Page Builder

High Level Design

Version 1.0

Author: Tebogo Tseka

AutAu

Table of Contents

1. Document History

Version	Changes	Owner
1.0	Initial Version	Tebogo Tseka

2. Introduction

This document is intended to serve as a reference point for the technical design of the **AI-Powered Landing Page Builder**. This solution will be based on **AWS serverless architecture; Amazon Bedrock (Claude Sonnet 4.5), CloudFront CDN, and DynamoDB** to enable **rapid, high-quality landing page generation for marketing teams, designers, developers** and depends on **AWS Bedrock Foundation Models and existing design asset libraries**.

This document has been broken into the following sections:

- **Contextual Design** - Provides context on how the components in this solution depend on each other
- **HLD** - High Level Design is intended to provide an end-to-end view of the solution

3. Business Requirements Overview

Currently, the challenges faced by **Marketing and Product teams** are:

Main pain points:

1. **Time to Market**: Creating landing pages for campaigns currently takes 2-4 weeks involving designers, developers, and QA teams. Marketing teams need the ability to launch campaigns within 24-48 hours.
2. **Resource Constraints**: Design and development resources are bottlenecked, with teams handling 20+ landing page requests per month but only able to deliver 5-7.
3. **Cost per Page**: Each custom landing page costs approximately R8,000-R15,000 in design and development resources.
4. **Inconsistent Quality**: Without a centralized system, landing pages vary in quality, brand consistency, and performance optimization.
5. **Limited A/B Testing**: Manual page creation makes it prohibitively expensive to create multiple variations for testing.
6. **Technical Dependency**: Marketing teams cannot iterate independently and must wait for developer availability for even minor changes.

Scope of requirements:

The solution will cover:

- AI-powered landing page generation from natural language prompts
- Component-based assembly using pre-approved design templates
- Real-time preview and iterative refinement capabilities
- Quality validation and brand consistency checking
- Multi-environment deployment (DEV, UAT, PROD)
- Performance monitoring and cost optimization
- Integration with existing design asset library
- Migration of legacy sites from Xneelo to AWS

4. Business Ask

The ask from **Marketing and Product teams** is to address the above-mentioned challenges with a **GenAI-powered landing page builder**. It should **enable non-technical users to generate production-ready landing pages through conversational AI** to improve **time-to-market, reduce costs, and maintain brand consistency**.

When complete, the solution should allow a user to:

1. Describe their landing page requirements in natural language (e.g., "Create a modern SaaS landing page with pricing table, testimonials, and contact form")
2. Receive an AI-generated preview within 10-15 seconds
3. Iteratively refine the page through conversational feedback
4. Validate the page meets quality, security, and performance standards
5. Deploy directly to production or staging environments
6. Track page performance and engagement metrics
7. Reuse and customize existing high-performing templates

Success Metrics:

5. Reduce landing page creation time from 2-4 weeks to 24-48 hours
6. Reduce cost per page from R8,000-R15,000 to under R2500
7. Enable 80% of landing page requests to be self-service
8. Maintain 95%+ brand consistency score
9. Achieve 90%+ user satisfaction rating

10. High Level Epic Overview

User story Number	Epic	User Story	Test Scenario(s)
1	1	As a **Marketing Manager**, I want to **describe my landing page**	When I submit "Create a product launch page for enterprise

		requirements in plain language so that I can **quickly generate a draft page without technical knowledge**	software with hero section, feature highlights, and pricing", then a preview is generated within 15 seconds, and when I review it, then all requested sections are present and properly formatted
3	1	As a **Content Strategist**, I want the AI to **use our existing design components and brand assets** so that I can **maintain consistency across all pages**	When I request a landing page, then the AI assembles it using only approved templates from our design library, and when I inspect the output, then it matches our brand guidelines
4	2	As a **Marketing Manager**, I want to **provide feedback and request changes conversationally** so that I can **refine the page without starting over**	When I say "Make the hero section more minimal and add a testimonial carousel", then the AI updates only those sections while preserving other content, and when I review, then changes are accurately applied
5	2	As a **Designer**, I want to **see generation history and rollback to previous versions** so that I can **experiment safely**	When I make multiple iterations, then each version is saved with timestamp, and when I want to revert, then I can restore any previous version
6	3	As a **Brand Manager**, I want **automatic validation of brand compliance** so that I can **ensure all pages meet our standards**	When a page is generated, then it's automatically scored for brand consistency (8/10 minimum), and when validation fails, then I receive specific feedback on issues
7	3	As a **Security Engineer**, I want **all generated code to be scanned for vulnerabilities** so that I can **prevent XSS and injection attacks**	When HTML is generated, then it passes through security validation, and when malicious patterns are detected, then generation is rejected with specific warnings
8	4	As a **Marketing Manager**, I want to **deploy pages to staging or production with one click** so that I can **quickly launch campaigns**	When I approve a page, then I can deploy to staging environment, and when ready, then I can promote to production with tracked versions
9	4	As a **DevOps Engineer**, I want **automated performance testing before production deployment** so that I can **ensure pages meet performance standards**	When deploying to production, then page load time is tested, and when performance is below threshold, then deployment is blocked with optimization suggestions
10	5	As a **Growth Marketer**, I want to **track which components perform**	When pages are live, then engagement metrics are

		best so that I can **optimize future page generations**	collected per component, and when I review analytics, then I see which templates drive highest conversion
11	5	As a **Product Owner**, I want to **see cost and performance metrics for AI generation** so that I can **optimize resource usage**	When reviewing dashboards, then I see token usage, generation costs, and latency per request, and when costs spike, then I receive alerts

11. Contextual Design

This section is intended to provide context of the solution design and how the systems will leverage each other's capabilities.

Architecture Overview

Architecture Design Principles

The AI Landing Page Builder follows a serverless, event-driven architecture with the following key design principles:

1. **Component Assembly over Full Generation**: Leverages existing design templates stored in S3 Designs and managed through the Prompt Library for consistency, brand compliance, and accelerated generation speed
2. **AI-Powered Generation**: Uses Amazon Bedrock's Claude Sonnet 4.5 model for intelligent landing page creation, with the SiteGenerator Lambda orchestrating the generation process
3. **Validation Pipeline**: Automated quality, security, and performance checks executed by the WebsiteValidator Lambda before deployment, ensuring all pages meet compliance thresholds and security standards
4. **Metadata-Driven Architecture**: DynamoDB tables (Customers, Prompts, Sites, Migrations) store component relationships, generation context, user preferences, and audit trails, enabling version control and rollback capabilities
5. **Asynchronous Processing**: SQS queue decouples site generation from deployment, allowing the SiteDeployer Lambda to process validated pages independently and scale based on demand
6. **Multi-Tenant Separation**: Cognito-secured API Gateway endpoints provide role-based access control for three distinct user personas (Generator UI users, Website Admins, and Migrator users), with separate data isolation per customer
7. **Edge-First Delivery**: CloudFront CDN at edge locations ensures low-latency access for both the React application and generated landing pages, with WAF protection for security

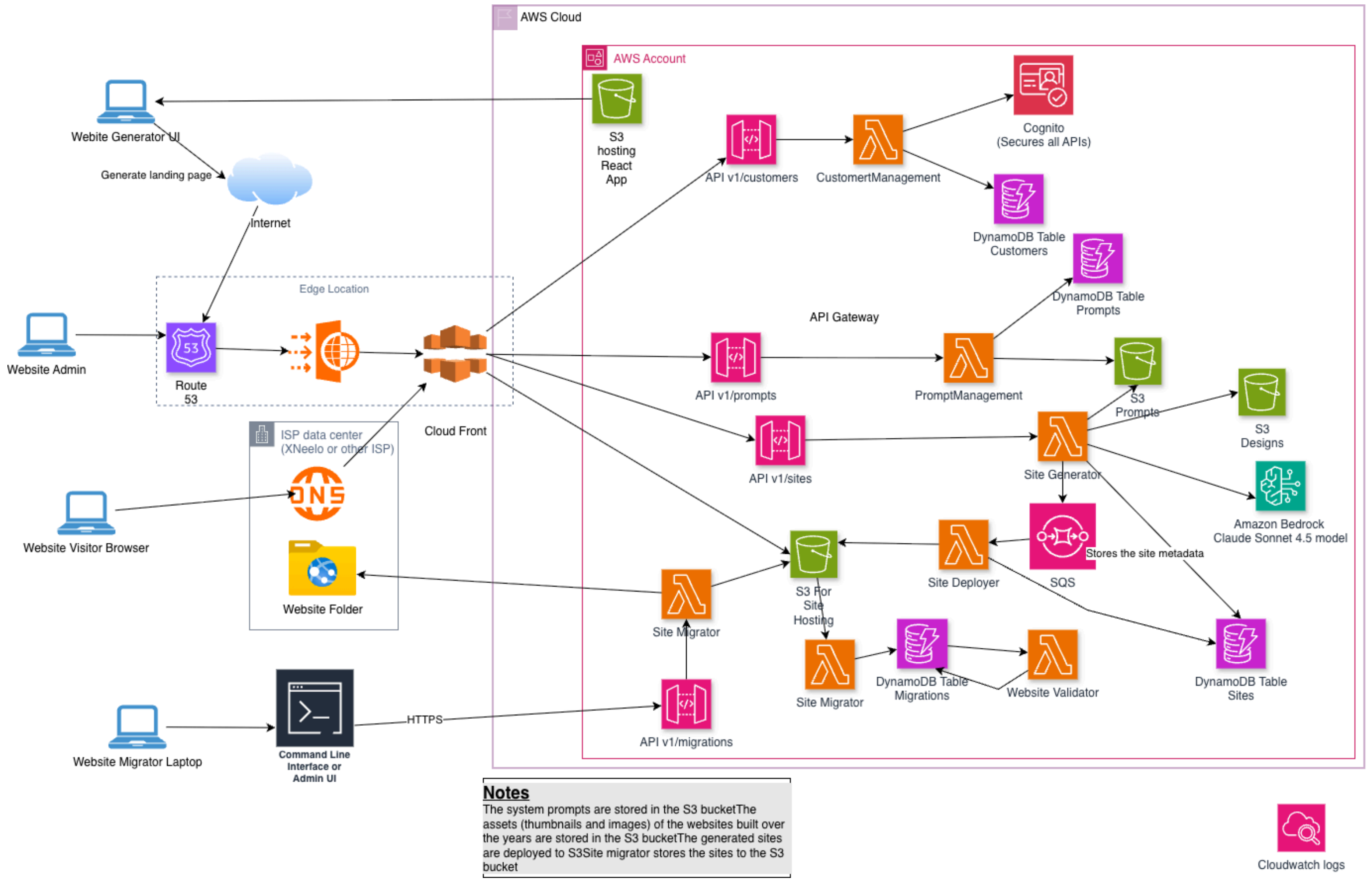
8. **Progressive Enhancement**: Flexible architecture supporting various service levels and use cases through configurable Lambda functions and DynamoDB-driven metadata

Context Diagram

The solution consists of the following major components:

1. **Site migrator** - A component responsible for migrating Websites from Wordpress to static HTML. This component uses lambda, S3 for hosting sites and DynamoDB table for site metadata.
2. **Page/Site builder** - An AI powered landing page builder that generates landing pages using input from the user and Amazon Bedrock - Claude Sonnet 4.5. The generated sites are deployed on to an S3 bucket and site metadata is kept in a DynamoDB table.
3. **Site deployer -** A component responsible for deploying migrated sites and newly built sites. The deployer ensures sites are available on well known URLs and workflowed through the DEV, SIT and PROD environments.
4. **Site designer -** An AI based, nuanced tasteful website site crafter that focuses on aesthetics, good taste and usability. This component infuses BBWS's years of tasteful web design into website design, whilst ensuring customers have choice and flexibility.

All components in this solution are serverless and scale on usage. By adopting a frugal by default serverless first approach the architecture ensures performance while not increasing cost.



Notes
The system prompts are stored in the S3 bucketThe assets (thumbnails and images) of the websites built over the years are stored in the S3 bucketThe generated sites are deployed to S3Site migrator stores the sites to the S3 bucket

11.1. Component List

Protocol	Component name	Existing / New/ Change	Description	Epic/ Phase	User Story #
Web	Frontend Application (S3/CloudFront)	New	React-based web interface for users to submit landing page requirements in plain language, select AI model preferences (preview/production), provide iterative feedback, review generation history, and manage deployment workflows	1, 2, 3, 4, 5	1,3,4,5,8,10,11
REST	API Gateway	New	Primary API entry point that routes all backend requests, implements authentication/authorization, enforces rate limiting, and provides unified interface for frontend to interact with Lambda orchestration layer	1, 2, 3, 4	1,3,4,5,6,7,8,9
Direct Integration	Claude Sonnet 4.5 (Production Model)	New	High-quality AI model for production-grade landing page generation with enhanced content quality, proper semantic structure, and optimized component assembly using brand-compliant templates	1, 2	3,4
Direct Integration	Claude Haiku (Preview Model)	New	Fast AI model for rapid preview generation enabling quick iteration cycles, allowing users to refine requirements before committing to production-quality generation	1, 2	4
REST	S3 Bucket - Design Assets	New	Cloud storage containing approved design templates, brand components, and reusable page patterns that AI models reference during generation to ensure brand consistency	1	3

REST	S3 Bucket - Generated Pages	New	Storage repository for finalized landing pages serving both staging and production versions with versioning enabled for rollback capabilities	2, 4	5,8
REST	DynamoDB - Template Metadata	New	Database storing design component catalog, template definitions, brand guidelines, and prompt library that the orchestrator queries to guide AI generation	1, 5	3,10
REST	DynamoDB - Generation History	New	Audit database tracking all generation requests, version history, customer feedback iterations, approval workflows, and rollback points for governance and analytics	2, 3, 5	4,5,6,10,11
REST	DynamoDB - User State	New	Session management database maintaining user preferences, active generation sessions, workflow status (review/approve/production), and personalization settings	2, 4	4,5,8
REST	DynamoDB - Prompt Library	New	Repository of successful prompts, proven generation patterns, and optimized component combinations learned from high-performing pages to improve future generations	5	3,10
REST	Orchestrator Lambda	New	Core business logic function that analyzes user requests, selects appropriate AI model, coordinates component assembly, manages generation workflow, and handles error scenarios	1, 2, 3, 4	1,3,4,6,7
REST	Analytics Lambda	New	Data processing function that collects performance metrics, tracks token usage and costs, analyzes component effectiveness, aggregates conversion data, and triggers cost alerts	5	10,11

REST	CloudWatch Logs	New	Centralized logging and monitoring service capturing errors, operational metrics, generation latency, security events, and performance data for troubleshooting and optimization	3, 5	7,9,11
REST	WAF (Web Application Firewall)	New	Security layer providing DDoS protection, rate limiting per user, IP-based access control, and protection against common web exploits (XSS, SQL injection)	3	7
REST	CloudFront Distribution (Dev)	New	CDN for development environment providing edge caching, SSL termination, low-latency content delivery, and integration with WAF for security	1, 4	1,8
REST	CloudFront Distribution (Prod)	New	Production CDN serving approved landing pages to end customers with optimized caching strategies, performance monitoring, and failover capabilities	4, 5	8,9,10
REST	Route 53	New	DNS service managing routing between development (dev.[customer-domain].co.za) and production (customer-domain.co.za) domains with health checks and failover	4	8
Direct Integration	Validation Service (embedded in Lambda)	New	Quality assurance module performing brand compliance checking (minimum 8/10 score), security scanning for XSS/injection vulnerabilities, and content validation before approval	3	6,7
Direct Integration	Performance Testing Service (embedded in Lambda)	New	Automated testing module that measures page load times, validates against performance thresholds, and blocks production deployment with optimization recommendations when standards aren't met	4	9

12. Messaging and notifications

As part of the operation of this solution there will be a need to send out notification to the operations team in case of SLA breach and or system failures. It is expected that a mailing list will be created and monitored by the ops team as indicated below.

Type	Target group	Description
Email	Operations team	On slow responses below NFR SLA
Email	Operations team	On unexpected failures
Email	FinOps team	Budget alerts on monthly threshold

Environment	Email Address Prefix	Description
DEV	DEV.support@bigbeard.co.za	
SIT	SIT.support@bigbeard.co.za	
Prod	support@bigbeard.co.za	There is no prefix for prod therefore it is blank

The above email addresses must be created across the following environments.

13. NFRs (TBC2)

Stats	Value
Time To First Token (TTFT)	< 10 ms
Time To Last Token (TTLT)	< 2 seconds (To be tested)
GenAI Conversation duration	TBC
Max Backend API Response times	1 second (For non-streaming)
Expected Monthly Customers	TBC
Max Tokens per conversation	< 500 K
Monthly cost estimate	TBC
Response times per API	< 10 ms

14. Risks and mitigations

Risk	Mitigation
LLM Model Jailbreak	Red team testing and penetration testing
Fraud risk mitigation	All purchases will be secured with an OTP
Ransomware risk mitigation	PII Masking, DR and Backup strategy will be used ensure PII customer data is not readable and all customer data is backed up and vaulted in-line with the RPO and RTO
DDoS risk mitigation	All public facing endpoints will be fronted with WAF, CloudFront and Shield
Infiltration risk mitigation	Security token will be passed from the webapp to SoulMachines and be echoed on all calls to the AWS account and validated.
Performance risk mitigation	Lambda will autoscale and LLM performance will be monitored.

15. Tagging

15.1. Company Specific Tags

Tag name	Value	Cost allocated	Description
Project	Switbuilder.ai	Y	
CreatedBy	<User Name>		
CostCenter	Swiftbuilder.ai	Y	

16. Governance

16.1. Cost Estimation

The following link provides cost estimate for the non-prod environment ***<Link to calculator>***

The following link provides cost estimate for the prod environment ***<Link to calculator>***

16.2. Saving Guidelines

- Non-production resources should be automatically shut-down daily as a cost saving mechanism.

- Adopting serverless first is intended to keep cost low as prescribed by https://thefrugalarchitect.com/

16.3. Monitoring Dashboards

Metric	Description
Lambda response time	The lambda will log the method duration with an identifiable business name and transaction id. The average value of this metric will drive this dashboard.
Error Rate	This dashboard will allow alarming in case of a high error rate

16.4. Troubleshooting Playbook

16.4.1. Slow/Lost transaction tracing

Each transaction will have an associated transaction ID which will be logged for all subsequent interactions. The unique transaction id will be a plain non-sortable GUID. The transaction ID will be propagated to the front-end to facilitate trouble shooting and tracing.

16.4.2. Business Error Handling

Business errors will not be logged unless the solution is running in DEBUG mode.

16.4.3. Technical Error Handling

All technical errors will be logged and reported on the error rate dashboard.

16.4.4. Disaster Recovery

This section details the BCP strategy adopted by business and the accompanying technical DR strategy.

	Decision/Value	Reason
Solution Criticality	TBC3	
BCP Impact analysis	TBD	
BCP Risk assessment	TBC4	
BCP Cost Considerations	This solution is serverless therefore would not incur costs if deployed in standby DR.	Backup and restore will serve the solution well as it is not business critical.
Average Cost Per second of outage	TBD	Expected average sales volumes for this solution is TBC
RTO	TBC5	

RPO	TBC6	
DR Strategy	Backup and restore	This is based on the criticality level
Multi-region DR	Single region	This solution does not store business critical data. The transaction database will be backed up and vaulted to ensure against ransomware.

16.5. Security

- All integration points will be on TLS
- All users will be authenticated and associated with a role
- Any role that access customer data will be assumed after an OTP verification
- All interactions with customer data will be logged
- Roles assumed by customer calls will not have delete rights
- Deletes and Drops will be blocked by an explicit deny
- This solution will be pen tested
- This GenAI parts of the solution will be red team tested
- All interactions with the GenAI model will pass through an LLM Guard Rail
-
- The knowledge base for this solution will not be shared with other projects to minimise blast radius.
- All financial transactions will be protected with an OTP

17. Signoff

Signatory name	Role	Feedback	Status
	Security		**Pending**
	Risk		**Pending**
	Product Owner		**Pending**
	Enterprise Architecture		**Pending**

18. TBC (To be confirmed)

TBC Number	Description	Status	Owner
~~TBC1~~	~~Notification mailing list for alarms and notifications~~	~~**Resolved**~~	~~Tebogo~~
TBC2	NFRs	**Pending**	Nate
TBC3	Solution Criticality	**Pending**	Nate
TBC4	Risk Assessment (By business)	**Pending**	Nate
TBC5	RPO	**Pending**	Team
TBC6	RTO	**Pending**	Team
TBC8	Risks and mitigation (Approval by Sec Team)	**Pending**	Team

19. Definition of Terms

Term	Definition
DEV	Development
UAT	User Acceptance Testing
PROD	Production
API	Application Programming Interface. A JSON/Rest service written to the JSON/REST/API standard.
BRS	Business Requirements Specification
BRD	Business Requirements Document
NFR	Non-Functional Requirements
TBC	To be confirmed
TBD	To be determined

20. Appendices

21. References

1.1. Enterprise Integration Patterns - Gregor Hohpe and Bobby Woolf
1.2. User Story workshop outcomes
1.3. User Stories - Available on request
1.4. Flight Plan - Available on request