Research document

Group Project: PhishGuard

By: Simeon Markov

Course: ICT

Date: 2026-01-21

Introduction

This document outlines my contribution to the research document, particularly the research tasks for the backend and database architecture. For the detailed document please check the '**Applied Research**' document, where the whole research for the project can be found.

Role: Software Engineer

Focus: Technology Stack Selection & Architecture (technical architect).

Research Task 1: Frontend Framework Comparison

Deliverable: A recommendation on the frontend technology with a "Why I chose this" justification.

Research Task 2: Backend & Database Options

Deliverable: An architecture diagram showing how the frontend, backend, and database will connect.

Research Task 3: Deployment & Collaboration

Deliverable: A "DevOps" plan setup guide for the team to start coding.

Research Task 1: Frontend Framework Comparison:

FRONTEND FRAMEWORK: Vue.js 3 + Nuxt 3

Why This Choice? [1] [7]

Performance Advantages:

- 19% faster startup time vs React (critical for mobile users)
- 36% faster DOM manipulation (interactive quizzes respond instantly)
- Smaller initial JavaScript bundle (reduced bandwidth for global audiences)
- Better performance on lower-end devices (reaches underserved markets)

Developer Experience:

- HTML-based templates (easier to learn than JSX)
- Pinia state management (significantly simpler than Redux/Context API)
- Auto-imports (less boilerplate code)
- Composition API with <script setup> syntax (modern, intuitive)
- TypeScript support (first-class, not bolted-on)
- File-based routing (folder structure = application structure)

Team Productivity:

- Smaller learning curve = faster hiring
- Built-in development server with hot module reloading
- Excellent developer tools (Vue DevTools with timeline debugging)

Educational Clarity:

- Vue mirrors HTML structure (closer to traditional web development)
- Easier to teach programming concepts without abstraction layers
- Clear separation: templates, scripts, styles

Styling: Tailwind CSS + shadcn/ui

Why Tailwind CSS:

- Industry standard in 2025 (definitively replaced Bootstrap)
- Utility-first approach enables rapid, consistent design
- Zero CSS naming conflicts (eliminates "what should I call this class?" decisions)
- Design tokens built-in (color palette, spacing, sizing)
- Tree-shakeable (unused styles don't ship to production)
- Dark mode support built-in (modern user expectation)

Why shadcn/ui (React) or Headless UI (Vue):

- **Copy-paste components** - Not locked into library updates.
- **Accessibility guaranteed** - WCAG 2.1 compliance by default
- **Full customization** - modifying components without dealing with CSS overrides
- **Small bundle** - only components that are needed
- **Type-safe** - TypeScript-first component design
- **Educational value** - beginners see production-ready component patterns

Research Task 2: Backend & Database Options

2. BACKEND FRAMEWORK: Nitro (Nuxt Server) + Node.js

Why Unified Nuxt Full-Stack? [6]

Single Application Deployment:

- Frontend (Vue components) + Backend (API routes) = one Git repo
- One deployment command instead of two
- Environment variables managed once
- Single monitoring/logging system

Simpler Development:

- One Node.js version to manage

- One package manager (pnpm)
- One testing setup
- One CI/CD pipeline

3. DATABASE: PostgreSQL [3]

ACID Transactions:

- User progress update cannot partially fail (all-or-nothing guarantee)
- Quiz submission with score recording must be atomic
- Payment processing requires guarantees

Complex Queries:

- Aggregate user progress: "Show me lesson completion %, average quiz score, learning streak"
- Requires multi-table joins: Users → Lessons → Quiz_Responses → Scores

Analytics:

- Window functions for ranking (leaderboards)
- Aggregation functions for statistics
- Time-series analysis (learning over weeks/months)

Research Task 3: Deployment & Collaboration

4. DEPLOYMENT: Railway [2]

- npm run deploy or GitHub integration
- Automatic SSL certificates (no manual setup)
- PostgreSQL included (add via UI, provisioned instantly)
- Environment variables stored securely

CI/CD Pipeline: GitHub Actions [4] [5]

Why GitHub Actions?

- Native GitHub integration (no external tools)
- Free for public repos, generous free tier for private
- Marketplace with pre-built actions

Testing:

- **Vitest** - Fast unit testing (10x faster than Jest)

- **Playwright** - End-to-end testing (test real user flows)
- **Percy** - Visual regression testing (catch UI changes)

Layer	Technology	Why
Frontend Framework	Vue.js 3 + Nuxt 3	Faster, simpler, better DX
Styling	Tailwind CSS + shadcn/ui	Industry standard, zero conflicts
Type Safety	TypeScript	Shared frontend-backend types
State Management	Pinia	Simpler than Redux, excellent DevTools
Backend	Nitro (Nuxt server)	Unified fullstack, single deployment
Database	PostgreSQL	ACID guarantees, complex queries, analytics
ORM	Prisma	Type-safe, auto-generated types
Deployment	Railway	Fast, cheap, PostgreSQL included
Repository	Monorepo + Turborepo	Shared types, single source of truth
Package Manager	pnpm	Faster, efficient, monorepo support

Testing	Vitest + Playwright	Fast unit tests, real user flow E2E
CI/CD	GitHub Actions	Native GitHub, no external tools
Code Quality	ESLint + Prettier + TypeScript	Automated standards enforcement

Conclusion

In conclusion, after the research is done, the realization of the project becomes very straightforward with clear steps to be followed. With proper time management and group collaboration the project would be finished within the given deadline.

References:

[01] Top 7 Frontend Frameworks for Web Development in 2025. https://webpinn.com/best-7-frontend-frameworks-that-accelerate-pace-of-web-development/

[02] Trang. chủ (2025). Railway Review 2025 – A Modern Deployment Platform for Developers, Startups, SMEs, and Students https://ikigaiteck.com/pages/railway-review-2025-modern-app-deployment-platform

[03] Hamza, Khan (2025). PostgreSQL vs. MongoDB in 2025: Which Database Should Power Your Next Project? https://dev.to/hamzakhan/postgresql-vs-mongodb-in-2025-which-database-should-power-your-next-project-2h97

[04] Learn continuous integration with GitHub Actions. https://learn.microsoft.com/en-us/training/modules/learn-continuous-integration-github-actions/

[05] Omar, Elhawary (2023). Building a full-stack TypeScript application with Turborepo. https://blog.logrocket.com/build-full-stack-typescript-application-turborepo/

[06] Arunangshu, Das (2025). Choosing the Right Node.js Framework: Options and Comparisons. https://arunangshudas.com/blog/choosing-node-js-framework-options-and-comparisons/

[07] Top Frontend Technologies, Tools, and Frameworks to Use in 2025. https://www.wedowebapps.com/frontend-technologies-tools-frameworks/