Stakeholders:

1. Recognized

* Stakeholder groups identified:

→ We know who will use or be impacted by the system—such as users, admins, developers etc.

* Key stakeholder groups represented:

→ Key individuals from each group (such as an admin rep or end-user tester) are engaged to provide feedback.

* Responsibilities defined:

→ Everyone knows what they are to do—users log in, admins maintain users data, developers implement features, etc.

2. Represented

* Responsibilities agreed:

→ Each stakeholder (like admins or third-party developers) understands and agrees to their role in the project.

* Representatives authorized:

→ Those speaking for each group have the right to make decisions or give feedback (e.g., lead developer or admin head).

* Collaboration approach agreed:

→ We’ve decided how to work together—via meetings, online tools, or documents.

* Way of working supported & respected:

→ Everyone's work style of choice is honored—such as remote access, timed feedback, or code review.

3. Involved

* Representatives are there to help the team:

→ Stakeholders provide assistance if and when necessary, such as testing features or offering improvements.

* Timely decisions and feedback delivered:

→ Rapid feedback is delivered to avoid slowing down development.

* Changes communicated quickly:

→ If there is something critical that is changing, it is communicated immediately to the team.

4. In Agreement

* Low expectations align:

→ Everyone is aligned on the things the system has to do—such as login using face or password, secure storage, etc.

* Reps satisfied with their stake:

→ Stakeholders are engaged just enough—not too much or not at all.

* Reps' voice heard:

→ If a stakeholder raises input, the team listens and acts.

* Team's voice heard:

→ Developers' and testers' voices are heard too when decisions are made.

* Priorities clear & perspectives balanced:

→ We all understand what goes and everyone's opinions are taken into account.

5. Satisfied for Deployment

* Stakeholder feedback received:

→ All stakeholders check and provide final feedback before releasing the system.

* System ready for deployment:

→ Stakeholders verify that the system satisfies requirements such as secure login, API integration, and user-friendly design.

6. Satisfied in Use

* Feedback on system use available:

→ Once the system has been activated, users and administrators can provide feedback in the form of reports, forms, or sessions.

* System meets expectations:

→ Once deployed, the stakeholders are satisfied that the system performs efficiently and as expected.

Opportunity:

1. Identified

* Opportunity identified and described:

→ The requirement for stronger, flexible authentication (such as face + username) was observed as an improvement in security over conventional login procedures.

* One or more interested stakeholders exist:

→ Stakeholders such as users, security analysts, and admins favor the concept and are willing to be engaged.

* Stakeholders affected by the opportunity identified:

→ All impacted groups—users (who will log in), developers (who'll create it), and admin (who'll edit or modify user’s data)—are well-defined.

2. Solution Needed

* Stakeholders' needs determined:

→ Users clearly need easy but secure login mechanisms, and admins need tracking and control mechanisms for user’s data.

* Root cause(s) of problem(s) identified:

→ Single-password or small-password mechanisms are very vulnerable to hacking, and most platforms don't have real-time tracking or biometric security—this is the deficiency.

* Need to act justified:

→ In order to enhance security and user confidence, a more robust system is obviously needed, particularly in high-risk environments.

* One or more candidate solutions suggested:

→ A multimodal authentication system based on username + biometrics, admin controls, username + face, forgot password etc.

3. Value Established

* Opportunity value quantified:

→ Enhanced login security and enhanced user management will save time and minimize risks for users and admins.

* Solution impact understood:

→ The solution will enhance safety, user satisfaction, and monitoring capacity across platforms.

* System value understood:

→ The value is in simple integration, multi-device compatibility, and robust authentication—significant to both users and admins.

* Success criteria clear:

→ Success is in the form of fewer login failures, quicker authentication, and safe recovery options.

* Outcomes clear and quantified:

→ Targets: 95%+ login success rate, <2s authentication time

4. Viable

* Solution outlined:

→ A secure system with face, fingerprint and password login, admin dashboard, and recovery features.

* Solution possible within constraints:

→ Can be implemented with existing tech (Node.js, JS, MySQL, APIs) within time and cost constraints.

* Risks acceptable & manageable:

→ Secure coding, testing, and user validation reduce the risk of misuse or failure.

* Solution profitable:

→ Time-saving for users and admin, prevents data leaks—resulting in long-term cost savings.

* Reasons to develop solution understood:

→ It addresses actual problems: weak passwords, no admin tracking, and bad security.

* Pursuit viable:

→ Team capabilities, stakeholder buy-in, and tech availability make it possible.

5. Addressed

* Opportunity addressed:

→ Major issues (poor login, email verification) are addressed by this new system.

* Solution worthy of deployment:

→ System delivers definite value, addresses genuine issues, and enhances trust and user experience.

* Stakeholders satisfied:

→ Feedback from the beginning indicates users and admins are content with system features.

6. Benefit Accrued

* Solution accrues benefits:

→ Users benefit from quicker, more secure logins; admins are able to monitor and manage the system with ease.

* ROI acceptable:

→ Low cost of development + high security and usability impact = excellent return on investment.

Software System:

1. Architecture Chosen

* Criteria agreed for selecting architecture

→ Chose multi-platform (web-based with Node.js/Python backend, HTML/JS frontend) supporting.

* HW platforms targeted

→ Planned to operate on desktops, laptops, and mobile browsers (Windows, Mac, Linux supported).

* Technologies chosen

→ Node.js, Express, MySQL (Cmd), JavaScript, HTML/CSS

* System boundary understood

→ System takes care of user login, register, password recovery and admin dashboard.

* Decisions on system organization taken

→ Clean separation: frontend does UI, backend does Database interaction and stores and modifies the data.

* Buy, build, reuse decisions taken

→ Primarily build in-house with open-source tools; reuse libraries for email, image capture, and validation.

* Key technical risks identified

→ Risks: Good accuracy of face/fingerprint model, reliability of email delivery, and browser compatibility.

2. Demonstrable

* Key architectural features demonstrated:

→ Successfully demonstrated smooth login, admin panel, password reset, and Database connection across browsers and all other methods.

* System exercised & performance monitored

→ Login time is less and the user can SignIn and SignUp fast without any issues

* Critical HW configurations proved

→ System tested on many OS, browser types, and devices.

* Critical interfaces proved

→ Face and fingerprint input, email token system, and admin controls all verified.

* Integration with environment proved

→ Email, browser microphone/camera, and DB server integrated and working.

* Architecture accepted as fit-for-purpose

→ Selected structure accommodates the functional and performance requirements gracefully.

3. Usable

* System can be used

→ Simple-to-use login, registration, and admin functionality—tested with sample users.

* System functionality tested

→ All modules (login, face/fingerprint, token reset, admin edit/search/delete) tested.

* System performance acceptable

→ Performs with good speed, does not crash, and can scale up to moderate traffic.

* Defect levels acceptable

→ Small bugs picked early through CI/CD and user testing—no critical defects.

* System fully documented

→ Architecture, features, and modules well documented; includes user manual as well.

* Release content known

→ Features for each version (basic login, then face/fingerprint, then admin panel) are pre-decided.

* Added value clear

→ Convenient access, admin control, and multi-platform compatibility.

4. Ready

* User documentation available

→ Easy-to-follow guides for users and admin to register, log in, reset password, and manage users.

* System accepted as fit-for-purpose

→ Meets functional, usability, and security tests—ready for real deployment.

* Stakeholders want the system

→ Testers', users', and reviewers' feedback is positive—they can see its advantages.

* Operational support in place

→ Clear support plan with token expiry handling, database backup, and error logs

5. Operational

* System available for use:

→ Hostel management system is live and accessible through browser now.

* System live:

→ Users can register, log in, reset passwords, and admins can manage data live.

* Agreed service levels supported:

→ Satisfies expected uptime, data response, and email delivery specified in SRS.

6. Retired

* Replaced or discontinued:

→ Will be retired in case a new version or another system replaces it

* No longer supported:

→ No updates or maintenance after system is archived.

* No authorized users:

→ Admin access stripped and user accounts shut down after retirement.

* Updates halted:

→ No new features, bug fixes, or support delivered.

Work

1. Initiated

* Required result evident

→ Target: create secure, user-friendly hostel system with admin capabilities.

* Constraints evident

→ Limited time for development, specified budget, and minimal system requirements.

* Funding stakeholders known

→ Project directed under academic resources or in-house funding.

* Initiator identified

→ We initiated the development plan.

* Accepting stakeholders known

→ Faculty/instructors or admin team who will use or review the system.

* Source of funding clear

→ Institutional/college-level project with no commercial cost.

* Priority clear

→ Project completion before deadline with full functionality is the main goal.

2. Prepared

* Commitment made

→ Team agreed to deliver project within given time.

* Cost and effort estimated

→ Time and tasks were estimated in Gantt chart and planning phase.

* Resource availability understood

→ Available: 1–2 developers, laptop/PC, MySQL server, and email APIs.

* Risk exposure understood

→ Risks such as token expiry failure, email undelivered, or DB errors identified.

* Acceptance criteria established

→ Login, reset, and admin controls should function without errors for acceptance.

* Sufficiently broken down to begin

→ Tasks divided into modules: auth, token, admin features, etc.

* Tasks identified and prioritized

→ Login and DB first, then email, admin dashboard, search/delete features.

* Credible plan in place

→ All modules traced to individual milestones with timelines.

* Funding arranged

→ No money funding required—support and tools available.

* One team member prepared

→ We (and any co-developers) were prepared to develop.

* Integration points established

→ DB, frontend, backend, and email API all established and connected.

3. Started

* Development commenced

→ Initial setup, database, and initial modules such as login implemented.

* Monitoring of progress

→ Periodic check-ins or personal tracking of progress employed.

* Definition of done established

→ "Done" refers to feature tested without bugs on local and remote servers.

* Tasks in progress

→ Modules actively written, debugged, and refined while developing.

4. Under Control

* Tasks getting done

→ Modules such as login, reset, dashboard were completed within schedule.

* Unplanned work in control

→ Controlled changes such as introducing search filter or resolving token expiry bugs.

* Risks in control

→ Email API tested and token expiry errors resolved proactively.

* Estimates updated to indicate performance

→ Revised schedule for surprise bugs or module updates.

* Progress tracked

→ Measured expected vs actual task completion in a chart or tracker.

* Re-work in check

→ Re-did or corrected portions such as admin table sorting or email error.

* Commitments made on time

→ Completed each module by its new or original deadline.

5. Completed

* Admin tasks remaining

→ Coding/testing complete—left only with deployment or documentation.

* Results attained

→ Functional system with working face/fingerprint module, admin features, and login.

* Resulting system accepted

→ Either evaluators/testers or mentor reviewed and signed off the final version.

6. Closed

* Lessons learned

→ Learned integration problems, email delay handling, and deployment tricks.

* Metrics available

→ Performance tested for login speed, DB response time, and bug counts.

* Everything archived

→ Final code, documentation, and diagrams stored securely.

* Budget reconciled & closed

→ No budget consumed; resource usage reviewed and closed.

* Team released

→ Development complete—no further coding required.

* No pending, incomplete tasks

→ All the features outlined in SRS finished and delivered.

Way of Working

1. Principles Formulated

* Team actively endorses principles

→ Everyone consented to adhere to well-defined coding and testing practices.

* Stakeholders concur with principles

→ Faculty/project guide endorsed the development approach.

* Tool requirements agreed

→ Agreed on Node.js, MySQL, Nodemailer, etc.

* Approach suggested

→ Selected iterative development with frequent testing and feedback.

* Operational context realized

→ Aware that the project was for student housing with admin control requirements

* Practice & tool limitations known

→ Aware of constraints such as email API rate, local DB storage, etc.

2. Foundation Established

* Selected practices & tools

→ Git, VS Code, MySQL, and Express chosen as tools.

* Practices to begin work decided

→ Coding, testing, and version control strategy finalized.

* Non-negotiable practices & tools determined

→ Should implement secure password storage and email verification.

* Gaps between available and necessary way of working understood

→ Had to learn password hashing or secure token management early enough.

* Gaps in capability understood

→ Recognized lack of experience in error handling and email APIs.

* Integrated approach to working available

→ Integrated frontend, backend, and DB work into seamless workflow.

3. In Use

* Practices & tools being used

→ GitHub commits, Nodemailer, token expiration—used every day.

* Regularly reviewed & optimized

→ Bugs checked over, code tuned after testing.

* Evolved to fit context

→ Altered logic slightly when email delivery didn't work in local mode.

* Supported by team

→ Everyone used same stack and adhered to agreed rules.

* Feedback mechanisms in use

→ Peer feedback, mentor comments, or individual retrospectives utilized.

* Practices & tools facilitate collaboration

→ Git and organized modules facilitated easy collaboration.

4. In Place

* Utilized by entire team

→ Entire team used agreed approach and shared tools.

* Accessible to entire team

→ Project files and tools were on Git or shared folder.

* Checked and modified by entire team

→ Code reviews and discussion enabled adaptation and feedback.

5. Working Well

* Predictable progress being achieved

→ Work completed largely on schedule with reliable updates.

* Practices emerge naturally

→ Implemented async calls, token verification, and user role checks effortlessly.

* Tools emerge naturally to support way-of-working

→ Tools such as Postman, Git, and MySQL supported smoothly without additional configuration.

* Continuously tuned

→ Configured API timeouts, DB queries, and email retries for improved flow.

6. Retired

* No longer in active use

→ Project development complete, no longer any active workflow or tool modifications.

* Lessons learned shared

→ Noted what was effective (e.g., token stream) and what was problematic (e.g., email issues).

Team

1. Seeded

* Mission defined

→ Establish a safe login system with biometrics (fingerprint, face) + password and admin as well as audit capabilities.

* Constraints known and defined

→ Have to adhere to security policies (e.g., encryption, GDPR), write using Python/JS stack, and complete within project timelines.

* Growth mechanisms in place

→ Team members are able to pick up tools (OpenCV, Flask, MySQL) as they go.

* Composition defined

→ Backend devs, frontend devs, and security contributors are on the team.

* Responsibilities outlined

→ Every team member knows their module (auth, admin panel, email reset, etc.).

* Required commitment level clear

→ Everyone committed to completing their pieces of work prior to the deadline.

* Required competencies identified

→ Required skills: Flask, API development, DB design, JS frontend, security practices.

* Size determined

→ Small project team of 3–4 developers.

* Governance rules defined

→ Team lead manages tasks, code review, and integration.

* Leadership model selected

→ One or two leads responsible for module allocation and milestone tracking.

2. Formed

* Enough members signed up

→ Complete team allotted, all basic modules covered.

* Roles clear

→ Everyone clear on what they are coding—no ambiguity.

* How to collaborate clear

→ Code collaboration using Git, comms using meetings or messaging.

* Members familiar with each other

→ Team is already familiar (college group/project team).

* Individual responsibilities accepted and mapped to competence

→ Task assigned based on strengths: frontend, backend,DB, etc.

* Members are accepting work

→ Each of them is accountable for their features assigned.

* External collaborators identified

→ If any, such as reviewers or testers, they are known.

* Communication mechanisms defined

→ Changes committed through version control, messages, or meetings.

* Members commit to team

→ All actively contribute and adhere to project flow.

3. Collaborating

* Works as one unit

→ Code is merged, tested, and integrated in harmony—not in silos.

* Communication open and honest

→ Team openly discusses design changes, blockers, or bugs.

* Focused on mission

→ Entire focus is secure, smooth, and functional authentication.

* Members know one another

→ Already familiar with working together as a college team.

4. Performing

* Always delivering on promises

→ Modules done and combined on schedule.

* Always changing to adapt

→ Adapted to problems such as token expiration, bug fixes, or new concepts.

* Solves problems

→ Bugs or design problems are addressed and resolved quickly.

* Rework and backtracking minimized

→ Proper planning minimized unnecessary changes.

* Waste continually eliminated

→ Unused or messy code eliminated to remain clean and fast.

5. Adjourned

* Duties completed

→ All components of the system (auth, admin, email, etc.) are complete.

* Available to other teams

→ Free to work on other academic projects or tasks now.

* Mission complete

→ Project done and accepted—objective met.

Requirements

**1. Conceived**

* **Stakeholders in agreement system is to be built**

**→ All those who are involved are in agreement with developing this system.**

* **Users identified**

**→ The system users are well known and identified.**

* **Funding stakeholders identified**

**→ The financial or resource stakeholders providing support are well identified.**

* **Opportunity clear**

**→ Why the system has to be constructed and its advantage are well understood.**

**2. Bounded**

* **Development stakeholders identified**

**→ Individuals who take part in constructing the system are well outlined.**

* **System purpose agreed**

**→ Everyone is on the same page with what the system should accomplish.**

* **System success unambiguous**

**→ The criteria that define the success of the system are established.**

* **Shared understanding of the solution**

**→ Everyone shares the same understanding of what the system will be like.**

* **Format of requirements agreed**

**→ There exists a standard format in which the requirements are written.**

* **Requirements management**

**→ There is a procedure for dealing with changes and keeping track of requirements.**

* **Scheme of prioritization clear**

**→ Everyone understands which requirements are of higher priority.**

* **Constraints identified & taken into account**

**→ Constraints such as time, budget, and tech are recognized.**

* **Assumptions transparent**

**→ Any assumptions made during the planning process are explicitly stated.**

**3. Consistent**

* **Requirements shared**

**→ Requirements have been shared with all stakeholders.**

* **Requirements' source transparent**

**→ It is transparent where each requirement originated from (e.g., user need, law).**

* **Rationale transparent**

**→ The rationale for each requirement is clearly explained.**

* **Conflicts resolved**

**→ Any conflicting requirements have been addressed.**

* **Essential characteristics transparent**

**→ The major attributes the system should possess are specified.**

* **Key usage scenarios explained**

**→ How users will utilize the system is explained.**

* **Priorities clear**

**→ The highest priority features are clearly ordered.**

* **Impact understood**

**→ Individuals understand how requirements impact the system.**

* **Team knows & agrees on what to deliver**

**→ The entire team is aligned about the scope of the project.**

**4. Acceptable**

* **Acceptable solution described**

**→ An acceptable solution that fulfills the requirements is described.**

* **Change under control**

**→ Changes to requirements are handled correctly.**

* **Value to be realized clear**

**→ Everybody is aware of what value the system will deliver.**

* **Clear how opportunity addressed**

**→ It's clear how the system addresses the problem or need.**

* **Testable**

**→ Requirements are detailed enough to be tested.**

**5. Addressed**

* **Enough addressed to be acceptable**

**→ Adequate requirements have been put in place for the system to operate.**

* **Requirements and system match**

**→ The built system satisfies the documented requirements.**

* **Value realized clear**

**→ The delivered benefits of the system are apparent.**

* **System worth making operational**

**→ The system is finalized sufficiently to be applied to real cases.**

**6. Fulfilled**

* **Stakeholders accept requirements**

**→ All stakeholders agree that the requirements have been fulfilled.**

* **No hindering requirements**

**→ There are no unmet blockers or needs left.**

* **Requirements fully satisfied**

**→ All specified requirements have been implemented and proven.**