Department of Computer Science & Engineering
The University of Texas at Arlington

System Requirements Specification
CSE 4316: Senior Design I
Fall 2025

Builders Squad
MavHousing

Aastha Khatri
Alok Jha
Aviral Saxena
Atiqur Rahman
Talha Tahmid
Md Rashidul Alam Sami

Revision History

Revision	Date	Author(s)	Description
0.1	09.25.2025	AJ	Document setup and outline
0.2	09.25.2025	AK, AJ	Added core project details and objectives
0.3	09.26.2025	AK	Designed Figma mockups for concept UI screens and user flows
0.4	10.28.2025	AJ	Made Conceptual Product Diagram
0.5	10.29.2025	AK	Expanded Features, Inputs/Outputs, and UI sections
0.6	10.30.2025	AJ	Completed Product Concept and Product Description sections
0.7	10.31.2025	AK	Completed Customer Requirements, Packaging Requirements, and Performance Requirements
0.8	11.1.2025	AJ, AK, TT	Completed Safety Requirements and Maintenance & Support Requirements
0.9	11.2.2025	AJ, AK, TT, AS, AR, MS	Reviewed and refined document for submission

CONTENTS

LIST OF FIGURES

1 Product Concept

This section describes the purpose, use, and intended user audience for the MavHousing system. MavHousing is a comprehensive housing management platform designed to centralize and automate student housing operations at the University of Texas at Arlington. The system streamlines housing applications, room assignments, maintenance requests, payment processing, and communication between students and housing staff through an intuitive web-based interface with AI-powered assistance. Users of MavHousing will be able to submit and track housing applications, manage room assignments, request and monitor maintenance services, process payments, and receive instant answers to housing-related questions through an integrated AI assistant accessible via web portal and Microsoft Teams.

1.1 Purpose and Use

MavHousing replaces UTA's fragmented housing infrastructure with a unified platform that automates student housing operations from application through move-out. Students can submit applications, track status, request maintenance with photos, and receive instant AI-powered answers via web or Microsoft Teams. Housing staff process applications, execute intelligent room assignments, manage maintenance workflows, and access real-time analytics on occupancy and operational metrics. The system provides 24/7 access, transparent tracking, and complete audit trails for data-driven decision-making.

1.2 Intended Audience

MavHousing is specifically designed for the University of Texas at Arlington housing ecosystem, serving four distinct user groups with tailored capabilities:

Students: The system's primary user base consists of current and prospective UTA students seeking on-campus housing, including incoming freshmen, transfer students, returning residents, and graduate students. Students interact with the platform to submit housing applications with roommate preferences, track application status, browse available housing options, manage room assignments, submit and monitor maintenance requests, and access payment information. The student interface prioritizes simplicity, mobile responsiveness, and 24/7 accessibility to accommodate diverse schedules and varying levels of technical proficiency.

Housing Staff: UTA Housing & Residence Life professional staff represent the administrative user group, including housing coordinators, assignment managers, and department administrators. These users manage operational aspects such as reviewing and processing applications, executing room assignments using the intelligent matching algorithm, managing building and room inventory and configuring system settings and permissions. Housing staff require robust performance, intuitive workflows, and powerful search capabilities for daily operations and intensive peak-period usage.

Maintenance Personnel: Facilities and maintenance staff use the system to manage the complete lifecycle of maintenance requests. This includes supervisors who assign and prioritize work orders, technicians who update request status and document completed repairs, and facilities managers who analyze maintenance trends.

University Administration: Secondary stakeholders include UTA administrators in Student Affairs, Budget and Finance who access system-generated reports and analytics dashboards.

2 Product Description

This section provides the reader with an overview of the MavHousing system. The primary operational aspects of the product, from the perspectives of end users, maintainers, and administrators, are defined here. The key features and functions of the system, as well as critical user interactions and interface components, are described in detail. The intent of this section is to provide sufficient context for understanding the systemâs architecture, data flow, and user experience as described in later sections of this document.

MavHousing is a full-stack web application designed to streamline and modernize student housing management at the University of Texas at Arlington. The platform consolidates housing applications, payments, maintenance requests, and communication into a single integrated system. It provides role-based access for students, housing staff, and administrators, ensuring that each user has secure, targeted access to relevant operations. Built using Next.js and NestJS, and deployed on Google Cloud Platform, MavHousing emphasizes scalability, performance, and maintainability while maintaining compliance with institutional security and data standards.

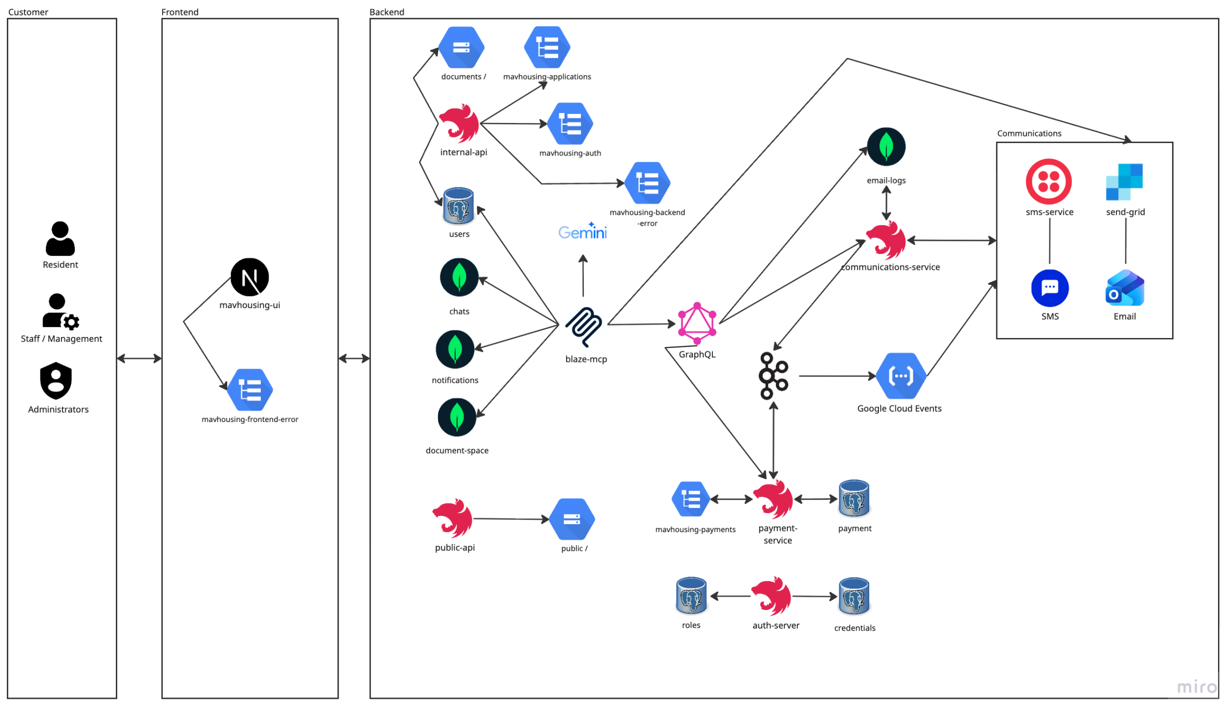


Figure 1: MavHousing System Architecture Overview

2.1 Features & Functions

EMAIL COMMUNICATION

The system will provide automated email communication capabilities to ensure timely updates and effective engagement between students, housing staff, and administrators. Email notifications will be automatically triggered for key user actions such as application submissions, payment confirmations, maintenance requests, and account updates.

Users will receive scheduled reminder emails for critical deadlines, including rent due dates, contract renewals, and move-in or move-out timelines. Administrators will have access to a visual campaign editor that allows them to create, schedule, and manage announcement campaigns with support for

attachments and basic content formatting.
Audience targeting will enable messages to be sent to specific user groups, such as residents of a particular building, floor, or role category. All email templates will support dynamic content and personalization fields (e.g., user names, due dates, and payment amounts) to improve message relevance and clarity. These communications will be securely transmitted using institutional email standards to maintain compliance with university policies and privacy requirements.

IN-APP COMMUNICATION

The system will feature a centralized communication center where users can access all messages, alerts, and announcements within the MavHousing portal. Real-time chat will enable direct interaction between residents and housing staff, supporting quick issue resolution and optional file sharing for maintenance requests. In-app alerts will notify users about maintenance updates, building announcements, and emergencies, with acknowledgment tracking to confirm receipt. These features improve transparency, responsiveness, and user engagement through seamless, real-time communication.

USER REGISTRATION & ACCOUNT CREATION

The system provides a secure, streamlined registration process for users to create accounts using their official UTA email addresses. Password policies will comply with institutional security standards, including minimum length, complexity, and periodic expiration requirements.After registration, users receive a verification email containing a unique confirmation link to validate their address. This step ensures that only authorized UTA users can activate accounts. Once verified, users complete an initial profile setup with essential information such as full name, student ID, contact details, and emergency contact. All data is securely stored in the system database and linked to the designated user role. An integrated account recovery mechanism allows users to reset passwords through their verified email using a secure, time-limited token.
Administrative users (e.g., housing staff or system administrators) can create new accounts and assign roles, but cannot grant privileges higher than their own, preserving the systems hierarchical access control and overall security integrity.

INCIDENT AND MAINTENANCE REQUEST MANAGEMENT

The system features an integrated maintenance and incident management module that enables users to efficiently report, track, and resolve facility-related issues. Students can submit requests with detailed descriptions and optional attachments directly from their dashboard, while housing staff and administrators can update statuses, assign tasks, and communicate within each ticket thread. Automated notifications keep users informed at every stage, ensuring timely responses and accountability. By providing full lifecycle visibility and performance tracking, this module improves operational efficiency, transparency, and overall user satisfaction.

STUDENT USER DASHBOARD & PROFILE MANAGEMENT

The system provides each student with a personalized dashboard that consolidates all housing-related informationâsuch as application status, room assignment, payments, and maintenance requests into a single, user-friendly interface. Quick access menus allow students to submit applications, make payments, request maintenance, or contact support with minimal navigation. The profile management module enables users to update personal details, housing preferences, and emergency contacts, all of which are securely stored to maintain data accuracy and privacy. Students can also access important documents, such as contracts and checklists, and manage visibility settings for shared profile information. Overall, the dashboard offers a centralized, transparent, and secure platform for managing every aspect of the student housing experience.

APPLICATION & DOCUMENT MANAGEMENT FEATURES

The system includes a comprehensive application and document management module that allows students to complete, submit, and track housing applications seamlessly within the MavHousing portal. Students can update personal details and upload required documents such as identification or proof of enrollment, which are securely stored in a cloud-based repository and linked to their applications. Housing staff have role-based access to review, approve, or reject applications, with automated validation to ensure correct file formats and prevent errors. Notifications are sent upon submission, approval, or when additional documents are needed. By centralizing this process, the system eliminates manual paperwork, reduces administrative workload, and maintains transparency and data security through encrypted storage and controlled access.

AUTHENTICATION & AUTHORIZATION - FEATURE IDEAS

The system implements a secure authentication and authorization framework that manages user access based on defined roles and privileges. Each user logs in with verified credentials, and permissions are assigned according to their role. Role-based authentication ensures that students, housing staff, and administrators can only access features relevant to their responsibilities. For example, administrators can manage users, review system data, and configure settings, while students can access housing applications, payments, and maintenance requests. The system enforces strict access controls to prevent unauthorized data access or modification. All authentication processes use encrypted communication, and sessions are automatically timed out after inactivity to maintain security. This framework forms the foundation for data integrity and privacy while ensuring compliance with institutional security standards.

OPTIMAL UNIT ASSIGNMENT AND ROOMMATE MATCHER

The system includes an intelligent unit assignment and roommate matching module that optimizes room allocation based on predefined criteria and student preferences. During the application process, students can specify factors such as building choice, room type, occupancy, budget, and lifestyle habits. The system analyzes these inputs to generate fair and efficient assignments that balance user preferences with room availability. The roommate matching component uses a rule-based algorithm enhanced with machine learning trained on anonymized housing data to calculate compatibility scores between applicants. Administrators can review and adjust suggested pairings before final approval. The process also accounts for gender policies, capacity limits, and accessibility requirements. Once assignments are finalized, students receive email and in-app notifications with their housing details. This module improves fairness, efficiency, and satisfaction through data-driven decision-making and transparent communication.

AI ASSISTANT (BLAZE)

The system features an integrated AI assistant named Blaze, designed to enhance user interaction, automate routine inquiries, and streamline communication between students and housing staff. Powered by the Model Context Protocol (MCP) framework and integrated with the Gemini large language model, Blaze provides secure, context-aware access to real-time system data. It can answer natural language questions about application status, rent deadlines, and maintenance progress with accurate, data-driven responses. Beyond chat support, Blaze sends personalized reminders, notifications, and updates through in-app messages, email, and Microsoft Teams. Users can also interact with Blaze directly through email or as a chatbot within Teams, ensuring convenient access across platforms. This feature improves operational efficiency and engagement by offering 24/7 intelligent assistance, faster responses, and consistent automated communication.

2.2 External Inputs & Outputs

This section describes the critical data flows into and out of the MavHousing system. These include user-generated inputs, external service integrations, and system outputs consumed by users or third-party systems. The tables below summarize key inputs and outputs along with their purposes.

2.3 External Inputs

Name / Source	Description	Use in System
User Registration Data	Input from students and staff during account creation, including UTA email, password, name, student ID, and emergency contact details.	Used to authenticate and create user profiles in the SQL user database with role-based access permissions.
Login Credentials	UTA email and password entered through the frontend login interface.	Validated via the authentication service to issue secure session tokens for authorized access.
Application & Document Submissions	Housing applications and uploaded forms or identification documents submitted by students.	Stored in cloud storage with reference URLs in the NoSQL database, accessible to authorized staff for review.
Maintenance Requests	Submissions containing issue descriptions, categories, locations, and optional attachments.	Recorded as service tickets for tracking, assignment, and status management through the maintenance module.
Payment Actions	Rent or deposit payment requests initiated by students.	Processed via the payment gateway API, updating transaction records and triggering confirmation emails.
Roommate Preferences	Lifestyle, budget, and building preferences provided during application.	Processed by the roommate and unit assignment model to compute compatibility and generate housing allocations.
Chat and Message Events	Real-time messages exchanged between users.	Stored in the NoSQL database for chat history, presence tracking, and message delivery.
Administrative Configurations	Admin inputs for announcements, campaigns, or user management.	Updates backend modules and triggers event-driven functions for automation.
System Event Triggers	Inputs from backend services or monitoring tools (e.g., downtime alerts).	Used to initiate automated notifications or escalation workflows.

Table 2: External Inputs

2.4 External Outputs

Name / Destination	**Description**	**Use in System**
Email Notifications	Automated emails for application submissions, payments, maintenance updates, and reminders.	Sent through SMTP with dynamic fields for personalization and compliance with university communication standards.
AI Assistant Responses (Blaze)	Natural language replies generated from MCP server queries to user requests.	Returned via in-app chat, email (`blaze@mavhousingexample.edu`), or Microsoft Teams integration.
Application Status Updates	Real-time application and document verification results.	Displayed in student dashboards and synchronized with admin review panels.
Maintenance Status Alerts	Automatic ticket updates and resolution notifications.	Displayed in user dashboards and optionally mirrored in Microsoft Teams.
Dashboard Data Feeds	Aggregated housing status, balances, and notifications fetched via GraphQL APIs.	Rendered dynamically on the Next.js frontend through GraphQL API calls.
Chat Notifications	WebSocket alerts for new messages, mentions, or replies.	Displayed in the in-app chat module and persisted in NoSQL.
Analytics and Reports	Data summaries for administrators such as occupancy and maintenance metrics.	Exported through dashboard visualizations or as CSV/PDF files.
Document Access Links	Pre-signed URLs from Google Cloud Storage for uploaded contracts or forms.	Shared securely with authorized users and automatically expired after a defined duration.
System Logs and Audit Trails	Records of authentication events, API calls, and admin actions.	Stored in secure audit logs for monitoring and compliance.
External API Data	Outgoing responses to systems such as payment gateways or Microsoft Teams.	Used to synchronize events, messages, and workflows across integrated platforms.

Table 3: External Outputs

2.5 Product Interfaces

The following images are samples of key high-level screens for students and administrators. These Figma mockups represent the main interfaces of the MavHousing system and may change as the design evolves. The student portal includes login, dashboard, maintenance requests, and payment pages, allowing students to manage housing tasks easily. The management view provides staff with access to maintenance requests and related details, while the integrated BlazeAI chat assists users with quick inquiries.

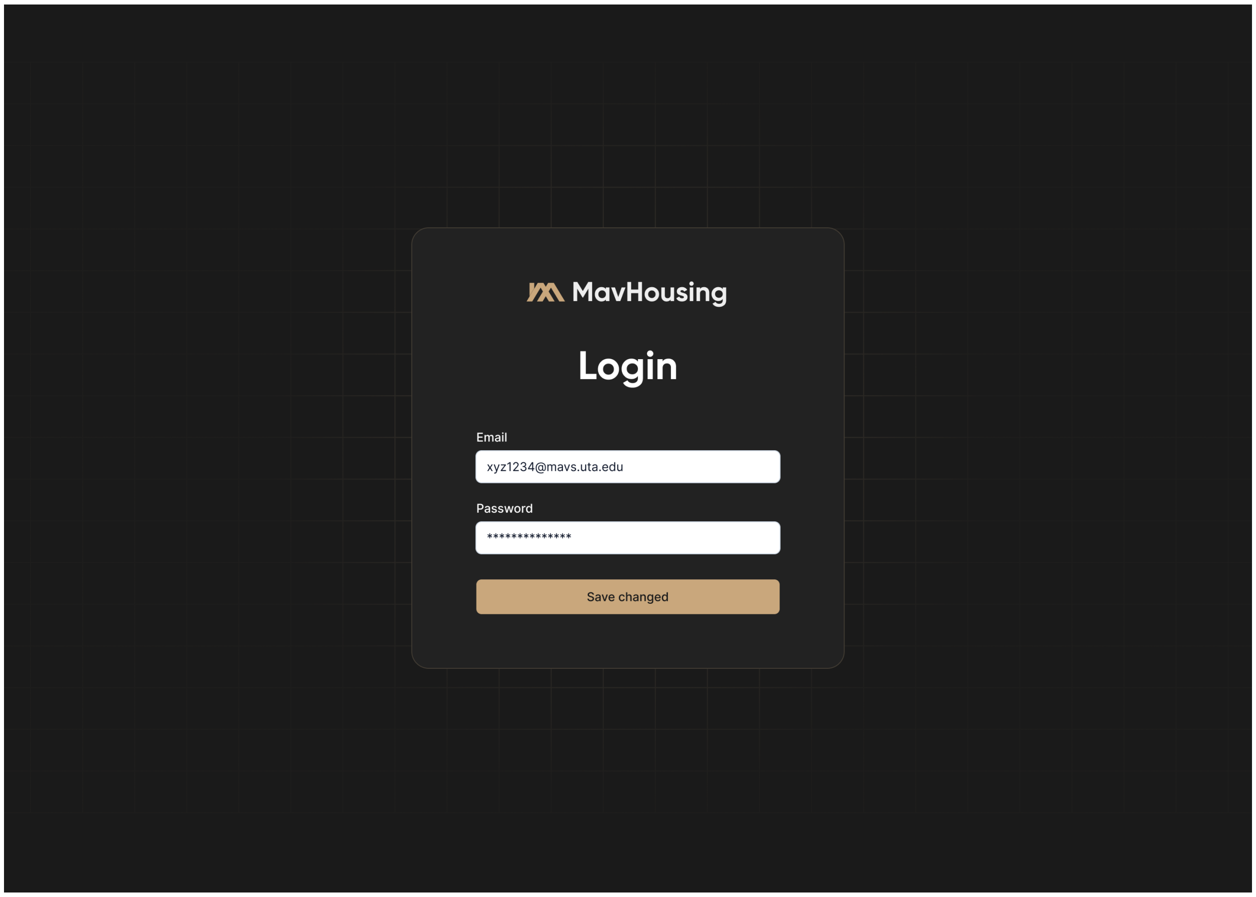
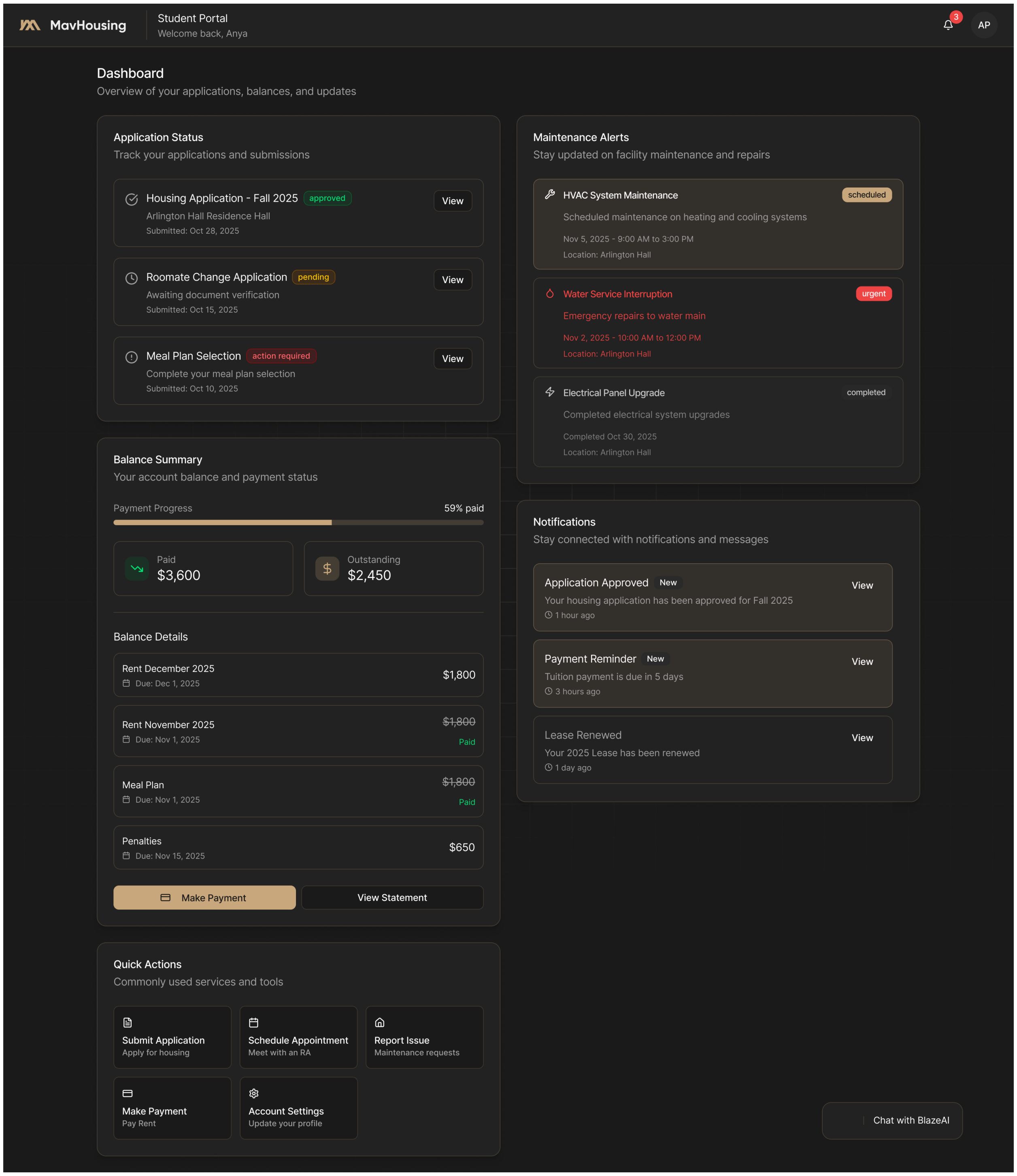


Figure 2: Login Page



Dashboard

Overview of your applications, balances, and updates

Application Status

Track your applications and submissions

Housing Application - Fall 2025 approved
Arlington Hall Residence Hall
Submitted: Oct 28, 2025
View

Roomate Change Application pending
Awaiting document verification
Submitted: Oct 15, 2025
View

Meal Plan Selection action required
Complete your meal plan selection
Submitted: Oct 10, 2025
View

Maintenance Alerts

Stay updated on facility maintenance and repairs

HVAC System Maintenance scheduled
Scheduled maintenance on heating and cooling systems
Nov 5, 2025 - 9:00 AM to 3:00 PM
Location: Arlington Hall

Water Service Interruption urgent
Emergency repairs to water main
Nov 2, 2025 - 10:00 AM to 12:00 PM
Location: Arlington Hall

Electrical Panel Upgrade completed
Completed electrical system upgrades
Completed Oct 30, 2025
Location: Arlington Hall

Balance Summary

Your account balance and payment status

Payment Progress 59% paid

Paid $3,600

Outstanding $2,450

Balance Details

Rent December 2025 Due: Dec 1, 2025	$1,800
Rent November 2025 Due: Nov 1, 2025	~~$1,800~~ Paid
Meal Plan Due: Nov 1, 2025	~~$1,800~~ Paid
Penalties Due: Nov 15, 2025	$650

Notifications

Stay connected with notifications and messages

Application Approved New
Your housing application has been approved for Fall 2025
1 hour ago
View

Payment Reminder New
Tuition payment is due in 5 days
3 hours ago
View

Lease Renewed
Your 2025 Lease has been renewed
1 day ago
View

Quick Actions

Commonly used services and tools

Submit Application
Apply for housing

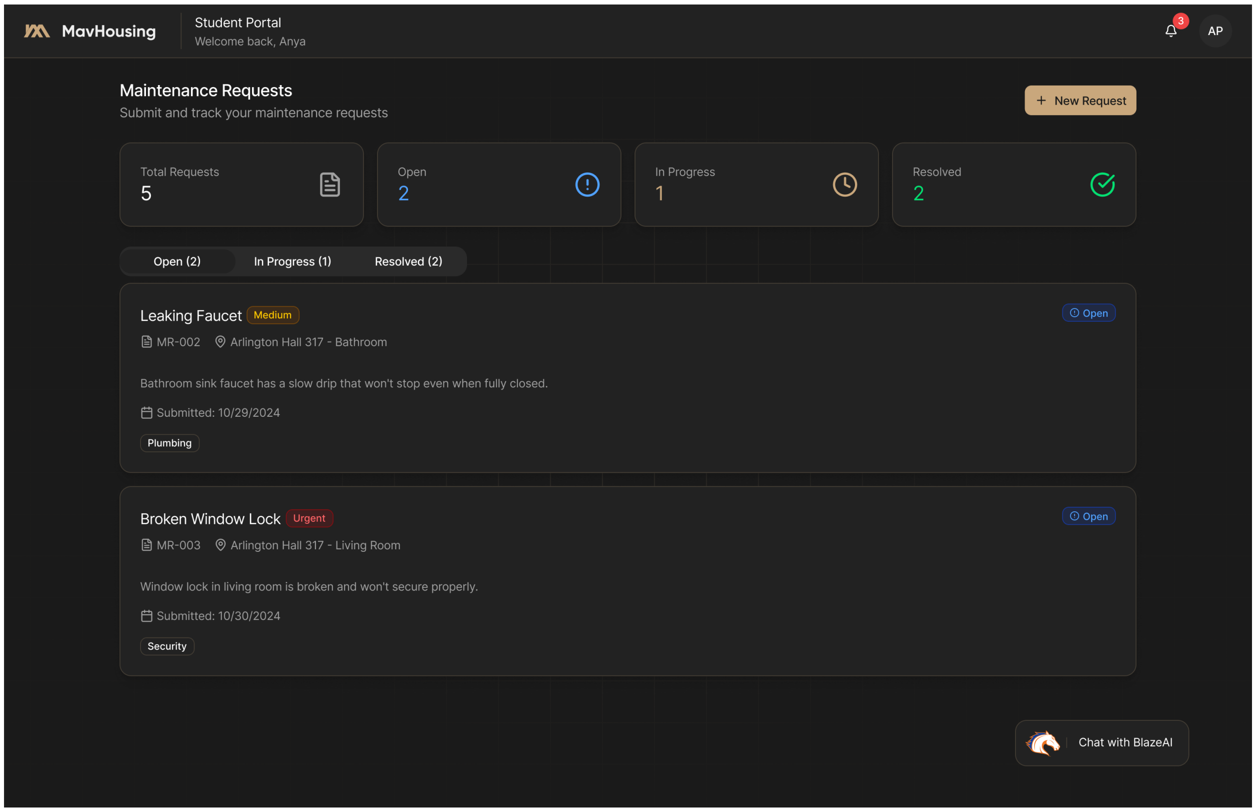
Schedule Appointment
Meet with an RA

Report Issue
Maintenance requests

Make Payment
Pay Rent

Account Settings
Update your profile

Figure 3: Student Dashboard



Maintenance Requests

Submit and track your maintenance requests

Total Requests	Open	In Progress	Resolved
5	2	1	2

Leaking Faucet Medium — Open

MR-002 · Arlington Hall 317 - Bathroom

Bathroom sink faucet has a slow drip that won't stop even when fully closed.

Submitted: 10/29/2024

Plumbing

Broken Window Lock Urgent — Open

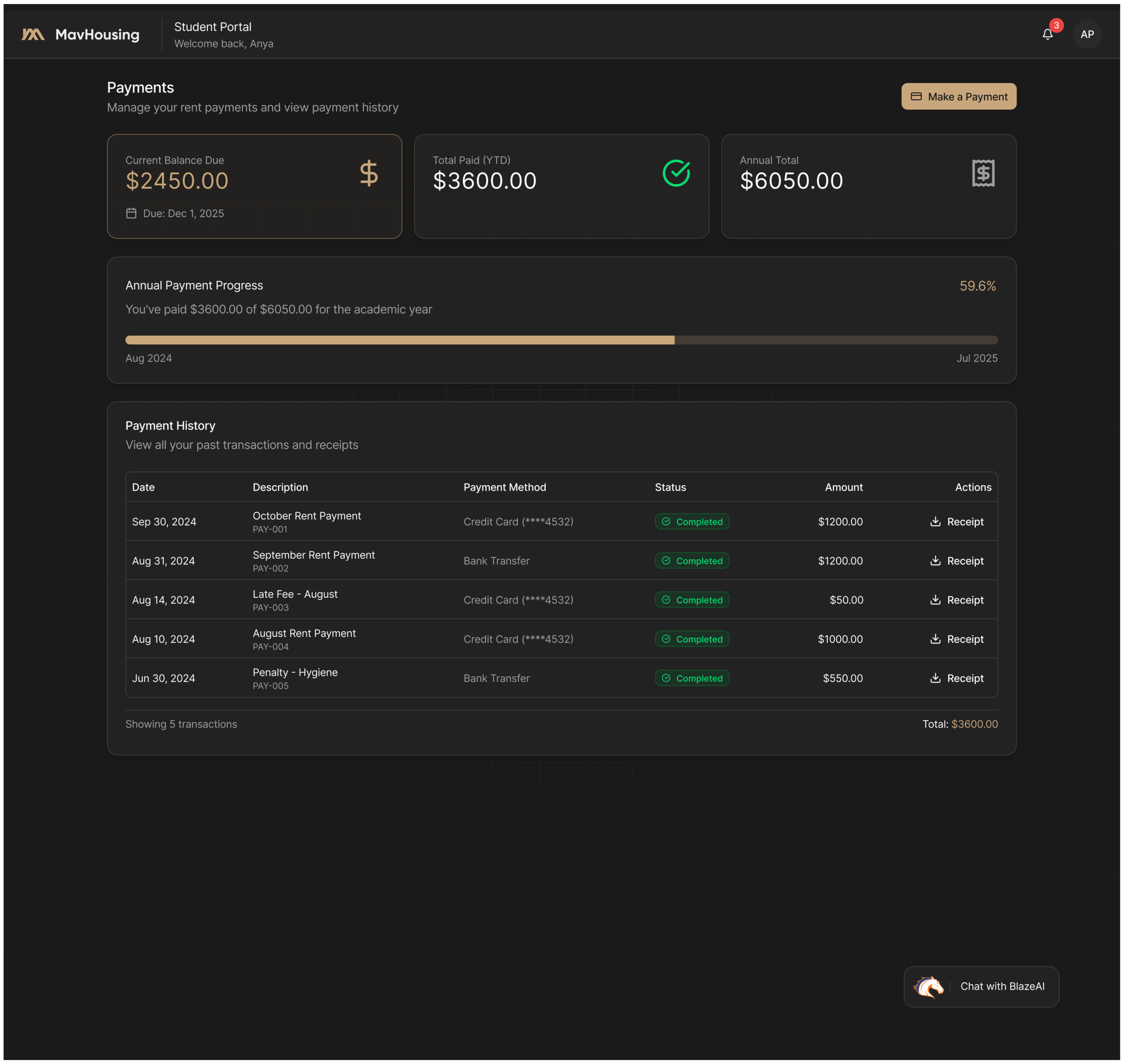
MR-003 · Arlington Hall 317 - Living Room

Window lock in living room is broken and won't secure properly.

Submitted: 10/30/2024

Security

Figure 4: Maintenance Request Page



MavHousing | Student Portal
Welcome back, Anya

Payments
Manage your rent payments and view payment history

Make a Payment

Current Balance Due
$2450.00
Due: Dec 1, 2025

Total Paid (YTD)
$3600.00

Annual Total
$6050.00

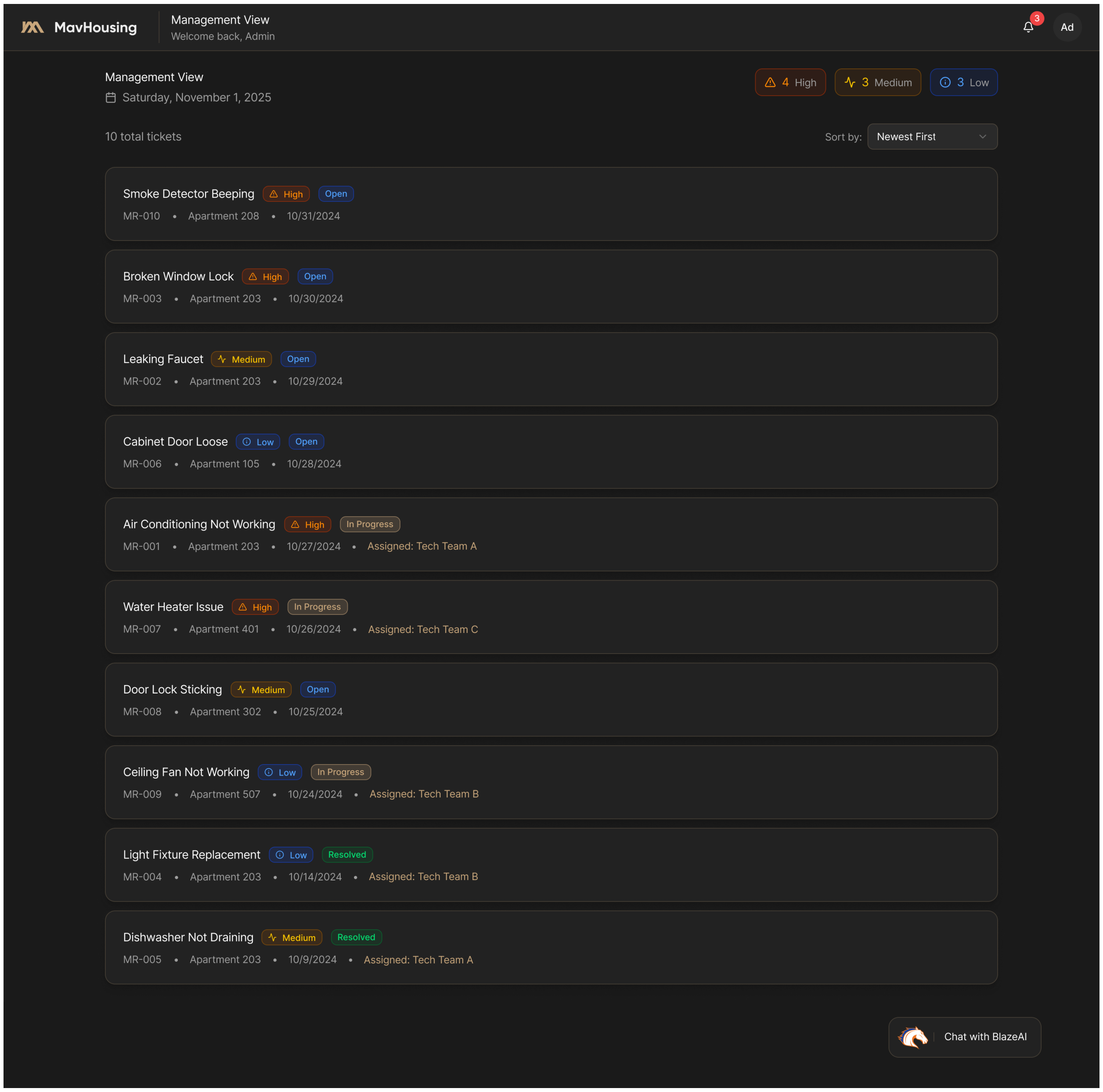
Annual Payment Progress
You've paid $3600.00 of $6050.00 for the academic year
59.6%
Aug 2024 — Jul 2025

Payment History
View all your past transactions and receipts

Date	Description	Payment Method	Status	Amount	Actions
Sep 30, 2024	October Rent Payment PAY-001	Credit Card (****4532)	Completed	$1200.00	Receipt
Aug 31, 2024	September Rent Payment PAY-002	Bank Transfer	Completed	$1200.00	Receipt
Aug 14, 2024	Late Fee - August PAY-003	Credit Card (****4532)	Completed	$50.00	Receipt
Aug 10, 2024	August Rent Payment PAY-004	Credit Card (****4532)	Completed	$1000.00	Receipt
Jun 30, 2024	Penalty - Hygiene PAY-005	Bank Transfer	Completed	$550.00	Receipt

Showing 5 transactions — Total: $3600.00

Figure 5: Payment Portal



Management View

Saturday, November 1, 2025

4 High | 3 Medium | 3 Low

10 total tickets

Sort by: Newest First

Smoke Detector Beeping High Open
MR-010 • Apartment 208 • 10/31/2024

Broken Window Lock High Open
MR-003 • Apartment 203 • 10/30/2024

Leaking Faucet Medium Open
MR-002 • Apartment 203 • 10/29/2024

Cabinet Door Loose Low Open
MR-006 • Apartment 105 • 10/28/2024

Air Conditioning Not Working High In Progress
MR-001 • Apartment 203 • 10/27/2024 • Assigned: Tech Team A

Water Heater Issue High In Progress
MR-007 • Apartment 401 • 10/26/2024 • Assigned: Tech Team C

Door Lock Sticking Medium Open
MR-008 • Apartment 302 • 10/25/2024

Ceiling Fan Not Working Low In Progress
MR-009 • Apartment 507 • 10/24/2024 • Assigned: Tech Team B

Light Fixture Replacement Low Resolved
MR-004 • Apartment 203 • 10/14/2024 • Assigned: Tech Team B

Dishwasher Not Draining Medium Resolved
MR-005 • Apartment 203 • 10/9/2024 • Assigned: Tech Team A

Figure 6: Management Dashboard

3 Customer Requirements

MavHousing is a web-based housing management system designed to provide students, housing staff, and administrators with an intuitive and visually consistent interface to manage applications, payments, maintenance requests, and communication. Customer requirements define the expected functionality, usability, and visual design of the system, ensuring that all end-users can efficiently complete tasks while easily understanding the status of requests and actions. Each requirement addresses a specific user need and reflects the look and feel of the product as defined in the Figma mockups.

3.1 User Interface

3.1.1 Description

The MavHousing interface will follow a consistent color scheme to enhance clarity and indicate item status. The primary background will be black, completed actions or positive status will be shown in green, pending or waiting items will be highlighted in yellow, and logo and branding elements will use a gold color (C9A77C). Status indicators will be applied throughout dashboards, maintenance request lists, and payment histories to provide immediate visual feedback. The interface is responsive, designed for quick navigation across all major functions, including the student portal, maintenance request management, payment interface, and AI chatbot interaction.

3.1.2 Source

CSE Senior Design Project specifications and end-user requirements.

3.1.3 Constraints

The web application must support multiple users interacting with the system simultaneously, including submitting housing applications, making payments, and requesting maintenance. It must maintain responsive performance, ensure data integrity, and comply with UTA security and privacy standards.

3.1.4 Standards

The application should comply with ISO 9241-110:2020, which sets ergonomic requirements for human-centered design of interactive systems. This ensures that dashboards, menus, and navigation are intuitive and accessible.

3.1.5 Priority

High.

3.2 Maintenance Request System

3.2.1 Description

The MavHousing system will allow students to submit maintenance requests for their housing units, specifying issue type, location, description, and optional attachments. Housing staff and maintenance personnel can view, update, and resolve requests, with status indicators showing progress: green for completed, yellow for pending. Notifications will be sent to users at key stages to ensure transparency. Administrators will have a management view listing all requests, including status, urgency, and assigned staff, enabling efficient tracking and prioritization.

3.2.2 Source

CSE Senior Design Project specifications and end-user requirements.

3.2.3 Constraints

The module must support multiple simultaneous requests without performance degradation. Status updates must be reflected in real-time across dashboards and management views. Attachments must comply with file type and size limits. All actions must adhere to UTA security and privacy standards.

3.2.4 Standards

The maintenance request module should comply with ISO 9241-110:2020 for human-centered design, ensuring that students can submit requests and staff can manage them efficiently and intuitively.

3.2.5 Priority

Critical.

3.3 Communication and Notifications

3.3.1 Description

MavHousing provides in-app messaging and email notifications. Students and staff receive updates about applications, maintenance requests, payments, and announcements. Notifications help keep all users informed and improve response time.

3.3.2 Source

CSE Senior Design Project specifications and end-user requirements.

3.3.3 Constraints

Notifications must be reliable and delivered in real-time. Messaging and alerts must comply with UTA privacy and security standards.

3.3.4 Standards

The communication module should comply with ISO 9241-110:2020 for usability and secure messaging guidelines to ensure privacy and data protection.

3.3.5 Priority

High.

3.4 Payment Management

3.4.1 Description

Students can view payment history, make payments, and update lease information. Payments are securely processed, recorded accurately, and confirmations are sent automatically. Administrators can monitor transactions and generate reports for accounting purposes.

3.4.2 Source

CSE Senior Design Project specifications and end-user requirements.

3.4.3 Constraints

High traffic may affect processing speed temporarily. All payment transactions must follow security and privacy rules, including encryption and access control.

3.4.4 Standards

The payment module should comply with ISO/IEC 25010 for software quality and PCI DSS standards for secure payment processing.

3.4.5 Priority

Critical.

4 Packaging Requirements

MavHousing will be accessible as a web-based application through modern web browsers. The system is designed to be responsive and usable on a variety of devices, including desktops, laptops, tablets, and smartphones. While it could potentially be adapted into a mobile app in the future, the current project focuses exclusively on the web application version.

4.1 Accessible Through Web Browser

4.1.1 Description

Students, housing staff, and administrators should be able to access MavHousing via any modern web browser by entering the application URL. The web app will provide full functionality, including housing applications, maintenance request submissions, payment management, and AI chatbot interaction, without requiring additional installations or plugins.

4.1.2 Source

CSE Senior Design project specifications, intended end-user requirements (UTA students, housing staff, and administrators), and the need for ease of access across devices.

4.1.3 Constraints

The web app requires an active internet connection. Functionality may vary slightly depending on browser version or device screen size. Offline access is not provided in the current version.

4.1.4 Standards

MavHousing will comply with:

- W3C standards for web accessibility and cross-browser compatibility.
- ISO 9241-110:2020 for ergonomic, human-centered interactive system design.
- WCAG 2.1 accessibility guidelines to ensure usability for all users, including those with disabilities.
- ECMA-262 (JavaScript) standard for scripting and client-side functionality.

4.1.5 Priority

Critical.

5 Performance Requirements

MavHousing must operate smoothly and reliably for students, housing staff, and administrators. Core operations including login, dashboard loading, viewing applications, submitting maintenance requests, managing payments, and using the Blaze AI chatbot must perform efficiently while maintaining data accuracy and system stability. The system should support multiple concurrent users, especially during peak periods like move-in or payment deadlines, without noticeable delay or data loss. All critical functions should complete within acceptable timeframes to provide a responsive and user-friendly experience.

5.1 System Initialization and Dashboard Load

5.1.1 Description

The system should be fully operational, and dashboards fully loaded, within 5-7 seconds after user login. All critical services, including database connections, authentication, payment APIs, and BlazeAI, must initialize quickly.

5.1.2 Source

CSE Senior Design project specifications and end-user convenience.

5.1.3 Constraints

Network latency, high server load, or backend service delays may slightly increase initialization time.

5.1.4 Standards

ISO/IEC 25010 (performance efficiency, reliability), ISO/IEC 25012 (data integrity), ISO 9241-11 (usability), W3C Performance Timing API.

5.1.5 Priority

High

5.2 Login and Authentication

5.2.1 Description

All users (students, staff, administrators) should log in and authenticate within 5-7 seconds. Credentials must be securely validated, session tokens issued, and multiple simultaneous logins handled reliably.

5.2.2 Source

CSE Senior Design project specifications and end-user convenience.

5.2.3 Constraints

High server load or slow network connections may increase login time.

5.2.4 Standards

ISO/IEC 25010, ISO/IEC 25012, ISO 9241-11, W3C Performance Timing API.

5.2.5 Priority

High

5.3 Maintenance Request Management

5.3.1 Description

Students can submit maintenance requests, and housing staff can view and resolve them. Status updates (pending, completed) should appear on dashboards and management views within 5-10 seconds. Multiple simultaneous requests should not compromise accuracy or responsiveness.

5.3.2 Source

CSE Senior Design project specifications and end-user needs.

5.3.3 Constraints

Large attachments or high simultaneous submissions may temporarily increase response time.

5.3.4 Standards

ISO/IEC 25010, ISO/IEC 25012, ISO 9241-11.

5.3.5 Priority

Critical

5.4 Payment and Lease Management

5.4.1 Description

Users should view payment history, submit rent or deposit payments, and update lease information within 5-7 seconds per action. Transactions must be securely processed, accurately recorded, and confirmations delivered promptly.

5.4.2 Source

CSE Senior Design project specifications and student/staff convenience.

5.4.3 Constraints

High server load or slow network may slightly increase processing time.

5.4.4 Standards

ISO/IEC 25010, ISO/IEC 25012, ISO 9241-11, PCI DSS compliance for payments.

5.4.5 Priority

Critical

5.5 BlazeAI Chat Response

5.5.1 Description

BlazeAI must respond to queries regarding housing applications, maintenance requests, or payments within 10 seconds. Multiple simultaneous queries should be handled efficiently, preserving chat history and ensuring no message loss.

5.5.2 Source

CSE Senior Design project specifications and user convenience.

5.5.3 Constraints

High simultaneous queries or complex requests may slightly delay responses.

5.5.4 Standards

ISO/IEC 25010, ISO/IEC 25012, ISO 9241-11, W3C WebSocket standards.

5.5.5 Priority

High

5.6 Data Accuracy and Integrity

5.6.1 Description

All user inputs, including registration data, housing applications, maintenance requests, and paymentsâ- must be accurately recorded and retrievable. Database updates must be atomic, and any operation should complete within 2-5 seconds under normal conditions to prevent inconsistencies.

5.6.2 Source

CSE Senior Design project specifications.

5.6.3 Constraints

Database downtime or network failures may temporarily affect data submission.

5.6.4 Standards

ISO/IEC 25012, SQL ACID compliance, ISO/IEC 25010 (reliability).

5.6.5 Priority

Critical

6 Safety Requirements

Although MavHousing is a software-based web application with no physical hardware components, it requires strong emphasis on information safety, data security, and system reliability. Safety in the context of this product primarily involves protecting sensitive student data from unauthorized access, preventing security breaches, ensuring compliance with educational privacy laws, and maintaining system integrity to avoid data loss or corruption that could harm users' housing status, financial records, or personal information.

6.1 Data Security and Privacy

6.1.1 Description

All user data including student personal information, housing applications, payment records, maintenance requests, and communication logs must be securely stored and transmitted. Data must be encrypted both at rest (in Google Cloud SQL) and in transit (via HTTPS). The system must implement role-based access controls ensuring users can only view data they are authorized to access. Student records must be handled in accordance with FERPA guidelines, and all team members must follow UTA's data protection policies during development and testing.

6.1.2 Source

FERPA requirements, UTA data protection policies, CSE Senior Design project specifications

6.1.3 Constraints

All sensitive data must be stored in Google Cloud Platform's secure database services with encryption enabled by default. The development team must use environment variables for all API keys and credentials, never hardcoding them in source code. Access to the production database will be limited to designated team members only. All code containing security implementations must be reviewed by at least one other team member before merging. Student test data must be anonymized and cannot contain real personal information.

6.1.4 Standards

Family Educational Rights and Privacy Act (FERPA), ISO/IEC 27001 for information security management, CSE Senior Design project specifications

6.1.5 Priority

Critical

6.2 Secure Access and Authentication

6.2.1 Description

The system must implement secure user authentication system. Role-based access control (RBAC) must restrict students, housing staff, and administrators to features appropriate for their role. Session management must include automatic logout after 30 minutes of inactivity. Passwords, if stored locally for testing, must be hashed using bcrypt with a minimum of 10 salt rounds. The system must protect against common attacks including brute force login attempts and session hijacking through rate limiting and secure token management.

6.2.2 Source

OWASP authentication guidelines, CSE Senior Design project specifications

6.2.3 Constraints

The system may or may not be able to use UTA's existing SSO system, limiting our ability to customize the authentication flow. Session tokens must expire after 30 minutes of inactivity to balance security and user convenience. Rate limiting for login attempts will be set to 5 failed attempts per 15 minutes to prevent brute force attacks without overly restricting legitimate users. The team must use established authentication libraries (Passport.js, JWT) rather than building custom authentication from scratch. All authentication-related code must be stored in a separate, access-controlled repository branch.

6.2.4 Standards

OAuth 2.0, bcrypt hashing standard, CSE Senior Design project specifications

6.2.5 Priority

Critical

6.3 System Integrity and Reliability

6.3.1 Description

The MavHousing platform must operate reliably under typical usage conditions and include error handling to prevent data loss or system crashes. All database operations must use transactions to maintain data consistency, with automatic rollback on errors. The system must implement input validation on both frontend and backend to prevent invalid data from corrupting the database. Automated daily backups must be configured using Google Cloud's backup features. Basic monitoring and logging must be implemented to help the team identify and troubleshoot issues during development and after deployment.

6.3.2 Source

Internal quality assurance guidelines, CSE Senior Design project specifications, Google Cloud Platform best practices

6.3.3 Constraints

Database backups will be scheduled during low-traffic hours (2-4 AM) using Google Cloud's automated backup features. The team has limited cloud budget ($800 from CSE department), so backup retention will be limited to 7 days of daily backups. Error logs must be reviewed weekly by the designated QA team member. Testing must be performed on a separate staging database to avoid corruption of development data. The system must gracefully handle GCP service outages by displaying clear error messages rather than crashing, though full functionality cannot be maintained during cloud provider downtime.

6.3.4 Standards

ISO/IEC 25010 for software quality and reliability, Google Cloud Operations logging standards

6.3.5 Priority

High

7 Maintenance & Support Requirements

Following deployment of the MavHousing system, ongoing maintenance and support will be essential to ensure continued functionality, address bugs, implement updates, and assist users with technical issues. These requirements define the documentation, tools, access credentials, and procedures necessary for future developers or UTA IT staff to maintain the system after the original development team graduates. Maintenance activities include fixing software defects, updating dependencies, managing cloud infrastructure, monitoring system performance, and providing user support. Clear documentation and accessible development environments will enable smooth handoff and long-term sustainability of the platform.

7.1 Source Code Documentation and Repository Access

7.1.1 Description

All system maintainers must have access to comprehensive source code documentation and the complete codebase stored in a version-controlled repository. This includes the frontend, backend, database schemas, API documentation, deployment scripts, and configuration files. Code must include inline comments explaining complex logic, and each module should have README files describing its purpose, dependencies, and usage. The repository must maintain a complete commit history showing the evolution of the codebase. API endpoints must be documented using Swagger/OpenAPI specifications, and database schemas must include entity relationship diagrams (ERDs).

7.1.2 Source

Internal development team standards, CSE Senior Design documentation requirements, software maintenance best practices

7.1.3 Constraints

The source code repository will be hosted on GitHub with access limited to authorized UTA personnel and future development teams. Documentation must be written in English and follow consistent formatting standards (Markdown for README files, JSDoc for JavaScript/TypeScript code). All sensitive credentials, API keys, and configuration secrets must be excluded from the repository and stored separately in environment variable documentation. The documentation must be updated whenever significant changes are made to the codebase, with the final comprehensive update completed before project handoff in April 2026.

7.1.4 Standards

Git version control best practices, JSDoc documentation standard for JavaScript/TypeScript, OpenAPI 3.0 specification for REST APIs, Markdown formatting standards, IEEE 1063 for software user documentation

7.1.5 Priority

Critical

7.2 Deployment and Environment Setup Documentation

7.2.1 Description

Maintainers must have access to detailed documentation describing how to set up development, staging, and production environments for MavHousing. This includes step-by-step instructions for installing required dependencies, configuring Google Cloud Platform services, setting up environment variables, initializing databases with schema migrations, and deploying updates. Documentation must cover both local development setup for testing changes and production deployment procedures. Troubleshooting guides should address common setup issues such as database connection failures, missing dependencies, or GCP authentication problems.

7.2.2 Source

Team deployment procedures, GCP documentation, internal setup guides

7.2.3 Constraints

Environment setup requires Google Cloud Platform account access with appropriate billing and permissions configured by UTA IT administration. Database migrations must be run in the correct sequence to avoid schema conflicts. The team will provide a setup script to automate environment configuration where possible, but some manual steps may be required. Complete environment setup should take no more than 2-4 hours for an experienced developer following the documentation.

7.2.4 Standards

Google Cloud Platform best practices, Docker containerization guidelines

7.2.5 Priority

High

7.3 User Support and Troubleshooting Guides

7.3.1 Description

The maintenance package must include user-facing support documentation and technical troubleshooting guides. User guides should explain how to perform common tasks such as submitting housing applications, requesting maintenance, making payments, and using the BlazeAI assistant. Troubleshooting guides for administrators should cover common issues like failed payment processing, maintenance request workflow problems, application submission errors, and database query performance issues. Documentation should include screenshots, step-by-step instructions, and solutions to frequently encountered problems. An FAQ section should address the most common user questions to reduce support burden.

7.3.2 Source

User feedback during testing, anticipated support scenarios

7.3.3 Constraints

Support documentation must be accessible in PDF format and as a searchable web page within the MavHousing portal. Guides must be written in clear, non-technical language for student users, while technical troubleshooting content can assume basic IT knowledge for housing staff and administrators. Screenshots and diagrams should be kept up-to-date when UI changes are made. The team will create initial documentation based on prototype testing, but guides may need updates as real usage patterns emerge after deployment.

7.3.4 Standards

IEEE 1063 for software user documentation, ISO 9126-1 for usability standards

7.3.5 Priority

High

8 Other Requirements

The MavHousing system must include a few more features to make sure it works well, can be updated easily, and can be used on different setups. These requirements cover how customers set up the system, how the system can grow in the future, and how it can run smoothly on many platforms. They make the product easier to manage, flexible, and ready for future updates.

8.1 Customer Setup and Configuration

8.1.1 Description

MavHousing must offer an easy setup process for housing administrators. Admins should be able to adjust settings like user roles, building data, and room details through a clear setup page or guide. The system should include short setup instructions and visual steps to help first-time users.

8.1.2 Source

CSE Senior Design project guidelines and housing staff feedback.

8.1.3 Constraints

Setup should stay simple but secure. Admins should be able to configure everything without needing programming skills. All stored credentials must follow university privacy policies.

8.1.4 Standards

ISO/IEC 27002 for security setup and configuration.

8.1.5 Priority

High

8.2 System Extensibility

8.2.1 Description

The MavHousing system should be built so new features can be added later. This means the design must be modular, and the APIs must allow smooth changes without breaking old features. Future updates could include more payment methods, new report tools, or other campus system links.

8.2.2 Source

CSE Senior Design project guidelines and Roadmap

8.2.3 Constraints

New updates must not affect how the system works now. All changes should be tested in separate modules before being added.

8.2.4 Standards

IEEE 1471 and RESTful API design practices.

8.2.5 Priority

Future

8.3 Source Code Portability

8.3.1 Description

MavHousings code should work on different platforms like Windows, macOS, and Linux. This helps the team test and deploy the system easily in different environments such as local servers or the cloud.

8.3.2 Source

CSE Senior Design project guidelines

8.3.3 Constraints

Some libraries or tools might behave differently on each platform. The team should use cross-platform tools and container systems to avoid issues.

8.3.4 Standards

CSE Senior Design project guidelines

8.3.5 Priority

High

8.4 Modularity for Component-Based Design

8.4.1 Description

MavHousing should be built using small, separate parts (modules) for each main feature such as login, payments, maintenance, and BlazeAI chat. This makes fixing or improving one part easier without affecting others.

8.4.2 Source

Team coding and design standards.

8.4.3 Constraints

Each module must stay independent and communicate through defined interfaces only. Too many cross-links between modules should be avoided.

8.4.4 Standards

SOLID design principles and IEEE 1471 for modular architecture.

8.4.5 Priority

High

9 Future Items

This section restates all requirements marked as Priority 5 (Future). These features were discussed but will not be included in the MavHousing prototype due to time and resource limitations. They are planned for future implementation.

9.1 System Extensibility

9.1.1 Description

The MavHousing system should be designed so new features can be added in the future. This includes having a modular design and APIs that support updates like new payment methods, advanced reports, or integration with other campus systems.

9.1.2 Source

Team development plan and architecture roadmap.

9.1.3 Constraints

Any new updates must not interrupt the current system. All changes should be developed and tested separately before being added to the main system.

9.1.4 Standards

IEEE 1471 and RESTful API design practices.

9.1.5 Priority

Future

References