IJCRT.ORG ISSN : 2320-2882

INTERNATIONAL JOURNAL OF CREATIVE RESEARCH THOUGHTS (IJCRT)

An International Open Access, Peer-reviewed, Refereed Journal

Design and Development of an API-Driven Virtual Try-On Web Application for Accessible Online Fashion Visualization

[1]Sarang Shrihari Acharya, [2] Sushil V Kulkarni

[1]MTech Student, [2] HOD of CSE & IT Department

Department of M.Tech. Computer Science and Information Technology,

M.B.E. Society's College of Engineering, Ambajogai, Maharashtra, India

Abstract: Online apparel shopping removes the tactile and fitting cues available in a physical store, creating uncertainty about appearance and contributing to avoidable purchase dissatisfaction. This paper presents the design and development of a lightweight, API-driven virtual try-on web application that allows a user to upload a front-facing photograph, choose a garment, submit both assets to an external image-processing service, and view the synthesized result in a browser. The contribution is not a new generative model; it is a deployable integration architecture for institutions and small retailers that cannot train or host graphics-processing-unit-intensive models. The prototype uses HTML and CSS for presentation, JavaScript for interaction, PHP for server-side validation and API orchestration, and MySQL for structured metadata. The paper documents requirements, architecture, data flow, interface logic, file-handling controls, threat considerations, failure recovery, and a reproducible evaluation framework. A qualitative comparison with model-hosting approaches shows the principal trade-off: API-based delivery reduces infrastructure and machine-learning engineering effort while increasing dependency on network quality, provider availability, cost, privacy terms, and version stability. The resulting system is best understood as a visualization aid rather than a guarantee of size, fit, fabric behavior, or purchase outcome. The study provides a practical reference architecture and identifies measurable criteria task completion, latency, failure rate, image plausibility, garment-detail preservation, perceived usefulness, and privacy compliance for future controlled evaluation.

I. INTRODUCTION

E-commerce has made clothing catalogues continuously accessible, but it has not removed a central limitation of remote apparel purchasing: the customer cannot directly observe how a garment may look on their own body.

Product photographs show garments on models or in isolation, whereas the buyer must mentally combine garment appearance, personal body shape, pose, and style. Virtual try-on (VTO) systems reduce part of this visualization gap by generating an image in which a selected garment is transferred to a photograph of a target person. The generated image can support exploration and confidence, although it cannot by itself reproduce tactile properties, exact size, drape under movement, or physical comfort.

Research systems such as VITON introduced a coarse-to-fine image-based pipeline; CP-VTON improved garment alignment and detail preservation; and later work, including VITON-HD and Dress Code, addressed higher resolution and broader garment categories [1]-[4]. These advances demonstrate the value of specialized generative models, but local deployment commonly requires curated datasets, trained weights, accelerator hardware, environment management, and continuing model maintenance. Such

requirements are difficult for a student prototype, a small retailer, or an institution whose primary expertise is web development.

This work therefore studies a different engineering question: how can a useful VTO experience be delivered through standard web technologies when image synthesis is supplied by a remote service? The proposed architecture separates the user-facing application from the computational model. The browser handles input and feedback; a PHP backend validates files, manages secrets, constructs API requests, interprets responses, and records limited metadata; the provider performs image synthesis. This separation converts a model-development problem into a systems-integration problem, but introduces its own concerns involving privacy, latency, reliability, cost, and provider lock-in.

The original five-page publication described the project at a high level. The present expanded manuscript adds the design rationale, formal requirements, end-to-end workflow, data model, secure file-handling strategy, testing framework, and a discussion of validity and responsible use. No unreported numerical experiment is invented. Where measured datasets are unavailable, the paper specifies tests and metrics that can be executed and reported in a subsequent empirical study.

2. Research Scope, Questions, and Contributions

The scope is a browser-based, image-to-image VTO prototype for still photographs. A user supplies one person image and selects one supported garment image. The system submits the assets to a configurable third-party VTO endpoint and displays a returned image or job result. Real-time augmented reality, three-dimensional body reconstruction, exact sizing, payments, inventory synchronization, and locally trained generative models are outside the present scope.

The work is organized around three research questions: (RQ1) What modular architecture permits an external VTO service to be integrated into a conventional web application? (RQ2) Which validation, privacy, security, and failure-handling controls are required for responsible image processing? (RQ3) How should an API-driven prototype be evaluated without conflating system usability with the visual quality of the external model?

The main contributions are:

- a layered reference architecture that isolates the browser, application controller, persistent metadata, temporary image storage, and remote inference provider;
- a provider-neutral request lifecycle that supports either synchronous generation or asynchronous job polling;
- a security and privacy checklist for personal-image upload, API-key protection, retention, access control, and deletion;
- a reproducible verification plan covering functionality, compatibility, performance, resilience, usability, accessibility, and visual-output quality; and
- an explicit analysis of API-based deployment trade-offs, limitations, and ethical boundaries.

3. Related Work

3.1 Image-Based Virtual Try-On

VITON framed two-dimensional try-on as conditional image synthesis without requiring a three-dimensional representation. Its clothing-agnostic person representation and coarse-to-fine strategy established a baseline for transferring an in-shop garment to a target person [1]. CP-VTON introduced a learnable geometric matching module based on thin-plate-spline transformation and a try-on module that

balances rendered content with the warped garment, with the goal of retaining textures, logos, and other garment characteristics [2].

VITON-HD addressed high-resolution synthesis at 1024 x 768 pixels and emphasized misalignment-aware normalization [3]. Dress Code expanded the problem beyond upper-body garments by introducing high-resolution, multi-category data for upper-body, lower-body, and full-body clothing [4]. Full-range VTO and diffusion-based methods further illustrate the field's movement toward more difficult poses, complex garment structures, and greater realism [5], [6]. These studies focus primarily on model architecture and output quality. The present work complements them by focusing on the web delivery layer and treating the generator as a replaceable external capability.

3.2 API-Centred Web Delivery

An API-centred design exposes image synthesis through a network contract. This pattern allows a client system to exchange an inference request for a completed image, a hosted result URL, or an asynchronous job identifier. It reduces local computation but makes robust integration essential: requests may time out, providers may impose file-size and rate limits, schemas may change, and failed jobs may still consume quota. Consequently, the web application must validate input before submission, use bounded timeouts and retries, distinguish transient from permanent errors, and avoid exposing provider credentials in JavaScript.

3.3 Gap Addressed

A review of VTO literature reveals abundant attention to synthesis quality but comparatively less detail about small-scale deployment using common server-side technologies. A practical paper must explain not only what an external model can generate, but also how user data crosses trust boundaries, how requests are tracked, how errors are communicated, and how quality is evaluated independently of the provider. This study addresses that systems-engineering gap.

4. Requirements Analysis

4.1 Stakeholders and Use Cases

The primary stakeholder is a shopper who wants a quick visual preview. Secondary stakeholders include catalogue administrators, application operators, privacy or security reviewers, and the API provider. The principal use case begins when the shopper uploads a suitable photograph, selects an available garment, grants informed processing consent, and requests generation. Alternative paths include invalid files, unsupported garments, provider rejection, timeout, cancellation, and deletion of submitted data.

4.2 Functional Requirements

ID	Requirement	Acceptance condition
FR-01	Accept JPG, JPEG, or PNG person images through a labelled upload control.	A valid image is previewed; unsupported types are rejected before API submission.
FR-02	Display a garment catalogue and permit one garment selection per request.	The selected item is visually identifiable and its identifier reaches the backend.
FR-03	Validate image type, size, decodability, and required inputs on the server.	Malformed or oversized input produces a clear, non-sensitive error.
FR-04	Submit validated assets to a configurable extemal VTO endpoint.	The backend creates a provider-compliant request without exposing the API key.
FR-05	Handle immediate responses or poll asynchronous jobs.	The interface reaches completed, failed, expired, or cancelled state.
FR-06	Present the generated image with retry. download, and delete controls.	A successful result is visible and its lifecycle can be controlled by the user.
FR-07	Record minimal operational metadata.	Request status and timestamps are retrievable without storing unnecessary personal data.

Table 1. Functional requirements and acceptance conditions.

4.3 Non-Functional Requirements

Usability requires visible system status, concise instructions for an appropriate front-facing image, reversible garment selection, and actionable errors. Performance should be assessed using end-to-end latency percentiles rather than a single average because network inference times are typically skewed. Reliability requires graceful behavior under timeouts, unavailable services, malformed JSON, incomplete images, and duplicate user actions. Maintainability requires an adapter around provider-specific fields so that an endpoint can be replaced without rewriting the user interface.

Security requires server-side MIME verification, randomized filenames, storage outside the public document root where feasible, strict upload limits, output encoding, parameterized database queries, secret management through environment configuration, transport encryption, and rate limiting. Privacy requires consent, a stated purpose, limited retention, deletion, access restriction, and disclosure that the image is transmitted to an external processor. Accessibility requires keyboard operation, programmatic labels, visible focus, sufficient contrast, descriptive status announcements, and meaningful alternative text for non-decorative images.

5. Proposed System Architecture

Layered Architecture of the API-Driven Prototype

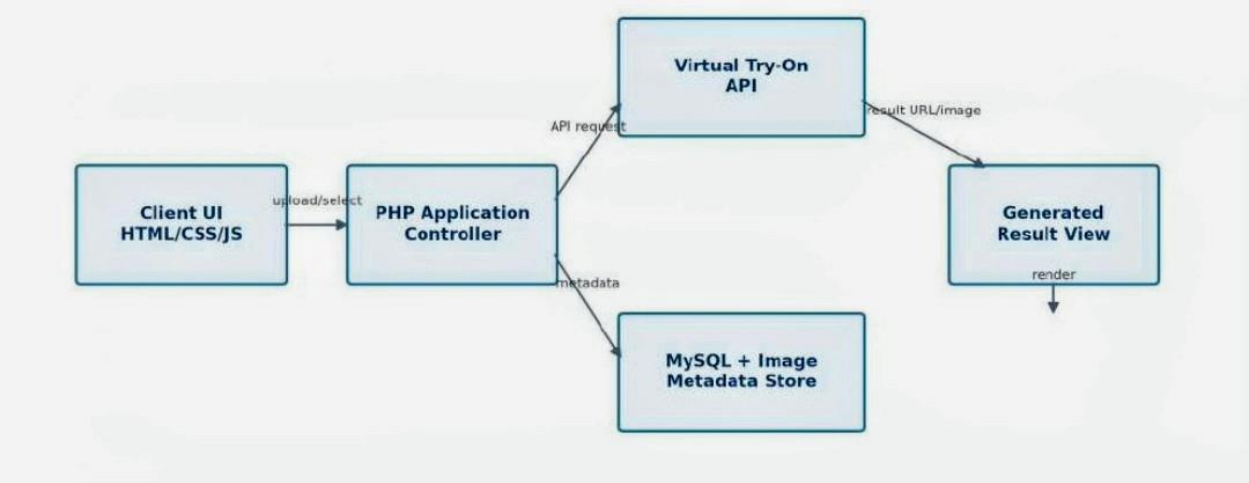

Figure 1. Layered architecture and principal data exchanges.

5.1 Presentation Layer

The presentation layer is implemented with semantic HTML, responsive CSS, and unobtrusive JavaScript. HTML provides labelled controls and page structure; CSS adapts the catalogue and preview to desktop and mobile widths; JavaScript performs immediate interaction such as local thumbnail preview, selection feedback, progress indication, and result rendering. Client-side checks improve responsiveness but are never treated as a security boundary because requests can bypass the browser interface.

5.2 Application and Integration Layer

PHP receives the multipart form request, validates the uploaded image, resolves the selected garment from a server-controlled catalogue, and invokes the provider adapter. The adapter translates an internal request object into the external API schema. It also normalizes provider responses into application states such as queued, processing, completed, failed, expired, or rate-limited. This normalization prevents provider-specific terminology from leaking into the rest of the application.

5.3 Data and Storage Layer

MySQL stores structured operational data rather than raw image bytes: a random request identifier, a non-sensitive session or user reference where applicable, garment identifier, provider job identifier, timestamps, status,error category, and deletion deadline. Images may be held in protected temporary storage or an object store using opaque names. The database should not contain local filesystem paths supplied by users, API keys, or verbose provider errors that could include sensitive information.

5.4 External Inference Layer

The remote provider receives the person and garment inputs, performs alignment and synthesis, and returns the generated result. Because implementation details vary, the application assumes only a documented contract. A synchronous provider returns a result within the initial request. An asynchronous provider returns a job identifier that must be polled with bounded frequency. The adapter records provider version information when available, since model upgrades can alter output quality even when application code is unchanged.

6. Detailed Workflow and Data Design

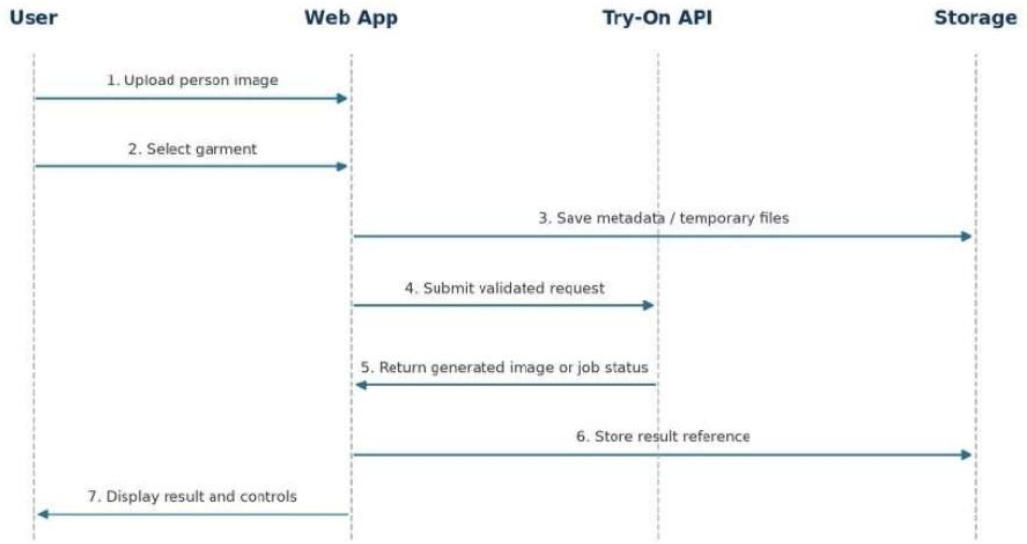

Figure 2. Request sequence from upload to result presentation.

6.1 Input Acquisition and Validation

The interface gives the user guidance before upload: use a well-lit, front-facing image; keep the torso visible; avoid severe occlusion; and do not upload another person's photograph without permission. On submission, the server checks that both files exist, the reported upload completed, the byte size is within policy, and the content can be decoded as an allowed image. The server derives the effective type from content rather than trusting the filename extension. If normalization is supported, the image is re-encoded to remove unexpected embedded content and to enforce a bounded resolution.

6.2 Request State Machine

A request progresses through VALIDATING, READY, SUBMITTED, PROCESSING, and COMPLETED. Terminal alternatives are REJECTED, FAILED, EXPIRED, and DELETED. State transitions should be monotonic: a completed or deleted request must not revert to processing. The backend assigns an idempotency key so repeated clicks or network retries do not unintentionally create multiple chargeable jobs. A short-lived server session maps the browser to the random request identifier.

6.3 Suggested Relational Schema

Entity	Key fields	Purpose
garments	garment id, category, asset uri, active	Server-approved catalogue and provider-compatible asset reference.
tryon_requests	request id, garment_id, status, created_at, expires_at	Lifecycle record for each attempt.
provider jobs	request id, provider_job_id, provider_version, last_poll_at	Isolation of external job details.
artifacts	artifact_id, request_id, kind, protected_uri, delete after	Controlled references to person, garment, and result assets.
events	event_id, request_id, event_type, occurred at, safe detail	Auditable state changes without storing image content in logs.

Table 2. Minimal logical data model.

6.4 Provider-Neutral Request Contract

Internally, the backend should represent a request with request_id, person_asset, garment_asset, garment_category, output_format, callback_or_polling_mode, and consent_timestamp. The provider adapter maps these fields to multipart upload, signed URLs, or base64 data according to the selected service. Responses are normalized to internal status, result_asset, retry_after, provider_job_id, and safe_error_code. Logging records only correlation identifiers, timestamps, durations, and categorized outcomes.

7. Implementation Design

7.1 Front-End Interaction

The page is divided into a person-image panel, garment catalogue, generation control, and result region. The Generate button remains disabled until the required inputs are valid. A local preview confirms the chosen person image without implying that generation has begun. When processing starts, the result region announces status visually and through an accessible live region. The user can cancel polling, try another garment, delete the request, or download a completed result where provider terms permit.

7.2 Backend Controller and API Adapter

The controller executes a deterministic sequence: authorize the request; validate CSRF protection where sessions are used; validate and normalize files; create a database transaction; stage protected assets; call the adapter with timeouts; store the provider job reference; and return a safe application response. API credentials are loaded only on the server. A provider error is mapped to a user-oriented category-invalid

input, quota exceeded, service busy, timeout, or internal failure while detailed diagnostics remain in protected operational logs.

7.3 Error Handling and Recovery

Retries are appropriate only for transient network errors, explicit rate limits, or server failures. They should use exponential backoff with jitter and a strict maximum. Validation failures and unsupported content must not be retried. If the provider accepts a request but the client disconnects, the job can continue independently and be recovered using the request identifier. Cleanup workers remove abandoned temporary files and expired artifacts even when the main request terminates unexpectedly.

7.4 Responsive and Accessible Interface

Responsive design changes the layout without changing task order. On a wide screen, person image, garment catalogue, and result can appear in adjacent regions; on a narrow screen, they appear sequentially. Touch targets should remain comfortably sized, colour should not be the only selection indicator, and loading animation must not conceal a text status. Input errors are associated with their controls, and the generated image uses alternative text that describes it as a synthetic preview.

8. Security, Privacy, and Ethical Considerations

8.1 Threat Model

The upload endpoint is exposed to untrusted content and may be abused for oversized files, polyglot files, path traversal, malicious metadata, denial of service, or attempts to execute stored content. The API boundary introduces credential theft, request tampering, replay, and leakage through logs or URLs. The result itself can create a different harm: users may interpret a plausible image as an accurate promise of fit. Security controls and user communication are therefore part of system correctness rather than optional enhancements.

Risk	Control	Verification
Disguised or malformed upload	Conment-based type detection, image decoding/re-encoding, byte and pixel limits.	Upload polyglot, truncated, oversized, and renamed non-image fixtures.
Public exposure of personal photos	Protected storage, opaque identifiers, authorization checks, short retention.	Attempt unauthenticated and cross-session retrieval: verity scheduled deletion.
API-key disclosure	Server-only secret storage; never embed keys in HTML or JavaScript.	Inspect built assets, responses, logs, and browser network traffic.
Injection or path manipulation	Allowlisted garment IDs, parameterized SQL, generated filenames, cutput encoding.	Run invalid identifiers and injection-oriented test cases.
Resource abuse and duplicate charging	Rate limits, idempotency keys, request quotas, bounded retries.	Repeat submissions and concurrent clicks; verify one logical job.
Misleading outpu	Clear "visualization only" notice and separation from	Usability test whether

	size recommendations.	participants understand the limitation.

Table 3. Principal risks, controls, and verification activities.

8.2 Privacy Lifecycle

Consent must be specific to image processing and external transmission. The privacy notice should identify the purpose, categories of data, provider role, retention period, deletion mechanism, and contact channel. Data minimization favours anonymous or short-lived sessions and avoids collecting demographic or account information unless needed. Result URLs should not be permanently public. Deletion should cover local person images, generated results, derived thumbnails, database references, caches, and provider-side artifacts to the extent supported by contract.

8.3 Responsible Interpretation

A synthesized image is an estimate of appearance under model assumptions. It may alter body shape, skin texture, garment colour, logos, patterns, boundaries, or occluded regions. The interface should not describe the output as an exact fit, and should not infer sensitive traits. Evaluation datasets should include varied poses, body shapes, skin tones, garment categories, and assistive-technology scenarios. Any disparity study must obtain appropriate consent and report subgroup sample sizes rather than making broad faimess claims from a small convenience sample.

9. Evaluation Methodology

A credible evaluation separates application engineering from provider image quality. The application can be verified with deterministic fixtures and a mock API; the integrated system can be measured against a live provider; visual quality requires human judgement or a paired dataset. The following protocol is designed for reproducibility and should accompany future measured results.

9.1 Functional and Compatibility Testing

Functional tests cover valid generation, each accepted image format, invalid content, missing garment, provider rejection, timeout, asynchronous completion, repeated submission, deletion, and expired results. Browser tests should include current stable desktop and mobile versions used by the target population. Each test records. environment, provider version, input fixture, expected state transition, observed state, and evidence. A test passes only if the final state, user message, database record, and cleanup behavior all agree.

9.2 Performance and Reliability Metrics

Metric	Definition	Reporting recommendation
Upload validation time	Server receipt to completion of local validation.	Median, P90, P95; stratify by file size.
Provider latency	Provider submission to completed result.	Median, P90, P95, maximum; report sample count and provider version.
End-to-end latency	Generate action to visible result.	Percentiles plus network and device context.
Completion rate	Completed requests/valid subraitted requests.	Overall and by garment category; give confidence interval.
Transient failure recovery	Retried transient failures that later complete/retryable failures.	Report tetry policy and added latency.
Cleanup compliance	Expired artifacts deleted within policy window /artifacts due.	Audit at multiple time points.
API cost per completion	Total provider charges/completed outputs.	State currency, tier, date, and excluded free quota

Table 4. Quantitative system metrics for a future benchmark.

9.3 Visual-Output Evaluation

Image quality should be assessed using a predefined rubric: garment identity preservation, logo or pattern fidelity, boundary quality, body and face preservation, pose consistency, occlusion handling, colour consistency, and overall plausibility. Two or more independent raters can score randomized outputs on a five-point scale after calibration. Inter-rater agreement should be reported. If paired ground truth is available, structural or perceptual metrics may supplement not replace human evaluation, because numerical similarity can penalize plausible alternatives or overlook semantically important garment errors.

9.4 Usability Study

A task-based study can ask participants to upload an image, select two garments, compare results, recover from one injected error, and delete their data. Measures include task completion, time on task, error count, assistance requests, a standardized usability score, and short questions about trust and limitations. Participants should be told that outputs are synthetic and should not use sensitive photographs. The report must state recruitment method, sample size, device distribution, prior online-shopping experience, and any ethics approval or consent procedure.

9.5 Acceptance Thresholds

Thresholds must be fixed before measurement to reduce post hoc interpretation. An example engineering gate is: all critical functional tests pass; no high-severity upload or authorization finding remains; valid jobs fail gracefully under provider outage; deletion completes within the stated retention window; and at least 95% of valid requests reach a clear terminal state. Latency and visual-quality thresholds should be chosen from user research and provider capabilities rather than inserted without evidence.

10. Results and Discussion

10.1 Demonstrated Prototype Capabilities

The documented prototype demonstrates the feasibility of combining HTML, CSS, JavaScript, PHP, MySQL, and an external image-processing API into a single VTO workflow. Its implemented feature set includes person-image upload, garment browsing and selection, remote generation, dynamic result display, responsive presentation, and image management. This confirms the central architectural proposition: a usable demonstration can be built without hosting or training the generative model locally.

The architectural analysis also shows that the lightweight claim applies to local compute, not to the entire system. Computation still occurs on provider infrastructure, and total performance depends on upload bandwidth, queueing, inference time, response transfer, and browser rendering. Similarly, "faster response" cannot be asserted universally without measurement against a defined baseline. The correct empirical conclusion is therefore limited to deployment feasibility and reduced local infrastructure requirements.

10.2 Comparative Analysis

Criterion	API-driven prototype	Self-hosted model
Initial infrastructure	Conventional web host plus provider account.	GPU-capable host, model weights, runtime, and storage.
Model expertise	Integration and prompt/input contract knowledge.	Training inference, dependency, optimization, and model monitoring expertise.
Control	Limited by provider schema, policy, and model version.	High control over weights, proprocessing. versioning, and retention.
Scaling	Provider absorbs inference scaling subject to quota and price.	Operator provisions capacity and manages queues.
Privacy	Image crosses an extemal trust boundary.	Can remain within operator-controlled Infrastructure,
Availability	Depends on provider and network.	Depends on operator infrastructure and model service.
Cost profile	Variable per request/subscription low entry	Higher setup and fixed compute cost; potentially

	cost.	lower marginal cost at scale.
Reproductibility	Provider changes may affect output unless versions are pinned.	Pinned code, weights, and environment can improve reproducibility.

Table 5. Engineering trade-offs between API-driven and self-hosted delivery.

10.3 Interpretation of Research Questions

RQ1 is answered by the layered architecture and adapter boundary: standard web components can deliver the interaction while the provider performs synthesis. RQ2 is answered by treating upload validation, secret isolation, protected storage, consent, retention, deletion, and honest output labelling as mandatory controls. RQ3 is answered by the evaluation decomposition: mock-backed application tests, live integration measurements, human visual ratings, and user-task evaluation measure different qualities and should not be collapsed into one success claim.

10.4 Practical Implications

For education and early retail pilots, the API approach shortens the path from concept to an interactive system and makes it possible to compare providers behind one interface. For production use, the same convenience creates governance work. Provider contracts, geographic processing location, retention, incident notification, service limits, model changes, and content policy become architecture inputs. A provider-neutral adapter and a small operational data model reduce migration cost but cannot eliminate commercial dependence.

11. Limitations and Threats to Validity

The principal limitation is the absence of a reported controlled benchmark containing measured latency, success rates, human quality scores, and usability data. The manuscript deliberately does not fabricate these values. Its results are therefore design-level and feasibility-oriented. A future submission making performance or user-impact claims should execute the protocol in Section 9 and attach anonymized test fixtures, configuration, provider version, and analysis code where licensing permits.

External validity is constrained by provider choice, garment categories, image guidance, region, network conditions, device capability, and participant population. Construct validity is threatened if "realism" is measured only by overall preference, since a visually pleasing result can still distort garment identity. Internal validity is threatened when provider updates occur during an experiment or when different garment categories receive different preprocessing. Reproducibility is threatened by closed APIs and unpinned model versions.

The application remains unsuitable for exact size recommendation unless combined with validated body measurement, garment dimensions, and a separately evaluated sizing model. It also does not reproduce fabric dynamics, movement, rear views, or physical comfort. Complex poses, hands covering the torso, loose garments, reflective fabrics, detailed text, layered outfits, and unsupported categories may produce artefacts. These limitations should be visible to users at decision time.

12. Future Work

The immediate next step is an empirical study using a version-pinned provider and a consented, diverse test set. The study should report system latency percentiles, completion and error rates, garment-category results, cost per successful generation, and a blinded visual-quality rubric. A usability study should test whether the interface improves decision confidence while maintaining correct understanding that the preview is not a guarantee of fit.

Technical extensions include automatic quality checks before upload, asynchronous queues and webhooks, multiple-provider routing, cached catalogue assets, content delivery for results, accessibility audits, and administrative dashboards for failure and cost monitoring. More ambitious work could add validated body-size estimation, garment recommendations, augmented-reality preview, multi-view or video output, and three-dimensional simulation. Each extension introduces additional privacy and validity requirements and should be evaluated independently.

Research extensions should compare API-based systems with open, self-hosted baselines under identical inpurs. Evaluation should examine not only photorealism but also preservation of garment identity, robustness to body diversity, calibration of user trust, and the effect of provider changes over time. Publishing a provider-neutral test harness would make these comparisons easier to reproduce.

13. Conclusion

This paper presented a detailed design for an API-driven virtual try-on web application built with common web technologies. The architecture separates interaction, orchestration, storage, and image synthesis so that computationally intensive generation can be consumed as an external service. This approach is accessible for prototypes and small deployments, but its success depends on disciplined validation, privacy controls, failure recovery, provider abstraction, and transparent communication of limitations.

The study's central contribution is a practical and evaluable systems design rather than a new generative model. By defining requirements, trust boundaries, request states, a minimal data schema, security controls, and separate evaluation tracks for application behavior, service performance, visual quality, and usability, the manuscript provides a foundation for a stronger empirical publication. The prototype supports visual exploration; it should not be presented as proof of physical fit. Future measured evaluation is necessary before claiming reduced return rates, faster performance than alternatives, or improved purchase outcomes.

Declarations

Funding: No external funding information was reported for this work.

Conflict of interest: The author declares no known conflict of interest. Any specific API provider used in an empirical deployment should be disclosed.

Data availability: No personal-image dataset is distributed with this manuscript. Future evaluation data should be shared only with appropriate consent and de-identification.

Ethics and consent: Users must provide informed consent before personal photographs are transmitted to an external processor. A human-participant study requires the institutionally appropriate review and consent process.

Author contributions: Acharya Sarang Shrihari-conceptualization, application development, investigation, and writing. Prof. Sushil V. Kulkami-supervision and guidance. These roles should be confirmed before submission.

References

[1] X. Han, Z. Wu, Z. Wu, R. Yu, and L. S. Davis, "VITON: An Image-Based Virtual Try-On Network," in Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, 2018, pp. 7543-7552. doi: 10.1109/CVPR.2018.00787.

[2] B. Wang, H. Zheng, X. Liang, Y. Chen, L. Lin, and M. Yang, "Toward Characteristic-Preserving Image-Based Virtual Try-On Network," in Proceedings of the European Conference on Computer Vision, 2018, pp. 589-604. doi: 10.1007/978-3-030-01246-5 36.

[3] S. Choi, S. Park, M. Lee, and J. Choo, "VITON-HD: High-Resolution Virtual Try-On via Misalignment-Aware Normalization," in Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition, 2021, pp. 14131-14140.

[4] D. Morelli, M. Fincato, M. Cornia, F. Landi, F. Cesari, and R. Cucchiara, "Dress Code: High-Resolution Multi-Category Virtual Try-On," in Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition Workshops, 2022, pp. 2231-2235.

[5] H. Yang, X. Yu, and Z. Liu, "Full-Range Virtual Try-On With Recurrent Tri-Level Transform," in Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition, 2022, pp. 3460-3469.

[6] L. Zhu et al., "TryOnDiffusion: A Tale of Two UNets," in Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition, 2023, pp. 4606-4615.

[7] OWASP Foundation, "File Upload Cheat Sheet," OWASP Cheat Sheet Series. Available:

https://cheatsheetseries.owasp.org/cheatsheets/File_Upload Cheat_Sheet.html (accessed Aug. 9, 2026).

[8] OWASP Foundation, "REST Security Cheat Sheet," OWASP Cheat Sheet Series. Available:

https://cheatsheetseries.owasp.org/cheatsheets/REST_Security_Cheat_Sheet.html (accessed Aug. 9, 2026).

[9] PHP Documentation Group, "Handling File Uploads," PHP Manual. Available: https://www.php.net/manual/en/features.file-upload.php (accessed Aug 9, 2026).

[10] Oracle, "MySQL 8.4 Reference Manual," 2026. Available: https://dev.mysql.com/doc/refman/8.4/en/ (accessed Aug. 9, 2026).

[11] World Wide Web Consortium, "Web Content Accessibility Guidelines (WCAG) 2.2," W3C Recommendation, Oct. 5, 2023. Available: https://www.w3.org/TR/WCAG22/.

[12] M. Brooke, "SUS: A 'Quick and Dirty Usability Scale," in Usability Evaluation in Industry, F. W. Jordan et al., Eds. London, UK: Taylor & Francis, 1996, pp. 189-194.

[13] J. Nielsen, Usability Engineering. San Francisco, CA, USA: Morgan Kaufmann, 1993.

[14] ISO, "ISO 9241-11:2018 Ergonomics of Human-System Interaction Part 11: Usability: Definitions and Concepts," International Organization for Standardization, 2018.

[15] NIST, "Digital Identity Guidelines: Authentication and Lifecycle Management," Special Publication 800-63B, 2017, updated.

2024. Available: https://pages.nist.gov/800-63-4/sp800-63b.html.

E-commerce has made clothing catalogues continuously accessible, but it has not removed a central limitation of remote apparel purchasing: the customer cannot directly observe how a garment may look on their own body.

Product photographs show garments on models or in isolation, whereas the buyer must mentally combine garment appearance, personal body shape, pose, and style. Virtual try-on (VTO) systems reduce part of this visualization gap by generating an image in which a selected garment is transferred to a photograph of a target person. The generated image can support exploration and confidence, although it cannot by itself reproduce tactile properties, exact size, drape under movement, or physical comfort.

Research systems such as VITON introduced a coarse-to-fine image-based pipeline; CP-VTON improved garment alignment and detail preservation; and later work, including VITON-HD and Dress Code, addressed higher resolution and broader garment categories [1]-[4]. These advances demonstrate the value of specialized generative models, but local deployment commonly requires curated datasets, trained weights, accelerator hardware, environment management, and continuing model maintenance. Such requirements are difficult for a student prototype, a small retailer, or an institution whose primary expertise is web development.

This work therefore studies a different engineering question: how can a useful VTO experience be delivered through standard web technologies when image synthesis is supplied by a remote service? The proposed architecture separates the user-facing application from the computational model. The browser handles input and feedback; a PHP backend validates files, manages secrets, constructs API requests, interprets responses, and records limited metadata; the provider performs image synthesis. This separation converts a model-development problem into a systems-integration problem, but introduces its own concerns involving privacy, latency, reliability, cost, and provider lock-in.

The original five-page publication described the project at a high level. The present expanded manuscript adds the design rationale, formal requirements, end-to-end workflow, data model, secure file-handling strategy, testing framework, and a discussion of validity and responsible use. No unreported numerical experiment is invented. Where measured datasets are unavailable, the paper specifies tests and metrics that can be executed and reported in a subsequent empirical study.

2. Research Scope, Questions, and Contributions

The scope is a browser-based, image-to-image VTO prototype for still photographs. A user supplies one person image and selects one supported garment image. The system submits the assets to a configurable third-party VTO endpoint and displays a returned image or job result. Real-time augmented reality, three-dimensional body reconstruction, exact sizing, payments, inventory synchronization, and locally trained generative models are outside the present scope.

The work is organized around three research questions: (RQ1) What modular architecture permits an external VTO service to be integrated into a conventional web application? (RQ2) Which validation, privacy, security, and failure-handling controls are required for responsible image processing? (RQ3) How should an API-driven prototype be evaluated without conflating system usability with the visual quality of the external model?

The main contributions are:

- a layered reference architecture that isolates the browser, application controller, persistent metadata, temporary image storage, and remote inference provider;

- a provider-neutral request lifecycle that supports either synchronous generation or asynchronous job polling;

- a security and privacy checklist for personal-image upload, API-key protection, retention, access control, and deletion;

- a reproducible verification plan covering functionality, compatibility, performance, resilience, usability, accessibility, and visual-output quality; and

- an explicit analysis of API-based deployment trade-offs, limitations, and ethical boundaries.

3. Related Work

3.1 Image-Based Virtual Try-On

VITON framed two-dimensional try-on as conditional image synthesis without requiring a three-dimensional representation. Its clothing-agnostic person representation and coarse-to-fine strategy established a baseline for transferring an in-shop garment to a target person [1]. CP-VTON introduced a learnable geometric matching module based on thin-plate-spline transformation and a try-on module that balances rendered content with the warped garment, with the goal of retaining textures, logos, and other garment characteristics [2].

VITON-HD addressed high-resolution synthesis at 1024 x 768 pixels and emphasized misalignment-aware normalization [3]. Dress Code expanded the problem beyond upper-body garments by introducing high-resolution, multi-category data for upper-body, lower-body, and full-body clothing [4]. Full-range VTO and diffusion-based methods further illustrate the field's movement toward more difficult poses, complex garment structures, and greater realism [5], [6]. These studies focus primarily on model architecture and output quality. The present work complements them by focusing on the web delivery layer and treating the generator as a replaceable external capability.

3.2 API-Centred Web Delivery

An API-centred design exposes image synthesis through a network contract. This pattern allows a client system to exchange an inference request for a completed image, a hosted result URL, or an asynchronous job identifier. It reduces local computation but makes robust integration essential: requests may time out, providers may impose file-size and rate limits, schemas may change, and failed jobs may still consume quota. Consequently, the web application must validate input before submission, use bounded timeouts and retries, distinguish transient from permanent errors, and avoid exposing provider credentials in JavaScript.

3.3 Gap Addressed

A review of VTO literature reveals abundant attention to synthesis quality but comparatively less detail about small-scale deployment using common server-side technologies. A practical paper must explain not only what an external model can generate, but also how user data crosses trust boundaries, how requests are tracked, how errors are communicated, and how quality is evaluated independently of the provider. This study addresses that systems-engineering gap.

4. Requirements Analysis

4.1 Stakeholders and Use Cases

The primary stakeholder is a shopper who wants a quick visual preview. Secondary stakeholders include catalogue administrators, application operators, privacy or security reviewers, and the API provider. The principal use case begins when the shopper uploads a suitable photograph, selects an available garment, grants informed processing consent, and requests generation. Alternative paths include invalid files, unsupported garments, provider rejection, timeout, cancellation, and deletion of submitted data.

4.2 Functional Requirements

ID	Requirement	Acceptance condition
FR-01	Accept JPG, JPEG, or PNG person images through a labelled upload control.	A valid image is previewed; unsupported types are rejected before API submission.
FR-02	Display a garment catalogue and permit one garment selection per request.	The selected item is visually identifiable and its identifier reaches the backend.
FR-03	Validate image type, size, decodability, and required inputs on the server.	Malformed or oversized input produces a clear, non-sensitive error.
FR-04	Submit validated assets to a configurable extemal VTO endpoint.	The backend creates a provider-compliant request without exposing the API key.
FR-05	Handle immediate responses or poll asynchronous jobs.	The interface reaches completed, failed, expired, or cancelled state.
FR-06	Present the generated image with retry. download, and delete controls.	A successful result is visible and its lifecycle can be controlled by the user.
FR-07	Record minimal operational metadata.	Request status and timestamps are retrievable without storing unnecessary personal data.

Table 1. Functional requirements and acceptance conditions.

4.3 Non-Functional Requirements

Usability requires visible system status, concise instructions for an appropriate front-facing image, reversible garment selection, and actionable errors. Performance should be assessed using end-to-end latency percentiles rather than a single average because network inference times are typically skewed. Reliability requires graceful behavior under timeouts, unavailable services, malformed JSON, incomplete images, and duplicate user actions. Maintainability requires an adapter around provider-specific fields so that an endpoint can be replaced without rewriting the user interface.

Security requires server-side MIME verification, randomized filenames, storage outside the public document root where feasible, strict upload limits, output encoding, parameterized database queries, secret management through environment configuration, transport encryption, and rate limiting. Privacy requires consent, a stated purpose, limited retention, deletion, access restriction, and disclosure that the image is transmitted to an external processor. Accessibility requires keyboard operation, programmatic labels, visible focus, sufficient contrast, descriptive status announcements, and meaningful alternative text for non-decorative images.

5. Proposed System Architecture

Layered Architecture of the API-Driven Prototype

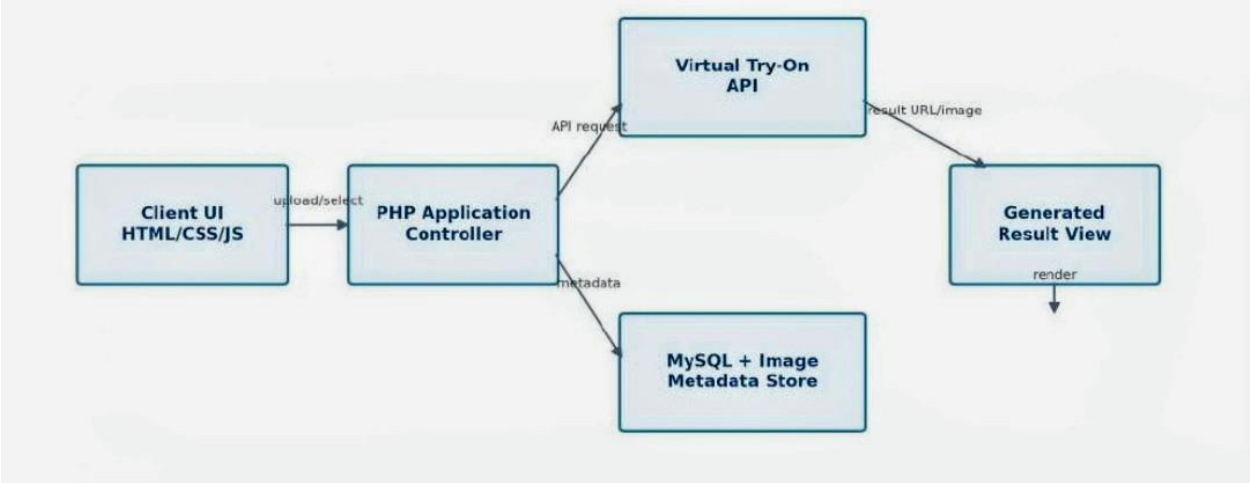

Figure 1. Layered architecture and principal data exchanges.

5.1 Presentation Layer

The presentation layer is implemented with semantic HTML, responsive CSS, and unobtrusive JavaScript. HTML provides labelled controls and page structure; CSS adapts the catalogue and preview to desktop and mobile widths; JavaScript performs immediate interaction such as local thumbnail preview, selection feedback, progress indication, and result rendering. Client-side checks improve responsiveness but are never treated as a security boundary because requests can bypass the browser interface.

5.2 Application and Integration Layer

PHP receives the multipart form request, validates the uploaded image, resolves the selected garment from a server-controlled catalogue, and invokes the provider adapter. The adapter translates an internal request object into the external API schema. It also normalizes provider responses into application states such as queued, processing, completed, failed, expired, or rate-limited. This normalization prevents provider-specific terminology from leaking into the rest of the application.

5.3 Data and Storage Layer

MySQL stores structured operational data rather than raw image bytes: a random request identifier, a non-sensitive session or user reference where applicable, garment identifier, provider job identifier, timestamps, status,error category, and deletion deadline. Images may be held in protected temporary storage or an object store using opaque names. The database should not contain local filesystem paths supplied by users, API keys, or verbose provider errors that could include sensitive information.

5.4 External Inference Layer

The remote provider receives the person and garment inputs, performs alignment and synthesis, and returns the generated result. Because implementation details vary, the application assumes only a documented contract. A synchronous provider returns a result within the initial request. An asynchronous provider returns a job identifier that must be polled with bounded frequency. The adapter records provider version information when available, since model upgrades can alter output quality even when application code is unchanged.

6. Detailed Workflow and Data Design

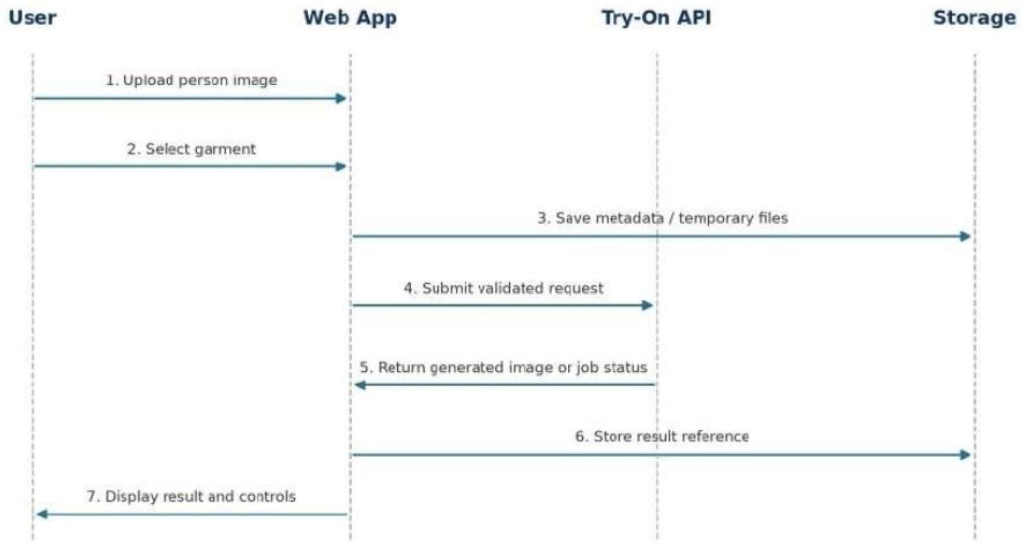

Figure 2. Request sequence from upload to result presentation.

6.1 Input Acquisition and Validation

The interface gives the user guidance before upload: use a well-lit, front-facing image; keep the torso visible; avoid severe occlusion; and do not upload another person's photograph without permission. On submission, the server checks that both files exist, the reported upload completed, the byte size is within policy, and the content can be decoded as an allowed image. The server derives the effective type from content rather than trusting the filename extension. If normalization is supported, the image is re-encoded to remove unexpected embedded content and to enforce a bounded resolution.

6.2 Request State Machine

A request progresses through VALIDATING, READY, SUBMITTED, PROCESSING, and COMPLETED. Terminal alternatives are REJECTED, FAILED, EXPIRED, and DELETED. State transitions should be monotonic: a completed or deleted request must not revert to processing. The backend assigns an idempotency key so repeated clicks or network retries do not unintentionally create multiple chargeable jobs. A short-lived server session maps the browser to the random request identifier.

6.3 Suggested Relational Schema

Entity	Key fields	Purpose
garments	garment id, category, asset uri, active	Server-approved catalogue and provider-compatible asset reference.
tryon_requests	request id, garment_id, status, created_at, expires_at	Lifecycle record for each attempt.
provider jobs	request id, provider_job_id, provider_version, last_poll_at	Isolation of external job details.

artifacts	artifact_id, request_id, kind, protected_uri, delete after	Controlled references to person, garment, and result assets.
events	event_id, request_id, event_type, occurred at, safe detail	Auditable state changes without storing image content in logs.

Table 2. Minimal logical data model.

6.4 Provider-Neutral Request Contract

Internally, the backend should represent a request with request_id, person_asset, garment_asset, garment_category, output_format, callback_or_polling_mode, and consent_timestamp. The provider adapter maps these fields to multipart upload, signed URLs, or base64 data according to the selected service. Responses are normalized to internal status, result_asset, retry_after, provider_job_id, and safe_error_code. Logging records only correlation identifiers, timestamps, durations, and categorized outcomes.

7. Implementation Design

7.1 Front-End Interaction

The page is divided into a person-image panel, garment catalogue, generation control, and result region. The Generate button remains disabled until the required inputs are valid. A local preview confirms the chosen person image without implying that generation has begun. When processing starts, the result region announces status visually and through an accessible live region. The user can cancel polling, try another garment, delete the request, or download a completed result where provider terms permit.

7.2 Backend Controller and API Adapter

The controller executes a deterministic sequence: authorize the request; validate CSRF protection where sessions are used; validate and normalize files; create a database transaction; stage protected assets; call the adapter with timeouts; store the provider job reference; and return a safe application response. API credentials are loaded only on the server. A provider error is mapped to a user-oriented category-invalid input, quota exceeded, service busy, timeout, or internal failure while detailed diagnostics remain in protected operational logs.

7.3 Error Handling and Recovery

Retries are appropriate only for transient network errors, explicit rate limits, or server failures. They should use exponential backoff with jitter and a strict maximum. Validation failures and unsupported content must not be retried. If the provider accepts a request but the client disconnects, the job can continue independently and be recovered using the request identifier. Cleanup workers remove abandoned temporary files and expired artifacts even when the main request terminates unexpectedly.

7.4 Responsive and Accessible Interface

Responsive design changes the layout without changing task order. On a wide screen, person image, garment catalogue, and result can appear in adjacent regions; on a narrow screen, they appear sequentially. Touch targets should remain comfortably sized, colour should not be the only selection indicator, and loading animation must not conceal a text status. Input errors are associated with their controls, and the generated image uses alternative text that describes it as a synthetic preview.

8. Security, Privacy, and Ethical Considerations

8.1 Threat Model

The upload endpoint is exposed to untrusted content and may be abused for oversized files, polyglot files, path traversal, malicious metadata, denial of service, or attempts to execute stored content. The API boundary introduces credential theft, request tampering, replay, and leakage through logs or URLs. The result itself can create a different harm: users may interpret a plausible image as an accurate promise of fit. Security controls and user communication are therefore part of system correctness rather than optional enhancements.

Risk	Control	Verification
Disguised or malformed upload	Conment-based type detection, image decoding/re-encoding, byte and pixel limits.	Upload polyglot, truncated, oversized, and renamed non-image fixtures.
Public exposure of personal photos	Protected storage, opaque identifiers, authorization checks, short retention.	Attempt unauthenticated and cross-session retrieval: verity scheduled deletion.
API-key disclosure	Server-only secret storage; never embed keys in HTML or JavaScript.	Inspect built assets, responses, logs, and browser network traffic.
Injection or path manipulation	Allowlisted garment IDs, parameterized SQL, generated filenames, cutput encoding.	Run invalid identifiers and injection-oriented test cases.
Resource abuse and duplicate charging	Rate limits, idempotency keys, request quotas, bounded retries.	Repeat submissions and concurrent clicks; verify one logical job.
Misleading outpu	Clear "visualization only" notice and separation from size recommendations.	Usability test whether participants understand the limitation.

Table 3. Principal risks, controls, and verification activities.

8.2 Privacy Lifecycle

Consent must be specific to image processing and external transmission. The privacy notice should identify the purpose, categories of data, provider role, retention period, deletion mechanism, and contact channel. Data minimization favours anonymous or short-lived sessions and avoids collecting demographic or account information unless needed. Result URLs should not be permanently public. Deletion should cover local person images, generated results, derived thumbnails, database references, caches, and provider-side artifacts to the extent supported by contract.

8.3 Responsible Interpretation

A synthesized image is an estimate of appearance under model assumptions. It may alter body shape, skin texture, garment colour, logos, patterns, boundaries, or occluded regions. The interface should not describe the output as an exact fit, and should not infer sensitive traits. Evaluation datasets should include varied poses, body shapes, skin tones, garment categories, and assistive-technology scenarios. Any disparity study must obtain appropriate consent and report subgroup sample sizes rather than making broad faimess claims from a small convenience sample.

9. Evaluation Methodology

A credible evaluation separates application engineering from provider image quality. The application can be verified with deterministic fixtures and a mock API; the integrated system can be measured against a live provider; visual quality requires human judgement or a paired dataset. The following protocol is designed for reproducibility and should accompany future measured results.

9.1 Functional and Compatibility Testing

Functional tests cover valid generation, each accepted image format, invalid content, missing garment, provider rejection, timeout, asynchronous completion, repeated submission, deletion, and expired results. Browser tests should include current stable desktop and mobile versions used by the target population. Each test records. environment, provider version, input fixture, expected state transition, observed state, and evidence. A test passes only if the final state, user message, database record, and cleanup behavior all agree.

9.2 Performance and Reliability Metrics

Metric	Definition	Reporting recommendation
Upload validation time	Server receipt to completion of local validation.	Median, P90, P95; stratify by file size.
Provider latency	Provider submission to completed result.	Median, P90, P95, maximum; report sample count and provider version.
End-to-end latency	Generate action to visible result.	Percentiles plus network and device context.
Completion rate	Completed requests/valid subraitted requests.	Overall and by garment category; give confidence interval.
Transient failure recovery	Retried transient failures that later complete/retryable failures.	Report tetry policy and added latency.
Cleanup compliance	Expired artifacts deleted within policy window /artifacts due.	Audit at multiple time points.
API cost per completion	Total provider charges/completed outputs.	State currency, tier, date, and excluded free quota

Table 4. Quantitative system metrics for a future benchmark.

9.3 Visual-Output Evaluation

Image quality should be assessed using a predefined rubric: garment identity preservation, logo or pattern fidelity, boundary quality, body and face preservation, pose consistency, occlusion handling, colour consistency, and overall plausibility. Two or more independent raters can score randomized outputs on a five-point scale after calibration. Inter-rater agreement should be reported. If paired ground truth is available, structural or perceptual metrics may supplement not replace human evaluation, because numerical similarity can penalize plausible alternatives or overlook semantically important garment errors.

9.4 Usability Study

A task-based study can ask participants to upload an image, select two garments, compare results, recover from one injected error, and delete their data. Measures include task completion, time on task, error count, assistance requests, a standardized usability score, and short questions about trust and limitations. Participants should be told that outputs are synthetic and should not use sensitive photographs. The report must state recruitment method, sample size, device distribution, prior online-shopping experience, and any ethics approval or consent procedure.

9.5 Acceptance Thresholds

Thresholds must be fixed before measurement to reduce post hoc interpretation. An example engineering gate is: all critical functional tests pass; no high-severity upload or authorization finding remains; valid jobs fail gracefully under provider outage; deletion completes within the stated retention window; and at least 95% of valid requests reach a clear terminal state. Latency and visual-quality thresholds should be chosen from user research and provider capabilities rather than inserted without evidence.

10. Results and Discussion

10.1 Demonstrated Prototype Capabilities

The documented prototype demonstrates the feasibility of combining HTML, CSS, JavaScript, PHP, MySQL, and an external image-processing API into a single VTO workflow. Its implemented feature set includes person-image upload, garment browsing and selection, remote generation, dynamic result display, responsive presentation, and image management. This confirms the central architectural proposition: a usable demonstration can be built without hosting or training the generative model locally.

The architectural analysis also shows that the lightweight claim applies to local compute, not to the entire system. Computation still occurs on provider infrastructure, and total performance depends on upload bandwidth, queueing, inference time, response transfer, and browser rendering. Similarly, "faster response" cannot be asserted universally without measurement against a defined baseline. The correct empirical conclusion is therefore limited to deployment feasibility and reduced local infrastructure requirements.

10.2 Comparative Analysis

Criterion	API-driven prototype	Self-hosted model
Initial infrastructure	Conventional web host plus provider account.	GPU-capable host, model weights, runtime, and storage.
Model expertise	Integration and prompt/input contract knowledge.	Training inference, dependency, optimization, and model monitoring expertise.
Control	Limited by provider schema, policy, and model version.	High control over weights, proprocessing. versioning, and retention.
Scaling	Provider absorbs inference scaling subject to quota and price.	Operator provisions capacity and manages queues.
Privacy	Image crosses an extemal trust boundary.	Can remain within operator-controlled Infrastructure,
Availability	Depends on provider and network.	Depends on operator infrastructure and model service.
Cost profile	Variable per request/subscription low entry cost.	Higher setup and fixed compute cost; potentially lower marginal cost at scale.
Reproductibility	Provider changes may affect output unless versions are pinned.	Pinned code, weights, and environment can improve reproducibility.

Table 5. Engineering trade-offs between API-driven and self-hosted delivery.

10.3 Interpretation of Research Questions

RQ1 is answered by the layered architecture and adapter boundary: standard web components can deliver the interaction while the provider performs synthesis. RQ2 is answered by treating upload validation, secret isolation, protected storage, consent, retention, deletion, and honest output labelling as mandatory controls. RQ3 is answered by the evaluation decomposition: mock-backed application tests, live integration measurements, human visual ratings, and user-task evaluation measure different qualities and should not be collapsed into one success claim.

10.4 Practical Implications

For education and early retail pilots, the API approach shortens the path from concept to an interactive system and makes it possible to compare providers behind one interface. For production use, the same convenience creates governance work. Provider contracts, geographic processing location, retention, incident notification, service limits, model changes, and content policy become architecture inputs. A provider-neutral adapter and a small operational data model reduce migration cost but cannot eliminate commercial dependence.

11. Limitations and Threats to Validity

The principal limitation is the absence of a reported controlled benchmark containing measured latency, success rates, human quality scores, and usability data. The manuscript deliberately does not fabricate these values. Its results are therefore design-level and feasibility-oriented. A future submission making performance or user-impact claims should execute the protocol in Section 9 and attach anonymized test fixtures, configuration, provider version, and analysis code where licensing permits.

External validity is constrained by provider choice, garment categories, image guidance, region, network conditions, device capability, and participant population. Construct validity is threatened if "realism" is measured only by overall preference, since a visually pleasing result can still distort garment identity. Internal validity is threatened when provider updates occur during an experiment or when different garment categories receive different preprocessing. Reproducibility is threatened by closed APIs and unpinned model versions.

The application remains unsuitable for exact size recommendation unless combined with validated body measurement, garment dimensions, and a separately evaluated sizing model. It also does not reproduce fabric dynamics, movement, rear views, or physical comfort. Complex poses, hands covering the torso, loose garments, reflective fabrics, detailed text, layered outfits, and unsupported categories may produce artefacts. These limitations should be visible to users at decision time.

12. Future Work

The immediate next step is an empirical study using a version-pinned provider and a consented, diverse test set. The study should report system latency percentiles, completion and error rates, garment-category results, cost per successful generation, and a blinded visual-quality rubric. A usability study should test whether the interface improves decision confidence while maintaining correct understanding that the preview is not a guarantee of fit.

Technical extensions include automatic quality checks before upload, asynchronous queues and webhooks, multiple-provider routing, cached catalogue assets, content delivery for results, accessibility audits, and administrative dashboards for failure and cost monitoring. More ambitious work could add validated body-size estimation, garment recommendations, augmented-reality preview, multi-view or video output, and three-dimensional simulation. Each extension introduces additional privacy and validity requirements and should be evaluated independently.

Research extensions should compare API-based systems with open, self-hosted baselines under identical inpurs. Evaluation should examine not only photorealism but also preservation of garment identity, robustness to body diversity, calibration of user trust, and the effect of provider changes over time. Publishing a provider-neutral test harness would make these comparisons easier to reproduce.

13. Conclusion

This paper presented a detailed design for an API-driven virtual try-on web application built with common web technologies. The architecture separates interaction, orchestration, storage, and image synthesis so that computationally intensive generation can be consumed as an external service. This approach is accessible for prototypes and small deployments, but its success depends on disciplined validation, privacy controls, failure recovery, provider abstraction, and transparent communication of limitations.

The study's central contribution is a practical and evaluable systems design rather than a new generative model. By defining requirements, trust boundaries, request states, a minimal data schema, security controls, and separate evaluation tracks for application behavior, service performance, visual quality, and usability, the manuscript provides a foundation for a stronger empirical publication. The prototype supports visual exploration; it should not be presented as proof of physical fit. Future measured evaluation

is necessary before claiming reduced return rates, faster performance than alternatives, or improved purchase outcomes.

Declarations

Funding: No external funding information was reported for this work.

Conflict of interest: The author declares no known conflict of interest. Any specific API provider used in an empirical deployment should be disclosed.

Data availability: No personal-image dataset is distributed with this manuscript. Future evaluation data should be shared only with appropriate consent and de-identification.

Ethics and consent: Users must provide informed consent before personal photographs are transmitted to an external processor. A human-participant study requires the institutionally appropriate review and consent process.

Author contributions: Acharya Sarang Shrihari-conceptualization, application development, investigation, and writing. Prof. Sushil V. Kulkami-supervision and guidance. These roles should be confirmed before submission.

References

[1] X. Han, Z. Wu, Z. Wu, R. Yu, and L. S. Davis, "VITON: An Image-Based Virtual Try-On Network," in Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, 2018, pp. 7543-7552. doi: 10.1109/CVPR.2018.00787.

[2] B. Wang, H. Zheng, X. Liang, Y. Chen, L. Lin, and M. Yang, "Toward Characteristic-Preserving Image-Based Virtual Try-On Network," in Proceedings of the European Conference on Computer Vision, 2018, pp. 589-604. doi: 10.1007/978-3-030-01246-5 36.

[3] S. Choi, S. Park, M. Lee, and J. Choo, "VITON-HD: High-Resolution Virtual Try-On via Misalignment-Aware Normalization," in Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition, 2021, pp. 14131-14140.

[4] D. Morelli, M. Fincato, M. Cornia, F. Landi, F. Cesari, and R. Cucchiara, "Dress Code: High-Resolution Multi-Category Virtual Try-On," in Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition Workshops, 2022, pp. 2231-

2235.

[5] H. Yang, X. Yu, and Z. Liu, "Full-Range Virtual Try-On With Recurrent Tri-Level Transform," in Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition, 2022, pp. 3460-3469.

[6] L. Zhu et al., "TryOnDiffusion: A Tale of Two UNets," in Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition, 2023, pp. 4606-4615.

[7] OWASP Foundation, "File Upload Cheat Sheet," OWASP Cheat Sheet Series. Available:

https://cheatsheetseries.owasp.org/cheatsheets/File_Upload Cheat_Sheet.html (accessed Aug. 9, 2026).

[8] OWASP Foundation, "REST Security Cheat Sheet," OWASP Cheat Sheet Series. Available:

https://cheatsheetseries.owasp.org/cheatsheets/REST_Security_Cheat_Sheet.html (accessed Aug. 9, 2026).

[9] PHP Documentation Group, "Handling File Uploads," PHP Manual. Available: https://www.php.net/manual/en/features.file-upload.php (accessed Aug 9, 2026).

[10] Oracle, "MySQL 8.4 Reference Manual," 2026. Available: https://dev.mysql.com/doc/refman/8.4/en/ (accessed Aug. 9, 2026).

[11] World Wide Web Consortium, "Web Content Accessibility Guidelines (WCAG) 2.2," W3C Recommendation, Oct. 5, 2023. Available: https://www.w3.org/TR/WCAG22/.

[12] M. Brooke, "SUS: A 'Quick and Dirty Usability Scale," in Usability Evaluation in Industry, F. W. Jordan et al., Eds. London, UK: Taylor & Francis, 1996, pp. 189-194.

[13] J. Nielsen, Usability Engineering. San Francisco, CA, USA: Morgan Kaufmann, 1993.

[14] ISO, "ISO 9241-11:2018 Ergonomics of Human-System Interaction Part 11: Usability: Definitions and Concepts," International Organization for Standardization, 2018.

[15] NIST, "Digital Identity Guidelines: Authentication and Lifecycle Management," Special Publication 800-63B, 2017, updated.

2024. Available: https://pages.nist.gov/800-63-4/sp800-63b.html.