24h Hackathon: Work Split & Tech Stack

🎯 Recommended Tech Stack

Frontend

- **UI Builder**: Lovable.dev or v0.dev (AI-powered, faster than Bolt/Replit)
- **Framework**: React + TypeScript (what Lovable generates)
- **Styling**: Tailwind CSS (built-in)
- **State Management**: React Context or Zustand (simple for MVP)

Backend

- **Framework**: FastAPI (Python) - faster development than Flask/Django
- **Task Queue**: Celery + Redis (for async image generation)
- **Database**: PostgreSQL + SQLAlchemy (or Supabase for quick setup)
- **File Storage**: AWS S3 or Cloudinary (for avatars/panels)

APIs

- **LLM**: OpenAI GPT-4 (story generation)
- **Image Gen**: FLUX (via Replicate or Together AI API)
- **Alternative**: Stable Diffusion if FLUX is too expensive

👥 5-Member Work Split

Person 1: Backend Lead + API Integration

Focus: Core backend architecture & AI orchestration

Tasks:

- Set up FastAPI project structure
- Design API endpoints (see interface contracts below)
- Integrate OpenAI API for story generation
- Integrate FLUX/image generation API
- Implement image processing pipeline
- Handle file uploads and storage

Time Split:

- Hours 0-4: Setup + API skeleton
- Hours 4-12: OpenAI + FLUX integration
- Hours 12-18: Testing & refinement
- Hours 18-24: Bug fixes + demo prep

Person 2: Database & Avatar Management

Focus: Data models, database, avatar system

Tasks:

- Design database schema (classrooms, students, avatars, stories, panels)
- Set up PostgreSQL/Supabase
- Create CRUD operations for all entities
- Implement avatar generation logic
- Build avatar customization system
- File upload handling

Time Split:

- Hours 0-3: Database setup + schema
- Hours 3-10: CRUD operations + avatar logic
- Hours 10-18: Integration with backend
- Hours 18-24: Testing + data seeding for demo

Person 3: Frontend Lead - Teacher Dashboard

Focus: Main teacher interface

Tasks:

- Teacher onboarding flow
- Classroom creation/setup form
- Story generation interface (prompt input + 3 options)
- Story preview/review interface
- Correction/regeneration UI
- Story library/history view

- Export functionality (PDF download)

Time Split:

- Hours 0-2: Setup Lovable project + design system
- Hours 2-8: Teacher dashboard core flows
- Hours 8-16: Story generation & review UI
- Hours 16-24: Polish + integration testing

Person 4: Frontend - Student Avatar Creation

Focus: Student-facing interface

Tasks:

- Student signup/onboarding page
- Avatar creation wizard
  - Name input
  - Photo upload with preview
  - Interest tags input
  - Avatar preview
- Student story viewing interface (read generated comics)
- Responsive design for tablets

Time Split:

- Hours 0-2: Setup + routing
- Hours 2-10: Avatar creation flow
- Hours 10-18: Story viewer interface
- Hours 18-24: Mobile/tablet optimization

Person 5: DevOps + Integration + Demo Prep

Focus: Glue everything together + presentation

Tasks:

- Set up deployment (Vercel for frontend, Railway/Render for backend)
- Environment configuration

- API contract testing between frontend/backend
- Create seed data for demo
- Pre-generate 2-3 complete stories for demo
- Build demo script/presentation
- Handle CORS, authentication basics
- Monitor API usage/costs

Time Split:

- Hours 0-4: Deployment setup
- Hours 4-12: Integration support
- Hours 12-18: Demo data preparation
- Hours 18-24: Final testing + presentation prep

🔌 API Interface Contracts (Work Simultaneously)

Backend Endpoints (Person 1 implements)

```python
# Classroom Management
POST   /api/classrooms
GET    /api/classrooms/{id}
PUT    /api/classrooms/{id}

# Student & Avatar Management
POST   /api/classrooms/{id}/students
POST   /api/students/{id}/avatar
GET    /api/students/{id}/avatar

# Story Generation
POST   /api/stories/generate-options    # Returns 3 story concepts
POST   /api/stories/generate-panels     # Generate full comic
POST   /api/stories/{id}/regenerate     # Regenerate specific panels
GET    /api/stories/{id}
GET    /api/classrooms/{id}/stories

# Export
GET    /api/stories/{id}/export/pdf
```

Request/Response Formats

typescript

```
// POST /api/classrooms
{
  name: string;
  subject: string;
  gradeLevel: string;
  storyTheme: string;
  designStyle: string;
  duration: string;
  learningMaterials?: File[];
}

// POST /api/stories/generate-options
{
  classroomId: string;
  lessonPrompt: string;  // "Today we covered Newton's Laws..."
}
// Response:
{
  options: [
    { id: string; title: string; summary: string; },
    { id: string; title: string; summary: string; },
    { id: string; title: string; summary: string; }
  ]
}

// POST /api/stories/generate-panels
{
  classroomId: string;
  selectedOptionId: string;
}
// Response:
{
  storyId: string;
  status: "generating" | "completed";
  progress: number;  // 0-100
  panels: [
    { id: string; imageUrl: string; order: number; }
  ]
}

// POST /api/students/{id}/avatar
{
  name: string;
  photo?: File;
  interests: string;
}
```

📋 Parallel Work Strategy

Hours 0-2: Foundation

- Person 1: FastAPI skeleton + route stubs
- Person 2: Database setup + models
- Person 3: Lovable setup + teacher layout
- Person 4: Lovable setup + student layout
- Person 5: Deployment infrastructure

Sync Point at Hour 2: Confirm API contracts work (use mock data)

Hours 2-12: Core Development

- **Frontends use mock data** (Person 3 & 4)
- **Backend implements real logic** (Person 1 & 2)
- Person 5: Helps with integration issues

Sync Point at Hour 12: Integration testing begins

Hours 12-18: Integration

- Connect frontend to real backend
- Test full user flows
- Fix integration bugs

Sync Point at Hour 18: Feature freeze, demo prep only

Hours 18-24: Polish & Demo

- Generate demo stories
- Polish UI
- Practice presentation
- Prepare backup plans

🎬 MVP Feature Prioritization

Must Have (Demo Killers)

✅ Teacher creates classroom
✅ Students create avatars (3-5 demo students)
✅ Teacher inputs lesson prompt
✅ System shows 3 story options
✅ Generate ONE complete story (pre-generate for demo)
✅ View story in nice format
✅ Export to PDF

Nice to Have (If Time Permits)

- Story corrections/regeneration
- Multiple classrooms
- Story history/library
- Real-time progress tracking

Skip for MVP

- User authentication (use demo accounts)
- Email notifications
- Advanced avatar customization
- Mobile app
- Analytics

💰 Cost Management (Important!)

API Costs for Demo

- OpenAI GPT-4: ~$0.03 per story script = $0.30 for 10 stories
- FLUX (via Replicate): ~$0.05 per image × 20 panels = $1 per story × 10 = $10
- **Total estimate**: ~$15-20 for entire hackathon

Pro Tips

1. Use OpenAI GPT-3.5-Turbo for development ($0.002/call)
2. Pre-generate 2-3 stories before demo
3. Cache everything aggressively
4. Have fallback placeholder images

🚨 Risk Mitigation

Biggest Risks

1. **FLUX is too slow**: Pre-generate stories, use progress bars
2. **Character consistency fails**: Use simpler style, focus on story
3. **APIs rate limit**: Implement retries, have backup API keys
4. **Integration breaks**: Daily sync points, clear contracts
5. **Ran out of time**: Feature prioritization, have v1/v2/v3 scope

Backup Plans

- If FLUX fails: Use Stable Diffusion or even DALL-E
- If generation is too slow: Show "in progress" and use pre-made examples
- If PDF export breaks: Use screenshot/HTML export

📞 Communication Protocol

Slack/Discord Channels

- `#general` - Coordination
- `#frontend` - Person 3 & 4
- `#backend` - Person 1 & 2
- `#devops` - Person 5
- `#blockers` - Urgent issues

Sync Schedule

- Hour 2: API contract confirmation
- Hour 8: Progress check-in
- Hour 12: Integration begins
- Hour 18: Feature freeze
- Hour 22: Presentation practice

🎯 Success Criteria

You've succeeded if you can demo:

1. ✅ Teacher sets up a "Physics Grade 8" classroom
2. ✅ Show 3 student avatars (with photos)
3. ✅ Teacher enters: "We learned about Newton's Laws today"
4. ✅ Show 3 AI-generated story concepts
5. ✅ Display a complete 10-20 panel comic with student avatars
6. ✅ Export to PDF and download

If you can do this smoothly, you'll impress the judges!

Good luck! 🚀