CSE251 – " Test Report "

1. Project Title

Smart Reverse Vending Machine with Gamification System

3. Team Members

ID	Name
23101608	Yasseen Ahmed El-Sayed
23101594	Mohamed Mustafa Awad
23101545	Ahmed Khaled Said
23101599	Marawan Khaled Ahmed
23101899	Mohamed Adel Abdulrahman

UseCases

1. Manage Authentication

DESCRIPTION: THIS USE CASE HANDLES USER REGISTRATION, LOGIN, AND PROFILE MANAGEMENT. IT ENSURES SECURE ACCESS TO THE PLATFORM BY VALIDATING USER CREDENTIALS, CREATING NEW ACCOUNTS WITH VALID INFORMATION, AND ALLOWING USERS TO UPDATE THEIR PROFILE DETAILS. THE SYSTEM PREVENTS UNAUTHORIZED ACCESS AND MAINTAINS DATA INTEGRITY THROUGH VALIDATION CHECKS.

1.1 Login

- **TR-LOG-01:** Validate that user can login through filling email, password
- **TR-LOG-02:** Validate that user cannot login with invalid credentials
- **TR-LOG-03:** Validate that user cannot login with non-existent email
- **TR-LOG-04:** Validate that system locks account after 3 failed login attempts
- **TR-LOG-05:** Validate that system displays appropriate error messages for login failures

1.2 Register

- **TR-REG-01:** Validate that user can register with valid information
- **TR-REG-02:** Validate that user cannot register with existing email
- **TR-REG-03:** Validate that password meets minimum security requirements (8 characters, special character)
- **TR-REG-04:** Validate that all required fields (name, email, password) are filled
- **TR-REG-05:** Validate that system sends confirmation email after successful registration

1.3 Update Profile

- **TR-UPD-01:** Validate that registered user can update personal information (name, phone)
- **TR-UPD-02:** Validate that user cannot change email to already existing email
- **TR-UPD-03:** Validate that profile changes are saved and reflected immediately
- **TR-UPD-04:** Validate that user must re-authenticate when changing password
- **TR-UPD-05:** Validate that guest users cannot access profile update functionality

2. Manage Transaction

Description: This use case manages the complete recycling transaction process. It handles the recycling operation from start to finish, validates items through RVM hardware sensors, and

generates transaction receipts. The system ensures accurate item recognition, proper processing, and maintains transaction records for audit and user reference.

2.1 Process Recycling

- **TR-PROC-01:** Validate that system initiates recycling session when user starts transaction
- **TR-PROC-02:** Validate that system processes items sequentially and maintains order
- **TR-PROC-03:** Validate that system handles transaction completion successfully
- **TR-PROC-04:** Validate that system allows transaction cancellation before completion
- **TR-PROC-05:** Validate that system saves transaction data for registered users

2.2 Validate Items

- **TR-VAL-01:** Validate that system accepts valid recyclable bottles and cans
- **TR-VAL-02:** Validate that system rejects non-recyclable items
- **TR-VAL-03:** Validate that RVM hardware correctly identifies item type (plastic, aluminum, glass)
- **TR-VAL-04:** Validate that system verifies item condition (undamaged, clean)
- **TR-VAL-05:** Validate that system communicates validation results within 2 seconds

2.3 Generate receipt

- **TR-REC-01:** Validate that system generates receipt with transaction details (date, time, items, rewards”Note : points instead”)
- **TR-REC-02:** Validate that receipt includes unique transaction ID
- **TR-REC-03:** Validate that receipt prints within 3 seconds of transaction completion
- **TR-REC-04:** Validate that registered users can access digital receipt in transaction history
- **TR-REC-05:** Validate that system handles printer errors gracefully with appropriate error message

3. Manage QR Code

Description: This use case manages the generation, scanning, and validation of QR codes used for reward redemption. It handles QR code creation with encrypted data, enables cafeteria systems to scan codes, and validates authenticity and expiration. The system ensures secure reward redemption and prevents fraud through proper QR code lifecycle management

3.1 Generate QR Code

- **TR-GEN-01:** Validate that system generates unique QR code after successful reward redemption
- **TR-GEN-02:** Validate that QR code contains encrypted transaction data (user ID, amount, timestamp)
- **TR-GEN-03:** Validate that QR code is generated within 2 seconds of redemption request
- **TR-GEN-04:** Validate that QR code includes expiration time (24 hours from generation)
- **TR-GEN-05:** Validate that user receives QR code through multiple formats (image, PDF)

3.2 Scan QR Code

- **TR-SCN-01:** Validate that cafeteria system can successfully scan QR codes
- **TR-SCN-02:** Validate that system decodes QR data correctly
- **TR-SCN-03:** Validate that scanning process completes within 2 seconds
- **TR-SCN-04:** Validate that system handles damaged/unclear QR codes with error message
- **TR-SCN-05:** Validate that system displays QR code value and details after successful scan

3.3 Validate QR Code

- **TR-VQR-01:** Validate that system verifies QR code authenticity through encryption signature
- **TR-VQR-02:** Validate that system checks QR code has not been previously used
- **TR-VQR-03:** Validate that system rejects expired QR codes (older than 24 hours)
- **TR-VQR-04:** Validate that system marks QR code as "used" after successful redemption
- **TR-VQR-05:** Validate that invalid or tampered QR codes display appropriate error message

4. Manage Rewards

Description: This use case manages the reward system including issuing rewards to users, tracking reward status and history, and managing reward schemes and policies. It handles reward distribution based on points, monitors reward redemption lifecycle, and enables administrators to configure reward programs. The system ensures fair reward allocation and comprehensive tracking.

4.1 Issue rewards

- **TR-ISS-01:** Validate that system issues reward voucher after successful point redemption
- **TR-ISS-02:** Validate that reward value corresponds correctly to redeemed points (1 point = 1 EGP)
- **TR-ISS-03:** Validate that reward includes unique identifier and expiration date
- **TR-ISS-04:** Validate that system sends reward details to user via email and app notification
- **TR-ISS-05:** Validate that issued rewards are recorded in system database

4.2 track rewards

- **TR-TRK-01:** Validate that users can view all issued rewards in their account
- **TR-TRK-02:** Validate that system displays reward status (active, used, expired)
- **TR-TRK-03:** Validate that administrators can track all rewards across all users
- **TR-TRK-04:** Validate that system updates reward status in real-time when redeemed
- **TR-TRK-05:** Validate that reward history includes issue date, redemption date, and location

4.3 reward schemes

- **TR-SCH-01:** Validate that administrator can create new reward schemes with custom rules
- **TR-SCH-02:** Validate that system applies correct reward scheme based on user tier (bronze, silver, gold)
- **TR-SCH-03:** Validate that promotional reward multipliers are applied correctly (e.g., 2x points)
- **TR-SCH-04:** Validate that reward scheme changes take effect from specified date
- **TR-SCH-05:** Validate that system displays active reward schemes to users

5. Manage Points

Description: This use case handles the complete lifecycle of user reward points including accumulation, redemption, and transfer operations. It manages point calculations based on recycling activities, enables users to redeem points for rewards, and facilitates point transfers between users. The system ensures accurate point balance management and transaction integrity

5.1 Accumulate Points

- **TR-ACC-01:** Validate that points are added to registered user account after each transaction
- **TR-ACC-02:** Validate that point balance is updated in real-time
- **TR-ACC-03:** Validate that system calculates correct points based on item type and quantity
- **TR-ACC-04:** Validate that guest users cannot accumulate points
- **TR-ACC-05:** Validate that system prevents duplicate point addition for same transaction

5.2 Redeem Points

- **TR-RED-01:** Validate that user can redeem points when balance is sufficient
- **TR-RED-02:** Validate that system prevents redemption when point balance is insufficient
- **TR-RED-03:** Validate that points are deducted correctly after successful redemption
- **TR-RED-04:** Validate that system generates confirmation after redemption
- **TR-RED-05:** Validate that redemption transaction is recorded in user history

5.3 Apply Point Expiration

- **TR-EXP-01:** Validate that points expire after 12 months of inactivity if configured
- **TR-EXP-02:** Validate that users receive notification 30 days before points expiration
- **TR-EXP-03:** Validate that expired points are automatically deducted from balance
- **TR-EXP-04:** Validate that point expiration policy is clearly displayed to users
- **TR-EXP-05:** Validate that administrator can adjust point expiration policies globally

6. Manage Notifications

Description: This use case handles all system notifications and alerts sent to users, administrators, and maintenance staff. It manages notification delivery through multiple channels (email, SMS, in-app), tracks notification history, and ensures timely communication for important events such as successful transactions, reward redemption, system alerts, and maintenance requirements.

6.1 Send System Alert

- **TR-SYS-01:** Validate that administrators receive immediate alerts for system errors or failures
- **TR-SYS-02:** Validate that maintenance staff receive alerts for machine full or malfunction

- **TR-SYS-03:** Validate that alert severity levels (low, medium, high, critical) are correctly classified
- **TR-SYS-04:** Validate that critical alerts use multiple channels (email + SMS)
- **TR-SYS-05:** Validate that alert resolution updates are sent after issue is fixed

6.2 Send Transaction Notification

- **TR-TXN-01:** Validate that registered users receive notification after successful transaction completion
- **TR-TXN-02:** Validate that notification includes transaction summary (items, points earned, balance)
- **TR-TXN-03:** Validate that notification is sent within 5 seconds of transaction completion
- **TR-TXN-04:** Validate that users can opt-in/opt-out of transaction notifications in settings
- **TR-TXN-05:** Validate that notification delivery failures are logged and retried

6.3 Send Reward Notification

- **TR-RWD-01:** Validate that users receive notification when reward is successfully redeemed
- **TR-RWD-02:** Validate that notification includes QR code and redemption instructions
- **TR-RWD-03:** Validate that reminder notification is sent before reward/QR code expiration
- **TR-RWD-04:** Validate that users receive notification when reaching reward milestones
- **TR-RWD-05:** Validate that notification includes current point balance after redemption

11. TestCase Diagram

UseCase: Manage Authentication

1.1 Login

TEST CASE ID	TEST SCENARIO	TEST CASE DESCRIPTION	PRE-CONDITION	TEST STEPS	TEST DATA	EXPECTED RESULT	POST CONDITION	ACTUAL RESULT	STATUS (PASS/FAIL)
TC_Auth_01	TR-LOG-01: Validate that user can login through filling email, password	Enter **valid email** and **valid password**	1. User is registered in system2. Login page is loaded3. Database is accessible	1. Enter User Email2. Enter Password3. Click "Login" button	Email: **yasseen@rv m.comPassword**: **Pass word_123**	Successful login, redirected to dashboard	User dashboard is shown		
TC_Auth_02	TR-LOG-02: Validate that user can login through email, password via mobile app	Enter **valid email** and **valid password** via mobile app	1. Mobile app installed2. User registered3. Network active	1. Enter User Email2. Enter Password3. Tap "Login" button	Email: **ahmed@rv m.comPassword**: **Secur e@2025**	Successful login, home screen displayed	App home screen is shown		
TC_Auth_03	TR-LOG-02: Validate that user cannot login with invalid password	Enter **valid email** and **invalid password**	1. User account exists2. Login page loaded	1. Enter User Email2. Enter Password 3. Click "Login" button	Email: **yasseen@rv m.com** Password: **WrongPass 123**	Error message "Invalid email or password" shown	Login fails, error displayed		
TC_Auth_04	TR-LOG-02: Validate that user cannot login with **invalid email** and valid password	Enter **invalid email** and **valid password**	1. Email not registered 2. Login page loaded	1. Enter User Email 2. Enter Password 3. Click "Login" button	Email: **fake @test.com** Password: **P ass@123**	Error message "Invalid email or password" shown	Login fails, error displayed		
TC_Auth_05	TR-LOG-03: Validate that user cannot login with non-	Enter **unregistered email**	1. Email not in database2. Login page accessible	1. Enter User Email2. Enter Password3. Click	Email: **not1 @valid.com** Password: **Pass@ 123**	Error message "No account	Login denied		

TEST CASE ID	TEST SCENARIO	TEST CASE DESCRIPTION	PRE-CONDITION	TEST STEPS	TEST DATA	EXPECTED RESULT	POST CONDITION	ACTUAL RESULT	STATUS (PASS/FAIL)
	existent email			"Login" button		found with this email"			
TC_Auth_06	TR-LOG-03: Validate that user cannot login with non-existent email	Enter **email with typo**	1. Correct email exists 2. User types wrong domain	1. Enter User Email 2. Enter Password 3. Click "Login" button	Email: **user @gmail.co m** Password: **P ass@12**	Error message "No account found with this email"	Login denied		
TC_Auth_07	TR-LOG-04: Validate that system locks login 3 times with wrong password	Attempt login 3 times with wrong password	1. Account not locked 2. Failed attempt counter at 0	1. Enter valid email wrong password (attempt 1) 2. Repeat (attempt 2) 3. Repeat (attempt 3)	Email: **yass een@rvm.c om** Password: **Wrong123 (attempt 1-3)**	"Account temporarily locked. Try again in 15 minutes" displayed	Account locked for 15 minutes		
TC_Auth_08	TR-LOG-04: Validate that system locks account after 3 failed login attempts	2 failed attempts th en success on 3rd	1. Account active 2. Failed counter at 0	1. Wrong password (attempt 1) 2. Wrong password (attempt 2) 3. Correct password (attempt 3)	Email: **yass een@rvm.c om** Attempt 1-2: **Wrong12** 3 Attempt 3: **Pass@123**	Login successful, failed counter reset to 0	User logged in successfully		
TC_Auth_09	TR-LOG-05: Validate that system displays appropriate error messages	Submit em pty email and password	1. Login page loaded 2. All fields empty	1. Leave all fields blank 2. Click "Login" button	Email: **{Emp ty}** Password: **{Empty}**	Two error messages: "Email is required" "Password is required"	Both fields highlighted		

TEST CASE ID	TEST SCENARIO	TEST CASE DESCRIPTION	PRE-CONDITION	TEST STEPS	TEST DATA	EXPECTED RESULT	POST CONDITION	ACTUAL RESULT	STATUS (PASS/FAIL)
	for login failures								
TC_Auth_10	TR-LOG-05: Validate that system displays appropriate error messages for login failures	Network connection lost during login	1. Login page loaded 2. Network active	1. Enter valid credentials 2. Disconnect network	Email: **yass een@rvm.c om** Password: **P ass@123**	Error message "Connectio n failed. Please check your internet"	Login fails, retry option shown		

1.2 Register

TEST CASE ID	TEST SCENARIO	TEST CASE DESCRIPTION	PRE-CONDITION	TEST STEPS	TEST DATA	EXPECTED RESULT	POST CONDITION	ACTUAL RESULT	STATUS (PASS/FAIL)
TC_Auth_1	TR-REG-01: Validate that user can register with all required fields valid	Register with all **required fields valid**	1. Registration page loaded 2. Email not in database	1. Enter full name 2. Enter unique email 3. Enter phone 4. Enter password 5. Click "Register"	Name: **Ahmed Ali** Email: **ali.ahmed.new@rvm.com** Phone: 010123456789 Password: **Secure@12 5**	Success message "Account created successfully!"	Account created, confirmation email sent		
TC_Auth_12	TR-REG-01: Validate that user can register with valid information	Register and receive confirmatio n email	1. Email service working 2. Valid registration	1. Complete registration 2. Submit form 3. Check email within 1 minute	Email: **newuser@rvm.com** Password: **Tes t@123**	Confirmation email received with verification link	Email delivered within 60 seconds		
TC_Auth_13	TR-REG-02: Validate that user cannot register with existing email	Register with **alread y registered email**	1. Email exists in database 2. Registration page open	1. Enter name and phone 2. Enter existing email 3. Enter password 4. Click Register	Name: **New User** Email: **yasseen@rvm.com** Password: **Pass @123**	Error "Email already registered. Please login or use different email"	Registration blocked		
TC_Auth_14	TR-REG-02: Validate that user cannot register with existing email	Register with same **email, different case**	1. Email **"User@Test.com"** exists 2. Use lowercase email	1. Enter all fields 2. Use lowercase "user@test.com" 3. Try lowercase email 4. Submit form	Email: **user@test.com** (exists as User@Test.com) Password: **Test@123**	System detects duplicate, error "Email already exists"	Registration blocked		

TEST CASE ID	TEST SCENARIO	TEST CASE DESCRIPTION	PRE-CONDITION	TEST STEPS	TEST DATA	EXPECTED RESULT	POST CONDITION	ACTUAL RESULT	STATUS (PASS/FAIL)
TC_Auth_15	TR-REG-03: Validate that password meets minimum security requirements	Register with passwo rd too short	1. Registration form open 2. User filling fields	1. Enter all fields 2. Enter 6-Char password 3. Attempt submit	Password: **Pass12** (only 6 chars)	Error "Password must be at least 8 characters long"	Registration blocked		
TC_Auth_16	TR-REG-03: Validate that password meets minimum security requirements	Password la cks special character	1. User on registration page	1. Fill required fields 2. Enter 8-char password without special 3. Submit	Password: **Password12**	Error "Password must contain at least one special character (@, #, $, etc.)"	Form not submitted		
TC_Auth_17	TR-REG-04: Submit with missin g name field	Submit form with name empty & **all required fields are filled**	1. Registration form open	1. Leave name empty 2. Fill email and password 3. Click Register	Name: **{Empty}** Email: **m.com** Password: **Test@123**	"Name is required" error message, name field highlighted	Registration blocked		
TC_Auth_18	TR-REG-04: Validate that all required fields must be missing fields	Submit with empty fields	1. User on registration page	1. Fill only email 2. Leave name and password empty 3. Click Register	Name: **{Empty}** Email: **all: user@test.com** Password: **{Empty}**	Multiple errors: "Name is required", "Password is required"	All empty fields highlighted		

TEST CASE ID	TEST SCENARIO	TEST CASE DESCRIPTION	PRE-CONDITION	TEST STEPS	TEST DATA	EXPECTED RESULT	POST CONDITION	ACTUAL RESULT	STATUS (PASS/FAIL)
TC_Auth_19	TR-REG-05: Register Validate that system sends confirmation email after successful registration	and check email delivery	1. Valid email provided 2. Email service operational	1. Complete registration 2. Submit 3. Wait 30 seconds 4. Check inbox	Email: **verify@rvm.co m**	Email received with subject "Verify Your RVM Account", contains verification link	Email delivered within 1 minute		
TC_Auth_20	TR-REG-05: Email Validate that system sends	contains verification link	1. Registration completed	1. Open email 2. Locate verification link 3. Click link	Email content includes: Ve	rify Email button	Clicking link opens page, account status: "Verified"	Account activated	

1.3 Update Profile

TEST CASE ID	TEST SCENARIO	TEST CASE DESCRIPTION	PRE-CONDITION	TEST STEPS	TEST DATA	EXPECTED RESULT	POST CONDITION	ACTUAL RESULT	STATUS (PASS/FAIL)
TC_Auth_21	TR-UPD-01: Validate that registered user can update personal information	Update name successfully	1. User logged in 2. Profile page open 3. Current name: "Ahmed Ali"	1. Navigate to Profile Settings 2. Change name 3. Click "Save Changes"	Old Name: **Ahmed Ali** New Name: **Ahmed Mohamed**	Success message "Profile updated successfully", name changed	Profile shows new name		
TC_Auth_22	TR-UPD-01: Validate that registered user can update personal information	Update phone number	1. Logged in 2. Current phone: 01012345678	1. Open profile editor 2. Change phone 3. Save changes	Old Phone: **010 12345678** New Phone: **010 98765432**	Phone updated, confirmation displayed	New phone saved		
TC_Auth_23	TR-UPD-02: Validate that user cannot change email to already taken email	Change email to **already taken email**	1. Logged as user1@test.com 2. user2@test.com exists	1. Open settings 2. Try to change to user2@test.com 3. Save	Current: **user1@test.com** New: **user2@test.com** (exists)	Error "This email is already in use. Please choose another"	Email blocked		
TC_Auth_24	TR-UPD-02: Validate that user cannot change email to already existing email	Change email to same email	1. Current: **ahmed@test.com**	1. Edit email field 2. Re-enter same email 3. Save	Current & New: **ahmed@test.com**	Update succeeds (no change), no error	Email unchanged		
TC_Auth_25	TR-UPD-03: Validate that profile changes are saved and reflected	Update name and verify immediate display	1. Logged in as "Ali" 2. Profile page open	1. Change name to "Yasseen" 2. Save 3. Refresh page	Old: **Ali** New: **Yasseen**	Dashboard shows "Welcome, Yasseen" immediately	New name visible everywhere		

TEST CASE ID	TEST SCENARIO	TEST CASE DESCRIPTION	PRE-CONDITION	TEST STEPS	TEST DATA	EXPECTED RESULT	POST CONDITION	ACTUAL RESULT	STATUS (PASS/FAIL)
	immediatel y			4. Check dashboard					
TC_Auth_2 6	TR-UPD-03: Validate that profile changes are then perfor m transaction saved and reflected immediatel y	Update phone	1. User updates phone 2. Goes to RVM	1. Update phone in profile 2. Save 3. Complete transaction 4. Check receipt	New Phone: **010 98765432**	Receipt shows updated phone	Real-time update confirmed		
TC_Auth_2 7	TR-UPD-04: Validate that user must re-authenticat e when changing password	Change password requires cur rent password	1. User wants to change password 2. In profile settings	1. Click "Change Password" 2. Enter current password 3. Enter new password 4. Confirm	Current: **Pa ss@123** New: **New Pass@456**	System verifies current password, then updates	Password changed		
TC_Auth_2 8	TR-UPD-04: Validate that user must re-authenticat e when changing password	Enter incorrect current password	1. Changing password	1. Click "Change Password 2. Enter wrong current password 3. Enter new password 4. Submit	Current: **Wr ongPass1 2** New: **New Pass@456**	Error "Current password is incorrect", blocked	Old password still active		

TEST CASE ID	TEST SCENARIO	TEST CASE DESCRIPTION	PRE-CONDITION	TEST STEPS	TEST DATA	EXPECTED RESULT	POST CONDITION	ACTUAL RESULT	STATUS (PASS/FAIL)
TC_Auth_29	TR-UPD-05: Guest tries Validate to access that guest profile users settings cannot access profile update functionality	Guest User profile hidden for guests	1. User not logged in (guest) 2. Has direct URL	1. Navigate to /profile URL 2. System checks auth	User Type: **Guest**	Redirected to login with message "Please login to access this feature"	Access denied		
TC_Auth_30	TR-UPD-05: Validate that guest users cannot access profile update functionality	Profile menu hidden for guests	1. Guest user on homepage 2. Look for Profile option	1. Check navigation menu 2. Look for "Profile" option	User: **Guest**	"Profile" not visible, only "Login" and "Register" shown	UI hides authenticated features		

USE CASE 2 - MANAGE TRANSACTION

2.1 Process Recycling

TEST CASE ID	TEST SCENARIO	TEST CASE DESCRIPTION	PRE-CONDITION	TEST STEPS	TEST DATA	EXPECTED RESULT	POST CONDITION	ACTUAL RESULT	STATUS (PASS/FAIL)
TC_Txn_01	TR-PROC-01: Validate that system initiates recycling session when user starts transaction	Registered user **starts recycling session**	1. RVM is idle 2. User registered 3. QR code ready	1. User scans QR code 2. System authenticates 3. Check welcome message	User: **Ahmed** Balance: **120 points**	Welcome message: "Hello Ahmed! Balance: 120 pts. Insert items to recycle"	Session started, ready for items		
TC_Txn_02	TR-PROC-01: Validate that system initiates recycling session when user starts transaction	Guest user starts **recycling session**	1. RVM idle 2. Guest mode enabled	1. User presses "Start as Guest" 2. System creates temp session 3. Check display	User Type: **Guest**	Display: "Welcome Guest! Insert items to recycle. Points won't be saved."	Guest session active		
TC_Txn_03	TR-PROC-02: Validate that system processes items sequentially and maintains order	Insert **3 items sequentially**	1. Session active 2. Items ready: plastic, aluminum, glass	1. Insert plastic bottle 2. Wait for confirmation 3. Insert aluminum can 4. Insert glass bottle 5. Verify order	Item 1: **Plastic 20g** Item 2: **Aluminum 15g** Item 3: **Glass 50g**	Items processed in order: 1→2→3, display shows running count	3 items accepted, totals updated		
TC_Txn_04	TR-PROC-02: Validate that system processes items sequentially and maintains order	System displays **running totals** after each item	1. Session active 2. No items processed yet	1. Insert item 1 2. Check display 3. Insert item 2 4. Check updated display	Item 1: **Plastic 20g (0.2 pts)** Item 2: **Aluminum 15g (0.225 pts)**	After item 1: "Items: 1 | Weight: 20g | Points: 0.2" After item 2: "Items: 2 | Weight: 35g | Points: 0.425"	Running totals accurate		

TEST CASE ID	TEST SCENARIO	TEST CASE DESCRIPTION	PRE-CONDITION	TEST STEPS	TEST DATA	EXPECTED RESULT	POST CONDITION	ACTUAL RESULT	STATUS (PASS/FAIL)
TC_Txn_05	TR-PROC-03: Validate that system handles transaction completion successfully	User completes transaction with **bonus points**	1. 5 items processed 2. Peak hour active 3. User has 3-day streak	1. Press "Finish Transaction" 2. System calculates bonuses 3. Verify total	Base: **5.0 pts** Peak: **+20%** Streak: **+10%**	Final calculation: 5.0 × 1.2 × 1.1 = 6.6 points, displayed with breakdown	Transaction complete, points awarded		
TC_Txn_06	TR-PROC-03: Validate that system handles transaction completion successfully	Transaction updates **user point balance**	1. User balance: 100 pts 2. Transaction earned: 6.6 pts	1. Complete transaction 2. Check database update 3. Verify new balance	Old: **100 pts** Earned: **6.6 pts**	New balance: 106.6 points, saved in database	Balance updated successfully		
TC_Txn_07	TR-PROC-04: Validate that system allows transaction cancellation before completion	User cancels transaction **before finishing**	1. 2 items inserted 2. User wants to cancel	1. Press "Cancel Transaction" 2. System prompts confirmation 3. Confirm cancellation	Items: **2** Points: **0.5 pts**	Confirmation: "Cancel transaction? Items will be returned." → Items ejected, no points saved	Transaction cancelled, RVM reset		
TC_Txn_08	TR-PROC-04: Validate that system allows transaction cancellation before completion	Transaction **timeout after inactivity**	1. User inserted 1 item 2. No activity for 2 minutes	1. Wait 120 seconds 2. System detects timeout 3. Auto-cancel	Inactivity: **120 sec**	Warning at 90 sec: "Complete transaction or press Cancel" At 120 sec: Auto-cancel, items returned	Session ended, RVM idle		
TC_Txn_09	TR-PROC-05: Validate that system saves transaction	Transaction saved in **history for**	1. User: Ahmed 2.	1. Complete transaction 2. Check database	User: **Ahmed**	Record saved with: timestamp, user ID, item	Transaction in history,		

TEST CASE ID	TEST SCENARIO	TEST CASE DESCRIPTION	PRE-CONDITION	TEST STEPS	TEST DATA	EXPECTED RESULT	POST CONDITION	ACTUAL RESULT	STATUS (PASS/FAIL)
	data for registered users	**registered user**	Transaction completed	3. Verify saved data	Items: **5** Points: **6.6**	list, points, transaction ID	accessible via app		
TC_Txn_10	TR-PROC-05: Validate that system saves transaction data for registered users	Guest transaction **not saved** to history	1. Guest session 2. Transaction completed	1. Complete as guest 2. Check database 3. Verify no user record	User: **Guest** Points: **3.5**	Transaction logged (for RVM stats) but not linked to any user account	Guest data not in user history		

2.2 Validate Items:

TEST CASE ID	TEST SCENARIO	TEST CASE DESCRIPTION	PRE-CONDITION	TEST STEPS	TEST DATA	EXPECTED RESULT	POST CONDITION	ACTUAL RESULT	STATUS (PASS/FAIL)
TC_Txn_11	TR-VAL-01: Validate that system accepts valid recyclable bottles and cans	Accept valid **plastic bottle**	1. Session active 2. Plastic bottle ready	1. Insert plastic bottle 2. System scans material 3. Check acceptance	Material: **PET Plastic** Weight: **25g**	Display: "✓ Accepted: Plastic Bottle - 25g - 0.25 points"	Item stored, points calculated		
TC_Txn_12	TR-VAL-01: Validate that system accepts valid recyclable bottles and cans	Accept valid **aluminum can**	1. Session active 2. Aluminum can ready	1. Insert aluminum can 2. System identifies material 3. Verify acceptance	Material: **Aluminum** Weight: **15g** Factor: **1.5**	Display: "✓ Accepted: Aluminum Can - 15g - 0.225 points" (15g × 10 × 1.5 / 1000)	Can accepted, higher point value applied		
TC_Txn_13	TR-VAL-02: Validate that system rejects non-recyclable items	Reject **paper cup**	1. Session active 2. Non-recyclable item inserted	1. Insert paper cup 2. System scans 3. Check rejection	Material: **Paper**	Display: "✗ Item not recognized. Please insert recyclable bottles or cans only" Item ejected	Item returned to user		
TC_Txn_14	TR-VAL-02: Validate that system rejects non-recyclable items	Reject **food container**	1. Session active 2. Plastic food box inserted	1. Insert food container 2. ML model classifies 3. Verify rejection	Material: **Non-PET Plastic**	Rejection message: "This plastic type is not recyclable in our system"	Item ejected, no points		
TC_Txn_15	TR-VAL-03: Validate that RVM hardware correctly identifies item type (plastic, aluminum, glass)	Correctly identify **glass bottle**	1. Session active 2. Glass bottle inserted	1. Insert glass bottle 2. Sensor analyzes 3. Check classification	Material: **Glass** Weight: **80g** Factor: **0.8**	Identified as Glass, points calculated: 80g × 10 × 0.8 / 1000 = 0.64 pts	Glass accepted, correct factor applied		

TEST CASE ID	TEST SCENARIO	TEST CASE DESCRIPTION	PRE-CONDITION	TEST STEPS	TEST DATA	EXPECTED RESULT	POST CONDITION	ACTUAL RESULT	STATUS (PASS/FAIL)
TC_Txn_16	TR-VAL-03: Validate that RVM hardware correctly identifies item type (plastic, aluminum, glass)	Distinguish between **plastic and aluminum**	1. Two similar-shaped items 2. One plastic, one aluminum	1. Insert plastic can 2. Note classification 3. Insert aluminum can 4. Compare	Item 1: **Plastic Can 20g** Item 2: **Aluminum Can 20g**	Item 1: 0.20 pts (factor 1.0) Item 2: 0.30 pts (factor 1.5) Correct differentiation	Both accepted with correct factors		
TC_Txn_17	TR-VAL-04: Validate that system verifies item condition (undamaged, clean)	Reject **crushed bottle**	1. Session active 2. Damaged bottle inserted	1. Insert crushed bottle 2. Condition sensor checks 3. Verify rejection	Condition: **Crushed/Damaged**	Rejection: "Item condition unacceptable. Please ensure items are clean and undamaged"	Damaged item ejected		
TC_Txn_18	TR-VAL-04: Validate that system verifies item condition (undamaged, clean)	Reject **contaminated can**	1. Session active 2. Can with liquid residue	1. Insert wet can 2. System detects contamination 3. Check response	Contamination: **Liquid Detected**	Message: "Item appears contaminated. Please rinse items before recycling"	Can ejected		
TC_Txn_19	TR-VAL-05: Validate that system communicates validation results within 2 seconds	Validation time for **accepted item**	1. Session active 2. Timer ready	1. Start timer 2. Insert bottle 3. Wait for result 4. Stop timer	Item: **Plastic 30g**	Validation result displayed within 2 seconds: "✓ Accepted: Plastic - 30g - 0.30 pts"	Response time ≤ 2 sec		

TEST CASE ID	TEST SCENARIO	TEST CASE DESCRIPTION	PRE-CONDITION	TEST STEPS	TEST DATA	EXPECTED RESULT	POST CONDITION	ACTUAL RESULT	STATUS (PASS/FAIL)
TC_Txn_20	TR-VAL-05: Validate that system communicates validation results within 2 seconds	Validation time for **rejected item**	1. Session active 2. Non-recyclable item ready	1. Start timer 2. Insert paper cup 3. Measure response time	Item: **Paper Cup**	Rejection message displayed within 2 seconds, item ejected promptly	Response time ≤ 2 sec		

2.3 Generate Receipt:

TEST CASE ID	TEST SCENARIO	TEST CASE DESCRIPTION	PRE-CONDITION	TEST STEPS	TEST DATA	EXPECTED RESULT	POST CONDITION	ACTUAL RESULT	STATUS (PASS/FAIL)
TC_Txn_21	TR-REC-01: Validate that system generates receipt with transaction details (date, time, items, rewards)	Receipt contains **complete transaction details**	1. Transaction completed 2. 3 items processed	1. Complete transaction 2. Generate receipt 3. Verify content	Items: **3** Weight: **60g** Points: **0.75**	Receipt includes: Date, Time, User Name, Item List (type, qty, weight), Points Breakdown, Total	Complete receipt generated		
TC_Txn_22	TR-REC-01: Validate that system generates receipt with transaction details (date, time, items, rewards)	Receipt shows **bonus breakdown**	1. Transaction with bonuses 2. Peak hour + streak active	1. Complete transaction 2. Check receipt detail 3. Verify bonus section	Base: **5.0 pts** Peak: **+1.0 pt** Streak: **+0.6 pt**	Receipt breakdown: Base Points: 5.0 Peak Hour Bonus (+20%): 1.0 Streak Bonus (+10%): 0.6 Total: 6.6 pts	Detailed bonus calculation visible		
TC_Txn_23	TR-REC-02: Validate that receipt includes unique transaction ID	Receipt has **unique transaction ID**	1. Transaction completed 2. Receipt generated	1. Generate receipt 2. Extract transaction ID 3. Verify format	Date: **2025-11-08** Machine: **RVM01** Sequence: **00523**	Transaction ID format: TXN-20251108-RVM01-00523, unique and traceable	ID saved in database		
TC_Txn_24	TR-REC-02: Validate that receipt includes unique transaction ID	Verify **sequential transaction IDs**	1. Two consecutive transactions 2. Same machine, same day	1. Complete transaction 1 2. Complete transaction 2 3. Compare IDs	TXN1: **00523** TXN2: **00524**	Sequential IDs: TXN-20251108-RVM01-00523 and TXN-20251108-RVM01-00524	IDs increment correctly		

TEST CASE ID	TEST SCENARIO	TEST CASE DESCRIPTION	PRE-CONDITION	TEST STEPS	TEST DATA	EXPECTED RESULT	POST CONDITION	ACTUAL RESULT	STATUS (PASS/FAIL)
TC_Txn_25	TR-REC-03: Validate that receipt prints within 3 seconds of transaction completion	Receipt prints **within 3 seconds**	1. Transaction completed 2. Printer operational 3. Timer ready	1. Press "Finish" 2. Start timer 3. Wait for print 4. Stop timer	Printer: **Ready**	Physical receipt printed and available at output slot within 3 seconds	Print time ≤ 3 sec		
TC_Txn_26	TR-REC-03: Validate that receipt prints within 3 seconds of transaction completion	Receipt quality is **readable**	1. Printer has paper and ink 2. Transaction completed	1. Print receipt 2. Collect from slot 3. Check readability	Receipt: **Thermal Print**	All text clearly visible: headers, item details, totals, QR code scannable	Receipt legible		
TC_Txn_27	TR-REC-04: Validate that registered users can access digital receipt in transaction history	Digital receipt saved in **user history**	1. Registered user 2. Transaction completed	1. Complete transaction 2. Open app 3. Navigate to "Transaction History" 4. Find receipt	User: **Ahmed** Transaction: **TXN-20251108-RVM01-00523**	Digital receipt accessible in app, includes all details + downloadable PDF	Receipt saved and accessible		
TC_Txn_28	TR-REC-04: Validate that registered users can access digital receipt in transaction history	User receives **email with receipt**	1. Registered user with email 2. Transaction completed	1. Complete transaction 2. Wait 30 seconds 3. Check email inbox	Email: **ahmed@rvm.com**	Email received with subject "Recycling Receipt - TXN-20251108-RVM01-00523", PDF attached	Email delivered with PDF		

TEST CASE ID	TEST SCENARIO	TEST CASE DESCRIPTION	PRE-CONDITION	TEST STEPS	TEST DATA	EXPECTED RESULT	POST CONDITION	ACTUAL RESULT	STATUS (PASS/FAIL)
TC_Txn_29	TR-REC-05: Validate that system handles printer errors gracefully with appropriate error message	Handle **out of paper** error	1. Printer paper empty 2. Transaction completed	1. Complete transaction 2. System tries to print 3. Check error handling	Printer Status: **Out of Paper**	Display: "Receipt could not be printed. Check your email/app for digital receipt" Alert sent to maintenance	Digital receipt available, maintenance notified		
TC_Txn_30	TR-REC-05: Validate that system handles printer errors gracefully with appropriate error message	Handle **paper jam** error	1. Printer jammed 2. User completing transaction	1. Complete transaction 2. System detects jam 3. Verify fallback	Printer Status: **Jammed**	Error message displayed, maintenance alert sent, user directed to digital receipt options	Transaction saved, user not blocked		

USE CASE 3 - MANAGE QR CODE

3.1 Generate QR Code

TEST CASE ID	TEST SCENARIO	TEST CASE DESCRIPTION	PRE-CONDITION	TEST STEPS	TEST DATA	EXPECTED RESULT	POST CONDITION	ACTUAL RESULT	STATUS (PASS/FAIL)
TC_QR_01	TR-GEN-01: Validate that system generates unique QR code after successful reward redemption	Generate **unique QR code** for redemption	1. User balance: 100 pts 2. Redemption: 50 pts ($5)	1. Request redemption 2. System deducts points 3. Generate QR code 4. Verify uniqueness	User: **Ahmed** Amount: **$5.00** ID: **RDM-20251108143025-U12345-A7F3C9**	Unique QR code generated with redemption ID, no duplicates	QR code active, balance updated		
TC_QR_02	TR-GEN-01: Validate that system generates unique QR code after successful reward redemption	Sequential redemptions create **different QR codes**	1. User redeems twice 2. Same user, different times	1. Redeem $5 (first) 2. Get QR code 1 3. Redeem $10 (second) 4. Get QR code 2 5. Compare	QR1: **RDM-...-A7F3C9** QR2: **RDM-...-B2E8D4**	Two distinct QR codes with different IDs and hashes	Both QR codes active independently		
TC_QR_03	TR-GEN-02: Validate that QR code contains encrypted transaction data (user ID, amount, timestamp)	QR code contains **encrypted payload**	1. QR code generated 2. Decryption key available	1. Generate QR code 2. Decode QR data 3. Decrypt payload 4. Verify content	User ID: **U12345** Amount: **$5.00** Time: **2025-11-08 14:30:25**	Decrypted data includes: User ID, User Name, Amount, Redemption ID, Timestamp, Signature	All data fields present		
TC_QR_04	TR-GEN-02: Validate that QR code contains encrypted transaction data (user ID, amount, timestamp)	QR code has **valid digital signature**	1. QR code generated 2. Signature verification enabled	1. Generate QR code 2. Extract signature 3. Verify HMAC-SHA256 4. Check authenticity	Signature: **HMAC-SHA256 Hash**	Signature verification passes, confirms authenticity and integrity	QR tamper-proof		

TEST CASE ID	TEST SCENARIO	TEST CASE DESCRIPTION	PRE-CONDITION	TEST STEPS	TEST DATA	EXPECTED RESULT	POST CONDITION	ACTUAL RESULT	STATUS (PASS/FAIL)
TC_QR_05	TR-GEN-03: Validate that QR code is generated within 2 seconds of redemption request	QR generation time **≤ 2 seconds**	1. User requests redemption 2. Timer ready	1. Start timer 2. Request redemption 3. Wait for QR code 4. Stop timer	Amount: **$5.00**	QR code displayed on mobile app within 2 seconds	Generation time ≤ 2 sec		
TC_QR_06	TR-GEN-03: Validate that QR code is generated within 2 seconds of redemption request	High-resolution QR image created **quickly**	1. Redemption approved 2. Image generation pending	1. Trigger QR generation 2. Measure time 3. Verify resolution	Resolution: **800×800px**	High-res QR image (800×800px) generated within 2 seconds	Image quality verified		
TC_QR_07	TR-GEN-04: Validate that QR code includes expiration time (24 hours from generation)	QR code has **24-hour expiration**	1. QR code generated at 14:30 2. Nov 8, 2025	1. Generate QR code 2. Check expiration field 3. Verify 24-hour window	Generated: **Nov 8, 2025 14:30** Expires: **Nov 9, 2025 14:30**	Expiration timestamp exactly 24 hours from generation	Expiration embedded in QR		
TC_QR_08	TR-GEN-04: Validate that QR code includes expiration time (24 hours from generation)	Display **countdown timer** on app	1. QR code shown on app 2. 23 hours remaining	1. Open app 2. View QR code 3. Check countdown display	Remaining: **23h 45m**	App displays countdown: "Valid for 23h 45m" and updates in real-time	User aware of expiration		

TEST CASE ID	TEST SCENARIO	TEST CASE DESCRIPTION	PRE-CONDITION	TEST STEPS	TEST DATA	EXPECTED RESULT	POST CONDITION	ACTUAL RESULT	STATUS (PASS/FAIL)
TC_QR_09	TR-GEN-05: Validate that user receives QR code through multiple formats (image, PDF)	QR code available as **PNG image**	1. QR code generated 2. Mobile app active	1. Generate QR code 2. Check app display 3. Verify image format	Format: **PNG 800×800px**	QR code displayed on app as high-quality PNG image	Image viewable on app		
TC_QR_10	TR-GEN-05: Validate that user receives QR code through multiple formats (image, PDF)	QR code sent via **email as PDF**	1. User email: ahmed@rvm.com 2. QR generated	1. Generate QR code 2. Wait 30 seconds 3. Check email 4. Open PDF	Email: **ahmed@rvm.com**	Email received with PDF attachment containing QR code, redemption details, and instructions	PDF accessible via email		

3.2 Scan QR Code:

TEST CASE ID	TEST SCENARIO	TEST CASE DESCRIPTION	PRE-CONDITION	TEST STEPS	TEST DATA	EXPECTED RESULT	POST CONDITION	ACTUAL RESULT	STATUS (PASS/FAIL)
TC_QR_11	TR-SCN-01: Validate that cafeteria system can successfully scan QR codes	Scan QR code from **mobile app screen**	1. QR code displayed on phone 2. Scanner active	1. User shows phone screen 2. Staff points scanner 3. System captures image 4. Verify scan	User: **Ahmed** Amount: **$5.00**	QR code scanned successfully, data decoded, beep sound + green checkmark	QR data retrieved		
TC_QR_12	TR-SCN-01: Validate that cafeteria system can successfully scan QR codes	Scan QR code from **printed PDF**	1. User printed PDF 2. Scanner ready	1. User presents paper 2. Staff scans printed QR 3. Check recognition	Format: **Printed PDF**	Printed QR code scanned successfully, same data extracted as digital version	Print quality sufficient		
TC_QR_13	TR-SCN-02: Validate that system decodes QR data correctly	Decode QR and extract **all fields**	1. QR code scanned 2. Encrypted payload captured	1. Scan QR code 2. Decrypt payload 3. Parse data fields 4. Verify completeness	Fields: **User ID, Name, Amount, ID, Timestamp, Signature**	All fields decoded correctly: Ahmed Ali, $5.00, RDM-20251108143025-U12345-A7F3C9	Data extraction complete		
TC_QR_14	TR-SCN-02: Validate that system decodes QR data correctly	Handle **encryption decryption** properly	1. QR encrypted with AES-256 2. Secret key available	1. Scan encrypted QR 2. Apply decryption 3. Verify readable data	Encryption: **AES-256**	Decryption successful, data readable and matches original redemption	Secure data transfer verified		
TC_QR_15	TR-SCN-03: Validate that scanning process	Scan time **≤ 2 seconds**	1. QR code ready 2. Scanner active	1. Start timer 2. Scan QR code 3. Wait for	QR: **Valid code**	Scan + decode + display	Response time ≤ 2 sec		

TEST CASE ID	TEST SCENARIO	TEST CASE DESCRIPTION	PRE-CONDITION	TEST STEPS	TEST DATA	EXPECTED RESULT	POST CONDITION	ACTUAL RESULT	STATUS (PASS/FAIL)
	completes within 2 seconds		3. Timer ready	result 4. Stop timer		completed within 2 seconds			
TC_QR_16	TR-SCN-03: Validate that scanning process completes within 2 seconds	Fast processing under **good conditions**	1. Good lighting 2. Clear QR code 3. Steady hand	1. Scan under optimal conditions 2. Measure speed 3. Verify instant feedback	Conditions: **Optimal**	Near-instant scan (< 1 second), immediate beep and display	Fast scan confirmed		
TC_QR_17	TR-SCN-04: Validate that system handles damaged/unclear QR codes with error message	Reject **blurry QR code**	1. QR code out of focus 2. Scanner attempts read	1. Present blurry QR 2. System tries to scan 3. Check error handling	Quality: **Blurry/Unclear**	Error: "Unable to read QR code. Please adjust lighting or show clearer image"	User prompted to retry		
TC_QR_18	TR-SCN-04: Validate that system handles damaged/unclear QR codes with error message	Reject **partially damaged QR**	1. QR code with missing corner 2. Scanner active	1. Scan damaged QR 2. System attempts decode 3. Verify error	Damage: **Corner missing**	Error: "QR code damaged or incomplete. Please use a clearer version"	Scan rejected		
TC_QR_19	TR-SCN-05: Validate that system displays QR code value and details after successful scan	Display **redemption details** on cafeteria terminal	1. QR scanned successfully 2. Data decoded	1. Scan QR code 2. Check terminal display 3. Verify information shown	User: **Ahmed Ali** Amount: **$5.00** Expires: **Nov 9, 14:30**	Terminal displays: "Student: Ahmed Ali | Amount: $5.00 | Valid until: Nov 9, 2025 2:30 PM"	All details visible to staff		

TEST CASE ID	TEST SCENARIO	TEST CASE DESCRIPTION	PRE-CONDITION	TEST STEPS	TEST DATA	EXPECTED RESULT	POST CONDITION	ACTUAL RESULT	STATUS (PASS/FAIL)
TC_QR_20	TR-SCN-05: Validate that system displays QR code value and details after successful scan	Show **visual confirmation** to user	1. QR scanned 2. User watching terminal	1. Scan QR code 2. Observe feedback 3. Verify user notification	Feedback: **Green checkmark + beep**	Green checkmark displayed, beep sound, message: "QR code scanned successfully"	User knows scan succeeded		

3.3 Validate QR Code:

TEST CASE ID	TEST SCENARIO	TEST CASE DESCRIPTION	PRE-CONDITION	TEST STEPS	TEST DATA	EXPECTED RESULT	POST CONDITION	ACTUAL RESULT	STATUS (PASS/FAIL)
TC_QR_21	TR-VQR-01: Validate that system verifies QR code authenticity through encryption signature	Verify **valid digital signature**	1. QR code scanned 2. Signature present	1. Extract signature 2. Compute HMAC-SHA256 3. Compare hashes 4. Verify match	Signature: **Valid HMAC**	Signature verification passes, QR code authenticated	QR authenticity confirmed		
TC_QR_22	TR-VQR-01: Validate that system verifies QR code authenticity through encryption signature	Reject **tampered QR code**	1. QR data modified 2. Signature doesn't match	1. Scan tampered QR 2. Verify signature 3. Detect mismatch 4. Reject	Status: **Signature Invalid**	Error: "Invalid QR code. Authentication failed. Possible tampering detected."	QR rejected, security alert logged		
TC_QR_23	TR-VQR-02: Validate that system checks QR code has not been previously used	Accept **unused QR code**	1. QR never used 2. Usage flag: "Unused"	1. Scan QR code 2. Check database status 3. Verify unused flag 4. Proceed	Status: **Unused**	Validation passes: "✓ Valid QR Code | Proceed with redemption"	Ready for redemption		
TC_QR_24	TR-VQR-02: Validate that system checks QR code has not been previously used	Reject **already used QR code**	1. QR used on Nov 8 2. Usage flag: "Used"	1. Scan same QR again 2. Check database 3. Detect previous use 4. Reject	Used: **Nov 8, 2025 15:00 at Cafeteria A**	Error: "QR code already redeemed on Nov 8, 2025 3:00 PM at Cafeteria A. Cannot be reused."	Duplicate use prevented		
TC_QR_25	TR-VQR-03: Validate that system rejects expired QR codes	Accept QR within **24-hour window**	1. Generated: Nov 8 14:30 2. Scanned: Nov 9	1. Scan QR code 2. Check timestamp 3. Calculate age	Age: **19.5 hours**	QR code accepted, within valid 24-hour period	Validation successful		

TEST CASE ID	TEST SCENARIO	TEST CASE DESCRIPTION	PRE-CONDITION	TEST STEPS	TEST DATA	EXPECTED RESULT	POST CONDITION	ACTUAL RESULT	STATUS (PASS/FAIL)
	(older than 24 hours)		10:00 (19.5 hrs later)	4. Verify valid					
TC_QR_26	TR-VQR-03: Validate that system rejects expired QR codes (older than 24 hours)	Reject **expired QR code**	1. Generated: Nov 8 14:30 2. Scanned: Nov 10 15:00 (25 hrs later)	1. Scan QR code 2. Check expiration 3. Detect expired 4. Reject	Age: **25 hours**	Error: "QR code expired. Please generate a new redemption code."	Expired QR rejected		
TC_QR_27	TR-VQR-04: Validate that system marks QR code as "used" after successful redemption	Update QR status to **"Used"**	1. Valid QR scanned 2. Staff confirms redemption	1. Validate QR 2. Staff approves 3. System updates DB 4. Check status	Before: **Unused** After: **Used**	Database updated: Status="Used", Timestamp=Current, Location=Cafeteria A	QR marked as used		
TC_QR_28	TR-VQR-04: Validate that system marks QR code as "used" after successful redemption	Send **confirmation notification** to user	1. QR redeemed successfully 2. User email active	1. Complete redemption 2. System marks as used 3. Check notification sent	User: **ahmed@rvm.com** Amount: **$5.00**	Notification sent: "You've successfully redeemed $5.00 at Cafeteria A"	User notified		
TC_QR_29	TR-VQR-05: Validate that invalid or tampered QR codes display	Reject **unrecognized QR code**	1. QR from external source 2. Not in database	1. Scan unknown QR 2. Query database 3. No record	Redemption ID: **Not Found**	Error: "QR code not recognized. Please contact support."	Fake QR rejected		

TEST CASE ID	TEST SCENARIO	TEST CASE DESCRIPTION	PRE-CONDITION	TEST STEPS	TEST DATA	EXPECTED RESULT	POST CONDITION	ACTUAL RESULT	STATUS (PASS/FAIL)
	appropriate error message			found 4. Reject					
TC_QR_30	TR-VQR-05: Validate that invalid or tampered QR codes display appropriate error message	Handle **corrupted QR data**	1. QR with corrupted payload 2. Decryption fails	1. Scan QR code 2. Attempt decryption 3. Detect corruption 4. Display error	Payload: **Corrupted/Invalid**	Error: "Unable to process QR code. Data corrupted or invalid format."	Corrupted QR rejected		

USE CASE 6 - MANAGE NOTIFICATIONS

6.1 Send System Alert

TEST CASE ID	TEST SCENARIO	TEST CASE DESCRIPTION	PRE-CONDITION	TEST STEPS	TEST DATA	EXPECTED RESULT	POST CONDITION	ACTUAL RESULT	STATUS (PASS/FAIL)
TC_Not_41	TR-SYS-01: Validate that administrators receive immediate alerts for system errors or failures	Validate that **critical error** sends alert immediately	1. Admin account active 2. Machine operational 3. Email/SMS configured	1. Trigger system error (machine full) 2. System detects error 3. Check alert delivery time	Error Type: **Machine Full** Severity: **Critical** Admin: **admin@rvm.com**	Alert sent within 5 seconds via email + SMS	Alert logged, admin notified immediately		
TC_Not_42	TR-SYS-01: Validate that administrators receive immediate alerts for system errors or failures	Validate **network failure** alert delivery	1. System connected 2. Admin configured	1. Disconnect network 2. System detects failure 3. Verify alert sent after reconnection	Error Type: **Network Offline** Severity: **High**	Alert queued, sent when connection restored	Alert delivered with timestamp of error		
TC_Not_43	TR-SYS-02: Validate that maintenance staff receive alerts for machine full or malfunction	Send alert when machine **reaches 80% capacity**	1. Machine capacity at 75% 2. Maintenance staff registered	1. Fill machine to 80% 2. System checks capacity 3. Verify alert sent	Capacity: **80%** Staff: **maint@rvm.com**	Email alert sent "Machine at Location X is 80% full"	Maintenance staff notified		
TC_Not_44	TR-SYS-02: Validate that maintenance staff receive alerts for machine full or malfunction	Send alert for **sensor malfunction**	1. Sensors working 2. Machine active	1. Simulate sensor failure 2. System runs diagnostic 3. Check alert delivery	Error: **Weight Sensor Failed** Machine: **RVM-001**	Alert sent to maintenance with error details	Sensor error logged, staff alerted		

TEST CASE ID	TEST SCENARIO	TEST CASE DESCRIPTION	PRE-CONDITION	TEST STEPS	TEST DATA	EXPECTED RESULT	POST CONDITION	ACTUAL RESULT	STATUS (PASS/FAIL)
TC_Not_45	TR-SYS-03: Validate that alert severity levels (low, medium, high, critical) are correctly classified	Classify **low severity** alert correctly	1. System operational 2. Alert system active	1. Trigger low priority event (capacity 50%) 2. Verify severity classification	Event: **Capacity Warning 50%**	Severity classified as "Low", email only (no SMS)	Alert sent via email only		
TC_Not_46	TR-SYS-03: Validate that alert severity levels (low, medium, high, critical) are correctly classified	Classify **critical severity** correctly	1. System active 2. Multi-channel configured	1. Trigger critical error (fire sensor) 2. Check classification 3. Verify channels used	Error: **Fire Sensor Activated**	Severity "Critical", sent via email + SMS + in-app	All channels notified immediately		
TC_Not_47	TR-SYS-04: Validate that critical alerts use multiple channels (email + SMS)	Verify **email + SMS** delivery for critical alert	1. Admin has email and phone 2. Both channels active	1. Trigger critical alert 2. Check email delivery 3. Check SMS delivery	Admin Email: **admin@rvm.com** Phone: **+20101 2345678**	Both email and SMS received within 10 seconds	Dual notification confirmed		
TC_Not_48	TR-SYS-04: Validate that critical alerts use multiple channels (email + SMS)	Handle **SMS failure**, email still delivered	1. Email working 2. SMS service down	1. Trigger critical alert 2. SMS fails 3. Verify email sent	SMS Status: **Failed** Email: **Active**	Email sent successfully, SMS retry scheduled	Email delivered, SMS error logged		
TC_Not_49	TR-SYS-05: Validate that alert resolution updates are	Send **resolution**	1. Active alert exists 2. Issue	1. Admin fixes issue 2. Mark alert as resolved 3. Check	Alert ID: **ALT-001**	Resolution notification "Issue resolved:	All stakeholders		

TEST CASE ID	TEST SCENARIO	TEST CASE DESCRIPTION	PRE-CONDITION	TEST STEPS	TEST DATA	EXPECTED RESULT	POST CONDITION	ACTUAL RESULT	STATUS (PASS/FAIL)
	sent after issue is fixed	**notification** after fix	being resolved	notification sent	Status: **Resolved**	Machine operational"	notified of resolution		
TC_Not_50	TR-SYS-05: Validate that alert resolution updates are sent after issue is fixed	Update **alert status** from active to resolved	1. Alert in database 2. Status = Active	1. Resolve issue 2. System updates status 3. Verify database update	Alert: **Machine Full** New Status: **Resolved**	Database updated, status changed, timestamp recorded	Alert history shows resolution		

6.2 Send Transaction Notification:

TEST CASE ID	TEST SCENARIO	TEST CASE DESCRIPTION	PRE-CONDITION	TEST STEPS	TEST DATA	EXPECTED RESULT	POST CONDITION	ACTUAL RESULT	STATUS (PASS/FAIL)
TC_Not_51	TR-TXN-01: Validate that registered users receive notification after successful transaction completion	Send notification **after successful** transaction	1. User logged in 2. Transaction completed 3. Notifications enabled	1. Complete transaction 2. Check notification delivery 3. Verify timing	User: **yasseen@rvm.com** Points: **80** Items: **5 bottles**	Notification received within 5 seconds	User notified with transaction summary		
TC_Not_52	TR-TXN-01: Validate that registered users receive notification after successful transaction completion	Verify **in-app notification** displayed	1. User has mobile app 2. App running	1. Complete transaction 2. Open mobile app 3. Check notification badge	App Version: **2.1.0** User: **ahmed@rvm.com**	In-app notification with sound alert, badge updated	Notification visible in app		
TC_Not_53	TR-TXN-02: Validate that notification includes transaction summary (items, points earned, balance)	Include **complete transaction details**	1. Transaction completed 2. Points calculated	1. Review notification content 2. Verify all fields present	Items: **5 bottles, 3 cans** Points: **80+10 bonus** Balance: **540**	Notification shows items, points, bonus, new balance	Complete summary displayed		
TC_Not_54	TR-TXN-02: Validate that notification includes transaction summary (items, points earned, balance)	Display **breakdown with bonus points**	1. Bonus applied 2. Transaction complete	1. Check notification 2. Verify bonus shown separately	Base Points: **80** Bonus: **+10** Total: **90**	Notification shows "80 points + 10 bonus = 90 total"	Bonus clearly indicated		

TEST CASE ID	TEST SCENARIO	TEST CASE DESCRIPTION	PRE-CONDITION	TEST STEPS	TEST DATA	EXPECTED RESULT	POST CONDITION	ACTUAL RESULT	STATUS (PASS/FAIL)
TC_Not_55	TR-TXN-03: Validate that notification is sent within 5 seconds of transaction completion	Measure **notification delivery time**	1. Transaction ready 2. Network active	1. Complete transaction 2. Start timer 3. Check notification received 4. Stop timer	Start Time: **14:30:00** End Time: **14:30:03**	Notification received in 3 seconds (< 5 seconds)	Timing requirement met		
TC_Not_56	TR-TXN-03: Validate that notification is sent within 5 seconds of transaction completion	Test **under high load** conditions	1. Multiple transactions 2. System busy	1. Trigger 10 concurrent transactions 2. Measure delivery time for each	Concurrent Users: **10** Load: **High**	All notifications delivered within 5 seconds despite load	Performance maintained under load		
TC_Not_57	TR-TXN-04: Validate that users can opt-in/opt-out of transaction notifications in settings	User **opts out** of notifications	1. Notifications enabled 2. User in settings	1. Navigate to settings 2. Disable transaction notifications 3. Complete transaction 4. Verify no notification	Setting: **Disabled** User: **test@rvm.com**	No notification sent, preference saved	User preference respected		
TC_Not_58	TR-TXN-04: Validate that users can opt-in/opt-out of transaction notifications in settings	User **opts back in** to notifications	1. Notifications disabled 2. User changes setting	1. Enable notifications 2. Save settings 3. Complete transaction 4. Verify notification received	Setting: **Enabled** Preference: **Updated**	Notification sent successfully after re-enabling	Opt-in working correctly		

TEST CASE ID	TEST SCENARIO	TEST CASE DESCRIPTION	PRE-CONDITION	TEST STEPS	TEST DATA	EXPECTED RESULT	POST CONDITION	ACTUAL RESULT	STATUS (PASS/FAIL)
TC_Not_59	TR-TXN-05: Validate that notification delivery failures are logged and retried	Handle **delivery failure** with retry	1. Network unstable 2. Transaction complete	1. Disconnect network 2. Trigger notification 3. Reconnect network 4. Verify retry	Attempt: **1 (Failed)** Retry: **2 (Success)**	Failure logged, automatic retry successful	Notification delivered on retry		
TC_Not_60	TR-TXN-05: Validate that notification delivery failures are logged and retried	Log **all retry attempts** in database	1. Delivery failing 2. Retry mechanism active	1. Trigger failure 2. System retries 3 times 3. Check database logs	Attempts: **3** Status: **All Logged**	All 3 attempts logged with timestamps and reasons	Complete audit trail maintained		

6.3 Send Reward Notification:

TEST CASE ID	TEST SCENARIO	TEST CASE DESCRIPTION	PRE-CONDITION	TEST STEPS	TEST DATA	EXPECTED RESULT	POST CONDITION	ACTUAL RESULT	STATUS (PASS/FAIL)
TC_Not_61	TR-RWD-01: Validate that users receive notification when reward is successfully redeemed	Send notification with **reward details**	1. User has points 2. Reward available 3. Redemption successful	1. Redeem reward 2. Check notification 3. Verify content	Reward: **Coffee** Cost: **100 points** User: **ahmed@rvm.com**	Notification shows "Coffee redeemed for 100 points"	User informed of redemption		
TC_Not_62	TR-RWD-01: Validate that users receive notification when reward is successfully redeemed	Display **updated point balance** after redemption	1. Previous balance: 540 2. Reward cost: 100	1. Redeem reward 2. Check notification 3. Verify new balance shown	Old Balance: **540** New Balance: **440**	Notification shows "Remaining balance: 440 points"	Balance update communicated		
TC_Not_63	TR-RWD-02: Validate that notification includes QR code and redemption instructions	Include **QR code** in notification	1. Reward redeemed 2. QR generated	1. Check notification 2. Verify QR code present 3. Test QR scannable	QR Code: **QR-2025 1101-001**	QR code embedded in notification, scannable	QR code ready for use		
TC_Not_64	TR-RWD-02: Validate that notification includes QR code and redemption instructions	Provide clear **redemption instructions**	1. QR code generated 2. Notification sent	1. Open notification 2. Read instructions 3. Verify clarity	Instructions: **"Show QR at cafeteria within 24hrs"**	Clear step-by-step redemption instructions included	User knows how to redeem		

TEST CASE ID	TEST SCENARIO	TEST CASE DESCRIPTION	PRE-CONDITION	TEST STEPS	TEST DATA	EXPECTED RESULT	POST CONDITION	ACTUAL RESULT	STATUS (PASS/FAIL)
TC_Not_65	TR-RWD-03: Validate that reminder notification is sent before reward/QR code expiration	Send **reminder 2 hours** before expiration	1. QR expires in 24hrs 2. 22 hours elapsed	1. Wait for reminder time 2. Check notification sent 3. Verify timing	Expiry: **24 hours** Reminder: **22 hours**	Reminder "QR expires in 2 hours!" sent	User warned before expiration		
TC_Not_66	TR-RWD-03: Validate that reminder notification is sent before reward/QR code expiration	Include **expiration countdown** in reminder	1. QR code active 2. Reminder triggered	1. Receive reminder 2. Check countdown display	Time Remaining: **2 hours** Exact Expiry: **16:30:00**	Reminder shows "Expires at 16:30 (in 2 hours)"	Clear expiration time shown		
TC_Not_67	TR-RWD-04: Validate that users receive notification when reaching reward milestones	Notify when user **reaches 500 points** milestone	1. User at 490 points 2. Transaction adds 15 points	1. Complete transaction 2. Points reach 505 3. Check milestone notification	Previous: **490** New: **505** Milestone: **500**	Notification "Congratulations! You've reached 500 points!"	Milestone achievement celebrated		
TC_Not_68	TR-RWD-04: Validate that users receive notification	Suggest **available rewards** at milestone	1. Milestone reached 2. Rewards available	1. Trigger milestone 2. Check notification content	Points: **500** Available: **Coffee, Sandwich**	Notification suggests "You can now redeem: Coffee, Sandwich"	User encouraged to redeem		

TEST CASE ID	TEST SCENARIO	TEST CASE DESCRIPTION	PRE-CONDITION	TEST STEPS	TEST DATA	EXPECTED RESULT	POST CONDITION	ACTUAL RESULT	STATUS (PASS/FAIL)
	when reaching reward milestones			3. Verify suggestions					
TC_Not_69	TR-RWD-05: Validate that notification includes current point balance after redemption	Show **remaining balance** prominently	1. Redemption complete 2. Points deducted	1. Redeem reward 2. Open notification 3. Check balance display	Redeemed: **100 points** Remaining: **440 points**	Notification highlights "Remaining balance: 440 points"	Balance clearly visible		
TC_Not_70	TR-RWD-05: Validate that notification includes current point balance after redemption	Calculate **points needed** for next reward	1. Balance: 440 2. Next reward: 500 points	1. Check notification 2. Verify calculation shown	Current: **440** Next Goal: **500** Needed: **60**	Notification shows "60 more points to next reward!"	User motivated to continue		

USE CASE 5 - MANAGE POINTS:

5.1 Accumulate Points:

TEST CASE ID	TEST SCENARIO	TEST CASE DESCRIPTION	PRE-CONDITION	TEST STEPS	TEST DATA	EXPECTED RESULT	POST CONDITION	ACTUAL RESULT	STATUS (PASS/FAIL)
TC_Pts_01	TR-PTS-01: Validate that points are calculated correctly based on weight and material type	Calculate points for **plastic bottles**	1. User logged in 2. Items validated 3. Weight measured	1. Deposit 5 plastic bottles 2. System weighs items (0.5kg) 3. Calculate points	Weight: **0.5kg** Material: **Plastic** Factor: **1.0**	Base points = 0.5 × 10 × 1.0 = 50 points	50 points calculated correctly		
TC_Pts_02	TR-PTS-01: Validate that points are calculated correctly based on weight and material type	Calculate points for **aluminum cans**	1. User logged in 2. Items validated as aluminum	1. Deposit 8 aluminum cans 2. System weighs (0.4kg) 3. Apply aluminum factor	Weight: **0.4kg** Material: **Aluminum** Factor: **1.5**	Base points = 0.4 × 10 × 1.5 = 60 points	60 points calculated with aluminum bonus		
TC_Pts_03	TR-PTS-02: Validate that bonus strategies (time-based, streak, milestone) are applied correctly	Apply **peak hour bonus** (+20%)	1. Current time: 10:00 AM (Monday) 2. Peak hours: 8AM-2PM	1. Complete transaction at 10:00 AM 2. Base points: 50 3. Apply peak hour bonus	Base: **50 pts** Time: **10:00 AM** Bonus: **+20%**	Total = 50 + (50 × 0.20) = 60 points	Peak hour bonus applied correctly		
TC_Pts_04	TR-PTS-02: Validate that bonus strategies (time-based, streak,	Apply **recycling streak bonus**	1. User has 5-day streak 2. Base	1. Check user's streak 2. Apply +10% streak bonus	Base: **50 pts** Streak: **5 days**	Total = 50 + (50 × 0.10) = 55 points	Streak bonus added to points		

TEST CASE ID	TEST SCENARIO	TEST CASE DESCRIPTION	PRE-CONDITION	TEST STEPS	TEST DATA	EXPECTED RESULT	POST CONDITION	ACTUAL RESULT	STATUS (PASS/FAIL)
	milestone) are applied correctly		points earned: 50	3. Calculate total	Bonus: **+10%**				
TC_Pts_05	TR-PTS-03: Validate that user's point balance is updated correctly after transaction	Update balance after **earning points**	1. Previous balance: 450 pts 2. Points earned: 65 pts	1. Retrieve current balance 2. Add earned points 3. Update database	Old Balance: **450** Earned: **65** New: **515**	Balance updated to 515 points successfully	User balance = 515 points		
TC_Pts_06	TR-PTS-03: Validate that user's point balance is updated correctly after transaction	Verify **transaction record** created	1. Transaction completed 2. Database accessible	1. Complete transaction 2. Check database for record 3. Verify all fields	User: **yasseen@rvm.com** Points: **65** Timestamp: **Logged**	Transaction record created with all details	Record exists in database		
TC_Pts_07	TR-PTS-04: Validate that milestone achievements are detected and notified	Detect **500 points milestone**	1. Previous balance: 490 pts 2. Earned: 15 pts	1. Update balance to 505 2. Check milestone crossed 3. Trigger notification	Previous: **490** New: **505** Milestone: **500**	Milestone notification "Reached 500 points!" sent	User notified of milestone		
TC_Pts_08	TR-PTS-04: Validate that milestone achievements are detected and notified	Suggest rewards at **milestone**	1. Milestone reached: 500 pts 2. Rewards available	1. Detect milestone 2. Query available rewards 3. Include in notification	Balance: **500** Available: **Meal (500 pts)**	Notification suggests "You can now redeem: Meal"	User encouraged to redeem		

TEST CASE ID	TEST SCENARIO	TEST CASE DESCRIPTION	PRE-CONDITION	TEST STEPS	TEST DATA	EXPECTED RESULT	POST CONDITION	ACTUAL RESULT	STATUS (PASS/FAIL)
TC_Pts_09	TR-PTS-05: Validate that leaderboard rankings are updated with new point totals	Update **student leaderboard** ranking	1. Previous rank: #15 2. New points earned	1. Update user's total points 2. Recalculate rankings 3. Update leaderboard	User: **Ahmed** Points: **515** New Rank: **#12**	Leaderboard shows user at rank #12	Rankings updated in real-time		
TC_Pts_10	TR-PTS-05: Validate that leaderboard rankings are updated with new point totals	Update **college leaderboard**	1. College total: 15,000 pts 2. Student adds: 65 pts	1. Add student points to college 2. Recalculate college ranks 3. Update display	College: **Engineering** New Total: **15,065**	College leaderboard updated with new total	College ranking may change		

5.2 Redeem Points:

TEST CASE ID	TEST SCENARIO	TEST CASE DESCRIPTION	PRE-CONDITION	TEST STEPS	TEST DATA	EXPECTED RESULT	POST CONDITION	ACTUAL RESULT	STATUS (PASS/FAIL
TC_Pts_11	TR-RDM-01: Validate that system checks if user has sufficient points before redemption	Redeem with **sufficient points**	1. User balance: 515 pts 2. Reward cost: 500 pts	1. Select reward (Meal) 2. Validate balance 3. Proceed redemption	Balance: **515** Cost: **500** Valid: **Yes**	Validation passes, redemption allowed	Redemption process continues		

TEST CASE ID	TEST SCENARIO	TEST CASE DESCRIPTION	PRE-CONDITION	TEST STEPS	TEST DATA	EXPECTED RESULT	POST CONDITION	ACTUAL RESULT	STATUS (PASS/FAIL
TC_Pts_12	TR-RDM-01: Validate that system checks if user has sufficient points before redemption	Block redemption with **insufficient points**	1. User balance: 50 pts 2. Reward cost: 100 pts	1. Select reward (Coffee) 2. Validate balance 3. Check error	Balance: **50** Cost: **100** Valid: **No**	Error "Insufficient points. Need 50 more points"	Redemption blocked		
TC_Pts_13	TR-RDM-02: Validate that points are deducted correctly from user balance after redemption	Deduct points for **successful redemption**	1. Balance: 515 pts 2. Reward selected: Meal (500 pts)	1. Confirm redemption 2. Deduct 500 points 3. Update balance	Old Balance: **515** Deducted: **500** New: **15**	Balance updated to 15 points	User has 15 points remaining		
TC_Pts_14	TR-RDM-02: Validate that points are deducted correctly from user balance after redemption	Verify **atomic transaction** (rollback on failure)	1. Redemption started 2. Database error occurs	1. Start redemption 2. Simulate DB failure 3. Check rollback	Balance: **515** Error: **DB Timeout**	Transaction rolled back, balance unchanged at 515	User balance remains 515, no points lost		
TC_Pts_15	TR-RDM-03: Validate that unique QR code is generated for each redemption	Generate **unique QR code**	1. Redemption confirmed 2. QR generator active	1. Confirm redemption 2. Generate QR code 3. Verify uniqueness	Format: **QR-20251108-USER001-MEAL**	Unique QR code generated successfully	QR code ready for use		

TEST CASE ID	TEST SCENARIO	TEST CASE DESCRIPTION	PRE-CONDITION	TEST STEPS	TEST DATA	EXPECTED RESULT	POST CONDITION	ACTUAL RESULT	STATUS (PASS/FAIL
TC_Pts_16	TR-RDM-03: Validate that unique QR code is generated for each redemption	Verify **QR code expiration** set to 24 hours	1. QR generated at 14:00 2. Expiration policy: 24hrs	1. Generate QR code 2. Check expiration timestamp 3. Verify calculation	Generated: **14:00, Nov 8** Expires: **14:00, Nov 9**	Expiration set correctly to 24 hours from generation	QR expires at correct time		
TC_Pts_17	TR-RDM-04: Validate that redemption record is created with all necessary details	Create **redemption record** in database	1. Redemption completed 2. All data available	1. Complete redemption 2. Create database record 3. Verify fields	User: **ahmed@rvm.com** Reward: **Meal** QR: **Generated**	Record created with user ID, reward ID, QR, timestamp, status	Complete record in database		
TC_Pts_18	TR-RDM-04: Validate that redemption record is created with all necessary details	Set initial status as **"pending"**	1. Record created 2. Not yet redeemed at cafeteria	1. Create record 2. Check status field 3. Verify value	Status: **pending** Redeemed At: **NULL**	Status field set to "pending" correctly	Status awaits cafeteria scan		
TC_Pts_19	TR-RDM-05: Validate that QR code can be scanned and validated at cafeteria	Scan and validate **valid QR code**	1. QR code generated 2. Not expired 3. Status: pending	1. Scan QR at cafeteria 2. Validate in system 3. Check all conditions	QR: **QR-2025 1108-001** Status: **Valid**	Validation passes: exists, not expired, not redeemed	Ready to mark as redeemed		
TC_Pts_20	TR-RDM-05: Validate that QR code can be scanned and	Reject **expired QR code**	1. QR generated 25 hours ago	1. Scan expired QR 2. Check expiration	Generated: **Nov 7, 13:00** Current:	Error "QR code expired. Please	Redemption denied		

TEST CASE ID	TEST SCENARIO	TEST CASE DESCRIPTION	PRE-CONDITION	TEST STEPS	TEST DATA	EXPECTED RESULT	POST CONDITION	ACTUAL RESULT	STATUS (PASS/FAIL
	validated at cafeteria		2. Expiration: 24 hours	3. Reject validation	**Nov 8, 14:00**	contact support"			

5.3 Apply Point Expiration:

TEST CASE ID	TEST SCENARIO	TEST CASE DESCRIPTION	PRE-CONDITION	TEST STEPS	TEST DATA	EXPECTED RESULT	POST CONDITION	ACTUAL RESULT	STATUS (PASS/FAIL)
TC_Pts_21	TR-EXP-01: Validate that points expire after 12 months of inactivity	Expire points after **12 months** inactivity	1. User inactive for 365 days 2. Points earned: 300 3. Last activity: 2024-11-08	1. Run expiration job 2. Calculate days inactive 3. Apply expiration	User: **Ahmed** Points: **300** Inactive: **365 days**	300 points expired, balance set to 0	User balance = 0, expiration logged		
TC_Pts_22	TR-EXP-01: Validate that points expire after 12 months of inactivity	Do NOT expire points for **active users**	1. User inactive for 300 days 2. Points: 200	1. Run expiration job 2. Check inactivity period 3. Skip user	User: **Sara** Points: **200** Inactive: **300 days**	No points expired, balance remains 200	User keeps all 200 points		
TC_Pts_23	TR-EXP-02: Validate that FIFO (First In, First Out) expiration logic is applied	Apply **FIFO expiration** logic	1. User has 500 pts 2. 300 pts from 2024-01-01 3. 200 pts from 2024-06-01	1. Identify oldest points 2. Expire 300 pts first 3. Keep newer 200 pts	Old: **300 pts (2024-01-01)** New: **200 pts (2024-06-01)**	Oldest 300 points expired first, 200 remain	User balance = 200 points		
TC_Pts_24	TR-EXP-02: Validate that FIFO (First In, First Out) expiration logic is applied	Verify **expiration record** includes original earn date	1. Points expiring 2. Original date known	1. Expire points 2. Create record 3. Check earn date field	Expired: **300 pts** Earned On: **2024-11-08**	Record includes original earn date for audit	Complete expiration record created		
TC_Pts_25	TR-EXP-03: Validate that expiration notification is sent to	Send **expiration notification** to user	1. Points expired: 300 2. User email: active	1. Expire points 2. Compose notification 3. Send to user	User: **ahmed@rvm.com** Expired: **300 pts**	Email sent: "300 points expired due to inactivity"	User notified via email and in-app		

TEST CASE ID	TEST SCENARIO	TEST CASE DESCRIPTION	PRE-CONDITION	TEST STEPS	TEST DATA	EXPECTED RESULT	POST CONDITION	ACTUAL RESULT	STATUS (PASS/FAIL)
	affected users								
TC_Pts_26	TR-EXP-03: Validate that expiration notification is sent to affected users	Include **motivational message** in notification	1. Expiration occurred 2. Notification prepared	1. Create notification 2. Add encouragement 3. Send message	Message: **"Start recycling to earn new points!"**	Notification includes positive call-to-action	User encouraged to re-engage		
TC_Pts_27	TR-EXP-04: Validate that warning notification is sent before points expire (30 days)	Send **30-day warning** notification	1. Points expire in 30 days 2. User inactive for 335 days	1. Check upcoming expirations 2. Send warning 3. Log notification	User: **Sara** Points: **150** Days Left: **30**	Warning sent: "150 points expire in 30 days!"	User warned in advance		
TC_Pts_28	TR-EXP-04: Validate that warning notification is sent before points expire (30 days)	Send **15-day warning** notification	1. Points expire in 15 days 2. Previous warning sent	1. Check expiration date 2. Send urgent warning 3. Suggest action	User: **Sara** Points: **150** Days Left: **15**	Warning sent: "Use points or deposit items to extend"	User receives urgent reminder		
TC_Pts_29	TR-EXP-05: Validate that user activity resets	Reset expiration after **new transaction**	1. Points expiring in 15 days 2. User	1. User completes transaction 2. Update last activity	User: **Sara** Activity:	Last activity updated, expiration	Points safe for 12 more months		

TEST CASE ID	TEST SCENARIO	TEST CASE DESCRIPTION	PRE-CONDITION	TEST STEPS	TEST DATA	EXPECTED RESULT	POST CONDITION	ACTUAL RESULT	STATUS (PASS/FAIL)
	expiration countdown		deposits items	3. Reset countdown	**Nov 8, 2025**	reset to 12 months			
TC_Pts_30	TR-EXP-05: Validate that user activity resets expiration countdown	Cancel **pending expiration** after activity	1. Pending expiration scheduled 2. User becomes active	1. User deposits items 2. Check pending expirations 3. Cancel for user	User: **Sara** Pending: **Cancelled**	Pending expiration removed, activity logged	User's points preserved		

2. Integration Testing

Integration testing verifies that modules/classes work together correctly across subsystems. This plan is aligned with the project architecture and uses traceable integration test numbering (IT-*).

2.1 Main Subsystems/Components and Their Main Classes

UserAccountService:
• User Manager
• Profile Manager
• Account Validator
• Role Assignor
• User Orchestrator

AuthenticationService:
• Auth Manager
• Session Manager
• Authentication Validator
• Token Orchestrator

TransactionService:
• Transaction Manager
• Catalog Manager
• Transaction (Entity)
• Transaction Processor
• Transaction Orchestrator

PointService:
• Point Manager
• Point Transition Manager
• Point Calculator
• Point Validator
• Point Orchestrator

RewardService:
• Reward Manager
• Catalog Manager
• Redemption
• Availability Checker
• Reward Orchestrator

InventoryService:
• Inventory Manager
• Machine Manager
• Alert Manager
• Collection Manager
• Digital Twin Sync
• Report Generator

NotificationService:
• Notification Manager
• Channel Router
• Template Manager

• Event Processor
• Preference Manager
• Notification Scheduler

ANALYTICSSERVICE:

• Report Manager
• Data Aggregator
• Trend Analyzer
• Statistics Calculator
• Ranking Manager
• Cache Manager
• Export Manager

OTHER MAIN COMPONENTS INCLUDED IN INTEGRATION:

• API Controller (mobile app/admin dashboard endpoints)
• RVM System
• Input Module (QR scanner, weight sensor, material classifier, liquid detector)
• Output Module (display/audio/printer)
• Database + Offline Cache
• External Cafeteria POS (redemption scanning/validation)

2.2 Class-Based Traceability Matrix (Classes → UT/IT/RT)

This matrix is class-based (not subsystem-based). It shows how each class is covered by unit tests, integration tests, and regression tests.

Service	Class	Unit Tests (UT)	Integration Tests (IT)	Regression (RT)
UserAccountService	User Manager	UT-UA-02	IT-UA-01, IT-SYS-01	RT-01, RT-08
UserAccountService	Profile Manager	UT-UA-02	IT-UA-01, IT-SYS-01	RT-01, RT-08
UserAccountService	Account Validator	UT-UA-01	IT-UA-01, IT-SYS-01	RT-01, RT-08
UserAccountService	Role Assignor	UT-UA-01	IT-UA-01, IT-SYS-01	RT-01, RT-08
UserAccountService	User Orchestrator	UT-UA-02	IT-UA-01, IT-SYS-01	RT-01, RT-08
AuthenticationService	Auth Manager	UT-AUTH-01	IT-AUTH-01, IT-SYS-01	RT-01, RT-08
AuthenticationService	Session Manager	UT-AUTH-01	IT-AUTH-01, IT-SYS-01	RT-01, RT-08
AuthenticationService	Authentication Validator	UT-AUTH-01	IT-AUTH-01, IT-SYS-01	RT-01, RT-08
AuthenticationService	Token Orchestrator	UT-AUTH-02	IT-AUTH-01, IT-SYS-01	RT-01, RT-08
TransactionService	Transaction Manager	UT-TXN-02	IT-TXN-01, IT-SYS-02, IT-SYS-03, IT-	RT-02, RT-05, RT-08

			SYS-06, IT-SYS-07	
TransactionService	Catalog Manager	UT-TXN-01	IT-TXN-01, IT-SYS-02	RT-02, RT-05, RT-08
TransactionService	Transaction (Entity)	UT-TXN-01	IT-TXN-01, IT-SYS-02	RT-02, RT-05, RT-08
TransactionService	Transaction Processor	UT-TXN-01	IT-TXN-01, IT-SYS-02	RT-02, RT-05, RT-08
TransactionService	Transaction Orchestrator	UT-TXN-02	IT-TXN-01, IT-SYS-02, IT-SYS-03	RT-02, RT-05, RT-08
PointService	Point Manager	UT-PTS-02	IT-PTS-01, IT-SYS-03, IT-SYS-04, IT-SYS-07	RT-02, RT-05, RT-08
PointService	Point Transition Manager	UT-PTS-02	IT-PTS-01, IT-SYS-03	RT-02, RT-05, RT-08
PointService	Point Calculator	UT-PTS-01	IT-PTS-01, IT-SYS-03	RT-02, RT-05, RT-08
PointService	Point Validator	UT-PTS-01	IT-PTS-01, IT-SYS-03	RT-02, RT-05, RT-08
PointService	Point Orchestrator	UT-PTS-02	IT-PTS-01, IT-SYS-03, IT-SYS-04	RT-02, RT-05, RT-08
RewardService	Reward Manager	UT-RWD-02	IT-RWD-01, IT-SYS-04, IT-SYS-05	RT-03, RT-06, RT-13, RT-08
RewardService	Catalog Manager	UT-TXN-01	IT-TXN-01, IT-SYS-02	RT-02, RT-05, RT-08
RewardService	Redemption	UT-RWD-02	IT-RWD-01, IT-SYS-04	RT-03, RT-06, RT-13, RT-08
RewardService	Availability Checker	UT-RWD-01	IT-RWD-01, IT-SYS-04	RT-03, RT-06, RT-13, RT-08
RewardService	Reward Orchestrator	UT-RWD-02	IT-RWD-01, IT-SYS-04	RT-03, RT-06, RT-13, RT-08
InventoryService	Inventory Manager	UT-INV-02	IT-INV-01, IT-SYS-06, IT-SYS-07	RT-07, RT-12, RT-08
InventoryService	Machine Manager	UT-INV-02	IT-INV-01, IT-SYS-06	RT-07, RT-12, RT-08
InventoryService	Alert Manager	UT-INV-01	IT-INV-01, IT-SYS-06	RT-07, RT-12, RT-08
InventoryService	Collection Manager	UT-INV-02	IT-INV-01	RT-07, RT-12, RT-08
InventoryService	Digital Twin Sync	UT-INV-02	IT-INV-01	RT-09, RT-12, RT-08

InventoryService	Report Generator	UT-INV-02	IT-INV-01, IT-SYS-07	RT-07, RT-12, RT-08
NotificationService	Notification Manager	UT-NOTIF-02	IT-NOTIF-01, IT-SYS-05, IT-SYS-06	RT-06, RT-10, RT-11, RT-12, RT-13, RT-08
NotificationService	Channel Router	UT-NOTIF-01	IT-NOTIF-01, IT-SYS-05	RT-06, RT-10, RT-11, RT-12, RT-13, RT-08
NotificationService	Template Manager	UT-NOTIF-02	IT-NOTIF-01, IT-SYS-05	RT-06, RT-10, RT-11, RT-12, RT-13, RT-08
NotificationService	Event Processor	UT-NOTIF-02	IT-NOTIF-01, IT-SYS-05	RT-06, RT-10, RT-11, RT-12, RT-13, RT-08
NotificationService	Preference Manager	UT-NOTIF-01	IT-NOTIF-01	RT-06, RT-10, RT-11, RT-12, RT-13, RT-08
NotificationService	Notification Scheduler	UT-NOTIF-02	IT-NOTIF-01	RT-06, RT-10, RT-11, RT-12, RT-13, RT-08
AnalyticsService	Report Manager	UT-ANL-02	IT-ANL-01, IT-SYS-07	RT-07, RT-08
AnalyticsService	Data Aggregator	UT-ANL-01	IT-ANL-01, IT-SYS-07	RT-07, RT-08
AnalyticsService	Trend Analyzer	UT-ANL-01	IT-ANL-01	RT-07, RT-08
AnalyticsService	Statistics Calculator	UT-ANL-01	IT-ANL-01	RT-07, RT-08
AnalyticsService	Ranking Manager	UT-ANL-01	IT-ANL-01, IT-SYS-07	RT-07, RT-08
AnalyticsService	Cache Manager	UT-ANL-01	IT-ANL-01	RT-07, RT-08
AnalyticsService	Export Manager	UT-ANL-02	IT-ANL-01, IT-SYS-07	RT-07, RT-08

2.3 Adopted Integration Approach

Adopted approach: Bottom-Up integration.
Rationale: We first validate low-level classes (validators/calculators/routers/managers) in isolation, then integrate them into each subsystem's Orchestrator, then integrate subsystems together, and finally integrate the full system (API + RVM + external POS).

2.4 Bottom-Up Integration Testing Plan (By Steps)

• Step 1 (within each subsystem): Integrate Manager/Validator classes together and verify their combined behavior.
• Step 2 (within each subsystem): Integrate those internal components into the subsystem Orchestrator.

• Step 3 (between subsystems): Integrate services in dependency order: Accounts → Auth → Transaction → Points → Rewards → Notifications → Inventory → Analytics.
• Step 4 (system level): Integrate API Controller and the RVM Input/Output Modules with the services for end-to-end flows.

2.5 Integration Test Cases (Traceable Numbering)

1. IT-UA-01: Integrate Account Validator + Role Assignor + User Manager + Profile Manager, then connect to User Orchestrator (UserAccountService internal integration).
2. IT-AUTH-01: Integrate Authentication Validator + Auth Manager + Session Manager, then connect to Token Orchestrator (AuthenticationService internal integration).
3. IT-TXN-01: Integrate Transaction Entity + Catalog Manager + Transaction Processor, then connect to Transaction Orchestrator (TransactionService internal integration).
4. IT-PTS-01: Integrate Point Validator + Point Calculator + Point Manager, then connect to Point Orchestrator (PointService internal integration).
5. IT-RWD-01: Integrate Availability Checker + Redemption + Reward Manager, then connect to Reward Orchestrator (RewardService internal integration).
6. IT-INV-01: Integrate Machine Manager + Inventory Manager + Collection Manager + Digital Twin Sync, then Alert Manager + Report Generator (InventoryService internal integration).
7. IT-NOTIF-01: Integrate Template Manager + Preference Manager + Channel Router, then Notification Scheduler + Event Processor (NotificationService internal integration).
8. IT-ANL-01: Integrate Data Aggregator + Statistics Calculator + Trend Analyzer + Cache Manager, then Ranking Manager + Report Manager + Export Manager (AnalyticsService internal integration).
9. IT-SYS-01: User onboarding integration: API Controller ↔ UserAccountService ↔ AuthenticationService (register/login, sessions).
10. IT-SYS-02: Recycling transaction integration: RVM Input Module ↔ TransactionService ↔ Output Module (display/receipt).
11. IT-SYS-03: Points update integration: TransactionService ↔ PointService (rounding/threshold/bonuses) ↔ user balance update.
12. IT-SYS-04: Reward redemption integration: PointService ↔ RewardService ↔ Cafeteria POS (scan/validate) ↔ mark redemption complete.
13. IT-SYS-05: Notifications integration: Authentication/Transaction/Reward/Inventory events ↔ NotificationService (templates, channels, scheduling).
14. IT-SYS-06: Inventory monitoring integration: TransactionService ↔ InventoryService ↔ NotificationService (capacity alerts).
15. IT-SYS-07: Analytics integration: Transaction/Points/Inventory ↔ AnalyticsService (aggregation, rankings, export).
16. IT-SYS-08: Offline mode transaction: RVM caches session when offline, then syncs in chronological order when connectivity returns (no duplicates).
17. IT-SYS-09: Conflict resolution: duplicate transaction submissions are detected and reconciled; balances match server-side validation after sync.
18. IT-SYS-10: Guest mode deposit: RVM allows guest deposit without QR, applies validation rules, records guest transaction, awards no points.
19. IT-SYS-11: Points expiration: inactivity triggers expiration workflow; notifications sent; expired points deducted and logged.

20. IT-SYS-12: Multi-machine aggregation: two or more RVMs send transactions to central DB; leaderboards/college totals reflect all machines.
21. IT-SYS-13: Admin monitoring & alerts: machine offline/malfunction/capacity thresholds generate dashboard alerts + notifications.
22. IT-SYS-14: Cafeteria POS outage handling: redemption requests queue/retry gracefully; no double-spending; final state is consistent.

2.6 Integration Testing Exit Criteria

• All IT-* tests pass without critical defects.
• End-to-end transaction flow completes within required performance limits (e.g., QR validation ≤2s, receipt ≤3s where applicable).
• No data integrity issues (e.g., no duplicate transactions, no negative point balances).

2.7 Integration Scenario Coverage Matrix

This matrix ensures we explicitly test all crucial end-to-end scenarios and can prove coverage.

Scenario	Integration Test ID(s)	Regression Re-check (RT)
Registration + Login	IT-SYS-01	RT-04, RT-08
Recycling Transaction (RVM)	IT-SYS-02	RT-05, RT-08
Points Calculation + Balance Update	IT-SYS-03	RT-05, RT-08
Reward Redemption + POS Scan	IT-SYS-04	RT-06, RT-08
Notifications Delivery	IT-SYS-05	RT-06, RT-08
Inventory Capacity Alerts	IT-SYS-06	RT-07, RT-08
Analytics/Leaderboards/Exports	IT-SYS-07	RT-07, RT-08
Offline Operation + Sync	IT-SYS-08	RT-09, RT-08
Conflict Resolution After Sync	IT-SYS-09	RT-09, RT-08
Guest Mode Deposit	IT-SYS-10	RT-10, RT-08
Points Expiration Workflow	IT-SYS-11	RT-11, RT-08
Multi-Machine Management	IT-SYS-12	RT-12, RT-08
Admin Monitoring & Alerts	IT-SYS-13	RT-12, RT-08
POS Outage Queue/Retry	IT-SYS-14	RT-13, RT-08

3. Unit Testing

Unit testing verifies the behavior of individual classes in isolation using mocks/stubs for external dependencies. As requested, this section lists the MAIN unit tests required for the MAIN classes (not all possible classes). Each unit test has a traceable ID (UT-*).

3.1 Main Unit Tests (Selected Core Classes)

1. UT-UA-01: Account Validator validates required fields, uniqueness, and role assignment rules.
2. UT-UA-02: User Orchestrator coordinates User Manager + Profile Manager flows and handles failures consistently.
3. UT-AUTH-01: Authentication Validator validates credentials and enforces lockout after 3 failed attempts.
4. UT-AUTH-02: Token Orchestrator issues/refreshes/invalidates tokens and checks expiry.
5. UT-TXN-01: Transaction Processor processes items sequentially and finalizes transaction state correctly.
6. UT-TXN-02: Transaction Orchestrator handles cancel/complete logic and persists outcomes.
7. UT-PTS-01: Point Calculator applies rounding to nearest 10g and minimum deposit threshold of 50g.
8. UT-PTS-02: Point Orchestrator applies bonuses correctly and updates balances atomically.
9. UT-RWD-01: Availability Checker verifies reward availability and user eligibility.
10. UT-RWD-02: Redemption creates a unique confirmation and prevents double-spending.
11. UT-INV-01: Alert Manager triggers alerts at 80/90/100% capacity thresholds.
12. UT-INV-02: Digital Twin Sync synchronizes machine status and handles retries.
13. UT-NOTIF-01: Channel Router selects correct channel(s) (email/SMS/in-app) based on event severity and preferences.
14. UT-NOTIF-02: Notification Scheduler schedules/reminds for expiring rewards and retries failed sends.
15. UT-ANL-01: Data Aggregator aggregates transactions correctly by date range and college.
16. UT-ANL-02: Export Manager exports to CSV/PDF/Excel formats without data loss.

3.2 Unit Testing Plan (Execution)

• Run UT-* on every code change (CI) and before integration milestones.
• Use mocks for: Database, external Cafeteria POS, Notification gateways, and hardware sensor drivers.
• Exit criteria: all UT-* tests pass before executing IT-* tests that depend on them.

4. Regression Testing

Regression testing re-runs a defined set of stable tests after changes to ensure previously working functionality still works. Regression is applied at unit level (re-running UT-*), at integration level (re-running IT-*), and before releases.

4.1 Regression Test Cases (Traceable Numbering)

1. RT-01: Re-run UT-UA-* and UT-AUTH-* after any authentication/account change (prevents login/registration breakage).
2. RT-02: Re-run UT-TXN-* and UT-PTS-* after any transaction/points rule change.
3. RT-03: Re-run UT-RWD-* after any reward catalog/redemption change.
4. RT-04: Re-run IT-SYS-01 after changes to API/Auth/Accounts.
5. RT-05: Re-run IT-SYS-02 and IT-SYS-03 after changes to RVM transaction flow or point calculation.
6. RT-06: Re-run IT-SYS-04 and IT-SYS-05 after changes to rewards or notifications.
7. RT-07: Re-run IT-SYS-06 and IT-SYS-07 after changes to inventory monitoring or analytics reports.
8. RT-09: Re-run IT-SYS-08 and IT-SYS-09 after changes to offline caching, sync logic, or conflict resolution.
9. RT-10: Re-run IT-SYS-10 after changes to guest mode rules or logging.
10. RT-11: Re-run IT-SYS-11 after changes to points expiration policy or notifications.
11. RT-12: Re-run IT-SYS-12 and IT-SYS-13 after changes to multi-machine/admin monitoring/alerts.
12. RT-13: Re-run IT-SYS-14 after changes to cafeteria POS integration or outage/queue mechanism.
13. RT-08: Full regression pack: execute RT-01..RT-07 + a final end-to-end sanity run before Release Testing.

4.2 Regression Testing Plan

• On every bug fix: run the regression items related to the changed subsystem(s) (RT-01..RT-07 as applicable).
• After completing any IT-* milestone: immediately re-run the related RT-* suite to confirm stability.
• When adding a new feature: add/extend at least one RT-* case and include it in RT-08 full pack.
• Before releases: RT-08 must pass with no critical defects.

5. Overall Testing Plan (Class-Based Bottom-Up + Gantt)

This is the master plan that lists all planned tests (unit, integration, regression, and system-level testing), estimated durations, and dependencies. This section is the most important part of the test report.
Bottom-Up functional sequence (required template) with class names replacing F1/F2/F3:
F1 = User Orchestrator
F2 = Transaction Orchestrator
F3 = Reward Orchestrator

5.1 Master Test Schedule Tables (Per Subsystem – Bottom-Up by Classes)

Each table below is a bottom-up plan inside ONE subsystem: we test lower-level classes first (UT), then integrate them together inside the subsystem (IT), then run a subsystem regression (RT).

UserAccountService – Bottom-Up Schedule (All Class Integration Scenarios + Regression)

Step	Test	Type	Class/Scenario	Duration (days)	Depends On	Planned Start	Planned End
1	UT-UA-01	Unit	Account Validator	0.2	-	Day 0	Day 0.2
2	UT-UA-02	Unit	Profile Manager	0.2	1	Day 0.2	Day 0.4
3	UT-UA-03	Unit	Role Assignor	0.2	2	Day 0.4	Day 0.6
4	UT-UA-04	Unit	User Manager	0.2	3	Day 0.6	Day 0.8
5	UT-UA-05	Unit	User Orchestrator	0.2	4	Day 0.8	Day 1
6	IT-UA-01	Integration	Integrate: Account Validator + Profile Manager	0.35	1, 2	Day 0.4	Day 0.75
7	RT-UA-01	Regression	Re-check integration scenario IT-UA-01	0.15	6	Day 0.75	Day 0.9
8	IT-UA-02	Integration	Integrate: Account Validator + Profile Manager + Role Assignor	0.35	3, 6	Day 0.75	Day 1.1

9	RT-UA-02	Regression	Re-check integration scenario IT-UA-02	0.15	8	Day 1.1	Day 1.25
10	IT-UA-03	Integration	Integrate: Account Validator + Profile Manager + Role Assignor + User Manager	0.35	4, 8	Day 1.1	Day 1.45
11	RT-UA-03	Regression	Re-check integration scenario IT-UA-03	0.15	10	Day 1.45	Day 1.6
12	IT-UA-04	Integration	Integrate: Account Validator + Profile Manager + Role Assignor + User Manager + User Orchestrator	0.35	5, 10	Day 1.45	Day 1.8
13	RT-UA-04	Regression	Re-check integration scenario IT-UA-04	0.15	12	Day 1.8	Day 1.95

AuthenticationService – Bottom-Up Schedule (All Class Integration Scenarios + Regression)

Step	Test	Type	Class/Scenario	Duration (days)	Depends On	Planned Start	Planned End
1	UT-AUTH-01	Unit	Authentication Validator	0.2	-	Day 0	Day 0.2
2	UT-AUTH-02	Unit	Auth Manager	0.2	1	Day 0.2	Day 0.4
3	UT-AUTH-03	Unit	Session Manager	0.2	2	Day 0.4	Day 0.6

4	UT-AUTH-04	Unit	Token Orchestrator	0.2	3	Day 0.6	Day 0.8
5	IT-AUTH-01	Integration	Integrate: Authentication Validator + Auth Manager	0.35	1, 2	Day 0.4	Day 0.75
6	RT-AUTH-01	Regression	Re-check integration scenario IT-AUTH-01	0.15	5	Day 0.75	Day 0.9
7	IT-AUTH-02	Integration	Integrate: Authentication Validator + Auth Manager + Session Manager	0.35	3, 5	Day 0.75	Day 1.1
8	RT-AUTH-02	Regression	Re-check integration scenario IT-AUTH-02	0.15	7	Day 1.1	Day 1.25
9	IT-AUTH-03	Integration	Integrate: Authentication Validator + Auth Manager + Session Manager + Token Orchestrator	0.35	4, 7	Day 1.1	Day 1.45
10	RT-AUTH-03	Regression	Re-check integration scenario IT-AUTH-03	0.15	9	Day 1.45	Day 1.6

TransactionService – Bottom-Up Schedule (All Class Integration Scenarios + Regression)

Step	Test	Type	Class/Scenario	Duration (days)	Depends On	Planned Start	Planned End
1	UT-TXN-01	Unit	Transaction (Entity)	0.2	-	Day 0	Day 0.2
2	UT-TXN-02	Unit	Catalog Manager	0.2	1	Day 0.2	Day 0.4

3	UT-TXN-03	Unit	Transaction Manager	0.2	2	Day 0.4	Day 0.6
4	UT-TXN-04	Unit	Transaction Processor	0.2	3	Day 0.6	Day 0.8
5	UT-TXN-05	Unit	Transaction Orchestrator	0.2	4	Day 0.8	Day 1
6	IT-TXN-01	Integration	Integrate: Transaction (Entity) + Catalog Manager	0.35	1, 2	Day 0.4	Day 0.75
7	RT-TXN-01	Regression	Re-check integration scenario IT-TXN-01	0.15	6	Day 0.75	Day 0.9
8	IT-TXN-02	Integration	Integrate: Transaction (Entity) + Catalog Manager + Transaction Manager	0.35	3, 6	Day 0.75	Day 1.1
9	RT-TXN-02	Regression	Re-check integration scenario IT-TXN-02	0.15	8	Day 1.1	Day 1.25
10	IT-TXN-03	Integration	Integrate: Transaction (Entity) + Catalog Manager + Transaction Manager + Transaction Processor	0.35	4, 8	Day 1.1	Day 1.45
11	RT-TXN-03	Regression	Re-check integration scenario IT-TXN-03	0.15	10	Day 1.45	Day 1.6
12	IT-TXN-04	Integration	Integrate: Transaction (Entity) + Catalog	0.35	5, 10	Day 1.45	Day 1.8

			Manager + Transaction Manager + Transaction Processor + Transaction Orchestrator				
13	RT-TXN-04	Regression	Re-check integration scenario IT-TXN-04	0.15	12	Day 1.8	Day 1.95

PointService – Bottom-Up Schedule (All Class Integration Scenarios + Regression)

Step	Test	Type	Class/Scenario	Duration (days)	Depends On	Planned Start	Planned End
1	UT-PTS-01	Unit	Point Calculator	0.2	-	Day 0	Day 0.2
2	UT-PTS-02	Unit	Point Transition Manager	0.2	1	Day 0.2	Day 0.4
3	UT-PTS-03	Unit	Point Validator	0.2	2	Day 0.4	Day 0.6
4	UT-PTS-04	Unit	Point Manager	0.2	3	Day 0.6	Day 0.8
5	UT-PTS-05	Unit	Point Orchestrator	0.2	4	Day 0.8	Day 1
6	IT-PTS-01	Integration	Integrate: Point Calculator + Point Transition Manager	0.35	1, 2	Day 0.4	Day 0.75
7	RT-PTS-01	Regression	Re-check integration scenario IT-PTS-01	0.15	6	Day 0.75	Day 0.9
8	IT-PTS-02	Integration	Integrate: Point Calculator + Point Transition Manager + Point Validator	0.35	3, 6	Day 0.75	Day 1.1

9	RT-PTS-02	Regression	Re-check integration scenario IT-PTS-02	0.15	8	Day 1.1	Day 1.25
10	IT-PTS-03	Integration	Integrate: Point Calculator + Point Transition Manager + Point Validator + Point Manager	0.35	4, 8	Day 1.1	Day 1.45
11	RT-PTS-03	Regression	Re-check integration scenario IT-PTS-03	0.15	10	Day 1.45	Day 1.6
12	IT-PTS-04	Integration	Integrate: Point Calculator + Point Transition Manager + Point Validator + Point Manager + Point Orchestrator	0.35	5, 10	Day 1.45	Day 1.8
13	RT-PTS-04	Regression	Re-check integration scenario IT-PTS-04	0.15	12	Day 1.8	Day 1.95

RewardService – Bottom-Up Schedule (All Class Integration Scenarios + Regression)

Step	Test	Type	Class/Scenario	Duration (days)	Depends On	Planned Start	Planned End
1	UT-RWD-01	Unit	Availability Checker	0.2	-	Day 0	Day 0.2
2	UT-RWD-02	Unit	Catalog Manager	0.2	1	Day 0.2	Day 0.4
3	UT-RWD-03	Unit	Redemption	0.2	2	Day 0.4	Day 0.6

4	UT-RWD-04	Unit	Reward Manager	0.2	3	Day 0.6	Day 0.8
5	UT-RWD-05	Unit	Reward Orchestrator	0.2	4	Day 0.8	Day 1
6	IT-RWD-01	Integration	Integrate: Availability Checker + Catalog Manager	0.35	1, 2	Day 0.4	Day 0.75
7	RT-RWD-01	Regression	Re-check integration scenario IT-RWD-01	0.15	6	Day 0.75	Day 0.9
8	IT-RWD-02	Integration	Integrate: Availability Checker + Catalog Manager + Redemption	0.35	3, 6	Day 0.75	Day 1.1
9	RT-RWD-02	Regression	Re-check integration scenario IT-RWD-02	0.15	8	Day 1.1	Day 1.25
10	IT-RWD-03	Integration	Integrate: Availability Checker + Catalog Manager + Redemption + Reward Manager	0.35	4, 8	Day 1.1	Day 1.45
11	RT-RWD-03	Regression	Re-check integration scenario IT-RWD-03	0.15	10	Day 1.45	Day 1.6
12	IT-RWD-04	Integration	Integrate: Availability Checker + Catalog Manager + Redemption + Reward Manager +	0.35	5, 10	Day 1.45	Day 1.8

			Reward Orchestrator				
13	RT-RWD-04	Regression	Re-check integration scenario IT-RWD-04	0.15	12	Day 1.8	Day 1.95

InventoryService – Bottom-Up Schedule (All Class Integration Scenarios + Regression)

Step	Test	Type	Class/Scenario	Duration (days)	Depends On	Planned Start	Planned End
1	UT-INV-01	Unit	Alert Manager	0.2	-	Day 0	Day 0.2
2	UT-INV-02	Unit	Collection Manager	0.2	1	Day 0.2	Day 0.4
3	UT-INV-03	Unit	Digital Twin Sync	0.2	2	Day 0.4	Day 0.6
4	UT-INV-04	Unit	Inventory Manager	0.2	3	Day 0.6	Day 0.8
5	UT-INV-05	Unit	Machine Manager	0.2	4	Day 0.8	Day 1
6	UT-INV-06	Unit	Report Generator	0.2	5	Day 1	Day 1.2
7	IT-INV-01	Integration	Integrate: Alert Manager + Collection Manager	0.35	1, 2	Day 0.4	Day 0.75
8	RT-INV-01	Regression	Re-check integration scenario IT-INV-01	0.15	7	Day 0.75	Day 0.9
9	IT-INV-02	Integration	Integrate: Alert Manager + Collection Manager + Digital Twin Sync	0.35	3, 7	Day 0.75	Day 1.1
10	RT-INV-02	Regression	Re-check integration	0.15	9	Day 1.1	Day 1.25

			scenario IT-INV-02				
11	IT-INV-03	Integration	Integrate: Alert Manager + Collection Manager + Digital Twin Sync + Inventory Manager	0.35	4, 9	Day 1.1	Day 1.45
12	RT-INV-03	Regression	Re-check integration scenario IT-INV-03	0.15	11	Day 1.45	Day 1.6
13	IT-INV-04	Integration	Integrate: Alert Manager + Collection Manager + Digital Twin Sync + Inventory Manager + Machine Manager	0.35	5, 11	Day 1.45	Day 1.8
14	RT-INV-04	Regression	Re-check integration scenario IT-INV-04	0.15	13	Day 1.8	Day 1.95
15	IT-INV-05	Integration	Integrate: Alert Manager + Collection Manager + Digital Twin Sync + Inventory Manager + Machine Manager + Report Generator	0.35	6, 13	Day 1.8	Day 2.15
16	RT-INV-05	Regression	Re-check integration scenario IT-INV-05	0.15	15	Day 2.15	Day 2.3

NotificationService – Bottom-Up Schedule (All Class Integration Scenarios + Regression)

Step	Test	Type	Class/Scenario	Duration (days)	Depends On	Planned Start	Planned End
1	UT-NOTIF-01	Unit	Channel Router	0.2	-	Day 0	Day 0.2
2	UT-NOTIF-02	Unit	Event Processor	0.2	1	Day 0.2	Day 0.4
3	UT-NOTIF-03	Unit	Notification Manager	0.2	2	Day 0.4	Day 0.6
4	UT-NOTIF-04	Unit	Notification Scheduler	0.2	3	Day 0.6	Day 0.8
5	UT-NOTIF-05	Unit	Preference Manager	0.2	4	Day 0.8	Day 1
6	UT-NOTIF-06	Unit	Template Manager	0.2	5	Day 1	Day 1.2
7	IT-NOTIF-01	Integration	Integrate: Channel Router + Event Processor	0.35	1, 2	Day 0.4	Day 0.75
8	RT-NOTIF-01	Regression	Re-check integration scenario IT-NOTIF-01	0.15	7	Day 0.75	Day 0.9
9	IT-NOTIF-02	Integration	Integrate: Channel Router + Event Processor + Notification Manager	0.35	3, 7	Day 0.75	Day 1.1
10	RT-NOTIF-02	Regression	Re-check integration scenario IT-NOTIF-02	0.15	9	Day 1.1	Day 1.25
11	IT-NOTIF-03	Integration	Integrate: Channel Router + Event Processor + Notification	0.35	4, 9	Day 1.1	Day 1.45

			Manager + Notification Scheduler				
12	RT-NOTIF-03	Regression	Re-check integration scenario IT-NOTIF-03	0.15	11	Day 1.45	Day 1.6
13	IT-NOTIF-04	Integration	Integrate: Channel Router + Event Processor + Notification Manager + Notification Scheduler + Preference Manager	0.35	5, 11	Day 1.45	Day 1.8
14	RT-NOTIF-04	Regression	Re-check integration scenario IT-NOTIF-04	0.15	13	Day 1.8	Day 1.95
15	IT-NOTIF-05	Integration	Integrate: Channel Router + Event Processor + Notification Manager + Notification Scheduler + Preference Manager + Template Manager	0.35	6, 13	Day 1.8	Day 2.15
16	RT-NOTIF-05	Regression	Re-check integration scenario IT-NOTIF-05	0.15	15	Day 2.15	Day 2.3

AnalyticsService – Bottom-Up Schedule (All Class Integration Scenarios + Regression)

Step	Test	Type	Class/Scenario	Duration (days)	Depends On	Planned Start	Planned End
1	UT-ANL-01	Unit	Statistics Calculator	0.2	-	Day 0	Day 0.2

2	UT-ANL-02	Unit	Cache Manager	0.2	1	Day 0.2	Day 0.4
3	UT-ANL-03	Unit	Data Aggregator	0.2	2	Day 0.4	Day 0.6
4	UT-ANL-04	Unit	Export Manager	0.2	3	Day 0.6	Day 0.8
5	UT-ANL-05	Unit	Ranking Manager	0.2	4	Day 0.8	Day 1
6	UT-ANL-06	Unit	Report Manager	0.2	5	Day 1	Day 1.2
7	UT-ANL-07	Unit	Trend Analyzer	0.2	6	Day 1.2	Day 1.4
8	IT-ANL-01	Integration	Integrate: Statistics Calculator + Cache Manager	0.35	1, 2	Day 0.4	Day 0.75
9	RT-ANL-01	Regression	Re-check integration scenario IT-ANL-01	0.15	8	Day 0.75	Day 0.9
10	IT-ANL-02	Integration	Integrate: Statistics Calculator + Cache Manager + Data Aggregator	0.35	3, 8	Day 0.75	Day 1.1
11	RT-ANL-02	Regression	Re-check integration scenario IT-ANL-02	0.15	10	Day 1.1	Day 1.25
12	IT-ANL-03	Integration	Integrate: Statistics Calculator + Cache Manager + Data Aggregator +	0.35	4, 10	Day 1.1	Day 1.45

			Export Manager				
13	RT-ANL-03	Regression	Re-check integration scenario IT-ANL-03	0.15	12	Day 1.45	Day 1.6
14	IT-ANL-04	Integration	Integrate: Statistics Calculator + Cache Manager + Data Aggregator + Export Manager + Ranking Manager	0.35	5, 12	Day 1.45	Day 1.8
15	RT-ANL-04	Regression	Re-check integration scenario IT-ANL-04	0.15	14	Day 1.8	Day 1.95
16	IT-ANL-05	Integration	Integrate: Statistics Calculator + Cache Manager + Data Aggregator + Export Manager + Ranking Manager + Report Manager	0.35	6, 14	Day 1.8	Day 2.15
17	RT-ANL-05	Regression	Re-check integration scenario IT-ANL-05	0.15	16	Day 2.15	Day 2.3
18	IT-ANL-06	Integration	Integrate: Statistics Calculator + Cache Manager + Data Aggregator + Export	0.35	7, 16	Day 2.15	Day 2.5

			Manager + Ranking Manager + Report Manager + Trend Analyzer				
19	RT-ANL-06	Regression	Re-check integration scenario IT-ANL-06	0.15	18	Day 2.5	Day 2.65

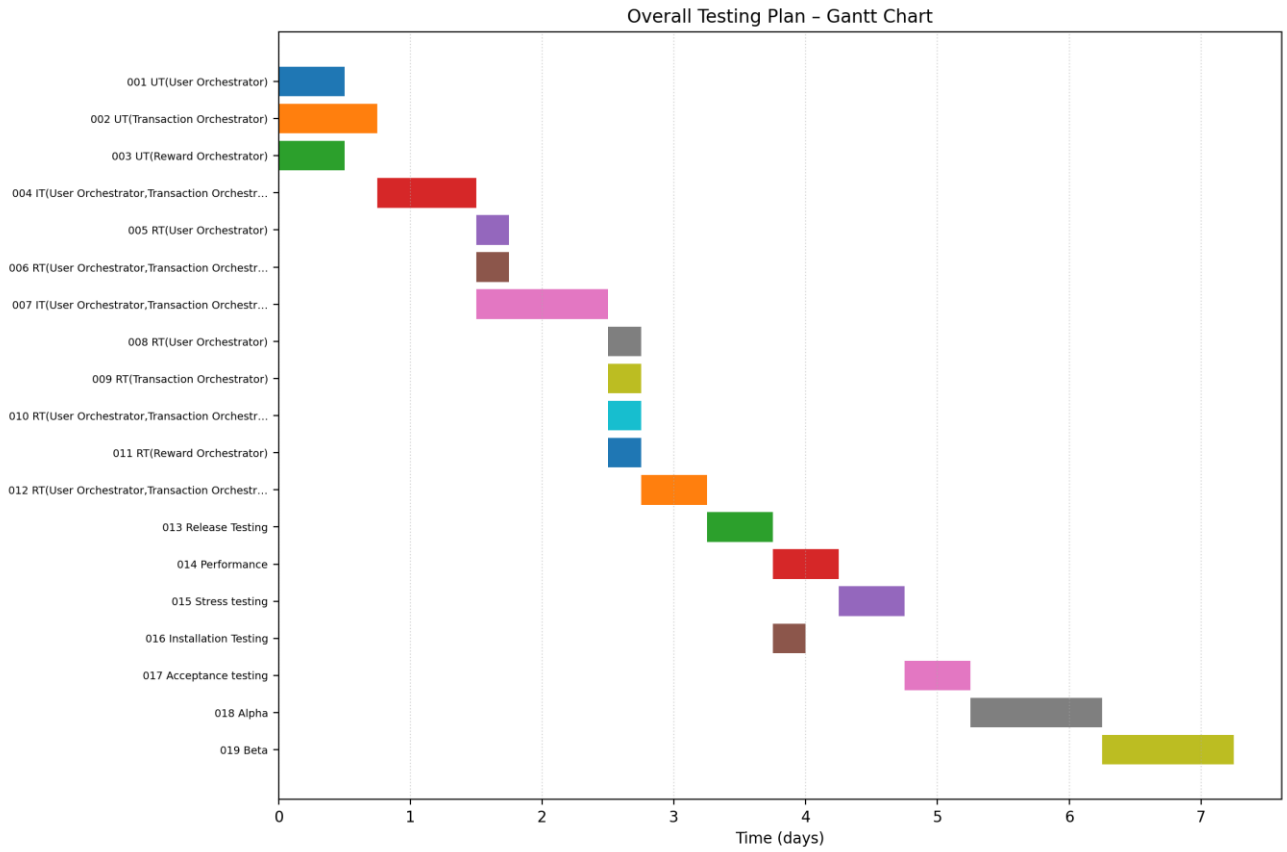
5.2 Overall Project Master Schedule (Template 1–19)

ID	Test	Type	Scope/Feature	Duration (days)	Depends On	Planned Start	Planned End
1	UT(User Orchestrator)	Unit	Execute UT suite covering User Orchestrator and its supporting classes	0.5	-	Day 0	Day 0.5
2	UT(Transaction Orchestrator)	Unit	Execute UT suite covering Transaction Orchestrator and its supporting classes	0.75	-	Day 0	Day 0.75
3	UT(Reward Orchestrator)	Unit	Execute UT suite covering Reward Orchestrator and its supporting classes	0.5	-	Day 0	Day 0.5
4	IT(User Orchestrator,Transaction Orchestrator)	Integration	Integrate User Orchestrator ↔ Transaction Orchestrator (login/onboarding then recycling transaction)	0.75	1, 2	Day 0.75	Day 1.5
5	RT(User Orchestrator)	Regression	Regression on User Orchestrator after IT changes	0.25	4	Day 1.5	Day 1.75
6	RT(User Orchestrator,Transaction Orchestrator)	Regression	Regression on User Orchestrator+Transaction Orchestrator integration	0.25	4	Day 1.5	Day 1.75
7	IT(User Orchestrator,Transaction	Integration	Full integration User Orchestrator ↔ Transaction	1	4, 3	Day 1.5	Day 2.5

	Orchestrator,Reward Orchestrator)		Orchestrator ↔ Reward Orchestrator (earn points then redeem)				
8	RT(User Orchestrator)	Regression	Re-check User Orchestrator after full integration	0.25	7	Day 2.5	Day 2.75
9	RT(Transaction Orchestrator)	Regression	Re-check Transaction Orchestrator after full integration	0.25	7	Day 2.5	Day 2.75
10	RT(User Orchestrator,Transaction Orchestrator)	Regression	Re-check User Orchestrator+Transaction Orchestrator after full integration	0.25	7	Day 2.5	Day 2.75
11	RT(Reward Orchestrator)	Regression	Re-check Reward Orchestrator after full integration	0.25	7	Day 2.5	Day 2.75
12	RT(User Orchestrator,Transaction Orchestrator,Reward Orchestrator)	Regression	Full functional regression pack	0.5	8, 9, 11	Day 2.75	Day 3.25
13	Release Testing	System	Release candidate verification	0.5	12	Day 3.25	Day 3.75
14	Performance	Non-Functional	Performance checks (timings/throughput)	0.5	13	Day 3.75	Day 4.25
15	Stress testing	Non-Functional	Stress/endurance checks	0.5	14	Day 4.25	Day 4.75
16	Installation Testing	System	Deployment + setup verification	0.25	13	Day 3.75	Day 4
17	Acceptance testing	System	Stakeholder acceptance	0.5	15, 16	Day 4.75	Day 5.25
18	Alpha	Release	Internal pilot	1	17	Day 5.25	Day 6.25
19	Beta	Release	External/limited pilot	1	18	Day 6.25	Day 7.25

5.3 Gantt Chart

Generated Gantt chart image:



5.4 Execution Notes

• Dependencies enforce the required order: Unit → Integration → Regression → Release/NFR/Acceptance.
• Unit plan is class-based (UT(ClassName)) to match the architecture class design and bottom-up approach.
• If multiple team members are available, class-based unit tests can be executed in parallel to shorten the schedule.
• Update actual start/end dates and replace estimated durations after execution.

6. Traceability Matrices (Class-Based)

The following matrices are class-based to ensure traceability from design classes to tests.

6.1 Class → Unit Test Traceability

Service	Class	UT IDs
UserAccountService	User Manager	UT-UA-02
UserAccountService	Profile Manager	UT-UA-02
UserAccountService	Account Validator	UT-UA-01
UserAccountService	Role Assignor	UT-UA-01
UserAccountService	User Orchestrator	UT-UA-02
AuthenticationService	Auth Manager	UT-AUTH-01
AuthenticationService	Session Manager	UT-AUTH-01
AuthenticationService	Authentication Validator	UT-AUTH-01
AuthenticationService	Token Orchestrator	UT-AUTH-02
TransactionService	Transaction Manager	UT-TXN-02
TransactionService	Catalog Manager	UT-TXN-01
TransactionService	Transaction (Entity)	UT-TXN-01
TransactionService	Transaction Processor	UT-TXN-01
TransactionService	Transaction Orchestrator	UT-TXN-02
PointService	Point Manager	UT-PTS-02
PointService	Point Transition Manager	UT-PTS-02
PointService	Point Calculator	UT-PTS-01
PointService	Point Validator	UT-PTS-01
PointService	Point Orchestrator	UT-PTS-02
RewardService	Reward Manager	UT-RWD-02
RewardService	Catalog Manager	UT-TXN-01
RewardService	Redemption	UT-RWD-02
RewardService	Availability Checker	UT-RWD-01
RewardService	Reward Orchestrator	UT-RWD-02
InventoryService	Inventory Manager	UT-INV-02
InventoryService	Machine Manager	UT-INV-02
InventoryService	Alert Manager	UT-INV-01
InventoryService	Collection Manager	UT-INV-02
InventoryService	Digital Twin Sync	UT-INV-02
InventoryService	Report Generator	UT-INV-02
NotificationService	Notification Manager	UT-NOTIF-02
NotificationService	Channel Router	UT-NOTIF-01
NotificationService	Template Manager	UT-NOTIF-02
NotificationService	Event Processor	UT-NOTIF-02
NotificationService	Preference Manager	UT-NOTIF-01
NotificationService	Notification Scheduler	UT-NOTIF-02
AnalyticsService	Report Manager	UT-ANL-02
AnalyticsService	Data Aggregator	UT-ANL-01
AnalyticsService	Trend Analyzer	UT-ANL-01

AnalyticsService	Statistics Calculator	UT-ANL-01
AnalyticsService	Ranking Manager	UT-ANL-01
AnalyticsService	Cache Manager	UT-ANL-01
AnalyticsService	Export Manager	UT-ANL-02

6.2 Class → Integration Test Traceability

Service	Class	IT IDs
UserAccountService	User Manager	IT-UA-01, IT-SYS-01
UserAccountService	Profile Manager	IT-UA-01, IT-SYS-01
UserAccountService	Account Validator	IT-UA-01, IT-SYS-01
UserAccountService	Role Assignor	IT-UA-01, IT-SYS-01
UserAccountService	User Orchestrator	IT-UA-01, IT-SYS-01
AuthenticationService	Auth Manager	IT-AUTH-01, IT-SYS-01
AuthenticationService	Session Manager	IT-AUTH-01, IT-SYS-01
AuthenticationService	Authentication Validator	IT-AUTH-01, IT-SYS-01
AuthenticationService	Token Orchestrator	IT-AUTH-01, IT-SYS-01
TransactionService	Transaction Manager	IT-TXN-01, IT-SYS-02, IT-SYS-03, IT-SYS-06, IT-SYS-07
TransactionService	Catalog Manager	IT-TXN-01, IT-SYS-02
TransactionService	Transaction (Entity)	IT-TXN-01, IT-SYS-02
TransactionService	Transaction Processor	IT-TXN-01, IT-SYS-02
TransactionService	Transaction Orchestrator	IT-TXN-01, IT-SYS-02, IT-SYS-03
PointService	Point Manager	IT-PTS-01, IT-SYS-03, IT-SYS-04, IT-SYS-07
PointService	Point Transition Manager	IT-PTS-01, IT-SYS-03
PointService	Point Calculator	IT-PTS-01, IT-SYS-03
PointService	Point Validator	IT-PTS-01, IT-SYS-03
PointService	Point Orchestrator	IT-PTS-01, IT-SYS-03, IT-SYS-04
RewardService	Reward Manager	IT-RWD-01, IT-SYS-04, IT-SYS-05
RewardService	Catalog Manager	IT-TXN-01, IT-SYS-02
RewardService	Redemption	IT-RWD-01, IT-SYS-04
RewardService	Availability Checker	IT-RWD-01, IT-SYS-04
RewardService	Reward Orchestrator	IT-RWD-01, IT-SYS-04
InventoryService	Inventory Manager	IT-INV-01, IT-SYS-06, IT-SYS-07
InventoryService	Machine Manager	IT-INV-01, IT-SYS-06
InventoryService	Alert Manager	IT-INV-01, IT-SYS-06
InventoryService	Collection Manager	IT-INV-01
InventoryService	Digital Twin Sync	IT-INV-01
InventoryService	Report Generator	IT-INV-01, IT-SYS-07

NotificationService	Notification Manager	IT-NOTIF-01, IT-SYS-05, IT-SYS-06
NotificationService	Channel Router	IT-NOTIF-01, IT-SYS-05
NotificationService	Template Manager	IT-NOTIF-01, IT-SYS-05
NotificationService	Event Processor	IT-NOTIF-01, IT-SYS-05
NotificationService	Preference Manager	IT-NOTIF-01
NotificationService	Notification Scheduler	IT-NOTIF-01
AnalyticsService	Report Manager	IT-ANL-01, IT-SYS-07
AnalyticsService	Data Aggregator	IT-ANL-01, IT-SYS-07
AnalyticsService	Trend Analyzer	IT-ANL-01
AnalyticsService	Statistics Calculator	IT-ANL-01
AnalyticsService	Ranking Manager	IT-ANL-01, IT-SYS-07
AnalyticsService	Cache Manager	IT-ANL-01
AnalyticsService	Export Manager	IT-ANL-01, IT-SYS-07

6.3 Class → Regression Test Traceability

Service	Class	RT IDs
UserAccountService	User Manager	RT-01, RT-08
UserAccountService	Profile Manager	RT-01, RT-08
UserAccountService	Account Validator	RT-01, RT-08
UserAccountService	Role Assignor	RT-01, RT-08
UserAccountService	User Orchestrator	RT-01, RT-08
AuthenticationService	Auth Manager	RT-01, RT-08
AuthenticationService	Session Manager	RT-01, RT-08
AuthenticationService	Authentication Validator	RT-01, RT-08
AuthenticationService	Token Orchestrator	RT-01, RT-08
TransactionService	Transaction Manager	RT-02, RT-05, RT-08
TransactionService	Catalog Manager	RT-02, RT-05, RT-08
TransactionService	Transaction (Entity)	RT-02, RT-05, RT-08
TransactionService	Transaction Processor	RT-02, RT-05, RT-08
TransactionService	Transaction Orchestrator	RT-02, RT-05, RT-08
PointService	Point Manager	RT-02, RT-05, RT-08
PointService	Point Transition Manager	RT-02, RT-05, RT-08
PointService	Point Calculator	RT-02, RT-05, RT-08
PointService	Point Validator	RT-02, RT-05, RT-08
PointService	Point Orchestrator	RT-02, RT-05, RT-08
RewardService	Reward Manager	RT-03, RT-06, RT-13, RT-08
RewardService	Catalog Manager	RT-02, RT-05, RT-08
RewardService	Redemption	RT-03, RT-06, RT-13, RT-08
RewardService	Availability Checker	RT-03, RT-06, RT-13, RT-08
RewardService	Reward Orchestrator	RT-03, RT-06, RT-13, RT-08
InventoryService	Inventory Manager	RT-07, RT-12, RT-08
InventoryService	Machine Manager	RT-07, RT-12, RT-08
InventoryService	Alert Manager	RT-07, RT-12, RT-08
InventoryService	Collection Manager	RT-07, RT-12, RT-08

InventoryService	Digital Twin Sync	RT-09, RT-12, RT-08
InventoryService	Report Generator	RT-07, RT-12, RT-08
NotificationService	Notification Manager	RT-06, RT-10, RT-11, RT-12, RT-13, RT-08
NotificationService	Channel Router	RT-06, RT-10, RT-11, RT-12, RT-13, RT-08
NotificationService	Template Manager	RT-06, RT-10, RT-11, RT-12, RT-13, RT-08
NotificationService	Event Processor	RT-06, RT-10, RT-11, RT-12, RT-13, RT-08
NotificationService	Preference Manager	RT-06, RT-10, RT-11, RT-12, RT-13, RT-08
NotificationService	Notification Scheduler	RT-06, RT-10, RT-11, RT-12, RT-13, RT-08
AnalyticsService	Report Manager	RT-07, RT-08
AnalyticsService	Data Aggregator	RT-07, RT-08
AnalyticsService	Trend Analyzer	RT-07, RT-08
AnalyticsService	Statistics Calculator	RT-07, RT-08
AnalyticsService	Ranking Manager	RT-07, RT-08
AnalyticsService	Cache Manager	RT-07, RT-08
AnalyticsService	Export Manager	RT-07, RT-08

6.4 Class → Overall Plan Traceability

This matrix links each class to the Overall Testing Plan IDs (1–12) where it is exercised, using the required template.

Service	Class	Overall Plan IDs
UserAccountService	User Manager	1, 4, 5, 6, 7, 8, 10, 12
UserAccountService	Profile Manager	1, 4, 5, 6, 7, 8, 10, 12
UserAccountService	Account Validator	1, 4, 5, 6, 7, 8, 10, 12
UserAccountService	Role Assignor	1, 4, 5, 6, 7, 8, 10, 12
UserAccountService	User Orchestrator	1, 4, 5, 6, 7, 8, 10, 12
AuthenticationService	Auth Manager	1, 4, 5, 6, 7, 8, 10, 12
AuthenticationService	Session Manager	1, 4, 5, 6, 7, 8, 10, 12
AuthenticationService	Authentication Validator	1, 4, 5, 6, 7, 8, 10, 12
AuthenticationService	Token Orchestrator	1, 4, 5, 6, 7, 8, 10, 12
TransactionService	Transaction Manager	2, 4, 6, 7, 9, 10, 12
TransactionService	Catalog Manager	2, 4, 6, 7, 9, 10, 12
TransactionService	Transaction (Entity)	2, 4, 6, 7, 9, 10, 12
TransactionService	Transaction Processor	2, 4, 6, 7, 9, 10, 12
TransactionService	Transaction Orchestrator	2, 4, 6, 7, 9, 10, 12
PointService	Point Manager	2, 4, 6, 7, 9, 10, 12
PointService	Point Transition Manager	2, 4, 6, 7, 9, 10, 12
PointService	Point Calculator	2, 4, 6, 7, 9, 10, 12
PointService	Point Validator	2, 4, 6, 7, 9, 10, 12
PointService	Point Orchestrator	2, 4, 6, 7, 9, 10, 12

RewardService	Reward Manager	3, 7, 11, 12
RewardService	Catalog Manager	3, 7, 11, 12
RewardService	Redemption	3, 7, 11, 12
RewardService	Availability Checker	3, 7, 11, 12
RewardService	Reward Orchestrator	3, 7, 11, 12
InventoryService	Inventory Manager	2, 7, 9, 10, 12
InventoryService	Machine Manager	2, 7, 9, 10, 12
InventoryService	Alert Manager	2, 7, 9, 10, 12
InventoryService	Collection Manager	2, 7, 9, 10, 12
InventoryService	Digital Twin Sync	2, 7, 9, 10, 12
InventoryService	Report Generator	2, 7, 9, 10, 12
NotificationService	Notification Manager	3, 7, 11, 12
NotificationService	Channel Router	3, 7, 11, 12
NotificationService	Template Manager	3, 7, 11, 12
NotificationService	Event Processor	3, 7, 11, 12
NotificationService	Preference Manager	3, 7, 11, 12
NotificationService	Notification Scheduler	3, 7, 11, 12
AnalyticsService	Report Manager	7, 12
AnalyticsService	Data Aggregator	7, 12
AnalyticsService	Trend Analyzer	7, 12
AnalyticsService	Statistics Calculator	7, 12
AnalyticsService	Ranking Manager	7, 12
AnalyticsService	Cache Manager	7, 12
AnalyticsService	Export Manager	7, 12