Vysoká škola báňská - Technická univerzita Ostrava
Fakulta elektrotechniky a informatiky

Vývoj informačních systémů

MiniGitHub

Jméno: Phat Tran Dai
Osobní číslo: TRA0163
Datum: 30. 09. 2025

Content

Vision

1.1. Introduction

The purpose of this document is to define the high-level needs and features of MiniGitHub, a web-based code collaboration platform. MiniGitHub is designed to help developers and teams manage code repositories, track changes, and collaborate effectively on software projects.

This document provides an overview of the product's purpose, positioning in the market, stakeholders and users, and its key features. It will serve as a guide for aligning development efforts and ensuring the solution meets user expectations.

1.2. Positioning

1.2.1. Problem Statement

The problem of	fragmented and inefficient collaboration on code projects
affects	developers/student teams, open-source contributors, and organizations that rely on version control and collaboration tools
the impact of which is	code conflicts, slower development cycles, miscommunication, lack of version tracking, and difficulties in managing distributed teams
a successful solution would be	a simple and accessible platform that enables users to store, manage, and share code repositories, track changes through versioning, and collaborate effectively with features such as issue tracking and pull requests.

Table 1: Problem statement

1.2.2. Product Position Statement

For	developers, student teams, and organizations seeking a lightweight but structured collaboration platform
Who	need a reliable and accessible version control and project collaboration tool
The MiniGitHub system	is a repository hosting and collaboration platform
That	provides version control, issue tracking, and lightweight collaboration features in a user-friendly environment
Unlike	existing heavyweight platforms such as GitHub or GitLab, which may be overly complex for small teams
Our product	offers a streamlined, focused solution with only the essential collaboration tools.

Table 2: Product Position Statement

1.3. Stakeholder and User Descriptions

1.3.1. Stakeholder Summary

Name	Description	Responsibilities
Project Sponsor	Provides funding or backing for development	Defines goals and approves overall direction
Development Team	Engineers building the system	Implements features, ensures maintainability
System Administrator	Manages hosting and deployment	Keeps system online and secure

Table 3: Stakeholder Summary

1.3.2. User Summary

Name	Descriptions	Responsibilities	Stakeholder
Developer	Individual writing and committing code	Creates repositories, commits changes, opens issues	Development Team
Maintainer	Oversees project repositories	Reviews contributions, manages issues and pull requests	Development Team
Contributor	External collaborator	Suggests changes, reports bugs, submits pull requests	Development Team

Table 4: User Summary

1.3.3. User Environment

Number of users per project	Typically 2–10, may scale for open collaboration.
Task cycle	Continuous; developers frequently commit, push, and review code.
Environment constraints	Primarily desktop/laptop through a web browser; later expansion to mobile is possible.
Platforms in use	Modern web browsers for front-end, PostgreSQL/SQL Server for data, ASP.NET/Python-based backend. WPF desktop application for Windows.

1.4. Product Overview

1.4.1. Product Perspective

MiniGitHub is an independent system designed as a lightweight GitHub alternative. It consists of three major layers:

- Presentation Layer: Web interface and desktop application for interaction.
- Domain Layer: Business logic for repository, commit, and issue management.
- Data Layer: Database with persistence services.

Interfaces:

- Desktop application (for end users)
- Web browser (for end users)
- REST API (for integration with Git clients)

1.4.2. Assumptions and Dependencies

System assumes internet-connected users with modern web browsers.

Database system (SQL Server or PostgreSQL) must be available.

Hosting environment supports .NET (if on Windows) or Python (if on Linux) runtime.

Full Git implementation is out of scope; repository management will be simplified.

1.5. Product Features

User management	Registration, login, authentication.
Repository management	Create, list, delete repositories.
Commits	Track file version and changes.
Issue	Open, comment on, and close issues.
Pull requests (optional)	Propose and merge proposed changes.
Search and browse	Find repositories and issues.
Collaboration tools	Comments, activity feeds.
Basic analytics	Number of commits, contributors, and issues per repo.

1.6. Other Product Requirements

Standards	Follows MVC architecture for the web front end, layered design (Data Access, Domain, Presentation), and selected patterns (Domain Model, Data Mapper, Unit of Work).
Performance	Must support 50–100 concurrent users with acceptable response time.
Robustness	Transactions must ensure consistent repository state.
Platform requirements	Runs on Linux/Windows server, database backend required.
Usability	Clean and intuitive UI, minimal learning curve.
Documentation	Basic user guide and technical developer documentation.
Constraints	Project scope limited to core GitHub-like features (full Git replication is out of scope). Only core repository and collaboration features are included.