**Project Scope**

The project involves developing a website and front-end for a SaaS platform that will integrate with a Python-based optimization model (already developed) as the backend, handling resource and scheduling optimization. The goal is to create a unified SaaS platform that includes both the marketing website and front-end software for the Python optimization model. This serves as a core concept for how the software should function, and it is expected that the geek will apply their expertise to enhance and improve the overall software.

**1. Project Overview**

* **Objective**: Create a unified SaaS platform that includes both a marketing website and a SaaS desktop application that runs your Python-based project scheduling optimization model.
* **Goal**: The marketing website should provide an overview of the product, pricing, and subscription options. The backend SaaS application should allow users to input data or upload excel file, run optimization tasks, view results, and manage their projects.
* **Tech Stack**:
  + **Backend**: Python with Flask/Django.
  + **Frontend**: HTML/CSS/JavaScript for the user interface.
  + **Deployment**: Cloud hosting on AWS/Heroku.
  + **Integration**: Python optimization code (already developed) will be integrated as the backend processing engine.

**2. Frontend Requirements**

**Landing Page (Marketing Website)**

* **Design**: Clean, responsive, user-friendly, compatible with mobile devices.
* **Sections**:
  1. **Hero Section**: A brief overview of the product (project scheduling optimization) with a clear CTA (e.g., “Sign Up”).
  2. **Features**: Highlight the key features of the SaaS platform, such as project management, resource optimization, scheduling, etc. This content will be given.
  3. **Pricing**: Detailed pricing structure with different plans (e.g., Basic & Customization). This content will be given.
  4. **Case Studies**: This content will be given.
  5. **Founders Profile**: This content will be given.
  6. **Contact Us**: Include a form to contact support or inquire about special pricing.
  7. **Signup/Login Links**: Clear paths for users to sign up/subscribe or log in.

**User Registration & Authentication**

* **Features**:
  + User registration form with email verification and password reset.
  + Login form with options to authenticate via email/password.
* **Security**:
  + Passwords must be hashed and stored securely.
  + SSL certificates for secure data transmission.
  + Other security protocols

**After Signing Up for the software**

**User Dashboard**

* **Main Features**:
  + **Project Management**: Users should be able to create, view, and manage their projects.
  + **Subscription Overview**: Display current subscription status (plan, renewal date).
  + **Optimization Task Summary**: View the status of previously run optimizations (pending, completed, etc.).
* **Design**: Simple, intuitive UI where users can quickly navigate between creating new projects, running optimization, and viewing results.

**Input Forms for Project Data**

* **Form Fields**:
  + Number of projects, tasks, workers, time periods, skills etc.
  + Input for project constraints such as skills, worker availability, project deadlines
  + Input for project requirements, skill availability per worker and cost per worker.
  + Control for adding other inputs as well
  + Support for both **manual data entry** and **Excel file upload**.
  + **File upload** should validate the format before accepting it (e.g., column validation, file type validation).

The input data fields will be provided in the excel file. The software must be available to allow add inputs field or upload excel file that will be connected to the optimization model

* **Data Validation**:
  + Validate input formats (e.g., numeric fields, date fields).
  + Provide error messages if inputs are incorrect or missing.

**Optimization Results Page**

* **Features**:
  + Display results of the optimization (e.g., task scheduling, worker assignments, cost breakdown).
  + Use **interactive charts/tables** to visualize the data.
  + Allow users to download results as Excel/PDF reports.

**User Settings Page**

* **Account Management**:
  + Change password, update profile information (email, name, etc.).
* **Subscription Management**:
  + Allow users to upgrade, downgrade, or cancel their subscription plans.
  + Integration with a payment gateway like Stripe/PayPal to handle payments.

**3. Backend Requirements**

**Integration of Python Optimization Model**

* **Backend Language**: Python-based backend (Flask or Django) will handle all API requests and process the optimization logic.
* **How the Model Works**:
  + Users provide input (manually or via Excel upload).
  + The Python model runs the optimization logic.
  + Results are returned and displayed on the results page.
* **Asynchronous Processing**:
  + Use asynchronous job processing (e.g., **Celery** with Redis) to handle long-running optimization tasks without freezing the user interface.

**Database Design**

* **Tables**:
  + **User Table**: Stores user account information, including hashed passwords, roles, and subscription status.
  + **Project Table**: Contains project details, including task and resource information.
  + **Optimization Results Table**: Stores results of each optimization run (project schedules, task assignments, cost data).
* **CRUD Operations**:
  + Implement CRUD functionality for managing user projects, input data, and results.
  + Allow users to delete old projects and results.

**4. Subscription Management**

* **Payment Integration**:
  + Integrate with payment gateways like **Stripe** or **PayPal** to handle subscriptions.
  + Allow users to enter their payment information securely.
  + Handle recurring payments, free trials, discounts, and upgrades/downgrades.
* **Subscription Check**:
  + On login or when trying to access premium features, check the user’s subscription status.
  + If the subscription has expired, prompt the user to renew or switch to a different plan.

**6. Performance and Scalability**

* **Caching**:
  + Use caching for static content (e.g., landing page, pricing information) to reduce server load.
* **Load Balancing**:
  + As traffic grows, ensure that the platform can scale horizontally by adding load balancers to distribute traffic between multiple servers.

**7. Deployment and Hosting**

* **Cloud Hosting**:
  + Deploy the entire platform to a cloud provider like **AWS**, **Heroku**, or **Google Cloud** Pay per use.
* **Monitoring and Error Logging**:
  + Integrate monitoring tools like **Datadog**, **New Relic**, or **Sentry** to track performance and detect errors in real time.

**8. Admin Panel**

* **Admin panel** for managing users, projects, and subscriptions.
* Track user activities, active subscriptions, and optimization performance.
* Chat box for customer support

**9. Email Integration**

* **Transactional Emails**:
  + Send confirmation emails for sign-ups and password resets.
  + Notify users when their optimization task has completed or when their subscription is about to renew.

**10. additional Features**

* **Free Trial**:
  + Set up a system to allow new users to try premium features for free for a limited period.

**All source code will be required to hand over to the client.**