6.170 Assignment 3 Documentation

Burak Firik

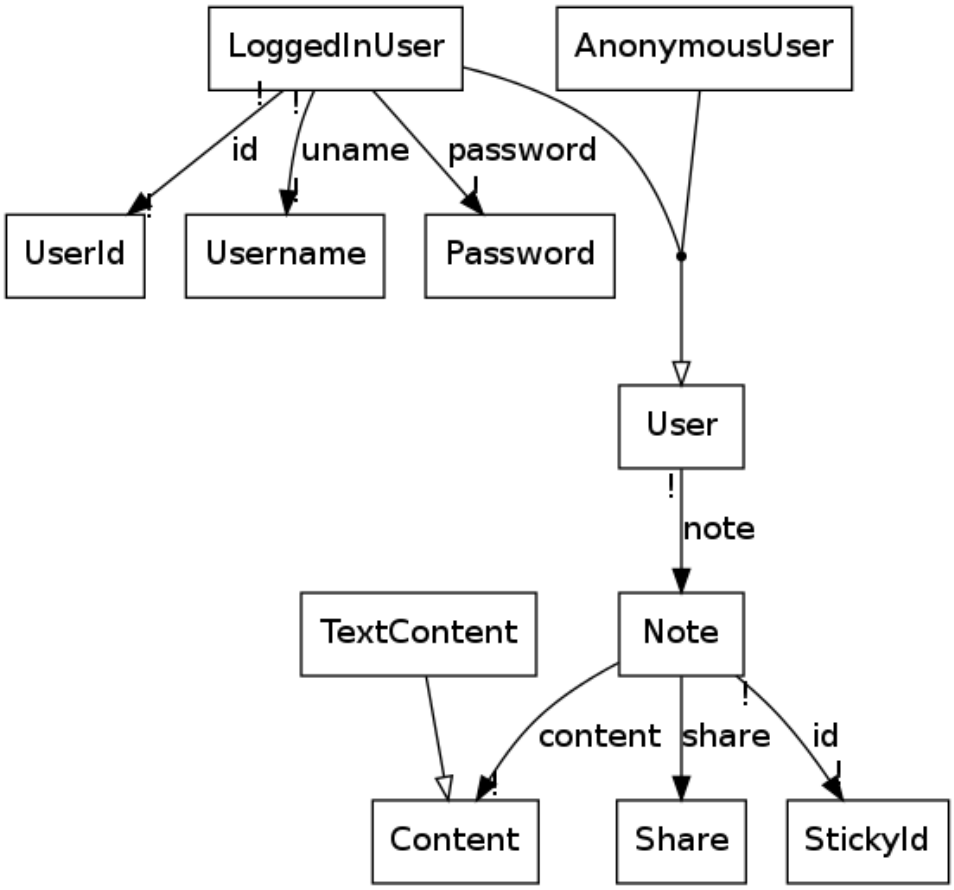
April 13, 2013

1 Models

1.1 Object Models

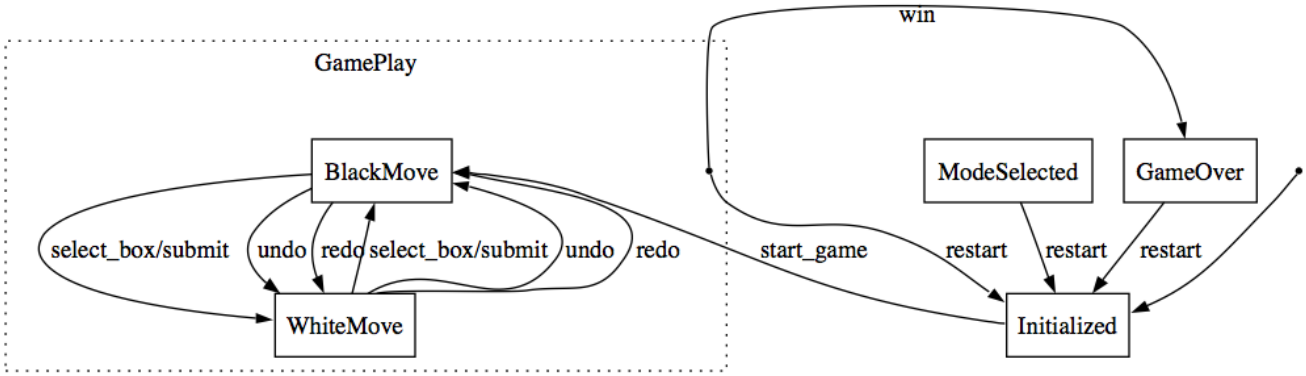
The object model for the problem domain is included in the figure below. The problem domain object model demonstrates the system that must be built.

Within the application, `User`s have a set of `Note`s. Every user has a `UserId`, `Password`, and `Username`. Every Note has a `NoteId`, `Position`, and some type of `Content`. For this application, every Note only contains an instance of `TextContent`, though an extension could allow a Note to contain other types of Content, such as an image.



1.2 State Machines

The following state machine describes how the state of the application changes as a user uses the sticky note application. The user is either logged in or out, and can only add, edit, or delete stickies from the logged-in state.



2 Design Notes

2.1 Key Challenges

- Does a user need to be logged in to create stickies? If not, should all "current stickies" be saved upon login?
  At first, I thought it would make the most sense to allow users to create stickies without being logged in. Then, upon login, all the current notes would be saved persistently for that user. This scheme was more complex than the final decision, which was to only allow users to edit stickies upon login.
- Editing notes on the fly or using a popup dialog.
  It would have been possible to pop up a dialog box whenever a user wanted to edit or add a new sticky. However, this is less usable, as it covers the window and prevents the user from interacting with the rest of the interface. Therefore, I decided to open up a textarea that allowed for editing but didn't prevent interaction with the rest of the functionalities of the application.
- Status of server – should it be stateless?
  In class we discussed that the paradigm for servers is for them to be stateless. For this reason, I refrained from storing a map of logged-in statuses for users in the server and let Flask deal with session management on a per-request basis.

2.2 Issues Arising

- Implementing login by using the flask-login module.
  Flask-login proved to be fairly hard to use correctly. Adding a registration capability is outside the scope of flask-login, and combining registration with authentication was not trivial. It proved much simpler and clearer to use session variables to represent a user's login state.

- Stickies "jump" upon save.
  For some reason, after stickies are dragged, when the new position is saved and then reloaded, the sticky often "jumps" to the left and downwards. I think this is because I am not taking into account the size of the sticky itself when saving its position. Time pressures of this project prevented debugging this, but fortunately, it does not detract much from the user's ability to move stickies at will.

2.3 Critique

This project implemented separation of concerns by using the MVC design pattern. Each module created had a clear specification and purpose, so the code itself was organized fairly well. All controller code was stored in `app/controllers`, while all model objects were stored in `app/models`. These objects followed directly from the object model in this document.

I learned a lot about building a web application from the ground-up from doing this project and it served as a good way to get acquainted with shelve, Flask, and remind myself about using GET and POST requests, and HTTP status codes.

3 Specification

3.1 Overview

This application is a web application written using ruby on rails

The application allows users to store and edit stickies containing text, and to access these stickies from any machine using their login information.

3.2 Key Features

The key features of this implementation of the 6.170 Network Stickies implementation are:

- Users can access their stickies from any browser. Login information is stored in the backend so the scheme is tolerant of browser failures.
- Smooth dragging allows users to arrange stickies anywhere in the field of view simply by dragging the mouse. The position of stickies on the page is stored across sessions.
- Features on the page are visible when applicable, and hidden when not. For instance, when a user clicks "Edit" on a sticky, the editing box appears for the user to edit. There are no popups or alerts, which generally hinder visibility.
- The size of the sticky is dependent on the amount of text entered, making the application more flexible.

- Passwords are not stored in the backend in cleartext, making the application more secure.

3.3 User Manual

The running application can be accessed at networkstickies.herokuapp.com

A new user should be registered by clicking on the "Sign up" link, and filling in the desired name and password. Thereafter, the user can login with that same name and password to access stickies.

New stickies are added via the "Add Note" button. The text of the sticky should be entered into the box that appears, and submitted by clicking the box called "Create Note".

Existing stickies are edited by clicking the "Edit" button on the sticky, updating the text that appears near the top of the window, and clicking the Update button.

Stickies are deleted by clicking the "Delete" button for that sticky.

No functionalities can be accessed without logging in first.

4 Implementation

4.1 Module Dependency Diagram

This code's modules can be seen in the context of the Model-View-Controller framework, which is roughly related to how the files were defined. The `app/controllers` folder contains the Controllers, while `app/models` contains the models, which includes the objects defined in the object model above. Concerns are separated cleanly in that no code in the model touches anything in the user interface. The model is further stored in a database where user information is kept persistently.

4.2 Code Notes

I implemented to share the notes with other users, users can share the created notes with other ones.

One interesting construct that appears in my JavaScript code has to do with using AJAX for adding stickies. At first, I was using form submission for adding stickies. However, the assignment prescribed that AJAX was used for user interaction, so I modified the JavaScript controller code to use POST requests for sticky interaction. However, to update the view upon return of the POST requests.

5 Testing

5.1 Test Plan

To test the application, the HTML was validated to ensure standards compliance. Also all the unit tests can be invoked by

- ‘rspec spec‘

In addition, manual testing was done to ensure that the application functioned as specified.

5.2 Rationale and Conclusions

This project fulfills the requirements of the assignment. The application runs as desired, is well-commented, and demonstrates separation of concerns.

Many of the design decisions implemented made the interface easier and cleaner to use, which improved the overall user experience. There is more work to be done to improve the appearance of the application, but overall it served as a good learning experience for using the Python Flask framework.