LAPORAN PRAKTIKUM
KONSTRUKSI PERANGKAT BERGERAK

MODUL VII

Aplikasi Bank Transfer dengan Runtime Configuration (Node.js)

Disusun Oleh :

Arwin Nabiel Arrofif

2211104057

S1SE-06-02

Asisten Praktikum :

Muhamad Taufiq Hidayat

Dosen Pengampu :

Riyan Dwi Yulian Prakoso, S.Kom., M.Kom.

PROGRAM STUDI S1 REKAYASA PERANGKAT LUNAK

DIREKTORAT TELKOM KAMPUS PURWOKERTO

2025

BAB I
PENDAHULUAN

A. DASAR TEORI

Node.js adalah lingkungan runtime JavaScript yang dibangun di atas mesin V8 milik Google Chrome. Node.js bersifat open-source dan dirancang untuk membangun aplikasi jaringan yang scalable, terutama aplikasi berbasis server. Dengan pendekatan non-blocking dan event-driven, Node.js memungkinkan pengolahan data secara efisien dan cepat**.**

B. MAKSUD DAN TUJUAN

pembuatan aplikasi *Bank Transfer dengan Runtime Configuration* ini adalah untuk menerapkan konsep konfigurasi runtime dan internationalization (i18n) dalam pengembangan aplikasi Node.js. Aplikasi ini dirancang agar dapat membaca pengaturan dari file konfigurasi eksternal yang fleksibel serta menyesuaikan bahasa antarmuka berdasarkan pilihan pengguna.
Dengan adanya runtime configuration dan i18n, aplikasi menjadi lebih mudah diadaptasi tanpa perlu mengubah kode utama, baik untuk kebutuhan pengguna individu maupun lingkungan pengembangan yang berbeda.

BAB II
IMPLEMENTASI (GUIDED)

config/ BankTransferConfig.js

```
const fs = require('fs');
const path = require('path');

class BankTransferConfig {
  constructor() {
    this.configPath = path.join(__dirname, '../data/bank_transfer_config.json');
    this.defaultConfig = {
      lang: "en",
      transfer: {
        threshold: 25000000,
        low_fee: 6500,
        high_fee: 15000
      },
      methods: ["RTO (real-time)", "SKN", "RTGS", "BI FAST"],
      confirmation: {
        en: "yes",
        id: "ya"
      }
    };
    this.config = this.loadConfig();
  }

  loadConfig() {
    if (fs.existsSync(this.configPath)) {
      const data = fs.readFileSync(this.configPath, 'utf8');
      return JSON.parse(data);
    } else {
      this.saveConfig(this.defaultConfig);
      return this.defaultConfig;
    }
  }

  saveConfig(config) {
    fs.writeFileSync(this.configPath, JSON.stringify(config, null, 2));
  }
}

module.exports = BankTransferConfig;
```

data/ bank_transfer_config.json

```
{
  "lang": "id",
  "transfer": {
    "threshold": 25000000,
    "low_fee": 6500,
    "high_fee": 15000
  },
  "methods": [
    "RTO (real-time)",
    "SKN",
    "RTGS",
    "BI FAST"
  ],
  "confirmation": {
    "en": "yes",
    "id": "ya"
  }
}
```

app/ BankTransferApp.js

```
const readline = require('readline');
const BankTransferConfig = require('../config/BankTransferConfig');

class BankTransferApp {
  constructor() {
    this.config = new BankTransferConfig().config;
    this.rl = readline.createInterface({
      input: process.stdin,
      output: process.stdout
    });
  }

  askQuestion(query) {
    return new Promise(resolve => this.rl.question(query, resolve));
  }

  async run() {
    const lang = this.config.lang;

    const promptAmount = lang === "en" ?
      "Please insert the amount of money to transfer: " :
      "Masukkan jumlah uang yang akan di-transfer: ";

    const amountStr = await this.askQuestion(promptAmount);
    const amount = parseFloat(amountStr);
```

```
        const fee = amount <= this.config.transfer.threshold
          ? this.config.transfer.low_fee
          : this.config.transfer.high_fee;

        const total = amount + fee;

        if (lang === "en") {
          console.log(`Transfer fee = ${fee}`);
          console.log(`Total amount = ${total}`);
        } else {
          console.log(`Biaya transfer = ${fee}`);
          console.log(`Total biaya = ${total}`);
        }

        console.log(lang === "en" ? "Select transfer method:" : "Pilih metode transfer:");
        this.config.methods.forEach((method, idx) => {
          console.log(`${idx + 1}. ${method}`);
        });

        await this.askQuestion(lang === "en" ? "Choose a method (press Enter after): " : "Pilih metode
(tekan Enter setelah): ");

        const confirmationPrompt = lang === "en" ?
          `Please type "${this.config.confirmation.en}" to confirm the transaction: ` :
          `Ketik "${this.config.confirmation.id}" untuk mengkonfirmasi transaksi: `;

        const confirmationInput = await this.askQuestion(confirmationPrompt);

        if (
          (lang === "en" && confirmationInput.trim().toLowerCase() === this.config.confirmation.en)
||
          (lang === "id" && confirmationInput.trim().toLowerCase() === this.config.confirmation.id)
        ) {
          console.log(lang === "en" ? "The transfer is completed" : "Proses transfer berhasil");
        } else {
          console.log(lang === "en" ? "Transfer is cancelled" : "Transfer dibatalkan");
        }

        this.rl.close();
    }
}

module.exports = BankTransferApp;
```

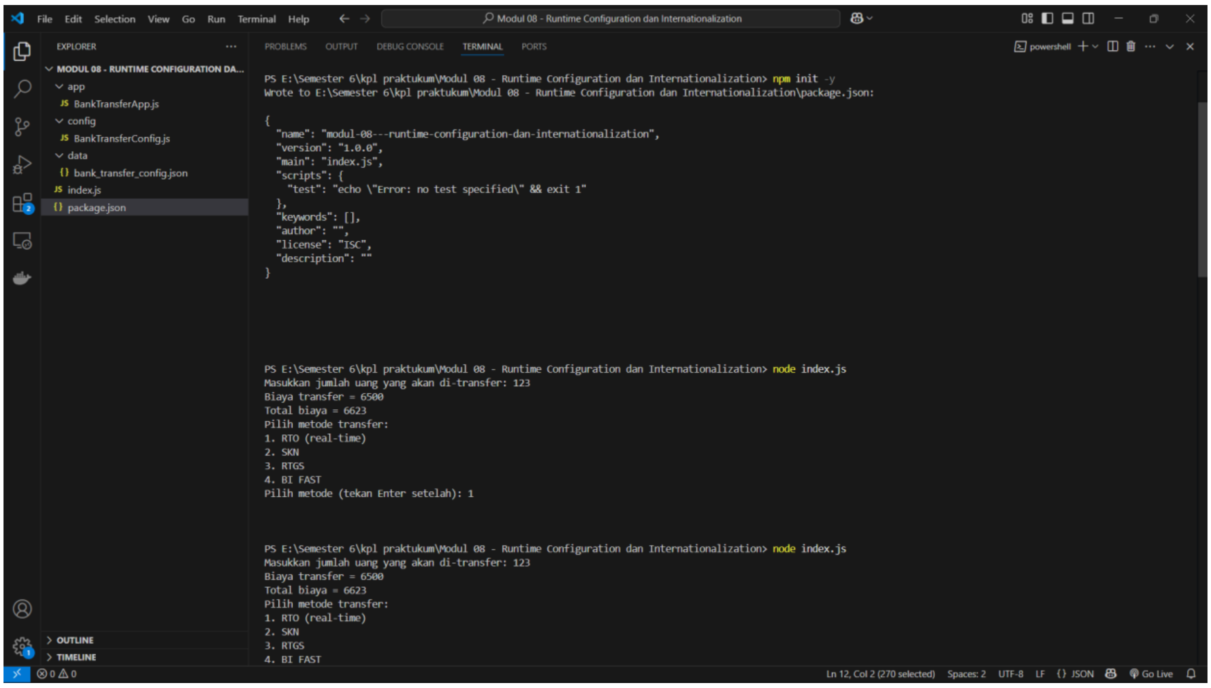
index.js

```
const BankTransferApp = require('./app/BankTransferApp');

const app = new BankTransferApp();
app.run();
```

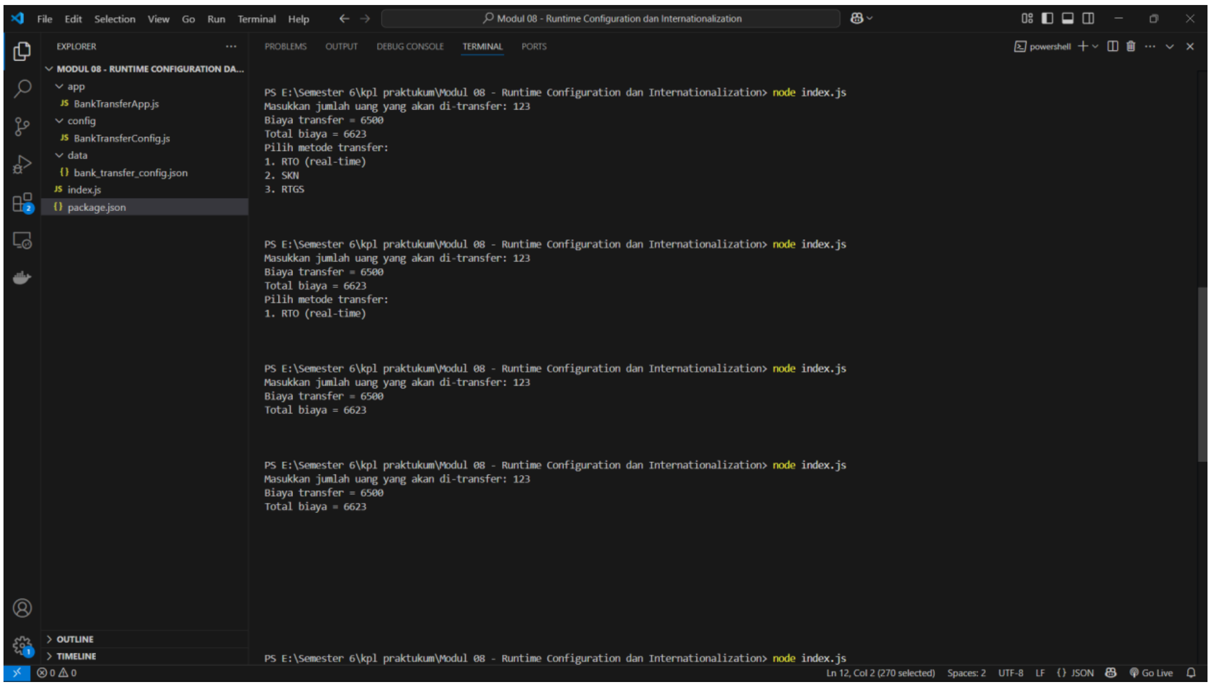
package.json

```
{
  "name": "modul-08---runtime-configuration-dan-internationalization",
  "version": "1.0.0",
  "main": "index.js",
  "scripts": {
    "test": "echo \"Error: no test specified\" && exit 1"
  },
  "keywords": [],
  "author": "",
  "license": "ISC",
  "description": ""
}
```

Output:



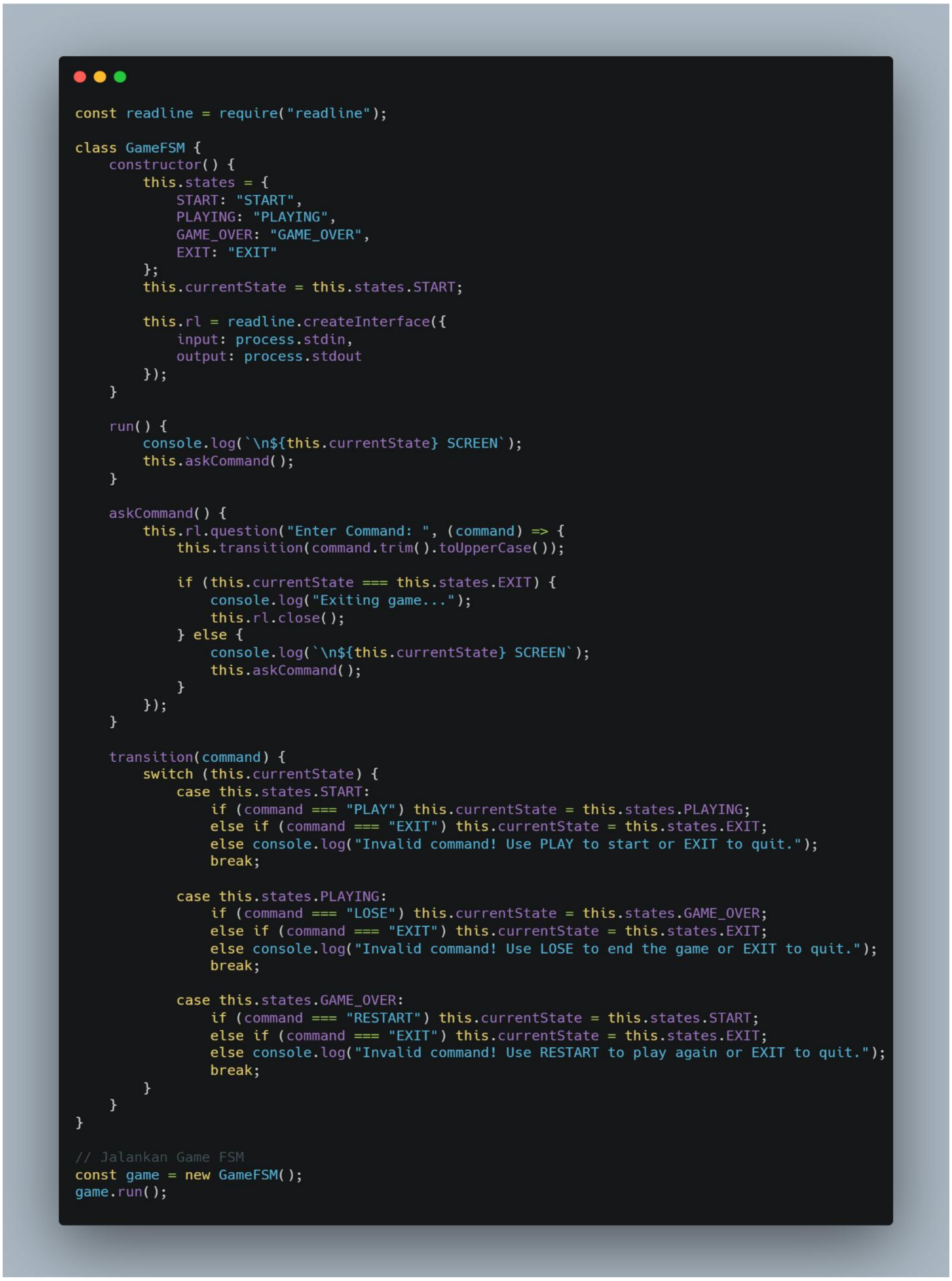
```
PS E:\Semester 6\kpl praktukum\Modul 08 - Runtime Configuration dan Internationalization> npm init -y
Wrote to E:\Semester 6\kpl praktukum\Modul 08 - Runtime Configuration dan Internationalization\package.json:

{
  "name": "modul-08---runtime-configuration-dan-internationalization",
  "version": "1.0.0",
  "main": "index.js",
  "scripts": {
    "test": "echo \"Error: no test specified\" && exit 1"
  },
  "keywords": [],
  "author": "",
  "license": "ISC",
  "description": ""
}



PS E:\Semester 6\kpl praktukum\Modul 08 - Runtime Configuration dan Internationalization> node index.js
Masukkan jumlah uang yang akan di-transfer: 123
Biaya transfer = 6500
Total biaya = 6623
Pilih metode transfer:
1. RTO (real-time)
2. SKN
3. RTGS
4. BI FAST
Pilih metode (tekan Enter setelah): 1


PS E:\Semester 6\kpl praktukum\Modul 08 - Runtime Configuration dan Internationalization> node index.js
Masukkan jumlah uang yang akan di-transfer: 123
Biaya transfer = 6500
Total biaya = 6623
Pilih metode transfer:
1. RTO (real-time)
2. SKN
3. RTGS
4. BI FAST
```



```
PS E:\Semester 6\kpl praktukum\Modul 08 - Runtime Configuration dan Internationalization> node index.js
Masukkan jumlah uang yang akan di-transfer: 123
Biaya transfer = 6500
Total biaya = 6623
Pilih metode transfer:
1. RTO (real-time)
2. SKN
3. RTGS


PS E:\Semester 6\kpl praktukum\Modul 08 - Runtime Configuration dan Internationalization> node index.js
Masukkan jumlah uang yang akan di-transfer: 123
Biaya transfer = 6500
Total biaya = 6623
Pilih metode transfer:
1. RTO (real-time)


PS E:\Semester 6\kpl praktukum\Modul 08 - Runtime Configuration dan Internationalization> node index.js
Masukkan jumlah uang yang akan di-transfer: 123
Biaya transfer = 6500
Total biaya = 6623


PS E:\Semester 6\kpl praktukum\Modul 08 - Runtime Configuration dan Internationalization> node index.js
Masukkan jumlah uang yang akan di-transfer: 123
Biaya transfer = 6500
Total biaya = 6623



PS E:\Semester 6\kpl praktukum\Modul 08 - Runtime Configuration dan Internationalization> node index.js
```

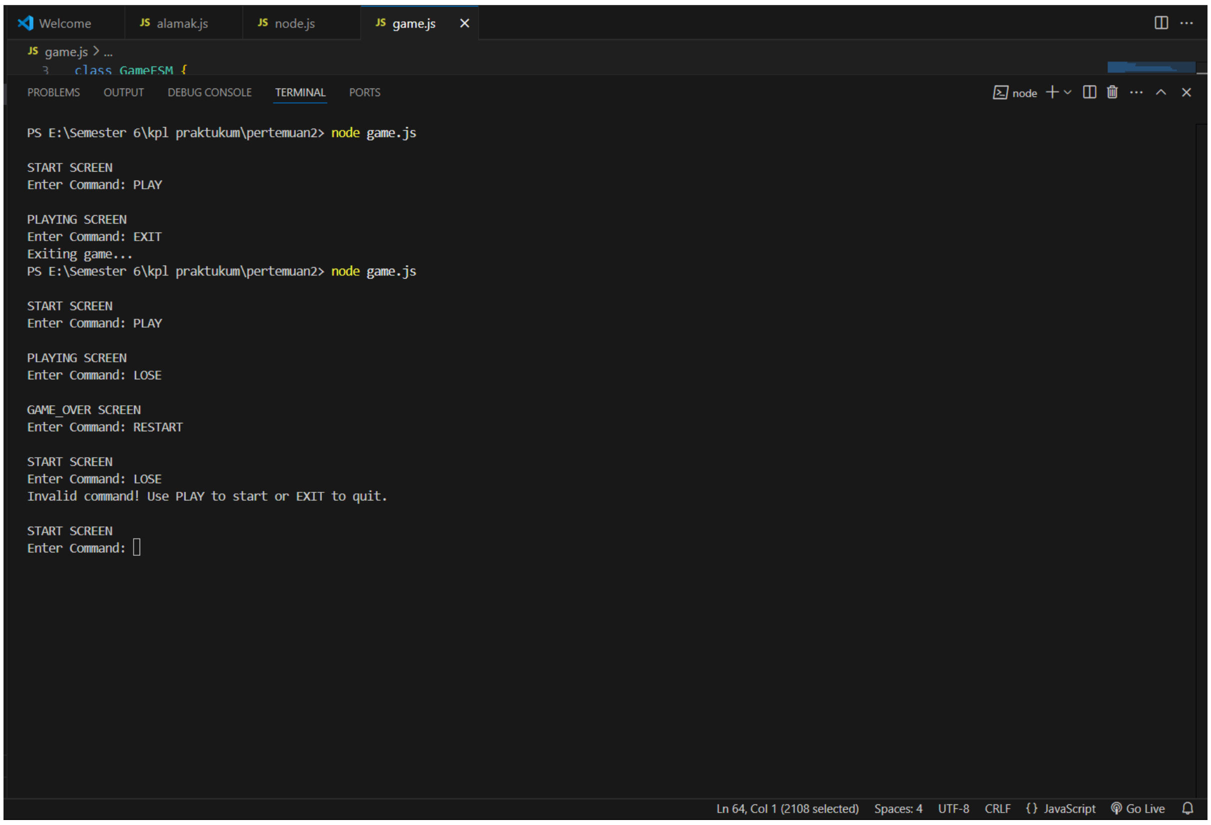
BAB III
PENUGASAN (UNGUIDED)

Game.js



```javascript
const readline = require("readline");

class GameFSM {
    constructor() {
        this.states = {
            START: "START",
            PLAYING: "PLAYING",
            GAME_OVER: "GAME_OVER",
            EXIT: "EXIT"
        };
        this.currentState = this.states.START;

        this.rl = readline.createInterface({
            input: process.stdin,
            output: process.stdout
        });
    }

    run() {
        console.log(`\n${this.currentState} SCREEN`);
        this.askCommand();
    }

    askCommand() {
        this.rl.question("Enter Command: ", (command) => {
            this.transition(command.trim().toUpperCase());

            if (this.currentState === this.states.EXIT) {
                console.log("Exiting game...");
                this.rl.close();
            } else {
                console.log(`\n${this.currentState} SCREEN`);
                this.askCommand();
            }
        });
    }

    transition(command) {
        switch (this.currentState) {
            case this.states.START:
                if (command === "PLAY") this.currentState = this.states.PLAYING;
                else if (command === "EXIT") this.currentState = this.states.EXIT;
                else console.log("Invalid command! Use PLAY to start or EXIT to quit.");
                break;

            case this.states.PLAYING:
                if (command === "LOSE") this.currentState = this.states.GAME_OVER;
                else if (command === "EXIT") this.currentState = this.states.EXIT;
                else console.log("Invalid command! Use LOSE to end the game or EXIT to quit.");
                break;

            case this.states.GAME_OVER:
                if (command === "RESTART") this.currentState = this.states.START;
                else if (command === "EXIT") this.currentState = this.states.EXIT;
                else console.log("Invalid command! Use RESTART to play again or EXIT to quit.");
                break;
        }
    }
}

// Jalankan Game FSM
const game = new GameFSM();
game.run();
```

Output:



```
PS E:\Semester 6\kpl praktukum\pertemuan2> node game.js

START SCREEN
Enter Command: PLAY

PLAYING SCREEN
Enter Command: EXIT
Exiting game...
PS E:\Semester 6\kpl praktukum\pertemuan2> node game.js

START SCREEN
Enter Command: PLAY

PLAYING SCREEN
Enter Command: LOSE

GAME_OVER SCREEN
Enter Command: RESTART

START SCREEN
Enter Command: LOSE
Invalid command! Use PLAY to start or EXIT to quit.

START SCREEN
Enter Command: 
```