



International Journal of Research Publication and Reviews

Journal homepage: www.ijrpr.com ISSN 2582-7421

Sting – A Social Media App

Sabeetha M^{#1}, Sri Aathipriyan M^{#2}, Vishal A^{#3}, Sundarabalan D T^{#4}, Sanjay S^{#5}, Supriyan P^{#6}

^{#1} Assistant Professor, Department of Computer Science and Engineering, Sri Shakthi Institute of Engineering and Technology, India.

^{#2} Student, Department of Computer Science and Engineering, Sri Shakthi Institute of Engineering and Technology, India.

^{#3} Student, Department of Computer Science and Engineering, Sri Shakthi Institute of Engineering and Technology, India.

^{#4} Student, Department of Computer Science and Engineering, Sri Shakthi Institute of Engineering and Technology, India.

^{#5} Student, Department of Computer Science and Engineering, Sri Shakthi Institute of Engineering and Technology, India.

^{#6} Student, Department of Computer Science and Engineering, Sri Shakthi Institute of Engineering and Technology, India.

ABSTRACT :

Mainstream social media platforms emphasize curated, filtered and edited content, which can weaken authenticity and encourage unhealthy social comparison. This paper presents Sting, a mobile social networking application built around live, unedited moments and consistent daily participation. Each day, every user receives one shared task prompt, such as “capture the view from where you’re sitting.” Users respond through a feature called Unfiltered, which allows only live camera capture. It provides no gallery uploads, filters, or editing. Posting once per day builds a public streak that other users can see, creating social accountability and encouraging consistent habits. The platform also includes the usual social features: user profiles, follower and following relationships, interest-based user suggestions, a personalized feed with a trending fallback, and post likes. The system has a Flutter mobile client for Android and a Node.js/Express REST API backed by a PostgreSQL relational database. Authentication uses OTP verification, bcrypt password hashing and JWT-based sessions. Streak logic and the one-post-per-day rule are enforced server-side so clients cannot tamper with them. To discourage re-photographing a screen or printed image, the client samples the device accelerometer before capture and flags the post as “verified live” or “low motion” using a lightweight motion-variance heuristic. The prototype currently has two limitations. OTP delivery is simulated through server logs, and the live-capture flag is reported by the client rather than verified on the server. Planned future work includes server-side authenticity verification, cloud media storage, push notifications, and gamification features such as streak freezes, badges and leaderboards.

Keywords: Social Networking, Authenticity, Habit Formation, Streaks, Flutter, Node.js, PostgreSQL, Mobile Application, Live Capture.

I. INTRODUCTION

Social media has become the main way many people share everyday life, and its core mechanics of profiles, connections and a feed of shared content are now well understood [1]. The dominant feed model, however, rewards polished, edited and carefully timed posts. Social comparison theory explains why this matters: people evaluate themselves against others, and curated content gives them a skewed standard to compare against [2]. Experimental evidence shows that exposure to upward social comparison on social media profiles can lower self-esteem [3]. When users decide what to post, when to post and how heavily to edit it, the shared picture of daily life drifts away from what actually happened.

A second weakness is that participation on most platforms is sporadic and unstructured. Habit research shows that behaviour becomes automatic through repetition in a stable context, and in one longitudinal study the time needed to reach automaticity varied widely between people, with a median of 66 days [4]. Game design elements such as streaks and badges are widely used to sustain engagement [5], [6], and behaviour-design models argue that a behaviour occurs when motivation, ability and a prompt coincide [7]. These ideas are usually applied separately rather than inside one social product.

This paper presents **Sting – A Social Media App**, a mobile social network that combines these ideas. Every user receives one task per day and answers it through Unfiltered, a capture flow that accepts only live camera input, with no gallery upload, filter or editing. Each daily post extends a public streak. The main contributions are:

- 1) A daily-task engine that issues one shared prompt per calendar day and persists it on the server so that all users see the same task.
- 2) The Unfiltered capture flow: live camera only, a visible timestamp, and a lightweight accelerometer-based check that flags captures that look static.
- 3) A server-enforced streak mechanism with a one-counted-post-per-day rule, exposing current and longest streaks on public profiles.
- 4) A social layer consisting of profiles, a follow graph, interest-based suggestions, a feed with a trending fallback, and likes.
- 5) A modular Flutter, Express and PostgreSQL architecture with OTP verification, hashed passwords and token-based sessions, together with a transparent statement of the prototype’s limitations and an evaluation protocol.

The remainder of the paper is organized as follows. Section II reviews related work. Section III describes the methodology and architecture. Section IV details the implementation of each module. Section V summarizes the advantages of the approach. Section VI reports the prototype status, limitations, and evaluation protocol. Sections VII and VIII present the conclusion and future work.

II. LITERATURE SURVEY

Social Network Sites and Self-Presentation. Social network sites are commonly defined by public or semi-public profiles, an explicit list of connections, and the ability to traverse those lists [1]. Kietzmann et al. describe social media through seven functional building blocks: identity, conversations, sharing, presence, relationships, reputation and groups [8]. Sting keeps identity, relationships and sharing, but narrows sharing to a single live capture per day.

Social Comparison and Authenticity. Festinger's theory holds that people evaluate their own opinions and abilities by comparing themselves with others [2], and experimental work on social media profiles shows that upward comparison can reduce self-esteem [3]. Removing editing tools and filters limits how much a user can polish a post before sharing it. Whether this improves well-being is a hypothesis that Sting does not yet test.

Habit Formation and Behaviour Design. Lally et al. followed people forming a new daily behaviour and found that automaticity grew with repetition along an asymptotic curve, with large individual differences [4]. They also found that missing a single opportunity did not materially affect the process, which suggests that a hard streak reset may be harsher than the underlying habit mechanism requires. Fogg's behaviour model treats behaviour as the product of motivation, ability and a prompt [7]. In Sting, the daily task is the prompt, the camera-only flow keeps the required effort low, and the streak provides a visible reward for repetition.

Gamification and Streaks. Deterding et al. define gamification as the use of game design elements in non-game contexts [5]. A review of empirical studies by Hamari et al. found mostly positive effects but stressed that outcomes depend on context and on the users involved [6]. Streak counters are one such element and are common in learning and fitness applications. Sting makes the streak public on the profile, adding social visibility to a mechanism that is otherwise private, and it therefore has to consider the pressure that public streaks may create.

Live-Capture and Daily-Prompt Platforms. Some consumer applications prompt users once a day to capture a photo within a limited window, and others display counts of consecutive days of activity. We treat these as inspiration rather than as prior art for quantitative comparison. Sting's combination of a shared task, live-only capture and a public streak is a design choice whose effects are still to be measured.

Authenticity and Media Integrity. Detection of manipulated media has mostly focused on image content; FaceForensics++ provides a large benchmark for detecting manipulated facial images [9]. Such detectors analyse the pixels of an image and need substantial models and data. Sting instead looks at how the image was captured: only the camera is exposed in the interface, and an on-device motion signal indicates whether the phone was held naturally. These signals are cheap but weaker, and they complement rather than replace content-based forensics.

Web Architecture and Security. Sting's back end follows the REST architectural style [10], authenticates requests with signed JSON Web Tokens [11], stores passwords with the adaptive bcrypt scheme [12], and verifies contact details with short-lived numeric one-time codes. The HOTP specification [13] describes a standard HMAC-based variant, whereas Sting uses random codes stored with an expiry time. The OWASP Top 10 lists the most common web application risks, including broken access control and cryptographic failures, and serves as a checklist for the security review that the prototype still requires [14].

Gaps Identified. Existing work addresses social comparison, habit formation, gamification and media integrity separately. Little of it describes a single social application that combines a shared daily task, edit-free capture, a lightweight liveness signal and a public streak, or that states plainly which of those parts are validated. Sting is built to join these parts and to report their current limits.

III. PROPOSED METHODOLOGY

Sting follows a client-server design in which the mobile client handles capture and presentation, while the server owns identity, time-sensitive rules and persistence. This separation keeps rules such as the daily limit and the streak out of the reach of a modified client.

Requirements. The requirements were derived from the product concept: (i) account creation with a verified phone number or e-mail address; (ii) one shared task per calendar day; (iii) live-only capture with no gallery access, filters or editing; (iv) at most one counted post per day, with streak updates performed on the server; (v) profiles, a follow graph and a feed; (vi) public visibility of each user's streak; and (vii) basic security for credentials, sessions and uploads.

Workflow. A user registers with an e-mail address or phone number and a password, and the server issues a four-digit one-time code that is valid for ten minutes. After verification the account is created and a signed token is returned. The user then chooses a unique username, display name, bio and interest. The home screen shows the day's task, fetched from the server. The user opens the camera and captures a photo, and the client attaches the capture time and the result of the motion check. After an optional caption, the client uploads the photo. The server checks the daily limit, stores the post and updates the streak, and the post appears in the feed.

Data Model. The core entities are the user (credentials, profile fields, interest, privacy flag, current streak, longest streak and last post date), the one-time code, the follow relation, the daily task, the post (task label, caption, image location, live-capture flag and capture time) and the post like. Composite primary keys prevent duplicate follows and likes, and indexes on the follower, following and post-owner columns support feed and list queries.

Daily Task and Streak Logic. The server selects the day's task from a curated pool and stores it by date, so repeated requests on the same day return the same prompt. When a post arrives, the streak is computed from the user's last post date. With no previous post the streak becomes one. A previous post on the same calendar day causes the request to be rejected with a conflict response and leaves the streak unchanged. A previous post on the preceding day increases the streak by one, and any longer gap resets it to one. The longest streak is the maximum value reached. Because this computation runs on the server, editing the client cannot change a streak.

Unfiltered Capture and Liveness Heuristic. The capture screen exposes only the camera preview and a shutter button. While the preview is open, the client keeps a rolling buffer of the last 40 accelerometer readings, each taken as the sum of the absolute values of the three axes. At the moment of capture, the variance of this buffer is compared with a threshold of 0.02. Natural hand movement produces small continuous variation, whereas a phone resting on a table or fixed on a stand produces very little. If fewer than eight readings are available, the capture is not penalised. The review screen shows the capture time and a badge reading either live capture verified or low motion. The threshold was set empirically and has not been calibrated across devices.

Social Graph and Feed. Following and unfollowing are idempotent operations. Suggestions list users who share the viewer's interest and are not yet followed, ordered by streak. The feed returns recent posts from followed users, newest first and limited to 50 items. When there are none, it falls back to trending posts, defined as the most-liked posts of the previous two days.

Security by Design. Passwords are hashed with bcrypt before storage, and sessions use signed tokens with an expiry time. Protected routes pass through an authentication middleware. One-time codes expire after ten minutes and are marked as used after verification. The upload route accepts only image content types and limits file size to 12 MB. The password-reset request returns the same response whether or not the account exists, which avoids revealing which e-mail addresses are registered.

IV. SYSTEM IMPLEMENTATION

The methodology is realized as an Android application and a REST service, organized into the following modules.

Technology Stack. The client is written in Flutter [15] using the camera, sensors_plus, http and shared_preferences packages. The server runs on Node.js with the Express framework [16] and exchanges JSON over HTTP, and media uploads use multipart forms handled by multer. Data is stored in PostgreSQL [17] through the pg driver, and the bcryptjs and jsonwebtoken libraries provide hashing and tokens.

Authentication Module. Registration, one-time-code verification and login are separate endpoints, and users can log in with an e-mail address, phone number or username. The signed token is stored on the device so that reopening the app skips the welcome screen. In the prototype, one-time codes are written to the server log instead of being sent by SMS or e-mail.

Profile Module. As the user types, the client checks username availability against the server; usernames are 3–20 characters of lowercase letters, digits, dots and underscores. The profile stores display name, bio and interest, and these can be edited later. A privacy toggle updates a private-account flag on the server.

Social Graph Module. Dedicated endpoints handle follow, unfollow, followers, following and suggestions. In the client, follow buttons show a busy state and change only after the server confirms the update.

Daily Task Module. A task endpoint returns the date's prompt from a pool of twelve prompts and stores it so that it remains stable throughout the day. The feed banner shows the prompt, and the camera screen displays it as the capture title.

Unfiltered Capture Module. The camera button in the navigation bar opens the capture screen and is hidden once the user has posted that day. After the shutter is pressed, the review screen shows the photo, the capture time and the verification badge, with an optional caption. The upload explicitly declares an image content type so that the server can validate it.

Post, Streak and Feed Module. The post endpoint accepts the multipart upload, rejects a second post on the same day, stores the image in the server's upload folder, records the post and updates the streak counters. The feed endpoint joins posts with author information and like counts, and like and unlike endpoints maintain the like table.

Client State and Error Handling. A shared state object caches the profile, streak, following set and own posts, and refreshes them from the server after login and after posting. Each screen shows loading and error states, including a message when the server cannot be reached.

Integration and Testing. The modules were integrated and exercised manually on a physical Android phone connected to a development server running on a PC, with the phone reaching the server through a forwarded USB port. This testing exposed and fixed integration issues in database permissions and in the content type of uploaded images. No automated test suite or user study has been completed.

V. ADVANTAGES

- 1. Authentic Sharing.** Because the interface offers only live capture, a post reflects a real moment rather than a prepared one.
- 2. Habit-Building Structure.** A single daily prompt removes the decision of what to post and acts as a consistent cue, while the streak gives visible progress.
- 3. Social Accountability.** A streak shown on the public profile makes consistency visible to followers, which may encourage users to keep posting.
- 4. Tamper-Resistant Rules.** The daily limit and streak calculation run on the server, so a modified client cannot award itself extra streak days.
- 5. Low-Friction Verification.** The motion check runs on the device, needs no additional hardware and sends no image to third-party services.
- 6. Modularity and Extensibility.** The client, API and database are separate, so the client-side heuristic can later be supplemented by server-side checks without changing the user interface.
- 7. Cost-Effective Stack.** The system uses open-source components, and the single Flutter code base can later be extended to iOS.

VI. RESULTS AND ANALYSIS

This section reports what the current prototype demonstrates, what it does not, and how it should be evaluated. No user study or performance benchmark has been run, so we make no quantitative claims about engagement or detection accuracy.

A. Prototype Status. The status of each module is listed below. The core flow from registration through profile setup, task display and feed loading was exercised on a physical device; the remaining items are implemented but not yet fully tested.

Authentication and OTP: Implemented; codes logged, not delivered. Validation status: functionally exercised; SMS/e-mail delivery pending.

Profile and username check: Implemented. Validation status: functionally exercised.

Follow graph and suggestions: Implemented. Validation status: not yet tested with multiple users.

Daily task: Implemented (12-prompt pool). Validation status: functionally exercised.

Unfiltered capture and upload: Implemented. Validation status: under testing; upload path fixed during testing.

Liveness heuristic: Implemented (client-side). Validation status: threshold uncalibrated; no accuracy claim.

Streak engine: Implemented (server-side). Validation status: midnight and time-zone cases not yet unit-tested.

Feed and likes: Implemented. Validation status: feed loads; photos not yet rendered; likes not yet exercised.

B. Limitations. (i) One-time codes are only written to the server log, and the password-reset request issues a code but no endpoint yet redeems it. (ii) The live-capture flag is reported by the client and could be forged by calling the API directly; the accelerometer heuristic is simple, uncalibrated, and may misclassify captures made with a steady hand or a tripod. (iii) Uploaded images are stored on the server's local disk, and the feed currently shows a placeholder instead of the photo. (iv) The private-account flag is stored but not yet enforced in follow or feed logic, and the notification toggle is not connected to push notifications. (v) Day boundaries follow the server's date, so users in other time zones may see inconsistent day changes. (vi) There is no rate limiting, reporting, blocking or moderation. (vii) Other users' post history and the follower count on one's own profile are not yet shown.

C. Proposed Evaluation Protocol. Before public release, each component should be evaluated against a defined reference. Detection metrics such as precision, recall and F-measure [18] and ROC analysis [19] suit the liveness heuristic, usability can be measured with the System Usability Scale [20], and the security review can follow the OWASP Top 10 [14]. The proposed protocol is listed below, component by component.

Liveness heuristic: False-accept and false-reject rate, ROC. Reference standard: labeled live captures and screen or print re-captures across devices.

Streak and daily-task logic: Correct streak after each simulated day sequence. Reference standard: unit tests including midnight and time-zone cases.

Feed and discovery: Suggestion precision, feed latency. Reference standard: pilot-user relevance ratings and server logs.

Engagement: Day-1 and Day-7 retention, streak-length distribution. Reference standard: pilot cohort analytics.

Security: Findings by severity. Reference standard: review against the OWASP Top 10.

Usability: SUS score, task time, error count. Reference standard: representative users.

Performance: 95th-percentile API latency under load. Reference standard: load tests at target concurrency.

D. Discussion. The prototype shows that the proposed architecture can connect a shared daily task, edit-free capture, server-side streaks and a social graph in one working application. It does not show that these features change user behaviour. Two design questions need particular attention. First, public streaks may motivate but can also create pressure, and the habit literature suggests that a single missed day need not undo a habit [4], so recovery mechanisms such as a freeze that a friend can gift should be tested. Second, the motion heuristic should be treated as a deterrent against casual misuse, not as proof of authenticity.

VII. CONCLUSION

This paper presented Sting, a mobile social media application that asks every user to answer one shared task per day through Unfiltered, a live-only capture flow, and records consistent participation as a public streak. The system combines a Flutter client, an Express REST API and a PostgreSQL database, with OTP-based registration, hashed passwords, token sessions, and server-side enforcement of the daily limit and the streak logic.

The current prototype is a functional proof of concept rather than a validated product. One-time codes are not yet delivered, the authenticity flag is client-reported, and no user study has been conducted. We therefore described the limitations explicitly and proposed an evaluation protocol covering detection quality, rule correctness, usability, security and engagement.

VIII. FUTURE WORK

Server-Side Authenticity. Future versions should send raw sensor samples with each capture and verify them on the server, and should use platform device-integrity services to confirm that an upload came from the app's own camera flow.

Streak Recovery and Social Features. Planned additions include streak freezes that friends can gift, small groups with shared streaks, leaderboards and a year-end recap of each user's activity.

Notifications and Timing. Push notifications can announce the daily task, and the release time can be randomized so that the prompt stays spontaneous.

Media and Storage. Photos should move to cloud object storage and be displayed in the feed, and the back end should be deployed to a stable public host.

Moderation and Privacy. Private accounts should be enforced, and reporting, blocking, rate limiting, account deletion and moderation tools should be added before public launch.

Platform. Longer-term goals include an iOS build, a Play Store release, per-user analytics on streaks and task categories, and accessibility improvements.

IX. REFERENCES

- [1]. Boyd, d. m., & Ellison, N. B. (2007). Social network sites: definition, history, and scholarship. *Journal of Computer-Mediated Communication*, 13(1), 210–230. <https://doi.org/10.1111/j.1083-6101.2007.00393.x>
- [2]. Festinger, L. (1954). A theory of social comparison processes. *Human Relations*, 7(2), 117–140. <https://doi.org/10.1177/001872675400700202>
- [3]. Vogel, E. A., Rose, J. P., Roberts, L. R., & Eckles, K. (2014). Social comparison, social media, and self-esteem. *Psychology of Popular Media Culture*, 3(4), 206–222. <https://doi.org/10.1037/ppm0000047>
- [4]. Lally, P., van Jaarsveld, C. H. M., Potts, H. W. W., & Wardle, J. (2010). How are habits formed: modelling habit formation in the real world.

European Journal of Social Psychology, 40(6), 998–1009. <https://doi.org/10.1002/ejsp.674>

- [5]. Deterding, S., Dixon, D., Khaled, R., & Nacke, L. (2011). From game design elements to gamefulness: defining “gamification”. Proceedings of the 15th International Academic MindTrek Conference (pp. 9–15). ACM. <https://doi.org/10.1145/2181037.2181040>
- [6]. Hamari, J., Koivisto, J., & Sarsa, H. (2014). Does gamification work? A literature review of empirical studies on gamification. Proceedings of the 47th Hawaii International Conference on System Sciences (pp. 3025–3034). IEEE. <https://doi.org/10.1109/HICSS.2014.377>
- [7]. Fogg, B. J. (2009). A behavior model for persuasive design. Proceedings of the 4th International Conference on Persuasive Technology (Article 40). ACM. <https://doi.org/10.1145/1541948.1541999>
- [8]. Kietzmann, J. H., Hermkens, K., McCarthy, I. P., & Silvestre, B. S. (2011). Social media? Get serious! Understanding the functional building blocks of social media. Business Horizons, 54(3), 241–251. <https://doi.org/10.1016/j.bushor.2011.01.005>
- [9]. Rössler, A., Cozzolino, D., Verdoliva, L., Riess, C., Thies, J., & Nießner, M. (2019). FaceForensics++: learning to detect manipulated facial images. Proceedings of the IEEE/CVF International Conference on Computer Vision (pp. 1–11). <https://doi.org/10.1109/ICCV.2019.00009>
- [10]. Fielding, R. T. (2000). Architectural Styles and the Design of Network-based Software Architectures (Doctoral dissertation). University of California, Irvine.
- [11]. Jones, M., Bradley, J., & Sakimura, N. (2015). JSON Web Token (JWT). RFC 7519, Internet Engineering Task Force. <https://doi.org/10.17487/RFC7519>
- [12]. Provos, N., & Mazières, D. (1999). A future-adaptable password scheme. Proceedings of the USENIX Annual Technical Conference, FREENIX Track (pp. 81–91).
- [13]. M'Raihi, D., Bellare, M., Hoornaert, F., Naccache, D., & Ranen, O. (2005). HOTP: an HMAC-based one-time password algorithm. RFC 4226, Internet Engineering Task Force. <https://doi.org/10.17487/RFC4226>
- [14]. OWASP Foundation. (2021). OWASP Top 10:2021. <https://owasp.org/Top10/>
- [15]. Google. (n.d.). Flutter documentation. <https://docs.flutter.dev>
- [16]. OpenJS Foundation. (n.d.). Express – Node.js web application framework. <https://expressjs.com>
- [17]. PostgreSQL Global Development Group. (n.d.). PostgreSQL documentation. <https://www.postgresql.org/docs/>
- [18]. Sokolova, M., & Lapalme, G. (2009). A systematic analysis of performance measures for classification tasks. Information Processing & Management, 45(4), 427–437.
- [19]. Fawcett, T. (2006). An introduction to ROC analysis. Pattern Recognition Letters, 27(8), 861–874. <https://doi.org/10.1016/j.patrec.2005.10.010>
- [20]. Brooke, J. (1996). SUS: a quick and dirty usability scale. In P. W. Jordan et al. (Eds.), Usability Evaluation in Industry (pp. 189–194). Taylor & Francis.