MAD CONTEST 2025

StudySync

AI-Powered Study Group Matcher

Dillon Opperman

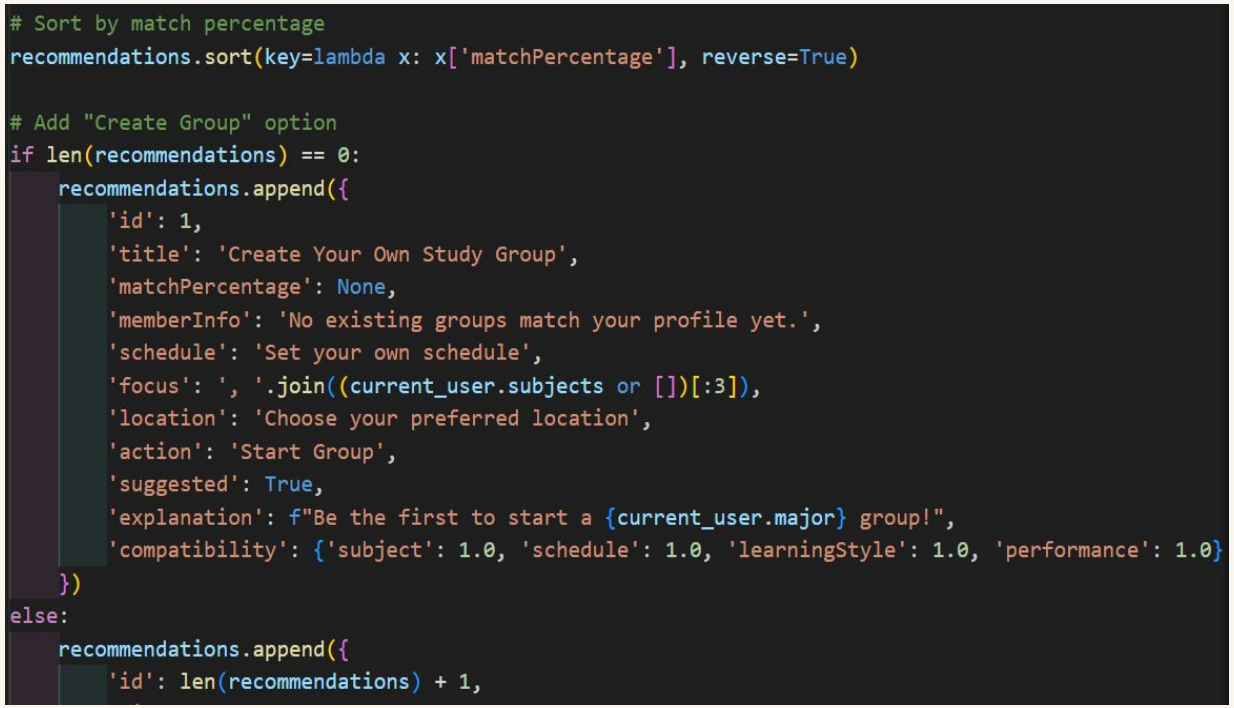
The Problem

Students are Disconnected/Lazy

Finding study partners is currently a manual, inefficient process

- **Social Anxiety:** Asking classmates "randomly" is scary. (thanks covid)
- **Mismatched Schedules:** Finding time overlap is a common problem. (everyone has individual lives)
- **Different Goals:** "Cramming" vs. "Deep Learning" styles clash. (learning for fun (not serious) compared to the graduate student (serious)

Result: Students study alone, or just rely on AI to be their study partners. (in the workforce, this won't cut the cake)



```
# Sort by match percentage
recommendations.sort(key=lambda x: x['matchPercentage'], reverse=True)

# Add "Create Group" option
if len(recommendations) == 0:
    recommendations.append({
        'id': 1,
        'title': 'Create Your Own Study Group',
        'matchPercentage': None,
        'memberInfo': 'No existing groups match your profile yet.',
        'schedule': 'Set your own schedule',
        'focus': ', '.join((current_user.subjects or [])[:3]),
        'location': 'Choose your preferred location',
        'action': 'Start Group',
        'suggested': True,
        'explanation': f"Be the first to start a {current_user.major} group!",
        'compatibility': {'subject': 1.0, 'schedule': 1.0, 'learningStyle': 1.0, 'performance': 1.0}
    })
else:
    recommendations.append({
        'id': len(recommendations) + 1,
```

Core Question 1: Is AI Necessary?

```python
# Import database and auth
from database import init_db, get_db, close_db
from models import User, StudyGroup, GroupMember, Message, MessageReaction, StudySession,
from auth import hash_password, verify_password, generate_token, token_required

app = Flask(__name__)
CORS(app)

# Initialize the BERT model
model = SentenceTransformer('sentence-transformers/all-MiniLM-L6-v2')

# Initialize database
with app.app_context():
    init_db()
```

Frankly, SQL filters aren't enough.

A simple database query matches "Biology" to "Biology." It misses the nuance of human compatibility.

```
Semantic Understanding (SBERT):
"I like visual learning"
≈ "I prefer diagrams and charts"
Score: 0.92
Similarity
```

StudySync AI will analyze **Learning Styles** and **Study Goals** to predict compatibility, not just availability.

Core Question 2: What is out there currently?

Bulletin Boards

Pros: Visible.
Cons: outdated, limited reach, no filtering, requires physical presence.
(old)

Discord/GroupMe

Pros: Chat features.
Cons: Chaotic, noisy, not designed for *forming* groups, just hosting them.
(disorganized)

StudySync

Advantage: Active *Discovery*. It doesn't just host the chat; it finds the people you *should* be chatting with.

(effective)

The Niche Landscape

Note Markets

(StudySoup, CourseHero)

The Flaw: "Transactional Content." Users pay for PDFs, they don't connect with people. It monetizes anxiety rather than solving the root cause of isolation.

(isolated)

Team Builders

(Fynk, CATME)

The Flaw: "Resume-Based." Matches are based on balancing hard skills (coding vs. design) for projects, not on behavioral compatibility for studying.

(lack of personalization)

Peer Assessment

(Peerceptiv, Kritik)

The Flaw: "Mandatory." Groups are assigned by professors for grading purposes. Collaboration ends the moment the assignment is submitted.

(forced study groups)

Core Question 3: Why an App?

Created for the Student Lifestyle

Students are mobile-first now days. A website doesn't fit their dynamic needs.

- **Push Notifications:** "Your study group is meeting in 10 mins." (high priority)
- **Location Services:** "Find groups in *this* library right now." (low priority)
- **Hardware Access:** Quickly snap a photo of notes to share in chat. (high priority)

High-Level Architecture

React Native

Cross-platform Frontend.
Redux for State.
AsyncStorage for caching.

REST API (JSON)

Flask Backend

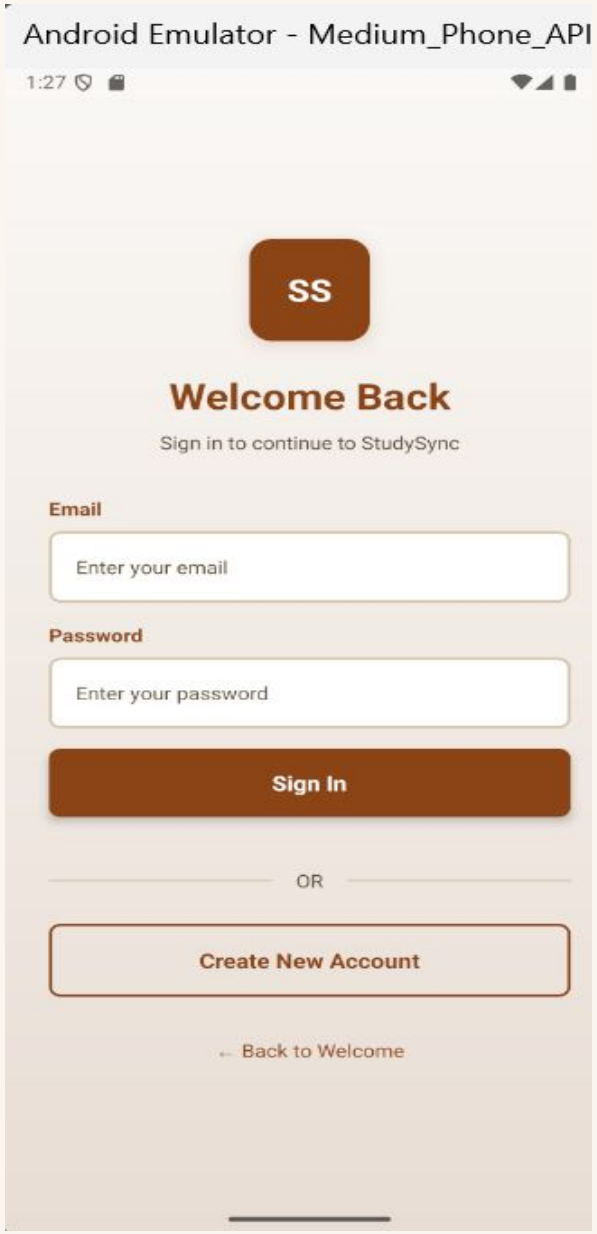
Authentication Logic.
AI Processing (Torch).
API Endpoints.

SQLAlchemy

MySQL Database

Persistent Storage.
Users, Groups, Chats.
Relational Data.

Screen 1: Secure Authentication



Security First

Before accessing the community, users must verify their identity. This prevents spam and ensures a safe environment.

Key Features:

- Email Format Validation (RegEx).
- .edu email enforcement (Student only).
- Secure Password Entry.

Backend Focus: Authentication Logic

Password Hashing

Instead of storing plain-text passwords. I will be using **Bcrypt** with salt.

```
def hash_password(password):
  salt = bcrypt.gensalt()
  hashed = bcrypt.hashpw(enc, salt)
  return hashed
```

JWT (JSON Web Tokens)

Stateless authentication for API requests.

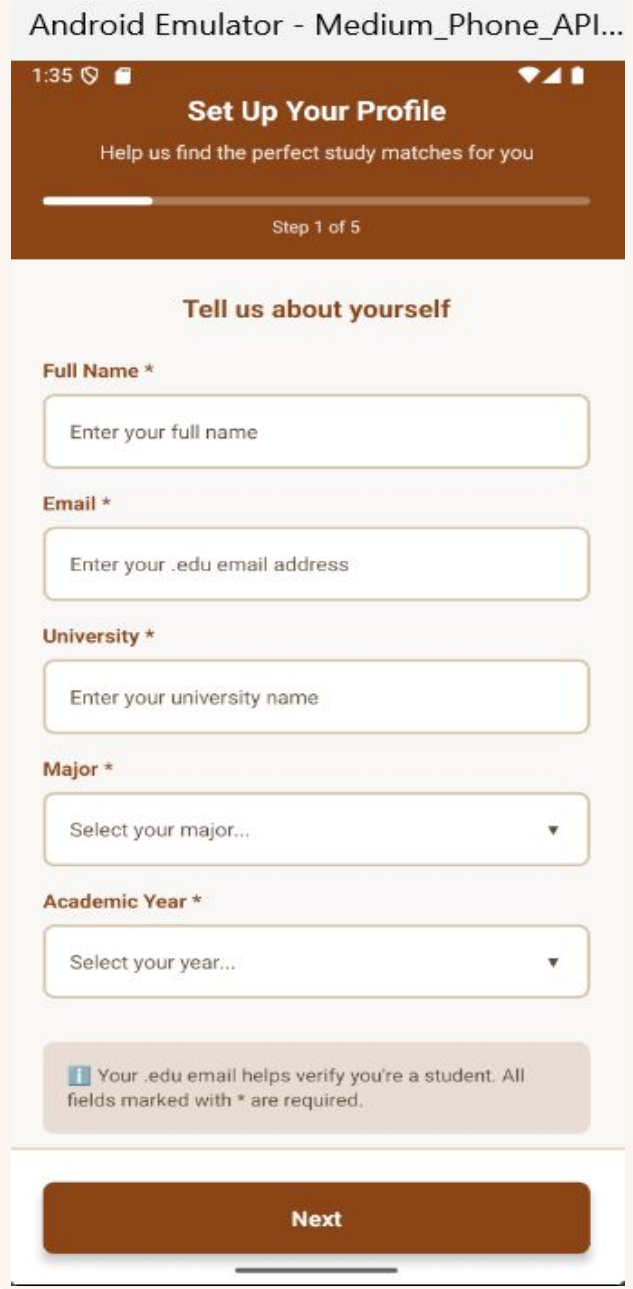
```
payload = {
  'user_id': user_id,
  'exp': datetime() + 7_days
}
token = jwt.encode(payload, KEY)
```

Screen 2: Profile Construction

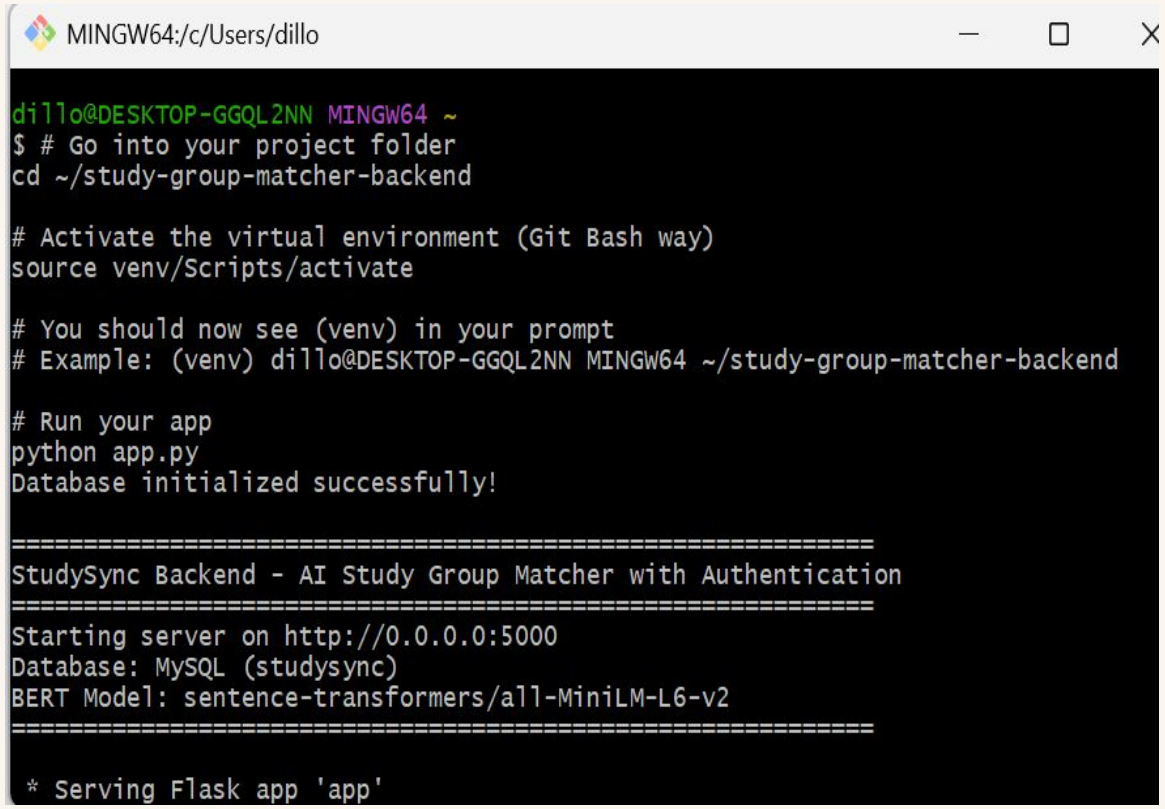
The 5-Step Onboarding

To make good matches, the app will collect the following:

1. **Basic Info:** Major, Year, University.
2. **Learning Style:** Visual vs. Auditory (for AI analysis).
3. **Schedule:** Availability Matrix.
4. **Preferences:** Group size, Study duration.
5. **Security:** Password Setup.



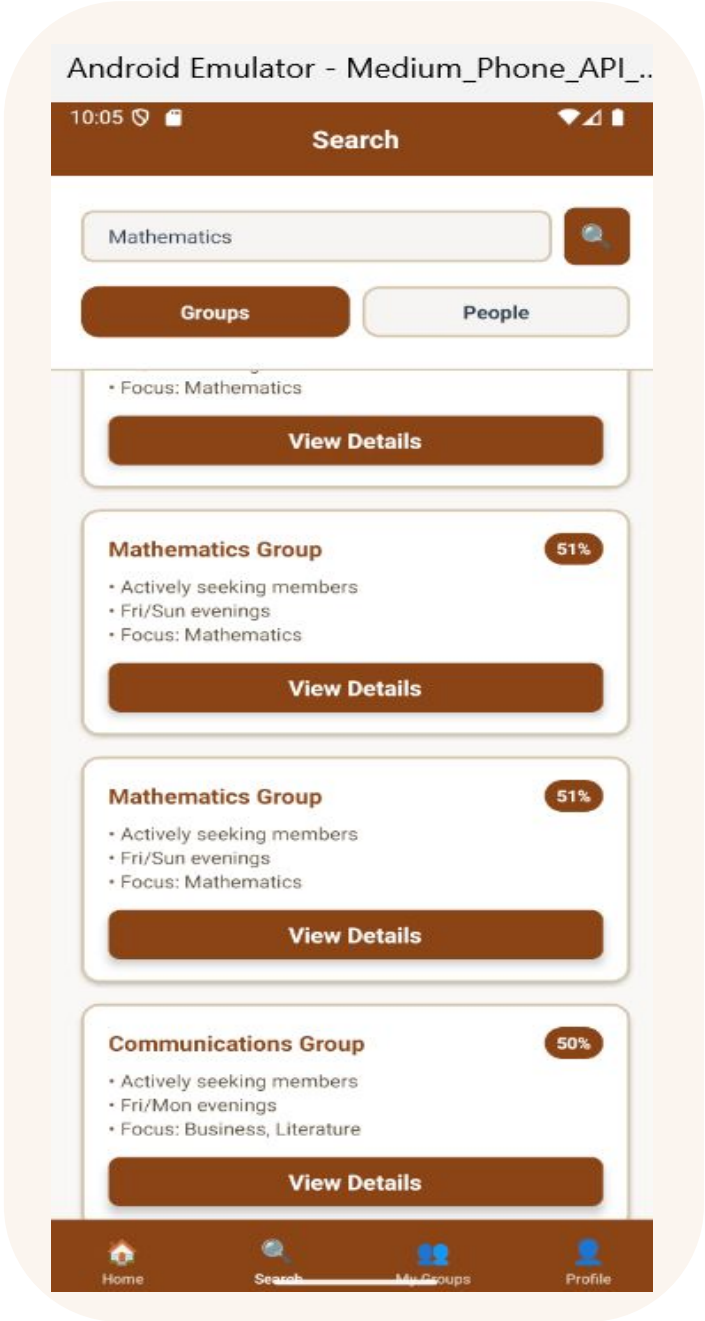
Backend Focus: The Matching Engine



How it calculates the **Compatibility Score**:

```
Subject Score (40%) +
Schedule Overlap (30%) +
Learning Style Similarity (20%) +
Performance Level (10%)
```

AI Component: I will/am using sentence-transformers to convert text descriptions (Learning Styles) into vector embeddings, then calculate Cosine Similarity.



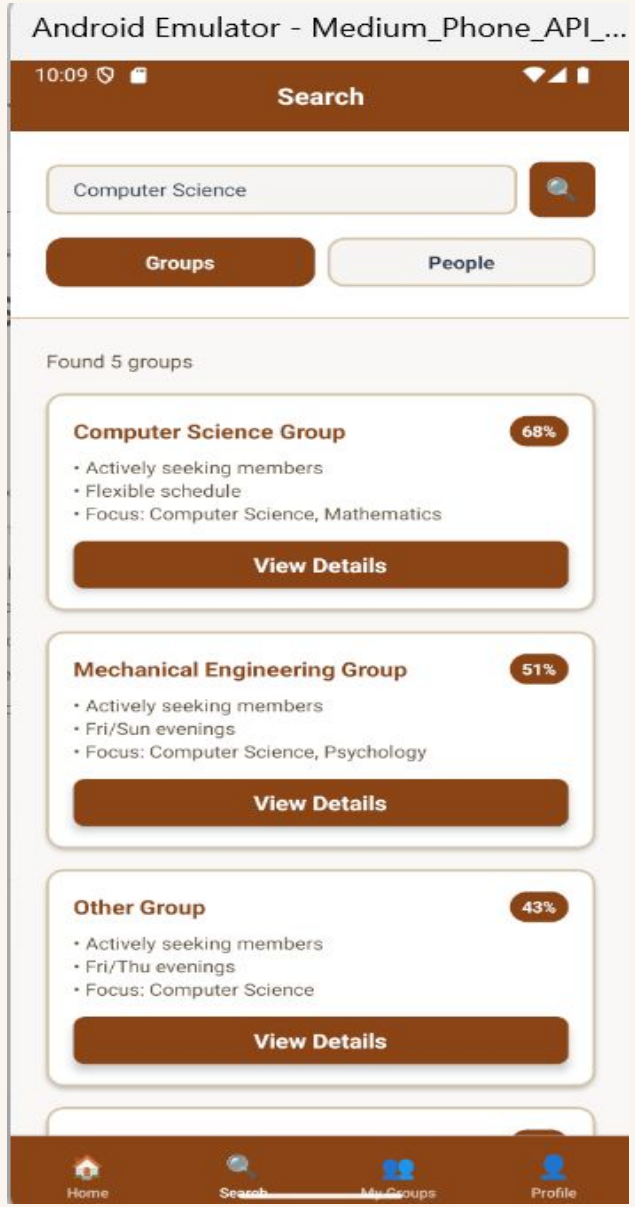
Screen 3: The Dashboard

AI Recommendations

The home screen isn't a static list. It's a processed feed of potential study groups.

- **Match Percentage:** Clearly displayed (e.g 92% Match).
- **Explanation:** The AI tells you *why* you matched (e.g., "Similar schedule and visual learning style").
- **Action:** One-tap "Request to Join".

Screen 4: Semantic Search

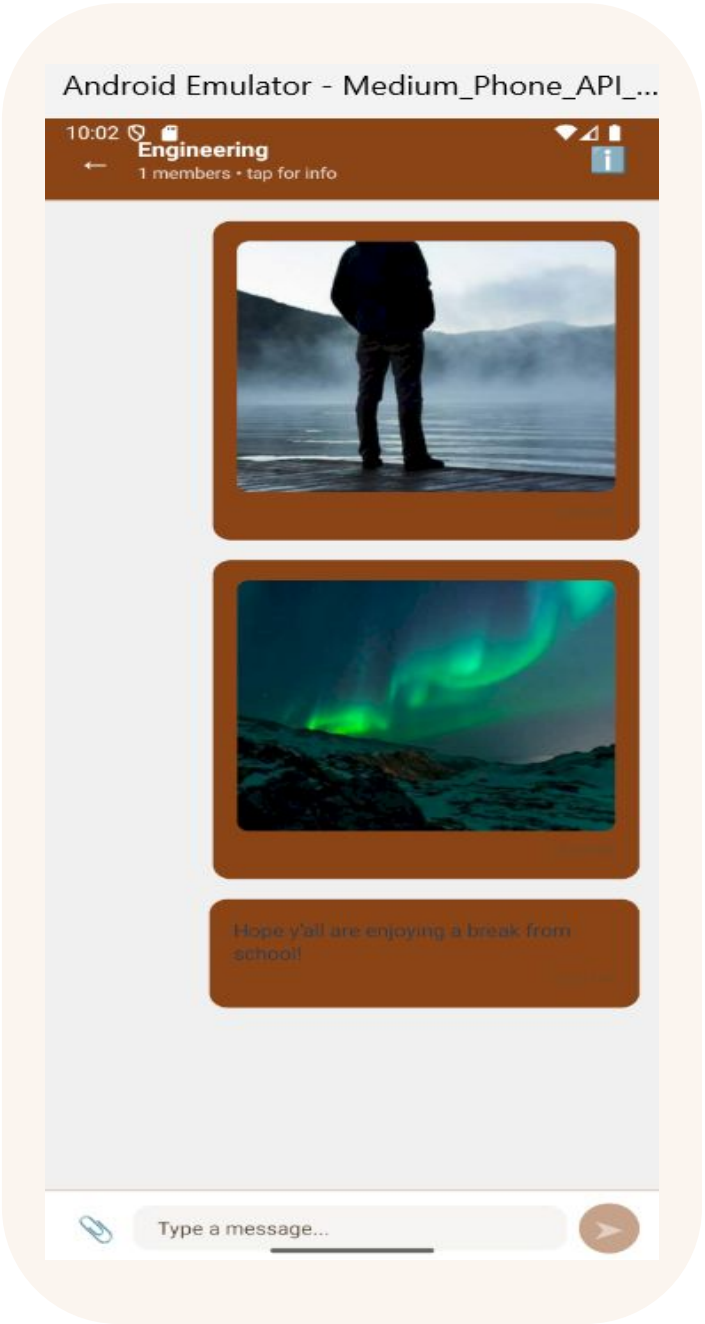


Beyond Keywords

Users can search for specific topics or broad concepts.

The backend /api/search endpoint uses the same BERT model to compare the search query vector against the user corpus vectors.

Example: Searching "coding" finds groups labeled "Python", "Java", and "Web Dev".



Screen 5: Collaboration

Community Building

Once matched, the app facilitates the actual study session.

- **My Groups:** Manage active members
- **Chat:** Text-based communication/live video calls.
- **Resources:** Share photos (camera/gallery) and notes.

Status & Roadmap

Phase 1

Core UI & Local Storage
(Completion 80%)

Phase 2

Flask AI & Auth
(Completion 70%)

Phase 3

Full Database Migration
(Completion 65%)

Phase 4

Real-time Sockets & Launch
(60%)

Current State: Prototype 70% Complete

Authentication (In Process), Database persistence (In Process), AI matching (Testing Phase).

The End

Thank You

https://github.com/DillonOpperman

Works Cited

https://www.reddit.com/r/ClaudeAI/comments/1o2bj9l/introducing_claude_code_plugins_in_public_beta/