**IntelliDART: End-to-End Project Proposal & Build Guide**

**1. Executive Summary**

**Product Name:** IntelliTeach / DeepTeach  
**Overview:** An online tutor marketplace for STEM students, providing multi-tutor, multi-topic, and multi-fidelity on-demand tutoring, personalized pedagogy, and comprehensive GenAI educational reports.  
**Target Audience:** STEM students (Grades 9-12), tutors, and parents.  
**Purpose:** To bridge skill gaps in STEM, support AP/college prep, and offer hyper-personalized learning and career guidance.

**2. Product Vision and Strategy**

**Vision Statement:**  
Enable dynamic micro-marketplaces for education, creating a personalized "eduverse" tailored to each student's aptitude and aspirations.

**Strategic Goals:**

* Phase 1: Tutor/student onboarding, agent-based matching
* Phase 2: STEM topic complexity analysis, dashboards
* Phase 3: Hyper-personalized career guidance and pedagogy plans

**Target Market:**

* US-based STEM students (Grades 9-12)
* Curated tutor ecosystem

**Value Proposition:**

* Hyper-personalized learning and career guidance
* Marketplace and personalization combined

**Competitive Edge:**

* Addresses unmet needs for tailored career guidance
* Combines marketplace with adaptive learning

**3. Requirements Gathering**

**Functional Requirements**

* Tutor profiles: expertise, keywords, sample videos
* Student profiles: requirements, editable preferences
* Matching panel: personalized tutor/topic recommendations
* Personalized pedagogy: AI-driven knowledge graph and learning plan
* Career stream workflow: milestone tracking, assessments, recommendations

**Non-Functional Requirements**

* Security: Cyber-physical standards, role-based access
* Scalability: Initial support for 1,000+ students
* GenAI reporting, asynchronous quizzes, and career briefs
* Separate logins for students, tutors, and admins

**User Stories**

* Student seeks career exploration and capstone projects
* Tutor bids for students based on expertise and analytics
* AI agent creates mentor-protégé plans for optimal matching

**4. System Architecture and Design**

**High-Level Architecture**

* **Frontend:** Web app (React/Angular/Vue)
* **Backend:** REST API (Node.js/Express, Python/FastAPI, or similar)
* **Database:** PostgreSQL or MongoDB
* **AI/ML:** Python microservices for GenAI and analytics
* **Cloud:** AWS/Azure/GCP for hosting, storage, and scaling

**Data Flow**

1. Student/tutor registration and profile creation
2. Student submits learning/career goals
3. AI agent matches students with tutors and generates plans
4. Sessions, quizzes, and reports managed via backend
5. Data stored securely in the database

**Database Design**

* Users (students, tutors, admins)
* Tutor profiles
* Student profiles
* Knowledge graphs
* Session logs
* Reports and analytics

**Integration Points**

* Video conferencing APIs (Zoom, WebRTC)
* Payment gateways (Stripe, PayPal)
* GenAI/LLM APIs (OpenAI, Azure OpenAI)

**Technology Stack**

* Frontend: React.js (recommended)
* Backend: Node.js/Express or Python/FastAPI
* Database: PostgreSQL (relational) or MongoDB (NoSQL)
* AI/ML: Python, TensorFlow/PyTorch, OpenAI API
* Cloud: AWS (EC2, S3, RDS), Docker, Kubernetes

**UI/UX Design**

* Intuitive dashboards for students and tutors
* Responsive design for web/mobile
* Accessibility and user-centric flows

**5. Development Process**

* **Methodology:** Agile/Scrum with 2-week sprints
* **Milestones:**
  + MVP: User onboarding, basic matching, session booking
  + Phase 2: AI-driven recommendations, reporting
  + Phase 3: Career guidance, advanced analytics
* **Version Control:** Git (feature branching, PR reviews)
* **Code Quality:** Linting, code reviews, unit/integration tests
* **CI/CD:** GitHub Actions, Jenkins, or GitLab CI for automated testing and deployment
* **Developer Docs:** Inline code comments, API docs (Swagger/OpenAPI), onboarding guides

**6. Security and Compliance**

* **Security:**
  + HTTPS, JWT/OAuth authentication
  + Role-based access control
  + Data encryption at rest and in transit
* **Compliance:**
  + GDPR (data privacy)
  + COPPA (for minors)
* **Risk Assessment:**
  + Regular security audits
  + Backup and disaster recovery plans

**7. Testing and Quality Assurance**

* **Test Plan:**
  + Unit, integration, and end-to-end tests
  + Performance/load testing
* **Test Cases:**
  + User registration/login
  + Tutor-student matching
  + Session booking and reporting
* **Performance Testing:**
  + Load/stress tests for 1,000+ users
* **Bug Tracking:**
  + Jira, GitHub Issues, or similar

**8. Deployment and Release Management**

* **Deployment Strategy:**
  + Staged rollout (dev, staging, production)
  + Blue-green deployment for zero downtime
* **Release Plan:**
  + Versioned releases, changelogs
* **Environment Setup:**
  + Dockerized services, environment variables for secrets
* **Rollback Plan:**
  + Automated rollback scripts, database backups

**9. Guide to Build and Extend**

**Step-by-Step Build Guide**

1. **Set up version control:**
   * Initialize Git repo, set up branching strategy
2. **Frontend setup:**
   * Scaffold React app, set up routing and state management
3. **Backend setup:**
   * Scaffold Node.js/Express or FastAPI app, connect to DB
4. **Database schema:**
   * Design and migrate initial schema (users, sessions, reports)
5. **AI/ML integration:**
   * Set up Python microservice for GenAI features
6. **APIs and integration:**
   * Implement REST APIs, integrate video/payment/AI services
7. **Testing:**
   * Write and run unit/integration tests
8. **CI/CD:**
   * Set up pipelines for build, test, deploy
9. **Deployment:**
   * Deploy to cloud (AWS/Azure/GCP), configure DNS/SSL
10. **Monitoring:**
    * Set up logging, monitoring, and alerting

**Extending the Platform**

* Add support for more subjects beyond STEM
* Integrate additional AI/LLM providers
* Expand to mobile apps (React Native, Flutter)
* Add gamification, badges, and community features

**10. Example User Flows**

**Student Flow**

1. Registers and creates a profile (interests, grade, goals)
2. Browses or searches for tutors by subject/topic
3. Receives AI-driven tutor and topic recommendations
4. Books a session and interacts via integrated video platform
5. Receives personalized learning plan and progress reports
6. Takes quizzes and receives GenAI-generated feedback
7. Tracks career milestones and receives guidance

**Tutor Flow**

1. Registers and creates a profile (expertise, credentials, sample content)
2. Reviews student requests and bids for suitable students
3. Matches with students via AI agent
4. Conducts sessions, uploads resources, and tracks student progress
5. Receives feedback and analytics on teaching effectiveness

**11. Sample Database Schema**

**Users Table/Collection**

* id (PK)
* name
* email
* role (student/tutor/admin)
* profile\_data (JSON)

**Tutors Table/Collection**

* id (PK)
* user\_id (FK)
* expertise
* keywords
* sample\_videos

**Students Table/Collection**

* id (PK)
* user\_id (FK)
* interests
* goals
* knowledge\_graph (JSON)

**Sessions Table/Collection**

* id (PK)
* student\_id (FK)
* tutor\_id (FK)
* scheduled\_time
* status
* session\_notes

**Reports Table/Collection**

* id (PK)
* student\_id (FK)
* report\_type
* content (JSON)
* created\_at

**12. Sample API Endpoints**

* POST /api/register — Register new user
* POST /api/login — Authenticate user
* GET /api/tutors — List/search tutors
* GET /api/students/{id} — Get student profile
* POST /api/sessions — Book a session
* GET /api/sessions/{id} — Get session details
* POST /api/reports — Generate/report progress
* GET /api/ai/recommendations — Get AI-driven tutor/topic matches

**13. Architecture Diagram (Text Description)**

* **Frontend (React):**
  + User dashboards, booking, messaging, reports
  + Communicates with backend via REST API
* **Backend (Node.js/Express or FastAPI):**
  + Handles authentication, business logic, session management
  + Connects to database and AI microservices
* **Database (PostgreSQL/MongoDB):**
  + Stores users, sessions, reports, knowledge graphs
* **AI/ML Microservice (Python):**
  + Provides recommendations, generates reports, analyzes progress
* **Integrations:**
  + Video API, payment gateway, GenAI/LLM API
* **Cloud Infrastructure:**
  + Dockerized services, CI/CD, monitoring, backups

**14. Expanded Build Guide (with References)**

1. **Version Control:**
   * Use GitHub/GitLab, set up main/dev branches, enable PR reviews
2. **Frontend:**
   * Use Create React App or Vite, set up React Router, Redux/Context API
   * Install UI library (e.g., Material-UI)
3. **Backend:**
   * Scaffold with Express Generator or FastAPI CLI
   * Set up RESTful routes, JWT authentication
4. **Database:**
   * Use Sequelize (Postgres) or Mongoose (MongoDB) for ORM/ODM
   * Write migration scripts for schema
5. **AI/ML Integration:**
   * Create Python Flask/FastAPI microservice
   * Use OpenAI API for GenAI features
   * Connect via REST or gRPC
6. **APIs & Integrations:**
   * Integrate Zoom/WebRTC SDK, Stripe/PayPal SDK
   * Document APIs with Swagger/OpenAPI
7. **Testing:**
   * Use Jest (frontend), Mocha/Chai (backend), Pytest (AI)
   * Set up test coverage reports
8. **CI/CD:**
   * Use GitHub Actions or GitLab CI for build/test/deploy
   * Lint, test, and deploy on merge to main
9. **Deployment:**
   * Use Docker Compose for local dev
   * Deploy to AWS (EC2/ECS), set up S3 for storage, RDS for DB
   * Configure SSL with Let's Encrypt
10. **Monitoring:**
    * Use Prometheus/Grafana for metrics, Sentry for error tracking