CSE251 – " Project Report "

1. Project Title

Smart Reverse Vending Machine with Gamification System

2. Project Description

The proposed project aims to design and implement a smart reverse vending machine (RVM) that encourages university students to recycle plastic waste through gamification and rewards. Students scan their unique QR codes, deposit plastic recyclables, and earn points based on weight. The system creates multi-level competition (student vs. student, college vs. college) with leaderboards to drive engagement.

Points earned can be redeemed at the university cafeteria for food and beverages, creating a closed-loop incentive system. The RVM is powered by Raspberry Pi with integrated sensors (QR scanner, weight sensor, material classifier, liquid detector) and operates with offline capability, automatically synchronizing data when connectivity is restored.

Material Classification: AI-powered sensors automatically identify whether deposited items are plastic or non-plastic waste. Non-plastic materials are diverted to a separate waste compartment, preventing contamination of recyclable materials.

Liquid Detection: Advanced weight and tilt sensors detect liquid content inside plastic bottles. Bottles containing liquid are automatically rejected with clear instructions displayed to users to empty them first, ensuring recycling quality and preventing machine damage.

The RVM includes a "Guest" button allowing unregistered campus members (doctors, TAs, faculty, staff, visitors) to contribute to recycling efforts without registration. Guest deposits contribute to overall campus sustainability metrics, promoting inclusive environmental responsibility across the entire university community.

The system consists of a mobile application for students to view their QR codes, track points, check leaderboards, and redeem rewards. Administrators monitor system performance, manage inventory, view analytics through a web dashboard, and manage multiple RVM machines deployed across the university campus. The system automatically alerts administrators when machine capacity reaches 80% for timely maintenance and emptying. Administrators can also manage cafeteria rewards and view detailed reports on transactions, rankings, and environmental impact.

By combining IoT hardware, AI-based material validation, gamification, renewable energy awareness, and smart campus technologies, this project transforms waste management into an engaging, competitive, and eco-friendly campus service available to ALL campus members. **The system architecture follows SOLID principles to ensure maintainability, scalability, and extensibility.**

> ***"This project demonstrates the integration of IoT, AI validation, gamification, and sustainable practices into a scalable and eco-friendly campus recycling solution, designed with SOLID principles for robust software architecture and inclusive campus-wide participation."***

3. Team Members

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

ID	Name
23101899	Mohamed Adel Abdulrahman

4. Functional Requirements

"The following requirements are organized hierarchically with 29 high-level functional requirements, each containing specific low-level implementation details."

1. User Registration Authentication Management

1.1 The system shall allow students to register using university email, student ID, name, and college affiliation

1.2 The system shall validate university email domains before registration completion

1.3 The system shall authenticate users securely using encrypted credentials

1.4 The system shall provide password recovery functionality via email verification

2. QR Code Generation Management

2.1 The system shall generate unique QR codes for each registered student

2.2 The system shall link QR codes permanently to user accounts

2.3 The system shall allow users to regenerate QR codes if compromised

2.4 The system shall display QR codes in the mobile application for easy scanning

3. QR Code Scanning Validation at RVM

3.1 The RVM machine shall scan QR codes using integrated camera module

3.2 The system shall validate scanned QR codes against the user database

3.3 The system shall reject invalid or expired QR codes with appropriate error messages

3.4 The system shall complete QR validation within 2 seconds

4. Material Classification Validation at RVM

4.1 The RVM shall use AI-powered sensors/camera to classify deposited items as plastic or non-plastic

4.2 The system shall achieve minimum 90% accuracy in material classification

4.3 The system shall automatically divert non-plastic waste to a separate waste compartment

4.4 The system shall log classification results for quality analytics

5. Liquid Detection Bottle Validation

5.1 The RVM shall detect liquid content inside plastic bottles using weight differential and tilt sensors

5.2 The system shall reject bottles containing liquid automatically

5.3 The system shall display clear instructions to users to empty bottles before resubmission

5.4 The system shall log all rejected items with rejection reasons

6. Weight Measurement Accuracy

6.1 The system shall measure weight of deposited plastic using calibrated load cell sensors

6.2 The system shall maintain measurement accuracy within ±5g tolerance

6.3 The system shall automatically calibrate sensors weekly

6.4 The system shall detect and report sensor malfunctions to administrators

7. Points Calculation System

7.1 The system shall calculate base points as 10 points per kilogram of plastic deposited

7.2 The system shall round weight measurements to nearest 10 grams for point calculation

7.3 The system shall apply minimum deposit threshold of 50 grams to earn points

7.4 The system shall calculate and display points within 3 seconds of deposit

8. Bonus Points Incentive System

8.1 The system shall award 20% bonus for first deposit of the day
8.2 The system shall award 50% bonus for bulk deposits exceeding 5kg
8.3 The system shall award 30% bonus for maintaining weekly deposit streak (7 consecutive days)
8.4 The system shall clearly display applicable bonuses during transaction confirmation

9. User Points Balance Management

9.1 The system shall maintain separate tracking for total earned points, redeemed points, and available points
9.2 The system shall update point balances in real-time after each transaction
9.3 The system shall provide point balance query functionality via mobile app
9.4 The system shall prevent negative balance scenarios with validation checks

10. Transaction Recording History

10.1 The system shall record all transactions including user ID, timestamp, weight, points earned, and machine location
10.2 The system shall store complete transaction history for minimum 2 months
10.3 The system shall provide transaction history access through mobile application
10.4 The system shall allow users to export their transaction history in CSV format

11. Real-Time Feedback Display

11.1 The RVM shall display transaction details (weight, points earned) within 3 seconds of deposit
11.2 The system shall provide animated visual feedback for successful deposits
11.3 The system shall display current user ranking and points balance after transaction
11.4 The system shall provide audio confirmation for successful transactions

12. Student Leaderboard System

12.1 The system shall maintain real-time leaderboards ranking students within their college

12.2 The system shall update leaderboard positions immediately after each transaction

12.3 The system shall show top 100 students in each college leaderboard

12.4 The system shall show user's current rank and points difference from next rank

13. College Competition System

13.1 The system shall aggregate points by college affiliation

13.2 The system shall maintain university-wide college leaderboard

13.3 The system shall calculate college rankings based on total points from all students

13.4 The system shall display college competition standings in mobile app and RVM displays

14. Time-Based Leaderboard Filtering

14.1 The system shall provide leaderboard views filtered by daily, weekly, monthly, and all-time periods

14.2 The system shall reset daily and weekly leaderboards automatically at midnight

14.3 The system shall archive historical leaderboard data for reporting purposes

14.4 The system shall allow users to switch between different time period views seamlessly

15. Badge Achievement System

15.1 The system shall highlight top 10 students with special badges (Gold, Silver, Bronze)

15.2 The system shall award "Top 3 College" badges to leading colleges

15.3 The system shall display badges prominently in user profiles and leaderboards

15.4 The system shall send notifications when users earn new badges

16. Cafeteria Integration System

16.1 The system shall integrate with university cafeteria POS system via secure API

16.2 The system shall synchronize available rewards with cafeteria menu items

16.3 The system shall support real-time point redemption at cafeteria checkout

16.4 The system shall handle cafeteria system outages gracefully with queue mechanism

17. Points Redemption Processing

17.1 The system shall validate user's available points before processing redemption
17.2 The system shall process point deduction atomically to prevent double-spending
17.3 The system shall generate unique confirmation codes for each redemption
17.4 The system shall complete redemption transactions within 5 seconds

18. Redemption History Management

18.1 The system shall record all redemption transactions with item details and points spent
18.2 The system shall display redemption history in mobile application
18.3 The system shall send email receipts for all redemption transactions
18.4 The system shall retain redemption history for minimum 2 months

19. Offline Mode Operation

19.1 The RVM shall keep operating when internet connectivity is unavailable
19.2 The system shall store transactions locally during offline periods
19.3 The system shall cache essential data (user QR codes, point balances) for offline validation
19.4 The system shall display offline status indicator on RVM screen

20. Automatic Data Synchronization

20.1 The system shall automatically detect when internet connectivity is restored
20.2 The system shall upload cached offline transactions with central server in chronological order
20.3 The system shall implement retry mechanism for failed synchronization attempts
20.4 The system shall notify administrators of synchronization status

21. Conflict Resolution System

21.1 The system shall detect and prevent duplicate transaction submissions
21.2 The system shall resolve timestamp conflicts using server-side validation
21.3 The system shall reconcile point balances after synchronization completion
21.4 The system shall log all conflict resolution actions for auditing

22. Guest Mode System

22.1 The RVM shall display prominent "Guest" button on main interface
22.2 The system shall allow unregistered users (faculty, staff, visitors) to deposit recyclables without QR scanning
22.3 The system shall apply same validation rules (material classification, liquid detection) to guest deposits
22.4 The system shall record guest transactions with timestamp, weight, and machine location marked as "guest"
22.5 The system shall display thank you message showing environmental impact contribution
22.6 The system shall not award points for guest deposits

23. Multi-Machine Management System

23.1 The system shall support management of multiple RVM machines across campus
23.2 The system shall assign unique identifiers to each RVM machine
23.3 The system shall centralize data from all machines in unified database
23.4 The system shall allow remote configuration updates to all machines

24. Points Expiration Policy

24.1 The system shall expire unused points after 15 days of account inactivity
24.2 The system shall send reminder notifications 7 days before points expiration
24.3 The system shall send final warning notification 3 days before expiration
24.4 The system shall execute expiration process automatically at midnight on expiration date
24.5 The system shall log all expired points transactions for auditing

25. Analytics Generation (Administrator)

25.1 The system shall generate detailed performance reports (transaction volumes, engagement metrics)
25.2 The system shall generate student ranking reports by college and time period
25.3 The system shall show college competition standing reports
25.4 The system shall generate environmental impact reports (total weight, CO2 savings)
25.5 The system shall generate guest usage analytics reports
25.6 The system shall generate rejection analytics for quality monitoring
25.7 The system shall export reports in PDF, Excel, and CSV formats
25.8 The system shall schedule automated report generation and email delivery

26. Reward Management (Administrator)

26.1 The system shall allow admins to create new cafeteria reward items with point costs
26.2 The system shall allow admins to update existing reward details (name, description, points)
26.3 The system shall allow admins to delete or discontinue reward items
26.4 The system shall allow admins to set reward availability status (active/inactive)
26.5 The system shall allow admins to view redemption statistics for each reward
26.6 The system shall allow admins to manage reward categories and groupings
26.7 The system shall allow admins to set seasonal or promotional reward offerings

27. Machine Management (Administrator)

27.1 The system shall allow admins to monitor real-time status of all RVM machines (online/offline/maintenance)
27.2 The system shall display locations of all machines on an interactive campus map
27.3 The system shall allow admins to check individual machine capacity levels (bin + waste)
27.4 The system shall allow admins to view machine health metrics (sensor status, error logs)
27.5 The system shall allow admins to remotely enable/disable machines for maintenance
27.6 The system shall allow admins to configure machine settings (point rates, bonus multipliers)
27.7 The system shall allow admins to manage machine deployment locations
27.8 The system shall allow admins to view machine maintenance history and schedules

27.9 The system shall allow admins to view machine usage statistics (deposits/day, peak hours)

28. System Monitoring Alerts (Administrator)

28.1 The system shall provide admins with real-time system status dashboard
28.2.1 The system shall send admins capacity alerts (80%, 90%, 100% full)
28.2.2 The system shall send admins machine malfunction alerts
28.3 The system shall notify admins of connectivity issues with offline machines
28.4 The system shall display system error logs and warnings to admins
28.5 The system shall allow admins to track overall system performance metrics
28.6 The system shall allow admins to monitor database connectivity and health
28.7 The system shall allow admins to view active user sessions and concurrent usage

29. User Management (Administrator)

29.1 The system shall allow admins to view registered user accounts
29.2 The system shall allow admins to verify student registrations manually if needed
29.3 The system shall allow admins to deactivate or suspend user accounts for policy violations
29.4 The system shall allow admins to reset user passwords upon request
29.5 The system shall allow admins to view user transaction histories
29.6 The system shall allow admins to handle user disputes and support tickets

Non-Functional Requirements

"The following non-functional requirements define the measurable quality attributes and constraints of the system."

NFR-1 :Performance

1.1 The system shall process bottle/can identification and validation within **2 seconds** of insertion.

1.2 The system shall generate receipt/voucher within **3 seconds** after the user completes their recycling session.

1.3 The system shall support processing at least **50 items per hour** during peak usage times.

NFR-2 : Reliability

2.1 The system shall maintain **99% uptime** during operational hours (excluding scheduled maintenance).

2.2 The system shall perform automatic data backup every **24 hours** to prevent data loss.

2.3 In case of hardware failure, the system shall display an error message within **5 seconds** and notify the administrator.

NFR-3 : Usability

3.1 90% of first-time users shall be able to successfully complete a recycling transaction without assistance within **5 minutes**.

3.2 The user interface shall support multi-language options (minimum 2 languages: **English and Arabic**).

3.3 All on-screen instructions shall be readable from a distance of **1.5 meters** with font size no less than **18pt**.

NFR-4 : Scalability

4.1 The system architecture shall support expansion to at least **10 RVM units** connected to a centralized database without performance degradation.

4.2 The database shall be capable of storing transaction records for up to **100,000 transactions** without affecting query response time (maximum 2 seconds).

4.3 The system shall support adding new bottle/can types (up to **20 different types**) without requiring system redesign.

NFR-5 : Security

5.1 User transaction data shall be encrypted using **AES-256 encryption** standard during transmission and storage.

5.2 The system shall automatically log out admin users after **5 minutes** of inactivity.

5.3 The system shall maintain an audit log of all transactions for a minimum of **12 months** with timestamp and user identification.

5. Initial Identified Subsystems (SOLID-Compliant Design)

> *"Each subsystem is designed following SOLID principles:* ***S=Single Responsibility, O=Open/Closed, L=Liskov Substitution, I=Interface Segregation, D=Dependency Inversion****"*

Subsystem Name	Classes	Subsystem Function	Subsystem Interface

1. Authentication User Management	UserAccount, AuthenticationHandler, SessionManager, PasswordManager	1.1 Manage user registration (registration responsibility only).
		
		1.2 Authenticate existing users (authentication responsibility only).
		
		1.3 Handle user sessions (session management only).
		
		1.4 Manage password security and recovery (password handling only).
			IUserAuth:
			
			`void registerUser(UserData) boolean loginUser(Credentials)`
			
			ISessionManager:
			
			`Session createSession(UserID) boolean validateSession(SessionID)`
			
			IPasswordManager:
			
			`void resetPassword(UserID) void`

2. QR Code Management	QRCodeGenerator, QRCodeValidator, QRScanner	2.1 Generate unique QR codes for each user (QR generation only).	**IQRCodeService:**

Subsystem Name	Classes	Subsystem Function	Subsystem Interface

		2.2 Validate QR codes at RVM machine (validation responsibility).
		
		2.3 Scan QR codes using camera module (scanning only).
			`String generateQRCode(UserID) boolean validateQRCode(String)`
			
			IQRScanner:
			
			`String scanQRCode()`

3. Material Classification Validation	MaterialClassifier, AIModel, ItemValidator, WasteSorter	3.1 Classify deposited materials as plastic or non-plastic using AI (classification only).
		
		3.2 Validate items before acceptance based on material type (validation only).
		
		3.3 Sort and divert non-plastic waste to separate compartment (sorting only).
		
		3.4 Extensible for new material types without modifying existing code.
			IMaterialClassifier:
			
			`MaterialType classifyMaterial(Item) boolean isPlastic(Item)`
			
			⚠ IItemValidator:
			
			`ValidationResult validate(Item)`
			
			(extensible for new validation rules)
			
			IWasteSorter:
			
			`void sortToWaste(Item) void sortToRecyclable(Item)`

Subsystem Name	Classes	Subsystem Function	Subsystem Interface

4. Liquid Detection Rejection	LiquidDetector, TiltSensor, RejectionHandler	4.1 Detect liquid content in bottles using sensors (detection only).
		
		4.2 Reject bottles containing liquid automatically (rejection handling).
		
		4.3 Display clear instructions to users to empty bottles (user feedback).
		
		4.4 Log all rejected items with rejection reasons (logging only).
			ILiquidDetector:
			
			`boolean hasLiquid(Item)`
			`float getLiquidLevel(Item)`
			
			IRejectionHandler:
			
			`void rejectItem(Item, String reason)`
			`void logRejection(RejectionData)`

5. Weight Measurement System	WeightSensor, SensorCalibrator, MeasurementValidator	5.1 Measure weight of deposited plastic (weight measurement only).
		
		5.2 Calibrate sensors automatically (calibration responsibility).
		
		5.3 Validate measurement accuracy (validation only).
			IWeightSensor:
			
			`float measureWeight()`
			`void calibrate()`
			
			IMeasurementValidator:
			
			`boolean validateMeasurement(float)`

6. Points Calculation Bonus Engine	PointsCalculator, IBonusStrategy, FirstDepositBonus,	6.1 Calculate base points from weight (calculation responsibility only).	**IPointsCalculator:** `int calculatePoints(float weight)`

Subsystem Name	Classes	Subsystem Function	Subsystem Interface
	BulkDepositBonus, WeeklyStreakBonus	🔧 6.2 Apply bonus multipliers using strategy pattern (extensible for new bonus types, strategies are substitutable). 6.3 Add new bonus types without modifying existing calculation logic.	🔧 **IBonusStrategy:** `int applyBonus(int basePoints, Context)` *(implements: FirstDepositBonus, BulkDepositBonus, WeeklyStreakBonus - substitutable)*

7. Transaction Processing Recording	TransactionHandler, TransactionRecorder, TransactionValidator, DatabaseUpdater, **GuestTransactionProcessor**	7.1 Create and record transaction details (transaction creation).
		
		7.2 Validate transaction data (validation only).
		
		7.3 Update central database in real-time (database update task).
		
		7.4 Process guest deposits without authentication (guest transaction handling).
			ITransactionCreator:
			
			`Transaction createTransaction(Data)`
			
			ITransactionRecorder:
			
			`void recordTransaction(Transaction)`
			
			IGuestTransactionProcessor:
			
			`Transaction processGuestDeposit(float weight)`

8. Points Balance Account Management	BalanceManager, ExpirationHandler, AccountTracker	8.1 Update user and college point balances (balance management only).	**IBalanceManager:**
			
			`void updateBalance(UserID, int)`
			`int getAvailablePoints(UserID)`

Subsystem Name	Classes	Subsystem Function	Subsystem Interface
		8.2 Track total, redeemed, and available points (tracking responsibility). 8.3 Expire points after 6 months of inactivity (expiration handling).	**IPointsExpiration:** `void expirePoints(UserID)`

9. Leaderboard Ranking System	LeaderboardManager, RankingEngine, StudentAggregator, CollegeAggregator, ITimeFilter, DailyFilter, WeeklyFilter, MonthlyFilter, AllTimeFilter, BadgeSystem	9.1 Maintain student rankings within colleges (ranking responsibility).
		
		🔧 9.2 Aggregate college-wide rankings (aggregation task only).
		
		9.3 Generate university-wide college rankings (campus-level task).
		
		🔧 9.4 Filter leaderboards by time period using strategy pattern (extensible and substitutable filters).
		
		9.5 Award badges to top contributors (badge assignment only).
		
		Note:
			IRankingEngine:
			
			`void updateRankings(UserID)`
			
			ILeaderboardQuery:
			
			`List getStudentLeaderboard(CollegeID)`
			`List getCollegeLeaderboard()`
			
			🔧 ITimeFilter:
			
			`List filterByPeriod(LeaderboardData)`
			
			(implements: DailyFilter, WeeklyFilter, MonthlyFilter, AllTimeFilter - substitutable)
			
			IBadgeAssigner:
			
			`void assignBadges(UserID)`

Subsystem Name	Classes	Subsystem Function	Subsystem Interface
		Guest contributions do not appear in leaderboards but contribute to campus totals.	

10. Redemption Cafeteria Integration	RedemptionHandler, ICafeteriaService, CafeteriaAPIAdapter, ConfirmationGenerator, PointsValidator, PointsDeductor, RewardManager	🔧 10.1 Interface with cafeteria systems depends on ICafeteriaService (depends on abstractions, not concrete API).
		
		10.2 Validate user's available points for redemption (validation only).
		
		10.3 Process point deduction atomically (transaction responsibility).
		
		10.4 Generate confirmation codes for cafeteria (code generation task).
		
		10.5 Record redemption transaction history (recording task only).
		
		10.6 Allow admins to create, update, delete rewards (reward CRUD operations).
		
		Note:
			🔧 ICafeteriaService:
			`boolean requestRedemption(RedemptionData)`
			(implements: CafeteriaAPIAdapter)
			IPointsValidator:
			`boolean validatePoints(UserID, int)`
			IPointsDeductor:
			`void deductPoints(UserID, int)`
			IConfirmationGenerator:
			`String generateConfirmation()`
			IRedemptionRecorder:
			`void recordRedemption(Transaction)`

Subsystem Name	Classes	Subsystem Function	Subsystem Interface
		Only registered users can redeem points.	**IRewardManager:** `void manageRewards(RewardData)`

11. Guest Mode Management	**GuestSessionHandler, GuestTransactionProcessor, GuestAnalytics**	11.1 Display prominent "Guest" button on RVM interface (UI responsibility).
		
		11.2 Handle anonymous guest sessions (guest management only).
		
		11.3 Process guest deposits without QR code/authentication (guest transaction handling).
		
		11.4 Record guest transactions with timestamp and machine location (recording only).
		
		11.5 Display thank you message showing environmental impact (feedback generation).
		
		11.6 Aggregate guest contributions separately (analytics tracking).
			IGuestSessionManager:
			`GuestSession createGuestSession()`
			`void endGuestSession(SessionID)`
			
			IGuestTransactionProcessor:
			`Transaction processGuestDeposit(float weight)`
			`String generateThankYouMessage(float weight)`
			
			IGuestAnalytics:
			`GuestData aggregateGuestData()`
			`float getGuestContribution()`

Subsystem Name	Classes	Subsystem Function	Subsystem Interface

12. Data Synchronization Offline Mode	StorageProvider, LocalStorageHandler, CloudStorageProvider, SyncManager, DatabaseUpdater, ConflictResolver	12.1 Continue RVM operation when internet unavailable (offline operation).
		
		12.2 Store transactions locally or in cloud (storage abstraction with substitutability).
		
		12.3 Synchronize offline data with server automatically (sync-only responsibility).
		
		12.4 Implement conflict resolution mechanisms (conflict handling only).
		
		12.5 Update central database when online (database update task).
			IStorageProvider:
			`void store(Transaction)`
			`List retrieve()`
			(implements: LocalStorageHandler, CloudStorageProvider - substitutable)
			ISyncManager:
			`SyncResult syncOfflineTransactions()`
			`boolean isOnline()`
			IConflictResolver:
			`void resolveConflict(Transaction)`
			IDatabaseUpdater:
			`void updateDatabase(Transaction)`

13. Admin Analytics Reporting	ReportGenerator, AnalyticsEngine, ExportFormatter, EmailScheduler, ChartBuilder, DataAggregator	13.1 Generate system performance reports (transaction volumes, user engagement metrics) (report generation responsibility only).	**IReportGenerator:**
			`Report generatePerformanceReport(DateRange)`
			`Report generateRankingReport(CollegeID,`

Subsystem Name	Classes	Subsystem Function	Subsystem Interface

		13.2 Generate student ranking reports by college and time period (ranking analytics responsibility only).
		
		13.3 Generate college competition standings reports (competition analytics responsibility only).
		
		13.4 Generate environmental impact reports (waste recycled, CO2 savings, sustainability metrics) (environmental analytics responsibility only).
		
		13.5 Generate guest usage analytics and contribution reports (guest analytics responsibility only).
		
		13.6 Generate rejection analytics for quality monitoring (rejection tracking responsibility only).
		
		13.7 Export reports in multiple formats without modifying existing code (export extensibility).
		
		13.8 Schedule automated report generation using abstracted email service (scheduling abstraction).
			`Period)`
			`Report generateEnvironmentalReport(DateRange)`
			
			IAnalyticsEngine:
			
			`Analytics aggregateTransactionData(DateRange)`
			`Analytics calculateEngagementMetrics(UserID)`
			
			IExportFormatter:
			
			`File exportToPDF(Report)`
			`File exportToExcel(Report)`
			`File exportToCSV(Report)`
			
			IEmailScheduler:
			
			`void scheduleReport(ReportType, Schedule)`
			`void sendPeriodicEmail(Report, Recipients)`

Subsystem Name	Classes	Subsystem Function	Subsystem Interface

14. Admin Reward Management	RewardManager, MachineController, SystemMonitor, UserAdministrator, AlertManager, ConfigurationManager	14.1 Manage cafeteria reward catalog (create, update, delete rewards) (reward CRUD responsibility only).
		
		14.2 Configure reward availability and point costs (pricing configuration responsibility only).
		
		14.3 Monitor real-time status of all RVM machines (online/offline) (monitoring responsibility only).
		
		14.4 Control machine operations remotely (enable, disable, configure settings) (machine control responsibility only).
		
		14.5 View machine locations on interactive campus map (location display responsibility only).
		
		14.6 Track machine capacity, health metrics, and analytics (analytics responsibility only).
		
		14.7 Display real-time system health dashboard (system monitoring responsibility only).
			IRewardManager:
			`void createReward(RewardData)`
			`void updateReward(RewardID, RewardData)`
			`void deleteReward(RewardID)`
			`List getActiveRewards()`
			`RewardStats getRedemptionStats(RewardID)`
			
			IMachineController:
			`MachineStatus getMachineStatus(MachineID)`
			`void toggleMachine(MachineID, boolean)`
			`void updateMachineConfig(MachineID, Config)`
			`List getAllMachines()`
			
			ISystemMonitor:
			`Health getSystemHealth()`
			`List getActiveAlerts()`
			`Status getSyncStatus()`
			
			IUserAdministrator:
			`User getUserProfile(UserID)`
			`void suspendUser(UserID, String reason)`

Subsystem Name	Classes	Subsystem Function	Subsystem Interface
		14.8 Send capacity and maintenance alerts to administrators (alert management responsibility only). 14.9 Monitor synchronization status and database health (sync monitoring responsibility only). 14.10 View and manage user accounts (view, verify, suspend) (user administration responsibility only). 14.11 Handle password reset requests from users (password management responsibility only). 14.12 Interface with cafeteria through abstracted service interfaces (dependency abstraction).	`void resetUserPassword(UserID)` `List getUserHistory(UserID)`

6. Traceability Matrix

> *"This traceability matrix links each functional requirement (from Section 4) to the subsystem(s) responsible for its implementation."*

Req. #	Requirement Description	Auth User	QR Code	Material	Liquid	Weight	Points	Balance	Leaderboard	Redemption	Guest	Sync	Analytics	Administrator
1	User Registration Authentication Management	✓												
2	QR Code Generation Management	✓	✓											
3	QR Code Scanning Validation at RVM	✓	✓											
4	Material Classification Validation			✓										

Req. #	Requirement Description	Auth User	QR Code	Material	Liquid	Weight	Points	Balance	Leaderboard	Redemption	Guest	Sync	Analytics	Administrator
5	Liquid Detection Bottle Validation				✓									
6	Weight Measurement Accuracy					✓								
7	Points Calculation System					✓	✓							
8	Bonus Points Incentive System						✓							

Req. #	Requirement Description	Auth User	QR Code	Material	Liquid	Weight	Points	Balance	Leaderboard	Redemption	Guest	Sync	Analytics	Administrator
9	User Points Balance Management							✓						
10	Transaction Recording History											✓		
11	Real-Time Feedback Display				✓	✓	✓	✓						
12	Student Leaderboard System								✓					

Req. #	Requirement Description	Auth User	QR Code	Material	Liquid	Weight	Points	Balance	Leaderboard	Redemption	Guest	Sync	Analytics	Administrator
13	College Competition System								✓					
14	Time-Based Leaderboard Filtering								✓					
15	Badge Achievement System								✓					
16	Cafeteria Integration System									✓				

Req. #	Requirement Description	Auth User	QR Code	Material	Liquid	Weight	Points	Balance	Leaderboard	Redemption	Guest	Sync	Analytics	Administrator
17	Points Redemption Processing							✓		✓				
18	Redemption History Management									✓				
19	Offline Mode Operation											✓		
20	Automatic Data Synchronization											✓		

Req. #	Requirement Description	Auth User	QR Code	Material	Liquid	Weight	Points	Balance	Leaderboard	Redemption	Guest	Sync	Analytics	Administrator
21	Conflict Resolution System							✓				✓		
22	Guest Mode System										✓			
23	Multi-Machine Management System											✓		✓
24	Capacity Monitoring Alerting				✓									✓

Req. #	Requirement Description	Auth User	QR Code	Material	Liquid	Weight	Points	Balance	Leaderboard	Redemption	Guest	Sync	Analytics	Administrator
25	Points Expiration Policy	✓						✓						
26	Analytics Generation										✓		✓	
27	Rewards Management									✓				✓
28	Machine Management													✓
29	System Monitoring Alerts											✓		✓

Req. #	Requirement Description	Auth User	QR Code	Material	Liquid	Weight	Points	Balance	Leaderboard	Redemption	Guest	Sync	Analytics	Administrator
30	User Management	✓												✓

7. Actors & Use Cases

PRIMARY ACTORS:

Actor 1: Guest User

Actor 2: Registered User

Actor 3: Administrator

EXTERNAL ACTORS:

Actor 4: RVM System (Hardware)

Actor 5: Cafeteria System

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

7. Manage User Account

7.1 Profile management

7.2 view history

7.3 update info

8. Manage Inventory

8.1 Track bottle/can storage

8.2 monitor capacity

8.3 alerts

9. Manage Items

9.1 Configure item types

9.2 pricing

9.3 validation rules

10. Admin Analytics Reporting

10.1 Generate reports

10.2 view statistics

10.3 export data

Use Case Descriptions

1. Manage Authentication

Name	Manage Authentication
Type	Primary

Actors	Administrator & Registered User
Description	1. User/Administrator can register a new account. 2. User/Administrator can login with email and password. 3. User/Administrator can update their profile information. 4. System validates user credentials during login. 5. System enforces password security requirements. 6. System locks account after failed login attempts. 7. System sends confirmation email after registration. 8. User must re-authenticate when changing password. 9. System prevents duplicate email registration. 10. System maintains data integrity through validation checks.

Cross Ref	R1, R2, R3, R4, R5
Preconditions	None (first use case in the system)
Postconditions	Manage Profile, Manage Orders, Manage Notifications
Alternative Scenario	- Invalid credentials entered. - Account locked due to multiple failed

	attempts. - Email already exists during registration. - Password does not meet security requirements. - Required fields not filled.

List of Requirements

1. R1: Valid email address
2. R2: Password (minimum 8 characters with special character)
3. R3: User name
4. R4: Phone number
5. R5: Internet access

2. Manage Transaction

Name	Manage Transaction
Type	Primary

Actors	Registered User, Guest User, RVM System
Description	1. User initiates a recycling transaction at the RVM. 2. System validates QR code or guest mode selection. 3. System accepts valid recyclable bottles/cans. 4. System measures weight, classifies materials, and detects liquid. 5. Points are calculated based on deposit

	weight. 6. Transaction data recorded with user ID, timestamp, and location. 7. System provides receipt and feedback after completion. 8. Guest transactions recorded separately (no points).

Cross Ref	R6, R7, R8, R9, R10
Preconditions	User authenticated or guest mode selected; RVM is online or in offline mode.
Postconditions	Manage Points, Manage Rewards, Manage Notifications
Alternative Scenario	- Non-recyclable item detected. - Item rejected due to liquid detection. - System fails to print receipt. - Transaction canceled before completion.

List of Requirements

6. R6: Valid QR code or guest mode
7. R7: Functional sensors (weight, AI classifier, liquid detector)
8. R8: Database access
9. R9: Transaction recording
10. R10: Receipt printing or digital record

3. Manage QR Code

Name	Manage QR Code
Type	Primary

Actors	Registered User, Cafeteria System
Description	1. System generates a unique QR code after successful point redemption. 2. QR code includes encrypted transaction data (user ID, timestamp). 3. Cafeteria system scans and validates the QR code. 4. System verifies QR authenticity and expiration. 5. Used QR codes are marked invalid to prevent reuse.

Cross Ref	R11, R12, R13, R14, R15
Preconditions	Successful reward redemption.
Postconditions	Manage Rewards, Manage Notifications
Alternative Scenario	- Invalid or tampered QR code. - QR code expired. - Cafeteria scanner error. - Network connection failure during validation.

List of Requirements

11. R11: QR code encryption
12. R12: QR expiration (24 hours)
13. R13: Unique identifier
14. R14: Scanning and validation time ≤ 2 seconds
15. R15: Cross-system verification

4. Manage Rewards

Name	Manage Rewards
Type	Primary

Actors	Administrator & Registered User
Description	1. Administrator manages reward catalog (create, update, delete items). 2. System issues rewards upon point redemption. 3. Users can view and track issued rewards. 4. System monitors reward status (active, used, expired). 5. System notifies users of issued or expiring rewards.

Cross Ref	R16, R17, R18, R19, R20
Preconditions	User has sufficient point balance; administrator has access privileges.
Postconditions	Manage Points, Manage Notifications
Alternative Scenario	- Reward catalog unavailable. - Invalid reward configuration. - Reward expired before redemption. - System fails to issue reward notification.

List of Requirements

16. R16: Reward catalog management
17. R17: Reward value linked to points
18. R18: Unique reward ID
19. R19: Reward expiration and tracking
20. R20: Notification of issued rewards

5. Manage Points

Name	Manage Points
Type	Primary

Actors	Registered User, Administrator
Description	1. System calculates points based on deposited item weight and bonuses. 2. User's total, redeemed, and available balances are tracked in real time. 3. System enables point redemption for rewards. 4. System enforces expiration policies for inactive accounts. 5. Administrators can modify or review user point policies.

Cross Ref	R21, R22, R23, R24, R25
Preconditions	Successful recycling transaction or administrative setup.
Postconditions	Manage Rewards, Manage Analytics
Alternative Scenario	- Insufficient points for redemption. - Duplicate point addition detected. - Points expired or policy misconfigured. - Connection error during point update.

List of Requirements

21. R21: Point calculation (10 points/kg)
22. R22: Real-time balance updates
23. R23: Redemption validation
24. R24: Expiration notifications
25. R25: Administrative point management

8. Detailed Use Case Interaction Scenarios

USE CASE 1: MANAGE AUTHENTICATION

1.1 Login

Actor Intentions	System Responsibility
1. User navigates to RVM system interface and selects "Login" option.	
2. User enters their registered email address and password into the login form.	3. System validates email format and checks credentials against database.
	4. System verifies password matches encrypted stored password for the email.

Actor Intentions	System Responsibility
	5. System checks account status (not locked, active).
	6. System creates authenticated session and generates session token.
7. User is redirected to their dashboard.	8. System displays personalized dashboard with user name, current point balance, and recent transactions.
	9. System logs successful login with timestamp and device information.

1.2 Register

Actor Intentions	System Responsibility
1. New user navigates to registration page and clicks "Create Account".	
2. User fills in registration form with name, email, phone number, and password.	3. System validates email format and checks for existing account with same email.
	4. System verifies password meets security requirements (minimum 8 characters, contains special character).
	5. System validates phone number format.
3. User clicks "Register" button to submit form.	6. System creates new user account with encrypted password.
	7. System generates unique user ID and initializes point balance to zero.

Actor Intentions	System Responsibility
	8. System sends confirmation email with account verification link.
	9. System displays success message "Account created successfully! Please check your email to verify."
4. User checks email and clicks verification link.	10. System activates account and marks email as verified.

1.3 Update Profile

Actor Intentions	System Responsibility
1. Registered user logs in and navigates to "Profile Settings".	2. System retrieves current user profile information from database.

Actor Intentions	System Responsibility
	3. System displays profile form pre-filled with existing data (name, email, phone).
4. User modifies name from "Ahmed Ali" to "Ahmed Mohamed" and updates phone number.	5. System validates new phone number format.
6. User clicks "Save Changes" button.	7. System checks if email was changed and validates uniqueness if modified.
	8. System updates user record in database with new information.
	9. System displays confirmation message "Profile updated successfully".
	10. System immediately reflects changes across all user interfaces.

USE CASE 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

Actor Intentions	System Responsibility
1. User approaches RVM and initiates recycling session (scan QR code or press "Start" for guest mode).	
	2. System authenticates user (if registered) or creates temporary guest session.
	3. System displays welcome message with user name (or "Guest User") and current point balance (if applicable).

Actor Intentions	System Responsibility
	4. System activates item acceptance mechanism and displays instruction: "Insert recyclable items one at a time".
5. User inserts first recyclable item (bottle or can) into RVM slot.	
	6. System captures item and triggers validation process (calls Validate Items sub-function).
	7. System displays validation result: "Item Accepted: Plastic Bottle - 15g - 0.15 points" (or rejection message if invalid).
	8. System updates running total on screen: "Items: 1 | Total Weight: 15g | Points Earned: 0.15".

Actor Intentions	System Responsibility
9. User continues inserting items or presses "Finish Transaction" button.	
	10. System repeats validation for each additional item (steps 6-8) until user completes transaction.
11. User presses "Finish Transaction" to complete the recycling session.	
	12. System calculates final totals: total items, total weight, base points, bonus points (peak hour/streak), and final points.
	13. System updates user's point balance in database (if registered user) or prepares guest transaction summary.

Actor Intentions	System Responsibility
	14. System generates transaction receipt (calls Generate Receipt sub-function).
	15. System displays completion message: "Transaction Complete! You earned 5.2 points. Thank you for recycling!"
	16. System saves transaction record to database with timestamp, user ID (if applicable), items processed, and points awarded.
	17. System sends transaction notification to user (if registered - calls Notification Service).
	18. System resets RVM to idle state and displays: "Ready for next user".

2.2 Validate Items

Actor Intentions	System Responsibility
1. User inserts item into RVM acceptance slot.	
	2. System's optical sensor captures item image and initiates material classification.
	3. System analyzes material type using ML-based Material Classification Service.
	4. System identifies item as: Plastic (PET), Aluminum, or Glass.
	5. If material type is unrecognizable, system rejects item with message: "Item not recognized. Please insert recyclable bottles or cans only".

Actor Intentions	System Responsibility
	6. System checks item condition using sensors (detects cracks, contamination, or damage).
	7. If item is damaged or contaminated, system rejects with message: "Item condition unacceptable. Please ensure items are clean and undamaged".
	8. System weighs item using integrated weight sensor (measures in grams).
	9. System validates weight is within acceptable range (5g - 500g for bottles/cans).
	10. If weight is out of range, system rejects with message: "Item weight invalid. Accepted range: 5-500 grams".

Actor Intentions	System Responsibility
	11. System retrieves material factor from configuration: Plastic=1.0, Aluminum=1.5, Glass=0.8.
	12. System calculates point value: Points = (Weight in kg) × 10 × Material Factor.
	13. System logs validation result with timestamp, item details, and point value.
	14. System displays validation result on screen: "✓ Accepted: Aluminum Can - 25g - 0.375 points".
	15. System physically accepts item (moves to storage compartment) or ejects rejected item.

Actor Intentions	System Responsibility
	16. System updates transaction item list and cumulative totals.
	17. System returns validation status to Process Recycling function.
18. User sees validation result and proceeds with next item or completes transaction.	

2.3 Generate Receipt

Actor Intentions	System Responsibility
1. User completes transaction and system prepares receipt generation.	

Actor Intentions	System Responsibility
	2. System retrieves transaction data: user info, timestamp, item list, weights, points.
	3. System generates unique transaction ID using format: TXN-{Date}-{MachineID}-{SequenceNo} (e.g., TXN-20251108-RVM01-00523).
	4. System compiles receipt header: "Smart RVM - Recycling Receipt", Date/Time, Transaction ID, Machine Location.
	5. System lists all accepted items with details: Item Type | Quantity | Weight (g) | Points.
	6. System calculates and displays breakdown: Base Points, Bonus Points (peak hour +20%, streak +10%), Total Points Earned.

Actor Intentions	System Responsibility
	7. System includes user information: User Name (or "Guest User"), Student ID (if registered), New Point Balance.
	8. System adds footer: "Thank you for recycling! Keep up the great work!", Environmental Impact Message, Support Contact.
	9. System formats receipt for physical printing (thermal printer format, 58mm width).
	10. System sends receipt data to thermal printer hardware.
	11. If printer is available and operational, system prints physical receipt.

Actor Intentions	System Responsibility
	12. If printer error occurs (paper jam, out of paper), system displays: "Receipt could not be printed. Check your email/app for digital receipt".
	13. System logs printer error and sends alert to maintenance team.
	14. For registered users, system saves digital receipt to user's transaction history in database.
	15. System generates QR code for receipt (encodes transaction ID for verification).
	16. System sends email notification with digital receipt attached (PDF format) to registered users.

Actor Intentions	System Responsibility
17. User collects printed receipt from RVM output slot (or accesses digital version via app/email).	
	18. System confirms receipt generation completion and returns to Process Recycling function.

USE CASE 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

Actor Intentions	System Responsibility
1. Student completes registration through mobile app or web interface.	
	2. System validates student registration data (university email, student ID, college affiliation).

	3. System generates unique QR code using QRCodeGenerator with encrypted user ID.
	4. System links QR code permanently to user account in database.
	5. System stores QR code data with metadata (generation timestamp, user ID, status: active).
	6. System displays generated QR code in mobile application profile section.
	7. System sends confirmation notification "Your QR code is ready for use at any RVM machine".

3.2 Scan QR Code

Actor Intentions	System Responsibility
1. Student approaches RVM machine and selects "I'm a Registered Student" button.	
	2. RVM displays camera viewfinder with "Scan your QR code" message.
3. Student positions phone with QR code in front of RVM camera module.	4. QRScanner captures QR code image using integrated camera.
	5. System decodes QR code string using QRCodeValidator.

	6. System validates QR code format and encryption signature.
	7. System queries database to verify QR code exists and is active (not expired/compromised).
	8. System retrieves associated user account (name, student ID, college, points balance).
	9. System completes validation within 2 seconds as per NFR-1.1.
	10. System displays welcome message "Welcome back, [Student Name]!" with current balance.
	11. System creates authenticated session for the transaction.

3.3 Validate QR Code

Actor Intentions	System Responsibility
Scenario A: Expired QR Code	
1. Student scans QR code that has expired due to account inactivity (15 days).	2. System decodes QR code and checks associated account status.
	3. System detects account is inactive (last transaction > 15 days ago).
	4. System rejects QR code with error message "Your account is inactive. Please reactivate through mobile app".

	5. System displays RVM error screen with reactivation instructions.
Scenario B: Compromised QR Code	
1. Student attempts to scan old QR code after regenerating new one.	2. System decodes QR code and queries database.
	3. System finds QR code marked as "compromised" status.
	4. System rejects QR code with error "This QR code is no longer valid. Please use your new QR code".
	5. System logs invalid scan attempt for security monitoring.
Scenario C: Invalid/Fake QR Code	
1. Someone attempts to scan non-RVM QR code or tampered code.	2. QRScanner captures image and attempts to decode.
	3. System validates encryption signature - validation fails.
	4. System rejects with error "Invalid QR code. Please use your RVM student QR code".
	5. System increments invalid scan counter for abuse prevention.

USE CASE 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

Actor Intentions	System Responsibility
1. System detects critical error (machine full, sensor malfunction, or network failure).	
	2. System classifies alert severity level (low, medium, high, critical) based on error type.
	3. System retrieves administrator contact information from database.

Actor Intentions	System Responsibility
	4. System determines appropriate notification channels based on severity (critical = email + SMS, high = email).
	5. System composes alert message including: error type, machine ID/location, timestamp, and severity level.
	6. System sends alert via email to administrator account (admin@rvm.com).
	7. For critical alerts, system also sends SMS to administrator's registered mobile number.
	8. System logs alert in notification history with delivery status and timestamp.

Actor Intentions	System Responsibility
9. Administrator receives notification on email/SMS and reviews the alert.	
10. Administrator investigates and resolves the issue (e.g., empties machine, repairs sensor).	
	11. System detects issue resolution and sends follow-up notification "Issue resolved: Machine at Location X is now operational".
	12. System updates alert status from "active" to "resolved" in database.

6.2 Send Transaction Notification

Actor Intentions	System Responsibility
1. Registered user completes recycling transaction at RVM machine.	
	2. System retrieves transaction details: user ID, items deposited (5 plastic bottles, 3 aluminum cans), points earned (80 points), bonus applied (+10 bonus points).
	3. System calculates new total point balance (previous: 450 points, new: 540 points).
	4. System checks user notification preferences from profile settings.
	5. System verifies user has opted-in for transaction notifications (preference = enabled).

Actor Intentions	System Responsibility
	6. System composes notification message including: transaction summary, items count, points earned, bonus points, new balance, and transaction timestamp.
	7. System sends in-app notification to user's mobile application.
	8. System sends email notification to user's registered email (yasseen@rvm.com) with detailed transaction receipt.
	9. System logs notification delivery with timestamp (within 5 seconds of transaction completion).
10. User receives notification on mobile app with sound alert and badge counter update.	

Actor Intentions	System Responsibility
11. User opens notification to view transaction details and current point balance.	
	12. System marks notification as "read" and updates notification history.
	13. If notification delivery fails, system logs error and schedules retry after 1 minute (maximum 3 retry attempts).

6.3 Send Reward Notification

Actor Intentions	System Responsibility
1. User successfully redeems reward from cafeteria (Coffee - 100 points).	

Actor Intentions	System Responsibility
	2. System deducts points from user balance (previous: 540 points, new: 440 points).
	3. System generates unique QR code for reward redemption with 24-hour expiration.
	4. System creates redemption record with: reward ID, user ID, QR code, expiration timestamp, and redemption status.
	5. System composes reward notification message including: reward name, points deducted, QR code image, redemption instructions, and expiration time.
	6. System sends in-app notification with embedded QR code.

Actor Intentions	System Responsibility
	7. System sends email notification to user (ahmed@rvm.com) with QR code attachment and redemption details.
	8. System displays success message "Reward redeemed successfully! Show QR code at cafeteria within 24 hours".
9. User receives notification and saves QR code to device.	
	10. System schedules reminder notification 2 hours before QR code expiration.
	11. When expiration approaches (22 hours elapsed), system sends reminder notification "Your reward QR code expires in 2 hours!"

Actor Intentions	System Responsibility
12. User presents QR code at cafeteria counter.	
13. Cafeteria staff scans QR code to validate redemption.	
	14. System validates QR code (checks expiration, redemption status).
	15. System marks reward as "redeemed" and updates redemption timestamp.
	16. System sends final confirmation notification "Your Coffee reward has been redeemed successfully! Remaining balance: 440 points".
	17. System logs complete redemption history for audit trail.

USE CASE 5: MANAGE POINTS

Description: This use case handles all point-related operations in the RVM system. It manages point accumulation based on deposited items and weight, handles point redemption for rewards, applies point expiration policies, and maintains accurate point balances for users. The system ensures fair point distribution, prevents point abuse, and provides transparency in point transactions.

5.1 Accumulate Points

Actor Intentions	System Responsibility
1. Registered user completes recycling transaction at RVM machine (deposits 5 plastic bottles).	
	2. System retrieves item validation results from MaterialClassifier (all items accepted).
	3. System retrieves total weight from weight sensor (0.5 kg total).

Actor Intentions	System Responsibility
	4. System calculates base points using formula: Points = Weight (kg) × 10 × Material Factor.
	5. System applies material factor: Plastic = 1.0, Aluminum = 1.5, Glass = 0.8.
	6. System calculates base points: 0.5 kg × 10 × 1.0 = 50 base points.
	7. System checks for applicable bonus strategies (time-based, streak, milestone).
	8. System detects peak hour bonus (Monday-Friday, 8AM-2PM): +20% bonus = 10 extra points.

Actor Intentions	System Responsibility
	9. System checks user's streak: 5 consecutive days recycling = +10% bonus = 5 extra points.
	10. System calculates total points: 50 (base) + 10 (peak hour) + 5 (streak) = 65 points.
	11. System retrieves user's current point balance from database (previous balance: 450 points).
	12. System adds earned points to balance: 450 + 65 = 515 new balance.
	13. System creates transaction record with: user ID, transaction timestamp, items deposited, weight, base points, bonus points, total points earned.

Actor Intentions	System Responsibility
	14. System updates user balance in database and commits transaction.
	15. System updates user's recycling streak (now 6 consecutive days).
	16. System checks if user reached milestone (500 points milestone crossed).
	17. System triggers milestone notification for reaching 500 points.
18. User receives notification showing: "You earned 65 points! (50 base + 15 bonus). New balance: 515 points".	
	19. System updates leaderboard rankings with new point total.

Actor Intentions	System Responsibility
	20. System logs complete point accumulation transaction for audit trail.

5.2 Redeem Points

Actor Intentions	System Responsibility
1. User opens mobile app and navigates to "Rewards" page.	
	2. System retrieves user's current point balance from database (515 points).
	3. System fetches available cafeteria rewards from database with point costs.

Actor Intentions	System Responsibility
	4. System displays reward catalog: Coffee (100 pts), Sandwich (200 pts), Meal (500 pts), showing which rewards are affordable.
5. User browses rewards and selects "Meal - 500 points".	
	6. System validates user has sufficient points (515 >= 500: Valid).
	7. System displays confirmation dialog: "Redeem Meal for 500 points? Remaining balance will be 15 points".
8. User confirms redemption by clicking "Redeem Now".	
	9. System starts database transaction to ensure atomicity.

Actor Intentions	System Responsibility
	10. System deducts 500 points from user balance (new balance: 15 points).
	11. System generates unique QR code for reward redemption (QR-20251108-USER001-MEAL).
	12. System sets QR code expiration time (24 hours from now).
	13. System creates redemption record with: redemption ID, user ID, reward ID, points deducted, QR code, generation timestamp, expiration timestamp, status = "pending".
	14. System updates user's point balance in database.
	15. System commits database transaction.

Actor Intentions	System Responsibility
	16. System sends reward notification with QR code to user's app and email.
	17. System displays success message with QR code: "Meal redeemed successfully! Show QR code at cafeteria within 24 hours".
18. User saves QR code and navigates to cafeteria.	
19. User presents QR code at cafeteria counter.	
20. Cafeteria staff scans QR code using cafeteria system.	
	21. System validates QR code (checks: exists, not expired, not already redeemed).

Actor Intentions	System Responsibility
	22. System marks redemption status as "redeemed" and records redemption timestamp.
	23. System sends confirmation notification to user: "Your Meal has been redeemed successfully! Remaining balance: 15 points".
	24. System logs complete redemption transaction for audit and analytics.

5.3 Apply Point Expiration

Actor Intentions	System Responsibility
	1. System scheduler triggers nightly point expiration job at 2:00 AM.

Actor Intentions	System Responsibility
	2. System retrieves expiration policy from configuration: Points expire after 12 months of inactivity.
	3. System queries database for all user accounts with point balances > 0.
	4. System identifies user "Ahmed" with 300 points and last activity date: 12 months ago (2024-11-08).
	5. System calculates days since last activity: 365 days (exceeds 12-month policy).
	6. System retrieves user's transaction history to determine which points are expiring.

Actor Intentions	System Responsibility
	7. System applies FIFO (First In, First Out) expiration: Oldest earned points expire first.
	8. System identifies 300 points earned on 2024-11-08 are eligible for expiration.
	9. System checks if user has any pending rewards or transactions (none found).
	10. System starts database transaction for point expiration.
	11. System deducts 300 expired points from user's balance (new balance: 0 points).

Actor Intentions	System Responsibility
	12. System creates expiration record with: user ID, expired points amount (300), expiration date, reason ("12 months inactivity"), original earn date.
	13. System updates user's point balance in database.
	14. System commits database transaction.
	15. System sends expiration notification to user via email and in-app: "300 points expired due to 12 months of inactivity. Start recycling to earn new points!"
	16. System logs expiration event for audit trail with detailed information.

Actor Intentions	System Responsibility
	17. System generates expiration report for administrators showing: total users affected, total points expired, reasons for expiration.
	18. System checks for users with points expiring soon (within 30 days).
	19. System identifies user "Sara" with 150 points expiring in 15 days.
	20. System sends warning notification to Sara: "Your 150 points will expire in 15 days! Use them or deposit items to extend expiration".
21. User Sara receives warning notification and decides to deposit items.	
22. User Sara completes a transaction (deposits 3 bottles).	

Actor Intentions	System Responsibility
	23. System updates Sara's last activity date to current date, resetting expiration countdown.
	24. System logs activity and cancels pending expiration for Sara's points.

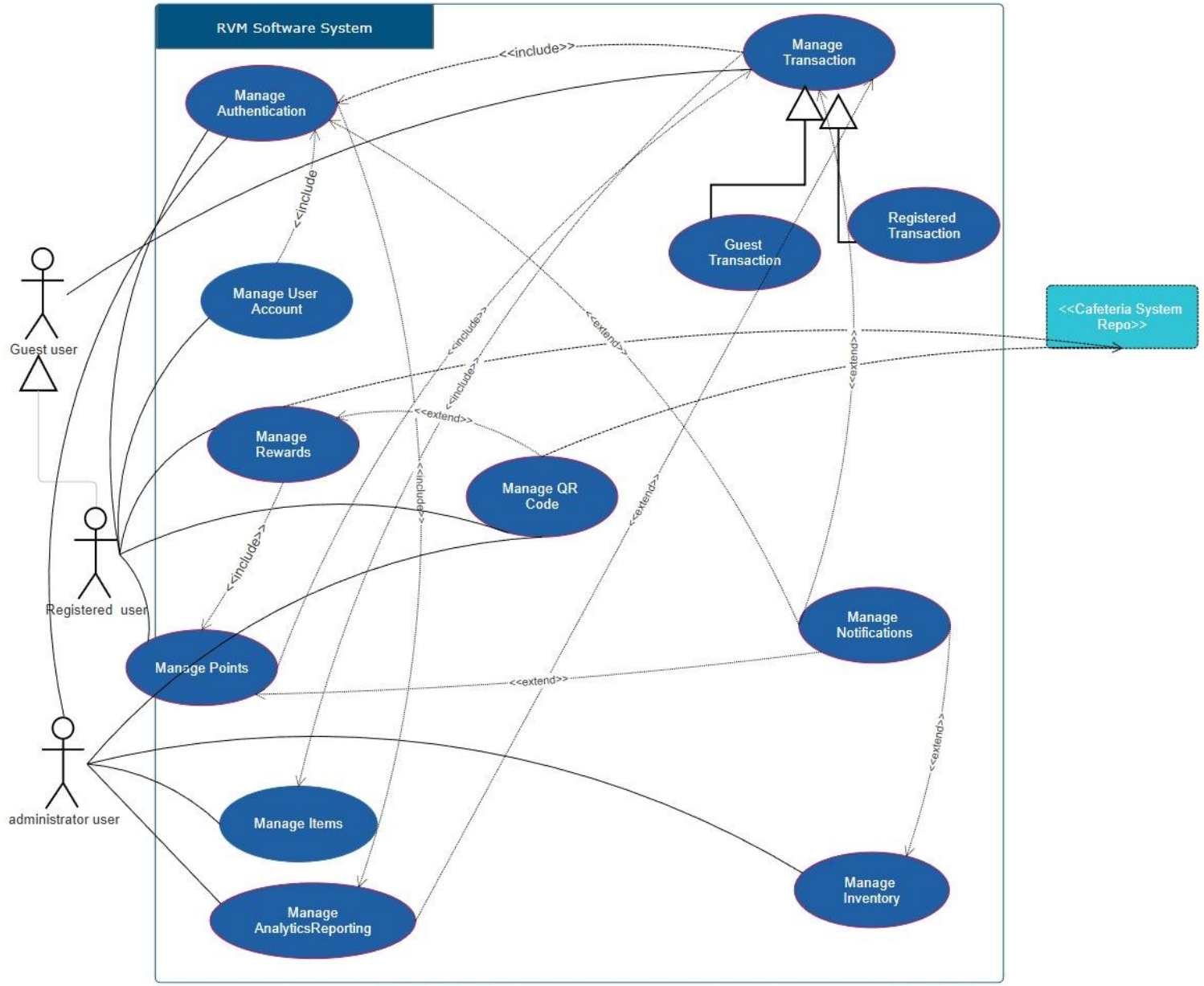
9. UseCase Relationships

Relation Type	Primary Use Case	Secondary Use Case
<<Include>>	Manage Transaction	Manage Authentication

Relation Type	Primary Use Case	Secondary Use Case
<<Include>>	Manage User Account	Manage Authentication
<<Include>>	Manage Rewards	Manage Points
<<Include>>	Manage Transaction	Manage Items
<<Extend>>	Manage Notifications	Manage Authentication
<<Extend>>	Manage Notifications	Manage Transaction
<<Extend>>	Manage QR Code	Manage Rewards

Relation Type	Primary Use Case	Secondary Use Case
<<Extend>>	Manage Notifications	Manage Inventory
Generalization	Manage Transaction	Guest Transaction
Generalization	Manage Transaction	Registered Transaction

10. Use Case Diagram


RVM Software System
Manage Transaction
Manage Authentication
Guest Transaction
Registered Transaction
Manage User Account
<<Cafeteria System Repo>>
Guest user
Manage Rewards
Manage QR Code
Registered user
Manage Notifications
Manage Points
administrator user
Manage Items
Manage Inventory
Manage AnalyticsReporting
<<include>>
<<extend>>

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.com**Password: **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.com**Password: **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		

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_05	TR-LOG-03: Validate that user cannot login with non-existent email	Enter **unregistered email**	1. Email not in database2. Login page accessible	1. Enter User Email2. Enter Password3. Click "Login" button	Email: **not1 @valid.com** Password: **Pass@ 123**	Error message "No account found with this email"	Login denied		
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		

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_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 for login failures	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		
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		

Figure 1: *Use Case Diagram showing the RVM Software System with actors and their interactions*

Diagram Key:

- **Solid lines:** Direct associations between actors and use cases
- **Dashed lines with «include»:** Required relationship - the base use case always includes the included use case
- **Dashed lines with «extend»:** Optional relationship - the extending use case adds behavior under certain conditions
- **Dashed lines (inheritance):** Generalization showing Guest Transaction and Registered Transaction are specialized types
- **System boundary box:** Shows the scope of the RVM Software System
- **External actors:** Systems outside the main software system that interact with it

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_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: **Ahm ed Ali** Email: **ali.ahmed.n ew@rvm.c om** Phone: 010123 456789 Password: **S ecure@12 5**	Success message "Account created successfully!"	Account created, confirmatio n email sent		
TC_Auth_1 2	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: **new user@rvm.c om** Password: **Tes t@123**	Confirmatio n email received with verification link	Email delivered within 60 seconds		
TC_Auth_1 3	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: **yass een@rvm.co m** Password: **Pass @123**	Error "Email already registered. Please login or use different email"	Registration blocked		
TC_Auth_1 4	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: **user @test.com** (exists as User@T est.com) Password: **T est@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)
				email 4. Submit form					
TC_Auth_1 5	TR-REG-03: Validate that password meets minimum security requiremen ts	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: **P ass12** (only 6 chars)	Error "Password must be at least 8 characters long"	Registration blocked		
TC_Auth_1 6	TR-REG-03: Validate that password meets minimum security requiremen ts	Password la cks special character	1. User on registration page	1. Fill required fields 2. Enter 8-char password without special 3. Submit	Password: **P assword12**	Error "Password must contain at least one special character (@, #, $, etc.)"	Form not submitted		
TC_Auth_1 7	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: **{Em pty}** Email: **m.com** Password: **T est@123**	"Name is required" error message, name field highlighted	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_1 8	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: **{Em pty}** Email: **all: user@t est.com** Password: **{Empty}**	Multiple errors: "Name is required", "Password is required"	All empty fields highlighted		
TC_Auth_1 9	TR-REG-05: Register Validate that system sends confirmatio n 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: **verif y@rvm.co m**	Email received with subject "Verify Your RVM Account", contains verification link	Email delivered within 1 minute		
TC_Auth_2 0	TR-REG-05: Email Validate that system sends	contains ve rification 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	

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_2 1	TR-UPD-01: Validate that registered user can update personal information	Update na me 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: **Ahm ed Ali** New Name: **Ahm ed Mohamed**	Success message "Profile updated successfully ", name changed	Profile shows new name		
TC_Auth_2 2	TR-UPD-01: Validate that registered user can update personal information	Update pho ne number	1. Logged in 2. Current phone: 0101234567 8	1. Open profile editor 2. Change phone 3. Save changes	Old Phone: **010 12345678** New Phone: **010 98765432**	Phone updated, confirmatio n displayed	New phone saved		
TC_Auth_2 3	TR-UPD-02: Validate that user cannot change email to already taken email	Change email to **already taken email**	1. Logged as user1@t est.com 2. user2@t est.com exi sts	1. Open settings 2. Try to change to user2@t est.com 3. Save	Current: **us er1@test.c om** New: **us er2@test.c om** (ex ists)	Error "This email is already in use. Please choose another"	Email blocked		
TC_Auth_2 4	TR-UPD-02: Validate that user cannot change email to already existing email	Change email to same email	1. Current: **ah med@test. com**	1. Edit email field 2. Re-enter same email 3. Save	Current & New: **ahme d@test.co m**	Update succeeds (no change), no error	Email unchanged		

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_2 5	TR-UPD-03: Validate that profile changes are saved and reflected immediatel y	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 4. Check dashboard	Old: **Ali** New: **Yasse en**	Dashboard shows "Welcome, Yasseen" immediatel y	New name visible everywhere		
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		

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_2 8	TR-UPD-04: Validate that user must re-authenticat e when changing password	Enter incorr ect 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		
TC_Auth_2 9	TR-UPD-05: Guest tries Validate to access that guest profile users settings cannot access profile update functionalit y	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_3 0	TR-UPD-05: Validate that guest users cannot access profile update functionalit y	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 authenticat ed features		

1.2 Register

1.3 Update Profile

Total Test Cases for MANAGE AUTHENTICATION: 30 test cases covering login, registration, and profile updates

Note: *These test cases validate the complete authentication flow including registration, login security, profile updates, and error handling mechanisms.*

TestCase Diagram: USE CASE 2 - MANAGE TRANSACTION

2.1 Process Recycling

CASE ID	TEST SCENARIO	TEST CASE DESCRIPTION	PRE-CONDITION	TEST STEPS	TEST DATA	EXPECTED RESULT	POST CONDITION	ACTUAL RESULT	STATUS (PASS/
xn_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		
xn_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		
xn_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		

CASE ID	TEST SCENARIO	TEST CASE DESCRIPTION	PRE-CONDITION	TEST STEPS	TEST DATA	EXPECTED RESULT	POST CONDITION	ACTUAL RESULT	STATUS (PASS/
xn_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		
xn_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		
xn_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		

CASE ID	TEST SCENARIO	TEST CASE DESCRIPTION	PRE-CONDITION	TEST STEPS	TEST DATA	EXPECTED RESULT	POST CONDITION	ACTUAL RESULT	STATUS (PASS/
xn_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		
xn_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		
xn_09	TR-PROC-05: Validate that system saves transaction data for registered users	Transaction saved in **history for registered user**	1. User: Ahmed 2. Transaction completed	1. Complete transaction 2. Check database 3. Verify saved data	User: **Ahmed** Items: **5** Points: **6.6**	Record saved with: timestamp, user ID, item list, points, transaction ID	Transaction in history, accessible via app		

CASE ID	TEST SCENARIO	TEST CASE DESCRIPTION	PRE-CONDITION	TEST STEPS	TEST DATA	EXPECTED RESULT	POST CONDITION	ACTUAL RESULT	STATUS (PASS/
xn_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		

CASE ID	TEST SCENARIO	TEST CASE DESCRIPTION	PRE-CONDITION	TEST STEPS	TEST DATA	EXPECTED RESULT	POST CONDITION	ACTUAL RESULT	STATUS (PASS/FAIL)
xn_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		
xn_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		
xn_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		
xn_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		
xn_15	TR-VAL-03: Validate that RVM hardware correctly identifies	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		

CASE ID	TEST SCENARIO	TEST CASE DESCRIPTION	PRE-CONDITION	TEST STEPS	TEST DATA	EXPECTED RESULT	POST CONDITION	ACTUAL RESULT	STATUS (PASS/FAIL)
	item type (plastic, aluminum, glass)								
xn_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		
xn_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		
xn_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		

CASE ID	TEST SCENARIO	TEST CASE DESCRIPTION	PRE-CONDITION	TEST STEPS	TEST DATA	EXPECTED RESULT	POST CONDITION	ACTUAL RESULT	STATUS (PASS/FAIL)
xn_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		
xn_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		

2.2 Validate Items

EST CASE ID	TEST SCENARIO	TEST CASE DESCRIPTION	PRE-CONDITION	TEST STEPS	TEST DATA	EXPECTED RESULT	POST CONDITION	ACTUAL RESULT	STATUS (PASS/FAIL)
C_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		
C_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		
C_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		
C_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		

EST CASE ID	TEST SCENARIO	TEST CASE DESCRIPTION	PRE-CONDITION	TEST STEPS	TEST DATA	EXPECTED RESULT	POST CONDITION	ACTUAL RESULT	STATUS (PASS/FAIL)
C_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		
C_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		
C_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		

EST CASE ID	TEST SCENARIO	TEST CASE DESCRIPTION	PRE-CONDITION	TEST STEPS	TEST DATA	EXPECTED RESULT	POST CONDITION	ACTUAL RESULT	STATUS (PASS/FAIL)
C_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		
C_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		
C_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		

2.3 Generate Receipt

CASE ID	TEST SCENARIO	TEST CASE DESCRIPTION	PRE-CONDITION	TEST STEPS	TEST DATA	EXPECTED RESULT	POST CONDITION	ACTUAL RESULT	STATUS (PASS/FAIL)
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		
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		
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		
QR_04	TR-GEN-02: Validate that QR code contains encrypted transaction data	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	Signature: **HMAC-SHA256 Hash**	Signature verification passes, confirms	QR tamper-proof		

CASE ID	TEST SCENARIO	TEST CASE DESCRIPTION	PRE-CONDITION	TEST STEPS	TEST DATA	EXPECTED RESULT	POST CONDITION	ACTUAL RESULT	STATUS (PASS/FAIL)
	(user ID, amount, timestamp)			4. Check authenticity		authenticity and integrity			
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		
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		
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		

CASE ID	TEST SCENARIO	TEST CASE DESCRIPTION	PRE-CONDITION	TEST STEPS	TEST DATA	EXPECTED RESULT	POST CONDITION	ACTUAL RESULT	STATUS (PASS/FAIL)
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		
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		
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		

Total Test Cases for MANAGE TRANSACTION: 30 test cases covering recycling process, item validation, and receipt generation

Note: *These test cases validate the complete transaction flow including session management, sequential item processing, material classification, condition verification, point calculation with bonuses, receipt generation, and error handling for hardware failures.*

TestCase Diagram: USE CASE 3 - MANAGE QR CODE

3.1 Generate QR Code

3.2 Scan QR Code

CASE ID	TEST SCENARIO	TEST CASE DESCRIPTION	PRE-CONDITION	TEST STEPS	TEST DATA	EXPECTED RESULT	POST CONDITION	ACTUAL RESULT	STATUS (PASS/FAIL)
R_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		
R_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		
R_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		

CASE ID	TEST SCENARIO	TEST CASE DESCRIPTION	PRE-CONDITION	TEST STEPS	TEST DATA	EXPECTED RESULT	POST CONDITION	ACTUAL RESULT	STATUS (PASS/FAIL)
R_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		
R_15	TR-SCN-03: Validate that scanning process completes within 2 seconds	Scan time **≤ 2 seconds**	1. QR code ready 2. Scanner active 3. Timer ready	1. Start timer 2. Scan QR code 3. Wait for result 4. Stop timer	QR: **Valid code**	Scan + decode + display completed within 2 seconds	Response time ≤ 2 sec		
R_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		
R_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		

CASE ID	TEST SCENARIO	TEST CASE DESCRIPTION	PRE-CONDITION	TEST STEPS	TEST DATA	EXPECTED RESULT	POST CONDITION	ACTUAL RESULT	STATUS (PASS/FAIL)
R_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		
R_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		
R_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		

T CASE ID	TEST SCENARIO	TEST CASE DESCRIPTION	PRE-CONDITION	TEST STEPS	TEST DATA	EXPECTED RESULT	POST CONDITION	ACTUAL RESULT	STATUS (PASS/FAIL)
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		
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		
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		
QR_24	TR-VQR-02: Validate that system checks QR code has not	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	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		

T CASE ID	TEST SCENARIO	TEST CASE DESCRIPTION	PRE-CONDITION	TEST STEPS	TEST DATA	EXPECTED RESULT	POST CONDITION	ACTUAL RESULT	STATUS (PASS/FAIL)
	been previously used			previous use 4. Reject					
QR_25	TR-VQR-03: Validate that system rejects expired QR codes (older than 24 hours)	Accept QR within **24-hour window**	1. Generated: Nov 8 14:30 2. Scanned: Nov 9 10:00 (19.5 hrs later)	1. Scan QR code 2. Check timestamp 3. Calculate age 4. Verify valid	Age: **19.5 hours**	QR code accepted, within valid 24-hour period	Validation successful		
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		
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		

T CASE ID	TEST SCENARIO	TEST CASE DESCRIPTION	PRE-CONDITION	TEST STEPS	TEST DATA	EXPECTED RESULT	POST CONDITION	ACTUAL RESULT	STATUS (PASS/FAIL)
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		
QR_29	TR-VQR-05: Validate that invalid or tampered QR codes display appropriate error message	Reject **unrecognized QR code**	1. QR from external source 2. Not in database	1. Scan unknown QR 2. Query database 3. No record found 4. Reject	Redemption ID: **Not Found**	Error: "QR code not recognized. Please contact support."	Fake QR rejected		
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		

3.3 Validate QR Code

Total Test Cases for MANAGE QR CODE: 30 test cases covering QR generation, scanning, and validation

Note: *These test cases validate the complete QR code lifecycle including unique code generation with encryption, multi-format delivery (PNG, PDF), cafeteria scanning from mobile/print, authentication through digital signatures, expiration enforcement (24 hours), usage tracking, and security measures against tampering, duplication, and fraud.*

TestCase Diagram: 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_4 1	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_4 2	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_4 3	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		

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_4 4	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		
TC_Not_4 5	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_4 6	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		

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_4 7	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_4 8	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_4 9	TR-SYS-05: Validate that alert resolution updates are sent after issue is fixed	Send **resolution notification** after fix	1. Active alert exists 2. Issue being resolved	1. Admin fixes issue 2. Mark alert as resolved 3. Check notification sent	Alert ID: **ALT-001** Status: **Resolved**	Resolution notification "Issue resolved: Machine operational"	All stakeholders notified of resolution		
TC_Not_5 0	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	Alert: **Machine Full** New Status: **Resolved**	Database updated, status changed, timestamp recorded	Alert history shows resolution		

TEST CASE ID	TEST SCENARIO	TEST CASE DESCRIPTION	PRE-CONDITION	TEST STEPS	TEST DATA	EXPECTED RESULT	POST CONDITION	ACTUAL RESULT	STATUS (PASS/FAIL)
				3. Verify database update					

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_5 1	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_5 2	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_5 3	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		

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_5 4	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		
TC_Not_5 5	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_5 6	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		

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_5 7	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_5 8	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		
TC_Not_5 9	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		

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_6 0	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		

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_6 1	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_6 2	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_6 3	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_6 4	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_6 5	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_6 6	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_6 7	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		

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_6 8	TR-RWD-04: Validate that users receive notification when reaching reward milestones	Suggest **available rewards** at milestone	1. Milestone reached 2. Rewards available	1. Trigger milestone 2. Check notification content 3. Verify suggestions	Points: **500** Available: **Coffee, Sandwich**	Notification suggests "You can now redeem: Coffee, Sandwich"	User encouraged to redeem		
TC_Not_6 9	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_7 0	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		

6.2 Send Transaction Notification

6.3 Send Reward Notification

Total Test Cases for MANAGE NOTIFICATIONS: 30 test cases covering all notification scenarios

Note: *These test cases validate the complete notification system including system alerts, transaction notifications, reward notifications, and all delivery mechanisms (email, SMS, in-app).*

TestCase Diagram: USE CASE 5 - MANAGE 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	Calculate points for **aluminum cans**	1. User logged in 2. Items validated as aluminum	1. Deposit 8 aluminum cans 2. System weighs (0.4kg)	Weight: **0.4kg** Material: **Aluminum** Factor: **1.5**	Base points = 0.4 × 10 × 1.5 = 60 points	60 points calculated with aluminum bonus		

TEST CASE ID	TEST SCENARIO	TEST CASE DESCRIPTION	PRE-CONDITION	TEST STEPS	TEST DATA	EXPECTED RESULT	POST CONDITION	ACTUAL RESULT	STATUS (PASS/FAIL)
	based on weight and material type			3. Apply aluminum factor					
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, milestone) are applied correctly	Apply **recycling streak bonus**	1. User has 5-day streak 2. Base points earned: 50	1. Check user's streak 2. Apply +10% streak bonus 3. Calculate total	Base: **50 pts** Streak: **5 days** Bonus: **+10%**	Total = 50 + (50 × 0.10) = 55 points	Streak bonus added to points		
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		

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_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		

5.1 Accumulate Points

5.2 Redeem Points

TEST CASE ID	TEST SCENARIO	TEST CASE DESCRIPTION	PRE-CONDITION	TEST STEPS	TEST DATA	EXPECTED RESULT	POST CONDITION	ACTUAL RESULT	STA
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		
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		

TEST CASE ID	TEST SCENARIO	TEST CASE DESCRIPTION	PRE-CONDITION	TEST STEPS	TEST DATA	EXPECTED RESULT	POST CONDITION	ACTUAL RESULT	STA
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		
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		

TEST CASE ID	TEST SCENARIO	TEST CASE DESCRIPTION	PRE-CONDITION	TEST STEPS	TEST DATA	EXPECTED RESULT	POST CONDITION	ACTUAL RESULT	STA
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 3. Reject validation	Generated: **Nov 7, 13:00**	Error "QR code expired. Please contact support"	Redemption denied		

TEST CASE ID	TEST SCENARIO	TEST CASE DESCRIPTION	PRE-CONDITION	TEST STEPS	TEST DATA	EXPECTED RESULT	POST CONDITION	ACTUAL RESULT	STA
	validated at cafeteria		2. Expiration: 24 hours		Current: **Nov 8, 14:00**				

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		

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_25	TR-EXP-03: Validate that expiration notification is sent to affected users	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		
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		

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_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 expiration countdown	Reset expiration after **new transaction**	1. Points expiring in 15 days 2. User deposits items	1. User completes transaction 2. Update last activity 3. Reset countdown	User: **Sara** Activity: **Nov 8, 2025**	Last activity updated, expiration reset to 12 months	Points safe for 12 more 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		

5.3 Apply Point Expiration

Total Test Cases for MANAGE POINTS: 30 test cases covering point accumulation, redemption, and expiration scenarios

Note: *These test cases validate the complete points management system including point calculation, bonus application, balance updates, milestone detection, reward redemption, QR code generation/validation, and point expiration policies.*

7. Initial GUI Sketches

Mobile Application:

- **Home Screen:** Displays large QR code for scanning, current points balance with animated counter, quick stats (deposits this month, rank), and college badge.
- **QR Scan Page (RVM Display):** Shows instructions for depositing plastic, real-time weight display during deposit, and transaction progress with animation.
- **Leaderboard Page:** Three tabs (My College, All Colleges, My Rank), filterable by time period (daily/weekly/monthly/all-time), displays top contributors with avatars and point totals.
- **Transaction History:** Scrollable list of all deposits showing date, time, weight, points earned, and machine location with search functionality.

- **Redemption Page:** Browse available cafeteria items with point costs, confirmation dialog before redemption, displays confirmation code after successful redemption.
- **Profile Page:** Displays student information, college badge, total points earned/redeemed, environmental impact stats (kg recycled, CO_2 saved equivalent).

RVM Machine Interface:

- **Welcome Screen:** Two prominent options: "I'm a Registered Student" (QR code icon) and "Guest Mode" (universal recycling icon). Shows machine status, capacity indicator, and motivational message.

Registered Student Flow:

- **QR Scan Screen:** Camera viewfinder with "Scan your QR code" message.
- **Item Validation:** "Analyzing item..." message while MaterialClassifier and LiquidDetector work.
- **Rejection Feedback:** If rejected: Clear message ("Please empty bottle" or "Non-plastic item detected - diverted to waste") with visual icons.
- **Weight Measurement:** Real-time weight display with animated progress bar (if item accepted).
- **Transaction Confirmation:** Shows weight, points earned (with bonuses), new balance, current rank.

Guest Mode Flow:

- **Welcome Message:** "Thank you for contributing to campus sustainability!"
- **Instructions:** Simple visual guide showing how to deposit plastic.
- **Item Validation:** Same validation as registered users (material classification + liquid detection).

- **Rejection Feedback:** Same clear rejection messages as registered flow.
- **Weight Display:** Real-time weight measurement with animated progress bar (if item accepted).
- **Thank You Screen:** Shows weight contributed, equivalent environmental impact (e.g., "5 water bottles recycled = 0.15kg CO2 saved"), and total campus contribution today. Option to "Learn More About Our Program" with QR code to download student app.

Mobile Application & RVM Machine Interface Prototype:

❖ View Figma Prototype - RVM Machine App & Interface

Admin Dashboard:

- **Overview Page:** Shows total active machines, machine capacity status with alerts for 80% full machines, system status indicators, and real-time alerts. Guest contribution widget showing daily guest deposits vs registered user deposits.
- **Analytics Page:** Displays student participation breakdown by college/class/rank, points distribution over time, statistics, and college competition standings. Guest analytics section with guest usage trends, peak guest usage times, and guest-to-registered user ratio. Rejection analytics showing rejection rates, common rejection reasons, over time.
- **Machine Control Page:** Map view of all RVM locations across university, individual machine status (online/offline/maintenance), fill levels (recyclable + waste compartments), remote enable/disable controls.

- **Reports Page:** Generate and download detailed reports for total transactions, student rankings, college performance, total weight recycled, and environmental impact metrics. Guest contribution report showing guest usage patterns. Rejection analysis showing invalid item type for targeted education campaigns.
- **Rewards Management Page:** Create, edit, and delete cafeteria reward items with point costs, view redemption statistics, and manage reward availability.
- **Quality Monitoring:** Dashboard showing validation system performance, rejection reasons breakdown, contamination prevention metrics, and user education opportunities.

❖ View Figma Prototype - Admin Dashboard

8. Impact Analysis: Enhanced Features

Benefits of Guest Mode:

- **Increased Participation:** Potentially 3-5x more recyclables collected by including entire campus community.
- **Inclusive Sustainability:** Faculty, staff, and visitors actively participate in environmental goals.
- **Marketing & Awareness:** Guest mode serves as a gateway to full program registration.

- **Better Metrics:** True campus-wide environmental impact measurement.
- **Social Proof:** Higher total contribution numbers motivate registered users.
- **Operational Efficiency:** Better capacity utilization of RVM machines.

Benefits of Intelligent Validation System:

- **Recycling Quality:** Prevents contamination by separating non-plastic waste automatically.
- **Machine Protection:** Liquid detection prevents machine damage and maintenance issues.
- **User Education:** Clear rejection feedback teaches proper recycling behavior.
- **Operational Efficiency:** Reduces manual sorting and post-processing costs.
- **Data-Driven Improvement:** Rejection analytics identify patterns for targeted education campaigns.
- **Environmental Impact:** Higher quality recyclables increase actual recycling rates vs. landfill diversion.
- **Scalability:** Automated validation allows deployment without constant human supervision.

SOLID Compliance:

- **SRP:** MaterialClassifier, LiquidDetector, ItemValidator, WasteSorter, GuestSessionHandler, GuestTransactionProcessor each have single, focused responsibilities.

- **OCP:** New validation rules (AluminumDetector), report types (RejectionAnalyticsReport, GuestAnalyticsReport), and bonus strategies can be added through extension, not by modifying existing code.
- **LSP:** Different storage providers, time filters, bonus strategies, and validators can be substituted seamlessly.
- **ISP:** Separate focused interfaces (IMaterialClassifier, ILiquidDetector, ItemValidator, IGuestSessionManager, IGuestTransactionProcessor) instead of one monolithic interface.
- **DIP:** High-level modules depend on abstractions (IItemValidator, IStorageProvider, ISensor) not concrete implementations.

Technical Considerations:

- **Database Schema:** Transaction table includes user_type (enum: 'registered', 'guest'), validation_status ('accepted', 'rejected_liquid', 'rejected_material'), material_type ('plastic', 'glass', 'aluminum', 'other').
- **Hardware Integration:** Camera module for visual classification, weight sensors for liquid detection, dual compartment system (recyclable + waste).
- **Abuse Prevention:** Rate limiting on guest deposits (max 5 per hour per machine); rejection logging to identify malicious behavior.
- **Offline Support:** Guest transactions and validation results cached locally during offline mode.

- **Analytics Separation:** Clear distinction between registered user metrics and guest metrics for accurate gamification.

9. Competitive Analysis in Egypt

*In Egypt, several startups and initiatives have emerged to address plastic waste management and recycling through smart technologies. The most relevant competitors to our Reverse Vending Machine (RVM) project are **Recyclobekia**, **Dawarha**, **ZeroPrime Technologies (Canbank)**, **Bekia**, and traditional campus recycling programs.*

Recyclobekia

Recyclobekia operates smart recycling bins and reverse vending machines in select Cairo locations, primarily targeting shopping malls and corporate offices. Their machines provide basic monetary rewards (cash or vouchers) but lack gamification elements and campus-specific features. Recyclobekia has limited presence in educational institutions and does not offer college competition frameworks or integration with campus services.

Dawarha

Dawarha is currently the most prominent company in the Egyptian recycling sector. It manufactures AI-powered Reverse Vending Machines capable of identifying and separating recyclables such as plastic bottles and cans. Dawarha has partnered with Nestlé and deployed machines across several Egyptian cities. However, Dawarha's solution is enterprise-focused and designed for large-scale deployment and public environmental awareness in commercial areas, rather than educational institutions. While technologically advanced, Dawarha's hardware is not incorporate student-centric features, college ranking mechanics, or campus service integration.

ZeroPrime Technologies (CanBank)

ZeroPrime Technologies, through its product **CanBank**, offers smart vending-style recycling machines placed in malls and some universities. The system reward users with phone credit and digital vouchers when they deposit bottles or cans. While ZeroPrime has notable local visibility and demonstrates effective public engagement through basic incentivization, its deployment scale remains smaller than Dawarha's. CanBank lacks comprehensive gamification features, student-facing system design, and no cloud-based campus integration nor project offers.

Bekia

Bekia operates differently as a digital platform for waste exchange rather than deploying physical RVMs. Users can exchange recyclables for goods or services through scheduled pickups, targeting households and small businesses. While Bekia successfully connects waste collectors with users, it does not provide automated, real-time recycling solutions or campus-specific functionality. It lacks the immediate gratification and gamified experience it unsuitable for institutional campus reach.

Traditional Campus Recycling Initiatives

Traditional campus recycling initiatives across Egyptian universities (AUC, GUC, Cairo University) rely on manual collection bins and plastic bottle recycling programs. They offer from inconsistent operator tracking mechanisms. Student participation rates due to absence of incentives, and no data analytics capabilities. Student engagement remains low, and environmental impact measurement is virtually non-existent.

> ***"Overall, while existing competitors have demonstrated market viability for recycling technologies in Egypt, none currently integrate secure QR-based student identification, university-level competition frameworks, campus service integration, guest participation, and comprehensive offline capability as our proposed project does. Our system fills this gap by combining secure RVM hardware with a reward-based mobile application, institutional gamification, and inclusive participation — specifically designed to foster sustainability and environmental awareness within Egyptian university communities."***