## **Table of Contents**

1. Project Title and Authors

2. Preface

3. Introduction

4. Architectural Design

5. Detailed Design

Package Diagram

Class Diagram

Class Interaction Diagram

Sequence Diagrams

**1. Project Title and Authors**

**Project Title:** CSUEB Survey App

**Team:**

**Team Members:** Robert Aguirre - kr8310

David Emono - demono@horizon.csueastbay.edu

Rajaarun Arulraj - sg8986

Lam Thai - jw2564

**Website:** <https://bob-02.github.io/CS401_Survey-App/>

**Repository:** [**https://github.com/LamThai13/CS401\_Survey-App**](https://github.com/LamThai13/CS401_Survey-App)

**2. Preface**

This document lays out the design specifics for the CSUEB Survey App. It includes both a high-level architectural overview and a detailed design structure, following standard UML designations. We've selected Django for our back-end operations and SQLite as our database due to its compatibility and simplicity. The implementation of the system in the next phase will follow this design document.

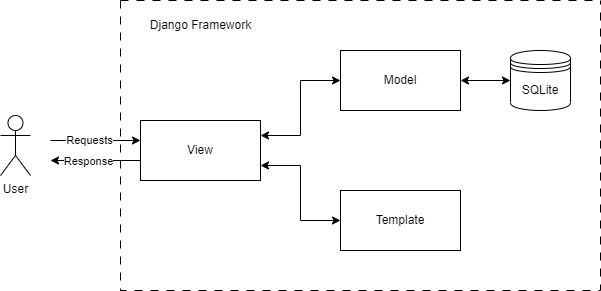
**3. Introduction**

The CSUEB Survey App is a digital platform that enables CSUEB students to express their opinions and view others' ratings about various entities, such as food, events, professors, courses, and more, anonymously. The design is developed to ensure an interactive and user-friendly application. Using Django, along with its Model-View-Template (MVT) structure and SQLite for the database, will streamline the development process. Django’s ability to handle much of the configuration and set-up allows us to focus on designing effective models, views, and templates to deliver a user-friendly, efficient, and interactive Survey App for the CSUEB students. Thereby allowing our team for rapid prototyping, development and deployment.

**4. Architectural Design**

The architectural design will follow the MVT (Model-View-Template) pattern typical of Django projects.

* **Model:** This layer will interact with the SQLite database, performing CRUD operations on entities such as Users, Polls, Ratings, and Votes.
* **View:** The view will manage the user requests and application responses, handling the business logic.
* **Template:** For the front-end user interface, Django's template system will be used to present data to the users. Django templates will also be used to create HTML pages that can interact with the back-end Django views.



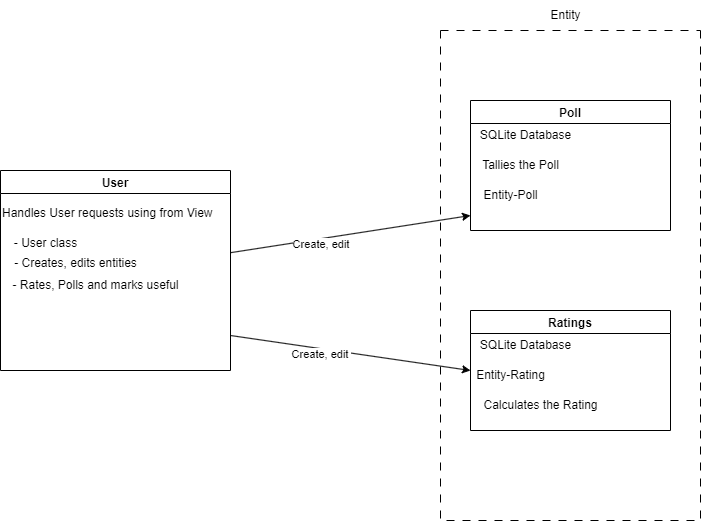
External frameworks/libraries to be used include Django for the back-end, SQLite for the database, and Django templates for the front-end.

**5. Detailed Design**

**5.1 Package Diagram**

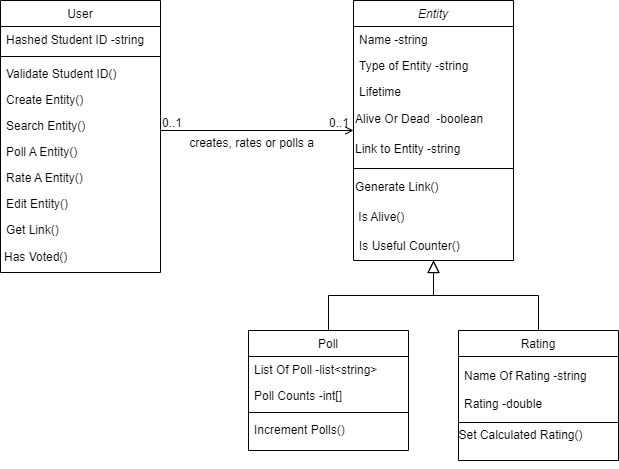
The Django application will be divided into three main packages:

* Users: It includes the User class, containing attributes and methods for user registration, login, and management.
* Polls: This package manages the creation, modification, and deletion of poll entities.
* Ratings: This package is dedicated to handling rating entities.



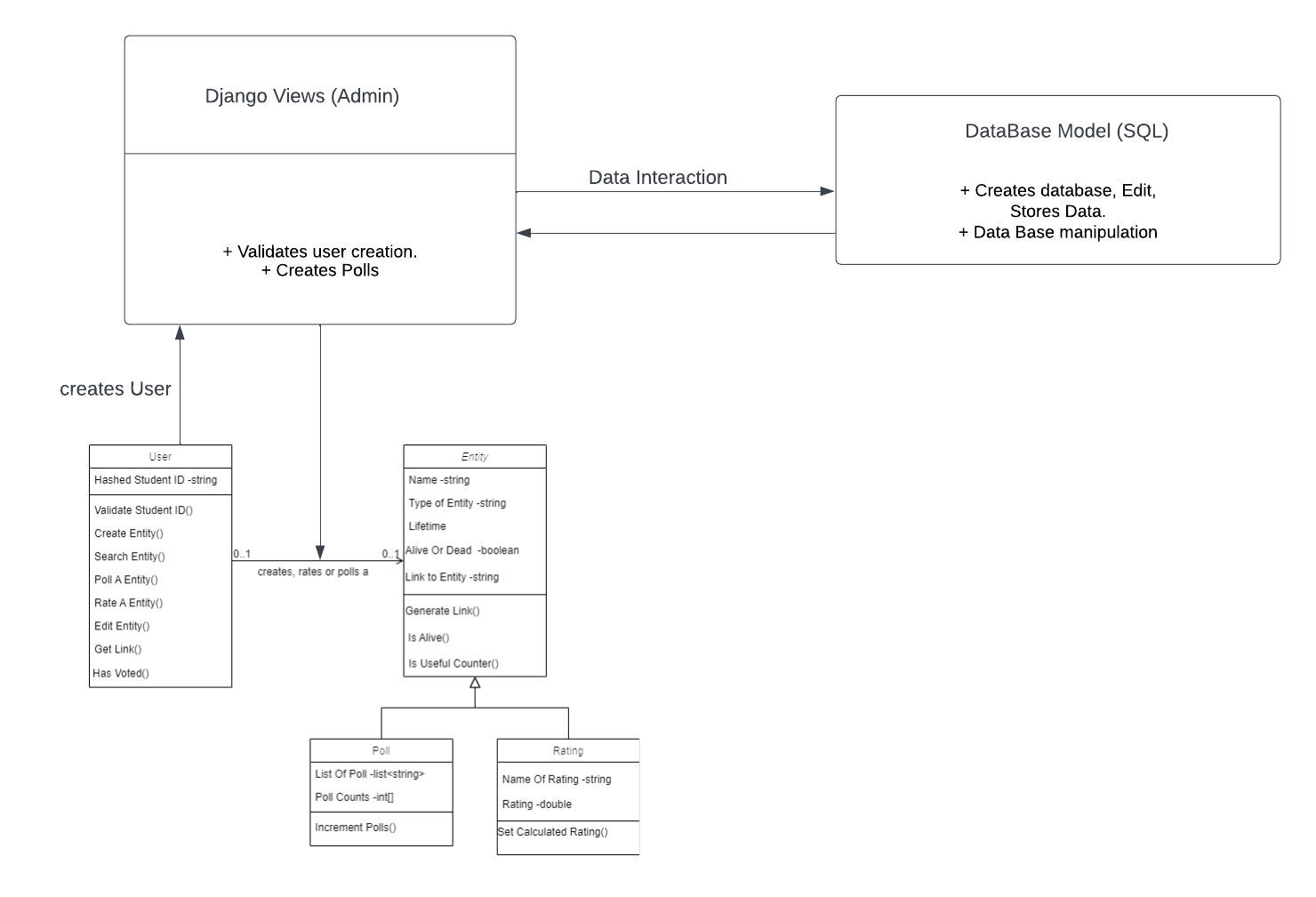
**5.2 Class Diagram**

Key classes within these packages will include User, Poll and Rating each having respective attributes and methods.

****

**5.3 Class Interaction Diagram**

Interaction among the classes will be managed via Django views. For example, the view handling user registration will interact with the User class to save new user data (key-pair) to the database.

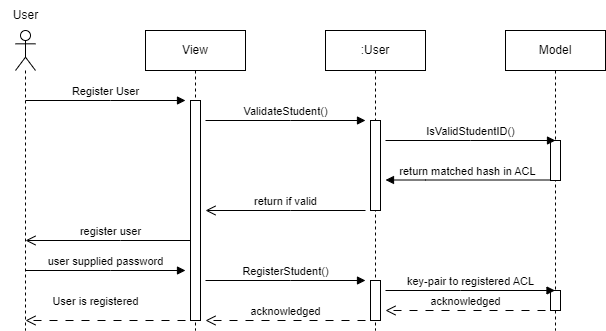
****

**5.4 Sequence Diagrams**

Key interactions among objects in the classes will be represented through sequence diagrams for functionalities like User Registration, User Login, Poll Creation, Rating Submission, and others.

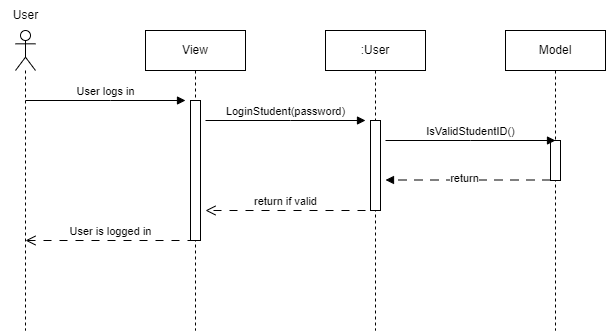
**User Registration**

User (valid student) uses their ID to see if they are valid for registration. If a valid ID is matched they will be acknowledged to supply a password. The password and student ID will be hashed together and added to a ‘registered’ ACL in the database.



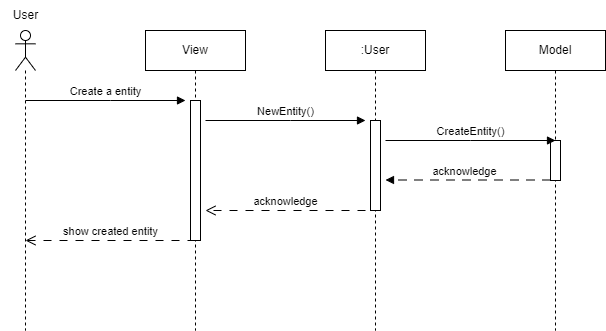
**User Login**

User logs in the view send the supplied password and checks if its valid in the database's ACL and then shows in a logged in screen.



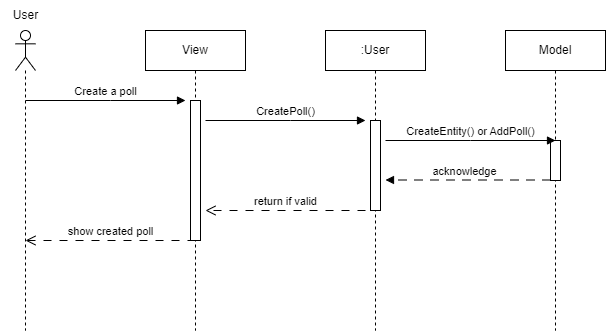
**Create a Entity**

User creates an entity that does not exist in the database yet.

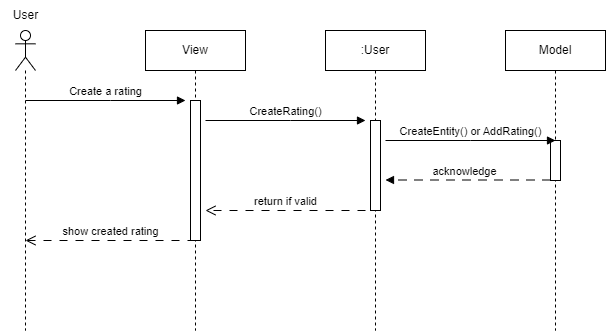


**Create a poll**

User creates a poll. If the entity does not exist it will create a new entity and add a poll. If an entity already exists it just adds a poll to the entity and updates the database.

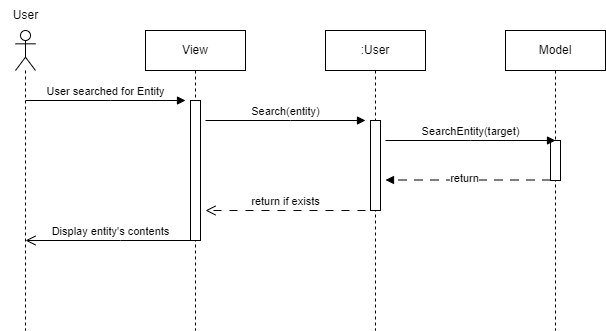


**Create a Rating**



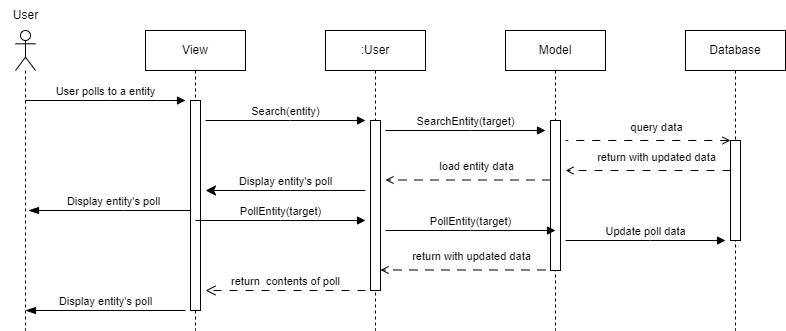
**Search an Entity**

A search is run by the user to display the contents of an entity that exists.

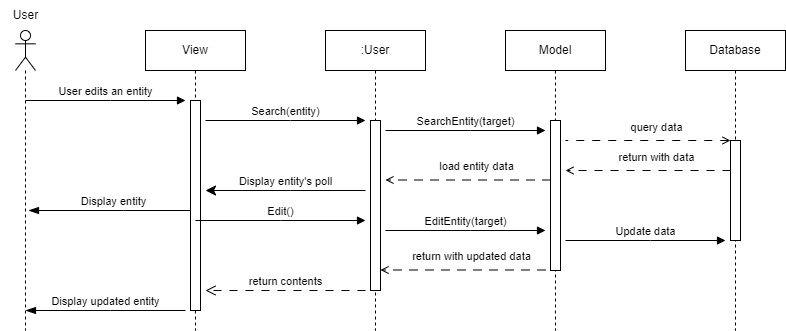
****

**Poll an Entity**

A user votes on a poll that exists. The model will query the intended poll to vote on in the database. Display the entity's contents and update the backend accordingly.

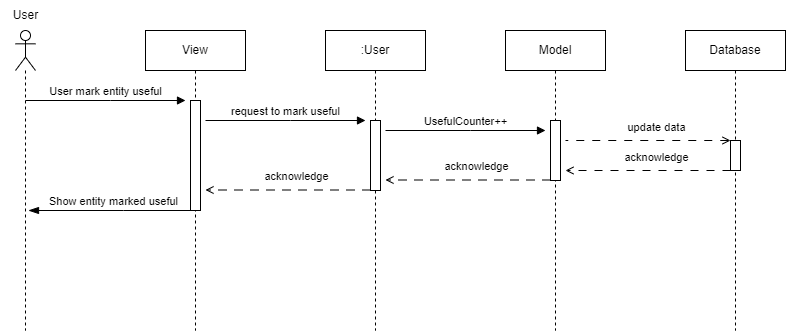
****

**Edit an Entity**

****

**Mark an Entity Useful**

A user marks an entity useful when they find information that should be shared. The User marks the entity as useful and the viewer handles the request. The User class will increment a counter in the entity and update the database.

****

**Generate Link**

When a user creates an entity the CreateEntity function will handle the creation of a new link that will be saved in the database for later retrieval. The User class holds a string of the generated link that will be passed back to the user by the viewer.

