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

Project Charter
CSE 4316: Senior Design I
Fall 2025

Builders Squad
MavHousing

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

Revision History

Revision	Date	Author(s)	Description
0.1	09.25.2025	AJ	Initial document setup and outline
0.2	09.28.2025	AJ, AK, AS	Added core project details and initial draft content
0.3	09.30.2025	AK, AR, AS	Expanded main sections and improved structure
0.4	10.02.2025	AJ, MS, AK, TT, AR	Revised layout, refined writing, and peer reviewed
1.0	10.7.2025	AJ, AK, AS, MS, TT	Finalized and approved for submission

CONTENTS

List of Figures

1 Problem Statement

Many students at UT Arlington face unnecessary barriers and frustration due to fragmented and outdated manual systems for managing on-campus housing, room assignments, communication, and maintenance requests. These issues can lead to delayed applications, lack of real-time updates, inefficient maintenance, and increased stress or dissatisfaction among residents and staff. Without a modern, unified solution, the university housing experience remains impersonal, slow, and needlessly complex. Solving this problem would not only streamline processes and boost operational efficiency, but would also create a more responsive, transparent, and supportive residential environment where students and staff feel informed, heard, and empowered to succeed academically and socially.

2 Methodology

To address the inefficiencies in UTA's current housing management system, we will develop a full-stack web application that centralizes and automates student housing operations, providing an intuitive interface for managing housing applications, room assignments, maintenance requests, payment processing, and communication between students and housing staff. The system will implement role-based access control for various entities, ensuring each user interacts with features appropriate to their responsibilities. A key innovation is the integration of an AI assistant with a Model Context Protocol (MCP) server, enabling context-aware natural language interactions that query the production database as the source of truth, accessible both within the web portal and through automated notification channels.

3 Value Proposition

This project delivers significant value to the University of Texas at Arlington by modernizing critical housing operations that directly impact thousands of students annually. For housing administrators and staff, the automated system will substantially reduce time spent on repetitive tasks such as processing applications, coordinating room assignments, and responding to routine inquiries, allowing them to redirect efforts toward complex student needs and strategic facility planning. The centralized platform eliminates current data fragmentation across multiple systems, providing real-time analytics on occupancy rates, maintenance backlogs, and application trends that enable informed resource allocation decisions. For students, the system transforms the housing experience by providing 24/7 access to information, transparent tracking of applications and maintenance requests, and immediate responses to common questions through the AI assistant, particularly during high-demand periods like application season and move-in when staff availability is most constrained. The platform's modular architecture ensures long-term value by enabling future expansion to additional residence life functions or integration with other university systems such as student information databases and payment processing services.

4 Development Milestones

This list of core project milestones should include all major documents, demonstration of major project features, and associated deadlines. Any date that has not yet been officially scheduled at the time of preparing this document may be listed by month.

Provide a list of milestones and completion dates in the following format:

- Project Charter first draft - October 2025
- System Requirements Specification - October 2025
- Architectural Design Specification - November 2025
- Demonstration of Database Schema and API Framework - November 2025

- Demonstration of User Authentication and Role-Based Access - December 2025
- Detailed Design Specification - December 2025
- Demonstration of Housing Application System - January 2026
- Demonstration of Room Assignment Algorithm - February 2026
- Demonstration of Maintenance Request Management - February 2026
- CoE Innovation Day poster presentation - unknown
- Demonstration of AI Assistant Integration - March 2026
- Demonstration of Microsoft Teams Bot Integration - April 2026
- Demonstration of Full System Integration - April 2026
- Final Project Demonstration - April 2026

5 BACKGROUND

The University of Texas at Arlington's current housing management system suffers from significant inefficiencies that impact both operational effectiveness and student satisfaction. Housing operations rely on a fragmented combination of outdated software, manual paper-based processes, and disparate databases that do not communicate with each other. This patchwork approach creates substantial administrative burden for housing staff who must manually coordinate room assignments, track maintenance requests across multiple platforms, and respond to routine student inquiries repeatedly throughout the academic year. During peak periods such as application season and move-in, the housing office becomes overwhelmed with requests, leading to delayed responses, assignment errors, and frustrated students who cannot access basic information about their housing status outside of business hours. Students face numerous pain points: the housing application process requires navigating multiple web portals with limited visibility into application status, maintenance issues lack real-time tracking or notification capabilities, and simple questions about payment deadlines or roommate information require contacting staff directly, creating bottlenecks that could be easily resolved through automated information access.

From an operational perspective, the current system prevents data-driven decision making. Housing administrators lack comprehensive analytics on occupancy trends, maintenance response times, or application patterns because data exists in silos across incompatible systems. Room assignment decisions rely heavily on manual processes that are time-intensive and prone to human error, particularly when accommodating student preferences for roommates, building locations, or amenities. The inability to quickly generate reports on facility maintenance needs or predict occupancy hampers strategic planning and resource allocation. As peer institutions invest in digital transformation of campus services, UTA's outdated housing infrastructure becomes increasingly apparent to prospective students who expect seamless digital experiences comparable to consumer applications they use daily.

Our development team approaches this project with strong alignment to UTA's strategic priorities in student success and operational excellence. The project addresses a documented need within the university community and provides an opportunity to demonstrate how modern software engineering practicesâincluding full-stack development, database architecture, and artificial intelligence integrationâcan solve real-world operational challenges. The system we propose will serve as a proof of concept that could be adopted by UTAś housing department or serve as a model for similar residential life management systems at other institutions, creating tangible value for the university while providing practical experience building enterprise-level applications that handle sensitive student data, support multiple user roles, and integrate with existing university infrastructure.

6 RELATED WORK

Modern housing management systems (HMS) have become essential for landlords, housing offices, and universities to manage leases, payments, maintenance, and communication through a single platform. These systems reduce manual paperwork, prevent errors, and improve the experience for both residents and administrators.

One example is Avail, designed for small and mid-size landlords. It offers online applications, rent collection, lease signing, and maintenance tracking in an affordable and user-friendly package [1]. However, Avail focuses on basic property management and lacks the advanced tools needed for more complex operations such as student housing or multi-unit institutions. Larger systems such as AppFolio, Buildium, and Yardi Breeze serve professional property managers with advanced features like account-

ing, reporting dashboards, and automated workflows [2–4]. While powerful, they come with higher subscription costs, setup requirements, and technical overhead that make them impractical for smaller organizations or campus housing offices.

In the student housing sector, platforms such as Entrata Student and StarRez (used by UTA) handle student-specific needs like online housing applications, roommate matching, contract management, and large-scale move-ins and move-outs [5, 6]. These systems are well integrated with university login and payment systems, showing how housing management technology has evolved from simple rental tools to specialized campus-wide platforms.

Despite these advancements, current systems have several limitations. Many commercial platforms are too expensive, charging per-unit fees or high monthly minimums [4]. Others are too generic, lacking features for campus housing operations that run on academic calendars or require complex room assignments [5, 6]. Integration is also a challenge as platforms often work best with their own ecosystems, making it difficult to connect to external systems such as student information or financial databases [2, 4]. Simpler systems like Avail are affordable but lack customization and robust data management features [1].

Our proposed housing management solution bridges this gap by combining the ease of use and affordability of systems like Avail with the campus-ready features of StarRez and Entrata. It emphasizes modular design, open integration, and adaptable workflows, providing a flexible and cost-effective system built specifically for universities and small housing organizations that need more control and reliability than existing commercial solutions offer.

7 System Overview

The systems architecture is built in layers, with each layer handling a specific set of tasks to keep the system organized and efficient. The top layer includes different modules that handle core functions. The Authentication & Role-Based Access module manages logins and permissions, ensuring students, staff, and administrators can only access what theyre supposed to. The Housing Management Module organizes housing applications, room assignments, and occupancy data, while the Maintenance Management Module keeps track of repair requests and updates their status as staff work on them. The Communication & Notifications Module automatically sends reminders, announcements, and progress updates to students and staff. The AI Integration (MCP Server) connects the AI assistant to the main database of the system so it can answer questions and provide information instantly.

Beneath these modules is the main application server layer, which handles all the logic and communication between the user interfaces and the database. It processes data requests, enforces security, and ensures smooth coordination among the different modules. The database layer stores all the housing information, user profiles, maintenance logs, and communication records in one central location. This design makes sure that all parts of the system use the same up-to-date data. The layered structure promotes separation of concerns, meaning each layer can be updated or scaled independently without affecting the others. It also improves performance and reliability by clearly defining how data flows between users, the application, and external systems.

Finally, the system connects to several external university services such as the student information system, payment gateway, and Microsoft Teams for notifications. These connections allow data to move

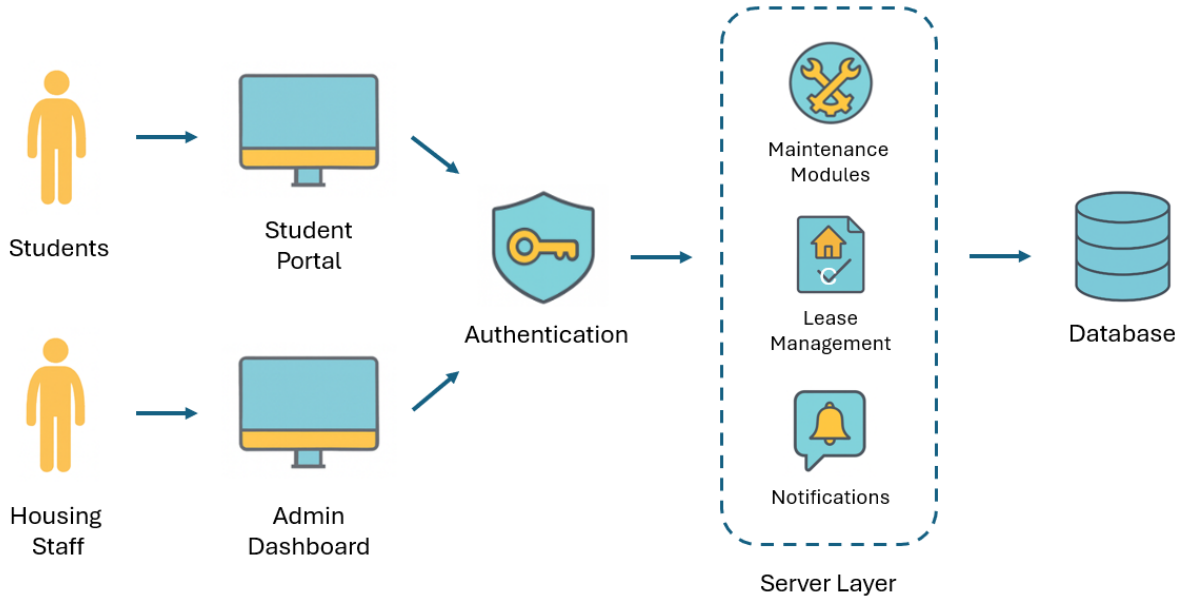

Figure 1: System Architecture Overview

smoothly between systems without duplicate entry. Altogether, this layered structure keeps the system secure, organized, and easy to maintain while allowing it to scale and integrate new features in the future.

8 Roles & Responsibilities

8.1 Stakeholders

The primary stakeholders of this project include the course instructor, Dr Chris Thomas Conly, who provides academic guidance and project oversight, and the Computer Science and Engineering department at the University of Texas at Arlington. As this is a senior design project, the potential end users include UTA housing administration, students seeking on-campus housing, maintenance staff, and residential advisors, though formal sponsorship from the housing department has not been established at this time.

8.2 Point of Contact

The point of contact from the sponsor or customer side is **Alok Jha**, Product Manager, who will coordinate communication between the development team and university stakeholders, gather requirements, and facilitate access to necessary resources and feedback from housing staff and students.

8.3 Team Members and Areas of Responsibility

The development team consists of six members, each contributing their expertise to ensure successful project delivery:

- **Alok Jha** - Point of Contact, MCP Server Development, Backend Development, Database Design
- **Aastha Khatri** - Frontend Development, UI/UX Design, User Interface Implementation

- **Talha Tahmid** - Frontend Development, Component Design, Client-Side Integration
- **Atiqur Rahman** - Backend Development, API Design, Authentication Services
- **Aviral Saxena** - Backend Development, Worker Queue, Integration Layer
- **Md Rashidul Alam Sami** - Quality Assurance, Testing, Documentation, Deployment

8.4 Product Owner and Scrum Master

The roles of Product Owner and Scrum Master will rotate among team members on a sprint-by-sprint basis, aligning with Agile principles of shared ownership and collaborative team dynamics. This rotation ensures shared leadership and comprehensive understanding of project management responsibilities across the team. The approach allows each member to develop skills in stakeholder communication, backlog prioritization, and team facilitation while maintaining continuity through structured handoffs at the end of each sprint.

9 Cost Proposal

This section contains the approximate budget for the project, where that money will come from, and any other support. Major expenses are mainly related to cloud services and materials for documentation and testing. Personal laptops and existing team resources will be used for development, so costs are minimized. The primary focus is to support the full-stack web application hosting, testing, and project deliverables.

9.1 Preliminary Budget

Item	Description
Cloud Services	GCP charges for backend, storage, and compute resources to support the web application
Documentation & Materials	Printing, posters, and presentation materials for project deliverables and demonstrations
Miscellaneous	Small expenses for demos, testing, or project-related tools

9.2 Current & Pending Support

The CSE Department provides approximately $800 in default funding for student projects, which will cover a portion of cloud hosting and materials costs. There are no external sponsors or grants for this project. The team will manage all expenses carefully, prioritizing essential items such as GCP cloud services, documentation, and testing materials.

10 Facilities & Equipment

The development of MavHousing will primarily rely on team members' personal laptops and other computer devices, which provide sufficient computing power for full-stack development, coding, and testing. No specialized laboratory space is required beyond standard computer access, as the project is a web-based application. The CSE department's computer labs may also be used as needed for collaborative work or testing on multiple devices to simulate real-world scenarios.

For hosting, deployment, and backend testing, the project will utilize GCP cloud services. GCP provides scalable servers, storage, and computing resources that allow the team to simulate real user interactions and test application performance without requiring on-site physical servers. This also supports secure data storage and enables collaboration among team members working remotely.

Testing and demonstrations of the application will be conducted on both personal devices and shared computers. Presentation materials, including posters, documentation, and demonstration setups, will be prepared using existing university resources that is the library. No additional hardware, specialized equipment, or outsourced resources are required, making the project fully feasible with current facilities and accessible technology.

11 Assumptions

The following list contains critical assumptions related to the implementation and testing of the project.

- All team members will have reliable access to laptops, IDEs, and internet connectivity throughout the project.
- Google Cloud Platform services (Compute Engine, Cloud SQL, Cloud Storage) will remain accessible and stable for hosting the backend, database, and static assets.
- Required APIs, libraries, and frameworks will remain compatible and available for integration with the project.
- Feedback from students and housing staff will be timely to validate functionality and usability.
- The version control system (GitHub) will operate reliably for collaborative development and tracking code changes.
- Development milestones can be completed as scheduled assuming no critical security or integration issues arise.
- We will have enough dataset about students to train models for matching the roommates and their apartment units.

12 Constraints

The following list contains key constraints related to the implementation and testing of the project.

- The final prototype demonstration must be completed by April 2026 to meet senior design project deadlines.
- Development must replicate the essential functionality of UTA's current housing portal, including applications, room assignments, and maintenance requests.
- Integration with Microsoft Teams via Blaze is limited to functionality supported by the Teams API and available access credentials.
- Cloud services (GCP) will be used for hosting and databases, and any outages or limits imposed by GCP could impact testing and deployment.
- All data and functionality must comply with UTA's security and privacy requirements, limiting what information can be used for testing and demonstration.
- Team access to UTA systems or data may be restricted, requiring the project to rely primarily on simulated or anonymized data.
- UTA may not provide the actual data from the students so we might have to create mock data for testing and simulating the applications.

13 Risks

The following high-level risk census contains identified project risks with the highest exposure. Mitigation strategies will be discussed in future planning sessions.

Risk description	Probability	Loss (days)	Exposure (days)
GCP service outage affecting backend or database	0.20	7	1.4
Delays in implementing AI assistant (Blaze) features	0.30	14	4.2
Critical bugs in room assignment or application processing modules	0.25	10	2.5
Delays in receiving feedback from students or housing staff	0.35	7	2.45
Compatibility issues with APIs, libraries, or frameworks	0.20	5	1.0
Team member availability or scheduling conflicts	0.15	5	0.75

Table 1: Overview of highest exposure project risks for MavHousing

14 Documentation & Reporting

14.1 Major Documentation Deliverables

14.1.1 Project Charter

The Project Charter will be maintained collaboratively in Overleaf as a shared LaTeX document accessible to all team members. It will be updated only when significant changes occur, such as modifications to the project scope, the addition of new requirements, or the completion of major milestones. The team will review the document during each Friday sprint retrospective to ensure continued alignment with the projectâs current status. The initial and final version will be delivered on October 10, 2025.

14.1.2 System Requirements Specification

The System Requirements Specification will be maintained in Overleaf and updated as new functional or non-functional requirements are identified through stakeholder feedback, testing results, or technical constraints. The team will review the document every Friday during sprint retrospectives to ensure it remains complete and accurate. The final approved version is scheduled for submission on October 31, 2025.

14.1.3 Architectural Design Specification

The Architectural Design Specification will be maintained in LaTeX format within Overleaf and supplemented with interactive diagrams created using tools such as Figma or draw.io to represent the system architecture. Updates will be made as architectural decisions are finalized, technology components are selected, or integration patterns are defined. The document will be reviewed during each Friday sprint retrospective to maintain consistency with evolving requirements. The final approved version is scheduled for submission on November 21, 2025.

14.1.4 Detailed Design Specification

The Detailed Design Specification will be maintained in Overleaf and updated as individual components are designed, APIs are defined, database schemas are finalized, and class diagrams are developed. The document will include UML diagrams, sequence diagrams, and detailed interface specifications for each major system component. The team will review the design documentation during each Friday sprint retrospective to validate technical decisions and ensure consistency across all modules. Development of this document will continue in the next semester (CSE 4317), with the final approved version scheduled for submission on January, 2026.

14.2 Recurring Sprint Items

14.2.1 Product Backlog

Items will be added to the product backlog from the System Requirements Specification as user stories, tasks, and technical requirements. Each backlog item will be categorized by type including features, bugs, spikes, technical debt, and research tasks. Items will be prioritized based on story points estimated through planning poker and business value determined through team discussion. Prioritization decisions will be made through a team poll during backlog refinement sessions conducted at the start of each sprint. The product backlog will be maintained and shared using Atlassian Jira within the Confluence suite, providing visibility to all team members and stakeholders with appropriate access permissions.

14.2.2 Sprint Planning

Sprint planning will occur at the beginning of each sprint cycle during scheduled Friday lab sessions or dedicated team meetings. The team will conduct approximately 8 sprints throughout the Fall 2025 and Spring 2026 semesters, with each sprint lasting 2 weeks. During planning sessions, the team will review the product backlog, estimate story points for upcoming items, and commit to a realistic sprint goal based on team velocity from previous sprints.

14.2.3 Sprint Goal

The sprint goal will be decided collaboratively by the team during sprint planning sessions, with the rotating Product Owner facilitating the discussion and making the final determination. Customer involvement will occur through regular feedback sessions where stakeholders review completed work and provide input on priorities for upcoming sprints. The sprint goal will align with project milestones, course deliverables, and stakeholder feedback to ensure continuous progress toward the final system prototype.

14.2.4 Sprint Backlog

The Product Manager, in collaboration with the development team, will decide which product backlog items make their way into the sprint backlog based on priority, dependencies, and team capacity. The sprint backlog will be maintained in Jira using a Scrum board view that provides visibility into backlog items, in-progress work, and completed tasks. The board will be updated daily as team members move tasks through workflow states including To Do, In Progress, In Review, QA/Test and Done.

14.2.5 Task Breakdown

Individual tasks will be assigned from the sprint backlog using a vertical slicing approach, where features are broken down into end-to-end increments that deliver complete functionality across all system layers. Team members will voluntarily claim tasks during sprint planning and daily stand-up meetings based on their expertise and current workload. Time spent on tasks will be documented in Jira through time tracking features, with team members logging hours spent on each task to support velocity calculations and sprint burn down chart generation.

14.2.6 Sprint Burn Down Charts

The Scrum Master for each sprint will be responsible for generating burn down charts using data automatically captured in Jira. Team members will update their task progress and remaining effort estimates daily, allowing the Scrum Master to access the total amount of effort expended by each individual through Jira reporting features. The burn down chart will display remaining story points or task hours on the vertical axis and sprint days on the horizontal axis, with an ideal burn down line showing expected progress and an actual line showing real progress. An example burn down chart is shown in

Figure 2.

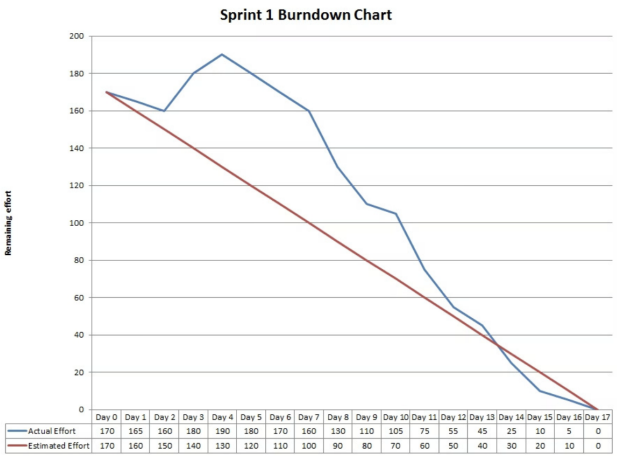

Figure 2: Example sprint burn down chart

14.2.7 Sprint Retrospective

The sprint retrospective will be conducted as a team immediately following each sprint review presentation, typically during Friday lab sessions. The retrospective will follow a structured format where team members discuss what went well, what could be improved, and action items for the next sprint. Group-level insights and action items will be documented in Confluence and reviewed at the start of the next sprint. Individual reflections will be included in each team member's weekly status report, due within 24 hours after the retrospective meeting.

14.2.8 Individual Status Reports

Each team member will submit an individual status report at the end of every sprint, documenting completed tasks, current work in progress, blockers encountered, time spent on various activities, and planned work for the upcoming sprint. Reports will include a self-assessment of contribution to the team and reflections on personal learning and growth. Key items in the report will include sprint goal alignment, story points completed, collaboration with teammates, and technical challenges overcome.

14.2.9 Engineering Notebooks

Each team member will update their engineering notebook at a minimum of once per week, with entries documenting design decisions, implementation approaches, testing results, and challenges encountered. A minimum of two pages must be completed per week, with detailed descriptions, diagrams, and code snippets as appropriate. In addition to individual notebooks, the team will maintain a working document section in Confluence that contains collaborative documentation for spike tickets, bug fixes, technical research, and other ad-hoc investigations as required. This shared workspace will serve as a centralized knowledge base for the team, capturing exploratory work and solutions that may not fit into formal documentation deliverables. Team accountability will be maintained through weekly peer reviews during lab sessions, where team members will verify each other's notebook entries.

REFERENCES

[1] Avail, "Rental property management software features," https://www.avail.co/landlords/features, 2025, accessed: 2025-10-10.

[2] AppFolio, Inc., "Property management software features," https://www.appfolio.com/property-manager, 2025, accessed: 2025-10-10.

[3] Buildium, LLC, "Buildium: Property management software," https://www.buildium.com/, 2025, accessed: 2025-10-10.

[4] Yardi Systems, Inc., "Residential software for property managers," https://www.yardibreeze.com/residential-features/, 2024, accessed: 2025-10-10.

[5] Entrata, Inc., "Student housing management software," https://www.entrata.com/solutions/student, 2025, accessed: 2025-10-10.

[6] University of Texas at Arlington Housing & Residence Life, "Welcome to the uta housing portal," https://uta.starrezhousing.com/StarRezPortalX, 2025, accessed: 2025-10-10.