System Design - Growth

Summer 2021 CSCC01 Project

Jaiteg Sahota | Ronish Bhatt | Yihang Cheng | Zhengyi Zhang | Shashwat Piyush Dosh | Naman Bhandari | Pritish Panda

Table of Contents

Front-End CRC Cards

Forum:
Parent class: none
Subclasses: none
Responsibilities:

- Provides a list of all posts:
    - With the author name
    - Date posted
    - Last modified
    - Post content
- makes posts through a GUI interface when logged in.
    - If user is not logged in, it redirects to a login page
- Allows editing, and deleting posts that are made by the current logged in user
- remove it.

Collaborations (models):

- Users
- Post
- Comment
- Reply

Post:
Parent class: none
Subclasses: none
Responsibilities:

- Provides a single post:
    - With the author name
    - Post title
    - Date posted
    - Last modified
    - Post content
- makes comments through a GUI interface when logged in.
    - If user is not logged in, it redirects to a login page
- Allows editing and deleting comments that are made by the current logged in user

Collaborations (models):

- Users
- Comment
- Reply

Comment:
Parent class: none
Subclasses: none
Responsibilities:

- Provides a list of all comments:
    - With the author name
    - Date posted
    - Post id
    - Comment content
- Stores replies to comments as Reply

Collaborations (models):

- Users
- Post
- Reply

Reply: Parent class: none Subclasses: none Responsibilities: - Provides replies to comment - All logged in users able to post reply to any comment Collaborations (models): - Users - Comment - Post	**Company Page** Parent class: none Subclasses: none Responsibilities: - Provide the user with information of their company - For users who have not yet registered a company, have a button to create and enroll in one. - For users who have a company, have a button to edit or remove it. Collaborations: - Users	
Messaging other users: Parent class: none Subclasses: none Responsibilities: - Provide text messaging capabilities to users who have access to the application - Users who have logged in can join a particular lobby with a security link/code that will allow them to communicate with other users in the same room - Users can join multiple rooms to communicate with	**Navigation-bar:** Parent class: none Subclasses: none Responsibilities: - Allows users to navigate between the homepage , the about us page, login page, and login page Collaborations : - Home page - Login page - About page - Blog Page - African Impact Challenge page	

different users in the web application Collaborations: - Login - Users		

Back-End CRC Cards

User Parent class: none Subclasses: none Responsibilities: - Save all user information including email, username, password, role and so on - Have abstract user information where built-in password verification can be used - Have the ability to chat, register and login - Able to authenticate if user is logged in or not Collaborations: - Sign up - Sign up for Partner Profiles - Login	**Login:** Parent class: none Subclasses: none Responsibilities: - Authenticates users who have registered themselves to access the application - Display an error message if authentication fails - The user can browse and access the application so long as they are not logged out or the session has timed-out. - Allows the user to reset their password - Provides a link to users who sign-up if they haven't registered themselves before. Collaborations: - Users	**Sign up:** Parent class: none Subclasses: Sign up for Partner Profile Responsibilities: - Require user to fill the mandatory fields for registering: - Full Name - Email Address - Password - User need to follow some rules while making a new password - Not similar with personal information - Passwords should be at least 8 characters. - Password can’t be entirely numeric - Display a Welcome message after signing up. Collaborations: - Users

Sign up for Partner Profile: Parent class: Sign up Subclasses: none Responsibilities: - Prompting user to enter the role as a user - Student - Instructor - Partner - The database will store information about the role of the user, so as to extract it in future. Collaborations: - Users		

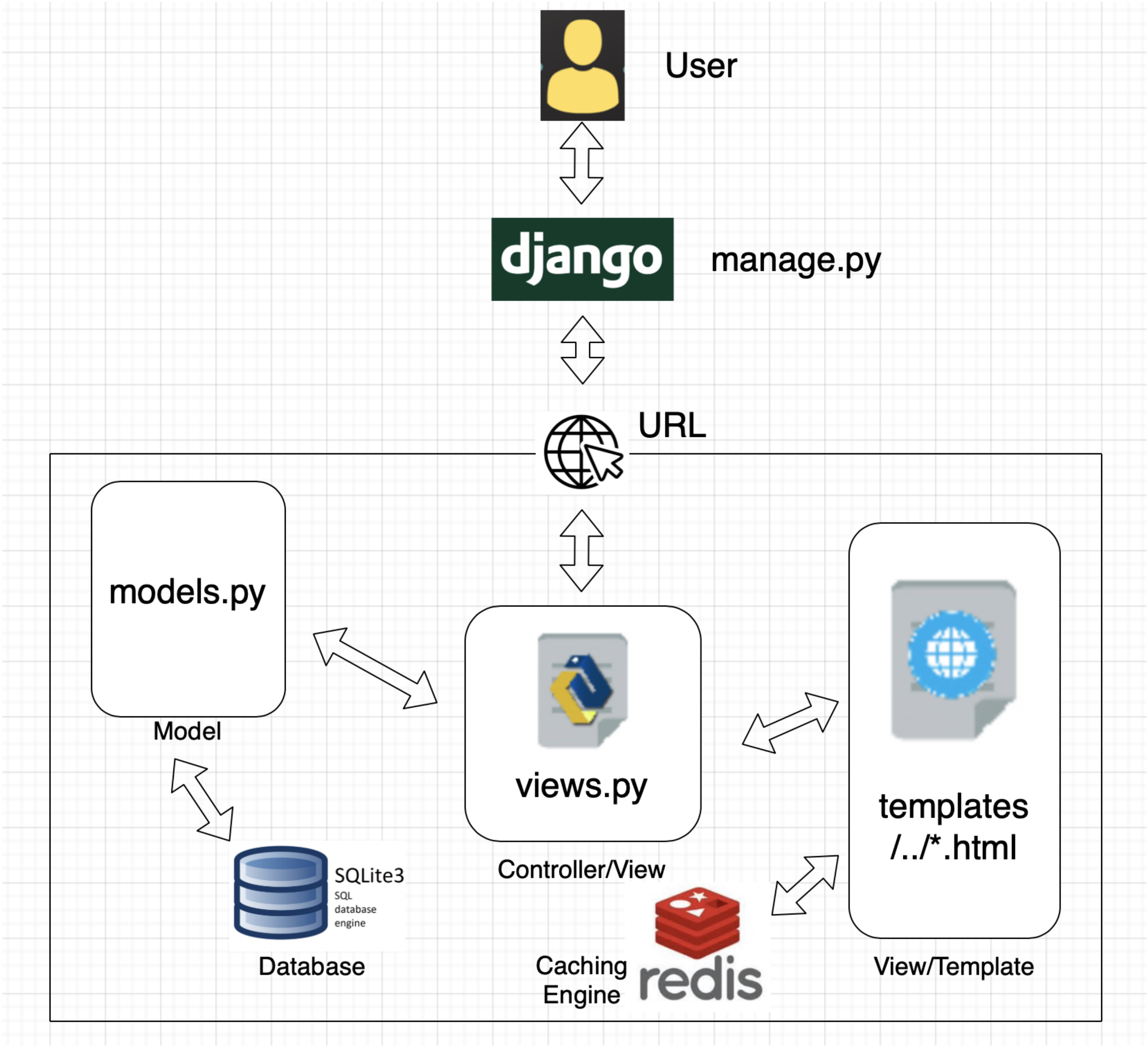
Description of System Interaction with Environment

The user must have a working browser with internet access on any technological device. Through the browser, the user is able to access the site and can login or sign up using a valid email and password. Upon logging in or signing up successfully, the user can browse and access the application so long as they are not logged out or the session has timed-out. Specific users such as instructors are able to upload course content in the e-learning section as well as view and moderate the posts in the community section. Every user will have a profile where their profile name, and other basic info can be added or changed at will. If the user is uploading any data from their device (i.e for profile pictures or uploading documents), access to their system file storage will be required.

To run the web application locally, the steps are as follow: clone the Git repo to your device and then navigate to the project-growth folder and run “sh firstTimeRun.sh” and then cd to Growth and run “python manage.py runserver” to start the web app. This django deployment takes care of both the front and back end. The sh script takes care of the django installation as well as the virtualenv installation and activation. The default port by which the app can be accessed is 127.0.0.1:8000.

Description of Architecture of the System

The software architecture model being used is the MVC model because of the dominant use of Django for the construction and coding of the system. The software architecture diagram below details the interaction of varying components in the system.



System Decomposition

Through the Django framework, the user can begin interacting with the web app using manage.py and the run instructions described above. After the local host server is up and running, all of the work is handled through Django's URL with the help of views.py as the Controller, various HTML templates as the View, and models.py as the Model, with the help of SQLite3 for keeping track of the Database. Redis is also used as a Caching Engine for live chat features.

There are several error prevention schemas in place, and a few are described in further detail: Errors during login and registration where common passwords are prevented from being used, as well as checking for valid email and password during each process. Username is also cross checked with the database to avoid creating duplicate usernames. Error prevention in forums is also done by preventing edits of other people's comments or posts.