Project “SimplestNote”

Version 1.0

Preamble

The course projects are inspired by real-world needs, and they usually refer to similar sites already on the web. Students must follow the specifications given by this document, but they could also refine them through an interaction with the teacher and the analysis of similar websites. In any case, the final realisation must be completely original. The published information must be well organized and accessible, to satisfy the different kinds of users of a public website.

Site Specifications

SimplestNote is a simple online notepad with note-sharing features. The site is accessible to registered users only. Each user has a virtual notebook, consisting of an arbitrary number of pages, which we shall call *notes* in the following. Each note has a title and a content (both plain text). The length of the title may be limited, whereas the content length should be unlimited. Notes are also associated with their creation date, last modification date, version number (corresponding to the number of times the note has been updated) and an arbitrary number of tags, defined as words or short sequences of words (e.g., "*news*", "*work*" or "*private messages*"). The system stores all the versions of each note, so as to prevent unwanted changes and provide a note history. It will be possible to share a note with other users of the system, making it appear in their notepads and possibly allowing them to modify it.

The following list contains a brief description of the contents and functionalities required by this site. Obviously, any further refinement or enrichment of these specifications will increase the value of the project.

– Users can register on *SimplestNote* providing their own personal data (name, surname, etc.) and a valid email address (the email verification, which can be performed by sending an email with a link to click to verify the address, is *optional*) and getting their access credentials. The email address will be used as the username and the password will be randomly generated at registration time.
– Once logged in, the user can see his/her list of notes (with title and last modification date). Selecting a note opens the reading/editing view. The interface should also allow the user to create new notes (with an initial default title, e.g., the current date/time) and delete existing ones.
– The reading/editing view of a note enables the user to change the note title, content and tags. *Optionally*, when adding a tag, the system may suggest one of the tags already used by the user in other notes, as well as allow the definition of a new tag.
– When a note is updated (changing its title, content or tags), the system saves its current state (title, content, tags, last modification date) in the note *history*, and then applies the required changes (saving the new state of the note, increasing the version number and updating the last modification date). In this way, the history will be gradually populated with all previous versions of the note.
– The user can access the history of a note, displaying the version number and the date of modification of all of its previous versions. Clicking on an history item displays (*as read-only*) the title, content and tags of the note for that version.
– A note can be shared with other users of the same system by specifying the user’s email address. The share can be configured as read-only or read/write. Shared notes are displayed within the user’s note list in such a way as to make clear their origin (email of the owner) and access permissions (read or read/write). Notes shared as read-only can be viewed but not modified. On the other

hand, the user should be able to make any kind of change on notes shared in read/write mode, just like on his own notes. The owner of a shared note can cancel the share at any time.

– *Optionally*, you can try to implement the connection endpoint for a mobile client of our service: a simple system to export the note data in JSON format. To this aim, the site should expose two special URLs, which we shall denote respectively with *http://server.it/notepad/getList?user=email&pass=password* and *http://server.it/notepad/getNote?user=email&pass=password&note=ID*. When requesting the first URL, the server must return a JSON structure (pay attention to the *content type*!) that represents the list of notes belonging to the user with the specified *email*, authenticated using the given *password*. Obviously, if the authentication fails, the response should be empty. The list must contain, for each note, its ID (of course your system will provide every note with an identifier, e.g., the primary key used to insert it into the database) and its title. On the other hand, when requesting the second URL, the server must return a JSON structure containing the creation date, modification date, version number, title, content and tags of the note whose *ID* is indicated in the URL, belonging to the user with the specified *email*, authenticated using the given *password*. *Note*: in a real application, sending username and password in every service call would be very inefficient and unsafe, but here we use this approach to simplify the implementation.

Technologies

– The basic structure of the site must be created using XHTML *strict* or *transitional*. HTML5 is also allowed, but always using XML syntax. The validation of all the site pages with respect to the chosen XHTML flavour is an important part of the development and **must** be reported in the documentation.

– The site layout must be realised using CCS style sheets. The layout can be freely based on third party layouts available on the web or shown during the lectures. In this case, the degree of **customization** of the layout will be taken into account in the final project assessment.

– For the *client-side* programming, Javascript is the **required** language. It is possible to include libraries developed by third parties, provided that they have a suitable cross-browser portability and that they are described in the project documentation. However, **any abuse of these technologies is not recommended**, especially when they can be replaced with a suitable use of HTML, CSS, etc. **In general, your site should be functional also with Javascript disabled**. Using the site without scripts may be less "friendly" or allow the access to "core" functionalities only, but sites whose dynamics is entirely script-based are not allowed. However, scripts can play a more important a role in the functionalities whose users are restricted and pre-determined (e.g., in the *back-end* functionalities for administrators, but not in the public *front-end* or in the login procedure).

– For the *server-side* programming, Java *(servlets, JSP)* is the **required** language. Any DBMS and template engine can be employed, if required. Again, it is possible to rely on external libraries.

– In general, the site must work and have a good *rendering* on Internet Explorer, Mozilla Firefox and Google Chrome, and *possibly* be compatible with older browsers (in this case it should at least *degrade well)* and with the latest versions of other browsers, like Opera. Browser compatibility **must** be explicitly stated in the documentation.

Project Development and Documentation

The specifications may not be exhaustive or completely defined. Every feature added or refined, also through an interaction with the stakeholder or the site end users, will be adequately assessed. All the design choices must be discussed and motivated.

The final project, developed following the guidelines given by the present specification, must be a fully functional website, whose contents and features will be assessed during the examination. The specification parts marked as *optional*, if not developed, will not make the project insufficient but, on the other hand, will not allow your project to reach the highest mark. If you choose to implement an optional feature, the result should not be necessarily perfect or complete: it must only show your commitment to deal with an advanced issue.

The documentation (**in electronic format**) which accompanies the project **must** contain at least the following information:

User Documentation

– Indication of the minimum installation requirements (with particular regard to the software dependencies).

– Installation instructions.

– Description of the main features and their usage.

Technical Documentation

– Diagram showing the content and navigability of the site.

– Relational schema of the database (if any).

- Analytical description of the site layout, also indicating which of its components are static/dynamic.
- Declaration of the XHTML type used, description of *any* validation problems, list of compatible browsers.
- Description of the adopted technical solutions (languages, standards, protocols, ...).
- Screenshots of the most important website pages (*optional*).

The actual contribution of each group member to the project **must** be declared in the documentation (indicating, for example, the members that mainly worked on server and client side programming, the layout designer, etc.). During the examination, each group member will describe its part of the realisation.

Project Evaluation

To evaluate the project, the following issues will be considered (in order of importance):

1. Compliance with the specifications.
2. Technical correctness.
3. Organizational clarity and correctness of the contents.
4. Accessibility and standards compliance.
5. Appropriate use of static and dynamic contents.
6. Design quality.
7. Adequacy of the documentation.

This evaluation will be combined with the result of the project discussion.

Additional Information

This specification is available in PDF format on the website of the Web Engineering course at the web address http://www.di.univaq.it/gdellape. Additional information on the specifications can be obtained directly via email by writing to giuseppe.dellapenna@univaq.it.

Please note that projects should carried out by *small* student groups (three members is the recommended number). Exceptions to this rule must be agreed by the teacher.

Summary

This summary should be compiled and sent to the teacher, in electronic form, *before* the examination. A copy of this form should be also attached to the project documentation.

Project name: ______________________________

Authors:

First Name	Last Name	ID	Contribution / Role in project development

Client-side technologies (XHTML strict, HTML5, CSS 2/3, Javascript, JQuery, …):

____

____

Server-side technologies (e.g., JSP, MySQL, template engines, librerie, …):

____

____

Compatible browsers:

Browser	Version	Compatible	Degrading	Not compatible	Not tried
Internet Explorer					
Mozilla Firefox					
Google Chrome					
Opera					
Safari					

Project discussion date: ______________________________