Farcaster Mini-App Development Project
Detailed Specifications

Project Title: Farcaster Application Rapid Development Environment	**Acronym:** FARDE
Duration: 8 weeks	**Team Size:** 5 students
Time Commitment: 40 hours per student	**Difficulty Level:** Intermediate to Advanced

1. Executive Summary

This project engages students in developing production-ready mini-applications for the Farcaster protocol, a decentralized social network with 500K+ users. Through an 8-week intensive program, students will create reusable infrastructure, explore novel social mechanics, and launch a live application with real users. The project emphasizes hands-on learning through rotating leadership roles and pair programming, ensuring all participants gain full-stack Web3 development experience.

2. Team Structure & Required Skills

Team Composition

The ideal team consists of 5 students operating in a rotating leadership model:

- **Weekly Lead (1 student):** Rotates each week, manages stakeholder communication, ensures deliverables, coordinates pairs
- **Programming Pairs (4 students):** Two pairs of 2 students each, focusing on collaborative development

Required Baseline Skills

All students must demonstrate proficiency in:

- Basic HTML/CSS/JavaScript
- Version control with Git/GitHub
- Command line basics
- Static site deployment (Vercel/Netlify)
- Effective use of LLMs for development assistance

Skills to be Developed

- React and TypeScript

- Farcaster protocol and mini-app constraints
- API integration and database management
- Web3 fundamentals
- Pair programming and agile methodology
- Project leadership and stakeholder communication

3. Project Overview & Problem Statement

Problem Statement

The Farcaster ecosystem lacks standardized development tools and templates for new developers entering the Web3 social space. Current barriers include complex authentication flows, no standardized UI components for constrained environments, limited examples of social graph utilization, and high friction for rapid prototyping.

Project Objectives

1. Create a production-ready mini-application demonstrating Farcaster capabilities
2. Develop reusable infrastructure components for future developers
3. Explore novel social mechanics using open protocol data
4. Establish best practices for constrained UI development
5. Generate open-source tools that accelerate ecosystem growth
6. Ensure all students gain full-stack Web3 development experience

Target Users

- **Primary:** Farcaster users seeking enhanced social experiences
- **Secondary:** Developers building on Farcaster protocol
- **Tertiary:** Web3 communities exploring decentralized platforms

4. Solution Approach & Core Components

Layer 1: Infrastructure Components

- **Authentication Module:** Plug-and-play Farcaster Quick Auth implementation
- **UI Component Library:** Pre-built responsive components for mini-app constraints
- **Data Access Layer:** Abstraction for Farcaster protocol interactions

Layer 2: Application Framework

- **Mini-App Scaffold:** Base template with routing and state management
- **Social Graph Engine:** Tools for analyzing network relationships
- **Engagement Analytics:** Real-time metrics tracking and reporting

Layer 3: User Applications (Choose One)

- **Option A:** Social Discovery - Algorithm-driven recommendations

- **Option B:** Gamification - Achievement system with on-chain actions
- **Option C:** Creative Tools - Content creation utilities
- **Option D:** Analytics Dashboard - Insights visualization

5. Technical Specifications

Technology Stack

Frontend

- React 18+ with TypeScript
- Next.js 14 for SSR/SSG
- Tailwind CSS for styling
- Zustand/Redux for state management

Backend

- Node.js with Express/Fastify
- Serverless functions (Vercel/AWS Lambda)
- PostgreSQL with Prisma ORM
- Redis for caching

Web3/Blockchain

- Farcaster Hub API
- Base L2 for on-chain interactions
- Viem for wallet connections

Technical Constraints

- Mini-app iframe dimensions: ~424x695px
- 10MB bundle size limit
- CORS restrictions for cross-origin requests
- Mobile-first design requirements

Testing Methodology

- Unit testing with Jest/Vitest (80% coverage target)
- Integration testing for API endpoints
- User acceptance testing with 20+ beta testers
- Performance testing (Core Web Vitals)
- Security audit for authentication flows

6. Implementation Timeline

> **Week 1: Technology Evaluation & Onboarding**
>
> **Objectives:** Assess baseline skills and establish foundation
>
> **Activities**
>
> - Setup communication channels (Discord/Slack, GitHub)
> - Create Farcaster accounts and follow team members
> - Explore and test 5+ existing mini apps
> - Build and deploy individual static sites with Farcaster profile links
> - Brainstorm 50+ mini app ideas
> - Establish LLM workflows for development
>
> **Week 1 Deliverables**
>
> - 5 individual static sites deployed
> - Team communication channels active
> - Skills assessment matrix completed
> - 50+ mini app ideas documented

Week 2: Mini App Foundation & Project Selection

Activities

- Convert static sites into Farcaster mini apps
- Implement frame metadata and test constraints
- Team workshop to evaluate all ideas
- Technical feasibility assessment
- Final project selection and scope definition

Week 2 Deliverables

- 5 functioning mini apps (individual)
- Project proposal document
- Technical architecture plan
- Rotation schedule finalized

Weeks 3-7: Iterative Development with Rotating Leadership

Week 3: Core Infrastructure (Lead: Student A)

- Pair A: Authentication system & database setup
- Pair B: UI component library & mini app scaffold
- Deliverable: Working authentication, 5+ base components

Week 4: MVP Development (Lead: Student B)

- Pair A: Core feature implementation
- Pair B: API endpoints & data flow
- Deliverable: MVP with 2-3 core features

Week 5: Beta Release (Lead: Student C)

- Pair A: User interface refinement
- Pair B: Testing framework & bug fixes
- Deliverable: Beta version with 20+ testers

Week 6: Launch & Iteration (Lead: Student D)

- Pair A: Performance optimization
- Pair B: Feature enhancements
- **Critical Milestone: Production launch by Wednesday**
- Deliverable: Live app with 50+ users

Week 7: User Feedback & Polish (Lead: Student E)

- Pair A: Critical bug fixes from user reports
- Pair B: Analytics implementation & documentation
- Deliverable: 100+ users, complete documentation

Week 8: Presentation & Final Iteration

Monday-Tuesday

- Address critical issues
- Final feature polish
- Documentation completion

Wednesday-Thursday

- Create presentation deck
- Record demo videos
- Practice run-through

Friday

- Final team presentation
- Live product demonstration
- Metrics and impact review
- Community showcase

Final Deliverables

- Live product with 200+ users
- Complete open-source repository
- Comprehensive documentation
- Component library and templates
- 15-20 minute presentation
- Individual reflection documents

7. Success Metrics

Weekly Milestones

- **Week 1:** 100% of students deploy static sites
- **Week 2:** 5 mini apps deployed, project selected
- **Week 3:** Authentication system functional
- **Week 4:** MVP with core features complete
- **Week 5:** Beta release with 20+ testers
- **Week 6:** Production launch achieved
- **Week 7:** 100+ active users onboarded
- **Week 8:** 200+ total users, presentation delivered

Technical Metrics

- 99%+ uptime post-deployment
- <3 second load time
- Zero critical security issues
- 80% code coverage on critical paths
- <5% error rate in production

Learning Outcomes

Each student will be able to:

- Deploy a Farcaster mini app independently
- Explain the entire system architecture
- Demonstrate pair programming proficiency
- Navigate Farcaster protocol documentation
- Use LLMs effectively for development

8. Innovation & Competitive Advantage

Innovative Aspects

- First comprehensive UI kit for Farcaster's constraints
- Novel interaction patterns for mini-app environment
- Direct access to entire social graph without API limitations

- Modular components allowing rapid experimentation
- Token-gated experiences without wallet popups
- Creator rewards integration for sustainable monetization

Competitive Advantages

- No platform lock-in
- Transparent algorithms
- User-owned social graph
- Superior developer experience
- Existing active user base
- Mobile-first approach

9. Deployment & Support Requirements

Infrastructure Requirements

- Production hosting on Vercel (free tier)
- PostgreSQL database (Supabase/Neon)
- Redis cache (Upstash)
- Domain and SSL certificate
- Development, staging, and production environments

Documentation Requirements

- README with setup instructions
- API documentation (OpenAPI/Swagger)
- Component storybook
- Architecture decision records
- Troubleshooting guide
- Video walkthroughs

10. Intellectual Property & Confidentiality

Open Source Commitment

- All infrastructure code open-sourced under MIT License
- Templates and libraries freely available
- Documentation published publicly

Student Rights

- Students retain ownership of specific implementations
- Right to continue development post-project
- Attribution in all derived works
- Portfolio and resume usage permitted

Publication Rights

- Students may publish about their work
- Case studies and metrics can be shared

- Technical innovations can be presented at conferences

11. Mentorship & Communication Plan

Week 1: Intensive Onboarding

- Daily 30-minute check-ins
- Technical setup assistance
- Farcaster platform guidance
- LLM best practices training

Week 2: Project Selection Support

- Mini app conversion workshop (2 hours)
- Idea evaluation session (1 hour)
- Project selection facilitation (2 hours)

Weeks 3-7: Development Support

- Monday: Kickoff with new lead (1 hour)
- Wednesday: Mid-week check-in (30 minutes)
- Friday: Week wrap-up and handoff prep (1 hour)
- On-demand code reviews (4-hour response)
- Weekly office hours (2 hours)

Week 8: Presentation Support

- Final bug fix assistance
- Presentation review and feedback
- Attendance at final presentation

Resources Provided

- Farcaster API documentation
- Example mini app repositories
- Authentication boilerplate code
- UI component library starter
- Development Farcaster accounts
- Analytics dashboard access
- Base testnet tokens (if needed)

12. Risk Mitigation

Critical Risk Indicators

- Student unable to deploy static site by Day 4
- No consensus on project by end of Week 2
- Lead handoff issues between weeks
- Production launch delayed past Week 6
- Less than 20 users by Week 7
- Team communication breakdown

Mitigation Strategies

- Early skill assessment and support
- Structured project selection process
- Clear handoff documentation templates
- MVP scope adjustment if needed
- Active community engagement plan
- Regular team retrospectives