Part I

a. If each zip file is generated only when requested, the server would be under a lot of load during busy times like before minors.
b. If each folder's zip is saved, there would be problems for storage when courses are to be updated.
c. To me, the best method seems to be that zip files should be stored for some fixed time after generation.
    I. When a zip is requested, generate it or serve it directly if it is stored already.
    II. After some fixed time from the request, delete the zip.
    III. If a zip is online and the course is updated, generate the new zip and replace it.

   This would reduce wait times when there are many requests and storage space is efficiently used.

Part III

1) a. **Websites**

   The browser sends the url to the ISP and its DNS server initiates a DNS query to find the IP address of the server that hosts the url. Then a TCP connection is initiated with the serveer using the IP adress obtained. Now, the browser sends HTTP requests to the web server. The web server uses a program to handle requests and send back the HTTP response with HTML, CSS and JS files which are interpreted by the browser which displays the website's content.

   The web server has programs (made using frameworks like Django, Laravel,etc.)

   b. **Local Network Server**

   Run a local server using 'python -m http.server 9000' in the directory containing the html file. Then access it from http:// [IP]:9000 from a computer in local area network.

   c. **Nginx and Apache**

   They are open source web servers and are the two most common ones. Apache allows more flexibility in choosing connections and request handling algorithms at the cost or resource consumption. Nginx was designed to use asynchronous algorithms. Its connection processing model allows its resource consumption to stay consistent even with heavy load. But Nginx can't process dynamic content (scripts) itself.

   d. **Outside network**

   For it to work, some router settings will have to be changed. Source

2) **Database system**

A database system is a software system that enables users to define, create, maintain and control access to the database. They are mainly used for data storage, retrieval and update applications in a project.

A SQL type data structure would be better as it allows relational queries, and enforces field constraints. We would need to get information about related things like projects(citations, publishing information, etc) by using SQL. A normal structured schema (SQL) would better fit this project. Scalability, which is better when using NoSQL is not required here as types of information about the professor would mostly remain the same.

There could be a model for professor like this.

Name(str)	Description (str)	Image	Projects(obj)	Research papers(obj)	Other personal info	

And models for Projects and Reasearch data.

3) **SetTImeout and setInterval**

SetTimeout and setInterval may cause bugs in some applications. For example(in asynchronous operation), when setInterval() is used to make calls to function using setTimeout(), the result would be unexpected as setInterval would not wait for setTimeout() to execute and setTimeout() would work in a different rhythm.

Also in synchronous operation, setInterval() changes its rhythm if it executes a time intensive operation. So, setTimeout and setInterval do not guarantee time delay.

There are alternatives using 'requestAnimationframe()'. Source

Browser makers intentionally degrade time measurement capability to make attacks using the computer security vulnerabilities Meltdown and Spectre difficult as precisely measuring time difference is a critical part of these attacks.

Rating Portal

Models

1. Review (author,content,date,information about the page, score, anonymity)
2. Like and Dislike (1 to 1 connection with review and Many to Many with Users)
3. Report (information about review, date reported)
4. Recent Action (message, user, link, info/alert)
5. Custom User (to use time till banned, created automatically on user creation)
6. Course (rating, no of reviews, page type (‘p’ or ‘c’), name, ordered using name by default)

Pages

1. Course
    a. To make a new page
    b. List of Courses/Professors
    c. Course/Professor page
2. Review
    a. Admin action
    b. List of all reviews(accessible only to admin)
    c. List of all issues(accessible only to admin)
    d. Form pages for deleting and reporting
    e. A template for a single review(flexible with original page)
3. User
    a. Pages for login, logout, register and changing password.
    b. Page to view top users(by points).

Features

1. For all
    a. View all courses and professors (ordered by name) and search through them
    b. View the pages of a course, professor or a user and see details that are allowed
    c. Register and login pages (using unique iitd email, also validate email by checking comparing prefixes)

2. For users
    a. Rate(0-10) and review courses(anonymously as well).
    b. Like, dislike (using buttons submitting post requests), get points (using relations with reviews written)
    c. Report other’s reviews(creating report objects connected to a review by foreignkey) and delete own reviews.
    d. View own recent actions (report,delete,like,dislike,review) (creating a RecentAction Object)
3. For admin
    a. View recent issues and ignore, delete and warn(using recent action object) user or ban user (remove account permanently or prevent login by validating on login attempt)
    b. View all reviews and user names(even if anonymous).
4. On a Page
    a. Ratings which are updated whenever all courses or any course is viewed.
    b. Search bar sending a guest request to make queries.

Possible Improvements

1. Sorting reviews by date,points,etc.
2. Pagination
3. Password reset by email
4. Social Authentication
5. Using javascript instead of POST for likes and dislikes.
6. Change camelCase to snake_case to correct form labels