LAMPIRAN

Lampiran 1. Foto Kegiatan

Lampiran 2. Lampiran Full Source Code

1. Index.ts

```ts
import { buyCommandHandler } from "../buy/buy-command-handler";
import { caraOrderCommandHandler } from "../cara-order/cara-order-command-
handler";
import { userMenuCommandHandler } from "../menu/user-menu-command-handler";
import { productCommandHandler } from "../product/product-command-handler";
import { Command } from "./command-type";

/**
 * List all available user commands.
 * */
export const userCommandList: Command[] = [
  {
    name: "menu",
    description: "Menampilkan semua command user yang tersedia pada bot ini",
    shortDescription: "Menampilkan menu",
    logo: "📌",
    handler: userMenuCommandHandler,
  },
  {
    name: "caraorder",
    description: "untuk menampilkan video tutorial cara order",
    shortDescription: "Video cara order",
    logo: "🎥",
    handler: caraOrderCommandHandler,
  },
  {
    name: "produk",
    description: "Menampilkan produk apa saja yang ada di bot",
    shortDescription: "Lihat produk",
    logo: "📦",
    handler: productCommandHandler,
  },
  {
    name: "buy",
    description: "Melakukan pemesanan dengan cara /buy codeProduk",
    shortDescription: "Melakukan pemesanan",
    logo: "🛒",
    handler: buyCommandHandler,
  },
];
```

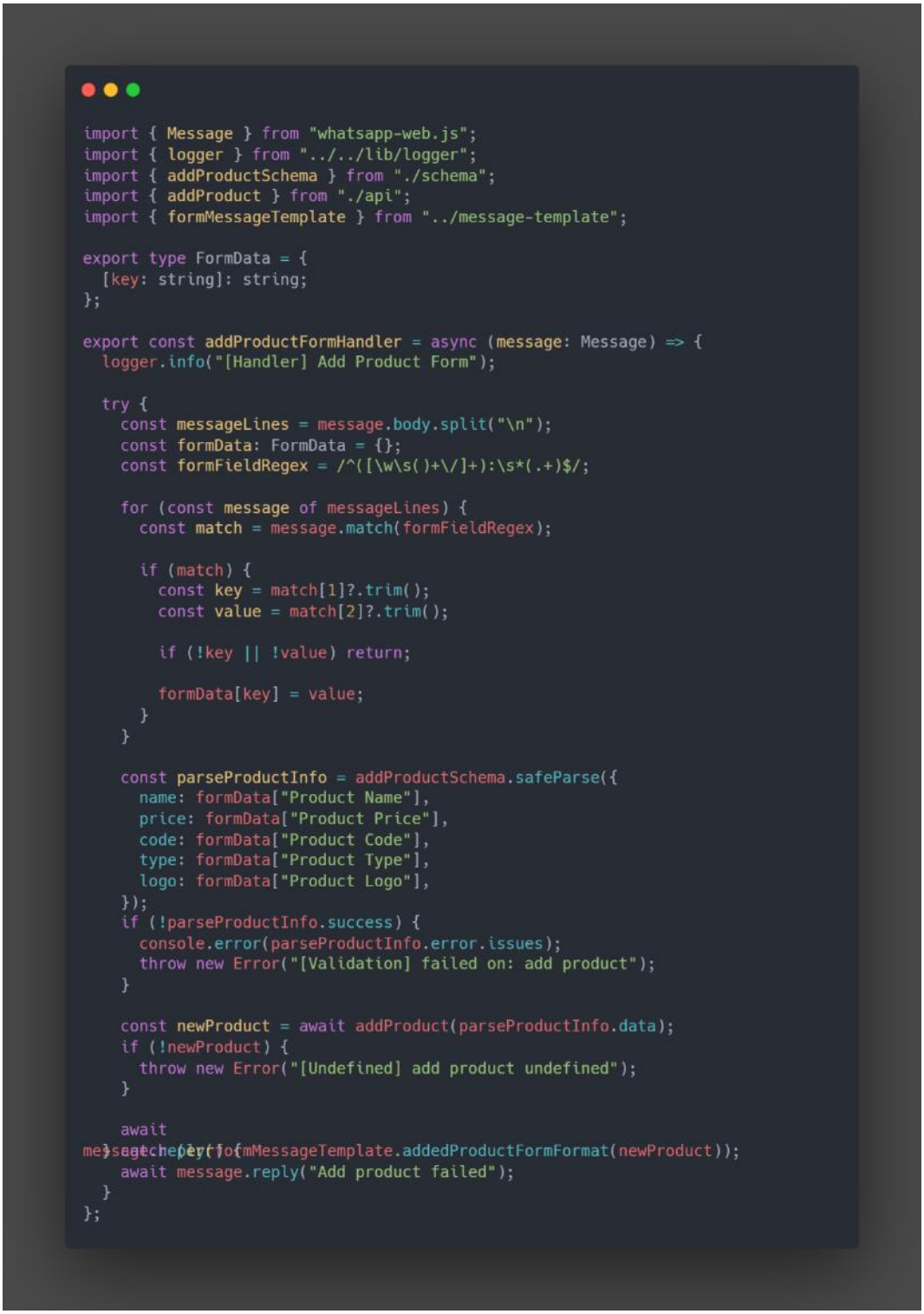
```
    /**
     * Default Message Handler
     * every message that doesn't get passed to condition above.
     * */
    const chat = await message.getChat();
    await chat.sendStateTyping();
    await chat.sendMessage(
      "Selamat datang di toko joki Fatkhi, silahkan lihat semua perintah yang tersedia
dibawah!",
    );
    if (role === "ADMIN") {
      await adminMenuCommandHandler(chat, []);
    } else {
      await userMenuCommandHandler(chat, []);
    }
  } catch (e) {
    if (e instanceof Error) {
      logger.error(On Message Handler error);
      logger.error({
        error: e.name,
        detail: e.message,
      });

      const chat = await message.getChat();
      chat.sendMessage("Bot mengalami kesalahan, tolong hubungi admin!");
    }
  }
};
```

2. User-menu-command-handler.ts

```
import { delayTime } from "../../lib/utils";
import { CommandHandler } from "../command/command-type";
import { userCommandList } from "../command/user-commands";

export const userMenuCommandHandler: CommandHandler = async (chat) => {
  await chat.sendStateTyping();
  await delayTime();

  const messages: string[] = [];

  messages.push(┌─ [ *Welcome To Joki Fatkhi* ] ─\n);
  messages.push(┆ \n);
  userCommandList.map((command) => {
    messages.push(
      ┆ ${command.logo ?? ""} */${command.name}* →
${command.shortDescription}\n,
    );
  });
  messages.push(┆ \n);
  messages.push(└──────────────\n\n);
  messages.push(KETIK PERINTAH DIATAS UNTUK MENGGUNAKAN BOT\n\n);
  messages.push(*CONTOH: /caraorder*);

  await chat.sendMessage(messages.join(""));
};
```

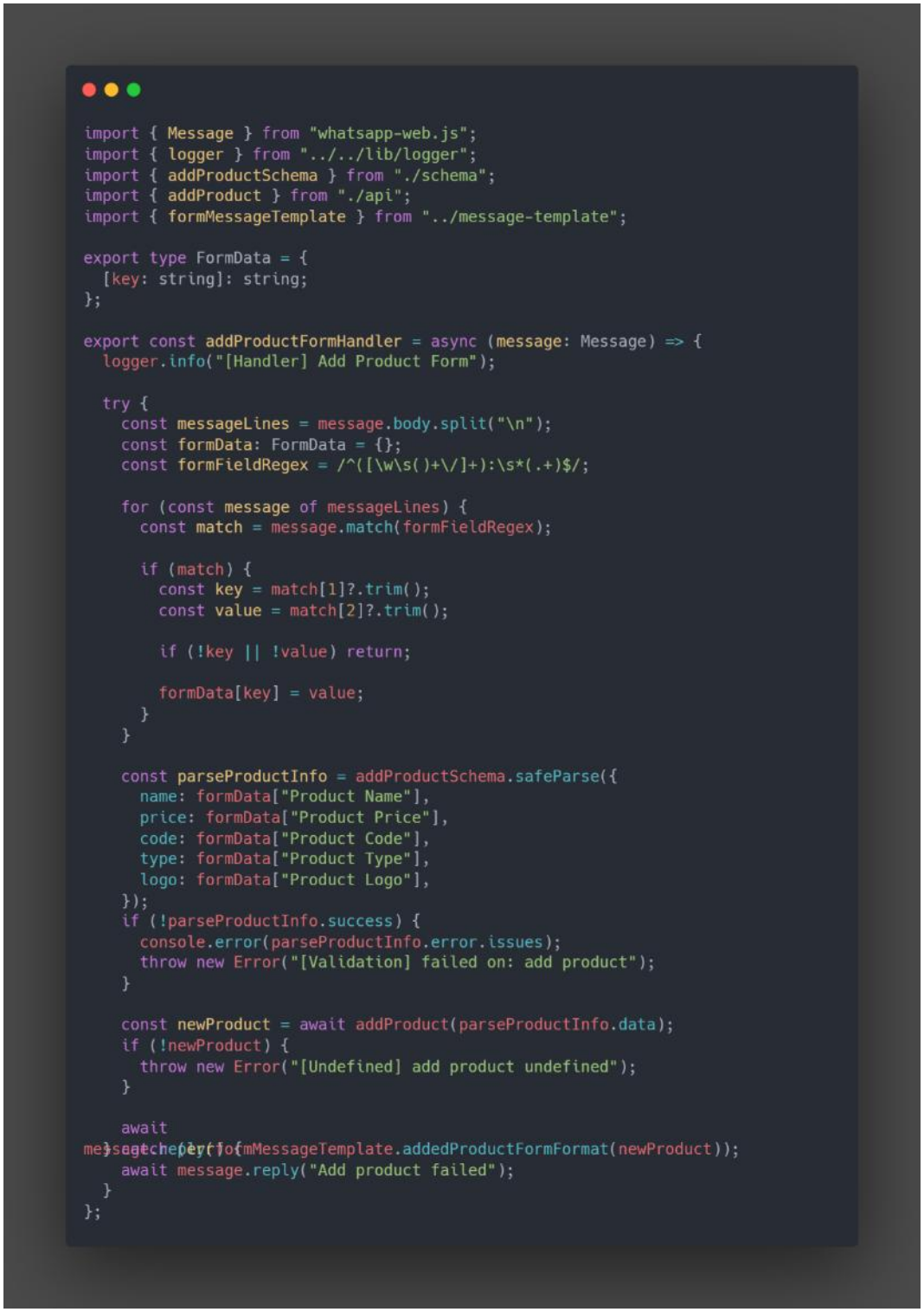
3. Admin-menu-command-handler.ts

```ts
import { delayTime } from "../../lib/utils";
import { adminCommandList } from "../command/admin-commands";
import { CommandHandler } from "../command/command-type";

/**
 * (ADMIN COMMAND)
 * Show available commands for admin
 * */
export const adminMenuCommandHandler: CommandHandler = async (chat) => {
  await chat.sendStateTyping();
  await delayTime();

  const messages: string[] = [];

  messages.push(┌─ [ *Welcome Boss!* ] ─\n);
  messages.push(┆ \n);
  adminCommandList.map((command) => {
    messages.push(
      ┆ ${command.logo ?? ""} */${command.name}* →
${command.shortDescription}\n,
  });
  messages.push(┆ \n);
  messages.push(└──────────────\n\n);
  messages.push(KETIK PERINTAH DIATAS UNTUK MENGGUNAKAN BOT\n\n);
  messages.push(*CONTOH: /konfirmasi nomorPelanggan*);

  await chat.sendMessage(messages.join(""));
};
```



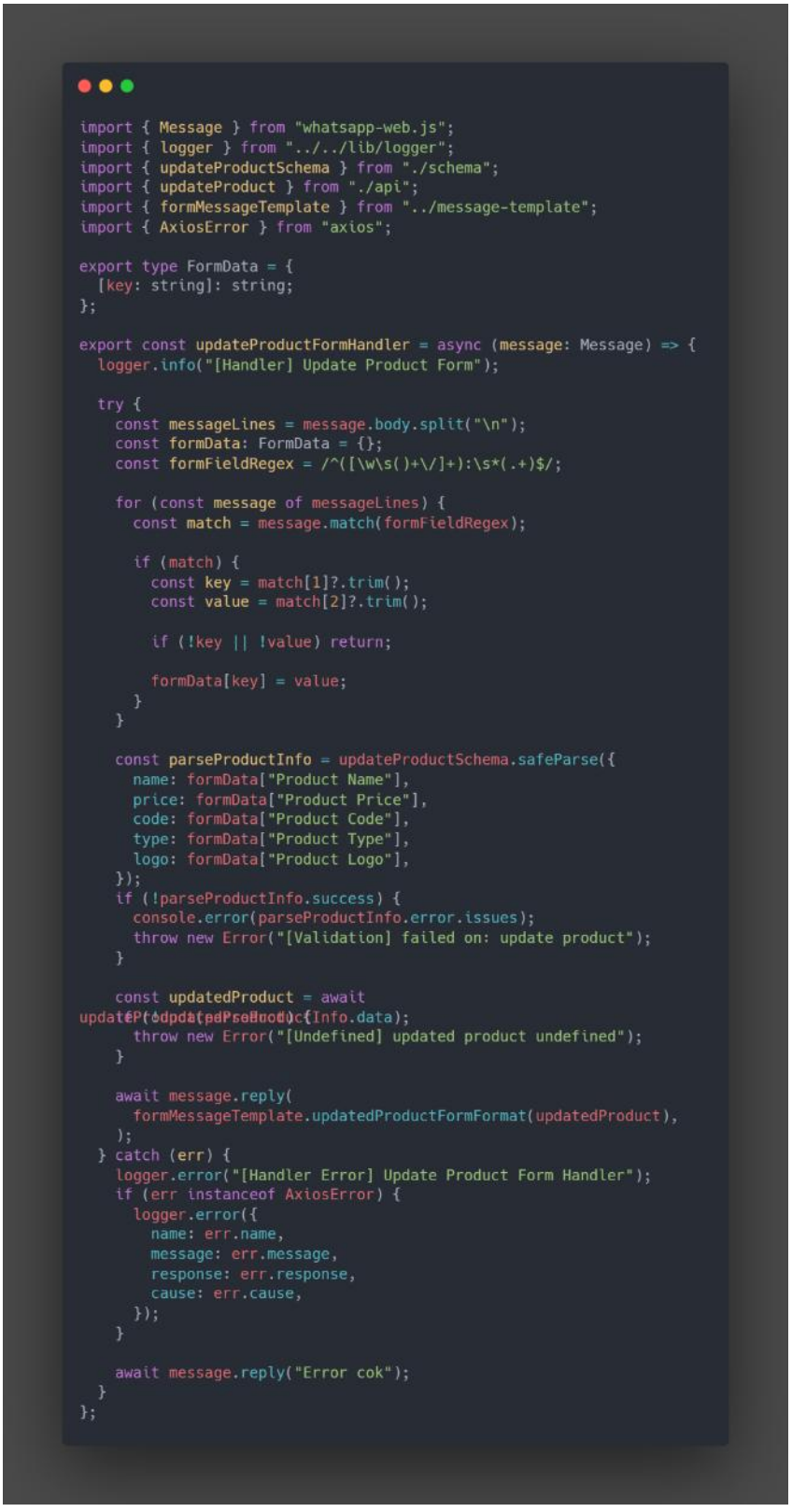
```
import { Message } from "whatsapp-web.js";
import { logger } from "../../lib/logger";
import { addProductSchema } from "./schema";
import { addProduct } from "./api";
import { formMessageTemplate } from "../message-template";

export type FormData = {
  [key: string]: string;
};

export const addProductFormHandler = async (message: Message) => {
  logger.info("[Handler] Add Product Form");

  try {
    const messageLines = message.body.split("\n");
    const formData: FormData = {};
    const formFieldRegex = /^([\w\s()+\/]+):\s*(.+)$/;

    for (const message of messageLines) {
      const match = message.match(formFieldRegex);

      if (match) {
        const key = match[1]?.trim();
        const value = match[2]?.trim();

        if (!key || !value) return;

        formData[key] = value;
      }
    }

    const parseProductInfo = addProductSchema.safeParse({
      name: formData["Product Name"],
      price: formData["Product Price"],
      code: formData["Product Code"],
      type: formData["Product Type"],
      logo: formData["Product Logo"],
    });
    if (!parseProductInfo.success) {
      console.error(parseProductInfo.error.issues);
      throw new Error("[Validation] failed on: add product");
    }

    const newProduct = await addProduct(parseProductInfo.data);
    if (!newProduct) {
      throw new Error("[Undefined] add product undefined");
    }

    await
message.reply(formMessageTemplate.addedProductFormFormat(newProduct));
  } catch (error) {
    await message.reply("Add product failed");
  }
};
```

4. Add-product-form-handler.ts



```ts
import { Message } from "whatsapp-web.js";
import { logger } from "../../lib/logger";
import { addProductSchema } from "./schema";
import { addProduct } from "./api";
import { formMessageTemplate } from "../message-template";

export type FormData = {
  [key: string]: string;
};

export const addProductFormHandler = async (message: Message) => {
  logger.info("[Handler] Add Product Form");

  try {
    const messageLines = message.body.split("\n");
    const formData: FormData = {};
    const formFieldRegex = /^([\w\s()+\/]+):\s*(.+)$/;

    for (const message of messageLines) {
      const match = message.match(formFieldRegex);

      if (match) {
        const key = match[1]?.trim();
        const value = match[2]?.trim();

        if (!key || !value) return;

        formData[key] = value;
      }
    }

    const parseProductInfo = addProductSchema.safeParse({
      name: formData["Product Name"],
      price: formData["Product Price"],
      code: formData["Product Code"],
      type: formData["Product Type"],
      logo: formData["Product Logo"],
    });
    if (!parseProductInfo.success) {
      console.error(parseProductInfo.error.issues);
      throw new Error("[Validation] failed on: add product");
    }

    const newProduct = await addProduct(parseProductInfo.data);
    if (!newProduct) {
      throw new Error("[Undefined] add product undefined");
    }

    await
message.reply(formMessageTemplate.addedProductFormFormat(newProduct));
  } catch (err) {
    await message.reply("Add product failed");
  }
};
```

5. Update-product-form-handler.ts



```ts
import { Message } from "whatsapp-web.js";
import { logger } from "../../lib/logger";
import { updateProductSchema } from "./schema";
import { updateProduct } from "./api";
import { formMessageTemplate } from "../message-template";
import { AxiosError } from "axios";

export type FormData = {
  [key: string]: string;
};

export const updateProductFormHandler = async (message: Message) => {
  logger.info("[Handler] Update Product Form");

  try {
    const messageLines = message.body.split("\n");
    const formData: FormData = {};
    const formFieldRegex = /^([\w\s()+\/]+):\s*(.+)$/;

    for (const message of messageLines) {
      const match = message.match(formFieldRegex);

      if (match) {
        const key = match[1]?.trim();
        const value = match[2]?.trim();

        if (!key || !value) return;

        formData[key] = value;
      }
    }

    const parseProductInfo = updateProductSchema.safeParse({
      name: formData["Product Name"],
      price: formData["Product Price"],
      code: formData["Product Code"],
      type: formData["Product Type"],
      logo: formData["Product Logo"],
    });
    if (!parseProductInfo.success) {
      console.error(parseProductInfo.error.issues);
      throw new Error("[Validation] failed on: update product");
    }

    const updatedProduct = await
updateProduct(parseProductInfo.data);
      throw new Error("[Undefined] updated product undefined");
    }

    await message.reply(
      formMessageTemplate.updatedProductFormFormat(updatedProduct),
    );
  } catch (err) {
    logger.error("[Handler Error] Update Product Form Handler");
    if (err instanceof AxiosError) {
      logger.error({
        name: err.name,
        message: err.message,
        response: err.response,
        cause: err.cause,
      });
    }

    await message.reply("Error cok");
  }
};
```

6. Delete-product.ts

```ts
import z from "zod/v4";
import { CommandHandler } from "../command/command-type";
import { logger } from "../../lib/logger";
import { deleteProduct } from "./api";

/**
 * (ADMIN COMMAND)
 * Update product
 * */
export const deleteProductCommandHandler: CommandHandler = async
( chat,
  args,
) => {
  logger.info("[Handler] Delete Product Hit");

  try {
    const productCode = z.string().parse(args[0]);

    const deletedProduct = await deleteProduct(productCode);
    if (!deletedProduct) {
      throw new Error("Delete product failed");
    }

    await chat.sendMessage(
      Produk dengan kode: ${deletedProduct.code} telah dihapus!,
    );
  } catch (err) {
    logger.error(err);
    await chat.sendMessage("Delete product failed");
  }
};
```