BOGAZICI WEB ANNOTATION TOOL

A PROJECT FOR BOGAZICI UNIVERSITY SWE574 CLASS

Bogaziçi Web Annotation Tool is an annotation tool for web pages that is compatible with W3C Web Annotation standard.

TABLE OF CONTENTS

BOGAZICI WEB ANNOTATION TOOL

A PROJECT FOR BOGAZICI UNIVERSITY SWE574 CLASS

DEVELOPERS

Özlem AKBAŞ

Sarah BEIRKDAR

Uğur HİÇYILMAM

Gökhan ÖZGEZEN

Mahmut Ali ÖZKURAN

Anıl Selim SÜRMELİ

WELCOME

This document is project report for the Boğaziçi Web Annotation Tool that created for Boğaziçi University Software Engineering SWE 574 Fall 2016 class.

Boğaziçi Web Annotation Tool (b.w.a.t) project started to developed in 2016 September.

A web annotation is meta information associated with a web resource. Since Web is bloated with lots of information in various formats, it is a common need to give users the ability to contribute, rate, explain, or criticize to existing web resources.

b.w.a.t project is about giving the users ability of creating web annotations, storing them, sharing the annotations with other users, and accessing the annotations created by other users online.

The project will consist of a web server and a Firefox plugin as client. Since it will be a RESTful server architecture, it will be extendable by a number of different client applications in the future.

The World Wide Web Consortium has introduced standards for representing and sharing annotations. Boğaziçi Web Annotation Tool is going to be compliant with the standards created by W3C.

For detailed information, please refer to Project Summary page.

PROJECT SUMMARY

Web Annotations

INTRODUCTION

There are many information on the web in many forms. Users can access information using a browser on websites, however there is no way for the users to contribute, rate, discuss, enter feedback on the existing information on the web unless the functionality is explicitly added by the developers.

It is a common need to have the ability to add meta information on web content. Here are the several use-cases for a web annotation tool to contribute to web itself:

- allowing users to discuss about a certain content,
- by giving ability to users to correct an information, or adding additional explanation,
- increasing user experience by giving users chance to highlight certain parts of a web page,
- giving users a standardized medium to discuss and share ideas about a content.,
- giving users ability to rate a certain content about its correctness, usefulness or about its user experience.
- allowing users to access detailed information and commentary about a web content.

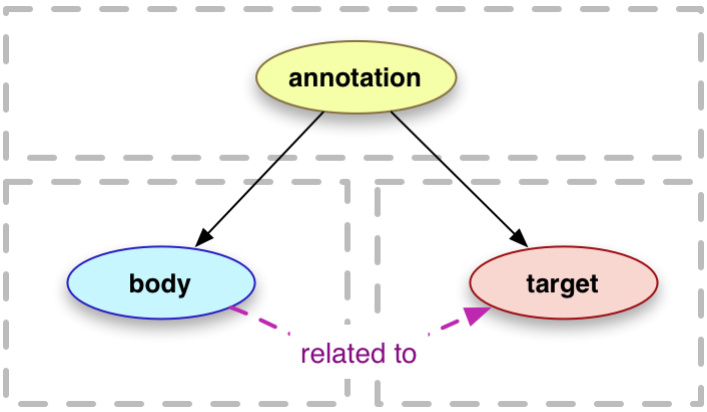
It would be best for all browsers to have a standardized solution to this problem. However, as of today, there is no standard solution to the problem.

World Web Consortium has a commission about the web annotations that introduced several standards about representation and transferring of annotation data.

W3C ANNOTATION DATA MODEL

According to W3C, an annotation itself is a web resource. Since each web resource has a standardized representation to allow sharing information, annotations also should have standardized representation.

The standard introduced by W3C typically consists of two main parts: a target, and a body.



(The image is taken from W3C website.)

An annotation data should be represented as an object in JSON format. Below is an example provided by W3C consortium:

EXAMPLE 1: Basic Annotation Model

```
{
  "@context": "http://www.w3.org/ns/anno.jsonld",
  "id": "http://example.org/anno1",
  "type": "Annotation",
  "body": "http://example.org/post1",
  "target": "http://example.com/page1"
}
```

According to W3C, an annotation can be added following types of web resources:

- Dataset
- Image
- Video
- Sound
- Text

The type of the web resource also should be included in the annotation data representation.

WHAT IS THIS PROJECT ABOUT?

The project will produce a web annotation tool. It will consist of a web server and a client application implemented as a Firefox plugin. However, it will be possible to create different client applications in the future.

The annotation tool created will give its users the ability to create and share annotation thorough web. Since it will be following the standards created by W3C, it will also be possible to consume other annotation tools created by Boğaziçi Web Annotation Tool, as long as they follow the same standards.

Similar Tools

- Genius: It is a tool that let's users to select text content on web pages and adding annotations. It let's users to discuss on an annotations. It's more like a commenting tool on a web resource. It has chrome extension but it is also possible to add web-pages through its JavaScript library.
- Hypothes.is: It is a chrome plug-in let's its users to take personal notes through annotations. It also let's users to discuss about a content using annotations. In addition to annotations, it also let's its user to highlight text on page.
- A.nnotate: A.nnotate is an online annotation, collaboration and indexing system for documents and images, supporting PDF, Word and other document formats. It is more about adding user notes on a content.
- Awesome Highlighter: This is a firefox plugin allows you to highlight any text on any website and save them for future reference. It is also possible to share the highlighted resource with other users.
- http://www.blerp.com/: It allows users to discuss about any web content anonymously.
- Bounce: It let's users to take a screenshot of any website and then add annotations to the captured screenshot.
- QuickFox: It is note-taking plug-in for Firefox.

ADDITIONAL RESOURCES

Extension Development

Chrome Extension Development

Firefox Extension Development

Web Annotation

Wiki-Web Annotation

W3C Annotation Standards

Web Annotation Data Model

Web Annotation Protocol

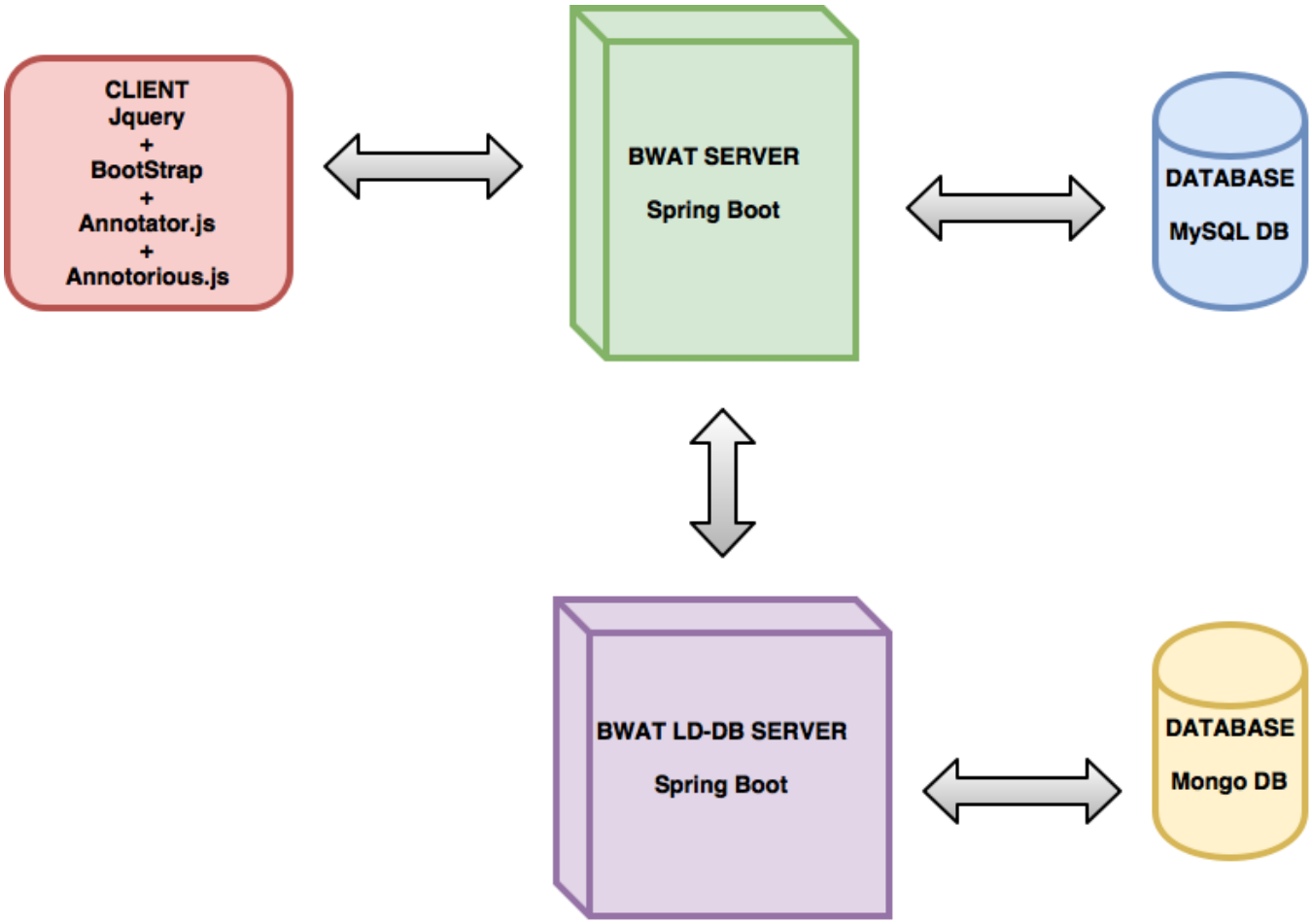
SCOPE OF THE DEVELOPMENT

This software project is to develop a web application for the ability to add meta information on web content. Requirements of the project can be summarized at the following statements:

1. A user will be able to create an account.
2. A user will be able to activate his account after creation.
3. An authenticated user will be able to create a text annotation.
4. An authenticated user will be able to create an image annotation.
5. An authenticated user will be able to edit his/her annotations after creation.
6. A user will be able to delete his/her annotations.
7. A user will be able to list all the annotations created on the web content.

SYSTEM ARCHITECTURE DESCRIPTION

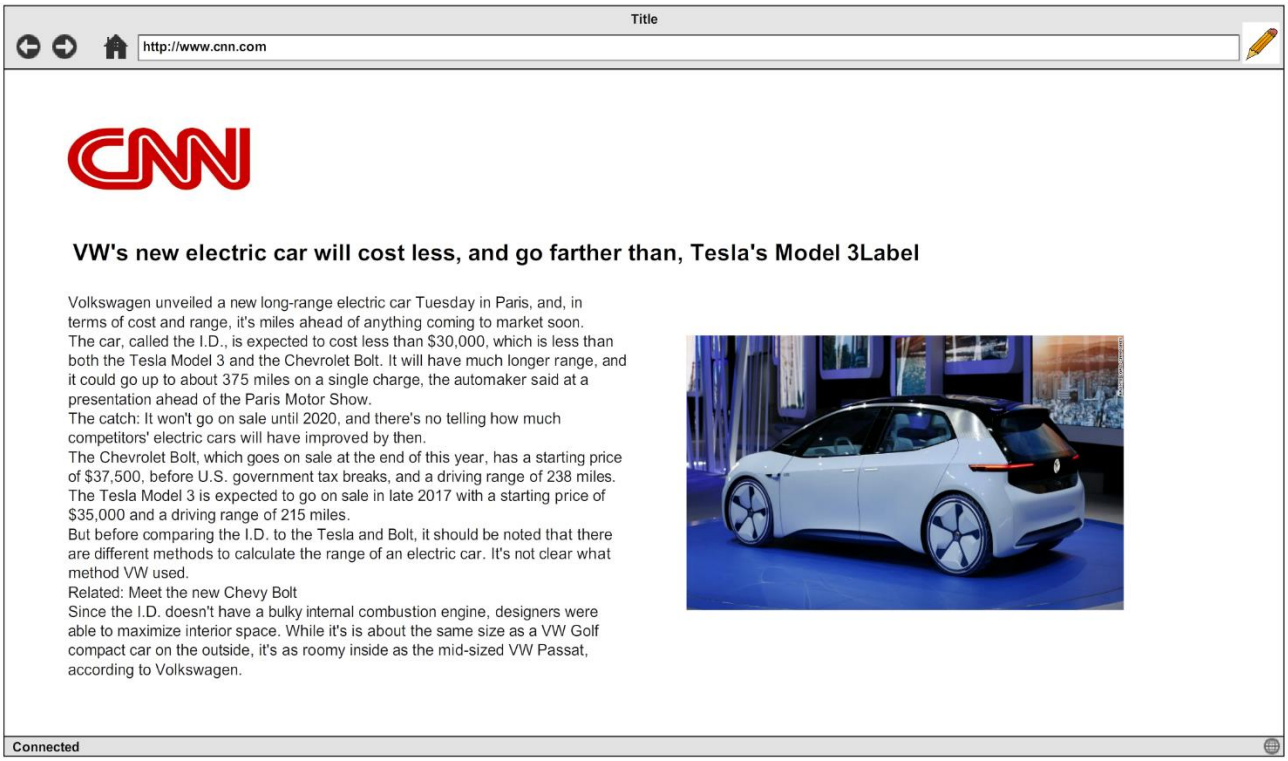
System architecture schema is below:



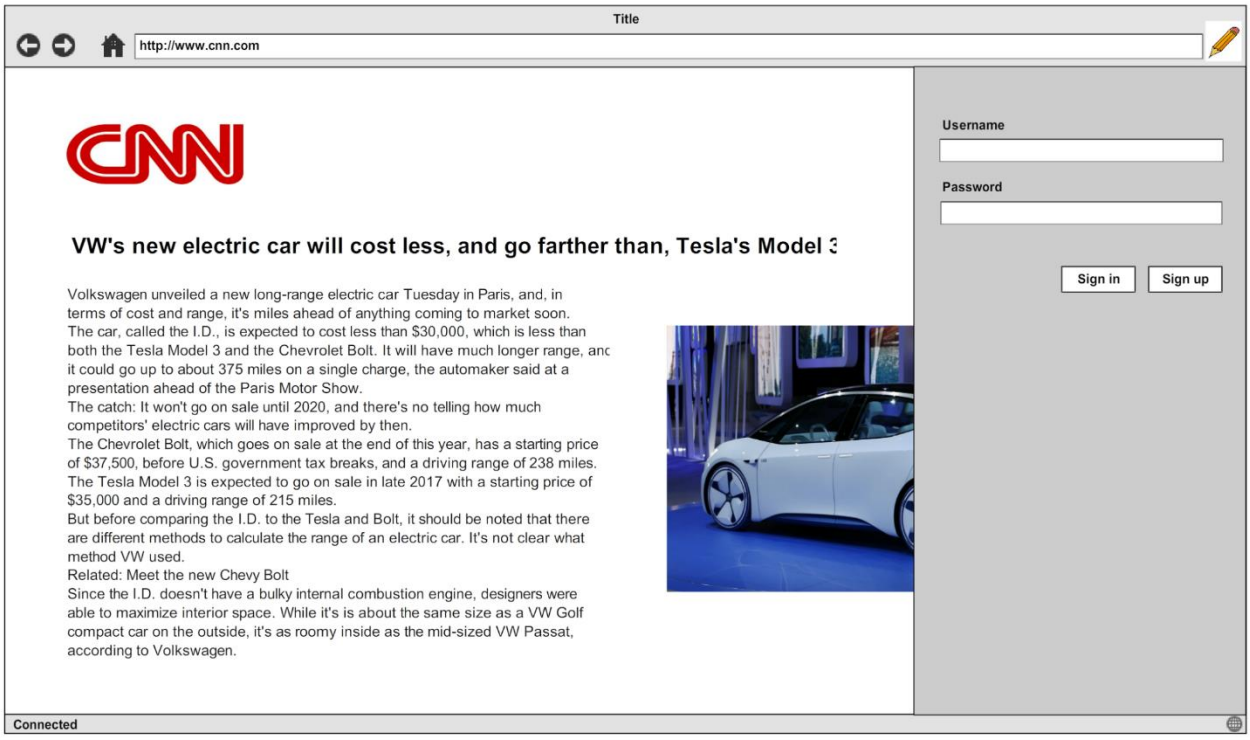
MOCKUPS

Annotation Tool

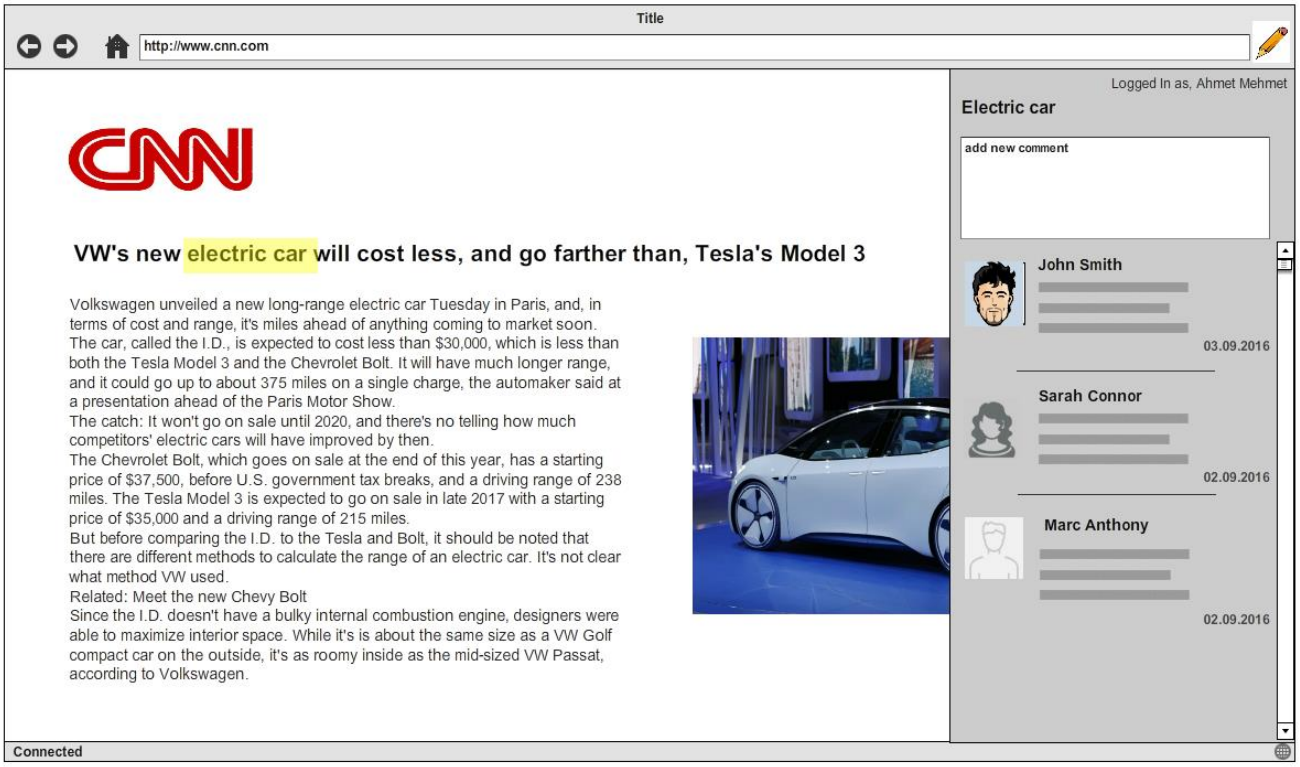
The application will be located in the upper corner of the browser. By clicking it, the user is required to enter his username and password for the first time, or sign up. Then the application will hold the log in information.



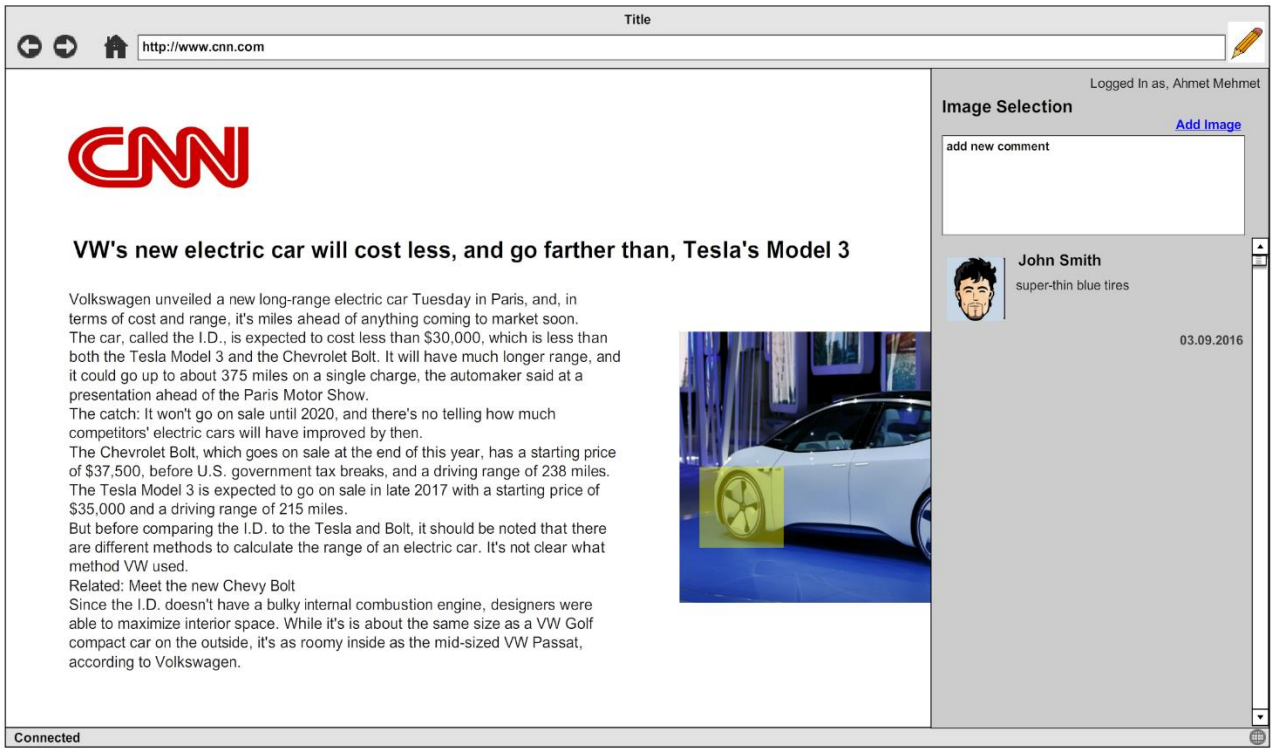
Login Screen



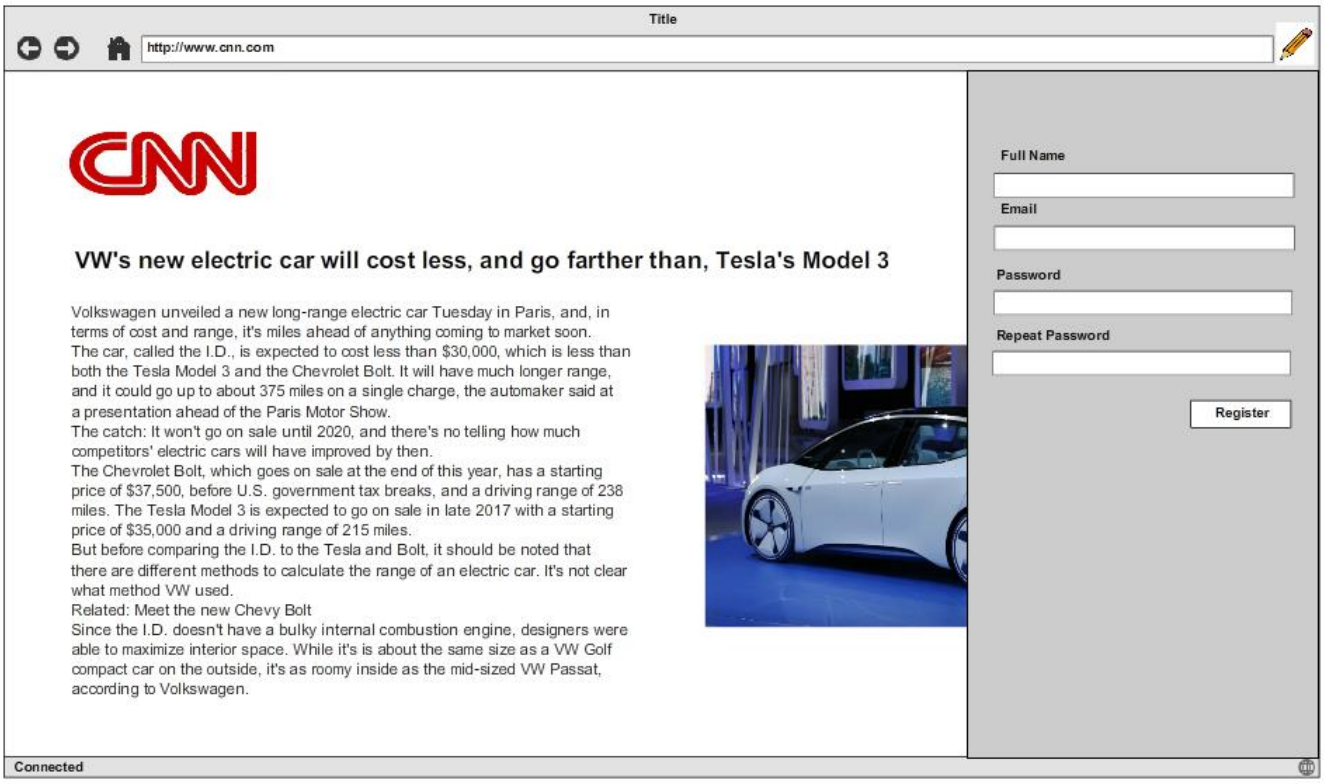
Adding Textual Annotations



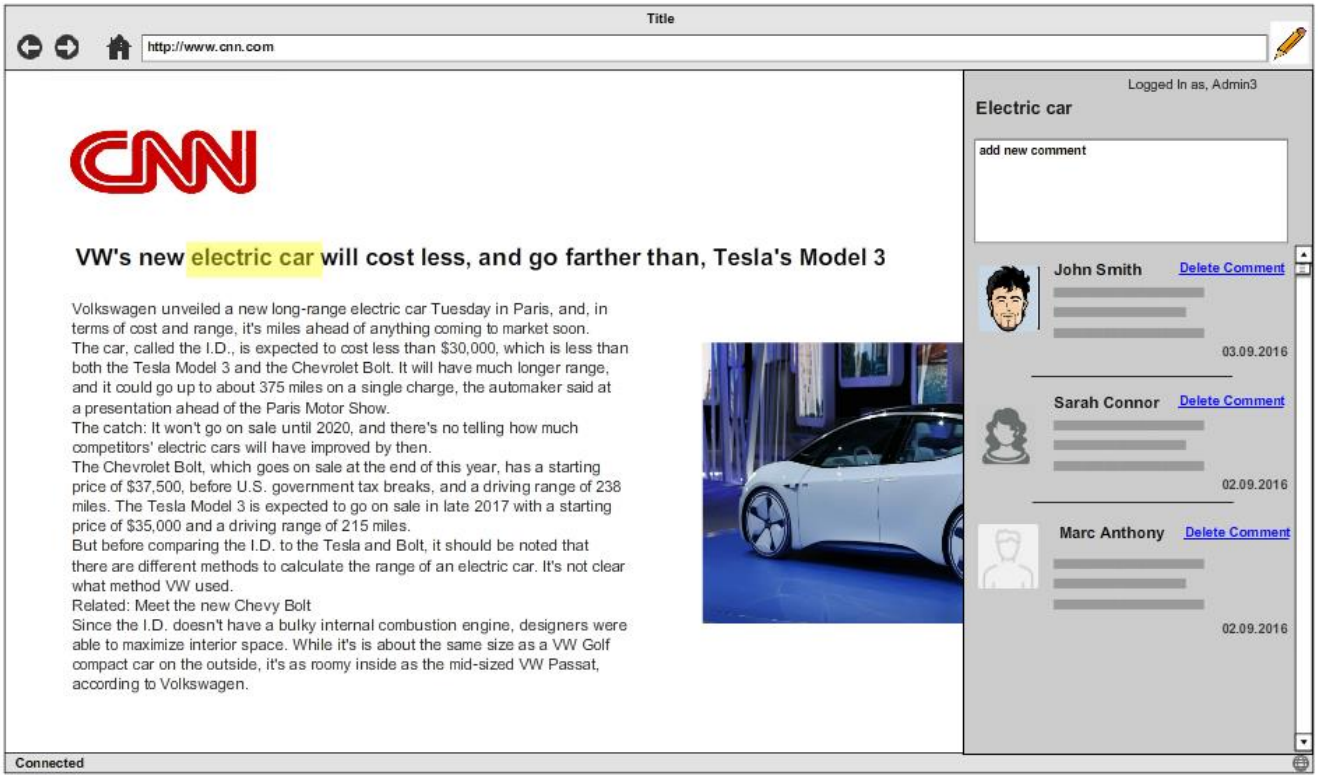
Adding Annotation to Graphical Media



Registration Page

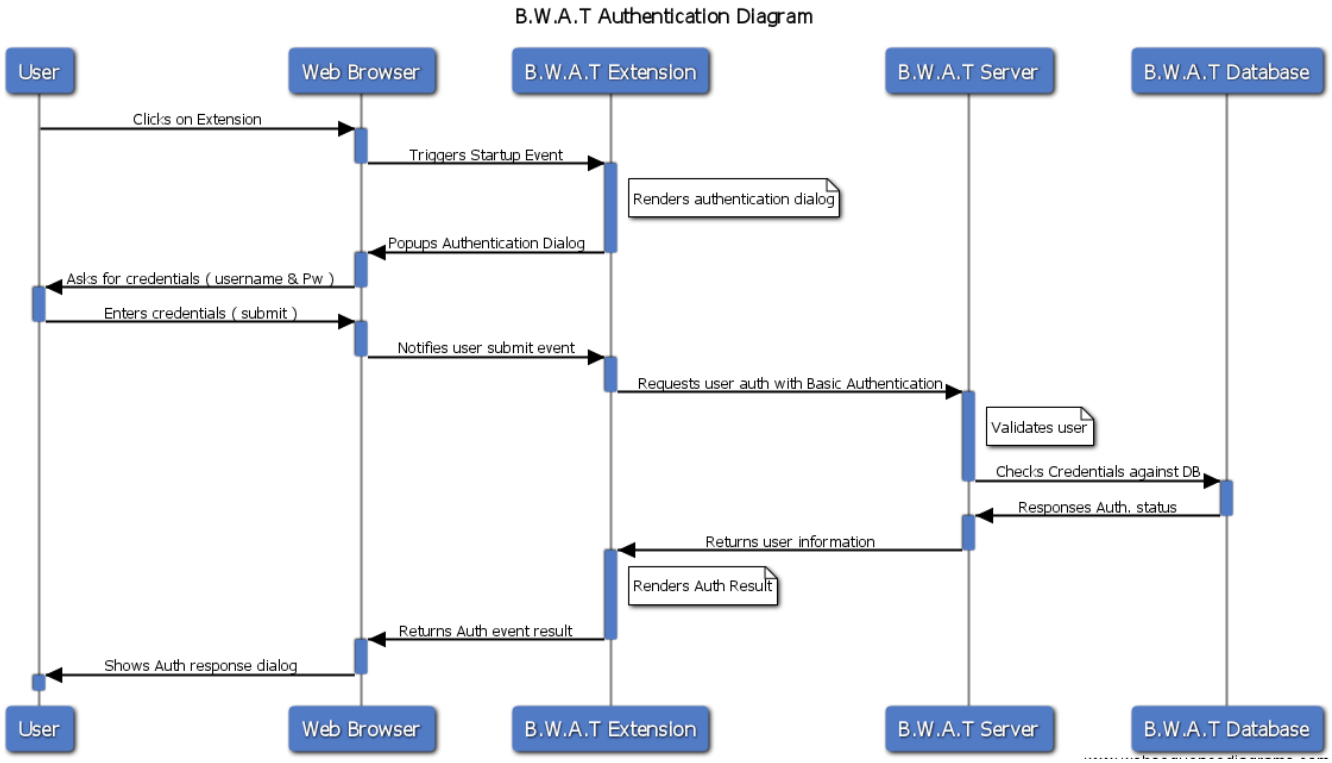


Administration Page

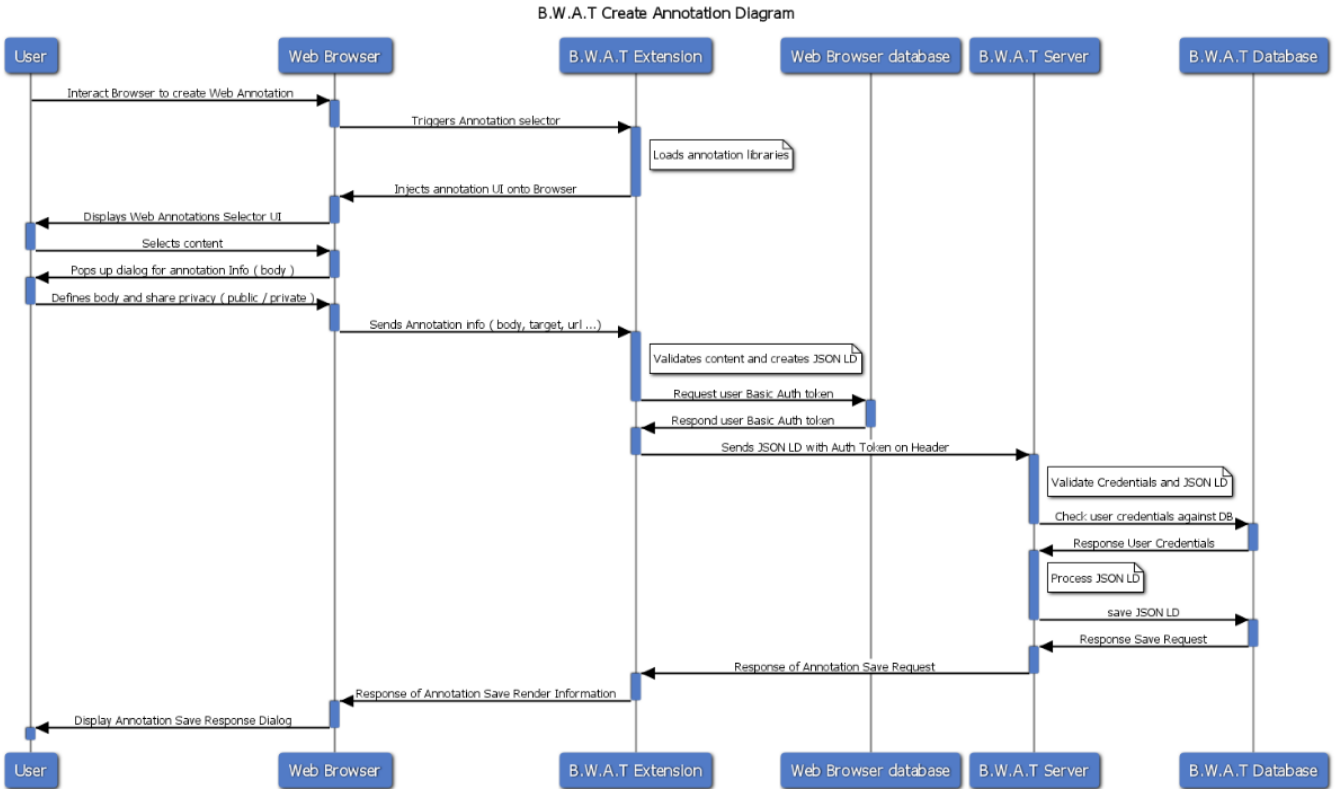


SEQUENCE DIAGRAMS

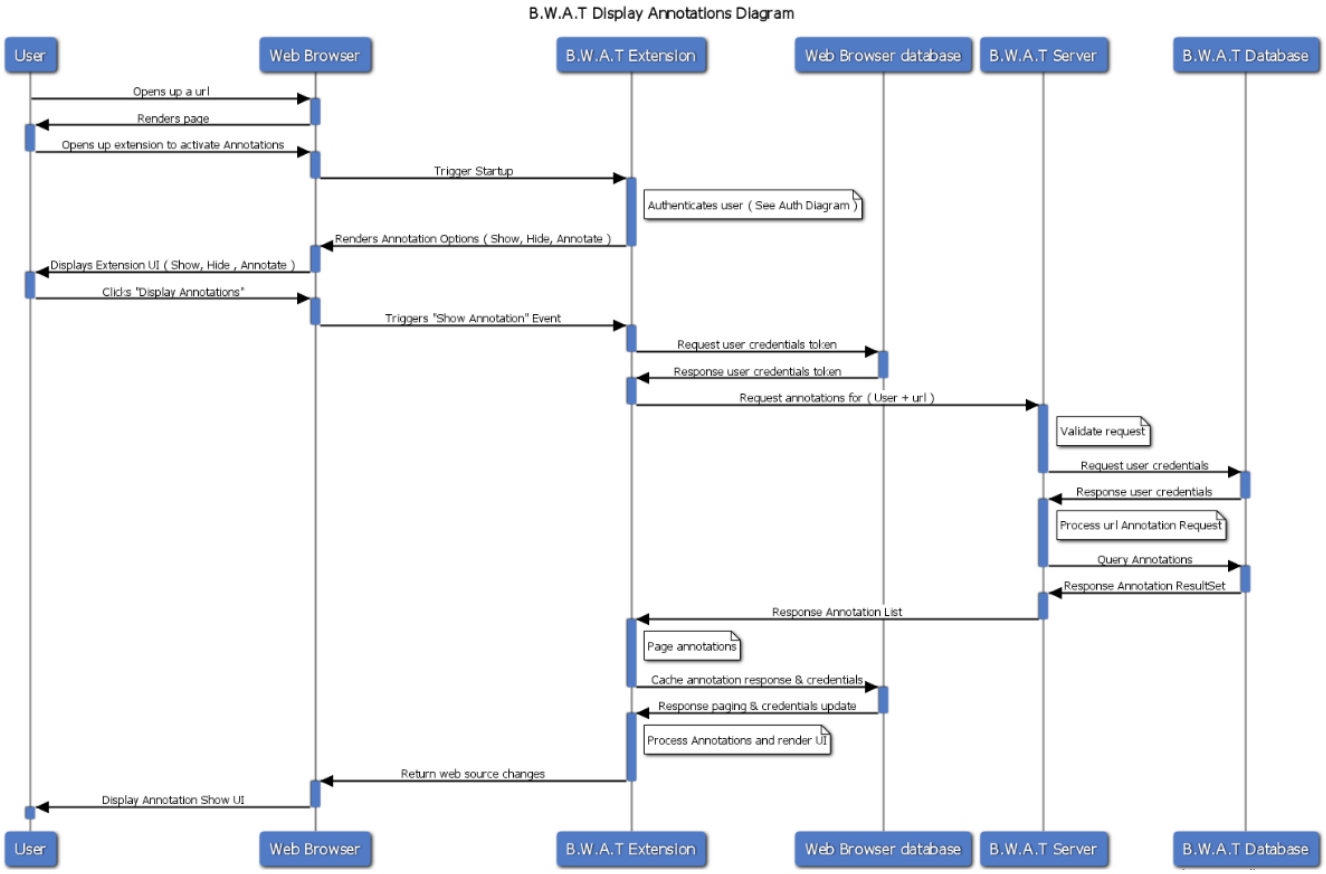
User Authentication



Create Annotation Diagram



Display Annotations on a Web Source Sequence Diagram



DEPLOYMENT

Backend of the application consists of two separate server instances. First one named “bwat-server” and it is responsible for authentication and binding annotation information to an account. The other one is named “bwat-ld-db” and its purpose is to provide a database implementation for annotation data. Each of them are created using Java 8 and Maven.

Server packages are provided in both .war and .jar formats. Server instances can be deployed as .war files to a servlet container such as Tomcat or WebLogic, and even better, they can be started directly as Java applications since the provided .jar package already contains an embedded tomcat server instance.

Deployment Using War Files

Deployment using the war files is quite straightforward since most of the configuration setting handled by the servlet container. Since there are two server instances and they should bound together, there is still a little configuration to be set.

On the “bwat-server”, there is file named “application.properties”, which provides a convenient way of externalizing platform specific configuration. In the file, there is a name-value pair as following:

app.config.annotation-server-base-url=http://localhost:8083

The default provided value is “http://localhost:8083”, which is a convenient default value since most probably, the two instances will be working on the same machine. As long as they work on the same machine, they can be accessed each other using “localhost” placeholder.

However, since using .war file means the deployment will be made on a servlet container, the server URLs will be provided by the container. Therefore, the configuration should be made accordingly.

Deployment Using Jar Files

Deployment using the “.jar” files, is the most convenient and simple way to deploy both server instances. Configuration files already contains the required settings two work together.

As default, “bwat-server” has been set to work on port 8080 and the “bwat-ld-db” server instance set to work on 8083.

As long as they operate on the same machine, default configuration will be working.

To start server instances, use below console commands:

java -jar -Dspring.profiles.active=dev bwat-server-0.0.1-SNAPSHOT.jar 2>errorOutput.log > output.log &

java -jar -Dspring.profiles.active=dev bwat-ld-db-0.0.1-SNAPSHOT.jar 2>ldErrorOutput.log > ldOutput.log &

First command is to start “bwat-server” and the second command is to start “bwat-ld-db” server.

Dependencies

The “bwat-server” has the following dependencies:

- spring-boot-starter-data-rest: 1.4.1.RELEASE
- spring-boot-devtools: 1.4.1. RELEASE
- lombok: 1.16.10
- spring-boot-starter-security: 1.4.1.RELEASE
- spring-boot-starter-web: 1.4.1.RELEASE
- h2: 1.4.192
- spring-security-test: 4.1.0.RELEASE
- json-path: 2.2.0
- json-path-assert: 2.2.0
- spring-boot-starter-test: 1.4.1.RELEASE
- spring-boot-starter-data-jpa: 1.4.1.RELEASE
- springfox-swagger-ui: 2.2.2
- springfox-swagger2: 2.2.2
- modelmapper-spring-boot-starter: 1.1.0

The “bwat-ld-db” has the following dependencies:

- spring-boot-starter-data-mongodb: 1.4.2.RELEASE

- spring-boot-starter-web: 1.4.2.RELEASE
- lombok: 1.16.10
- fongo: 2.0.6
- spring-boot-starter-test: 1.4.2.RELEASE
- jsonld-java: 0.8.3

Using Deployed Server

Currently, the authentication and the annotation servers are deployed on an AWS instance.

To access server, please use: ec2-35-162-70-40.us-west-2.compute.amazonaws.com

The server also has a Swagger instance running, which allows for developers to discover API. You can access swagger using: ec2-35-162-70-40.us-west-2.compute.amazonaws.com/swagger-ui.html

Deployment of Web-Server on Local

The simplest way to deploy the server on your local machine is using "Maven" build tool.

DEPLOYMENT USING MAVEN

In order to deploy using maven, maven should be installed on your machine. Some IDEs comes with embedded maven installations and you can use them if you wish (i.e. Intellij).

If you don't have maven installed on your machine, you can go and download it on this page. Then you can proceed to this page for installation instructions.

After you successfully install maven on your machine, open a bash (cmd or PowerShell for windows) terminal and go into the bwat-server folder located in your project copy.

The following command runs tests and starts server:

$ mvn spring-boot:run

DATABASE SETTINGS

Currently, bwat-server is configured to work with MySQL instance running on Amazon RDS.

If for some reason, you wish to run MySQL on your local machine, you can use application.properties file located in \bwat-server\src\main\resources. To connect to the MySQL instance on your local machine, update the "spring.datasource.url", "spring.datasource.username", "spring.datasource.password" fields accordingly.

Rest API

Registering User

Field	Value
Title	Register
Url	/api/users
Method	POST
Url Params	None
Data Params	firstName:String, lastName:String, password:String, mail:String Example: `{ "firstName" : "John", "lastName": "Doe", "password": "123456", "mail": "john.doe@gmail.com" }`
Description	This endpoint should be used to create a new user.

User Login

Field	Value
Title	Login
Url	/api/users/login
Method	GET
Url Params	None

Field	Value
Data Params	None
Description	This endpoint returns user information as long as basic authentication token placed into the header.

REST API FOR ANNOTATION DATABASE API

Creating Annotation

Field	Value
Title	Creating annotation
Url	/annotation
Method	POST
Url Params	None
Data Params	annotationObject : object
Description	This endpoint is for creating annotation in json-ld format, if the object does not satisfy the Annotation standard, it returns bedrequest. API returns HTTP Status OK for valid annotation objects, and persists it to mongo db.

Getting annotation by Id

Field	Value
Title	Getting annotation by id
Url	/annotation/{id}
Method	GET
Url Params	id: String
Data Params	None
Description	Returns the annotation for requested id.

Getting all annotations

Field	Value
Title	Get all annotations
Url	/annotation/all
Method	GET
Url Params	None
Data Params	None
Description	Returns all annotations persisted on db.

Updating annotation by Id

Field	Value
Title	Update Annotation
Url	/annotation/{id}
Method	PUT
Url Params	id: String
Data Params	annotationObject : object
Description	Updates the annotation with given Id if it is valid. Returns Http Not Found if there is no annotation with the id.

Deleting annotation by Id

Field	Value
Title	Delete Annotation
Url	/annotation/{id}
Method	DELETE
Url Params	id: String
Data Params	None
Description	Deletes the annotation with given id.

CLASS DIAGRAM