

# Lab 04/Project

## Adding Extra Functionality

TDDD97/732A56

Web Programming

Department of Computer Science

Linköping University

## Introduction

In this lab, you will learn additional web programming concepts, techniques and technologies. In addition you will independently search, assess, apprehend, and apply information about new technologies and third-party resources as well as downloading, installing, configuring and troubleshooting relevant libraries and frameworks.

## Grading

Your task is to extend the Twidder application, which you developed in the first three labs, with additional functionality. You can choose to implement different functionality from a list of alternatives described below, where each criteria gives a certain number of points. Your final grade for the course will be determined by the total number of points your implementation collects (see Table 1).

Table 1. The number of points required for achieving each grade

Total number of criteria points	Course grade
3	3
6	4
9 + well-documented code(Server)	5

Below you will find a description of each alternative criteria, including suggestions for third-party libraries or frameworks, and the number of points they give. You may use other libraries and frameworks, but in this case you must discuss them with you lab assistant before starting the work.

### NOTE

The mandatory section needs to be done first before moving to alternatives. This section gives two points and also mandatory in order to get passed in the project.

## Mandatory

### Using HTTP status codes and upgrading the feedback mechanism[2 points]

Your task is to use HTTP status codes instead of field `status`, introduced in lab 2. Both the client and server code shall be upgraded after you are done. Each server function shall be upgraded properly and use status codes to cover all cases in the code. The following status codes shall “at least” be used:

#### 200 OK

The request succeeded.

#### 201 Created

The request succeeded, and a new resource was created as a result. This is typically the response sent after POST requests, or some PUT requests.

#### 400 Bad Request

The server cannot or will not process the request due to something that is perceived to be a client error (e.g., malformed request syntax, invalid request message framing, or deceptive request routing).

#### 401 Unauthorized

Although the HTTP standard specifies “unauthorized”, semantically this response means “unauthenticated”. That is, the client must authenticate itself to get the requested response.

#### 404 Not Found

The server cannot find the requested resource. In the browser, this means the URL is not recognized. In an API, this can also mean that the endpoint is valid but the resource itself does not exist. Servers may also send this response instead of 403 Forbidden to hide the existence of a resource from an unauthorized client. This response code is probably the most well known due to its frequent occurrence on the web.

#### 405 Method Not Allowed

The request method is known by the server but is not supported by the target resource. For example, an API may not allow calling DELETE to remove a resource.

#### 409 Conflict

This response is sent when a request conflicts with the current state of the server.

#### 500 Internal Server Error

The server has encountered a situation it does not know how to handle.

Some of the status codes may be returned automatically by Flask. In such cases it is only the client code which needs to be updated in order to handle the problem. Handling the problem mostly requires at least notifying the user about the issue. Feel free and add more status codes to your implementation if possible.

Please keep in mind that you are also required to upgrade the client-side code in a way that all feedback shown to the user is decided by the client-side and not the server-side as you may have done in lab 2.

### HTTP status codes

[https://developer.mozilla.org/en-US/docs/Web/HTTP/Status#successful\\_responses](https://developer.mozilla.org/en-US/docs/Web/HTTP/Status#successful_responses)

**NOTE**

As a status code can be initiated by a broad range of causes you may still need to rely on the *message* field coming from the server. This field shall contain more specific information about the status of the request and what exactly has happened at the server-side. Please remember the message you show to the user shall be decided by the client and not the server in order to **decrease coupling**. In other words the message you get from the server turns more into an error key intended for the client code and not the end user.

**Example, the user tries to read a message received from another user:**

**Server returns {message:"user not found"}, 404**

**Client displays "The entered username does not exist! Please try again!"**

**Server returns {message:"message not found"}, 404**

**Client displays "Incorrect message id! The message may have been deleted!"**

At the client-side it is important to first check the status code in order to determine the status and then possible error key. It can happen that a status code does only have a possible error key or it can happen that it doesn't matter which error key has been returned and the same feedback shall be shown to the user anyway.

**Example, the user tries to create a new account:**

**Server returns {message:"user exists"}, 409**

**Client displays "The username is already taken! Please try another one!"**

In the above example we assume that there is only one possible error key. In such cases we don't even need to check the error key.

## Status Code Checklist

Function	Expected Status Codes
sign_in	200, 400, 401, 405, 500
sign_up	201, 400, 409, 405, 500
sign_out	200, 400, 401, 405, 500
change_password	200, 400, 401, 405, 500
get_user_data_by_token	200, 401, 405, 500
get_user_data_by_email	200, 401, 404, 405, 500
get_user_messages_by_token	200, 401, 405, 500
get_user_messages_by_email	200, 401, 404, 405, 500
post_message	201, 401, 400, 405, 500

## Alternatives

### 1. Providing Live Data Presentation [3 points]

This criterion requires your application to present dynamic data on the fly. The data may change over time while being presented as diagrams or charts using a third-party library such as *D3JS* and *ChartJS*. The data must be application-related and be composed of at least three different fields, which are updated by the backend. For example, the user could be able to see a live presentation of the number of posts on his/her wall, number of views of his/her page, and total number of users currently online. You need to use websockets or other available technologies to push the data to the clients on the fly. **At least two types of diagrams shall be used.**

#### D3 official website

<http://d3js.org/>

#### ChartJS official website

<http://www.chartjs.org/>

## 2. Use of HTML5 for Drag and Drop [1 point]

This criterion requires your application to provide drag-and-drop functionality for performing a specific task. Please note that the drag-and-drop functionality itself matters in this assignment even though you can also add a new requirement, which can be used by using the drag-and-drop functionality. For example, the user can copy the posts on his/her wall, posted by himself/herself or other users, into a text area to be edited and reused.

### **HTML5 Drag and Drop**

[http://www.w3schools.com/html/html5\\_draganddrop.asp](http://www.w3schools.com/html/html5_draganddrop.asp)

## 3. Performing Client-side Routing + Overriding Back/Forward buttons using the History API [2 point]

This criterion requires your application to provide a valid URL associated with each view or resource. The difference from the traditional form of URL handling, is that it shall happen at the client-side and shall mainly be done by using Javascript and third-party libraries. This approach requires having at least three separately-assigned URLs. Note that your implementation must tolerate any unwanted page refreshes and maintain the correct URL in the address bar upon any refresh event. You are also required to use the History API to make back and forward buttons usable for changing views in your application. At the current stage clicking on the back button will cause the browser to leave your application.

### **PageJS a small client-side routing library**

<http://smalljs.org/client-side-routing/page/>

### **Manipulating the browser history**

[https://developer.mozilla.org/en-US/docs/Web/API/History\\_API](https://developer.mozilla.org/en-US/docs/Web/API/History_API)

## 4. Third-Party Authentication Using OpenID/OAuth 2.0 [3 point]

Give the user the possibility to login via a third-party service, such as Facebook, Google and Microsoft. Please note, since your application is based on SPA architecture, you need to avoid page refreshes as much as you can.

### **What is OpenID**

<https://openid.net/developers/how-connect-works/>

### **Google OAuth 2.0**

<https://developers.google.com/accounts/docs/OAuth2>

## 5. Applying Further Security Measures [3 points]

This criterion requires the following two security measures to be applied:

1. Protecting passwords from being stolen directly from the server, in case of the database being compromised, by hashing them before storing in the database. In other words instead of the password itself, its hashed value shall be stored in the database once the user signs up.

### **Password Hashing**

<https://crackstation.net/hashing-security.htm>

### **Flask BCrypt**

<http://flask-bcrypt.readthedocs.org/en/latest/>

2. Protecting data while being transmitted. Since data, including tokens, are transmitted unencrypted over the network, they are subject to be retrieved and abused by a third party tapping the network. In this case, a villain can read a request to the server, change some parameter in it and then resend it to the server on behalf of the victim until the token becomes invalid by logout. This is called REPLAY attack. You are required to follow the provided instructions and prevent such attack. At the same time this technique does not wire the token itself on the network which prevents anyone from reading the token itself and using it later on to make its own requests. This technique does not prevent villain from seeing other data parameters.

### **Guideline**

<https://www.ida.liu.se/~TDDD97/labs/hmacarticle.pdf>

### **Replay Attack**

<https://www.kaspersky.com/resource-center/definitions/replay-attack>

## 6. Testing Using Selenium [2 points]

This criterion requires you to test your application by using the *Selenium* UI testing framework. You need to test at least five (5) different UI features of your application like *sign in*, *sign up* and *post messages*. Please note that it is not allowed to use the *Selenium IDE* and instead you need to use another language like python for writing the test cases.

### **Selenium official website**

<http://www.seleniumhq.org/>

### **Selenium documentation**

<http://docs.seleniumhq.org/docs/index.jsp>

## 7. Client-side Templating Using a Third-Party API [1 point]

By using templates, you can define an HTML structure that always remains the same. Inside the template there are variable parts that change each time the template is rendered depending on the data provided. Templates can be used to implement different views inside of an application, for example the profile view. Here you are required to use a third-party library, such as *Mustache* and *HandlebarsJS*, to perform client-side templating to reimplement the existing views inside of Twidder application. **To re-do only one view is enough.**

**Mustache official website**

<https://mustache.github.io/>

**HandlebarsJS official website**

<http://handlebarsjs.com/>

## 8. Media Streaming [3 points]

This criterion requires you to have both Audio/Video and Image files stored at the server-side and streamed to the client when needed. Users should be able to upload media files to the server, which stores them in a database or in a directory on the disc. For example the user shall be able to share media on his/her wall to be seen by other users. The user could also upload and set a profile picture for himself/herself to be seen by other users.

**HTML5 Video**

[http://www.w3schools.com/html/html5\\_video.asp](http://www.w3schools.com/html/html5_video.asp)

### NOTE

In case of storing media files on disc, make sure they are **NOT** publicly accessible by using URL pointing to them in the static directory. You need to add a service(s) that sends media files to the client based on the received parameters.

## 9. Styling and Responsive Design [2 points]

This criterion requires you to use a third-party framework, such as Bootstrap and Pure, for handling layout and styling. Also, you may need to use your own CSS code for certain cases. Using only CSS is also OK. You should follow the responsive design principles to make your application adaptable to different display resolutions from mobile to desktop for at least three different display-resolution ranges:



1. Mobile view
2. Tablet view
3. Desktop view

As a guideline, there are two areas in which your application needs to adapt to different screen sizes:

- The layout
- The resources such as images which are displayed to the user.

For example, in the desktop view all media and images shall be displayed in multiple columns while in the mobile view no or a few images are displayed and all text is displayed in one column. The tablet view stands in between the two.

#### **Bootstrap official website**

<http://getbootstrap.com/>

#### **Pure official website**

<http://purecss.io/>

#### **Guide to Responsive Web Design**

<http://blog.teamtreehouse.com/modern-field-guide-responsive-web-design>

#### **Responsive Web Design Basics**

<https://developers.google.com/web/fundamentals/layouts/rwd-fundamentals/>

#### **HTML Responsive Web Design**

[http://www.w3schools.com/html/html\\_responsive.asp](http://www.w3schools.com/html/html_responsive.asp)

## **10. Deploy Your Solution on a PaaS [2 points]**

Platform as a service (PaaS) is a cloud computing model in which a third-party provider delivers hardware and software needed by developers and companies to deploy their applications on the cloud.

Your task is to deploy your application on the cloud with the help of a PaaS provider. In result, the users shall be able to open your web application by using a domain address, given by the provider after deployment, instead of localhost.

#### **Top 10 PaaS providers of 2023 and what they offer you**

<https://www.techtarget.com/searchcloudcomputing/feature/Top-10-PaaS-providers-and-what-they-offer-you>

**NOTE**

We do know that Microsoft Azure provides free student accounts at the time of writing the instructions. You can follow the following instructions from another course, TDDD80. The instructions are in Swedish:

<https://www.ida.liu.se/~TDDD80/laborationer/server/lab3.sv.shtml>

**NOTE**

Switching to a real DBMS like MySQL or PostgreSQL **shall give an extra point.**

## 11. Geolocation [2 points]

Your task is to show the user's location by using HTML5 geolocation. The location shall be displayed to other users in the form of a functionality. For example, Every message shall also include the user's location who has posted it. It is a requirement to use Geocode.xyz API to convert the coordinates to a readable address.

**Geocode.xyz API**

<https://geocode.xyz/api>

**NOTE**


It is the coordinates, instead of the human-readable address, which need to be stored in the database. The conversion of the coordinates to an address can either happen at the client-side or the server-side.

## 12. Recover Your Password [2-3 points]

Your task is to allow the user to recover his/her password via e-mail or sms. As the task is about security, most of your code about password recovery is located at the server-side. Having said that, the user shall have the required UI at the client-side to initiate the process. Please note that the new password can be generated for the user and sent to him/her. In this scenario this alternative shall only give two points. The user can get the chance to enter the new password directly on a page whose link has been sent to him/her. The page shall receive a unique id as a part of the link which determines the password recovery regards which user. In this scenario the alternative gives 3 points.

### 13. Verification via SMS[3 points]

The user shall be able to add a phone number under Profile tab which is going to be shown under Home tab, if verified. The verification process includes sending a random code to the entered phone number which shall later on be entered inside of the web application for completing the verification process. The user shall be able to request a new SMS with a new code at any time which itself invalidates the previous sent code.

- If the user enters an incorrect code then a feedback shall be displayed and the user shall have the chance to ask for another code via SMS. No SMS is sent automatically after the user has entered a wrong code. The user can enter wrong codes in a row for a limited number of times, for example 3 times. After that the phone number shall be blocked for that specific user. The user shall not be able to use that phone number anymore and shall receive an error message before even starting the verification process.
- If the user succeeds then a feedback shall be displayed and the phone number shall be saved.
- The user shall have the possibility to remove the phone number at any time without verification.
- In a real world scenario the number of SMS which are sent to a specific user shall be limited during a specific time period, for example 10 SMS per day. This requirement is not included in this alternative. Having said that, if you decide to implement this requirement then you shall get an extra point reaching the total number of **4 points**  for the whole alternative.

For implementing the verification process via SMS you need to use one of the following CPaaS, CPaaS stands for Communication Platform as a Service:

- Sinch, <https://www.sinch.com/>
  - <https://developers.sinch.com/docs/sms>
  - [https://dashboard.sinch.com/signup?utmcsr=%28direct%29&utmcmd=%28none%29&utmccn=%28not%20set%29&utmcsr=google&utmcmd=cpc&utmccn=Search\\_US\\_Sinch\\_TM&utmctr=sinch%20sms&utmctt=](https://dashboard.sinch.com/signup?utmcsr=%28direct%29&utmcmd=%28none%29&utmccn=%28not%20set%29&utmcsr=google&utmcmd=cpc&utmccn=Search_US_Sinch_TM&utmctr=sinch%20sms&utmctt=)
- Twilio, <https://www.twilio.com/en-us>
  - <https://www.twilio.com/docs/messaging>
  - <https://www.twilio.com/en-us/blog/get-started-twilio-free-sms-businesses>