K-Beauty Token (KBT)

해외 성형 관광객을 위한 블록체인 리워드 플랫폼

Version 1.0 | 2025년 1월

목차

---

1. Executive Summary

K-Beauty Token(KBT)은 한국 성형 의료관광 산업에 특화된 블록체인 기반 리워드 시스템입니다.

핵심 가치 제안

대상	혜택
고객	시술비 최대 15% 리워드, 제휴 서비스 할인, VIP 혜택
성형외과	해외 고객 유치 마케팅 채널, 재방문 유도, 고객 데이터 인사이트
제휴 파트너	고소득 해외 고객층 접근, 크로스 마케팅 기회

토큰 흐름

```
시술 결제 → 토큰 적립 → 다음 시술 할인
                → 제휴 호텔/항공 할인
                → 친구 추천 보너스
                → VIP 등급 혜택
```

---

2. 시장 분석

2.1 한국 성형 의료관광 시장

한국은 세계 최고 수준의 성형 기술과 합리적인 가격으로 '성형 관광의 메카'로 자리 잡았습니다.

지표	현재 (2024)	예상 (2028)
연간 외국인 환자 수	약 50만 명	약 80만 명
시장 규모	약 3조 원	약 5조 원
평균 시술 비용	500~800만 원	600~1,000만 원
재방문율	약 25%	목표 40%

2.2 주요 타겟 국가

순위	국가	비중	특징
1	중국	35%	고가 시술 선호
2	동남아	25%	급성장 시장
3	일본	20%	자연스러운 시술 선호, 높은 재방문율
4	중동	10%	초고가 시술, VIP 서비스 중시
5	구미권	10%	품질 중시, SNS 영향력

---

3. 비즈니스 모델

3.1 생태계 구조

참여자	역할	인센티브
고객	시술 결제, 토큰 적립/사용	리워드, 할인, VIP 혜택
성형외과	토큰 발행, 결제 수령	고객 유치, 재방문 증가
제휴 파트너	토큰 수령, 서비스 제공	신규 고객층 확보
플랫폼	시스템 운영, 토큰 관리	거래 수수료, 데이터 가치

3.2 수익 모델

수익원	설명	예상 비중
거래 수수료	토큰 거래 시 1~2% 수수료	40%
가맹점 수수료	제휴 병원/업체 월정액 또는 거래 기반	30%
프리미엄 서비스	VIP 등급 구독료, 프리미엄 기능	20%
데이터 인사이트	익명화된 시장 분석 리포트 판매	10%

---

4. 토큰 이코노미

4.1 토큰 기본 정보

항목	내용
토큰명	K-Beauty Token
심볼	KBT
총 발행량	1,000,000,000 KBT (10억 개, 고정)
소수점	18 decimals
네트워크	Polygon (ERC-20 호환)
초기 가치 기준	1 KBT = 100원 (소프트 페깅)

4.2 토큰 배분

항목	비율	수량	베스팅
리워드 풀	40%	4억 KBT	5년간 점진적 방출
파트너 인센티브	20%	2억 KBT	3년간 마일스톤 기반
팀 & 어드바이저	15%	1.5억 KBT	6개월 클리프, 24개월 베스팅
운영 준비금	15%	1.5억 KBT	필요시 방출 (멀티시그)
투자자 (Seed)	5%	0.5억 KBT	3개월 클리프, 12개월 베스팅
투자자 (Series A)	5%	0.5억 KBT	즉시 10%, 나머지 12개월

4.3 적립률 구조

시술 금액 × 회원 등급별 적립률

시술 금액	SILVER	GOLD	PLATINUM	DIAMOND
100만원 미만	5%	6%	7%	8%
100~500만원	7%	8%	10%	11%
500~1,000만원	10%	11%	12%	13%
1,000만원 이상	12%	13%	14%	15%

회원 등급 기준

등급	누적 적립 기준	시술 횟수	추가 혜택
🥈 SILVER	가입 시	-	기본 적립
🥇 GOLD	50,000 KBT	2회 이상	통역 서비스
💎 PLATINUM	200,000 KBT	5회 이상	공항 픽업, 우선 예약
👑 DIAMOND	500,000 KBT	10회 이상	전담 컨시어지, 호텔 업그레이드

보너스 적립 이벤트

이벤트	보너스
첫 시술 보너스	기본 적립률 + 5% 추가
친구 추천 보너스	추천인/피추천인 각 25,000 KBT
재방문 보너스	12개월 내 재방문 시 적립률 +3%
시즌 프로모션	성수기(여름/겨울) 더블 적립

4.4 토큰 사용처

카테고리	사용처	사용 조건
성형외과	시술비 결제	총액의 최대 30%까지 토큰 결제
호텔	숙박비 할인	제휴 호텔 최대 20% 토큰 결제
뷰티/쇼핑	올리브영 등 뷰티샵	100% 토큰 결제 가능
항공	마일리지 전환	1 KBT = 10 마일리지
프리미엄	VIP 서비스 구매	통역, 픽업, 컨시어지

4.5 토큰 소각 메커니즘

자동 소각

소각 유형	소각 비율	트리거 조건
거래 수수료 소각	수수료의 50%	모든 토큰 사용 거래 시
만료 토큰 소각	100%	24개월 미사용 토큰
파트너 보증금 소각	위약금의 100%	파트너 계약 위반 시

분기별 바이백 & 소각

- **소각 재원:** 플랫폼 순이익의 20%
- **실행 주기:** 매 분기말 (3월, 6월, 9월, 12월)

- **방법:** 시장에서 토큰 매입 후 즉시 소각
- **투명성:** 소각 내역 블록체인에 기록, 월간 리포트 공개

예상 소각 시나리오

연도	예상 소각량	누적 소각량	유통량 대비
Year 1	500만 KBT	500만 KBT	0.5%
Year 2	2,000만 KBT	2,500만 KBT	2.5%
Year 3	5,000만 KBT	7,500만 KBT	7.5%
Year 5	1억 KBT	2억 KBT	20%

---

5. 기술 구현

5.1 블록체인 선택 근거

Polygon 네트워크 선택 이유:

항목	설명
낮은 거래 비용	건당 $0.01 미만의 가스비
빠른 처리 속도	평균 2초 블록 타임
이더리움 호환성	ERC-20 표준, 주요 지갑 지원
검증된 보안	이더리움 레이어2로서 높은 보안성

5.2 스마트 컨트랙트 구조

KBT 토큰 컨트랙트 (ERC-20)

```solidity
```

```solidity
// SPDX-License-Identifier: MIT
pragma solidity ^0.8.19;

import "@openzeppelin/contracts/token/ERC20/ERC20.sol";
import "@openzeppelin/contracts/token/ERC20/extensions/ERC20Burnable.sol";
import "@openzeppelin/contracts/access/AccessControl.sol";

contract KBTToken is ERC20, ERC20Burnable, AccessControl {
    bytes32 public constant MINTER_ROLE = keccak256("MINTER_ROLE");
    uint256 public constant MAX_SUPPLY = 1_000_000_000 * 10**18;

    constructor() ERC20("K-Beauty Token", "KBT") {
        _grantRole(DEFAULT_ADMIN_ROLE, msg.sender);
        _grantRole(MINTER_ROLE, msg.sender);
    }

    function mint(address to, uint256 amount) external onlyRole(MINTER_ROLE) {
        require(totalSupply() + amount <= MAX_SUPPLY, "Exceeds max supply");
        _mint(to, amount);
    }
}
```

리워드 풀 컨트랙트

```solidity
```

```
// SPDX-License-Identifier: MIT
pragma solidity ^0.8.19;

import "@openzeppelin/contracts/access/AccessControl.sol";
import "@openzeppelin/contracts/security/ReentrancyGuard.sol";

contract RewardPool is AccessControl, ReentrancyGuard {
    KBTToken public token;

    mapping(address => uint256) public rewards;
    mapping(address => bool) public partners;
    mapping(address => uint8) public userTier; // 0: Silver, 1: Gold, 2: Platinum, 3: Diamond

    // 적립률 (basis points): 500 = 5%
    uint256[4] public baseRates = [500, 700, 1000, 1200]; // 금액대별
    uint256[4] public tierBonus = [0, 100, 200, 300];     // 등급별 보너스

    event RewardEarned(address indexed user, uint256 amount, address indexed partner);
    event RewardClaimed(address indexed user, uint256 amount);

    modifier onlyPartner() {
        require(partners[msg.sender], "Not a partner");
        _;
    }

    constructor(address _token) {
        token = KBTToken(_token);
        _grantRole(DEFAULT_ADMIN_ROLE, msg.sender);
    }

    function earnReward(
        address user,
        uint256 paymentAmount,
        uint8 amountTier // 0: <100만, 1: 100-500만, 2: 500-1000만, 3: >1000만
    ) external onlyPartner {
        uint256 rate = baseRates[amountTier] + tierBonus[userTier[user]];
        uint256 reward = (paymentAmount * rate) / 10000;
        rewards[user] += reward;
        emit RewardEarned(user, reward, msg.sender);
    }

    function claimReward() external nonReentrant {
        uint256 amount = rewards[msg.sender];
        require(amount > 0, "No rewards to claim");
```

```
        rewards[msg.sender] = 0;
        token.transfer(msg.sender, amount);
        emit RewardClaimed(msg.sender, amount);
    }

    function addPartner(address partner) external onlyRole(DEFAULT_ADMIN_ROLE) {
        partners[partner] = true;
    }

    function updateUserTier(address user, uint8 tier) external onlyRole(DEFAULT_ADMIN_ROLE) {
        require(tier <= 3, "Invalid tier");
        userTier[user] = tier;
    }
}
```

VIP 등급 컨트랙트

solidity

```solidity
// SPDX-License-Identifier: MIT
pragma solidity ^0.8.19;

contract VIPTier {
    enum Tier { Silver, Gold, Platinum, Diamond }

    struct UserInfo {
        Tier tier;
        uint256 totalEarned;
        uint256 treatmentCount;
        uint256 lastActivity;
    }

    mapping(address => UserInfo) public users;

    uint256[4] public tierThresholds = [
        0,          // Silver: 가입 시
        50000e18,   // Gold: 50,000 KBT
        200000e18,  // Platinum: 200,000 KBT
        500000e18   // Diamond: 500,000 KBT
    ];

    function updateTier(address user) external {
        UserInfo storage info = users[user];

        if (info.totalEarned >= tierThresholds[3] && info.treatmentCount >= 10) {
            info.tier = Tier.Diamond;
        } else if (info.totalEarned >= tierThresholds[2] && info.treatmentCount >= 5) {
            info.tier = Tier.Platinum;
        } else if (info.totalEarned >= tierThresholds[1] && info.treatmentCount >= 2) {
            info.tier = Tier.Gold;
        } else {
            info.tier = Tier.Silver;
        }
    }

    function recordTreatment(address user, uint256 earnedAmount) external {
        UserInfo storage info = users[user];
        info.totalEarned += earnedAmount;
        info.treatmentCount += 1;
        info.lastActivity = block.timestamp;
    }
}
```

토큰 소각 컨트랙트

solidity

```solidity
// SPDX-License-Identifier: MIT
pragma solidity ^0.8.19;

contract BurnMechanism {
    KBTToken public token;

    uint256 public totalBurned;
    uint256 public constant EXPIRE_PERIOD = 730 days; // 24개월

    mapping(address => uint256) public lastActivity;

    event TokensBurned(uint256 amount, string reason);

    // 거래 수수료 소각 (50%)
    function burnTransactionFee(uint256 feeAmount) external {
        uint256 burnAmount = feeAmount / 2;
        token.burn(burnAmount);
        totalBurned += burnAmount;
        emit TokensBurned(burnAmount, "Transaction Fee");
    }

    // 만료 토큰 소각
    function burnExpiredTokens(address user) external {
        require(
            block.timestamp > lastActivity[user] + EXPIRE_PERIOD,
            "Tokens not expired"
        );
        uint256 balance = token.balanceOf(user);
        // 실제 구현에서는 관리자 권한 필요
        totalBurned += balance;
        emit TokensBurned(balance, "Expired");
    }

    // 분기별 바이백 소각
    function quarterlyBuybackBurn(uint256 amount) external {
        token.burn(amount);
        totalBurned += amount;
        emit TokensBurned(amount, "Quarterly Buyback");
    }
}
```

5.3 개발 환경 설정

필수 도구

```bash
# Node.js v18 이상 설치 후
mkdir kbt-token && cd kbt-token
npm init -y
npm install --save-dev hardhat @openzeppelin/contracts
npm install --save-dev @nomicfoundation/hardhat-toolbox
npx hardhat init
```

Hardhat 설정 (hardhat.config.js)

```javascript
require("@nomicfoundation/hardhat-toolbox");
require("dotenv").config();

module.exports = {
  solidity: "0.8.19",
  networks: {
    mumbai: {
      url: process.env.MUMBAI_RPC_URL,
      accounts: [process.env.PRIVATE_KEY]
    },
    polygon: {
      url: process.env.POLYGON_RPC_URL,
      accounts: [process.env.PRIVATE_KEY]
    }
  },
  etherscan: {
    apiKey: process.env.POLYGONSCAN_API_KEY
  }
};
```

배포 스크립트 (scripts/deploy.js)

```javascript
```

```
const { ethers } = require("hardhat");

async function main() {
  const [deployer] = await ethers.getSigners();
  console.log("Deploying with:", deployer.address);

  // 1. Deploy KBT Token
  const KBT = await ethers.getContractFactory("KBTToken");
  const kbt = await KBT.deploy();
  await kbt.waitForDeployment();
  console.log("KBT Token:", await kbt.getAddress());

  // 2. Deploy RewardPool
  const Pool = await ethers.getContractFactory("RewardPool");
  const pool = await Pool.deploy(await kbt.getAddress());
  await pool.waitForDeployment();
  console.log("RewardPool:", await pool.getAddress());

  // 3. Deploy VIPTier
  const VIP = await ethers.getContractFactory("VIPTier");
  const vip = await VIP.deploy();
  await vip.waitForDeployment();
  console.log("VIPTier:", await vip.getAddress());

  // 4. Deploy BurnMechanism
  const Burn = await ethers.getContractFactory("BurnMechanism");
  const burn = await Burn.deploy(await kbt.getAddress());
  await burn.waitForDeployment();
  console.log("BurnMechanism:", await burn.getAddress());

  // 5. Grant MINTER_ROLE to RewardPool
  const MINTER_ROLE = await kbt.MINTER_ROLE();
  await kbt.grantRole(MINTER_ROLE, await pool.getAddress());
  console.log("MINTER_ROLE granted to RewardPool");
}

main().catch((error) => {
  console.error(error);
  process.exitCode = 1;
});
```

배포 실행

```
bash
```

```
# 테스트넷 배포
npx hardhat run scripts/deploy.js --network mumbai

# 메인넷 배포 (주의!)
npx hardhat run scripts/deploy.js --network polygon

# 컨트랙트 검증
npx hardhat verify --network polygon <CONTRACT_ADDRESS>
```

5.4 백엔드 API 연동

ethers.js 서비스 클래스

```
javascript
```

```
const { ethers } = require("ethers");

class KBTService {
  constructor() {
    this.provider = new ethers.JsonRpcProvider(process.env.RPC_URL);
    this.wallet = new ethers.Wallet(process.env.PRIVATE_KEY, this.provider);

    this.token = new ethers.Contract(
      process.env.TOKEN_ADDRESS,
      require("./abis/KBTToken.json"),
      this.wallet
    );

    this.pool = new ethers.Contract(
      process.env.POOL_ADDRESS,
      require("./abis/RewardPool.json"),
      this.wallet
    );
  }

  // 잔액 조회
  async getBalance(address) {
    const balance = await this.token.balanceOf(address);
    return ethers.formatEther(balance);
  }

  // 리워드 적립
  async earnReward(userAddress, paymentAmount, amountTier) {
    const tx = await this.pool.earnReward(
      userAddress,
      ethers.parseEther(paymentAmount.toString()),
      amountTier
    );
    return tx.wait();
  }

  // 리워드 조회
  async getPendingReward(address) {
    const reward = await this.pool.rewards(address);
    return ethers.formatEther(reward);
  }

  // 사용자 등급 조회
  async getUserTier(address) {
```

```
    const tier = await this.pool.userTier(address);
    const tiers = ["Silver", "Gold", "Platinum", "Diamond"];
    return tiers[tier];
  }
}

module.exports = KBTService;
```

Express API 예시

```
javascript
```

```
const express = require("express");
const KBTService = require("./services/KBTService");

const app = express();
const kbt = new KBTService();

app.use(express.json());

// 잔액 조회
app.get("/api/balance/:address", async (req, res) => {
  try {
    const balance = await kbt.getBalance(req.params.address);
    res.json({ balance, unit: "KBT" });
  } catch (error) {
    res.status(500).json({ error: error.message });
  }
});

// 리워드 적립 (파트너 전용)
app.post("/api/reward/earn", async (req, res) => {
  try {
    const { userAddress, paymentAmount, amountTier } = req.body;
    const receipt = await kbt.earnReward(userAddress, paymentAmount, amountTier);
    res.json({
      success: true,
      txHash: receipt.hash
    });
  } catch (error) {
    res.status(500).json({ error: error.message });
  }
});

// 사용자 정보
app.get("/api/user/:address", async (req, res) => {
  try {
    const address = req.params.address;
    const [balance, pending, tier] = await Promise.all([
      kbt.getBalance(address),
      kbt.getPendingReward(address),
      kbt.getUserTier(address)
    ]);
    res.json({ balance, pendingReward: pending, tier });
  } catch (error) {
    res.status(500).json({ error: error.message });
```

```
  }
});

app.listen(3000, () => console.log("API Server running on :3000"));
```

5.5 보안 체크리스트

항목	설명	상태
OpenZeppelin 사용	검증된 라이브러리 사용	✅
정적 분석	Slither, Mythril 실행	⬜
전문 감사	CertiK, Trail of Bits	⬜
멀티시그 지갑	Gnosis Safe 사용	⬜
비상 정지	Pausable 구현	⬜
재진입 방지	ReentrancyGuard 적용	✅
Private Key 관리	환경 변수 분리	✅

---

6. 서비스 앱 설계

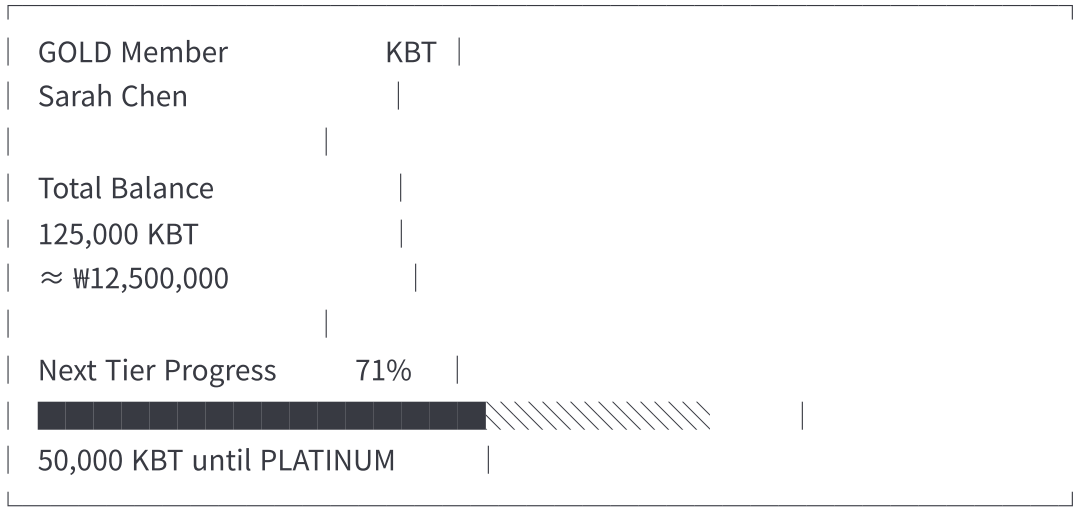
6.1 앱 구조

```
K-Beauty Token App
├── 🏠 홈 (Home)
│   ├── 잔액 카드 (등급, 잔액, 다음 등급 진행률)
│   ├── 퀵 액션 (결제, 리워드, 친구추천, 교환)
│   └── 최근 거래 내역
├── 🏪 파트너 (Partners)
│   ├── 카테고리 필터 (전체/성형외과/호텔/뷰티/항공)
│   └── 파트너 목록 (적립률 표시)
├── 🎁 리워드 (Rewards)
│   ├── 통계 (누적 적립/사용)
│   ├── 이번 달 미션
│   └── VIP 전용 혜택
└── 👤 프로필 (Profile)
    ├── 사용자 정보
```

```
├─── 언어/알림/보안 설정
└─── 고객센터
```

6.2 주요 화면 설계

홈 화면 - 잔액 카드

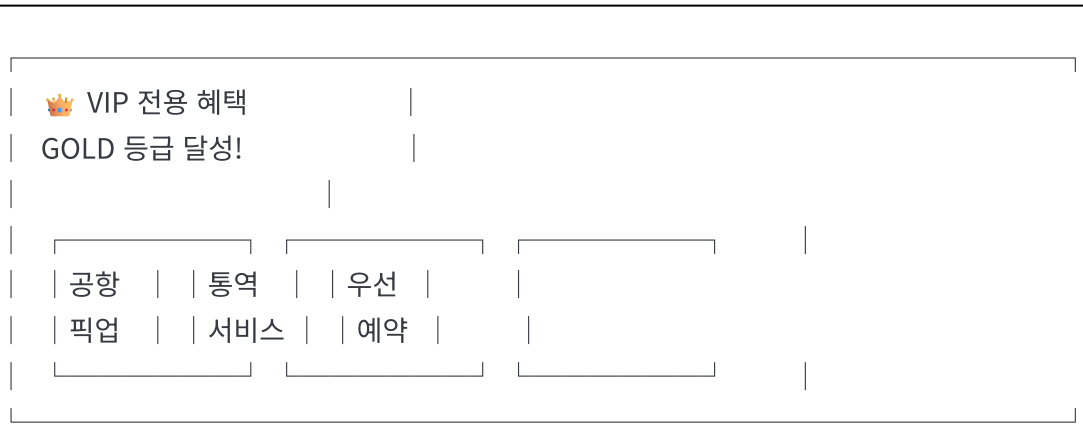


리워드 미션

```
🎯 이번 달 미션

✅ 첫 시술 예약          +5,000 KBT
⬜ 친구 3명 추천 (1/3)   +15,000 KBT
⬜ 앱 리뷰 작성          +2,000 KBT
```

VIP 혜택 카드



6.3 다국어 지원

언어	우선순위	비고
KR 한국어	기본	-
CN 중국어 (간체)	1순위	최대 시장
JP 일본어	2순위	고재방문율
US 영어	3순위	글로벌
VN 베트남어	4순위	성장 시장
SA 아랍어	5순위	VIP 시장

7. 로드맵

전체 로드맵

단계	기간	주요 마일스톤
Phase 1	2025 Q1-Q2	스마트 컨트랙트 개발 및 감사, MVP 앱 개발, 파일럿 병원 5곳 계약
Phase 2	2025 Q3-Q4	베타 런칭, 병원 30곳 확대, 호텔/항공 파트너십
Phase 3	2026 Q1-Q2	정식 런칭, 병원 100곳, 중국/일본 마케팅 강화
Phase 4	2026 Q3-Q4	거래소 상장, 동남아 시장 확대, 프리미엄 기능 출시
Phase 5	2027+	글로벌 확장 (태국, 일본 성형 시장), NFT 인증서, 메타버스 연동

상세 마일스톤

Phase 1: Foundation (2025 Q1-Q2)

- [ ] 스마트 컨트랙트 개발 완료
- [ ] 보안 감사 (CertiK)
- [ ] MVP 앱 개발 (iOS/Android)
- [ ] 파일럿 병원 5곳 MOU
- [ ] 시드 투자 유치 (5억원)

Phase 2: Beta Launch (2025 Q3-Q4)

- [ ] 베타 서비스 런칭
- [ ] 제휴 병원 30곳 확대
- [ ] 호텔 파트너십 (그랜드 인터컨티넨탈 등)
- [ ] 항공사 파트너십 (대한항공/아시아나)
- [ ] 사용자 10,000명 달성

Phase 3: Official Launch (2026 Q1-Q2)

- [ ] 정식 서비스 런칭
- [ ] 제휴 병원 100곳
- [ ] 중국 마케팅 캠페인
- [ ] 일본 마케팅 캠페인
- [ ] 시리즈 A 투자 유치 (20억원)

Phase 4: Scale (2026 Q3-Q4)

- [ ] 거래소 상장 (업비트, 빗썸)
- [ ] 동남아 시장 진출
- [ ] 프리미엄 구독 서비스 출시
- [ ] 사용자 100,000명 달성

Phase 5: Global Expansion (2027+)

- [ ] 태국 성형 시장 진출
- [ ] 일본 성형 시장 진출
- [ ] NFT 시술 인증서 발행
- [ ] 메타버스 가상 상담 연동
- [ ] 시리즈 B 투자 유치 (50억원)

---

8. 팀 및 투자

8.1 필요 인력

직군	인원	역할	예상 연봉
블록체인 개발	2명	스마트 컨트랙트, 블록체인 연동	8,000만원
풀스택 개발	3명	웹/앱 개발, API 서버	7,000만원
BD/마케팅	2명	병원 제휴, 해외 마케팅	6,000만원

직군	인원	역할	예상 연봉
운영/CS	2명	다국어 고객 지원, 운영	4,500만원
PM	1명	프로젝트 관리	7,000만원

연간 인건비 예상: 약 6.5억원

8.2 투자 유치 계획

라운드	목표 금액	시기	사용처
시드	5억 원	2025 Q1	MVP 개발, 파일럿 운영
시리즈 A	20억 원	2026 Q1	정식 런칭, 마케팅, 팀 확장
시리즈 B	50억 원	2027	글로벌 확장, 거래소 상장

8.3 자금 사용 계획 (시드)

항목	비율	금액
개발비	40%	2억 원
마케팅	25%	1.25억 원
운영비	20%	1억 원
법률/컴플라이언스	10%	0.5억 원
예비비	5%	0.25억 원

9. 리스크 및 법적 고려사항

9.1 주요 리스크

리스크	설명	대응 방안
규제 리스크	한국 암호화폐 규제 변화 가능성	법률 자문 지속, 유연한 사업 구조
시장 리스크	성형 관광 수요 변동	다각화된 파트너십, 비성형 의료관광 확대

리스크	설명	대응 방안
기술 리스크	보안 취약점	정기 감사, 버그 바운티 프로그램
경쟁 리스크	유사 서비스 등장	선점 효과, 차별화된 파트너십
환율 리스크	토큰 가치 변동	소프트 페깅, 안정화 기금

9.2 법적 준수사항

한국 규제

- ✅ 특정금융정보법(특금법) 준수
- ✅ VASP(가상자산사업자) 신고
- ✅ 자금세탁방지(AML) 시스템 구축
- ✅ 고객확인제도(KYC) 적용

개인정보보호

- ✅ 개인정보보호법 준수 (다국적 고객 데이터)
- ✅ GDPR 준수 (유럽 고객 대응)
- ✅ 중국 개인정보보호법 준수

의료 관련

- ✅ 의료법 관련 마케팅 규정 준수
- ✅ 의료광고 심의 기준 준수
- ✅ 의료 중개 관련 규정 검토

9.3 컴플라이언스 로드맵

시기	항목
2025 Q1	법률 자문사 선정, VASP 신고 준비
2025 Q2	VASP 신고 완료, AML/KYC 시스템 구축
2025 Q3	개인정보보호 정책 수립, 보안 감사
2026 Q1	정기 컴플라이언스 감사 시작

부록

A. 용어 정의

용어	정의
KBT	K-Beauty Token의 약자
베스팅	토큰을 일정 기간에 걸쳐 점진적으로 지급하는 방식
클리프	베스팅 시작 전 대기 기간
소각	토큰을 영구적으로 유통에서 제거하는 것
바이백	시장에서 토큰을 매입하는 것
멀티시그	여러 서명이 필요한 지갑

B. 참고 자료

- OpenZeppelin Contracts: https://docs.openzeppelin.com
- Hardhat Documentation: https://hardhat.org/docs
- Polygon Documentation: https://wiki.polygon.technology
- ethers.js: https://docs.ethers.org

C. 연락처

- **이메일:** contact@kbeautytoken.io
- **웹사이트:** www.kbeautytoken.io
- **텔레그램:** @KBeautyToken
- **트위터:** @KBeautyToken

면책조항

본 문서는 정보 제공 목적으로 작성되었으며, 투자 권유를 구성하지 않습니다. 암호화폐 투자는 높은 위험을 수반하며, 원금 손실이 발생할 수 있습니다. 투자 결정 전 전문가와 상담하시기 바랍니다.
