

## User Stories and Backend Stories for Arifpay Web App UI Screens

---

### 1. Dashboard Overview (Image 1)

**WebApp User Story:** As an admin, I want to see an overview of branches, POS, clerks, and weekly transaction performance, so that I can monitor key operational metrics at a glance.

**Backend Story:**

- Create an API to fetch counts for branches, POS terminals, and clerks.
  - Create an API to aggregate weekly transaction data by day.
  - Create an API to retrieve the most used payment methods with counts and amounts.
  - Implement caching for dashboard data for faster loading.
- 

### 2. Payments Screen (Image 2)

**WebApp User Story:** As a merchant or admin, I want to access different wallet and payment options, So that I can easily process transactions and see recent activities.

**Backend Story:**

- Integrate with Telebirr, CBE Birr, Kacha, and MPESA APIs.
  - Create a service to generate dynamic QR codes.
  - Create CRUD APIs for card, QR, and cash transactions.
  - Create an API to retrieve recent transactions filtered by method.
- 

### 3. Transactions List (Image 3)

**WebApp User Story:** As an admin, I want to view and filter all transaction history, So that I can monitor activity and export data.

**Backend Story:**

- Create an endpoint to fetch transactions with filters (date, method, status, clerk).
- Implement pagination and export functionality (CSV/Excel).
- Secure API with role-based access control.

---

## 4. QR Management (Image 4)

**WebApp User Story:** As a merchant, I want to manage static and dynamic QR codes by store, so that I can assign and reuse them easily.

**Backend Story:**

- Create endpoints for static and dynamic QR code creation and retrieval.
- Assign QR codes to stores with appropriate metadata.
- Store QR codes securely with download URLs.

---

## 5. Inventory Management (Image 5)

**WebApp User Story:** As a store manager, I want to track inventory status, top-selling items, and low-stock items, so that I can manage stock effectively.

**Backend Story:**

- Implement product, category, and unit models.
- Create APIs to track sales, reorder levels, and current stock.
- Create reports for on-demand and low-stock items.

---

## 6. Store Management (Image 6)

**WebApp User Story:** As an admin, I want to add, edit, disable, or delete store branches, So that I can manage store structure efficiently.

**Backend Story:**

- Create CRUD endpoints for branch/store management.
- Attach QR code download functionality to store data.
- Handle soft-delete and reactivation workflows.

---

## 7. Events Management (Image 7)

**WebApp User Story:** As an event organizer, I want to create, edit, and display events, So that I can manage campaigns and public-facing activities.

**Backend Story:**

- Design event model with fields: title, date, location, description, banner.
  - Create a CRUD API for event management.
  - Store event media in a CDN or cloud storage.
- 

**8. Clerk Management (Image 8)**

**WebApp User Story:** As an admin, I want to view, edit, disable, or delete clerk accounts, So that I can manage access to POS and store functions.

**Backend Story:**

- Create a clerk user model with role and location fields.
  - Implement APIs for enabling/disabling and deleting clerks.
  - Secure endpoints based on admin permissions.
- 

**9. Reporting & Analytics (Image 9)**

**WebApp User Story:** As a financial analyst, I want to generate filtered transaction reports so that I can analyze business performance.

**Backend Story:**

- Implement dynamic report generation based on filters.
  - Generate downloadable files in PDF/CSV formats.
  - Schedule and log report generation timestamps.
  - Secure access with role-based access.
- 

Let me know if you'd like these stories formatted into JIRA-compatible tasks or with effort estimates.





## Frontend Task: Generate Dynamic QR Modal Flow this sprint


Description	Acceptance Criteria	Assigned to / Story points
Modal opens with title "Assign Payment to Table QR"	<ul style="list-style-type: none"><li>- Modal is full-width with header text</li><li>- Dropdown for "Business Type" Table</li></ul>	Yoseph / Yashe - 1 story point

	- Amount input shown	
Cashier selects table/order and enters amount	- Table dropdown is searchable - Required validation for amount > 0	Yoseph /Yashe -1 story
Click "Push Payment to QR"	- Calls API to create transaction and returns QR link - QR is displayed inside modal or redirects to confirmation	Yoseph/ Yashe -1 story point
UI shows status: Unpaid / Paid	- Initially status = "Unpaid" - Updates automatically when payment is completed (poll or WebSocket) - Toast: "Table 7 paid 200 ETB"	Yoseph -Yashe - 1 story point

## FRONTEND TASKS + ACCEPTANCE CRITERIA Generate Dynamic QR

Feature	Description	Acceptance Criteria	Needs	Assigned to / Story point
Table UI Page	Display all tables with status	- Page: /tables - Tables shown in responsive grid - Each card shows: Table #,  Paid /  Unpaid badge - Button: "Generate QR" per table	Table creation needed.	Yashe- jojo -2 storypoint
Select & Generate QR	Generate QR from table card	- On "Generate QR", show modal - Modal includes: Amount input + submit - Calls POST /transactions		Yashe -Jojo - 2 story

		and renders QR code - Status initially shows "Unpaid"		
Periodic Polling  (should be real time)	Fetch status updates every 5s	- Use setInterval() to call GET /tables/status every 5 seconds - If status changes: update card, show toast: " Table 7 has paid"		Yashe -Jojo -1 story
WebSoc ket Sync (Optional )	Real-time updates without polling	- On dashboard load, open WebSocket - Listen for payment_completed - On event: update matching table - Show alert: "Table 10 paid 250 ETB"	Backlog	
Visual Notificati on	Toast on payment success	- If status changes: show toast <Table X has paid Y ETB> - Toast auto-dismisses in 3–5 sec - Optional: Play success sound or animate		Yashe-Jojo - 2 story point.
Transacti on History	View transaction s for table	- On card click or "Details" button, open modal/page - Call GET /transactions?table_id=X		Yashe -Jo 1 story point

		<ul style="list-style-type: none"> <li>- List: Amount, Time, Status</li> <li>- Most recent at top</li> </ul>		
Reflect QR Status	Update QR status if payment made	<ul style="list-style-type: none"> <li>- If QR is still open and payment is completed: Update QR status badge to  Paid</li> <li>- Disable QR re-use button based on the table status</li> </ul>		Yashe-jojo-2 story point
Error Handling & Fallbacks	UX when data fails	<ul style="list-style-type: none"> <li>- Show loader on API call</li> <li>- Handle network failures gracefully (e.g. "Try again" option)</li> <li>- Use default "Unpaid" if backend is unreachable</li> </ul>		Yashe-jojo-2 story point



## Verify QR (Universal Scanner)

### Frontend Tasks + Acceptance Criteria

Feature	Description	Acceptance Criteria	Assigned to / Story point
QR Scanner UI	Allow user to scan any QR code using camera or image upload	<ul style="list-style-type: none"><li>- Button opens camera scanner or image upload</li><li>- Accepts both live scan and uploaded .jpg/.png</li><li>- Fallback message if QR is unreadable</li></ul>	Yashe-jo- 1 story point
Display Decoded Info	Show decoded QR content (URL, text, etc.)	<ul style="list-style-type: none"><li>- Decoded text shown in large readable block</li><li>- If content is a URL, auto-link it</li><li>- Show type badge (Text, URL, Contact, etc.)</li></ul>	1 story point
Save Scanned QR	Save scanned QR content for later	<ul style="list-style-type: none"><li>- "Save QR" button appears after successful scan</li><li>- Input field to label the QR (e.g. "WiFi at Office")</li><li>- Save to local storage or user profile</li></ul>	1 story point

View Saved QRs	Show list of previously saved QR scans	<ul style="list-style-type: none"> <li>- Page: /saved-qr</li> <li>- List of saved entries with title, type, timestamp</li> <li>- “Open” or “Copy” buttons for reuse</li> </ul>	1 story point
Category Filter	Allow users to tag/organize saved QRs	<ul style="list-style-type: none"> <li>- Category dropdown: Work, Event, WiFi, Link</li> <li>- Filter/search saved QRs by label or content</li> </ul>	1 story point

## Frontend Tasks – Dynamic QR Creation (QR Management Page)

Feature	Description	Acceptance Criteria	Assigned to.
Tab Switch (Static / Dynamic)	Toggle between Static QR and Dynamic QR views	<ul style="list-style-type: none"> <li>- Tabs update UI state when clicked</li> <li>- Default tab: “Dynamic QR”</li> <li>- Active tab is visually highlighted</li> </ul>	Yashe -Jo 1 story point
QR Card Listing	Display dynamic QR cards per store	<ul style="list-style-type: none"> <li>- Each card shows QR Name, Store</li> <li>- Cards are responsive and clickable (options via menu)</li> </ul>	Yash-Jo 1-story point
Create QR Button	Opens modal to add new Dynamic QR	<ul style="list-style-type: none"> <li>- “+ Create new QR” button opens modal</li> <li>- Modal overlay darkens background</li> <li>- Form has: QR Name, Store Selector, Amount (optional)</li> </ul>	Yash-jo 1 story point
Store Selector	Dropdown to choose a store	<ul style="list-style-type: none"> <li>- Lists existing stores</li> <li>- Has “Create New Store” button</li> <li>- Store is required for QR creation</li> </ul>	

QR Amount (Optional)	Field to optionally bind an amount to the QR	<ul style="list-style-type: none"> <li>- Accepts decimal numbers only</li> <li>- Placeholder shown if empty</li> <li>- Max length or amount validation enforced</li> </ul>	1 story point
Submit QR Form	Submit and generate QR	<ul style="list-style-type: none"> <li>- Clicking "Create new Dynamic QR" triggers API</li> <li>- Form validation before submission</li> <li>- Success: toast "QR Created" + refresh list</li> <li>- Error: shows error message</li> </ul>	1 story point
Refresh QR List	After QR is created, show updated QR list	<ul style="list-style-type: none"> <li>- Newly created QR appears at top</li> <li>- Page reload not required</li> </ul>	1 story point