Creating Password-Secured Accounts:

1. - Utilize Django's authentication system for account creation and login.
2. - Store passwords as hash values using secure algorithms like PBKDF2 with SHA256.
3. - Extend the Django User model for additional information as needed.

Distinct User Types (Students and Teachers):

1. - Implement groups and permissions in Django for different user roles.
2. - Assign specific capabilities and access levels to students and teachers.
3. - Design distinct functionalities for each user type, like course creation for teachers and course enrollment for st

Collecting and Storing User Information:

1. - Gather essential details such as username, email, real name, and photo.
2. - Include additional fields for teachers like bio/qualifications.
3. - Ensure privacy and data protection in line with relevant laws.

Entities and Relationships Consideration:

1. - Plan an overview of entities like users, courses, enrollments, feedback, and materials.
2. - Focus on the relationships and interactions between these entities for detailed future discussion.

User Home Page Features:

1. - Display user information prominently, including username, real name, and photo.
2. - Show registered courses, upcoming deadlines, and status updates dynamically.
3. - Plan for home page discoverability and visibility with appropriate privacy controls.

Students Posting Status Updates on Home Page:

1. - Enable students to post status updates on their home pages.
2. - Include features for text input, multimedia attachments, and real-time visibility.
3. - Consider privacy and visibility settings for these updates.

Course Feedback by Students (Finalized Discussion Point):

1. - Implement a feedback mechanism for students to leave ratings and reviews on courses.
2. - Design an intuitive feedback form within the course interface, accessible at appropriate times.

3. - Establish a CourseFeedback table, relating it to Student and Course tables to store feedback.
4. - Map relationships to allow for a one-to-many link between Course and Course_Feedback, and a many-to-mar
5. - Use feedback data for continuous course improvement and to inform student course selection.

Search Functionality for Teachers (Finalized Discussion Point):

1. - Equip teachers with the ability to search for students and other teachers within the platform.
2. - Optimize search with indexes and leverage Django's Q objects for advanced queries.
3. - Set boundaries on searchable student information for privacy compliance.
4. - Implement intuitive UI for search, with considerations for student search limitations and privacy.
5. - Design the search functionality to ensure a balance between user discoverability and data protection.

Course Creation and Material Upload by Teachers (Finalized):

1. - Enable teachers to create courses and upload materials.
2. - Implement file upload functionality and organize files by course and user role.
3. - Plan for dynamic directory structures and efficient file retrieval.
4. - Address version control for course materials to manage updates.

Course Management and Student Enrollment Visibility for Teachers (Finalized):

1. - Develop features for teachers to view their courses and see lists of enrolled students.
2. - Define course fields and manage the many-to-many relationship between students and courses.
3. - Provide functionalities for real-time enrollment updates and enrollment management.

Real-Time Communication with Web Sockets (Finalized):

1. - Establish real-time communication channels using WebSockets for features like text chat.
2. - Utilize Django Channels to manage WebSocket connections and messages.
3. - Implement consumers for WebSocket session management.
4. - Secure WebSocket connections with appropriate authentication.

5. - Design the client-side to handle real-time UI updates and user notifications.
6. - Ensure scalability and robustness through testing and using a channel layer like Redis.

Comprehensive Requirements for eLearning Platform (Finalized):

1. - Account creation and management functions with secure login and logout processes.
2. - Teachers' ability to search for students and others, add new courses, and remove/block students.
3. - Students' ability to enroll in courses, leave feedback, and chat in real-time.
4. - Both students and teachers can post status updates, with teachers additionally able to upload course files.
5. - Notifications for user actions such as course enrollments and material additions.
6. - Technical requirements for the proper use of Django models, migrations, forms, validators, serialization, Djang
7. - A robust database model to effectively handle the relationships between accounts, courses, and user interacti
8. - Implementation of a RESTful interface for user data access and server-side code testing.