Bilkent University

Department of Computer Engineering

CS 319 Project : *Xtrify*

Group No #10

System Design Report

July 28, 2017

Ali Atlı - 21302442

Seda Gülkesen - 21302403

Atakan Özdemir - 21301134

Usama Saqib - 21500116

Course Instructor: Bora Güngören

This report is submitted to the GitHub in partial fulfilment of the requirements of the Object Oriented Software Engineering Project, course CS319.

Table of Contents

Table of Figures

1.Introduction

1.1. Purpose of the system

Xtrify is a web based note-taking application aiming to record the user's notes and suggest related articles from his/her notes. Different from similar note-taking applications, Xtrify serves as a recommendation system. Once the user save his/her notes, the notes will be extracted into keywords. The user can edit the keywords in order to change the scope of suggestion. These keywords will be used in suggestion stage. Whenever the user wants to find related articles about his/her notes, only thing the user has to do is clicking on advice button. The system requires register in order to go back to his/her history, all the notes and suggested lists are saved into his/her account. The notes are preserved and whenever the user wants to reach them, only need is to login to the system.

1.2. Design Goals

In this section, the design goals which are provided in analysis report will be discussed in order to build proper application.

1.2.1. Reliability

This application is aimed to be bug-free because meeting unexpected terminations could be annoying for the user. To be able to avoid such experiences, the tests will be done for each function of the application.

1.2.2. Modifiability

Our system will be modifiable. Whenever the developer wants to add new features on the application, an update will be easily implemented by the developer. Thanks to the object

oriented design, our project has a modular structure and the new modules can be always mounted to the project easily.

1.2.3. Adaptability

Comparing with other applications, web based apps are platform independent, they are accessible anywhere, anytime and by a computer with internet connection. Xtrify is a web based application could run on a computer from any operating system. To sum up, the only requirement to use Xtrify is having web browser and internet connection.

1.2.4. Good Documentation

Documentation is one of the important part in software engineering. Team members should give an enough importance of implementation part as well as the documentation. With a proper documentation, requirements are determined and each future of the program is described. As a result, well-documented projects face less failures during the implementation stage. It is also good for providing healthy communication among the team members. While working in a team, documentation is helpful to each member of team.

1.2.5. Well-defined interfaces

Xtrify's user interface is not complicated. It might look like a basic sticky note application, but this interface has been planned intentionally. Since people are struggling to remember things, they want to save his/her notes easily. Well designed, user friendly interface saves the user's time.

1.2.6. Ease of Use

The visitors of the page is impatient and at the first look they mostly decide whether the page meets their expectation or not. Most of the people still prefer to writing their notes by hand. If Xtrify was an ordinary note-taking app, it would be understandable. However, Xtrify is not only note-taking app but also it is regarded as a suggestion system. Xtrify is the easiest way to reach scholarly journals and articles. Since people from different age

groups can easily benefit, simple and easy to understand interface is chosen intentionally. Also, without downloading any extra apps or program, the user can access the application using any web browser and works with internet connection.

2.Software Architecture

2.1. Subsystem Decomposition

The model view controller architecture style is chosen for the system decomposition of the project. In order to keep all the classes in the system steady and organized, MVC style is planned to use. Since we plan to use Django framework and the Django framework is best suited for MVC applications the aforementioned design decisions were made. Instead the views are delegated to what is known as templates in the Django. Templates are combination of html and python code in the same document that has pretty much the same functionality for a regular View class in other MVC projects.

The classes with similar functions will be included in the same system component. The components are designed to be as independent as possible in order to provide modularity.

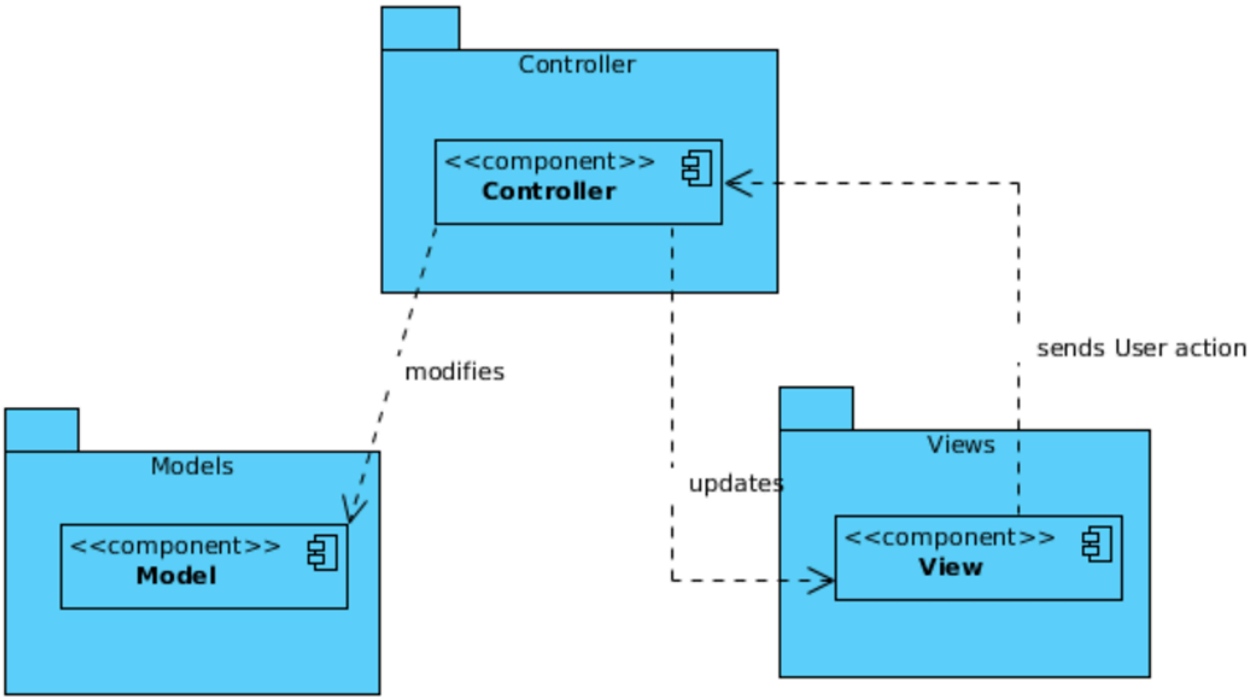

Figure 1: Subsystem Decomposition Diagram

2.1.1. User Interface Subsystem

User Interface subsystem will be composed of view components of the system. Its responsibility is to show correct page of the application according to the user's actions.

2.1.2. Model Subsystem

In this subsystem, all the application's model objects and the relations between them are represented.

2.1.3. Controller Subsystem

In the controller subsystem, the flow of the application will be controlled according to the user's action.

2.2. Hardware/Software Mapping

We are dealing with three primary hardware devices. Personal computers where the application interface will be available to the end-users via a browser. Backend server hosting the application itself, and therefore forming the logic layer of the application, in a UNIX environment. Finally a database server, also hosted in a UNIX environment, which will be accessed via the abstraction API provided by Django. The database server is the persistent data storage layer.

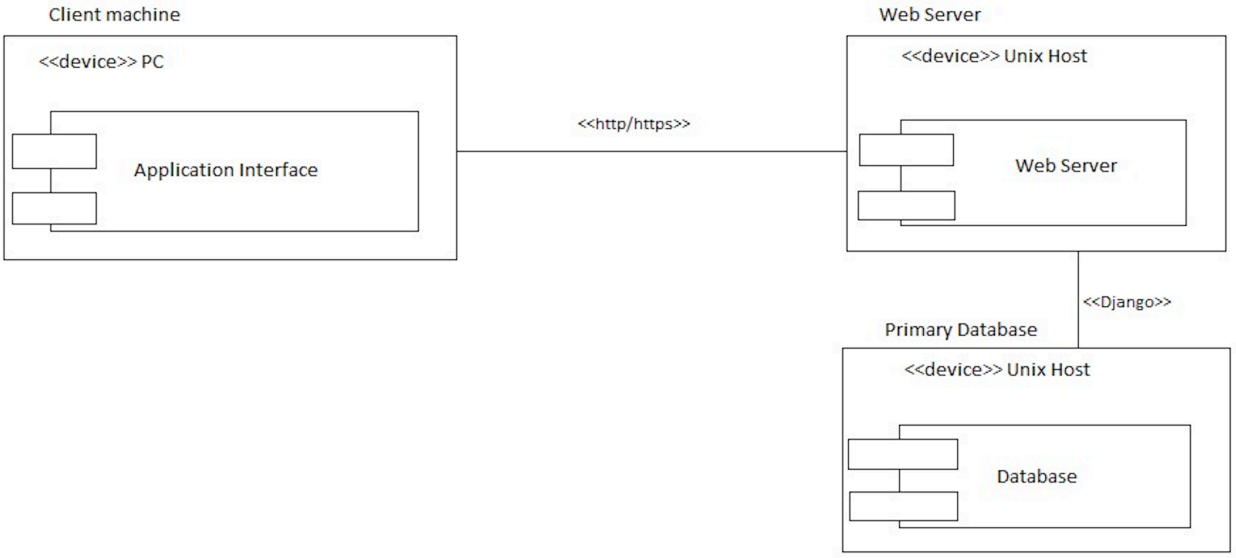

Figure 2: Deployment Diagram for Hardware/Software Mapping

2.3. Data Management

Django allows for smooth interfacing of the application with a database backend. The default database used is SQLite, which is also the database backend we have opted for. Connection management can be handled via configuration files, which can be set according to the details of the application. For example, if the application requires constant database access a persistent connection can be set up in order to avoid the overhead of establishing new connections each time, by setting the timeout age to CONN_MAX_AGE. In our case, we do not require a connection of this sort, therefore a more appropriate setting of the age would be 0 (zero), meaning that the connection is closed once the database interaction is completed. This is because interactions with the database are only required when the user explicitly triggers them.

Django automatically gives you a database-abstraction API that lets you create, retrieve, update and delete objects. These can then be used in your python code to manipulate the database entries as the logic of the application desires.

2.4. Access Control and Security

Xtrify requires internet connection and the users need to have an internet browser to access the application. On account of the fact that it is a web based application, nearly

every kind of device which has an internet browser can run the application without problem. It just requires an internet connection. When the user is able to run the application with proper internet browser and internet connection, he/she needs to register with a new account or login with an existing account to system with his/her username and a password. The system checks whether the username and password are valid and match then let user in. The user can also change his/her username and password but firstly they need to rewrite their current username and password.

2.5. Boundary Conditions

Initialization

- When the user opens executable file and after s/he access to the internet browser, the application is ready to be used. Register menu screen is shown when the application starts.
- Python and Django framework is used to create pages of the application.
- On the register screen there are three choices: Login, Register and Contact us.
- When the user selects register, he/she needs to write a new username and a password to get a new account, then she/he logins to the system. When the users have any questions or they have in trouble they can contact with the developers via Contact us button.
- When user selects sign in, he/she uses a previously created username and password to login the system.
- On the main page of the application, there are three choices: Home, My Profile and Logout.
- If user selects Home button from the main page, he /she can access his/her previously added notes. In the home page, there are two buttons: save and extract. Save button is for saving the newly added note into the system. Extract button loads keywords page which contains extracted keywords from the corresponding note.
- Keywords page includes advice button which loads suggestions page includes research options of the keywords from the internet database. It also has edit keywords button which loads edit keywords page.

- If user selects my profile from the main page, he/she can access settings which has 2 buttons called change password and change username to change and secure his//her username and password.
- Finally, if the user chooses to logout, he/she will redirect to the login page again.

Termination

- On the main page, the user can choose to logout to exit from the system.
- User can exit with use of “X” button of the browser whenever he/she wants, ongoing process will be lost in this condition.

Failure

- -If user manually closes the browser during the extraction or note taking process, the application will not be able to save ongoing process.
- -The possible problems about power of the system cause the data loss.

3.Subsystem Services

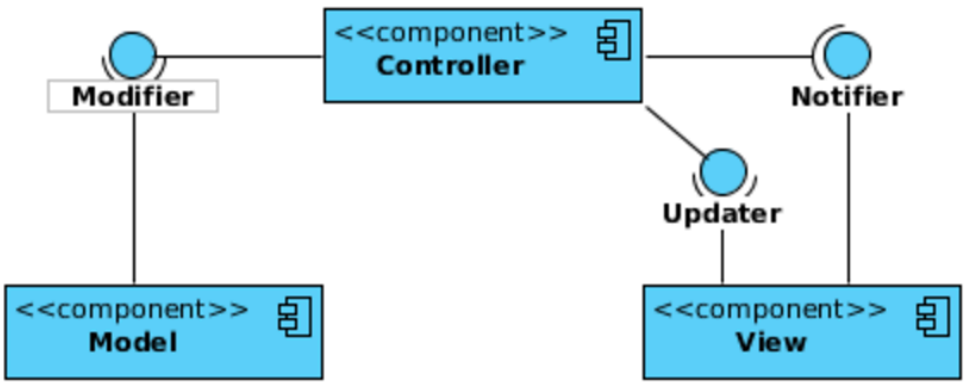

Figure 3: Subsystem Services Diagram

The system is decomposed into 3 parts as model, view, controller. The services shown in the figure do not have concrete one-to-one class name correspondence. However, the services of the systems can be best described by those names.

3.1. Services of the View

The View component is basically responsible for notifying the controller regarding the user input. After the user input is received as a request it is sent to the Controller component through the browser. This component is realised with what is known as templates in Django terminology. They are basically, html pages with python codes embedded in them. Through this html codes, the View component generates requests and sends them to the Controller component.

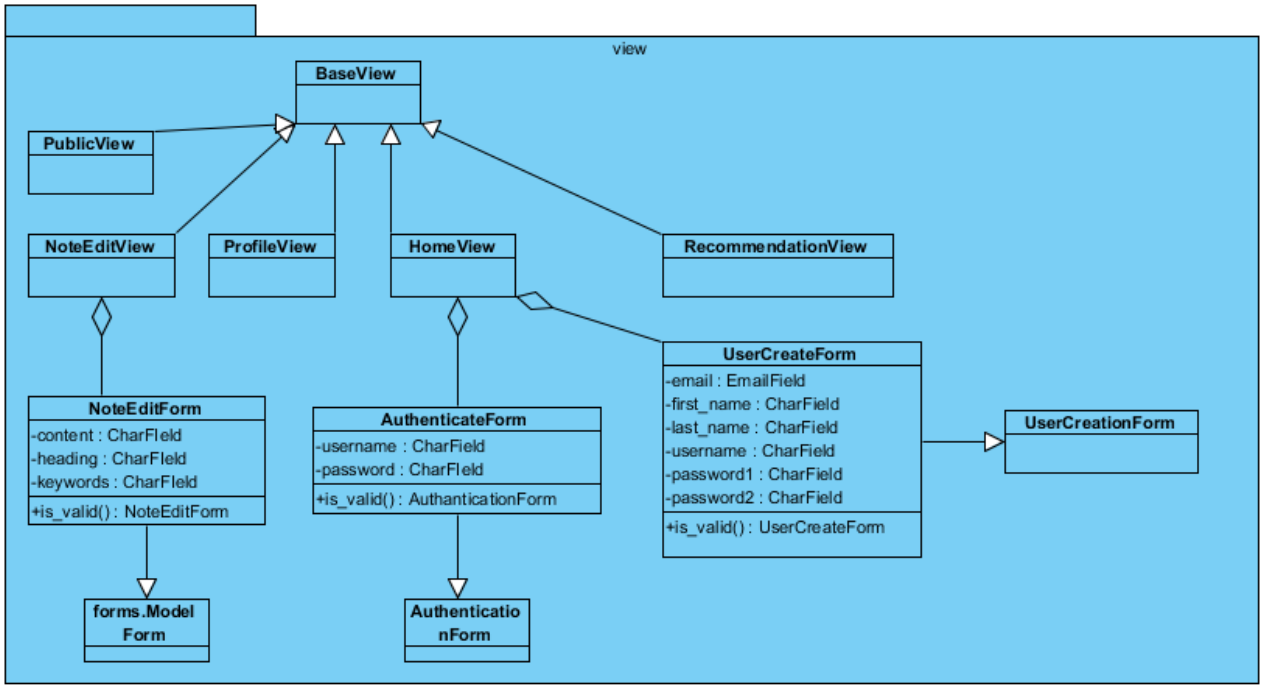

Figure 4 : View Package

3.2. Services of the Model

The Models in this form does not offer any service. They are mere placeholders for our application.

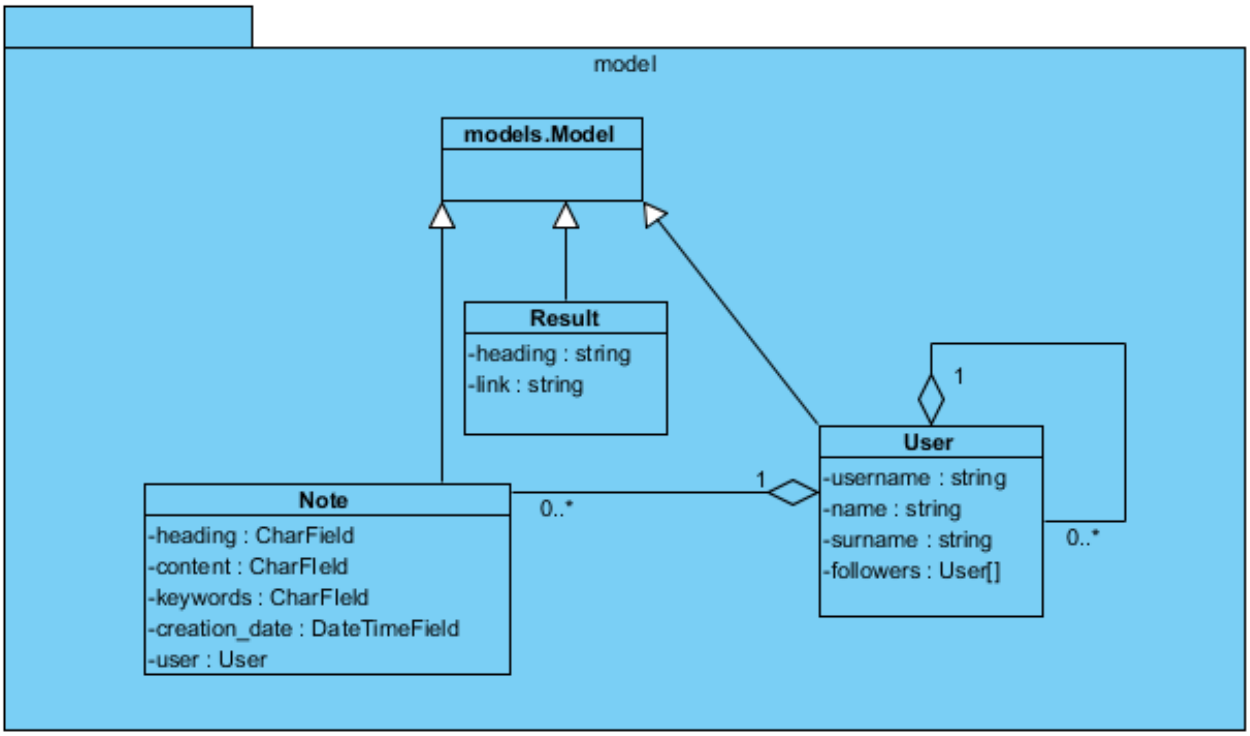

Figure 5: Model Package

3.3. Services of the Controller

The controller is mainly involved in the in-site navigation. When the user input is received, it is stored in a variable called ‘request’, then they are passed to the controller classes as a parameter. This request also stores information about the user has submitted. Depending on the type of action, whether it is a POST request or a GET or the kind of button pressed, the site navigates the user.

Controller passes information across different sites through inserting data to the url. A code piece in the Controller then checks if the url is fit for pre-determined types.

Controller updates the Model classes and their low level representation in the data request upon request.Controller is also responsible for talking to the keyword extraction and recommendation related classes. It receives data form the database about the models, then sends them to the keyword extraction and recommendation part, based on the input of the user, then updates the model.

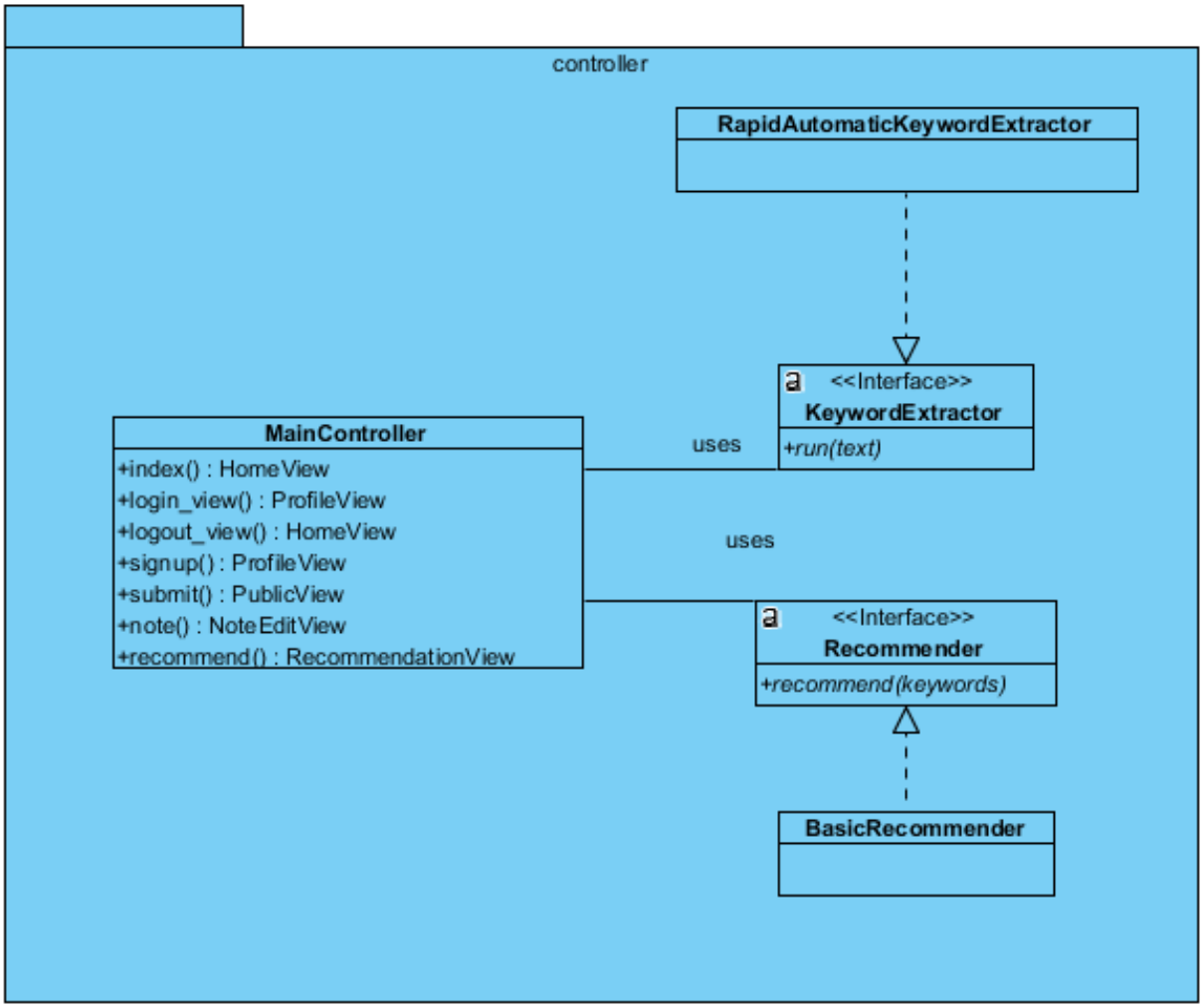

Figure 6: Controller Package

4.Low-Level Design

4.1. Object Design Trade-Offs

4.1.1. Rapid Development vs Functionality

Due to the fact that we have limited time to develop and implement the application, we need to sacrifice some functionalities of the program to develop faster. At the beginning of the project deciding which functionalities are inevitable and will be implemented primarily is necessary. Firstly, on account of the fact that we use static pages rather than dynamic ones, the user cannot add his/her profile picture into his/her account. Also, if the

user forgets his/her password, s/he cannot use his/her e-mail account to reset password. “Forget your password?” function cannot be added because of time issues.

4.1.2. Extendibility vs Functionality

The application at the moment implements only the core functionality. However, in order for it to fulfil all the roles required of a modern note taking application, more features need to be added. Given the time constraints we were not able to add these, therefore, the team’s focus from the beginning was to make the app extensible. This was so that more features could be readily integrated into the current model without changes to the core functionality. This would allow for more rapid development in later periods.

4.1.3. Security vs Performance

Security considerations are ignored in a number of places due to the time constraints and performance requirements. We have decided not to tunnel HTTP requests through SSL as that would require time consuming research. Similar other good practices from a security perspective have been foregone. We have chosen not to encrypt the passwords of the users when storing them in the database, nor perform any salting operations to secure against brute-force attacks, so as to avoid incurring the performance overhead such operations would result in. On top of that the proper implementation of such operations is time consuming and subject to rigorous testing, which would cause the schedule of our project to be extended. Care will be taken while coding to reduce common malpractices which result in command injection vulnerabilities are avoided. However, ensuring that these practices have been properly followed, require time consuming security audits, and therefore such vulnerabilities will not be protected against.

4.2. Final Object Design

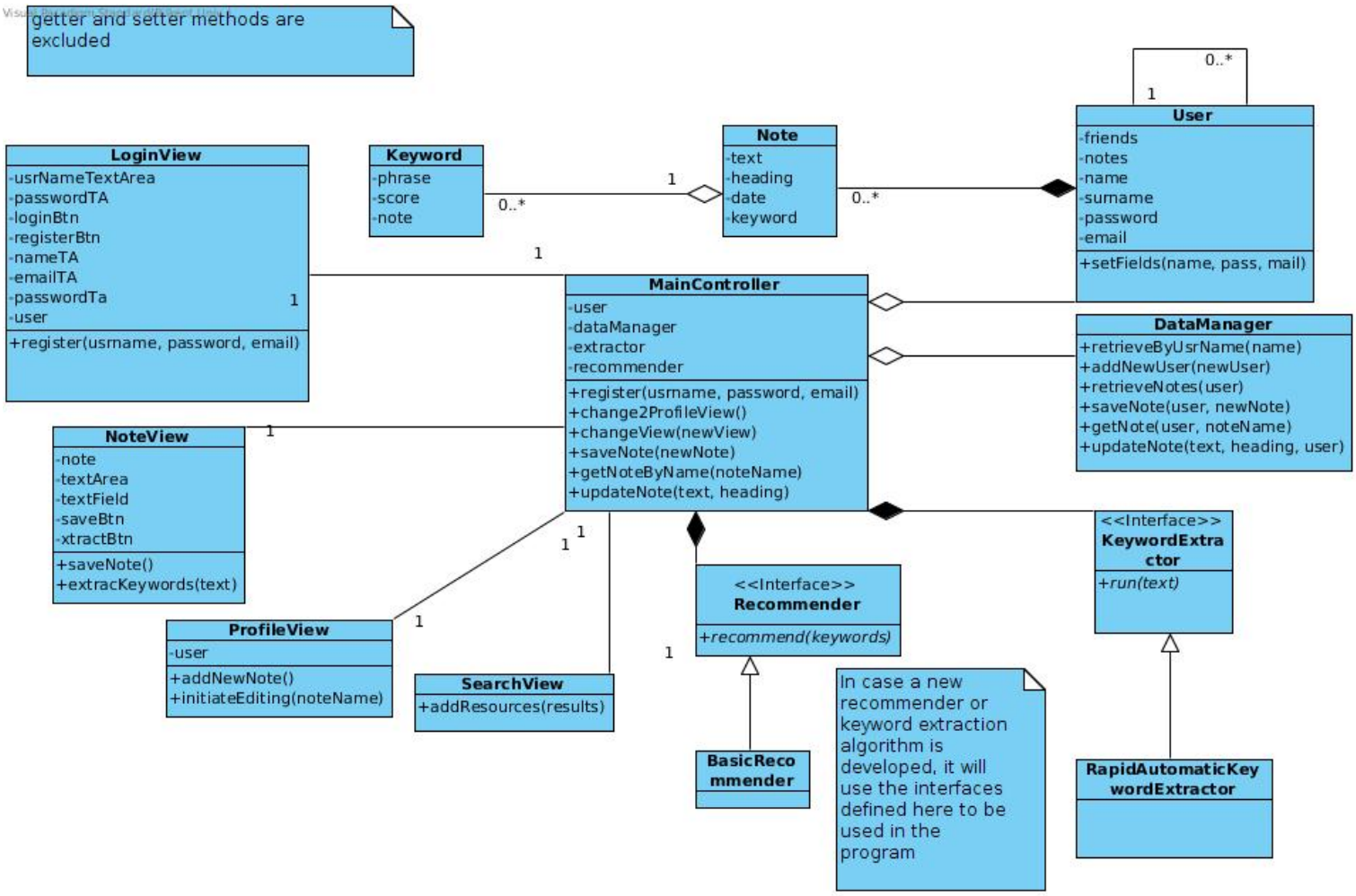

Figure 7: Final Class Diagram

4.2.1. Strategy Pattern

The interfaces KeywordExtractor and Recommender are strategy interfaces whereas the classes that implement are concrete strategies. This kind of design pattern is used as recommendation and keywords extraction often go hand in hand. Meaning that one class may, in fact, implement both of the functionalities. Since we wanted to avoid getting involved with multiple inheritance issues, we decided it was best if we used interfaces instead.

This also, we believe, allows us to test different kinds of strategies for aforementioned jobs easily without changing other parts of the code much.

4.3. Class Interfaces

In this part, detailed class diagram will be discussed according to the final object diagram. For each component the detailed class relations will be explained.

4.3.1. Model Classes

-Note Class

Attributes:

heading: This attribute holds header of the note.

content: This attribute holds contents of the note in a char field.

keywords: This attribute holds extracted keywords from the content of the note.

creation_date: This attribute holds precise creation date of the note.

user: This attribute holds a reference to a specific user's notes.

-User Class

User class represents a user with account's personal information, it simply holds data like username, password, etc. of an account.

Attributes:

username: This attribute holds username of the user.

name: This attribute holds name of the user.

surname: This attribute holds surname of the user.

followers: This attribute holds followers of the user in an array.

-Result Class

Attributes:

heading: This attribute holds title.

link: This attribute holds recommended links from Recommender..

4.3.2. Controller Classes

-MainController Class

MainController class deals with navigating the user and sending and receiving information from the strategy interfaces.

Methods:

def index(request): This method checks if the user is authenticated, it redirects to the ProfileView page. If the user is not authenticated, it redirects to HomeView page where sign up and login functions occurs.

def login_view(request): This method checks username and password then let user login into the system, then it redirects to ProfileView page.

def logout_view(request): This method is responsible for logout, closing the session for the user under the hood. Then redirects user to HomeView page.

def signup(request): This method is responsible for signing up the user. It gets the information then checks if the form filled by the user is valid. If so redirects to the ProfileView.

def submit(request): This method helps user to save notes to the account, then redirects to PublicView page.

def note(request): There are three buttons in NoteEditView page. When this page is shown to the user the requests are sent to this method. If the clicks from the ProfileView on the Note titles, it gets the note id from the url, finds the corresponding note from the database by its id, then fills the text areas with those values.

- If the user presses on Xtract button, the KeywordExtractor instance is invoked and the keywords are shown to the user in a text area.
- If the user presses on Xtrify button, the button name is taken from the request parameter, the keywords are given to the Recommender instance and the user is redirected to the RecommendationView page.

def recommend(request): This method simply terminates the RecommendationView page and redirects the user to the ProfileView page.

4.3.3. View Classes

The view classes are actually html pages in which some python code lives. They are responsible for creating requests. They do not have class attributes as the components are written as html code.

BaseView Class: This class is the base view class that contains the skeleton of the all of the pages. For instance, the header is not repeated across different view classes. It allows certain blocks to be inserted in it via extension.

NoteEditView Class: This class is a page for editing the notes that are created previously.

ProfileView Class: This class is a page for the profile screen, login and sign up functions can be accessed from this class.

HomeView Class: This class is a page for the main home screen.

RecommendationView Class:

-NoteEditForm Class

This class is extended from forms.ModelForm class in Django project. Therefore, it requires the binding with a Model class. The corresponding model class is Note class in this case. The input is recorded and returned as a Note instance for ease of use later on.

Attributes:

content: This attribute holds content of a note from user's account.

heading: This attribute holds header of a note.

keywords: This attribute holds extracted keywords from a specific note.

Methods:

def is_valid(self): This method checks if keyword edition is validly completed or not.

-AuthenticateForm Class

This is extended from pre-written AuthenticationForm in Django framework and makes the validation of the user's credentials during sign in process much more easy, with only overriding the is_valid() method.

Attributes:

username: This attribute holds username of the user.

password: This attribute holds password of the user.
Methods:

def is_valid(self): This method controls if username and password match with each other or not.

-UserCreateForm Class

This class is extended from UserCreationForm as above cases. Since the user creation for web sites are wide, it is automated by extending this class then only overriding the is_valid() method.

Attributes:

email: This attribute holds e-mail account of the new user.

first_name: This attribute holds first name of the new user.

last_name: This attribute holds last name of the new user.

username: This attribute holds username of the new user.

password1: This attribute holds password of the new user.

password2: This attribute holds confirmation password which needs to be same with password1.

Methods:

def is_valid(self): This methods decides if user creation process is valid or not.