TUTOR LINK

Jorge Ramirez: 38883171
Gio Kandelaki: 38883147
Nada Elhaddad: 38883074

Team Roles

Nada

- Requirement Analysis
- Requirement Elicitation
  - AS Functional Requirements
  - AS Non-Functional Requirements
  - Constraints

Jorge

- Project Management
- Static Architectural Design
  - Component Diagrams
  - Dynamic Architectural Design
- State Machine Diagrams
  - Sequence Diagrams

Gio

- Static Architectural Design
  - Class Diagrams
- API Definition
  - Swagger Documentation
- Technology Selection

REQUIREMENTS

https://1drv.ms/w/s!AmZXq25nPhR1hrImDuoKJE9LeRJEIg?e=IzKGRL

SRS V1.0

- Clear Scope Delineation
- Structured Documentation
- Guidance for Development and Testing Requirements
- Basis for Project Planning and Effort Estimation

Functional Requirements

User Registration for tutors and users:

- **Mandatory Information**: data require from the users
- **Optional Information**: not mandatory information
- **Authentication**: to make sure its not fake users

Tutor Profile Information:

- **Subject Selection**: subjects added by the tutors
- **Hourly Rates**: rates specified accoring to the subject and the tutor
- **Tutor Description**: Brief about the tutor

User Profile Management

- **Profile Updates**:Users can edit their profile details.

Communication

- **Messaging**: Users and tutors are supposed tom commmunicstet bt messageing one another.

Ratings and Reviews

- **Rating System**: students can rate their tutors
- **Review feature**: users can leave review for their tutors.

User Types and Access Levels

- **Silver Users (Free Tier)**:Limit users to send one message and profile viewings
- **Gold Users (Paid Tier)**: unlimited messages and profile viewings

Student realtime progress: The app should update the student progress in real time.

Matchmaking: The app should get tutor suggestions for student depends on their needs

Use Case Name	**User Registration**
XRef	1.2 User Registration for users
Trigger	The user must access the website first
Precondition	The user shall have an internet connection
Basic Path	**1**. The user must register 2. The user should enter the details required as: first name, last name, birth date 3. The user must authenticate themselves by their email address 4. The user must create a password and rewrite it again
Alternative Paths	**1. The user can register by their google account**
Postcondition	**The user now can login**
Exception Paths	**The user might have registered twice so it will not work**
Other	**N/A**

Use Case Name	Communication
XRef	1.7 Messaging
Trigger	User read the tutor description
Precondition	The user sends the tutor a brief message
Basic Path	1.User chooses to text the tutor 2.user send what subjects they're willing to learn 3. the tutor receives the message
Alternative Paths	N/A
Postcondition	The tutor starts communication with the user
Exception Paths	N/A

Use Case Name	**Rates and reviews**
XRef	1.8 Rates
Trigger	The user communicates with the tutor
Precondition	User gets into a session with the tutor
Basic Path	1. User finished all the sessions or trial session with their tutor 2. The user rates the tutor with a brief message or a star from 1-5 3. The review is visible to other people
Alternative Paths	N/A
Postcondition	Other users choose the tutor because of the review
Exception Paths	Users write a preview
Other	N/A

Non-functional requirements

- **Performance Efficiency**
  - **Response Time**: Time taken for the system to respond to a user request.
  - **Throughput**: Amount of work done by the system in a given period.
  - **Capacity**: Maximum number of users or transactions the system can handle.
- **Reliability**
  - **Availability**: Percentage of time the system is operational and accessible.
  - **Fault Tolerance**: System's ability to continue operating after failure.
  - **Recoverability**: Time and effort to restore the system after a failure.

Security

- **Confidentiality**: Protection of data from unauthorized access.
- **Integrity**: Prevention of unauthorized data modification.
- **Authentication and Authorization**: Verification of user identities and access control.

Maintainability

- **Modularity**: Degree to which components of a system can be separated and recombined.
- **Reusability**: Use of system components in different applications.
- **Analyzability**: Ease with which the system can be analyzed for errors or improvements.
- **Testability**: Ease of testing the system to verify functionality and performance.

Usability

- **Learnability**: How easily users can learn to use the system.
- **Operability**: The ease with which users can operate the system.
- **User Error Protection**: The system's ability to prevent user mistakes.

Portability

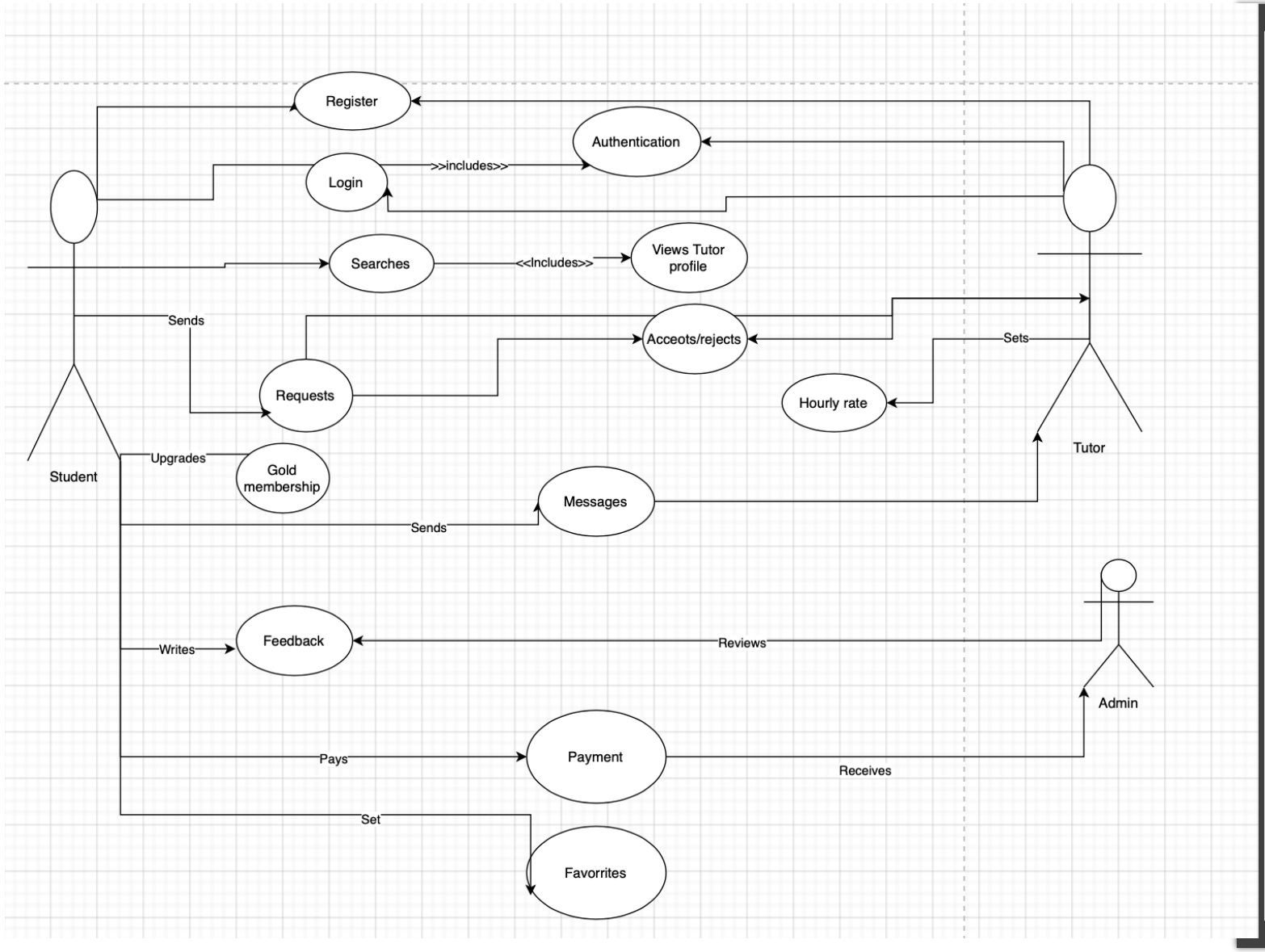
- **Adaptability**: The ability to adapt the software to different environments.
- **Installability**: Ease of installing the software in different environments.
- **Replaceability**: Ease of replacing the software with another.

Accessbility everywhere: The website should be accessed on any device with any software

Use Case Name	Performance efficiency
XRef	2.1Performance
Performance	The system should experience a high volume of users
Requirement	System should handle 700 users using it at the same time and 200 transactions
Constraints	1. Request should be handled evenly 2. System should be working without any lagging no matter how many numbers of users
Verification method	Simulate user requests
Notes	N/A

Use Case Name	Reliability
XRef	2.2 Reliability
Requirement	1. The system should respond to 95% of user requests within 2 seconds 2. The app should process up to 1000 transactions per minute 3. The system must support up to 500 concurrent users
Constraints	1.The app should have backup resources
Verification method	1. Node simulation to verify any fault 2. Testing recovery times
Notes	N/A

XRef	2.3 Security
Scenario	Secures the users data
Requirement	1. All user data must be encrypted using AES-256 encryption. 2. The system should log any unauthorized attempts to modify sensitive data. 3. All users must authenticate using two-factor authentication.
Constraints	1. Data security shouldn't affect the app 2. Security testing 3. Logs should be secured
Verification method	1. testing security features
Notes	N/A



USE CASE DIAGRAM

Constraints

Data security:

- GPDR Compliance: ensures data collection
- Encryption: the user data shall be protected

User access:

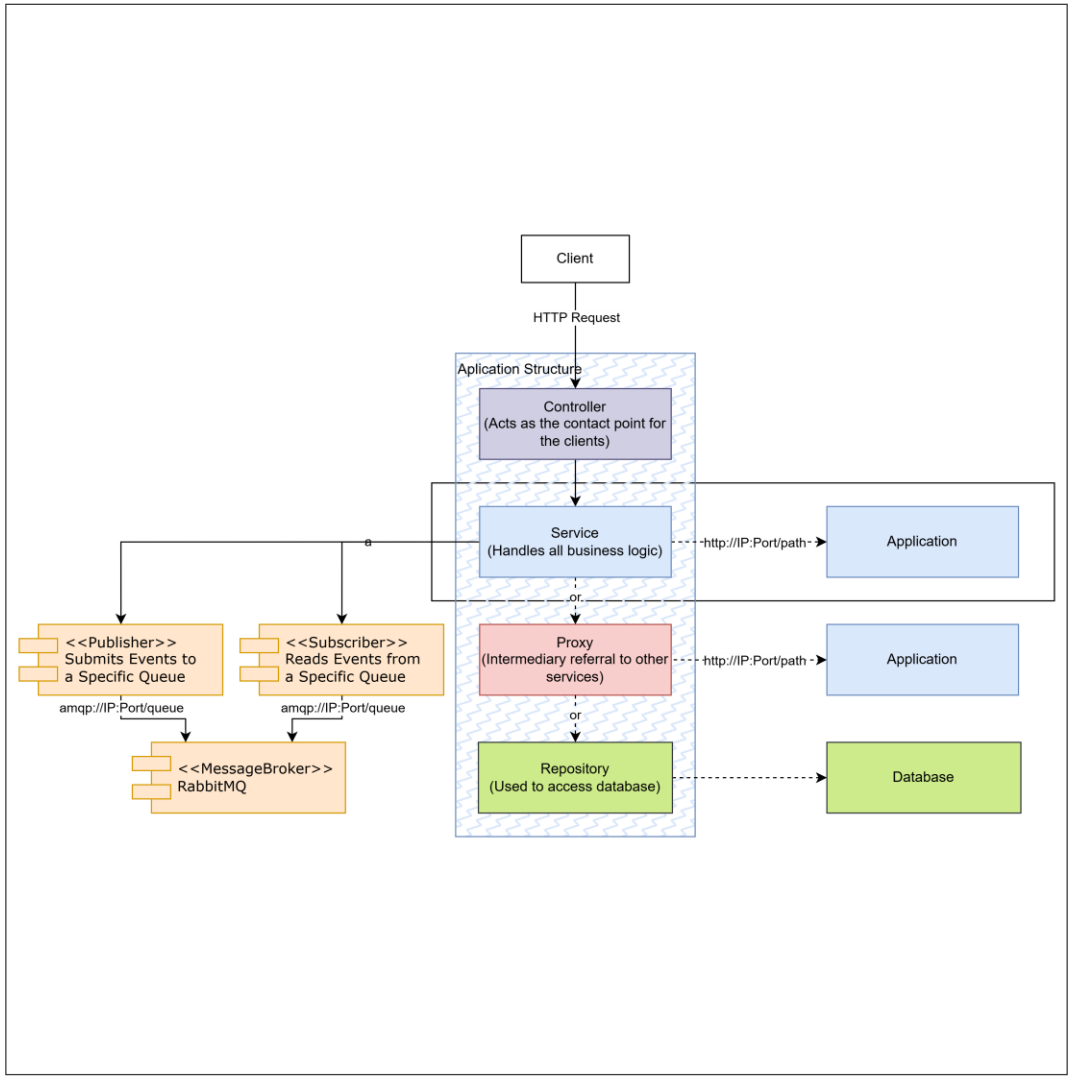
- User level: Silver users get limites functionalities while gold ones gets unlimited.
- Portability: Ensures that the website can be accessed on every device

Data mangement: data should be backed up every 24 hours

Responisve deisgn: the design should be accessed via different screen sizes

User support: Provide user support for users 24/7

SYSTEM DESIGN



Architectural Style

- Layered Architecture
- Multi-Tenant Architecture
- Microservices Architecture
- Client-Server Architecture
- Event-Driven Architecture

Design Choices

Session-Based-Authentication

Authentication-Layer stores SessionID.

SessionID is used to retrieve user data.

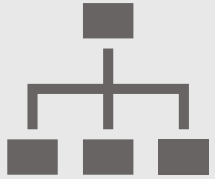
Allows for implementation with Google's OAuth v2.0

Session Token stored securely in client.

OAuth 2.0 Integration

Support third-party logins.

Uses Session Tokens, implementable with Session-Based-Authentication.



Role-Based Access Control

Segregation of User Responsibility.

Granural Control of Feature Access.

Simplified User Management via User Groups.

Design Choices

PROXY PATTERN

REPOSITORY PATTERN

DATABASE-PER-SERVICE PATTERN

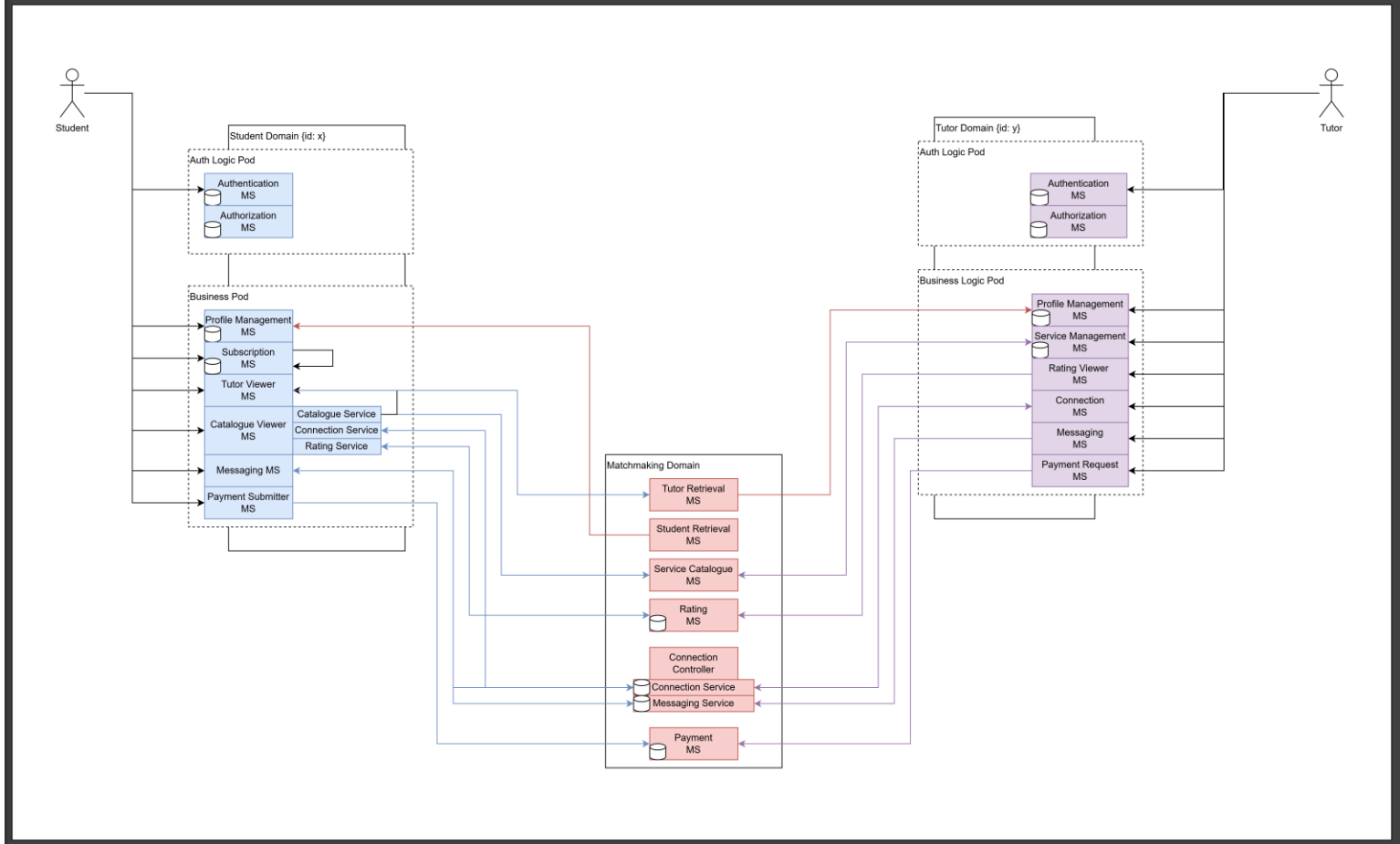
STATIC ARCHITECTURE

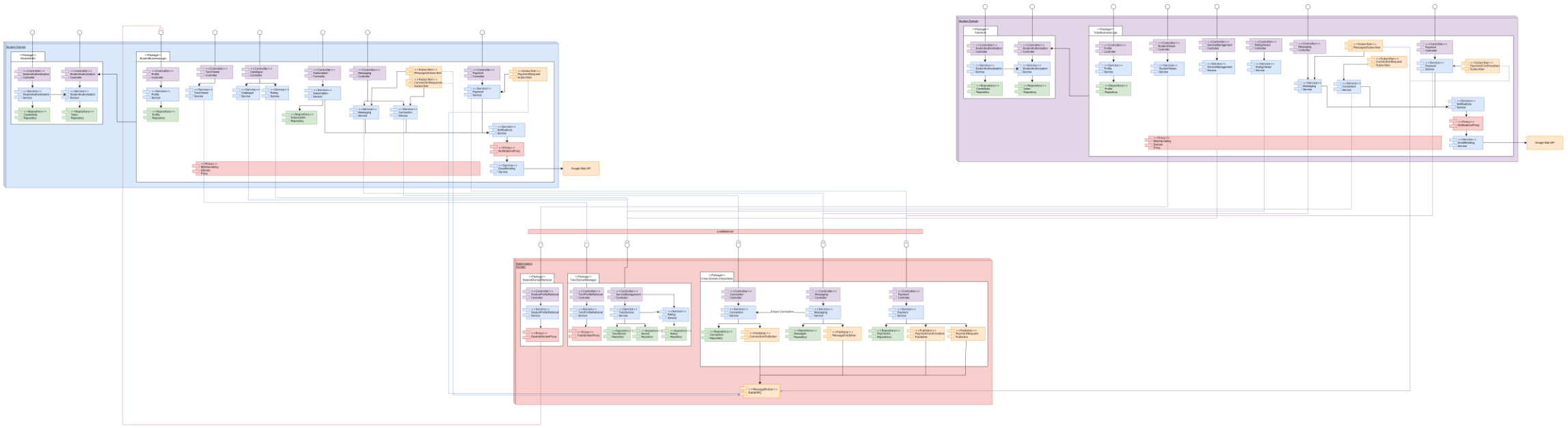
Component Diagrams

https://drive.google.com/file/d/1PVj5r5ewWK62QK9xlNflBjKJaIPgiDx4/view?usp=sharing

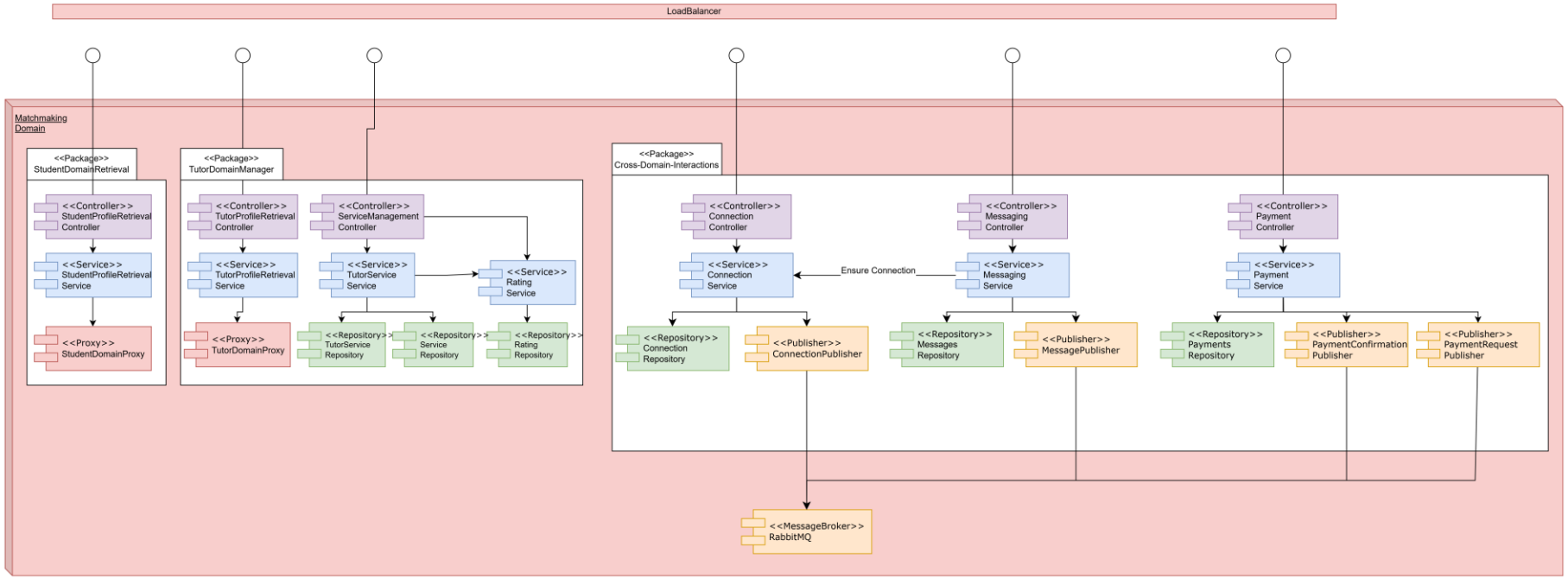
COMPONENT DIAGRAM – SIMPLIFIED

3-DOMAIN PRINCIPLE

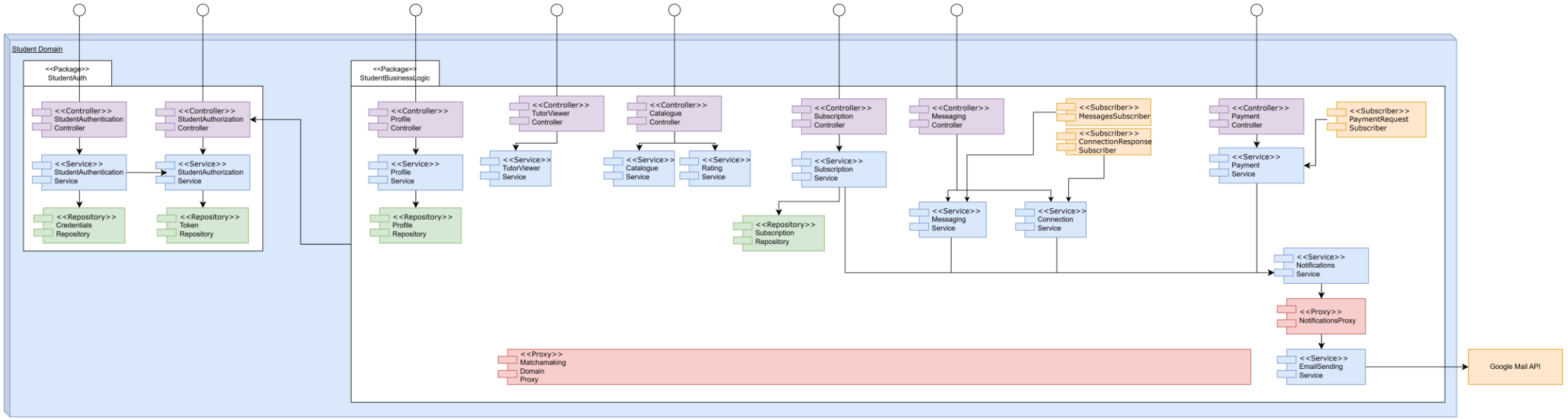




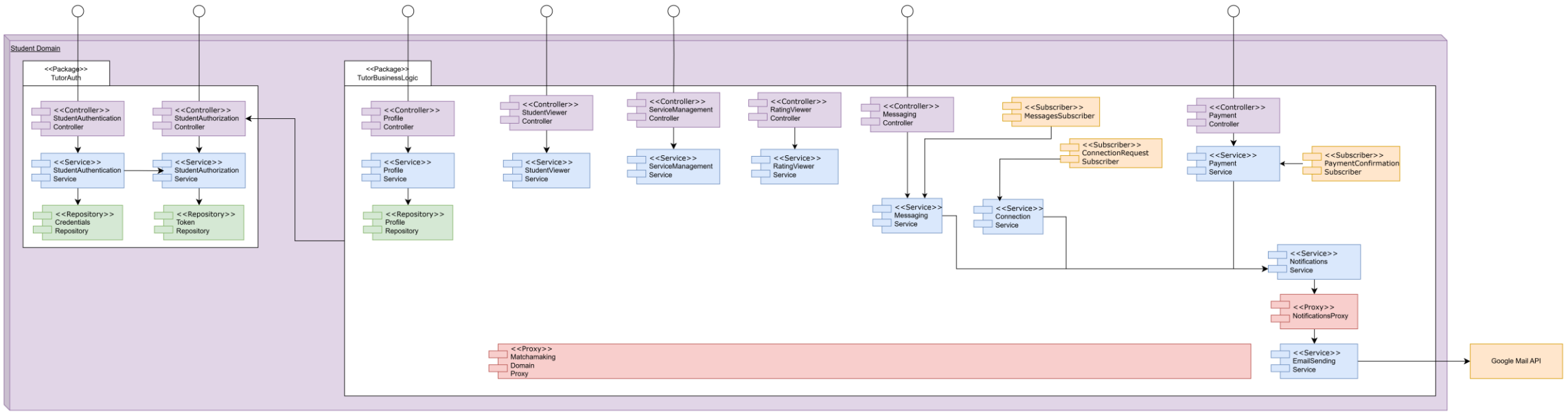
COMPONENT DIAGRAM – COMPLETE



COMPONENT DIAGRAM – MATCHMAKING DOMAIN



COMPONENT DIAGRAM – STUDENT DOMAIN

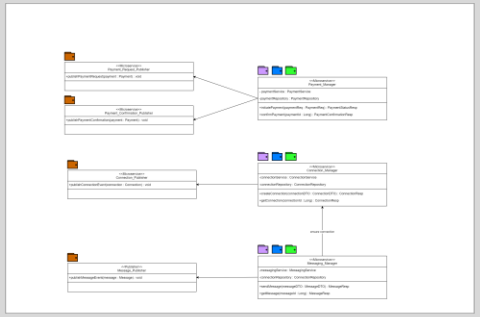


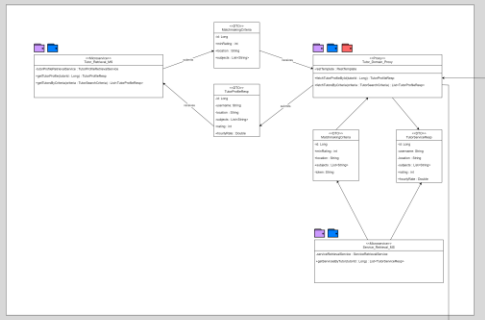
COMPONENT DIAGRAM – TUTOR DOMAIN

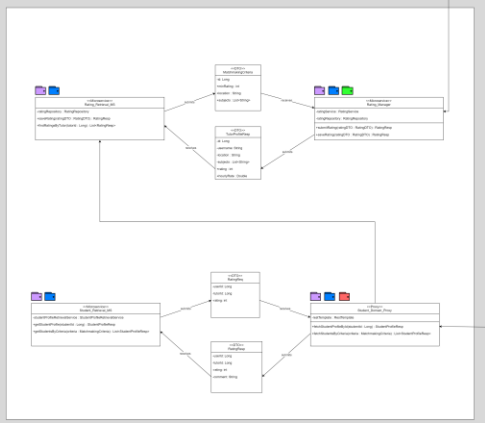
STATIC ARCHITECTURE

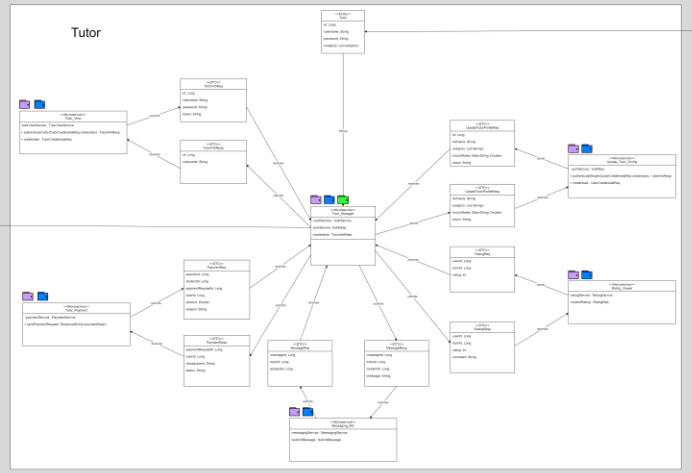
Class Diagrams
https://drive.google.com/file/d/1niVgLKVa9R_M1vTGFtiNROoErm3rIv9J/view?usp=sharing

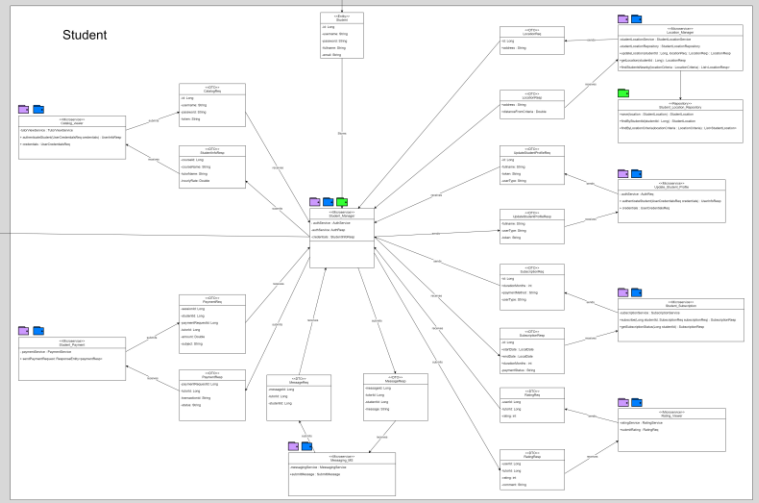
Full Class Diagram

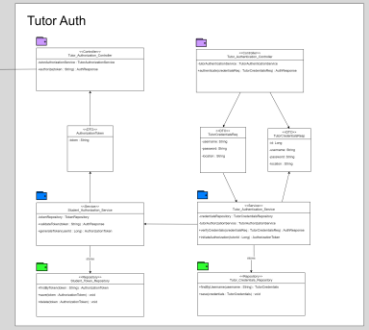


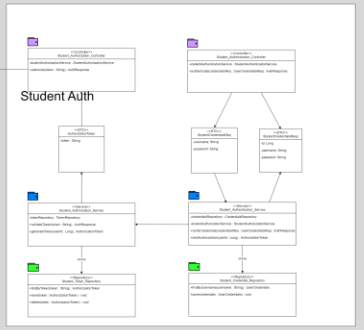








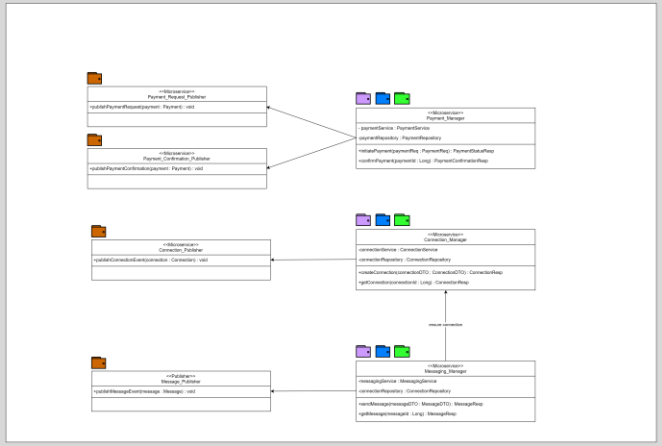


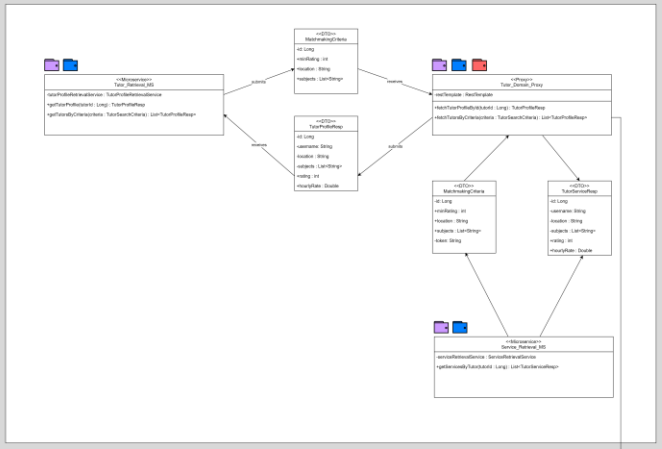


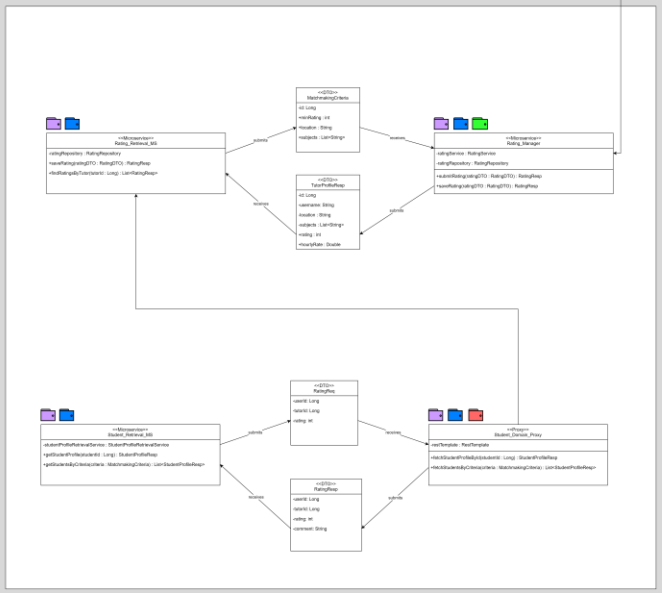
[Full Resolution Diagram]

Matchmaking Domain

- Manages connections
- Connects domains
- Manages Payments
- Manages Messaging

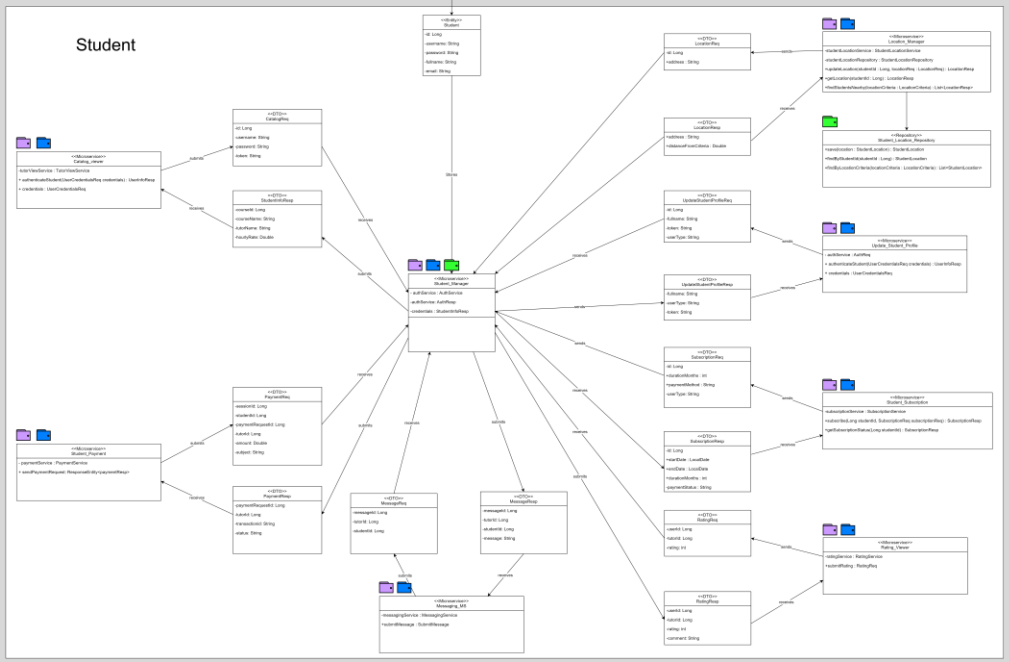


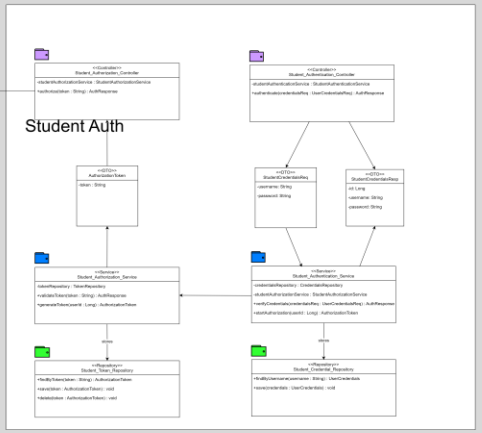




Student Domain

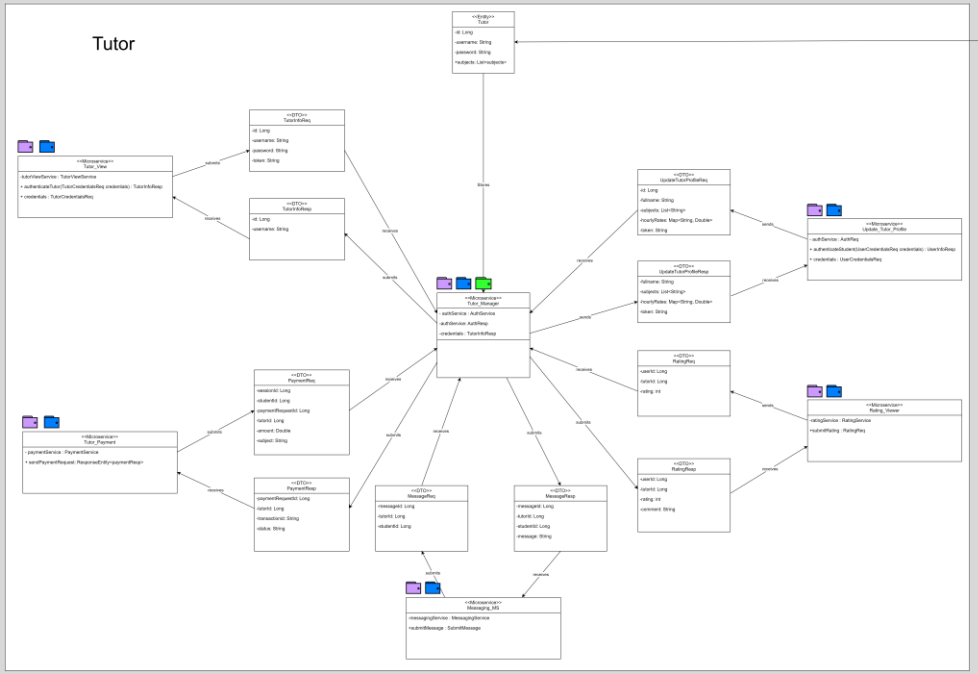
- Manages student profiles (keeps track of name, subjects, etc.)
- Keeps record of student location
- Manages student login and permissions

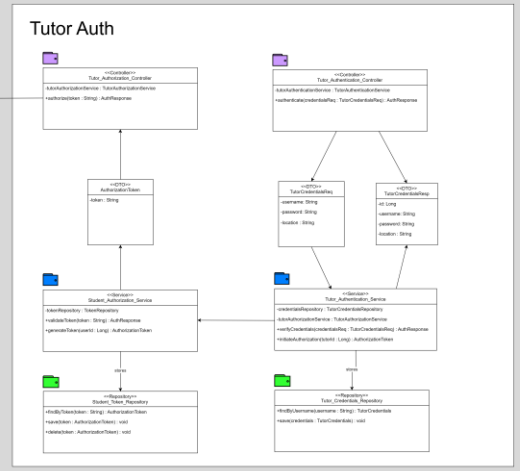




Tutor Domain

- Manages tutor profiles
- Collaborates for matchmaking
- Lists tutor services





POST /tutor/auth/register Tutor registration

POST /tutor/auth/login Tutor login

Catalog Controller Catalog of Courses

GET /catalog/courses View Course Catalog

GET /catalog/courses/search Search Courses

Intermediary Payment Controller Intermediary controller handling payment requests

POST /payments/intermediary/forward Forward payment request

Tutor Payment Controller Tutor sends payment requests

POST /payments/tutor/request Send payment request

Tutor Controller Tutor profile management endpoints

PUT /tutors/{tutorId}/subjects Select or modify tutor subjects

GET /tutors/{tutorId}/profile Retrieve tutor profile

PUT /tutors/{tutorId}/profile Modify tutor profile

DELETE /tutors/{tutorId}/delete Delete tutor account

API OVERVIEW

HTTPS://JRAMIREZDD.GITHUB.IO/TUTORLINK/DEVELOPMENT/API/

OpenAPI definition v0 OAS 3.0

swaggerStuff.json

Servers

http://localhost:8080 - Generated server url

Tutor Auth Controller Tutor Authentication and Authorization Endpoints

POST /tutor/auth/register Tutor registration

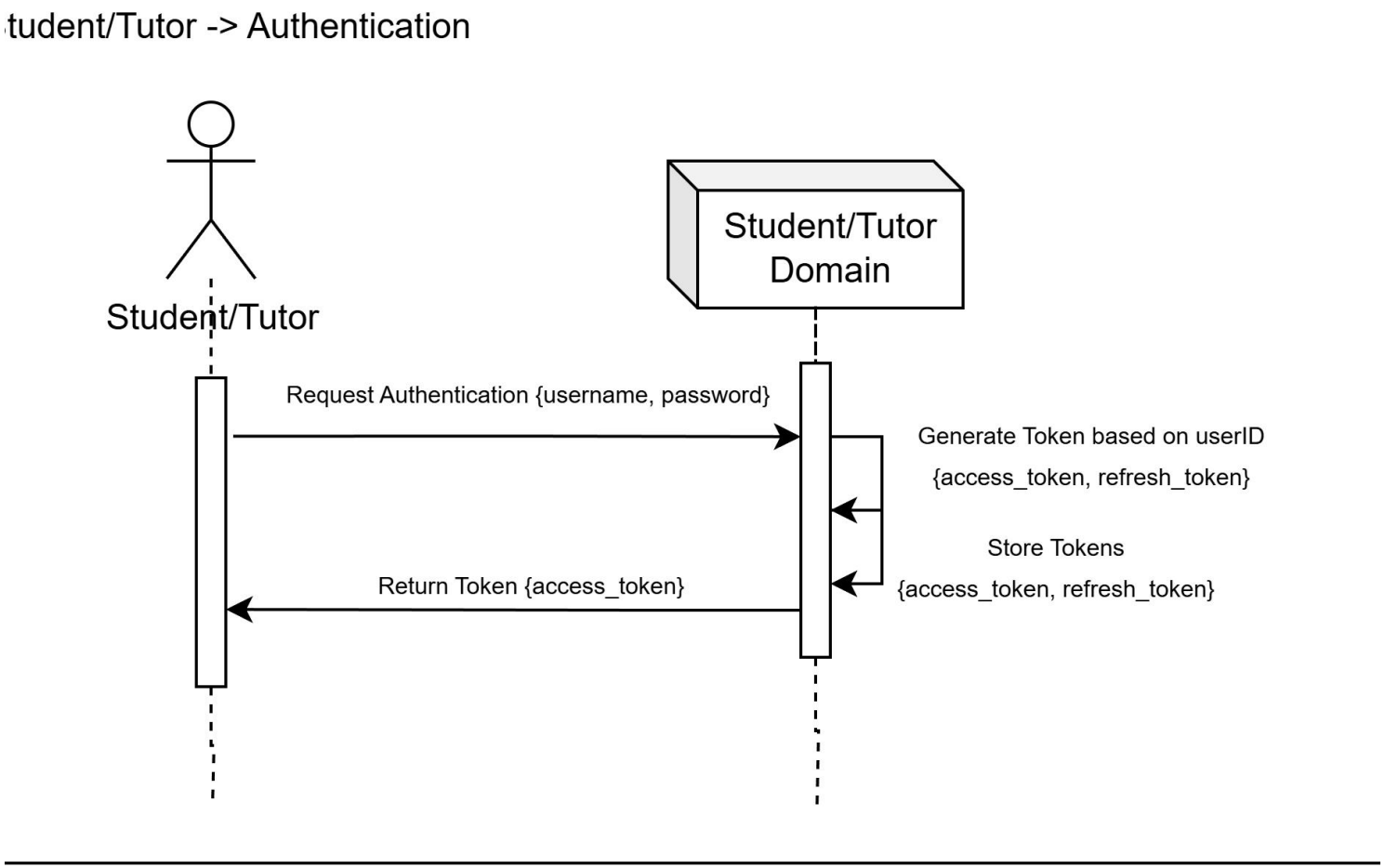
POST /tutor/auth/login Tutor login

SWAGGER DOCS

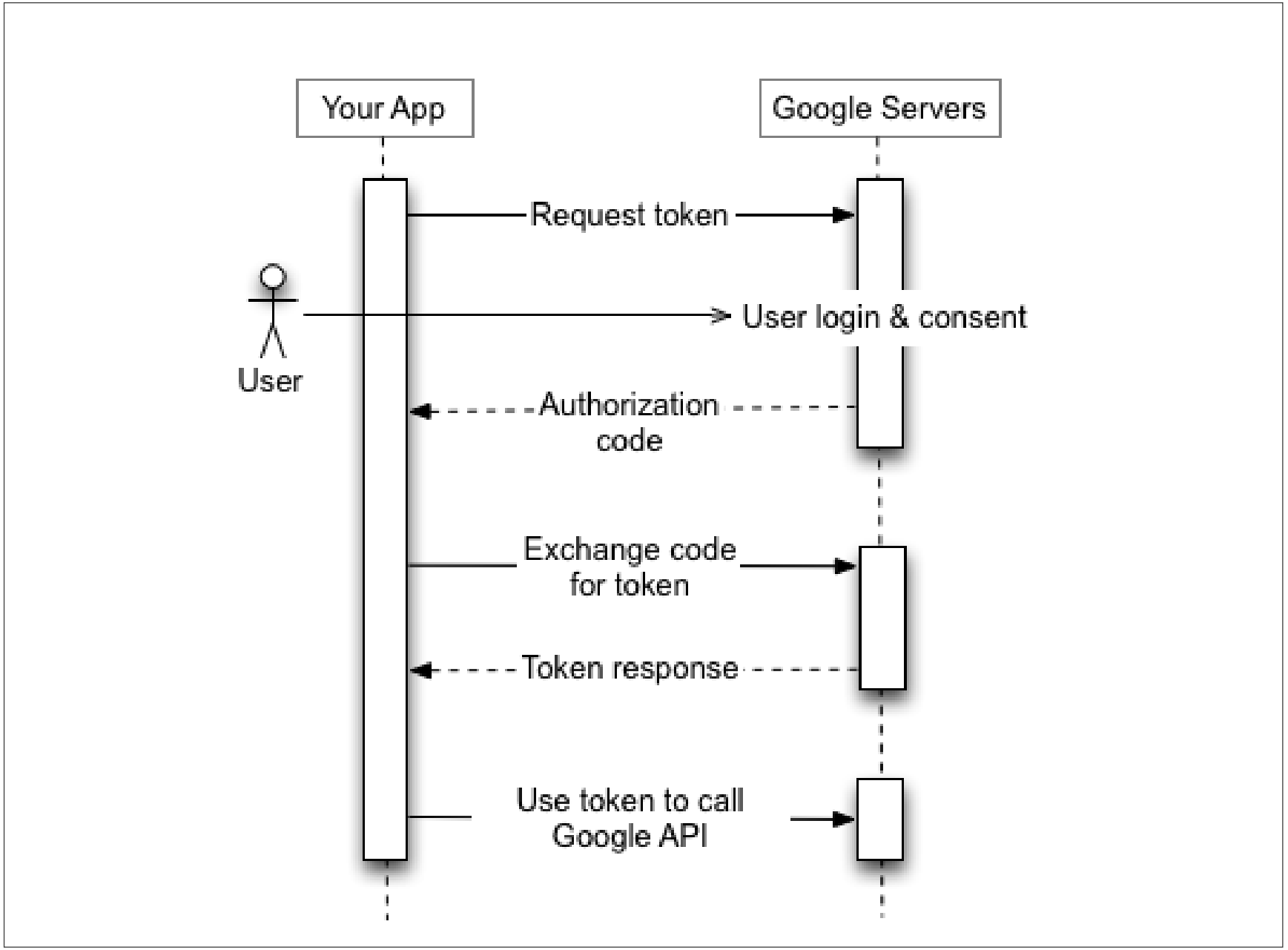
DYNAMIC ARCHITECTURE

Sequence Diagrams
https://drive.google.com/file/d/1ONL8x6l7vIXcOL_NI3w9PK9b_mENpOEH/view?usp=sharing

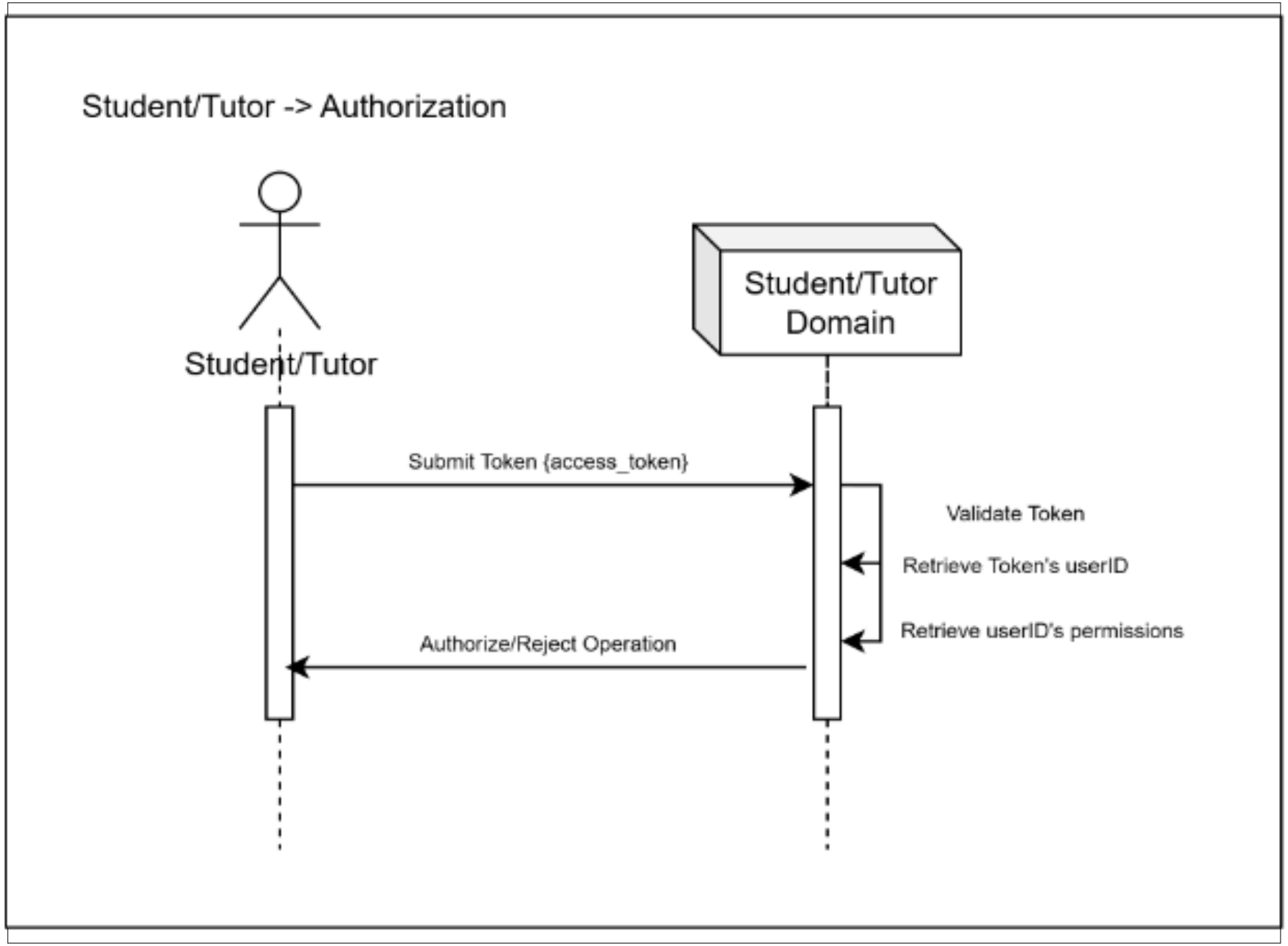
Student/Tutor: Authentication



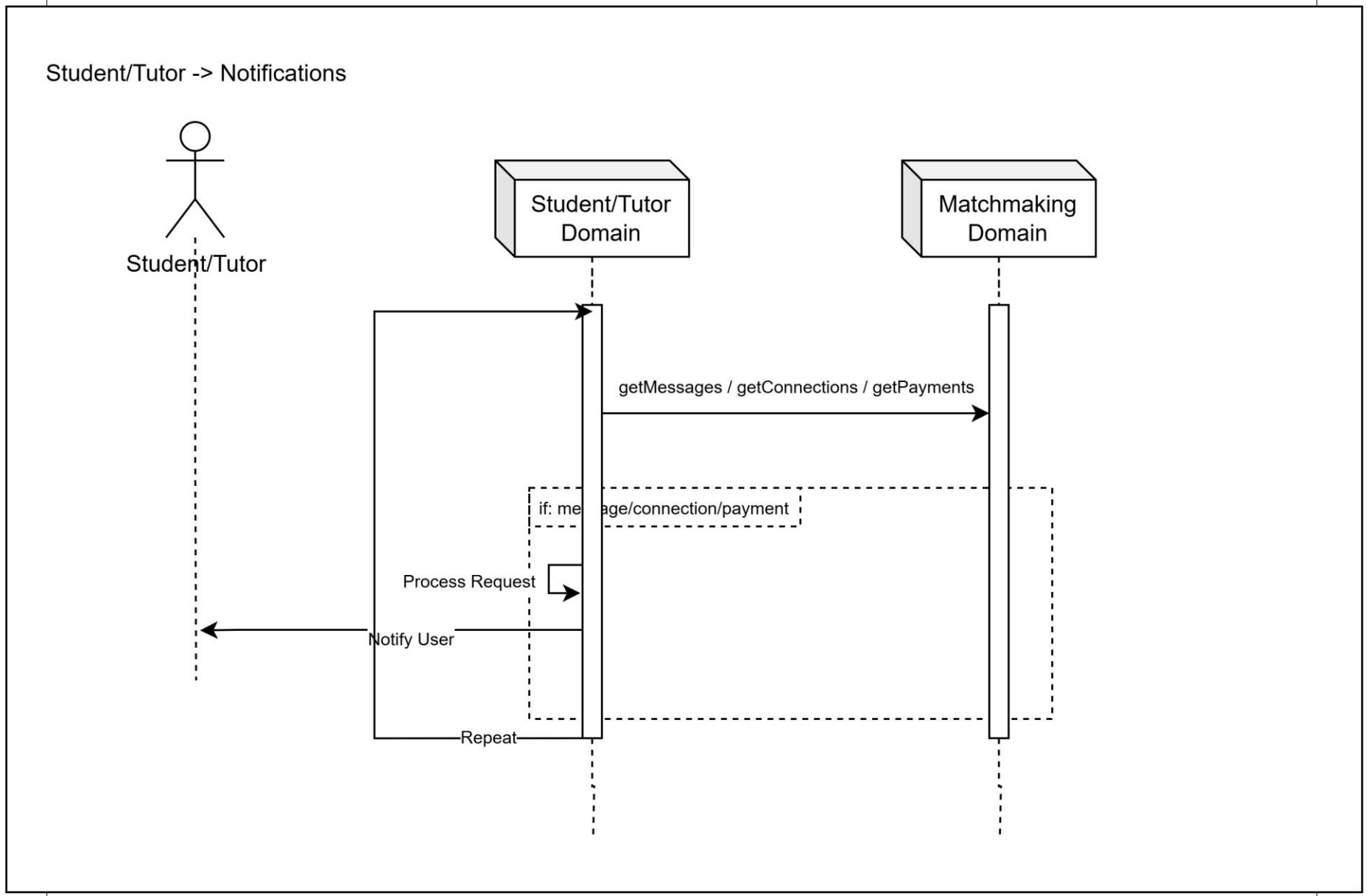
Student/Tutor: Authentication via Google's OAuth 2.0 API



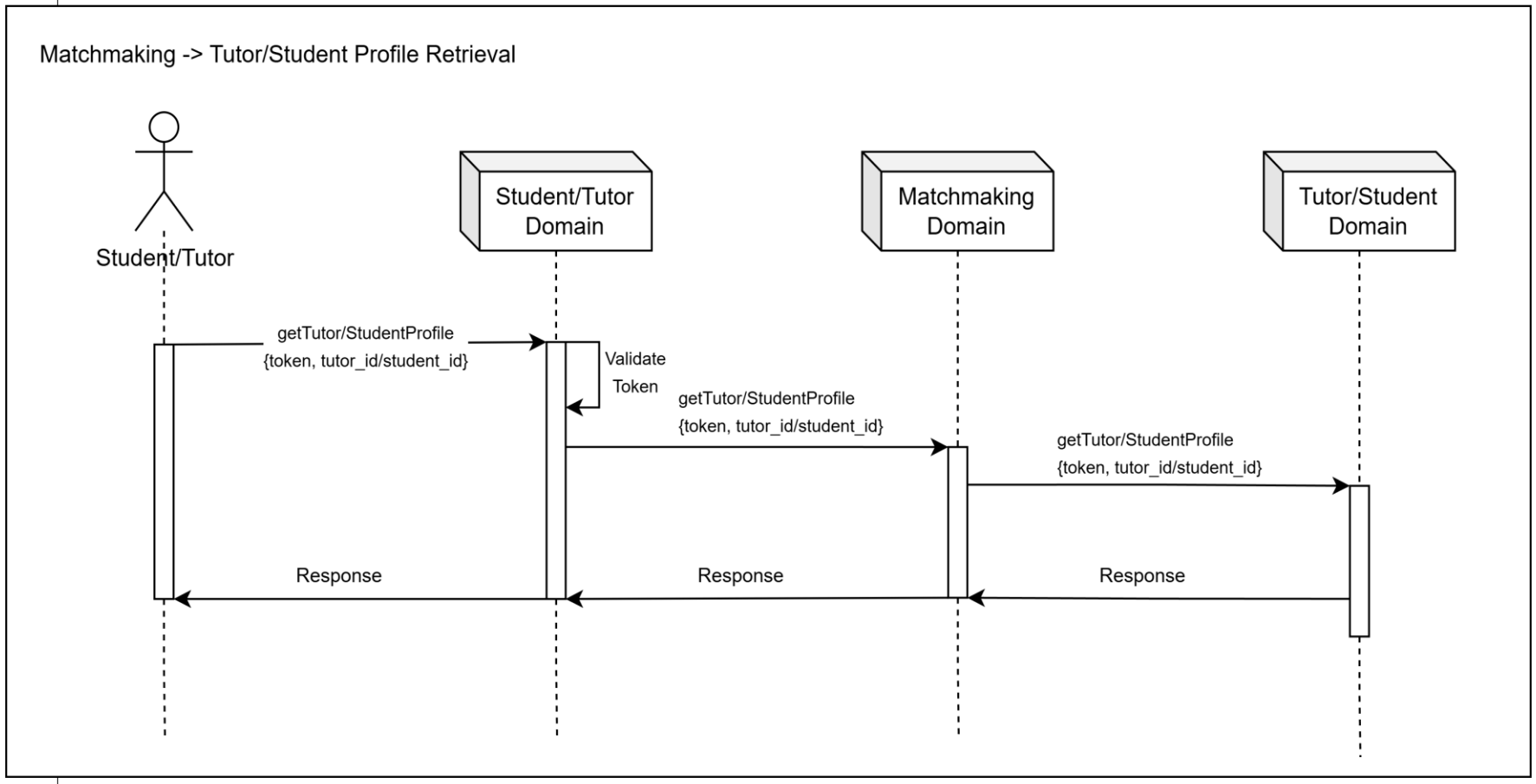
Student/Tutor: Authorization



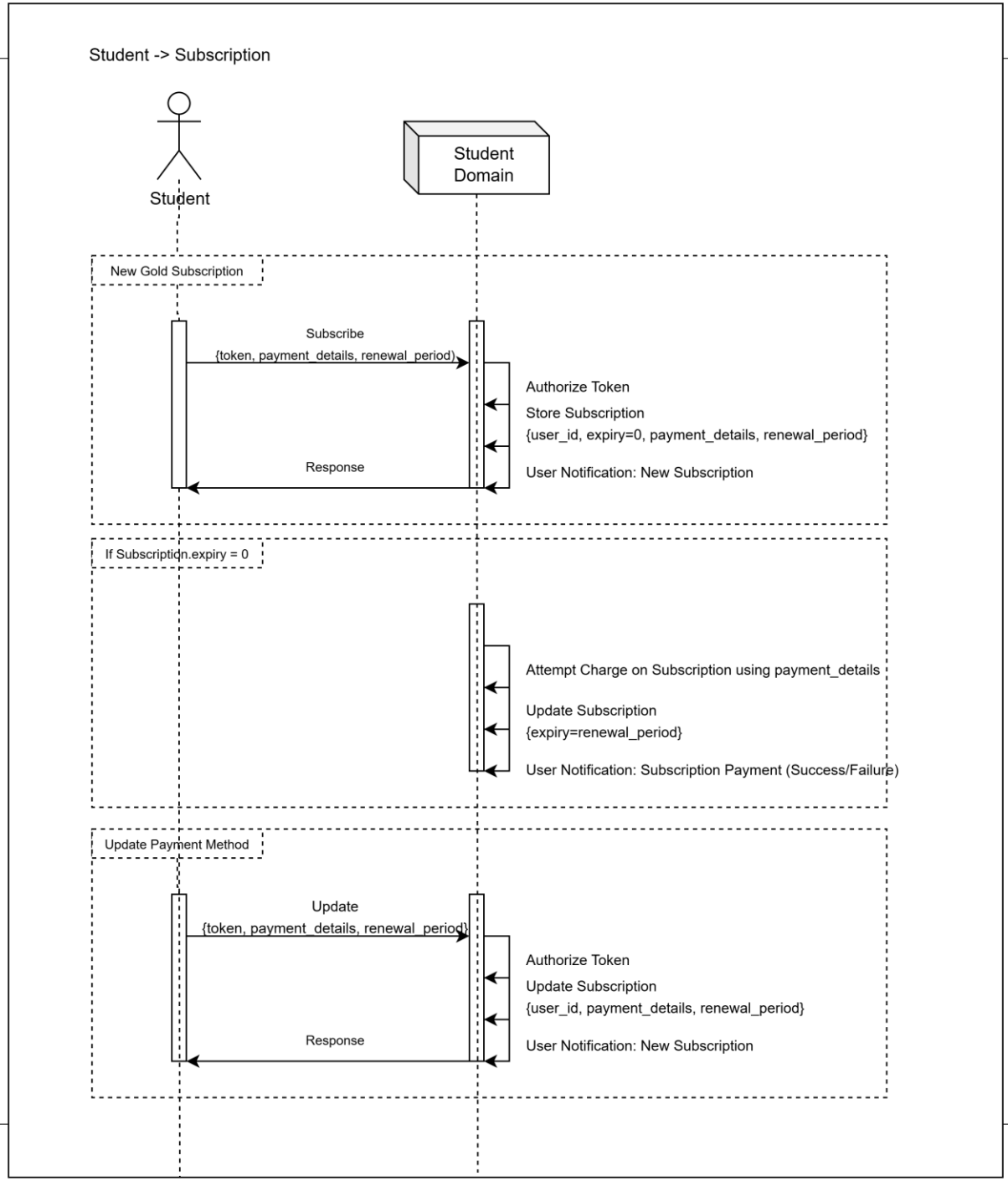
Student/Tutor: Notifications



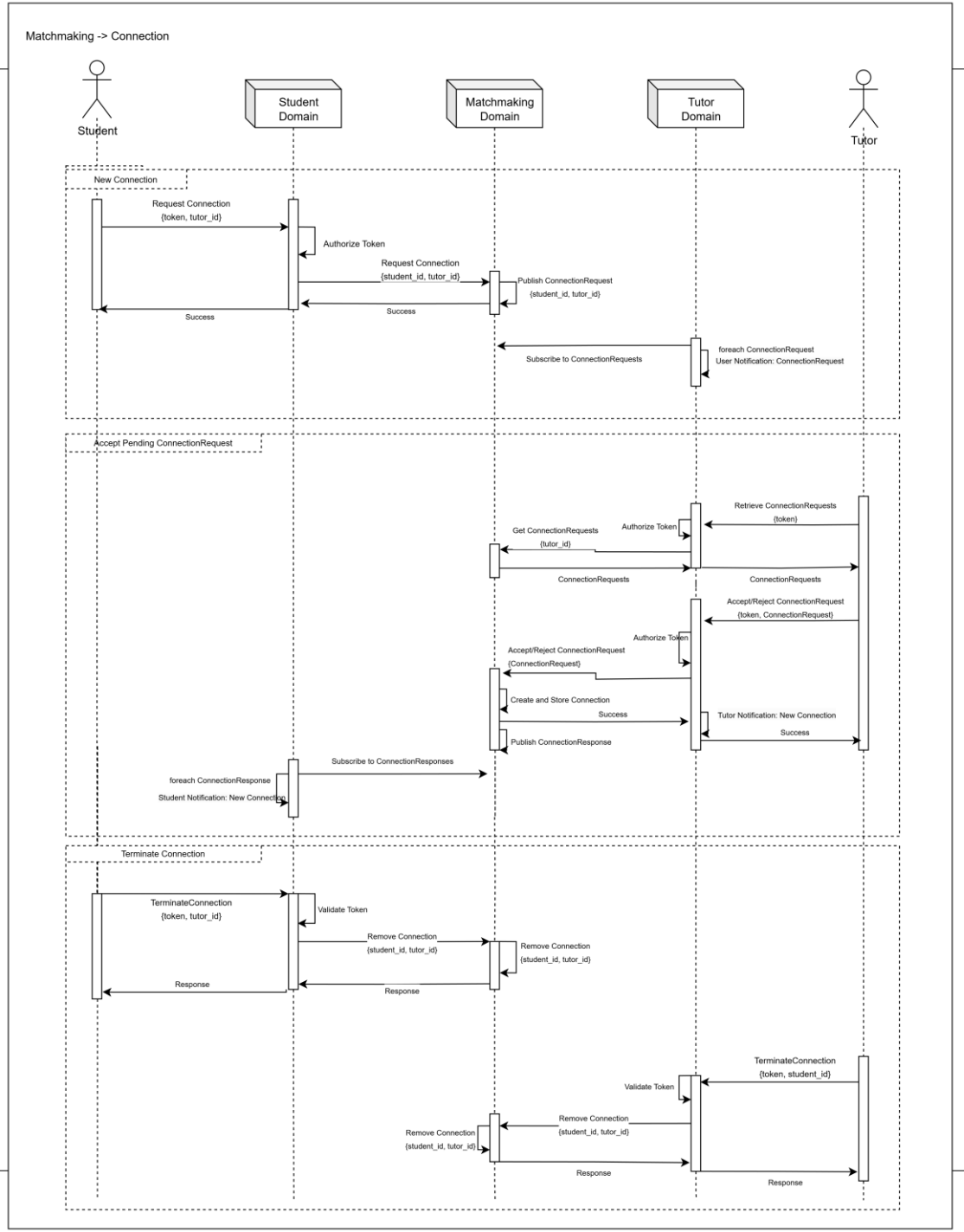
Matchmaking Domain: Tutor/Student Profile Retrieval



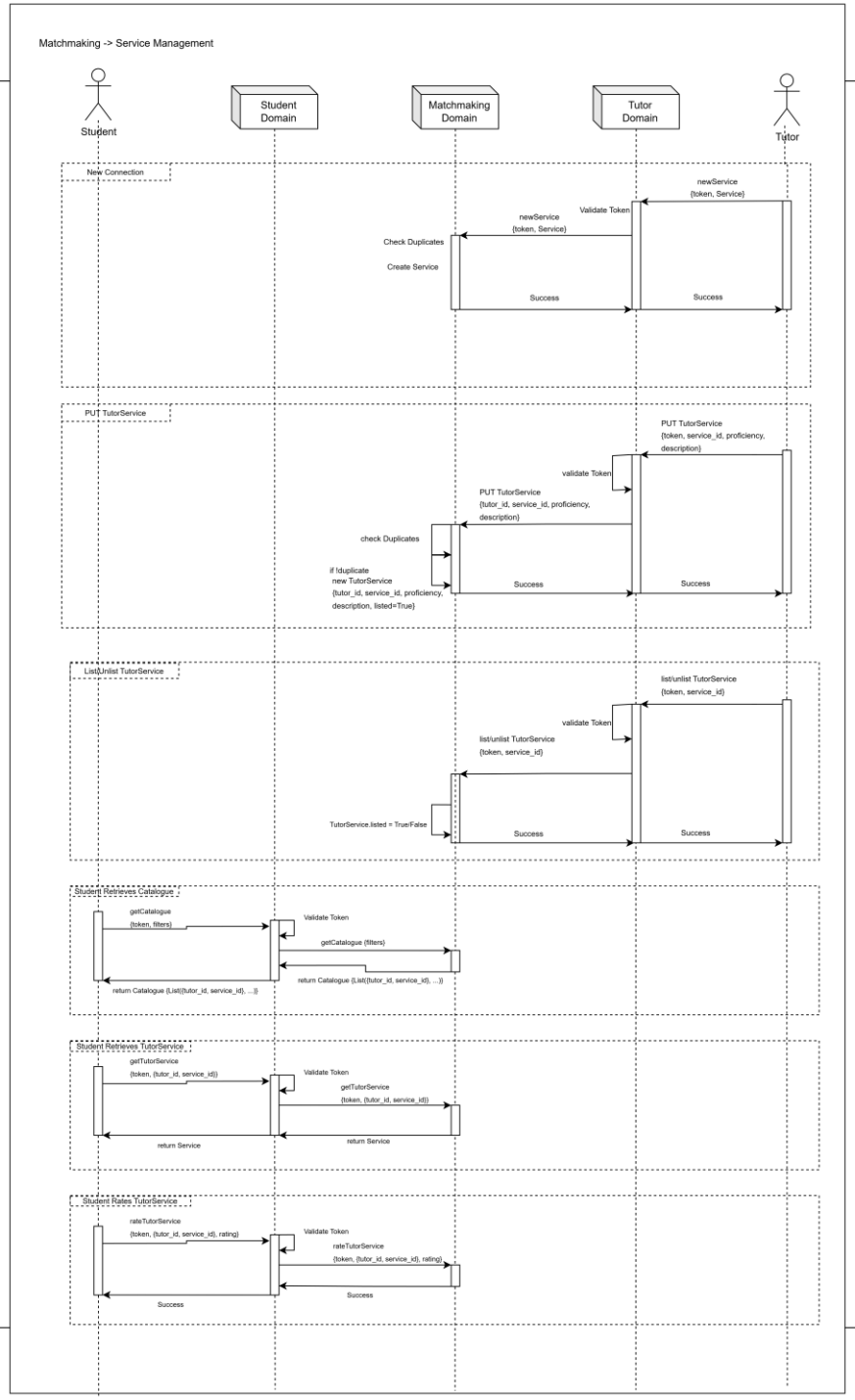
Student Domain: Subscription



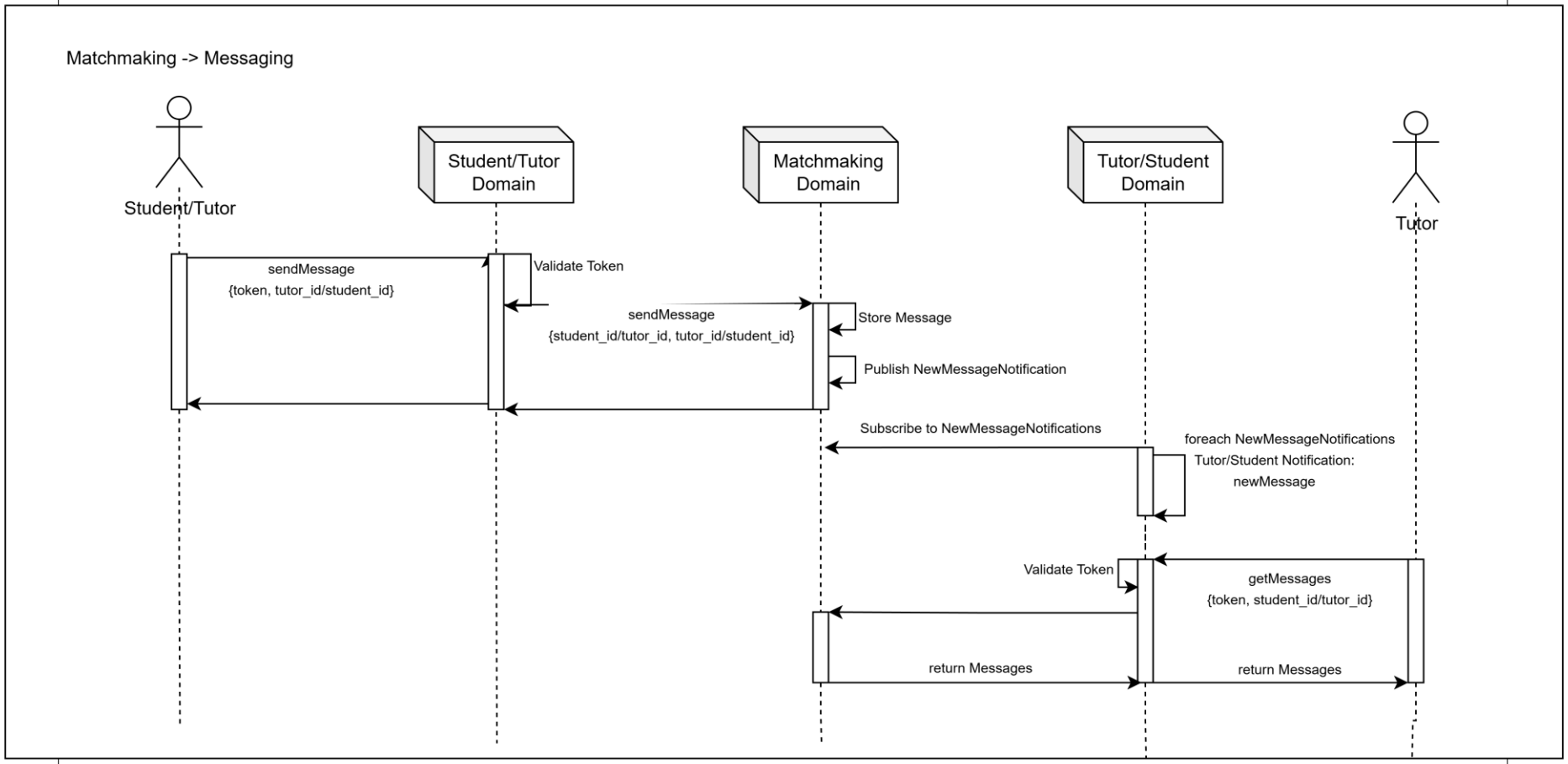
Matchmaking Domain: Connection



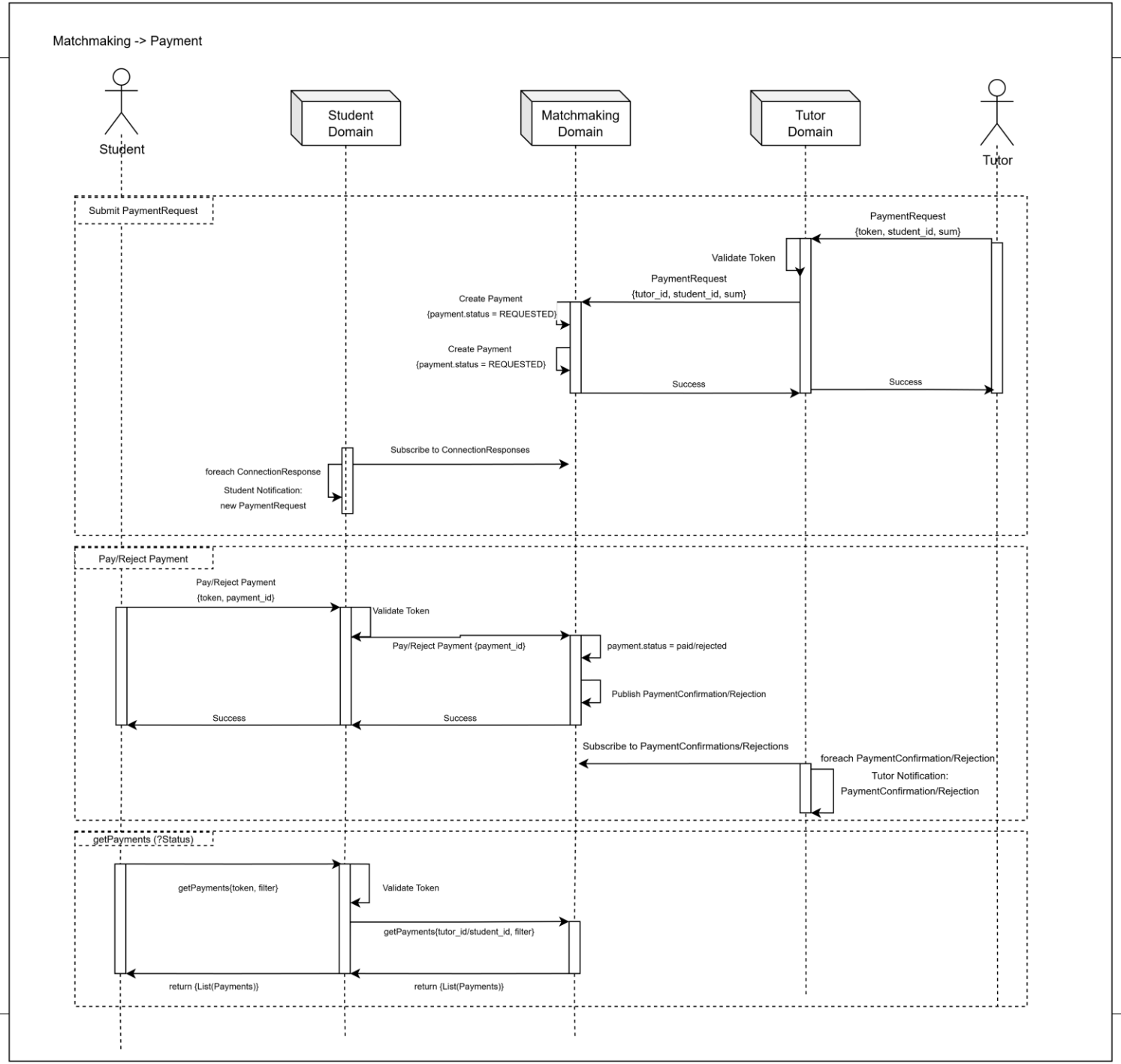
Matchmaking Domain: Service Management



Matchmaking Domain: Messaging



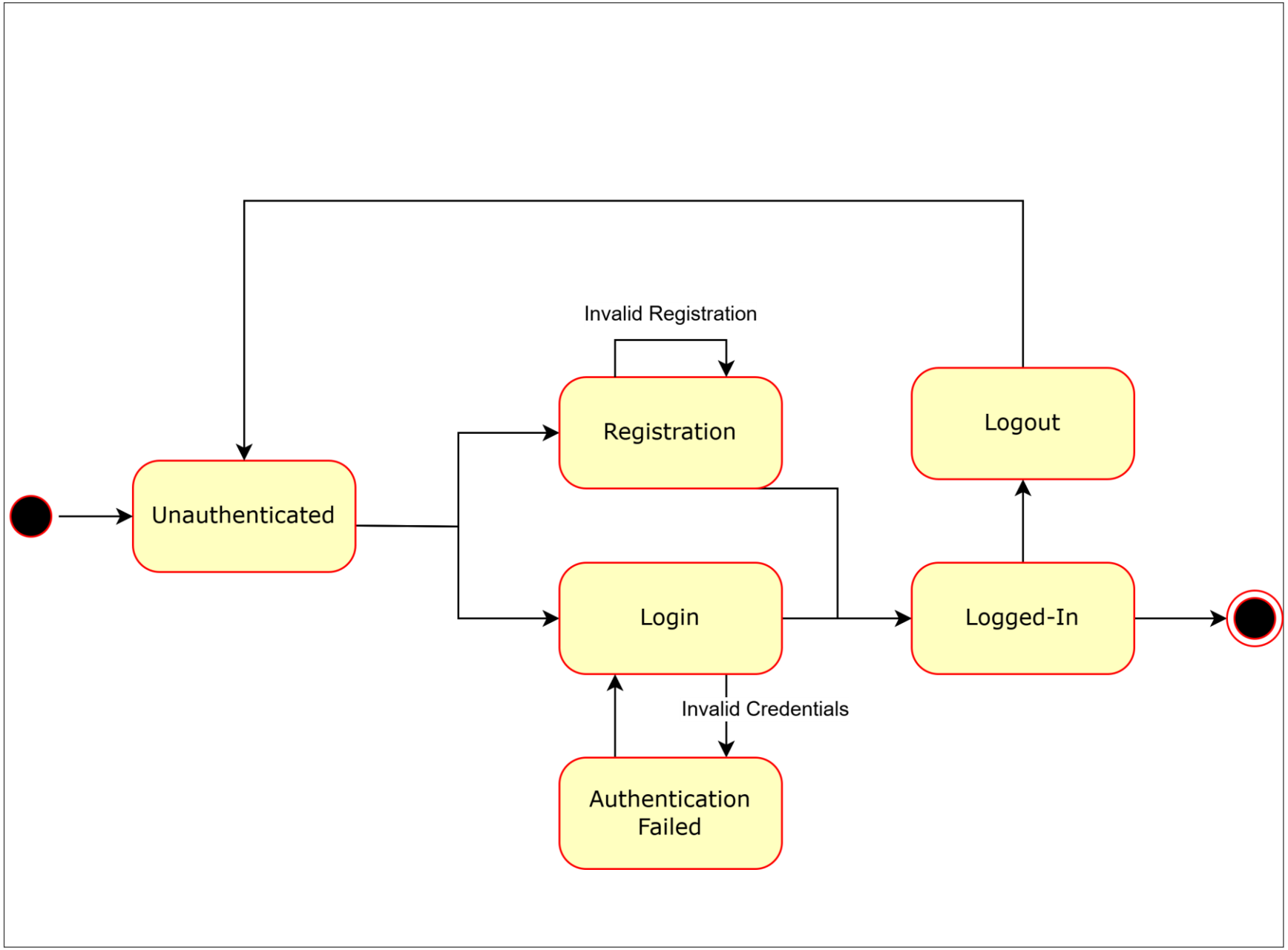
Matchmaking Domain: Payment



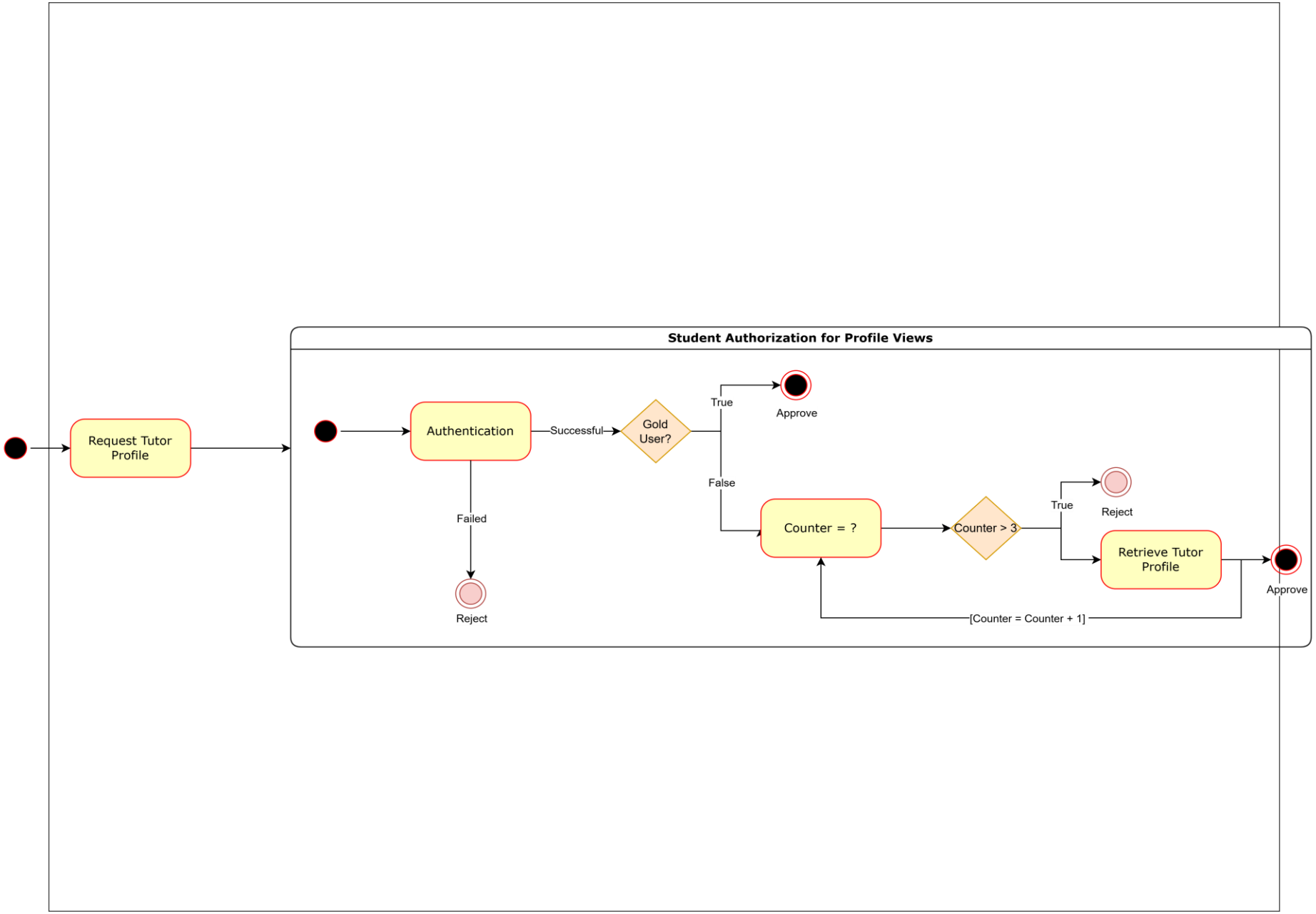
DYNAMIC ARCHITECTURE

State-Machine Diagrams
https://drive.google.com/file/d/1yUlqQQ4kG3UiwkVJw0CO0xhTI5UG40xc/view?usp=sharing

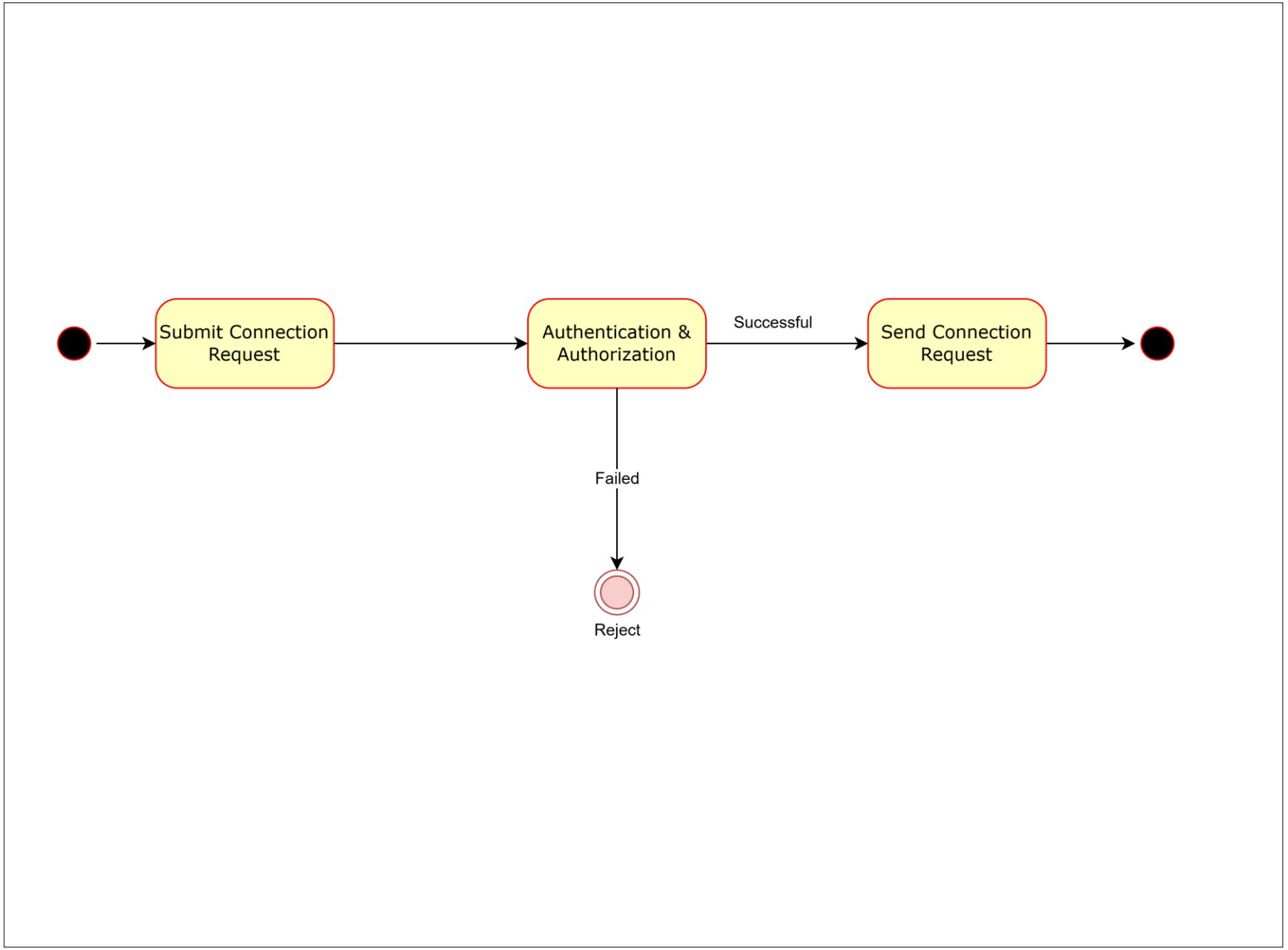
Student/Tutor Domain: Authentication



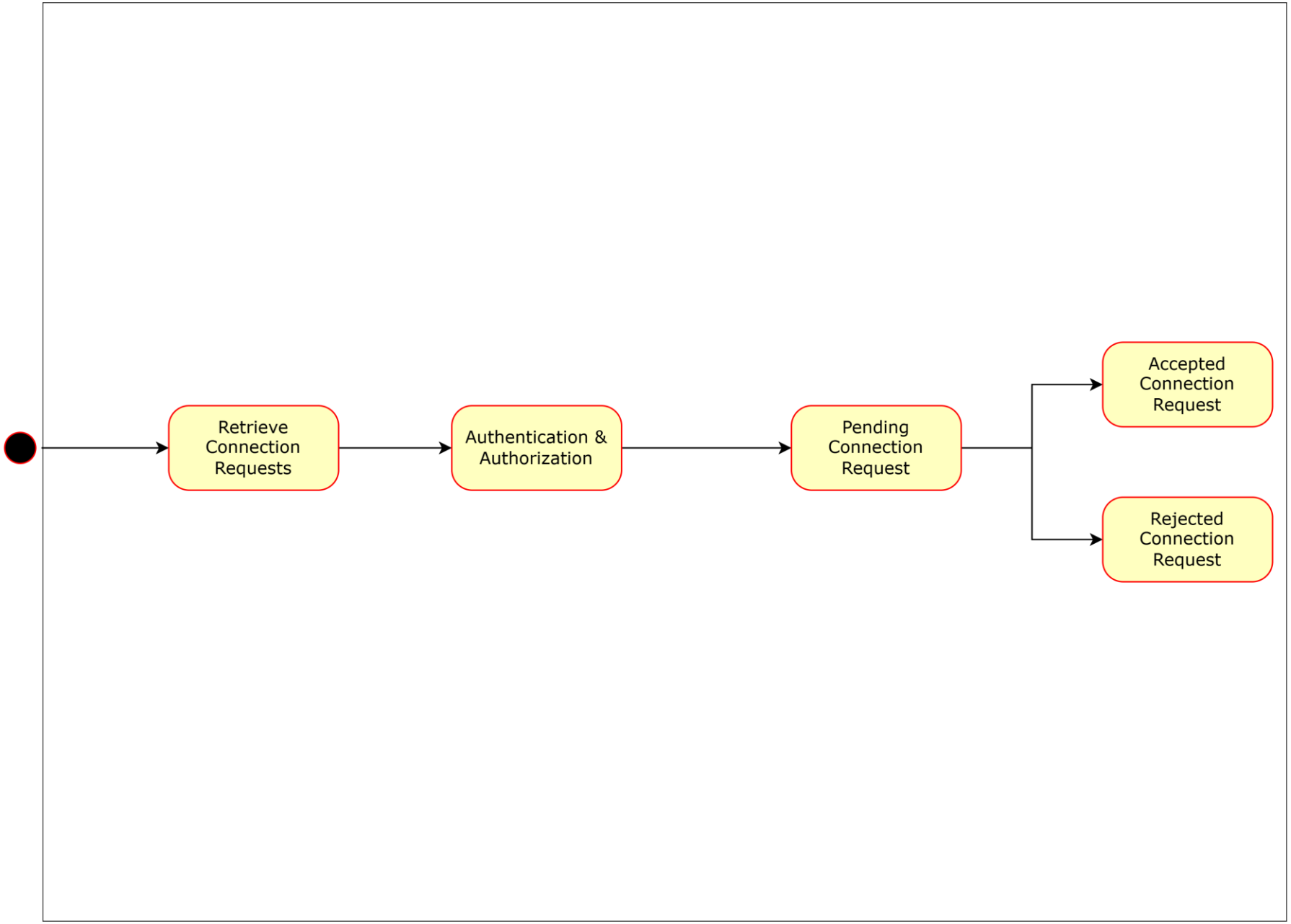
Student Domain: View Tutor Profiles



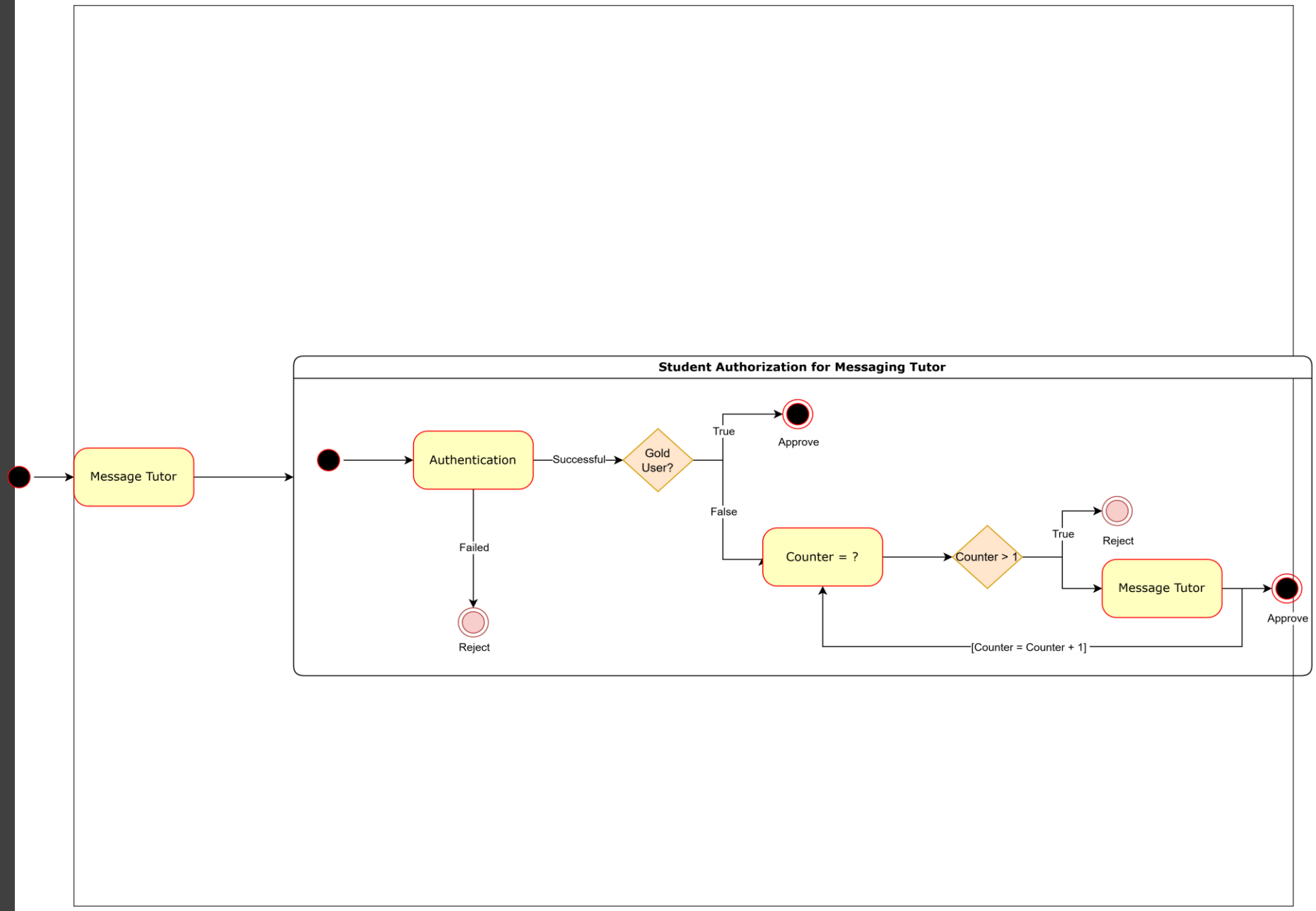
Tutor/Student Domain: Submit Connection Request



Tutor/Student Domain: Accept Connection Request



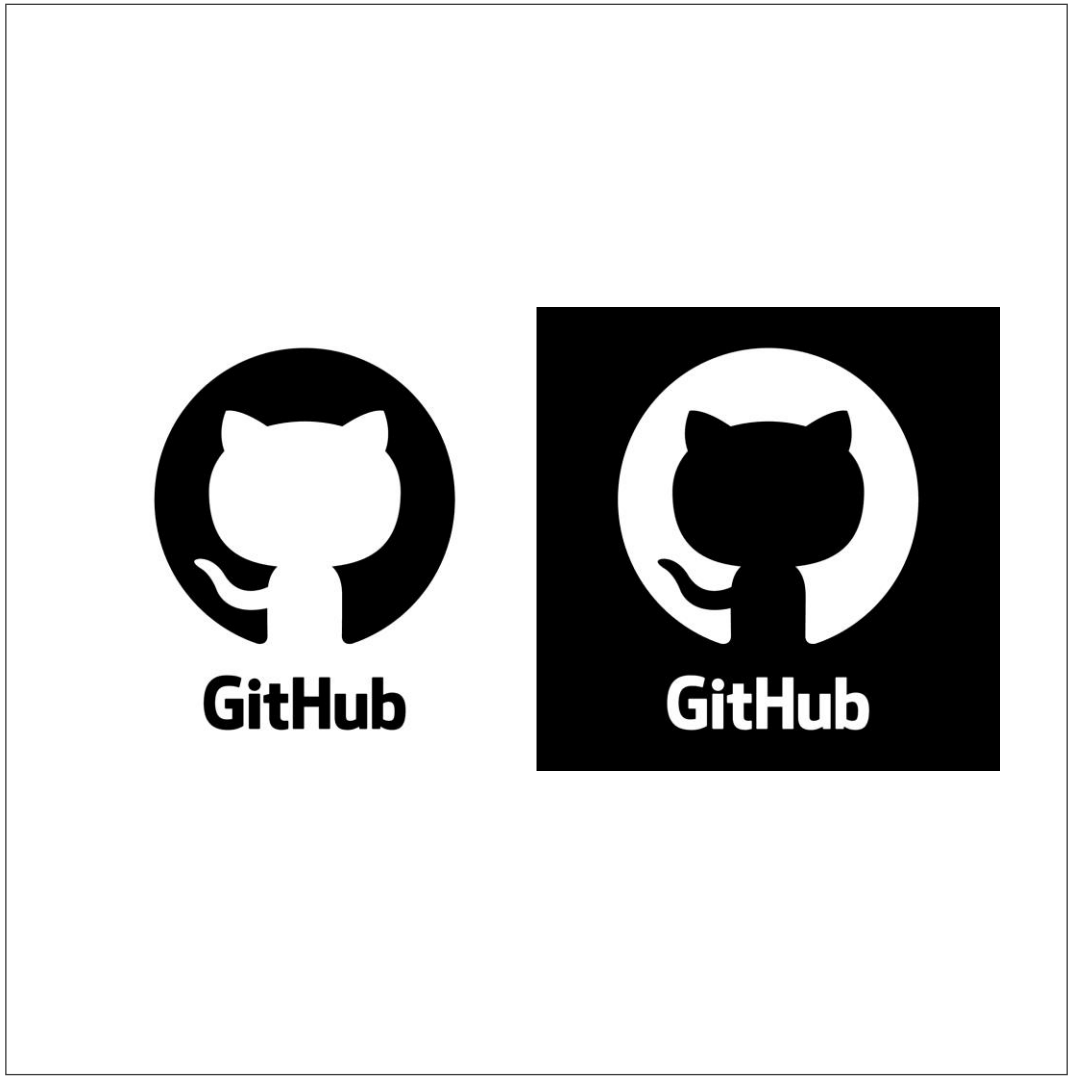
Student Domain: Message Tutor



TOOLS AND TECHNOLOGIES

Jira

- Simple task management
- Enables team communication
- Agile workflow
- Progress tracking



GitHub

- Reliable version control system
- Branching/Merging & Pull requests
- Online Hosting of Documentation via GitHub Pages
- Automation of CI/CD Jobs via GitHub Actions

HackMD

- Collaborative documentation
- Markdown support
- Tracks document changes

HackMD

Swagger

- Automated API documentation
- Interactive UI
- Clear request/response parameters/status codes
- Consistent/Standardized documentation

Flutter

- UX/UI design
- Real time updates
- Responsive layouts
- Data handling
- Backend integration

Spring

- Provides reliable microservices framework
- RESTful APIs
- Scalability

RabbitMQ

- Message broker of choice
- Reliable messaging service
- Manages and prioritizes message queues

Proxies

- Make inter-service communication easy
- Enhance security (by acting as intermediaries)
- Separation of concerns maintained
- "Gateway" for fast routing

Docker

- Packages microservices into their own containers
- No need to have MS running on separate OS kernels
- Resource efficient
- Ensures identical environments across most systems