**Vision Document**

**Web-based UML Class diagram editor “JUmlIt”**

**Rev.: 0.3**



Revision History

|  |  |  |  |
| --- | --- | --- | --- |
| **Rev.** | **Date** | **Author** | **Change Description** |
| 0.1 | 2015-10-09 | Kovaljov Eugene | Initial draft for review/discussion has been created |
| 0.2 | 2015-10-16 | Kovaljov Eugene | Mistakes are corrected. New information has been added |
| 0.3 | 26.10.2015 | Kovaljov Eugene | Mistakes are corrected. |

**1** **System vision**

[**1.1** **Document purpose**](#h.tyjcwt)

[**1.2** **Scope**](#h.3dy6vkm)

[1.2.1 High-level requirements](#h.1t3h5sf)

[1.2.2 Out of scope](#h.4d34og8)

**1.3** **Terms, abbreviation and definitions**

**1.4** **References**

[**2** **Preliminary Epics and User Stories**](#h.3rdcrjn)

2.1.1 Preliminary use case diagram

[2.1.2 Preliminary epics and user stories](#h.lnxbz9)

[**3** **Solution Highlights**](#h.35nkun2)

[**Attachment A. High-level design**](#h.2jxsxqh)

# System vision

## 1.1 Terms, abbreviation and definitions

|  |  |
| --- | --- |
| UML | Unified modeling language is a general-purpose, developmental, modeling language in the field of software engineering, that is intended to provide a standard way to visualize the design of a system |
| JS | JavaScript |
| ER | Entity-relationship notation |
| Diagram | A diagram is a two-dimensional geometric (can be three-dimensional also) symbolic representation of information according to some visualization technique. |
| CRUD | Create, retrieve, update, delete operations |
| JPEG | Joint Photographic Experts Group |
| XML | eXtensible Markup Language |

## 1.2 Background

Each developer wants to start coding as soon as possible, but a big part of developers doesn’t pay enough attention to the conceptual part of programming. It is also often the case that developing the basic structure seems to be rather a routine procedure.

## 1.3 Document purpose

The main goal of writing this document is to define high-level needs and features of the “JUmlIt”. It shows why the product should be developed and emphasizes the primary needs of clients and target users. Common details about work principles of the “JUmlIt” are described in the attachment.

## Scope

A UML-class diagram editor is going to be developed. This tool is supposed to simplify the development of applications. The main problem in terms of the application development process lies in the fact that no application can be built without the design stage. The tool should help developers to create a structural class model of the application and generate the Java code base, which will be improved in the further stages of the development process. Editor should support following elements and shapes: rectangle, arrowed lines.

### High-level requirements

The following are high-level requirements that are requested to be fulfilled by the solution.

* Managing account (attachment A, Figure 4)
  + Edit username, email, password
  + log out
  + log in
  + sign up
  + recover password
* Project management:
  + create new project
  + add team members
  + edit privacy of the diagram
  + save project
  + delete project
  + save project as file
  + open project
  + open latest project
  + open related to the user project
* manage UML2 class diagram (attachment A, figure 3):
  + add following shapes: square, arrows and lines
  + Mark shapes as one of the statements: class, interface, enumerable.
  + add shape to the working sheet
  + remove a shape
  + move shape across the working sheet
  + set class name of the object
  + set access modifiers of class members, have them displayed differently (e.g. public has green color, private - red etc)
  + set field types (primitive or custom types)
  + add class methods
  + set method’s return value (primitive or custom type)
  + connect shapes together, setting the type of the relationship between different types (generalization, association, realization, aggregation, composition) and setting the appropriate style of the connection line.
  + view autogenerated source code. This source code will be available only for reading.
  + handle the relationships between classes when generating Java source code (e.g. if the diagram has generalization, the generated class must have an “extends”-directive, if the diagram has a composition relationship - instance of a class must be present in another class)
  + generate each drawn class in a separate .java-file.
  + Simultaneous work on the same diagram for multiple users.
  + export the diagram as image-file or as an xml-file
  + Leave comments for diagram and it’s parts.
  + Email notification about collaboration invitation to added users
  + sign up via Google+, vk.com, facebook.com, twitter.com or using built-in registration

### Out of scope

### The following is out of scope of the application:

* Generating .class –files (Compiled .java files)
* Formatting the java-source code with tabulations and spacing
* Formatting the Java code according to code conventions.
* Application internationalization
* Java syntax highlighting/colorization for Java key words

### 2.2 Scope of Initial Release

The initial version of the product should include the implementation of the basic functionality. The site visitor must be able to sign in or create an account using a profile on Google, Twitter, VK, Facebook. The user should be able to interact with the working sheet and add objects on it. Manipulate these objects (move, delete, drag etc.), set the relationships between them. The system must autosave the diagram. System must allow user to download Java-files, image and xml files.

**2.3 Scope of Subsequent Releases**

In subsequent versions, it is planned to implement complete functionality allowing users to conveniently manage their diagram and conversate using comments, set project privacy, share diagrams, view history and be notificated via email.

**1.4 Business Risks**

The greatest risk in creating apps is the high level of competition in the market, which may impede the progress and development of the product. The probability of this risk is the highest. Fighting with it is possible only by improving the product and adding an exclusive functionality. One of the major risk factors associated with the implementation of the product, for our team is the lack of experience in the development of such applications and the lack of a theoretical basis. This factor is probably not much effect on product quality. The fact that we are full-time students may negatively affect on product development process due to lack of time for doing project.

Thus:

1. The whole functionality will not be finished in time by 15.12.2015
2. Key team members become sick
3. Environment will not accept software during deployment
4. System will crash during the presentation.
5. Team member leaves the group.

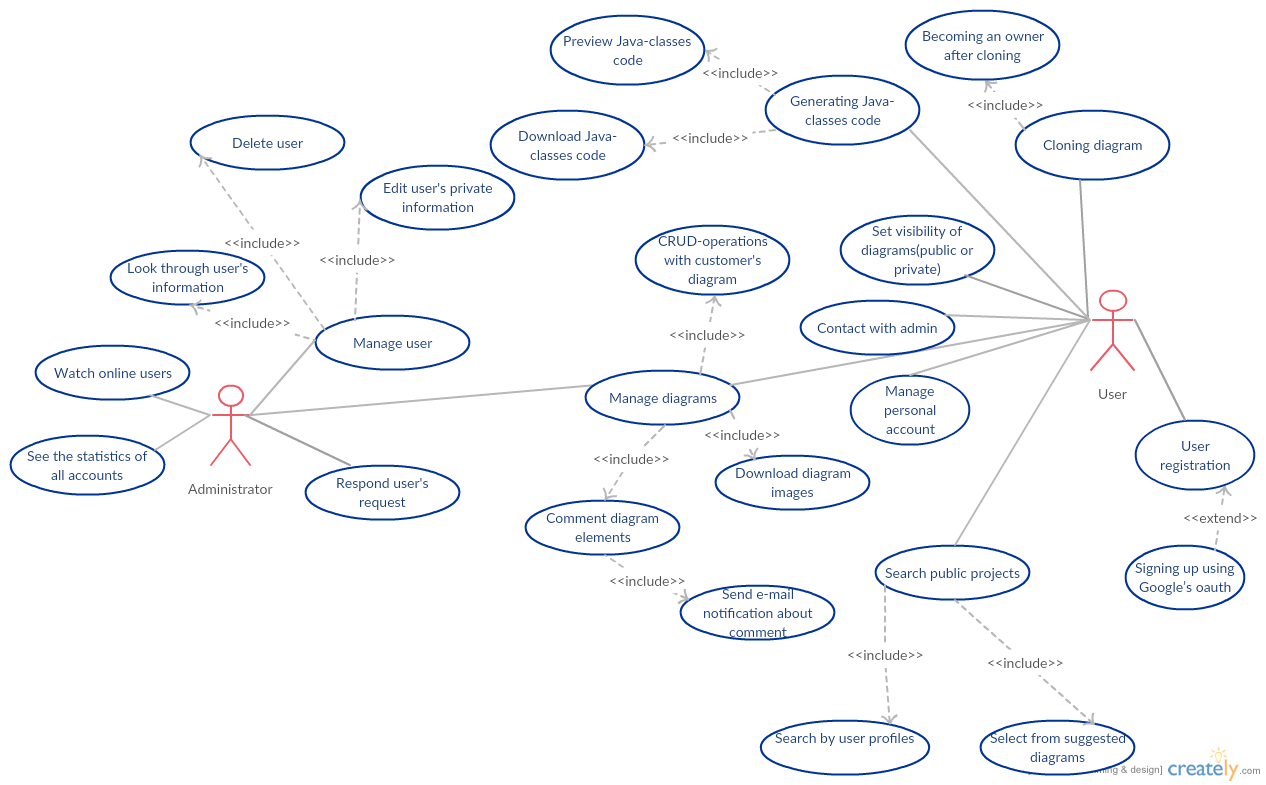
# Preliminary Epics and User Stories

### Preliminary use case diagram

As it can be seen in the figure 1, the system has such roles as an administrator, user.

Administrator performs functions concerning user and system management. His responsibilities include deleting, inserting, editing user’s private information. Also administrator is allowed to block user and establish the configuration of the whole application.

User is a core part of the application. User does whole work with regards to the diagram and account management. Those include creating a project, saving, editing, deleting it, generating Java source code, exporting diagrams into XML and PNG.



**Figure 1 UML-class diagram editor actors and use-cases**

### Preliminary epics

* Allow users of the system to create and manage their accounts.
* Allow team leads to create and manage projects and their teams.
* Allow administrators to manage all users and projects.
* Allow users to view and select projects, they take part in.
* Allow team members to create a UML 2 standard class diagram.
* Allow team members to import/export the diagram.
* Allow team members to communicate on the diagram.
* Allow team members to work simultaneously.
* Allow team members to comment parts of the diagram.
* Allow for the project to be saved and the change history to be reviewed.
* Allow team members to preview and download the Java classes, generated from the corresponding diagram.
* Allow users to get notifications via the email.

# Solution Highlights

The following should be considered during implementation:

* Final version of the application must be built till 15.12.2015.
* Draft version of the application must be finished during 2 sprints
* The application may be built with different technologies (e.g. ReactJS, Bootstrap, Spring MVC, Hibernate)
* Solution will be hosted on OpenShift (using JBoss) which will allow the Customer to avoid the needs of having local hosting environment.
* It is possible to use Tomcat servlet container for application deployment
* The system will support role-based access control. It will be possible to manage permissions for maintenance plans.
* It will be possible to export diagrams to PNG format. Save Java source code in .java-format, also the application will allow user to watch preview of the source code.
* The feature will be implemented to allow users to organize team rooms, where each team member will be allowed to edit shared diagram.

**3.2 Stakeholder Profiles**

This application will be interesting for users because of its accessibility, user-friendly and clear design. In addition, the interesting features may attract users to use the product.

|  |  |  |
| --- | --- | --- |
| **Interests** | **Limitations** | **Stakeholder** |
| Usable and simple tool for simplifying the developing java-applications | No limitations | User |
| Development of high quality software. | Funds and development time | Developer |
| Getting the finished product with the relevant documentation at the right time. | No limitations | Product Owner |

**3.3 Project Priorities**

Time is the main limitation for developer. We must emphasize that implementation of all main functionality and creating working user interface is the most prioritized goal. After doing this we will think about additional features of product.

|  |  |  |  |
| --- | --- | --- | --- |
| **Measuring** | **Key factor** | **Limitation** | **Degree of freedom** |
| Schedule | The product must be fully prepared and launched on December 15th in 2015. |  |  |
| Scope |  | All the basic functionality and the interface must be implemented until the release. |  |
| Quality |  | All the basic functionality and the interface must be tested until the release. |  |
| Team |  | There is a team, which consists of just 4 people. |  |
| Price |  | There is no financial support of this project. |  |

# Project dependencies

1. Application will have next dependencies:

**Spring MVC + AngularJS**

**Spring MVC**

Pros:

- easy configuration with annotations;

- integrates with many view options seamlessly: JSP/JSTL, Excel, PDF, JSON;

- good REST support;

Cons:

- instant reload not built-in, need JRebel or Spring Roo

- no open development process, need to be Spring Source;

- Requires 3rd-party libraries

**AngularJS**

Pros:

-enabling quick prototyping and delivery of dashboard-style, dynamic single-page-applications;

-fast development; very expressive, therefore need less code for same result as with other libraries;

Cons:

- basics are easy, but then learning curve becomes very steep making it hard to learn;

- scopes are easy to use, but hard to far;

- directives are powerful, but difficult to use;

**Redux Framework + React JS (FLUX)**

**Redux:**

Pros:

-time efficient;

-faster page load speeds;

-high web standards;

-easy to upgrade;

-increased functionality;

-design freedom;

-SEO(Search Engine Optimization);

Cons:

-cost;

-time to learn;

-framework limitations;

**React JS**

Pros:

-extremely easy to write UI tests cases. This is due to the virtual DOM system implemented entirely in JS;

-react will automatically manage all UI updates when your underlying data changes;

-ease of debugging;

-works nicely with commonJS / AMD patterns;

Cons:

-There's a learning curve for beginners whom are new to web development;

-Integrating reactJS into a traditional MVC framework such as rails would require some configuration. (i.e.: substituting erb with reactJS)

-It's not a full framework. There's no router nor model management libraries built into reactJS - unlike Angular or Ember.

We decided to use the second technology stack(Redux+React), because of its big variety of functionality, ease of debugging and good community.

## References

|  |  |  |  |
| --- | --- | --- | --- |
| **##** |  | **Date** | **Document Title** |
| 1. | UML2 | Specification | http://www.omg.org/spec/UML/2.0/Infrastructure/PDF |
| 2. | ECMA-262 | Language specification | http://www.ecma-international.org/publications/standards/Ecma-262.htm |
| 3. | Java | Language specification | https://docs.oracle.com/javase/specs/jvms/se8/jvms8.pdf |

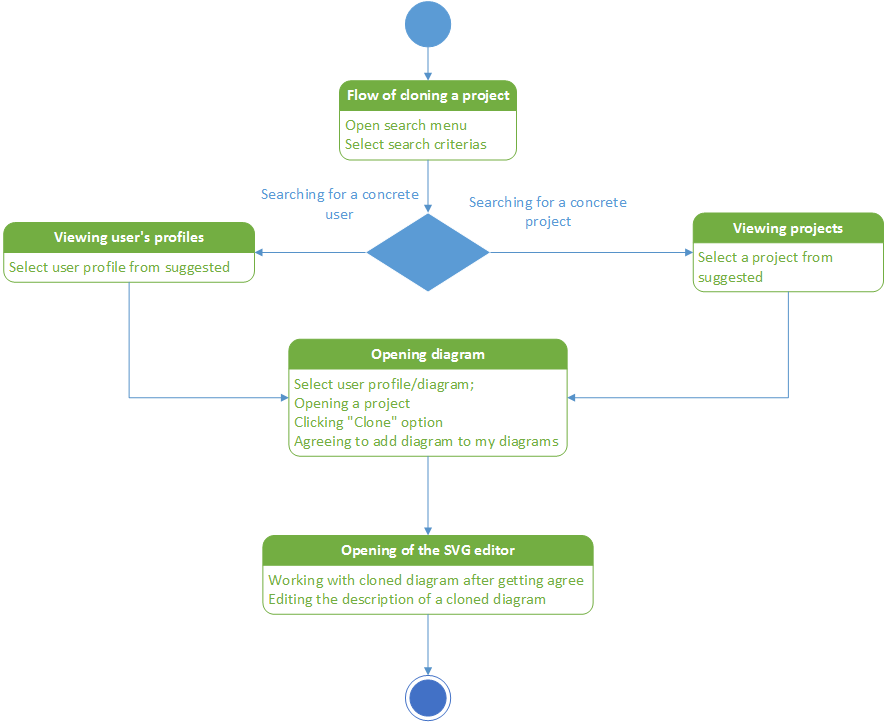
# Attachment A. UML diagrams of the application

1.png

**Figure 2 UML component diagram**

Sequence_diagram.png

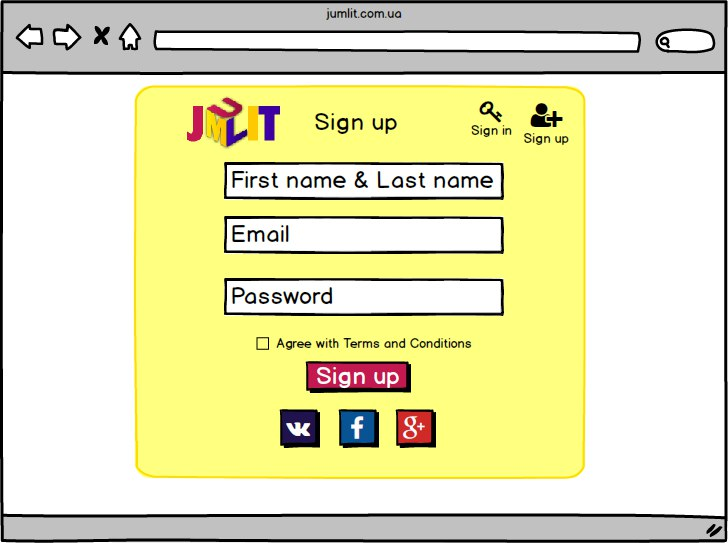
**Figure 3 UML Sequence diagram**



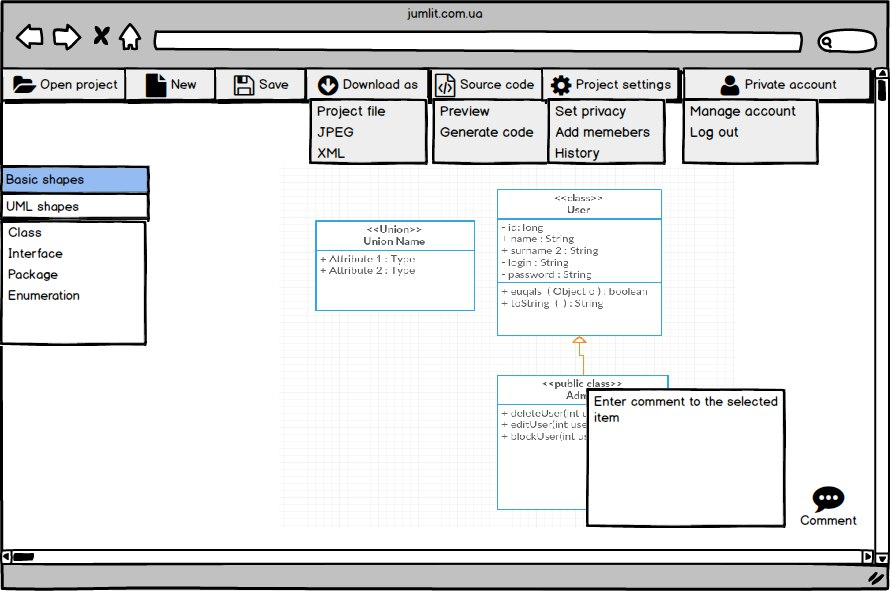
**Figure 4 UML activity diagram**



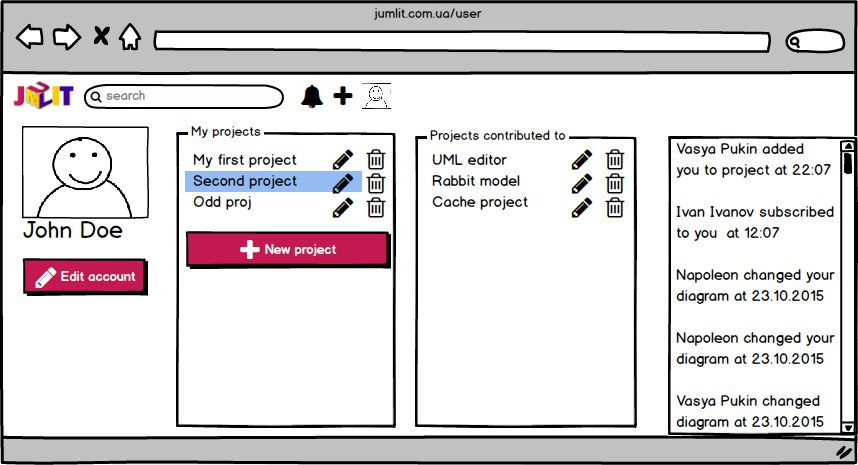
**Figure 5 Moqups of the sign in page.**



**Figure 6 Moqups of the sign up**



**Figure 7 Moqups of the main page**



**Figure 8 Moqups of the private account**