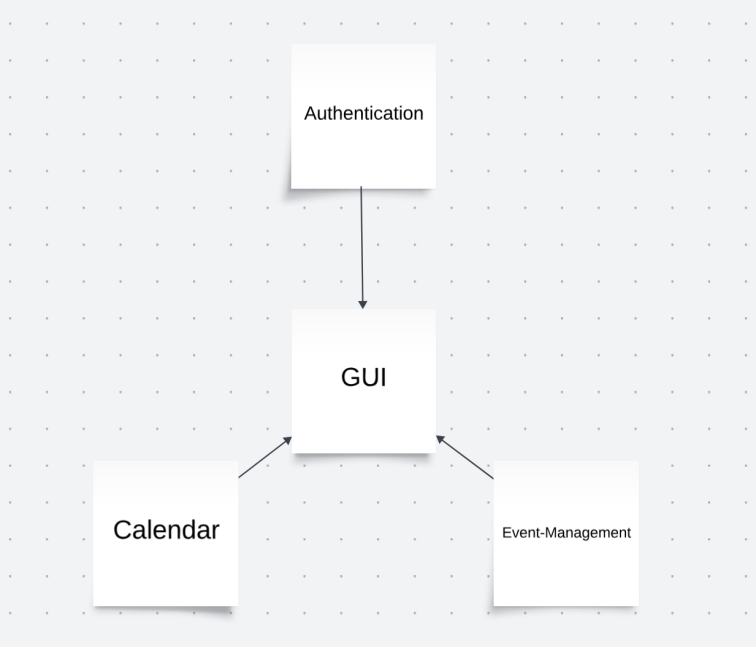
DS - Assignment 2

Liam Leirs

May 2024

1 SOA Overview



1.1 Authentication Service

Features: Handles user login and registration
Data Stored: Username and passwords for users
Connections: connects with GUI service for handling login and registration

1.2 Event Management Service

Features: Even creation and invitation management
Data Stored: Stores events and invitations (along with status of the invitation)
Connections: connects with GUI service for retrieving invites and events

1.3 Calendar Service

Features: Calendar data and invites to calendars
Data Stored: Stores events to which users participate (their calendar) and invitees to a users calendar
Connections: connects with GUI service for retrieving a users calendar and checking if a user is invited to a specific calendar

2 Decomposition Explanation

2.1 Authentication

Separating authentication allows for isolated management of user credentials and enhances security. It can scale independently to handle user authentication requests, which are critical for system access.
If the authentication service fails, users cannot log in or register, effectively locking them out of the system.

2.2 Event Management

Grouping event-related features together allows focused management of events, their attributes, and related user interactions. This allows for scalability as the number of events and users grows.
If the event management service fails, users cannot create or manage events. Invites can also not be retrieved.

2.3 Calendar

Handling user calendars and invitations separately allows optimized storage and retrieval of calendar data. It enables efficient calendar sharing and retrieval.
If the calendar service fails, users cannot view or update their calendars, or other calendars for that matter.

3 API Documentation

3.1 Authentication API

- Register:
  - Endpoint: **POST** /auth/register
  - Description: Registers a new user by storing their username and password
  - Request body: { "username": "string", "password": "string" }
  - Responses:
    - 200: Registration succesful
    - 400: Username already exists
- Login:
  - Endpoint: **POST** /auth/login
  - Description: Logs in a user by verifying their username and password
  - Request body: { "username": "string", "password": "string" }
  - Responses:
    - 200: Login successful
    - 400: Invalid username or password

4 Event Management API

- Create event:
  - Endpoint: **POST** /events/<username>/<title>
  - Description: Creates a new event with the specified details
  - Request body: { "description": "string", "date": "string", "publicprivate": "string" }
  - Responses:
    - 200: Event created successfully
    - 400: Error occurred during event creation
- Get event details:
  - Endpoint: **GET** /events/<event_id>
  - Description: Retrieves details of a specified event
  - Responses:
    - 200: Event details and invitees
    - 400: Error occurred fetching events

- Get public events:
  - Endpoint: **GET** /events
  - Description: Retrieves a list of all public events.
  - Responses:
    - 200: List of public events
    - 400: Error occurred fetching public events
- Invite users to an event:
  - Endpoint: **POST** /events/<username>/<title>/invites
  - Description: Sends invitations to specified users for an event.
  - Request body: { "invites": ["string"] }
  - Responses:
    - 200: Invites sent out correctly
    - 400: Error while sending out invites

5 Calendar API

- Invite user to calendar:
  - Endpoint: **POST** /calendar/<username>/invites
  - Description: Sends an invite to a specified user
  - Request body: { "invitee": "string" }
  - Responses:
    - 200: Successfully sent invite
    - 400: Error occurred sending invite
- Get invites sent by a user:
  - Endpoint: **GET** /calendar/<username>/invites
  - Description: Retrieves all invites sent by a specified user
  - Responses:
    - 200: List of invites
    - 400: Error occurred fetching invites
- Get a user's calendar:
  - Endpoint: **GET** /calendar/<username>
  - Description: Retrieves the calendar of a specified user
  - Responses:
    - 200: User calendar

        * 400: Error occurred fetching calendar

- Add an event to a user's calendar:
  - Endpoint: **POST** /calendar/<username>
  - Description: Adds an event to the calendar of a specified user.
  - Request body: { "event_id": "integer" }
  - Responses:
    * 200: Added event to calendar
    * 400: Error occurred adding event to calendar