Web API Gateway Requirements

Jason Nguyen - 014232658
Charles Kwak - 015368728
Daniel Gione - 016513144
Dylan Thorin - 015938089
Jingyan Du - 014436615
Fa Fu - 014073642
Kevin Phan 015260958

April 2, 2020

Overview:

The purpose of an API gateway is to expose backend services through a uniform entry point. It allows for increased interoperability. It is also responsible for reconciling the different data formats between the caller and the services. For example, if one team makes use of the JSON format and another uses XML, the API gateway can reconcile this.

Features:

Team Registration/ Team Login

□ Registration □

Functional Requirements:
Description: To allow individuals to register themselves to the web API gateway through a frontend component. Individuals are required to register before using and adding services to the web API.

Preconditions:

- Individual must not be logged in.
- Individual must have selected the registration option.

Required Input:

- Username/Application name.
    - 200 characters maximum, alphanumeric. (encoding: UTF-8)
    - Must be globally unique among users.
- Password.
    - Can be up to 2000 characters.
    - Can be alphanumeric with special characters.
        - The allowed special characters are every special character on the US standard keyboard except for < and >.
    - Should be a minimum of 12 characters.
    - The password should be compared a list of values known to be commonly-used, expected, or compromised:
        - Passwords obtained from previous breach corpuses.

        - Words contained in a dictionary.
        - Repetitive or sequential characters (e.g. ‘1234’, ‘bbbbbb’).
        - Context specific words, such as the name of the application or the current username.
    - A strength meter should be offered to assist the user in choosing a strong password.
    - The password should be able to be copied and pasted into the field.
    - Individual’s entered characters should be displayed for a short time before being hidden to verify correct entry.
    - An option to display the password should be available.
- Re-enter password.
    - Must match entry above.
    - The password should be able to be copied and pasted into the field.
    - An option to display the password should be available.
- Website url
    - used to ensure the request is coming from a valid place.
    - Must be https.
    - Can be up to 2000 characters.
- Call-back url:
    - Used to send authorization code response back to server see []
    - Must be https.
    - Can be up to 2000 characters.

Successful Postconditions:

- Individuals are successfully registered to the web API.
- Display API key/client id to individual [pg 8].
    - Globally unique 32 character hexstring
- Display client secret to individual [pg 9].
    - 64 character hexstring

Failure Postconditions:

- Any of the above required inputs failing.
- Username already exists.
- Password does not meet required strength.

<u>Non-Functional Requirements:</u>

- The process of registering should be completed within 3 seconds.
- The saved information is accurate.

*Note: After successfully registering the developer should immediately store the client secret. This secret is only displayed one time. Individuals can revoke secrets if exposed see [pg 9].

□ Login □

Functional Requirements:
Description: To allow teams to login to the web API gateway through a frontend component and manage their application.

Preconditions:

- Individual must not be logged in.
- Individual must be at the proper login view.

Required Input:

- Username.
- Password.
    - Individuals should be allowed to copy and paste into this field.
    - Individual's entered characters should be displayed for a short time before being hidden to verify correct entry.
    - An option to display the password should be available.

Successful Postconditions:

- Individual now has access to functionality requiring being logged in.
- Access token is returned.

Failure Postconditions:

- Any of the above required inputs failing.
- Username and password do not match.
- Username does not exists.
- After three failed login attempts user is locked and has to message the system administrator.

Non-Functional Requirements:

- The process of login should be completed within 3 seconds.
- User is accurately logged in.

Note: when an individual fails to login the system will prompt “one of the above fields was incorrect”.

Registration / Unregistration Of Services:

□ Registration □

Functional Requirements:
Description: To allow teams to register their web services to the web API gateway.

Preconditions:

- Individual is logged in.
- Individual must have selected register service option.

Required Input:

- Valid access token
- The url of the webservice to be registered.
- The route to access the web service using the web API gateway.
- Privacy rules: who is allowed to use this web service:
    - Open to only your team
    - Open to other teams, including your own (select who is allowed to use it).
    - Closed.

  *note: can only select one of the above choices.
  *note: once the service is registered the url **cannot** be changed. Teams will need

  Reregister to update url.
- Description of what type of datastores is compatible with the service.
    - Up to 200 characters.

Optional Input:

- Connection String Requirement:

Successful Postconditions:

- The service is successfully registered.

Failure Postconditions:

- Any of the above required inputs failing.
- The url to web service is invalid. Must return a 200 status code.
- Service is already registered.
- Connection string is unreachable.
- Url provided is unreachable.

Non-Functional Requirements:

- The process of registering a service should be completed within 3 seconds.

□ Unregister □

Functional Requirements:

Description: To allow teams to take off their web services from the web API gateway.

Preconditions:

- Has to be logged in.
- Individual must have selected unregister service option.

Required Input:

- Valid access token
- Select the service/services to unregister from the list previously registered owned services.

Successful Postconditions:

- The web services are unregistered from the web API gateway.
- The route to the web service through the web API gateway cannot be used anymore.
- The unregistered web services will not appear in service discovery [pg 8].
- If someone tries to use an unregistered service they will get a status code of 410 Gone.

Failure Postconditions:

- Any of the above required inputs failing.

Non-Functional Requirements:

- The process of unregistering a service should be completed within 3 seconds.

Configure Services

□ Enable And Configure Another Team's Service □

Functional Requirements:
Description: To allow a team to use another team's service with a different data store than the one it is registered with by default.

Preconditions:

- Individual is logged in.
- Individual must have selected service discovery option.

Required Input:

- Search and select web service to configure. (web services that are available for discovery must be set public for your team). [pg 8]
- Connection string to datastore.

Successful Postconditions:

- Future use of the configured service is updated the team.

Failure Postconditions:

- Any of the above required inputs are invalid.
- Settings were not saved correctly and accurately.

Non-Functional Requirements:

- The process of configuring a service should be completed within 3 seconds.
- The information is accurately updated.

Manage Registered Services

□ Change Privacy Rules □

Functional Requirements:
Description: To allow registree to change the privacy settings on their own web services.

Preconditions:

- Individual is logged in.
- Individual must have selected the “manage privacy” option and selected a service from a list of previously registered owned services.

Required Input:

- Valid access Token
- Privacy Rules - updated rules about who is allowed to use this web service
    - Open to your team.
    - Open to other teams,including your own (select who is allowed to use it).
    - Closed.

  *note: can only select one of the above choices.

Successful Postconditions:

- The new privacy rules on that web service are updated.
- If someone tries to use a service they don’t have access to the server will return a 403 Forbidden status code.

Failure Postconditions:

- Any of the above required inputs failing.
- The privacy rules of the web service are not updated.
- The updated privacy rules are not displayed.

Non-Functional Requirements:

- The process of updating a service’s privacy rules should be completed within 3 seconds.

Service Discovery

□ View Services By Filter□

Functional Requirements:
Description: Allow registrees to view available services on the web API gateway by filter.

Preconditions:

- Backend services are available.

- For the web services to be available for discovery it must be either public or available to the team that is browsing services.
- Registree must be logged in.
- Must have selected the service discovery option.

Required Input:
- Valid access token.
- Filter options (options include: team name, input data type, output data type, dataformat)
- Filter options can be empty, empty filter will show all services that are public or belong to the team that is browsing services.

Successful Postconditions:
- Display a list of all services that match the filter.
    - Included with the service name is information about how to call that service. (inputs, outputs, dataformat etc.)
- Display can return no results if no service matched the filter.

Failure Postconditions:
- Any of the above required inputs failing.
- Displays services that do not match filter criterias.
- Does not display results that match the filter criteria.

Non-Functional Requirements:
- The process of displaying for matched service should be completed within 4 seconds.
- Filter results are accurate.

API Key/ client id

□ Revoke API Key/Get New API key □

Functional Requirements:
Description: Revoke an API key for a specific team and get a new one.

Preconditions:

- Be logged in.

Required Input:
- Valid access token
- Selected the revoke API key option.

Successful Postconditions:
- Previous API can no longer be used to request tokens.
- Displays new 32 character globally unique hex string API key for team to save.

Failure Postconditions:
- Any of the above required inputs failing.
- Previous API Key is still valid.

Non-Functional Requirements:
- The process of revoking should take a maximum of 3 seconds.

Client Secret

□ Revoke client secret/Get new client secret□

Functional Requirements:
Description: Revoke a client secret if it gets exposed.

Preconditions:
- Be logged in.

Required Input:
- Valid access token.
- Selected the revoke client secret option.

Successful Postconditions:
- Previous secret can no longer be used to request tokens.
- Displays new 64 character hex string for team to save.

Failure Postconditions:
- Previous secret is still valid.

Non-Functional Requirements:

- The process of revoking should take a maximum of 3 seconds.

ID Token

□ Get ID token□

Functional Requirements:

Description: ID tokens are used to perform additional authentication checks to secure our APIs.

Preconditions:

- Previously registered application.
- Previously requested authorization code.

Required Input:

- Authorization code
- Client credentials: API key/client id and client secret.

Successful Postconditions:

- Return valid JWT ID token which represents that the application “logged in”

Failure Postconditions:

- Does not return a valid ID token.

Non-Functional Requirements:

- The process of requesting a new token should take a maximum of 3 second.
- ID token should be accurate to that user.

Access Token/Refresh Token

□ Get access token and Refresh Token □

Functional Requirements:
Description: After sending the authorization code and client credentials the authorization server will send a short lived access token and a long lived refresh token back.

Preconditions:

- Previously registered application.
- Previously requested authorization code.

Required Input:

- Authorization code
- Client credentials: API key/client id and client secret.

Successful Postconditions:

- Return a JWT access token allowing a team to use web services available to them.
    - Expire time: 5 minutes
- Return a JWT refresh token allowing teams to obtain a new access token when the previous one becomes expired or invalid.
    - Expire time: 2 hours
- Structure of JWT:
    - **Header**

      {
      “alg”: “RS256”,
      “typ”: “JWT”
      }
    - **Payload**

      {
      “Iss”: “https://webapi.com”,
      “client_id” : “1231231231213”,
      "iat": 14700011113,
      "exp": 14700111113,
      “Scope”: “1231231231213”,
      }

*note: iss: issuer, client_id: gotten from registration, iat: issued at, exp: expired. Scope is equal to client_id

- **Signature**
  {
  RSASIGN(HEADER + “.” + Payload, private key)
  }

Failure Postconditions:

- Does not return a valid access token.
- Does not return a valid refresh token.

Non-Functional Requirements:

- The process of requesting a new token should take a maximum of 3 second.

□ Use Refresh Token □

Functional Requirements:

Description: Refresh tokens are used to obtain new access tokens. They are issued by the authorization server and used by the authorization server.

Preconditions:

- Previously registered application.
- Previously requested authorization code.
- Previously sent authorization code and credentials to get a refresh token.

Required Input:

- Valid refresh token.

Successful Postconditions:

- Access token has an equal scope as the previous one.
- Structure of JWT:
  - **Header**
    {
    “alg”: “RS256”,
    “typ”: “JWT”
    }
  - **Payload**
    {

}

*note: iss: issuer, client_id: gotten from registration, iat: issued at, exp: expired. Scope is equal to client_id

- **Signature**

  {

  }

Failure Postconditions:

- Access token has a different scope than the previous.
- Refresh token is invalid.

Non-Functional Requirements:

- The process of requesting a new access token should take a maximum of 3 second.

□ Use Access Token □

Functional Requirements:

Description: Allows a team to use a protected resource(web service). Tokens have a restricted scope and time.

Preconditions:

- Previously registered application.
- Previously requested authorization code.
- Previously sent authorization code and credentials to get an access token.
- Have permission to use the requested web service.

Required Input:

- Token.
- Resource.
- host(irl to API gateway)

Note: used following specification of authorization request header field: https://tools.ietf.org/html/rfc6750#section-2.1

Successful Postconditions:

- Allows the caller to access the protected resource(web service).

Failure Postconditions:
- Access token expired.
- Access token has incorrect scope.

Non-Functional Requirements:
- The process of using an access token should take a maximum of 3 second.

Logging

□ Log all web API gateway actions □

Functional Requirements:
Description: Log all actions that the web API gateway performs into a datastore.

Preconditions:
- Datastore must exist.

Required Input:
- Timestamp (YYYY-MM-DD hh-mm-ss-ms) *ms is milliseconds
- Action performed.
- Web API key that performed action.
- Status code returned.

Optional Input:
- Failure message.

Successful Postconditions:
- Log is inserted into the specified datastore.

Failure Postconditions:
- Datastore connection is down.

Non-Functional Requirements:
- The process of logging should take a maximum of 1 second.

Universal Data Formatting

□ Reconcile conflicting data formats □

Functional Requirements:
Description: Allows for the seamless translation between data formats of differing services and the web API. The API gateway will communicate using Json format.

Preconditions:
- Service expects data format that differs from what the web API usually sends.

Required Input:
- Data to send to web API gateway.

Successful Postconditions:
- Translated data format correctly to the web service.

Failure Postconditions:
- Format is not translated correctly.

Non-Functional Requirements:
- The process of translation should take a maximum of 1 second.

Routing

Functional Requirements:
Description: Allows for more complex calls to backend services to be aggregated into one call, as well as increasing performance by calling some services asynchronously.

Preconditions:
- Backend Services being called must be registered in the API Gateway.
- Call to API Gateway must be authorized.
- Data submitted to API Gateway must be formatted through universal formatter before being processed.

Required Input:

- HTTP Request for specific services.
- Data needed by backend services called.
    - Stored inside the HTTP request body.

Successful Postconditions:

- HTTP response is returned to the client.
- HTTP response contains the expected data.

Failure Postconditions:

- Request does not receive a response from API Gateway.
- Route does not exist.
- Requested backchannel is not found.
- Backend service required is down.

Non-Functional Requirements:

- All transactions complete within four seconds.