Context Project: Cultural Heritage

Monumentzo

Analysis and Design

Group 2

Date
28 March 2012

Group members
Bojana Dumeljic (4103246)
Mick van Gelderen (4091566)
Kevin de Quillettes (4077482)
Salim Salmi (4089715)

Advisors
Martha Larson
Christoph Kofler (Teaching Assistant)

Table of contents

Table of contents
1. Introduction
2. Current system
    2.1. Rijksmonumenten
    2.2. Deventer op de Kaart
3. Proposed system
    3.1. Overview
    3.2. Functional requirements
    3.3. Nonfunctional requirements
        3.3.1. Constraints
        3.3.2. Documentation
        3.3.3. Failure Management
        3.3.4. Language
        3.3.5. Performance
        3.3.6. Platform Compatibility
        3.3.7. Robustness
    3.4. Analysis models
        3.4.1. Use case diagram
        3.4.2. Use cases
            3.4.2.1. View monument
            3.4.2.2. Browse by time
            3.4.2.3. Browse by attribute
            3.4.2.4. Browse by place
            3.4.2.5. Information from an external source
            3.4.2.6. Search by name
            3.4.2.7. Search by place
            3.4.2.8. Leave a comment
            3.4.2.9. Login
            3.4.2.10. Register
        3.4.3. Business object model
        3.4.4. Dynamic models
            3.4.4.1 State Diagram
            3.4.4.2 Sequence Diagram
    3.5. User interface
4. Glossary

1. Introduction

This document contains the analysis and design for Monumentzo. Monumentzo is a system that helps its users learn about monuments in the Netherlands.

Our goal is to make a system that is as intuitive and interactive as possible. Thereby making it a fun user experience that will make the user want to look further and thus learn more.

The system's main feature is its graphically rich interface that is designed to be as user friendly as possible which should make the system simple to use and thus intuitive.

The outline of this document is as follows. The second chapter describes existing systems that provide functionality similar to that of the system.

In chapter three the analysis and design can be found of the proposed system, Monumentzo. The third chapter first discusses the requirements and constraints of the system, then the analysis models and finally the user interface concept.

Finally, chapter four contains a glossary explaining the jargon used in this document.

2. Current system

Monumentzo is not a direct improvement of an existing system. The functionality that it will provide is similar to a number of existing systems. Two of these systems will be addressed in this chapter.

2.1. Rijksmonumenten

The Rijksmonumenten system is a very rich one. It has applications on multiple platforms. The web version[1] provides a map on which you can select a monument. It is possible to filter the visible monuments and also to search for monuments by address.

When you have selected a monument it shows the name and address of the monument along with extensive information and multiple photos if available, which they are for a lot of the monuments.

The mobile application is available for iPhone, Android and Nokia OVI. The Android version incorporates the Layar technology. The app can show a map, list or an augmented reality view. It is possible to view the interior completely by means of panorama pictures that have been taken. Users can also upload photos directly to Wikimedia Commons from their iPhone.

The data sources are: Rijksdienst voor het Cultureel Erfgoed, Wikipedia, Wikimedia Commons, Flickr, historical archives and museums.

Weaknesses:

- The style of the web interface is inconsistent.
- When there are more than 60 images available there is a next page button, but no previous page button.
- There is no way of commenting on the monuments or providing feedback on the correctness of the information that is provided.

2.2. Deventer op de Kaart

The Deventer op de Kaart website[2] is similar to the web version Rijksmonumenten but simpler. It features information from Rijksdienst voor het Cultureel Erfgoed and Wikipedia, photos, commenting, Google street view integration and hiking routes.

Weaknesses:

- It is pretty likely that the data has been manually added by administrators for the larger part. It is not realistic to do this for all the monuments in the Netherlands.
- There is no way to filter for a certain type of monument.
- Not a single comment has been placed for 15 of the 41 inspected monuments .

[1] www.rijksmonumenten.info/kaart

[2] www.deventeropdekaart.nl

3. Proposed system

3.1. Overview

In this chapter the requirements, the use cases, the use case diagram, the abstract class diagram, the sequence diagram and the state diagram for Monumentzo are discussed.

These were based on the following user story:

My name is Noah and I am a real estate agent. For my job I sell old houses which sometimes have characteristics of monuments. Therefore I would like to be able to look up monuments that have these attributes. I would also like to get an overview of these monuments on a map. I can then proceed to select one of the monuments that I want more information about.

When I select a monument I would like to see additional information related to that monument. This information should consist of the name, location, age and architectural attributes but also the past events or people related to the monument. I would like to have as many images as possible to get a good view of the monument and I would also like to see its location on a map.

I would like to see articles related to the monument for example from Wikipedia. I am also interested to know if there are videos related to this monument. Finally for further research I could use a list of books which are also related.

I would also like to be able to have an account. This way I would be able to save monuments so I can look them up later. I would also like to be able to save the books in my personal read list.

Using this system I would be able to learn more about the architectural attributes that some of the houses I sell have. This way I could inform my clients better about the house I am trying to sell them. Which would boost my sales but would also help me acquire interesting historical and architectural knowledge.

3.2. Functional requirements

1. **Let user register.**
   A user can make an account for himself on the system. They have to fill in a username, password and email address. Each username has to be unique and the email address has to be valid.
2. **Let user log in and out.**
   A registered user can log in to the system by typing in their username and corresponding password.
3. **Provide user with information about each monument.**
   When a user views a monument they are provided with the following information about that monument:
   - Name
   - Images
   - Location on a map
   - Date of construction
   - Category
   - Architectural attributes
   - Related articles
   - Related books
   - Related historical events
   - Related people
   - Related videos

   The data is either fetched from the database or obtained from external resources.
4. **Option to browse.**
   a. Browsing can be done by:
      - Place
      - Time
      - Architectural attribute
      - Category

   b. Results are displayed in a 3D environment where the items are ordered according to the selected browse option. Multiple browsing options can be selected up to the number of axis.
5. **Option to search.**
   Searching for monuments can be done using the following criteria:
   - Name
   - Address

   After providing the system with a query the user gets a list of monuments back from the system that satisfy the query. Each result shows at least the name and image of the monument.

6. **Gather information from external sources.**
   Using the APIs of the following resources data will be either stored in the database or shown to the user real time. We will use the following APIs:
   - Wikipedia
   - Wikibooks
   - Wikimedia Commons
   - Vimeo
   - YouTube
   - Google Books
   - Rijksmonumenten.info
   - Rijksdienst voor het Cultureel Erfgoed
7. **Option to save a monument to list so the user can look them up later.**
   a. There should be three different lists:
      - Favorite monuments
      - Visited monuments
      - Monuments that a user wants to visit
   b. Each monument in the lists should be shown on a map. Differentiation between the monuments in the different lists will be made by different colors for each monument.
8. **Tag enrichment by descriptions.**
   Gather the monuments architectural attributes by looking for certain words in the description obtained from the Monument Registry and add these words as information to the monument in the form of tags.
9. **Show graphically similar monuments.**
   All obtained images of the monuments should be indexed using Lire[3]. Also using Lire monuments with similar graphical attributes can be found by comparing images of the monuments. This relationship between monuments is saved in the database. When a user looks at a monument that has a graphically similar monument present in the database than the other monument is shown as a recommendation to the user.
10. **Read list.**
    A user can save or remove books to and from their personal read list. These books are either from Google Books or Wikibooks. The read lists consists of links to the actual books. This way the user can read them online.
11. **Comments.**
    Provide users with the option to leave comments about each monument.

[3] **Lire**: A tool used for content based image retrieval. (http://www.semanticmetadata.net/lire/)

3.3. Nonfunctional requirements

3.3.1. Constraints

- There are only two types of users, anonymous and registered ones.
- Everyone has to be able to search, browse and view monuments. From now on this type of user will be called the anonymous user as was done in the diagram 3.1 that displays the use case diagram.
- Only registered and logged in users can perform all actions. This type will be called users.

3.3.2. Documentation

- Documentation has to be kept up to date with each development iteration.

3.3.3. Failure Management

- If errors occur the error page should give non-cryptic information to the user about what went wrong.

3.3.4. Language

- All the text visible to the user in the application is in Dutch.
- Everything else is in English.

3.3.5. Performance

- The user should get a response of the system within 10 seconds. Not all the data has to load but the user should know that there are things happening (for example a loading bar or text).
- At least 25 users should be able to use the system simultaneously.

3.3.6. Platform Compatibility

- Works in Chrome and Firefox. Not in Internet Explorer.
- Written in HTML5, PHP, CSS3, JavaScript and OpenGL.

3.3.7. Robustness

- When there is no information to show this should be resolved by displaying a message that there is no information instead of a blank page.
- When there are no pictures of a monument the top down google satellite view should be used as the image.

3.4. Analysis models

3.4.1. Use case diagram

The use case diagram shown in diagram 3.1 shows the main use cases of the system. It also shows which actors interact with which use cases and how the different functionalities are built up from multiple use cases.

The blue use cases are the ones that can be performed by an anonymous user. The anonymous user himself is also blue. This is a user that does not have an account or is not logged in. A user with an account that is logged in can be do all the actions shown in the diagram.

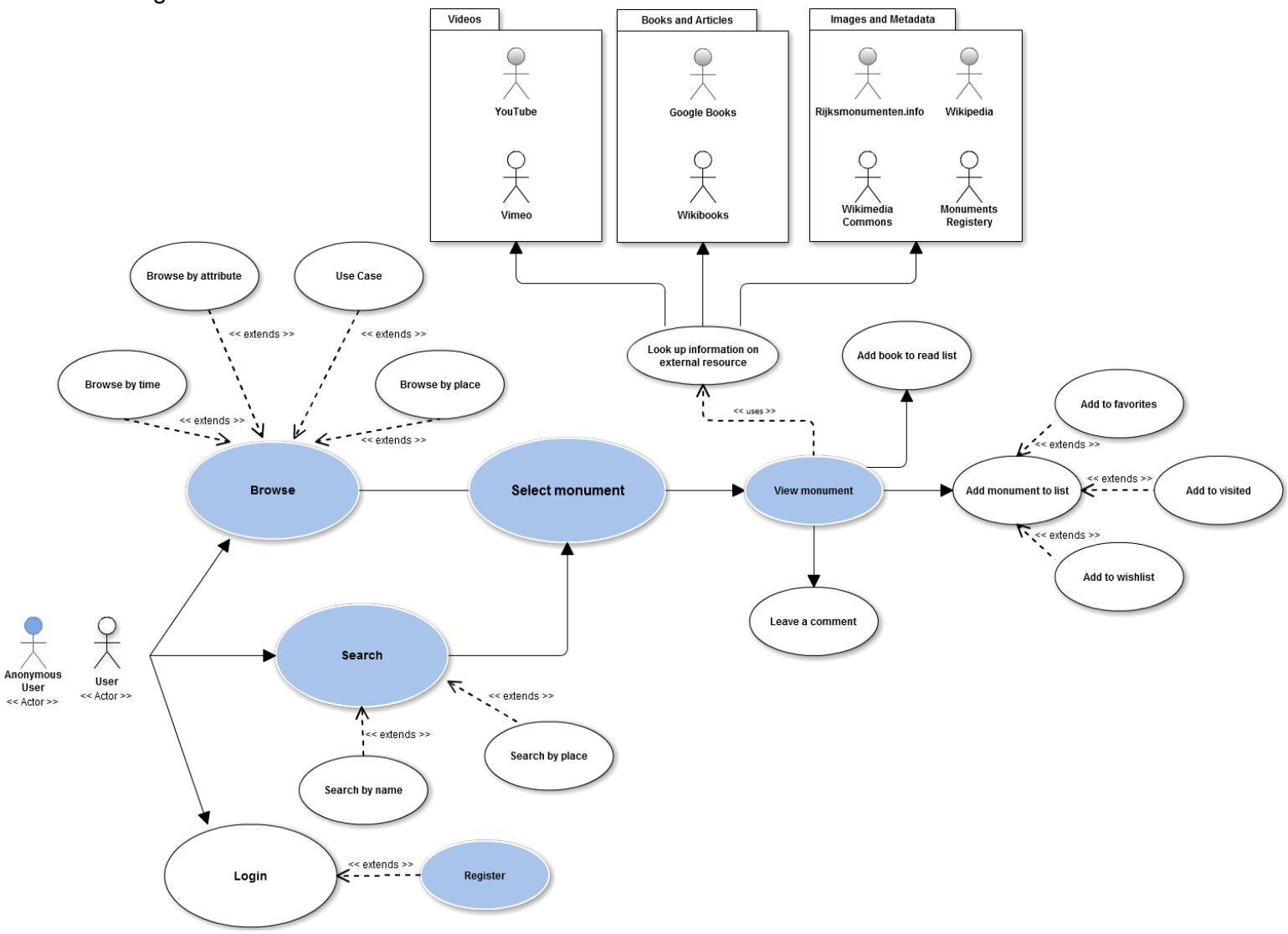

Diagram 3.1: The use case diagram

3.4.2. Use cases

3.4.2.1. View monument

Use case	**View monument**
Actors(s)	User, Anonymous user
Goal(s)	The user gets to see information and learn about a selected monument.
Trigger(s)	The user requests to see data for a particular monument.
Preconditions	User has found a monument that he wants more information about.
Postcondition on success	The user sees a page with detailed information about the selected monument.
Postcondition on failure	The user sees an error page explaining what went wrong.
Description basic course	1. The system is asked to find information about the viewed monument (use case: Information from an external source). 2. The system displays information about the selected monument. 3. The user reads and learns from the given information about the selected monument.
Related use cases	Information from an external source
Related requirements	#3 and #6

3.4.2.2. Browse by time

Use case	**Browse by time**
Actor(s)	User, Anonymous user
Goal(s)	The user is able to browse a list of monuments and to filter the results.
Trigger(s)	The user selects to browse by time and possibly sets filters.
Preconditions	Browse by time must be selected.
Postcondition on success	When the user selects the browse by time option the resulting monuments are ordered in the 3D environment by their construction date as was stated in the monuments register. Now the user can click on a monument to go to a page that shows detailed information about it or browse further either by time, attribute or place.
Postcondition on failure	The user sees an error page explaining what went wrong.
Description basic course	1. The system displays the monuments in the 3D environment. 2. The user browses the list of monuments.
Related use cases	View monument
Related requirements	#3 and #4

3.4.2.3. Browse by attribute

Use case	**Browse by attribute**
Actor(s)	User, Anonymous user
Goal(s)	User browses for monuments with a certain attribute.
Trigger(s)	User clicks on an attribute that he wants to browse by.
Preconditions	Browse by attributes must be selected
Summary	When the user selects the browse by attribute option the resulting monuments are grouped in the 3D environment by attribute. Now the user can click on a monument to go to a page that shows detailed information about it or browse further either by time, attribute or place.
Postconditions	The blocks in the 3D environment are ordered in separate groups. Each group represents an attribute.
Related use cases	View monument
Related requirements	#3 and #4

3.4.2.4. Browse by place

Use case	**Browse by place**
Actor(s)	User, Anonymous user
Goal(s)	The user wants to find monuments near a place.
Trigger(s)	The user selects the browse by place option.
Preconditions	Browse by place must be selected.
Summary	The monuments are sorted by place. Monuments with close geographical coordinates are shown close to each other and vice versa. Now the user can click on the monuments to go to a page that shows more information about it or browse further either by time, attribute or place.
Postconditions	Monuments are sorted by geographical coordinates.
Related use cases	View monument
Related requirements	#3 and #4

3.4.2.5. Information from an external source

Use case	**Information from an external source**
Actor(s)	User, External source
Goal(s)	Show information about a monument.
Trigger(s)	User clicks on a monument to get more information about it.
Preconditions	Information has to be retrieved from the database and from the external sources.
Summary	When a user goes to an item page they can get external information such as wikipedia articles and media content.
Postconditions	Information is retrieved from external source.
Related use cases	View monument
Related requirements	#3 and #6

3.4.2.6. Search by name

<table>
<tr><td>Use case</td><td>Search by name</td></tr>
<tr><td>Actor(s)</td><td>User, Anonymous user</td></tr>
<tr><td>Goal(s)</td><td>Find monuments with the specified name.</td></tr>
<tr><td>Trigger(s)</td><td>User fills in a name in the search field and presses the search button.</td></tr>
<tr><td>Preconditions</td><td>A search query has been written in the search field.</td></tr>
<tr><td>Postcondition on success</td><td>A list of monuments is provided which have the given name as a substring in their actual name.</td></tr>
<tr><td>Postcondition on failure</td><td>A message is displayed that lets the user know that no monuments could be found using the input they provided.</td></tr>
<tr><td>Description basic course</td><td>
<table>
<tr><th>Actor</th><th>System</th></tr>
<tr><td>1. The user types in a name in the search field.
2. The user clicks on the search button.

4. The user selects a monument from the results.</td><td>3. The system tries to find a monument in its database which name includes the search query as a substring and displays the results to the user.</td></tr>
</table>
</td></tr>
<tr><td>Related use cases</td><td>Search by place, View monument</td></tr>
<tr><td>Related requirements</td><td>#3 and #5</td></tr>
</table>

3.4.2.7. Search by place

Use case	**Search by place**
Actor(s)	User, Anonymous user
Goal(s)	User searches for monuments near a certain place
Trigger(s)	User clicks/fills in the name of the place near to the monument.
Preconditions	User has to know the location of the monument.
Summary	When the user fills in the location of the monument or a location that is close by the monument he is redirected to a page with a list of all the monuments in the database that have this location or are near it. The list is made based on how far the monument is from the given place. Now the user can click on the monuments to go to a page that holds more information about the monument or type in a new search query.
Postconditions	User sees a list of monuments near a certain location.
Related use cases	View monument
Related requirements	#3 and #5

3.4.2.8. Leave a comment

Use case	**Leave a comment**
Actor(s)	User
Goal(s)	User can leave their thought, impressions, etcetera about the monument.
Trigger(s)	User clicks on comment button then types in a comment in a dialog box and hits another button to either confirm or cancel the placement of the comment.
Preconditions	User has to be on the information page of a monument.
Summary	When the user is on the page of the monument there will be a button that indicates the user can comment if he clicks on it. When the user clicks the button a dialog box, a save button and a cancel button will appear. The user can type in the message and save it or he can hit the cancel button and the action will be stopped.
Postconditions	The comment is saved for other users to see.
Related use cases	View monument
Related requirements	#3 and #11

3.4.2.9. Login

Use case	**Login**
Actor(s)	User
Goal(s)	The user is logged in so he or she is able to do member actions, like commenting on monuments.
Trigger(s)	The user selects to login
Preconditions	The user has a registered user account (use case register)
Summary	1. The user fills in his/her username and password. 2. The system checks the username and password combination 3a. [Username and password combination is valid] The system goes to postconditions on success. 3b. [Username and password combination is invalid] The system goes to postconditions on failure.
Postconditions on success	The user is logged in.
Postconditions on failure	The user is not logged.
Related use cases	Register
Related requirements	#2

3.4.2.10. Register

Use case	**Register**
Actor(s)	Anonymous user
Goal(s)	Make a new account so the user can login and comment
Trigger(s)	The user selects to register a new account
Preconditions	The register dialog is shown correctly.
Summary	1. The user fills a form with necessary information, among which are a unique username and a password of the user's preference. 2. The user clicks the register button 3. The system checks the information that the user provided. 4a. [Information correct] The system goes to postconditions on success 4b. [Information incorrect] The system goes to postconditions on failure.
Postconditions on success	The system has created a new account for the user and lets the user know that it has created a new account. Also redirects the user to the login page, so the user can login with his/her new account (use case login).
Postconditions on failure	The system lets the user know that something went wrong and if the systems thinks that the user can fix his/her fault then it goes back to the information form and lets the user correct the fault.
Related use cases	Login
Related requirements	#1

3.4.3. Business object model

Diagram 3.2 contains the abstract class diagram for the system. It shows the main classes: user, monument, comment, read list and monument list. In more detail it shows their attributes and functions and also the relationships between the classes. The monument class has three different types: visited, favorite and wish. These are shown with inheritance.
Furthermore it shows that the browse and search classes will use the monument class to display their results to the user.

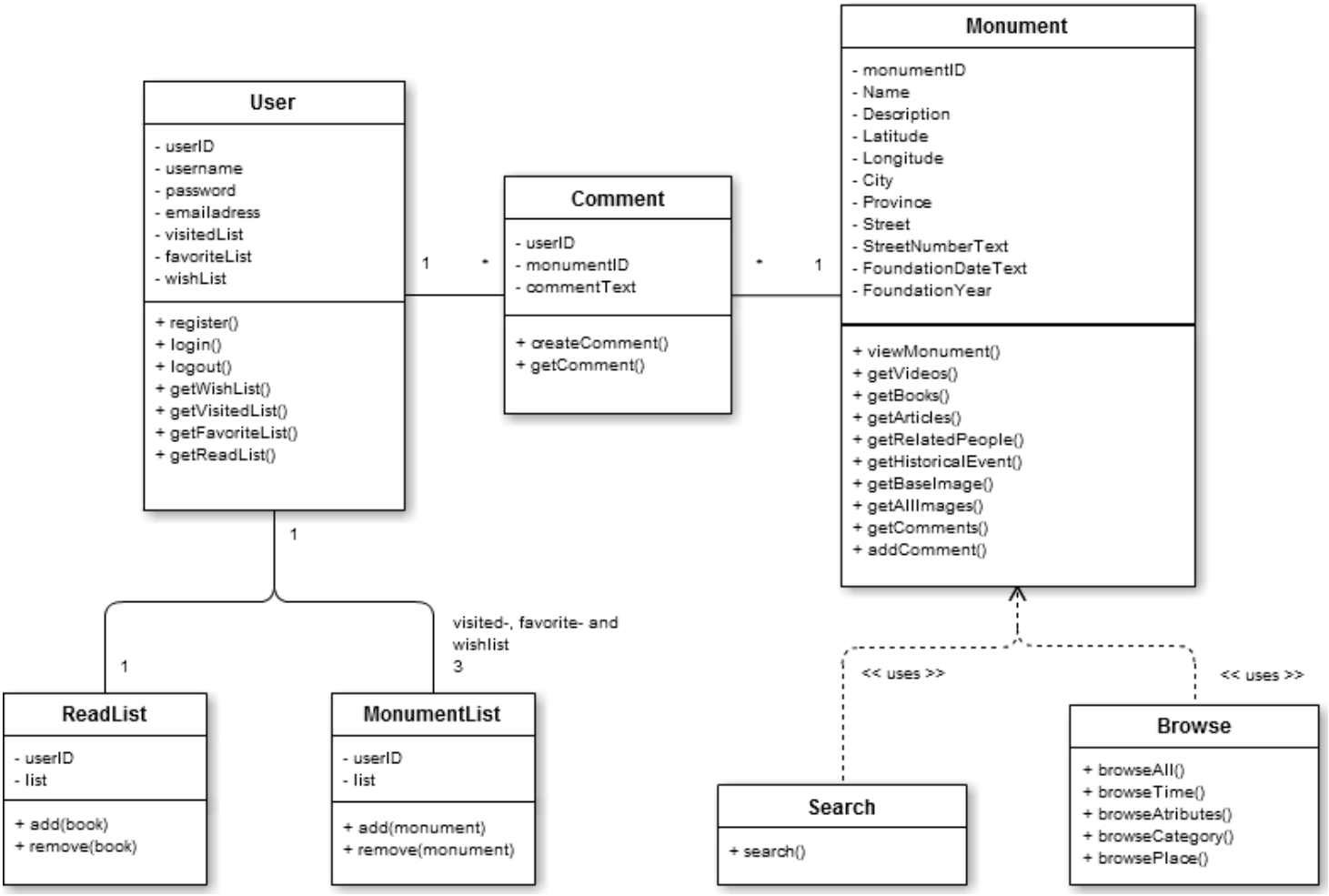

Diagram 3.2: Abstract class diagram

3.4.4. Dynamic models

3.4.4.1 State Diagram

Monumentzo has three main states. The main states are browsing, searching and viewing an item and the user is able to switch from each main state to one of the other states. The way how the states interact and the interactions the user can make to change the current state are shown below.

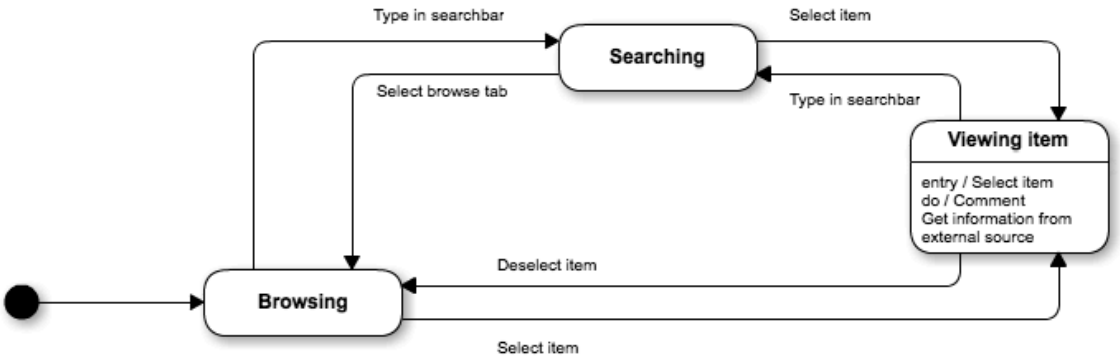

Diagram 3.3: The state diagram

3.4.4.2 Sequence Diagrams

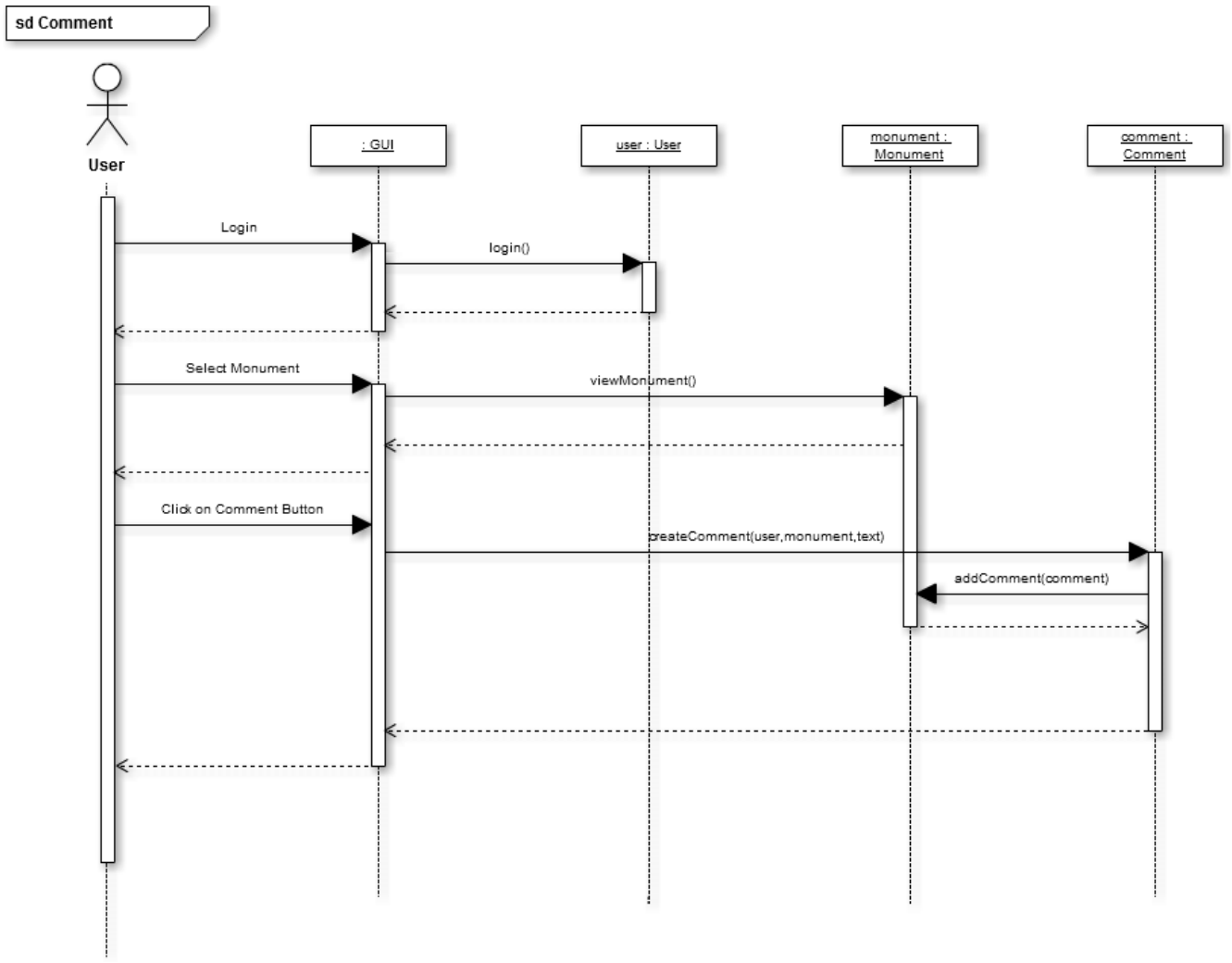

Diagram 3.4.1: The sequence diagram for comment

Diagram 3.4.1 is the first sequence diagram. In this diagram you can see what happens when a user wants to place a comment. The user first has to log in. When the user is logged in he can then proceed to click on a monument to view it. While he is on the monument page he can leave comments there. When he types something in the field and submits it, the comment is saved for that particular monument by storing it into the database.

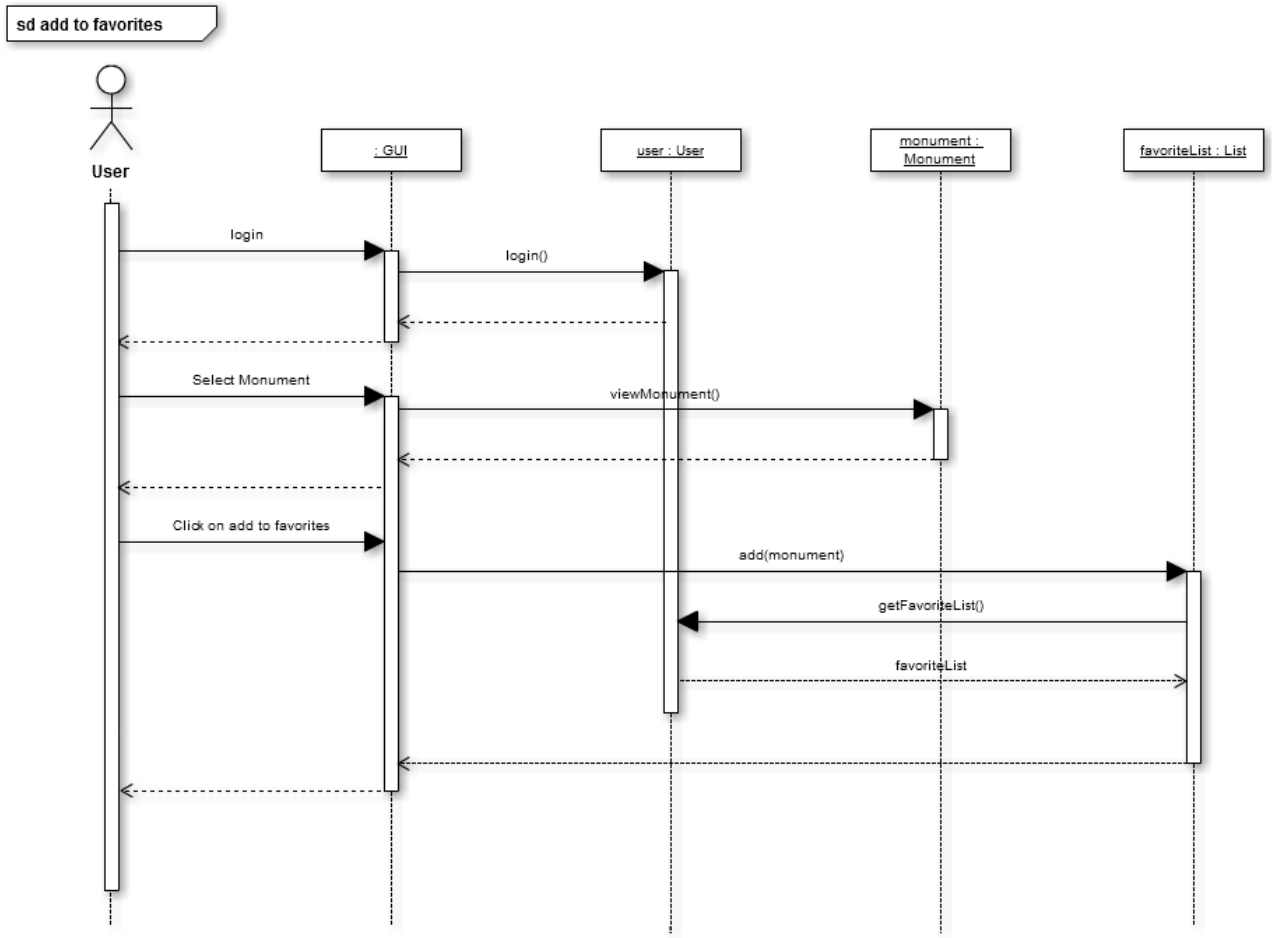

Diagram 3.4.2: The sequence diagram for add to favorites

Diagram 3.4.2 is the second sequence diagram. The above diagram shows you what happens when a user wants to add a monument to his favorite list. The user has to be logged in to do this. To add the monument, a registered user must be viewing the monument. While he is on the monument page he can click on a button to add it to his favorite list. Internally, the monument is then added to the favorites list of the user. These changes are stored in the database.

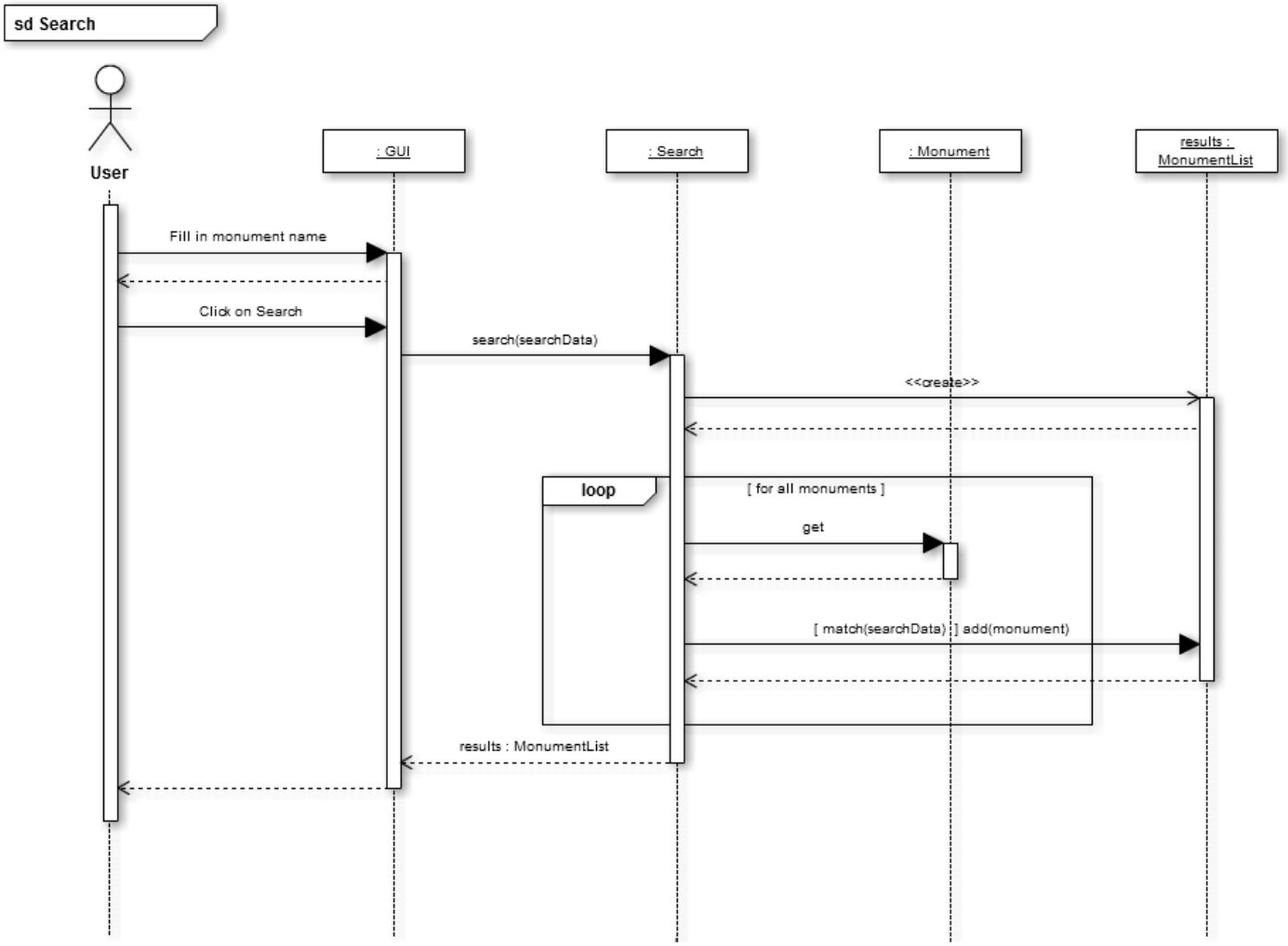

Diagram 3.4.3: The sequence diagram for search

Diagram 3.4.3 is the third sequence diagram. In this diagram you can see what happens when a user wants to search a monument by filling in its name. The user fills in a name in the search bar and submits it. The searchdata will then be compared to the monuments data and any matching monuments will be added to the results. The results with then be displayed to the user.

3.5. User interface

The user interface should be simple and neat. The browsing experience of the user should be as user friendly as possible by providing the user with a graphically rich interface. Which would also make it easy to use. It should be easy to navigate with clear and unambiguous buttons. The user interface plays a very large part in the system because we need it to be as simple to use as possible to make the learning process intuitive. It has to be graphically rich and fun to use to keep the user's attention and make them want to interact with the system a little bit longer.

An initial design for the user interface is shown in the figures below. Figure 3.1 and 3.2 show the register and login dialogs respectively. When a user clicks on the register or login button the respective boxes will appear. The background will be darkened out so the focus is solely on the boxes. These two figures display requirement 1 and 2.

The user has the conventional option to search specified in requirement 5. They can type a name or place in the search bar and are then directed to a page with the search results as shown in figure 3.3. A result consists of an image of the monument and under it the name and place of that monument.

Browsing is shown in figure 3.4. The 3D environment will be displayed over the whole browser. The only other thing that will be displayed is the top navigation bar. Monuments are represented by blocks. The sides of a block show images of that particular monument. The user can click on a block to get more information about the monument. When a block is clicked the user is shown basic information about the monument in question. When the block is double clicked it takes the user to a page where they can see pictures, articles and other external resources and leave a comment as shown in figure 3.5.

The browse tab on the right has different browsing options displayed as buttons to select or deselect. When an option is selected the monuments shown in the cloud of blocks are ordered according to the selected option. This tab can be pinned to stay visible to the user otherwise it hides itself to let the user have a better view of the blocks. A part of the tab will still be visible so that when the user moves his moves over it the tab slides back into view.

This part of the graphical user interface satisfies requirement 4.

Figure 3.5 displays the template for the monument page. This page shows the detailed information specified in requirement 3. The template does not show the graphically similar monuments from requirement 9 and the comments from requirement 11. The graphically similar monuments should be displayed in the right column after the description and the comments should be at the bottom after the videos and articles.

Requirement 7 is represented by the add buttons on the right above the description and requirement 10 by the add sign present by each book.

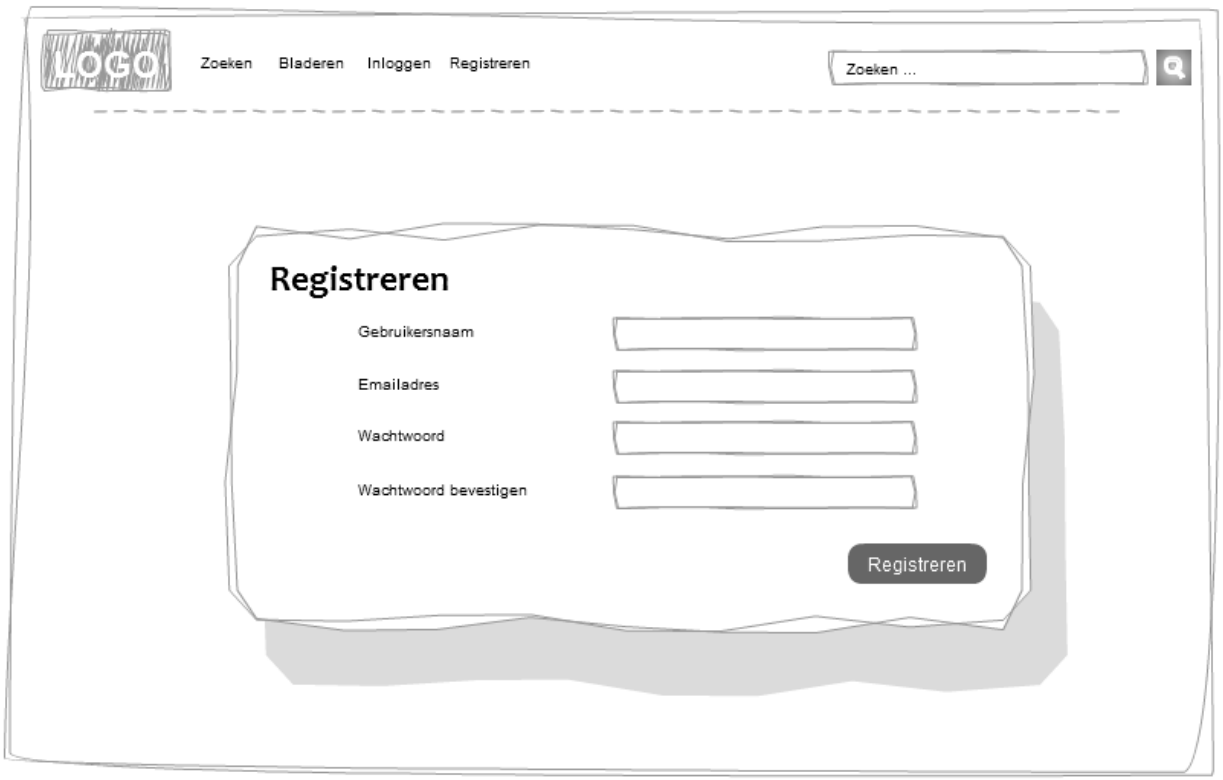

Figure 3.1: The register dialog

Figure 3.2: The login dialog

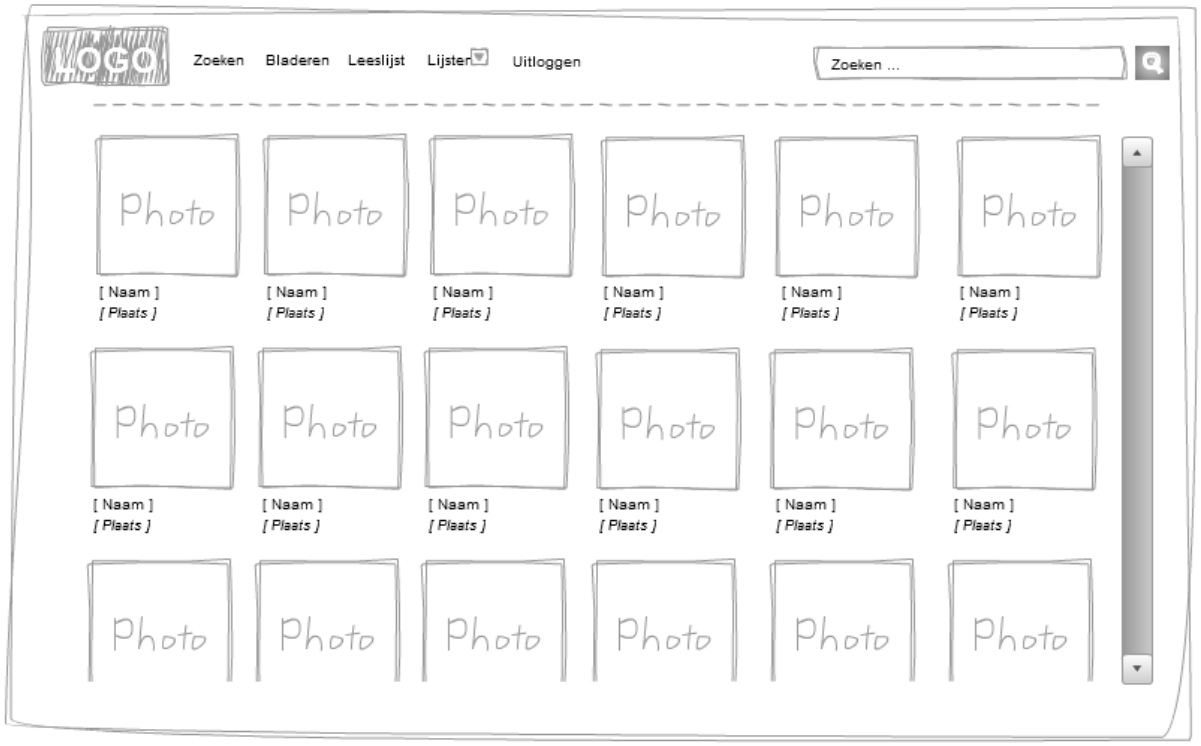

Figure 3.3: Search results template

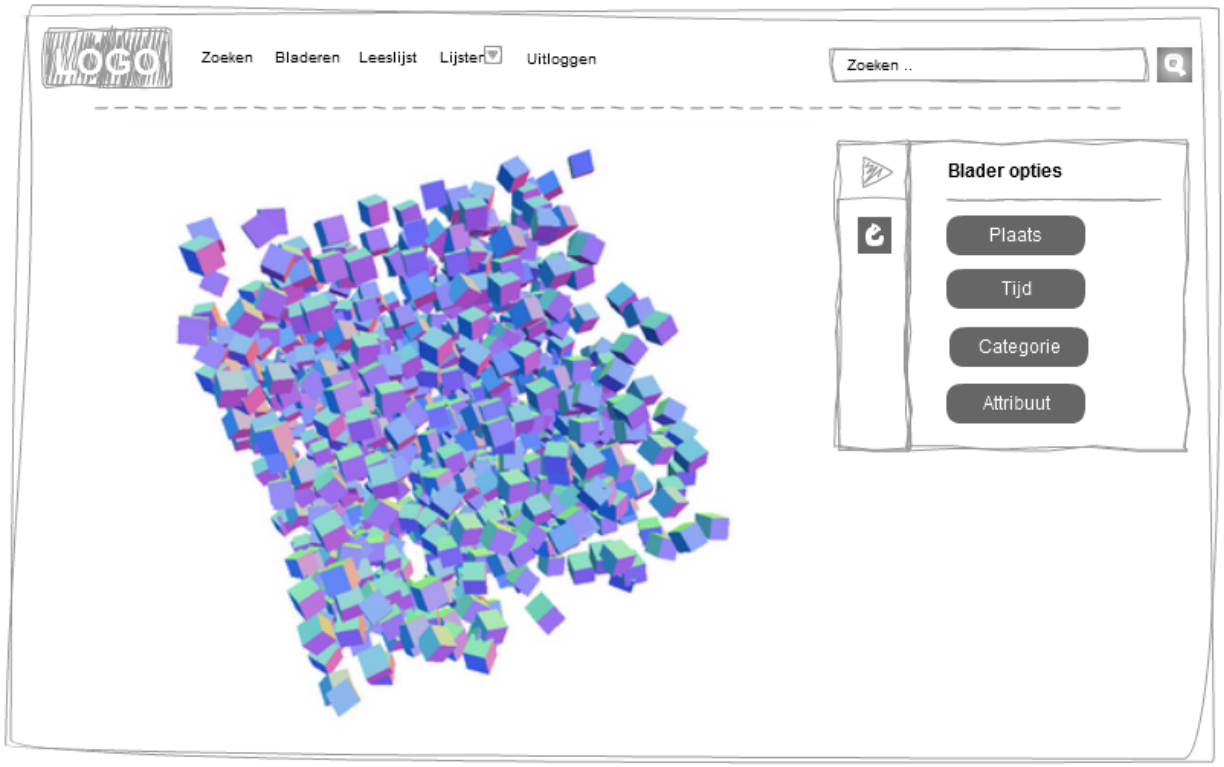

Figure 3.4: Browse template

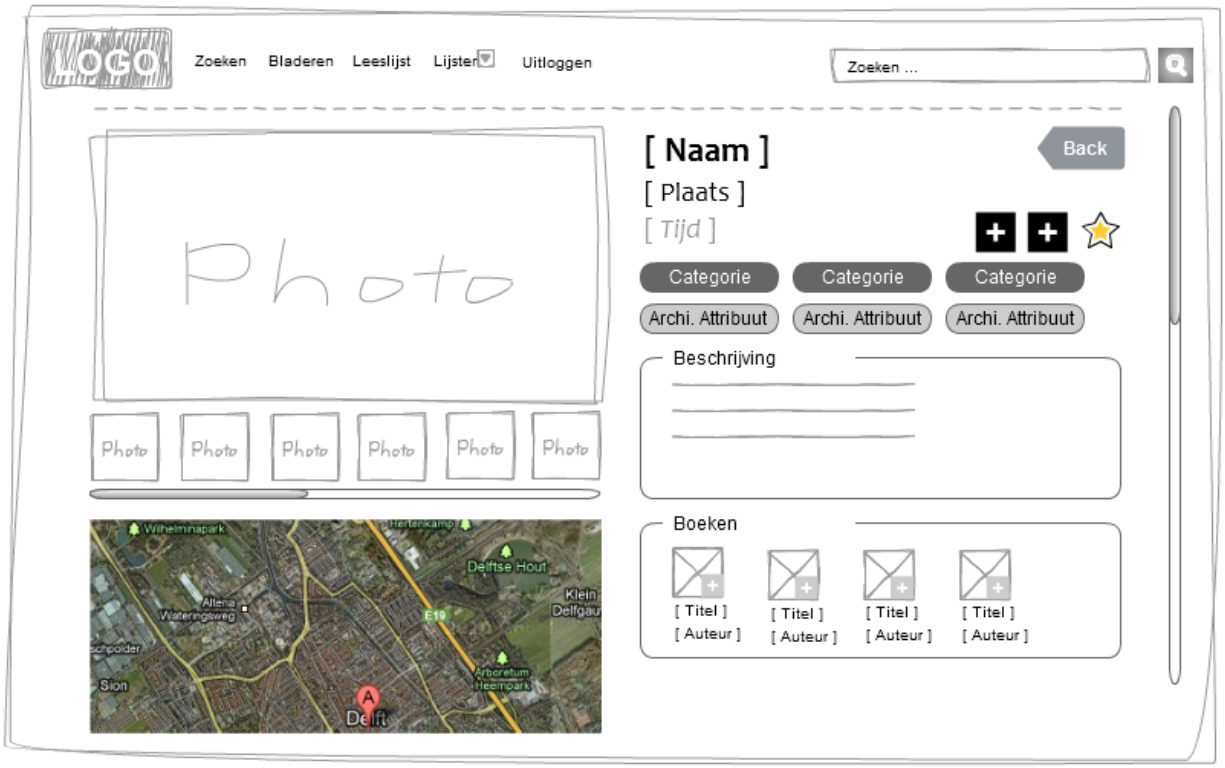

Figure 3.5.1: The first part of the monument template

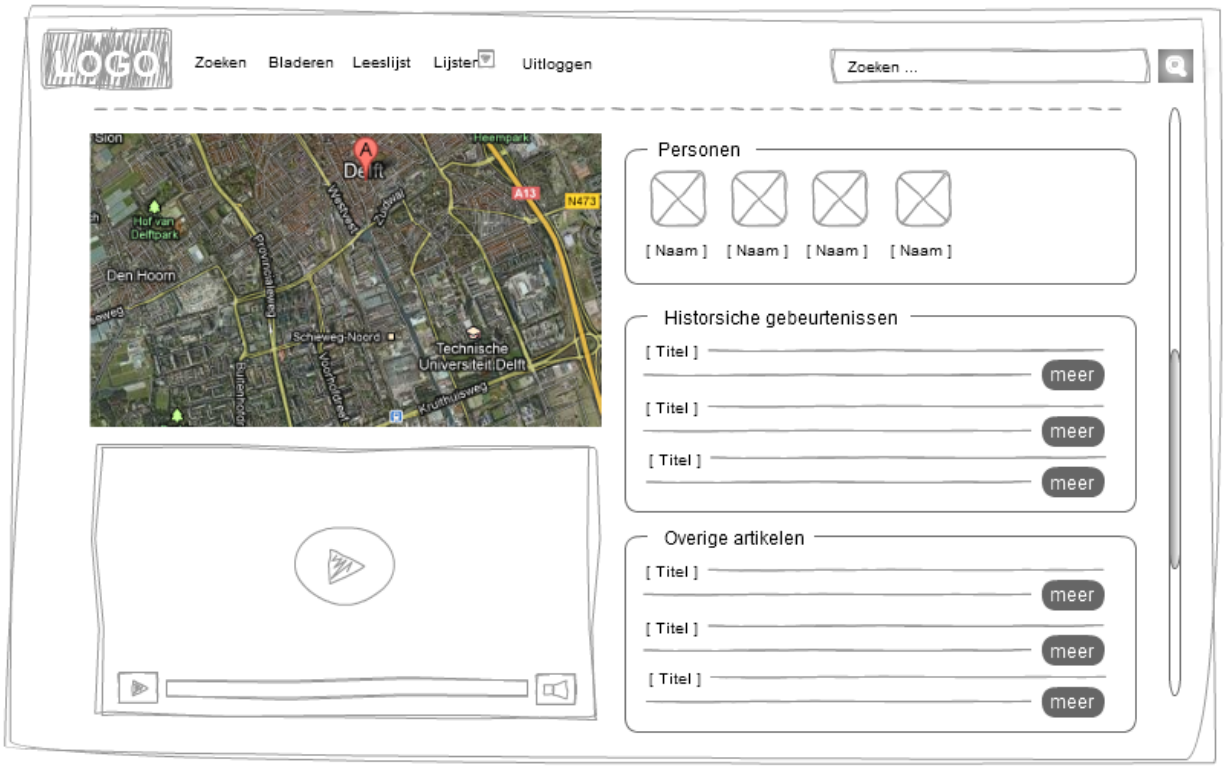

Figure 3.5.2: The second part of the monument template

4. Glossary

- **Monumentzo:** The name of this system.
- **Rijksmonumenten.info:** A multi platform system to search for monuments and more information on them.
  (source & more information: http://rijksmonumenten.info)
- **Deventeropdekaart:** A website where you can browse and search for monuments in Deventer, the Netherlands.
  (source & more information: http://deventeropdekaart.nl)
- **Vimeo:** A social video sharing website on which users can watch videos and upload their own content. (http://vimeo.com/)
- **Youtube:** A video sharing website where users can upload their own content. (http://www.youtube.com/)
- **Google Books:** A web service provided by Google Inc. where users can search the text of all books Google Inc. has scanned. (API: http://code.google.com/apis/books/)
- **Wikibooks:** Wikibooks is a wiki hosted by the Wikimedia Foundation for the creation of free content textbooks and annotated texts that anyone can edit.
  (source & more information: http://en.wikipedia.org/wiki/Wikibooks)
- **Wikipedia:** Wikipedia is a free, collaborative, multilingual Internet encyclopedia supported by the Wikimedia Foundation.
  (source & more information: http://en.wikipedia.org/wiki/Wikipedia)
- **Wikimedia Commons:** This is an online repository of free-use images, sound and other media files.
  (source & more information: http://en.wikipedia.org/wiki/Wikimedia_Commons)
- **Monuments Register:** A register containing information on dutch monuments. (http://monumentenregister.cultureelerfgoed.nl/php/main.php)
- **Three.js:** An open source JavaScript 3D library. (http://mrdoob.github.com/three.js/)
- **Lire**: A tool used for content based image retrieval. (http://www.semanticmetadata.net/lire/)