YieldMax: 15-Day Sprint Plan

Chainlink Hackathon - Zero to Hero Strategy

TEAM STRUCTURE & AI LEVERAGE

You: Backend/Integration + AI Strategy **Nikita:** Smart Contracts + Frontend
AI Tools: Code generation, debugging, documentation

DAILY SCHEDULE:

- 9 AM: 30-min standup
- 10 AM - 6 PM: Deep work blocks
- 6 PM: Daily demo to each other
- 7 PM: Plan next day

PHASE 1: FOUNDATION (Days 1-5)

Day 1: CHAINLINK CRASH COURSE + SETUP

YOU (Backend Lead):

- [ ] Complete Chainlink documentation speed-run (4 hours max)
- [ ] Set up development environment with all Chainlink testnet configs
- [ ] Clone and study 3 existing CCIP examples from Chainlink GitHub
- [ ] Set up Claude/Cursor/GitHub Copilot for maximum AI assistance

NIKITA (Contract Lead):

- [ ] Review basic DeFi vault patterns (Yearn, Compound examples)
- [ ] Set up Hardhat/Foundry with multi-chain deployment
- [ ] Create basic ERC4626 vault structure
- [ ] Study existing yield aggregator contracts

AI PROMPT TEMPLATES TO SET UP:

```
"You are a Chainlink expert helping build a cross-chain yield optimizer.
Always provide working code examples with proper error handling.
Focus on testnet implementations first."
```

END OF DAY 1 MILESTONE: Both devs can deploy a basic contract to 2 testnets

Days 2-3: CORE CCIP INTEGRATION

PARALLEL DEVELOPMENT:

YOU: CCIP Cross-Chain Logic

- [ ] Implement basic CCIP sender/receiver contracts
- [ ] Test cross-chain messaging between Sepolia ↔ Polygon Mumbai
- [ ] Build cross-chain token transfer functionality
- [ ] Create gas estimation and fee handling

NIKITA: Vault Core Logic

- [ ] ERC4626-compliant vault with deposit/withdraw
- [ ] Multi-asset support (USDC, USDT, DAI)
- [ ] Basic access control and security features
- [ ] Integration points for cross-chain calls

CRITICAL SUCCESS METRIC: End of Day 3 = Cross-chain message successfully sent

Days 4-5: DATA STREAMS + AUTOMATION BASICS

YOU: Data Integration

- [ ] Chainlink Data Streams integration for ETH/USD, USDC rates
- [ ] Build yield data aggregation from Aave, Compound APIs
- [ ] Create simple yield comparison logic
- [ ] Implement basic automation trigger conditions

NIKITA: Strategy Logic

- [ ] Simple rebalancing algorithm (move to highest yield)
- [ ] Slippage protection and minimum thresholds
- [ ] Integration with your data feeds
- [ ] Basic liquidation protection

END OF PHASE 1: Working cross-chain vault that can move funds and read yield data

---

PHASE 2: OPTIMIZATION ENGINE (Days 6-10)

Days 6-7: AI OPTIMIZATION (SIMPLIFIED)

STRATEGY PIVOT: Instead of complex ML, build rule-based "AI" that looks smart

YOU: Smart Rebalancing Engine

- [ ] Multi-factor scoring: Yield + Risk + Liquidity + Gas costs
- [ ] Historical performance tracking
- [ ] Predictive rebalancing (if yield gap > X%, switch in Y hours)
- [ ] Risk assessment scoring for each protocol

NIKITA: Chainlink Functions Integration

- [ ] Off-chain computation for complex yield calculations
- [ ] Integration with your optimization engine
- [ ] Error handling and fallback mechanisms
- [ ] Response validation and security

OPTIMIZATION ALGORITHM (Keep It Simple):

```python
def optimize_yield(protocols, user_balance, risk_tolerance):
    scores = []
    for protocol in protocols:
        score = (
            protocol.yield_rate * 0.4 +
            protocol.liquidity_score * 0.2 +
            (1 - protocol.risk_score) * 0.3 +
            protocol.gas_efficiency * 0.1
        )
        scores.append(score)
    return best_protocol_index
```

Days 8-9: AUTOMATION PERFECTION

YOU: Chainlink Automation Setup

- [ ] Time-based triggers (daily optimization)
- [ ] Performance-based triggers (yield differential > 2%)
- [ ] Custom logic triggers (risk score changes)
- [ ] Gas optimization for automation calls

NIKITA: Security & Testing

- [ ] Comprehensive test suite for all functions
- [ ] Reentrancy protection verification
- [ ] Oracle manipulation protection

- [ ] Emergency pause and recovery mechanisms

Day 10: INTEGRATION & BUG FIXES

BOTH: Full Integration Testing

- [ ] End-to-end user journey testing
- [ ] Cross-chain failure recovery testing
- [ ] Edge case handling verification
- [ ] Performance optimization

PHASE 3: FRONTEND & POLISH (Days 11-15)

Days 11-12: PROFESSIONAL FRONTEND

NIKITA: Frontend Development (Use AI heavily here)

- [ ] React dashboard with real-time data
- [ ] Wallet connection (MetaMask, WalletConnect)
- [ ] Transaction status tracking
- [ ] Yield comparison tables

YOU: Backend APIs & Integration

- [ ] REST APIs for frontend data
- [ ] WebSocket connections for real-time updates
- [ ] Cross-chain transaction tracking
- [ ] Performance analytics backend

AI PROMPT FOR FRONTEND:

```
"Create a professional DeFi dashboard component that shows:
- Real-time yield rates across protocols
- User portfolio allocation
- Transaction history with status
- APY projections and earnings
Use Tailwind CSS, make it look like Aave or Compound"
```

Days 13-14: DEMO PREPARATION

PARALLEL TASKS:

YOU: Demo Content Creation

- [ ] 5-minute demo video script
- [ ] Live demo environment setup
- [ ] Backup demo scenarios (in case live fails)
- [ ] Performance metrics compilation

NIKITA: Documentation & Pitch

- [ ] Technical documentation
- [ ] Architecture diagrams
- [ ] Pitch deck for judges
- [ ] ROI calculations and projections

Day 15: FINAL DEPLOYMENT & SUBMISSION

BOTH: Final Push

- [ ] Mainnet/testnet deployment verification
- [ ] All Chainlink integrations tested and documented
- [ ] Video demo recorded and edited
- [ ] Submission package completed

CHAINLINK SERVICES INTEGRATION CHECKLIST

MUST-HAVE (For Eligibility):

- **CCIP:** ✅ Cross-chain fund movement
- **Data Streams:** ✅ Real-time yield data
- **Automation:** ✅ Automated rebalancing

BONUS POINTS:

- **Functions:** ✅ Off-chain optimization computation
- **Price Feeds:** ✅ Additional price data validation

TARGET: 4 Services = Maximum Scoring

AI DEVELOPER TOOLS STACK

PRIMARY TOOLS:

1. **Claude.ai (You have this)** - Architecture decisions, complex problem solving
2. **GitHub Copilot** - Code completion and function generation

3. **Cursor IDE** - AI-powered development environment
4. **v0.dev by Vercel** - Frontend component generation

AI PROMPTING STRATEGY:

For Complex Logic:

```
"I'm building a cross-chain yield optimizer for Chainlink hackathon.
Current context: [paste relevant code]
Problem: [specific issue]
Requirements: [constraints]
Please provide working Solidity code with proper error handling."
```

For Frontend:

```
"Create a React component for DeFi yield optimization dashboard.
Features needed: [list features]
Styling: Tailwind CSS, dark theme, professional look
Make it look like established DeFi protocols."
```

COMPETITIVE ADVANTAGE STRATEGY

WHAT OTHERS WILL BUILD:

- Basic yield aggregators with manual rebalancing
- Single-chain solutions with limited Chainlink integration
- Complex but broken cross-chain features
- Poor user experience and documentation

WHAT YOU'RE BUILDING:

- **Automated** cross-chain optimization
- **Risk-adjusted** returns, not just highest APY
- **Professional** frontend that actually works
- **Complete** Chainlink services integration

SECRET WEAPONS:

1. **AI-Assisted Development:** You'll code 3x faster than manual teams
2. **Focused Scope:** Simple but perfect vs complex but broken

3. **Professional Polish:** Most hackathon projects look amateur
4. **Real Business Model:** Clear path to revenue and growth

DAILY PROGRESS TRACKING

WEEK 1 MILESTONES:

- Day 3: Cross-chain message working
- Day 5: Basic yield optimization working

WEEK 2 MILESTONES:

- Day 8: Full automation working
- Day 10: End-to-end testing complete
- Day 12: Professional frontend deployed
- Day 15: Submission complete

RISK MITIGATION:

- **If Behind by Day 3:** Cut AI features, focus on basic automation
- **If Backend Issues:** Nikita helps with integration, you focus on demo
- **If Frontend Delays:** Use AI tools aggressively, simplify UI
- **If Integration Fails:** Have backup single-chain version ready

---

WINNING CRITERIA ALIGNMENT

TECHNICAL EXCELLENCE (40%):

- ✅ Multiple Chainlink services working perfectly
- ✅ Clean, well-tested smart contract code
- ✅ Professional architecture and documentation

INNOVATION (30%):

- ✅ AI-powered optimization (even if simple)
- ✅ Cross-chain automation
- ✅ Risk-adjusted yield optimization

Business Potential (20%):

- ✅ Clear revenue model (2% performance fees)
- ✅ Large market opportunity ($100B+ DeFi)
- ✅ Scalable technical architecture

Presentation (10%):

- ✅ Professional demo and pitch
- ✅ Working live demonstration
- ✅ Clear value proposition

SUCCESS METRICS

TECHNICAL BENCHMARKS:

- **Cross-chain transaction:** <60 seconds completion
- **Gas efficiency:** <$25 average rebalancing cost
- **Uptime:** >99% during demo period
- **Yield advantage:** >1.5% APY improvement demonstrable

DEMO REQUIREMENTS:

- **Live transaction:** Show real cross-chain optimization
- **Performance proof:** Historical data showing yield improvement
- **User experience:** 3-click optimization from deposit to active
- **Professional presentation:** 5-minute pitch that impresses judges

---

BOTTOM LINE: 15 days is tight but doable. Your advantage is focus + AI tools + parallel development. While others waste time on fancy features, you're building something that works perfectly.

QUESTIONS:

1. Are you both available full-time for these 15 days?
2. Do you have testnet ETH/tokens for all the testing?
3. Which AI coding tools do you currently have access to?

NEXT STEP: Start Day 1 immediately. Set up your development environments and dive into Chainlink docs. The team that moves fastest in Week 1 usually wins the hackathon.