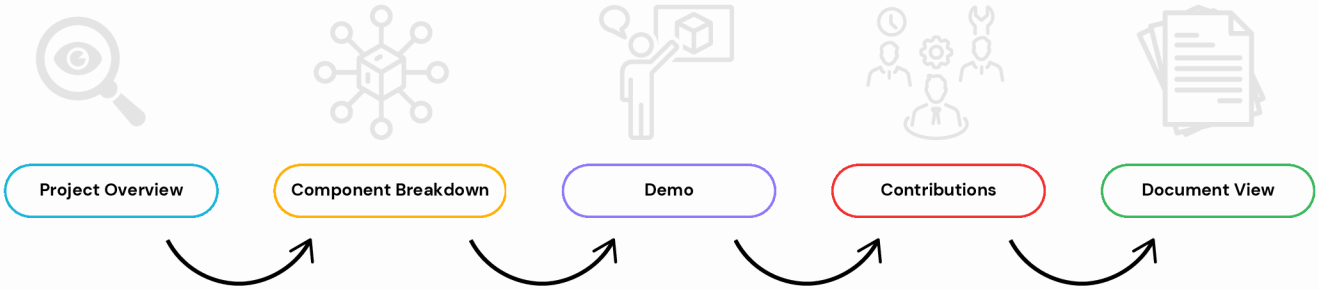
DECEMBER 10, 2024

COMS4153 - CLOUD COMPUTING

StudyLink

Find your study group

AGENDA


Project Overview
Component Breakdown
Demo
Contributions
Document View

PROJECT OVERVIEW

CHALLENGE

- Student isolation in large lecture settings
- Scary to go up to someone you don't know and ask if they want to study together!

REMEDY

- Let StudyLink do the work of finding similarly motivated classmates for you!

PROJECT OVERVIEW

CHALLENGE

- Student isolation in large lecture settings
- Scary to go up to someone you don't know and ask if they want to study together!

REMEDY

- Let StudyLink do the work of finding similarly motivated classmates for you!

Canvas API | **Messaging** | **Group Chats** | **Calendar**

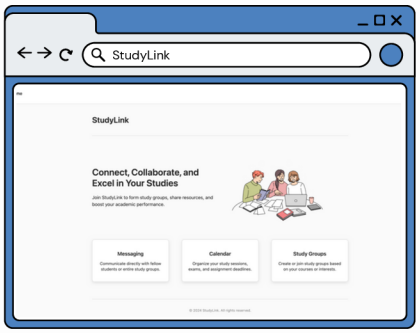
STUDYLINK

AUTOMATED CLASSMATE DISCOVERY

STUDY GROUP FORMATION

COMMUNICATION TOOLS

ACADEMIC CALENDAR



reducing anxiety associated with large lectures

collaborative learning environment + more interconnected student body

enhanced academic performance, student satisfaction, and community building

USER INTERFACE

Built on React and Redux

Hosted in Google Cloud

Backend from Composite Microservice via API Gateway

DEMO

END OF DEMO

MICROSERVICES SHARED FEATURES

Hosted in AWS or GCP

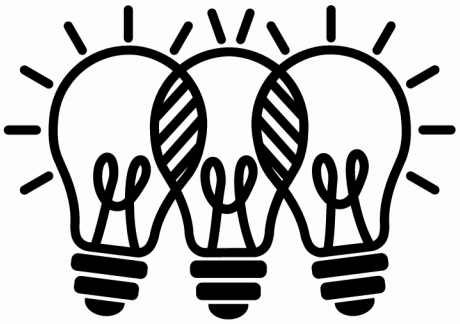
Built using Python/FastAPI

Connects to RDS in AWS

Logging and tracing

Correlation ID propagation

HATEOAS Links



USER MICROSERVICE

The user microservice is responsible for pulling user-specific information from Canvas API and the application database.

When creating an account, the service verifies that user is a Barnard/Columbia student via **Google Authentication**.


```
{
    "user_id": "azs2117"
    "canvas_token": "123456789",
    "first_name": "Amelie",
    "last_name" : "Scheil",
    "short_name": "Amelie",
    "email" : "azs2117@barnard.edu"
    "pronouns": "she/her",
    "courses": ["COMSW4153_001_2024_3 - Cloud Computing"],
    "created_at": "2024-12-09",
    "updated_at": "2024-12-09",

}
```


Functionality (FastAPI)	**Headers**
POST (create/update) a user profile	*google_token*
GET a user profile	*jwt_token, google_token*
GET users by *user_id* or *course*	*jwt_token, google_token*
DELETE a user profile	*jwt_token, google_token*

COURSE ENROLLMENT MICROSERVICE

Communicates with Courseworks via API

Searches through student's current, active courses

Communicates with the other Microservices

"Deployed on Commit" through GitHub

Get courses for student:

```
curl -X 'GET' \ 'http://35.174.4.121:8000/users/<uni>/courses' \
-H 'accept: application/json' \
-H 'token: <*courseworks token without quotes>'
```

Get students in course:

```
curl -X 'GET'\'http://35.174.4.121:8000/course/<*coursecode>
/students' \
-H 'accept: application/json' \
-H 'token: <*courseworks token without quotes>'
```

*coursecode ≈ "AMSTGU4300_001_2024_3"
*courseworks token = token generated from courseworks that allows us to access student's profile

Functionality (FastAPI)

- GET courses for a specific student — *uni, courseworks token*
- GET students in a course — *coursecode, courseworks token*

STUDYGROUP MICROSERVICE

Docker container on a VM

Students can create new groups or join existing

Manages memberships, meeting details

Functionality (through FastAPI):

- GET all study groups
- GET a single study group by GROUP_ID
- PUT a study group by GROUP_ID
- POST a study group, return new GROUP_ID
- DELETE a study group by GROUP_ID

Example


```
{
    "id": 4,
    "name": "Cloud Computing",
    "created_by": "jyl2196",
    "created_at": "2024-12-06T21:15:01.926Z",E
    "is_recurring": false,
    "meeting_date": "2024-12-07",
    "recurrence_frequency": "",
    "start_time": "16:00",
    "end_time": "16:30",
    "recurrence_end_date": "",
    "course_id": "COMSW4153_001_2024_3 -
                  Cloud Computing",
    "members": [
      "jyl2196",
      "er2788"
    ]
}
```

CHAT MICROSERVICE

Users can communicate with one another via
- Direct Message
- Group Chat

Messages are stored in SQL database

Pagination to handle large convo history

Async API calls for improved performance

Functionality (FastAPI)
- GET a conversation by ID
- GET all conversation details
- POST Create a conversation
- PUT Update a conversation

Sample Conversation JSON

```
{
    "name": "Coding Workshop",
    "participants": [
        "John Doe",
        "Jane Smith",
        "Bob Johnson"
    ],
    "messages": [
        {
            "text": "Welcome to our project group!",
            "sender": "CurrentUser",
            "timestamp": "9:00 AM"
        },
        {
            "text": "Hi guys!",
            "sender": "Jane Smith",
            "timestamp": "3:00 PM"
        }
    ],
    "isGroup": true
}
```

COMPOSITE MICROSERVICE

PAAS Deployment.

Implemented in Python.
Use the Synchronize and Asynchonize approach.

Apply the Chroegraphy and orchestration approach to communicate with the services Functionalities (Fast API)

- GET
- POST
- PUT
- DELETE

Example

```
@router.get("/StudyLink/v1/course/{course_id}/students")
@router.post("/StudyLink/v1/{user_id}/conversations", tags=["conversations"])
@router.put("/StudyLink/v1/{user_id}/conversations/{conversation_id}", tags=["conversations"])
```

TEAM CONTRIBUTIONS

JEANNIE

- Github Repos
- Course Enrollment Microservice
  - REST functionality
  - HATEOAS links
  - Middleware logging
  - Correlation ID
  - Communication with Courseworks through API
- Github action that deploys microservice on commit

JONATHAN

- Chat Microservice
  - Rest functionality
  - Logging/Tracing
  - Pagination
  - Asynchronous API
  - Correlation ID propagation
- Application Deployment
  - GCP Instance

JESS

- StudyGroup Microservice
  - REST functionality
  - HATEOAS links
  - Middleware logging
  - Logging/tracing
  - Correlation ID propagation
  - Put in container on VM

SUMYA

- Composite Microservice
  - Rest Functionalites
  - Ochestration and Chroreography Approch
  - Logging/tracing
  - PaaS deployment

EMANUELA

- User Interface
  - Built Interface
  - UI as "Blob" in GCP
- Accessibility:
  - Google Login
  - JWT Tokens
- RDS in AWS
- API Gateway
- FaaS
- Elastic IPs

AMELIE

- User Microservice:
  - REST functionality
  - HATEOAS links
  - Middleware logging
  - Communication with Courseworks through API
  - Logging/tracing
  - Correlation ID propagation
- CQRS

THANK YOU