Biophotovoltaic System Specification

Author :- Neha Mahajan

1. Introduction

1.1 Purpose

This document defines the objectives, scope, architecture, requirements, and deliverables of the Biophotovoltaic System—a full-stack application that allows users worldwide to estimate renewable energy generation from fruit-juice-based biophotovoltaic panels, based on climate, location, fruit chemistry, and device requirements.

1.2 Intended Audience

- Developers / Engineering Team
- Project Managers / Product Leads
- Potential Stakeholders or Collaborators
- Testers / QA Engineers
- Investors / Reviewers (for viability or feasibility checks)

1.3 Scope & Goals

Provide a global, web-based interface to compute potential solar energy generation using fruit-juice electrolytes + UV/light data. Offer climate-adaptive fruit recommendations based on user's city/weather. Allow users to configure panel size, device-power needs, and output expected energy + cost + environmental metrics.

2. Background & Motivation

Traditional solar (silicon-based) has high production/embodied-energy and environmental impact. Biophotovoltaic approach (using fruit juices / natural electrolytes)—though experimental—has shown promising lab results (e.g. power densities around 0.8–1.08 mW/cm², membraneless cells using fruit juices, natural redox species like NADPH/flavins). A software tool to simulate, plan, and analyze fruit-based solar installations globally can help researchers, hobbyists, eco-enthusiasts, and early adopters evaluate feasibility before any hardware commitment. It integrates climate data, chemical data, economic data—enabling informed decision-making for sustainable energy generation with local resources.

3. High-Level System Overview / Architecture

3.1 Major Components

- Frontend (Client): React + TypeScript + Three.js (for 3D UI) + CSS / styling + data visualization libraries (charts, graphs)
- Backend (Server): Node.js + Express—acts as API gateway & orchestration layer for multiple external APIs, performs calculations, serves results.
- Database (Shared): Schemas for Fruit Data, Regional/Electrolyte Data, Device Data—stored (e.g., in MongoDB Atlas).
- External APIs: Weather & climate data, geographic / location data, fruit/agricultural data, economic / pricing data, translation/localization data, device energy data, environmental data.

3.2 Data Flow & Interfaces

User enters a city or uses geolocation (frontend) → frontend calls backend endpoint → backend uses geographic + weather APIs → fetch climate & location → climatic parameters forwarded to fruit-selection engine → fruit database (with chemical properties) + economic data fetched → backend returns a set of fruit recommendations + predicted power output & cost for different panel configurations. For user-chosen fruit + panel size + device category → backend computes energy generation forecast, juice & resin requirements, cost analysis, lifespan & degradation schedule, device compatibility, environmental metrics → results sent to frontend → displayed via 3D visualization + dashboards + charts.

3.3 Module / Directory Structure

text

/client — React frontend (UI components)

/server — Express backend + API routes

/shared (or planned) — Data schemas: fruitSchema, deviceSchema, regionSchema, etc.

4. Requirements Specification

4.1 Functional Requirements

Feature / Endpoint	**Description / Behavior**
4.1.1 Welcome / Home UI	3D animated logo, welcome screen with tagline, smooth transition to city input screen
4.1.2 Location Input & Weather Fetch	Autocomplete city search + geolocation detection → fetch current weather, UV index, forecast

Feature / Endpoint	Description / Behavior
4.1.3 Climate-Adaptive Fruit Recommendation	Given location & weather, algorithm selects top fruits optimizing for local climate (sunny vs cloudy, seasonal variations, fruit chemical properties, availability, cost)
4.1.4 Fruit Properties Retrieval	Given fruit name & location, return full chemical & physical properties (pH, conductivity, pigment levels, redox potential, cost/kg, etc.)
4.1.5 Panel Configuration Feature	Allow user to input panel dimensions (sq ft), choose fruit, pick device category / desired device(s)
4.1.6 Energy & Cost Calculator	Compute juice + resin required, UV/light-based generation estimates (hourly/daily), cost analysis, ROI, maintenance schedule, degradation over time
4.1.7 Results Dashboard & Visualization	Show 3D model of panel + charts/gauges for power output, cost, environmental impact, maintenance schedule, device compatibility
4.1.8 Chemical & Environmental Analysis View	pH, acidity, conductivity, redox potential, pigment data; carbon footprint comparison vs silicon; sustainability metrics
4.1.9 Multi-language / Localization Support	Translate fruit names, UI labels, data using translation APIs (for global users)
4.1.10 API Gateway & External Integration	Orchestrate multiple external APIs (weather, geography, fruit/agriculture database, pricing, environment) with fallback & caching / rate limiting
4.1.11 Deployment & Config	Provide environment variable support, build scripts, deployment instructions (frontend & backend)

4.2 Non-Functional Requirements

- Performance: API responses should return within acceptable latency (e.g. < 2s for weather + climate + fruit recommendation fetch)

- Scalability: Able to handle multiple concurrent users globally; caching and rate-limiting to avoid API overuse
- Maintainability: Clean modular architecture, clear separation between services, schema-defined data modeling, version control
- Usability: Responsive UI, smooth 3D interactions, fallback for devices with limited WebGL support
- Extensibility: Easy addition of new APIs, new fruits, more device categories, localization to additional languages
- Reliability: Error handling for API failures, network issues, invalid inputs; fallback mechanisms and graceful degradation

4.3 Assumptions & Constraints

External APIs used (weather, fruit data, economic data) will remain accessible and free (or under free-tier limits). Fruit chemical / electrochemical data will be available or approximated reliably. Users have internet access and modern browsers (supporting WebGL / Three.js). Electrolyte-based generation remains hypothetical/experimental—energy estimates are for hypothetical/demonstration purposes, not guaranteed real-world performance.

5. Data Models / Schemas

typescript

```
interface Fruit {
  fruitName: string;
  scientificName: string;
  phLevel: number;
  acidity: 'low' | 'medium' | 'high';
  conductivity: number;
  redoxPotential: number;
  efficiency: number;
  costPerKg: number;
  juiceRequiredPerSqFt: number;
  resinRatio: number;
  powerDensityPerSqFt: number;
```

```
climateSpecialization: 'cloudy' | 'sunny' | 'versatile';
lowLightEfficiency: number;
highUVEfficiency: number;
photosyntheticPigments: {
  chlorophyll: number;
  carotenoids: number;
  anthocyanins: number;
  betalains: number;
  flavonoids: number;
};
activationThreshold: {
  uvIndex: number;
  lightIntensity: number;
  cloudCoverMax: number;
};
regionalOptimization: {
  bestClimateZones: string[];
  seasonalPerformance: Record<string, number>;
  humidityTolerance: number;
  temperatureRange: { min: number; max: number };
};
uvTriggeredGeneration: boolean;
operationalLifespan: number;
degradationRateMonthly: number;
environmentalAdaptation: number;
ionConductivity: number;
availabilityByRegion: Record<string, any>;
seasonalAvailability: string[];
```

```
  panelReplacementInterval: number;
  resinCuringTime: number;
  installationComplexity: 'simple' | 'moderate' | 'complex';
}
```

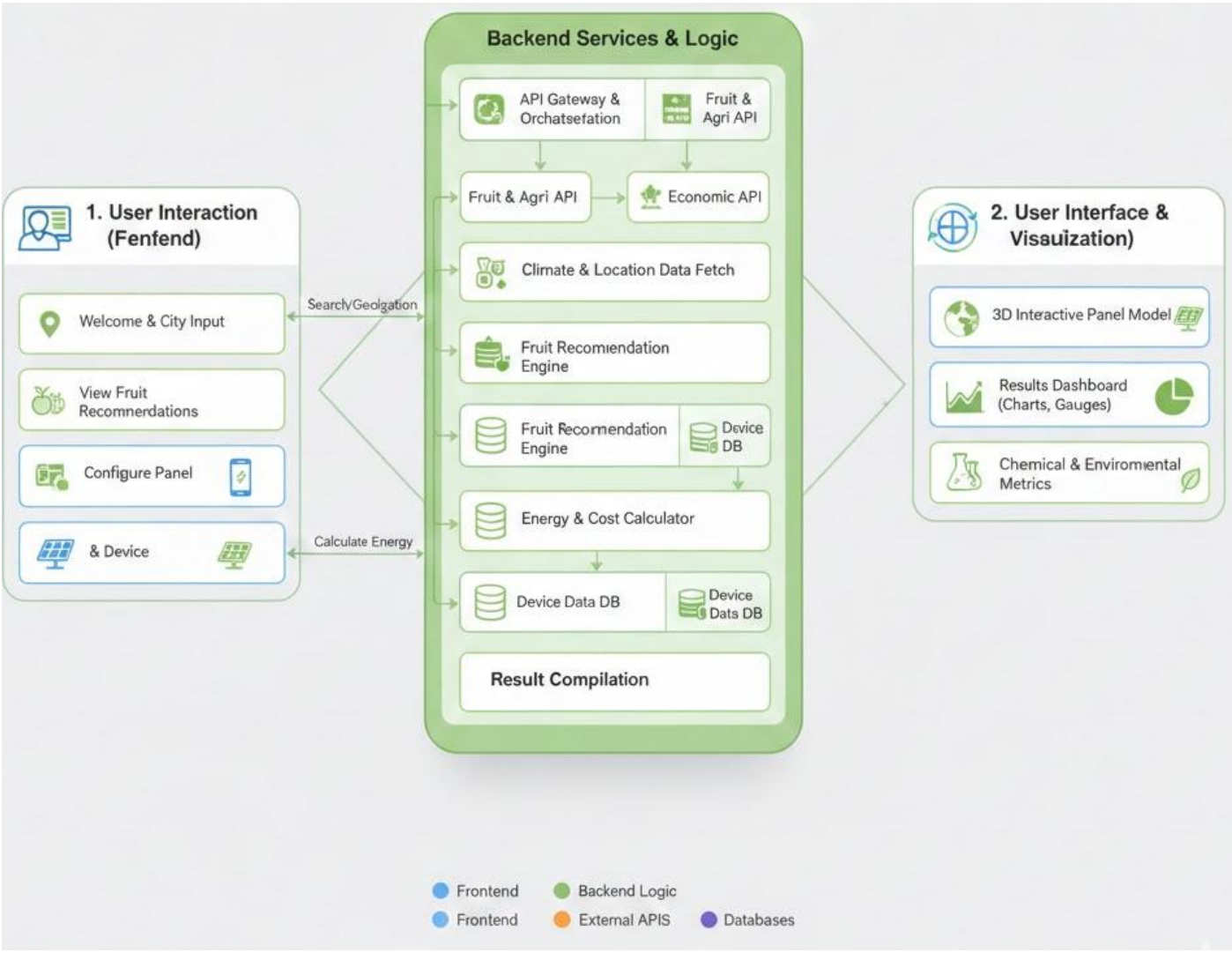
Similar TypeScript/JSON schemas apply to Region and Device entities.

6. API Endpoints Specification

Method	Path	Input Parameters	Output Structure	Possible Error Case
GET	/api/weather/:city	city (string)	{ temperature, humidity, UVIndex, cloudCover, forecast[], ... }	API unavailabl invalid city
GET	/api/fruits/recommendations/:location	location (string)	[{ fruitName, suitabilityScore, costPerKg, chemicalProperties, climateSpecialization }, ...]	No data, rate limit exceeded
POST	/api/calculate-energy	{ fruit, panelSize, deviceCategory }	{ energyForecast, costAnalysis, juiceRequired, roi, degradation }	Invalid inputs, computati error

(Expanded endpoints follow similar patterns for /api/uv-activated-panel, etc.)

7. UI / UX Flow & Mockups



User stories guide navigation: e.g., "As a user, I enter my city to get tailored fruit suggestions leading to energy estimates."

8. Risks, Limitations & Assumptions

- Data reliability: fruit chemical properties may vary by region, season, fruit variety—database might have approximations
- External API dependency: If APIs are unavailable, recommendations/calculations may fail.
- Real-world feasibility: Since fruit-juice electro-cells are experimental, actual energy output may differ significantly from estimates.

- Performance constraints: 3D visuals may not run smoothly in low-end devices or old browsers.
- Maintenance & scalability: Frequent API calls, global coverage—may hit rate limits or cost constraints.
- Regulatory / safety aspects (if used in real world panels): not covered

9. Project Roadmap & Milestones

- Phase 0: Requirement & Design Finalization (this document + stakeholder review)
- Phase 1: Setup project skeleton (frontend, backend), initial 3D scene, basic UI
- Phase 2: Implement core endpoints & fruit database + weather integration
- Phase 3: Build climate-adaptation engine + fruit recommendation logic
- Phase 4: Build calculator, panel config, device model, energy prediction logic
- Phase 5: UI dashboards, visualizations, chemical data view, environmental metrics
- Phase 6: Testing, caching, rate-limiting, optimization
- Phase 7: Deployment, documentation, user acceptance

10. Testing & Validation Strategy

- Unit tests for individual modules / calculations (e.g. juice-to-power formula, degradation)
- Integration tests for API calls & data fetching (simulate API failures, fallback logic)
- UI tests / E2E tests for user flows (city selection → recommendations → results)
- Performance testing (page load, 3D render performance, API response times)
- Data validation (fruit data, chemical values, edge cases, invalid inputs)

11. Deliverables & Success Criteria

Working frontend + backend, basic UI flows. At least 5 sample fruits in database with chemical & cost data. Weather integration for at least one city globally. Fruit recommendation logic working & returns plausible fruit list. Energy calculator giving sample output for small device + panel config. Documentation (this spec + API spec + schema + readme) complete. Basic tests passing.

12. Future Enhancements / Stretch Goals

- Add more fruits, more accurate chemical & region-specific data
- Add more APIs for better data coverage (agriculture, pricing, environment)
- Add localization / multi-language support
- Add mobile-friendly UI or mobile app version
- Integrate machine learning for better fruit selection / efficiency estimation
- Provide exportable reports (PDF / CSV) for users
- Prepare for real-world hardware integration / prototype design

This document aligns with standard practices as a Project Initiation Document (PID) and Software Requirements Specification (SRS), combining goals, scope, and technical design for clarity and collaboration.