**Assignment #2 - Use-Case and Sequence Models**

**Task 1 - Updated System Description**

**Banking - Communication Services System (Email & SMS)**

**System Overview :** The Banking Communication Services System is a secure, role-based platform designed to create, schedule, and deliver customer-facing messages through Email and SMS channels. Building upon the domain model established in Assignment #1, this system now explicitly defines user interactions through use cases and operational sequences.

**Extended System Description**

**Core Functionality :** The system supports both transactional messaging (account alerts, OTPS, password resets) and marketing campaigns (promotional offers, product announcements). It provides comprehensive message lifecycle management from creation through delivery, with full audit trails for banking compliance.

**Purpose :** The system enables a bank to create, schedule, send, and audit customer-facing communications across **Email** and **SMS**. It supports both **transactional** traffic (e.g., OTPs, account alerts) and **marketing** campaigns. It provides message templating, multi-recipient delivery with retries, tracking and reporting per recipient, and **sustainability** metrics (energy and CO₂ estimates) at campaign level.

**Actors**

**Primary Actors**These are real users who directly interact with the system’s interface (web portal, dashboard, or admin console). They all inherit conceptually from the abstract actor **User**, mirroring the class model from Assignment #1.

**1. Customer :** Customers are the recipient of messages. They view received email and SMS messages in their messsage inbox, opt-in/opt-out of marketing communications, update communication preferences (email/SMS) and report delivery issues or spam.

**2. Bank Staff : Bank staff are authors and managers of campaigns and messages. They create and manage marketing campaigns, design message templates (email and SMS), schedule message delivery, monitor campaign performance and delivery rates, generate campaign reports and manage customer segments for targeted campaigns.**

**3. Channel Admin : Channel admins configure email/SMS gateways and retry/throttling policies. They configure email/SMS gateway settings, monitor channel health and throughput, manage delivery retry policies, optimize channel performance, troubleshoot delivery failures and configure rate limits and throttling.**

**4. System Admin : System admins manage roles, security, audit policies. They manage user accounts and roles, configure system security settings, generate audit logs and compliance reports, monitor system-wide activities, manage data retention policies and configure sustainability tracking parameters.**

**Secondary Actors  
These are entities outside the system boundary that communicate with it through APIs or interfaces.  
They are necessary for representing real-world integrations and compliance contexts.  
  
5. Messaging Providers - external systems: Messaging providers are Email Gateway and SMS Gateway (deliver messages and return status callbacks).  
- Email Gateway : Email Gateway is the e**xternal messaging provider responsible for physically delivering **email messages** to customers. The system sends prepared emails to this gateway, and the gateway returns delivery confirmations or failure reports.

#### - S****MS Gateway :** SMS Gateway is the e**xternal messaging provider for SMS delivery. It works similarly to the Email Gateway but operates over telecom channels.

**6. Compliance Auditor : Compliance auditor is the external, read-only actor representing regulatory or auditing personnel. They a**ccess immutable audit trails to verify message compliance, data retention, and sustainability reports.

**Key Domain Classes**

* **User** (abstract) → **{Customer, BankStaff, ChannelAdmin, SystemAdmin}**
* **Message** (abstract) → {**EmailMessage**, **SMSMessage**}
* **Campaign**
* **Delivery** (association class between Message and Customer)
* **SustainabilityMetric**
* **MessageStatus** (enumeration: DRAFT, SCHEDULED, SENT, DELIVERED, FAILED)

**Notable Attributes/Constraints**

* Message./deliveryRate (derived), SMSMessage./smsParts (derived).
* **Delivery** keeps per-recipient status and attempts ≤ 5 (retry policy).
* Only **BankStaff** can create messages/campaigns; **ChannelAdmin** cannot edit message content; **SystemAdmin** manages roles & security.
* Each **Campaign** may reference one **SustainabilityMetric** (0..1).

**Application-Level Operations (Interaction Model)**

**Campaign Management Operations :**

* createCampaign(name: String, startDate: Date, endDate: Date) : Campaign
  + Caller: **BankStaff (via BankStaffPortal)**
  + Layer: Boundary → CampaignController → Campaign (Entity)
  + Throws: ValidationException (dates), AuthException
  + Notes: Creates a new campaign in DRAFT state. Validation ensures start and end dates are logical and non-overlapping.
* scheduleCampaign(campaignId: String, scheduledAt: DateTime) : boolean
  + Caller: **BankStaff**
  + Layer: SchedulingController
  + Throws: NotFoundException, ValidationException
  + Notes: Sets the scheduled execution time of a campaign and updates its status to SCHEDULED.
* assignMessagesToCampaign(campaignId: String, messageIds: List<String>) : void
  + Caller: **BankStaff**
  + Layer: CampaignController
  + Throws: NotFoundException, PermissionException
  + Notes: Links one or more message templates to a campaign. Idempotent operation; skips already-assigned messages.
* getCampaignStatus(campaignId: String) : CampaignStatusDTO
  + Caller: **BankStaff / SystemAdmin**
  + Layer: ReportingController
  + Returns: delivery counts, deliveryRate (derived), sustainability metric link.
  + Notes: Retrieves the campaign’s progress, including total messages, successful deliveries, and sustainability metrics.
* generateCampaignReport(campaignId: String, format: Enum{PDF,CSV}) : ReportRef
  + Caller: **BankStaff / SystemAdmin / Compliance Auditor**
  + Layer: ReportingController → AuditController
  + Notes: Produces an auditable campaign report summarizing deliveries, success rates, and sustainability metrics. If invoked by a Compliance Auditor, the report is immutable and digitally signed.

**Message Operations :**

* createEmailMessage(campaignId: String, subject: String, body: String, attachments: List<File>) : Message
  + Caller: **BankStaff**
  + Layer: MessageController
  + Throws: AttachmentTooLargeException, ValidationException.
  + Notes: Creates an email message template within a campaign. Attachments are validated for size and format.
* createSMSMessage(campaignId: String, body: String) : Message
  + Caller: **BankStaff**
  + Layer: MessageController
  + Notes: Creates an SMS message template. Automatically derives the /smsParts attribute from text length and encoding.
* validateMessage(messageId: String) : ValidationResult
  + Caller: **BankStaff (pre-send) / System (automated checks)**
  + Layer: ComplianceController / MessageValidator
  + Notes: Checks message content against compliance and technical rules (e.g., forbidden terms, encoding, or attachments).
* scheduleMessage(messageId: String, scheduledAt: DateTime) : boolean
  + Caller: **BankStaff**
  + Layer: SchedulingController
  + Notes: Optionally overrides a campaign’s schedule for individual messages; must remain within campaign validity dates.
* getMessageDeliveryRate(messageId: String) : Double
  + Caller: **BankStaff / ReportingController**
  + Layer: MetricsService
  + Notes: Derived attribute: deliveryRate = successfulDeliveries / totalDeliveries. Used for calculating delivery success rate and performance analysis.

**Delivery Operations :**

* sendMessage(messageId: String, customerId: String) : DeliveryRef
  + Caller: **DispatchController (system)**
  + Layer: DeliveryController → GatewayAdapter → (EmailGateway/SMSGateway)
  + Notes: Initiates message delivery asynchronously. GatewayAdapter selects the appropriate provider and waits for acknowledgment (ACK/NACK). Enqueues retry logic when necessary.
* retryDelivery(deliveryId: String) : boolean
  + Caller: **DeliveryController / ChannelAdmin (manual)**
  + Layer: DeliveryController
  + Throws: MaxAttemptsReachedException
  + Notes: Performs redelivery for failed attempts, respecting the policy attempts ≤ 5.
* updateDeliveryStatus(deliveryId: String, status: MessageStatus, providerInfo?: Map) : void
  + Caller: **EmailGateway / SMSGateway (via callback)**
  + Layer: CallbackEndpoint → DeliveryController → Delivery (Entity)
  + Notes: Updates delivery information based on provider feedback. Must be idempotent and log all changes to the audit trail.
* trackDelivery(deliveryId: String) : DeliveryDetailDTO
  + Caller: **BankStaff / Customer (limited view)**
  + Layer: DeliveryController / Portal
  + Notes: Retrieves delivery attempt history, timestamps, and provider response codes.

**Customer Operations :**

* updatePreferences(customerId: String, emailOpt: Boolean, smsOpt: Boolean) : boolean
  + Caller: **Customer Portal**
  + Layer: CustomerController → Customer (Entity)
  + Notes: Updates the customer’s notification preferences. Triggers unsubscribe logic and logs the event for compliance.
* viewMessageHistory(customerId: String, page: Int, size: Int) : Page<MessageSummary>
  + Caller: **Customer**
  + Layer: CustomerPortal → ReportingService
  + Notes: Displays the customer’s received messages with pagination and privacy filtering.
* reportIssue(customerId: String, deliveryId: String, issueType: Enum, description?: String) : TicketRef
  + Caller: **Customer**
  + Layer: SupportController
  + Notes: Allows customers to report delivery or content issues. Creates a support ticket linked to the affected delivery.

**Channel Operations :**

* configureEmailGateway(gatewayConfig: GatewayConfig) : GatewayRef
  + Caller: **ChannelAdmin**
  + Layer: AdminConsole → ChannelAdminController
  + Notes: Configures the email provider connection and registers callback URLs with the gateway for delivery notifications.
* configureSMSGateway(gatewayConfig: GatewayConfig) : GatewayRef
  + Caller: **ChannelAdmin**
  + **Layer: AdminConsole → ChannelAdminController**
  + **Notes:** Sets up SMS provider integration and registers callback endpoints for delivery reports.
* monitorChannelHealth() : ChannelHealthDTO
  + Caller: **ChannelAdmin / SystemAdmin**
  + Layer: MonitoringService
  + Notes: Monitors provider API latency, error rates, and last synchronization time. Checks gateway availability and performance.
* adjustRetryPolicy(maxAttempts: Int, retryIntervalSeconds: Int) : boolean
  + Caller: **ChannelAdmin**
  + Layer: ChannelPolicyService
  + Notes: Updates delivery retry rules for the messaging subsystem. Changes are versioned and audit logged.

**Administrative Operations :**

* createUser(userType: Enum, userData: UserDTO) : UserRef
  + Caller: **SystemAdmin**
  + Layer: UserAdminController
  + **Notes:** Creates a new system user with predefined role and permissions.
* assignRole(userId: String, role: String) : boolean
  + Caller: **SystemAdmin**
  + **Layer: UserAdminController**
  + Notes: Assigns or revokes system permissions. Role-based access control (RBAC) enforced; all changes audit logged.
* generateAuditLog(startDate: Date, endDate: Date, filters?: Map) : LogExportRef
  + Caller: **SystemAdmin / Compliance Auditor**
  + Layer: AuditController
  + Notes: Exports an immutable audit trail within a selected period. Accessible in read-only mode for the Compliance Auditor.
* configureSustainabilityMetrics(params: SustainabilityConfig) : boolean
  + Caller: **SystemAdmin / ChannelAdmin**
  + Layer: SustainabilityController
  + Notes: Defines sustainability parameters such as energy estimation model and reporting granularity (per-message or per-campaign).

**External Provider Operations (Email/SMS Gateways) :**

* sendEmail(apiKey: String, payload: EmailPayload) : ProviderResponse
  + **Caller:** GatewayAdapter (system)
  + **Layer:** External API (EmailGateway)
  + **Notes:** Delivers email messages through the provider API. Returns providerDeliveryId, status, and metadata.
* sendSMS(apiKey: String, payload: SMSPayload) : ProviderResponse
  + **Caller:** GatewayAdapter (system)
  + **Layer:** External API (SMSGateway)
  + **Notes:** Sends SMS messages via external gateway; returns delivery reference and initial status.
* getDeliveryStatus(providerDeliveryId: String) : ProviderDeliveryStatus
  + **Caller:** GatewayAdapter / MonitoringService
  + **Layer:** External API
  + **Notes:** Queries provider status if asynchronous callbacks are not available.
* registerCallback(callbackUrl: String) : RegistrationResult
  + **Caller:** SystemAdmin / ChannelAdmin (during setup)
  + **Layer:** External API (Email/SMS Gateway)
  + **Notes:** Registers callback URLs used by providers to notify the system about delivery updates.
* POST /callbacks/delivery (provider → system)
  + **Caller:** EmailGateway / SMSGateway
  + **Layer:** CallbackEndpoint → DeliveryController.updateDeliveryStatus
  + **Notes:** Provider-initiated callback that communicates final delivery results or failure events.

**Compliance and Audit Operations :**

* viewAuditLog(startDate: Date, endDate: Date, filters?: Map) : AuditPage
  + **Caller:** Compliance Auditor / SystemAdmin
  + **Layer:** AuditorPortal → AuditController
  + **Notes:** Displays the system’s audit trail filtered by user, campaign, or event type. Data is immutable and non-editable.
* requestImmutableExport(startDate: Date, endDate: Date, format: Enum{PDF, CSV}) : ExportJobRef
  + **Caller:** Compliance Auditor / SystemAdmin
  + **Layer:** AuditController → ExportService
  + **Notes:** Generates a digitally signed export of the audit log for compliance verification. The resulting file is stored in non-rewritable (WORM) storage.
* verifyExportSignature(exportRef: String) : VerificationResult
  + **Caller:** Compliance Auditor
  + **Layer:** ExportService
  + **Notes:** Verifies authenticity and integrity of exported audit data through cryptographic signature validation.
* annotateAuditFinding(exportRef: String, finding: String) : void (optional)
  + **Caller:** Compliance Auditor
  + **Layer:** AuditorPortal → AuditController
  + **Notes:** Allows auditors to record findings or comments linked to an immutable export without altering the original logs.

**Relationships & multiplicity (recap).**

* BankStaff (1) → (0..\*) Campaign (creates)
* Campaign (1) → (1..\*) Message
* Message (1) → (0..\*) Delivery → (1) Customer (association class)
* Campaign (0..1) → SustainabilityMetric
* Qualifier: Customer → Message [messageSeq]

**Sustainability Integration (Artifact/Process/Application)**

* Artifact: explicit **SustainabilityMetric** linked to **Campaign**.
* Process: auto-estimation after dispatch based on channel and volume.
* Application: dashboards comparing kWh/CO₂ across channels and time.

**Task 2 - Use-case Model(s)**

**Actors (with generalization)**

* **User (actor, abstract)**  
  ↳ **Customer**  
  ↳ **Staff (actor)** → **BankStaff**, **Admin (actor)** → **ChannelAdmin**, **SystemAdmin**
* **External Systems:** **Email Gateway**, **SMS Gateway**
* **Compliance Auditor**

Actor generalization is used to avoid duplication of common associations (e.g., all Users authenticate).

**Use Cases and Key Relationships**

1. **Authenticate User** (included by many)
2. **Manage Users & Roles** (SystemAdmin)
3. **Configure Channel** (ChannelAdmin)
   * **Extend:** **Configure Retry Policy** (optional feature)
4. **Create Campaign** (BankStaff)
   * **Include:** **Log Audit Event**
5. **Create Email Message** (BankStaff) **- Generalization** of **Create Message**
6. **Create SMS Message** (BankStaff) **- Generalization** of **Create Message**
7. **Schedule Campaign** (BankStaff)
   * **Extend:** **Define A/B Variants** (optional, at extension point “before dispatch”)
   * **Include:** **Validate Scheduling Constraints**
8. **Dispatch Campaign** (system-initiated or manual)
   * **Include:** **Compute Sustainability Metrics**
   * **Include:** **Update Delivery Status**
   * **Extend:** **Apply Retry Policy on Failure**
9. **Receive Delivery Callback** (from Email/SMS Gateway) - updates Delivery
10. **Monitor Deliveries & KPIs** (BankStaff)
11. **View Audit Log** (Compliance Auditor, SystemAdmin)
12. **Export Compliance Report** (SystemAdmin/Compliance Auditor)

**Generalization (two types) present**

* **Actor generalization:** User → {Customer, Staff, Admin}, and Admin → {SystemAdmin, ChannelAdmin}.
* **Use-case generalization:** **Create Message** → {**Create Email Message**, **Create SMS Message**}.

**Include / Extend :**

* **Include:** common, mandatory sub-behaviors  
  Authenticate User, Log Audit Event, Validate Scheduling Constraints, Compute Sustainability Metrics, Update Delivery Status.
* **Extend:** optional or conditional extras  
  Define A/B Variants extends Schedule Campaign; Apply Retry Policy on Failure extends Dispatch Campaign; Configure Retry Policy extends Configure Channel.

### Use-Case Descriptions

Below are concise templates (goal, actors, preconditions, main/alternative flows, postconditions):

|  |  |
| --- | --- |
| **Use-Case ID** | UC-01 |
| **Name** | Authenticate User |
| **Primary Actor(s)** | User (Abstract → Customer, BankStaff, ChannelAdmin, SystemAdmin) |
| **Supporting Actor(s)** | System Authentication Service |
| **Goal** | Verify user credentials and establish an authenticated session. |
| **Preconditions** | User exists; account active. Credentials provided. |
| **Main Flow** | 1. User enters username/password. 2. System validates credentials. 3. System generates session token. 4. Audit event logged. |
| **Alternative Flow** | A1. Invalid credentials → deny access + log failure. A2 Locked account → show lockout message. |
| **Postconditions** | Either an authenticated session exists, or failure is recorded. |
| **Extensions/Includes** | - |

|  |  |
| --- | --- |
| **Use-Case ID** | UC-02 |
| **Name** | Manage Users & Roles |
| **Primary Actor(s)** | SystemAdmin |
| **Supporting Actor(s)** | UserAdminController, AuditController |
| **Goal** | Allow SystemAdmin to create, update, deactivate users and assign roles. |
| **Preconditions** | SystemAdmin authenticated (UC-01). |
| **Main Flow** | 1. Admin opens user admin panel. 2️. Selects “Create” or “Edit.” 3️. Enters user info + role(s). 4️. System validates data. 5️. User saved. 6️. Audit event recorded. |
| **Alternative Flow** | A1 Duplicate username → reject. A2 Invalid role → error. |
| **Postconditions** | User successfully created/updated/deactivated. |
| **Extensions/Includes** | <<include>> Log Audit Event |

|  |  |
| --- | --- |
| **Use-Case ID** | UC-03 |
| **Name** | Configure Channel |
| **Primary Actor(s)** | ChannelAdmin |
| **Supporting Actor(s)** | EmailGateway, SMSGateway, ChannelAdminController |
| **Goal** | Configure Email/SMS gateway connections for message delivery. |
| **Preconditions** | ChannelAdmin authenticated. |
| **Main Flow** | 1️. Admin enters API keys, callback URL. 2️. System saves encrypted credentials. 3️. System registers callback URL with provider. 4️. System performs connection test. 5️. Audit entry logged. |
| **Alternative Flow** | A1 Invalid gateway config → reject. A2 Provider unreachable → error. |
| **Postconditions** | Channel configuration active and verified. |
| **Extensions/Includes** | <<include>> Log Audit Event  <<extend>> Configure Retry Policy |

|  |  |
| --- | --- |
| **Use-Case ID** | UC-04 |
| **Name** | Create Campaign |
| **Primary Actor(s)** | BankStaff |
| **Supporting Actor(s)** | CampaignController, AuditController |
| **Goal** | Define a new campaign. |
| **Preconditions** | BankStaff authenticated. |
| **Main Flow** | 1️. BankStaff enters campaign name + dates. 2️. System validates time window. 3️. Campaign created in state = DRAFT. 4️. Audit event recorded. |
| **Alternative Flow** | A1 Invalid timeline → error. |
| **Postconditions** | Campaign exists in DRAFT with no messages yet. |
| **Extensions/Includes** | <<include>> Log Audit Event |

|  |  |
| --- | --- |
| **Use-Case ID** | **UC-05** |
| **Name** | **Create Email Message** (child of Create Message) |
| **Primary Actor(s)** | BankStaff |
| **Supporting Actor(s)** | MessageController |
| **Goal** | Create a new email content template. |
| **Preconditions** | Campaign exists in DRAFT. |
| **Main Flow** | 1️. Select campaign. 2️. Enter subject, body, attachments. 3️. <<include>> Validate Message Content. 4️. System saves EmailMessage object. 5️. Audit logged. |
| **Alternative Flow** | A1 Invalid attachment → rejection. A2 Forbidden terms → validation fail. |
| **Postconditions** | EmailMessage linked to campaign. |
| **Extensions/Includes** | <<include>> Validate Message Content  <<extend>> Attach Files |

|  |  |
| --- | --- |
| **Use-Case ID** | **UC-06** |
| **Name** | **Create SMS Message** (child of Create Message) |
| **Primary Actor(s)** | BankStaff |
| **Supporting Actor(s)** | MessageController |
| **Goal** | Create SMS text content template. |
| **Preconditions** | BankStaff authenticated; campaign in DRAFT. |
| **Main Flow** | 1️. Enter SMS text. 2️. <<include>> Validate Message Content. 3️. System checks length + derives smsParts. 4️. Message saved. |
| **Alternative Flow** | A1 Excessive length → reject or auto-split. A2 Encoding invalid → error. |
| **Postconditions** | SMSMessage stored and linked to campaign. |
| **Extensions/Includes** | <<include>> Validate Message Content |

|  |  |
| --- | --- |
| **Use-Case ID** | UC-07 |
| **Name** | Schedule Campaign |
| **Primary Actor(s)** | BankStaff |
| **Supporting Actor(s)** | SchedulingController |
| **Goal** | Schedule a campaign for future delivery. |
| **Preconditions** | Campaign has ≥1 message. |
| **Main Flow** | 1️. Select schedule time. 2️. Select recipients. 3️. <<include>> Validate Scheduling Constraints. 4️. ScheduledAt set; Status = SCHEDULED. 5️. Audit event recorded. |
| **Alternative Flow** | A1 Invalid schedule → reject. A2 No recipients → error. |
| **Postconditions** | Campaign is ready for dispatch. |
| **Extensions/Includes** | <<include>> Validate Scheduling Constraints <<include>> Log Audit Event  <<extend>> Define A/B Variants |

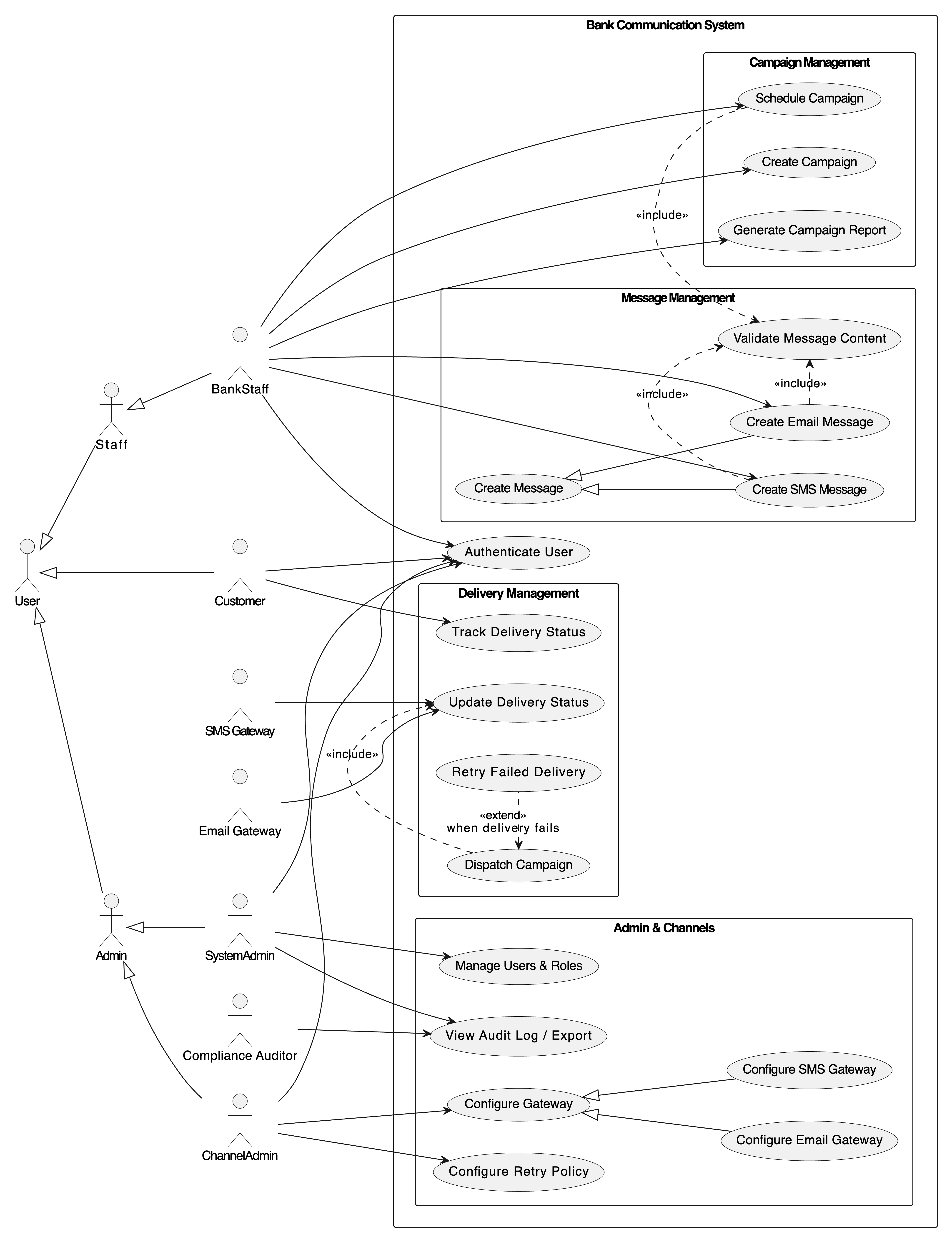
|  |  |
| --- | --- |
| **Use-Case ID** | UC-08 |
| **Name** | Dispatch Campaign |
| **Primary Actor(s)** | System (Auto Job or Manual Trigger) |
| **Supporting Actor(s)** | EmailGateway, SMSGateway, DeliveryController |
| **Goal** | Send all messages in a campaign to target customers. |
| **Preconditions** | Campaign = SCHEDULED; channels configured. |
| **Main Flow** | 1️. Dispatcher fetches recipients. 2️. Determines channel (Email/SMS). 3️. Sends message via GatewayAdapter. 4️. <<include>> Update Delivery Status. 5️. <<include>> Compute Sustainability Metrics. |
| **Alternative Flow** | A1 Gateway returns failure → <<extend>> Apply Retry Policy on Failure. A2 Partial successes → system reports. |
| **Postconditions** | Delivery records exist (SENT / DELIVERED / FAILED). |
| **Extensions/Includes** | <<include>> Update Delivery Status <<include>> Compute Sustainability Metrics  <<extend>> Apply Retry Policy on Failure |

|  |  |
| --- | --- |
| **Use-Case ID** | UC-09 |
| **Name** | Receive Delivery Callback |
| **Primary Actor(s)** | EmailGateway / SMSGateway |
| **Supporting Actor(s)** | CallbackEndpoint, DeliveryController |
| **Goal** | Process asynchronous delivery updates from providers. |
| **Preconditions** | Callback URL registered; provider sends POST. |
| **Main Flow** | 1️. Provider sends callback payload. 2️. System validates authenticity. 3️. DeliveryController updates Delivery status. 4️. Audit event logged. |
| **Alternative Flow** | A1 Invalid signature → reject callback. A2 Missing fields → system logs error. |
| **Postconditions** | Delivery status updated consistently (idempotent). |
| **Extensions/Includes** | <<include>> Update Delivery Status |

|  |  |
| --- | --- |
| **Use-Case ID** | UC-10 |
| **Name** | Monitor Deliveries & KPIs |
| **Primary Actor(s)** | BankStaff |
| **Supporting Actor(s)** | ReportingService |
| **Goal** | View campaign performance metrics and delivery details. |
| **Preconditions** | Deliveries exist for the campaign. |
| **Main Flow** | 1️. Open dashboard. 2️. System computes deliveryRate (derived). 3️. User filters by status/channel. 4️. Drill-down into delivery events. |
| **Alternative Flow** | A1 No data → show info message. |
| **Postconditions** | No state change; insights presented. |
| **Extensions/Includes** | - |

|  |  |
| --- | --- |
| **Use-Case ID** | UC-11 |
| **Name** | View Audit Log / Export Compliance Report |
| **Primary Actor(s)** | SystemAdmin, Compliance Auditor |
| **Supporting Actor(s)** | AuditController, ExportService |
| **Goal** | Allow administrators/auditors to review and export immutable audit logs. |
| **Preconditions** | User authenticated; audit data available. |
| **Main Flow** | 1️. Actor selects filters/date range. 2️. System displays matching logs. 3️. Actor chooses export format. 4️. System generates signed immutable export. 5️. Export stored + checksum logged. |
| **Alternative Flow** | A1 No logs found → empty export. A2 Signature verification fails → error. |
| **Postconditions** | Export file ready; audit of audit (!) logged. |
| **Extensions/Includes** | <<include>> Log Audit Event |

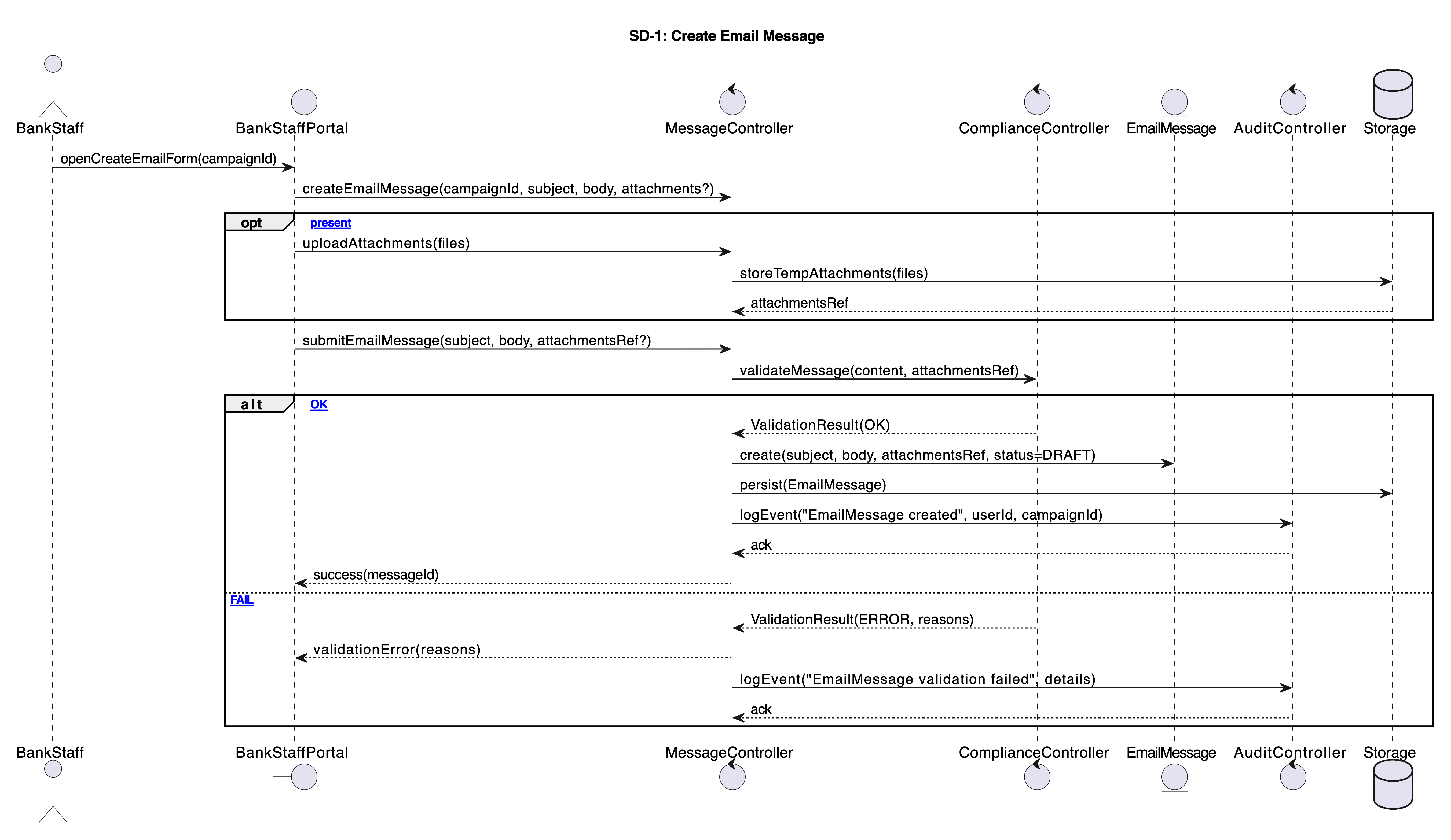
**Use-Case Diagram**



**Task 3 - Three Sequence Diagrams**

**3.1 Sequence Diagram 1 - Create Email Message** (Validation + Optional Attachments)  
  
**Frames used:** opt (attachments present) and alt (validation OK vs FAIL).  
**Operations surfaced:** createEmailMessage, uploadAttachments, validateMessage, EmailMessage.create, Storage.persist, AuditController.logEvent.

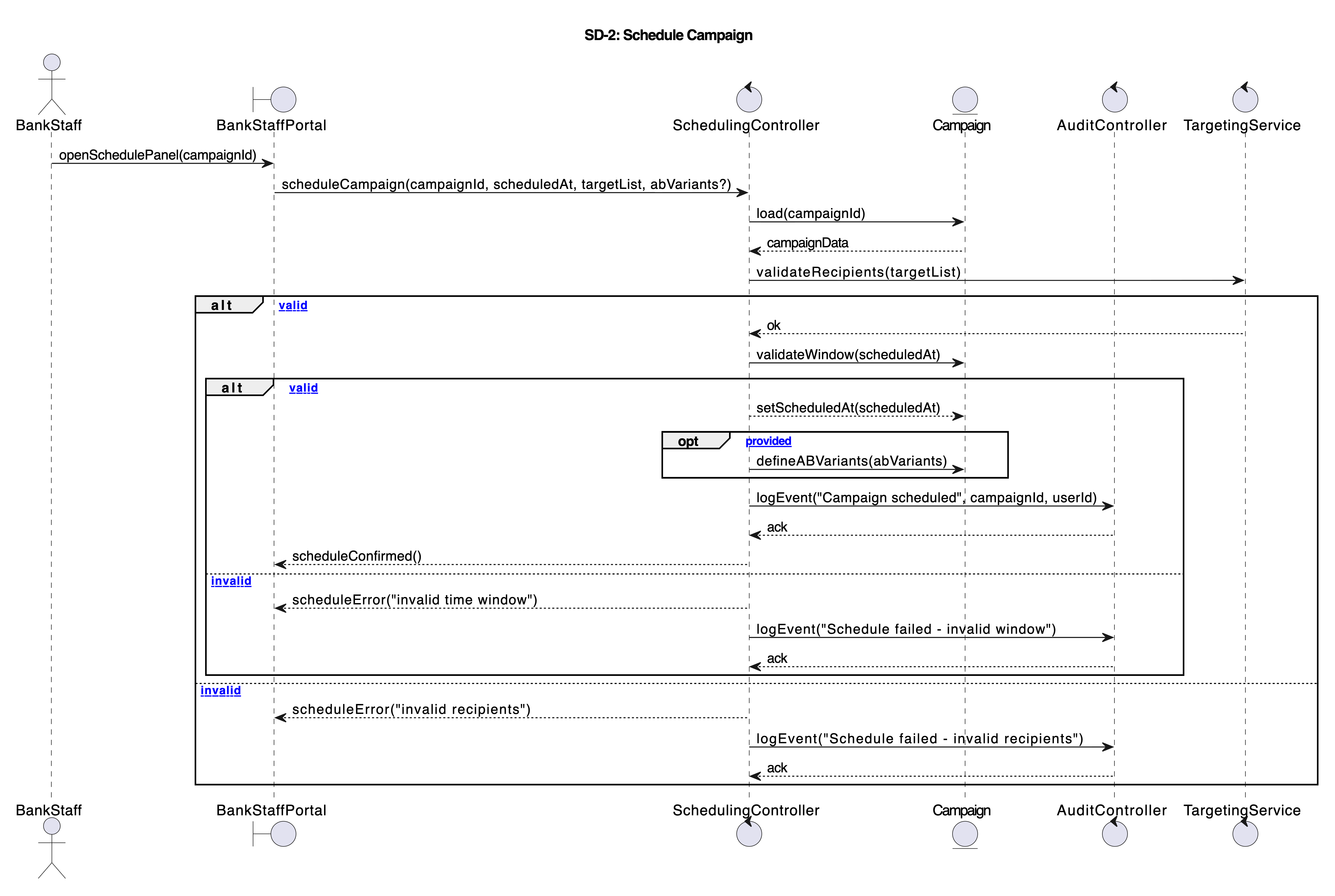
This diagram shows UI → controller → compliance checks and optional attachments (include/extend behavior), and it is good to document Validate Message Content include and Attach Files extend.

****

**3.2 Sequence Diagram 2 - Schedule Campaign (with A/B Variants and Validation)**

**Frames used:** alt **(target validation and time window valid/invalid nested),** opt **(A/B variants optional block).  
Operations surfaced:** scheduleCampaign**,** validateRecipients **(TargetingService),** Campaign.validateWindow**,** Campaign.setScheduledAt**,** defineABVariants**,** AuditController.logEvent**.**

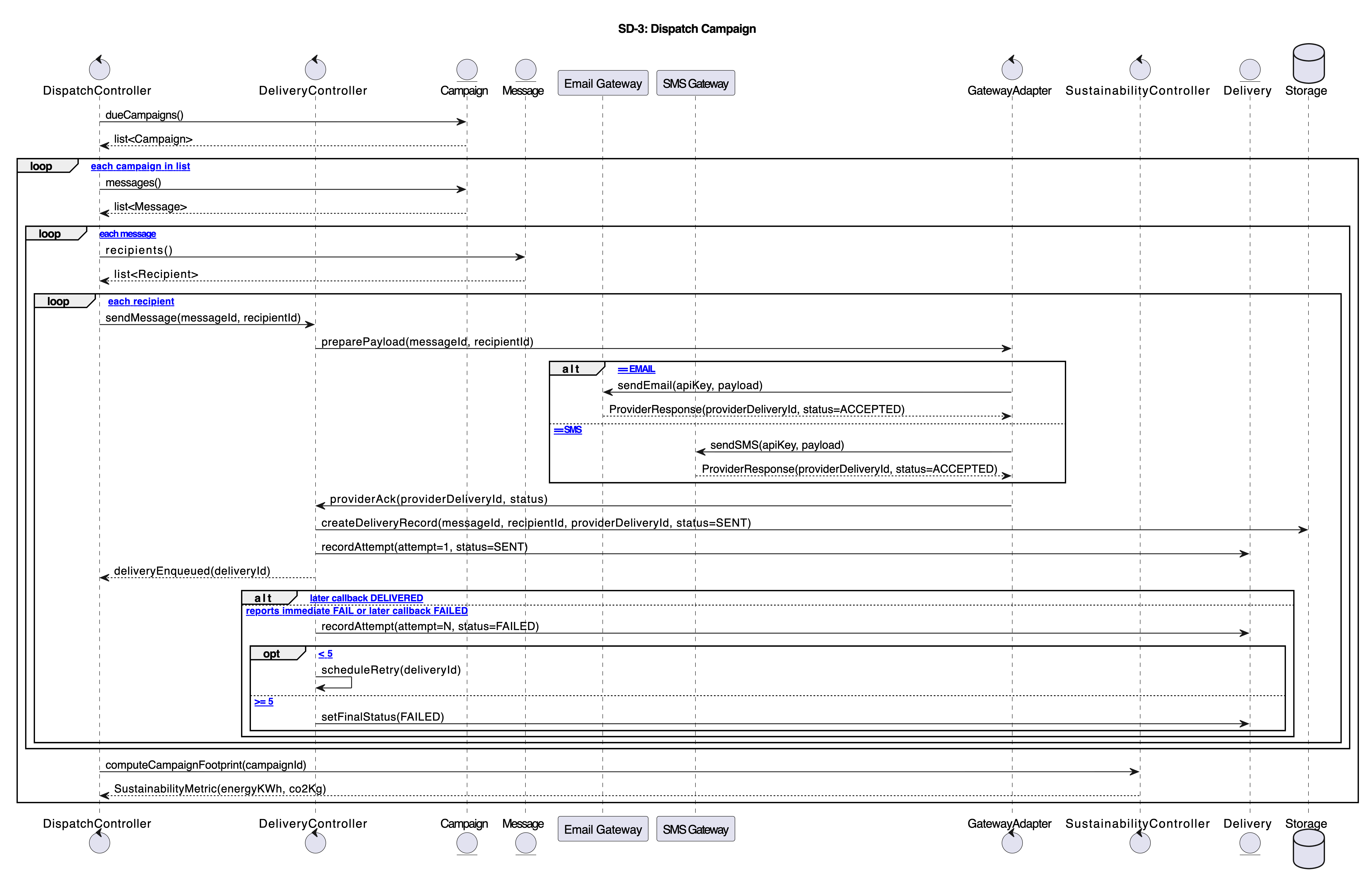
**This diagram models** Schedule Campaign **UC, shows mandatory validation includes and optional A/B extension.**



**3.3. Sequence Diagram 3 - Dispatch Campaign (Loop Over Campaigns/Messages/Recipients, Retry Logic, Sustainability Computation)**

**Frames used:** loop **(three nested loops, for campaigns/messages/recipients),** alt **(channel decision Email/SMS),** opt **(retry scheduling conditional).  
This diagram uses multiple frames (>= 2) and also demonstrates interaction with external gateways and sustainability computation.  
Operations surfaced:** dueCampaigns**,** messages**,** recipients**,** sendMessage**,** preparePayload**,** sendEmail**/**sendSMS **(gateway ops),** createDeliveryRecord**,** recordAttempt**,** scheduleRetry**,** computeCampaignFootprint**.**

**This diagram fully models the heavy, asynchronous dispatch logic, retry policy (**attempts ≤ 5**), and sustainability metric computation included after dispatch. It also shows GatewayAdapter as intermediary for provider calls.**



**Task 4 - Modelling Environment & Rationale**

**For designing the use-case diagrams and sequence diagrams, we used draw.io (diagrams.net) together with its PlantUML integration. This environment provided a balanced combination of visual modelling convenience and text-based modelling precision, which was necessary for representing a system with multiple actors, generalizations, include/extend relations, and complex sequence logic.**

**Reasons for choosing draw.io + PlantUML**

1. **Free, Browser-Based, and Collaborative**  
   Draw.io works entirely in the browser, does not require installation, and supports real-time collaboration via Google Drive/OneDrive. This made the modelling workflow accessible from anywhere and consistent with team-based assignment requirements.
2. **Easy UML Shapes & Clean Layout Control**  
   Draw.io provides a rich built-in UML shape set, allowing clear visual organization of actors, use cases, and system boundaries. This helps avoid visual clutter in diagrams that include many relationships and external systems.
3. **PlantUML Import & Round-Trip Editing**  
   The PlantUML plug-in supports both importing .puml text and visually rendering it. This enables round-trip modelling:
   * architect writes/edit diagrams textually (versionable),
   * draw.io renders them visually,
   * diagrams can be refined back in text form.  
     This was essential for keeping use-case and sequence diagrams consistent as the system evolved.
4. **Version Control Compatibility (GitHub)**  
   Since PlantUML is stored as plain text, it integrates smoothly with GitHub/GitLab. Every change in the models becomes traceable, enabling proper history tracking and collaborative editing.
5. **Export Capabilities (PNG, PDF, XML)**  
   The tool supports exporting diagrams into PNG, SVG, PDF, and even .drawio XML files required for academic submission and documentation. This ensured that the final report satisfied the formatting constraints.
6. **Support for Sequence Diagram Authoring Tools**  
   Alongside draw.io + PlantUML, we used **WebSequenceDiagrams** and **SequenceDiagram.org** as quick syntax checkers to validate lifelines, frames (alt/opt/loop/par), and call structures. These tools allowed rapid iteration on textual sequence logic before rendering the final version in draw.io.  
   This approach follows the tool recommendations demonstrated in the course’s practical sessions.

For these reasons, it was chosen as the primary environment for producing all use-case and sequence diagrams in this assignment.

**Task 5 – Team work evidence**

**Task 6 – Generative AI capabilities**

This task explores the capability of three Generative AI models – ChatGPT, Claude 4.5, and Google Gemini Advanced – to support modelling activities for the Banking – Communication Services System (Email & SMS).

The objective was to find out if AI could help create use-case and sequence diagrams comparing with those created by human in Tasks 2 and 3.

To ensure a fair comparison under reasonable accessibility constraints, all three models were utilized in their free versions.

1. **1. Tools Used and Justification**

|  |  |
| --- | --- |
| **AI Model** | **Why It Was Selected** |
| ChatGPT (GPT-5 free version) | Strong reasoning abilities, good at structured outputs (PlantUML/XML), widely used in modelling research. |
| Claude 3.5 Sonnet (free version) | Best natural language understanding among all three tools; often generates diagrams directly instead of code. |
| Google Gemini Advanced (free version) | Provides long explanations and sometimes code; included for diversity and comparison. |

When AI code in PlantUML and Draw.io failed, a specific modelling tool-Eraser.io was utilized as a backup renderer.

This also made it easier to assess how "strict" or "forgiving" the AI techniques are when generating UML that is syntactically correct.

**2. Prompts Used**

Each AI model was given the same instruction set (repeated here for transparency):

**Prompt for Use-Case Model**

Generate a full UML use-case model for a Banking – Communication Services System (Email & SMS). Include at least 4 actors, 10+ use-cases, include/include/extend/generalization.

After getting non relatable diagram provided some details of the model from Assignment 1 and asked again to generate model.

**Prompt for Sequence Diagrams**

Provide at three sequence diagrams for the system described in Task #1.

**Error-Handling Prompt (used when diagrams failed to render)**

There is error in the given code please generate the code again resolving the error.  
Error:<pasted error>

1. **3. AI-Generated Models**

**3.1 Use-Case Models**

|  |  |  |  |
| --- | --- | --- | --- |
| **AI Model** | **Accuracy** | **Issues** | **Notes** |
| **ChatGPT** | High | Initial PlantUML had errors; needed refinement prompts | Provided XML + PlantUML; |
| **Claude** | Very high | None; produced direct rendered diagrams, not code | Most complete and correct model |
| **Gemini** | Low | Ignored system description; invented use cases | Vague, ambiguous; least aligned with Task 1 |

ChatGPT

A computer screen shot of a computer

Description automatically generated

Claude AI

A diagram of a diagram

Description automatically generated

Gemini

A screenshot of a computer screen

Description automatically generated

**Use Case Model after providing each and every small details about relationship between actors, roles, classes of the system.**

ChatGPT

A diagram of a network

Description automatically generated with medium confidence

Gemini

A computer screen shot of a diagram

Description automatically generated

ClaudeAI

A diagram of a diagram

Description automatically generated with medium confidence

**3.2 Sequence Diagrams**

|  |  |  |  |
| --- | --- | --- | --- |
| **AI Model** | **Quantity** | **Validity** | **Notes** |
| **ChatGPT** | 3/3 | 2 diagrams correct, 1 had syntax errors | Closest to original diagrams |
| **Claude** | 1/3 initially, 3/3 after follow-up | Some diagrams incomplete on first attempt | Needed re-asking to generate missing diagrams |
| **Gemini** | 1/3 initially, 3/3 after follow-up | 2 diagrams correct, 1 had syntax errors | Required extra fixing; still less reliable |

All three diagrams generated by each AI Models:

**ChatGPT**

A diagram of a company

Description automatically generated

Monitor Campaign Performance & Sustainability Metrics

A diagram of a software project

Description automatically generated

Configure Messaging Channel

A diagram of a software application

Description automatically generated with medium confidence

Schedule and Send Message

**ClaudeAI**

**A close-up of a blueprint

Description automatically generated**

Create and Schedule Campaign

**A close-up of a document

Description automatically generated**

Message Delivery with Retry Logic

**A close-up of a document

Description automatically generated**

View Sustainability Dashboard

**Gemini**

Marketing Campaign Creation and Multi-Delivery

**A screenshot of a computer

Description automatically generated**

**A screenshot of a diagram

Description automatically generated**

**A diagram of a diagram

Description automatically generated**

1. **4. Comparison with Manually Designed Models**

**4.1 Use-Case Model Differences**

|  |  |  |
| --- | --- | --- |
| **Aspect** | **Manual Model** | **AI Models** |
| **Correct Actors** | All actors correctly identified | ChatGPT & Claude OK; Gemini invented or omitted |
| **Use-Case Completeness** | ≥10 use-cases | ChatGPT (OK), Claude (OK), Gemini (incomplete) |
| **Relationships (include/extend)** | Accurate | ChatGPT OK after revision; Claude OK; Gemini inconsistent |
| **Generalization** | Correct (e.g., User roles) | ChatGPT & Claude included; Gemini missing |
| **Clarity** | Clean, aligned visually | AI diagrams lacked consistent layout (except Claude) |

**Additional Insight:**

**Claude** was the only model that produced visual diagrams directly without relying solely on PlantUML.

**4.2 Sequence Diagram Differences**

|  |  |  |
| --- | --- | --- |
| **Aspect** | **Manual Model** | **AI Models** |
| **Frames (loop/alt/opt)** | Used correctly | ChatGPT used consistently; Claude sometimes missed; Gemini mostly missed |
| **Actors and Lifelines** | Well aligned with Task 1 | Gemini confused system components; Claude accurate after follow-up |
| **Delivery Logic** | Retry loops and callback events included | ChatGPT replicated retry logic well; Claude partially; Gemini inconsistent |
| **Sustainability Metric Flow** | Included | ChatGPT included; Claude included; Gemini missed details |

**Additional Insight:**

**ChatGPT and ClaudeAI** produced the closest overall diagrams when compared visually to team’s original diagrams.

**5. Critical Assessment of Each AI Tool**

**5.1 ChatGPT (GPT-5)**

Strengths

* Best balance of correctness and consistency
* All 3 sequence diagrams valid on first attempt
* Responded well to refinement prompts
* Able to generate both XML and PlantUML for multiple platforms

Weaknesses

* Initial code occasionally contained syntax errors
* Needed explicit reminders to include generalization and relationships
* Does not auto-render diagrams — requires external tool

**5.2 Claude 3.5 Sonnet**

Strengths

* Best natural understanding of system context
* Generated diagrams visually, not only code
* Produced the most accurate use-case model
* Very readable explanations and structure

Weaknesses

* Forgot to generate 2 out of 3 sequence diagrams until asked again
* Sequence diagram details sometimes missing (frames, activation bars)
* Slightly vague wording unless given strict prompts

**5.3 Google Gemini**

Strengths

* Fast generation
* Occasionally produced correct UML code (2 out of 3 sequences)
* Good for alternative creative suggestions

Weaknesses

* Least aligned with the system description
* Produced unrelated or imaginary use-cases
* Code often contained syntax errors and could not be rendered
* Misinterpreted requirements even when repeated
* Provided vague or generic system behaviors

**6. Additional Comparison**

Model Accuracy Ranking

1. Claude AI (best for use-case model)
2. ChatGPT (best for sequence diagrams)
3. Gemini (least accurate)

Reliability Ranking

1. ChatGPT – most reliable across all outputs
2. Claude – reliable but occasionally forgetful
3. Gemini – inconsistent and sometimes incorrect

**Rendering Compatibility**

|  |  |  |  |
| --- | --- | --- | --- |
| **AI** | **PlantUML Rendering** | **Draw.io Rendering** | **Eraser Rendering** |
| **ChatGPT** | Sometimes errors until corrected | Inconsistent | Works fully |
| **Claude** | Works (XML/diagram direct) | Works | Works |
| **Gemini** | Often fails | Fails for multiple diagrams | Eraser was the only tool that could render them |

**7. Did AI Improve the Models or Description?**

Yes — Partially.

AI tools helped clarify hidden aspects of the system such as:

* Missing alternative flows
* Expanded sustainability interactions
* Additional message delivery details
* Role boundaries in ChannelAdmin vs SystemAdmin
* More nuanced lifecycle of campaigns

However AI could NOT replace manual modelling because:

* Relationship semantics were often wrong
* Sequence logic lacked consistency
* Use-case dependencies needed manual correction
* Some diagrams were unusable without fixing (especially Gemini)

AI served well as a supporting assistant, not as a primary modeller.