Backend Documentation

Logins

POST: /logins/

Parameters: {}

Body: { “firstname”: “”, “lastname”: “”, “password”:“”}

Response: 200: { “firstname”: “”, “lastname”: “”, “password”:“”, “uuid”: “”, “chapter”: “”, “grade”: “”}, 400: invalid login

Description: Validates a users login. If they entered a valid login, their account is sent to the application as the response. If this application was an official application, better security standards, such as encryption would be used.

Backend Documentation

Chapters

POST: /chapters/

Parameters: {}

Body: { “school”: “”, “userUUID”: “”}

Response: 200: {“chapterUUID”: “”}, 400: Invalid or missing parameters

Description: Creates a new chapter. The school parameter is the name of the chapter, and the userUUID is the owners uuid.

GET: /chapters/

Parameters: {}

Body: {}

Response: 200: [{“uuid”:””, “school”:””}, ….], 400: Database Error

Description: Returns a list of all of the chapters in the database. It is returned in a JSON array that could be either one entity or an infinite amount of entities.

DELETE: /chapters/:chapterUUID/:userUUID

Parameters: {“chapterUUID”:””, “userUUID”:””}

Body: {}

Response: 200: {}, 400: Invalid or missing parameters

Description: Removes userUUID from the specified chapterUUID if and only if the userUUID is already in the chapterUUID.

GET: /chapters/:uuid

Parameters: {“uuid”: “”}

Body: {}

Response: 200: {“uuid”: “”, “school”:””}, 400: Invalid or missing parameters

Description: Returns the information of the chapter specified.

GET: /chapters/:uuid/people

Parameters: {“uuid”: “”}
Body: {}
Response: 200: [{uuidPerson: “”}, …], 400: Invalid or missing parameters
Description: Returns the uuids that are in the specified chapter. Returns a JSON array that could be either one entity or an infinite amount of entities.

POST: /chapters/:uuid

Parameters: {“uuid”:””}
Body: {“uuidPerson”: “”}
Response: 200: {}, 400: Invalid or missing parameters
Description: Adds uuidPerson to the uuid chapter. This is essentially a person joining the chapter.

GET: /chapters/:uuid/meetings

Parameters: {“uuid”: “”}
Body: {}
Response: 200: {“uuid”: “”, “name”:””, “chapter”:””}, 400: Invalid or missing parameters
Description: Retrieves all of the meetings associated with chapter, uuid. The response uuid is the meeting’s uuid, the chapter uuid is the chapter to which the meeting belongs to. The name of the meeting is also returned.

Backend Documentation

People

DELETE: /people/:uuid

Parameters: {“uuid”: “”}
Body: {}
Response: 200: {}, 400: Invalid or missing Parameters

Description: This endpoint will delete the stored chapter of the person, uuid. This essentially will make the user leave the chapter.

POST: /people/

Parameters: {}
Body: {“firstname”: “”, “lastname”: “”,“grade”: “”, **Optional:**“password”: “”}
Response: 200: {“firstname”: “”, “lastname”: “”,“grade”: “”, “uuid”: “”}, 400: Invalid or missing Parameters

Description: This endpoint creates a new user account. If there is a password field provided an account will be created as a usable account. If there is no password, the account is just a name-holding account.

GET: /people/:uuid

Parameters: {“uuid”: “”}
Body: {}
Response: 200: {“firstname”: “”, “lastname”: “”,“grade”: “”, “uuid”: “”, “password”: “”, “chapter”: “”}, 400: Invalid or missing Parameters

Description: Returns the information of the user, uuid. This returns ALL information including password. If this were to become an official app, security measures and encryption would be put into effect.

POST: /people/:uuid/chapter/:chapterid

Parameters: {“uuid”:””, “chapterid”: “”}
Body: {}
Response: 200: {}, 400: Invalid or missing Parameters

Description: This endpoint sets the logged in user’s chapter to the chapter they just joined. This makes sure that a user may only be in one chapter at once.

POST: /people/:uuid/chapter

Parameters: {“uuid”:””}

Body: {}

Response: 200: {“chapter”: “”}, 400: Invalid or missing Parameters

Description: This endpoint returns the logged in user’s chapter. If they are not in a chapter a 400 code is thrown to stop any Null errors on the client side.

Errors

POST: /errors/

Parameters: {}
Body: {“error”:””}
Response: 200: {}, 400: Invalid or missing Parameters
Description: This adds the error received from the client to the database

GET: /errors/

Parameters: {}
Body: {}
Response: 200: [{“uuid”:””, “error”:””},....], 400: Invalid or missing Parameters
Description: Retrieves all errors from the database. Returned as a JSON Array. The Array could be empty but could be infinitely large. Each error has its own uuid. This feature is not incorporated into the application.

Backend Documentation

Meetings

GET: /meetings/:uuid

Parameters: {“uuid”: “”}
Body: {}
Response: 200: [{“uuidPerson”: “”, “attendance”:””}, ...], 400: Invalid or missing Parameters
Description: Returns the meeting associated with the uuid. It will be a JSON Array. It could be empty, or it could be infinitely large. uuidPerson is the accounts uuid. Attendance is a binary of 0, absent, and 1, present

PUT: /meetings/:uuid

Parameters: {“uuid”: “”}
Body: {“uuidPerson”: “”,”attendance”: “”,”meetingUUID”: “” }
Response: 200: {}, 400: Invalid or missing Parameters
Description: Updates the person, uuidPerson. This endpoint is called when the user on the front end changes the attendance for a meeting.

POST: /meetings/

Parameters: {}
Body: {“chapterUUID”: “”, “name”: “”}
Response: 200: {}, 400: Invalid or missing Parameters
Description: Creates a new meeting with the name specified. The meeting will belong to the chapter specified. When it is created every member that is in the chapter gets cloned into the meeting.

DELETE: /meetings/:uuid

Parameters: {“uuid”: “”}
Body: {}
Response: 200: {}, 400: Invalid or missing Parameters
Description: Deletes the chapter from the meeting. Currently not implemented