Product Requirements Document (PRD)

Technical Skills Learning & Practice Platform

Document Version: 1.0
Last Updated: November 19, 2025
Project Status: Planning Phase
Document Owner: Product Team

Executive Summary

This document outlines the requirements for developing a comprehensive technical skills learning and practice platform, inspired by successful platforms like GeeksforGeeks and HackerRank. The platform will provide structured courses, topic-based practice problems, interactive code execution, daily/weekly challenges, community discussions, and a scoring system that learners can showcase to potential employers.

Key Objectives

- Provide structured learning paths for technical skills (programming, data structures, algorithms, web development, etc.)
- Enable hands-on practice with instant feedback through an integrated code editor
- Facilitate skill validation through time-bound challenges with scoring/ranking
- Build an active community for peer learning and discussion
- Create an admin portal for content management and platform maintenance

---

Product Vision

Vision Statement: To become the go-to platform for aspiring developers and tech professionals to learn, practice, and validate their technical skills through structured courses and competitive challenges.

Target Audience:

- College students preparing for technical interviews
- Self-learners transitioning into tech careers
- Working professionals upskilling in new technologies
- Coding bootcamp participants seeking additional practice
- Job seekers wanting to showcase verified skills to employers

---

Core Features & Requirements

1. User Management System

1.1 User Roles

Learner/Student (Primary User)

- Browse and enroll in courses
- Practice problems and take challenges
- Track personal progress and scores
- Participate in discussions
- View and submit solutions
- Build a skills profile/portfolio

Admin/Content Manager

- Create, update, and delete courses
- Organize courses into topics and modules
- Add/edit practice problems and test cases
- Create and schedule challenges
- Moderate discussions
- View platform analytics

Guest User

- Browse course catalog (limited preview)
- View sample problems
- Access registration/login pages

1.2 Authentication & Authorization

Requirements:

- Email/password registration and login
- Social authentication (Google, GitHub OAuth)
- Email verification for new accounts
- Password reset functionality
- Role-based access control (RBAC)
- Session management with JWT tokens
- Two-factor authentication (optional, future enhancement)

User Profile Features:

- Personal information (name, bio, location)
- Skills and interests tags
- Progress tracking dashboard
- Challenge participation history
- Score/ranking display
- Badges and achievements
- Solution submission history
- Social links (GitHub, LinkedIn)

---

2. Course Management System

2.1 Course Structure

Hierarchical Organization:

```
Course (e.g., "Data Structures & Algorithms")
├── Topic 1 (e.g., "Arrays")
│   ├── Theory/Content
│   ├── Code Examples
│   ├── Practice Problems (5-10 problems)
│   └── Solutions
├── Topic 2 (e.g., "Linked Lists")
│   ├── Theory/Content
│   ├── Code Examples
│   ├── Practice Problems
│   └── Solutions
└── Topic N...
```

Course Components:

- Course title and description
- Difficulty level (Beginner, Intermediate, Advanced)
- Estimated completion time
- Prerequisites (if any)
- Learning outcomes/objectives
- Table of contents (topic index)
- Instructor/author information
- Tags and categories

2.2 Topic/Module Features

Content Elements:

- Rich text editor for theory content
- Markdown support for formatting
- Code syntax highlighting
- Embedded images and diagrams
- Video embeds (YouTube, Vimeo)
- External resource links
- Downloadable supplementary materials

Practice Problems per Topic:

- Problem statement with examples
- Input/output specifications
- Constraints and edge cases
- Difficulty rating (Easy, Medium, Hard)
- Topic tags and categories
- Sample test cases (visible)
- Hidden test cases (for validation)
- Time and memory limits
- Editorial/approach hints
- Multiple solution approaches

Example Code:

- Code snippets demonstrating concepts
- Multiple language support (Python, Java, C++, JavaScript)
- Runnable code examples
- Commented explanations

---

3. Code Practice Environment

3.1 Integrated Code Editor

Editor Features:

- Syntax highlighting for multiple languages
- Auto-completion and IntelliSense
- Code formatting (Prettier/Black integration)
- Line numbering
- Theme selection (Light/Dark mode)
- Font size adjustment
- Keyboard shortcuts support
- Code templates for different languages

Supported Languages:

- Python 3.x
- Java (JDK 11+)
- C++ (C++14/17)
- JavaScript (Node.js)
- C
- Additional languages (PHP, Ruby, Go - future phases)

3.2 Code Execution Engine

Execution Requirements:

- Sandboxed code execution environment
- Real-time output display
- Error handling and debugging messages
- Test case execution and validation
- Time limit enforcement (typically 2-5 seconds)
- Memory limit enforcement (typically 256MB)
- Multiple test case evaluation
- Detailed feedback on failures

Security Measures:

- Containerized execution (Docker)
- Resource limitations (CPU, memory, network)
- Prohibited system calls blocking
- Execution timeout handling
- Code size limitations

3.3 Problem Solving Interface

User Workflow:

1. Read problem statement
2. Select programming language
3. Write/paste code in editor
4. Run against sample test cases
5. Submit solution for evaluation
6. View results (Accepted/Wrong Answer/TLE/MLE/Runtime Error)
7. View detailed test case results
8. Access discussion forum for the problem
9. View solutions (after solving or unlocking)

Submission Tracking:

- Submission history per problem
- Success/failure status
- Execution time and memory usage
- Code versioning (view previous submissions)
- Language used for each attempt
- Timestamp of submissions

---

4. Challenge System

4.1 Challenge Types

Daily Challenge

- One problem released daily at fixed time (e.g., 12:00 AM UTC)
- Available for 24 hours
- Scoring based on completion time and accuracy
- Leaderboard resets daily

Weekly Challenge

- Multiple problems (3-5 problems)
- Released every Monday/Weekend
- Duration: 7 days
- Combined scoring from all problems
- Weekly leaderboard and rankings

Monthly Contest

- Competitive programming style contest
- 5-8 problems of varying difficulty
- Duration: 2-3 hours
- Real-time leaderboard
- Prizes/recognition for top performers

Special Themed Challenges

- Topic-specific (e.g., "Dynamic Programming Week")
- Company-specific problem sets (e.g., "Google Interview Prep")

- Seasonal events (e.g., "New Year Coding Marathon")

4.2 Challenge Features

Challenge Structure:

- Challenge title and description
- Start and end datetime
- Problem set (1-8 problems)
- Scoring mechanism
- Participation rules
- Prize/reward information (if applicable)

Scoring System:

- Base points for problem difficulty (Easy: 100, Medium: 200, Hard: 300)
- Time bonus (earlier submission = higher bonus)
- Penalty for wrong submissions (-10 to -20 points per failed attempt)
- Partial scoring for some test cases passed
- Final score = Sum of all problem scores + time bonuses - penalties

Leaderboard:

- Real-time ranking during active challenges
- Display: Rank, Username, Score, Problems Solved, Last Submission Time
- Filter by country/region (optional)
- Historical challenge rankings
- Profile integration (showcase top rankings)

4.3 Challenge Administration

Admin Capabilities:

- Create new challenges
- Schedule challenge start/end times
- Add/remove problems from challenge
- Set scoring rules and time bonuses
- Enable/disable challenges
- Monitor participation and submissions
- Generate challenge analytics and reports

---

5. Discussion & Community Features

5.1 Discussion Forum

Forum Structure:

- Problem-specific discussion threads
- Course/topic general discussions
- Challenge discussion boards (unlocked after challenge ends)
- Upvote/downvote mechanism
- Threaded replies and comments
- Markdown formatting support
- Code snippet embedding

Discussion Features:

- Create new discussion posts
- Reply to existing discussions
- Edit/delete own posts
- Report inappropriate content
- Follow/unsubscribe from threads
- Tag users with @ mentions
- Search discussions by keywords
- Sort by: Latest, Most Upvoted, Oldest

Moderation:

- Admin/moderator roles for content moderation
- Flag system for inappropriate content
- Ban/mute users for violations
- Content deletion and editing
- Automated spam detection

5.2 Solution Sharing

Solution Access Rules:

- Locked until user solves the problem OR
- Unlockable after N failed attempts OR
- Premium feature (access all solutions)

Solution Display:

- Multiple solutions by different users
- Language-specific solutions
- Upvote/downvote solutions
- Comment on solutions
- Explanation/approach description
- Time and space complexity analysis
- Alternative approaches

Solution Submission:

- Submit solution with explanation
- Code formatting and syntax highlighting
- Add approach description
- Tag complexity analysis
- Optional video explanation link

---

6. Progress Tracking & Gamification

6.1 User Dashboard

Progress Metrics:

- Total problems solved (by difficulty)
- Course completion percentage
- Current streak (consecutive days of practice)
- Longest streak achieved
- Total submissions made
- Success rate percentage
- Topics mastered
- Languages used

Visual Representations:

- Progress bars for courses
- Heatmap calendar (GitHub-style activity)
- Skill level graphs
- Problem-solving trends over time
- Language distribution pie chart

6.2 Achievement System

Badges:

- First Problem Solved
- 10/50/100/500/1000 Problems Solved
- Complete a Course
- 7/30/100 Day Streak
- Win a Daily Challenge
- Top 10 in Weekly Challenge
- Language Specialist (solve 50 problems in one language)
- Topic Master (complete all problems in a topic)
- Community Contributor (10+ upvoted discussions)

Levels/Ranks:

- Beginner (0-50 problems)
- Intermediate (51-200 problems)
- Advanced (201-500 problems)
- Expert (501-1000 problems)
- Master (1000+ problems)

Leaderboards:

- Global leaderboard (all users)
- Country/region-specific leaderboards
- Course-specific leaderboards
- Weekly active users
- All-time top solvers

6.3 Skills Profile/Portfolio

Shareable Profile:

- Public profile URL (username-based)
- Skills and expertise tags
- Total problems solved with breakdown
- Challenge rankings and scores
- Badges and achievements display
- Activity heatmap
- Top languages used
- Recent submissions
- Solution contributions

Resume Integration:

- Downloadable skills certificate
- Verified challenge scores
- Skill endorsements
- Embed widget for LinkedIn/portfolio sites
- Direct link for recruiters/interviewers

---

7. Admin Dashboard & Content Management

7.1 Course Management

Admin Capabilities:

- Create new courses
- Edit course details (title, description, level, tags)
- Organize course topic/module structure
- Reorder topics within a course
- Publish/unpublish courses
- Delete courses (with confirmation)
- Clone/duplicate courses
- Import course content (JSON/CSV)

Topic Management:

- Create topics within courses
- Add rich text content/theory
- Upload images and attachments
- Embed videos and external resources
- Set topic prerequisites
- Reorder topics in course index
- Publish/draft status

Problem Management:

- Create new problems
- Add problem statement (Markdown editor)
- Set difficulty level and tags
- Add sample input/output examples

- Create test cases (visible and hidden)
- Set time and memory limits
- Assign problems to topics/courses
- Multi-language editorial/solution
- Problem versioning and history

7.2 Challenge Management

Challenge Creation:

- Create challenge from template
- Set challenge type (Daily/Weekly/Monthly/Special)
- Schedule start and end times
- Add problems to challenge
- Configure scoring rules
- Set time bonuses and penalties
- Add description and rules
- Enable/disable submissions

Challenge Monitoring:

- View real-time participation stats
- Monitor submission trends
- Check leaderboard status
- Handle user reports/issues
- Extend/modify challenge duration
- Pause/resume challenges

7.3 Analytics & Reporting

Platform Metrics:

- Total registered users
- Active users (daily/weekly/monthly)
- Course enrollment statistics
- Problem-solving trends
- Challenge participation rates
- User retention metrics
- Popular courses and topics
- User geographic distribution

Content Analytics:

- Problem difficulty vs solve rate
- Most/least solved problems
- Average time per problem
- Language usage distribution
- Discussion engagement metrics
- Solution view counts

Export Capabilities:

- Download reports as CSV/Excel
- Generate PDF summaries

- Scheduled email reports
- Custom date range filtering

Technical Architecture

8.1 System Architecture

Architecture Pattern: Microservices-based with Event-Driven components

Core Services:

1. **User Service** - Authentication, authorization, profile management
2. **Course Service** - Course, topic, and content management
3. **Problem Service** - Problem creation, storage, and retrieval
4. **Code Execution Service** - Sandboxed code execution and evaluation
5. **Challenge Service** - Challenge creation, scheduling, and scoring
6. **Discussion Service** - Forum threads, comments, and moderation
7. **Analytics Service** - Usage tracking and reporting
8. **Notification Service** - Email, push notifications, and alerts

8.2 Technology Stack Recommendations

Frontend:

- Framework: React.js or Angular (based on your existing expertise)
- UI Library: Material-UI or Tailwind CSS
- Code Editor: Monaco Editor (VS Code editor) or CodeMirror
- State Management: Redux Toolkit or Context API
- API Communication: Axios or Fetch API

Backend:

- Framework: Python Flask or FastAPI (aligns with your experience)
- Alternative: Node.js with Express (if preferred)
- API: RESTful API with OpenAPI/Swagger documentation
- WebSocket: For real-time features (live challenges, notifications)

Database:

- Primary: PostgreSQL (relational data - users, courses, problems)
- Cache: Redis (sessions, leaderboards, frequently accessed data)
- Search: Elasticsearch (problem search, discussion search)
- File Storage: AWS S3 or MinIO (images, attachments, code submissions)

Code Execution:

- Containerization: Docker
- Orchestration: Kubernetes (for scaling) or Docker Compose (for smaller scale)
- Code Runner: Judge0 API (open-source) or custom Docker executor
- Queue: RabbitMQ or Celery (for async code execution jobs)

Infrastructure:

- Cloud Provider: AWS, Google Cloud, or Azure
- CDN: CloudFlare or AWS CloudFront
- Load Balancer: Nginx or AWS ALB
- Monitoring: Prometheus + Grafana, or New Relic
- Logging: ELK Stack (Elasticsearch, Logstash, Kibana)

DevOps:

- CI/CD: GitHub Actions or GitLab CI
- Version Control: Git (GitHub/GitLab)
- Containerization: Docker
- Infrastructure as Code: Terraform (optional)

8.3 Database Schema Overview

Core Entities:

- Users (id, email, username, password_hash, role, profile_data, created_at)
- Courses (id, title, description, difficulty, author_id, status, created_at)
- Topics (id, course_id, title, content, order, created_at)
- Problems (id, topic_id, title, statement, difficulty, constraints, test_cases, created_at)
- Submissions (id, user_id, problem_id, code, language, status, score, execution_time, created_at)
- Challenges (id, title, type, start_time, end_time, problems[], scoring_rules, created_at)
- ChallengeParticipations (id, challenge_id, user_id, score, rank, submissions_count)
- Discussions (id, problem_id, user_id, title, content, upvotes, created_at)
- Comments (id, discussion_id, user_id, content, parent_comment_id, created_at)
- UserProgress (id, user_id, course_id, topic_id, completion_percentage, last_accessed)
- Achievements (id, user_id, badge_type, earned_at)

8.4 Security Requirements

Application Security:

- HTTPS/TLS encryption for all communications
- Input validation and sanitization (prevent XSS, SQL injection)
- CSRF protection for form submissions
- Rate limiting on API endpoints
- SQL injection prevention (parameterized queries/ORM)
- Secure password storage (bcrypt hashing)
- Content Security Policy (CSP) headers

Code Execution Security:

- Sandboxed containers for code execution
- Network isolation (no outbound connections from execution containers)
- Resource limits (CPU, memory, disk I/O)
- Timeout enforcement
- Prohibited system calls blocked
- File system restrictions (read-only access)
- Regular security audits of execution environment

Data Privacy:

- GDPR compliance (for EU users)
- Data encryption at rest
- Personal data anonymization in analytics
- User data export capability
- Account deletion functionality
- Privacy policy and terms of service

User Interface & Experience

9.1 Key User Flows

New User Onboarding:

1. Land on homepage → View featured courses
2. Sign up with email or social auth
3. Complete profile setup (skills, interests)
4. Browse course catalog
5. Enroll in first course
6. Guided tutorial for code editor and submission
7. Solve first practice problem
8. Earn "First Problem Solved" badge

Practice Problem Solving:

1. Navigate to course → Select topic
2. View list of practice problems
3. Click on a problem
4. Read problem statement and examples
5. Select programming language
6. Write code in editor
7. Run code against sample test cases
8. Debug if needed
9. Submit final solution
10. View results and feedback
11. Access discussion forum or solutions

Challenge Participation:

1. Navigate to Challenges section
2. View active/upcoming challenges
3. Register for a challenge
4. Wait for challenge to start (receive notification)
5. Start solving problems
6. Submit solutions
7. View real-time leaderboard
8. Complete challenge before deadline
9. View final rank and score
10. Share achievement on profile/social media

9.2 UI Components & Pages

Public Pages:

- Homepage (hero, featured courses, statistics, testimonials)
- Course catalog (filters, search, sorting)
- Course detail page (syllabus, topics, enrollment CTA)
- About Us
- Pricing (if freemium model)
- Login/Signup pages

Authenticated User Pages:

- User Dashboard (progress overview, active courses, challenges)
- Course learning page (topic content, problems list)
- Problem solving page (statement, editor, submission panel)
- Challenges page (active, upcoming, past challenges)
- Challenge leaderboard
- Discussion forum (all threads, create new)
- Individual discussion thread page
- Solutions page (filtered by problem)
- User profile (personal stats, badges, activity)
- Settings (account, preferences, notifications)

Admin Pages:

- Admin dashboard (platform analytics)
- Course management (CRUD operations)
- Topic management (content editor)
- Problem management (problem editor, test case creator)
- Challenge management (create, schedule, monitor)
- User management (view users, roles, moderation)
- Discussion moderation
- Analytics and reports

9.3 Responsive Design Requirements

- Mobile-first approach
- Breakpoints: Mobile (<640px), Tablet (640-1024px), Desktop (>1024px)
- Touch-friendly UI elements for mobile
- Responsive code editor (consider mobile-friendly alternatives)
- Hamburger menu for mobile navigation
- Optimized images and lazy loading
- Progressive Web App (PWA) capabilities (future enhancement)

Non-Functional Requirements

10.1 Performance

- Page load time: <2 seconds on 3G connection
- Code execution: Results within 5 seconds for most problems
- API response time: <500ms for 95% of requests
- Support for 10,000+ concurrent users
- Database query optimization (indexes, caching)
- CDN for static assets
- Lazy loading for images and components
- Code splitting and bundle optimization

10.2 Scalability

- Horizontal scaling capability for all services
- Load balancing across multiple servers
- Auto-scaling based on traffic patterns
- Queue-based async processing for heavy operations
- Database read replicas for high-traffic queries
- Microservices architecture for independent scaling

10.3 Reliability & Availability

- 99.9% uptime SLA
- Automated backups (daily full, hourly incremental)
- Disaster recovery plan
- Database replication (master-slave)
- Health checks and automated failover
- Error tracking and monitoring (Sentry, Rollbar)
- Graceful degradation (fallbacks for service failures)

10.4 Accessibility

- WCAG 2.1 Level AA compliance
- Keyboard navigation support
- Screen reader compatibility
- Sufficient color contrast ratios
- Alt text for images
- ARIA labels for interactive elements
- Skip navigation links
- Resizable text without layout breaking

10.5 Internationalization (Future)

- Multi-language support (English, Hindi, Spanish, etc.)
- RTL layout support (Arabic, Hebrew)
- Localized content and problem statements
- Currency and date format localization
- Timezone handling for challenges

Monetization Strategy

11.1 Revenue Models

Freemium Model:

- Free tier: Access to all courses and practice problems (with ads)
- Premium tier ($9.99/month or $99/year):
  - Ad-free experience
  - Unlimited solution access
  - Premium challenges and contests
  - Priority support
  - Certificates of completion
  - Offline access (downloadable content)

B2B/Enterprise:

- Company accounts for technical hiring
- Custom assessment creation
- Candidate evaluation reports
- Interview preparation packages
- Bulk licenses for educational institutions

Affiliate & Partnerships:

- Job board integrations (earn commission on placements)
- Sponsored challenges by tech companies
- Affiliate links for recommended courses/books
- Certification partnerships

Launch Strategy & Roadmap

12.1 MVP (Minimum Viable Product) - Phase 1 (3-4 months)

Core Features:

- User authentication (email/password only)
- Basic course catalog (5-10 starter courses)
- Topic-based content and practice problems (50-100 problems)
- Integrated code editor with execution (Python, Java, C++)
- Problem submission and evaluation
- Basic user dashboard and progress tracking
- Admin panel for course/problem management

Success Metrics:

- 500+ registered users
- 1,000+ problem submissions
- 60% user retention after first week

12.2 Phase 2 - Enhanced Features (2-3 months post-MVP)

New Features:

- Challenge system (Daily and Weekly challenges)
- Discussion forum and community features
- Solution sharing (locked/unlockable)
- Advanced progress tracking and badges
- Leaderboards and rankings
- Social authentication (Google, GitHub)
- Email notifications
- Additional programming languages (JavaScript, C)

Success Metrics:

- 5,000+ registered users
- 500+ daily active users
- 10,000+ monthly problem submissions
- 50+ discussions created weekly

12.3 Phase 3 - Monetization & Scale (3-4 months post-Phase 2)

New Features:

- Premium subscription tier
- Premium-only challenges and contests
- Certificates of completion
- Skills profile/portfolio (shareable)
- Advanced analytics for users
- Mobile app (iOS/Android)
- API for third-party integrations
- Improved code editor (more languages, themes)
- AI-powered hints and explanations

Success Metrics:

- 20,000+ registered users
- 5% conversion to premium
- $10,000+ monthly recurring revenue
- 80% user satisfaction rating

12.4 Phase 4 - Advanced Features (Ongoing)

- Live coding interviews (peer-to-peer or with mentors)
- Video courses and tutorials
- Company-specific interview prep packages
- Hackathon hosting capabilities
- Job board integration
- AI-powered problem recommendations
- Code review and mentorship features
- Advanced gamification (tournaments, team challenges)

Success Metrics & KPIs

13.1 User Engagement

- Daily Active Users (DAU)
- Monthly Active Users (MAU)
- DAU/MAU ratio (stickiness)
- Average session duration
- Problems solved per user per week
- Course completion rate
- Challenge participation rate
- Discussion forum activity

13.2 Platform Health

- User registration growth rate
- User retention (Day 1, Day 7, Day 30)
- Churn rate
- Average time to first submission
- Code execution success rate
- Platform uptime percentage
- Average page load time
- API error rate

13.3 Content Metrics

- Total courses available
- Total problems available
- Problem solve rate (% of users solving each problem)
- Average difficulty vs completion correlation
- Discussion threads per problem
- Solution quality ratings

13.4 Business Metrics

- Premium conversion rate
- Monthly Recurring Revenue (MRR)
- Customer Acquisition Cost (CAC)
- Lifetime Value (LTV)
- Churn rate for premium users
- Net Promoter Score (NPS)

Risks & Mitigation

14.1 Technical Risks

Risk	Impact	Mitigation
Code execution security breach	High	Strict containerization, security audits, penetration testing
Platform downtime during challenges	High	Load testing, auto-scaling, redundancy, monitoring
Database performance degradation	Medium	Indexing, caching, read replicas, query optimization
Scalability issues with user growth	High	Microservices architecture, horizontal scaling, CDN

14.2 Business Risks

Risk	Impact	Mitigation
Low user adoption	High	Strong marketing, SEO, referral program, free tier
High user churn	Medium	Gamification, daily challenges, community building
Premium conversion too low	Medium	Clear value proposition, free trial, A/B testing
Competition from established platforms	High	Unique features, better UX, localized content, community focus

14.3 Content Risks

Risk	Impact	Mitigation
Plagiarized problems	Medium	Review process, plagiarism detection, original content focus
Poor quality user solutions	Low	Moderation, upvote/downvote system, featured solutions
Spam in discussions	Medium	Automated spam detection, moderation tools, reporting

Appendix

A. Glossary

- **Test Case**: Input-output pair used to validate code correctness
- **Submission**: User's code attempt for a problem
- **Editorial**: Official solution and explanation for a problem
- **AC (Accepted)**: Code passed all test cases
- **WA (Wrong Answer)**: Code failed one or more test cases
- **TLE (Time Limit Exceeded)**: Code exceeded time limit
- **MLE (Memory Limit Exceeded)**: Code exceeded memory limit
- **RE (Runtime Error)**: Code crashed during execution
- **Sandbox**: Isolated execution environment for running untrusted code
- **Leaderboard**: Ranked list of users based on scores
- **Streak**: Consecutive days of activity on the platform

B. References

1. GeeksforGeeks Platform Architecture[1]
2. HackerRank Features and Capabilities[2][5]
3. Educational Coding Platform Best Practices[3]
4. Software Architecture Patterns[4][7][10]
5. Real-World Assessment Design[8]

C. Open Questions

- Should we support pair programming or collaborative coding features?
- Should we integrate with external job boards immediately or later?
- What should be the free vs premium feature split?
- Should we allow user-generated content (problems, courses)?
- Should we implement a referral/affiliate program at launch?
- What level of analytics should be available to free users?

D. Future Considerations

- Machine learning-based problem recommendations
- Adaptive learning paths based on user performance
- Virtual coding assistants/AI tutors
- Integration with popular IDEs (VS Code extension)
- Blockchain-based skill verification and certificates
- VR/AR coding environments (experimental)
- Code collaboration and peer review features
- Live mentorship and 1-on-1 coaching marketplace
- Corporate training and B2B SaaS offerings

Document Approval:

Role	Name	Signature	Date
Product Owner	_____	_____	_____
Tech Lead	_____	_____	_____
Design Lead	_____	_____	_____
Stakeholder	_____	_____	_____

End of Document