# Review Process Activities

**# Source**: **Chatgpt (GPT-5-Standard) at [8/14/2025]**

The **Review Process Activities** (as defined in ISTQB and industry best practices) typically follow a structured cycle to ensure systematic checking of artifacts like requirements, design documents, code, or test cases.

## Process Activities

**# Source**: **Chatgpt (GPT-5-Standard) at [8/14/2025]**

Here’s the breakdown:

### 1. Planning

**# Source**: **Chatgpt (GPT-5-Standard) at [8/14/2025]**

* **Purpose:** Define the review’s scope, objectives, participants, and schedule.
* **Key Actions:**
  + Identify the artifact(s) to review (e.g., SRS, UML diagram, source code).
  + Choose review type (informal, walkthrough, technical review, inspection).
  + Assign roles: author, reviewer(s), moderator, scribe.
  + Prepare review plan & set timelines.
* **Example:** Plan to review the "Payment Module" requirements document with 3 reviewers from business, QA, and development.

### 2. Review initiation orKick-off Meeting *(optional but recommended in formal reviews)*

**# Source**: **Chatgpt (GPT-5-Standard) at [8/14/2025]**

* **Purpose:** Ensure everyone understands the artifact, objectives, and review process.
* **Key Actions:**
  + Author explains the content and context.
  + Reviewers clarify doubts before deep reading.
  + Confirm review checklists or criteria to be used.
* **Example:** Author walks through the ride-booking flow diagram before reviewers start.

### 3. Individual Preparation (or Individual Review)

**# Source**: **Chatgpt (GPT-5-Standard) at [8/14/2025]**

* **Purpose:** Reviewers examine the artifact independently to find issues.
* **Key Actions:**
  + Read the document/code carefully.
  + Compare against requirements, standards, and checklists.
  + Record findings (potential defects, ambiguities, missing info).
* **Example:** Reviewer notes that "login" is sometimes called "sign in" in the SRS.

### 4. Review Meeting

**# Source**: **Chatgpt (GPT-5-Standard) at [8/14/2025]**

* **Purpose:** Discuss and consolidate findings, agree on defects, and propose corrections.
* **Key Actions:**
  + Each reviewer presents their observations.
  + Moderator keeps discussion on track.
  + Scribe records defects and agreed changes.
* **Example:** Group agrees that the UML diagram is missing error-handling flows.

### 5. Rework

**# Source**: **Chatgpt (GPT-5-Standard) at [8/14/2025]**

* **Purpose:** Author fixes the defects found during the review.
* **Key Actions:**
  + Update the artifact according to the review report.
  + Clarify and resolve any disputed issues.
* **Example:** Designer updates database ERD to match the requirement for “multi-currency support.”

### 6. Follow-up

**# Source**: **Chatgpt (GPT-5-Standard) at [8/14/2025]**

* **Purpose:** Ensure corrections were made and objectives met.
* **Key Actions:**
  + Moderator or reviewers verify rework.
  + Close review or schedule another round if needed.
* **Example:** Moderator confirms that PayPal payment method has been added to the SRS as agreed.

## Real-world review process example [**formal inspection]**

**# Source**: **Chatgpt (GPT-5-Standard) at [8/14/2025]**

Alright — here’s a **real-world review process example** told as an actual project scenario so you can see the **Review Process Activities** in action.

### Scenario: Review of the "Online Payment Requirements" Document for an E-commerce Platform

### **1. Planning**

**# Source**: **Chatgpt (GPT-5-Standard) at [8/14/2025]**

* **Artifact:** Payment Module section of the Software Requirements Specification (SRS).
* **Objective:** Ensure payment requirements are complete, correct, and compliant before design starts.
* **Participants:**
  + **Author:** Business Analyst (Sara)
  + **Reviewers:** Backend Developer (Ahmed), QA Lead (Lina), Security Specialist (Omar), Product Owner (Maya)
  + **Moderator:** Project Manager (Ali)
  + **Scribe:** QA Lead (Lina)
* **Schedule:** Review to be completed in 3 working days.

### **2. Kick-off Meeting**

**# Source**: **Chatgpt (GPT-5-Standard) at [8/14/2025]**

* **Sara** (Author) walks everyone through the SRS payment section:  
  + Supported methods: Credit Card, PayPal.
  + Transaction steps: Checkout → Payment → Confirmation.
* **Omar** asks if encryption requirements are specified.
* **Maya** reminds that the target market also expects Apple Pay.
* **Ali** confirms everyone will use the company’s “Requirements Review Checklist.”

### **3. Individual Preparation**

**# Source**: **Chatgpt (GPT-5-Standard) at [8/14/2025]**

* **Ahmed** notices there’s no mention of refund flow.
* **Lina** spots missing acceptance criteria for handling payment failures.
* **Omar** points out that PCI DSS compliance and TLS encryption details are missing.
* **Maya** highlights the absence of Apple Pay in supported payment methods.
* Each reviewer logs these issues into the review tool (e.g., Jira or Confluence page).

### **4. Review Meeting**

**# Source**: **Chatgpt (GPT-5-Standard) at [8/14/2025]**

* **Moderator Ali** leads the discussion:  
  + Team agrees to add Apple Pay to supported methods.
  + Refund and payment failure scenarios will be explicitly documented.
  + Encryption and compliance requirements will be added per security standards.
* **Lina** (Scribe) records all agreed changes with defect IDs.

### **5. Rework**

**# Source**: **Chatgpt (GPT-5-Standard) at [8/14/2025]**

* **Sara** updates the SRS:  
  + Adds **Apple Pay** as a supported method.
  + Documents **refund process** and **failure-handling scenarios**.
  + Includes **PCI DSS compliance clause** and **TLS 1.3 encryption requirement**.

### **6. Follow-up**

**# Source**: **Chatgpt (GPT-5-Standard) at [8/14/2025]**

* **Omar** re-checks the security section → confirms PCI DSS and encryption are now covered.
* **Lina** verifies that new acceptance criteria for failures are included.
* **Ali** closes the review, marking the SRS as “Ready for Design.”

✅ **Outcome:** By following the structured review process, the team identified and fixed critical gaps **before any code was written**, avoiding costly rework later.

## WHAT IS HAPPENED NEXT

**# Source**: **Chatgpt (GPT-5-Standard) at [8/14/2025]**

After that review process was completed and the SRS was marked **“Ready for Design”**, here’s what happened next in a real-world flow:

### **1. Design Phase Kick-off**

**# Source**: **Chatgpt (GPT-5-Standard) at [8/14/2025]**

* The **system architects** took the updated SRS and began creating:  
  + Database schema including **payment transactions table** (with encryption in mind).
  + Service architecture to integrate **Credit Card, PayPal, and Apple Pay APIs**.
  + Refund and payment failure handling in the sequence diagrams.

### **2. Test Planning Alignment**

**# Source**: **Chatgpt (GPT-5-Standard) at [8/14/2025]**

* **QA team** (led by Lina) updated:  
  + The **test plan** to include refund and payment failure scenarios.
  + Test cases for **Apple Pay flows**.
  + Security test cases to check for **TLS 1.3** and **PCI DSS compliance**.

### **3. Compliance Check Preparation**

**# Source**: **Chatgpt (GPT-5-Standard) at [8/14/2025]**

* **Omar** (Security Specialist) started preparing:  
  + Vendor questionnaires for payment gateway PCI DSS certification.
  + Automated static scanning scripts for encryption verification.

### **4. Stakeholder Confirmation**

**# Source**: **Chatgpt (GPT-5-Standard) at [8/14/2025]**

* **Maya** (Product Owner) arranged a short demo with stakeholders:  
  + Walked them through the updated payment process on the wireframes.
  + Got **final business sign-off** before developers wrote the first line of code.

### **5. Development Start**

**# Source**: **Chatgpt (GPT-5-Standard) at [8/14/2025]**

* Backend developers integrated:  
  + API connectors for Credit Card, PayPal, and Apple Pay.
  + Refund endpoints with audit logging.
  + Payment failure retry logic.

### **6. Parallel Static Testing**

**# Source**: **Chatgpt (GPT-5-Standard) at [8/14/2025]**

* Even as coding began, **static code analysis** (SonarQube) was configured from day one to catch:  
  + SQL injection risks.
  + Logging of sensitive card data.
  + Unused variables or overly complex methods.

**Result:** Because of that early **static review** and the “fix before code” approach, the project avoided a major issue — in a similar past project, missing Apple Pay support was only caught *after launch*, which cost an extra 3 weeks of emergency development and retesting.

# 