Novellium

A modern, web-based visual novel engine with cloud integration, visual builder, and seamless game sharing capabilities. Create, play, and share interactive stories without any setup required.A simple browser-based visual novel engine where you can create and play interactive stories.

> **Author:** James Hill
> *Final year RMIT student building pet projects in my spare time.*

✨ Features## What is this?

Core EngineNovellium lets you:

- **Modern Web Technology**: Built with vanilla JavaScript, no frameworks required- **Create** visual novels using a simple web interface
- **Responsive Design**: Works on desktop, tablet, and mobile devices- **Play** visual novels in your browser
- **Save System**: Automatic and manual save/load functionality with export/import- **Share** your stories as files
- **Asset Management**: Support for images, audio, and multiple file formats
- **Character System**: Dynamic character sprites with multiple expressionsNo downloads, no complex setup - just open it in your browser and start creating.
- **Event-Driven Architecture**: Flexible story progression with choices and branching

Quick Start

Visual Builder

- **Drag & Drop Interface**: Intuitive visual story creation1. **Start the server**:
- **Real-Time Preview**: See your story as you build it ```bash
- **Character Management**: Easy character creation with sprite assignment # Install http-server if you don't have it
- **Event Flow**: Visual event connection and branching logic npm install -g http-server
- **Asset Integration**: Simple file upload and management
- **Export System**: Package games as ZIP files for sharing # Run the server

  http-server -p 8000 -c-1

Cloud Integration ("Bottle in the Sea") ```

- **Anonymous Sharing**: Upload games to the cloud without registration
- **Automatic Asset Hosting**: Images and assets stored in Supabase Storage2. **Open in browser**:
- **Global Game Library**: Discover games shared by other creators - Go to `http://localhost:8000`
- **Dual Mode Import**: Support both local and cloud game deployment - Click "Builder" to create stories
- **Real-time Sync**: Games appear instantly in the global library - Click "Library" to play stories

🚀 Quick Start## How to Create a Story

Option 1: Use Online (Recommended)1. **Open the Builder** (`build.html`)

Visit **novellium.vercel.app** to start playing and creating immediately.2. **Add your game info** - title, author, description

3. **Create characters** - give them names and colors

Option 2: Local Development4. **Write events** - dialogue, choices, narration

```
# Clone the repository6. **Test** - go back to the library and play your story

git clone https://github.com/SpeedyDuck790/Novelluim.git

cd Novelluim## How Stories Work



# Install dependenciesStories are made of **events** that link together:

npm install

- **Dialogue**: Characters talking

# Start local server- **Narration**: Story text without a character

npm run dev- **Choices**: Let players make decisions

- **Scene**: Change backgrounds or music

# Or use Python

python -m http.server 8000Each event can link to the next one, creating your
story flow.



# Or use Node.js http-server## File Structure
```

```
npx http-server -p 8000 -c-1
```

Novellium/

Open `http://localhost:8000` in your browser. ├── index.html # Main library page (play games)

├── build.html # Game builder interface

📖 How to Use ├── styles.css # Global styles and themes

├── package.json # Project configuration

Playing Games ├── README.md # This documentation

1. **Browse Library**: View local and cloud games on the home page │
2. **Click to Play**: Select any game to start playing immediately ├── src/ # Core engine code
3. **Save Progress**: Use manual saves or rely on auto-save functionality │ ├── engine.js # Main game engine
4. **Import Games**: Drag & drop ZIP files to add new games │ ├── models/ # Data models

│ │ ├── Character.js # Character class definition

Creating Games │ │ ├── Event.js # Story event class

1. **Open Builder**: Click "Create Game" or visit `/build.html` │ │ └── GameState.js # Game state management
2. **Set Game Info**: Add title, author, and description │ ├── managers/ # System managers
3. **Create Characters**: Add characters with names, colors, and sprites │ │ ├── AssetLoader.js # Load images/audio assets
4. **Build Story**: Create events and connect them with choices │ │ ├── ConditionEvaluator.js # Handle conditional logic
5. **Test Locally**: Use "Deploy Local" to test your game │ │ └── SaveManager.js # Save/load game progress
6. **Share to Cloud**: Use "Deploy Cloud" to share globally │ └── ui/ # User interface

│ └── Renderer.js # Display engine for scenes

Importing/Exporting │

- **Local Mode**: Games stored in browser, can be exported as ZIP ├── config/ # Configuration files

- **Cloud Mode**: Games uploaded to Supabase, available to everyone│ └── games-list.json # Registry of available games
- **ZIP Format**: Standard export format for sharing between users│

├── scripts/ # Utility scripts

🏗 Technical Architecture│ ├── check-game-data.js # Validate game data

│ └── navbar.html # Shared navigation component

Frontend│

- **Engine**: `src/engine.js` - Core game engine and rendering ├── gamefolder/ # Game storage directory
- **Builder**: `build.html` - Visual story creation interface│ ├── adventure-game/ # Example adventure game
- **UI Components**: `src/ui/` - Reusable interface components│ │ ├── config.json # Game metadata
- **Asset Management**: `src/managers/AssetLoader.js` - File loading and caching│ │ ├── characters.json # Character definitions

│ │ ├── story.json # Story events and flow

Backend (Cloud Features)│ │ ├── backgrounds/ # Background images

- **Database**: Supabase PostgreSQL with simplified schema│ │ └── sprites/ # Character sprites
- **Storage**: Supabase Storage for game assets│ └── dating-game/ # Example dating sim
- **API**: Vercel serverless functions in `/api/`│ ├── config.json # Game metadata
- **CDN**: Automatic asset delivery via Supabase CDN│ ├── characters.json # Character definitions

│ ├── story.json # Story events and flow

Cloud Infrastructure│ ├── backgrounds/ # Background images

```
User Browser → Vercel (Frontend) → Supabase (Database + Storage)│
                    ↓├── NovelliumLogo/                    # Brand assets and
icons
              API Functions (Node.js)│    ├── logo.png                     #
Main logo
                    ↓│    ├── favicon.ico                    # Browser icon
```

~~~
            Game Data + Assets│    ├── favicon.svg                     # Vector
browser icon

```│    ├── apple-touch-icon.png            # iOS home screen icon

│    ├── favicon-96x96.png               # High-res favicon

## 🗂 Project Structure│    ├── web-app-manifest-192x192.png    # PWA icon
(192x192)

│    ├── web-app-manifest-512x512.png    # PWA icon (512x512)

```│    └── site.webmanifest                # PWA manifest

Novelluim/│

├── 🗀 api/                    # Vercel API functions├── docs/
# Documentation

│    ├── games.js             # Game CRUD operations│    └── README.pdf
# PDF version of docs

│    └── downloads.js         # Download tracking│

├── 🗀 config/                 # Configuration files└── .git/
# Git repository data

│    └── games-list.json      # Local games registry```

├── 🗀 database/               # Database schema and migrations

│    ├── schema.sql           # Main database schema## Features

│    └── update-bucket-mime-types.sql

├── 🗀 docs/                   # Documentation and demos**For Creators:**

├── 🗀 gamefolder/             # Local game storage- Visual editor with forms

│    ├── dating-game/         # Example game- JSON editor for advanced users

│    └── adventure-game/      # Example game- Asset upload (images, music)

├── 🗀 src/                    # Core engine source- Export/import game files

│    ├── 🗀 managers/          # System managers- Live preview

│    ├── 🗀 models/            # Data models

│    ├── 🗀 ui/                # UI components**For Players:**

│    └── 🗀 config/            # Configuration- Save/load games
~~~

~~~
├── 🗀 scripts/              # Utility scripts- Customizable themes

├── 🗀 NovelliumLogo/        # Brand assets- Typewriter text effects

├── index.html              # Main application- Choice-driven stories

├── build.html              # Visual builder

├── styles.css              # Global styles## Tech Stuff

└── README.md               # This file

```- **No dependencies** - pure HTML/CSS/JavaScript

- **Browser storage** - saves in localStorage

## ⚙ Configuration- **ES6 modules** - modern JavaScript

- **Canvas rendering** - for backgrounds

### Environment Variables- **File exports** - share as ZIP files

Create `.env.local` for local development:

```env## Need Help?

SUPABASE_URL=your_supabase_url

SUPABASE_ANON_KEY=your_supabase_anon_key- Check the builder's help sections

```- Look at example games in `gamefolder/`

- File issues on GitHub if something breaks

### Supabase Setup

1. Create a Supabase project## License

2. Run the SQL schema from `database/schema.sql`

3. Create a storage bucket named `game-assets`Created by James Hill. Use it
however you want.

4. Set up RLS policies for anonymous access

5. Configure CORS for your domain---



### Vercel Deployment**Simple. Clean. It just works.** 🎮

1. Connect your GitHub repository to Vercel
~~~

~~~
2. Add environment variables in Vercel dashboard---

3. Deploy automatically on push to main branch

## Solutions for Persistent Game Imports on Vercel

## 🎮 Game Format

### 🟢 **Simple Solutions** (Easy Implementation)

### ZIP Structure

```**1. GitHub Integration**

game-name.zip- Use GitHub API to commit imported games directly to repository

├── config.json              # Game configuration- Requires GitHub token and
automatic commits

├── characters.json          # Character definitions- Games become part of the
repo and persist for all users

├── story.json              # Events and story flow- ✅ No backend needed,
uses GitHub as storage

└── assets/                 # Game assets- ❌ Requires authentication, public
commits

    ├── backgrounds/        # Background images

    ├── sprites/           # Character sprites**2. Vercel KV Storage**

    └── audio/            # Sound effects and music- Use Vercel's built-in
Redis-like key-value storage

```- Simple API calls to store/retrieve game data

- Fast access, built into Vercel platform

### JSON Schemas- ✅ Easy setup, integrated with Vercel

See `docs/GAME-FORMAT.md` for detailed format specifications.- ❌ Paid
feature, data limits



## 🛠️ Development**3. Browser IndexedDB Enhancement**

- Upgrade from localStorage to IndexedDB for larger storage

### Adding New Features- Add import/export features for sharing
~~~

```
1. **Frontend**: Modify engine or UI components in `src/`- Better performance
for large games

2. **Builder**: Update `build.html` for creation tools- ✅ Still client-side,
no backend needed

3. **Backend**: Add API functions in `api/` folder- ❌ Still per-user, not
globally shared

4. **Database**: Update schema in `database/` folder

### 🟡 **Medium Solutions** (Moderate Setup)

### Testing

- **Local Games**: Test with example games in `gamefolder/`**4. Vercel
Serverless Functions + Database**

- **Cloud Features**: Verify upload/download functionality- Add API endpoints
via Vercel Functions

- **Cross-Platform**: Test on different devices and browsers- Connect to
external database (MongoDB, PostgreSQL)

- Full CRUD operations for games

### Debugging- ✅ Scalable, proper backend architecture

- **Browser Console**: Check for JavaScript errors- ❌ Requires database setup
and management

- **Network Tab**: Monitor API requests and asset loading

- **Supabase Dashboard**: Monitor database and storage usage**5. Firebase
Integration**

- Use Firebase Firestore for game storage

## 🌐 Cloud Features- Real-time sync across users

- Built-in authentication

### Anonymous Sharing- ✅ Google-managed, real-time features

- No registration required- ❌ Google dependency, learning curve

- Games become public immediately

- "Bottle in the sea" concept - share and discover**6. Supabase Backend**

- PostgreSQL database with REST API

### Asset Management- Built-in auth and file storage
```

```
- Automatic image optimization- Open-source alternative to Firebase

- CDN delivery for fast loading- ✅ Full-featured, good free tier

- MIME type validation- ❌ Another service to manage

- 50MB file size limit per game

### 🔴 **Advanced Solutions** (Complex Implementation)

### Analytics (Basic)

- Download counting**7. Headless CMS Integration**

- Game popularity metrics- Use Strapi, Sanity, or Contentful

- Storage usage tracking- Treat games as content entries

- Admin interface for game management

## 🤝 Contributing- ✅ Professional content management

- ❌ Overkill for simple games, costly

1. Fork the repository

2. Create a feature branch: `git checkout -b feature-name`**8. Blockchain/IPFS
Storage**

3. Make your changes and test thoroughly- Store games on decentralized storage

4. Commit with descriptive messages- Immutable, censorship-resistant

5. Push and create a Pull Request- Unique game NFTs or tokens

- ✅ Decentralized, future-proof

### Development Guidelines- ❌ Complex, slow, expensive

- **Code Style**: Use consistent formatting and meaningful names

- **Documentation**: Update README and docs for new features**9. Custom Backend
Service**

- **Testing**: Test both local and cloud functionality- Separate Node.js/Python
backend

- **Backwards Compatibility**: Maintain compatibility with existing games-
Deploy on Railway, Render, or DigitalOcean

- Full control over architecture
```

```
## 📄 License- ✅ Complete flexibility

- ❌ Most complex, separate hosting costs

This project is open source. Feel free to use, modify, and distribute according
to the license terms.

### 🔧 **Hybrid Solutions** (Best of Both Worlds)

## 🆘 Support

**10. Static + Dynamic Hybrid**

- **Issues**: Report bugs via GitHub Issues- Keep static games in repository

- **Discussions**: Use GitHub Discussions for questions- Add optional cloud
sync for user imports

- **Documentation**: Check `docs/` folder for detailed guides- Graceful
degradation when offline

- **Examples**: Study games in `gamefolder/` for reference- ✅ Works
everywhere, enhanced when connected

- ❌ More complex state management

## 🔮 Roadmap

**11. Pull Request Automation**

- [ ] Advanced analytics dashboard- Users submit games via automated PRs

- [ ] Game rating and review system- GitHub Actions validate and merge

- [ ] Collaborative editing features- Community moderation workflow

- [ ] Plugin system for custom components- ✅ Transparent, version controlled

- [ ] Mobile app wrapper- ❌ Requires approval workflow

- [ ] Advanced audio features

- [ ] Localization support**12. CDN + Edge Functions**

- Store games on CDN (Cloudflare R2, AWS S3)

---- Use edge functions for fast access

- Global distribution

**Novellium** - Empowering storytellers to create and share interactive
narratives effortlessly.- ✅ Fast worldwide, scalable
- ❌ Multiple services to configure
```

```
### 📊 **Recommended Implementation Order**

**Phase 1: Quick Win**
```

1. GitHub API integration for direct commits
2. Enhanced export/import with better UX

```


**Phase 2: Proper Backend**
```

3. Vercel Functions + Vercel KV
4. User authentication (GitHub OAuth)

```


**Phase 3: Scale & Polish**
```

5. Migration to full database if needed
6. Advanced features (ratings, search, etc.)

```


### 💡 **Code Examples Available**

Each solution above can be implemented with specific code examples:
- API endpoint structures
- Database schemas
- Authentication flows
- Import/export mechanisms

Choose based on your priorities: **simplicity**, **cost**, **features**, or
**scalability**.
```