ThesisHub Project Documentation

Table of Contents

Overview

ThesisHub is a collaborative academic platform designed to facilitate thesis/project management between students and teachers. The platform enables students to form groups, create proposals, find supervisors, and manage their thesis progress, while teachers can supervise groups, review proposals, assign tasks, and track student progress.

Technologies Used

Frontend Framework

- **React 19** - JavaScript library for building user interfaces
- **Vite** - Next-generation frontend tooling for fast development
- **Tailwind CSS** - Utility-first CSS framework for styling
- **Framer Motion** - Production-ready motion library for React
- **Lucide Icons** - Beautiful & consistent icon toolkit

Development Tools

- **ESLint** - Pluggable JavaScript linter
- **PostCSS** - Tool for transforming CSS with JavaScript
- **Autoprefixer** - Plugin to parse CSS and add vendor prefixes

Package Management

- **npm** - Node package manager for dependency management

Framework Architecture

Project Structure

```
src/
├── Entities/              # Data models in JSON format
├── Pages/                 # UI pages for different user roles
├── api/                   # API clients and services
├── components/            # Reusable UI components
│   ├── cards/             # Card components for data display
│   ├── chat/              # Chat interface components
│   ├── dashboard/         # Dashboard layout and navigation
│   ├── profile/           # Profile management components
│   ├── proposal/          # Proposal creation and management
│   └── ui/                # Core UI components
├── services/              # Business logic and data services
└── utils/                 # Utility functions and helpers
```

Component Architecture

1. **Dashboard Layout** - Provides consistent navigation across the application
2. **Entity System** - JSON-based data modeling for all system objects
3. **Service Layer** - Centralized data management and business logic
4. **UI Components** - Modular, reusable interface elements

State Management

- **LocalStorage** - Client-side data persistence
- **React Hooks** - Component state management (useState, useEffect, useContext)
- **Context API** - Global state sharing between components

API Implementation

Current Implementation (LocalStorage)

The application currently uses browser LocalStorage for data persistence:

- **base44Client.js** - Mock database client simulating API interactions
- **databaseService.js** - Centralized service layer abstracting data operations
- **EntityManager** - Class providing CRUD operations for each entity

Data Operations

All entities support the following operations:

- `list()` - Retrieve all records
- `filter(criteria)` - Filter records by criteria
- `findById(id)` - Retrieve record by ID
- `create(data)` - Create new record
- `update(id, updates)` - Update existing record
- `delete(id)` - Delete record

Entities

1. **Student** - Student profiles and group information
2. **Teacher** - Faculty profiles and supervision details
3. **StudentGroup** - Group formation and management
4. **GroupInvitation** - Peer invitation system
5. **Proposal** - Thesis proposal documents
6. **Message** - Inter-user communication
7. **Meeting** - Scheduled meetings between groups and supervisors
8. **Task** - Assigned tasks and progress tracking
9. **SharedFile** - Document sharing system
10. **WeeklyProgress** - Weekly progress reporting
11. **SupervisionRequest** - Teacher supervision requests

API Endpoints (Previous MongoDB Implementation)

When connected to MongoDB, the application used the following RESTful endpoints:

Student Endpoints

- `GET /api/students` - Retrieve all students
- `GET /api/students/:id` - Retrieve specific student
- `POST /api/students` - Create new student
- `PUT /api/students/:id` - Update student
- `DELETE /api/students/:id` - Delete student

Teacher Endpoints

- `GET /api/teachers` - Retrieve all teachers
- `GET /api/teachers/:id` - Retrieve specific teacher
- `POST /api/teachers` - Create new teacher
- `PUT /api/teachers/:id` - Update teacher
- `DELETE /api/teachers/:id` - Delete teacher

AI Integrations

Proposal Creation Assistant

The platform features an AI-powered proposal assistant that helps students:

- Generate thesis titles based on interests
- Develop research methodologies
- Structure proposal documents
- Provide content suggestions

Implementation Details

- **Floating Chatbot** - Accessible via icon in proposal creation page
- **Natural Language Processing** - Understands student inputs and context
- **Real-time Suggestions** - Provides instant feedback during proposal writing
- **Context Awareness** - Remembers conversation history for coherent assistance

Technical Implementation

- **Frontend Integration** - React component with WebSocket connectivity
- **Prompt Engineering** - Carefully crafted prompts for optimal responses
- **Response Formatting** - Structured output for easy consumption
- **Error Handling** - Graceful degradation when AI services are unavailable

Features

1. **Title Generation** - Suggests relevant thesis titles
2. **Methodology Guidance** - Recommends research approaches
3. **Content Enhancement** - Improves proposal quality and clarity
4. **Structure Assistance** - Helps organize proposal sections

Database Structure

Student Entity

```
{
  "student_id": "string",
  "password_hash": "string",
  "full_name": "string",
  "email": "string",
  "profile_photo": "string",
  "department": "string",
  "group_id": "string",
  "is_group_admin": "boolean",
  "status": "enum(active, graduated, inactive)"
}
```

Teacher Entity

```
{
  "teacher_id": "string",
  "password_hash": "string",
  "full_name": "string",
  "email": "string",
  "profile_photo": "string",
  "department": "string",
  "research_field": "string",
  "publications": "array",
  "accepted_topics": "array",
  "max_students": "number",
  "current_students_count": "number",
  "status": "enum(active, on_leave, inactive)"
}
```

StudentGroup Entity

```
{
  "group_name": "string",
  "member_ids": "array",
  "project_title": "string",
  "project_description": "string",
  "project_type": "string",
  "assigned_teacher_id": "string",
  "status": "enum(forming, proposal_submitted, supervised, completed)"
}
```

Proposal Entity

```
{
  "group_id": "string",
  "title": "string",
  "description": "string",
  "methodology": "string",
  "technologies": "array",
  "status": "enum(draft, submitted, under_review, approved, revision_required)",
  "submitted_to_teacher_id": "string",
  "feedback": "string"
}
```

User Roles and Features

Student Features

1. **Registration & Login** - Secure account creation and authentication
2. **Profile Management** - Personal information and avatar upload
3. **Group Formation** - Create/join student groups
4. **Partner Selection** - Browse and invite peers
5. **Proposal Creation** - AI-assisted thesis proposal development
6. **Supervisor Search** - Find and request supervision
7. **Communication** - Chat with group members and supervisors
8. **Task Management** - Track assigned tasks and deadlines
9. **Progress Tracking** - Weekly progress reporting
10. **File Sharing** - Document collaboration

Teacher Features

1. **Registration & Login** - Secure account creation and authentication
2. **Profile Management** - Professional information and research interests
3. **Student Supervision** - Review and accept supervision requests
4. **Proposal Review** - Evaluate student proposals
5. **Task Assignment** - Create and assign tasks to groups
6. **Progress Monitoring** - Track student weekly progress
7. **Meeting Scheduling** - Organize group meetings
8. **Communication** - Chat with supervised groups
9. **File Management** - Access shared documents
10. **Dashboard Analytics** - Overview of supervised groups and workload

Administrator Features

1. **User Management** - Overview of all students and teachers

2. **System Monitoring** - Track platform usage and statistics
3. **Data Management** - View and manage all system entities
4. **Configuration** - System settings and parameters

Installation and Setup

Prerequisites

- Node.js (version 16 or higher)
- npm (comes with Node.js)

Installation Steps

1. Clone the repository:

```
git clone <repository-url>
cd thesis-project
```

2. Install dependencies:

```
npm install
```

3. Start the development server:

```
npm run dev
```

4. Access the application at `http://localhost:5173`

Available Scripts

- `npm run dev` - Start development server
- `npm run build` - Build for production
- `npm run preview` - Preview production build locally
- `npm run lint` - Run ESLint for code quality checks

Project Configuration

- **Vite Config** - `vite.config.js` for build settings
- **Tailwind Config** - `tailwind.config.js` for styling
- **PostCSS Config** - `postcss.config.js` for CSS processing
- **ESLint Config** - `eslint.config.js` for code linting

Deployment

Static Deployment Options

1. **Vercel** - Zero-configuration deployment
2. **Netlify** - Continuous deployment platform
3. **GitHub Pages** - Free hosting for static sites
4. **Firebase Hosting** - Google's CDN for web apps

Deployment Steps (Vercel)

1. Create a Vercel account
2. Connect your GitHub repository
3. Configure build settings:
    - Build Command: `npm run build`
    - Output Directory: `dist`
4. Deploy and receive live URL

Environment Variables

The application uses the following environment variables:

- `VITE_APP_TITLE` - Application title
- `VITE_API_BASE_URL` - API endpoint (when connected to backend)

Performance Optimization

- **Code Splitting** - Automatic route-based splitting
- **Asset Optimization** - Image compression and minification
- **Lazy Loading** - Component lazy loading for faster initial load
- **Caching Strategies** - Browser caching for static assets

Future Enhancements

Backend Integration

- RESTful API development with Node.js/Express
- Database integration with MongoDB or PostgreSQL
- Authentication system with JWT tokens
- Real-time features with WebSocket

Advanced Features

- Notification system

- Analytics dashboard
- Mobile application
- Multi-language support
- Advanced search and filtering

Documentation generated on December 2, 2025
ThesisHub - Academic Collaboration Platform