

# 간단한 지출 관리 챗봇 **머니챗**(MoneyChat)



# 목차



# 지출 관리는 일상 생활에서 매우 중요한 부분



특히, 비교적 수입이 적은 10대,  
20대에겐 더욱 중요한 부분이다.

## 다양한 지출 관리 어플



편한가계부

Realbyte Inc.

광고 포함 · 인앱 구매

4.7★  
리뷰 40.5만개

1,000만+  
다운로드

3세 이상 ⓘ



좋은가계부

HS App

광고 포함

4.9★  
리뷰 3.14만개













100만+  
다운로드

3세 이상 ⓘ

출처 - Google Play

## 모든 지출이 포함되는 문제

지출이 자동으로 반영된다는 장점이 있지만, 원치않는 지출도 모두 포함시키는 불편함

		20:09	18,000원
		20:05	15,000원
	 치킨	20:01	-63,500원
		21:31	10,000원
		21:31	10,000원
		18:10	-30,000원

예) 친구들을 대표해서 내 카드로 대신 계산을 하고 더치페이할 때, 해당 지출이 내 지출 내역에 포함됨

## 복잡한 지출 입력

하나의 지출 입력도 지불 날짜, 지불 방법, 카테고리 분류, 금액, 내용 등 입력할 것이 많은 불편함



구글플레이 - 편한가계부



구글플레이 - 좋은 가계부

# 간단한 지출 관리 챗봇 머니챗(MoneyChat)

**간결함**  
한 줄 입력으로  
빠르고 간편하게 지출 기록

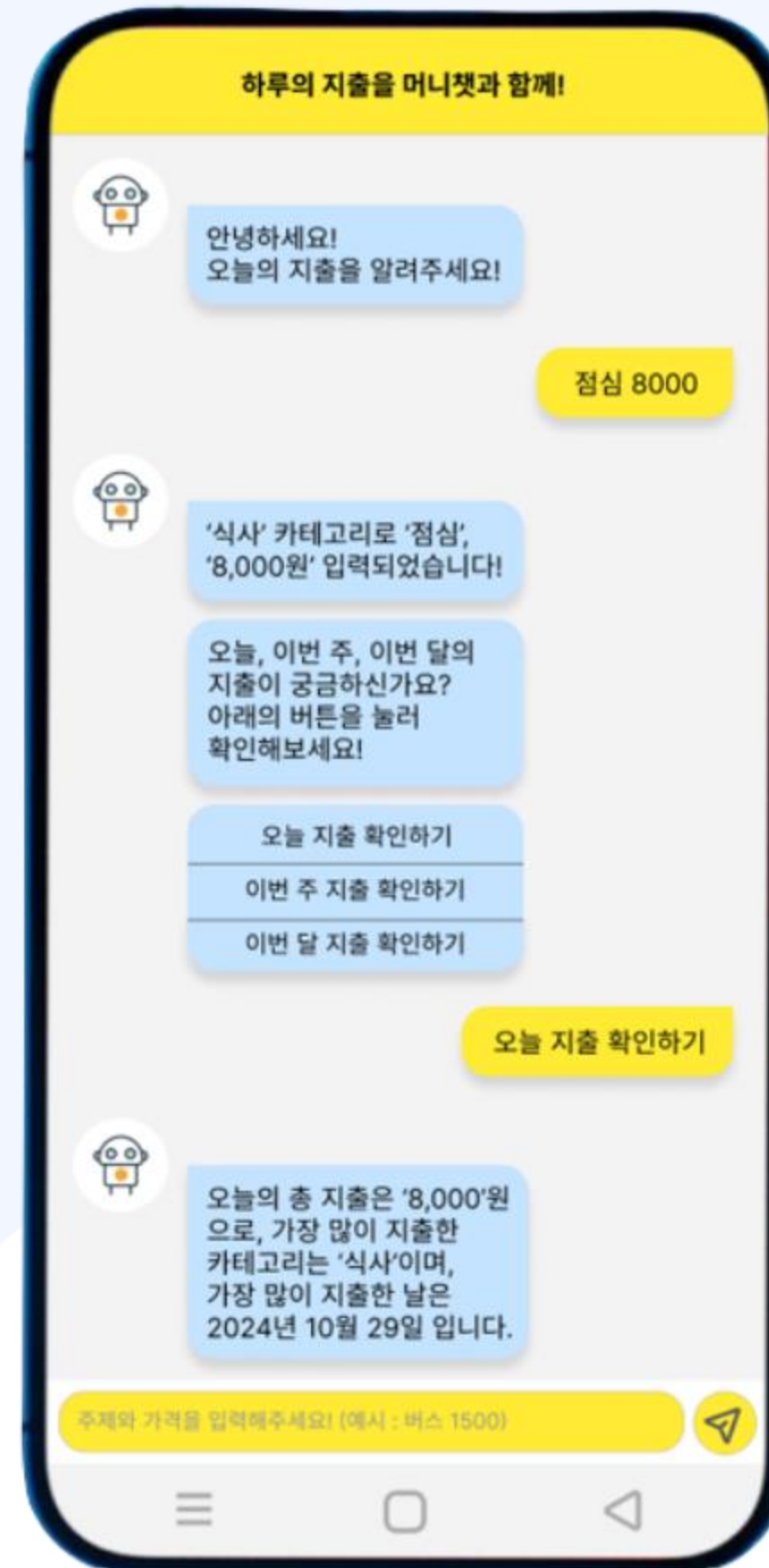
**사용 용이성**  
직관적인 인터페이스로  
누구나 쉽게 사용 가능

**카테고리화**  
주제와 가격만 입력하면  
챗봇이 자동으로 카테고리 분류

01

02

03



머니챗 완성 예시

04

**시간 절약**  
복잡한 입력 과정 없이  
간단한 채팅만으로 기록 가능

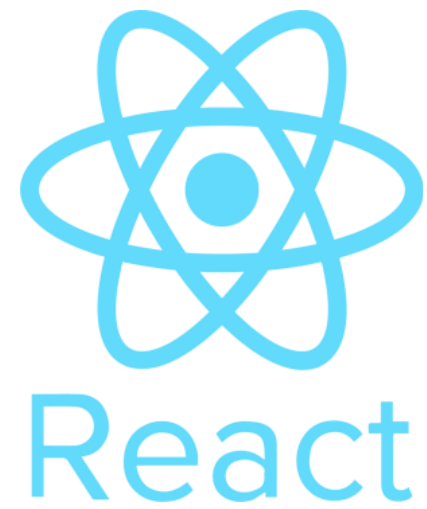
05

**필요한 정보만**  
불필요한 입력 없이  
실제로 반영될 금액만 입력

06

**지출 통합 요약**  
오늘, 이번 주, 이번 달 동안의  
지출 내역을 정리

프론트



백엔드



챗봇 API



DB



하루의 지출을 머니챗과 함께!

로그아웃

MoneyChat

B

안녕하세요!  
오늘 하루 지출한 금액을 주제와 함께 작성해주세요!

점심 8000

B

음식 카테고리로 점심 8,000원 입력되었습니다!

오늘, 이번 주, 이번 달의 지출이 궁금하신가요?  
아래의 버튼을 눌러 확인해보세요!

오늘 지출 확인하기

이번 주 지출 확인하기

이번 달 지출 확인하기

B

오늘의 총 지출 금액은 8,000원입니다.

주제와 가격을 입력해주세요! (예시 : 버스 1500)

실제 구현 화면

이메일 주소, 전화번호 또는 사용자 UID로 검색

사용자 추가

식별자	제공업체	생성한 날짜 ↓	로그인한 날짜	사용자 UID
123@naver.com		2024. 10. 27.	2024. 10. 27.	
son@naver.com		2024. 10. 22.	2024. 10. 27.	
kim@naver.com		2024. 10. 11.	2024. 10. 11.	
park@naver.com		2024. 10. 11.	2024. 10. 11.	

페이지당 행 수: 50 1 - 4 of 4

사용자 별 이메일 관리

expenses > y1mV7jbAKThm... > userExpenses >

Google Cloud의 추가 기능

y1mV7jbAKThmOHbAwR...	userExpenses	
+ 컬렉션 시작	+ 문서 추가	+ 컬렉션 시작
userExpenses >		+ 필드 추가
		amount: 8000
		category: "점심"
		expenseCategory: "음식"
		timestamp: 2024년 10월 27일 오후 6시 52분 19초 UTC+9
+ 필드 추가		

저장된 데이터

# 개발 진척도 (전체 : 35%)

	9주	10주	11주	12주	13주	14주	15주	16주
기능 개발	35% ~ 기능 개발							
알파테스트	30% ~ 알파테스트							
베타테스트				베타테스트				
사용설명서 작성						사용설명서 작성		
발표자료 제작							발표자료 제작	



## 향후 계획

기본 기능 개발 -> 테스트 (+ 추가 기능 개발)

### 기본 기능 개발

- 현재까지 챗봇의 기본적인 틀을 잡고, 데이터 저장과 조회까지 가능.
- OpenAI의 GPT 챗봇 API를 프로젝트에 반영하여 자연스러운 대화형 챗봇으로 개발
- 지출 데이터를 자동으로 요약하고 지출에 따른 피드백을 제공해주는 기능

### 테스트

- 기본 기능 개발 완료 후 다양한 테스트를 통해 버그나 부족한 부분 보완
- 일상생활에 실제로 사용할 수 있도록, 가능하다면 웹사이트 배포 고려 중
- 프로젝트 마무리

### 추가 기능 개발

- 테스트 완료 이후 여유가 된다면 챗봇의 다양한 기능 추가
- 사용자의 지출 패턴 분석, 지출 경고 알림, 피드백에 포함될 시각자료 추가, 이전 대화 기록 불러오기 등 다양한 기능 추가

# 목차



## 기존 지출 관리 앱들의 불편한 점들

### 모든 지출이 포함되는 문제



실제 지출 내역 속 중복 지출

지출이 자동으로 반영된다는 장점이 있지만, 원치않는 지출도 모두 포함시키는 불편함

### 낮은 가시성



구글플레이 - 편한 가계부

많은 정보가 있어 구체적인 내역을 확인할 수 있어 좋지만, 그만큼 가시성이 떨어짐

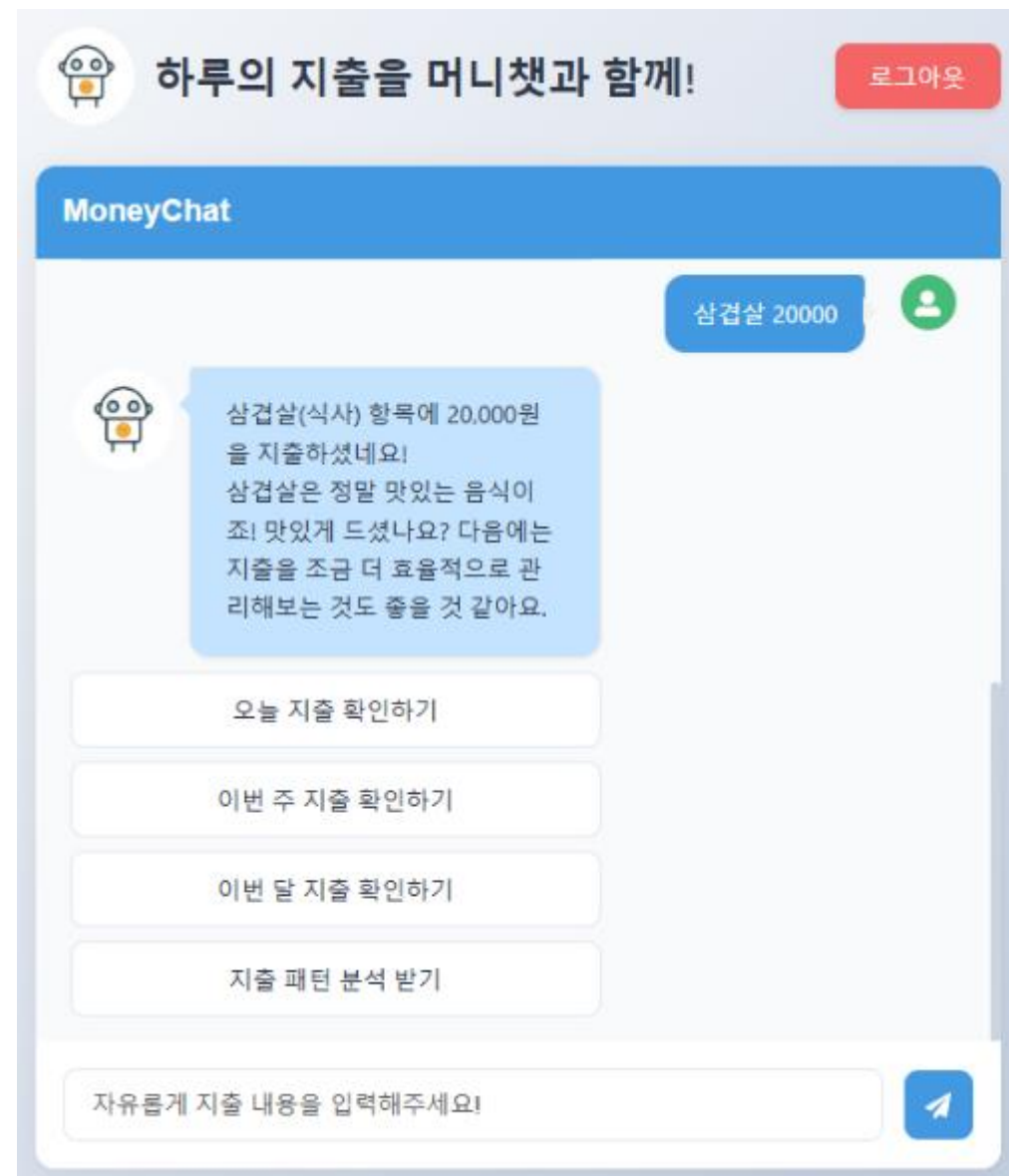
### 복잡한 지출 입력

구글플레이 - 좋은 가계부

하나의 지출 입력도 지불 날짜, 지불 방법, 카테고리 분류, 금액, 내용 등 입력할 것이 많은 불편함


## 머니챗(MoneyChat)으로 해결!


자신이 원하는 지출만, 한 문장으로 간단하고  
쉽게 입력 가능



하루의 지출을 머니챗과 함께! [로그아웃](#)

MoneyChat

삼겹살 20000 


 삼겹살(식사) 항목에 20,000원을 지출하셨네요!  
삼겹살은 정말 맛있는 음식이죠! 맛있게 드셨나요? 다음에는 지출을 조금 더 효율적으로 관리해보는 것도 좋을 것 같아요.

오늘 지출 확인하기

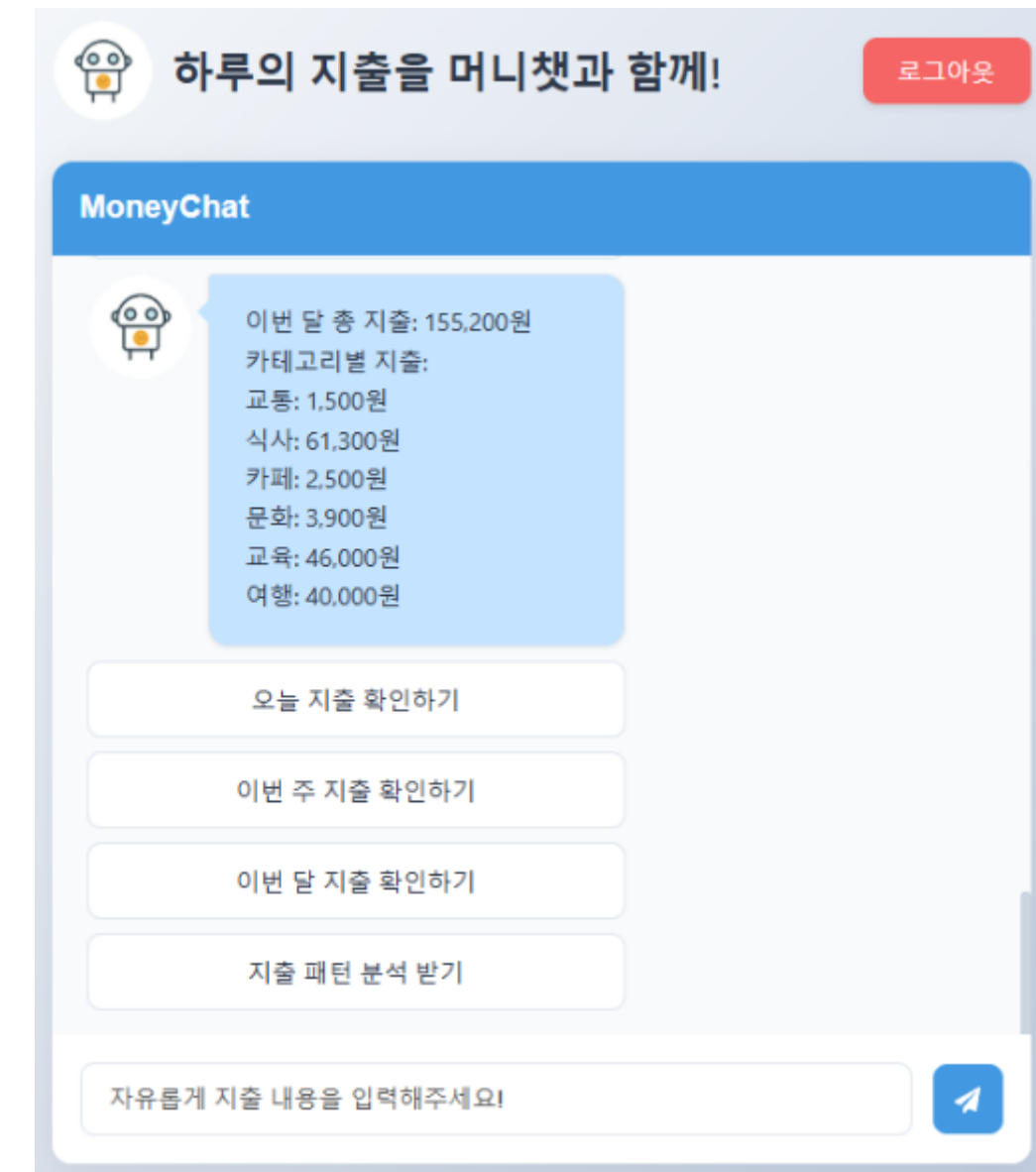
이번 주 지출 확인하기

이번 달 지출 확인하기

지출 패턴 분석 받기


자유롭게 지출 내용을 입력해주세요! 

버튼 클릭 한 번으로, 한 눈에 볼 수 있는  
간단한 지출 내역 확인



하루의 지출을 머니챗과 함께! [로그아웃](#)

MoneyChat


 이번 달 총 지출: 155,200원  
카테고리별 지출:  
교통: 1,500원  
식사: 61,300원  
카페: 2,500원  
문화: 3,900원  
교육: 46,000원  
여행: 40,000원

오늘 지출 확인하기

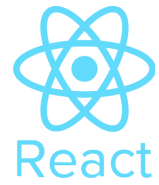
이번 주 지출 확인하기

이번 달 지출 확인하기

지출 패턴 분석 받기

자유롭게 지출 내용을 입력해주세요! 

## 프론트



React-chatbot-kit

ChatbotPage

MessageParser

ActionProvider



Firebase Hosting

## 백엔드



Node.js + Express



Backend Hosting

## API



OpenAI GPT-3.5

Message Analysis

Spending Analysis

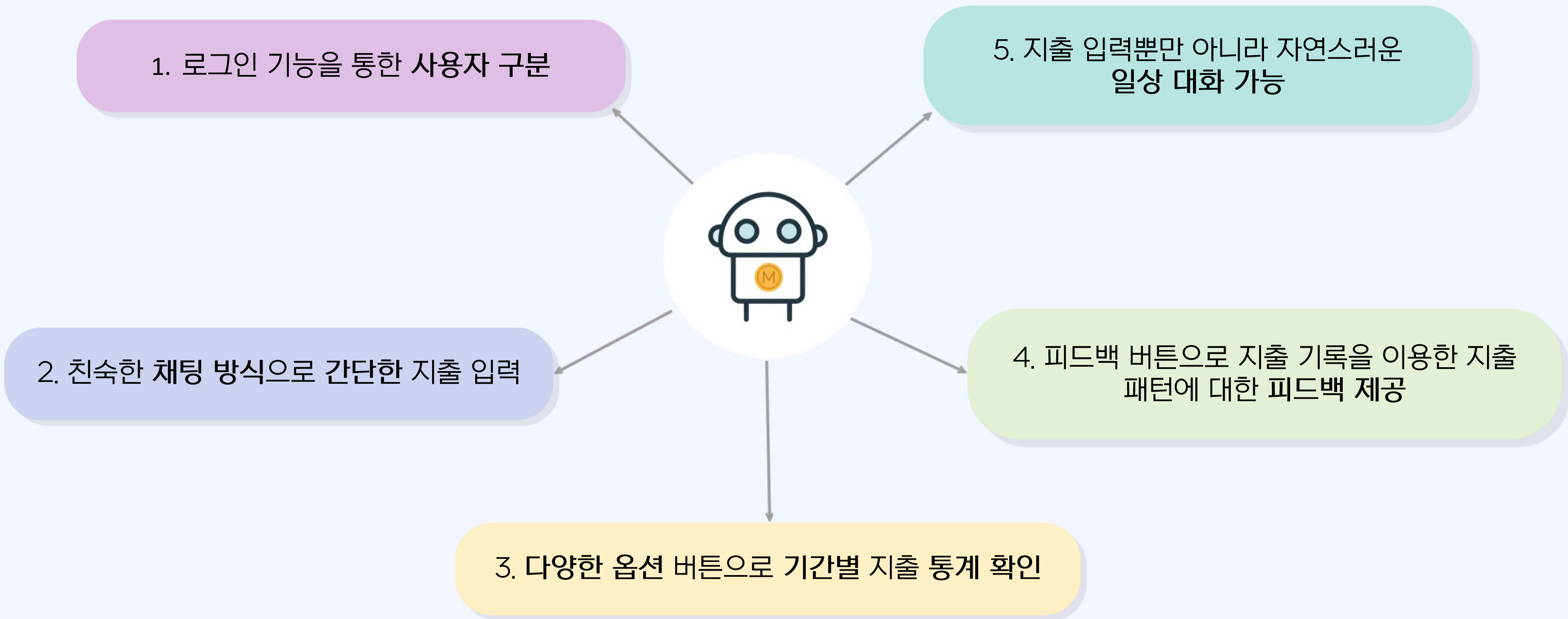


Firebase Authentication

## DB



Firebase Firestore



## Firebase를 이용한 사용자 로그인 관리

```
// 회원가입
const handleSignup = async (e) => {
  e.preventDefault();
  try {
    const userCredential = await createUserWithEmailAndPassword(auth, email, password);
    const user = userCredential.user;

    // Firestore에 사용자의 닉네임 저장
    await setDoc(doc(db, "users", user.uid), {
      // nickname: nickname,
      email: email
    });
    alert('환영합니다!');
    navigate('/');
  } catch (error) {
    alert('회원가입 실패: ' + error.message);
  }
};
```

```
// 로그인
const handleLogin = async (e) => {
  e.preventDefault();
  try {
    // 모듈식으로 signInWithEmailAndPassword 호출
    await signInWithEmailAndPassword(auth, email, password);
    navigate('/chatbot'); // 챗봇 화면으로 이동
  } catch (error) {
    alert('로그인 실패: ' + error.message);
  }
};
```

```
// 로그인 확인
useEffect(() => {
  const unsubscribe = auth.onAuthStateChanged((user) => {
    if (!user) {
      navigate('/'); // 메인화면(로그인)으로 이동
    }
  });

  return () => unsubscribe();
}, [navigate]);

// 로그아웃
const handleLogout = async () => {
  try {
    await signOut(auth);
    alert("로그아웃 되었습니다!");
    navigate('/');
  } catch (error) {
    console.error("로그아웃 실패: ", error);
  }
};
```



## 사용자가 입력한 메시지를 GPT API에 메시지 분석 요청

```
// 사용자 입력 메시지를 분석하고 처리하는 함수
async handleMessage(message) {
  console.log("ActionProvider handling message:", message);
  try {
    // 빈 메시지 체크
    if (!message.trim()) {
      throw new Error('메시지가 비어있습니다.');
```

gpt-4o

```
    }

    // API를 통해 메시지 분석 요청
    const response = await fetch('https://moneychat', {
      method: 'POST',
      headers: {
        'Content-Type': 'application/json',
        'Accept': 'application/json'
      },
      body: JSON.stringify({ message })
    });

    console.log('Response status:', response.status);
    if (!response.ok) {
      const errorData = await response.text();
      console.error('Error response:', errorData);
      throw new Error(`HTTP error! status: ${response.status}`);
    }

    const analysis = await response.json();
    console.log("Message analysis:", analysis);
```

ganglike, 3주 전 · 배포 시도2

```
    // 지출 정보가 포함된 경우와 아닌 경우를 구분하여 처리
    if (analysis.hasExpense && analysis.amount && analysis.category) {
      // 지출 정보를 DB에 저장
      await this.saveExpense(analysis.subject, analysis.category, analysis.amount);
    }
  }
}
```

```
    // 지출 정보가 포함된 경우와 아닌 경우를 구분하여 처리
    if (analysis.hasExpense && analysis.amount && analysis.category) {
      // 지출 정보를 DB에 저장
      await this.saveExpense(analysis.subject, analysis.category, analysis.amount);

      // 지출 정보와 간단한 피드백을 포함한 응답 메시지 생성
      const responseMessage = this.createChatBotMessage(
        `${analysis.subject}(${analysis.category}) 항목에 ${analysis.amount.toLocaleString()}원을  
지출하셨네요!\n${analysis.feedback}`,
        {
          widget: "options",
        }
      );
      this.updateChatbotState(responseMessage);
    } else {
      // 일반(일상) 대화에 대한 응답 메시지 생성
      const defaultMessage = this.createChatBotMessage(analysis.feedback,
        {
          widget: "options",
        }
      );
      this.updateChatbotState(defaultMessage);
    }
  } catch (error) {
    console.error("Error in handleMessage:", error);
    // 에러 발생 시 사용자에게 알림
    const errorMessage = this.createChatBotMessage(
      "죄송합니다. 처리 중 문제가 발생했어요. 다시 시도해주세요.",
      {
        widget: "options",
      }
    );
    this.updateChatbotState(errorMessage);
  }
}
```



## 사용자의 한 달 데이터를 바탕으로 피드백 생성 요청

```
// 이번 달 지출에 대한 분석 피드백을 생성하는 함수
async handleExpenseFeedback() {
  try {
    // 사용자 로그인 상태 확인
    // 오류 대비 혹시 몰라 설정함
    const user = auth.currentUser;
    if (!user) {
      this.updateChatbotState(this.createChatBotMessage(
        "로그인이 필요한 서비스입니다.",
        {
          widget: "options",
        }
      ));
      return;
    }

    // 이번 달 지출 내역 조회
    const monthSummary = await this.calculateExpenseSummary('month');
    console.log('Month summary:', monthSummary);

    // 지출 내역이 없는 경우 처리
    if (monthSummary.total === 0) {
      this.updateChatbotState(this.createChatBotMessage(
        "아직 이번 달 지출 내역이 없습니다.",
        {
          widget: "options",
        }
      ));
      return;
    }

    // 일평균 지출 계산
    const today = new Date();
    const daysInMonth = today.getDate();
    const dailyAverage = monthSummary.total / daysInMonth;

    // API 요청 데이터 준비
    const requestData = {
      total: monthSummary.total,
      dailyAverage,
      byCategory: monthSummary.byCategory,
      daysInMonth
    };
  }
}
```

```
// API를 통해 지출 분석 요청
const response = await fetch('https://moneychat', {
  method: 'POST',
  headers: {
    'Content-Type': 'application/json',
    'Accept': 'application/json'
  },
  body: JSON.stringify(requestData)
});

if (!response.ok) {
  throw new Error(`HTTP error! status: ${response.status}`);
}

// 분석 결과를 포매팅하여 표시
const data = await response.json();
const formattedFeedback = data.feedback
  .split(/(?:\d+\.\s)/)
  .filter(text => text.trim())
  .map(text => text.trim())
  .join('\n\n');

const feedbackMessage = this.createChatBotMessage(formattedFeedback,
  {
    widget: "options",
  });
this.updateChatbotState(feedbackMessage);

catch (error) {
  console.error("Error getting feedback:", error);
  const errorMessage = this.createChatBotMessage(
    "죄송합니다. 피드백을 생성하는 중 문제가 발생했어요. 다시 시도해주세요.",
    {
      widget: "options",
    }
  );
  this.updateChatbotState(errorMessage);
}
```

## GPT 프롬프트 구성

```
const { message } = req.body;
console.log('Analyzing message:', message);

// GPT-3.5 모델을 사용하여 메시지 분석
// 원하는 챗봇의 기능 프롬프트 작성
const response = await openai.chat.completions.create({
  model: "gpt-3.5-turbo", // gptlike, 3주 전 배포 시도
  messages: [{
    role: "system",
    content: `당신은 친근하고 도움이 되는 챗봇 AI 도우미입니다.
    사용자의 메시지에서 지출 관련 정보를 추출하고, 주제와 카테고리로 정리해주세요. 또한, 다른 일상적인 대화에도
    자연스럽게 응답할 수 있습니다.

    카테고리 정리 규칙:
    1. 원본 주제는 사용자가 입력한 실제 지출 항목 (예: 나이키 신발, 아메리카노)
    2. 카테고리는 더 넓은 분류 (예: 패션, 카페)
    3. 기본 카테고리: 식사, 카페, 교통, 패션, 문화, 의료, 교육, 생활 등

    일상적인 대화 규칙:
    1. 자연스럽게 친근한 톤으로 응답
    2. 대화 맥락을 고려한 적절한 답변 제공
    3. 가능한 한 지출 관리나 재무 관련 주제로 자연스럽게 연결
    4. 사용자의 메시지에 지출 관련 정보가 포함되지 않아도 자연스러운 답변 제공

    응답은 다음 JSON 형식으로 제공:
    {
      "hasExpense": boolean,
      "amount": number | null,
      "subject": string | null,
      "category": string | null,
      "feedback": string
    }
  }, {
    role: "user",
    content: message
  }],
  response_format: { type: "json_object" } // JSON 형식으로 응답 요청
});

// 분석 결과 파싱 및 응답
const analysis = JSON.parse(response.choices[0].message.content);
console.log('Analysis result:', analysis);
```

```
// 월별 지출 패턴 분석 및 피드백 제공
app.post('/api/analyze-spending', async (req, res) => {
  try {
    console.log('Request received:', req.body);
    // 필요한 데이터 추출
    const { total, dailyAverage, byCategory, daysInMonth } = req.body;

    // 필수 데이터 검증
    if (!total || total !== 0) {
      return res.status(400).json({ error: 'Total amount is required' });
    }

    // GPT 모델에 전달할 메시지 구성
    const messages = [
      {
        role: "system",
        content: "당신은 친근하고 전문적인 재무 상담사입니다. 사용자의 지출을 분석하고 실용적인 조언을 제공해주세요."
      },
      {
        role: "user",
        content: `현재 사용자의 지출 현황을 분석해주세요:
        - 이번 달 총 지출: ${total.toLocaleString()}원
        - 일일 평균 지출: ${Math.round(dailyAverage).toLocaleString()}원
        - 경과 일수: ${daysInMonth}일
        - 카테고리별 지출:
          ${Object.entries(byCategory)
            .map(([category, amount]) => ` - ${category}: ${amount.toLocaleString()}원`)
            .join('\n')}

        다음 사항을 포함하여 간단히 각 주제에 대해 1-2줄 정도로 분석해주세요:
        1. 현재 지출 패턴의 특징
        2. 개선이 필요한 부분이 있다면 구체적인 제안
      `
      }
    ];

    // GPT 모델을 사용하여 지출 분석 및 피드백 생성
    const completion = await openai.chat.completions.create({
      model: "gpt-3.5-turbo",
      messages: messages,
      temperature: 0.7, // 응답의 창의성 조절
      max_tokens: 500, // 응답 길이 제한
      top_p: 1,
      frequency_penalty: 0,
      presence_penalty: 0
    });

    // 분석 결과 응답
    res.json({
      feedback: completion.choices[0].message.content
    });
  } catch (openaiError) {
    console.error('OpenAI API Error:', openaiError);
    throw new Error(`OpenAI API Error: ${openaiError.message}`);
  }
});
```



<https://moneychat-3155a.web.app/>



## 사용성

- 우리가 매일 수십번씩 사용하는 채팅이라는 방식을 사용해 친숙하고 편리한 사용 가능
- 복잡한 입력이 아니라도 정말 간단히 지출 입력 가능
- 손쉬운 사용
- PWA 설정으로 웹 앱으로도 사용가능

## 시장성

- 기존에 복잡한 입력과 낮은 가시성으로 사용하기 번거롭고 어려운 것에 비해 간단하고 쉬운 지출 관리
- 실제로 사용해보며 편리하다고 느낌
- 친구 : 간단해서 좋다. 캐릭터가 귀엽다. 로그인 기능이 있어 일회성이 아니라 좋다.



### 실사용

- 꾸준히 직접 사용해보며 필요한 기능이나 불편한 점, 부족한 예외 처리, 버그, 편의사항 추가 검토

### 다양한 기능

- 기간별 지출 통계와 피드백 뿐만 아니라 다양한 기능 추가
- 더욱 사용하기 편리하도록 UI 수정

### 사용자의 피드백

- 친구나 가족에게 홍보하여 사용하도록 독려
- 사용자들의 불편한 점이나 피드백 수용하여 더욱 사용하기 좋은 서비스로 성장

- 프로젝트 배포 웹사이트 주소 :  
<https://moneychat-3155a.web.app/>
- 프로젝트 깃허브 레포지토리 :  
<https://github.com/ganglike248/MoneyChat>

# | 감사합니다.

