COMS3009A- Software Design

Web Application Project 2025

Freelancing Platform - Comprehensive Project Plan

Group Members:

Names	Student Numbers
Dolf Shungube	2683067
Faranani Nemutanzhela	2651532
Bokang Madondo	2681261
Mashego Mabeloane Gideon	2654606
Mutsawashe Njowa	2651487
Nelson Bhekiswayo	2681283

1. Executive Summary

This document outlines the comprehensive project plan for developing a web-based freelancing platform that connects clients with freelancers. The platform addresses the common challenges businesses and individuals face when hiring freelancers, including inefficient contract management, payment tracking, and deliverable monitoring. Similarly, it solves freelancers' difficulties in project and payment tracking.

Key Objectives

- Create a centralized platform for freelancer-client interactions
- Implement secure user authentication and authorization
- Facilitate seamless job posting and application processes
- Enable real-time communication and project tracking
- Integrate milestone-based payment systems
- Provide comprehensive reporting and analytics

2. Project Overview

2.1 Problem Statement

Many businesses and individuals hire freelancers but struggle to manage contracts, payments, and deliverables efficiently. Freelancers also face challenges in tracking their projects and payments, leading to miscommunication, delayed payments, and project management inefficiencies.

2.2 Solution Approach

Development of a web-based platform using modern technologies that enables:

- Streamlined contract and task management
- Transparent communication channels
- Automated payment tracking and processing
- Comprehensive project progress monitoring
- Detailed reporting and analytics

2.3 Technology Stack

- **Frontend**: HTML5, CSS3, JavaScript
- **Backend-as-a-Service**: Supabase
- **Authentication**: Third-party identity providers (Google OAuth)
- **Payment Integration**: Payment gateway APIs
- **Deployment**: Cloud hosting platform-Microsoft Azure
- **CI/CD**: Automated testing and deployment pipelines-git

3. Project Methodology

3.1 Development Approach

- **Methodology**: Agile Development with 4 Sprint cycles
- **Testing Strategy**: Test-Driven Development (TDD) with bottom-up testing approach
- **CI/CD Integration**: Continuous Integration and Continuous Deployment principles
- **Version Control**: Git-based workflow with feature branches

3.2 Quality Assurance

- Unit testing for all core functions
- Integration testing for API endpoints
- User acceptance testing for each sprint deliverable
- Performance testing for scalability

4. Core Features and Requirements

4.1 User Management and Authentication

Feature	Description	Priority
User Registration	Email/password registration for all user types	High
Third-party Authentication	Google OAuth integration	High
User Verification	Identity verification for Freelancers, Clients, and Admins	High
Profile Management	User profile creation and management	High
Role-based Access Control	Different permissions for each user type	High

4.2 Job Management System

Feature	Description	Priority
Job Posting	Clients can create and publish job opportunities	High
Job Browsing	Freelancers can view and search available jobs	High
Job Applications	Freelancers can apply for jobs with CV upload	High
Application Management	Clients can review and select freelancers	High
Contract Generation	Automated contract creation and signing	Medium

4.3 Project Management

Feature	Description	Priority
Task Tracking	Clients can create and monitor project tasks	High

Feature	Description	Priority
Milestone Setting	Define project milestones and deadlines	High
Progress Monitoring	Real-time project progress tracking	High
Feedback System	Client feedback and rating system	Medium
Document Management	File upload and sharing capabilities	Medium

4.4 Communication System

Feature	Description	Priority
Real-time Messaging	Chat system between clients and freelancers	High
Message History	Conversation history and search	Medium
Notification System	Unread message notifications	High
File Sharing	Document and media sharing in chat	Low

4.5 Payment Integration

Feature	Description	Priority
Milestone-based Payments	Payment release based on milestone completion	High
Payment Gateway Integration	Secure payment processing	High
Payment Tracking	Transaction history and status monitoring	High
Invoice Generation	Automated invoice creation	Medium
Payment Disputes	Dispute resolution system	Low

4.6 Reporting and Analytics

Feature	Description	Priority
Project Completion Dashboard	Visual project completion rates	High
Payment Reports	Payments received and pending analysis	High
Custom Reports	User-defined report generation	Medium
Export Functionality	CSV and PDF export capabilities	Medium
Performance Analytics	Platform usage and performance metrics	Low

4.7 Bonus Features

Feature	Description	Priority
AI Job Matching	Intelligent freelancer-job matching system	Low

5. Sprint Planning and Timeline

Sprint 1: Foundation and Authentication (April 7-14, 2025)

Duration: 7 days **Sprint Goal**: Establish core authentication and user management system

User Stories

- As a user, I want to register with email and password so that I can access the platform
- As a user, I want to login securely so that I can access my account
- As a user, I want to logout safely so that my account remains secure
- As a user, I want to see a welcome message after registration/login for confirmation
- As a user, I want to authenticate via Google OAuth for convenience (Bonus)

Deliverables

- User registration system with email/password
- Secure login/logout functionality
- User role management (Client, Freelancer, Admin)
- Welcome dashboard for each user type
- Google OAuth integration (if time permits)
- Basic user profile setup

Acceptance Criteria

- Users can successfully register with valid email/password
- Login system authenticates users and redirects to appropriate dashboard
- Logout functionality clears session securely
- Role-based redirection works correctly
- Basic error handling for authentication failures

Technical Tasks

- Set up Supabase authentication
- Create user registration forms
- Implement login/logout logic
- Design user dashboards

- Set up database schemas for users
- Implement Google OAuth (bonus)

Sprint 2: Job Management Core (April 14-21, 2025)

Duration: 7 days **Sprint Goal**: Implement core job posting and application functionality

User Stories

- As a client, I want to post job opportunities so that freelancers can apply
- As a freelancer, I want to view posted jobs so that I can find work opportunities
- As a freelancer, I want to search and filter jobs so that I can find relevant opportunities
- As a freelancer, I want to accept jobs so that I can start working
- As a freelancer, I want to view my current jobs so that I can track my workload
- As both client and freelancer, I want to track job progress so that I can monitor project status
- As a client, I want to see who applied and accepted jobs so that I can manage my projects

Deliverables

- Job posting interface for clients
- Job browsing and search functionality for freelancers
- Job application system
- Job acceptance workflow
- Progress tracking dashboard
- Application management for clients

Acceptance Criteria

- Clients can successfully create and publish job posts
- Freelancers can browse, search, and filter available jobs
- Job application process works end-to-end
- Progress tracking displays accurate project status
- Clients can view all applicants and accepted freelancers

Technical Tasks

- Design job posting forms and validation
- Create job database schema
- Implement job search and filtering

- Build application submission system
- Create progress tracking components
- Develop client job management interface

Sprint 3: Communication and Contract Management (April 21 - May 5, 2025)

Duration: 14 days **Sprint Goal**: Implement communication system and contract management

User Stories

- As a freelancer, I want to upload and update my CV so that clients can evaluate my skills
- As a client, I want to choose freelancers for jobs so that I can build my project team
- As a client, I want to send contracts to freelancers so that we can formalize our agreement
- As both client and freelancer, I want to communicate via chat so that we can collaborate effectively
- As both client and freelancer, I want to view past conversations so that I can reference previous discussions
- As a user, I want milestone-based payments so that I can ensure fair compensation (Bonus)

Deliverables

- CV upload and management system
- Freelancer selection and hiring workflow
- Contract generation and signing system
- Real-time chat functionality
- Message history and search
- Payment system integration (if time permits)

Acceptance Criteria

- Freelancers can upload, update, and manage their CVs
- Clients can successfully hire freelancers and send contracts
- Chat system enables real-time communication
- Message history is preserved and searchable
- Contract workflow is complete and functional

Technical Tasks

- Implement file upload for CVs and documents

- Create freelancer selection interface
- Design contract templates and workflow
- Build real-time messaging system
- Implement message storage and retrieval
- Integrate payment gateway (bonus)

Sprint 4: Notifications, Payments, and Polish (May 5-19, 2025)

Duration: 14 days **Sprint Goal**: Complete platform with notifications, payments, and final polish

User Stories

- As a user, I want to receive notifications for unread messages so that I don't miss important communications
- As both client and freelancer, I want milestone-based payments so that compensation is tied to project progress
- As a user, I want a polished, professional interface so that the platform is easy and pleasant to use
- As a user, I want reliable performance so that I can depend on the platform for my work

Deliverables

- Comprehensive notification system
- Milestone-based payment implementation
- UI/UX improvements and polish
- Performance optimization
- Bug fixes and quality improvements
- Documentation and user guides

Acceptance Criteria

- Users receive timely notifications for unread messages
- Payment system processes milestone-based payments accurately
- Platform performs reliably under normal usage
- User interface is intuitive and professional
- All major bugs are resolved

Technical Tasks

- Implement notification system
- Complete payment system integration

- Conduct UI/UX review and improvements
- Performance testing and optimization
- Comprehensive bug testing and fixes
- Create user documentation

6. Risk Management

6.1 Technical Risks

Risk	Impact	Probability	Mitigation Strategy
Third-party API Integration Issues	High	Medium	Early integration testing, fallback options
Database Performance Issues	High	Low	Regular performance monitoring, optimization
Security Vulnerabilities	High	Medium	Regular security audits, best practices implementation
Browser Compatibility Issues	Medium	Low	Cross-browser testing, progressive enhancement

6.2 Project Risks

Risk	Impact	Probability	Mitigation Strategy
Scope Creep	High	Medium	Clear requirements documentation, change control
Timeline Delays	Medium	Medium	Buffer time allocation, regular progress monitoring
Team Availability	Medium	Low	Resource planning, knowledge sharing
Requirement Changes	Medium	Medium	Agile methodology, regular stakeholder communication

6.3 Business Risks

Risk	Impact	Probability	Mitigation Strategy
Market Competition	Medium	High	Focus on unique features, user experience
User Adoption	High	Medium	User feedback integration, iterative improvement
Scalability Issues	High	Low	Cloud-based architecture, performance testing

7. Success Metrics and KPIs

7.1 Technical Metrics

- **Code Quality**: 80%+ test coverage, zero critical security vulnerabilities
- **Performance**: Fast page load time and high uptime percentage
- **Reliability**: Zero data loss incidents, successful automated deployments

7.2 Functional Metrics

- **User Authentication**: 100% successful login rate for valid credentials
- **Job Management**: Complete job posting to completion workflow
- **Communication**: Fast real-time message delivery
- **Payment Processing**: 100% accurate milestone-based payment calculations

7.3 User Experience Metrics

- **Usability**: high successful task completion rate
- **User Satisfaction**: good post-sprint user feedback scores
- **Platform Adoption**: Active user engagement metrics

8. Resource Allocation

8.1 Team Structure

- **Product Owner**: Requirements management, stakeholder communication
- **Scrum Master**: Process facilitation, impediment removal
- **Frontend Developers**: User interface implementation
- **Backend Developers**: Server-side logic and database management
- **Test Analyst**: Test frontend and backend code

8.2 Technology Resources

- **Development Environment**: Cloud-based development tools
- **Testing Environment**: Automated testing frameworks
- **Production Environment**: Scalable cloud hosting
- **Monitoring Tools**: Performance and error monitoring systems

9. Quality Assurance Plan

9.1 Testing Strategy

- **Unit Testing**: Test-driven development for all core functions
- **Integration Testing**: API endpoint and service integration testing
- **System Testing**: End-to-end workflow testing
- **User Acceptance Testing**: Stakeholder validation of sprint deliverables

9.2 Code Quality Standards

- **Code Reviews**: Mandatory peer review for all code changes
- **Coding Standards**: Consistent coding conventions and documentation
- **Static Analysis**: Automated code quality and security scanning
- **Performance Testing**: Load testing and optimization

10. Deployment and DevOps

10.1 CI/CD Pipeline

- **Continuous Integration**: Automated testing on code commits
- **Continuous Deployment**: Automated deployment to staging and production
- **Environment Management**: Separate development, staging, and production environments
- **Rollback Strategy**: Quick rollback capability for production issues

10.2 Monitoring and Maintenance

- **Application Monitoring**: Real-time performance and error monitoring
- **Security Monitoring**: Continuous security threat detection
- **Backup Strategy**: Regular automated backups with recovery testing
- **Update Management**: Regular security updates and dependency management

11. Post-Launch Considerations

11.1 Maintenance and Support

- **Bug Fix Protocol**: Rapid response to critical issues
- **Feature Enhancement**: Regular feature updates based on user feedback
- **Performance Optimization**: Ongoing performance monitoring and improvement
- **Security Updates**: Regular security patches and vulnerability assessments

11.2 Scalability Planning

- **Infrastructure Scaling**: Cloud-based auto-scaling capabilities
- **Database Optimization**: Performance tuning and query optimization
- **CDN Implementation**: Content delivery network for global performance
- **Load Balancing**: Distributed load handling for high traffic

12. Conclusion

This comprehensive project plan provides a structured approach to developing a robust freelancing platform that addresses real market needs. The Agile methodology with four focused sprints ensures iterative development and continuous improvement. The emphasis on testing, quality assurance, and CI/CD practices will result in a reliable, scalable platform that can grow with user demands.

The success of this project depends on adherence to the defined timeline, maintaining code quality standards, and regular communication with stakeholders. With proper execution of this plan, the platform will provide significant value to both freelancers and clients in managing their professional relationships and projects.