PENGEMBANGAN BOT TELEGRAM JALA TECH DENGAN PENERAPAN *STATE DESIGN PATTERN* DAN STRATEGI *CACHING* MENGGUNAKAN REDISLABS

Disusun Oleh:

N a m a : Mohammad Reza Ali Firdaus
NIM : 18523161

PROGRAM STUDI INFORMATIKA – PROGRAM SARJANA
FAKULTAS TEKNOLOGI INDUSTRI
UNIVERSITAS ISLAM INDONESIA
2022

HALAMAN PENGESAHAN DOSEN PEMBIMBING

PENGEMBANGAN BOT TELEGRAM JALA TECH DENGAN PENERAPAN STATE DESIGN PATTERN DAN STRATEGI CACHING MENGGUNAKAN REDISLABS

TUGAS AKHIR JALUR MAGANG

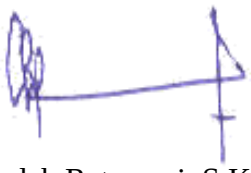
Disusun Oleh:

N a m a : Mohammad Reza Ali Firdaus

NIM : 18523161

Yogyakarta, 3 Agustus 2022

Pembimbing,



(Chanifah Indah Ratnasari, S.Kom., M.Kom.)

HALAMAN PENGESAHAN DOSEN PENGUJI

PENGEMBANGAN BOT TELEGRAM JALA TECH DENGAN PENERAPAN *STATE DESIGN PATTERN* DAN STRATEGI *CACHING* MENGGUNAKAN REDISLABS

TUGAS AKHIR JALUR MAGANG

Telah dipertahankan di depan sidang penguji sebagai salah satu syarat untuk memperoleh gelar Sarjana Komputer dari Program Studi Informatika – Program Sarjana di Fakultas Teknologi Industri Universitas Islam Indonesia

Yogyakarta, 3 Agustus 2022

Tim Penguji

Chanifah Indah Ratnasari, S.Kom., M.Kom. _____

Anggota 1

Erika Ramadhani, S.T., M.Eng. _____

Anggota 2

Arrie Kurniawardhani, S.Si., M.Kom. _____

Mengetahui,

Ketua Program Studi Informatika – Program Sarjana

Fakultas Teknologi Industri

Universitas Islam Indonesia

(Dr. Raden Teduh Dirgahayu, S.T., M.Sc.)

HALAMAN PERNYATAAN KEASLIAN TUGAS AKHIR

Yang bertanda tangan di bawah ini:

Nama : Mohammad Reza Ali Firdaus

NIM : 18523161

Tugas akhir dengan judul:

PENGEMBANGAN BOT TELEGRAM JALA TECH DENGAN PENERAPAN STATE DESIGN PATTERN DAN STRATEGI CACHING MENGGUNAKAN REDISLABS

Menyatakan bahwa seluruh komponen dan isi dalam tugas akhir ini adalah hasil karya saya sendiri. Apabila di kemudian hari terbukti ada beberapa bagian dari karya ini adalah bukan hasil karya sendiri, tugas akhir yang diajukan sebagai hasil karya sendiri ini siap ditarik kembali dan siap menanggung risiko dan konsekuensi apapun.

Demikian surat pernyataan ini dibuat, semoga dapat dipergunakan sebagaimana mestinya.

Yogyakarta, 14 Juli 2022

(Mohammad Reza Ali Firdaus)

HALAMAN PERSEMBAHAN

Halaman ini berisi kalimat-kalimat persembahan serta rasa syukur atas pemberian Allah SWT yang telah memberikan saya kekuatan serta kesempatan untuk menyelesaikan laporan akhir. Laporan ini dipersembahkan sebagai hasil akhir dari perkuliahan yang saya jalani kepada orang tua, keluarga, sahabat, dan teman-teman saya. Saya sangat berterima kasih kepada pihak-pihak yang disebutkan, karena telah senantiasa mendukung penulis untuk menjalani hidup dan akhirnya bisa sampai ke titik ini. Tak lupa juga rasa terima kasih kepada orang tua saya, yang senantiasa memberikan jasa dan pengorbanan supaya saya bisa menjadi manusia yang terdidik dan bermanfaat dalam kehidupan bermasyarakat. Kemudian rasa terima kasih juga saya sampaikan kepada seluruh elemen di Informatika UII, yang telah memberikan saya pengetahuan serta pengalaman unik semasa perkuliahan.

Saya mohon maaf sebesar-besarnya kepada pihak-pihak yang telah disebutkan, atas kesalahan-kesalahan yang telah saya perbuat di masa lampau. Semoga kita semua dapat menjadi pribadi yang lebih baik lagi kedepannya, Aamiin.

HALAMAN MOTO

"*The result is irrelevant, because the effort was there*" – xqc

"Allah SWT akan selalu memudahkan segala urusan hambanya di dunia, ketika hamba tersebut lebih memprioritaskan urusan agama dibandingkan dunia" – Orang tua saya

KATA PENGANTAR

Assalamu'alaikum Warahmatullahi Wabarakatuh.

Pertama, Alhamdulillah puji dan syukur kepada Allah SWT yang telah memberikan penulis kesempatan dan kekuatan untuk menyelesaikan laporan akhir berjudul "Pengembangan Bot Telegram Jala Tech Dengan Penerapan *State Design Pattern* Dan Strategi *Caching* Menggunakan Redislabs". Laporan akhir disusun untuk menyampaikan bukti pelaksanaan magang di Jala Tech dan sekaligus menjadi persyaratan penulis untuk bisa lulus dan memperoleh gelar sarjana pada Program Studi Informatika di Universitas Islam Indonesia.

Selama melaksanakan magang hingga momen-momen terakhir saat menyelesaikan laporan akhir ini, banyak sekali hambatan dan rintangan yang penulis hadapi. Akan tetapi, atas izin Allah SWT akhirnya penulis dapat menyelesaikan keduanya dengan baik. Tidak terlepas juga dukungan moral maupun materi dari orang-orang terdekat penulis, maka dari itu izinkan penulis untuk berterima kasih kepada:

1. Bapak Hendrik, S.T., M.Eng., selaku kepala Jurusan Informatika Fakultas Teknologi Industri Universitas Islam Indonesia, Dr. Raden Teduh Dirgahayu, S.T., M.Sc., selaku kepala Program Studi Informatika - Program Sarjana Fakultas Teknologi Industri Universitas Islam Indonesia.
2. Ibu Chanifah Indah Ratnasari, S.Kom., M.Kom, selaku dosen pembimbing yang senantiasa selalu sabar dan ikhlas meluangkan waktu serta tenaganya dalam memberikan bimbingan dan arahan kepada saya untuk menyelesaikan laporan akhir.
3. Kedua orang tua saya yang tak henti-hentinya memberikan banyak pengorbanan, jasa, serta dukungan baik dalam segi materi maupun non materi.
4. Kedua saudara saya yang selama ini menemani saya di rumah.
5. Segenap keluarga Informatika Universitas Islam Indonesia yang telah menjadi wadah untuk mengais ilmu pengetahuan serta memberikan saya aneka ragam pengalaman, kesan dan pesan yang unik semasa perkuliahan.
6. Mas Farid Inawan, Mas Syauqy Nurul Aziz, Mbak Liris Maduningtyas, Mas Jihad Maulana, Mas Wildan Pratama, Mbak Anastasia Arsanti, dan karyawan Jala Tech lainnya yang telah memberikan saya kesempatan, pengetahuan, dan semangat untuk melaksanakan magang di Jala.
7. Albarra Naufala dan Muhammad Farhan, selaku sahabat serta teman satu kontrakan. Terima kasih karena telah memberikan kesan dan pengalaman yang unik pada kehidupan saya.

8. Teman-teman dan pihak lainnya yang selalu mendukung saya bagaimanapun situasinya.

Dengan bantuan dari pihak-pihak yang telah disebutkan, penulis dapat menyelesaikan masa magang dan laporan akhir dengan baik. Tentunya penulis menyadari, bahwa pada laporan ini masih jauh dari kata sempurna sehingga penulis akan menerima semua masukan berupa kritik dan saran untuk dijadikan pembelajaran sebagai penulis kedepannya. Mohon maaf sebesar-besarnya apabila terdapat kesalahan kata dalam laporan ini, semoga laporan akhir ini juga dapat memberikan manfaat kepada pembacanya.

Wassalamu'alaikum Warahmatullahi Wabarakatuh

Yogyakarta, 14 Juli 2022

(Mohammad Reza Ali Firdaus)

SARI

Jala Tech merupakan salah satu perusahaan penyedia layanan budi daya udang digital dengan bantuan teknologi, layanan yang ditawarkan seperti kemudahan dalam manajemen tambak udang, pengelolaan data kolam udang yang dapat direpresentasikan menggunakan grafik, serta sistem pendukung keputusan untuk pengguna. Pada saat penulis melaksanakan magang di Jala Tech, terdapat suatu kasus di mana *field assistance* pada *smart farm* milik Jala memiliki kendala terkait pengelolaan kolam budi daya udang serta pengujian fitur proses pengolahan data kolam pada aplikasi *mobile* Jala karena terbatasnya koneksi internet dan *smartphone* yang digunakan. Hal ini menyebabkan proses pengujian produk dan pengelolaan *smart farm* Jala menjadi terhambat, selain itu juga membuat aplikasi *mobile* Jala harus dioptimalkan kembali supaya dapat digunakan pada *smartphone* dengan versi *android* apa pun. Atas permasalahan tersebut, diperlukan solusi alternatif yang dapat menunjang proses pengolahan data kolam yang dapat digunakan pada *smartphone* dengan segala versi *android* dan mampu digunakan dengan koneksi internet yang tidak terlalu besar. Hasil dari permasalahan tersebut adalah pengembangan bot Telegram yang memuat fitur inti pada aplikasi Jala seperti pencatatan data kolam serta pemeriksaan hasil data pencatatan, dengan menerapkan *state design pattern* yang berguna untuk mengelola fungsi dan respon bot sesuai dari aksi pengguna dan *caching* memanfaatkan Redislabs yang berguna untuk menyimpan data-data yang bersifat sementara dari pengguna. Dengan demikian, *field assistance* pada *smart farm* Jala dapat mengelola dan memproses data kolam udang melalui bot Telegram yang mudah diakses pada *smartphone* atau komputer pribadi serta tanpa perlu mengkhawatirkan koneksi internet yang besar.

Kata kunci: *Chatbot*, *Caching*, Redislabs, *Scrum*, *State Design Pattern*, Telegram.

GLOSARIUM

Anco	Wadah pakan pada kolam udang yang berguna untuk mengukur nafsu makan udang.
API	Media penengah pada aplikasi utama yang memungkinkan komunikasi antara dua aplikasi.
Array	Himpunan dari data-data pada pemrograman.
Backlog	Daftar yang memuat tugas atau spesifikasi kebutuhan sistem.
Caching	Penyimpanan data yang bersifat sementara dalam sebuah memori komputer.
Callback	Pemanggilan suatu fungsi program di dalam fungsi program lainnya.
Chatbot	Akun otomatis yang dapat menerima masukan dari pengguna dan mengembalikan respon dengan baik dan sesuai.
Context	Nama pada suatu konteks keadaan.
Debugging	Identifikasi *error* dan perbaikan pada pemrograman.
Design Pattern	Pola desain pada penulisan kode program.
EasyRetro	Aplikasi untuk menunjang proses evaluasi pada pengerjaan tugas.
Invoke	Pemanggilan suatu fungsi.
Request	Bentuk permintaan berbasis protokol HTTP pada suatu API aplikasi dan mengembalikan respon tertentu.

DAFTAR ISI

DAFTAR TABEL

DAFTAR GAMBAR

BAB I
PENDAHULUAN

1.1 Latar Belakang

Jala Tech merupakan salah satu perusahaan yang bergerak pada bidang akuakultur khususnya pada sektor budi daya udang. Jala Tech berkomitmen untuk membantu para petambak udang dalam meningkatkan produktivitas dan menciptakan budi daya udang yang berkelanjutan dengan bantuan teknologi. Jala Tech menciptakan produk perangkat keras bernama Jala Baruno yang berguna untuk mengukur parameter tertentu pada kolam udang, selain itu Jala Tech menciptakan produk digital berupa aplikasi berbasis *web* dan *mobile* yang berguna untuk memantau dan mengolah data tambak maupun kolam udang secara daring (Jala Tech, 2022).

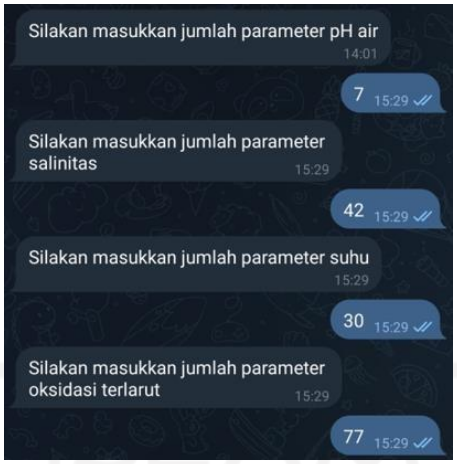
Penulis melaksanakan magang di Jala Tech selama enam bulan terhitung dari bulan September dan berakhir pada bulan Maret sebagai *front end developer*. Umumnya, *front end developer* memiliki pekerjaan yang bertanggung jawab atas pengembangan pada tampilan aplikasi serta distribusi data yang didapatkan dari *back end*. Penulis lebih banyak berkontribusi pada proyek dalam pengembangan aplikasi *web* Jala. Gambaran proyek-proyek yang dikerjakan seperti membuat *file* yang memuat *preset* bahasa asing, menerapkan tampilan baru pada halaman *dashboard* utama, menerapkan fungsi mata uang pada halaman fitur finansial dan simulasi, merombak halaman pada fitur *input excel*, memperbaiki *bug* dan mengerjakan tugas pada *backlog*. Setiap proyek yang dikerjakan dijalankan dengan metode pengembangan *scrum* yang terkenal sebagai metode pengembangan yang cepat dan adaptif (Grebić & Stojanović, 2021). Semua proses pengembangan dalam *Scrum* dilakukan dalam satu *sprint*, waktu *sprint* yang ditentukan oleh tim *software* Jala tech adalah sebanyak dua minggu.

Selain mendapatkan proyek yang berkaitan dengan aplikasi *web* Jala, penulis juga mendapatkan sebuah proyek di luar pengembangan aplikasi *web* Jala yakni pengembangan Bot Telegram. Alasan proyek ini dikembangkan karena adanya suatu kasus di mana *field assistance* pada *smart farm* milik Jala Tech mendapatkan kendala terkait pengelolaan kolam serta pengujian fitur pengolahan data kolam pada aplikasi *mobile* Jala karena keterbatasan koneksi internet serta gawai yang digunakan, dan juga kurang optimalnya aplikasi *mobile* Jala. Jala Tech memiliki *smart farm* yang berguna sebagai subjek pengujian produk-produk Jala sebelum

disampaikan kepada pengguna atau konsumen utama, kemudian *smart farm* Jala dipantau dan dikelola oleh *field assistance*.

Berdasarkan permasalahan pada paragraf di atas, proyek pengembangan Bot Telegram diangkat sebagai topik skripsi karena menurut penulis proyek pengembangan Bot Telegram sangat unik dibandingkan dengan pengembangan aplikasi *web*. Selain itu, Bot Telegram Jala Tech menjadi salah satu proyek yang di dalamnya terdapat kontribusi penuh dari penulis sekaligus menjadi karya penulis pada saat melaksanakan magang di Jala Tech. Setelah menyelesaikan proyek ini, penulis telah memahami bahwa Bot Telegram dapat dijadikan sebagai sistem informasi berskala kecil yang dapat memuat fitur-fitur inti dari aplikasi ketiga yang hendak diintegrasikan. Telegram dipilih karena aplikasi Telegram sangat populer dengan perkembangan layanan *chatbot*nya, dan juga Telegram dapat digunakan pada *smartphone* dengan segala macam versi *android* serta tidak menggunakan koneksi internet yang besar. Faktanya, pengembangan Bot Telegram telah diimplementasikan dalam berbagai bidang seperti bidang *Internet of Things* (IoT), rekam medis, layanan sosial, dan layanan lainnya yang dapat dilakukan secara virtual (Nobari et al., 2017).

Hal menarik lainnya bagi penulis adalah pengembangan Bot Telegram Jala menerapkan *state design pattern* sesuai pada pemrograman *JavaScript*, selain itu Bot Telegram Jala juga menerapkan *caching* data dengan memanfaatkan Redislabs. Diperlukannya *caching* data pada Bot Telegram Jala yakni untuk meyimpan data-data yang berkaitan dengan pengguna seperti konteks fitur yang diakses, *state* pada setiap konteks fitur, status akun Jala terhubung, dan penyimpanan data masukan pengguna. Bot Telegram Jala dibangun dengan model bot percakapan interaktif, di mana interaksi antara bot dengan pengguna dibuat secara tanya-jawab seperti pada Gambar 1.1.



Gambar 1.1 Model percakapan interaktif Bot Telegram Jala

1.2 Ruang Lingkup

Dalam proyek pengembangan Bot Telegram Jala, diterapkan metode pengembangan *scrum* yang bersifat cepat dan adaptif. Terdapat empat orang termasuk penulis yang berkontribusi pada proyek ini, masing-masing peran dari empat tersebut adalah satu *product manager*, satu *product designer*, satu *quality assurance*, dan penulis sebagai *developer*. Proyek ini menghabiskan waktu sebanyak dua *sprint* (empat minggu) mulai dari bulan Oktober hingga bulan November. Kontribusi penulis pada proyek ini dapat disimak sebagai berikut:

a. Mengimplementasikan fitur integrasi akun Jala dengan akun Telegram pengguna
b. Mengimplementasikan fitur pencatatan data kolam pada budi daya udang
c. Mengimplementasikan fitur pemeriksaan data hasil pencatatan pada budi daya udang
d. Mengimplementasikan fitur profil pada Bot Telegram
e. Mengimplementasikan fitur daftar tambak dan kolam pada Bot Telegram

1.3 Tujuan

Tujuan dari proyek pengembangan Bot Telegram adalah untuk menjadikan Bot Telegram sebagai solusi alternatif bagi para petambak udang dalam menunjang proses pengolahan data kolam udang pada *smart farm* Jala Tech. Sehingga Bot Telegram diharapkan dapat menjadi produk digital baru dari Jala Tech dan menjadi solusi yang selalu tersedia bagi para petambak udang yang memiliki kendala terkait keterbatasan pada saat menggunakan aplikasi Jala.

1.4 Manfaat

Manfaat dari pengembangan bot Telegram dengan menerapkan konsep *state and management design pattern* adalah sebagai berikut:

a. Memudahkan Bot Telegram dalam memproses setiap masukan pengguna berdasarkan *state* dan *context* yang dimiliki pengguna.

b. Menghapus kemungkinan dari respon Bot Telegram yang tidak sesuai dengan masukan dari pengguna. Karena setiap pengguna memiliki data *state* dan data konteks fitur yang diakses yang bersifat sementara, respon Bot Telegram dapat diolah sesuai *state* dan konteks fitur yang diakses pengguna sehingga hal ini menjadi *validator input* supaya setiap aksi pengguna dapat berjalan sesuai prosedur fitur yang diakses.

c. Memudahkan pengembang yang akan berkontribusi pada proyek ini ke depannya dengan mengklasifikasi setiap fitur dengan masing-masing perintah (*bot commands*).

d. Memudahkan pengguna dalam melakukan pencatatan data kolam serta pemeriksaan data hasil pencatatan.

e. Memahami penggunaan teknologi Redislabs sebagai *database cache*.

1.5 Sistematika Penulisan

Sistematika penulisan yang digunakan dalam penyusunan laporan akhir magang oleh penulis adalah sebagai berikut:

a. BAB I – PENDAHULUAN

   Pada bab ini, diuraikan gambaran umum perusahaan Jala Tech tempat penulis melaksanakan magang. Kemudian membahas garis besar kontribusi penulis saat melaksanakan magang serta proyek yang diangkat untuk dijadikan topik bahasan pada laporan akhir magang ini. Selain itu, bab ini juga memuat ruang lingkup magang, tujuan, manfaat, dan sistematika penulisan pada laporan.

b. BAB II – LANDASAN TEORI DAN TINJAUAN PUSTAKA

   Bab ini membahas teori-teori dasar yang mendukung topik utama yang diangkat pada laporan akhir ini. Teori-teori tersebut adalah pembahasan Telegram *chatbot*, *scrum*, *cache*, Redislabs, dan *state design pattern*.

c. BAB III – PELAKSANAAN MAGANG

   Pada bab ini, diuraikan pelaksanaan magang yang dilakukan penulis di Jala Tech. Bahasan terkait pelaksanaan magang seperti rutinitas yang dilakukan dalam sebuah *sprint* pada metode pengembangan *scrum* yang diterapkan, dan pembahasan aktivitas yang dilakukan

pada saat mengembangkan Bot Telegram dengan menerapkan *state design pattern* dan strategi *caching* menggunakan Redislabs.

d. BAB IV – REFLEKSI PELAKSANAAN MAGANG

   Bab ini membahas terkait refleksi terkait pembelajaran akademik yang didapatkan penulis semasa kuliah dengan aplikasi teori yang diterapkan pada saat pelaksanaan magang penulis di Jala Tech.

e. BAB V – PENUTUP

   Bab ini berisi kesimpulan dari laporan akhir magang penulis yang membahas pengembangan bot Telegram Jala Tech dengan menerapkan konsep *state and context management* dengan *state design pattern*. Adapun saran yang diberikan penulis mengenai perbaikan dan peningkatan yang dapat diterapkan kedepannya pada bot Telegram Jala Tech.

f. DAFTAR PUSTAKA

g. LAMPIRAN

BAB II
LANDASAN TEORI DAN TINJAUAN PUSTAKA

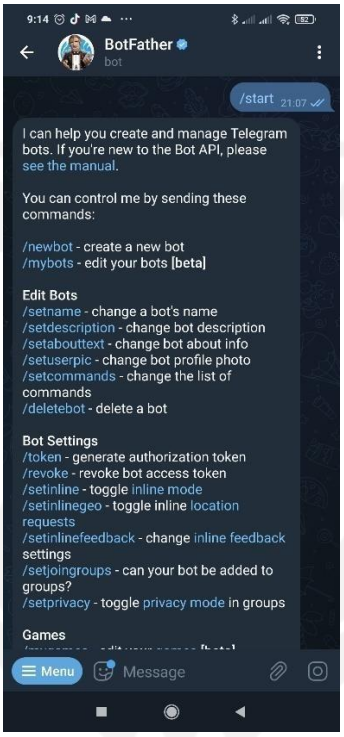
2.1 Telegram *Chatbot*

Telegram *chatbot* merupakan akun otomatis yang tersedia pada Telegram yang dapat merespon pengguna sesuai pesan yang diberikan (Sutikno et al., 2016), serta memungkinkan para pengembang untuk mengintegrasikan suatu layanan pada aplikasi pihak ketiga ke dalam *chatbot*. Sesuai dokumentasi Telegram (2020), bot dapat memuat banyak jenis layanan yang dapat diintegrasikan. Sebagai contoh, bot dapat dimanfaatkan menjadi sebuah portal informasi, berintegrasi dengan layanan pihak ketiga, menjadi *platform* pembayaran, membuat permainan, dan masih banyak lagi layanan yang dimuat pada sebuah bot selama layanan tersebut dapat dilakukan secara virtual.

Terdapat dua pola interaksi yang umumnya diketahui oleh pengembang Bot Telegram, yakni pola interaksi *command-based* dan *event-based*. Pola interaksi *command-based* merupakan suatu pola interaksi di mana Bot Telegram mengolah pesan dan memberikan respon kepada pengguna berdasarkan orientasi perintah berupa pesan yang diberikan. Sedangkan pola interaksi *event-based* merupakan suatu pola interaksi di mana Bot Telegram mengolah pesan dan memberikan respon kepada pengguna berdasarkan orientasi pesan yang spesifik atau *callback data* yang memanfaatkan *interactive action* pada fitur *inline keyboard button*. Fitur *inline keyboard* ini telah disediakan oleh Telegram supaya para pengembang bot dapat menggunakan variasi respon bot yang interaktif. Kemudian terdapat dua metode komunikasi yang dapat diterapkan pada Bot Telegram, yaitu dengan *long polling* dan *webhook*. *Long polling* merupakan metode komunikasi di mana bot harus melakukan *request* kepada server Telegram untuk mendapatkan pesan terbaru dari pengguna. Sedangkan *webhook* merupakan metode komunikasi di mana bot mendapatkan *update* pesan terbaru dari pengguna pada *url* yang telah ditetapkan oleh pengguna, dengan begitu bot tidak perlu melakukan *request* kepada server Telegram karena Telegram akan meneruskan pesan yang dikirim dari pengguna kepada *url* tersebut yang berbentuk *request*. Sehingga metode *webhook* sering digunakan karena bot dapat merespon *request* serta pesan dari pengguna dengan cepat (Setiaji & Paputungan, 2018).

Telegram menyediakan bot induk untuk para pengembang dalam mendaftarkan botnya supaya dapat diakses secara publik, bot induk tersebut bernama BotFather. Sesuai namanya, BotFather dapat mengelola pengaturan setiap bot sesuai kehendak pengembang. Seperti

mengatur daftar perintah, mengganti nama, membuat bot baru, mengelola bot yang dimiliki pengguna, dan masih banyak lagi pengaturan lainnya. Tampilan akun BotFather dapat dilihat pada Gambar 2.1.



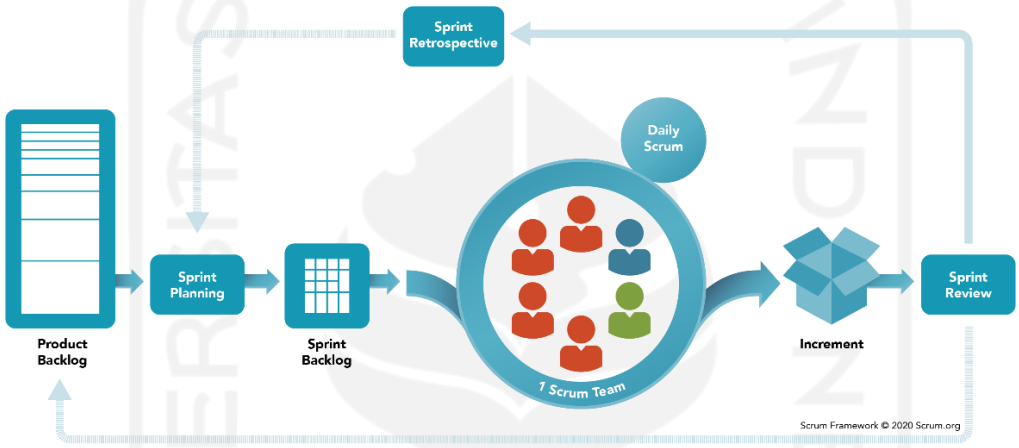
Gambar 2.1 Akun BotFather pada Telegram

2.2 Scrum

Scrum merupakan salah satu *framework* pengembangan modern yang bersifat cepat dan adaptif terhadap perubahan spesifikasi sistem. Perlunya menerapkan salah satu metode pengembangan adalah agar proses pengembangan produk dapat berjalan dengan terstruktur, sehingga dengan diterapkannya suatu metode pengembangan akan menghasilkan produk dengan *output* yang maksimal. *Scrum* ditemukan oleh Jeff Sutherland dan Ken Schwaber pada tahun 1993, filosofi *scrum* adalah pengembangannya yang mampu menunjang suatu aktivitas yang padat, tugas-tugas yang kompleks dan produk yang adaptif. *Scrum* dipercaya dapat melatih orang-orang yang terlibat di dalamnya untuk meningkatkan efektivitas pada kolaborasi, melatih kreativitas dalam menyusun strategi untuk mencapai suatu target, dan meningkatkan produktivitas setiap anggota tim (Bhavsar et al., 2020).

Scrum memiliki sebuah *sprint*, yakni sebuah entitas yang memuat segala proses dalam pengembangan suatu produk. Proses-proses yang ada di dalam *scrum* yaitu *sprint planning*,

sprint, *sprint review*, *sprint retrospective*, dan *daily scrum*. *Sprint* idealnya diterapkan selama satu sampai empat minggu dan tidak lebih atau kurang daripada itu (Grebić & Stojanović, 2021). *Scrum* juga memiliki elemen-elemen penting seperti *product backlog* dan *sprint backlog*. Kemudian, setiap peran yang ada saat menerapkan *scrum* harus ada setidaknya satu orang profesional supaya orang tersebut dapat memimpin peran yang mampu menunjang anggota timnya. Berikut ini adalah ilustrasi pengembangan *scrum* pada umumnya yang dapat disimak pada Gambar 2.2.



Gambar 2.2 Ilustrasi alur pengembangan dengan *framework scrum*

Sumber: (Scrum.org, 2020)

2.3 Cache

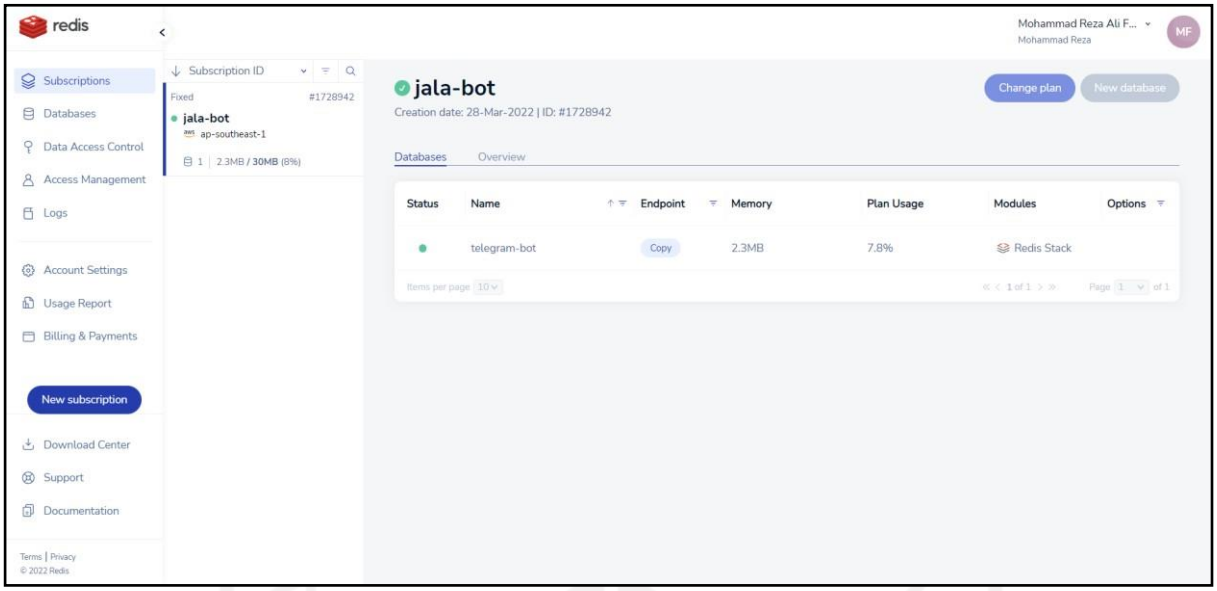
Cache merupakan data yang disimpan bersifat sementara dengan tujuan untuk mengurangi beban yang berpengaruh pada performa suatu sistem aplikasi. *Cache* diperlukan apabila terdapat data-data yang harus diolah secara cepat dan sering diminta oleh sistem aplikasi. Menurut (Alam et al., 2017), terdapat faktor-faktor yang menjadi alasan dasar penerapan *caching* dalam sebuah aplikasi, antara lain:

a. Untuk meningkatkan performa aplikasi.
b. Mengurangi beban *load* pada aplikasi.
c. Mengurangi beban *load* pada *web service* dan *database* yang digunakan.
d. Mendapatkan data yang bersifat *read-only* dengan lebih cepat

2.4 Redislabs

Redis merupakan *NoSQL database* bersifat *open-source* yang umumnya digunakan untuk menyimpan *cache* dan *session* pengguna aplikasi web secara *real-time* dengan model *key-value storing* (Syaefulloh & Yusrizal, 2019). Redis menggunakan memori pengguna (RAM) yang dapat diatur kapasitasnya dalam melakukan penyimpanan data, sehingga permintaan data dapat diproses dengan cepat (Redis, 2022). Redis dapat memberikan respon dengan waktu yang sangat singkat sehingga dapat menampung jutaan permintaan yang diminta dalam satu waktu, oleh karena itu Redis menjadi pilihan populer dalam penerapan *caching*. Redis mendukung berbagai macam tipe data seperti *string*, *list, set*, dan *JSON*. *JSON* memudahkan pengembang dalam mengolah data ke dalam bentuk objek dan *string* dengan mudah, sehingga pada pengembangan Bot Telegram Jala *JSON* digunakan untuk mengolah data pengguna dengan penerapan *state design pattern* pada pemrograman *JavaScript*.

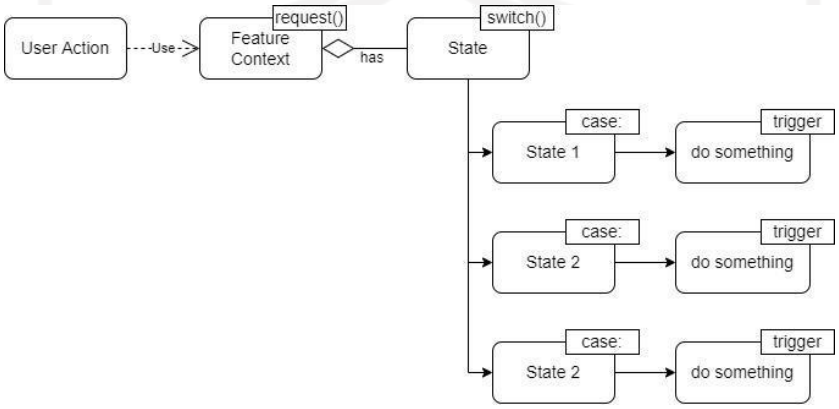
Pada pengembangan Bot Telegram Jala, penulis memanfaatkan Redislabs yang merupakan salah satu layanan berbasis *cloud* dari Redis. Redislabs adalah aplikasi layanan *database as a service* (DBaaS) berbasis *cloud* dengan performa yang setara dengan Redis yang menerapkan penyimpanan pada memori pengguna (Redis, 2022), sehingga pada pengembangan ini Redislabs sangat cocok untuk digunakan karena Bot Telegram Jala perlu mengelola *context* dan *state* dari setiap fitur yang diakses sesuai masukan pengguna. Dengan menggunakan Redislabs, pengembang tidak perlu repot-repot dalam mengatur *database* karena Redislabs menggunakan layanan penyimpanan pada AWS Amazon sehingga segala pengaturan telah otomatis diatur oleh AWS (AWS Amazon, 2022). Tampilan dari aplikasi Redislabs dapat dilihat pada Gambar 2.3.



Gambar 2.3 Tampilan aplikasi Redislabs

2.5 State Design Pattern

State design pattern merupakan salah satu konsep pola desain *behavioral* yang mengatur logika sistem untuk menghasilkan luaran yang tepat berdasarkan konteks fitur atau prosedur yang diakses oleh pengguna (Fodor Paul, 2020). Dengan bahasa pemrograman *JavaScript*, penerapan *state design pattern* pada pengembangan Bot Telegram Jala mengatur *context* dan *state* yang dimiliki pengguna dengan salah satu pendekatan *switch* dan *if else* tergantung dari sedikit atau banyaknya *state* pada suatu konteks fitur. Pendekatan *state design pattern* dengan *JavaScript* dapat disimak pada Gambar 2.4.



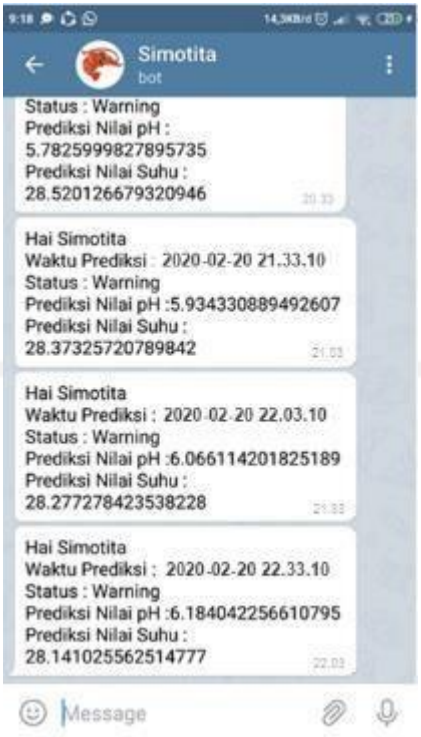
Gambar 2.4 Pendekatan *state design pattern*

2.6 Tinjauan Pustaka

Dalam penelitian sesuai dengan topik yang diangkat pada laporan akhir ini, penulis melakukan pencarian informasi terhadap penelitian yang sejenis terkait pengembangan Bot Telegram. Penulis menggali informasi dari beberapa artikel jurnal dengan topik sejenis untuk mendapatkan informasi-informasi yang nantinya dapat dijadikan perbandingan dengan penelitian penulis.

Pengembangan Bot Telegram sudah pernah diterapkan dalam beberapa aspek domain seperti *Internet of Things* (IoT), penunjang akademik, informasi layanan sosial, dan masih banyak lagi. Penulis mengambil informasi dari empat artikel jurnal nasional yang tersedia, berikut ini adalah referensi penelitian sejenis yang diambil oleh penulis:

a. Wisnu Alfiansyah et al., 2021, mengembangkan sebuah Bot Telegram yang diintegrasikan dengan perangkat keras berbasis IoT sebagai layanan *chatbot* untuk *push notification*. Pada penelitian ini, perangkat IoT memiliki peranan sebagai alat pengukur data kolam udang seperti data pH dan suhu air. Kemudian setelah perangkat tersebut berhasil mengukur data kolam, data tersebut akan dikirimkan ke dua *platform* yaitu *website monitoring* buatan peneliti terkait dan Bot Telegram. Bot Telegram pada penelitian ini berfungsi sebagai *push notification* kepada penggunanya terkait data yang berhasil diukur dengan perangkat IoT, sedangkan *website monitoring* berperan sebagai proses inti dalam melakukan pemantauan dan pengolahan data kolam. Data yang dikirimkan pada Bot Telegram diolah dengan metode *Double Exponential Smoothing* (DES) dengan model Holt, data tersebut diolah untuk dijadikan sebagai prediksi atau *forecasting* nilai kelayakan pH dan suhu air pada kolam udang. Menurut penulis, penelitian ini kurang mengemukakan sisi pengembangan Bot Telegram sehingga aspek utama pada penelitian ini adalah perhitungan pada perangkat IoT yang menggunakan metode DES dengan model Holt pada pengukuran kolam udang. Hasil penelitian tersebut dapat dilihat pada Gambar 2.5.



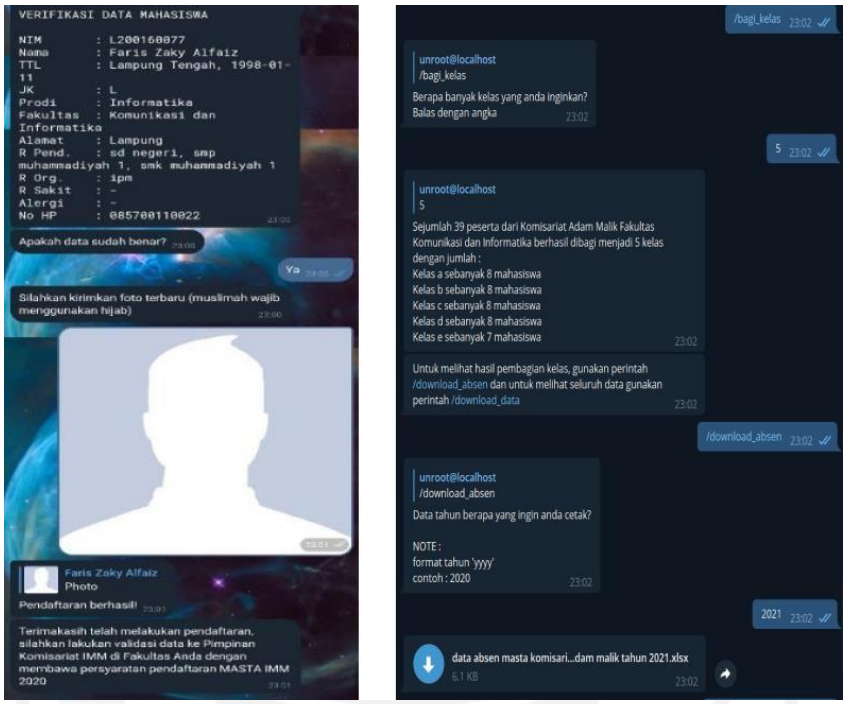
Gambar 2.5 Tampilan Bot Telegram sebagai *push notification* dari EWS

b. Gde Sastrawangsa, 2017, mengembangkan sebuah Bot Telegram yang bertujuan untuk menunjang layanan akademik sebagai salah satu penerapan *smart campus*. Bot Telegram yang dikembangkan pada penelitian tersebut memuat fitur-fitur inti seperti layanan KRS, informasi jadwal perkuliahan mahasiswa, presensi daring, informasi transkrip nilai matakuliah, layanan pendaftaran sidang, dan masih banyak lagi layaknya layanan akademik perkuliahan pada umumnya. Bot Telegram juga dikembangkan dengan konsep *command-based* yang berarti pengguna harus memasukkan perintah secara manual, namun hampir semua fitur di dalamnya menerapkan percakapan interaktif yakni Bot Telegram dibuat dengan model tanya-jawab kepada pengguna. Kumpulan teknologi yang digunakan pada pengembangan Bot Telegram *smart campus* adalah menggunakan bahasa pemrograman PHP sebagai bahasa pokok pemrogramannya, MariaDB sebagai *database* pengguna, dan *webhook* digunakan sebagai komunikasi antara bot dengan *server* yang memuat API *request* dari *database*. Setelah Bot Telegram diimplementasi, pengujian dilakukan dengan cara manual percobaan dari setiap daftar perintah yang tersedia sehingga pengujian difokuskan untuk menguji ambiguitas prosedur dari setiap fitur perintah. Hasil perancangan Bot Telegram tersebut dapat dilihat pada Gambar 2.6.

Gambar 2.6 Hasil rancangan Bot Telegram untuk penerapan *smart campus*

c. Alfaiz & Maryam, 2021, mengembangkan sebuah Bot Telegram yang bertujuan sebagai *platform* untuk pendaftaran masa orientasi mahasiswa guna untuk mengenalkan organisasi kampus serta elemen-elemen perkuliahan lainnya pada Universitas Muhammadiyah Surakarta. Bot Telegram pada penelitian tersebut perlu dikembangkan untuk menunjang segala proses pendaftaran seperti validasi, pengolahan data peserta, pengelolaan absensi dan daftar peserta, dan mengelola kepanitiaan. Adapun metode pengembangan yang digunakan pada pengembangan Bot Telegram adalah dengan menggunakan metode *waterfall*. Pada penelitian tersebut *waterfall* digunakan untuk mempermudah proses perbaikan yang berfokus pada tahap pengujian dan evaluasi, sehingga peneliti tersebut dapat berfokus pada analisis kebutuhan beserta implementasinya. Sama seperti penelitian yang telah disebut sebelumnya, Bot Telegram pada penelitian ini dibuat dengan konsep *command-based* dan juga memanfaatkan fitur *keyboard button* sehingga pengguna tidak perlu mengetik secara manual. Akan tetapi, penulis tidak melihat adanya pola interaksi antara bot dengan pengguna secara interaktif. Setelah Bot Telegram diimplementasi, pengujian dilakukan dengan metode pengujian *blackbox* yang berguna untuk menguji fungsionalitas sistem. Selain itu, survey kepada pengguna juga dilakukan dengan menerapkan jenis survey *System Usability Scale* (SUS) untuk mengevaluasi kegunaan Bot Telegram dengan kuesioner. Bot Telegram pada penelitian ini dikembangkan dengan bahasa pemrograman *python* dengan beberapa *library*-nya, dan MySQL sebagai *database* utama. Bot Telegram pada penelitian ini juga menerapkan *webhook* sebagai komunikasi antara bot dengan *server database* peneliti terkait, sehingga bot dapat langsung

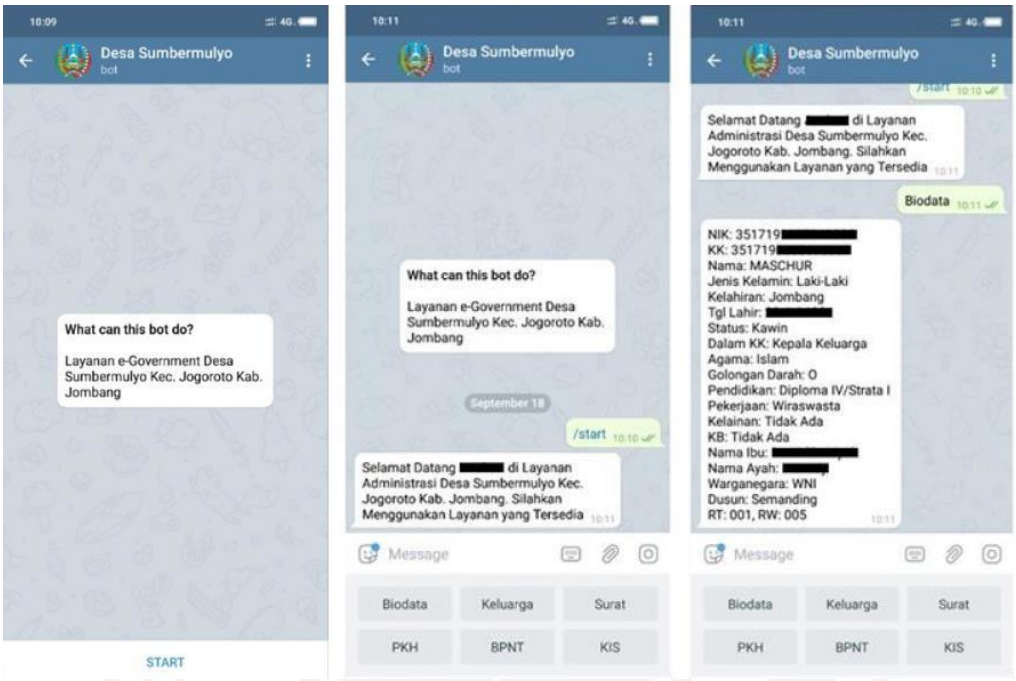
mengirimkan respon kepada pengguna tanpa perlu menunggu *update* dari *server* Telegram. Hasil penelitian dari Bot Telegram yang dikembangkan dapat dilihat pada Gambar 2.7.



Gambar 2.7 Tampilan Bot Telegram untuk layanan orientasi mahasiswa

d. Anshori et al., 2020,mengembangkan sebuah Bot Telegram untuk menerapkan layanan *E-Government* dengan Bot Telegram sebagai *platform*-nya. Pada penelitian tersebut, diperlukannya Bot Telegram sebagai *platform* layanan *E-Government* karena aplikasi Telegram populer serta cara penggunaannya yang mudah dibandingkan dengan aplikasi *web* dan *mobile*. Selain itu, dengan menggunakan Bot Telegram maka layanan *E-Government* dapat diakses dengan mudah di mana saja dan kapan saja tanpa menyita waktu produktif pada warga. Bot Telegram dibangun dengan model interaktif, di mana interaksi bot dengan pengguna di buat dengan menggunakan *keyboard button*. Saat pengguna menekan *keyboard button* tersebut, secara otomatis pengguna akan mengirimkan pesan kepada bot sesuai kata pada *keyboard button* tersebut. Bot Telegram pada penelitian tersebut juga memanfaatkan *webhook* sebagai metode komunikasi bot. Adapun metode penelitian yang digunakan adalah metode *Research and Development* (R&D) dengan membuat sebuah prototipe Bot Telegram untuk memaksimalkan pengembangan produk. Pada penelitian tersebut, tidak dijelaskan terkait kumpulan teknologi yang digunakan saat mengembangkan Bot Telegram sehingga penelitian tersebut berfokus pada penjelasan

deskriptif terkait solusi pada permasalahan yang diangkat. Hasil Bot Telegram yang telah dikembangkan pada penelitian tersebut dapat dilihat pada Gambar 2.8.



Gambar 2.8 Tampilan Bot Telegram untuk layanan *E-Government*

Berdasarkan referensi artikel yang telah dijabarkan, penulis mampu memahami konsep pengembangan Bot Telegram untuk menunjang aktivitas pengolahan data kolam udang untuk Jala Tech. Adapun tabel perbandingan untuk memperjelas perbedaan antar referensi-referensi digunakan pada Tabel 2.1.

Tabel 2.1 Referensi penelitian terdahulu

No.	Judul	Peneliti	Tujuan Penelitian	Metode dan Teknologi
1.	Implementasi IoT Untuk EWS Menggunakan Metode DES Model Holt Pada Tambak Udang Vaname	M. Wisnu Alfiansyah, I. G. P. Wirarama, A. Zafrullah Mardiansyah	Mengembangkan alat prediksi kualitas air pada tambak udang dengan pemanfaatan Bot Telegram	Penelitian Deskriptif, Metode DES Model Holt, Telegram
2.	Pemanfaatan Telegram Bot Untuk Automatisasi Layanan dan	Gde Sastrawangsa	Menciptakan layanan *smart campus* dengan memanfaatkan Bot Telegram	Penelitian Deskriptif, Teknik *Prototyping*, Telegram

	Informasi Mahasiswa Dalam Konsep *Smart Campus*		sebagai media *platform* untuk penerapan layanan *smart campus*	
3.	Implementasi Telegram *Chatbot* Pada Sistem Pendaftaran Masa Orientasi Mahasiswa Guna Efisiensi Pengelolaan Data	Faris Alfaiz, Maryam	Mengembangkan Bot Telegram sebagai *platform* penunjang layanan orientasi mahasiswa pada Universitas Muhammadiyah Surakarta	Penelitian Studi Kasus, Metode *Waterfall*, Bahasa Pemrograman *Python*, Telegram, *Database MySQL*, *Blackbox Testing*
4.	Pengembangan Telegram Bot *Engine* Menggunakan Metode Webhook Dalam Rangka Peningkatan Waktu Layanan *E-Government*	Moh. Anshori, Primaadi Airlangga	Implementasi Bot Telegram untuk menciptakan layanan *E-Government*	Penelitian Studi Kasus, Metode *Research and Development*, *Webhook*, Teknik *Prototyping*, *Blackbox* dan *Whitebox Testing*

BAB III
PELAKSANAAN MAGANG

Pada pelaksanaan aktivitas magang setiap harinya, penulis mengawali hari dengan mengulas kembali pekerjaan yang telah dilakukan pada hari sebelumnya. Hal tersebut bertujuan untuk mempersiapkan *daily meeting* yang dilakukan oleh seluruh anggota tim *software* di Jala Tech. Pada saat *daily meeting* dilakukan, penulis bergabung ke dalam sebuah pertemuan virtual menggunakan *google meet* bersama anggota tim lainnya. *Daily meeting* dilakukan untuk membahas perkembangan proyek dan juga menyampaikan rencana aktivitas yang akan dilakukan pada hari tersebut oleh masing-masing anggota tim. Setelah melaksanakan *daily meeting*, penulis melanjutkan aktivitas untuk menyelesaikan pekerjaan sesuai pada rencana aktivitas yang sudah disampaikan sebelumnya. Ketika penulis mendapati kendala pada saat mengerjakan tugas yang diberikan, biasanya penulis akan menghubungi senior atau *supervisor* penulis untuk melakukan diskusi terkait kendala yang didapat.

Kemudian di setiap hari Jum'at, penulis melakukan *weekly meeting* pada sore hari bersama seluruh anggota tim *software*. *Weekly meeting* diadakan untuk menyampaikan perkembangan proyek yang telah dikerjakan selama satu minggu penuh dan juga merencanakan target yang akan dicapai pada minggu berikutnya. *Weekly meeting* biasanya dipimpin oleh *supervisor* penulis yakni Farid Inawan selaku *Chief Technology Officer* (CTO) di Jala Tech.

3.1 Penerapan Metode *Scrum* di Jala Tech

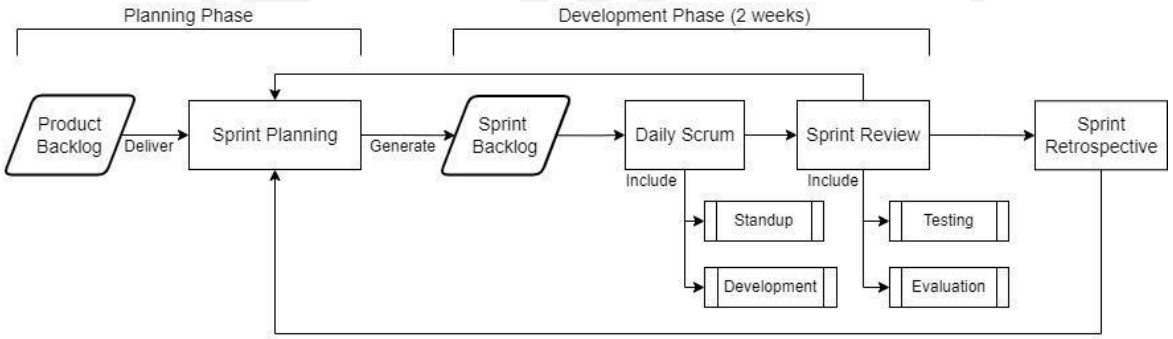
Metode pengembangan yang diterapkan di Jala Tech pada setiap proyek pengembangan produk *software* adalah *scrum*. *Scrum* merupakan salah satu metode pengembangan perangkat lunak yang bersifat cepat dan adaptif sehingga pada fase perencanaan hingga fase pengembangan yang memuat implementasi, pengujian dan evaluasi dapat bersinkronisasi dan dijalankan dengan cepat (Grebić & Stojanović, 2021). Terdapat dua fase dalam metode *scrum* yang diterapkan di Jala Tech, yaitu fase perencanaan dan fase pengembangan. Adapun empat tahapan dari kedua fase tersebut, yaitu *sprint planning*, *sprint development* (*daily scrum*), *sprint review*, dan *sprint retrospective*. Berikut ini adalah penjelasan dari masing-masing tahapan dari kedua fase tersebut:

a. *Sprint planning*, merupakan tahapan untuk merencanakan implementasi dalam suatu proyek. Mulai dari pembahasan *product backlog* oleh *product owner*, kemudian

pembahasan *sprint backlog* atau daftar penugasan sesuai *product backlog* yang telah disampaikan, dan pemberian tugas-tugas tersebut yang dilakukan oleh *product manager* kepada masing-masing anggota tim.

b. *Sprint development*, merupakan tahapan pengembangan terkait implementasi pada daftar *backlog* yang telah dibuat pada tahapan *planning*. Pada tahapan *development* juga dilakukan *daily meeting* dan juga *weekly meeting* untuk sinkronisasi perkembangan proyek terhadap seluruh anggota tim.
c. *Sprint review*, merupakan tahapan yang memuat kegiatan evaluasi hasil implementasi yang dilakukan dalam setiap minggunya maupun dalam satu *sprint*. Kegiatan ini juga dilakukan pemaparan hasil pengujian dari fitur pada spesifikasi suatu produk yang telah diimplementasikan pada tahapan *development*.
d. *Sprint retrospective*, merupakan tahapan yang memuat kegiatan evaluasi performa dari masing-masing anggota tim pada tahap pengembangan. Selain itu, seluruh anggota tim juga mendiskusikan perihal kendala yang didapati saat mengerjakan tugas beserta solusi untuk mengatasi kendala tersebut yang nantinya dapat dijadikan sebagai pembelajaran dan juga peningkatan efektivitas pengerjaan tugas pada *sprint* selanjutnya.

Adapun ilustrasi tahapan dari metode pengembangan *scrum* di Jala Tech yang dapat disimak pada Gambar 3.1.



Gambar 3.1 Ilustrasi alur metode pengembangan *scrum* di Jala Tech

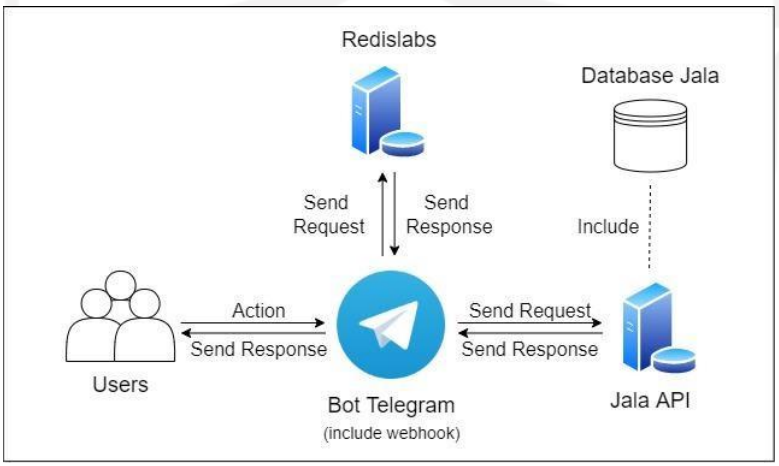
3.2 Spesifikasi Kebutuhan Bot Telegram Jala

Pada tahapan *planning*, dilakukan penyampaian *product backlog* oleh *product owner* untuk mendiskusikan implementasi terkait spesifikasi yang diperlukan. Farid Inawan selaku *product owner* pada proyek pengembangan Bot Telegram Jala, menyatakan bahwa Bot

Telegram harus menerapkan pola interaksi yang interaktif layaknya percakapan antar pengguna. Selain itu, Bot Telegram harus memuat fitur inti terkait pengolahan data kolam sama seperti aplikasi Jala pada umumnya. Bot Telegram Jala perlu mengolah pesan-pesan dari pengguna dan memberikan respon kepada pengguna dalam waktu yang cukup singkat, karena terdapat banyaknya prosedur yang perlu dilakukan oleh pengguna dalam mengakses fitur pada Bot Telegram Jala nantinya. Maka dari itu, pola interaksi *event-based* dan juga metode komunikasi perlu diterapkan supaya bot dapat dengan mudah dan cepat dalam mengolah logika respon dan pesan yang diberikan (Anshori et al., 2020; Telegram, 2020).

Untuk menerapkan pola interaksi *event-based*, maka konsep *state design pattern* perlu diterapkan supaya Bot Telegram dapat dengan mudah mengolah pesan yang didapat berdasarkan *context* fitur dan *state* yang diakses oleh pengguna. Selain itu, *state design pattern* juga dimanfaatkan untuk mengolah *callback data* setiap kali pengguna menggunakan *interactive menu* yang menggunakan fitur *inline keyboard button* pada Telegram. Kemudian *webhook* diterapkan supaya Bot Telegram tidak perlu melakukan *request* secara periodik untuk mendapatkan pesan terbaru dari pengguna, karena server Telegram akan mengirimkan semua pesan yang dikirimkan oleh pengguna ke dalam bentuk *request* kepada *url* yang telah ditetapkan sebagai *event listener* oleh pengembang. Tidak seperti *long-polling* di mana dalam setiap *request* kepada server Telegram, Telegram hanya dapat memberikan pesan yang terakhir dikirimkan oleh pengguna.

Adapun gambaran arsitektur pada Bot Telegram Jala sesuai kebutuhan yang dipaparkan dapat dilihat pada Gambar 3.2.

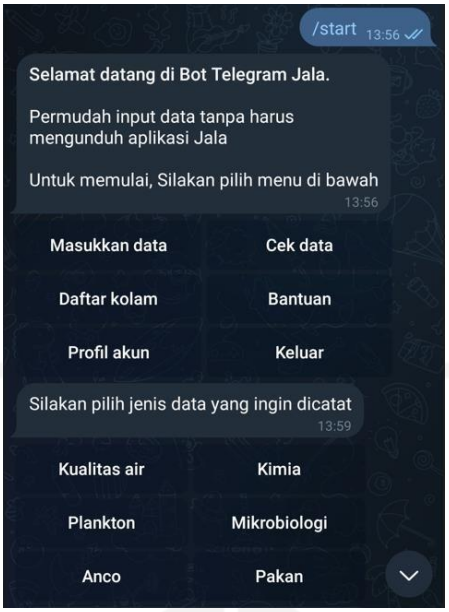


Gambar 3.2 Arsitektur Bot Telegram Jala

Sesuai pada Gambar 3.2, adapun alur dalam sistem Bot Telegram Jala yang dapat disimak berikut ini:

a. Ketika pengguna mengirimkan pesan atau melakukan suatu aksi untuk mengakses suatu fitur, maka Bot Telegram Jala akan memproses pesan atau aksi pengguna yang didapatkan dengan menggunakan *webhook* pada Telegram. Bot Telegram Jala perlu melakukan validasi terkait pesan atau aksi yang dikirimkan pengguna sebelum Bot Telegram Jala melanjutkan ke proses berikutnya.
b. Setelah pesan atau aksi tersebut tervalidasi, maka dapat memproses respon berdasarkan pesan atau aksi yang diberikan dan juga berdasarkan *context* dan *state* fitur yang diakses oleh pengguna. Bot Telegram Jala perlu memperbarui *context* dan *state* yang diakses oleh pengguna supaya tahapan prosedural dalam suatu fitur dapat terus berjalan hingga pengguna tersebut mengakhiri prosesnya sendiri. Oleh karena itu, Bot Telegram Jala juga perlu menerapkan *caching* untuk menyimpan data-data pengguna ke dalam *database* menggunakan aplikasi Redislabs.
c. Pada saat mengolah pesan atau aksi dari pengguna, jika *request* ke API Jala diperlukan maka Bot Telegram Jala akan sekaligus melakukan *request* sesuai instruksi dari pengguna untuk mendapatkan data-data dari *database* Jala. Setelah mendapatkan *response*, Bot Telegram Jala dapat memproses respon sesuai *response* yang didapat dan dijadikan sebuah pesan yang bersifat sebagai umpan balik Bot Telegram Jala kepada penggunanya.

Sebagai penunjang pola interaksi *event-based*, Bot Telegram Jala dibangun dengan model *chatbot* percakapan interaktif yaitu interaksi antara pengguna dengan bot mengandung unsur tanya-jawab. Bot Telegram Jala juga perlu dilengkapi dengan *interactive menu* dengan mengimplementasikan fitur *inline keyboard button* yang telah disediakan oleh Telegram Bot API, supaya pengguna tidak perlu mengetik perintah seperti pada pola interaksi *command-based*. Mengimplementasikan *inline keyboard* pada Bot Telegram Jala juga cukup mudah, pengembang hanya perlu mendefinisikan pilihan menu beserta *callback data* pada keluaran yang menjadi respon. *Callback data* dapat diolah dengan menggunakan *event handler* dari Telegram bernama *callback query*. Contoh implementasi *inline keyboard button* dan *callback query* dapat dilihat pada Gambar 3.3 dan Gambar 3.4.



Gambar 3.3 Fitur *inline keyboard button*

```
asnyc featureFunction ({ message: {chat} }) {
    // do something...
    return await telegramBot.sendMessage(
        chat.id,
        `Isi pesan dari respon bot`,
        {
            reply_markup: JSON.stringify({
                inline_keyboard: [
                    // Array button baris pertama
                    [
                        { text: 'Menu 1', callback_data: '/menu-1' },
                    { text: 'Menu 2', callback_data: '/menu-2' }
                    ],
                    // Array button baris lainnya
                ]
            })
        }
    );
}
telegramBot.on('callback_query', async (callbackQuery) => {
    const { data, message: { chat } } = callbackQuery;
    let userContext = JSON.parse(await redisClientGet(chat.id));
    await telegramBot.answerCallbackQuery(callbackQuery.id, {});

    if (userContext.isConnected) {
        switch(data) { // Cek callback_data
            case '/menu-1':
                // do something...
                break;
            case '/menu-2':
                // do something...
                break;
        }
    }
})
```

Gambar 3.4 Kode implementasi *callback query*

Caching pada Bot Telegram Jala diterapkan untuk menyimpan data-data pengguna yang bersifat sementara, data-data tersebut berupa nama *context* pada fitur yang diakses beserta *state* pada fitur tersebut, status terhubung akun Jala, dan *temporary* data sesuai masukan dari pengguna saat mengakses fitur pencatatan data. *Temporary* data tersebut dibuat karena model bot yang dibuat berbasis percakapan interaktif, sehingga masukan pengguna dibuat secara tanya-jawab. Data pengguna nantinya akan disimpan ke dalam *database* Redislabs yang menunjang penyimpanan berbasis *cloud*. Data-data pengguna tersebut disatukan dalam sebuah objek yang memuat struktur data sesuai pada Gambar 3.5.

```
userContext = {
    context: String,
    state: String,
    isConnected: Boolean,
    data: Object,
};
```

Gambar 3.5 Struktur data objek konteks pengguna

3.3 Sprint Planning

Pada tahap ini, dilakukan pertemuan dengan dihadiri oleh semua anggota tim dalam suatu proyek untuk membahas *product backlog* dari *product owner* pada proyek terkait. *Product backlog* merupakan daftar kebutuhan fungsional sistem yang diperlukan untuk diimplementasi oleh para *developer* (Bhavsar et al., 2020). *Sprint planning* juga membahas terkait pembagian tugas-tugas yang perlu dikerjakan sesuai dari *product backlog* untuk dibagikan kepada *developer*, pembagian tugas dilakukan dengan mengklasifikasi bobot tugas dan *scope* fitur kebutuhan sistem. Adapun daftar *backlog* yang dikerjakan pada proyek pengembangan Bot Telegram yang dapat disimak pada Tabel 3.1.

Tabel 3.1 Daftar *backlog* pada proyek pengembangan Bot Telegram Jala

No.	Nama *Backlog*	Keterangan
1.	Integrasi Akun Jala	Bot Telegram harus memuat fitur untuk penyambungan serta pemutusan akun Jala supaya Bot Telegram dapat berintegrasi dengan data-data pengguna sesuai akun Jala terkait.
2.	Pencatatan Data Kolam	Bot Telegram harus memuat fitur untuk mencatat semua jenis data kolam pada budi daya udang. Jenis data tersebut yaitu data kualitas air, kimia, plankton, mikrobiologi, pakan udang, wadah pakan (*anco*).
3.	Pemeriksaan Data Hasil Pencatatan	Bot Telegram harus memuat fitur untuk memeriksa hasil semua jenis data dari pencatatan yang dilakukan pengguna. Jenis data sama dengan yang disebutkan sebelumnya.

4.	Profil Pengguna	Bot Telegram dapat menyediakan fitur pengecekan profil terkait akun Jala yang terhubung.
5.	Daftar Tambak dan Kolam	Bot Telegram dapat menyediakan fitur untuk memberikan respon terkait daftar tambak beserta kolam yang dimiliki oleh pengguna.

Setelah mengetahui *backlog* dari proyek pengembangan Bot Telegram Jala, *product manager* kemudian membahas terkait pembagian tugas bersama *developer* dan menyusun tugas-tugas sesuai *product backlog* untuk diimplementasikan sehingga hasil penyusunan tugas tersebut dijadikan sebagai *sprint backlog*. *Sprint backlog* merupakan daftar tugas-tugas yang disusun berdasarkan *product backlog* untuk diimplementasikan pada konteks suatu pengembangan (Bhavsar et al., 2020). Berikut ini terdapat hasil *sprint backlog* yang perlu diimplementasikan yang disajikan pada Tabel 3.2.

Tabel 3.2 *Sprint backlog* yang telah disusun

No.	**Fitur**	**Definisi Tugas**	**Bobot Tugas (Skala 1-10)**
1.	Integrasi Akun Jala	Membuat *file handler* yang memuat fungsi *connect* untuk menyambungkan akun Telegram dengan akun Jala	5
		Membuat *file handler* yang memuat fungsi *disconnect* untuk memutuskan akun Telegram dengan akun Jala	5
2.	Pencatatan Data Kolam	Membuat *file handler* yang memuat fungsi *measurement* untuk mencatatkan data kualitas air pada kolam udang	9
		Membuat *file handler* yang memuat fungsi *chemical* untuk mencatatkan data kimia pada kolam udang	9
		Membuat *file handler* yang memuat fungsi *plankton* untuk mencatatkan data plankton pada kolam udang	8
		Membuat *file handler* yang memuat fungsi *microbio* untuk mencatatkan data mikrobiologi pada kolam udang	8
		Membuat *file handler* yang memuat fungsi *feeds* untuk mencatatkan data pakan udang pada kolam udang	6
		Membuat *file handler* yang memuat fungsi *feed-tray* untuk mencatatkan data *anco* pada kolam udang	9
3.	Pemeriksaan Data Hasil Pencatatan	Membuat *file handler* yang memuat fungsi *check-measurement* untuk memeriksa hasil pencatatan data kualitas air	8
		Membuat *file handler* yang memuat fungsi *check-chemical* untuk memeriksa hasil pencatatan data kimia	8
		Membuat *file handler* yang memuat fungsi *check-plankton* untuk memeriksa hasil pencatatan data plankton	7

		Membuat *file handler* yang memuat fungsi *check-microbio* untuk memeriksa hasil pencatatan data mikrobiologi	7
		Membuat *file handler* yang memuat fungsi *check-feeds* untuk memeriksa hasil pencatatan data pakan	6
		Membuat *file handler* yang memuat fungsi *check-feed-tray* untuk memeriksa hasil pencatatan data *anco*	7
4.	Profil Pengguna	Bot Telegram dapat menyediakan fitur pengecekan profil terkait akun Jala yang terhubung.	4
5.	Informasi Daftar Kolam	Bot Telegram dapat menyediakan fitur untuk memberikan respon terkait daftar kolam beserta tambak yang dimiliki oleh pengguna.	4

3.4 Sprint Development

Setelah merencanakan *sprint backlog*, selanjutnya masuk ke fase *development* di mana para pengembang mengerjakan tugas-tugas yang telah dibagikan pada *sprint backlog*. Fase *development* dilakukan selama dua minggu, yang memuat tahapan seperti *daily scrum* dan *sprint review* yang dilakukan pada hari terakhir sebuah *sprint*. *Daily scrum* merupakan segala aktivitas terkait kolaborasi antar anggota tim yang membahas kemajuan pengembangan, *daily scrum* di Jala Tech memuat aktivitas seperti *daily meeting*. *Daily meeting* di Jala Tech biasanya membahas perkembangan tugas yang dikerjakan oleh setiap individu, membahas rencana aktivitas harian, dan membahas kendala yang terjadi pada saat mengerjakan tugas dari *backlog*. Minggu pertama pada sebuah *sprint* difokuskan untuk pengerjaan tugas-tugas atau implementasi, lalu pada minggu kedua mulai diberlakukan pengujian terhadap fitur-fitur yang telah diimplementasi pada minggu pertama. Di hari terakhir *sprint, review* diberlakukan untuk membahas tugas-tugas yang sudah diselesaikan maupun belum selesai. Pada *sprint review* juga dilakukan pengambilan keputusan oleh *product manager* untuk menentukan kebutuhan *sprint* tambahan pada proyek Bot Telegram Jala. Selain itu, *sprint retrospective* dilakukan juga di hari terakhir *sprint* untuk merumuskan masalah dan solusi terkait kendala dalam implementasi supaya seluruh anggota tim dapat meningkatkan efisiensi dalam pengerjaan tugas.

Pada proyek Bot Telegram Jala, pengembangan dengan *scrum* diharapkan dapat selesai setidaknya dalam dua *sprint* sehingga proses *production* dapat segera dilakukan. Sebelum melakukan pengembangan Bot Telegram Jala, penulis berdiskusi dengan Farid Inawan selaku *product owner* pada proyek Bot Telegram Jala untuk membahas pola desain kode pemrograman yang akan diterapkan nantinya. Beliau menyatakan bahwa Bot Telegram perlu memanipulasi serta memeriksa *context* dan *state* pengguna sehingga pada saat pengguna

mengakses suatu fitur, bot dapat merespon konteks dari fitur dan memberikan respon awal dari fitur yang diakses. Maka dari itu, telah disepakati untuk menerapkan pemrograman dengan *state design pattern* dengan bahasa pemrograman *JavaScript* menggunakan *syntax* kondisional *switch-case* dan *if-else*. *Context* diperlukan untuk menyimpan nama konteks dari sebuah fitur, sedangkan *state* diperlukan untuk menyimpan kondisi atau situasi pada saat pengguna mengakses suatu *context* fitur. Akan tetapi, pemeriksaan *context* hanya dilakukan setiap kali pengguna perlu memasukkan data dengan mengetik secara mandiri sehingga fitur-fitur yang hanya mengembalikan satu respon dari bot dapat dimanipulasi dengan *callback query* pada *inline keyboard button* dari Telegram. Berikut ini disajikan daftar *context* dan *state* yang diperlukan untuk mengelola respon Bot Telegram Jala pada saat pengguna mengetikkan *input* pada Tabel 3.3.

Tabel 3.3 Daftar *context* fitur beserta *state* yang diperlukan

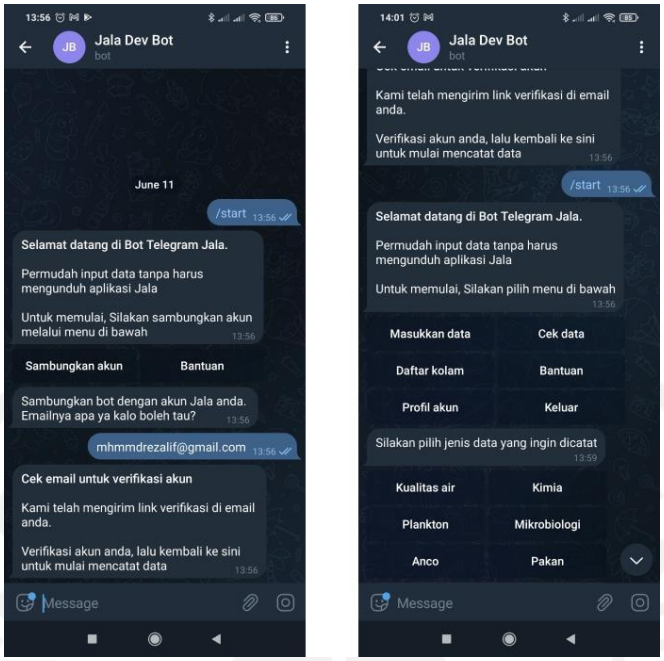
Context	***State***	***Context***	***State***
/connect	*input-email*	***/plankton***	*input-measured_date*
/measurement	*input-measured_date*		*input-measured_time*
	input-measured_time		*choose-farm*
	choose-farm		*choose-pond*
	choose-pond		*input-ga*
	input-ph		*input-bga*
	input-salinity		*input-dinoflagellata*
	input-temperature		*input-prozoo*
	input-do		*input-total_plankton*
	input-weather_parameter		*more-parameter-confirm*
	input-water_color		*more-parameter-input*
	more-parameter-confirm		*more-ponds-confirm*
	more-parameter-input	***/microbio***	*input-measured_date*
	more-ponds-confirm		*input-measured_time*
/chemical	*input-measured_date*		*choose-farm*
	input-measured_time		*choose-pond*
	choose-farm		*input-yellow_vibrio*
	choose-pond		*input-black_vibrio*
	input-ammonia		*input-green_vibrio*
	input-alkalinity		*input-luminance*
	input-nitrat		*input-total_vibrio*
	more-parameter-confirm		*input-total-bacteria*
	more-parameter-input		*more-parameter-confirm*
	more-ponds-confirm		*more-parameter-input*
			more-ponds-confirm
/feeds	*input-logged_date*	***/feed-tray***	*choose-farm*
	input-logged_time		*choose-pond*

	choose-farm		*input-feed_date*
	choose-pond		*choose-feed_time*
	input-fasting		*input-logged_date*
	input-quantity		*input-logged_time*
	input-remark		*input-tray_status_question*
	more-ponds-confirm		*input-tray_answer*
			input-remark
/measurement_check	*choose-farm*	***/microbio_check***	*choose-farm*
	choose-pond		*choose-pond*
/chemical_check	*choose-farm*	***/feeds_check***	*choose-farm*
	choose-pond		*choose-pond*
/plankton_check	*choose-farm*	***/feed_tray_check***	*choose-farm*
	choose-pond		*choose-pond*

Setelah mengetahui *context* beserta *state* yang diperlukan saat pengguna mengetikkan *input* data, penulis mendapatkan gambaran dalam mengelola *context* dan *state* pada setiap fitur. Kemudian penulis mengerjakan tugas-tugas sesuai *backlog* yang telah diberikan kepada penulis dengan menerapkan *state design pattern* dan *caching* menggunakan Redislabs pada pengembangan Bot Telegram Jala.

3.4.1 Fitur Integrasi Akun Jala

Untuk mengerjakan fitur ini, penulis membuat dua *file* bernama *connect.js* dan *disconnect.js* yang keduanya memuat fungsi untuk mengintegrasikan akun Telegram pengguna dengan akun Jala. *File connect.js* perlu memuat dua fungsi karena pada fitur penyambungan akun Jala, pengguna perlu memasukkan data berupa *email* pada akun Jala terkait supaya bot dapat melakukan *request* ke API Jala terkait penyambungan akun Telegram dengan akun Jala. Fitur ini memiliki *context* bernama "*/connect*" dan memuat satu *state* bernama "*input-email*", lalu dua fungsi yang telah disebutkan diberi nama *connect* dan *connectWithContext*. Tampilan Bot Telegram saat pengguna telah melakukan penyambungan akun Jala dapat dilihat pada Gambar 3.6.



Gambar 3.6 Tampilan Bot Telegram pada fitur integrasi akun Jala

Fungsi *connect* digunakan dan dipanggil (*invoke*) saat pengguna menekan tombol "Sambungkan akun" pada perintah bot "*/start*", pada fungsi ini penulis menerapkan pengecekan kondisi terkait pengguna yang sudah terhubung dengan akun Jala sebelumnya sehingga pengguna harus melakukan *disconnect* terlebih dahulu apabila ingin mengganti akun Jala. Setelah pengguna memasukkan *email*, bot akan mengirimkan *request* ke API Jala untuk menyambungkan akun Telegram pengguna dengan akun Jala dengan *email* terkait. Jika akun Jala tersebut tersedia, API Jala akan mengirimkan verifikasi melalui *email* kepada pengguna. Pada sisi bot, bot akan mengirimkan pesan kepada pengguna untuk memeriksa *email* terkait. Tetapi jika akun Jala tersebut tidak tersedia, maka API Jala akan mengirimkan pesan *error* yang kemudian diolah melalui bot sebagai indikasi bahwa *email* yang dimasukkan pengguna tidak valid atau tidak tersedia. Keseluruhan kode program pada fungsi *connect* dapat disimak pada Gambar 3.7.

```
module.exports.connect = async (message = { chat }) => {
    // Cek akun jala yang sudah terhubung
    const user = await userJala(chat.id).catch(() => null);
    if (user) {
        // Respon bot
        return await telegramBot.sendMessage(chat.id,
            'Akun anda telah terhubung dengan akun Jala berikut:\n\n' +
            `Nama: ${user.name}\n` +
            `Email: ${user.email}\n` +
            `Telp: ${user.phone}`
        );
    } else {
        const bot = await telegramBot.getMe();
        const userContext = {
            context: '/connect',
            state: 'input-email',
            isConnected: false,
            data: {
                email: '',
                redirect: `https://t.me/${bot.username}?start=start`
            }
        };
        await redisClientSet(chat.id, JSON.stringify(userContext));
        return await telegramBot.sendMessage(chat.id,
            'Sambungkan bot dengan akun Jala anda. Emailnya apa ya kalo boleh
            tau?'
        );
    }
}
```

Gambar 3.7 Kode program fungsi *connect*

Pada Gambar 3.3, penulis menerapkan *caching* pada kondisi *else* di mana tidak terdapat akun Jala yang sedang terhubung dengan akun Telegram pengguna. *Caching* diterapkan dengan membuat sebuah objek bernama *userContext* yang memuat properti *context*, *state*, *isConnected*, dan *data*. *Caching* perlu diterapkan pada setiap kali terjadi perubahan data pada objek *userContext.* Properti *data* diisi dengan yang memuat data-data yang diperlukan dari pengguna sesuai pada fitur yang diakses, pada fitur ini penulis memberikan properti *email* dan *redirect*. *Email* dibutuhkan untuk mendapatkan masukan data berupa *email* dari pengguna, sedangkan *redirect* merupakan properti yang memuat *link url* Bot Telegram Jala dan digunakan pada saat verifikasi berhasil dilakukan oleh pengguna. Setelah membuat objek *userContext*, penulis menuliskan kode *request* ke Redislabs dengan menggunakan fungsi *redisClientSet* yang memuat parameter id Telegram pengguna sebagai *key* dan parameter objek *userContext* yang perlu diubah ke tipe *String* dengan fungsi *JSON.stringify* pada *JavaScript* sebagai *value* pada data Redis.

Fungsi *connectWithContext* digunakan dan dipanggil pada saat pengguna memasukkan data *email* untuk menghubungkan akun Jala. Fungsi ini memuat *request* ke API Jala untuk menghubungkan akun Telegram pengguna dan *error handler* dari *email* yang telah dimasukkan

oleh pengguna apabila format *email* yang dimasukkan pengguna tidak valid atau *email* tersebut tidak tersedia, sehingga gambaran kode program pada fungsi *connectWithContext* dapat disimak pada Gambar 3.8.

```
module.exports.connectWithContext = async (message = {}, context = {}) => {
    switch (context.state) {
        case 'email':
            context.data.email = message.text;
            await axiosJala.post('/register/telegram', {
                telegram_id: message.chat.id,
                ...context.data,
            }).then(async () => {
                await redisClientSet(message.chat.id, JSON.stringify({}));
                await telegramBot.sendMessage(
                    message.chat.id,
                    `*Cek email untuk verifikasi akun*\n\n` +
                    `Kami telah mengirim link verifikasi di email anda.\n\n` +
                    `Verifikasi akun anda, lalu kembali ke sini untuk mulai
mencatat data`,
                    {parse_mode: 'Markdown'}
                );
            }).catch(async (error) => {
                if (error.response && error.response.status === 422) {
                    const errorMessage = Object.keys(error.response.data.errors)
                        .map((key) => error.response.data.errors[key].join('\n'))
                        .join('\n')
                    await telegramBot.sendMessage(
                        message.chat.id,
                        errorMessage
                    );
                }
            });
    }
};
```

Gambar 3.8 Kode program fungsi *connectWithContext*

Setelah menuliskan kode pada kedua fungsi tersebut, penulis kemudian menuliskan kode untuk memanggil kedua fungsi tersebut (*invocation*) yang diolah melalui *event callback_query* dan pengecekan kondisial *context* dengan *switch-case* pada *file index.js*. Kode program *invocation* untuk fungsi *connect* dan *connectWithContext* pada *file* induk dapat disimak pada Gambar 3.9.

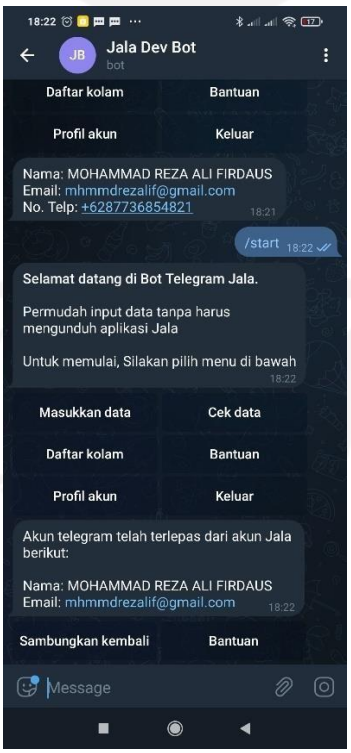
```
index.js

// Trigger bot setiap menerima pesan dari pengguna
telegramBot.on('message', async (message) => {
    let userContext = JSON.parse(await redisClientGet(message.chat.id));
    switch(userContext.context) {
        case '/connect':
            await connectWithContext();
            break;
        ... // Case untuk context lainnya...
        default:
            await telegramBot.sendMessage(message.chat.id, 'Silahkan pilih menu');
    }
});

// Trigger bot setiap pengguna menekan tombol pada inline keyboard button
telegramBot.on('callback_query', async (callbackQuery) => {
    const { data, message: { chat } } = callbackQuery;
    let userContext = JSON.parse(await redisClientGet(chat.id));
    await telegramBot.answerCallbackQuery(callbackQuery.id, {});

    if (userContext.isConnected) {
        switch(data) { // Cek callback_data
            case '/connect':
                await connect({ chat });
                break;
            ... // Case untuk callback_data lainnya...
        }
    }
});
```

Gambar 3.9 Pemanggilan kedua fungsi fitur integrasi akun Jala

Setelah mengimplementasikan fitur untuk penyambungan akun Jala, penulis kemudian membuat fitur untuk pemutusan akun Jala dengan membuat *file handler* bernama *disconnect.js*. Fitur ini hanya memuat satu fungsi bernama *disconnect* dengan berisikan *request* untuk memutuskan akun Jala yang terkait dengan akun Telegram pengguna. Kemudian bot akan mengembalikan respon berupa pesan kepada pengguna yang mengindikasikan bahwa akun Jala sudah terputus dari akun Telegram pengguna. Kode program pada fungsi *disconnect* beserta tampilan Bot Telegram pada saat mengakses fitur pemutusan akun Jala dapat dilihat pada Gambar 3.10 dan Gambar 3.11.

```
module.exports.disconnect = async (message = {}, match = []) => {
    let context = {
        context: '/start-menu',
        state: '',
        isConnected: false,
        data: {},
    };
    const user = await userJala(message.chat.id);
    if (user) {
        await redisClientSet(message.chat.id, JSON.stringify(context));
        await unregisterUserJala(message.chat.id);
        await telegramBot.sendMessage(
            message.chat.id,
            `Akun telegram telah terlepas dari akun Jala berikut:\n\n` +
            `Nama: ${user.name}\n` +
            `Email: ${user.email}\n`,
            {
                reply_markup: JSON.stringify({
                    inline_keyboard: [
                        [{text: "Sambungkan kembali", callback_data:
'/sambungkan'}, {text: "Bantuan", callback_data: '/bantuan'}],
                    ]
                })
            }
        );
    } else {
        throw Error('Gagal memutuskan akun');
    }
};
```

Gambar 3.10 Kode program pada fungsi *disconnect*

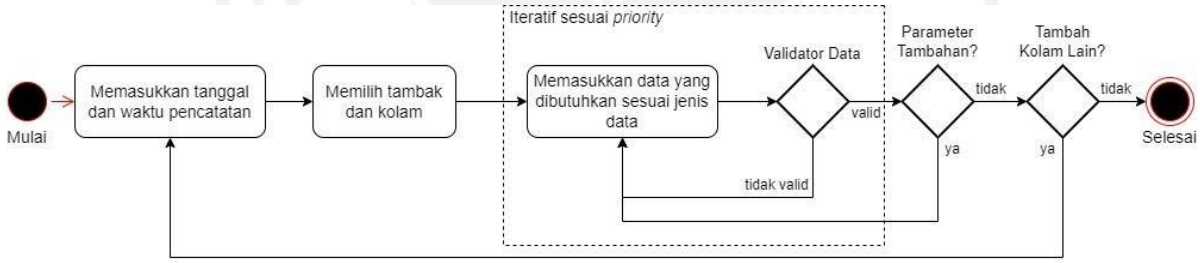


Gambar 3.11 Tampilan Bot Telegram Jala pada fitur pemutusan akun Jala

3.4.2 Fitur Pencatatan Data Kolam

Fitur pencatatan data kolam merupakan fitur yang berfungsi untuk melakukan pencatatan data kolam dengan model interaksi percakapan tanya-jawab antara bot dengan pengguna. Jenis-jenis data yang perlu dimuat dalam fitur ini adalah data kualitas air, kimia, plankton, mikrobiologi, pakan, dan *anco*. Fitur ini perlu dipecah menjadi beberapa bagian *file handler* sesuai dari jenis datanya, maka dari itu penulis membuat *file handler* dari setiap masing-masing jenis data supaya pendefinisian fungsi dapat ditulis dengan mudah.

Sebelum implementasi, penulis perlu berdiskusi bersama Farid Inawan terkait alur interaksi dan parameter-parameter yang perlu dijadikan prioritas sebagai *input* utama. Setelah berdiskusi, akhirnya disepakati alur interaksi yang akan ditetapkan beserta parameter-parameter yang perlu dijadikan prioritas *input*. Prosedur serta alur interaksi pencatatan data pada semua jenis data kolam udang memiliki prosedur yang sama, yang membedakan hanyalah macam-macam kebutuhan datanya beserta *state* yang diakses saja. Berikut ini adalah alur untuk melakukan pencatatan data pada Gambar 3.12.



Gambar 3.12 Diagram alur interaksi pencatatan data kualitas air

Tabel 3.4 Daftar parameter pada data kualitas air

Nama Parameter	Tipe Data
ph	*Float* | *priority*
salinity	*Float* | *priority*
temperature	*Float* | *priority*
do (*Dissolved Oxygen*)	*Float* | *priority*
weather	*String* | *priority*
water_color	*String* | *priority*
water_level	*Float*
do_percent	*Float*
ec	*Float*
tds	*Float*
gravity	*Float*
transparency	*Float*
turbidity	*Float*

Pada Tabel 3.4, disajikan data-data parameter yang tersedia pada jenis data kualitas air. Parameter yang bersifat prioritas utama adalah parameter yang akan dimasukkan ke dalam pertanyaan bot dalam mengakses pencatatan data kolam sesuai jenis data, kemudian ketika pengguna hendak menambahkan data lain maka bot akan memberikan instruksi untuk memasukkan data parameter lainnya dalam satu *chat bubble*. Sebagai contoh implementasi pencatatan data kolam, penulis membuat *file* bernama *measurement.js* yang memuat fungsi pencatatan data kualitas air. Pertama-tama penulis membuat fungsi *measurement* sebagai fungsi awal yang dipanggil (*invoke*) saat pengguna menekan tombol ”Kualitas air” pada bagian “Masukkan data”. Penulis menuliskan objek *userContext* baru dan menerapkan *caching* pada fungsi ini, kemudian objek *userContext* diisi dengan properti *context* dengan nama “*/measurement*”, *state* awal dengan nama “*input-measured_date*”, serta *data* yang memuat nilai *string measured_date*, *measured_time*, dan *array measurements* karena pengguna dapat melakukan pencatatan data lebih dari satu kolam. Kode program dari fungsi *measurement* dapat disimak pada Gambar 3.13.

```
module.exports.measurement = async (message = {}, match = []) => {
    let context = {
        context: '/measurement',
        state: 'measured_date',
        isConnected: true,
        data: {
            measured_date: '',
            measured_time: '',
            measurements: [],
        }
    }
    await redisClientSet(message.chat.id, JSON.stringify(context));
    return await telegramBot.sendMessage(
        message.chat.id,
        'Silakan masukkan tanggal dengan format (YYYY-MM-DD, contoh: 2020-01-
01)'
    );
};
```

Gambar 3.13 Kode program fungsi *measurement*

Setelah itu penulis membuat fungsi *measurementWithContext* untuk menangani masukan dari pengguna. Namun, terdapat beberapa *state* yang menggunakan *callback_query* dengan *inline keyboard* seperti pemilihan tambak, kolam, konfirmasi tambahan parameter, dan konfirmasi pencatatan data pada kolam lainnya. Maka dari itu diperlukan fungsi tambahan untuk menangani *state* tersebut. Urutan pencatatan data umumnya sama pada semua jenis data terkecuali untuk data *anco*, yang membedakan hanyalah parameter atau *input* data yang

dibutuhkan dalam setiap jenis data. Adapun urutan alur pencatatan data kualitas air yang dapat disimak sebagai berikut:

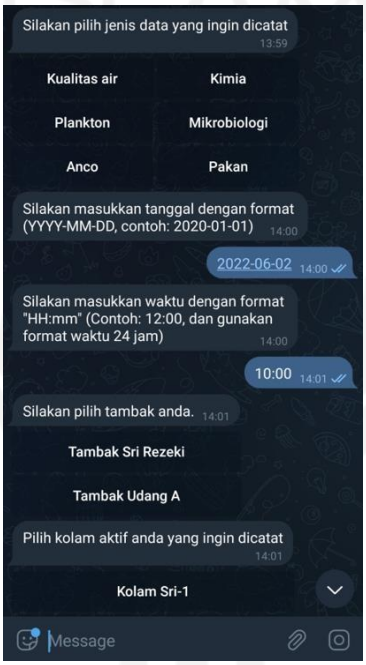
a. Memasukkan tanggal dan waktu pencatatan
b. Memilih tambak dan kolam yang ingin dicatat
c. Memasukkan data *ph*
d. Memasukkan data *salinity*
e. Memasukkan data *temperature*
f. Memasukkan data *do*
g. Memasukkan data *weather*
h. Memasukkan data *water_color*
i. Konfirmasi *input* data parameter lainnya
j. Konfirmasi *input* data pada kolam lainnya

Dengan diketahuinya alur pencatatan pada data kualitas air, penulis kemudian menuliskan kode pada fungsi *measurementWithContext* untuk melakukan pengecekan setiap *state* pada jenis data kualitas air dengan *switch-case*. Pesan yang dikirimkan oleh pengguna bisa didapatkan dari parameter *message*. Kemudian saat pengguna perlu berpindah *state* dalam setiap *input* yang dilakukan, penulis hanya perlu mengganti nilai *state* pada *userContext* yang didapatkan dari parameter *context* dan melakukan *request* ke Redislabs itu menyimpan *userContext*. Gambaran kode program pada fungsi *measurementWithContext* dapat disimak pada Gambar 3.14.

```
module.exports.measurementWithContext = async (message = {}, userContext = {})
=> {
    switch(context.state) {
        case 'measured_date':
            if (...) { // Logika untuk validasi input
                userContext.data.measured_at = message.text;
                userContext.state = measured_time;
                await redisClientSet(message.chat.id, userContext);
                ... // return respon bot
            } else {
                ... // return respon bot
            }
            break;
        ... // case untuk state lainnya
    }
};
```

Gambar 3.14 Kode program fungsi *measurementWithContext*

Setelah pengguna memasukkan waktu pencatatan data, Bot Telegram Jala akan menampilkan menu yang berisi daftar tambak yang dimiliki pengguna. Begitu juga dengan pemilihan kolam, bot akan menampilkan daftar kolam yang dimiliki pengguna sesuai dari tambak yang telah dipilih sebelumnya. Menu tersebut memanfaatkan *inline keyboard button* yang menggunakan *callback_data* dan dapat diolah melalui *event callback_query* pada *file index.js* Ilustrasi pesan dari Bot Telegram Jala yang menyediakan menu tambak dan kolam dapat disimak pada Gambar 3.15.



Gambar 3.15 Pemilihan tambak dan kolam pada fitur pencatatan data kolam

Callback_data pada pemilihan tambak perlu dimanipulasi dengan menambahkan *id* pada *callback_data*-nya, sedangkan untuk *callback_data* pada pemilihan kolam dimanipulasi dengan menambahkan *id* dan *timezone* kolam tersebut. Hal ini dilakukan karena saat pengecekan kondisional terkait *callback_data* pada *file index.js*, setiap *id* beserta data lain yang dimanipulasi tidak selalu sama. Maka dari itu pada *switch-case* pengecekan *callback_data*, diperlukan *default handler* untuk memeriksa *callback_data* yang dimanipulasi. Supaya data tersebut dapat dimanipulasi lagi untuk dijadikan parameter yang disisipkan berdasarkan fungsi terkait pemilihan tambak dan kolam. Berikut ini adalah contoh kode program implementasi pemilihan tambak yang dapat disimak pada Gambar 3.16 dan Gambar 3.17.

```
index.js // nama file
telegramBot.on('callback_query', async (callbackQuery) => {
    const { data, message: { chat } } = callbackQuery;
    let userContext = JSON.parse(await redisClientGet(chat.id));
    await telegramBot.answerCallbackQuery(callbackQuery.id, {});

    if (userContext.isConnected) {
        switch(data) { // Cek callback_data
            ... // Case untuk callback_data lainnya
            default:
                if (data.includes('measurement-choose_farm')) {
                    let farmId = Number(data.split('_')[1]);
                    await farmMeasurementRespond({ chat }, farmId);
                }
                ... // Kondisional lainnya
        }
    }
});

measurement.js // nama file
module.exports.measurementWithContext = async (message = {}, userContext = {})
=> {
    switch(context.state) {
        ... // case untuk state lainnya
        case 'measured_time':
            ... // kode olah data
            // Request tambak pengguna
            const farms = await axiosJala.get('/farms', {
                headers: { 'X-Telegram-ID': message.chat.id }
            }).then((response) => response.data.data).catch(async (error) => {
                if (error.response && error.response.status === 401) return [];
                throw error;
            });

            if (farms) {
                return await telegramBot.sendMessage(
                    message.chat.id,
                    'Silakan pilih tambak anda.',
                    {
                        reply_markup: JSON.stringify({
                            inline_keyboard: farms.map((farm) => {
                                return [
                                    {
                                        text:
`${farm.name.toLowerCase().includes("tambak") ? farm.name : `Tambak
${farm.name}`}`,
                                        callback_data: `input-measurement-choose-
farm_${farm.id}`,
                                    }
                                ]
                            })
                        })
                    }
                );
            } else {
                throw Error('No ponds found');
            }
        ... // case untuk state lainnya
    }
);
            break;
        ... // case untuk state lainnya
    }
};
```

Gambar 3.16 Kode program implementasi pemilihan tambak

```
measurement.js

module.exports.farmMeasurementRespond = async (message = {}, farmId = 0) => {
    let context = JSON.parse(await redisClientGet())
    if (context.context !== '/measurement') {
        ... // return respon bot
    } else {
        if (context.state !== 'choose-farm') {
            ... // return respon bot
        } else {
            const cycles = await axiosJala.get(`/farms/${farmId}`, {
                params: {
                    with: 'unfinished_cycles.pond',
                },
                headers: { 'X-Telegram-ID': message.chat.id }
            }).then((response) => response.data.data).catch(async (error) => {
                if (error.response && error.response.status === 401) return [];
                throw error;
            });

            if (cycles) {
                context.state = 'choose-pond';
                let pondCycles = cycles.unfinished_cycles.map((cycle) => cycle);
                await redisClientSet(message.chat.id, JSON.stringify(context));
                return await telegramBot.sendMessage(
                    message.chat.id,
                    'Pilih kolam aktif anda yang ingin dicatat',
                    {
                        reply_markup: JSON.stringify({
                            inline_keyboard: pondCycles.map((cycle) => {
                                return [
                                    {
                                        text: `Kolam ${cycle.pond.name}`,
                                        callback_data: `input-measurement-choose-
pond_${cycle.pond_id}_${cycle.farm_id}_${cycle.timezone}`,
                                    }
                                ]
                            })
                        })
                    }
                );
            } else {
                throw Error('No farms found');
            }
        }
}};
```

Gambar 3.17 Kode program fungsi untuk respon pemilihan tambak

Kemudian pada fungsi yang merespon pemilihan kolam, penulis memanipulasi *callback_data* pada *button* pilihan kolam dengan menambahkan data *id* dan *timezone* pada kolam. Setelahnya, penulis membuat objek baru dan dimasukkan ke dalam *array* data *measurements* pada *userContext* untuk dijadikan data *request* pada saat pengguna hendak menyimpan data catatan kolamnya. Berikut ini adalah kode program pada fungsi untuk respon pemilihan kolam yang bernama *setMeasurementPondId* pada Gambar 3.18.

```
module.exports.setMeasurementPondId = async (message = {}, pondId = 0, timezone
= '') => {
    let context = JSON.parse(await redisClientGet(message.chat.id));

    if (context.context !== '/measurement') {
        ... // return respon bot
    } else {
        if (context.data.measurements.filter((measurement) =>
measurement.pond_id === pondId).length > 0) {
            ... // return respon bot
        } else if (context.state !== 'choose-pond') {
            ... // return respon bot
        } else {
            // objek per  kolam
            const measurementObj = {
                pond_id: parseInt(pondId),
                ph: 0,
                salinity: 0,
                temperature: 0,
                do: 0,
                transparency: 0,
                color: 0,
                weather: 0,
            }

            context.data.timezone = timezone;
            context.data.pond_id = pondId;
            context.data.measurements.push(measurementObj);
            context.state = 'ph-parameter';

            await redisClientSet(message.chat.id, JSON.stringify(context));

            return await telegramBot.sendMessage(
                message.chat.id,
                'Silakan masukkan jumlah parameter pH air',
            );
        }
    }
}
```

Gambar 3.18 Kode program fungsi *setMeasurementPondId*

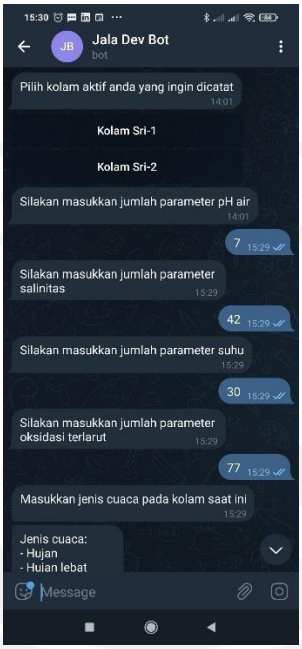
Selanjutnya, penulis mengerjakan pengolahan *input* data berdasarkan *state* yang diperlukan dalam mencatatkan data kualitas air. Semua *input* dari pengguna diolah pada fungsi *measurementWithContext* yang sudah dijelaskan sebelumnya. Berikut ini adalah potongan kode program untuk mengolah *state* pencatatan data pada parameter sesuai data kualitas air pada Gambar 3.19.

```
module.exports.measurementWithContext = async (message = {}, userContext = {})
=> {
    switch(context.state) {
        ... // case untuk state lainnya
        case 'input-ph':
            let phValue = message.text;
            if (phValue.includes(",")) {
                let splitVal = phValue.split("");
                let filterComma = splitVal.filter((char) => char === ",");
                if (filterComma.length > 1) {
                    return await telegramBot.sendMessage(
                        message.chat.id,
                        '<b>[ERROR]</b>\n' +
                        'Masukkan nilai dengan format angka yang benar.',
                        { parse_mode: 'HTML' }
                    );
                }
                phValue = phValue.replace(",", ".");
            }
            if (phValue && !isNaN(phValue)) {
                let paramValue = parseFloat(phValue);
                context.data.measurements.filter((measurement)=>
measurement.pond_id === context.data.pond_id)[0].ph = paramValue;
                context.state = 'salinity-parameter';
                await redisClientSet(message.chat.id, JSON.stringify(context));
                await telegramBot.sendMessage(
                    message.chat.id,
                    'Silakan masukkan jumlah parameter salinitas',
                );
            } else {
                ... // respon error bot
            }
            break;
        ... // case untuk state lainnya
    }
};
```

Gambar 3.19 Kode program saat mengolah parameter *ph* pada data kualitas air

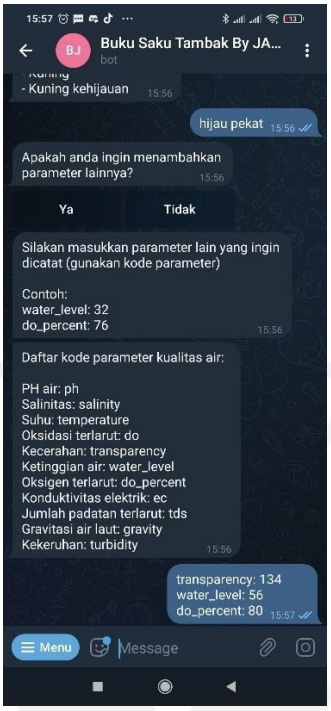
Sesuai pada kode program di atas, saat pengguna melakukan *input* dengan *state* "*input-ph*" diperlukan validasi data yang dimasukkan terlebih dahulu. Karena tipe data *ph* adalah *float*, maka diperlukan validasi terkait penggunaan desimal dan nilai *input* itu sendiri. Namun nilai parameter *ph* tidak selamanya desimal, maka dari itu penulis menerapkan kondisional untuk mengecek ada atau tidaknya koma dalam *input* pengguna. Jika nilai *input* bukan desimal, selanjutnya adalah kondisional untuk mengecek jenis nilai *input*. Jika nilai *input* adalah berupa teks, maka bot akan langsung menghasilkan *output* pada *else* untuk mengirimkan respon mengindikasikan *input* pengguna yang tidak valid tanpa merubah *state* sehingga pengguna harus memasukkan data yang valid. Apabila semua data yang dimasukkan adalah data valid, nilai data tersebut akan di*parsing* terlebih dahulu untuk menjadi nilai *float*. Untuk memasukkan data ke objek sesuai data kolam yang dicatatkan, penulis memanfaatkan fungsi *filter* pada *high order array JavaScript* dengan mencari *id* kolam yang sudah tersimpan pada *userContext*.

Setelah itu *state* pada *userContext* perlu diubah untuk melanjutkan *input* parameter berikutnya. Tampilan Bot Telegram Jala dalam melakukan *input* pada parameter-parameter prioritas dapat dilihat pada Gambar 3.20.



Gambar 3.20 Tampilan Bot Telegram Jala saat mencatat parameter data kualitas air

Setelah semua parameter prioritas telah dimasukkan, bot akan memberikan respon terkait pertanyaan sebagai konfirmasi penambahan parameter jika pengguna ingin menambahkan data parameter lainnya. Untuk menambahkan parameter lainnya, model interaksi bot tidak dibuat pola interaksi tanya-jawab karena hal ini akan memengaruhi performa bot yang semakin banyak dalam memproses *state* nantinya. Sehingga untuk menambahkan *input* parameter lainnya, pengguna dapat langsung mencatatkan parameter-parameter yang diperlukan dan dijadikan dalam satu *chat bubble*. Dengan begitu bot hanya perlu mengolah kode parameter beserta nilai datanya saja dengan melakukan *parsing*. Tampilan Bot Telegram Jala saat melakukan penambahan parameter lainnya dapat dilihat pada Gambar 3.21.



Gambar 3.21 Tampilan Bot Telegram Jala saat menambahkan parameter lainnya

Adapun potongan kode program ketika pengguna memilih opsi untuk menambahkan parameter tambahan yang dapat disimak pada Gambar 3.22.

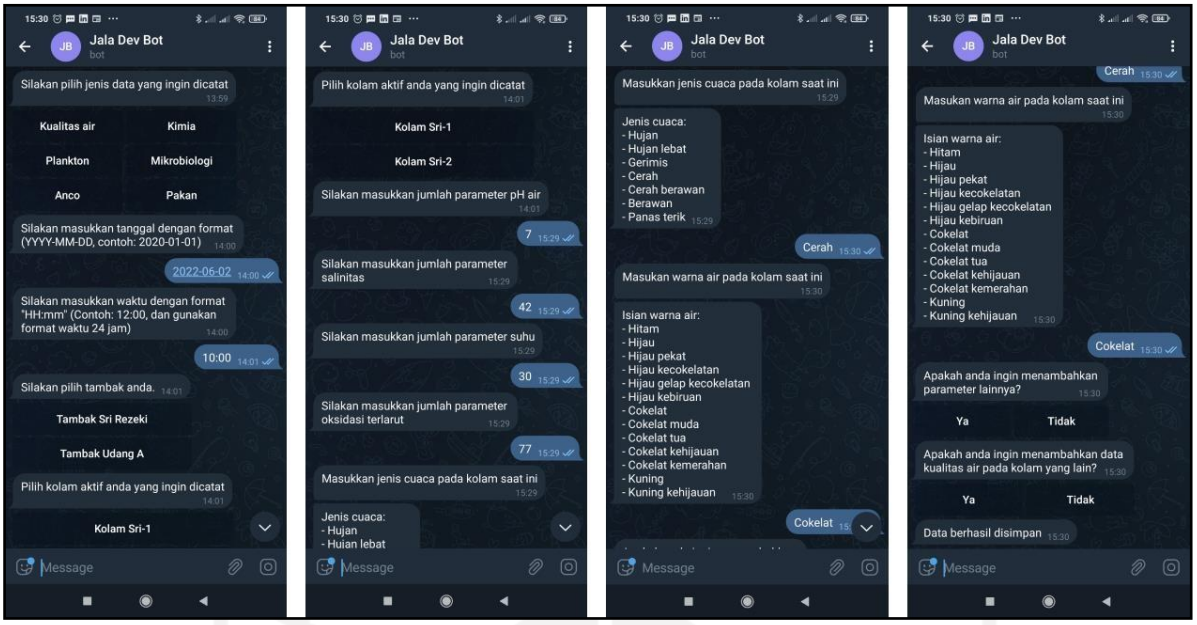
```
const parameterData = message.text.split('\n');
const pondData = context.data.measurements.filter((measurement) =>
measurement.pond_id !== 0)[context.data.measurements.length - 1];
const newPondData = {
    ...pondData,
}
parameterData.forEach(async (parameter) => {
    if (parameter.includes(":")) {
        const parameterData = parameter.trim().split(':');
        const parameterName = parameterData[0].trim().toLowerCase();
        let parameterValue = parameterData[1].trim();
        if (parameterList.includes(parameterName)) {
            if (parameterValue.includes(",")) {
                let splitVal = parameterValue.split("");
                let filterComma = splitVal.filter((char) => char === ",");
                if (filterComma.length > 1) {
                    return await telegramBot.sendMessage(
                        message.chat.id,
                        '<b>[ERROR]</b>\n' +
                        'Masukkan nilai dengan format angka yang benar.',
                        { parse_mode: 'HTML' }
                    );
                }
                parameterValue = parameterValue.replace(",", ".");
            }
            if (!isNaN(parameterValue)) {
                parameterValue = parseFloat(parameterValue);
                newPondData[parameterName] = parameterValue;
            } else {
                return isError = true;
            }
        } else {
            return isError = true;
        }
    } else {
        return isError = true;
    }
});
if (!isError) {
    context.data.measurements[context.data.measurements.length - 1] =
newPondData;
    context.state = 'more_pond_input';
    await redisClientSet(message.chat.id, JSON.stringify(context));
    return await telegramBot.sendMessage(
        message.chat.id,
        'Apakah anda ingin menambahkan data kualitas air pada kolam yang lain?',
        {
            reply_markup: JSON.stringify({
                inline_keyboard: [
                    [
                        { text: 'Ya', callback_data: 'more-input-measurement_yes'
},
                        { text: 'Tidak', callback_data: 'more-input-
measurement_no' }
                    ],
                ]
            })
        }
    );
} else {
    ... // respon error bot
}
```

Gambar 3.22 Kode program untuk menangani *input* parameter lainnya

Selanjutnya, pengguna akan diberikan respon pertanyaan oleh bot terkait pencatatan data pada kolam lainnya. Apabila pengguna hendak menambahkan kolam lainnya untuk dicatat, bot akan mereset *state* pengguna untuk memilih kolam dengan menu *inline keyboard button*. Kemudian prosedur pencatatan diulang dari pemilihan kolam hingga konfirmasi pencatatan data pada kolam lain, tidak ada batasan terkait jumlah kolam yang dapat ditambahkan. Jika pengguna memilih pilihan "tidak", maka bot akan melakukan *request* ke API Jala untuk menyimpan data-data yang telah dimasukkan. Jika *request* tersebut mengembalikan *response*, maka data hasil pencatatan berhasil disimpan ke *database* Jala. Kemudian *context* dan *state* pengguna pada *userContext* akan diubah menjadi konteks "*/start-menu*" dengan *state* kosong supaya pengguna tidak melakukan aksi-aksi dari respon bot terkait pencatatan data untuk menghindari *error*. Berikut ini adalah potongan kode program dan keseluruhan tampilan ilustrasi pencatatan data yang dapat dilihat pada Gambar 3.23 dan Gambar 3.24.

```
context.data.measurements.forEach((measurement) => {
    measurement.measured_date = moment(`${context.data.measured_date}
${context.data.measured_time}`).format("YYYY-MM-DD");
    measurement.measured_time = moment(`${context.data.measured_date}
${context.data.measured_time}`).format("HH:mm:ss");
});

let measurementsRequest = await axiosJala.post(`/farms/${context.data.farm_id}`,
    { multi_measurements: context.data.measurements },
    { headers: { 'X-Telegram-ID': message.chat.id } }
).then((response) => response.data.data).catch(async (error) => {
    return await telegramBot.sendMessage(
        message.chat.id,
        '<b>[ERROR]</b>\n' +
        'Terjadi kesalahan saat menyimpan data kualitas air, Silakan coba lagi',
        { parse_mode: 'HTML' }
    );
});

if (measurementsRequest) {
    context = {
        context: '/start-menu',
        state: '',
        isConnected: true,
        data: null,
    }
    await redisClientSet(message.chat.id, JSON.stringify(context));
    return await telegramBot.sendMessage(
        message.chat.id,
        'Data berhasil disimpan'
    );
}
```

Gambar 3.23 Kode program untuk menangani *request* pencatatan data



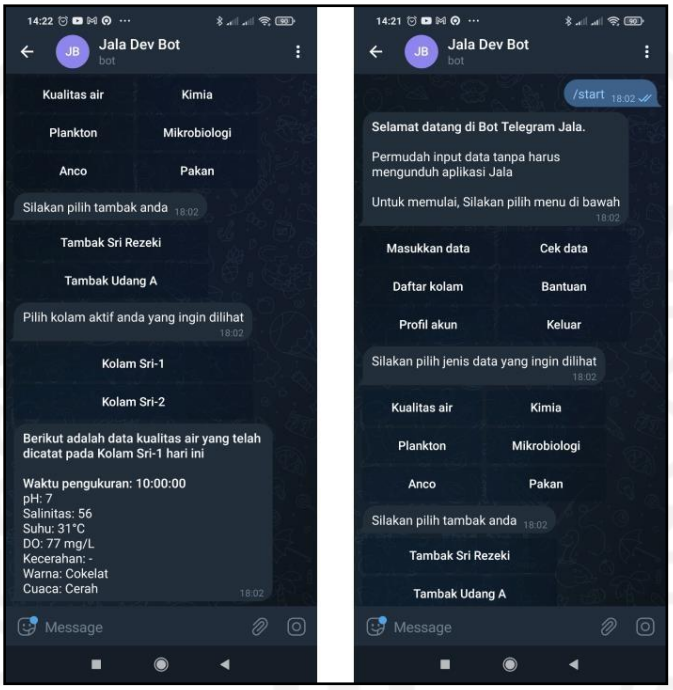
Gambar 3.24 Ilustrasi pencatatan data kualitas air Bot Telegram Jala

3.4.3 Fitur Pemeriksaan Data Hasil Pencatatan

Setelah menyelesaikan fitur pencatatan data, selanjutnya adalah menyelesaikan fitur pemeriksaan data hasil pencatatan yang dilakukan. Pada fitur ini, *state* yang dimuat kali ini tidak serumit saat mengimplementasikan fitur pencatatan data. Akan tetapi data yang berhasil didapatkan dengan *request* ke API Jala harus diolah dengan baik untuk menghasilkan respon yang sesuai, terlebih lagi jika ada banyak data yang dicatatkan dalam satu hari tertentu. Pemeriksaan data juga harus menunjang semua jenis data yang ada, dan menyajikan data parameter sesuai prioritas. Selain itu pemeriksaan data hasil pencatatan tidak memerlukan masukkan tanggal dan waktu dari mengguna, karena data hasil pencatatan yang diberikan mengikuti sesuai hari terkini.

Sebagai contoh, penulis mengimplementasikan pemeriksaan data hasil pencatatan untuk jenis data kualitas air. Dimulai dengan membuat *file handler* yang akan memuat fungsi pemeriksaan data kualitas air, karena sudah diketahui *state* yang dibutuhkan penulis maka bobot dalam pengerjaan fitur ini pun lebih ringan dibandingkan fitur pencatatan data. Perlu diketahui juga alur dari pemeriksaan data memiliki prosedur yang cukup singkat, sehingga yang menjadi tantangan disini adalah bagaimana cara penulis dalam menuliskan kode program supaya bot dapat mengolah data sesuai *request* dari API Jala dan menyajikan data tersebut dengan baik dan mudah dibaca. Alur pemeriksaan data dimulai dari penulis yang telah memilih menu “Cek data” dengan jenis data “Kualitas air”, kemudian bot akan merespon dengan

mengirimkan pesan yang memuat pilihan tambak yang dimiliki pengguna. Setelah memilih tambak, bot akan memberikan respon kepada pengguna untuk pemilihan kolam dari tambak yang dipilih. Selanjutnya, bot akan mengirimkan *request* ke API Jala untuk meminta data hasil pencatatan yang telah dilakukan sesuai dengan hari terkini dan kolam yang telah dipilih pengguna. Tampilan pemeriksaan data hasil pencatatan dapat dilihat pada Gambar 3.25.



Gambar 3.25 Tampilan pemeriksaan data hasil pencatatan

Pertama, penulis menuliskan kode program untuk fungsi awal pada saat pengguna mengakses fitur pemeriksaan data kualitas air. Pada fungsi awal, penulis mendefinisikan *userContext* beserta *caching request* ke Redislabs. Setelahnya tidak untuk membuat *request* ke API Jala untuk mendapatkan daftar tambak pengguna, karena langkah awal dalam melakukan pemeriksaan data adalah dengan memilih tambak. Penulis menuliskan beberapa kode program yang serupa dengan fitur pencatatan sebelumnya yang menerapkan *callback_data* serta pengolahannya dengan *callback_query*. *Callback_data* pada pemilihan tambak perlu disisipkan *id* tambak sama seperti pada pencatatan data sebelumnya, sedangkan *callback_data* pada pemilihan kolam perlu disisipkan *id* siklus kolam yang telah dicatatkan. *State* yang diubah pun juga sama yakni dengan mengubah *state* ke "*choose-farm*" dan "*choose-pond*". Berikut ini potongan kode program pada fungsi awal pemeriksaan data kualitas air yang bernama *measurementCheck* pada Gambar 3.26.

```
module.exports.measurementCheck = async (message = {}) => {
    let context = {
        context: '/measurement_check',
        state: 'choose-farm',
        isConnected: true,
        data: {},
    };
    await redisClientSet(message.chat.id, JSON.stringify(context)); // caching

    const farms = await axiosJala.get('/farms', {
        headers: { 'X-Telegram-ID': message.chat.id }
    }).then((response) => response.data.data).catch(async (error) => {
        if (error.response && error.response.status === 401) return [];
        throw error;
    });

    return await telegramBot.sendMessage(
        message.chat.id,
        'Silakan pilih tambak anda',
        {
            reply_markup: JSON.stringify({
                inline_keyboard: farms.map((farm) => {
                    return [
                        {
                            text: `${farm.name.toLowerCase().includes("tambak")
? farm.name : `Tambak ${farm.name}`}`,
                            callback_data: `check-measurement-choose-
farm_${farm.id}`,
                        }
                    ]
                })
            })
        }
    );
}
```

Gambar 3.26 Kode program pada fungsi *measurementCheck*

Selanjutnya, penulis membuat fungsi *farmRespond* yang dipanggil ketika pengguna telah memilih tambak. Fungsi ini memuat pengecekan *context* apabila pengguna mencoba menekan tombol dari respon pesan di luar konteks pemeriksaan data kualitas air, sehingga pengguna tidak bisa melakukan aksi di luar konteks suatu fitur. Fungsi ini juga memuat *request* untuk meminta daftar kolam pengguna dan respon yang berisikan pilihan kolam sesuai tambak yang dipilih. Berikut ini adalah potongan kode program pada fungsi *farmRespond* pada Gambar 3.27.

```
module.exports.farmRespond = async (message = {}, farmId = 0) => {
    let context = JSON.parse(await redisClientGet(message.chat.id));

    if (context.context !== '/measurement_check') {
        ... // return respon error bot
    } else {
        context.state = 'choose-pond';
        await redisClientSet(message.chat.id, JSON.stringify(context));
       // Request daftar tambak dengan kolam
        const cycles = await axiosJala.get(`/farms/${farmId}`, {
            params: {
                with: 'unfinished_cycles.pond',
            },
            headers: { 'X-Telegram-ID': message.chat.id }
        }).then((response) => response.data.data).catch(async (error) => {
            if (error.response && error.response.status === 401) return [];
            throw error;
        });

        if (cycles) {
            let pondCycles = cycles.unfinished_cycles.map((cycle) => cycle);
            await telegramBot.sendMessage(
                message.chat.id,
                'Pilih kolam aktif anda yang ingin dilihat',
                {
                    reply_markup: JSON.stringify({
                        inline_keyboard: pondCycles.map((cycle) => {
                            return [
                                {
                                    text: `Kolam ${cycle.pond.name}`,
                                    callback_data: `check-measurement-choose-
pond_${cycle.id}_${cycle.timezone}_${cycle.pond.name}`,
                                }
                            ]
                        })
                    })
                }
            );
        } else {
            throw Error('No farms found');
        }
    }
}
```

Gambar 3.27 Kode program pada fungsi *farmRespond*

Kemudian fungsi *measurementCheckRespond* dibuat sebagai respon setelah pengguna memilih kolam yang akan diperiksa. Fungsi ini memuat *request* ke API Jala untuk meminta data kolam yang hendak diperiksa. Untuk mengolah data kolam yang didapatkan bot, diperlukan fungsi *forEach, map,* dan *filter* pada *high order array function JavaScript* supaya data dapat diolah dengan baik dan disajikan sesuai dengan bentuk pesan yang telah disepakati oleh *product designer*. Berikut ini adalah potongan kode program pada fungsi *measurementCheckRespond* terkait *request*, pengolahan data *response*, dan *return* yang menjadi respon bot kepada pengguna pada Gambar 3.28, Gambar 3.29, dan Gambar 3.30.

```
module.exports.measurementCheckRespond = async (message = {}, cycleId = 0,
timezone = '', pondName = '') => {
    ... // kode lainnya
    // Request API Jala
    const measurements = await axiosJala.get(`/multi_measurements`, {
        params: {
            cycle_id: cycleId,
            measured_date: moment().format("YYYY-MM-DD"),
            sort: 'measured_at,asc',
            per_page: 1000
        },
        headers: { 'X-Telegram-ID': message.chat.id }
    }).then((response) => response.data.data).catch(async (error) => {
        throw Error("Terjadi kesalahan saat mengambil data kualitas air");
    });
};
```

Gambar 3.28 Kode program fungsi *measurementCheckRespond* yang memuat *request*

```
module.exports.measurementCheckRespond = async (message = {}, cycleId = 0,
timezone = '', pondName = '') => {
    ... // kode lainnya
    const measurementKeys = [
        'ph', 'salinity', 'temperature', 'do', 'transparency', 'color',
'weather', 'water_level', 'ec', 'turbidity',
        'do_percent', 'tds', 'gravity'
    ];
    // Array baru
    const mappedMeasurements = measurements.map((measurement) => {
        let mappedMeasurementItem = {
            id: measurement.id,
            cycle_id: measurement.cycle_id,
            pond_id: measurement.pond_id,
            farm_id: measurement.farm_id,
            timezone: measurement.timezone,
            measured_date: measurement.measured_date,
            measured_time: measurement.measured_time,
        };
        // Perulangan untuk mengolah parameter yang dimiliki setiap kolam
        for (const [key, value] of Object.entries(measurement)) {
            if (value !== null && !key.includes("anomaly")) {
                if (measurementKeys.indexOf(key) > -1) {
                    mappedMeasurementItem[key] = value;
                }
            }
        }
        return mappedMeasurementItem;
    });
    ... // kode lainnya
};
```

Gambar 3.29 Kode program lanjutan untuk mengolah *response*

```
module.exports.measurementCheckRespond = async (message = {}, cycleId = 0,
timezone = '', pondName = '') => {
    ... // lanjutan dari kode di atas
    let templates = ''; // Sebagai variabel return
    const weather_parameter = {
        ... // Berisi key dan value dari kode dan nama cuaca
    }
    const color_param = {
        ... // Berisi key dan value dari kode dan nama warna kolam
    }
    mappedMeasurements.forEach(async (measurement, index) => {
        const arrayKeys = Object.keys(measurement);
        if (index === 0) {
            templates += `<b>Waktu pengukuran: ${measurement.measured_time}</b>
\n`;
            if (arrayKeys.length > 7) {
                if (arrayKeys.indexOf("ph") > -1) {
                    templates += `pH: ${measurement.ph ? measurement.ph : '-'}
\n`;
                }
                ... // Pengecekan parameter lainnya
            } else {
                templates += '<i>Tidak ada parameter yang tersedia untuk
ditampilkan</i> \n';
            }
        } else {
            templates += `-------------------------------------------------------------------------
----------- \n` +
            `<b>Waktu pengukuran: ${measurement.measured_time}</b> \n`;
            if (arrayKeys.length > 7) {
                if (arrayKeys.indexOf("ph") > -1) {
                    templates += `pH: ${measurement.ph ? measurement.ph : '-'}
\n`; // Assign ke variabel templates
                }
                ... // Pengecekan parameter lainnya
            } else {
                templates += '<i>Tidak ada parameter yang tersedia untuk
ditampilkan</i> \n';
            }
        }
    });
    // Return respon bot
    return await telegramBot.sendMessage(
        message.chat.id,
        `<b>Berikut adalah data kualitas air yang telah dicatat pada Kolam
${pondName} hari ini</b> \n\n` +
        templates,
        {
            parse_mode: 'HTML'
        }
    );
    ... // kode lainnya
};
```

Gambar 3.30 Kode program lanjutan untuk membuat *string templates* dan respon bot

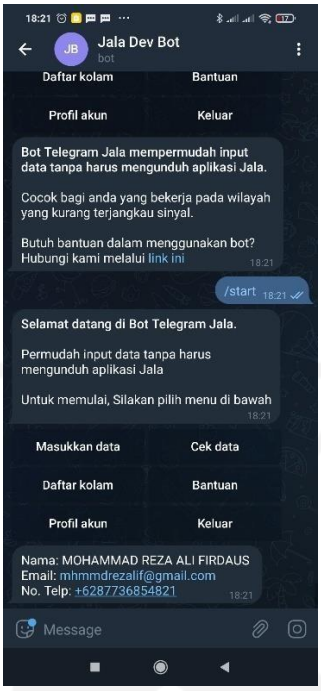
Pertama dimulai dengan melakukan *request* ke API Jala untuk meminta data hasil pencatatan yang sudah dibuat. Ketika *request* tersebut mengembalikan *response*, selanjutnya penulis membuat *array* baru bernama *mappedMeasurements* dengan memanfaatkan fungsi *map* untuk memisahkan data-data yang tidak diperlukan. Saat melakukan *mapping*, di

dalamnya harus diterapkan perulangan untuk membaca parameter-parameter yang dimiliki setiap kolam. Karena tentunya saat terdapat banyak kolam yang dicatat dalam satu waktu, kolam-kolam tersebut akan memiliki parameter-parameter yang berbeda-beda. Maka dari itu penulis menerapkan *for loop* pada setiap objek *measurement* pada *map* supaya objek tersebut menyimpan parameter-parameter yang tersedia.

Selanjutnya penulis membuat variabel *templates* dengan *string* kosong untuk menyimpan data *string* yang akan menjadi respon bot nantinya. Selain itu penetapan objek cuaca dan warna kolam juga diperlukan karena pada saat melakukan *input* parameter cuaca dan warna kolam, *string* diubah menjadi kode yang bersifat *integer* karena pada API Jala nama-nama cuaca dan warna kolam disimpan dalam bentuk kode. Kemudian penulis menerapkan perulangan *forEach* pada *array mappedMeasurements* tadi untuk membuat teks yang akan menjadi respon bot, dengan memasukkan teks *string* ke dalam variabel *templates* dan menggunakan operator tambah (+) untuk menambahkan *string* lainnya. Di dalam perulangan *forEach* juga perlu dicek kondisi terkait *index* pada objek yang dibaca, hal ini karena jika terdapat lebih dari satu objek pesan perlu dipisah dengan gabungan tanda penghubung (-) yang dapat diperhatikan pada kondisi *else*. Setelah perulangan *forEach* selesai, penulis hanya perlu melakukan *return* pada bot untuk mengirim pesan yang memuat variabel *templates* kepada pengguna.

3.4.4 Fitur Profil Pengguna

Fitur profil pengguna merupakan fitur yang berguna untuk menampilkan status akun Jala yang terhubung dengan akun Telegram yang digunakan. Fitur ini merupakan kebutuhan non-fungsional yang cukup membantu pengguna dalam memeriksa status akun Jala yang terkait. Pada fitur ini, data yang ditampilkan adalah data nama lengkap, *email* dari akun Jala, dan nomor telpon pengguna. Berikut ini adalah tampilan Bot Telegram pada saat mengakses fitur profil pada Gambar 3.31.



Gambar 3.31 Tampilan fitur profil pada Bot Telegram Jala

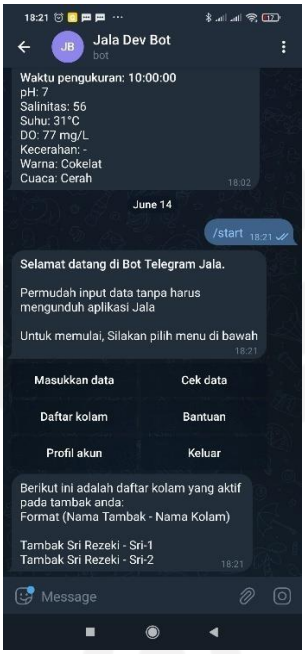
Untuk mengimplementasikan fitur ini, penulis membuat *file handler* bernama *profile.js* yang akan memuat fungsi saat fitur ini dipanggil. Pada fitur ini hanya terdapat satu fungsi dan tidak memerlukan manipulasi *userContext* dan *caching* saat pengguna mengakses fitur ini, karena *callback* yang dilakukan hanyalah sekali dengan melakukan *request* untuk mendapatkan data akun Jala yang digunakan. Setelah itu penulis membuat fungsi bernama *profile* yang berisikan *request* data pengguna Jala dan mengembalikan respon bot sesuai data yang diambil. Tidak lupa juga diperlukan pengecekan terkait *request* pengguna tersebut supaya pada saat akun Telegram pengguna tidak terhubung dengan akun Jala, maka bot dapat mengembalikan respon *error* yang mengindikasikan pengguna perlu menghubungkan akun Jala terlebih dahulu sebelum mengakses fitur profil ini. Berikut ini adalah kode program pada fungsi *profile* pada Gambar 3.32.

```
module.exports.profile = async (message = {}, match = []) => {
    const user = await userJala(message.chat.id);
    await telegramBot.sendMessage(
        message.chat.id,
        `Nama: ${user.name}\n` +
        `Email: ${user.email}\n` +
        `No. Telp: ${user.phone && user.phone !== null ? user.phone : "-"}\n`
    );
};
```

Gambar 3.32 Kode program pada fungsi *profile*

3.4.5 Fitur Informasi Daftar Kolam

Fitur daftar tambak dan kolam merupakan fitur yang mengembalikan pesan berupa data-data terkait daftar tambak dan kolam yang dimiliki oleh pengguna. Fitur ini merupakan kebutuhan non-fungsional pada Bot Telegram Jala. Pada fitur ini, pengguna dapat memastikan siklus kolam yang masih aktif sebelum melakukan pencatatan. Fitur ini juga tidak memerlukan manipulasi *state* pada *userContext*, karena fitur ini hanya memiliki sebuah fungsi yang menjadi *callback*. Tampilan Bot Telegram Jala pada saat mengakses fitur ini dapat dilihat pada Gambar 3.33.



Gambar 3.33 Tampilan fitur daftar kolam pada Bot Telegram Jala

Sebagai langkah awal, penulis membuat *file pond-lists.js* untuk menuliskan fungsi *callback* pada saat pengguna hendak mengakses fitur ini. Kemudian, penulis membuat fungsi bernama *getLists* yang berisikan *request* data tambak yang memuat kolam dengan siklus aktif yang dimiliki pengguna. Setelah melakukan *request*, *response* dari *request* tersebut kemudian diolah untuk menjadi respon bot yang dikirimkan kepada pengguna. Karena *response* tersebut berbasis *array of objects*, maka diperlukan *mapping* untuk menghasilkan *array* yang berisikan kolam yang dimiliki oleh sebuah tambak. Kode program dari fungsi *getLists* dapat dilihat pada Gambar 3.34.

```
module.exports.getLists = async (message = {}) => {
    const ponds = await axiosJala.get('/farms', {
        params: {
            with: 'unfinished_cycles.pond'
        },
        headers: { 'X-Telegram-ID': message.chat.id }
    }).then((response) => response.data.data).catch(async (error) => {
        if (error.response && error.response.status === 401) return [];
        throw error;
    });

    if (ponds) {
        const allCycles = ponds.map((farm) => farm.unfinished_cycles).flat();
        return await telegramBot.sendMessage(
            message.chat.id,
            `Berikut ini adalah daftar kolam yang aktif pada tambak anda: \n` +
            `Format (Nama Tambak - Nama Kolam) \n\n` +
            `${allCycles.map(cycle => {
                return `${cycle.pond.farm_pond_name}\n`
            }).join("")}`,
            {parse_mode: "HTML"}
        )
    } else {
        return await telegramBot.sendMessage(
            message.chat.id,
            'Tidak ada kolam dengan siklus aktif yang tersedia, Silakan buat
kolam dan aktifkan siklus pada kolam tesebut.'
        );
    }
}
```

Gambar 3.34 Kode program pada fungsi *getLists*

3.5 Sprint Review

Setiap memasuki hari terakhir pada tiap satu minggu menjalankan *sprint*, maka dilakukanlah *sprint review* yang berfokus pada pengujian dan evaluasi kinerja tim. Pada tahapan ini juga dilakukan *weekly meeting* untuk menyampaikan kemajuan proyek, rekapan aktivitas mingguan, dan aktivitas yang akan dilakukan pada minggu selanjutnya. *Weekly meeting* biasanya dipimpin oleh *product manager* dan juga dihadiri *product owner*. Karena mayoritas anggota tim bekerja secara *remote* atau *Work from Home* (WFH), *weekly meeting* diadakan secara daring menggunakan aplikasi *Google Meet*.

Pada minggu pertama dalam menjalankan *sprint*, *sprint review* dimanfaatkan untuk mengevaluasi kinerja serta merencanakan aktivitas dan target yang akan dikejar pada minggu kedua. Setiap anggota dari tim yang tergabung akan menyampaikan kemajuan dalam pengerjaan proyek serta merencanakan aktivitas selanjutnya terkait penyelesaian proyek. Pada tahapan ini, penulis ikut berkontribusi untuk menyampaikan *progress* dan ekspektasi fitur yang akan diselesaikan pada minggu selanjutnya.

Pada minggu kedua, *weekly meeting* dilakukan untuk menyampaikan seluruh *progress* yang dilakukan selama satu *sprint* dan menyampaikan rekapan aktivitas pada minggu kedua. Selain itu, *product manager* juga berdiskusi bersama *developer* untuk membahas *sprint* tambahan apabila proyek belum terselesaikan dalam satu *sprint*. Pada *sprint* pertama, penulis hanya dapat menyelesaikan fitur integrasi akun Jala dan fitur pencatatan data sebanyak tiga *subtask* berdasarkan jenis data yang diperlukan. Maka dari itu diperlukan *sprint* tambahan untuk menyelesaikan fitur-fitur lainnya. Minggu kedua juga difokuskan untuk melakukan pengujian terhadap Bot Telegram Jala untuk memastikan fitur-fitur yang telah diimplementasikan pada tahapan pengembangan sebelumnya.

3.5.1 Pengujian Bot Telegram Jala

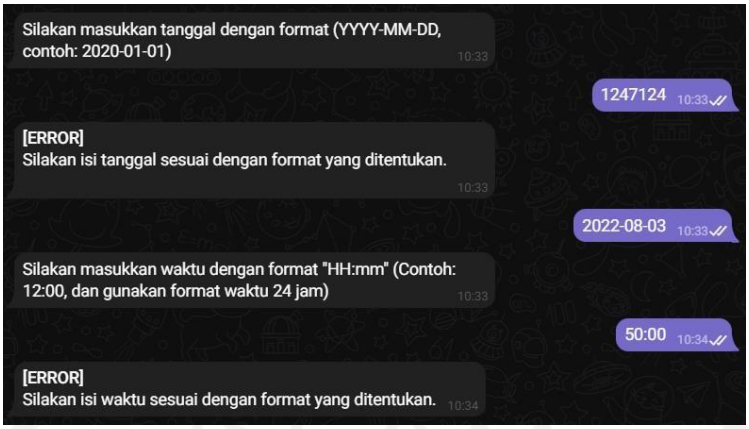
Selanjutnya pada minggu kedua, minggu ini difokuskan untuk melakukan pengujian terkait fitur-fitur yang telah diimplementasi dan melakukan perbaikan *bug* sesuai hasil pengujian. Pada proyek pengembangan Bot Telegram Jala, *Quality Assurance* (QA) dari Jala mulai melakukan pengujian secara mandiri dengan menguji langsung fungsionalitas pada Bot Telegram Jala. Untuk pengujian secara otomatis belum bisa diterapkan karena minimnya teknologi untuk menguji sebuah *chatbot* pada Telegram. Penulis berkoordinasi dengan QA untuk membahas pengujian serta batasan-batasan aksi yang dapat dilakukan dengan Bot Telegram Jala. Pengujian dilakukan dengan metode pengujian *blackbox*, *blackbox testing* merupakan salah satu metode pengujian perangkat lunak yang bertujuan untuk melakukan validasi spesifikasi fungsional yang telah diimplementasikan ke suatu sistem perangkat lunak dengan spesifikasi kebutuhan fungsional sistem sesuai rancangan (Jaya, 2018). Hasil pengujian dapat diklasifikasikan dengan kategori *success*, *success but need improvement*, dan *failed*. Berikut ini adalah tabel rekapan pengujian yang telah dilakukan pada masa pengembangan dapat disimak pada Tabel 3.5.

Tabel 3.5 Hasil pengujian Bot Telegram Jala pada *sprint* pertama

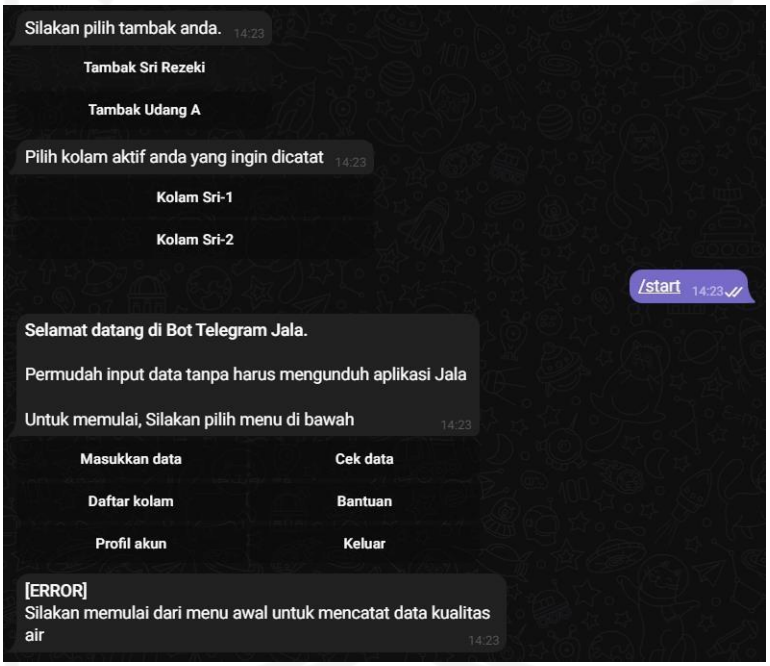
Fitur	Deskripsi	Ekspektasi	Hasil
Sambungkan akun	Mendaftarkan *email* yang valid dan tersedia pada akun Jala	Bot memberikan respon verifikasi *email* untuk tahapan selanjutnya	*Done, success*
	Mendaftarkan *email* tidak valid dan tidak tersedia pada akun Jala	Bot memberikan respon *error*	*Done, success*

Putuskan akun	Memutuskan akun Jala dari akun Telegram pengguna	Bot memberikan respon detail akun Jala yang terputus	*Done, success*
Record data (kualitas air)	Melakukan pencatatan data dengan memilih tambak yang memiliki kolam aktif yang tersedia dengan format data yang valid	Bot berhasil menyimpan data pencatatan	*Done, success*
	Melakukan pencatatan data dengan memilih tambak yang tidak memiliki kolam aktif	Bot mengembalikan respon *error*	*Done, success but need improvement*
	Melakukan *input* desimal dengan standar format nasional (,) pada *input* parameter	Bot berhasil mengganti *input* desimal ke format internasional (.) dan mengembalikan respon untuk pertanyaan parameter lainnya	*Done, success*
	Melakukan pencatatan dengan opsi menambah parameter lainnya	Bot memberikan respon petunjuk untuk menambah parameter lainnya dan menyimpan data	*Done, success*
	Melakukan pencatatan lebih dari satu kolam	Bot berhasil menyimpan data pada kolam-kolam yang dicatat	*Done, success*

Selain melakukan validasi atas fungsional atau prosedural kebutuhan utama pada Bot Telegram Jala, pengujian terhadap batasan *input* dan akses pengguna juga dilakukan untuk menguji batasan akses yang dapat dilakukan pengguna pada saat mengakses suatu fitur pada Bot Telegram Jala. Oleh karena itu, Bot Telegram Jala perlu menanggapi aksi dan *input* pengguna dengan sebuah *handler* berupa pesan yang berguna sebagai indikasi terkait aksi atau *input* yang tidak valid. Adapun contoh *handler* pesan yang menanggapi aksi dan *input* pengguna yang tidak valid dapat dilihat pada Gambar 3.35 dan 3.36.



Gambar 3.35 Contoh *handler* pesan untuk menanggapi *input* yang tidak valid



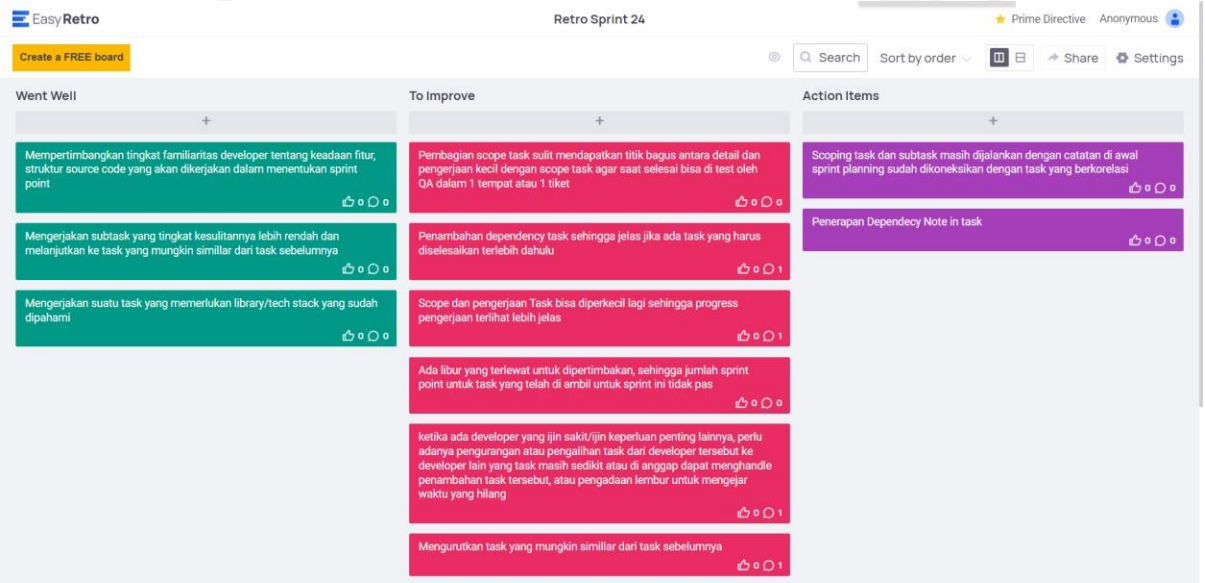
Gambar 3.36 Contoh *handler* pesan untuk menanggapi aksi di luar konteks pencatatan data

3.6 Sprint Retrospective

Pada hari terakhir dalam satu *sprint*, *sprint retrospective* dilakukan untuk mendiskusikan terkait evaluasi kinerja individu, membahas kendala yang terjadi saat implementasi, dan hal-hal yang dapat ditingkatkan untuk meningkatkan efektivitas implementasi pada *sprint* berikutnya (Grebić & Stojanović, 2021). *Sprint retrospective* dihadiri oleh seluruh anggota tim untuk berdiskusi mencari solusi dari kendala-kendala yang dimiliki baik oleh anggota lain maupun diri sendiri.

Pada proyek pengembangan Bot Telegram Jala, penulis ikut berkontribusi dalam *sprint retrospective* yang dilakukan di Jala. *Sprint retrospective* dilakukan secara daring dengan

menggunakan aplikasi *Google Meet*, serta aplikasi *EasyRetro* digunakan untuk mendapatkan terkait hal-hal yang ingin disampaikan dalam satu kartu. Para anggota tim kemudian membuat kartu sebanyak yang diperlukan, kartu tersebut berisikan kendala dan hal yang perlu ditingkatkan untuk *sprint* berikutnya. Kartu-kartu yang sudah terkumpul kemudian dibahas satu per satu untuk menemukan masing-masing solusi pada setiap masalah yang diangkat. Selain membahas kendala dan hal yang ditingkatkan, para anggota tim juga perlu menulis hal-hal yang telah membantu mereka dalam mengerjakan suatu tugas untuk dipertahankan pada *sprint* berikutnya. Tampilan aplikasi *EasyRetro* yang digunakan Jala Tech untuk melakukan *sprint retrospective* dapat dilihat pada Gambar 3.37.



Gambar 3.37 Aplikasi *EasyRetro* yang digunakan pada *sprint retrospective*

BAB IV
REFLEKSI PELAKSANAAN MAGANG

4.1 Relevansi Akademik

Saat melaksanakan magang di Jala Tech, terdapat beberapa teori yang telah dipelajari oleh penulis yang justru penerapannya memerlukan adaptasi dikarenakan kondisi yang kurang ideal sesuai dengan teori yang diacu. Sehingga hal ini menjelaskan bahwa dalam penerapan teori yang dipelajari selama masa perkuliahan, tidak semuanya selalu berjalan sesuai dengan luaran teori yang ideal. Cukup banyak sekali hal-hal yang terjadi di luar perkiraan, contohnya hal-hal yang tidak terduga dalam mengerjakan suatu proyek menggunakan metode pengembangan *scrum*. Hal-hal tidak terduga lainnya yang terjadi akan dijelaskan oleh penulis di bawah ini.

4.1.1 Penerapan Metode Pengembangan *Scrum*

Selama pelaksanaan magang di Jala, metode pengembangan *scrum* diterapkan pada setiap proyek pengembangan yang ada. Metode pengembangan *scrum* digunakan karena metode pengembangan ini bersifat cepat dan juga adaptif terhadap perubahan kebutuhan sistem yang terjadi selama dalam masa *sprint*. Menurut Grebić & Stojanović (2021), *Scrum* secara teori memuat lima aktivitas, yaitu *sprint planning*, *sprint*, *sprint review*, *sprint retrospective*, dan *daily scrum*. Aktivitas tersebut merupakan aktivitas-aktivitas yang penting karena keperluan-keperluan dalam aktivitasnya menyangkut pada kesuksesan produk hasil implementasi. Pada proyek pengembangan Bot Telegram Jala, terdapat empat anggota di dalamnya yakni satu orang *product manager*, satu orang *product designer*, satu orang *developer*, dan satu orang *quality assurance*. Sehingga struktur dapat dikatakan belum optimal, karena dalam sebuah tim proyek setidaknya harus ada satu orang yang profesional dalam setiap perannya (Grebić & Stojanović, 2021). Pada proyek ini, penulis bekerja sendirian sebagai pengembang sehingga tidak ada satu orang yang profesional untuk mengarahkan penulis dalam mengerjakan tugas proyek Bot Telegram Jala

Sprint planning di Jala Tech membahas terkait perencanaan untuk pembagian tugas-tugas yang perlu dikerjakan sesuai pada *product backlog* yang diperoleh dari *product owner*. *Sprint planning* dilakukan di awal *sprint*, pembahasan pada *sprint planning* di Jala Tech yakni membahas tugas-tugas yang telah dikonversi dari *product backlog*, bobot pengerjaan setiap

tugas, perencanaan pengerjaan tugas sesuai prioritas, dan pembagian tugas kepada setiap anggota tim oleh *product manager*. Namun kenyataannya, *product backlog* yang seharusnya transparan kepada pengembang justru dibatasi sehingga pengembang hanya dapat memperkirakan penyelesaian proyek sesuai konversi *product backlog* oleh *product manager*. Alhasil penulis yang berperan sebagai *developer* cukup kesulitan dalam menentukan prioritas fitur serta menentukan ekspektasi *progress* pengembangan Bot Telegram Jala dalam satu *sprint*.

Daily scrum merupakan aktivitas pertemuan terhadap anggota tim proyek untuk membahas perkembangan implementasi dan rencana aktivitas yang akan dilakukan di hari tersebut. Seperti namanya, *daily scrum* diadakan setiap hari sebelum anggota tim memulai bekerja pada pekerjaan masing-masing. Di Jala Tech, aktivitas *daily scrum* digabung dengan seluruh proyek yang berjalan sehingga aktivitas ini yang seharusnya dilakukan setidaknya selama tiga puluh menit menjadi berjam-jam karena banyaknya individu yang perlu menyampaikan *progress* pada proyek masing-masing. Hal ini menjadi memakan waktu pengembangan sehingga tugas-tugas yang dimiliki penulis menjadi terhambat.

Sprint review merupakan aktivitas yang membahas terkait pemeriksaan spesifikasi produk dan meninjau fitur-fitur yang telah diimplementasi. Pada tahapan ini, keputusan terkait penambahan *sprint* diambil dengan mengukur ekspektasi masa penyelesaian produk. Apabila masih terdapat beberapa fitur yang belum diimplementasi serta adanya tambahan kebutuhan sistem terhadap spesifikasi produk, maka durasi pengembangan akan diperpanjang dan *sprint* akan diinisialisasi ulang oleh *product manager*. Di Jala Tech, *sprint review* dijadikan aktivitas untuk meninjau pengujian produk serta menentukan prioritas pengerjaan suatu tugas atau fitur berdasarkan efektivitas pengembangan yang dilakukan pada saat implementasi.

Supaya menjadi pembelajaran, diperlukan edukasi dan praktik pengembangan dengan *framework scrum* pada setiap mata kuliah yang memiliki proyek pengembangan. Karena *framework scrum* sudah sangat umum digunakan di beberapa industri IT, serta banyak sekali manfaat yang bisa diambil dengan memanfaatkan *scrum*. Manfaat tersebut menurut penulis yaitu melatih kolaborasi, melatih kreativitas dalam menyusun strategi untuk menyelesaikan suatu tugas, dan masih banyak lagi manfaat lainnya.

4.1.2 Pengujian Sistem Secara Manual

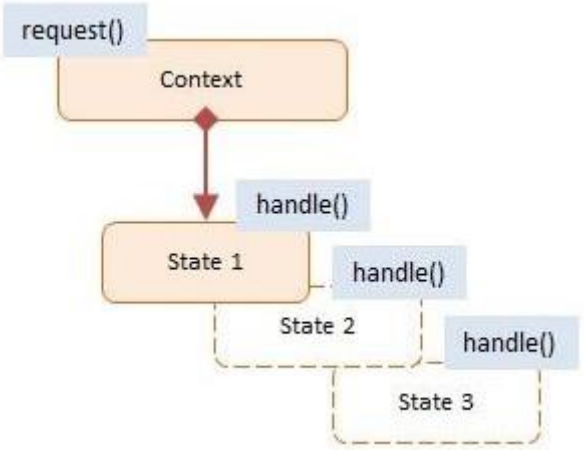
Pada proyek pengembangan Bot Telegram Jala, pengujian dilakukan oleh *quality assurance* dengan menguji bot secara manual dan mandiri tanpa menggunakan bantuan

teknologi. Hal ini dikarenakan kurangnya teknologi untuk menunjang pengujian otomatis terhadap sistem *chatbot* Telegram. Dengan penerapan sistem bekerja dari rumah atau WFH, penulis perlu berkoordinasi secara daring dengan *quality assurance* terkait batasan pengujian yang dilakukan. Namun kenyataannya, koordinasi secara daring tidak efektif karena banyaknya kendala atau hambatan eksternal yang menyebabkan koordinasi menjadi terganggu.

Selama masa perkuliahan, penulis mempelajari pengujian otomatis pada mata kuliah Pengujian Perangkat Lunak. Penulis mempelajari langkah-langkah dalam menguji sistem aplikasi yang berbasis *web* dan *mobile*, serta mempelajari cara menerapkan pengujian otomatis di dalamnya. Sehingga pada kasus yang dialami penulis, penulis tidak bisa merasakan hasil pengujian yang dilakukan secara otomatis pada saat melaksanakan magang di Jala Tech.

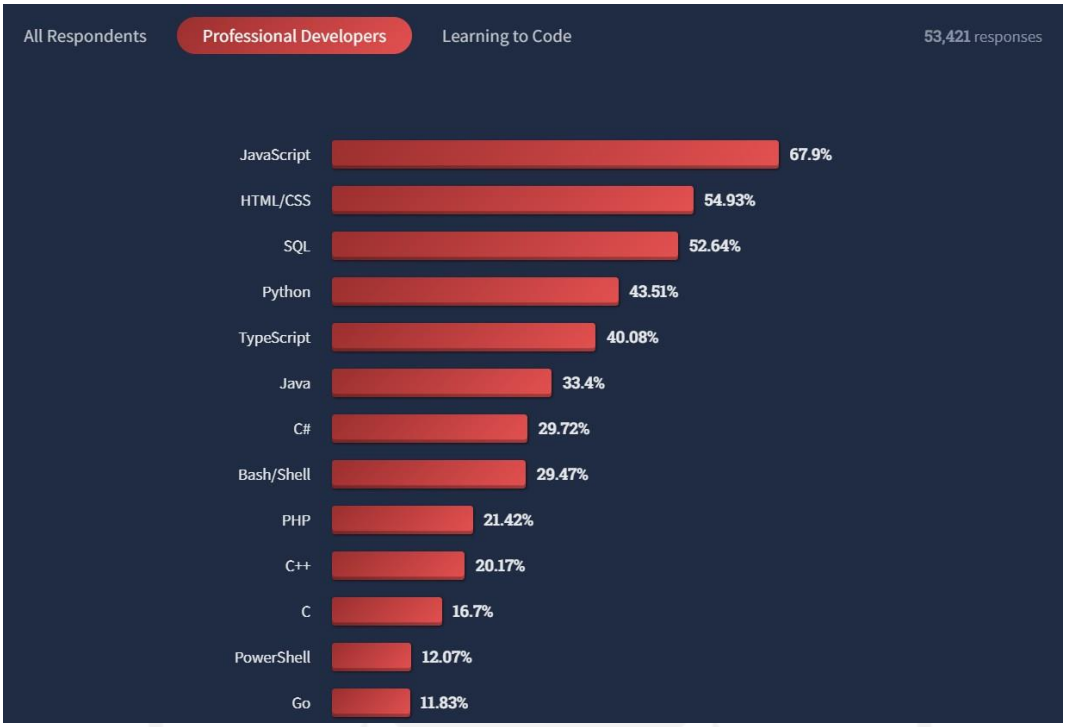
4.1.3 Penerapan *State Design Pattern* Pada Bot Telegram Jala

Sesuai dengan teori yang dikemukakan sebelumnya, *state design pattern* dimanfaatkan ketika prosedur pada suatu sistem bergantung kepada aksi pengguna. Pada proyek pengembangan Bot Telegram Jala, telah disebutkan bahwa pola interaksi *chatbot* yang disepakati adalah model percakapan dengan tanya-jawab sehingga interaksi antara bot dengan pengguna dapat berjalan secara interaktif. Kemudian untuk mewujudkan pola interaksi tersebut, dilakukan implementasi bot dengan memanfaatkan *inline keyboard button* sebagai menu bot yang memungkinkan pengguna untuk mengakses suatu fitur tanpa harus mengetik secara mandiri seperti pada pola interaksi *command-based*. Sesuai pada dokumentasi Telegram, implementasi *inline keyboard button* pada Bot Telegram diterapkan dengan memanfaatkan *callback_data* yang dapat diolah dalam *event callback_query*. Maka dari itu, konsep *state design pattern* sesuai dengan bahasa pemrograman *JavaScript* diterapkan pada pengembangan Bot Telegram Jala supaya pengguna hanya dapat mengakses suatu fitur yang diakses dan tidak dapat mengakses fitur lainnya di luar konteks fitur tersebut. Konsep *state design pattern* dengan *JavaScript* dapat diwujudkan dengan pendekatan *switch-case* maupun *if-else* tergantung dari banyaknya *state* serta cara mengolah *state* tersebut. Sejatinya, konsep *state design pattern* dapat disimak pada Gambar 4.1.



Gambar 4.1 Konsep *State Design Pattern* dengan *JavaScript*
Sumber: *DoFactory* (2022)

Pada masa perkuliahan, penulis mengenal bahasa pemrograman *JavaScript* pada mata kuliah Pengembangan Aplikasi Berbasis Web. Pada mata kuliah tersebut, *JavaScript* dikenalkan sebagai "otak" pada struktur aplikasi *web*. Padahal *JavaScript* mampu melakukan pemrograman terhadap banyak jenis aplikasi, seperti aplikasi *mobile* dengan teknologi *React Native*, pemrograman independen dengan *Node.js*, bahkan pada pengembangan gim dapat dilakukan dengan *JavaScript* melalui aplikasi *Unity*. Maka dari itu, diperlukan pengenalan *JavaScript* lebih lanjut karena bahasa pemrograman ini menjadi salah satu tren bahasa pemrograman yang tinggi dibandingkan dengan bahasa pemrograman seperti *Java*, *Python*, dan bahasa pemrograman lainnya. Berikut ini adalah grafik tren perkembangan bahasa pemrograman yang digunakan berdasarkan survei dari *stackoverflow* pada tahun 2022 dapat disimak pada Gambar 4.2.



Gambar 4.2 Grafik bahasa pemrograman terpopuler dalam survey *stackoverflow* 2022
Sumber: (Stack Overflow, 2022)

4.2 Pembelajaran Magang

Magang merupakan salah satu sarana untuk meningkatkan kemampuan dalam mengembangkan serta membentuk karir yang diinginkan. Karena dengan magang, para pemagang akan dihadapkan oleh kasus nyata untuk menerapkan teori-teori yang telah dipelajari selama masa perkuliahan. Penulis mendapatkan banyak pengalaman serta manfaatnya baik dalam segi teknis maupun non teknis, di bawah ini akan dijabarkan beberapa manfaat tersebut.

4.2.1 Teknis

Selama melaksanakan magang di Jala Tech, penulis mendapatkan banyak sekali manfaat terutama pada pembelajaran terkait proses pengembangan suatu produk dalam dunia industri IT serta menerapkan pemrograman dengan *JavaScript*. Manfaat tersebut sebagian besar didapat dengan pembelajaran otodidak, namun bukan berarti pembelajaran yang telah dipelajari selama perkuliahan tidak mendapatkan manfaat sama sekali pada praktik magang.

Pada masa perkuliahan, mata kuliah *Pengembangan Aplikasi Berbasis Web* menarik perhatian penulis dalam dunia pemrograman. Pengembangan *web* menurut penulis sangatlah unik terlebih lagi sebagai *front end* karena harus mengimplementasikan hasil desain serta mengontrol interaksi yang dapat dilakukan pada suatu aplikasi *web*, selain itu *front end* juga

mengelola distribusi data yang didapatkan dari *back-end* untuk diolah menjadi tampilan informasi pada aplikasi *web*. Kemudian proses pengembangan aplikasi dipelajari lagi lebih lanjut pada mata kuliah *Pengembangan Sistem Informasi*. Pada mata kuliah tersebut selain mempelajari proses pengembangan, penulis juga mempelajari perancangan suatu proses bisnis dari sudut pandang perusahaan. Pemahaman penulis terkait proses bisnis suatu perusahaan diperkuat lagi dengan mengikuti magang di Jala Tech, perusahaan yang mengembangkan dan memelihara suatu produk. Untuk manfaat teknis selengkapnya, akan dijabarkan manfaat lainnya di bawah ini.

A. Mendalami Pemahaman Pemrograman Dengan *JavaScript*

Dengan ikut berkontribusi dalam proyek Bot Telegram Jala, penulis mempelajari lebih lanjut terkait pemrograman *JavaScript* secara alami tanpa menggunakan bantuan teknologi seperti *React*, *Vue.js*, dan teknologi lainnya. Bot Telegram Jala dikembangkan dengan *JavaScript* menggunakan *Node.js* yang mewujudkan pemrograman *standalone*, sehingga pemrograman kali ini berbeda dengan pemrograman *web* yang biasanya memanfaatkan *JavaScript*. Penulis berhasil memahami konsep-konsep dasar beserta konsep lanjutan pada *JavaScript* seperti pengolahan data *array* dan *object*, memaksimalkan penggunaan *ternary operator* yang mewujudkan penerapan kondisial seminimal mungkin. Contoh penggunaan manipulasi *array*, *object*, serta *ternary operator* pada *JavaScript* pada proyek Bot Telegram Jala dapat dilihat pada Gambar 4.3.

```
const mappedMeasurements = measurements.map((measurement) => {
    let mappedMeasurementItem = {
        id: measurement.id,
        cycle_id: measurement.cycle_id,
        pond_id: measurement.pond_id,
        farm_id: measurement.farm_id,
        timezone: measurement.timezone,
        measured_date: measurement.measured_date,
        measured_time: measurement.measured_time,
    };
    // Perulangan untuk mengolah parameter yang dimiliki setiap kolam
    for (const [key, value] of Object.entries(measurement)) {
        if (value !== null && !key.includes("anomaly")) {
            if (measurementKeys.indexOf(key) > -1) {
                mappedMeasurementItem[key] = value;
            }
        }
    }
    return mappedMeasurementItem;
});
... // Kode lainnya
return await telegramBot.sendMessage(
    message.chat.id,
    `Pond ID: ${pond_id ? pond_id : '-'}` + // Contoh ternary operator
    `Pond Name: ${pond_name && (pond_name !== '') ? pond_name : '-'}`
);
```

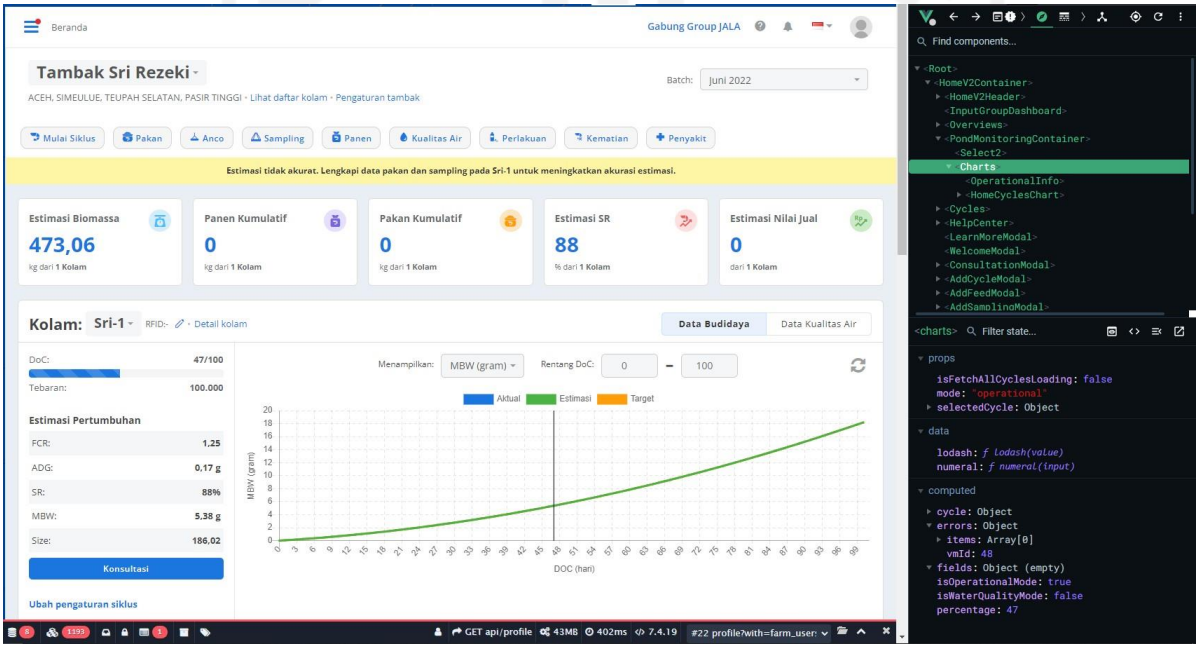
Gambar 4.3 Kode program pengolahan *array*, *object*, dan penggunaan *ternary operator*

B. Memahami Penggunaan *Vue.js* Dalam Pengembangan *Web*

Dalam setiap proyek yang berkaitan dengan aplikasi *web* Jala, penulis perlu mempelajari struktur dan penggunaan *framework vue.js* terlebih dahulu. *Vue.js* adalah sebuah *framework JavaScript* untuk pembangunan *user interface* pada sebuah aplikasi *web* dengan pendekatan pemrograman yang *declarative* dan *component-based* untuk mempermudah pengembangan *user interface* (Vue.js, 2022). Penulis memahami struktur serta fungsi-fungsi yang digunakan dalam mengembangkan *front end* menggunakan *vue.js* pada aplikasi *web* Jala. Pemahaman dasar yang dicakup penulis seperti penerapan *component-based*, struktur dasar pada *vue.js*, serta pengelolaan data antara komponen *parent* dengan *child* menggunakan *props* dan *emit*. Berikut ini merupakan implementasi *component-based* pada aplikasi *web* Jala menggunakan *vue.js* yang dapat disimak pada Gambar 4.4.

Gambar 4.4 *Component-based* pada antarmuka aplikasi *web* Jala

Selain itu, penulis juga memahami *debugging* pada *vue.js* dengan menggunakan Vue Devtools. Vue Devtools merupakan salah satu alat berupa ekstensi pada aplikasi *browser* seperti *Google Chrome* yang berguna untuk melakukan *debugging* pada *vue.js*. Vue Devtools sangat membantu penulis dalam melakukan *debugging* seperti pengecekan data *props*, *data* pada suatu komponen, data *computed*, inspeksi struktur komponen, dan pengecekan *event* yang terjadi. Berikut ini adalah tampilan dari Vue Devtools yang digunakan penulis pada Gambar 4.5.



Gambar 4.5 Penggunaan Vue Devtools untuk *debugging* pada *vue.js*

C. Mendalami Penggunaan Higher Order Array Function Pada JavaScript

Higher order array function merupakan fungsi yang memuat fungsi lain untuk dijadikan sebuah parameter atau mengembalikan fungsi tersebut. Konsep ini biasanya digunakan pada data *array* untuk membantu para pengembangnya dalam memanipulasi *array* menjadi lebih mudah. Fungsi-fungsi yang terdapat pada konsep ini adalah *forEach*, *map, reduce*, dan *filter*. Penulis lebih sering menerapkan *higher order array function* pada proyek Bot Telegram Jala, lebih tepatnya dengan menerapkan *forEach*, *map*, dan *filter*. Karena penulis harus mengolah data yang diberikan serta data yang dikirim dalam bentuk *array of objects* untuk dimanipulasi menjadi respon bot maupun data pada suatu *request*.

4.2.2 Non Teknis

Selain mendapatkan manfaat yang berdampak pada kemampuan teknis penulis, magang juga memberikan manfaat dalam segi pengembangan diri yang diperlukan dalam menjalani aktivitas pekerjaan sehari-hari. Proses pengembangan diri juga bermanfaat untuk membentuk kepribadian yang baik serta mempermudah adaptasi ketika para pekerja mendapatkan suasana atau lingkungan yang baru. Hal tersebut juga berpengaruh dalam menguatkan serta memudahkan koordinasi antara sesama rekan kerja. Selama melaksanakan magang di Jala Tech, penulis mendapatkan banyak manfaat serta pengalaman yang diharapkan dapat membantu penulis dalam mengembangkan diri untuk menjadi pekerja yang profesional. Manfaat tersebut akan dijabarkan di bawah paragraf ini.

A. Mendapatkan Pengalaman Bekerja Sebagai *Front End Developer* Secara Profesional

Pengalaman merupakan salah satu faktor pembelajaran yang paling penting dalam mempelajari sesuatu. Pengalaman juga dapat dijadikan sebagai ajang praktik dari teori-teori yang telah dipelajari sebelumnya. Saat melaksanakan magang di Jala Tech, penulis mendapatkan banyak sekali pengalaman dalam proyek pengembangan aplikasi *web* Jala sebagai *front end developer*. Mulai dari pengalaman dalam menangani kode *front end* dengan *vue.js*, mengimplementasi hasil desain antarmuka dari *product designer* pada aplikasi *web* Jala, hingga pengalaman buruk yang dapat dijadikan pelajaran oleh penulis yaitu kekeliruan dalam melakukan *push* pada *gitlab*. Semua pengalaman tersebut sangat berharga bagi penulis, sehingga dapat dijadikan pelajaran dan mendapatkan wawasan atau hikmah pembelajaran untuk pengembangan diri penulis.

B. Belajar Manajemen Diri

Sebelum melaksanakan magang, penulis tidak memiliki kebiasaan tetap dalam melakukan aktivitas sehari-harinya. Hal ini dapat mempengaruhi hasil aktivitas yang dilakukan karena proses yang dilakukan tidak terstruktur. Saat penulis melaksanakan magang di Jala Tech, penulis mendapatkan *insight* terkait manajemen dari proses pengembangan di Jala yang menggunakan *scrum*. Sehingga hal ini diterapkan oleh penulis dengan mengatur setiap aktivitas yang telah direncanakan sehari-hari supaya penulis dapat fokus pada setiap aktivitas yang dilakukan dan membuahkan hasil yang optimal. Contohnya, mengatur tugas kuliah dan tugas magang dengan memberikan bobot prioritas pada setiap tugasnya.

C. Mendapatkan Perbandingan Pandangan Terhadap Perkuliahan dan Industri IT

Dengan mengikuti perkembangan zaman sekarang yang serba modern, dunia industri turut berevolusi dan bersaing untuk menghasilkan sebuah inovasi baru berdasarkan permasalahan di era sekarang ini. Pada saat melaksanakan magang di Jala Tech, penulis mendapatkan wawasan baru terkait industri yang menyediakan solusi bagi para petambak udang dengan mewujudkan tambak udang digital. Jala Tech merupakan salah satu industri yang menyediakan produk dengan prospek jangka panjang. Penulis yang bekerja sebagai *front end developer*, memiliki tanggung jawab terhadap perkembangan dan pemeliharaan produk digital Jala. Dalam setiap proyek Jala, penulis berorientasi untuk menyelesaikan implementasi dengan baik untuk membantu meningkatkan efektivitas produk-produk Jala. Serta dengan rekan-rekan yang mendukung, penulis memiliki motivasi lebih untuk menyelesaikan tugas dalam setiap proyek pengembangan produk Jala.

Dibandingkan dengan perkuliahan dahulu, dalam setiap tugas proyek yang dikerjakan penulis selalu berorientasi untuk menyelesaikan tugas proyeknya saja. Sehingga tidak ada proses berkelanjutkan yang dapat diterapkan pada tugas proyek perkuliahan. Terlebih jika ada rekan yang kurang mendukung, membuat penulis semakin kehilangan motivasi dalam menyelesaikan tugas-tugas perkuliahan. Maka dari itu, praktik magang diperlukan untuk mendapatkan wawasan serta suasana baru untuk menerapkan teori yang didapatkan pada perkuliahan dengan ikut berkontribusi dalam suatu proyek perusahaan.

D. Mendapatkan Pengalaman Berbicara di Depan Umum

Ketika mendapatkan kesempatan untuk presentasi pada masa perkuliahan, penulis cenderung gugup dan kurang mendapatkan perhatian dari audiensnya. Sehingga setiap kali

terdapat kesempatan untuk presentasi, penulis lebih sering menolak untuk melakukannya. Selain karena gugup, penulis juga cenderung mengalami rasa pesimis yang berlebih. Kurangnya motivasi dalam menjalankan perkuliahan juga mempengaruhi faktor terhambatnya penulis dalam melatih berbicara di depan umum.

Pada saat melaksanakan magang, Jala sering memberikan kesempatan untuk menyampaikan opini maupun presentasi. Penulis sering melatih untuk berbicara dengan umum melalui aktivitas *daily scrum* pada suatu proyek pengembangan, serta presentasi dalam *weekly meeting* yang melibatkan seluruh tim *software* di Jala. Oleh karena itu, sekarang penulis tidak lagi memiliki rasa gugup dan optimis yang berlebihan pada saat diberikan kesempatan untuk berbicara di depan umum. Praktik magang sangat membantu penulis dalam melatih kemampuan tersebut.

E. Penerapan Etika Profesi Islami Dalam Bekerja

Pada masa perkuliahan, penulis mendapatkan mata kuliah *Etika Profesi* yang mempelajari norma dan aturan dalam dunia bekerja. Teori-teori yang didapatkan dapat diterapkan dengan baik oleh penulis, sehingga penulis berhasil menyelesaikan praktik magang dengan baik. Etika dalam bekerja sangat dibutuhkan dalam membangun relasi dan kolaborasi dengan rekan kerja dalam suatu perusahaan, serta melatih kedisiplinan dengan menaati peraturan yang ada.

Menurut Hidayat (2006), terdapat beberapa hal yang patut dan tidak patut dilakukan yang kemudian perlu diingat oleh setiap Muslim dalam mengembangkan karirnya pada dunia kerja. Berdasarkan hal tersebut, beberapa prinsip bekerja sesuai derajat sebagai kaum muslimin diterapkan oleh penulis. Sebagai contoh dalam mengerjakan setiap tugas yang diberikan, penulis selalu menyelesaikan tugas tersebut dengan ikhlas dan profesional. Kemudian nilai kejujuran dan tanggung jawab dalam menyelesaikan setiap tugas-tugas juga perlu diprioritaskan sebagai ajang untuk mencari ridho Allah SWT. Dengan begitu, penulis berharap Allah SWT selalui meridhoi dan memudahkan penulis dalam melaksanakan magang di Jala Tech.

F. Melatih Kemampuan Beradaptasi Pada Lingkungan Baru

Pada saat melaksanakan magang di hari pertama, lingkungan bekerja dalam suatu perusahaan merupakan hal yang baru saat dijumpai penulis. Tentunya setiap perusahaan memiliki budaya atau kultur yang berbeda-beda, maka dari itu hal tersebut menjadi tantangan

pertama bagi penulis untuk beradaptasi supaya terbiasa dalam menjalankan aktivitas di Jala Tech. Sejatinya, manusia harus dapat beradaptasi ketika menjumpai suatu lingkungan yang baru. Sehingga adaptasi merupakan salah satu faktor yang penting dalam keberhasilan pelaksanaan magang.

Pentingnya adaptasi diperlukan untuk menyesuaikan diri penulis dengan rekan-rekan yang menganut suku dan budaya yang berbeda. Adaptasi juga diperlukan untuk mempermudah komunikasi dengan rekan kerja. Selain itu, penulis juga perlu beradaptasi dalam segi teknis. Karena teknologi-teknologi yang digunakan untuk menunjang aktivitas di Jala Tech tentunya asing bagi penulis, sehingga penulis perlu mempelajari teknologi yang digunakan di Jala Tech.

BAB V
PENUTUP

5.1 Kesimpulan

Berdasarkan permasalahan yang diangkat pada saat melaksanakan magang, pengembangan Bot Telegram Jala Tech dengan penerapan *state design pattern* dan strategi *caching* menggunakan Redislabs berhasil menjadi solusi alternatif untuk mengolah data kolam pada *smart farm* Jala. Bot Telegram Jala Tech juga dapat diproyeksikan sebagai produk digital baru selain aplikasi Jala yang berbasis *web* dan *mobile*. Adapun poin-poin lainnya yang dapat diambil dari pengembangan Bot Telegram Jala Tech adalah sebagai berikut:

a. Setelah melakukan *sprint planning* pada metode pengembangan *scrum* yang diterapkan. Disepakati Bot Telegram Jala perlu menjawab permasalahan terkait pengolahan data kolam disaat pengguna memiliki kendala dalam mengakses aplikasi Jala. Saat mendiskusikan *sprint backlog*, Bot Telegram Jala harus memuat fitur-fitur seperti integrasi akun Jala, pencatatan data pada semua jenis data kolam beserta pemeriksaan data hasil pencatatan, informasi profil pengguna, dan informasi daftar kolam pada tambak yang dimiliki oleh pengguna.
b. Setelah melakukan tahap implementasi, Bot Telegram Jala berhasil dikembangkan dengan pola interaksi percakapan tanya-jawab interaktif dan menggunakan sistem menu *inline keyboard button* yang tersedia pada Telegram. Karena bot memuat pola interaksi yang interaktif, hal ini dapat diwujudkan dengan konsep *state design pattern* pada bahasa pemrograman *JavaScript* dengan memanfaatkan *switch-case* dan *if-else* pada pemrogramannya. Fitur-fitur yang telah disepakati pada tahap *planning* juga berhasil diimplementasikan pada Bot Telegram Jala.
c. Pada saat pengujian yang telah dilakukan oleh *quality assurance* Jala, penulis berkoordinasi untuk menyepakati konteks pengujian yang akan dilakukan. Pengujian pada Bot Telegram Jala menggunakan teknik pengujian *blackbox* yang bersifat untuk menguji fungsionalitas utama pada sistem. Penulis langsung memperbaiki *bug* apabila terdapat kesalahan pada Bot Telegram Jala yang telah diimplementasikan saat pengujian. Pada akhirnya, Bot Telegram Jala dapat dijalankan sesuai prosedur pada setiap fiturnya dengan baik. Dengan begitu Bot Telegram Jala siap untuk diterapkan pada kasus nyata dan dirilis untuk menjadi produk digital baru dari Jala Tech.

5.2 Saran

Adapun saran yang dapat diberikan oleh penulis terkait pengembangan Bot Telegram Jala sebagai berikut:

a. Perlu diterapkan *clean code* pada beberapa fitur jika ada pengembangan terkait Bot Telegram Jala kedepannya. Karena terdapat beberapa fitur di mana struktur kode programnya belum tertulis dengan rapi. Hal tersebut karena pengembangan Bot Telegram Jala yang harus diselesaikan dalam waktu dekat untuk menguji kasus dari permasalahan yang diangkat.
b. Pada fitur pemeriksaan data hasil pencatatan, tanggal pemeriksaan tidak bersifat fleksibel di mana pemeriksaan data harus mengikuti hari terkini. Maka dari itu, menurut penulis fitur ini harus memberikan fleksibilitas kepada pengguna yang hendak memeriksa data hasil pencatatan pada tanggal tertentu.
c. Perlu dilakukan riset terkait peningkatan performa Bot Telegram Jala dalam memproses pesan dari pengguna. Karena menurut penulis, kode program yang telah dituliskan masih dapat dioptimalkan sehingga dapat mempengaruhi performa Bot Telegram untuk memproses pesan atau *request* dengan lebih cepat.

DAFTAR PUSTAKA

Alam, S. I., Wazir, S., khalique, A., & Hassan, S. I. (2017). Data Cache with Distributed Cache: A Design Approach. *International Journal of Computer Science and Engineering*, *4*(6), 17–23. https://doi.org/10.14445/23488387/IJCSE-V4I6P104

Alfaiz, F. Z., & Maryam, M. (2021). IMPLEMENTATION TELEGRAM CHAT BOT ON STUDENT ORIENTATION PERIOD REGISTRATION SYSTEM FOR EFFICIENCY OF DATA MANAGEMENT. *Jurnal Teknik Informatika (Jutif)*, *2*(2), 85–93. https://doi.org/10.20884/1.jutif.2021.2.2.56

Anshori, M., Widya, A., Airlangga, P., & Hasbullah, K. H. A. W. (2020). PENGEMBANGAN TELEGRAM BOT ENGINE MENGGUNAKAN METODE WEBHOOK DALAM RANGKA PENINGKATAN WAKTU LAYANAN E-GOVERNMENT. *SAINTEKBU: Jurnal Sains Dan Teknologi*, *12*(2).

AWS Amazon. (2022). Redis: Penyimpanan data di dalam memori yang cepat dengan sumber terbuka untuk digunakan sebagai database, cache, perantara pesan, dan antrean. Retrieved July 11, 2022, from https://aws.amazon.com/id/redis/

Fodor Paul. (2020). *Behavioral Design Patterns, Fundamentals of Software Development*. New York: Stony Brook University. Retrieved from http://www.cs.stonybrook.edu/~cse316

Gde Sastrawangsa. (2017). Pemanfaatan Telegram Bot Untuk Automatisasi Layanan Dan Informasi Mahasiswa Dalam Konsep Smart Campus. In STMIK STIKOM Bali (Ed.), *Konferensi Nasional Sistem & Informatika* . Bali. Retrieved from https://knsi.stikom-bali.ac.id/index.php/eproceedings/issue/view/1

Grebić, B., & Stojanović, A. (2021). Application of the Scrum Framework on Projects in IT Sector. *European Project Management Journal*, *11*(2), 37–46. https://doi.org/10.18485/epmj.2021.11.2.4

Hidayat, I. S. (2006). ETOS KERJA SESUAI DENGAN ETIKA PROFESI ISLAM. *MIMBAR: Jurnal Sosial Dan Pembangunan*, *XXII*(1), 130–142. Retrieved from https://ejournal.unisba.ac.id/index.php/mimbar/article/view/204

Jala Tech. (2022). Tentang Jala Tech. Retrieved July 6, 2022, from https://jala.tech/about/

Krunal Bhavsar, Vrutik Shah, & Samir Gopalan. (2020). Scrum: An Agile Process Reengineering in Software Engineering. *International Journal of Innovative Technology and Exploring Engineering*, *9*(3), 840–848. https://doi.org/10.35940/ijitee.c8545.019320

Nobari, A. D., Reshadatmand, N., & Neshati, M. (2017). Analysis of telegram, an instant messaging service. *International Conference on Information and Knowledge Management, Proceedings*, *Part F131841*, 2035–2038. Association for Computing Machinery. https://doi.org/10.1145/3132847.3133132

Redis. (2022). Redis Enterprise Cloud Documentation. Retrieved July 8, 2022, from https://docs.redis.com/latest/rc/

Sandhika Jaya, T., Studi Manajemen Informatika, P., Ekonomi dan Bisnis, J., & Negeri Lampung JlnSoekarno, P. (2018). Pengujian Aplikasi dengan Metode Blackbox Testing Boundary Value Analysis (Studi Kasus: Kantor Digital Politeknik Negeri Lampung). *Jurnal Informatika: Jurnal Pengembangan IT (JPIT)*, *03*(02).

Scrum.org. (2020). What is Scrum? Retrieved July 12, 2022, from https://www.scrum.org/resources/what-is-scrum

Setiaji, H., & Paputungan, I. v. (2018). Design of Telegram Bots for Campus Information Sharing. *IOP Conference Series: Materials Science and Engineering*, *325*(1). Institute of Physics Publishing. https://doi.org/10.1088/1757-899X/325/1/012005

Stack Overflow. (2022). Stack Overflow Developer Survey 2022. Retrieved July 12, 2022, from https://survey.stackoverflow.co/2022/#technology-most-popular-technologies

Sutikno, T., Handayani, L., Stiawan, D., Riyadi, M. A., & Subroto, I. M. I. (2016). WhatsApp, viber and telegram: Which is the best for instant messaging? *International Journal of Electrical and Computer Engineering*, *6*(3), 909–914. https://doi.org/10.11591/ijece.v6i3.10271

Syaefulloh, A., & Yusrizal, F. (2019). *Implementasi Dan Analisa Performa DataBase Cache Redis*. Retrieved from https://www.researchgate.net/publication/338195534

Telegram. (2020). Bots: an introduction for developers. Retrieved June 10, 2022, from https://core.telegram.org/bots

Wisnu Alfiansyah, M., Gede Putu Wirarama Wedashwara, I., & Zafrullah Mardiansyah, A. (2021). Implementasi IoT Untuk EWS Menggunakan Metode DES Model Holt Pada Tambak Udang Vaname (Implemented of IoT for EWS Using the Holt Model Des Method in Vaname Shrimp Ponds). *Journal of Computer Science and Informatics Engineering*, *5*(1). Retrieved from http://jcosine.if.unram.ac.id/

LAMPIRAN

A. Sertifikat Magang

DocuSign Envelope ID: 979CBB6C-9898-4F70-9C32-23696545B353

JALA

SERTIFIKAT MAGANG

Magang: 004/VI/JALA/2022

Sertifikat ini diberikan kepada

Mohammad Reza Ali Firdaus

Sebagai penghargaan atas perannya sebagai
Front-End Developer Intern
pada 1 September 2021 sampai 1 Maret 2022 dengan hasil SANGAT BAIK.

Reza berpartisipasi dalam membangun Telegram chat bot dengan menggunakan bahasa pemrograman Node.js, Serverless Framework dan mengembangkan front-end menggunakan Vue.js

LIRIS MADUNINGTYAS
CHIEF EXECUTIVE OFFICER
JALA TECH

www.jala.tech
contact@jala.tech

B. Daily Scrum