MVP Implementation

1. Project Setup:

Requirement Analysis:

- Identify the main objectives of the project.
- Understand the project scope and define the minimum features required for an MVP.

Technology Stack Selection:

- Frontend: React / Angular / Flutter (depending on web or mobile).
- Backend: Node.js / Django / Spring Boot.
- Database: SQLite / MySQL / MongoDB (for scalability).

Development Environment Setup:

- Install IDEs such as VS Code, Android Studio, IntelliJ.
- Configure project folders for frontend, backend, database, and documentation.
- Install Git and connect with GitHub for version control.

Basic Configuration:

- Initialize project using commands like Django - admin start project, or flutter create.
- Install dependencies: Axios/Fetch for APIs, Tailwind/Bootstrap for UI, ORM (Hibernate) for database.
- Configure coding standards with Prettier.

2. Core Features Implementation:

Authentication System:

- User registration with fields like name, email, and phone.
- Login/Logout functionality with validation.
- Role-based access control: Admin, Student, Mentor.

Dashboard:

- Personalized dashboard for each role.
- Easy navigation to all important modules.

Main Functional Modules:

- Students: Access courses, training materials, and assessments.
- Mentors: Upload content, assign tasks, monitor student progress.
- Admin: Manage users, courses, and announcements.

Notifications / Updates:

- Display important announcements and updates.
- Simple notification system (local alerts or email).

Basic UI/UX Design:

- Responsive layout for desktop and mobile.
- Clean and accessible interface with user-friendly navigation.

3. Data Storage (Local State / Database):

Local State Management:

- Use React Context API / Redux or Flutter Provider for temporary data management.
- Store session data, forms, and temporary user actions in local state.

Database Design:

- Entities: Users, Courses, Modules, Assignments, Notifications.

Relationships:

- One user can enrol in multiple courses.
- One course can have multiple modules.
- Assignments are linked to students and mentors.

Database Operations:

- CRUD operations for all entities.
- Use ORM/ODM for better database handling.

Sample Implementation:

- Local SQLite for MVP phase.
- Upgrade to MySQL/PostgreSQL for production-ready version.

4. Testing Core Features:

Unit Testing:

- Test individual components like login, registration, course listing.
- Use Jest (React), Py Test (Django), or JUnit (Java).

Integration Testing:

- Ensure frontend communicates correctly with backend via APIs.
- Verify accurate data storage and retrieval.

User Testing:

- Deploy MVP to a small group of users.
- Collect feedback on usability, design, and performance.

Bug Fixing & Optimization:

- Identify crashes, errors, or slow-loading features.
- Optimize database queries and API response time.

5. Version Control (GitHub):

Repository Setup:

- Create a GitHub repository.
- Add a README.md with project description, setup instructions, and contributors.

Branching Strategy:

- Main Branch: Stable release.
- Development Branch: Active development work.
- Feature Branches: Separate branches for individual features.

Commit Practices:

- Use meaningful commit messages like Added login API, Fixed dashboard UI.

- Commit frequently to track progress and changes,

Collaboration:

- Use GitHub Issues for task management.
- Review requests before merging to main branch.

Documentation & Releases:

- Maintain a changelog.
- Use GitHub Releases to track MVP, Beta, and Production versions.