**Project X: Issues Encountered and Implemented Solutions**

|  |  |
| --- | --- |
| Problems / Questions Encountered | Research Solutions |
| 1. How to prevent unauthorized devices from accessing the system? | We implemented a **device registration system** that only allows pre-approved device IDs to access core system functions. Each device is linked to a specific user role. |
| 2. GPS accuracy issues during location-based attendance. | We refined the **geofencing radius** and used **device-level GPS calibration prompts** to ensure accurate location detection within acceptable margins. |
| 3. Students attempting to spoof QR codes using screenshots. | Added **dynamic QR codes** with timestamps and encryption, which invalidate after a short duration to prevent reuse or spoofing. |
| 4. OTPs were not being delivered to some students in remote areas. | Introduced **offline OTP fallback using device authentication**, and allowed **email-based OTPs** as an alternative to SMS delivery. |
| 5. How to generate different types of reports from one system? | We created a **modular reporting module** with filters and export options (CSV/PDF), which allows dynamic selection of report types and data ranges. |
| 6. Low image quality from student ID photo capture. | Enforced **minimum camera resolution requirements** and integrated **photo preview/retake** features before final submission. |
| 7. How to manage different access roles for Admins, Instructors, and Students? | Implemented a **role-based access control (RBAC)** system to restrict features and data visibility based on user roles. |
| 8. What if an instructor loses their registered device? | Enabled an **admin panel option** to remotely deregister lost devices and reassign access to a new registered device. |
| 9. Students unable to view their attendance logs in real time. | Optimized the **real-time synchronization** between the database and student app using efficient polling and caching mechanisms. |
| 10. Securing API access to the cloud database. | Applied **token-based authentication (JWT)** and enforced **HTTPS** for all API communications to restrict and encrypt access. |
| 11. Difficulties in testing modules independently. | Adopted a **modular development approach** and used **unit testing frameworks** like Jest (for frontend) and Postman tests (for APIs). |
| 12. Confusion over system requirements during UAT. | Created a **clear checklist and UAT script** aligned with our requirement list (R01–R015) and clarified expectations through demos. |
| 13. Initial UML diagrams lacked clarity. | Reviewed system flow with the team and **refined the diagrams** using tools like Draw.io or Lucidchart to better reflect entity relationships. |