CraftDoc
Your Docs, Your Way

SOFTWARE REQUIREMENTS SPECIFICATION (SRS)

CraftDoc
Your Docs, Your Way

1. Introduction

1.1 Purpose

CraftDoc is an online document customization platform that allows users to create, edit, and download professional documents such as letterheads, visiting cards, flyers, product catalogs, and company profiles. The platform will operate on a credit-based subscription model and will provide AI-powered design suggestions in future versions.

1.2 Scope

The system will:

- ***Provide pre-designed templates for customization***.

Provide pre-designed templates for customization, covering various document types such as letterheads, business cards, flyers, product catalogs, company profiles, and other professional documents. Templates will be designed to accommodate diverse industries and personal branding needs.

- ***Allow users to edit specific sections of templates using credits.***

Allow users to edit specific sections of templates using credits, covering essential fields such as name, address, title, phone number, email, website, social media links, logo, brand colors, slogan, mission statement, and other customizable elements. Users can selectively modify these fields based on their needs while maintaining a structured design.

- ***Offer download and printing options.***

Offer download and printing options, allowing users to save their customized designs in multiple formats such as PDF, PNG, and SVG. Additionally, users can integrate with third-party printing services or directly place printing orders through the platform for professional-quality output.

- ***Include a credit-based economy where users can earn or purchase credits.***

Include a credit-based economy where users receive an initial set of free credits, can watch ads to earn additional credits, or purchase credits via a secure payment gateway. Credits will be required for specific customization features, premium templates, and additional downloads.

- ***Implement AI-powered design suggestions in future iterations.***

Implement AI-powered design suggestions in future iterations, providing users with automated recommendations for color schemes, typography, layout adjustments, and branding consistency. These suggestions will be based on industry standards, user input, and machine learning analysis of successful design trends.

1.3 Target Audience

- Small and medium-sized businesses (SMBs)
- Professionals and freelancers
- Corporate users
- Students and academicians

1.4 Definitions, Acronyms, and Abbreviations

- SRS: Software Requirements Specification
- UI: User Interface
- UX: User Experience
- AI: Artificial Intelligence
- CMS: Content Management System
- CRUD: Create, Read, Update, Delete

2. Functional Requirements

2.1 User Management

1. User Registration and Login

- ***Step 1: Setup User Authentication***
  - Use authentication libraries like Firebase Authentication, Auth0, or implement your own authentication system if you have.
  - Allow users to register using their email and social media accounts (Google, Facebook, etc.).
    - Implement email verification after registration to ensure valid email addresses.
- ***Step 2: Email Registration Flow***
  - Sign Up: Users enter their email, password, and confirmation.
  - Send Verification Email: After user submits the form, send a verification email with a unique link to confirm the account.
  - Confirm Email: On clicking the verification link, confirm the email in the database and redirect the user to the login page.
- ***Step 3: Social Media Authentication Flow***
  - Integrate third-party APIs (Google/Facebook OAuth) to enable users to log in via their social media accounts.
  - Implement error handling for failed login attempts or denied permissions by the user.
- ***Step 4: User Login***
  - Create a login form where users can enter their email and password.
  - Session Management: Upon successful login, create a session token and store it in cookies or local Storage for persistent authentication.
  - Implement password reset functionality through email links to reset a forgotten password.

2. Profile Management

- ***Step 1: Create User Profile Model***
  - In the backend, define a User model that stores user data: name, email, password hash, profile picture, payment details (tokenization method for payment security), and credits.
  - Implement profile CRUD operations (Create, Read, Update, Delete) for managing profile information.
- ***Step 2: Edit Profile Section***
  - Create a profile page where users can update their details:
    - Name
    - Email (with email validation)
    - Password (change functionality with old-password verification)
    - Profile Picture (allow uploading and cropping images)
    - Payment Details (credit card or PayPal details stored securely, possibly via third-party payment APIs like Stripe)
    - Additional Info (phone number, bio, etc.)
- ***Step 3: Implement Security Measures***
  - Ensure password hashing (e.g., bcrypt) for secure storage.
  - Use two-factor authentication (2FA) as an additional layer of security (optional).
  - Rate limiting on login attempts to prevent brute-force attacks.
- ***Step 4: Payment Details Management***
  - Primary: Local Payment gateway like bkash or others local
  - Secondary: Integrate a payment gateway for international users like Stripe, PayPal, or Razorpay to manage payment details securely.
  - Implement tokenization of payment details for security. Only store token references, not sensitive card details.
  - Allow users to update or remove their saved payment methods securely.

3. Credit Tracking System

- **Step 1: Create a Credit Model**
  - In the database, add a credit balance field linked to the user account.
  - Implement actions like:
    - Credit Addition (for watching ads, purchasing, or other activities).
    - Credit Deduction (for using features like document creation or downloading).
- **Step 2: Display Credit Balance**
  - Show the user's current credit balance on the dashboard or profile page.
  - Allow users to earn credits through:
    - Watching ads.
    - Purchasing credits via a payment gateway.
    - Completing other actions that you choose (e.g., signing up for a newsletter).

- ***Step 3: Credit Deduction Mechanism***
  - Define actions that will deduct credits (e.g., generating designs, customizing templates).
  - Ensure real-time updates for credit balances after each transaction.
- ***Step 4: Implement Notification System***
  - Set up email and/or in-app notifications when:
    - Credits are added or deducted.
    - The credit balance falls below a threshold (e.g., a low-credit warning).
- ***Step 5: Credit Earning and Redemption***
  - **Ad-Watching Flow:** Set up an Ad-Watching system using services like Google AdMob to allow users to watch ads in exchange for credits.
  - **Referral System:** Implement a referral program where users can invite friends and earn credits for each successful referral.

2.2 Document Customization

1. Template Selection

- ***Step 1: Template Categorization and Display***
  - Organize templates into different categories (e.g., letterheads, resumes, visiting cards).
  - On the user dashboard, show a preview of the available categories.
  - Display a grid or list view of templates within the selected category.
  - Implement a search and filter system to help users find templates by keyword or style (e.g., modern, professional, creative).
- ***Step 2: Template Selection Mechanism***
  - Allow users to click on a template to select it.
  - When a user selects a template, open the template editor where they can modify editable fields (name, address, etc.).

2. Editable Fields Implementation

- ***Step 1: Define Editable Fields***
  - Identify and define the editable fields in the template (e.g., name, address, phone number, social media links).
  - For each template, map the editable sections that users can modify (e.g., using placeholders like {{name}}, {{address}}).
- ***Step 2: Develop a Custom Editor Interface***
  - Create an **editor interface** where users can click on each editable field to modify it.
  - **Text Fields:** Allow users to edit text, such as name, address, and slogan.
  - **Date Fields:** For fields like establishment date, implement a date picker or calendar widget.
  - **Image Fields:** Enable users to upload and change logos or images (e.g., drag-and-drop upload).
  - **Social Media Links:** Allow users to input URLs for social media handles.

- ***Step 3: Editing Features***
  - **Text Editing:** Add basic text editing features like font size, font family, color, alignment, and bold/italic/underline options for editable fields.
  - **Image Uploading:** Users can upload images for logos. Implement drag-and-drop functionality and automatic resizing/cropping to fit the designated area.
  - **Dynamic Fields:** For each editable field, track the field type (text, image, date, etc.) to handle user input appropriately.

3. Credit Deduction System for Specific Sections

- ***Step 1: Define Credit Deduction Rules***
  - Identify which editable sections cost credits to edit (e.g., changing name, address, or adding a logo might cost credits).
  - Implement a mechanism that checks whether the user has enough credits to edit a particular section.
- ***Step 2: User Feedback on Credit Costs***
  - Show the credit cost for each editable section before the user makes the change (e.g., a tooltip or a small icon next to the field indicating that editing will cost credits).
  - Upon editing, check if the user has sufficient credits. If not, show a prompt to either purchase more credits or watch ads to earn credits.
- ***Step 3: Real-Time Credit Updates***
  - Deduct credits when a user modifies an editable field and confirm the deduction in real-time.
  - Implement an instant credit check during the editing process to prevent any accidental credit usage.

4. Live Preview of Edited Templates

- ***Step 1: Real-Time Rendering***
  - Implement live preview functionality that immediately reflects any changes made to editable fields in the template (e.g., text, images, logo).
  - Use JavaScript or a front-end framework (like React or Vue.js) to dynamically update the preview area as the user makes changes.
- ***Step 2: Synchronization of Changes***
  - Ensure that every edit (text, image, or date) made by the user is synchronized with the live preview.
  - For images (like logos), display the uploaded image in the preview in real-time.
- ***Step 3: Responsive Layout***
  - Ensure that the live preview is responsive and looks correct on different screen sizes (mobile, tablet, desktop).
  - Allow users to toggle between the desktop and mobile versions of the template during the preview process.

5. Save and Resume Editing Later

- ***Step 1: User Session Management***
  - Implement session-based saving of the user's editing progress.
  - When a user starts editing a template, automatically save their progress in the database or local storage, so they can resume later.
- ***Step 2: Save Button and Notification***
  - Provide a Save button in the editor. When clicked, the current changes are saved in the backend (e.g., in the user's account or database).
  - If the user has not saved their changes, show a prompt asking them to save before they navigate away or close the page.
- ***Step 3: Resume Editing Later***
  - When the user returns to their dashboard or editing page, display an option to resume editing from where they left off.
  - On resuming, load the saved template with all previous edits pre-filled.
  - Allow the user to continue editing the document or start fresh with a new template.

6. Saving Finalized Document

- **Step 1: Finalize Document**
  - Once the user is happy with the customizations, provide an option to finalize the document. This would save the customized version in a format (e.g., PDF, PNG).
- **Step 2: Export Options**
  - Provide download/export options to save the document locally or share it via email or social media.
  - Add a credit deduction for the download action, which will be applied after the final document is generated.
- **Step 3: Version Control**
  - Allow users to save multiple versions of their customized documents. This can be useful if they want to test out different variations before finalizing.

7. Testing & Debugging

- ***Step 1: Unit Tests***
  - Write unit tests for the editor functionality, including:
    - Text and image editing.
    - Live preview updates.
    - Credit deduction checks when editing fields.
- ***Step 2: Integration Tests***
  - Test the end-to-end flow from template selection to saving and exporting the final document.

2.3 Credit System

1. Credit Allocation on Signup

- ***Step 1: Create User Model with Credit Balance***
  - In the user model, include a field for credits (e.g., credit_balance).
  - Set the initial value of credits to 10 when a user signs up.
  - This can be done by adding a function that runs after the user completes their registration, which sets the initial credits.
- ***Step 2: Store Credit History***
  - Maintain a credit history log for each user (e.g., credit_transactions table) to track:
    - How credits are added (e.g., through sign-up, purchase, ad-watching).
    - When credits were added.
    - When credits are deducted (e.g., when the user uses credits for document customization).
- ***Step 3: Provide Feedback to Users***
  - Upon successful registration, display a notification or message informing the user that they've received 10 free credits.
  - Update the credit balance visible on the user dashboard.

2. Credit Purchases via Payment Gateways

- **Step 1: Integrate Payment Gateways (Stripe, PayPal, bkash)**
  - Integrate Stripe, PayPal, and bkash (for local payments) to handle credit purchases.
  - For Stripe and PayPal, you can use their APIs to process payments.
  - For bkash (or any local gateway), use the specific API for that service, making sure it supports secure transactions.
- ***Step 2: Payment Flow***
  - When users choose to purchase credits, show them a page to select how many credits they want to buy (e.g., 10, 50, 100 credits).
  - Upon selecting the desired amount, redirect the user to the payment gateway for processing the payment.
  - After the transaction is successful, add credits to the user's account (based on the purchased amount).
  - Implement transaction status handling (e.g., success, failure, pending) and ensure the credit balance is updated accordingly.
- ***Step 3: Credit Purchase Confirmation***
  - Send a confirmation email or in-app notification to the user once the transaction is successful, confirming the amount of credits purchased.
  - Provide a summary of the updated credit balance.
- ***Step 4: Implement Refund and Dispute Mechanism***
  - Implement functionality to handle refund requests in case of failed transactions, where credits are not credited despite successful payments.
  - Integrate with the payment gateway's dispute and refund mechanisms.

3. Credit Earning via Watching Ads

- ***Step 1: Integrate Ad-Watching Service***
  - Integrate an ad-watching service like Google AdMob, StartApp, or Vungle.
  - These services provide APIs to display ads and reward users with credits after watching them.
- ***Step 2: Set Credit Earning Rules for Ads***
  - Determine how many credits the user earns per ad view (e.g., 1 credit per 30-second ad).
  - Create a system that will:
    - Show users a button to watch an ad and earn credits.
    - Track whether the user watches the ad completely before awarding credits.
- ***Step 3: Implement Ad-Watching Flow***
  - When a user clicks the "Earn Credits by Watching Ads" button, show a loading screen or message indicating that an ad is being prepared.
  - After the ad finishes, the system should automatically credit the user's account with the earned credits and provide feedback (e.g., a message or pop-up saying, “You’ve earned 1 credit!”).
  - Deduct credits immediately when the user uses them for a service, and ensure a real-time update of the credit balance.
- ***Step 4: Limit Ad Views***
  - Set limits on how often users can watch ads per day or per session to prevent abuse (e.g., 5 ads per day).
  - Display a message when the ad limit for the day has been reached.
- ***Step 5: User Feedback on Ad Earnings***
  - After watching an ad, provide the user with a summary of how many credits they earned and how many they have left.
  - Update the credit balance in real-time and reflect it in the user interface (dashboard/profile).

4. Credit Deduction and Usage

- ***Step 1: Deduct Credits for Document Customization***
  - When users customize documents (e.g., changing name, logo, address), deduct credits according to the pricing rules set for each editable section.
  - Ensure credits are deducted at the point of modification (e.g., before the user can preview or download the finalized document).
- ***Step 2: Handle Insufficient Credit Balances***
  - If a user doesn’t have enough credits to make a change, show a warning message indicating that they need more credits and provide options to:
    - Purchase credits via a payment gateway.
    - Earn credits by watching ads.

- ***Step 3: Real-Time Credit Updates***
  - Ensure the user's credit balance is updated in real-time as credits are used or earned, providing a seamless experience.

5. Credit Balance Display and Notification

- ***Step 1: Show Credit Balance on User Dashboard***
  - Display the current credit balance in a prominent location on the user dashboard and other relevant pages (e.g., the editing page).
  - Display credit usage history in the user's account or profile section, including how credits were earned (ad-watch, purchase) and how they were spent.
- ***Step 2: Notify User of Credit Changes***
  - Send notifications or emails to users when:
    - Credits are earned (e.g., after watching an ad or making a purchase).
    - Credits are used for document customization.
    - The user's credit balance falls below a certain threshold (e.g., low-credit warning).
- ***Step 3: Optional: Set Up Credit Expiration***
  - You may choose to set an expiration date for credits (e.g., credits that expire after 1 year), and notify users accordingly.

2.4 Document Export & Printing

- Download options: PDF, PNG, JPEG.
- Print-ready formats.
- Integration with third-party printing services (future feature).

2.5 Payment & Subscription

- **Payment Gateway Integration:**
  - Integrate secure payment gateways (e.g., Stripe, PayPal, bkash) for handling payments.
  - Ensure PCI-DSS compliance and use tokenization to securely handle payment information.
  - Users can purchase credits or subscribe via local and international payment methods.
  - Provide payment confirmation and receipts after successful transactions.
  - Handle refunds and cancellations via the payment gateway.
  - Implement payment flow: users choose a method, enter details, and receive confirmation.

- **Subscription Plans (Future Feature):**
  - Offer business subscription plans (e.g., Basic, Standard, Premium) with benefits like monthly credits, premium templates, and support.
  - Integrate subscription management tools to handle recurring billing, renewals, cancellations, and plan upgrades/downgrades.
  - Include free trials with automatic transition to paid plans.
  - Provide invoices for each billing cycle, and allow users to manage their subscription status through a dashboard.
  - Implement notifications for subscription renewal, payment failures, and plan changes.

2.6 Admin Panel

1. User Management:

- **CRUD Operations:** Admins can create, read, update, and delete user accounts.
- **User Activity Monitoring:** Track user actions, login history, and document usage.
- **Audit Logging:** Record changes made to user accounts for accountability.

2. Template Management:

- **CRUD Operations:** Admins can upload, edit, or remove templates (e.g., letterheads, business cards).
- **Template Categorization:** Organize templates into categories and allow for multiple categories per template.
- **Template Analytics:** Monitor template usage and display usage statistics.

3. Credit Transaction Monitoring:

- **Transaction Logs:** Track credit purchases, refunds, and adjustments.
- **Manual Adjustments:** Admins can modify user credit balances as needed.
- **Transaction Alerts:** Get notifications for unusual credit activity.
- **Financial Reporting:** Generate reports on credit transactions, revenue, and refunds.

4. Advertisement Management:

- **Ad Creation and Scheduling:** Upload and schedule ads, set targeting options.
- **Ad Performance Tracking:** Monitor impressions, click-through rates, and conversions.
- **Budget Management:** Set and track ad campaign budgets.
- **Approval Workflow:** Admins review and approve ads before they go live.
- **Ad Revenue Reporting:** Generate reports on ad revenue and spending.

5. Admin Panel Dashboard:

- **User-Friendly Interface:** Provide quick access to key sections with summary statistics.
- **Data Visualization:** Display data through charts and graphs for easier analysis.
- **Responsive Design:** Ensure the panel is mobile-friendly for on-the-go management.

2.7 AI-Powered Design Suggestions (Future Development)

- AI-generated design recommendations.
- Auto-suggest color schemes and fonts.

2.8 Revenue Model

CraftDoc's revenue model consists of the following streams:

- Revenue from Google AdSense (Ads Revenue) – Display ads on the platform to generate passive income.
- Revenue from Google Ad Manager – Utilizing Google's base ads service for rewarded video ads and display ads to earn revenue.
- Revenue from Credit Sales – Selling credits to users for customization options and document downloads.

2.9. Development Plan for Revenue Integration

- **Week 1:**
  - Define revenue model implementation strategy.
  - Research and integrate Google AdSense and Google Ad Manager APIs.
  - Set up initial backend infrastructure for ad placements.
- **Week 2:**
  - Develop frontend components to display ads dynamically.
  - Implement rewarded ad logic to allow users to earn credits.
  - Integrate credit purchase system with secure payment gateways.
- **Week 3:**
  - Conduct testing for ad revenue and credit purchase functionality.
  - Optimize ad placements to maximize revenue without disrupting user experience.
  - Deploy revenue-related features and monitor performance.

3. Non-Functional Requirements

3.1 Performance Requirements

- The system should handle up to 10,000 concurrent users.
- Page load time should be under 3 seconds.

3.2 Security Requirements

- SSL encryption for secure transactions.
- Role-based access control (RBAC) for user and admin functionalities.
- Secure API communication with token-based authentication.

3.3 Usability Requirements

- User-friendly and intuitive UI/UX.
- Mobile-responsive design.

3.4 Availability Requirements

- System uptime should be at least 99.5%.
- Automatic backups every 24 hours.

3.5 Maintainability & Scalability

- Modular codebase for easy future enhancements.
- Microservices architecture (if applicable).
- Cloud hosting support (AWS/GCP/Azure).

4. Development Timeline

Week 1: Core Functionality Development

Day 1-3: User Management (CRUD Operations)

- Set up authentication and authorization system.
- Implement user registration and login via email and social media.
- Implement CRUD functionality for user accounts (Create, Read, Update, Delete).
- Develop user activity tracking and audit logs.

Day 4-5: Template Management (CRUD Operations)

- Design and implement a template upload system (for different categories).
- Develop template CRUD functionality (upload, edit, delete).
- Create an interface for admins to manage template metadata (category, name, description).

Day 6-7: Credit Transaction Monitoring

- Design and implement a credit transaction system.
- Implement credit transaction logs and transaction types.
- Set up manual credit adjustments and transaction alerts.

Week 2: Advanced Features & Admin Panel Management

Day 8-9: Advertisement Management

- Develop ad creation and scheduling interface.
- Implement ad performance tracking (impressions, CTR, etc.).
- Set up budget management and ad revenue reporting.

Day 10-11: Payment & Subscription Management

- Integrate payment gateway (e.g., Stripe, PayPal).
- Set up secure payment handling and subscription plans (if needed).
- Implement subscription management and payment logs.

Day 12-13: Admin Dashboard & UI

- Design a user-friendly admin dashboard with key metrics and quick access to functionalities.
- Implement data visualization (charts and graphs for analytics).
- Ensure responsive design for mobile compatibility..

Day 14: Template Categorization & Management Interface

- Finalize template categorization and allow multiple categories per template.
- Implement a system for template usage analytics.

Week 3: Finalizing Core Functionalities

Day 15-17: User Experience Improvements

- Refine UI/UX for user management, template management, and credit transaction systems.
- Polish the admin panel interface for smoother navigation and access to key features.

Day 18-19: Revenue and Ad Campaign Analytics

- Implement ad revenue tracking and detailed reporting.
- Finalize financial reporting features for credit transactions and ad revenue.

Day 20-21: Final Integration and Polishing

- Integrate all features (User Management, Template Management, Credit Monitoring, Ads, Payment & Subscription).
- Perform final adjustments on the backend for smooth data flow between systems.
- Finalize user roles, permissions, and security measures.

Post-Development: Testing & Feedback (After Week 3)

Once the 3-week development phase is completed, testing and gathering feedback will begin, which will be a separate process.

This breakdown ensures that each key aspect of the product is developed systematically, with a focus on core functionality first and then polishing and integrating all parts together by the end of the third week.

5. Conclusion

The platform development includes features like user management, document customization, credit tracking, template management, advertisement management, and payment integration, creating a comprehensive solution for customized documents with a credit-based subscription model.

- **A structured three-week timeline focused on key functionalities:**
  - User authentication
  - Template management
  - Transaction monitoring
  - Payment integration

The Admin Panel ensures effective management of users, transactions, templates, and advertisements. Users can efficiently create personalized documents while utilizing a robust credit system for monetization.

With integrated payment gateways and subscription options, the platform supports flexible revenue models. As it enters the testing phase, refinements will enhance performance, usability, and security.

The overall goal is to offer an intuitive, user-friendly solution for document customization and credit subscriptions, boosting productivity for individuals and businesses while ensuring sustainable growth.