# HOA Management System: Complete Project Documentation

December 2024

**Student Information**

\* Name: Justin Pike

\* Major: Computer Science

\* Degree: Bachelor of Science - Computer Science

\* Project Advisor: Michael O’Neill

\* Expected Graduation: December 2024

## 1. Statement of Purpose

### Project Overview

The HOA Management System represents a comprehensive solution designed to address the complex challenges faced by homeowners associations in today’s residential communities. This web-based platform aims to modernize and streamline the entire spectrum of HOA operations, from basic data management to complex financial tracking and communication systems, while acknowledging the critical importance of proper system initialization and deployment for successful implementation.

### Problem Statement

Homeowners associations (HOAs) have become an integral part of modern residential communities, affecting millions of homeowners across the United States. Despite their widespread presence, HOAs frequently face criticism from homeowners due to issues ranging from rigid rules and fees to ineffective management and lack of transparency. While some challenges, such as fee structures or community-specific restrictions, are subjective and difficult to standardize, many operational inefficiencies stem from poor administrative practices. These inefficiencies contribute significantly to homeowner frustrations and hinder the ability of HOAs to foster a positive community environment. The key operational challenges facing HOAs can be categorized into four main areas:

1. Data Management Issues

* Inconsistent collection and storage of homeowner information. Many HOAs rely on outdated or piecemeal systems, leading to errors in record-keeping.
* Difficulty maintaining up-to-date contact details. Homeowners frequently move or change contact information, which can lead to communication gaps.
* Lack of centralized record-keeping. Some HOAs still use paper-based systems or multiple, disconnected tools, resulting in inefficiencies.
* Inadequate systems for tracking property ownership changes. Ownership transfers (e.g., sales or inheritance) can be poorly documented, affecting HOA dues and compliance.
* Poor organization of historical data. Important records, such as past meeting minutes or financial statements, are often not easily accessible.

1. Communication Barriers

* Ineffective distribution of critical announcements and updates. Without standardized systems, HOAs may fail to disseminate essential information timely.
* Lack of standardized communication channels. Homeowners often receive inconsistent updates via physical mail, email, or word of mouth.
* Inconsistent delivery and engagement tracking for important information. HOAs typically have no mechanism to confirm receipt or homeowner acknowledgment of updates.
* Limited ability to document and reference communication history. Past communications may not be easily retrieved or tracked for disputes or transparency.
* Overreliance on informal methods of communication. HOAs often default to physical notices or uncoordinated individual outreach.

1. System Access and Security

* Complex user authentication needs to accommodate various roles. HOAs often need to differentiate access levels for board members, homeowners, and administrators.
* Privacy protection requirements. Handling sensitive homeowner information (e.g., financial details) demands robust safeguards.
* Secure management of shared HOA assets (e.g., document access). HOAs struggle to balance transparency with ensuring restricted access to sensitive records.

1. Financial Management

* Inefficient collection and tracking of dues, fines, and other payments. Homeowners often complain about unclear payment methods or late fee policies.
* Lack of transparency in financial reporting. Many HOAs fail to provide clear breakdowns of how dues are spent, leading to homeowner dissatisfaction.
* Difficulty managing multiple payment methods. Homeowners prefer diverse payment options (e.g., online, check, ACH), but HOAs often rely on manual processes.
* Challenges with delinquent payments. Collecting overdue fees is time-intensive and can strain community relationships.

The consequences of these issues extend far beyond mere inconvenience. They can lead to:

1. Legal Vulnerabilities
   * Inconsistent rule enforcement leading to discrimination claims
   * Poor documentation creating liability issues
   * Inadequate record-keeping affecting legal proceedings
   * Non-compliance with state and local regulations
2. Financial Impacts
   * Increased operational costs
   * Lost revenue from poor payment tracking
   * Inefficient use of volunteer time
   * Potential legal expenses from disputes
   * Higher insurance costs due to poor risk management
3. Community Discord
   * Frustration with inefficient processes
   * Lack of trust due to poor transparency
   * Increased conflicts due to communication breakdowns
   * Reduced community engagement
   * Resistance to HOA leadership and decisions
4. Administrative Burden
   * Overwhelming workload for board members
   * Difficulty in recruiting new board members
   * Increased stress on volunteers
   * Inefficient use of time and resources
   * Risk of burnout among board members

These issues are further compounded when HOAs are managed by homeowners themselves rather than professional management companies, as they often lack: - Professional management expertise - Established procedures and systems - Dedicated staff and resources - Technological infrastructure - Training and support systems

## 2. Research & Background

### Project Evolution and Technical Pivot

One of the most significant aspects of this project was its fundamental transformation during the development phase. This evolution represents not just a technical change, but a crucial learning experience that shaped both the project outcomes and my development as a software engineer.

#### Initial Technical Approach

The project initially began with what seemed to be a straightforward and logical technical stack:

1. Oracle Database
   * Chosen for its robustness, reliability, and familiarity
   * Built-in security features
   * Strong data integrity controls
   * Enterprise-level capabilities
   * Easy integration with APEX application
2. Oracle APEX (Application Express)
   * Low-code development platform
   * Built-in components and templates
   * Integrated with Oracle Database
   * Promise of rapid development
   * Web-based development environment
3. Initial Development Process
   * Database design and implementation
   * Data loading and verification
   * APEX application configuration
   * Page and form creation
   * Component customization

#### Challenges with Initial Approach

The Oracle APEX approach revealed several significant limitations and challenges:

1. Permission and Access Issues
   * Restricted ability to implement mass user creation
   * Limited control over authentication processes
   * Constraints on custom functionality
2. Development Limitations
   * Development became heavily focused on configuration rather than programming
   * “Buttonology” - excessive time spent navigating complex UI settings
   * Limited ability to implement custom solutions in timely manner
   * Restricted creative problem-solving
   * Slow development due to platform constraints
3. Learning Constraints
   * Focus on platform-specific knowledge rather than transferable skills
   * Limited exposure to modern development practices
   * Restricted ability to implement current web technologies
   * Reduced opportunities for genuine coding experience
4. Technical Frustrations
   * Complex configuration requirements for simple changes
   * Difficult debugging process
   * Limited control over implementation details
   * Poor development feedback loop
   * Restricted access to modern development tools
   * Inability to save state of development when error present

#### The Pivot Decision

The decision to pivot away from Oracle APEX was driven by several factors:

1. Personal Development Goals
   * Desire for more hands-on coding experience
   * Interest in modern web development practices
   * Alignment with current industry trends
   * Better career skill development
2. Project Quality Concerns
   * Limited ability to create an optimal user experience
   * Restricted customization options
   * Poor mobile responsiveness options
   * Limited modern feature implementation
3. Development Efficiency
   * Slow progress with APEX
   * Time spent fighting platform limitations
   * Reduced enthusiasm for development
   * Limited ability to implement creative solutions

#### Transition to Modern Stack

The project was reimagined with a completely new technical stack:

1. Frontend
   * React.js for component-based development
   * Modern JavaScript (ES6+)
   * Tailwind CSS for styling
   * Responsive design principles
   * Modern UI/UX practices
2. Backend
   * Node.js server
   * RESTful API architecture
   * JWT authentication
   * Modern security practices
   * Flexible routing and middleware
3. Database
   * MySQL for data storage
   * Simplified schema management
   * Easier local development
   * Better integration with modern tools
   * More accessible development environment

#### Impact of the Pivot

The transition to a modern stack had several positive outcomes:

1. Development Experience
   * Increased enthusiasm for the project
   * Better learning opportunities
   * More creative problem-solving
   * Faster development cycles
   * Greater sense of accomplishment
2. Technical Benefits
   * Better control over implementation
   * Improved performance options
   * More flexible architecture
   * Better testing capabilities
   * Enhanced security options
3. Skill Development
   * Experience with modern web development
   * Improved understanding of full-stack development
   * Better alignment with industry practices
   * More transferable skills
   * Deeper technical knowledge
   * Increased exposure to integration of third party applications
4. Project Quality
   * Better user experience
   * More modern features
   * Improved mobile support
   * Better performance
   * More maintainable codebase

### Market Research and Existing Solutions

Extensive research into existing HOA management solutions revealed several categories of available options:

### Market Research and Existing Solutions (continued)

A comprehensive analysis of existing HOA management solutions revealed several categories of available options, each with distinct advantages and limitations:

#### 1. Custom Web Development Solutions

Custom-built HOA management platforms offered through web development agencies:

Advantages: - Tailored to specific HOA needs - Full control over features and functionality - Potential for unique solutions - Direct input into development process

Limitations: - High initial development costs - Ongoing maintenance requirements - Long development timelines (6-12 months) - Risk of developer dependency - Requires technical expertise to manage - High cost of updates and modifications

#### 2. Commercial HOA Management Software

Pre-built software solutions marketed specifically to HOAs:

Advantages: - Immediate availability - Professional support - Regular updates - Established feature sets - Proven reliability

Limitations: - High monthly costs, hundreds to thousands per month - Limited customization options - Feature bloat - Locked into vendor ecosystem - Often designed for large communities - Complex learning curves - Difficult data migration

#### 3. Generic Community Management Platforms

General-purpose community management software adapted for HOA use:

Advantages: - Lower cost than specialized solutions - Basic communication features - Simple implementation - Familiar interfaces

Limitations: - Lack of HOA-specific features - Limited financial management - Poor integration options - Generic functionality - Inadequate security for HOA needs - Limited document management - Basic reporting capabilities

#### 4. Website Builders with HOA Templates

Website building platforms offering HOA-specific templates:

Advantages: - Low initial cost - Easy to set up - Basic web presence - Simple content management - DIY approach

Limitations: - Very limited functionality - No financial management - Basic communication tools only - Poor data organization - Limited security features - No integration capabilities - Not scalable for growing communities

#### 5. Open-Source Management Systems

Community-developed open-source solutions:

Advantages: - No licensing costs - Full code access - Community support - Customization potential - Transparent security

Limitations: - Requires significant technical expertise - Limited professional support - Security concerns - Inconsistent updates - Complex setup process - High maintenance burden - Risk of project abandonment

### Gap Analysis

This research revealed several critical gaps in existing solutions:

1. Accessibility Gap
   * Most solutions either too expensive or too complex
   * Few options suitable for small to medium HOAs
   * Limited options for self-managed communities
2. Feature Balance Gap
   * Solutions either too simple or overly complex
   * Lack of middle-ground options
   * Poor balance of features vs. usability
3. Cost-Effectiveness Gap
   * High costs for comprehensive solutions
   * Limited functionality in affordable options
   * Poor value proposition for small HOAs
4. Technical Expertise Gap
   * Most solutions require significant technical knowledge
   * Limited self-service options
   * Poor documentation and support
5. Integration Gap
   * Limited ability to connect with other systems
   * Poor data portability
   * Closed ecosystems

### Opportunity Identification

Analysis of these gaps revealed several key opportunities:

1. Market Opportunity
   * Need for mid-range solution
   * Demand for user-friendly systems
   * Space for modern technology adoption
   * Potential for scalable solution
2. Technical Opportunity
   * Apply modern web technologies
   * Implement user-centric design
   * Create flexible architecture
   * Build scalable solution
3. User Experience Opportunity
   * Simplify complex processes
   * Improve accessibility
   * Enhance communication
   * Streamline workflows

### Project Positioning

Based on this research, the project was positioned to:

1. Target Requirements
   * Focus on essential features
   * Prioritize user experience
   * Emphasize simplicity
   * Maintain flexibility
2. Technical Approach
   * Use modern technology stack
   * Implement responsive design
   * Create scalable architecture
   * Ensure security
3. Development Strategy
   * Iterative development
   * User-centered design
   * Continuous testing
   * Regular feedback

## 4. Implementation Details

### Database Design and Evolution

#### Initial Oracle Implementation

The project’s database design began in Oracle with a focus on enterprise features:

1. Original Schema Design
   * Complex relationships for HOA entities
   * Heavy use of Oracle-specific features
   * Stored procedures for business logic
   * Complex trigger system
   * Materialized views
2. Data Migration Challenges
   * Complex data types conversion
   * Relationship restructuring
   * Trigger logic translation
   * Stored procedure conversion
   * Permission system redesign

#### MySQL Migration and Optimization

1. Schema Translation
   * Data type transition (Number, Varchar2, Date)
   * Sequences vs. AUTO\_INCREMENT
   * Triggers in database to handling flows outside of database (personal preference)
   * Environment setup and permissions
2. Key Improvements
   * Simplified relationships
   * Better indexing strategy
   * Improved data integrity
   * Enhanced query performance
   * Simplified maintenance

#### MySQL Implementation

1. Account Management Tables
   * ACCOUNT
     + Account identifiers and balances
     + ACCOUNT\_ID (PK), OWNER\_ID (FK), PROPERTY\_ID (FK), BALANCE
   * PROPERTY
     + Property details and location
     + PROP\_ID (PK), UNIT, STREET, CITY, STATE, ZIP\_CODE
   * OWNER
     + Resident information and authentication
     + OWNER\_ID (PK), LAST\_NAME, FIRST\_NAME, PHONE, EMAIL, VOTING\_RIGHTS, PASSWORD\_HASH, IS\_TEMPORARY\_PASSWORD, NOTIFICATION\_PREF\_ID (FK)
2. Relationship Mapping Tables
   * ACCOUNT\_OWNER\_MAP
     + Links accounts to owners with timeline
     + ACCOUNT\_ID (FK), OWNER\_ID (FK), PURCHASE\_DATE, SELL\_DATE
   * PROPERTY\_OWNER\_MAP
     + Property ownership history
     + PROPERTY\_ID (FK), OWNER\_ID (FK), PURCHASE\_DATE, SELL\_DATE
3. Financial Management Tables
   * CREDIT\_CARDS
     + Payment method storage and management
     + CARD\_ID (PK), ACCOUNT\_ID (FK), CARD\_TYPE, CARD\_NUMBER\_LAST\_4, EXPIRY\_MONTH, EXPIRY\_YEAR, IS\_DEFAULT, IS\_ACTIVE
   * PAYMENT
     + Payment transaction records
     + PAYMENT\_ID (PK), ACCOUNT\_ID (FK), OWNER\_ID (FK), DATE\_OF\_PAYMENT, PAYMENT\_AMOUNT, CARD\_ID (FK)
   * ACCOUNT\_CHARGE
     + Charges and assessments tracking
     + CHARGE\_ID (PK), ACCOUNT\_ID (FK), CHARGE\_TYPE, PAYMENT\_DUE\_DATE, VIOLATION\_DATE, VIOLATION\_TYPE\_ID (FK)
4. Assessment and Violation Tables
   * ASSESSMENT\_TYPE
     + Types of assessments
     + TYPE\_ID (PK), ASSESSMENT\_DESCRIPTION
   * ASSESSMENT\_RATE
     + Assessment rate information
     + RATE\_ID (PK), ASSESSMENT\_YEAR, AMOUNT, CHANGED\_BY (FK), IS\_YEARLY\_ASSESSMENT
   * VIOLATION\_TYPE
     + Violation categories and fines
     + TYPE\_ID (PK), VIOLATION\_RATE, VIOLATION\_DESCRIPTION
5. Board Management Tables
   * BOARD\_MEMBER\_ADMIN
     + Board member roles and permissions
     + MEMBER\_ID (PK), MEMBER\_ROLE, ASSESS\_FINES, CHANGE\_RATES, CHANGE\_MEMBERS
   * OWNER\_BOARD\_MEMBER\_MAP
     + Board member tenure tracking
     + OWNER\_ID (FK), BOARD\_MEMBER\_ID (FK), START\_DATE, END\_DATE
6. Communication Tables
   * MESSAGE
     + Internal messaging system
     + MESSAGE\_ID (PK), MESSAGE, CREATED, SENDER\_ID (FK), RECEIVER\_ID (FK), PARENT\_MESSAGE\_ID (FK)
   * OWNER\_MESSAGE\_MAP
     + Message status tracking
     + OWNER\_ID (FK), MESSAGE\_ID (FK), IS\_READ
   * ANNOUNCEMENT\_NEWS
     + Community announcements and events
     + ANNOUNCEMENT\_ID (PK), TITLE, MESSAGE, TYPE, EVENT\_DATE, EVENT\_END\_DATE, FILE\_BLOB
7. Document and Survey Tables
   * DOCUMENT
     + Document storage and management
     + DOCUMENT\_ID (PK), TITLE, DESCRIPTION, FILE\_BLOB, FILE\_MIME, FILE\_NAME, CATEGORY
   * SURVEY
     + Community surveys
     + SURVEY\_ID (PK), MESSAGE, ANSWER\_1-4, CREATED\_BY (FK), END\_DATE, STATUS
   * OWNER\_SURVEY\_MAP
     + Survey response tracking
     + OWNER\_ID (FK), SURVEY\_ID (FK), RESPONSE, RESPONSE\_DATE
8. Preference Management Tables
   * NOTIFICATION\_PREFERENCES
     + User notification settings
     + PREF\_ID (PK), EMAIL\_ENABLED, MESSAGES\_ENABLED, NEWS\_DOCS\_ENABLED, PAYMENTS\_ENABLED, CHARGES\_ENABLED

### Backend Implementation

#### Node.js Server Architecture

1. Core Technologies

* Node.js runtime environment
  + Purpose: Server-side JavaScript execution
  + Version: Latest LTS version
  + Key Features:
    - Event-driven architecture
    - Non-blocking I/O
    - NPM package management
    - Cross-platform compatibility
* Express.js web framework
  + Purpose: Web application framework
  + Implementation:
    - Route handling
    - Middleware support
    - Static file serving
    - Error handling
  + Configuration:
    - JSON parsing
    - URL-encoded parsing
    - Static file paths
    - Error middleware
* MySQL2 for database connectivity
  + Purpose: Database interaction
  + Features:
    - Connection pooling
    - Prepared statements
    - Promise-based queries
    - Transaction support
  + Implementation:
    - Connection pool configuration
    - Query error handling
    - Result parsing
    - Connection management
* JSON Web Tokens (JWT)
  + Purpose: Authentication mechanism
  + Implementation:
    - Token generation with payload
    - Token verification middleware
    - Refresh token handling
    - Token blacklisting
  + Configuration:
    - Token expiration
    - Secret key management
    - Refresh token window

1. API Architecture

* RESTful API design
  + Endpoints follow REST conventions
  + Resource-based routing
  + HTTP method semantics:
    - GET for retrieval
    - POST for creation
    - PUT for updates
    - DELETE for removal
* Modular routing structure
  + Route organization:
    - /auth for authentication
    - /api for protected routes
    - /public for public access
  + Route handlers:
    - Controller separation
    - Service abstraction
    - Error boundary

1. Service Layer Organization

* Authentication Service
  + Files:
    - authService.js
    - refreshToken.js
    - axiosPrivate.js
  + Features:
    - Login validation
    - Token generation
    - Password hashing
    - Session management
    - Password reset flow
    - Registration handling
* Data Services
  + Account Service:
    - Balance calculations
    - Payment processing
    - Transaction history
    - Assessment handling
  + Document Service:
    - File storage
    - Metadata management
    - Access control
    - Search functionality

1. Middleware Components

* Authentication verification
  + Purpose: Route protection
  + Implementation:
    - Token validation
    - Role verification
    - Permission checking
    - Session validation
* Request logging
  + Purpose: Activity tracking
  + Features:
    - Request timing
    - Route logging
    - Error capture
    - User identification

1. Scheduled Tasks

* Survey status updates
  + Schedule: Daily at midnight
  + Actions:
    - Check end dates
    - Update survey status
    - Process results
    - Send notifications
* Annual assessment generation
  + Schedule: Yearly on January 1
  + Process:
    - Calculate assessments
    - Create charges
    - Update balances
    - Send notifications

1. Route Structure

* Public Routes:
* // Authentication endpoints  
  POST /auth/login  
  POST /auth/register  
  POST /auth/refresh-token  
  POST /auth/forgot-password  
  POST /auth/reset-password  
    
  // Public access  
  GET /public/announcements  
  GET /public/documents
* Protected Routes:

*// User Management*

GET /api/profile

PUT /api/profile

PUT /api/contact-info

PUT /api/notification-preferences

PUT /api/password

*// Document Management*

GET /api/documents

POST /api/documents

PUT /api/documents/:id

DELETE /api/documents/:id

GET /api/documents/:id/download

*// Financial Management*

GET /api/payments/history

POST /api/payments

GET /api/cards

POST /api/cards

DELETE /api/cards/:id

PUT /api/cards/:id/default

*// Message Center*

GET /api/messages

POST /api/messages

GET /api/messages/:id

PUT /api/messages/:id/read

DELETE /api/messages/:id

GET /api/messages/thread/:id

*// Announcements*

GET /api/announcements

POST /api/announcements

PUT /api/announcements/:id

DELETE /api/announcements/:id

*// Surveys*

GET /api/surveys

POST /api/surveys

POST /api/surveys/:id/responses

GET /api/surveys/:id/results

*// Board Member Routes*

GET /api/board/verify

GET /api/board/roles

PUT /api/board/roles/:id

POST /api/board/members

PUT /api/board/members/:id/end

*// Violations*

GET /api/violations/types

POST /api/violations/types

PUT /api/violations/types/:id

POST /api/violations

GET /api/violations/:id

*// Assessments*

GET /api/assessments/types

GET /api/assessments/rates

POST /api/assessments/types/batch

POST /api/assessments/rates/batch

POST /api/assessments/issue

#### 7. Database Access Layer

* Connection Management
  + Pool Configuration:
    - Connection limits
    - Timeout settings
    - Retry logic
  + Error Handling:
    - Connection failures
    - Query timeouts
    - Transaction rollbacks
* Query Building
  + Prepared Statements:
    - Parameter binding
    - SQL injection prevention
    - Query optimization
  + Transaction Support:
    - ACID compliance
    - Rollback mechanisms
    - Deadlock handling

#### 8. Security Implementation

* Request Validation
  + Input Sanitization:
    - SQL injection checks
    - Data type validation
  + Access Control:
    - Role-based permissions
    - Resource ownership
    - Action authorization
* Rate Limiting
  + Configuration:
    - Request thresholds
    - Window periods
    - IP tracking
  + Implementation:
    - Rate counter
    - Block logic
    - Reset timing

#### 9. Error Handling System

* Error Categories
  + Validation Errors:
    - Input validation
    - Business rules
    - Data constraints
  + Authentication Errors:
    - Invalid credentials
    - Token expiration
    - Permission denied
  + System Errors:
    - Database failures
    - External service errors
    - Network issues
* Error Responses
  + Structure:
    - Error code
    - Message
    - Details
    - Stack trace (development)

#### 10. External Services Integration

* Email Service (SendGrid)
  + Configuration:
    - API keys
    - Template management
    - Rate limits
  + Implementation:
    - Welcome emails
    - Password reset
    - Notifications
    - Assessment notices
* File Storage
  + Configuration:
    - Storage limits
    - File types
    - Compression settings
  + Implementation:
    - Upload handling
    - Download streaming
    - Cleanup processes

#### 11. Logging System

* Log Categories
  + Application Logs:
    - Error tracking
    - Performance metrics
    - User actions
  + Security Logs:
    - Authentication attempts
    - Permission checks
    - Resource access

### Frontend Implementation Strategy

#### Component Architecture

1. Layout Components

* Header with dynamic navigation
  + Component: Header.js
  + Purpose: Provides top-level navigation and user controls
  + Features: User authentication status display, theme toggle, responsive menu for mobile
  + Integrates with AuthContext for user status and ThemeContext for appearance



Figure 1: Header. Basic navigation of public access pages and information about the HOA.

* Responsive sidebar
  + Component: Sidebar.js
  + Purpose: Main navigation for authenticated users
  + Features: Collapsible design, role-based menu items, mobile drawer mode
  + Uses Lucide-React icons for menu items and custom styling for active states

A screenshot of a computer

Description automatically generated A screenshot of a computer

Description automatically generated

Figures 2 & 3: (Left) Expanded Sidebar for larger screens. (Right) Responsive dropdown header for smaller screens.

* Protected route wrapper
  + Component: ProtectedRoute.js
  + Purpose: Route-level authentication control
  + Features: Automatic redirect to login, role verification, session check
  + Works with AuthContext to manage authentication state
* Theme provider
  + Component: ThemeContext.js
  + Purpose: Global theme management
  + Features: Light/dark mode toggle, persistent theme preference
  + Implements Tailwind light and dark mode classes across application
  + Allows sitewide changes to be made in a single file if required

1. Page Components

* Dashboard views
  + Components: Dashboard.js, BoardMemberDashboard.js
  + Purpose: Main interface after authentication
  + Features:
    - Resident dashboard shows account info, recent charges, quick links
    - Board dashboard provides administrative actions through card interface
  + Integrates with multiple contexts for data and theme management

A screenshot of a computer

Description automatically generated

Figure 4: Dashboard for residents. Offers a quick glance at the standing of the account, charges, updates in the HOA and quick links to other user information and actions.

A screenshot of a board

Description automatically generated

Figure 5: Board Member Dashboard. Offers quick links to board member actions based on the privileges of the board member. A board member with all privileges is shown, however links would be disabled and visibly different if the privileges were not available.

* Owner Information
  + Components: OwnerInfo.js, PersonalInfoModal.js, ContactInfoModal.js
  + Purpose: Profile management interface
  + Features:
    - * Personal information display and editing
      * Contact information management
      * Password management
      * Notification preferences management

A screenshot of a computer

Description automatically generated

Figure 6: Owner Information Page. View and change owner information and preferences.

1. Feature Components

* Card management
  + Components: CardManagementModal.js, PaymentModal.js
  + Purpose: Credit card information handling
  + Features:
    - Add/remove payment methods
    - Set default card
    - Last four digits display
    - Card type detection

A screenshot of a payment method

Description automatically generated A screenshot of a card

Description automatically generated

Figure 7 & 8: Manage Payment Modal: Allows for the selecting of default method of payment, adding additional forms of payment or removing them.

* Payment processing
  + Components: PaymentModal.js, PaymentHistoryModal.js
  + Purpose: Payment simulation and record keeping
  + Features:
    - Payment method selection
    - Amount validation
    - Receipt generation
    - Transaction history display
  + Implements mock payment processing for demonstration

A screenshot of a payment method

Description automatically generated

Figure 9: Payment Modal. Allows payments to be made to an owner account if a balance is on the account. Amounts are validated.

* Notification Preferences
  + Component: NotificationPreferencesModal.js
  + Purpose: Email notification management
  + Features:
    - Master email toggle
    - Granular notification categories
    - Bulk enabe/disable options

A screenshot of a computer

Description automatically generated

Figure 10: Notification Preferences Modal. Update user notification preferences. A user can enable or disable all communications via email or be more granular with their choices.

1. Shared Components

* Loading states
  + Component: LoadingSpinner.js
  + Purpose: Consistent loading indication across app
  + Features:
    - Centralized loading component
    - Customizable size and color
    - Implements loading context for global state
    - Used in data fetching operations
* Error boundaries
  + Component: ErrorBoundary.js
  + Purpose: Graceful error handling
  + Features:
    - Catches JavaScript errors
    - Prevents app crashes
    - Provides user-friendly error messages
    - Logs errors for debugging
* Modal templates
  + Components: BaseModal.js, ModalContext.js
  + Purpose: Standardized modal implementation
  + Features:
    - Reusable modal structure
    - Backdrop click handling
    - Keyboard accessibility (Esc to close)
    - Z-index management
    - Prevents background scrolling

### Communication System Components

1. Messaging Features

* One-to-one messaging
  + Components: Messages.js, MessageList.js, MessageDetail.js
  + Purpose: Direct communication between users
  + Features:
    - Message composition
    - Thread view
    - Timeline view
    - Recipient selection
    - Message status tracking

A screenshot of a computer

Description automatically generated

Figure 11: Message. Allows for creation of new messages, viewing of all messages, and replying to messages. The timeline view will only show a single message.

A screenshot of a computer

Description automatically generated

Figure 12: Message. Allows for creation of new messages, viewing of all messages, and replying to messages. This composition view will show a thread of messages.

* Message composition
  + Components: NewMessageModal.js, ReplyMessageModal.js
  + Purpose: Message creation interface
  + Features:
    - Rich text input
    - Recipient search
    - Draft saving
    - Reply context
    - Character limits

A screenshot of a computer

Description automatically generated A screenshot of a computer

Description automatically generated

Figure 13 & 14: (Left) Recipient selection for message. (Right) Message content creation

A screenshot of a computer

Description automatically generated

Figure 15: Message Reply Modal

1. Announcement Management

* Event creation
  + Components: CreateAnnouncementModal.js, AnnouncementModal.js
  + Purpose: Community event management
  + Features:
    - Date/time selection
    - Location input
    - Image attachment
    - Duration management
    - Scheduling options

A screenshot of a computer

Description automatically generated

Figure 16: Announcement Creation Modal. Allows a board member to create and schedule announcements and events to be posted to the announcement page.

A screenshot of a computer

Description automatically generated

Figure 17: Announcement Carousel. Shows all active announcements and allows user to scroll through them.

* Calendar integration
  + Components: AnnouncementCalendar.js, AnnouncementCarousel.js
  + Purpose: Event visualization
  + Features:
    - Monthly calendar view
    - Event details popup
    - Date navigation
    - Event categorization
    - Mobile-responsive design

A screenshot of a computer

Description automatically generated

Figure 18: Events Calendar. Tracks all current events that are scheduled by HOA board members and displays them visually for the user.

1. Survey System

* Survey creation
  + Components: CreateSurveyModal.js, Survey.js
  + Purpose: Community feedback collection
  + Features:
    - Question input
    - Answer option management
    - End date setting
    - Preview functionality
    - Board member access control

A screenshot of a computer

Description automatically generated

Figure 19: Survey. Allows user to view all active and inactive surveys and participate in surveys that they have not yet participated in.

A screenshot of a survey

Description automatically generated

Figure 20: Survey Creation Modal. Allows are board member to quickly generate a new text based survey. A basic survey question and a minimum of 2 answer options are required with a maximum of 4 answers being allowed.

* Results visualization
  + Components: SurveyResultsModal.js, SurveyCompletionModal.js
  + Purpose: Survey data presentation
  + Features:
    - Real-time results
    - Charts using Recharts
    - Percentage calculations
    - Response tracking
    - Export functionality

A screenshot of a survey results

Description automatically generated

Figure 21: Results Preview Modal. Allows users to view the responses to a survey and displays the results graphically. Results are anonymous. Results are also sent to HOA members in message format upon the close of a survey.

### Document Management System

1. Document Features

* Document listing
  + Components: Documents.js, DocumentCard.js
  + Purpose: File organization and display
  + Features:
    - Grid/list view toggle
    - Sort functionality

A screenshot of a computer

Description automatically generated

Figure 22: Documents. Displays all current documents available for HOA members to download. Two views are available with sorting options. Only board members are allowed to upload documents.

* Upload interface
  + Components: UploadDocumentModal.js, ManageDocumentsModal.js
  + Purpose: File upload and management
  + Features:
    - Drag-drop support
    - File validation
    - Progress indication
    - Metadata input
    - Category assignment

A screenshot of a computer

Description automatically generated

Figure 23: Document Upload Modal. Allows a board member to upload a number of different file types, and categorize the files while adding a title and description to the document. Documents can be downloaded by any member of the HOA.

### Payment System Components

1. Payment Processing

* Card management
  + Components: CardManagementModal.js
  + Purpose: Payment method handling
  + Features:
    - Card input validation
    - Card type detection
    - Default card selection
    - Card deletion
    - Secure storage simulation
* Transaction handling
  + Components: PaymentModal.js, PaymentHistoryModal.js, PaymentReceipt.js
  + Purpose: Payment simulation
  + Features:
    - Amount validation
    - Payment method selection
    - Receipt generation
    - Transaction history
    - Balance updates

1. Financial Records

* History display
  + Components: PaymentHistoryModal.js, ChargeDetailsModal.js
  + Purpose: Transaction record keeping
  + Features:
    - Filterable history
    - Detailed transaction view
    - Receipt generation
    - Export functionality
    - Payment type categorization

A screenshot of a computer

Description automatically generated

Figure 24: Account History: Displays all of the transactions, categorized by year, allowing the user access to anyone of the transactions and further details and receipts for those transactions.

## 6. Test Plan / Evolution of the Test Plan

### Initial Automated Testing Approach

The project initially aimed for comprehensive automated testing but encountered several challenges:

1. Challenges Faced
   * Steep learning curve with testing frameworks
   * Complex setup requirements
   * Time-intensive test creation
   * Unreliable test results
   * Framework compatibility issues
2. Impact on Development
   * Delayed feature implementation
   * Reduced confidence in tests
   * Increased development complexity
   * Resource allocation issues
   * Timeline concerns

### Pivot to Manual Testing

The decision to shift to manual testing proved beneficial:

1. Manual Testing Approach
   * Comprehensive test cases
   * Detailed documentation
   * Real-world scenarios with user-centric testing
2. Testing Documentation
   * Comprehensive set of tests
   * Log of testing completions
   * Expected results examined in both UI and database
   * Issue tracking and resolution
3. Testing Areas
   * User Interface
   * Business Logic
   * Data Integrity
   * Security Features

The test plan is outlined in HOA\_MANAGEMENT\_TP\_01.docx

The majority of testing took place utilizing the Requirements Traceability Matrix and verifying that each test passed. If there was an issue, it was addressed on the spot before moving on to the next set of test and logged in the issues log.

Full program flows were tested at most parts of development and during individual components testing, however a separate testing document was created for Integration Testing. HOA\_MANAGEMENT\_INT\_TP\_01.docx serves as the guide for all integration testing of the HOA Management System, including steps and expectations for each test.

Another separate testing document was created specifically for security testing as this isn’t something that can easily be spotted and required more attention to detail. HOA\_MANAGEMENT\_SEC\_TP\_01.docx serves as the guide for all security related testing for the HOA Management System, including steps and expectations for each test.

## 7. Test Results

### Comprehensive Testing Outcomes

1. User Interface Testing
   * Responsive Design Testing
     + Verified layouts at breakpoints: <768px, 768px-1024px, >1024px
     + Tested all modals for responsive behavior
     + Confirmed sidebar collapse functionality
     + Validated calendar views on mobile devices
     + Tested grid/list view transitions Results:
     + Fixed modal scrolling issues on mobile devices
     + Improved table responsiveness for payment history
     + Enhanced navigation menu for tablet views
     + Optimized image scaling in announcements
     + Resolved carousel control visibility issues
2. Authentication Testing
   * Login Flow Testing
     + Verified email format validation
     + Tested password requirement enforcement
     + Validated temporary password workflow
     + Confirmed “remember me” functionality
   * Session Management
     + Tested token refresh mechanism
     + Verified timeout handling
     + Validated multi-tab behavior
     + Tested board member permission inheritance Results:
     + Fixed race condition in token refresh
     + Improved session timeout notifications
     + Enhanced error messages for login failures
     + Resolved permission verification delays
3. Data Management Testing
   * Database Operations
     + Tested all CRUD operations across 20 tables
     + Verified foreign key constraint enforcement
     + Validated cascade operations for deletions
     + Tested transaction integrity
   * Data Validation
     + Verified input constraints
     + Tested duplicate entry handling
     + Validated date format consistency
     + Confirmed numeric precision handling Results:
     + Resolved owner-property mapping issues
     + Fixed payment record consistency
     + Improved error handling for duplicate entries
     + Optimized query performance for document retrieval
4. Payment System Testing
   * Card Management
     + Tested card addition workflow
     + Verified card number validation
     + Confirmed default card selection
     + Tested card deletion handling
   * Transaction Processing
     + Validated balance calculations
     + Tested payment allocation logic
     + Verified receipt generation
     + Tested payment history recording Results:
     + Enhanced card validation feedback
     + Improved payment confirmation display
     + Fixed balance calculation precision
     + Resolved receipt formatting issues
     + Optimized payment history loading
5. Communication System Testing
   * Message Management
     + Tested thread creation and replies
     + Verified read/unread status updates
     + Tested message search functionality
     + Validated notification delivery
   * Announcement Handling
     + Tested scheduled publication
     + Verified image attachment handling
     + Validated event date management
     + Tested calendar integration Results:
     + Improved thread organization
     + Enhanced message status tracking
     + Fixed scheduled announcement timing
     + Resolved event display issues
6. Document System Testing
   * File Operations
     + Tested upload size limits
     + Verified file type restrictions
     + Validated download functionality
     + Tested category organization
   * Search and Display
     + Verified search functionality
     + Tested grid/list view switching
     + Validated sort operations
     + Confirmed filter functionality Results:
     + Improved upload error handling
     + Enhanced download reliability
     + Fixed category filtering issues
     + Resolved sort order persistence
7. Integration Testing

* HOA-INT-IT-PAY-010 was decided to be not a requirement that needed to be tested or implemented.
* HOA-INT-IT-DOC-002, 005, 009, 010 were decided to be not a requirement that needed to be tested or implemented.
* HOA-INT-IT-MSG-002, 008, 009, 010 were decided to be not a requirement that needed to be tested or implemented.
* Card deletion was discovered to be an issue as removing it from the system created an error on the foreign key. A new card deletion flow was created which led to a better system of checks for adding cards as well.

In total 21 issues were raised in the testing of the HOA Management System that required changes to the program to be made. These ranged from responsiveness, database issues, broken flows, improper field validation and navigation issues.

## 8. Challenges Overcome

### Technical Challenges

1. Platform Migration
   * Challenge: Complex transition from Oracle APEX to React/Node.js
   * Solution:
     + Complete rebuild
     + Used APEX as wireframe/template for development
   * Outcome: Successfully transitioned the application quickly and built momentum to continue development
2. Authentication System
   * Challenge: Token refresh race conditions and session management
   * Solution:
     + Implemented token memoization
     + Enhanced session tracking
     + Improved error handling
   * Outcome: Robust and reliable authentication system, that allowed proper redirects and accesses
3. Mobile Responsiveness
   * Challenge: Complex UI elements not scaling properly
   * Solution:
     + Redesigned component architecture
     + Implemented responsive design patterns
     + Enhanced mobile-first approach
   * Outcome: Fully responsive application across all devices
4. Database Performance
   * Challenge: Slow queries and inefficient data access
   * Solution:
     + Optimized database schema
     + Implemented proper indexing
     + Enhanced query optimization
   * Outcome: Significant performance improvements particularly with loading and retrieval of documents

### Development Process Challenges

1. Testing Strategy Pivot
   * Challenge: Ineffective automated testing approach
   * Solution:
     + Developed comprehensive manual testing plan
     + Created detailed test documentation
     + Implemented structured test procedures
   * Outcome: More reliable and efficient testing process
2. Feature Prioritization
   * Challenge: Balancing essential features vs nice-to-have features
   * Solution:
     + Created priority matrix
     + Focused on core functionality
     + Implemented phased development
   * Outcome: Successfully delivered essential features with clear path for enhancements
3. Learning Curve
   * Challenge: Adapting to new technology stack
   * Solution:
     + Dedicated learning time
     + Proof of concept development
     + Iterative implementation
   * Outcome: Successfully mastered new technologies while maintaining progress

### Implementation Challenges

1. Payment System Design
   * Challenge: Creating realistic payment simulation
   * Solution:
     + Developed mock payment processor
     + Implemented comprehensive transaction tracking
   * Outcome: Functional demonstration system with future integration capability
2. Document Management
   * Challenge: Efficient handling of various document types
   * Solution:
     + Implemented structured storage system
     + Created flexible categorization
     + Enhanced search functionality
   * Outcome: Robust document management system
3. Communication System
   * Challenge: Managing complex message threading
   * Solution:
     + Implemented hierarchical message structure
     + Enhanced notification system
     + Improved message tracking
   * Outcome: Comprehensive communication platform

## 9. Future Enhancements

### Technical Enhancements

1. Integration Capabilities
   * Real payment gateway integration
   * Third-party service connections
2. Mobile Applications
   * Native mobile app development
   * Cross-platform compatibility
   * Offline capabilities
   * Push notification system ### Feature Enhancements
3. Advanced Communication Tools
   * Real-time chat functionality
   * Community forums
4. Financial Management
   * Real payment gateway integration
   * Automated payment plans
   * Tax document generation
5. Community Features
   * Event management system
     + Resource booking
     + Attendance management
   * Maintenance request system
     + Work order tracking
     + Maintenance scheduling
6. Administrative Enhancements
   * Advanced reporting system
     + Custom report builder
     + Scheduled reports
     + Export options
   * Compliance tracking
     + Appeal management ### Onboarding / HOA Setup There is currently no implementation for the initial setup of the application for a new customer. The actual hosting of the web application would not be the issue, it would be the database setup itself that would become the main problem. The list of issues include:
7. Initial Access Bootstrap Problem

* No users exist in the system initially, making login impossible.
* A first administrator account with full permissions is required to initiate setup.
* System must enable at least one board member to configure properties, owners, and roles.

1. Property Records Bootstrap

* Property records need to be created before assigning homeowner accounts.
* Without properties, owner accounts cannot be properly linked.
* Initial property data is crucial for effective HOA management.

1. Board Structure Bootstrap

* Board member roles must be defined before any system features can be used.
* At least one administrator with elevated privileges is required.
* Role assignments are essential to configure board-specific functionality.

The potential approaches in an incremental improvement process to address this issue are as follows: 1. Manual First-Time Setup A straightforward approach where the system implementer performs initial setup tasks: - Manually insert necessary database records for: \* First administrator account. \* Initial board member role(s). \* Mapping admin to appropriate role. - Provide secure credentials to HOA leadership. - Document the initial setup process for future reference. 2. Setup Wizard Approach An interactive, guided solution for HOAs to self-configure the system: - Accessible only when no accounts exist. - Guides users through essential steps, including: \* First admin account setup. \* Initial role assignments. \* Property data entry. - Requires a secure token or key for access.

## 10. Lessons Learned

### Technical Insights

1. Technology Stack Decisions
   * Early platform decisions have long-lasting impacts
   * Importance of flexibility in technology choices
   * Value of modern development approaches
   * Need for scalable solutions
   * Impact of technical debt
2. Development Process
   * Benefits of iterative development
   * Importance of proper planning
   * Value of continuous testing
   * Need for documentation
   * Significance of code quality
3. Testing Approach
   * Reality of testing frameworks
   * Value of manual testing
   * Importance of test planning
   * Need for balanced approach
   * Documentation significance

### Project Management Insights

1. Planning and Execution
   * Importance of flexible planning
   * Value of regular reassessment
   * Need for clear communication
   * Significance of documentation
   * Impact of proper scoping
2. Resource Management
   * Time allocation importance
   * Learning curve considerations
   * Tool selection impact
   * Documentation needs
   * Support requirements
3. Risk Management
   * Technology risk assessment
   * Learning curve impact
   * Timeline management
   * Resource allocation
   * Scope control

### Personal Development

1. Technical Growth
   * Modern web development expertise
   * Database management skills
   * Security implementation knowledge
   * Testing methodologies
   * Architecture design experience
2. Professional Development
   * Project management skills
   * Documentation practices
   * Problem-solving abilities
   * Decision-making capabilities
   * Time management skills

## 11. Conclusion

The HOA Management System project represents a significant learning experience and technical achievement. The pivot from Oracle APEX to a modern web stack, while challenging, proved to be a crucial decision that enhanced both the project outcome and personal development. The resulting system provides a solid foundation for HOA management while offering clear paths for future enhancement and expansion. While certain aspects, such as payment processing, remain in simulation, the system architecture supports future integration of real-world payment systems and other enhancements.