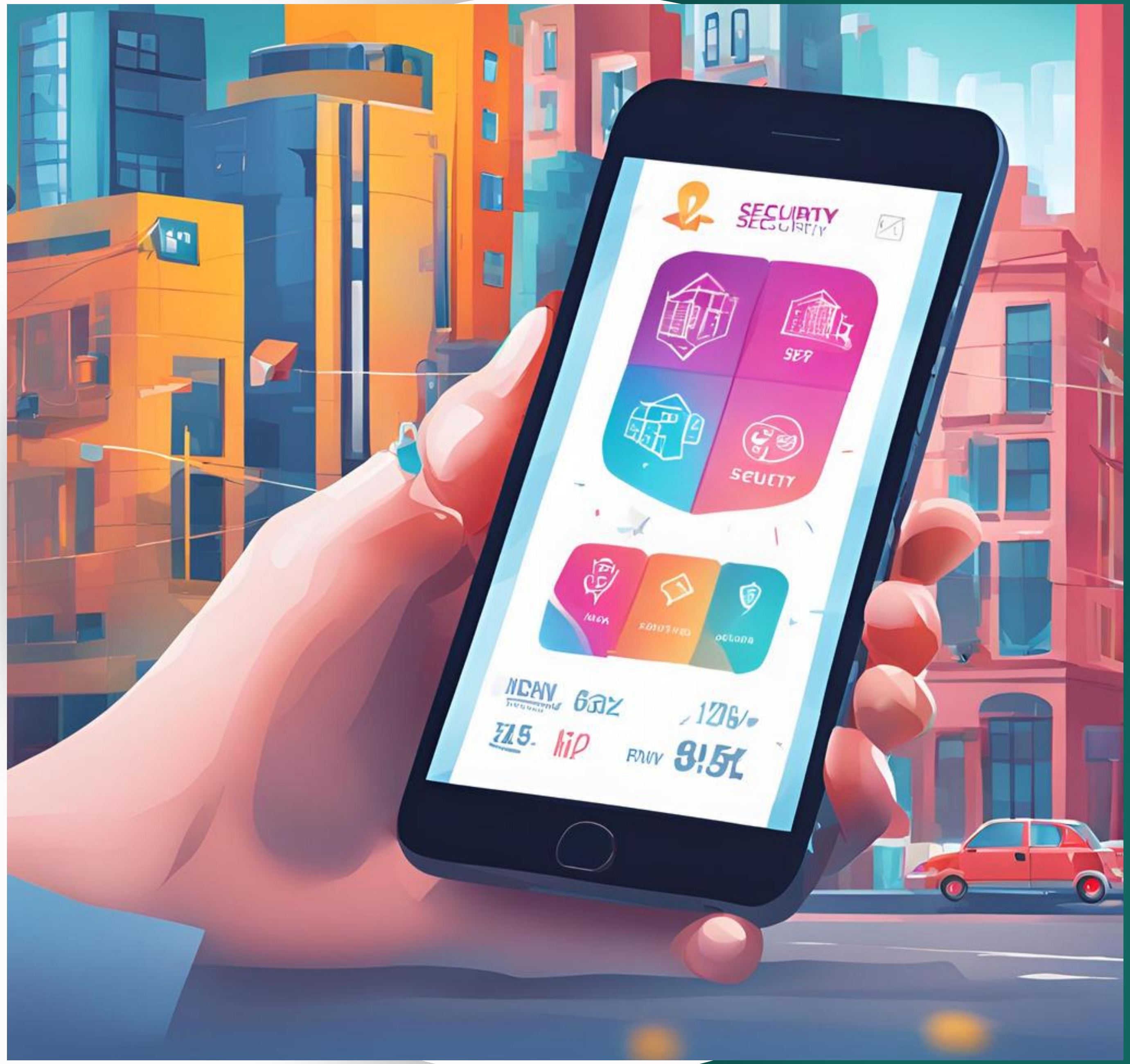
SecureHer

Layered Architecture Overview

Group No. 02

Team Members:
2005009 – Ahmmad Nur Swapnil
2005010 – Shovon Roy
2005011 – Tanvirul Islam Turad
2005014 – Tanvir Hossain
2005022 – Ekramul Haque Amin
2005025 – Sonia Khatun



presented by
Shovon Roy (205010)

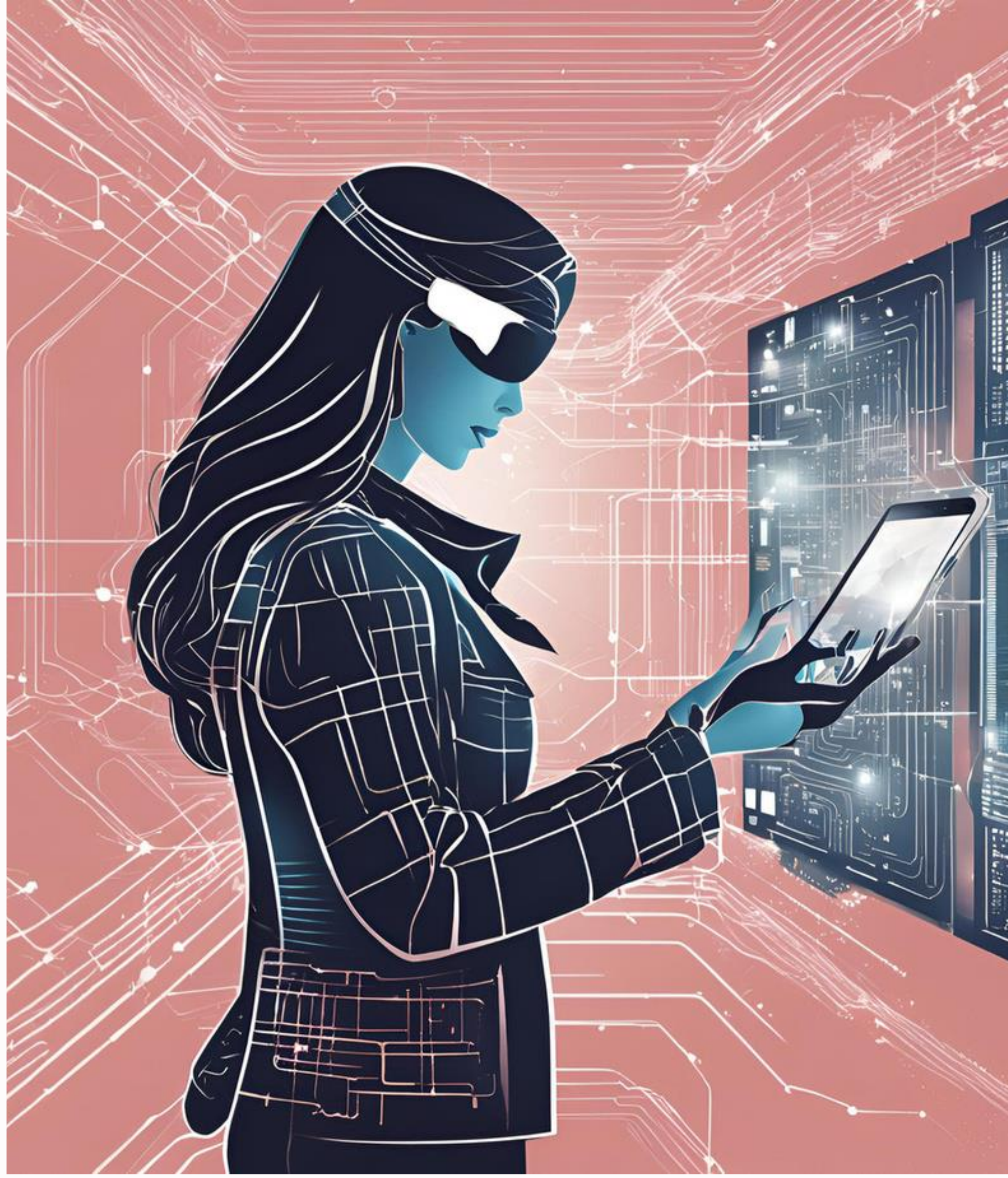
INTRODUCTION TO LAYERED ARCHITECTURE

Speaking Points:

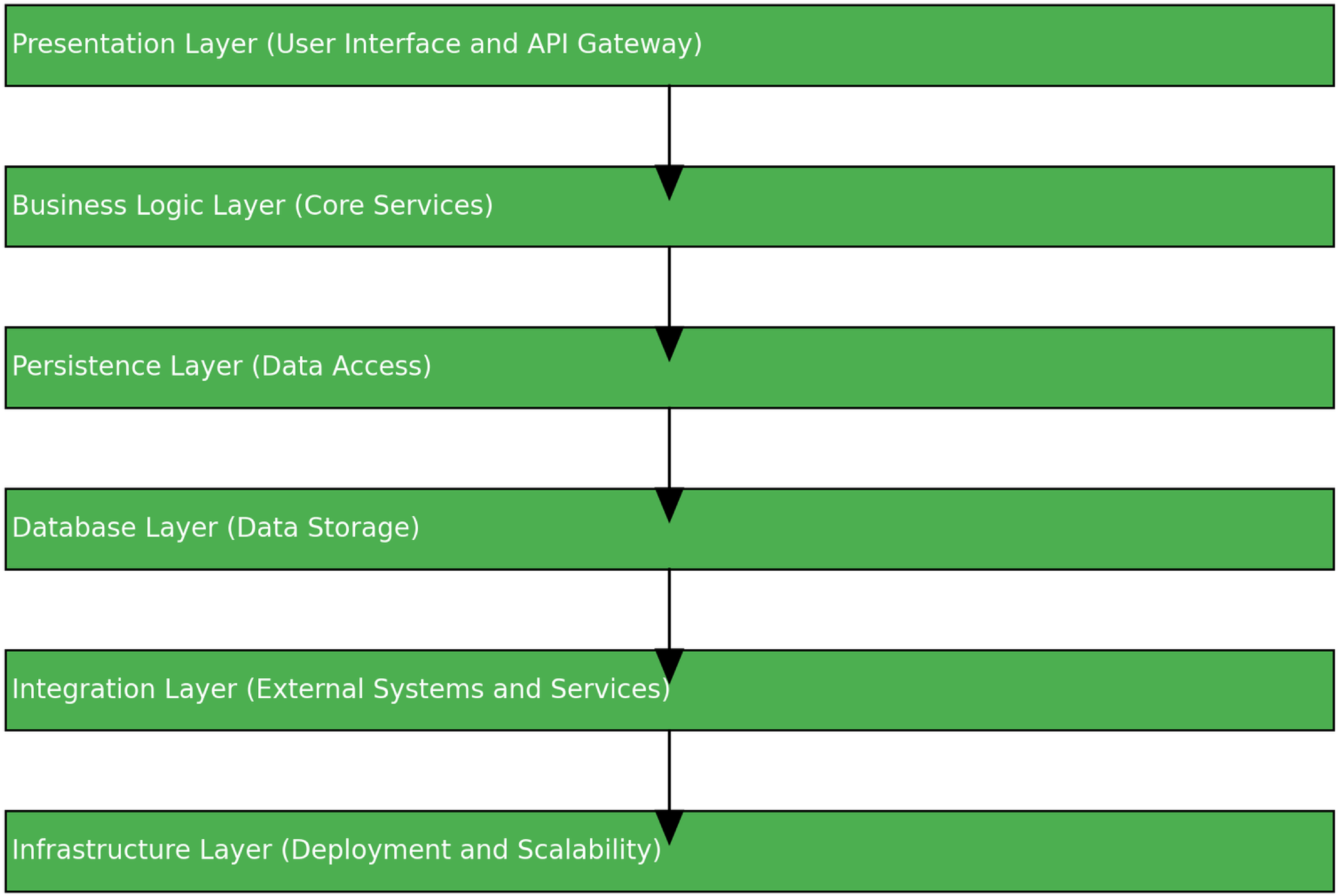
- What is Layered Architecture?
  A software design pattern where components are organized into distinct layers, each with specific responsibilities.
- Why Use It for SecureHer?
  Ensures that as our women's safety application evolves, we can easily maintain, scale, and update it without disrupting other parts of the system.

Key benifits:

- Clear separation of concerns.
- Enhanced maintainability and scalability.
- Easier to integrate new features and services.



Layered Architecture Diagram for SecureHer


Presentation Layer (User Interface and API Gateway)
Business Logic Layer (Core Services)
Persistence Layer (Data Access)
Database Layer (Data Storage)
Integration Layer (External Systems and Services)
Infrastructure Layer (Deployment and Scalability)

WALKTHROUGH OF IMPORANT TYPICAL SCENARIOS

1.SOS PRESS

2.SOS RECIEVE

3.*Heat Map*

4.Risky Zone ALERT

5.View & Filter Report

1.SOS PRESS

Presentation Layer:
The user's phone screen (UI) captures the SOS request and sends it to the backend API Gateway

Business Logic Layer:
The SOSService validates the user's location and Identifies the nearest agents, and checks if evidence is attached.

Persistence Layer:
The SOSService fetches user's previously stored data, and trusted contacts from the repositories, ensuring correct data retrieval.

Database Layer:
Under the hood, queries run on MySQL/PostgreSQL to get user data, agents availability, and any pre-existing incident records.

Integration Layer:
Calls map/geocoding API to find the nearest agents. Sends SMS or push notifications to trusted contacts and agents.

Infrastructure Layer:
Ensures all these operations scale under heavy load, logs important events, handles containerized deployments, and keeps the system stable.

2.SOS RECEIVE

Presentation Layer:
Alerts are displayed on the agents' interface, showing the SOS request details and the user’s live location.

Business Logic Layer:
Determines which agent is closest to the user and sends them the request.
Updates the agent interface to accept or reject the request.

Persistence Layer:
Stores the agent's response and SOS status updates in the database.
Tracks the status of the user-agent interaction.

Database Layer:

Updates records to reflect which agent is handling the case.
Logs any comments or actions performed by the agent during the SOS.

Integration Layer:

Sends updates to the user regarding the agent's response and expected arrival time.
Allows communication between the user and agent through audio, video call or text.

Infrastructure Layer:

Ensures all these operations scale under heavy load, logs important events, handles containerized deployments, and keeps the system stable.

3.Heat Map

Presentation Layer:

A heat map interface is displayed, using color gradients to indicate areas with high incident density.

Real-time updates ensure the map reflects the latest reported incidents.

Business Logic Layer:

Aggregates data from multiple reports to calculate risk levels in specific areas.

Dynamically updates the heat map when new reports are added or incidents are resolved.

Persistence Layer:

Fetches stored risk zone data .

Maintains a cache for faster loading of frequently accessed data.

Database Layer:

Stores geolocation data for all reports and their severity levels.
Runs queries to calculate risk levels based on report density.

Integration Layer:

Interfaces with geocoding APIs to map the incident locations accurately.
Retrieves real-time updates about ongoing incidents to modify the heat map dynamically.

Infrastructure Layer:

Ensures smooth rendering of the heat map and supports concurrent user access.
Manages heavy traffic during peak times, such as mass incident reporting.

4.Risky Zone ALERT

Presentation Layer:
Pushes real-time alerts if the user enters or approaches a risky zone.
Displays a map interface highlighting risky zones with appropriate visual markers.

Business Logic Layer:
Continuously monitors the user’s location.
Matches the user's position with stored risky zone coordinates.
Notifies the user of potential risks.

Persistence Layer:
Fetches pre-loaded data about risky zones and threat levels.
Logs instances when users are notified about these zones.

Database Layer:

Stores detailed risky zone data including type, severity, and geolocation.

Updates risk zones dynamically every one hour.

Integration Layer:

Interfaces with geocoding APIs for location tracking.

Updates the map dynamically based on external inputs about risky zones.

Infrastructure Layer:

Ensures the map loads quickly and runs smoothly.

Scales services for handling multiple users simultaneously.

5.View & Filter Report

Presentation Layer:

A detailed interface shows individual report markers on the map.
Users and agents can filter reports by date, type, severity, and location.
Clicking on a marker opens detailed information about the incident.

Business Logic Layer:

Processes user queries for specific reports or incident types.
Dynamically filters and displays only the relevant reports based on user input.

Persistence Layer:

Retrieves stored reports from the database based on search parameters.

Database Layer:

Stores detailed information for each report, including time, location, type and severity.

Runs queries to provide search results quickly and efficiently.

Integration Layer:

Uses geocoding APIs to display accurate report locations on the map.

Synchronizes live updates, ensuring users see the latest incident data.

Infrastructure Layer:

Ensures that map interactions, searches, and report viewing are seamless and responsive.

Handles large-scale user access during high-incident periods or emergencies.

Presentation Layer Details

What It Does

Handles user interactions, such as registering, logging in, pressing SOS, and viewing reports.

Components

- *Victim's Mobile App: For sending SOS requests, uploading evidence, and viewing allocated responders.*
- *Agent's Portal: Police/NGO login, accept cases, update statuses.*
- *Admin Dashboard: Manage rules, system health, and data analytics.*

Tech

Frontends in React.js (Web) and Flutter/React Native (Mobile) NGINX as API Gateway and Reverse Proxy for load balancing.

Business Logic Layer

Purpose

Defines how requests are processed, rules for SOS handling, evidence checks, notification logic.

Key services

- ***UserService: Manages user authentication and profiles.***
- ***SOSService: Validates SOS requests, finds nearest responders.***
- ***EvidenceService: Handles uploads, ensures correct visibility (public/private).***
- ***ReportService: Manages incident reports, filtering, and linking to SOS history.***
- ***HeatMapService: Aggregates data to highlight high-risk areas.***

Benifit

All core rules reside here, so changing how an SOS is processed doesn’t affect the UI or databases directly

Persistence & Database Layers

Persistence Layer

Uses repositories/DAOs to abstract away SQL queries. The business layer only sees clean methods like getUserById() instead of raw queries.

Database Layer

MySQL/PostgreSQL store structured data. We can add read replicas to handle high traffic after an SOS event.

Caching

With Redis to speed up repetitive lookups, like frequently accessed location data.

Benefit:

If we change from MySQL to another database, only this layer needs updates, not the entire application.

Integration Layer

Connects SecureHer to external systems

- ***SMS/Email gateways for sending emergency alerts to trusted contacts.***
- ***Map/Geocoding APIs for route guidance and location services.***
- ***Cloud storage (S3/GCP) for evidence files.***

Benefit

Isolates external dependencies in one place. If the SMS provider changes, we update only this layer.

Infrastructure Layer

Infrastructure Ensures

- ***Smooth deployments with Docker and Kubernetes.***
- ***Scalability: Autoscale during peak demands (e.g., multiple SOS calls at once).***
- ***Monitoring & Logging: Prometheus, Grafana, and ELK stack to catch and fix issues early.***
- ***Backup & Recovery: Automated database backups ensure data safety.***

Benefit

System remains reliable, easily maintained, and can handle sudden spikes in usage without collapsing.

Recap & Benefits

- ***Clear Boundaries: Each layer handles a distinct aspect of the system.***
- ***Maintainability: Changing the UI, adding a new evidence format, or switching databases affects only one layer.***
- ***Scalability: Infrastructure layer lets us handle growth. If user demand doubles, we scale without redesigning the whole app.***
- ***Security & Reliability: Layer isolation helps contain failures, making the system more resilient.***

Conclusion

- *The layered architecture aligns with SecureHer’s mission:*
  - *Providing quick, reliable assistance in emergencies.*
  - *Ensuring easy evolution of features, integrations, and scaling as the user base grows.*
- *This structure ensures long-term flexibility, supporting new functionalities and enhancements without major rewrites.*