Documentation

Contents

1 Introduction **2**

2 General view of the application **3**

3 System's datacontents **5**

4 Relation data base chart **8**

5 System's architecture **9**

6 User interface **10**

7 Installation **12**

8 How to run and use the app **12**

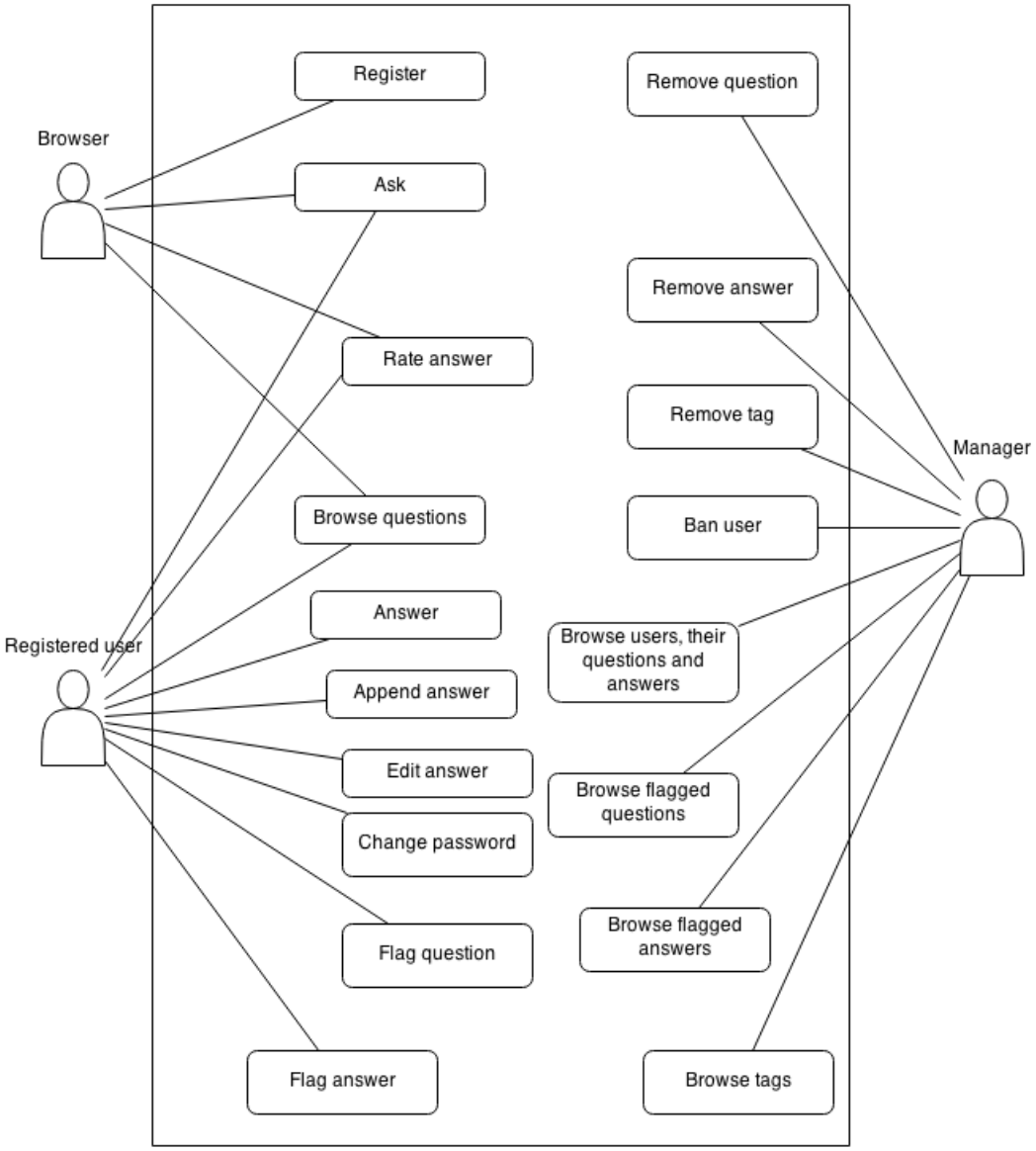
1 Introduction

This web application is a forum of questions and answers, akin to *stackoverflow* and *yahoo answers*. The web service will let a user find solutions to problems either by browsing the web site or asking a question oneself. Questions can be answered by other users and each answer can be rated. Highest rated solution offers are shown at the top. With enough questions and answers in the database, the service can provide a quick and convenient way to look up a solution for a problem of any kind. Tags can be added to the asked question to help browsing questions by category.

Service will be hosted on university's users-server via Tomcat. Code will be written in Java via NetBeans IDE. App will use PostgreSQL database. User's browser will not require other scripts or plug-ins.

2 General view of the application

The following is an actor chart, followed with a description for each user and action.



Browser: Unregistered user, that got access to the webapp simply by retrieving an url-link.
Registered user: User that has access to exclusive features by creating an account and logging into the app.
Manager: Webapp manager obligated to managing the forum, keeping it accessible to other users.

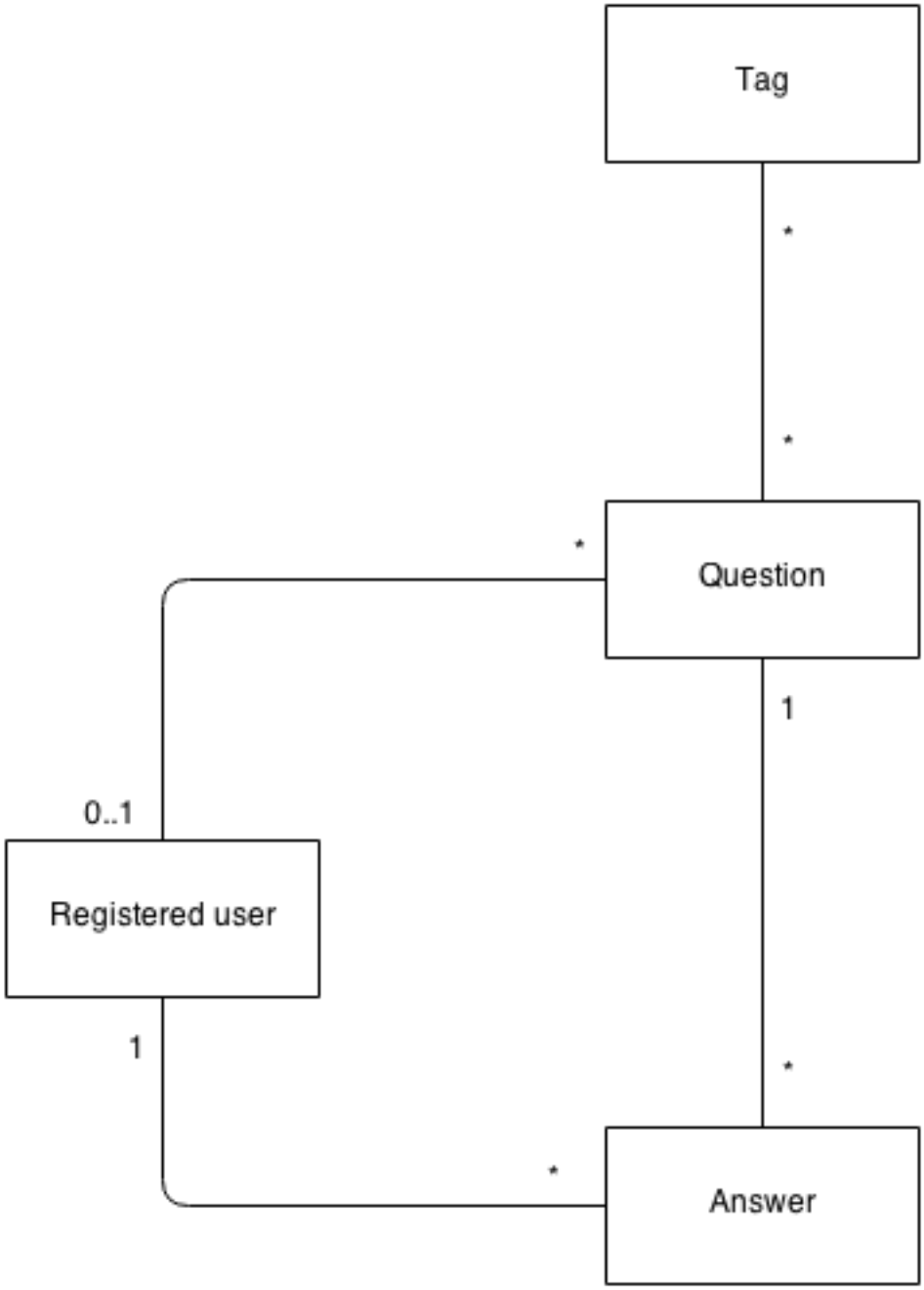
User actions:

Register: Browser creates an account and promotes oneself into a registered user. User can then login to that account at any time if not banned.
Ask: User creates a question, awaiting a set of answers. Also sets tags for the question to improve the transparency of the question.
Rate answer: User marks answer as helpful and correct. Answer variant with most ratings is shown on top.
Browse questions: Lists questions by tags, (possibly a search function).
Answer: Adds an answer to a specified question.
Append answer: User edits one's answer by adding information into it. This way of editing will retain the rating of the answer.
Edit answer: User edits one's answer freely. This will reset the rating for the answer.
Change password: Registered user changes one's password.
Flag question: Notifies manager that specified question is inappropriate.
Flag answer: Notifies manager that specified answer is inappropriate.

Remove question: Removes the specified question along with its answers.
Remove answer: Removes the specified answer.
Remove tag: Removes a tag from the database.
Ban user: Removes the user from the database. Questions and answers made by the user are either marked as 'banned', or removed completely.
Browse users, their questions and answers
Browse flagged questions: Lists questions by their amount of flags. Helps finding inappropriate questions.
Browse flagged answers: Lists answers by their amount of flags. Helps finding inappropriate answers.
Browse tags: Lists tags. Whether they are inappropriate or not is apparent directly from the tag.

3 System's datacontents

The following is a chart representing the required data contents of the system on a conceptual level:



Four datablocks shown in the chart have following properties:

Datablock: Registered user

Property	Value interval	Description
Nickname/Username	16 characters	User's calling name in the forum. Can't be empty.
E-mail	64 characters	User's email to contact one. Can be omitted.
Password	128 characters	User's password to access the account. Can't be empty.
Time user registered the account	Date and time	
Avatar	512x512 sized picture	User's picture to identify him by. Can be omitted.

Datablock: Question

Property	Value interval	Description
Question title	96 characters	Question's essence displayed in search results. Can't be empty
Question body	Any amount of text	Main part of the question.
Time when question was asked	Date and time	
Amount of flags for the question	Integer starting from 0	How many times question was deemed by users as inappropriate. Questions with most flags are easier for a manager to find.

Datablock: Tag

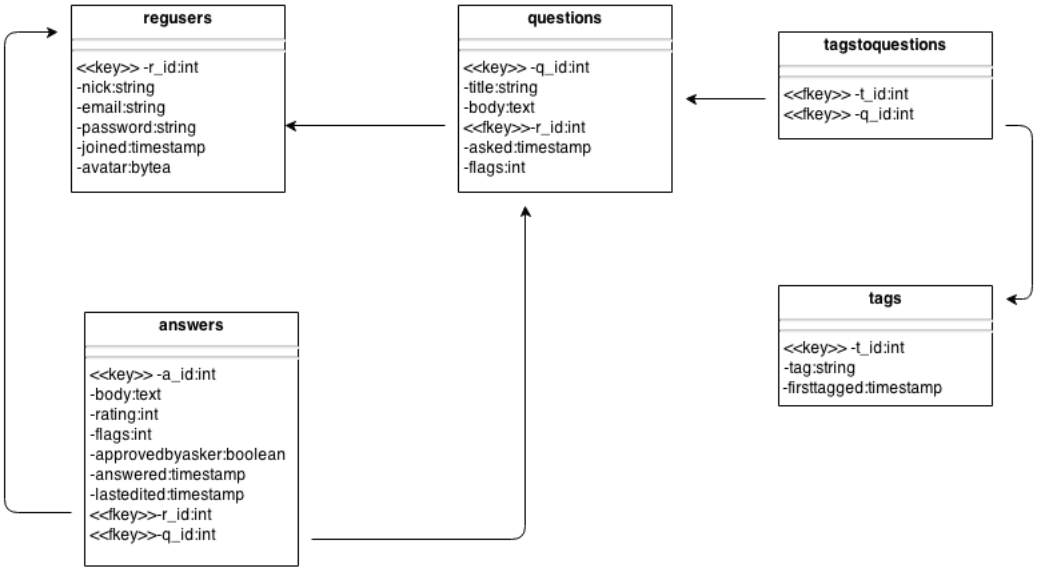
Property	Value interval	Description
Tag title	12 characters	Can't be empty
Time when tagged	Date and time	Time when tag was first used.

Datablock: Answer

Property	Value interval	Description
Answer body	Any amount of text	
Rating	Integer starting from 0	Amount of positive points other users have given to this answer.
Amount of flags for the answer	Integer starting from 0	How many times answer was deemed by users as inappropriate. Answers with most flags are easier for a manager to find.
Approved by asker	Boolean	True if poster of the question which this answer addresses has rated this answer.
Time when answer was posted	Date and time	This value is reset if answer is edited.
Time when answer was last edited	Date and time	This value is reset if new information is appended to the answer.

User can post multiple questions and answers. Each question and answer belongs to one user, although question may be posted anonymously, that way it won't belong to any user. Question may contain several answers. Each answer belongs to one question. Question may contain several tags and several question may be tagged with the same tag.

4 Relation data base chart



5 System's architecture

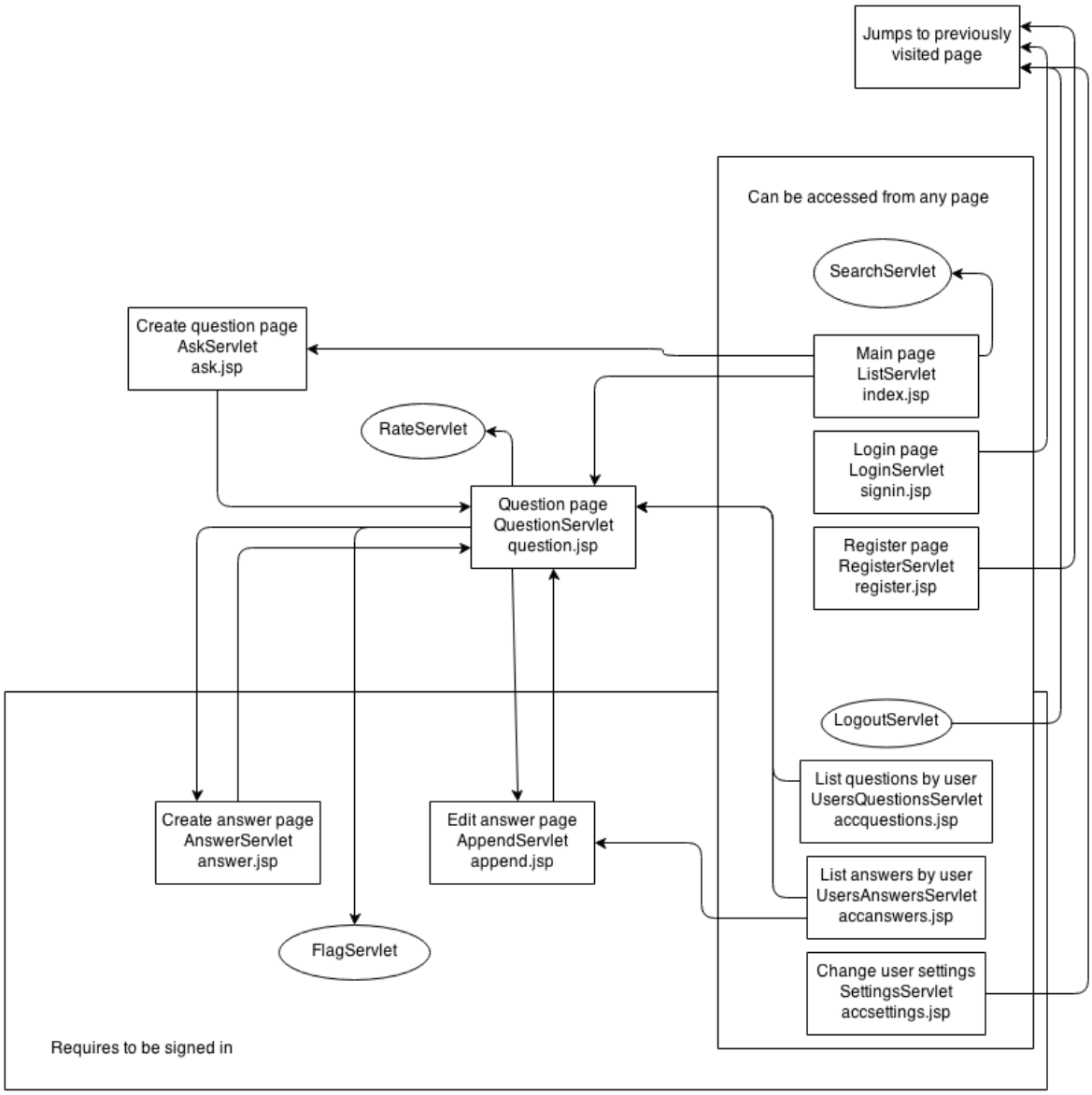
This app uses the traditional *MVC-model*, meaning that the code is divided to three parts:
Models, which access database data and stores it into objects.
Views, which is client-side code mainly consisting of html-code.
Controllers, which let correct users access correct app controls under correct premises.

Models are stored in the Models package, which has four classes accessing four main database tables (registered user, question, answer, tag). Code for forming and closing connections with the database is placed at *QAConnection* class.
Views are stored in the *web* directory, outside *src* directory where most of the code is placed. Each webpage of the app is written in a .jsp file. Some parts of the html-code are stored in tag files, which are then referenced in .jsp files. These are placed at *web/WEB-INF/tags*.
Controllers are synonymous with Servlets in this app. These are placed in *Servlets* package, controls exclusive to registered users are stored at *Servlets.RegisteredUser* and controls exclusive to moderators are stored at *Servlets.Moderator*. Each concrete Servlet inherits QAServlet, which provides useful methods that are often used.

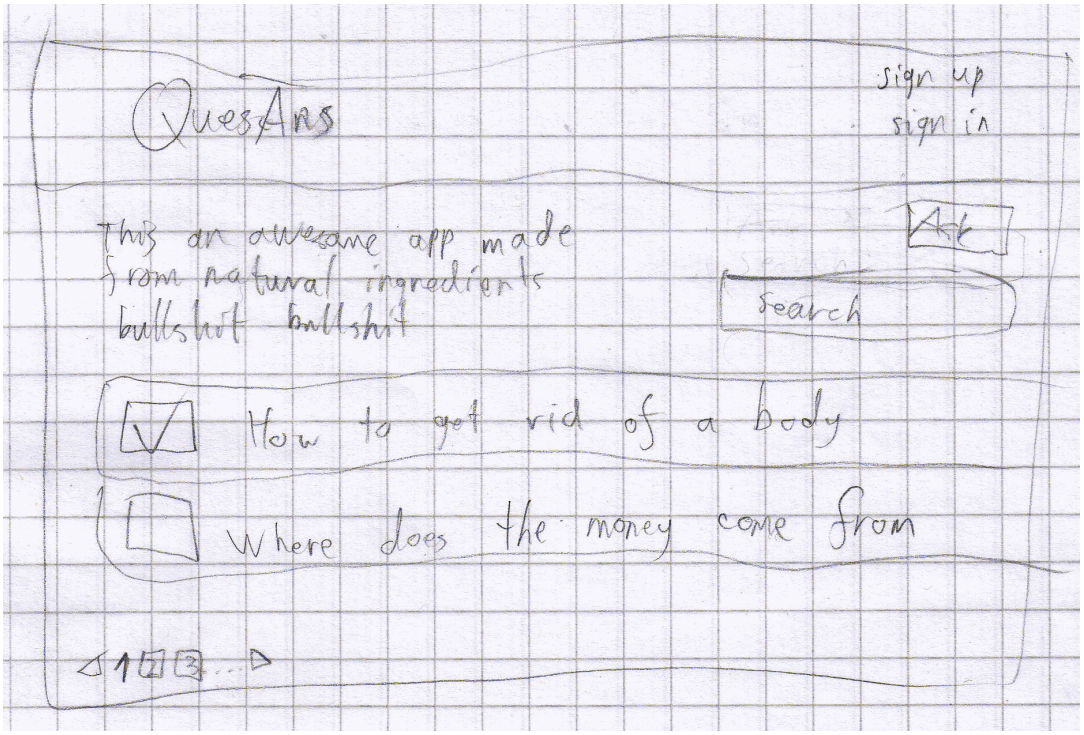
Sessions are used to store the user currently signed in and previous URL. For example, if a user browses questions and only then decides to log in, after entering the information app redirects user to the question that was last browsed using the saved URL.

6 User interface

The following is a map which represents all the sites that are accessible in the application, and what sites user can access next from the specified site:



This is a design concept of the front page. Notice that it lists asked questions, offers user to ask a question, sign in or register.



7 Installation

To install the application, copy *QuesAns.war* file to *tomcat/webapps/* folder. Copying the file should be enough to make the tomcat unpack the file within the *.war* file.
Next, configure *context.xml* file at *tomcat/webapps/QuesAns/META-INF/* to connect to the Postgres database. Use *context.xml.dist* for reference.
Now, as long as tomcat is running, web application should now be working.

8 How to run and use the app

Web application is accessed at the url: *servername.com/QuesAns/*. At the moment of writing it is accessed at *http://t-pasmpasm.users.cs.helsinki.fi/QuesAns/*. Press sign up at top-right corner to get started by registering a user. Notice that reading other questions and even asking a question doesn't require the user to be registered.