IMPLEMENTATION DOCUMENT

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

Introduction

This chapter explains how the BuildFlow system was implemented following an **Agile and DevOps approach**. After completing the requirements gathering, analysis, and design phases, the focus shifted to coding, testing, and deployment. The implementation involved integrating the Laravel backend with a Next.js frontend, connecting them to a PostgreSQL database, and deploying using Vercel and Render.

Development Environment

Component	Technology
Frontend	Next.js (React.js)
Backend	PHP Laravel
Database	PostgreSQL
Styling	TailwindCSS
API Testing	Postman
Version Control	Git + GitHub
Deployment (Frontend)	Vercel
Deployment (Backend)	Render
CI/CD	GitHub Actions
Communication	Slack, Trello

Implementation Strategy (Agile & Sprint Breakdown)

The BuildFlow team followed an Agile approach with 5 sprints:

SpriNt	Activities	Deliverables

1	Project setup, UI wireframes, Laravel boilerplate	GitHub repo, initial auth views
2	User auth system, dashboard UI	Working login/signup system
3	Project, task, and worker modules	Fully functional core modules
4	Expense tracking, payment integration, testing	End-to-end flows, test results
5	Final deployment, performance tuning, documentation	Live app, implementation report

Module-by-Module Implementation

a) User Authentication

- Laravel's built-in authentication with **Sanctum** for token-based access.
- **Role-based access control (RBAC)** for admins, managers, and workers.

b) Core Business Logic

- Construction project creation and breakdown into units, tasks, and deadlines.
- Worker registration and payroll management linked to specific projects.

c) API Implementation

- Laravel APIs follow **RESTful standards**.
- **Postman** used for testing endpoints like `/api/projects`, `/api/payments`.

Frontend Development

- Built with **Next.js** using **React components** and **TailwindCSS**.
- Pages include Login, Dashboard, Project List, Expense Manager, and Reports.
- Responsive design for mobile and desktop.

Database Implementation

- **PostgreSQL** stores all core entities: users, projects, tasks, expenses, payments.
- Foreign keys ensure relational integrity.
- **Eloquent ORM** in Laravel used for database interactions.

Testing and Debugging

- Unit tests written in Laravel using **PHPUnit** for models and controllers.
- Manual UI testing for forms and navigation.
- Bugs tracked via GitHub Issues and addressed in sprint retrospectives.

Security Implementation

- Passwords hashed using **bcrypt**.
- API access protected via **Sanctum tokens**.
- Input validation and sanitization to prevent SQL injection and XSS.
- HTTPS enforced on all deployed endpoints.

Performance Optimization

- **Frontend:** Lazy loading of dashboard components and code splitting.
- **Backend:** Query optimization, eager loading with Eloquent.
- **Caching:** Redis planned for future implementation to speed up frequent queries.

CI/CD and Deployment

- **GitHub Actions** used for automated deployment pipelines.

- **Frontend deployed to Vercel**, backend to Render with Laravel optimizations.
- Environment variables stored securely for API keys (e.g., M-Pesa integration).

Collaboration and Workflow

- Used **Trello** for task management and sprint tracking.
- Daily stand-ups and sprint reviews done on Slack.
- Git branches for each feature; pull requests reviewed before merging to main.

Challenges and Solutions

Challenge	Solution
Switching backend from FastAPI to Laravel	Refactored architecture and database bindings in Laravel
Role-based complexity in views	Implemented middleware for user-type-based redirects
M-Pesa API sandbox issues	Mocked payment flows and isolated the logic
Time management with overlapping school work	Balanced with strict sprints and evening work sessions

Summary

The BuildFlow system was successfully implemented using modern technologies such as Laravel, Next.js, and PostgreSQL. The team followed Agile practices, applied secure and scalable architecture, and completed functional testing. The project is now fully deployed and ready for real-user feedback and future improvements.