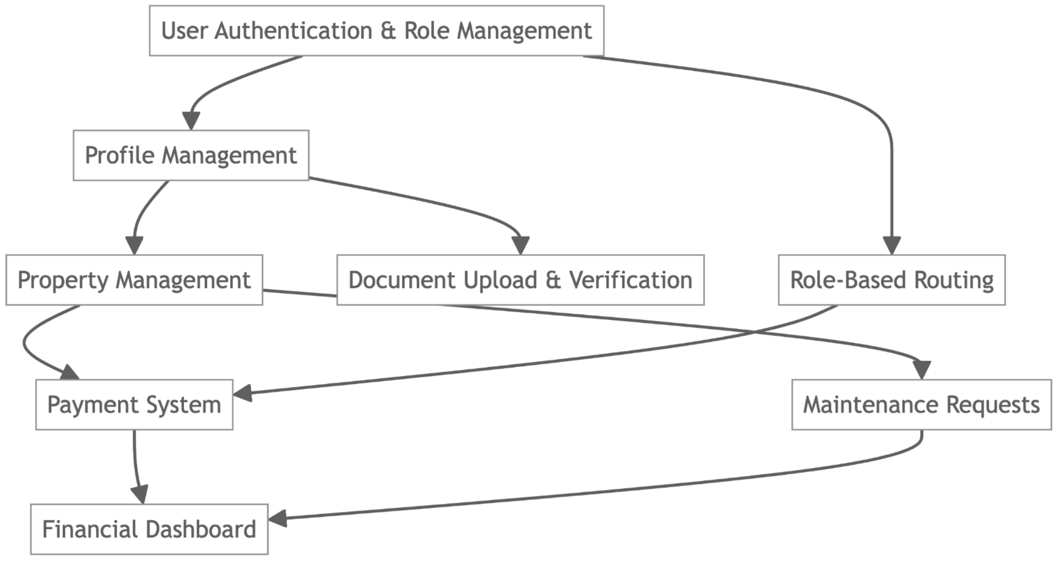
Schedule & Dependencies Analysis

Network Diagram



A. User Authentication & Role Management → B. Profile Management → C. Property Management → D. Payment System → E. Financial Dashboard
- Property Management also enables Maintenance Requests
- Document Upload & Verification is linked to Profile Management
- Role-Based Routing ensures users access the correct features

Dependencies

- User Authentication & Role Management is foundational; all other features depend on users being authenticated and assigned a role.
- Profile Management (tenant/landlord) must be completed before users can manage properties or submit requests.
- Property Management depends on profiles and is required for both payment and maintenance flows.
- Payment System depends on property and user data, as payments are tied to specific properties and users.
- Financial Dashboard aggregates data from the payment system and maintenance requests.
- Document Upload & Verification is linked to profile management for KYC and compliance.

- - Role-Based Routing ensures users only access features relevant to their role.

Critical Path

1. User Authentication & Role Management
2. Profile Management
3. Property Management
4. Payment System
5. Financial Dashboard

Any delay in these steps will delay the overall project delivery.

Risk Areas

- Scope Creep: Mid-sprint scope changes can disrupt planning and velocity.
- Integration Risks: Payment and dashboard features depend on accurate, timely data from earlier modules.
- Resource Bottlenecks: If key team members are unavailable, critical path tasks may be delayed.
- External Dependencies: Reliance on third-party services (e.g., Supabase, payment APIs) introduces risk if those services are unavailable or change unexpectedly.
- Testing & QA: Insufficient testing of integrated flows (e.g., payment + dashboard) could lead to bugs surfacing late.

What Went Wrong & Lessons Learned

What Went Wrong

- Mid-Sprint Scope Changes: New requirements and re-estimation of tasks led to a spike in remaining work and disrupted the planned burndown.
- Reduced Development Days: An extra-long weekend reduced the number of effective working days, impacting velocity.
- Underestimation: Some tasks were underestimated, leading to rework and modifications.

Lessons Learned

- Lock Scope Early: Aim to finalize sprint scope before starting to avoid mid-sprint disruptions.
- Account for Non-Working Days: Adjust sprint planning to account for holidays and known absences.
- Improve Estimation: Break down tasks more granularly and involve the whole team in estimation to improve accuracy.
- Document Dependencies: Make dependencies explicit in planning to avoid integration surprises.
- Continuous Integration: Integrate and test features early and often to catch issues before they block the critical path.