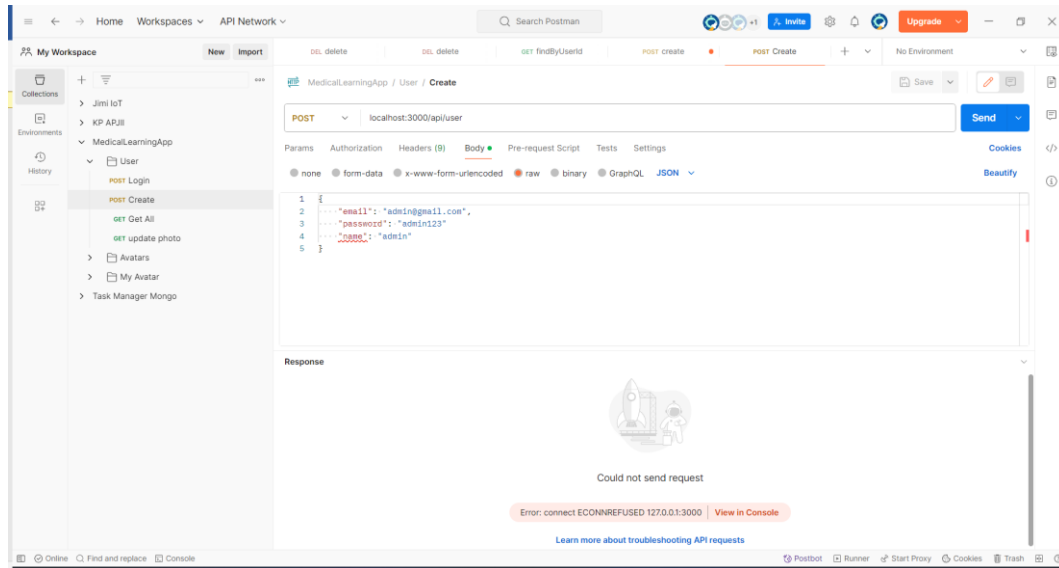
Berdasarkan Lampiran A5, digunakan *jsonwebtoken* untuk memverifikasi *payload* yang dibawa *user* menggunakan tokennya yang didapat ketika login. Fungsi dari pembuatan *middleware isLogin* ini adalah membatasi hak akses pada *router* tertentu sehingga hanya *user* yang telah *login* saja yang dapat mengakses *router* tersebut tanpa menghiraukan *role* nya.

f. Pembuatan *Utils*

Utils (utilities) merupakan kumpulan fungsi yang menjalankan tugas-tugas kecil khusus. Pada konteks penelitian ini, fungsi yang digunakan adalah *error handling* untuk mengatur *error output* ketika terjadi *error* dan mengirimkan *error* tersebut ke *middleware logger* untuk dicatat ke dalam *file* khusus. Contoh *source code* dapat dilihat pada halaman Lampiran A6.

3.4.3.2. Testing

Testing dilakukan menggunakan metode *blackbox testing* untuk menguji fungsionalitas dari setiap *endpoints* yang disediakan. Pengujian dilakukan dengan mendefinisikan fitur yang akan diuji, skenario pengujian, *output* yang diharapkan, dan hasil pengujian. Parameter dalam pengujian ini adalah persentase keberhasilan pengujian terhadap semua API. *Tools* yang digunakan dalam pengujian ini adalah *software* bernama *Postman* untuk melakukan *request* terhadap API yang akan diuji. Pada *software* tersebut juga dapat melakukan *input* apabila diperlukan dan melihat *output* yang dikeluarkan.



Gambar 9. Tampilan *Postman*

Testing dilakukan di dalam pelaksanaan *sprint* sehingga kode *API* yang telah dibuat telah melalui pengujian oleh pengembang yang membuat *API* tersebut pada lingkungan lokal pengembang. Hal ini dilakukan untuk memastikan *API* yang dibuat bekerja dengan baik sesuai ekspektasi dan kebutuhan, tidak terdapat *error* yang memiliki efek pada fungsi lain, dan tidak ada konflik ketika digabungkan (*merge*) dengan kode utama di github. Adapun format tabel yang digunakan untuk *testing* dapat dilihat pada Tabel 5.

Tabel 5. Tabel *Testing*

No.	[Metode] <i>Endpoints</i>	Skenario <i>Input</i>	Ekspektasi <i>Output</i>	Hasil
1	...	...	...	...

Pada Tabel 5 akan didefinisikan metode dan *endpoints* yang akan diuji, skenario pengujian yang mencakup data *input* sebagai masukkan nya, ekspektasi dari *output* yang akan dikeluarkan, dan hasil pegujian yang akan dicatat sebagai sukses atau gagal.

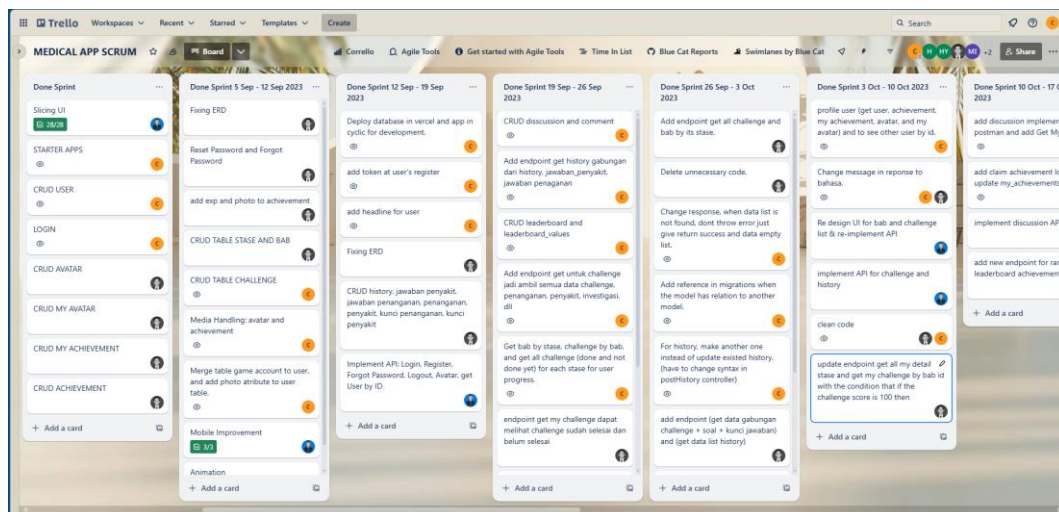
3.4.4. *Sprint Review*

Tahap *sprint review* dilakukan setelah masa *sprint* selesai. Dilakukan pertemuan bersama tim dan *stakeholders* untuk membahas hasil *sprint* yang sudah dilakukan

dan melakukan demonstrasi dari hasil kerja agar mendapat *feedback* dari *stakeholders*. *Feedback* tersebut dapat berupa persetujuan atas fitur yang telah dikembangkan pada sistem, kritik dan saran terhadap fitur yang telah dikembangkan, dan penambahan fitur apabila diinginkan. Berdasarkan *feedback* tersebut, apabila ada perubahan atau penambahan fitur pada sistem dapat dimasukkan ke dalam *product backlog* dan dipertimbangkan untuk dimasukkan ke dalam *sprint backlog* untuk *sprint* berikutnya.

3.4.5. *Sprint Retrospective*

Pada tahap *sprint retrospective* dilakukan pertemuan bersama tim tanpa *stakeholders* untuk membahas kinerja selama *sprint* yang sudah dilakukan sebagai tim, agar dapat bekerjasama dengan baik pada proses *sprint* selanjutnya. Pada tahap ini tim mendiskusikan apa yang sudah berjalan dengan baik dan harus dipertahankan, apa kendala yang dihadapi dan bagaimana solusinya, serta apa yang harus ditingkatkan dalam bekerja sebagai tim. Tahap ini dilakukan melalui perangkat lunak bernama *Discord* untuk melakukan pertemuan secara *online*. Pada tahap ini juga tim mengelola dokumentasi pekerjaan di *Trello* agar menjadi rapih, sesuai dengan kenyataan, dan dapat dipahami bersama.



Gambar 10. Tampilan Trello

V. KESIMPULAN DAN SARAN

5.1. Kesimpulan

Berdasarkan hasil yang telah disampaikan didapatkan kesimpulan sebagai berikut:

1. Sistem *backend API* untuk aplikasi QuestMD telah berhasil dibangun menggunakan *framework* ExpressJS dengan memenuhi semua kebutuhan dalam *product backlog*.
2. Sistem *backend* berhasil dibangun dalam 17 iterasi *sprint* yang terdiri dari 10 iterasi pengembangan produk awal, 2 iterasi pengembangan fitur tambahan, 3 iterasi pengembangan fitur admin, dan 2 iterasi penambahan data dengan total *endpoint* yang dibentuk adalah 72 *endpoints* dan total tugas dalam *sprint backlog* yang dikerjakan adalah 81 tugas.
3. Berdasarkan hasil pengujian pada setiap iterasi menggunakan metode *blackbox testing*, semua fitur yang dikembangkan telah berhasil melewati pengujian *blackbox* dengan persentase keberhasilan 100%. Fitur-fitur ini telah memenuhi semua *sprint backlog* pada setiap iterasinya, serta terintegrasi dengan *frontend* dan *database*.

5.2. Saran

Saran untuk pengembangan selanjutnya adalah sebagai berikut:

1. Penelitian selanjutnya agar dapat melakukan *testing* untuk menguji kelayakan dan kecepatan *response* sistem dengan melakukan *performance testing* yang dilakukan melalui *load testing* tool bernama BlazeMeter. *Performance testing* dilakukan untuk memastikan bahwa sistem tidak akan *crash* di bawah situasi krisis.

2. Pada saat penelitian ini berlangsung sedang tren *runtime* JavaScript baru bernama Bun di mana pada website resminya mengatakan bahwa performa dari Bun lebih baik daripada NodeJs. Pada penelitian selanjutnya dapat dilakukan pengembangan dengan menggunakan *runtime* Bun agar dapat menjadi perbandingan.

DAFTAR PUSTAKA

- [1] Y. A. Mannas, “Hubungan Hukum Dokter dan Pasien Serta Tanggung Jawab Dokter dalam Penyelenggaraan Pelayanan Kesehatan,” *Jurnal Cita Hukum*, vol. 6, no. 1, pp. 163–182, Juni 2018, doi: 10.15408/jch.v6i1.8274.
- [2] Presiden Republik Indonesia, Undang-Undang Republik Indonesia nomor 20 Tahun 2013 Tentang Pendidikan Kedokteran. Jakarta, 2013.
- [3] L. Fajriati, Y. Yaunin, and L. Isrona, “Perbedaan Derajat Kecemasan pada Mahasiswa Baru Preklinik dan Klinik Fakultas Kedokteran Universitas Andalas,” *Jurnal Kesehatan Andalas*, vol. 6, no. 3, p. 546, Februari 2018, doi: 10.25077/jka.v6i3.736.
- [4] N. Dianti and A. Findyartini, “Hubungan Tipe Motivasi terhadap Kejadian Burnout pada Mahasiswa Fakultas Kedokteran Universitas Indonesia pada Masa Transisi dari Pendidikan Preklinik ke Klinik Tahun 2018,” *eJournal Kedokteran Indonesia*, vol. 7, pp. 115–121, September 2019, doi: 10.23886/ejki.7.10771.
- [5] L. Da Rocha Seixas, A. S. Gomes, and I. J. De Melo Filho, “*Effectiveness of Gamification in the Engagement of Students*,” *Computer in Human Behavior*, vol. 58, pp. 48–63, Mei 2016, doi: 10.1016/j.chb.2015.11.021.
- [6] H. Jusuf, “Penggunaan Gamifikasi dalam Proses Pembelajaran,” *Jurnal TICOM*, vol. 5, no. 1, pp. 1–6, September 2016.
- [7] C. McCarron, “*7 Splendid Reasons to Keep Frontend and Backend Separate*,” *ITChronicles*. Accessed: Maret 10, 2024. [Online]. Available: <https://itchronicles.com/software-development/7-splendid-reasons-to-keep-frontend-and-backend-separate/>
- [8] Nasution, “Implementasi Mongo DB, Express Js, React Js dan Node Js (Mern) pada Pengembangan Aplikasi Formulir, Kuis, dan Survei Online,” Universitas Islam Indonesia, Yogyakarta, 2022.
- [9] L. Mulana and K. Prihandani, “Analisis Perbandingan Kinerja Framework Codeigniter dengan Express.Js pada Server RESTful Api,” *Jurnal Ilmiah Wahana Pendidikan*, vol. 8, no. 16, pp. 316–326, September 2022, doi: 10.5281/ZENODO.7067707.

- [10] “Home | Scrum Guides,” *Scrum Guides*. Accessed: November 13, 2023. [Online]. Available: <https://scrumguides.org/>
- [11] M. Rifai, A. Wibawa, I. Zaeni, and D. Setiawan, “Implementasi *Scrum* pada Pengembangan Aplikasi Sistem ADI STETSA SMAN 4 Malang: Bidang Kesiswaan,” *Jurnal Inovasi Teknologi dan Edukasi Teknik*, vol. 1, no. 7, pp. 469–478, Juli 2021, doi: 10.17977/um068v1i72021p469-478.
- [12] I. Hanindya, D. Fachrudin, and O. Rahadiani, “Persepsi Tingkat Kesiapan Dokter Muda di Rotasi Klinik dalam Program Profesi Dokter di Fakultas Kedokteran Universitas Swadaya Gunung Jati,” *Tunas Medika Jurnal Kedokteran & Kesehatan*, vol. 8, no. 2, Desember 2022, Accessed: Mar. 03, 2024. [Online]. Available: <https://jurnal.ugj.ac.id/index.php/tumed/article/view/8026>
- [13] KEMENRISTEK-DIKTI, Peraturan Menteri Riset, Teknologi, dan Pendidikan Tinggi Republik Indonesia nomor 18 Tahun 2015 Tentang Tata Cara Pelaksanaan Uji Kompetensi Mahasiswa Program Profesi Dokter atau Dokter Gigi. Jakarta, 2015.
- [14] F. Marisa, T. M. Akhriza, A. L. Maukar, A. R. Wardhani, S. W. Iriananda, and M. Andarwati, “Gamifikasi (*Gamification*) Konsep dan Penerapan,” *JOINTECS (Journal of Information Technology and Computer Science)*, vol. 5, no. 3, p. 219, September 2020, doi: 10.31328/jointecs.v5i3.1490.
- [15] D. Ariani, “Gamifikasi untuk Pembelajaran,” *Jurnal Pembelajaran Inovasi*, vol. 3, no. 2, pp. 144–149, November 2020, doi: 10.21009/JPI.032.09.
- [16] Putra Prima Arhandi, “Pengembangan Sistem Informasi Perijinan Tenaga Kesehatan dengan Menggunakan Metode *Back End* dan *Front End*,” *Jurnal Teknologi Informasi: Teori, Konsep, dan Implementasi*, vol. 7, no. 1, 2016.
- [17] Donni Prabowo, “*Website E-commerce Menggunakan Model View Controller (Mvc)* dengan *Framework Codeigniter* Studi Kasus : Toko Miniatur,” *Data Manajemen dan Teknologi Informasi*, vol. 16, no. 1, pp. 23–29, 2015.
- [18] R. Pangestika, “Pengembangan *Back-end* Sistem Informasi Pendataan Sekolah Desa Komunitas Pendar Foundation Yogyakarta.,” Universitas Islam Indonesia, Yogyakarta, 2021.
- [19] Y. Sari and H. Riyansah, “Aplikasi *Tracking* Pedagang Keliling Dengan GPS Google Maps *API* Berbasis Android,” *Jurnal IKRAITH-INFORMATIKA*, vol. 5, no. 3, 2021.
- [20] D. Westerveld, *API Testing and Development with Postman: A practical guide to creating, testing, and managing APIs for automated software testing*. Packt Publishing Ltd, 2021.
- [21] P. Kukhnavets, “How to run *Scrum* efficiently in 2019? Quick guide for beginners,” *Habr*. Accessed: Nov. 13, 2023. [Online]. Available: <https://habr.com/en/companies/hygger/articles/455022/>

- [22] E. Hahn, *Express in action: writing, building, and testing Node.js applications*. Shelter Island, NY: Manning Publications, 2016.
- [23] “JSON: Introducing JSON.” Accessed: Juni 20, 2024. [Online]. Available: <https://www.json.org/json-id.html>
- [24] A. D. Praba, “Aplikasi Rekap Mengajar Berbasis Webiste dengan Database PostgreSQL,” *Indonesian Journal of Network & Security*, vol. 8, no. 1, 2018.
- [25] M. A. Rosid, “Implementasi JSON untuk Minimasi Penggunaan Jumlah Kolom Suatu Tabel pada Database PostgreSQL,” *JOINCS (Journal of Informatics, Network, and Computer Science)*, vol. 1, no. 1, p. 33, Juli 2017, doi: 10.21070/joincs.v1i1.802.
- [26] N. M. D. Febriyanti, A. A. K. O. Sudana, and I. N. Piarsa, “Implementasi *Black Box Testing* pada Sistem Informasi Manajemen Dosen,” *Jurnal Ilmiah Teknologi dan Komputer*, vol. 2, no. 3, pp. 535–544, Desember 2021.
- [27] J. Shadiq, A. Safei, and R. W. R. Loly, “Pengujian Aplikasi Peminjaman Kendaraan Operasional Kantor Menggunakan *BlackBox Testing*,” *INFORMATION MANAGEMENT FOR EDUCATORS AND PROFESSIONALS : Journal of Information Management*, vol. 5, no. 2, pp. 97–110, Juli 2021, doi: 10.51211/imbi.v5i2.1561.
- [28] W. G. Wardhana, I. Arwani, and B. Rahayudi, “Implementasi Teknologi Restful Web Service dalam Pengembangan Sistem Informasi Perekaman Prestasi Mahasiswa Berbasis Website (Studi Kasus: Fakultas Teknologi Pertanian Universitas Brawijaya),” *Jurnal Pengembangan Teknologi Informasi dan Ilmu Komputer*, vol. 4, no. 2, Art. no. 2, Mei 2020.
- [29] A. A. KARINA, “Analisis dan Perancangan *User Interface* dan *User Experience* Aplikasi *Mobile E-Learning* Berbasis Gamifikasi dengan Metode *Design Thinking* (Studi Kasus : Program Profesi Dokter Universitas Lampung),” Universitas Lampung, Bandar Lampung, 2023.
- [30] D. Ariani, D. S. Prawiradilaga, and W. Fatharani, “Inovasi Gamifikasi pada Pembelajaran Matakuliah Pengembangan Media Sederhana,” *Jurnal Pembelajaran Inovatif*, vol. 5, no. 1, pp. 41–48, Maret 2022, doi: 10.21009/JPI.051.05.
- [31] P. D. Mardika, Ahmad Fauzi, and Nilma, “Implementasi Metode *Scrum* pada Perancangan Sistem Informasi Tata Usaha Sekolah Berbasis Web,” *Jurnal Publikasi Teknik Informatika*, vol. 1, no. 1, pp. 53–60, November 2021, doi: 10.55606/jupti.v1i1.188.
- [32] I. Kurniawan, Humaira, and F. Rozi, “REST API Menggunakan NodeJS pada Aplikasi Transaksi Jasa Elektronik Berbasis Android,” *JITSI : Jurnal Ilmiah Teknologi Sistem Informasi*, vol. 1, no. 4, pp. 127–132, Desember 2020, doi: 10.30630/jitsi.1.4.18.

- [33] I. A. Kaniya Pradnya Paramitha, D. M. Wiharta, and I. M. Arsa Suyadnya, "Perancangan dan Implementasi Restful Api pada Sistem Informasi Manajemen Dosen Universitas Udayana," *Jurnal SPEKTRUM*, vol. 9, no. 3, p. 15, September 2022, doi: 10.24843/SPEKTRUM.2022.v09.i03.p3.
- [34] R. Fajrul Falah and M. Komarudin, "Perancangan Microservice Berbasis Rest Api pada Google Cloud Platform Menggunakan NodeJS dan Python," *Jurnal Informatika dan Teknik Elektro Terapan*, vol. 11, no. 3s1, September 2023, doi: 10.23960/jitet.v11i3s1.3506.