**EcoVision Integration Guide**

PoC/Working Demo

|  |  |
| --- | --- |
| **Author:** | **Praveen Kumar Madhava Rao** |
| **Owner:** | **Praveen Kumar Madhava Rao** |
| **Contributors:** |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |

|  |  |
| --- | --- |
| **Version:** | **1.0** |
| **Date:** | **14/03/2024** |

**Table of Contents**

[**Preface** 3](#_Toc161340736)

[**Introduction:** 3](#_Toc161340737)

[**Frontend Design using Figma:** 3](#_Toc161340738)

[**Backend Design with Node.js:** 5](#_Toc161340739)

[**Visualization:** 5](#_Toc161340740)

[**Conclusion:** 6](#_Toc161340741)

# **Preface**

The purpose of this document, the EcoVision Integration Guide, is to streamline the integration process of frontend and backend components within the EcoVision project. It provides detailed instructions for achieving the following tasks:

* Frontend Design using Figma: Set up the Figma project, create wireframes, design visual elements, and define reusable components to ensure a cohesive and visually appealing frontend interface.
* Connect with Backend: Define API endpoints, simulate user interactions using Figma's prototyping feature, and collaborate with backend developers to align frontend design with backend specifications.
* Integration with CI/CD Pipeline: Export design assets from Figma, streamline collaboration with developers using Figma plugins or integrations, and incorporate design reviews into the CI/CD pipeline for iterative improvements.
* Backend Design with Node.js: Initialize and configure the Node.js project, define RESTful API endpoints, implement business logic, handle authentication and authorization, and manage data storage and integration with frontend components.
* Visualization: Implement real-time monitoring, carbon footprint prediction, and optimization recommendation visualization components, ensuring seamless integration with the frontend UI and utilizing JavaScript frameworks for enhanced interactivity.

**Introduction:**EcoVision is an innovative software solution designed to monitor, analyze, and optimize the environmental impact of software development processes. By integrating real-time monitoring, predictive analytics, and optimization recommendations, EcoVision empowers development teams to reduce carbon emissions, energy consumption, and resource usage, contributing to environmental sustainability and corporate responsibility goals.  
  
**Prerequisites:**

* Frontend Design with Figma
* Backend Development with Node.js
* Knowledge of data visualization libraries such as Chart.js, D3.js, or Plotly.js.
* Proficiency in JavaScript programming language for implementing visualization components in frontend applications.
* Understanding of CI/CD pipelines and deployment processes using tools like Jenkins, Travis CI, or GitHub Actions.

**Frontend Design using Figma:**

1. **Set Up Figma Project:**
   * Create a new project in Figma for EcoVision frontend design.
   * Organize frames, components, and stylesheets for better management.
2. **Wireframe Creation:**
   * Start with wireframes to outline the layout and structure of each page.
   * Design wireframes for key screens such as Dashboard, Settings, and Reports.
   * Include placeholders for data visualization components and interactive elements.
3. **Visual Design:**
   * Add visual elements such as colors, typography, icons, and images to enhance the design.
   * Create a style guide to maintain consistency across the application.
   * Use eco-friendly color schemes and nature-inspired imagery to reflect the environmental focus of EcoVision.
4. **Component Design:**

1. **Buttons:**
   * **Sample Button Name:** PrimaryButton
   * **Description:** PrimaryButton is a reusable UI component used to represent primary actions or call-to-action buttons within the application. It typically features a prominent color to draw attention and encourage user interaction. Examples include "Submit," "Save Changes," or "Add Item."
   * **Properties:**
     + Color: Specifies the background color of the button.
     + Size: Defines the size of the button (e.g., small, medium, large).
     + Disabled: Indicates whether the button is disabled or not.
   * **States:**
     + Hover: Changes appearance when the mouse hovers over the button.
     + Active: Changes appearance when the button is clicked or tapped.
   * **Interactions:**
     + onClick: Triggers a specific action or function when the button is clicked.
     + onFocus: Handles focus events when the button receives focus.
2. **Forms:**
   * **Sample Form Name:** LoginForm
   * **Description:** LoginForm is a reusable UI component used to collect user credentials for authentication. It consists of input fields for username and password, along with a submit button for login. This component helps streamline the login process and provides a consistent user experience across the application.
   * **Properties:**
     + Username: Stores the value entered in the username field.
     + Password: Stores the value entered in the password field.
   * **States:**
     + Invalid: Indicates validation errors if the form submission fails due to incorrect credentials.
     + Loading: Displays a loading indicator while the login request is being processed.
   * **Interactions:**
     + onSubmit: Submits the form data to the backend for authentication.
     + onChange: Updates the state of the form fields as the user types.
3. **Cards:**
   * **Sample Card Name:** ProductCard
   * **Description:** ProductCard is a reusable UI component used to display information about a specific product within the application. It typically includes an image, title, description, and action buttons (e.g., "Add to Cart," "View Details"). ProductCard helps showcase products in a visually appealing and organized manner, facilitating browsing and purchasing decisions for users.
   * **Properties:**
     + Image URL: Specifies the URL of the product image to be displayed.
     + Title: Displays the name or title of the product.
     + Description: Provides a brief overview or description of the product.
   * **States:**
     + Hover: Changes appearance when the mouse hovers over the card.
     + Selected: Indicates if the card is selected or highlighted.
   * **Interactions:**
     + onClick: Triggers a specific action (e.g., adding the product to the cart) when the card is clicked.
4. **Navigation Menu:**
   * **Sample Navigation Menu Name:** MainMenu
   * **Description:** MainMenu is a reusable UI component used to navigate between different sections or pages within the application. It typically includes links or buttons for common destinations such as Home, Dashboard, Settings, and Logout. MainMenu provides users with easy access to key functionalities and promotes efficient navigation throughout the application.
   * **Properties:**
     + Items: Specifies the list of menu items to be displayed.
   * **States:**
     + Active: Indicates the currently selected or active menu item.
   * **Interactions:**
     + onClick: Navigates to the corresponding page or section when a menu item is clicked.
5. **Connect with Backend:**
   * Define API endpoints and data requirements for each screen.
   * Use Figma's prototyping feature to simulate user interactions and data fetching from the backend.
   * Collaborate with backend developers to ensure alignment between frontend design and API specifications.
6. **Integration with CI/CD Pipeline:**
   * Export design assets from Figma and share them with developers for implementation.
   * Use Figma plugins or integrations (e.g., Zeplin, Avocode) to streamline handoff and collaboration with developers.
   * Incorporate design reviews and feedback loops into the CI/CD pipeline to iterate on design changes efficiently.

**Backend Design with Node.js:**

1. **Set Up Node.js Project:**
   * Initialize a new Node.js project for the EcoVision backend using npm or yarn.
   * Install necessary dependencies such as Express.js for building RESTful APIs, MongoDB or PostgreSQL for data storage, and other libraries for authentication, validation, and logging.
2. **Define API Endpoints:**
   * Design RESTful API endpoints to support frontend interactions, including data retrieval, analysis, and visualization.
   * Define routes for user authentication, data fetching, prediction requests, and optimization recommendations.
3. **Implement Business Logic:**
   * Write controller functions to handle requests from the frontend and perform data processing, analysis, and optimization.
   * Integrate third-party libraries or APIs for machine learning algorithms, environmental data retrieval, and carbon footprint prediction.
4. **Authentication and Authorization:**
   * Implement authentication middleware to verify user credentials and generate access tokens.
   * Set up authorization logic to restrict access to certain endpoints based on user roles and permissions.
5. **Data Storage and Management:**
   * Define data models and schemas using Mongoose for MongoDB or Sequelize for SQL databases.
   * Implement CRUD operations (Create, Read, Update, Delete) for managing environmental data, user settings, and prediction results.
6. **Integration with Frontend:**
   * Expose API endpoints for frontend consumption and ensure compatibility with data requirements specified in the frontend design.
   * Collaborate with frontend developers to test API integrations and resolve any compatibility issues.
7. **Continuous Integration and Deployment:**
   * Set up CI/CD pipelines using tools like Jenkins, Travis CI, or GitHub Actions to automate testing, building, and deployment processes.
   * Configure automated tests for API endpoints, data validation, and error handling to ensure reliability and stability.

**Visualization:**

1. **Real-time Monitoring:**
   * Use charting libraries like Chart.js or D3.js to visualize real-time environmental data such as carbon emissions, energy consumption, and resource usage.
   * Create dynamic charts and graphs to display trends, patterns, and anomalies in environmental metrics.
2. **Carbon Footprint Prediction:**
   * Develop visualization components to display predictions and forecasts generated by machine learning models.
   * Present prediction results in an intuitive and informative manner, including confidence intervals, uncertainty estimates, and actionable insights.
3. **Optimization Recommendations:**
   * Design visualization tools to present optimization recommendations generated based on data analysis and performance metrics.
   * Provide visualizations of performance improvements, efficiency gains, and environmental impact reductions resulting from optimization strategies.
4. **Integrating Visualization with Frontend:**
   * Integrate visualization components into the frontend UI, ensuring seamless interaction and data synchronization.
   * Use JavaScript frameworks like React or Angular to embed visualization elements within dashboard screens and reports.

# **Conclusion:**

The EcoVision Integration Guide facilitates seamless integration between frontend and backend components, ensuring a cohesive and visually appealing user experience. By following this guide, teams can streamline collaboration, implement robust backend infrastructure, and enhance the project's sustainability impact through dynamic data visualization.