

EST

ZERO DEBT

SAVE NOW
BUY LATER

REGEX-BASED FINANCIAL BOT

Pemrosesan Bahasa Alami

Melvin Waluyo

(22/492978/TK/53955)

Muhammad Grandiv Lava Putra

(22/493242/TK/54023)

INGS

BUDGET



The background is a green grid. In the top left, an orange pot with a plant has a sign that says 'INVEST'. In the top center, a white starburst. To its right, a winged gold coin with 'SPEND' on it. Further right, a pink piggy bank with gold coins on top. In the bottom left, another orange pot with a plant has a sign that says 'INVEST'. In the bottom center, a pink piggy bank with a sun on top and the word 'SAVE' on its side. On the right, a man in a black shirt sits at a desk with a laptop, looking at a graph. A woman in a purple shirt stands next to him, pointing at a flowchart on a whiteboard. The flowchart starts with an orange oval, followed by a purple rectangle, then a yellow diamond. From the diamond, three arrows point down to three purple rectangles, each followed by an orange circle. A woman in a white lab coat stands to the right of the whiteboard, holding a clipboard. A white starburst is at the bottom right.

LATAR BELAKANG

Financial Bot dikembangkan untuk mengatasi kesulitan umum dalam pengelolaan keuangan pribadi, seperti pencatatan transaksi manual yang sering terlupakan dan rumitnya penggunaan aplikasi keuangan tradisional.

FITUR UTAMA

Pencatatan Transaksi



Melvin-Grandiv 14:02

@FinancialBot Saya dapat gaji 5000000 dari kantor



FinancialBot APP 14:02

Baik! Saya telah mencatat pemasukan kamu sebesar Rp 5,000,000 untuk kategori 'gaji' dengan deskripsi 'kantor'.

💰 Saldo terbaru: Rp 5,000,000



Melvin-Grandiv 14:02

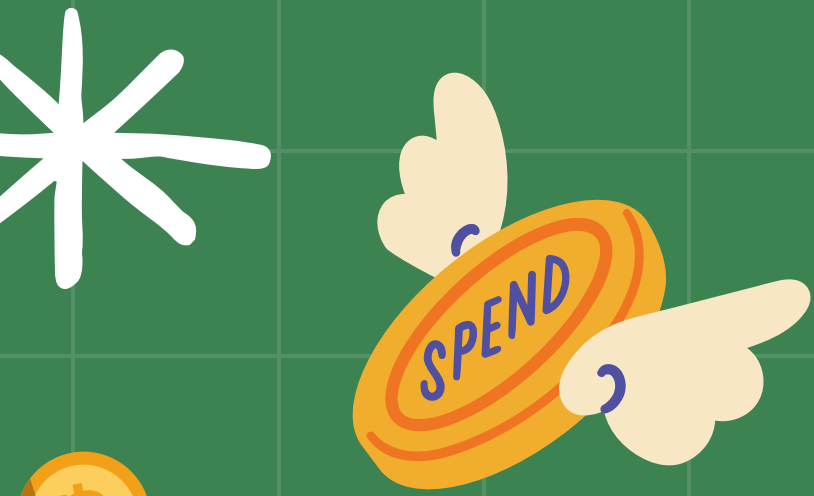
@FinancialBot Habis 50000 untuk makan siang di Sambal SS



FinancialBot APP 14:02


Oke! Saya sudah catat pengeluaran kamu sebesar Rp 50,000 untuk kategori 'makan' dengan deskripsi 'siang di Sambal SS'.


💰 Saldo terbaru: Rp 4,950,000



FITUR UTAMA

Pencatatan Transaksi

 **Melvin-Grandiv** 14:06
@FinancialBot Laporan keuangan

 **FinancialBot APP** 14:06
📊 **Laporan Keuangan per Kategori:**

💰 **Total Pemasukan:** Rp 5,000,000
📉 **Total Pengeluaran:** Rp 50,000
📈 **Saldo:** Rp 4,950,000

💚 **Pemasukan:**
- Gaji: Rp 5,000,000

📉 **Pengeluaran:**
- Makan: Rp 50,000

 **Melvin-Grandiv** 14:05
@FinancialBot Saldo saya

 **FinancialBot APP** 14:05
💰 **Ringkasan Keuangan Kamu:**

- Pemasukan: Rp 5,000,000
- Pengeluaran: Rp 50,000
- Saldo: Rp 4,950,000 (positif)

📋 **5 Transaksi Terakhir:**

1. 📉 Rp 50,000 - makan (siang di Sambal SS)
2. 💚 Rp 5,000,000 - gaji (kantor)

FITUR UTAMA

Kategori Otomatis



Melvin-Grandiv 14:45

@FinancialBot gue dapet 50000000 dari proyek



FinancialBot APP 14:45

Baik! Saya telah mencatat pemasukan kamu sebesar Rp 50,000,000 untuk kategori 'gaji'.

💰 Saldo terbaru: Rp 50,150,000



Melvin-Grandiv 14:45

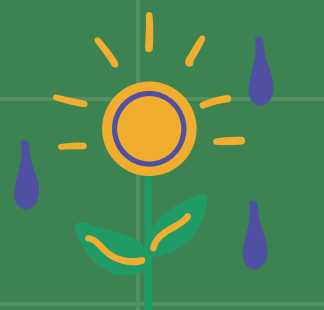
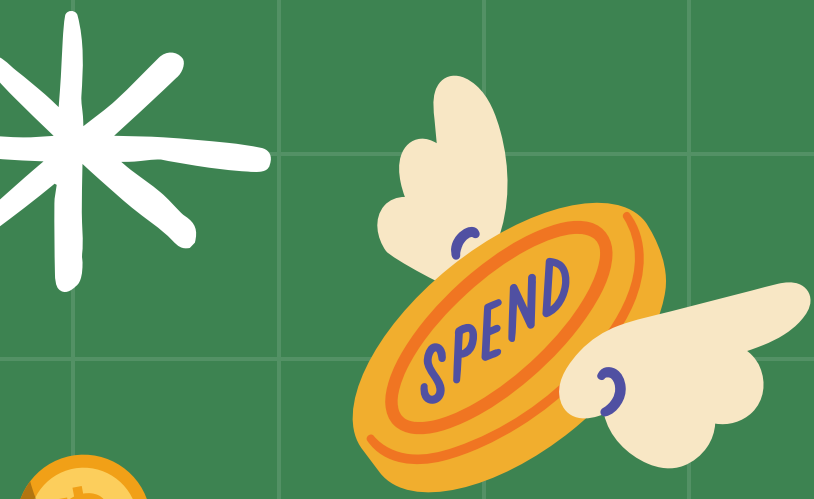
@FinancialBot gue habis 40000 untuk bensin



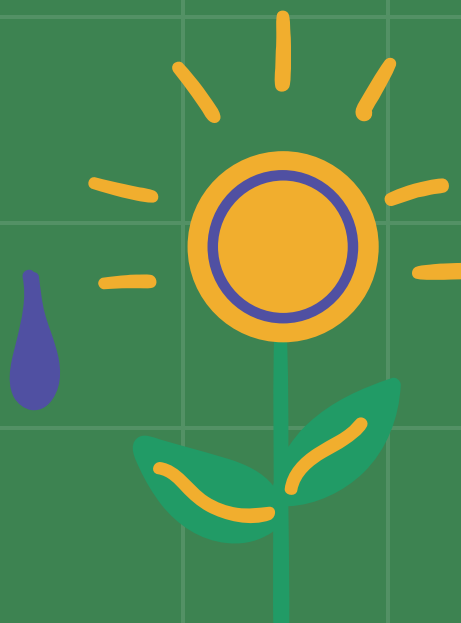
FinancialBot APP 14:45

Oke! Saya sudah catat pengeluaran kamu sebesar Rp 40,000 untuk kategori 'transport'.

💰 Saldo terbaru: Rp 50,110,000




INVEST





INVEST


FITUR UTAMA


Analisis Pembelian

 Melvin-Grandiv 14:04
@FinancialBot Saya mau beli laptop 15000000

 FinancialBot **APP** 14:04
 **Analisis Beli Laptop:**


Harga Barang: Rp 15,000,000
Gaji Bulanan: Rp 5,000,000
Saldo Sekarang: Rp 4,950,000

 **Kurang Dana:** Masih kurang Rp 10,050,000

 **Pilihan untuk Kamu:**

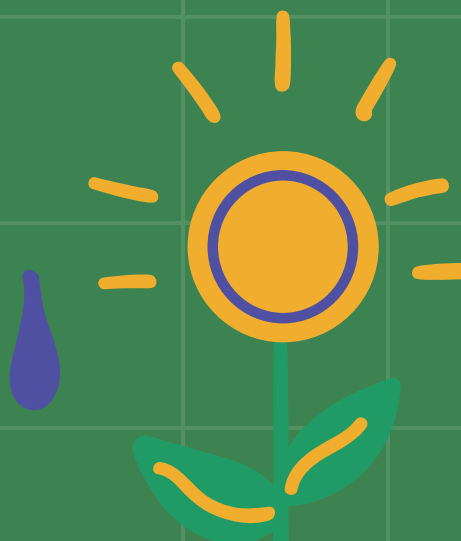
Pilihan 1: Nabung Dulu (2 bulan)
• Nabung Rp 4,950,000 per bulan
• Bisa beli cash tanpa hutang

Pilihan 2: Cari Alternatif Lebih Murah (Rp 9,000,000)
• Lebih mudah dijangkau
• Bisa beli lebih cepat

 **Saran Saya:** Nabung dulu sekitar 2 bulan. Lebih aman dan tidak ada beban hutang.



INVEST



INVEST

FITUR UTAMA

Saran Anggaran



Melvin-Grandiv 14:02

@FinancialBot Bantuan anggaran saya



FinancialBot APP 14:02

💰 Ringkasan Keuangan Bulanan:

- Pemasukan: Rp 5,000,000
- Pengeluaran: Rp 50,000
- Sisa: Rp 4,950,000

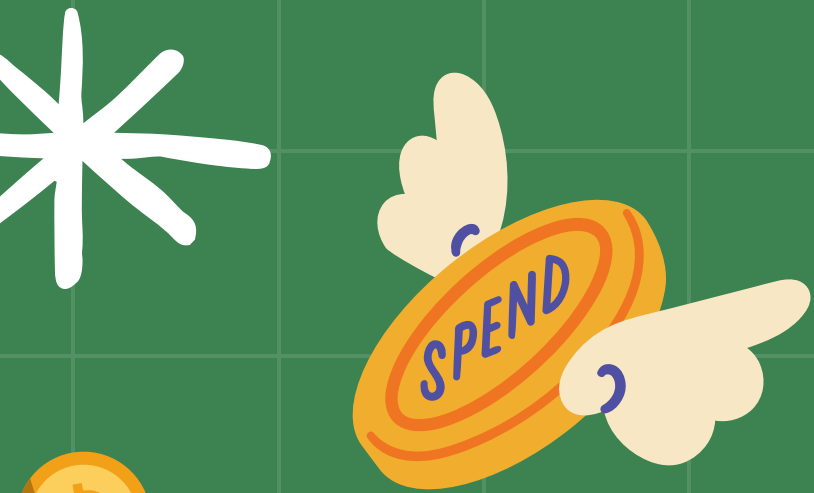
📊 Saran Penggunaan Sisa Uang:

- Dana Darurat: Rp 750,000 (15% dari gaji)
- Tabungan: Rp 1,485,000 (30% dari sisa)
- Sisanya untuk Kamu: Rp 2,715,000

👍 **Bagus Sekali:** Pengeluaran kamu cuma 1% dari gaji. Kamu bisa nabung banyak!

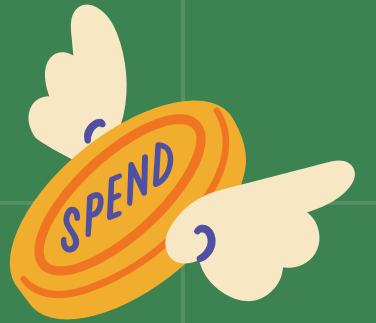
💡 Yang Harus Dilakukan:

- Sisihkan dana darurat dulu (target 6 bulan pengeluaran)
- Tabung sisanya secara rutin
- Kalau udah cukup, coba investasi





RULES



Regex Pemasukan

```
self.income_patterns = [  
    # Pattern: !income 100000 gaji bonus bulan ini  
    r'!income\s+(\d+(?:\.\d+)?)\s+(\w+)(?:\s+(.+))?',  
    # Pattern natural: saya dapat gaji 5000000 dari kantor  
    r'(?:(saya|aku)\s+(?:dapat|terima|dapat|menerima|meraih|peroleh)\s+(\w+)\s+(\d+(?:\.\d+)?)\s*(?:dari\s+(.+))?',  
    # Pattern: dapat 50000 dari ortu (FIXED: swap amount and category order)  
    r'(?:(dapat|terima|dapat|menerima|meraih|peroleh)\s+(\d+(?:\.\d+)?)\s+(?:dari\s+)?(\w+)(?:\s+(.+))?',  
    # Pattern: income 1000000 kategori deskripsi  
    r'(?:(income|pemasukan|masuk)\s+(\d+(?:\.\d+)?)\s+(\w+)(?:\s+(.+))?'  
]
```

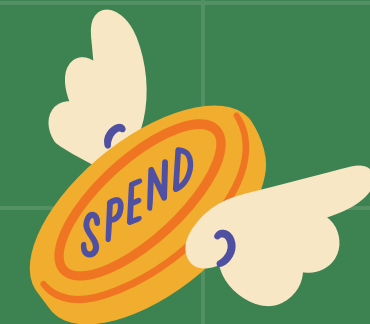
Regex Pengeluaran

```
self.expense_patterns = [  
    # Pattern: !expense 50000 makanan makan siang  
    r'!expense\s+(\d+(?:\.\d+)?)\s+(\w+)(?:\s+(.+))?',  
    # Pattern natural: saya habis 50000 untuk makanan  
    r'(?:(saya|aku)\s+(?:habis|abis|keluar|bayar|beli|menghabiskan|mengeluarkan|belanja|pakai|gunakan)\s+(\d+(?:\.\d+)?)\s+(?:untuk\s+)?(\w+)(?:\s+(.+))?',  
    # Pattern: keluar 75000 transport  
    r'(?:(keluar|habis|abis|bayar|beli|expense|pengeluaran|menghabiskan|mengeluarkan|belanja|pakai|gunakan)\s+(\d+(?:\.\d+)?)\s+(?:untuk\s+)?(\w+)(?:\s+(.+))?',  
    # Pattern: beli makanan 25000  
    r'(?:(beli|bayar|belanja)\s+(\w+)\s+(\d+(?:\.\d+)?)(?:\s+(.+))?'  
]
```




RULES

Regex Cek Saldo dan Laporan



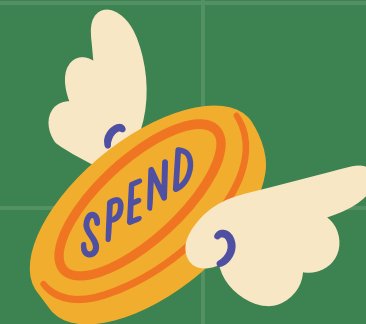
```
# Pattern untuk saldo
self.balance_patterns = [
    r'^!balance$',
    r'^(?:saldo|balance)(?:\s+(?:saya|aku))?$ ',
    r'^(?:cek|lihat|check)\s+(?:saldo|balance|uang)$ ',
    r'^(?:berapa)\s+(?:saldo|uang)(?:\s+(?:saya|aku))?$ ',
    r'^(?:saya|aku)\s+(?:punya|ada)\s+(?:berapa)$ '
]

# Pattern untuk laporan
self.report_patterns = [
    r'^!report$',
    r'^(?:laporan|report)$ ',
    r'^(?:lihat|show|cek)\s+(?:laporan|report)$ ',
    r'^(?:ringkasan|summary)\s+(?:keuangan|finansial)$ ',
    r'^summary\s+keuangan$',
    r'^(?:laporan|report)\s+(?:keuangan|finansial)$ '
]
```



RULES

Regex Bantuan Anggaran

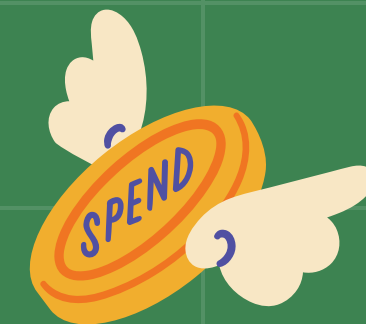


```
self.budget_patterns = [  
    # Indonesian patterns  
    r'(?:(bantuan|buatkan|buat)\s+(?:saya|aku)?\s*(?:buat|bikin)?\s+(?:anggaran|budget)',  
    r'(?:(saya|aku)\s+(?:mau|ingin|butuh|perlu)\s+(?:anggaran|budget|bantuan anggaran|budget  
advice)',  
    r'(?:(gimana|bagaimana)\s+(?:cara|bikin|buat)\s+(?:anggaran|budget)',  
    r'(?:(tolong|bantuan)\s+(?:anggaran|budget)',  
    r'(?:(analisis|analisa|cek)\s+(?:anggaran|budget)\s+(?:saya|aku)',  
    r'(?:(anggaran|budget)\s+(?:saran|rekomendasi)',  
    r'(?:(buat|bikin|buatkan)\s+(?:anggaran|budget)',  
    r'(?:(bantuan)\s+(?:anggaran|budget)',  
    r'(?:(saran|rekomendasi)\s+(?:anggaran|budget)',  
    r'(?:(konsultasi)\s+(?:anggaran|keuangan|budget)',  
    # English patterns  
    r'(?:(help|create|make).*(?:budget)',  
    r'(?:(budget).*(?:help|advice|recommendation|create)',  
    r'(?:(i|I)\s+(?:need|want|require)\s+(?:a\s+)?(?:budget)',  
    r'(?:(how to|how)\s+(?:create|make)\s+(?:a\s+)?(?:budget)',  
    r'(?:(budget).*(?:planning|advice)',  
    r'(?:(financial).*(?:advice|planning)',  
    r'(?:(help me create a budget)'  
]
```



RULES

Regex Rencana Pembelian

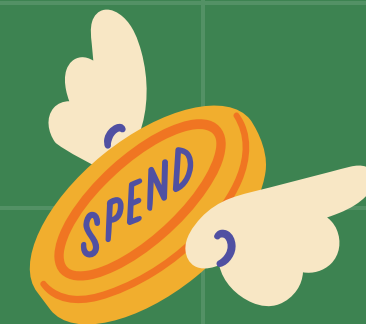


```
self.purchase_patterns = [  
    # Indonesian patterns  
    r'(? :saya|aku)\s+(? :mau|ingin|pengen)\s+(? :beli)\s+(.+?)(? :\s+(? :harga|seharga)?\s*(\d+(? :,?\d+)*))?',  
    r'(? :mau|ingin|pengen)\s+(? :beli)\s+(.+?)(? :\s+(? :Rp)?\s*(\d+(? :,?\d+)*))?',  
    r'(? :rencana)\s+(? :beli|membeli)\s+(.+?)(? :\s+(? :Rp)?\s*(\d+(? :,?\d+)*))?',  
    r'(? :analisis|analisa)\s+(? :beli)\s+(.+?)(? :\s+(? :Rp)?\s*(\d+(? :,?\d+)*))?',  
    r'(? :konsultasi)\s+(? :beli|pembelian)\s+(.+?)(? :\s+(? :Rp)?\s*(\d+(? :,?\d+)*))?',  
    # English patterns  
    r'(? :i|I)\s+(? :want to|wanna)\s+(? :buy)\s+(? :a\s+)?(.+?)(? :\s+(? :for|at|price)?\s*(\d+(? :,?\d+)*))?',  
    r'(? :buying|purchase)\s+(? :a\s+)?(.+?)(? :\s+(? :for|at|price)?\s*(\d+(? :,?\d+)*))?',  
    r'(? :planning to buy)\s+(? :a\s+)?(.+?)(? :\s+(? :for|at|price)?\s*(\d+(? :,?\d+)*))?',  
    r'(? :can i buy)\s+(? :a\s+)?(.+?)(? :\s+(? :for|at|price)?\s*(\d+(? :,?\d+)*))?'  
]
```



RULES

Regex Kategori Otomatis



```
# 1. Keywords dalam deskripsi
# 2. Jumlah transaksi (>1M = kemungkinan gaji)
# 3. Context patterns (waktu, lokasi, dll)
income_keywords = {
    'gaji': ['gaji', 'kantor', 'kerja', 'pekerjaan'],
    'freelance': ['freelance', 'projek', 'kontrak', 'lepas', 'sampingan'],
    'investasi': ['saham', 'reksadana', 'dividen', 'profit', 'trading', 'investasi'],
    'hadiah': ['hadiah', 'bonus', 'reward', 'kado', 'pemberian']
}

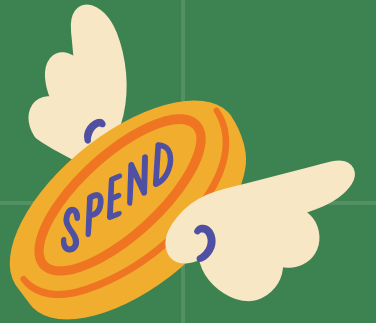
expense_keywords = {
    'makanan': ['makan', 'nasi', 'ayam', 'restaurant', 'cafe', 'snack', 'lapar', 'kenyang',
    'minuman'],
    'transport': ['transport', 'bensin', 'ojek', 'taksi', 'bus', 'kereta', 'grab', 'gojek',
    'motor', 'mobil'],
    'hiburan': ['film', 'game', 'spotify', 'netflix', 'youtube', 'concert', 'musik', 'hiburan'],
    'belanja': ['beli', 'shopping', 'baju', 'sepatu', 'elektronik', 'gadget', 'belanja', 'mall',
    'toko'],
    'tagihan': ['listrik', 'air', 'internet', 'wifi', 'telepon', 'cicilan', 'bayar', 'tagihan'],
    'kesehatan': ['dokter', 'obat', 'hospital', 'rumah sakit', 'vitamin', 'therapy', 'clinic'],
    'pendidikan': ['kursus', 'buku', 'sekolah', 'kuliah', 'training', 'seminar', 'workshop']
}
```



RULES



Regex Reflection Noun

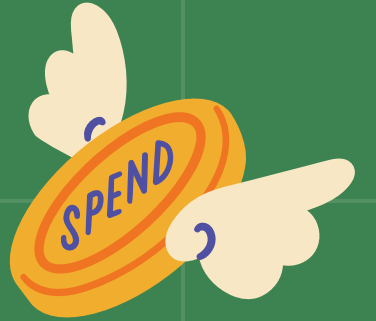


```
self.pronoun_reflections = {  
    'saya': 'kamu',  
    'aku': 'kamu',  
    'kamu': 'saya',  
    'anda': 'saya'  
}  
  
# Mapping untuk kata kerja  
self.verb_reflections = {  
    'punya': 'punya',  
    'adalah': 'adalah',  
    'ingin': 'ingin',  
    'mau': 'mau',  
    'bisa': 'bisa',  
    'akan': 'akan',  
    'sudah': 'sudah',  
    'belum': 'belum'  
}
```



RULES

Algoritma Saran Anggaran



```
expense_percentage = (expense / income * 100) if income > 0 else 0
available = balance
```

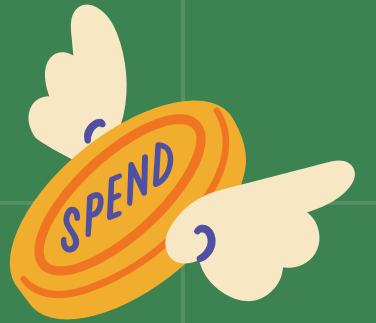
```
dana_darurat = income * 0.15 # 15% untuk dana darurat
tabungan = available * 0.30 if available > 0 else 0 # 30% dari sisa untuk tabungan
sisanya = max(0, available - dana_darurat - tabungan)
```

```
if expense_percentage > 80:
    response += "⚠️ **Perhatian**: Pengeluaran kamu sudah {:.0f}% dari gaji. Coba kurangi  
pengeluaran yang tidak perlu.\n\n".format(expense_percentage)
elif expense_percentage > 60:
    response += "✅ **Lumayan Baik**: Pengeluaran kamu {:.0f}% dari gaji. Sisanya bisa ditabung  
atau investasi.\n\n".format(expense_percentage)
else:
    response += "👍 **Bagus Sekali**: Pengeluaran kamu cuma {:.0f}% dari gaji. Kamu bisa nabung  
banyak!\n\n".format(expense_percentage)
```

```
# Simple actionable tips
response += "💡 **Yang Harus Dilakukan**:\n"
if available > 0:
    response += "• Sisihkan dana darurat dulu (target 6 bulan pengeluaran)\n"
    response += "• Tabung sisanya secara rutin\n"
    response += "• Kalau udah cukup, coba investasi\n"
else:
    response += "• Catat semua pengeluaran selama sebulan\n"
    response += "• Cari pengeluaran yang bisa dikurangi\n"
    response += "• Coba cari tambahan pendapatan\n"
```


* RULES

Algoritma Analisis Pembelian



```
balance_info = user_data.get('balance', {})  
income = balance_info.get('income', 0)  
expense = balance_info.get('expense', 0)  
current_balance = balance_info.get('balance', 0)  
  
response = f"🛒 **Analisis Beli {item.title()}**:\n\n"  
response += f"**Harga Barang**: Rp {price:,.0f}\n"  
response += f"**Gaji Bulanan**: Rp {income:,.0f}\n"  
response += f"**Saldo Sekarang**: Rp {current_balance:,.0f}\n\n"  
  
# Simple affordability check  
can_afford_now = current_balance >= price  
months_salary = (price / income) if income > 0 else 0
```

Akan generate 3 alternatif:
Opsi 1: Nabung dulu (aman)
Opsi 2: Cari alternatif lebih murah
Opsi 3: Beli sekarang (dengan syarat)

TEST

```
test_database.py
test_edge_cases.py
test_indonesian_features.py
test_integration.py
test_new_features.py
test_performance.py
test_rules.py
```

12 cases

18 cases

7 cases

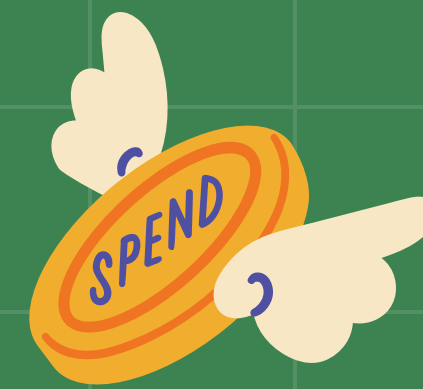
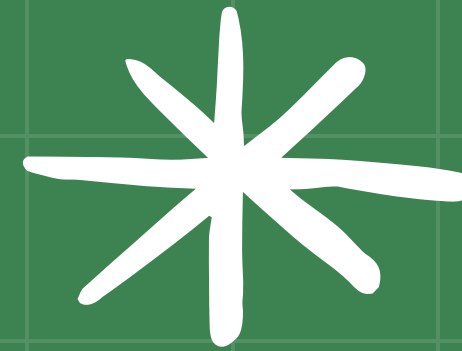
11 cases

9 cases

7 cases

15 cases

79 cases



TEST

test_database.py (12 cases)

Menuji semua operasi database, termasuk CRUD operations dan kalkulasi saldo.

Key Tests:

- Database initialization dan schema
- Add/delete transactions
- Balance calculation dengan multiple transactions
- User data isolation
- Category reporting
- Large number handling

test_indonesian_features (7 cases)

Menguji fitur dan respons yang spesifik menggunakan Bahasa Indonesia.

Key Tests:

- Indonesian budget patterns recognition
- Purchase planning patterns
- Response validation (pure Indonesian)
- No English leakage dalam responses

test_edge_cases (18 cases)

Menguji ketahanan bot terhadap input yang tidak terduga, salah, atau ekstrem.

Key Tests:

- Empty/whitespace messages
- Invalid amount formats
- Unicode characters
- Extremely long descriptions
- Database corruption recovery
- Rapid successive requests

test_integration.py (11 cases)

Memastikan semua komponen bot bekerja sama dengan baik secara end-to-end.

Key Tests:

- Complete income/expense flow
- Balance query variations
- Report generation
- Multi-user isolation
- Negative balance warnings
- Error handling untuk invalid inputs



TEST

test_new_features.py (9 cases)

Memvalidasi akurasi fitur analisis seperti saran anggaran dan rencana pembelian.

Key Tests:

- Budget advice parsing dan response
- Purchase planning analysis
- Different language pattern support
- Data-driven recommendations

test_rules.py (15 cases)

Memvalidasi pemahaman bot terhadap berbagai format perintah dan percakapan natural bahasa Indonesia.

Key Tests:

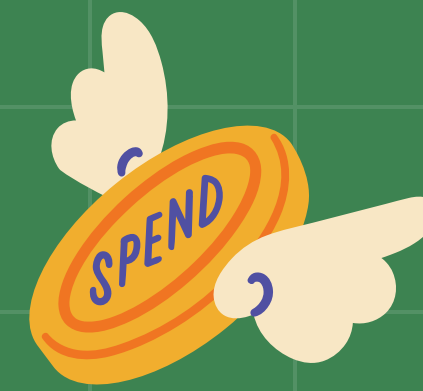
- Pronoun reflection (saya ↔ kamu)
- Income/expense command parsing (4 variasi format)
- Balance/report/help command recognition
- Automatic categorization
- Response generation dengan context

test_performance.py (7 cases)

Mengukur kecepatan, stabilitas, dan skalabilitas bot saat beban kerja tinggi.

Key Tests:

- Response time basic commands (<100ms)
- Large transaction volume handling
- Pattern matching efficiency
- Memory usage stability
- Concurrent users simulation



HASIL TEST

Script run semua tests

```
@echo off
echo =====
echo Financial Bot Discord - Test Runner
echo =====
echo.

echo [1/3] Activating virtual environment...
call venv\Scripts\activate.bat

echo [2/3] Running all tests...
echo.
python -m pytest tests/ -v --tb=short

echo.
echo [3/3] Test Summary:
echo =====
echo Tests completed!
echo =====
pause
```



```
(venv) PS D:\Data Melvin\Vscode\FinancialBot> ./run_tests.bat
tests/test_new_features.py::TestNewFeatures::test_budget_advice_with_data PASSED [ 64%]
tests/test_new_features.py::TestNewFeatures::test_budget_advice_without_data PASSED [ 65%]
tests/test_new_features.py::TestNewFeatures::test_purchase_planning_different_languages PASSED [ 67%]
tests/test_new_features.py::TestNewFeatures::test_purchase_planning_expensive_item PASSED [ 68%]
tests/test_new_features.py::TestNewFeatures::test_purchase_planning_parsing PASSED [ 69%]
tests/test_new_features.py::TestNewFeatures::test_purchase_planning_with_data PASSED [ 70%]
tests/test_new_features.py::TestNewFeatures::test_purchase_planning_without_data PASSED [ 72%]
tests/test_performance.py::TestPerformance::test_concurrent_users PASSED [ 73%]
tests/test_performance.py::TestPerformance::test_database_query_performance PASSED [ 74%]
tests/test_performance.py::TestPerformance::test_large_transaction_volume PASSED [ 75%]
tests/test_performance.py::TestPerformance::test_memory_usage_stability PASSED [ 77%]
tests/test_performance.py::TestPerformance::test_pattern_matching_performance PASSED [ 78%]
tests/test_performance.py::TestPerformance::test_regex_pattern_efficiency PASSED [ 79%]
tests/test_performance.py::TestPerformance::test_response_time_basic_commands PASSED [ 81%]
tests/test_rules.py::TestReflectionEngine::test_pronoun_reflection_indonesian PASSED [ 82%]
tests/test_rules.py::TestFinancialRulesEngine::test_about_command_parsing PASSED [ 83%]
tests/test_rules.py::TestFinancialRulesEngine::test_amount_parsing PASSED [ 84%]
tests/test_rules.py::TestFinancialRulesEngine::test_automatic_categorization PASSED [ 86%]
tests/test_rules.py::TestFinancialRulesEngine::test_balance_command_parsing PASSED [ 87%]
tests/test_rules.py::TestFinancialRulesEngine::test_capability_command_parsing PASSED [ 88%]
tests/test_rules.py::TestFinancialRulesEngine::test_delete_command_parsing PASSED [ 89%]
tests/test_rules.py::TestFinancialRulesEngine::test_expense_command_parsing PASSED [ 91%]
tests/test_rules.py::TestFinancialRulesEngine::test_goodbye_command_parsing PASSED [ 92%]
tests/test_rules.py::TestFinancialRulesEngine::test_help_command_parsing PASSED [ 93%]
tests/test_rules.py::TestFinancialRulesEngine::test_income_command_parsing PASSED [ 94%]
tests/test_rules.py::TestFinancialRulesEngine::test_report_command_parsing PASSED [ 96%]
tests/test_rules.py::TestFinancialRulesEngine::test_response_generation PASSED [ 97%]
tests/test_rules.py::TestFinancialRulesEngine::test_thanks_command_parsing PASSED [ 98%]
tests/test_rules.py::TestFinancialRulesEngine::test_unknown_command PASSED [100%]

===== 79 passed in 7.33s =====

[3/3] Test Summary:
=====
Tests completed!
```

79 cases berhasil 100% passed

THANK YOU

