SkinSavvy: AI-Powered Skincare Analysis & Recommendation System

Documentation Version: 1.0

Date: February 19, 2025

Project Team: Amit Kumar

Methodology: Agile SDLC

Status: In Development

Table of Contents

SkinSavvy: AI-Powered Skincare Analysis & Recommendation System....1
Documentation Version: 1.0....1
1. Project Overview....3
1.1 Problem Statement....3
1.2 Solution....3
1.3 Key Features....3
1.4 Target Audience....4
2. Agile SDLC Methodology....4
2.1 Agile Framework Implementation....4
2.2 Sprint Structure....4
2.3 Team Roles....4
3. System Architecture....4
3.1 High-Level Architecture....4
3.2 Docker Container Architecture....5
3.3 Data Flow....5
4. Technical Specifications....5
4.1 Frontend Technologies....5
4.2 Backend Technologies....6
4.3 AI/ML Technologies....6
4.4 DevOps Technologies....6
5. Implementation Details....7
5.1 Database Schema....7
5.2 API Endpoints....7
5.3 AI Model Architecture....8
6. Testing Strategy....8
6.1 Test Pyramid Implementation....8
6.2 Test Categories....8
6.3 Test Automation....8
7. Deployment Plan....9
7.1 Environment Strategy....9
7.2 Deployment Process....9
7.3 Infrastructure Requirements....9
8. Project Timeline....10
8.1 Sprint Schedule....10
8.2 Milestones....10
9. Risk Management....10
9.1 Risk Assessment Matrix....10
9.2 Compliance Requirements....11
10. Future Enhancements....11
10.1 Phase 2 Features....11
10.2 Phase 3 Features....11
10.3 Technical Roadmap....11
11. Appendices....11
11.1 Installation Guide....11
11.2 API Documentation....12
11.3 Support Channels....12
11.4 License Information....12

1. Project Overview

1.1 Problem Statement

Current skincare solutions lack personalized, AI-driven analysis capabilities, leading to ineffective product recommendations and poor user experiences.

1.2 Solution

SkinSavvy provides an intelligent skincare analysis platform that uses computer vision and machine learning to analyze skin conditions and recommend personalized products.

1.3 Key Features

•AI-powered skin condition detection
•Personalized product recommendations
•User profile management
•E-commerce integration
•Mobile-responsive design

1.4 Target Audience

•Individuals seeking personalized skincare solutions
•Dermatology clinics
•Skincare product retailers

2. Agile SDLC Methodology

2.1 Agile Framework Implementation

Diagram

Code

2.2 Sprint Structure

Sprint	Duration	Focus Areas
Sprint 1	2 weeks	Project setup, requirements gathering, architecture design
Sprint 2	2 weeks	Backend development, database design
Sprint 3	2 weeks	AI model development and training
Sprint 4	2 weeks	Frontend development, API integration
Sprint 5	2 weeks	Testing, deployment, documentation

2.3 Team Roles

- **Project Owner:** Business requirements
- **Scrum Master:** Process facilitation
- **Development Team:** Implementation
- **QA Engineer:** Testing and validation

3. System Architecture

3.1 High-Level Architecture

Diagram

Code

3.2 Docker Container Architecture

yaml

```yaml
version: '3.8'
services:
  reverse-proxy:
    image: nginx:alpine
    ports: ["80:80", "443:443"]

  frontend:
    build: ./frontend
    environment: [NODE_ENV=production]

  backend:
    build: ./backend
    environment: [DB_HOST=database, AI_SERVICE_URL=ai-service:8501]
```

```
  ai-service:
    build: ./ai-service
    volumes: [models-volume:/models]

  database:
    image: postgres:15-alpine
    volumes: [postgres-data:/var/lib/postgresql/data]

volumes: [postgres-data, models-volume]
```

3.3 Data Flow

1.User uploads image via frontend

2.Image processed through AI service

3.Results stored in database

4.Recommendations generated

5.Response delivered to user

4. Technical Specifications

4.1 Frontend Technologies

Technology	Version	Purpose
React.js	18.2.0	UI Framework
React Native	0.72.0	Mobile App
Chakra UI	2.8.0	Component Library
Axios	1.6.0	HTTP Client

4.2 Backend Technologies

Technology	Version	Purpose
Node.js	18.17.0	Runtime Environment
Express.js	4.18.0	Web Framework
JWT	9.0.0	Authentication
PostgreSQL	15.0	Database

4.3 AI/ML Technologies

Technology	Version	Purpose
TensorFlow	2.15.0	ML Framework
MobileNetV2	-	Base Model
OpenCV	4.8.0	Image Processing

4.4 DevOps Technologies

Technology	Purpose
Docker	Containerization
Docker Compose	Orchestration
GitHub Actions	CI/CD
AWS EC2	Deployment

5. Implementation Details

5.1 Database Schema

sql

```
-- Users Table
CREATE TABLE users (
    id SERIAL PRIMARY KEY,
    email VARCHAR(255) UNIQUE NOT NULL,
    password_hash VARCHAR(255) NOT NULL,
    created_at TIMESTAMP DEFAULT CURRENT_TIMESTAMP
);

-- Analysis History Table
CREATE TABLE analysis_history (
    id SERIAL PRIMARY KEY,
    user_id INTEGER REFERENCES users(id),
    image_url VARCHAR(500),
    condition_type VARCHAR(100),
    confidence_score DECIMAL(5,4),
    created_at TIMESTAMP DEFAULT CURRENT_TIMESTAMP
);

-- Products Table
CREATE TABLE products (
    id SERIAL PRIMARY KEY,
```

```
    name VARCHAR(255) NOT NULL,
    description TEXT,
    ingredients TEXT,
    target_conditions VARCHAR(100)[],
    image_url VARCHAR(500),
    buy_link VARCHAR(500),
    brand VARCHAR(100)
);
```

5.2 API Endpoints

Method	Endpoint	Description
POST	/api/auth/register	User registration
POST	/api/auth/login	User login
POST	/api/analyze	Skin analysis
GET	/api/history	Analysis history
GET	/api/products	Product recommendations

5.3 AI Model Architecture

```python
# Model Configuration
base_model = MobileNetV2(
    weights='imagenet',
    include_top=False,
    input_shape=(224, 224, 3)
)

# Custom Head
model = Sequential([
    base_model,
    GlobalAveragePooling2D(),
    Dense(256, activation='relu'),
    Dropout(0.3),
    Dense(128, activation='relu'),
    Dropout(0.3),
    Dense(1, activation='sigmoid')
])
```

6. Testing Strategy

6.1 Test Pyramid Implementation

Diagram

Code

6.2 Test Categories

Test Type	Tools	Coverage Target
Unit Tests	Jest, pytest	80%
Integration Tests	Supertest	70%
E2E Tests	Cypress	50%
Performance Tests	k6	100%

6.3 Test Automation

•GitHub Actions for CI/CD

•Automated test execution on PR

•Coverage reporting

•Performance monitoring

7. Deployment Plan

7.1 Environment Strategy

Environment	Purpose	URL
Development	Feature testing	dev.skinsavvy.com
Staging	Pre-production	staging.skinsavvy.com
Production	Live users	skinsavvy.com

7.2 Deployment Process

1.Code commit to feature branch

2.Automated testing

3.PR review and approval

4.Merge to main branch

5.Automated deployment to staging

6.Manual approval for production

7.Zero-downtime deployment

7.3 Infrastructure Requirements

Resource	Specification	Quantity
EC2 Instance	t3.medium	2
RDS Instance	db.t3.micro	1
S3 Bucket	Standard	1
CloudFront	CDN	1

8. Project Timeline

8.1 Sprint Schedule

Sprint	Dates	Deliverables
Sprint 1	Jan 1-14	Project setup, requirements
Sprint 2	Jan 15-28	Backend development
Sprint 3	Jan 29-Feb 11	AI model development
Sprint 4	Feb 12-25	Frontend development
Sprint 5	Feb 26-Mar 10	Testing & deployment

8.2 Milestones

- **M1:** Requirements Complete (Jan 14)
- **M2:** Backend Complete (Jan 28)
- **M3:** AI Model Trained (Feb 11)
- **M4:** Frontend Complete (Feb 25)
- **M5:** Production Launch (Mar 10)

9. Risk Management

9.1 Risk Assessment Matrix

Risk	Probability	Impact	Mitigation Strategy
Data privacy concerns	Medium	High	GDPR compliance, data encryption
Model inaccuracy	High	High	Continuous training, human oversight
Scalability issues	Medium	Medium	Load testing, auto-scaling
Integration failures	Low	Medium	Fallback mechanisms, monitoring

9.2 Compliance Requirements

•GDPR compliance
•HIPAA considerations (medical data)
•PCI DSS (if handling payments)
•ADA accessibility standards

10. Future Enhancements

10.1 Phase 2 Features

•Multi-language support
•Advanced analytics dashboard
•Social features community
•Professional dermatologist reviews

10.2 Phase 3 Features

•AR skin analysis
•IoT device integration
•Subscription model
•White-label solutions

10.3 Technical Roadmap

•Microservices architecture

•Kubernetes orchestration

•Machine learning pipeline

•Real-time notifications

11. Appendices

11.1 Installation Guide

```bash
# Clone repository
git clone https://github.com/username/skinsavvy.git

# Setup with Docker
docker-compose up --build

# Manual setup
cd backend && npm install
cd ../frontend && npm install
```

11.2 API Documentation

Full API documentation available at:

/api/docs endpoint with Swagger UI

11.3 Support Channels

•Email: support@skinsavvy.com

•GitHub Issues: Bug reports

•Documentation: docs.skinsavvy.com

11.4 License Information