Weekly Project Status Update

Project: Loan Management System (Laravel + MySQL + Eloquent ORM)
Report Date: September 4, 2025
Changes: Email option added

---

1. Summary of Accomplishments

All previously listed tasks have been **completed**, including:

- **Infrastructure & Setup**: Laravel project initialized, configured `.env`, and connected to MySQL.
- **Authentication & Authorization**: Roles for Receptionist and Loan Officer seeded, authentication implemented, role-based redirection applied.
- **Receptionist Forms**: Customer intake workflows built.
- **Data Modeling**: ERD created; tables set up; schemas and model relationships documented.
- **Validation & Sanitization**: Full JS frontend and PHP backend validation and sanitization implemented.
- **Model Mappings & Filtering**: Zip code filtering for Loan Officer territories; clear table-to-model mappings established.
- **Loan Officer Dashboard Enhancements**: Clickable customer summaries with detailed retrieval.
- **Loan Application Module**: Backend logic and frontend components scaffolded.

NEWLY COMPLETED THIS WEEK:

- **Pending Loans Dashboard Enhancements**:
    - Brief retrieval of loan application and customer info on Loan Officer's pending list.
    - "Approve" button implemented to update customer status to Approved.
    - Backend now auto-creates loan accounts for approved customers using data from `loan_application`, `business_type`, `customer`, `guarantors`, and `collaterals` tables.
    - Loan schedule calculation and display functionality added for Loan Officers.
    - Email button included to send loan schedules via email to approved customers.
    - Right-click event added on customer list items to allow deletion of selected customers.

---

2. Current Work In Progress

- **Route Security Enhancements**:

- Securing login and other application routes.
- Implementing robust logout functionality to ensure secure session termination for staff.

---

3. Upcoming Deliverables (Next Week)

- **Complete Security Hardening**:
  - Finish securing all routes and finalizing logout logic.
- **Apply Software Design Best Practices ('Schology's requirement')**:
  - Begin integrating **SOLID design principles** into the codebase to enhance modularity, maintainability, and extensibility. For example, use **Single Responsibility** to ensure classes do only one thing, and **Dependency Inversion** to depend on abstractions not concrete implementations.
- **RESTful API Exploration**:
  - Begin designing how a **RESTful API** could be applied to your system to support potential future integrations or client-server decoupling. RESTful architecture principles such as statelessness, uniform interface, and resource-based endpoints will be considered.
