ANALYSIS AND DESIGN DOCUMENT

- Amy Njeri - SCT211-0010/2021
- Teddy Muli - SCT211-0023/2022
- Collins Omollo - SCT211-0021/2022
- Kimberely Njoroge - SCT211-0060/2022

Introduction

BuildFlow is a construction management system designed to simplify and improve how construction projects are planned, tracked, and executed. The platform replaces traditional tools like pen-and-paper and spreadsheets with an all-in-one solution that helps project managers handle tasks such as budgeting, scheduling, worker management, and expense tracking more efficiently. BuildFlow aims to support construction teams by improving collaboration, reducing delays, and ensuring that projects stay on time and within budget.

BuildFlow is the first system of its kind in the Kenyan market and it aims to completely change how projects are managed. With the rising number of construction projects all over the country, BuildFlow is assured to be able to handle any type of projects from small ones to the biggest.

Functional Requirements.

These define the system's core features and behaviors.

User Management

- User registration and authentication (email, password, OAuth options).
- Role-based access control (**Admin, Project Manager, Worker, Accountant, etc.**).
- Profile management.

Project & Task Management

- Ability to create, update, and delete **projects, tasks, and milestones**.
- Assign roles and responsibilities to different users.
- Visual representation of deadlines (Gantt charts).

Financial & Payroll Management

- Automated payroll processing for workers.
- Expense tracking for **materials, labor, and other costs**.
- Payment integration (**Paystack**).
- Real-time budget tracking and reporting.

Communication & Collaboration

- In-app messaging or notification system for project updates.
- Commenting and tagging functionality for tasks.
- Email or SMS notifications for important project events.

Dashboard & Reporting

- Real-time analytics on **project progress, budget, and workforce management**.
- Dynamic dashboards with customizable widgets.
- Exportable reports in **PDF, CSV, and Excel** formats.

Non-Functional Requirements

Performance & Scalability

- The system should handle **multiple concurrent users** without lag.
- Optimize frontend for **fast load times (<3s)** even on low-bandwidth connections.
- Backend should support **asynchronous processing** for large transactions.

Security

- End-to-end **encryption** for data storage and transmission (**SSL/TLS**).
- **Multi-Factor Authentication (MFA)** for admins and sensitive accounts.
- Role-based access control (**RBAC**) to prevent unauthorized actions.
- **Audit logs** to track system activity and prevent fraud.

Availability & Reliability

- **99.9% uptime** with auto-scaling mechanisms.
- Regular **automated backups** to prevent data loss.
- Graceful degradation – core features should work even if some services fail.

Usability & Accessibility

- Intuitive **UI/UX with mobile responsiveness**.
- Support for **multiple languages** (e.g., English, Swahili).
- WCAG compliance for **accessibility** (screen readers, color contrast).

Compliance & Legal

- Adhere to **Kenyan data protection laws**.
- Secure handling of **financial transactions** with industry best practices.
- Maintain GDPR compliance if handling **international clients**.

Platform Requirements

Frontend (Client-Side)

- **Next.js (React.js)** for **SEO-friendly, fast, and scalable UI**.
- State management using **Redux Toolkit / React Context API**.
- Tailwind CSS

Backend (Server-Side & API)

- **Laravel** for high-performance backend APIs.
- PostgreSQL as the primary relational database.
- Redis for **caching** and improving API response time.

Deployment & DevOps

- **Frontend:** Vercel
- **Backend:** Azure
- **Database:** PostgreSQL

- CI/CD pipelines via **GitHub Actions**.

Integrations

- **Payment APIs:** Paystack
- **Email/SMS Notifications:** SendGrid .

System Architecture

1. Frontend (Client-Side)

The frontend is developed using Next.js (React.js), allowing for fast loading speeds and improved SEO for web access. It provides an intuitive and responsive user interface that allows users to manage projects, track progress, and monitor finances.

Technologies Used:

- Next.js (React) – For building UI pages and handling routing.
- React Context API – For managing application state (e.g., user sessions, project data).
- tailwind CSS – For consistent and modern styling of UI components.
- Deployment Platform: Vercel (ideal for Next.js applications, with automatic deployment from GitHub).

2. Backend (Server-Side & API Layer)

The backend is built using Laravel, an open-source PHP web framework. It handles business logic, API endpoints, and communication between the frontend and the database.

Key Backend Responsibilities:

- Handling user authentication and authorization
- Managing project data, payroll, reports, and expenses
- Serving RESTful APIs to the frontend

Technologies Used:

- Laravel – For routing, authentication and database interactions.
- PostgreSQL – As the main relational database for structured data (e.g., users, projects,

expenses)

- Redis – For caching frequently used data and speeding up response times
- Deployment Platform: Azure

3. Integrations & External Services

BuildFlow integrates with various third-party services to enhance its functionality.

- Payments:
    - Paystack: This allows users to make and receive payments directly within the platform.

4. CI/CD & DevOps

BuildFlow follows modern DevOps practices to streamline development and deployment. ● CI/CD Tools: GitHub Actions for:

- ○ Running automated tests
- ○ Building and deploying the application
- ○ Ensuring consistent code quality and faster delivery

● Database Hosting: Aiven provides managed PostgreSQL with automated backups and high availability

UI AND UX DESIGN

User Interface (UI) and UX Design of BuildFlow

BuildFlow is designed with a clean, modern, and intuitive user interface to ensure that both technical and non-technical users can manage construction projects with ease. The main focus is on simplicity, clarity, and efficiency.

1. Easy Usability

- The interface is minimalistic and clutter-free, using clear icons and labels so users can find what they need quickly.

- Navigation is simple, with a left-hand sidebar that contains main sections like *Dashboard, Projects, Payments, and Settings*.
- Each page follows a consistent layout, so users don't feel lost when switching between features.

2. Interactive Dashboards

- The Dashboard is a key part of the user experience, showing important information at a glance:
    - Project progress using visual indicators (progress bars, timelines)
    - Budget status with charts and figures
    - Upcoming deadlines and alerts
- Users can filter, sort, and drill down into specific data (e.g., view expenses for a particular project).

- Interactive graphs and reports help users track performance without needing deep technical knowledge.

3. Smooth Login and Signup Process

- The signup process is simple, asking only for essential information (name, email, password, role).

- After signing up, users are guided through a short onboarding process (tooltips, hints) to help them get started.

- Login is fast and secure, with options for password recovery and two-factor authentication (2FA) for added security.

4. Responsive and Accessible Design

- BuildFlow is fully responsive, meaning it works well on desktops, tablets, and mobile devices.

- The design considers accessibility, using high contrast colors and clear fonts to support all users, including those with visual impairments.

5. UX Highlights

- Real-time updates allow users to see changes (e.g., new payments or updated tasks) without refreshing the page.

- Role-based views ensure users only see information relevant to them, keeping the interface uncluttered.

- Quick actions (like creating a new project or paying workers) are easily accessible from any screen.

DATABASE DESIGN

The BuildFlow database is designed using a relational model to ensure data consistency, maintainability, and efficiency. It supports all critical operations—from managing workers and payments to tracking projects and expenses—while staying scalable for future growth. Some of the tables included in the database design is:

1) Users table
2) Projects table
3) Tasks table
4) Worker's table
5) Payments table
6) Expenses table
7) Activity logs table

SECURITY ARCHITECTURE

BuildFlow uses a **client-server model** where the **frontend (Next.js)** communicates with the **backend (Laravel)** through secure APIs. The goal is to protect user data, prevent unauthorized access, and ensure safe handling of financial and project information.

1. Secure Communication (Frontend ↔ Backend)

- All communication between the frontend and backend is done over **HTTPS** to **encrypt data in transit**, preventing interception (e.g., man-in-the-middle attacks).
- The frontend sends **API requests** (e.g., login, fetch projects, submit expenses) to the backend's **RESTful endpoints**.
- The backend responds with the required data, only if the request is **authenticated and authorized**.

2. Authentication & Authorization

- Users log in via a secure **login API**, and receive a **JWT (JSON Web Token)** upon successful login.
- The token is stored securely in the browser (e.g., HTTP-only cookies or local storage) and attached to each request header.
- The backend verifies the token before allowing access to protected routes.
- **Role-based access control (RBAC)** ensures users can only perform actions allowed for their role (e.g., a worker cannot access admin functions).

3. Other Key Security Measures

- **Input validation and sanitization** on both frontend and backend to prevent XSS and SQL Injection.
- **Rate limiting and logging** to detect suspicious activity or brute force attacks.
- **Environment variables** are used to securely store API keys and credentials (e.g., M-Pesa, database access).

SCALABILITY AND DEPLOYMENT STRATEGY

1. Scaling Strategy

Scalability means the system can grow and still perform well. BuildFlow uses both horizontal and vertical scaling techniques to manage traffic, data, and user load.

a) Frontend (Next.js on Vercel)

- Auto-scaling: Vercel automatically handles high traffic by scaling the frontend across

multiple servers.

- CDN (Content Delivery Network): Ensures fast loading by delivering pages from the nearest server to the user.

b) Backend (Laravel)

- Containerized deployments: Using Docker allows backend services to scale quickly.
- Horizontal scaling: Multiple instances of the backend can be deployed to handle more API requests.
- Load balancing: Distributes user requests across several backend servers to prevent overload.

c) Database (PostgreSQL on Aiven)

- Read replicas: Used to handle heavy read operations separately from writes.
- Database sharding: Can be implemented in the future to split data across multiple databases.
- Caching with Redis: Speeds up frequent queries (e.g., dashboard data, user sessions).

2. Deployment Strategy

The deployment process ensures that BuildFlow is updated quickly, reliably, and without downtime.

a) Frontend Deployment (Vercel)

● Automatic deployment on every push to the main branch.

● Preview environments for testing features before they go live.

b) Backend Deployment (Render / EC2)

● Dockerized deployment for consistency across environments.

● CI/CD pipelines (GitHub Actions) to build, test, and deploy code automatically. ●

Rollback options in case of deployment failure.

c) Database Deployment

- Hosted on Aiven for managed PostgreSQL with automatic backups, high availability, and monitoring.

- Migration scripts are used to safely update the database schema during new deployments.

d) Monitoring and Alerts

- Tools like Prometheus, Grafana, or Sentry are used to monitor system health. ● Logs and alerts help detect problems early and reduce downtime.

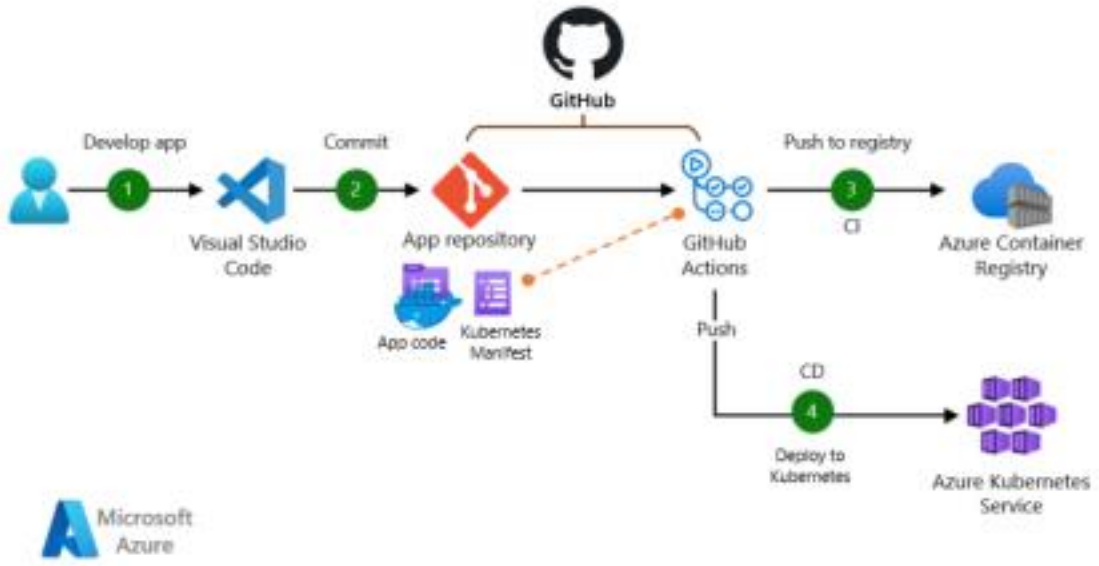
AGILE DEVELOPMENT PLAN

Week 1: Develop the application.

Week 2: Commit to GitHub

Week 3: Push to registry (Docker hub)

Week 4: Deploy to Azure



RISK ANALYSIS

1. Risk: Data Breach or Unauthorized Access

Description: Sensitive information like user details, payroll data, or financial records could be exposed if the system is not properly secured.

Mitigation Plan:

- Implement **role-based access control (RBAC)** to limit data access by user role.
- Use **JWT tokens with expiry** and secure authentication (e.g., 2FA).
- Store passwords with strong **hashing algorithms** (e.g., bcrypt).
- Ensure all data is sent over **HTTPS** and stored in **encrypted form**.

2. Risk: System Downtime or Crash During High Usage

Description: When multiple users access BuildFlow at once (e.g., end-of-month payroll), the system could crash or become very slow.

Mitigation Plan:

- Use **auto-scaling** infrastructure (e.g., Render, Vercel) to handle traffic spikes.
- Implement **load balancing** to distribute requests across servers.
- Optimize backend with **asynchronous processing** (using Celery + RabbitMQ).

3. Risk: Loss of Data Due to System Failure or Bugs

Description: Bugs or server failures could result in data loss, such as missing project records or payments.

Mitigation Plan:

- Schedule **automated daily backups** using the database host (e.g., Aiven).
- Test all new features on **staging environments** before deploying.
- Maintain **version-controlled migration scripts** for safe database changes.

- Use **unit and integration tests** to catch errors early.

4. Risk: Resistance to Adoption by Construction Firms

Description: Users may be hesitant to switch from familiar manual methods like spreadsheets and paper.

Mitigation Plan:

- Provide **simple onboarding tutorials** and in-app guidance.
- Offer a **free trial** or demo version to encourage exploration.
- Focus on **user-friendly design** with clear dashboards and navigation.
- Collect feedback and iterate using **Agile methodology** to better meet user needs.

COLLABORATION TOOLS AND WORKFLOW

1. Visual studio code
2. GitHub
3. Laravel
4. Docker hub
5. Azure

CONCLUSION

BuildFlow is a modern, scalable, and user-friendly construction management system designed to streamline project workflows, enhance collaboration, and improve financial tracking. Through careful planning, secure architecture, and intuitive design, it addresses key industry challenges and positions itself as a practical solution for construction teams.