UNIVERSITY OF CALGARY

ENSF 607

Advanced Software Development and Architecture
Design Document for Stock Recommendation System

Objective:

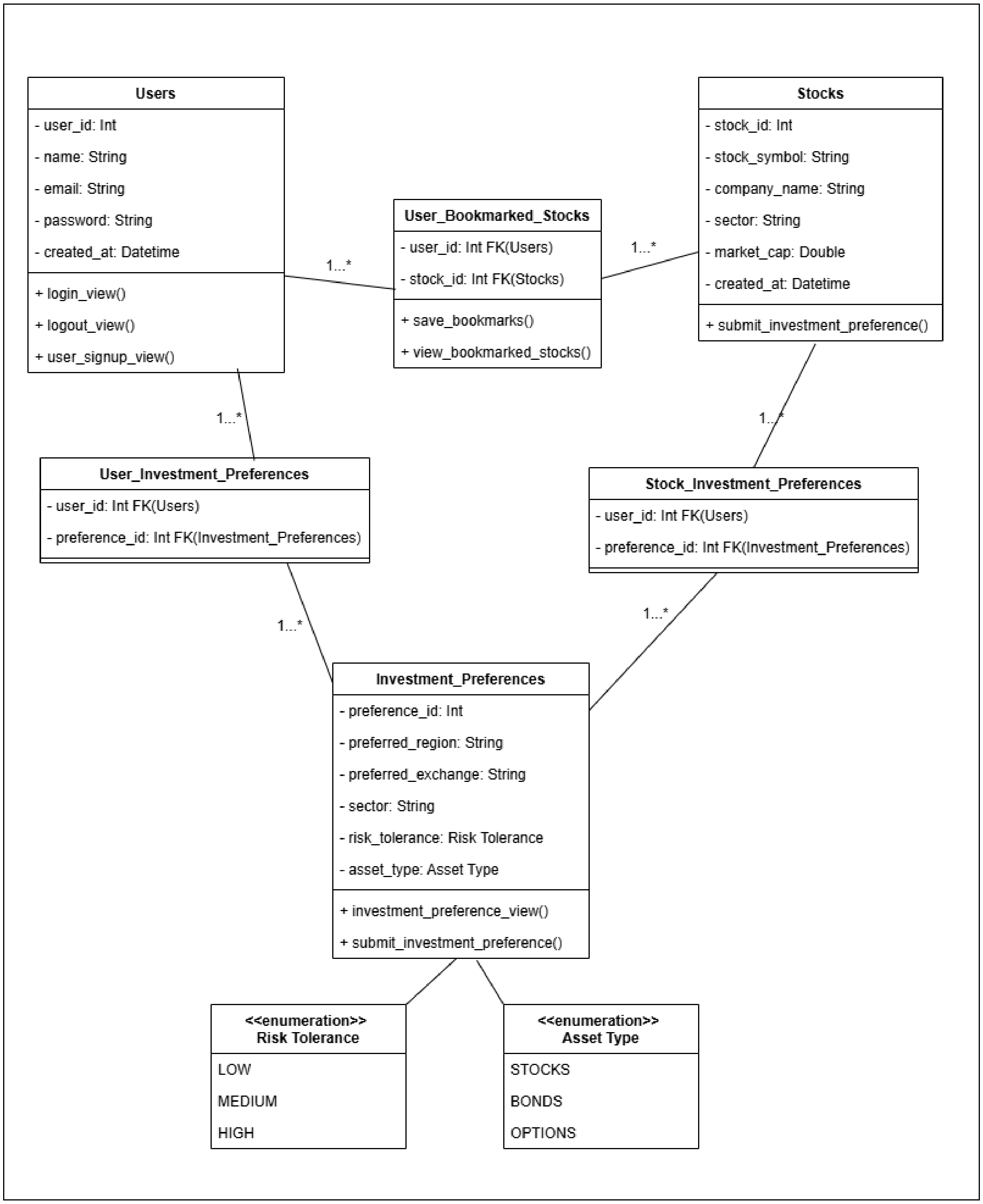
Present the design and architecture of your application, demonstrating your understanding of design patterns and principles.

Architecture Definition:

The system uses a layered architecture with the following components:

1. **Models (M)**: Define database structure using Django ORM for entities such as Users, Stocks and Investment Preferences.
2. **Views (V)**: Handle user requests and return appropriate responses.
3. **Services**: Contain business logic, such as fetching stock recommendations or validating preferences.
4. **SQL Queries**: Perform low-level database operations for performance-critical tasks.
5. **Controller (C)**: Django's URL routing system acts as the controller, delegating requests to appropriate views

Class Diagram:



Design Patterns:

Our chosen design pattern is the Model-View-Controller (MVC) architecture, which we adopted to ensure a clear separation of concerns between different layers of the application. This design pattern enhances maintainability, scalability, and collaborative development by compartmentalizing the application into three distinct components:

The **model**, defined in models.py, represents the application's data structure and manages database interactions. By encapsulating the data layer, it ensures that changes to the schema or business logic remain independent of the view or controller layers, promoting modularity and maintainability. The models.py file contains the logic for defining database tables, relationships, and methods for querying and manipulating data.

The **view** in this project refers to the HTML files, which render the web app's user interface. For instance, login.html serves as a part of the view layer, defining the presentation logic for the login page and determining how it appears to the user.

The **controller**, implemented in views.py, acts as the intermediary between the model and the view. It processes the HTTP requests, executes the required business logic, and prepares data to be sent to the front end. It acts as the intermediary between the model and view, ensuring smooth communication and functionality.

Why MVC is Ideal for Our Project

- **Separation of Concerns**: By dividing the application into Model, View, and Controller components, we reduce the interdependencies between the data, logic, and presentation layers. This makes debugging and updates simpler and safer.
- **Scalability**: MVC supports the addition of new features and functionalities without disrupting existing components. For example, we could add new Views or Controllers for additional features without modifying the core data Model.
- **Collaborative Development**: Different team members can work independently on the Model, View, or Controller, improving development efficiency. Designers can focus on the View, while backend developers concentrate on the Model and Controller.
- **Testability**: The modular nature of MVC makes unit testing and integration testing more effective. We can independently test the data models, controllers, and views, ensuring robust application behaviour.
- **Maintainability**: Over time, as requirements evolve, the application remains easier to update and refactor since each layer has a defined responsibility.

SOLID Principles:

Single Responsibility Principle (SRP)
Every class or module should have one reason to change, meaning it should only have one job

Example in our project:

- Model solely responsible for defining the data schema and business logic related to the database
- Views focus on the presentation layer, managing how data is displayed to the user
- Controllers manage request/response cycles and business logic

Interface Segregation Principle (ISP)
Clients should not be forced to depend on interfaces they do not use

Example in our project:

- In the HTML files, we use specific CSS and JavaScript only for the pages that need them, avoiding bloated dependencies for simpler pages like the login page