Software Requirements Specification

for

TaskNest, Release 1.0

Version 1.0 approved

Prepared by Krishna Sharma K

Vellore Institute of Technology, Chennai

July 21, 2025

Table of Contents

Revision History ..... **ii**
1. Introduction **1**
1.1 Purpose 1
1.2 Document Conventions 1
1.3 Project Scope and Product Features 1
1.4 References 2
2. Overall Description **2**
2.1 Product Perspective 2
2.2 Product Functions 2
2.3 User Classes and Characteristics 2
2.4 Operating Environment 2
2.5 Constraints 3
2.6 Assumptions and Dependencies 3
3. System Features **4**
3.1 Functional Requirements 4
3.2 Non-Functional Requirements 5
4. Data Requirements **6**
4.1 Logical Data Model 6
4.2 Data Dictionary 6
4.3 Reports 9
4.4 Data Integrity, Retention, and Disposal 9
5. External Interface Requirements **10**
5.1 User Interfaces 10
5.2 Software Interfaces 10
5.3 Hardware Interfaces 10
5.4 Communications Interfaces 10
6. Quality Attributes **10**
6.1 Usability Requirements 10
6.2 Performance Requirements 10
6.3 Security Requirements 11
6.4 Safety Requirements 11
6.5 Availability Requirements 11
6.6 Robustness Requirements 11
7. Appendix **11**
7.1 Appendix A: Glossary 11
7.2 Appendix B: References 15
7.3 Appendix C: Abbreviations 15

Revision History

Name	Date	Reason For Changes	Version
Krishna Sharma	7/21/25	Initial Draft	1.0

1. Introduction

1.1 Purpose

The purpose of this document is to define the functional and non-functional requirements for the TaskNest web application - a personal productivity tool that implements a Kanban-style task management system. The document will serve as a contractual baseline between the developer (solo software engineer) and potential stakeholders such as evaluators, instructors, or portfolio reviewers. It aims to ensure that all features, design decisions, constraints, and quality attributes are specified before implementation begins, supporting traceability throughout the development lifecycle.
This SRS also functions as a reference point for maintenance, testing, and future enhancements.

1.2 Document Conventions

This document follows the IEEE 830-1998 standard for Software Requirements Specifications, with the following notational conventions:

- **Shall** indicates a mandatory requirement.
- **Should** indicates a desirable but non-mandatory requirement.
- **May** indicates an optional feature.
- Requirements are uniquely identified using the format: `REQ-[Category]-[ID]` (e.g., `REQ-FUNC-01` for functional requirements).
- Diagrams follow UML 2.x notation.
- Technical terms are written in *italic* on first occurrence and defined in the glossary.

1.3 Project Scope and Product Features

TaskNest (Solo Edition) is a **single-user Kanban task management web application** designed to demonstrate disciplined software engineering practices rather than maximize feature count.
The application will enable users to:

- Register and log in securely
- Create and manage projects
- Create, edit, and delete tasks within projects
- View and manage tasks via a drag-and-drop Kanban board
- Monitor project/task status via a dashboard

The system will be built with Next.js for the frontend, an Python/Express.js backend, and PostgreSQL for persistent storage. The deployment will follow a CI/CD pipeline, with automated testing and documentation.

Core Features

- User Authentication (signup, login, logout)
- CRUD Operations for projects and tasks
- Kanban Board View with status columns: *To Do*, *Doing*, *Done*
- User Dashboard summarizing task and project status

Optional Features (time-permitting)

- Comments or basic messaging per task

- Task reminders
- Real-time updates using WebSockets
- Dark mode theme toggle

1.4 References

1. IEEE Std 830-1998, *IEEE Recommended Practice for Software Requirements Specifications*.
2. OpenAPI Specification, Version 3.1.0: https://spec.openapis.org/oas/v3.1.0/
3. React Documentation: https://react.dev/
4. Next.js Documentation: https://nextjs.org/docs/
5. PostgreSQL Documentation: https://www.postgresql.org/docs
6. GitHub Actions Documentation: https://docs.github.com/actions/

2. Overall Description

2.1 Product Perspective

TaskNest is a **standalone web application** intended for personal productivity management. It is not dependent on third-party applications for its core functionality, although it may integrate with deployment platforms (Vercel, Render or AWS) and optional APIs (e.g., email notifications).

The product follows a **three-tier architecture**:

1. **Frontend (Presentation Layer)** – Built using Next.js (React framework), providing the user interface for authentication, task/project management, Kanban interaction, and dashboard view.
2. **Backend (Application Layer)** – Implemented in Python/Express.js, exposing RESTful APIs for business logic and handling CRUD operations, authentication, and data validation.
3. **Database (Data Layer)** – PostgreSQL for persistent storage of users, projects, and tasks.

External Interfaces include:
- Browser-based UI (desktop and mobile responsive)
- REST API endpoints for frontend-backend communication
- Optional WebSocket channel for real-time updates

2.2 Product Functions

At a high level, TaskNest will:
1. **Authenticate Users** - register, log in, and log out securely.
2. **Manage Projects** - create, view, update, and delete projects.
3. **Manage Tasks** - create, view, update, and delete tasks associated with projects.
4. **Kanban Visualization** - display tasks in columns (*To Do*, *Doing*, *Done*) and allow drag-and-drop status updates.
5. **Dashboard Summary** - provide a visual overview of tasks and projects (counts, completion rates).
6. **Optional Add-ons** - support comments, reminders, dark mode, or real-time updates if time allows.

2.3 User Classes and Characteristics

The primary user is:

- **Individual users** (single account owner per deployment).
- Basic to intermediate computer literacy.
- Familiarity with task and project management concepts.
- No requirement for advanced technical skills; user interface will be intuitive and self-explanatory.

2.4 Operating Environment

OE-1: TaskNest shall operate correctly with the following web browsers: Microsoft Edge versions 139.0.0+; Firefox versions 141.0.0+; Google Chrome versions 139.0.0+; and Apple Safari versions 18.5+.

OE-2: The COS shall operate on web and mobile interfaces.

2.5 Constraints

2.5.1 Technical Constraints:

T-CO-1: Frontend must be implemented in Next.js.

T-CO-2: Backend must use Python/Express.js with PostgreSQL.

T-CO-3: Deployment targets: Vercel (frontend), Render (backend), or alternative, on AWS.

T-CO-4: Testing must include unit, integration, and E2E coverage.

2.5.2 Non-Technical Constraints:

NT-CO-1: Development is performed by a single developer.
NT-CO-2: Timeline and scope are limited by available hours and deadlines.
NT-CO-3: Primary focus is on **software engineering process discipline** rather than maximal feature set.

2.5.3 Security Constraints

S-CO-1: Authentication must use secure password storage.
S-CO-2: Sensitive information must not be stored in client-side localStorage in plain text.

2.6 Assumptions and Dependencies

AS-1: The user has a stable internet connection and a modern web browser.

DE-1: Deployment platforms (Vercel, Render) will remain operational and accessible.

DE-2: PostgreSQL database hosting will support required performance and storage.

DE-3: Dependencies such as Next.js, Express.js, and PostgreSQL client libraries will remain maintained during the development lifecycle.

3. System Features

3.1 Functional Requirements

Each functional requirement is uniquely identified and traceable using the format REQ-FUNC-XX.
Priority:

- **H** = High (must be implemented for MVP)
- **M** = Medium (important but can be deferred)
- **L** = Low (optional/stretch feature)

3.1.1 User Authentication

- **REQ-FUNC-01 (H):** The system shall allow a user to create an account by providing a unique email and password.
- **REQ-FUNC-02 (H):** The system shall securely store passwords using services or a salted hashing methodology (e.g., bcrypt)**.**
- **REQ-FUNC-03 (H):** The system shall allow registered users to log in with their email and password.
- **REQ-FUNC-04 (H):** The system shall maintain a session using HTTP-only cookies or equivalent secure method.
- **REQ-FUNC-05 (H):** The system shall allow a logged-in user to log out, invalidating their session.

3.1.2 Project Management

- **REQ-FUNC-06 (H):** The system shall allow a user to create a project with a title and optional description.
- **REQ-FUNC-07 (H):** The system shall allow a user to view all projects they own.
- **REQ-FUNC-08 (H):** The system shall allow a user to edit the title or description of a project.
- **REQ-FUNC-09 (H):** The system shall allow a user to delete a project, removing all associated tasks.

3.1.3 Task Management

- **REQ-FUNC-10 (H):** The system shall allow a user to create a task within a project, specifying title, description, and status.
- **REQ-FUNC-11 (H):** The system shall allow a user to update the title, description, and status of a task.
- **REQ-FUNC-12 (H):** The system shall allow a user to delete a task.
- **REQ-FUNC-13 (H):** The system shall allow a user to change a task’s status by drag-and-drop on the Kanban board.

3.1.4 Dashboard and Views

- **REQ-FUNC-14 (H):** The system shall provide a dashboard summarizing total projects, total tasks, and tasks grouped by status.
- **REQ-FUNC-15 (M):** The system should display visual indicators (colors, icons) for task urgency or approaching due dates.

3.1.5 Optional Features

- **REQ-FUNC-16 (M):** The system should allow users to add comments to tasks.
- **REQ-FUNC-17 (M):** The system should allow users to set reminders for tasks.
- **REQ-FUNC-18 (M):** The system should update Kanban boards in real time without requiring a page refresh.
- **REQ-FUNC-19 (L):** The system may provide a dark mode theme toggle.

3.2 Non-Functional Requirements

Non-functional requirements are identified using REQ-NFR-XX.

3.2.1 Performance

- **REQ-NFR-01:** The system shall load the dashboard view within 3 seconds on a standard broadband connection.
- **REQ-NFR-02:** The system shall support at least 50 simultaneous active sessions without performance degradation (for demo/testing scale).

3.2.2 Security

- **REQ-NFR-03:** The system shall encrypt all client-server communication using HTTPS in production.
- **REQ-NFR-04:** The system shall validate all user inputs server-side to prevent injection attacks.
- **REQ-NFR-05:** The system shall implement rate-limiting on authentication endpoints to mitigate brute-force attacks.

3.2.3 Usability

- **REQ-NFR-06:** The system shall provide a responsive design for desktop and mobile devices.
- **REQ-NFR-07:** The system shall use intuitive icons and labels for all primary actions.

3.2.4 Maintainability

- **REQ-NFR-08:** The codebase shall follow established style guidelines.
- **REQ-NFR-09:** The system shall include automated unit, integration, and E2E tests covering at least 60% of code

3.2.5 Reliability

- **REQ-NFR-10:** The system shall automatically reconnect to the database after transient network failures.
- **REQ-NFR-11:** The system shall recover gracefully from backend restarts without losing persisted data.

4. Data Requirements

4.1 Logical Data Model

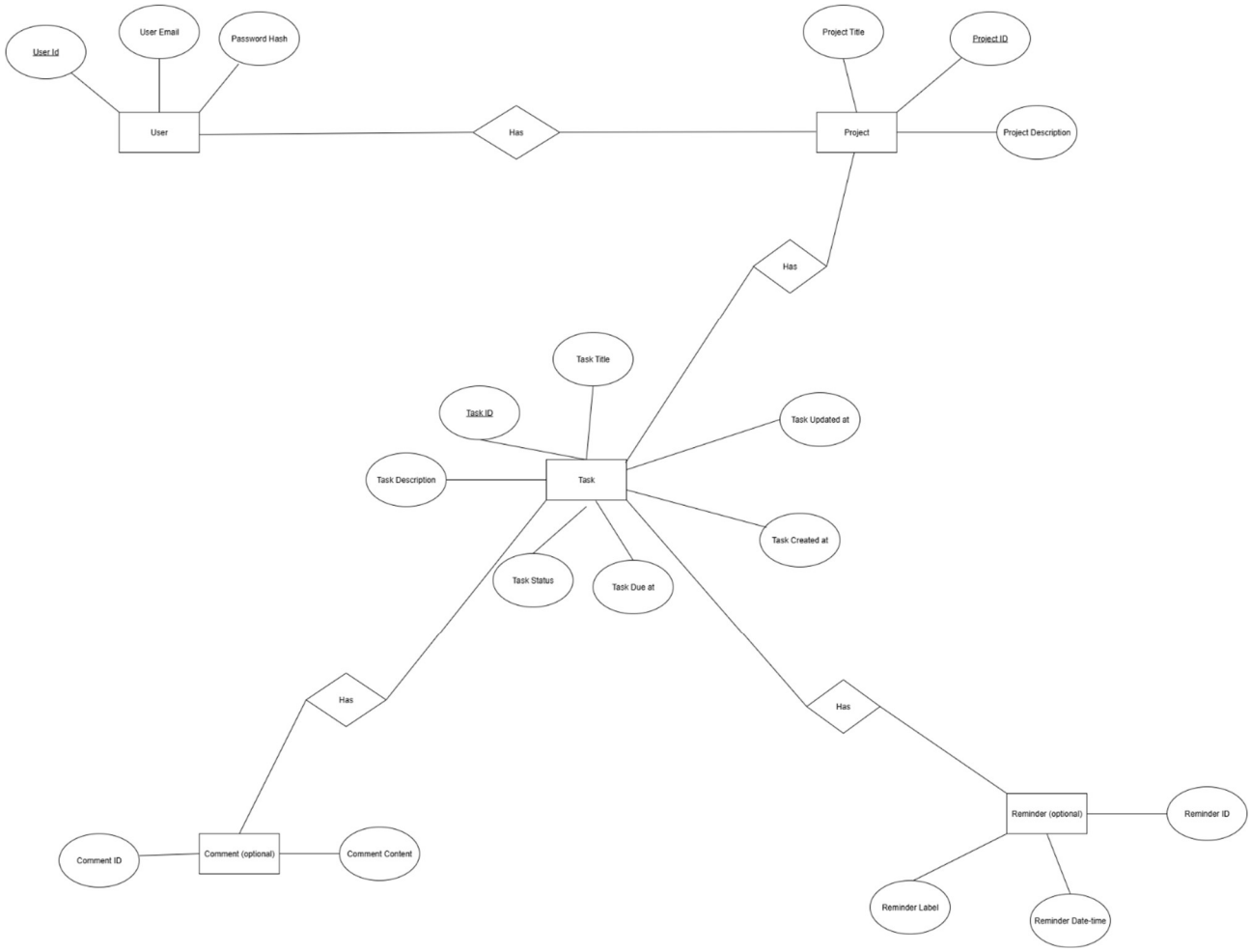

Figure 1.1. Partial data model for release 1.0 of TaskNest.

4.2 Data Dictionary

Data Element	Description	Composition / Data Type	Length (or maximum length, where applicable)	Values
User ID	Unique identifier for each user	UUID	36	Auto-generated
User Email	Email address for login	Alphanumeric	254	Must be unique, RFC 5322 format
Password Hash	Secure hash of user password	Alphanumeric (hash)	60	bcrypt hash
Project ID	Unique identifier for each project	UUID	36	Auto-generated
Project Title	Name of the project	Alphanumeric	100	Letters, numbers, spaces, hyphens
Project Description	Optional longer description	Alphanumeric (text)	500	Markdown allowed (optional)
Task ID	Unique identifier for each task	UUID	36	Auto-generated
Task Title	Short description of task	Alphanumeric	150	Letters, numbers, spaces, punctuation
Task Description	Detailed description	Alphanumeric (text)	2000	Markdown allowed (optional)
Task Status	Workflow state	Enum	10	"To Do", "Doing", "Done"
Task Due Date	Optional deadline	Date (YYYY-	10	ISO 8601 date

Data Element	Description	Composition / Data Type	Length (or maximum length, where applicable)	Values
		MM-DD)		
Task Created At	Timestamp of creation	DateTime	-	ISO 8601 datetime
Task Updated At	Timestamp of last update	DateTime	-	ISO 8601 datetime
Comment ID *(optional)*	Unique identifier for comment	UUID	36	Auto-generated
Comment Content *(optional)*	Text of comment	Alphanumeric (text)	1000	Markdown allowed
Reminder ID *(optional)*	Unique identifier for reminder	UUID	36	Auto-generated
Reminder Date-Time *(optional)*	Date/time when reminder should trigger	DateTime	-	ISO 8601 datetime

4.3 Reports

4.3.1 Project and Task Summary Report

Report ID:	TN-RPT-1
Report Title:	Project and Task Summary
Report Purpose:	Allow the user to view all active projects and associated task counts (by status) in a single view for better workload tracking.
Priority:	High
Report Users:	Authenticated User
Data Sources:	Database tables for projects and tasks
Frequency and Disposition:	Generated on demand; displayed in browser dashboard.
Latency:	Report should load within 2 seconds of request.
Visual Layout:	Card/grid view per project, showing progress bar for task completion.
Header and Footer:	Header includes report title and current date; no footer for web view.
Report Body:	• Project Title • Number of Tasks (Total) • Number of Tasks by Status (To Do, Doing, Done) • Completion Percentage Selection Criteria: All projects belonging to authenticated user. Sort Criteria: Projects ordered by creation date (newest first).
End-of-Report Indicator:	None
Interactivity:	Clicking a project opens its detailed Kanban view.
Security Access Restrictions:	A user can only access their own projects and tasks.

[Note: Other TaskNest reports are not provided in this example.]

4.4 Data Integrity, Retention, and Disposal

DI-1: The system **shall** retain all user accounts until explicitly deleted by the user or project administrator.

DI-2: The system **shall** retain project and task data for at least 12 months after deletion request, to support recovery and audit, unless otherwise required by applicable data protection laws.

DI-3: The system **shall** ensure referential integrity (tasks cannot exist without an associated project; projects cannot exist without an associated user).

DI-4: The system **shall** ensure all deletion operations are soft-deletes by default (mark as inactive) unless explicitly purged.

DI-5: The system **shall** remove any purged data from all active databases and backups within 30 days of purge.

5. External Interface Requirements

5.1 User Interfaces

UI-1: Browser-based UI, accessible via common modern browsers (Chrome, Firefox, Edge, Safari).

UI-2: UI elements include navigation bar, project list, Kanban board, dashboard, and authentication forms.

5.2 Software Interfaces

SI-1: PostgreSQL for data persistence.

SI-2: RESTful API between frontend and backend.

SI-3: Optional WebSocket server for real-time updates.

5.3 Hardware Interfaces

Not applicable as the system is a cloud-based web-application.

5.4 Communications Interfaces

CI-1: HTTPS for secure data transfer.

CI-2: JSON for data exchange between client and server.

6. Quality Attributes

6.1 Usability Requirements

USE-1: The system **shall** allow a user to create a project and its first task within three interactions (clicks or form submissions) from the dashboard.

USE-2: 95% of first-time users **shall** be able to create and update a task without consulting documentation or help resources.

USE-3: The Kanban board **shall** support drag-and-drop task updates that immediately reflect the new status visually.

6.2 Performance Requirements

PER-1: The system **shall** support at least 50 concurrent active sessions without performance degradation during peak usage periods.

PER-2: 95% of dashboard views **shall** fully load within 3 seconds over a 20 Mbps or faster internet connection.

PER-3: Task status changes made via drag-and-drop **shall** be acknowledged by the backend and updated in the UI within an average of 1 second, and a maximum of 3 seconds.

PER-4: Search and filter operations within a user's projects and tasks **shall** return results within 2 seconds for datasets up to 500 tasks.

6.3 Security Requirements

SEC-1: All communication containing authentication tokens or personal data **shall** be encrypted using HTTPS with TLS 1.2 or higher.

SEC-2: Users **shall** be required to log in before performing any project or task management operations.

SEC-3: The system **shall** restrict all data access to resources owned by the authenticated user.

SEC-4: Passwords **shall** never be stored in plain text and must be hashed.

SEC-5: The system **shall** enforce a limit of 3 failed login attempts per minute before temporarily locking the account for 24 hrs.

6.4 Safety Requirements

SAF-1: The system **shall** prompt the user for confirmation before permanently deleting any project or task.

SAF-2: The system **shall** prevent accidental data loss by implementing undo capability for at least 10 seconds after a delete action (where feasible in UI).

6.5 Availability Requirements

AVL-1: The system **shall** be available at least 99% of the time, excluding scheduled maintenance windows, for both frontend and backend services.

AVL-2: Scheduled maintenance periods **shall** be communicated to users at least 24 hours in advance, via an in-app notification or deployment notes.

6.6 Robustness Requirements

ROB-1: If the connection between the client and server is lost during a task update, the system **shall** queue the change locally and attempt to resubmit automatically upon reconnection.

ROB-2: The system **shall** recover gracefully from backend restarts without requiring users to re-enter unsaved data, provided they have not logged out.

7. Appendix

7.1 Appendix A: Glossary

Term / Acronym	Definition
API	Application Programming Interface - a set of defined endpoints allowing communication between frontend and backend.

Term / Acronym	Definition
bcrypt	A password-hashing function that incorporates a salt to protect against rainbow table attacks.
CRUD	Create, Read, Update, Delete - the four basic operations for persistent data storage.
CI/CD	Continuous Integration / Continuous Deployment - automated processes for building, testing, and deploying code.
CSS	Cascading Style Sheets - language used to style HTML content.
Drag-and-drop	A UI interaction allowing users to select an object and move it to a different position or category via mouse or touch.
E2E Testing	End-to-End Testing - tests that simulate real user actions from the frontend through to the backend and database.
Enum	A special data type consisting of a set of predefined constant values.
ERD	Entity-Relationship Diagram - a visual model showing entities (tables) and their relationships.
ESLint	A linting utility for JavaScript/TypeScript that enforces code style and quality rules.
Express.js	A minimalist web framework for Node.js used for building backend APIs.

Term / Acronym	Definition
Frontend	The client-side part of the application (UI) that runs in the user's browser.
HTTP-only Cookie	A cookie that cannot be accessed by JavaScript, reducing the risk of XSS attacks.
ISO 8601	An international standard for representing date and time formats.
JWT	JSON Web Token - a compact and secure way to represent claims between two parties, often used for authentication.
Kanban Board	A visual workflow tool showing tasks in columns that represent different stages of completion.
Markdown	A lightweight markup language for formatting text.
Next.js	A React framework with built-in routing, server-side rendering, and API support.
NFR	Non-Functional Requirement - a system quality or constraint, such as performance or security.
Node.js	A JavaScript runtime environment that executes code outside the browser, often used for backend services.
OpenAPI	A specification for documenting RESTful APIs in a standard, machine-readable format.

Term / Acronym	Definition
PostgreSQL	A powerful open-source relational database management system.
Prisma	A type-safe ORM (Object Relational Mapping) tool for Node.js and TypeScript.
REST API	Representational State Transfer - an architectural style for designing networked applications via HTTP requests.
RTM	Requirements Traceability Matrix - a table mapping each requirement to its corresponding test(s).
SRS	Software Requirements Specification - a formal document describing software functionality, constraints, and design.
Swagger	A set of tools for designing, building, documenting, and consuming RESTful APIs.
TLS	Transport Layer Security - protocol for encrypting data between client and server.
TypeScript	A superset of JavaScript that adds static type definitions.
UML	Unified Modeling Language - a standard way to visualize system design.
UUID	Universally Unique Identifier - a 128-bit number used to uniquely identify data.

Term / Acronym	Definition
Vercel	A cloud platform for deploying and hosting frontend applications.
WebSocket	A protocol enabling full-duplex communication between client and server for real-time updates.

7.2 Appendix B: References

1. IEEE Std 830-1998, *IEEE Recommended Practice for Software Requirements Specifications*.
2. OpenAPI Specification, Version 3.1.0 - https://spec.openapis.org/oas/v3.1.0
3. React Documentation - https://react.dev
4. Next.js Documentation - https://nextjs.org/docs
5. PostgreSQL Documentation - https://www.postgresql.org/docs
6. GitHub Actions Documentation - https://docs.github.com/actions
7. bcrypt Documentation - https://github.com/pyca/bcrypt
8. Argon2 Documentation - https://github.com/P-H-C/phc-winner-argon2/blob/master/argon2-specs.pdf
9. Prisma ORM Documentation - https://www.prisma.io/docs

7.3 Appendix C: Abbreviations

Abbreviation	Full Form
API	Application Programming Interface
CI/CD	Continuous Integration / Continuous Deployment

CRUD	Create, Read, Update, Delete
E2E	End-to-End
ERD	Entity-Relationship Diagram
HTTP	Hypertext Transfer Protocol
HTTPS	Hypertext Transfer Protocol Secure
JWT	JSON Web Token
ORM	Object Relational Mapping
REST	Representational State Transfer
RTM	Requirements Traceability Matrix
SRS	Software Requirements Specification
TLS	Transport Layer Security
UML	Unified Modeling Language
UUID	Universally Unique Identifier