**Week 2**

**Functional requirements:**

**Use Cases**

* User Registration and Authentication
  + UC-1: Register with a university email (e.g., .edu) to ensure only students join.
  + UC-2: Log in using email and password for secure access.
  + UC-3: Reset forgotten password via email for account recovery.
* **Profile Management** 
  + UC-4: Edit profile information (e.g., name, bio, profile picture) to personalize accounts.
  + UC-5: View other users’ profiles to connect with peers.
* **Home Feed** 
  + UC-6: View a feed of posts from friends, groups, or university announcements for updates.
  + UC-7: Post text, images, or links to the feed to share content.
  + UC-8: Like, comment on, or share posts to engage with others.
* **Group Chats and Study Groups** 
  + UC-9: Create or join study groups for specific courses to collaborate academically.
  + UC-10: Send and receive real-time messages in group chats for instant communication.
* **Event Calendar** 
  + UC-11: View a calendar of campus events, deadlines, and club activities for awareness.
  + UC-12: Add personal events or deadlines to manage schedules.
  + UC-13: Receive notifications for upcoming events to stay informed.
* **Coursework Forums** 
  + UC-14: Post questions or resources in course-specific forums to support learning.
  + UC-15: Reply to forum threads to foster discussion.
* **Marketplace** 
  + UC-16: List items for sale (e.g., textbooks, furniture) to trade within the community.
  + UC-17: Browse and search for items in the marketplace to find needed goods.
  + UC-18: Contact sellers or buyers through in-app messaging for transactions.

**User Stories**

* As a student, I want to join study groups for my courses so I can collaborate with classmates.
* As a student, I want to see all campus events in one place so I don’t miss important activities.
* As a student, I want to buy and sell items within a trusted university community.
* As a club leader, I want to post events and updates to reach more students.
* As a user, I want real-time chat features to communicate instantly with my peers.
* **Non-Functional requirements:**

**1. Performance**

* The system should load the homepage/feed within 2 seconds.
* Support at least 100 concurrent users with no significant slowdown (initial goal).
* Use caching (e.g., Django cache, Redis) for frequently accessed data (e.g., user profile, trending posts).

**2. Scalability**

* The app should be designed to scale horizontally (e.g., add more servers).
* Use modular components (e.g., microservices or well-separated Django apps).
* Easily upgradeable to CDN/Cloud Storage for handling high volumes of media files.

**3. Availability**

* The system should be available 99.5% of the time.
* Scheduled maintenance should not exceed 1 hour/month.
* Use robust hosting (e.g., Heroku, Render, DigitalOcean, or AWS) with health checks.

**4. Security**

* All user data must be stored securely with password hashing (e.g., bcrypt).
* The system should implement CSRF protection, input validation, and rate limiting.
* HTTPS should be enforced for all communication.

**5. Usability**

* UI must be responsive and work on mobile, tablet, and desktop.
* Users should be able to perform common tasks (post, like, comment) within 3 clicks.
* The interface should follow accessibility standards (WCAG 2.1 AA).

**6. Maintainability**

* Code should follow Django best practices and PEP 8 style.
* Use clear separation of concerns (models, views, templates, static files).
* Unit tests should cover at least 60% of backend logic.
* **System Constrains:**

**1. Technology Stack Constraint**

* + Must build with Django’s architecture (MVT pattern)
  + Limited to Django-compatible libraries
  + Real-time features may require Django Channels or polling (not native in base Django)

**2. Hosting/Deployment Constraint**

* May use platforms like Heroku, Render, or a VPS
* Must respect platform limits:
  + Free/low-tier plans might sleep apps after inactivity
  + Limited storage and bandwidth

**3. Storage Constraint**

* **Image/video content requires file storage:**
  + If storing locally: limited by server disk space
  + If using cloud (e.g., S3): must handle integration and cost

**4. Performance Constraint**

* Running Django with SQLite (in dev) limits concurrent user support
* Without caching (e.g., Redis), high traffic may slow down feed loading

**5. Security Constraint**

* You must implement secure password storage and session management
* Django helps here, but you’re responsible for:
  + CSRF protection
  + Input sanitization
  + Access control

**6. Budget Constraint**

* Likely working with free tools and services
* Can’t afford premium APIs, cloud plans, or team collaboration tools
* **Assumptions:**

|  |  |
| --- | --- |
| **Assumption** | **Why It Matters** |
| |  | | --- | | Users have stable internet connections |  |  | | --- | |  | | |  | | --- | | Required for web interaction and real-time features |  |  | | --- | |  | |
| Users access the system from modern browsers | So that JavaScript and responsive CSS behave correctly |
| |  | | --- | | The server environment supports Python and Django |  |  | | --- | |  | | |  | | --- | | Basic requirement to run the backend |  |  | | --- | |  | |
| Database server is always available and responsive | Posts, users, comments rely on DB queries |
| |  | | --- | | Media uploads (e.g., images) won't exceed a reasonable size (e.g., <5MB) |  |  | | --- | |  | | |  | | --- | | Prevents storage overload or upload failures |  |  | | --- | |  | |
| |  | | --- | | Admin or dev team will monitor and maintain the system |  |  | | --- | |  | | Assumes someone will fix issues, deploy updates, etc. |

* **Dependencies:**

|  |  |
| --- | --- |
| **Dependencies** | **Role In Your App** |
| Django Framework | |  | | --- | | Core backend engine for handling routing, models, views |  |  | | --- | |  | |
| Python | Runtime for Django |
| Database | Stores all app data (users, posts, comments, etc.) |
| |  | | --- | | Static files tools (optional, like Tailwind) |  |  | | --- | |  | | For styling the UI |
| File storage (local or cloud) | To store media (images, videos) |
| Web server | To serve your Django app in production |
| Hosting platform | Required to run the app live on the web |