ZimBuilds Building Plan Approval System

By

Ropafadzo Esalencia Munetsi

(H210159W)

HIT400 Capstone Project Submitted in Partial Fulfillment of the

Requirements of the degree of

Bachelor of Technology

In

Software Engineering

In the

School of Information Sciences and Technology

Harare Institute of Technology

Zimbabwe

Supervisor: Mr. Chiworera

.....

May/2025

ABSTRACT

The increasing requirement for successful urban growth in Zimbabwe has laid the need to improve traditional plan submission and approval processes. This capstone project presents a digital Residential Plan Approval System powered by artificial intelligence that computerizes checks for compliance and streamlines application handling for city councils. The system facilitates role-based interactions for three significant sets of users, i.e., applicants, inspectors, and administrators. It incorporates AI technologies for checking submitted plans against building regulations, as well as automated scheduling of inspections and online payments. Through the replacement of paper-based, manual procedures with a digital process flow, the system aims to reduce processing time, improve transparency, and deliver better services. This research outlines the design architecture of the system, functional and non-functional requirements, and provides a prototype implementation as a solution to the problem.

PREFACE

This report demonstrates the design and development of an AI-based Plan Approval System to revamp the building plan approval process in local city councils in Zimbabwe. It was rooted in personal experience and observation of inefficiency, delay, and lack of transparency that have long plagued manual procedures in urban government offices that I undertook this project. Through this research, I sought to study how emerging technologies such as artificial intelligence and web-enabled systems can be used to bring revolutionary change in the delivery of public services.

The project combines several fundamental areas of my academic experience, such as software engineering, database management, artificial intelligence, and human-computer interaction. The design process involved extensive interaction with actual users—applicants, inspectors, and administrative personnel—via interviews, surveys, and testing, enabling me to craft a solution that is both functional and user-friendly.

Moreover, the project also reinforced my understanding in system architecture, UI/UX design principles, data protection, and agile development methodologies. It not only provided technical knowledge but also taught me the importance of iterative design, stakeholder interaction, and scalability in designing sustainable digital systems for the public sector.

This report is drafted with the intention of guiding the reader from problem identification through system design, implementation considerations, testing phases, and final evaluation of the system. It is both academic research writing and a proposal for practical usage.

I trust this work is a meaningful contribution to ongoing efforts towards e-governance, digitalization, and smart city development in Zimbabwe and other places.

ACKNOWLEDGEMENT

I would like to extend my warmest gratitude to all the persons who helped bring this project to successful fruition.

First and foremost, I would like to thank the Lord Almighty, whose presence and grace has been with me from start of this program till now that l am achieving a great milestone.

Special appreciation goes to my project supervisor, **Mr. Chiworera,** for their sound guidance, valuable criticism, and continuous encouragement throughout this study.

I also would like to express my gratitude to the city council members and inspectors who were involved in the interviews, sharing information regarding the current planning approval processes and justifying the necessity for this proposed solution. They brought invaluable input from their field experience to the system design.

My sincere gratitude also goes to my loved ones, my peers and family for their moral encouragement and support. Without their encouragement and trust in my ambition, this experience would not have been attainable. I would also like to thank them for the resources they helped with for this study to be a success.

Finally, I am grateful to the administrative and faculty staffs of Harare Institute of Technology for providing the academic environment and resources necessary to complete this capstone project.

DEDICATION

This project is dedicated to my family, whose unwavering love, prayers, and moral support have been the backbone of my academic and personal growth. Your belief in me has carried me through challenging times and motivated me to pursue excellence.

I also dedicate this work to the city council boards of Zimbabwe, planning departments, and inspection teams whose daily efforts often go unrecognized, yet play a critical role in shaping safe and sustainable urban environments. Your dedication to public service has inspired the creation of this system, and it is my hope that this solution contributes meaningfully to making your work more efficient and impactful.

To the citizens who navigate the complex journey of development and compliance, this system is for you—a step toward fairness, transparency, and improved access to public services.

Lastly, I dedicate this work to my peers, mentors, and the wider academic and professional community working toward the digital transformation of Africa's public institutions. May this serve as a small but significant contribution to our collective pursuit of innovation and service to society.

This is to certify that HIT 400 Project entitled "**ZimBuilds Building Plan Approval System" has** been completed by **Ropafadzo Esalencia** (H210159W) for partial fulfilment of the requirements for the award of **Bachelor of Technology** degree in **Software Engineering**. This work is carried out by **her** under my supervision and has not been submitted earlier for the award of any other degree or diploma in any university to the best of my knowledge.

Your Supervisor Name	**Approved/Not Approved**
Mr Chiworera	
Project Supervisor	Project Coordinator
Signature: ……	**Signature: ……**
Date: ……	**Date: ……**

Certificate of Declaration

This is to certify that work entitled "ZimBuilds Residential Plan Approval System "is *submitted in partial fulfillment of the requirements for the award of Bachelor of Technology (Hons) in Software Engineering, Harare Institute of Technology. It is further certified that no part of research has been submitted to any university for the award of any other degree.*

(Supervisor)	Signature……..	Date…….
(Mentor)	Signature……..	Date……
(Chairman)	Signature……..	Date……..

Project Documentation Marking Guide

ITEM	TOTAL MARK /%	ACQUIRED/%
PRESENTATION- Format-Times Roman 12 for ordinary text, Main headings Times Roman 14, spacing 1.5. Chapters and sub-chapters, tables and diagrams should be numbered. Document should be in report form. Range of document pages. Between 50 and 100.Work should be clear and neat	**5**	
Pre-Chapter Section Abstract, Preface, Acknowledgements, Dedication & Declaration	**5**	
Chapter One-Introduction Background, Problem Statement, Objectives – smart, clearly measurable from your system. Always start with a TO… Hypothesis, Justification, Proposed Tools Feasibility study: Technical, Economic & Operational Project plan –Time plan, Gantt chart	**10**	
Chapter Two-Literature Review Introduction, Related work & Conclusion	**10**	
Chapter Three –Analysis Information Gathering Tools, Description of system Data analysis –Using UML context diagrams, DFD of existing system Evaluation of Alternatives Systems, Functional Analysis of Proposed System-Functional and Non-functional Requirements, User Case Diagrams	**15**	
Chapter Four –Design Systems Diagrams –Using UML Context diagrams, DFD, Activity diagrams Architectural Design-hardware, networking Database Design –ER diagrams, Normalized Databases Program Design-Class diagrams, Sequence diagrams, Package diagrams, Pseudo code Interface Design-Screenshots of user interface	**20**	
Chapter Five-Implementation & Testing Pseudo code of major modules /Sample of real code can be written here Software Testing-Unit, Module, Integration, System, Database & Acceptance	**20**	
Chapter Six –Conclusions and Recommendations Results and summary, Recommendations & Future Works	**10**	
Bibliography –Proper numbering should be used Appendices –templates of data collection tools, user manual of the working system, sample code, research papers	**5**	
	100	**/100**

Contents

Table of Figures

Chapter One - Introduction

1.1 Background

The rate of urbanization in Zimbabwe has increased in the past two decades due to population growth, rural-urban migration, and economic growth. The increase in urbanization has created higher demands for housing, thereby putting pressure on the city councils to approve building proposals efficiently. Yet, the process of approval is still characterized by manual, antiquated, and inefficient processes at the expense of applicants and municipal authorities.

The application process begins with applicants submitting physical building plans and supporting documents to the planning department of the council. They are reviewed by different departments like planning, engineering, and health. Architectural drawings, site plans, and structural design must be submitted in duplicate. Plans are presented to different departments for review, each looking at zoning, building codes, and health codes. The process has multiple stages of authorization, requiring the applicants to make amendments to their proposals based on the comments raised by the department. As for inspections, they are carried out at various stages of building to ensure adherence. Physical records are utilized for keeping records, making it difficult to track progress and access information.

The manual approval delay process lasts several months. The applicants don't always get any idea regarding the application status, leading to frustration. Employees at the city council also bear a heavy administrative burden because of the manual processing of documents and coordination with various departments. Manual processes are error-prone, leading to wrongful approvals and delays. The lack of centralization renders it difficult to track application statuses, inspection timetables, and document revisions efficiently. Disapprovals cause delays, impacting urban development and construction activities. Although digital connectivity has improved in Zimbabwe, the adoption of technology at municipal public sector councils is fragmented.

1.2 Problem Statement

The existing manual process of residential building plan approval is cumbersome, slow, and prone to errors. Rapid urbanization and population growth in Zimbabwe have created a heightened demand for residential housing, putting substantial pressure on city councils to efficiently approve building plans. However, the current process for approving residential building plans is mostly

manual, outdated, and inefficient. This results in numerous challenges for both applicants and city council officials, including:

- Prolonged processing times for plan approvals.
- A lack of transparency and accountability in the approval process.
- An increased administrative burden on city council staff.
- The potential for errors and inconsistencies in plan assessments.
- Challenges in tracking application statuses and inspection schedules.
- Delays in revenue collection due to inefficient payment processing.

1.3 Objectives

- To develop a web app that streamlines the application process for plan approval.
- To automate document verification using AI-powered document verification.
- To automate payment by integrating online payments
- To implement automated inspector appointment scheduling

1.4 Hypothesis

The critical assumption of this project is that a web-based system supported by AI for residential building plan approval will make a significant difference in the efficiency of the city council, particularly by reducing the time taken to approve these plans. This assumption is a falsifiable statement, attempting to link the utilization of the web-based system with improved approval efficiency, measured in terms of reduced processing time and better workflow, utilization of resources, and service delivery.

Processing time is crucial because it directly affects the residents and the city council. For the residents, slower approvals translate to delayed construction, higher expenses, and aggravation. For the council, it highlights the inefficiencies that overtax resources and slow down development. Reducing processing time solves a significant issue in the system. AI-powered plan review is at the core of this hypothesis. It will endeavor to automate portions of plan review, including code compliance and structural analysis. This will cut down dramatically on review time that must be handled manually, decreasing overall processing time.

To test this hypothesis, we advance two conflicting propositions:

Null Hypothesis (H0): The null hypothesis shall be the current state, implying that the web application will not reduce plan approval times significantly or make any increase in efficiency. It supposes that there will be no effect on the existing procedures.

Alternative Hypothesis (H1): This hypothesis states that the web application will significantly reduce the time taken for plan approvals while also improving overall efficiency. We aim to prove this through the implementation and evaluation of the project.

Significance of the Hypothesis: The present hypothesis will be examined to see the efficacy of the proposed system. If established, the study would reveal technology's potential for improving Zimbabwean public services.

1.5 Justification

Currently, Zimbabwe is witnessing rapid urbanization, and there is an enormous demand for housing. Because of this, there is a dire need to modernize and streamline the currently outdated municipal processes. The manual system is slow and inefficient and has lots of human errors; it slows down the approval of residential building plans, thereby delaying developers and applicants, reducing transparency and accountability, and even increasing the administrative burden on city council staff.

The reasons for creating a digital platform assisted by AI are triple:

- Operational Efficiency

  Through digital submission, automated document verification, and inspection scheduling through an intelligent system, will reduce the time for application processing, increasing service delivery for applicants and allowing the municipal staff to handle more applications without an increase in workload.
- Transparency and Accountability

  Applications can be trackable in real time with the system. Applicants can receive automated status updates, and there is an audit trail for every action taken. This minimizes corruption chances, applicant uncertainty, and departmental accountability.

- Error Reduction and Compliance

  A manual review of plans presents the possibility of overlooking noncompliance with which structures can be approved or plans rejected on avoidable grounds. By integrating AI-based compliance, we enable consistent and accurate evaluations of building plans by verifying adherence to regulations

1.6. Proposed Tools

- Programming Languages: JavaScript, Python
- Frameworks: Next JS, Node JS
- Database: PostgreSQL
- AI Tools: OpenCV, Tesseract.js for document verification, Pretrained model for semantic analysis and compliance feedback.
- Code Editor: VS Code
- Testing Endpoints: Postman

1.7 Feasibility Study

1.7.1 Technical Feasibility:

- Existing Infrastructure

  Urban local governments in Zimbabwe have, for the most part, a foundation of basic ICT infrastructure such as internet access, networked computers, and digital records. Yet, the majority of these are outdated and not maximally utilized.
- User Access

  Internet penetration is getting better in Zimbabwe but there still exists a digital divide. Yet, most stakeholders in the building construction sector (e.g., contractors, architects) have access to internet-enabled devices.
- Technology Compatibility

  The compliance tool for AI requires a back-end to handle digital plan files, such as PDF and DWG. Python libraries are used to develop the AI engine, and APIs are used to connect them to the front end. A web-based front-end through Next.js or React will give

browser compatibility, while the architecture file formats can be supported through suitable upload and conversion tools.

1.7.2Economic Feasibility:

A cost-benefit analysis was drawn up to conclude the economic viability of implementing this AI-based plan approval system.

- Estimated Costs

Cost Item	Estimated Cost (USD)
System Development	$25,000
AI Model Development & Training	$10,000
Hardware Upgrades (if needed)	$5,000
Cloud Hosting & Maintenance (Annual)	$3,000
User Training & Documentation	$2,000
Total Initial Investment	$45,000

- Potential ROI

  The AI compliance engine can significantly enhance efficiency by reducing the time taken to approve plans by over 60%, enabling councils to recover revenue more quickly. Faster approvals will be anticipated to lead to increased building activity that will ultimately result in higher fee collections. Automated confirmation also reduces human effort, decreases errors, and minimizes the use of paper, leading to cost savings overall.

- Payback Period

  With improved operating efficiency and greater plan processing, the system can easily recover its investment within 2–3 years through administrative savings and greater permit returns.

1.7.3 Operational Feasibility

- User Readiness

  Government staff are familiar with other computer programs but will need to be trained to use AI systems, for which moderate to high levels of take-up will be expected. Contractors

and applicants already use online platforms and will likely support AI-driven approvals that speed up processing.

- Required Changes

  The work process will be adapted to include AI checks for architectural plan compliance. Internal procedures must be updated to legally accept documents certified by AI, and short training sessions (1–2 weeks) will be required for employees and users to adapt to the new system.

- Resistance to Change

  Resistance at an early stage is expected, particularly from staff familiar with handwork. Involving them at an early point within system planning and illustrating efficiency savings will reduce resistance.

1.8 Project Plan

1.8.1 Time Plan

PHASE	DURATION (Weeks)	STARTING DATE	ENDING DATE
Requirements Analysis	3 weeks	01 Sep 2024	21 Sep 2024
System Design	4 weeks	22 Sep 2024	19 Oct 2024
AI Model Development	6 weeks	20 Oct 2024	30 Nov 2024
Backend Development (APIs)	5 weeks	01 Dec 2024	04 Jan 2025
Frontend Development	5 weeks	05 Jan 2025	08 Feb 2025
AI Integration & File Support	3 weeks	09 Feb 2025	01 Mar 2025
Testing & Debugging	3 weeks	02 Mar 2025	22 Mar 2025
User Training & Documentation	2 weeks	23 Mar 2025	05 Apr 2025
Implementation	4 weeks	06 Apr 2025	03 May 2025
Maintenance & Support	2 weeks	04 May 2025	17 May 2025

Figure 1: Time Plan

1.8.2 Gantt Chart

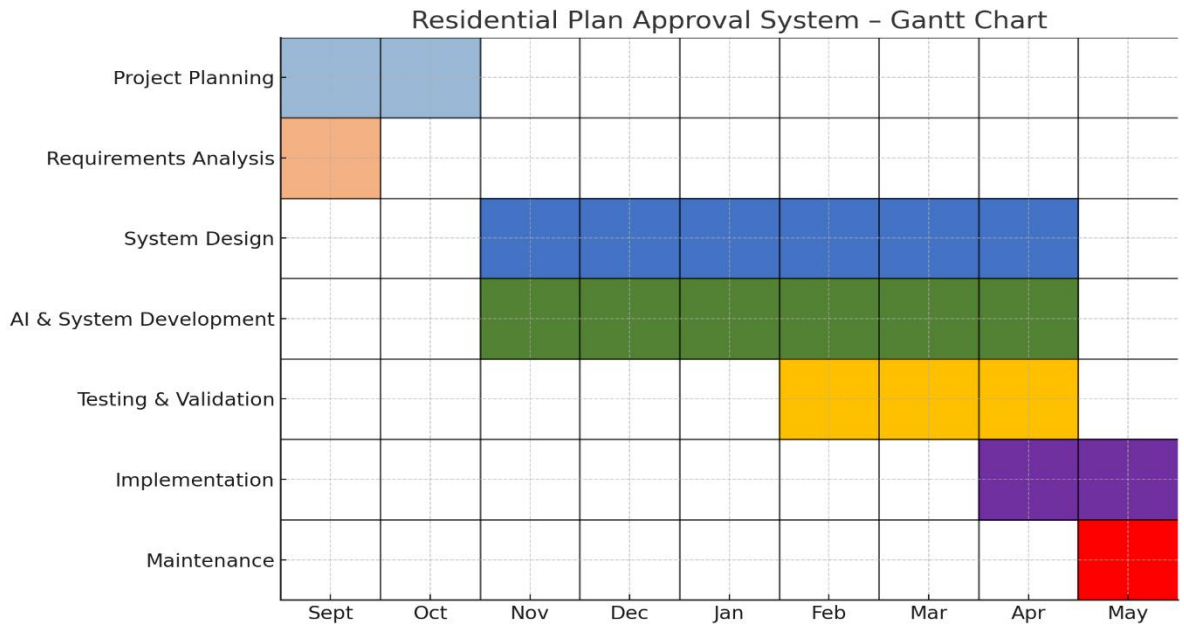


Figure 2: Gantt Chart

Chapter Two – Literature Review

Introduction

The approval of building plans is an essential regulatory procedure that guarantees urban development complies with defined criteria, fostering safety, sustainability, and organized progress. To guarantee adherence to zoning laws, structural integrity, and environmental requirements, city councils in Zimbabwe are in charge of examining and approving building designs. Effective urban governance is hampered by approval process inefficiencies, such as a lack of digital integration, bureaucratic delays, and corruption. With an emphasis on issues, developments in technology, and best practices relevant to city councils in Zimbabwe, this review of the literature looks at the body of research on building plan approval systems.

Related work

Digital Transformation in Urban Planning and Building approvals

Over the past decade, digitalization in urban planning has caught global attention as city councils seek to ensure transparency, reduce bureaucratic processes, and streamline service delivery. The application of e-government solutions in urban building processes has been worthwhile in reducing processing times and improving regulatory compliance.

An early and successful example among the pioneering is Singapore's CORENET (Construction and Real Estate Network) that allows completely digital submission of building plans, concurrent agency checking, and automatic checking of compliance. In [1] Tan et al., 2016, the project reportedly reduced the time taken to process plans by 65%, streamlining communication between developers, architects, and regulatory bodies. Likewise, in [2] Dubai's Smart Permit System by Smart Dubai initiative launched AI-powered platforms that automated building code checking and incorporated Building Information Modeling (BIM) for real-time inspection, [2], 2020. In Odisha [3], India has also implemented electronic systems to allow for rapid approval of building plans, such as the Online Building Plan Approval System (OBPAS). OBPAS has the SmartDCR feature that allows for automated checks for compliance, reducing the risk of human errors and shortening approval time. In a case study in Odisha [3], the system improved transparency, shortened physical visits, and reduced approval times to at least to seven days.

Studies are however not limited to Asia; some European and North American municipalities have also used digital technologies to streamline urban development and building permit procedures. Finland is one such country where Finland's Lupapiste service offers a single online platform for construction permits to significantly accelerate turnaround time, as well as to aid inter-agency collaboration [4]. Similarly in the USA, Boston implemented the 'Inspecting Boston' platform through a synergy of GIS and mobile apps to advance automation of inspection routing and promote greater access of data to inspectors [5]. Similarly, Canada's City of Edmonton uses an e-permitting system that synergizes AI to compare virtual building models with code requirements [6]. These are evidence of the way governments in the Global North have adopted systemic digital methods that improve not just efficiency but also transparency, citizen engagement, and accountability in the planning urban procedures.

Conversely, most African, and indeed Zimbabwean, cities continue to use manual, paper-based approval processes. Studies like [7] acknowledge inefficiency in municipalities in Zimbabwe because of the duplication of powers, procedural transparency, and lack of central databases. Late approvals of the plan and scheduling of inspections are prevalent, fueling informal settlements and unsafe structures [7]. The African Development Bank, in a report [8], observed that digitization had the potential to minimize possible channels of corruption and cut the time taken to deliver services. However, in ways Africa is not lagging behind, for instance Rwanda's Irembo platform, which digitized more than 100 government services such as construction permits and land registration, suggest that centralized data architecture and strong political will are at the heart of digital governance success according to World Bank, 2020 in [9]. Kenya's electronic construction permit system is another useful case study, having cut approval times from more than 30 days to less than 10 in some counties (ICT Authority Kenya, 2019). Reyes-Carranza et al. [10] examine Ke DAMS, a web-based system in Nairobi, Kenya, developed in 2020 to automate and streamline building permit approvals which aims to promote transparency and combat corruption in the construction sector. While the system marks progress in digital governance, the authors note that it often neglects the unique needs of informal or peripheral areas [10]. They argue for context-aware, flexible digital solutions that address both formal and informal development practices to ensure inclusivity and effectiveness in urban planning.

AI in Building Plan Review and Compliance Checking

The latest innovations in artificial intelligence (AI) and machine learning (ML) have opened up the possibility of automatic building plan scrutiny going beyond simple static rule-based scrutiny. Algorithms are now capable of interpreting Computer-Aided Design (CAD) drawings, carrying out zoning conformity checks, and detecting structural anomalies based on national and local building codes. In 2016 [11] developed a natural language processing system to extract regulatory rules from building codes and automatically enforce them against architectural drawings—a method proven to identify as much as 80% of compliance errors in pilot testing. Similarly, Auto Codes, an automated rule-based checker, has been implemented in certain U.S. jurisdictions to automatically verify plans against fire safety, egress, and structural codes [12] Hjelseth, 2017. Deep learning models have also been applied to examine blueprint photos and visually identify non-compliance. Lee et al. (2019) suggested a convolutional neural network (CNN) system that is capable of recognizing structural elements and comparing them against code requirements. Such tools are especially useful in detecting breaches in spatial arrangements, setback lines, and accessibility pathways in high-rise development. In Africa, however, AI application in this regard is not prevalent. A 2021 article by Adegun and Oduwaye compared the feasibility of AI applications to Lagos' urban planning system and found severe setbacks: inconsistent digital plan formats, poor-quality internet infrastructure, and poor training in ML technology. These setbacks are also present in Zimbabwe, necessitating locally anchored solutions [13].

Automated Scheduling systems for Inspections

Scheduling inspections is another critical task in municipal government with inefficiencies. Manual scheduling procedures lead to inefficient allocation of resources, inspector burnout, and lengthy waiting times for applicants. Optimized software programs based on optimization algorithms and predictive analytics have proven to be the solution to these issues. In [14] New York City DOB NOW system, implemented by the Department of Buildings, automates appointment scheduling for inspections and dynamically assigns inspectors based on workload and location (NYC Department of Buildings, 2019). The system has reduced the average response times for inspections by 30%. AI-based systems have applied genetic algorithms and heuristic optimization techniques to route inspectors efficiently. Chen et al. (2017), for example, created an algorithm to minimize travel time based on learning from historical inspection activity and predicting optimum

schedules. In [17], in a 2020 study in South Africa analyzed Johannesburg city departments' smart inspection models. Although pilots were yielding results, poor connectivity and lack of integrated data systems constrained scalability—problems repeated across Zimbabwe.

Digital Transformation in Zimbabwe and Challenges and Lessons from Prior implementations

Zimbabwean e-government has increasingly used technology in taxation and education but urban planning is largely still analog. Internet penetration levels have been around 60% meaning that most citizens are out of reach of online services, according to the Postal and Telecommunications Regulatory Authority of Zimbabwe according to POTRAZ, 2021. Initiatives like the ZIMRA e-services portal and the Zimbabwe National Spatial Data Infrastructure (ZNSDI) provide a basis for digital urban planning, but there is no end-to-end municipal platform for plan approvals as yet. Research by Mukwashi (2020) and Chigwedere (2021) identifies institutional inertia, disjointed ICT strategies, and capacity constraints as major hindrances [18][19]. In contrast to this, Rwanda's Irembo and Kenya's Huduma Centers have succeeded with centralized platforms, strong digital policy environments, and public-private partnerships. The subsequent case studies offer practical lessons on stakeholder engagement and phased expansion, which can be emulated by Zimbabwean city councils in encouraging sustainability and user trust.

However, these systems are not with challenges, AI systems in public sectors face special challenges. Among the most persistent are:

- Resistance to Change: Public sector employees will be hesitant to accept automation because they fear job loss or diminishing discretion in decision-making (Heeks, 2018).
- Quality of Data: AI algorithms require plenty of clean and labeled data—exiguous in cities with poor digitization histories.
- Skills Deficiencies: Lack of trained personnel in AI, data science, and urban digital planning are principal system adoption barriers (Mhlanga, 2020).
- Ethical Concerns: Biased training data can perpetuate itself within AI models, leading to issues of fairness and accountability in plan approval (Mehrabi et al., 2021).
- Privacy and Security: As AI systems handle urban sensitive information (e.g., locations of utilities, building materials), ensuring privacy of such information becomes the priority (UN-Habitat, 2020).

The Estonian e-governance system experience illustrates the necessity for open algorithms, protection by law, and education of citizens in fostering trust in online spaces (Margetts & Naumann, 2017).

Conclusion

Across the globe, technologies such as CORENET, DOB NOW, and Smart Dubai have demonstrated that AI and automation can revolutionize building plan approvals and inspections. However, the majority of African applications are nonexistent, rule-based, or dispersed. Zimbabwe lacks an integrated platform for AI-based plan compliance validation with automated inspection scheduling.

This research proposes a localized, web-based system that:

- Digitally captures architectural plans and performs rule-based and ML compliance checks.
- Automatically schedules with resource-efficient routing.
- Provides real-time application and document status tracking to applicants.
- Gears up with Zimbabwean building regulations and municipal procedures.

Such an integrated system addresses a root missing link in modern urban planning practice, with a scalable, context-sensitive solution based on global best-of-breed practices yet addressing local constraints.

.

Chapter Three – Analysis

3.1 Information Gathering Tools

When gathering data and information for my system, I sought out various stakeholders, including architects and designers, city council officials and inspectors, construction engineers and builders who use the plans in construction, and urban planners and zoning officials. The tools used include surveys like interviews and questionnaires, and also reviewing the existing process documentation and building bylaws for Zimbabwe.

- Sample Interview Questions

  General Questions

  1. What is the existing procedure for submitting and approving building plans?
  2. What are the major bottlenecks or challenges to this process?
  3. What do you anticipate a digital solution can improve in order to enhance this process?

  For City Council Officials / Planning Department

  4. What departments are responsible for the approval of building plans?
  5. What documents need to be submitted for a general residential application?
  6. How do you currently handle storage and tracking of applications?
  7. How do you provide feedback on plans received from applicants?
  8. What manual compliance checks do you perform?
  9. What are your primary key performance indicators (KPIs) for gauging processing times?
  10. Are there any regulatory or legal guidelines the system must enforce?

  For Inspectors

  11. 11. How are you currently distributing inspection work?
  12. 12. What do you need before conducting an inspection?
  13. 13. How and where do you enter and submit inspection reports?
  14. 14. What are the challenges in processing inspections manually?

  For Applicants

  15. 15. What is your process for submitting a building plan?
  16. 16. Have you experienced delays in receiving feedback or approvals?
  17. 17. How do you remit for approvals or inspections?
  18. 18. Would you be comfortable uploading documents and tracking your application online?
  19. 19. What would make the system more user-friendly for you?

- Sample Questionnaire

User Requirements Questionnaire

Section 1: General Information

1. Full Name: _____

2. User Role:

☐ Architect/Designer ☐ City Council Official

☐ Construction Engineer/Builder ☐ Other (Please specify_____

3. Occupation/Designation: _____Architect_____

4. Years of experience in this role: ____More than 10 years_____

Section 2: Current Process Evaluation

How satisfied are you with the current plan approval process?

☐ Very Satisfied ☐ Satisfied ☐ Neutral ☐ Dissatisfied ☐ Very Dissatisfied

What are the biggest challenges you face in the approval process? *(Select up to 3)*

☐ Long waiting times ☐ Inconsistent feedback from reviewers

☐ Lack of transparency in status updates ☐ Manual paperwork & physical submissions

☐ Complex submission requirements ☐ Other: _____

How long does it typically take to get plans approved?

☐ < 1 week ☐ 1-4 weeks ☐ 1-3 months ☐ >3 months

Section 3: Desired System Improvements

Which features would most improve the approval process? *(Select up to 3)*

☐ Online submission portal ☐ Digital document verification

☐ Automated compliance checks ☐ AI-based error detection

☐ Real-time application tracking ☐ Other: _____

Would a digital dashboard for tracking application status be useful?

☐ Yes

☐ No

☐ Maybe

How should the system notify users about approval updates? *(Select all that apply)*

☐ Email ☐ Web Portal Alerts

☐ SMS ☐ Other: _____

☐ Mobile App Notification

Section 4: Open-Ended Feedback

Do you have any other suggestions for improving the plan approval system?

- Existing process documentation.

To understand and outline the requirements of the proposed residential plan approval system, I began by reviewing existing process documents utilized by Zimbabwean municipal authorities. These included building application forms, departmental review checklists, inspection report templates, and physical payment receipts. By reviewing these documents, I reconstructed the current manual process, mapped all requirements of inputs, decision-making points, and approval flow. The documents identified critical pain points such as resubmissions, poor communication of rejections, and traceability. Based on these observations, I derived functional requirements such as multi-step approval procedures, AI-powered document verification, electronic submission of building plans, and integration of inspection scheduling. Non-functional requirements such as system usability, form consistency, and data protection were also inferred from where current processes fell short. Paper-based analysis ensured the new system would be regulation-compliant while significantly improving efficiency, transparency, and user experience.

3.2 Description of Existing System

Zimbabwe's current building plan approval system is a multi-stage process governed by national legislation and administered by local authorities, ensuring that new construction complies with planning, zoning, and safety regulations. The process follows a regulatory and legal framework. It is governed by the Regional, Town and Country Planning Act under section 12 of chapter 29, Urban Councils Act (Chapter 29:15), Rural District Council Act (Chapter 29:13), Housing Standards and Control Act (Chapter 29:08), and the Environmental Management Act (Chapter 20:27). The process flows through the following steps:

1) Town planning and zoning approval: applicants guarantee if land complies with the intended usage. This is checked by local authorities against the zoning by-laws and development to verify land compatibility with permitted land use.

2) Submission of building plans: Usually three copies of detailed architectural plans which are prepared by qualified professionals, are presented to the planning or engineering branches of the local council for intra-departmental checking.

3) Plan Review and Departmental Verification: The plans are examined for compliance with local requirements, and also the Zimbabwe Building By-laws

4) Payment of Fees: The fee for approval is calculated based on the nature of construction and the location in which the project is being undertaken. For example, in Harare, fees for a main residence in low-density zones are currently set at US$150.

5) Approval and Stamping: Final approval is executed by Director of Engineering Services or equivalent official, stamping the plans in confirmation of compliance. This is done before legally initiating the construction. The approved plans are registered by the local authority formally.

6) Issuance of Building Permit: A building permit is issued, assuring that the planned construction is as per all legal and safety norms.

7) Inspections During Construction: The local authority inspectors visit the site during construction to verify compliance with approved plans. Non-compliance may lead to stoppage of work, penalties, or orders for demolition

8) Issuance of Certificate of Compliance: A certificate is given, attesting that the construction is in all ways as per law and safe.

3.3 Data Analysis

3.4 DFD of Existing System

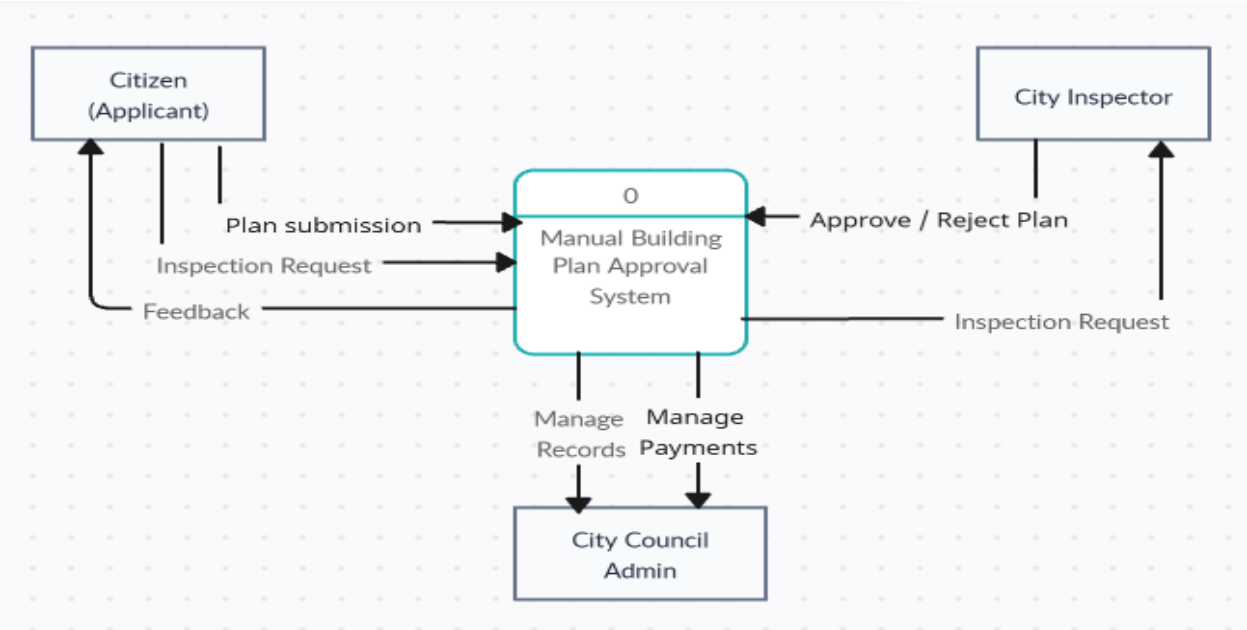


Figure 3: DFD Level 0 of Existing System

3.5 Evaluation of Alternative Systems

ZimBuilds Residential plan approval system as of now is a crucial solution that will reduce burden and improve efficiency in the plan approval and building inspection process. However, there is need to evaluate alternative systems to weigh on cost, security, accuracy, scalability and efficiency of the system etc. By doing so we can identify potential gaps and adjustments to improve proposed solution to make it a more powerful solution.

1. Manual Verification

This is the current traditional practice where paper documents are handled by city council staff manually without using computer tools. Each department checks compliance with building codes, zoning laws, and health requirements manually before approval. Paperwork-intensive processes, file transfer by hand, and written feedback define this process.

Advantages:

- There is no upfront investment in infrastructure or technology.
- Staff is already trained in the manual system.
- Independence from electricity and the internet.

Disadvantages:

- Excessive processing time due to paperwork and delays between departments.
- High risk of omission or misunderstanding of rules.
- No open tracking of applicants, so potential for more corruption.

2. Increasing Number of Inspectors

This answer highlights hiring more building inspectors to prevent on-site physical inspection delays. Efficiency is targeted through added manpower and field coverage, mostly in urban areas of high growth. It does not address document verification, payment, or tracking issues.

Advantages:

- More personnel might reduce the backlog and accelerate inspections.
- Supports job creation and addresses unemployment and capacity gaps in municipalities.

Limitations:

- Recurring cost burden from the recruitment and training of inspectors.
- Fails to address documentation, payment, and approval process choke points
- Still prone to human errors.

3. Off-the-Shelf Software

This involves the procurement of a commercially available off-the-shelf software solution for managing building plan approvals. Such solutions usually have standard workflows, dashboards, and document management functionality. They are typical for public consumption and can be configured to support local rules and processes.

Advantages:

- Rapid deployment: Available solutions can be deployed faster than tailored systems.
- Vendor support: Includes patches, updates, and technical support
- Proven features: Often tested and validated in other jurisdictions.

Disadvantages:

- Lacks customization in the sense that it may be out of step with Zimbabwean laws, workflow, or document formats.
- Periodic subscription fees can be expensive
- Risk of vendor lock-in or having sensitive data on third-party websites

3.5 Functional Analysis of Proposed System

3.5.1 Functional Requirements

1. User Registration and Authentication: Users (Applicants, Inspectors, Admins) must be able to register, log in, and access role-based dashboards using secure credentials.
2. Building Plan Submission: Applicants ought to be able to fill in application forms, upload building plans.
3. AI-Based Document Verification: The platform must automatically check uploaded plans for compliance with local building codes using an in-built AI engine.
4. Inspection Scheduling: Automatically schedule inspection dates and inspectors based on workload and project location. Inspection results can be reported online by the inspectors.
5. Real-Time Application Tracking: The candidates should be able to view the status of their application and receive automatic updates at each stage of the approval process.
6. Payment Integration: The applicants should have the ability to pay online for plan approvals, inspections, and certificates.
7. Revision Management: Revised documents can be uploaded by applicants in accordance with departmental feedback. Version management must be ensured.

8. Audit Logging and Reporting: Administrators should have the ability to generate reports (e.g., applications processed per month, common errors) and to get a full audit trail of activities performed in relation to every application.

3.5.2 Non-Functional Requirements:

1. Performance: The system must handle file uploads and check results in 60 seconds. It must support at least 500 users at the same time without slowing down.
2. Usability: Interfaces should be simple to use and consistent for roles, forms should provide inline validation and tooltips to assist user and system should be mobile responsive.
3. Security: The system will use JWT for authentication and role-based access control, and uploaded files must be scanned for malware.
4. Scalability: The system must be scalable by adding additional servers to serve more users, more towns, or more regions.
5. Availability: The system should be available 99.5% of the time, and planned maintenance should never impact key features during business hours.
6. Maintainability: Modular structure should render bug fixing and updates easy, and the code should be well documented to collaborate or share.

3.7 User Case Diagram

This shows the visual representation of user interactions with the system.

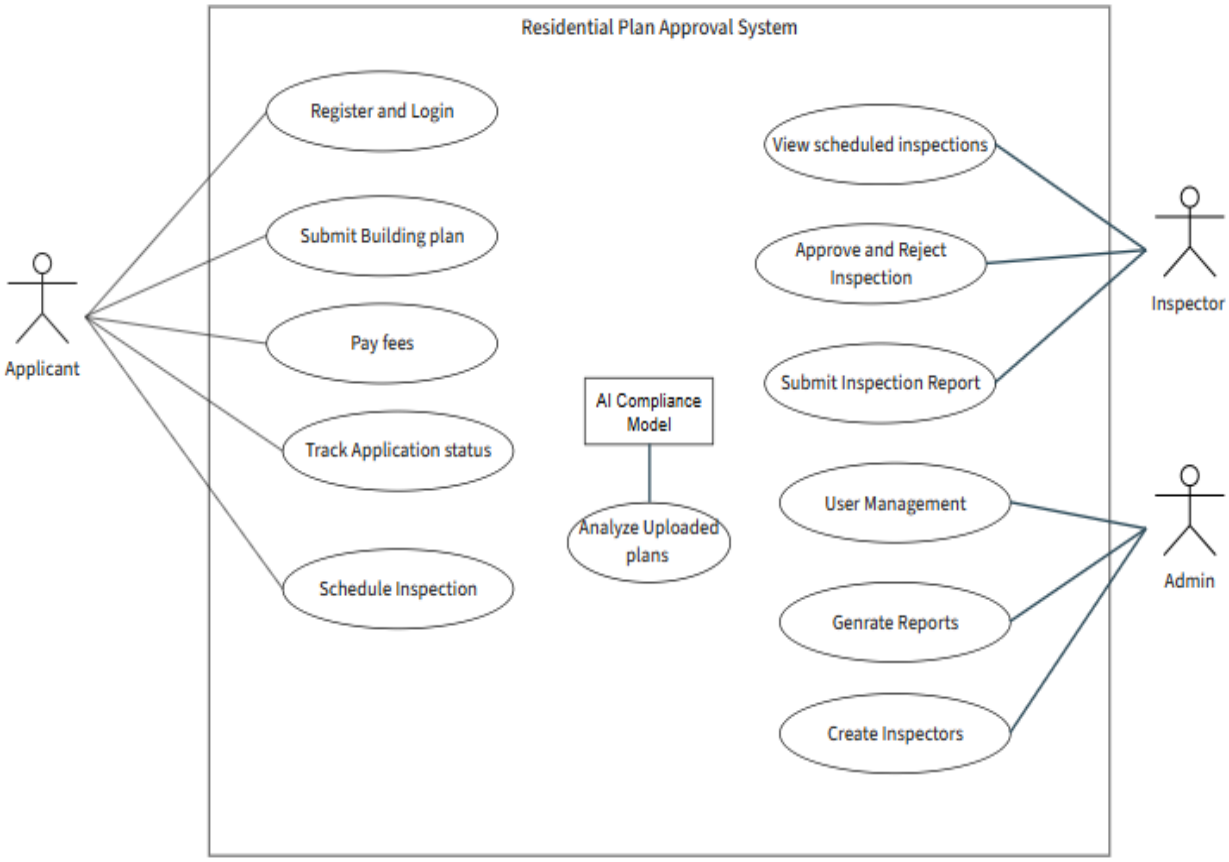


Figure 4:Use Case Diagram

Chapter Four – Design

This Chapter describes the system architecture and how the proposed system is going to function. It will illustrate how the system is designed and how components will be combined to into one functional system.

4.1 Systems Diagrams

4.1.1 UML Context Diagrams

These illustrate system boundaries and interactions. I am going to include a DFD level 0 that shows the high-level design of the system. A DFD level 1 of the system to show the sub processes of the plan approval system.

4.1.2 DFD Diagrams

DFD level 0

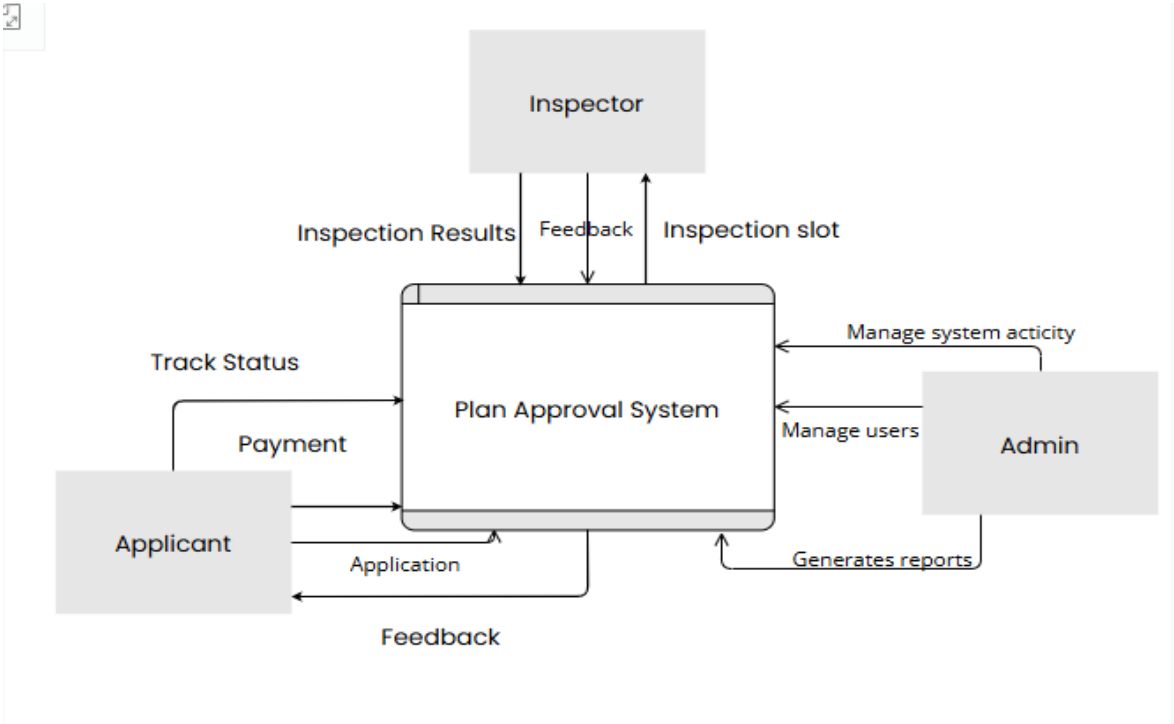


Figure 5: DFD Level 0

DFD Level 1

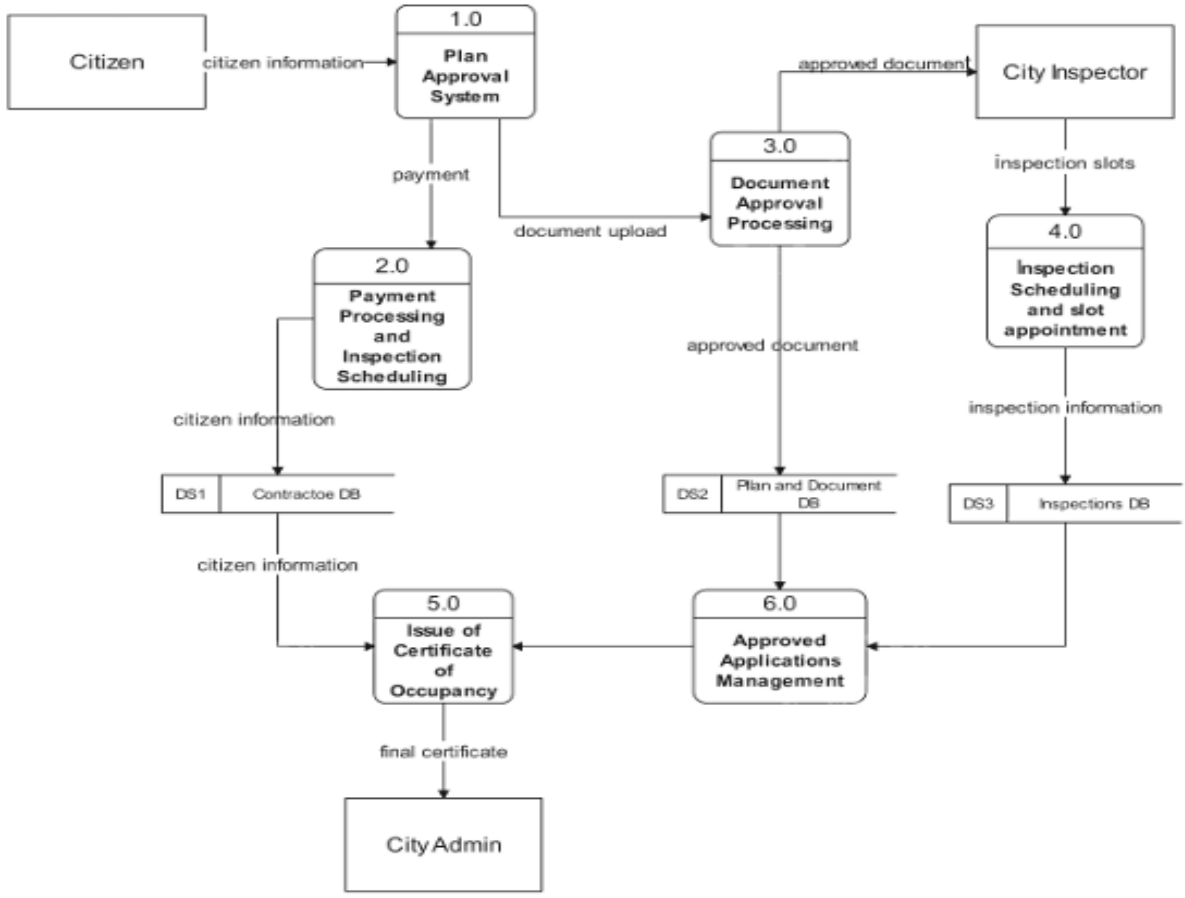


Figure 6: DFD Level 1

4.1.3 Activity Diagram

Show the workflow of the system.

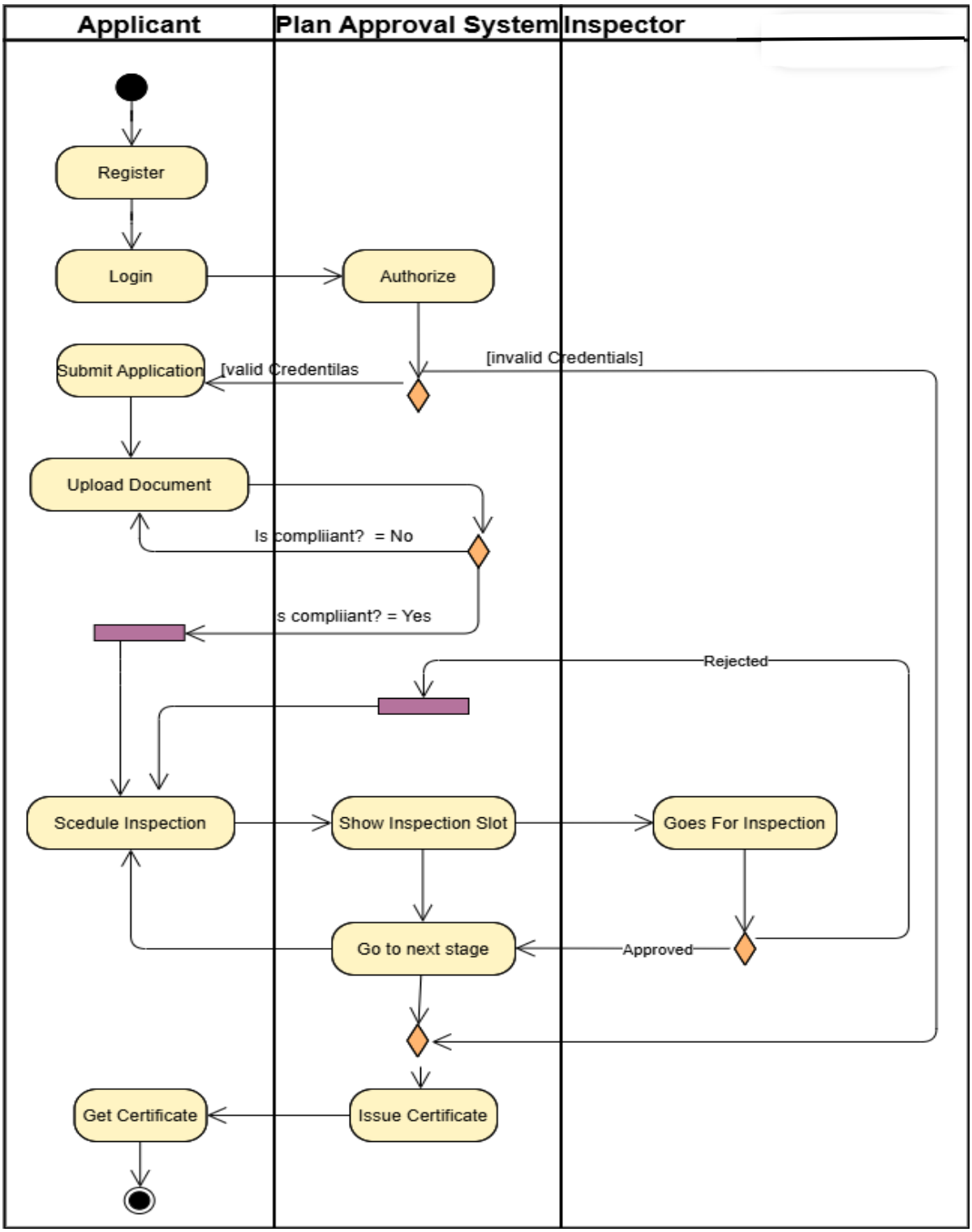


Figure 7: Activity Diagram

4.2 Architectural Design

I am going to show the hardware and networking diagram that shows the specifications of the server and user devices, and the required network setup and configuration, respectively.

4.2.1Hardware and Networking

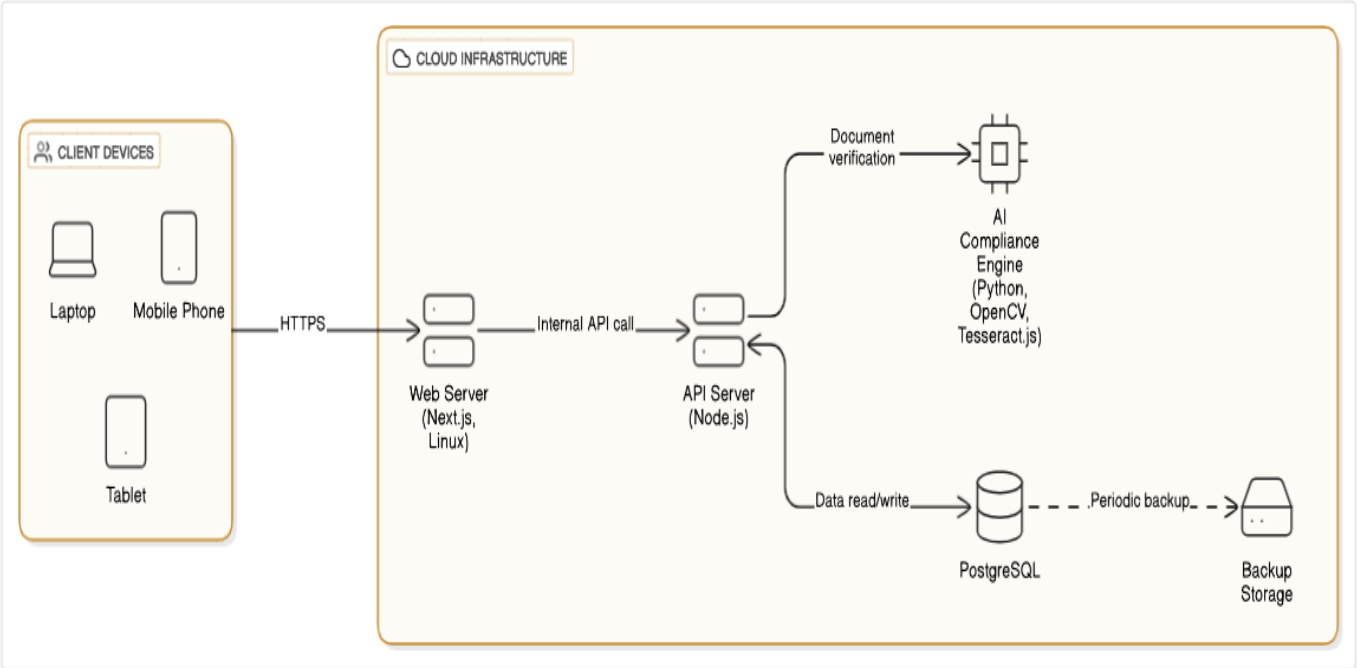


Figure 8:Architectural Diagram

4.3 Database Design

4.3.1 ER Diagram

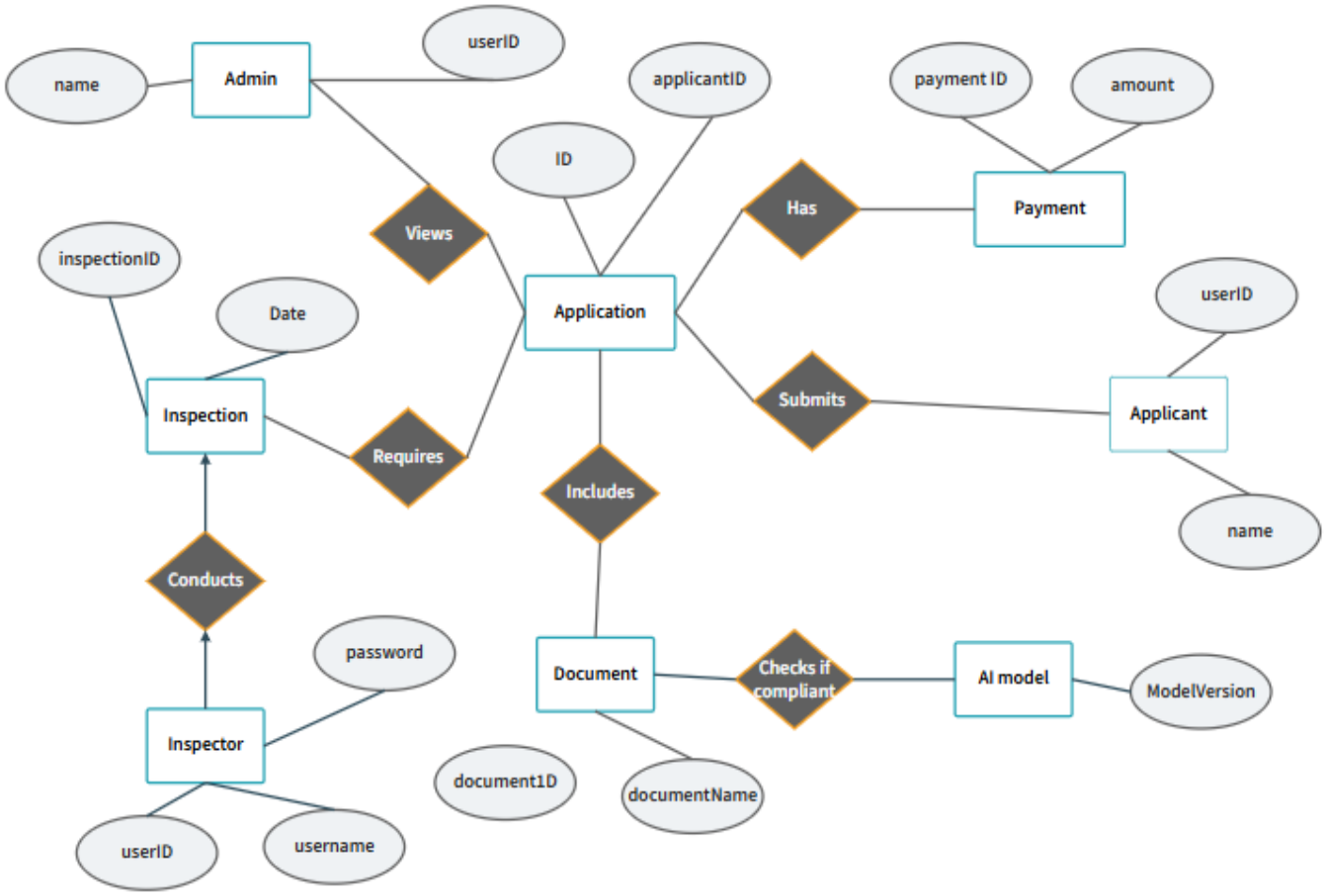


Figure 9: ERD Diagram

4.3.2 Normalized Databases

Users

Column Name	Data Type	Description
user_id	SERIAL PRIMARY KEY	Unique ID for each user
full_name	VARCHAR(100)	Full name of the user
Email	VARCHAR(100) UNIQUE	User email for login
password_hash	TEXT	Hashed user password
Role	VARCHAR(20)	Role: 'Applicant', 'Inspector', or 'Admin'
created_at	TIMESTAMP	When the account was created

Application

Column Name	Data Type	Description
application_id	SERIAL PRIMARY KEY	Unique ID for each building plan application
user_id	INTEGER	FK to users.user_id (Applicant)
application_title	VARCHAR(100)	Title or name of the project
Status	VARCHAR(50)	e.g. 'Pending', 'Approved', 'Rejected'
submitted_at	TIMESTAMP	Date and time the application was submitted

Document

Column Name	Data Type	Description
document_id	SERIAL PRIMARY KEY	Unique document ID
application_id	INTEGER	FK to building_applications.application_id
file_path	TEXT	File storage path or URL
document_type	VARCHAR(50)	E.g., 'Plan'
uploaded_at	TIMESTAMP	When the document was uploaded

Compliance Checks

Column Name	Data Type	Description
check_id	SERIAL PRIMARY KEY	Unique ID for compliance check
document_id	INTEGER	FK to documents.document_id
Result	VARCHAR(50)	'Passed', 'Failed', 'Pending'
Comments	TEXT	Any feedback or issues detected by AI
checked_at	TIMESTAMP	When the check was performed

Inspections

Column Name	Data Type	Description
inspection_id	SERIAL PRIMARY KEY	Unique ID for the inspection
application_id	INTEGER	FK to building_applications.application_id
inspector_id	INTEGER	FK to users.user_id where role='Inspector'
scheduled_date	DATE	Inspection date
Result	VARCHAR(50)	'Pass', 'Fail', 'Pending'
Remarks	TEXT	Inspector remarks
submitted_at	TIMESTAMP	When the inspection was submitted

Payments

Column Name	Data Type	Description
payment_id	SERIAL PRIMARY KEY	Unique payment ID
application_id	INTEGER	FK to building_applications.application_id
Amount	DECIMAL(10, 2)	Amount paid
payment_type	VARCHAR(50)	'Plan Approval', 'Inspection',
payment_status	VARCHAR(20)	'Paid', 'Pending', 'Failed'
receipt_path	TEXT	Path to downloadable electronic receipt
paid_at	TIMESTAMP	Date and time of payment

4.4 Program Design

4.4.1 Class Diagrams:

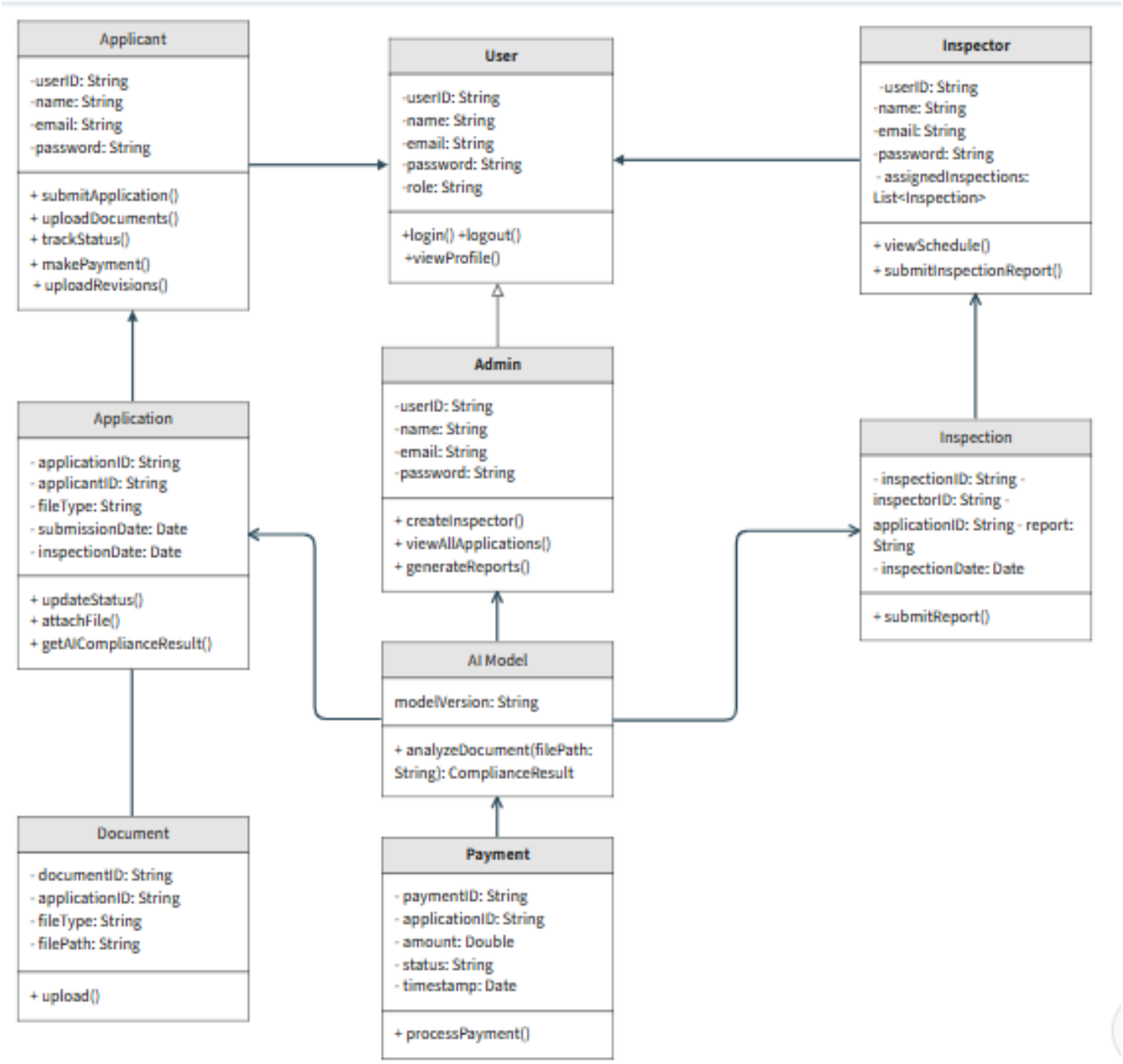


Figure 10: Class Diagram

4.4.2 Sequence Diagram

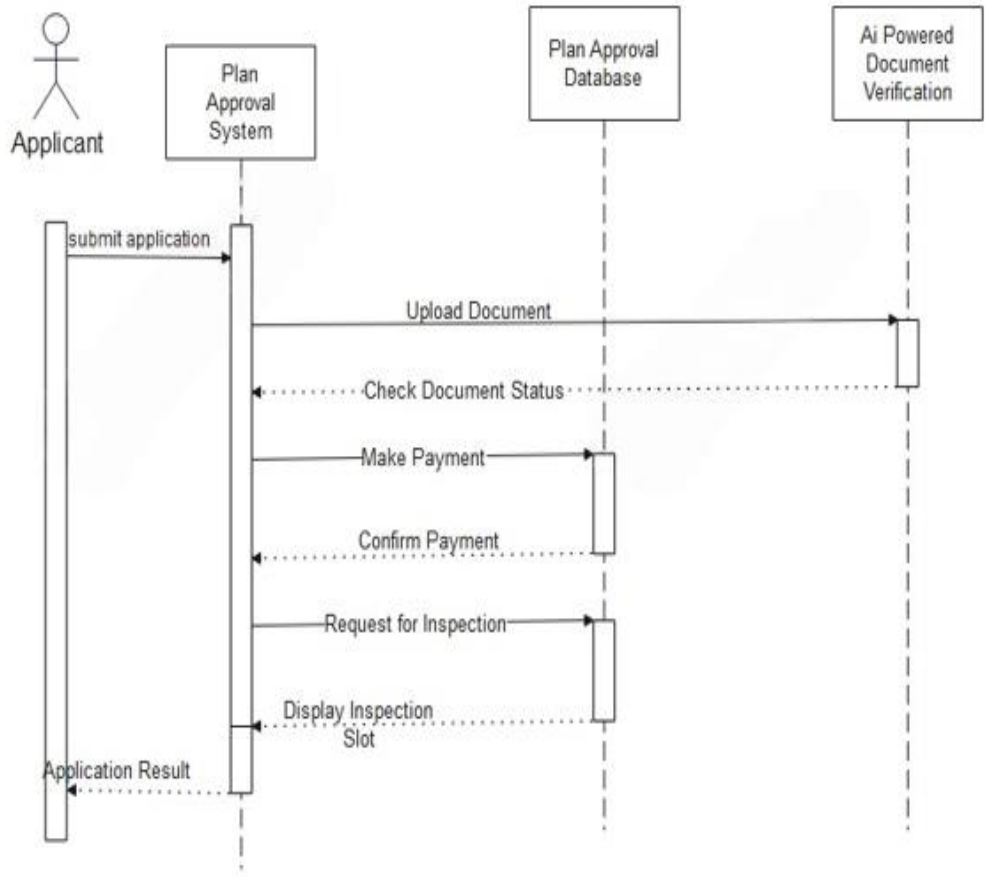


Figure 11: Sequence Diagram

4.4.2 Package Diagrams

Show system organization.

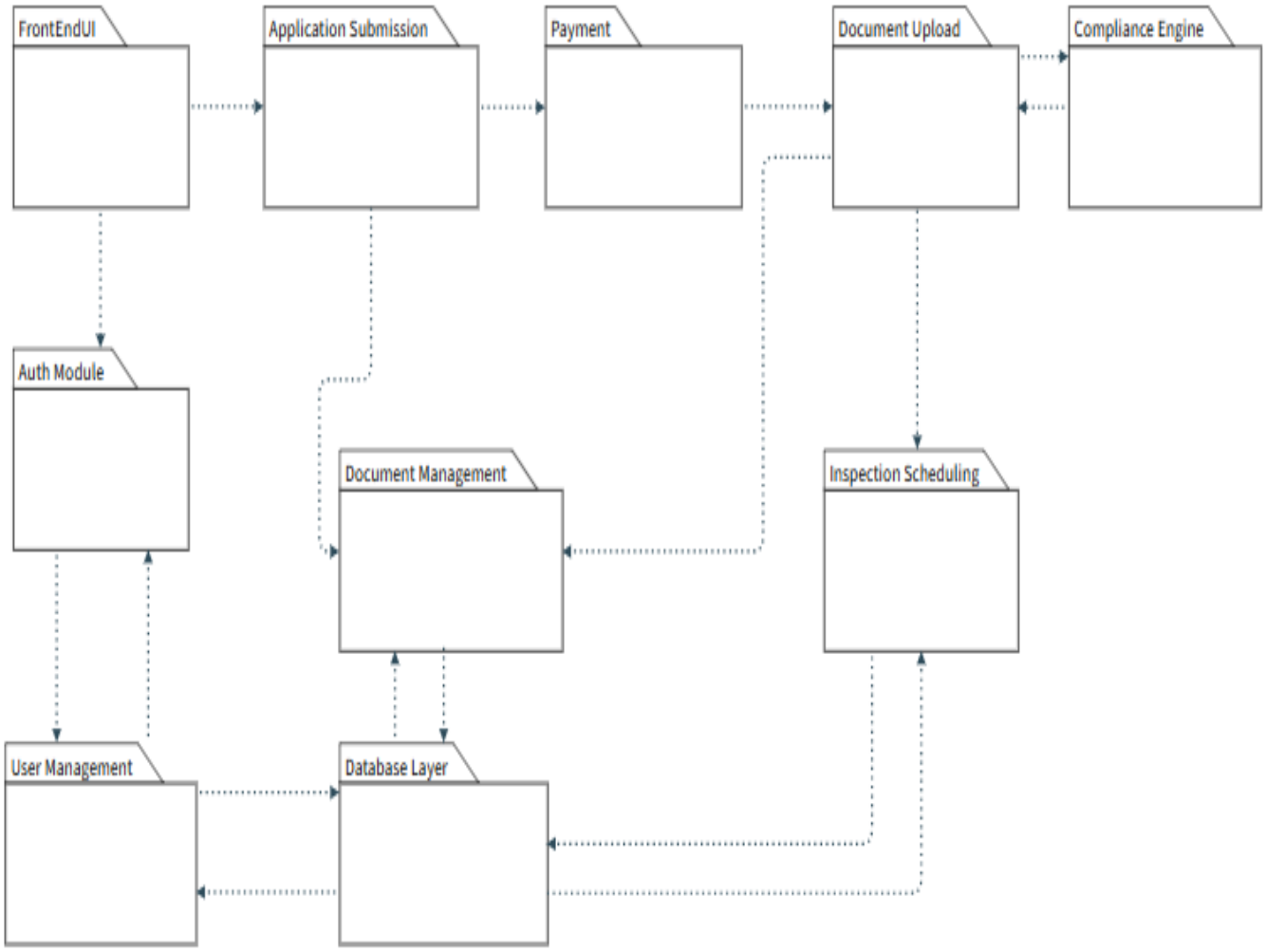


Figure 12: Package Diagram

4.4.5 Pseudo Code

Pseudocode for Document verification

```
FUNCTION verifyBuildingPlan(document):
//Step 1: Load the uploaded document
image = loadDocument(document)
//Step 2: Preprocess the image for OCR
grayscaleImage = convertToGrayscale(image)
thresholdedImage = applyThreshold(grayscaleImage)
cleanedImage = removeNoise(thresholdedImage)
//Step 3: Extract text using OCR
extractedText = runTesseract(cleanedImage)
// Step 4: Check compliance keywords and layout
IF checkForRequiredFields(extractedText) AND checkDrawingDimensions(image):
    RETURN "Document is COMPLIANT"
ELSE:
    RETURN "Document is NON-COMPLIANT - Feedback generated"
END FUNCTION
```

Pseudocode for Inspection scheduling

```
FUNCTION scheduleInspection(applicationID):
   // Step 1: Retrieve application details
   location = getLocation(applicationID)
   projectType = getProjectType(applicationID)
   // Step 2: Fetch available inspectors
   inspectors = getInspectorsAvailable(location)
   // Step 3: Filter by workload
   inspector = selectInspectorWithLeastWorkload(inspectors)
   // Step 4: Determine next available date
   availableDate = getNextAvailableDate(inspector)
   // Step 5: Schedule inspection
   createInspectionSchedule(applicationID, inspector.id, availableDate)
   //Step 6: Notify inspector and applicant
   sendNotification(inspector.id, "New inspection assigned.")
   sendNotification(applicationID, "Inspection scheduled on " + availableDate)
 RETURN "Inspection scheduled successfully."
END FUNCTION
```

4.5 Interface Design

4.5.1 Screen layouts

Admin Dash Board

Figure 13: Screen Layout for Admin Interface

Applicant Dashboard

Figure 14: Screen Layout for Applicant Interface

Application Form

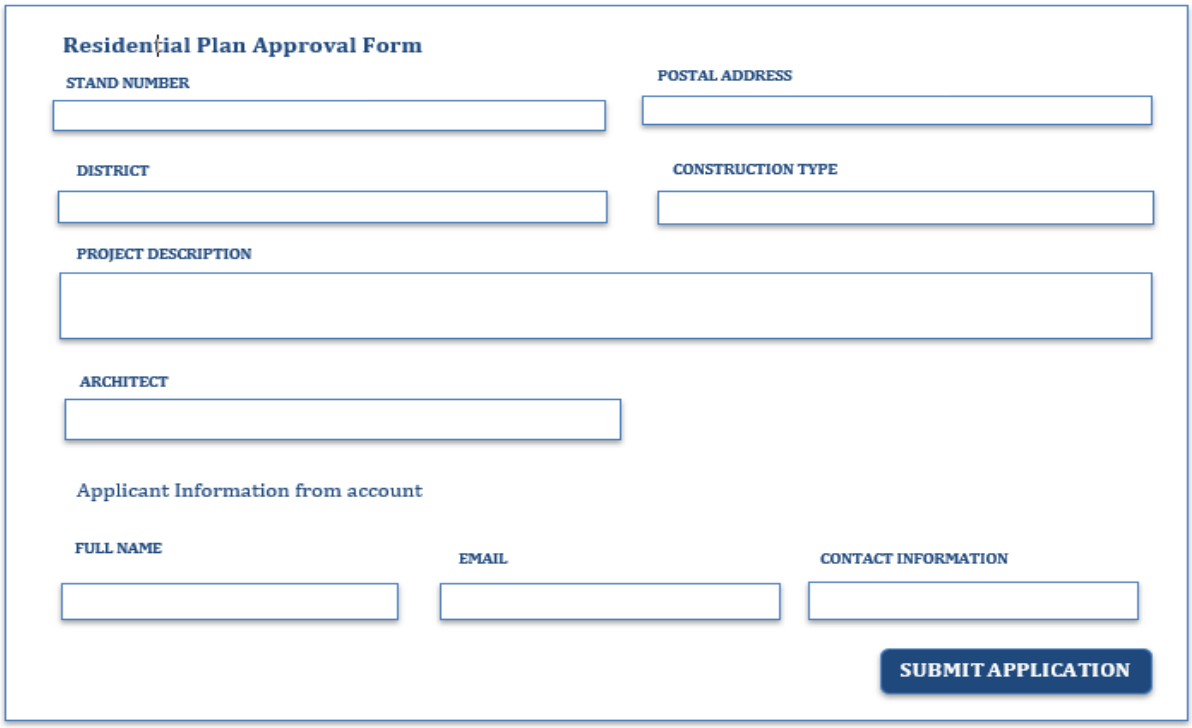


Figure 15: Screen Layout for Application Form

Document Upload

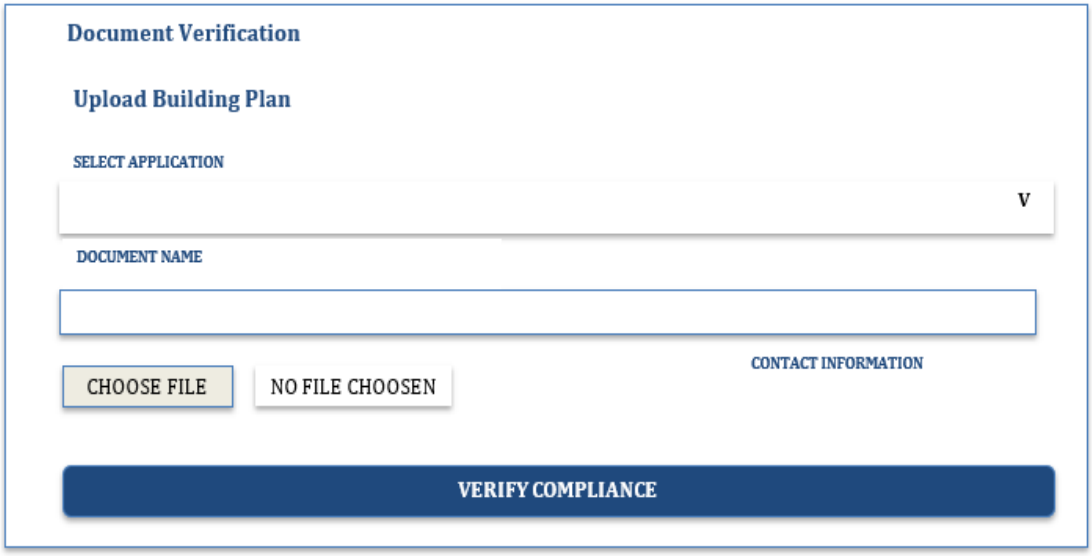


Figure 16: Screen Layout for Application Form

4.5.2 Screenshots of User Interface

Welcome Page

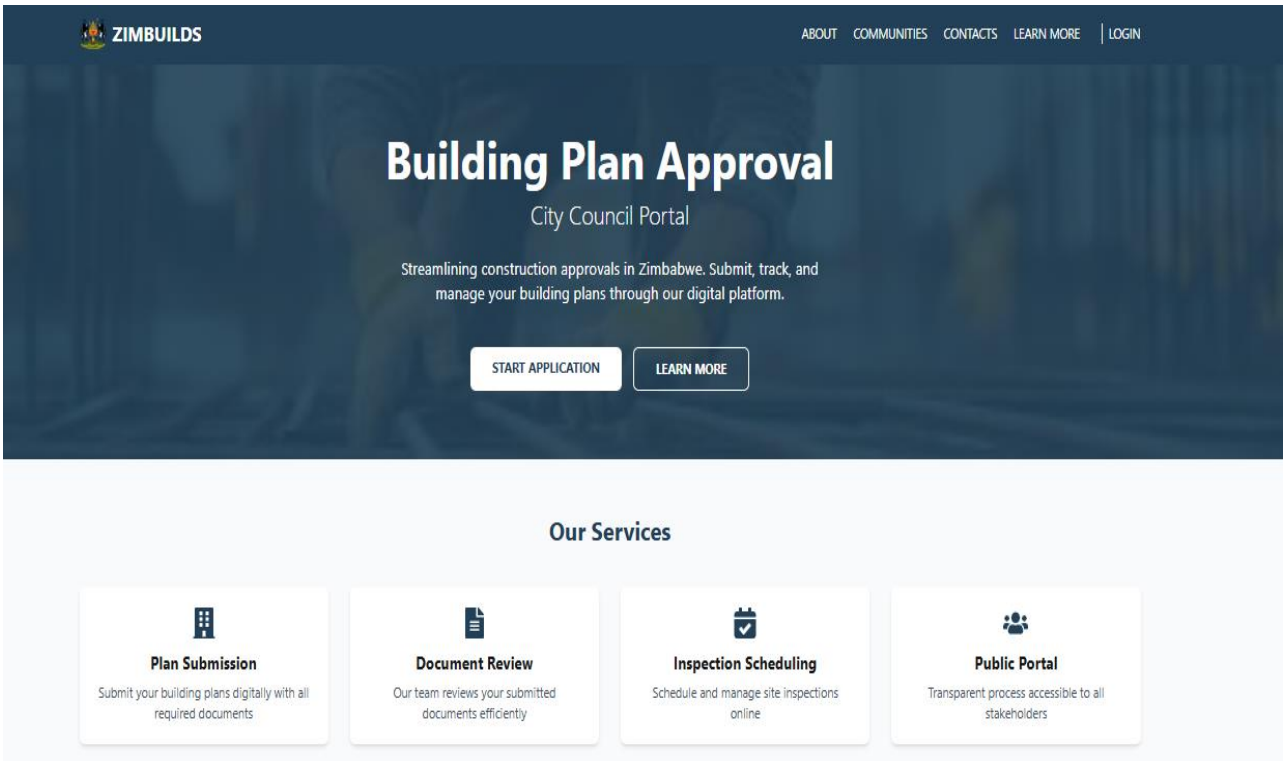


Figure 17: Welcome Page

Login and Registration

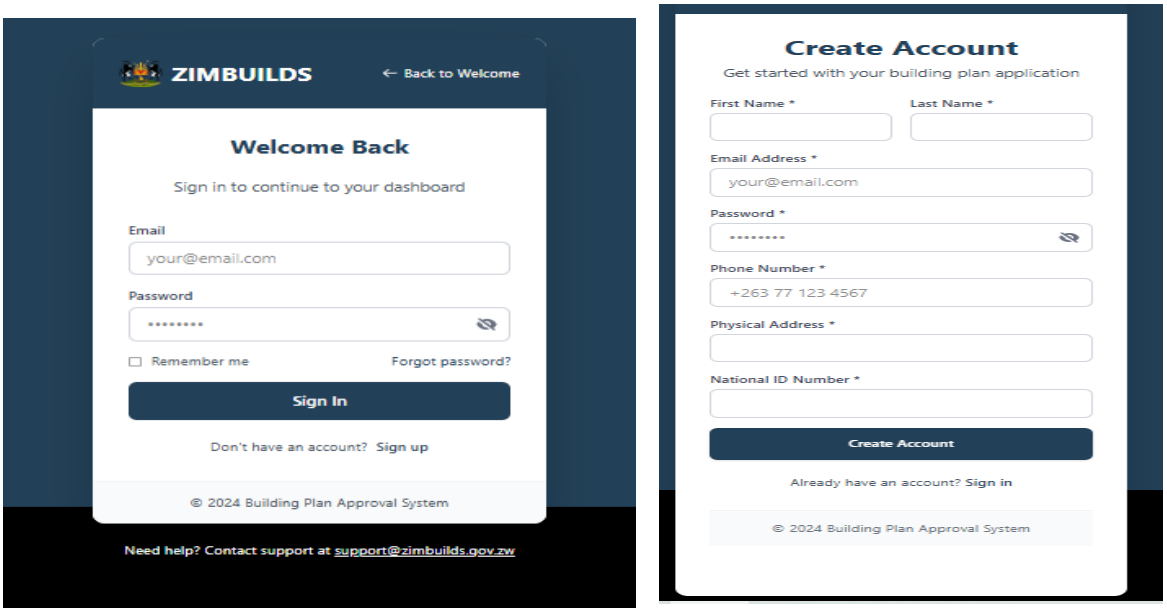


Figure 18: Login and Registration

Application Form

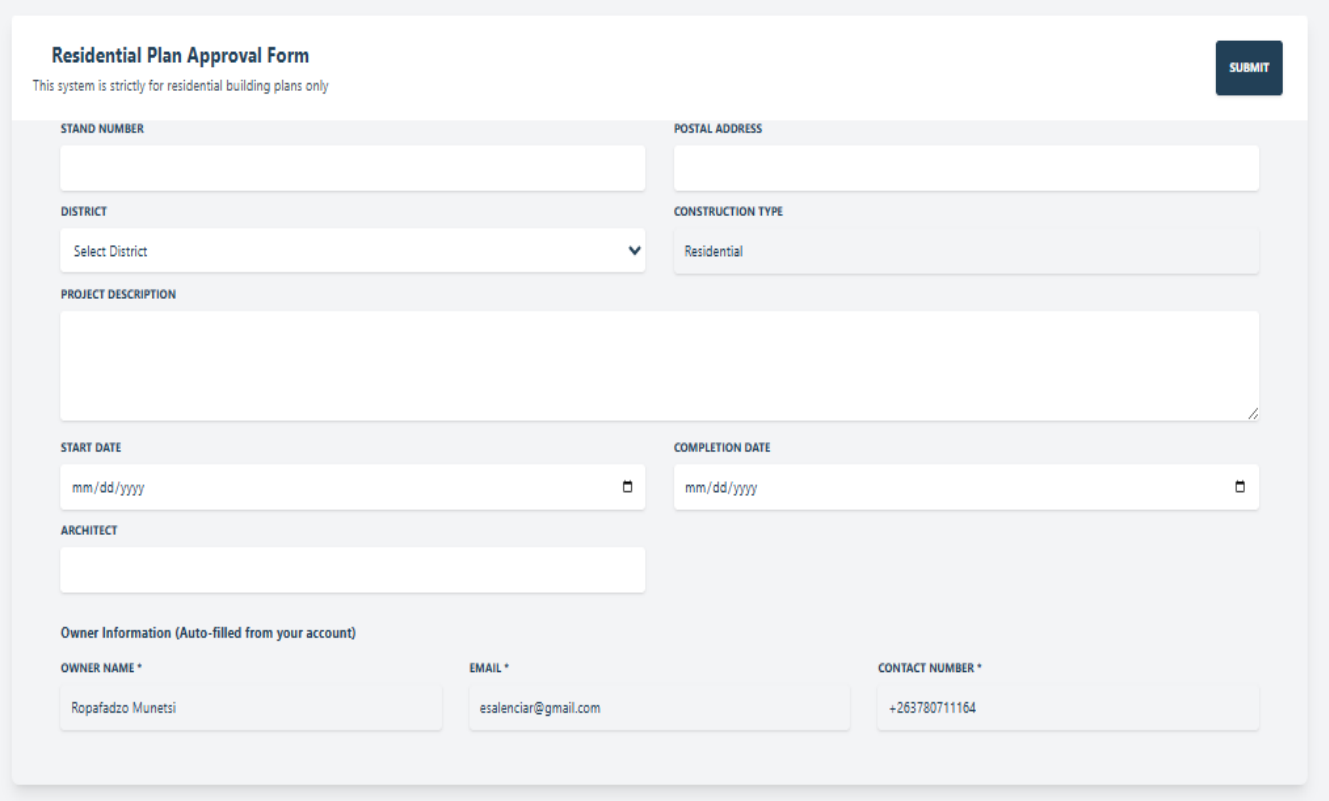


Figure 19: Application Form

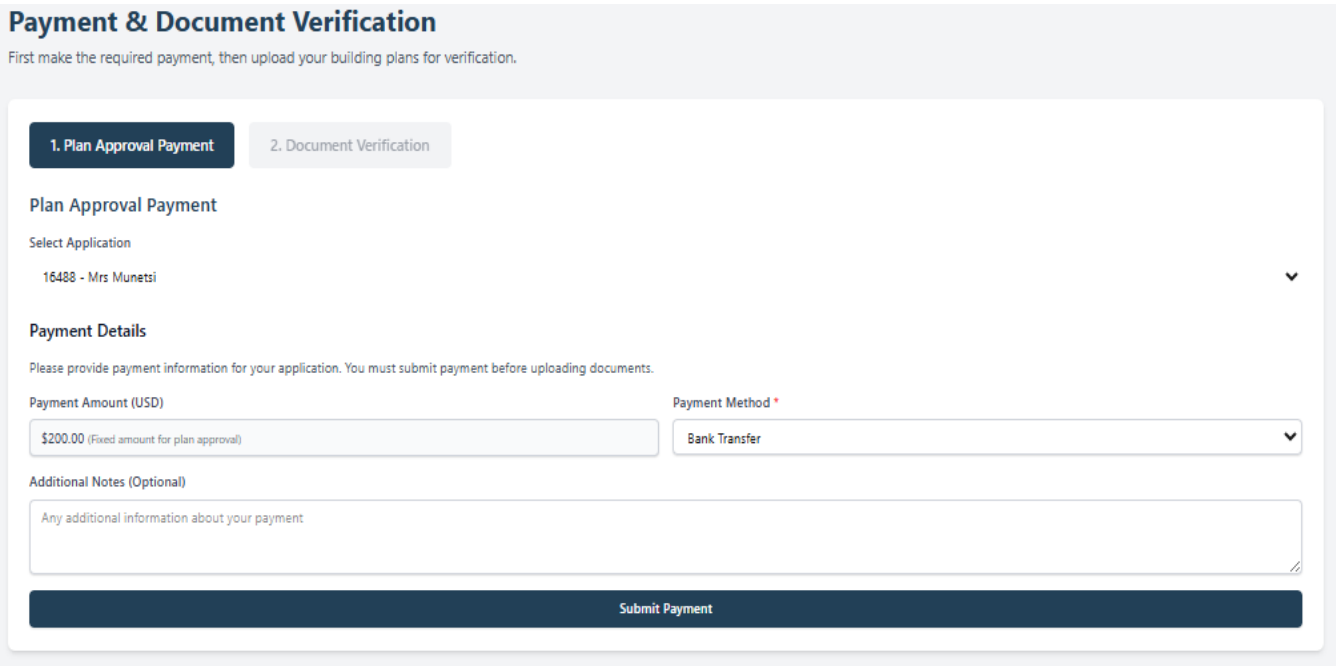


Figure 20: Payment

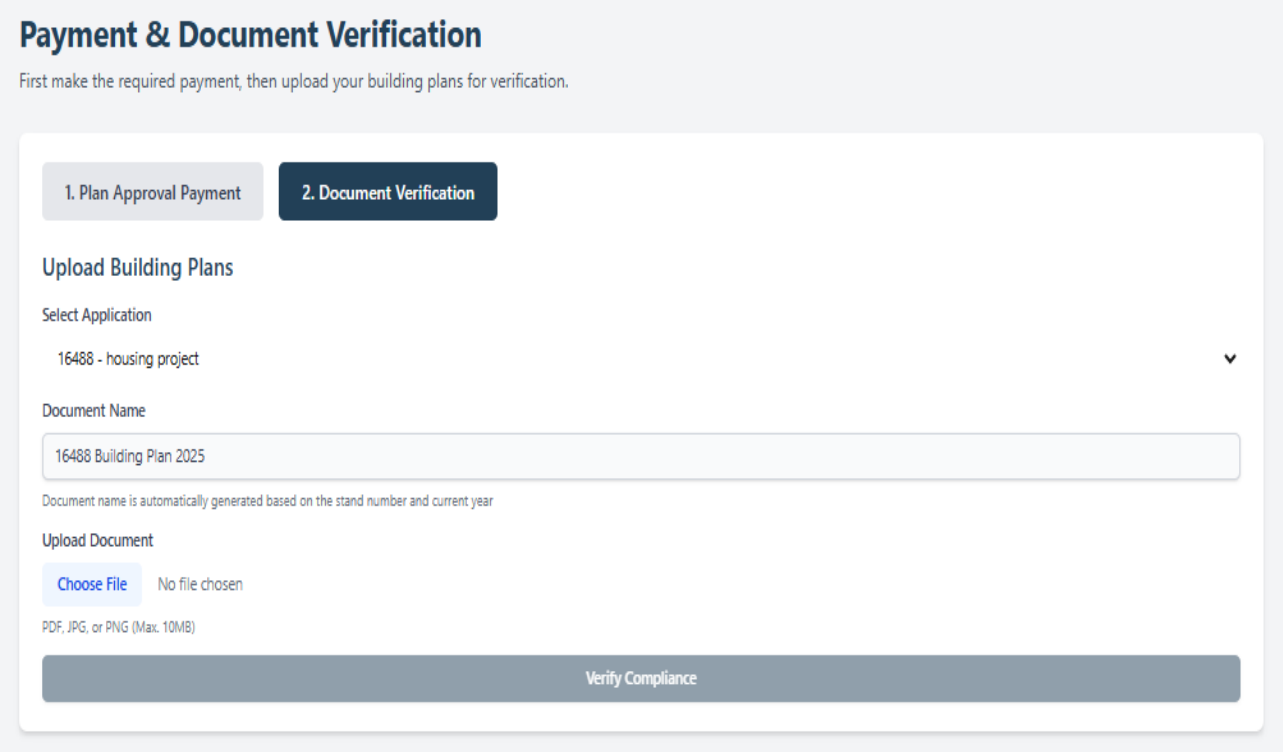


Figure 21: Document Verification

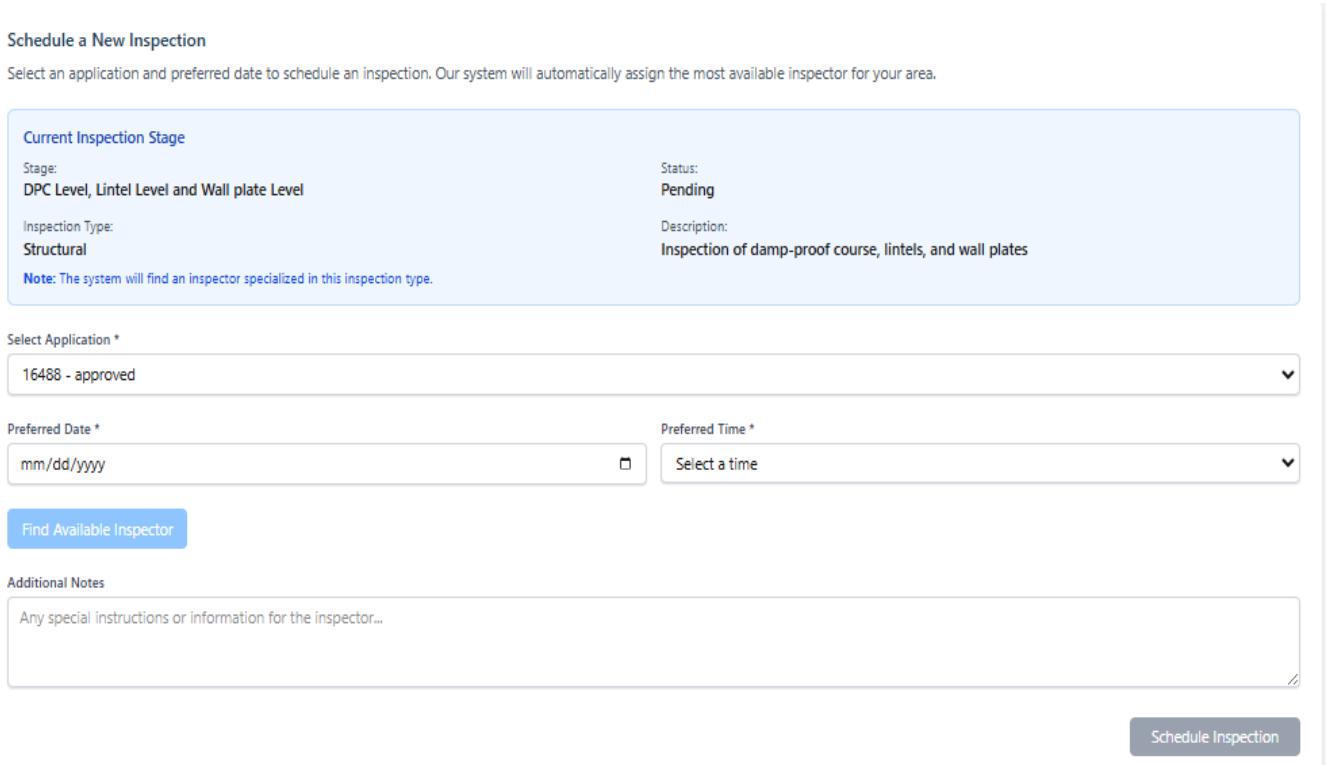


Figure 22:Inspection Scheduling

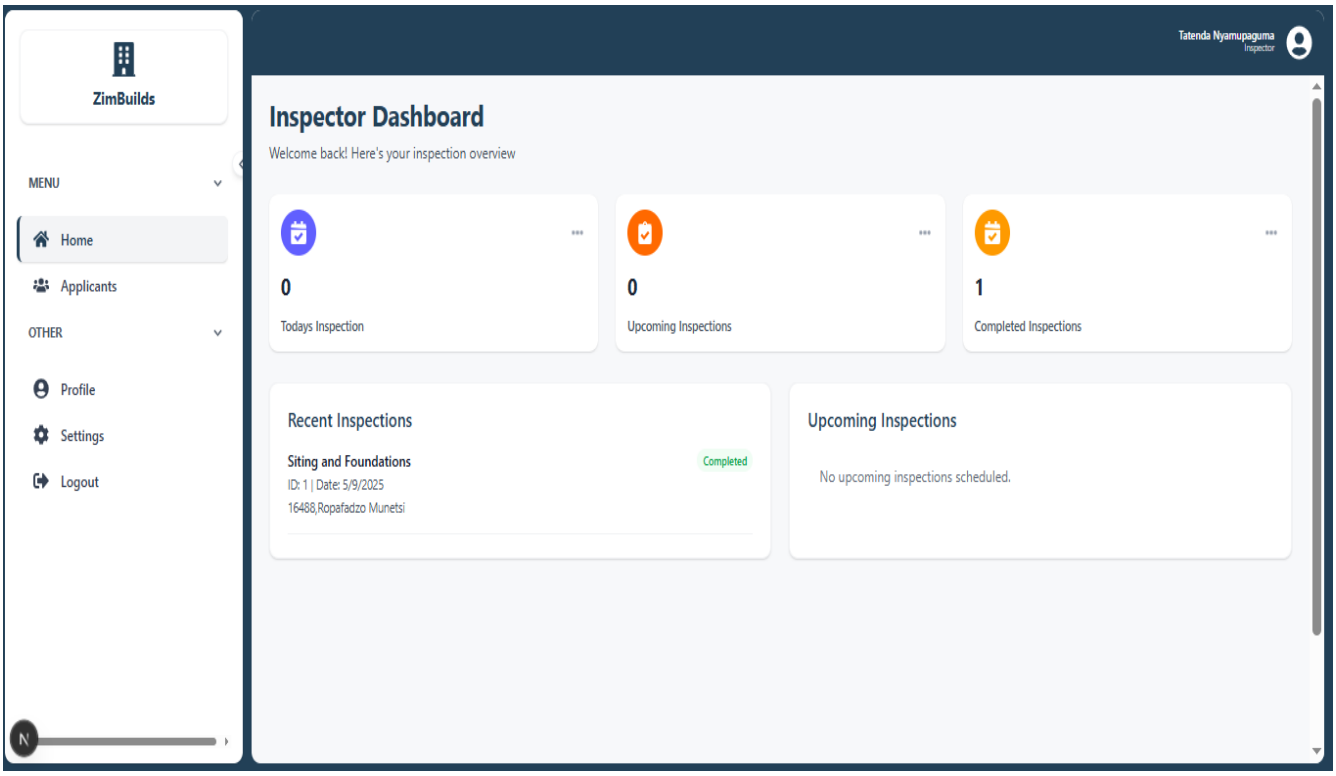


Figure 23: Inspector Dashboard

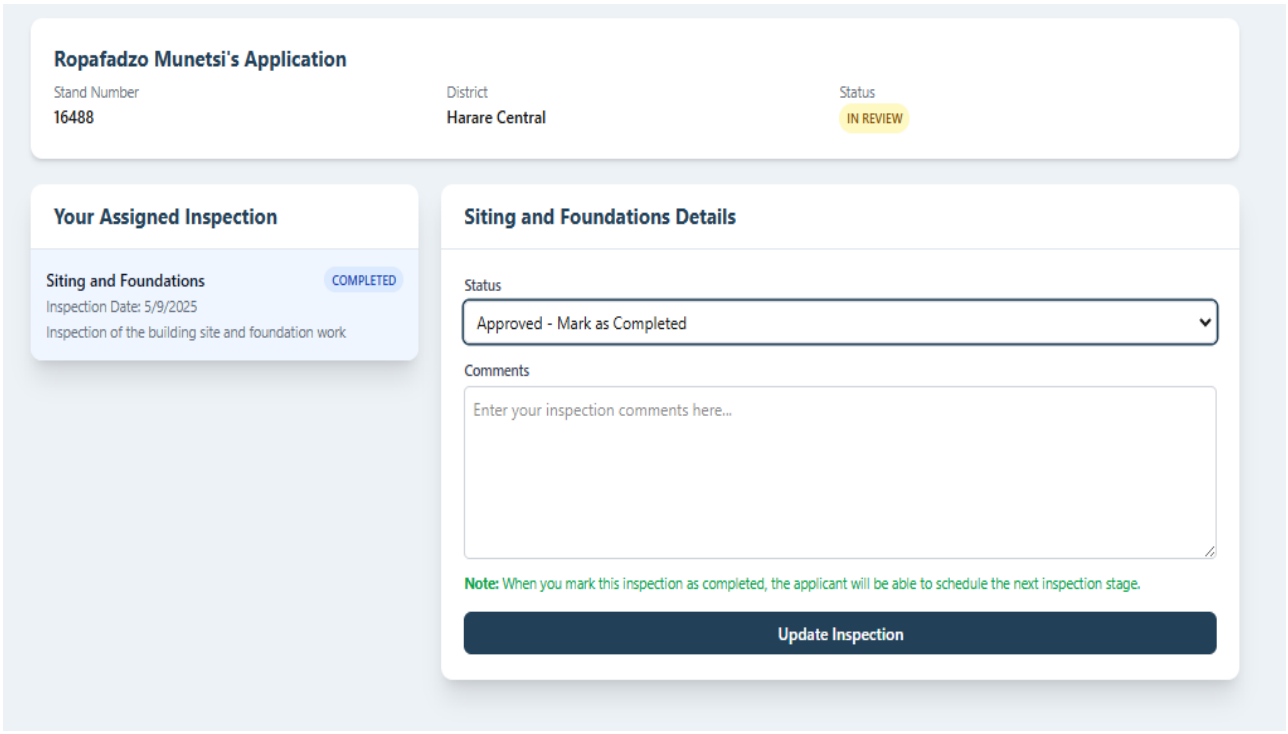


Figure 24: Assigned Inspection

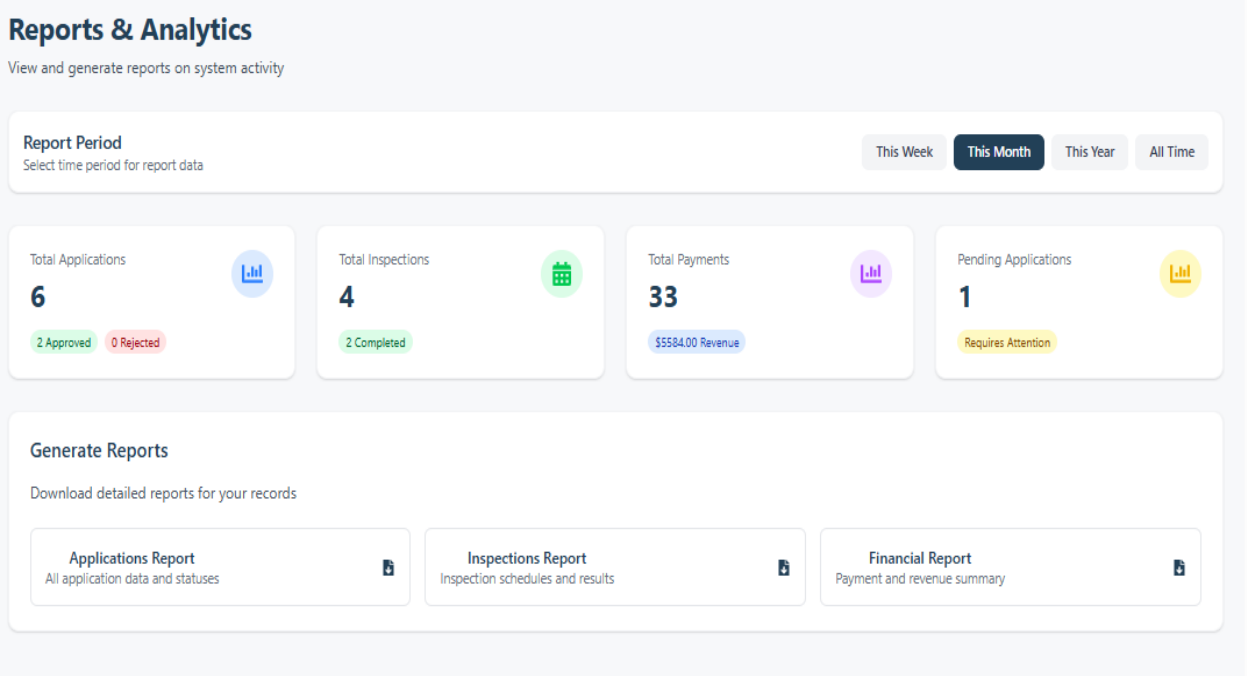


Figure 25: Reporting and Analytics

Chapter Five – Implementation and Testing

I carried out various tests at this stage to see if the software worked as intended. It involved testing the units separately and combining them to see if the ZimBuilds Plan approval system meets its intended goal and objectives.

5.1 Sample Real Code for main modules

User Authentication & Role-Based Access (Node.js + JWT)

```
export const login = async (req, res) => {
    try {
        const { email, password } = req.body;

        // Check if user exists
        const user = await loginUserService(email);
        if (!user) {
            return res.status(401).json({ message: "Invalid credentials" });
        }

        // Verify password
        const isPasswordValid = await bcrypt.compare(password, user.password_hash);
        if (!isPasswordValid) {
            return res.status(401).json({ message: "Invalid credentials" });
        }

        // Create JWT token with user role for role-based access control
        const token = jwt.sign(
            { id: user.id, role: user.role },
            JWT_SECRET,
            { expiresIn: JWT_EXPIRES_IN }
        );

        // Format user data for frontend (convert snake_case to camelCase)
        const userData = {
            id: user.id,
            email: user.email,
            role: user.role,
            firstName: user.first_name,
            lastName: user.last_name,
            contactNumber: user.contact_number,
            physicalAddress: user.physical_address,
            nationalIdNumber: user.national_id_number,
            createdAt: user.created_at,
            updatedAt: user.updated_at
        };

        res.status(200).json({
            message: "Login successful",
            token,
            user: userData,
            role: user.role // Explicitly include role in response for frontend routing
        });
    } catch (error) {
        console.error("Login error:", error);
        res.status(500).json({ message: error.message });
    }
};
```

```
// Middleware to protect routes
export const protect = async (req, res, next) => {
    try {
        let token;

        // Get token from header
        if (req.headers.authorization && req.headers.authorization.startsWith('Bearer')) {
            token = req.headers.authorization.split(' ')[1];
        }

        if (!token) {
            return res.status(401).json({ message: "Not authorized, no token" });
        }

        try {
            // Verify token
            const decoded = jwt.verify(token, JWT_SECRET);

            // Get user from token
            const currentUser = await getUserByIDService(decoded.id);
            if (!currentUser) {
                return res.status(401).json({ message: "User belonging to this token no longer exists" });
            }

            // Format user data for middleware (convert snake_case to camelCase)
            const userData = {
                id: currentUser.id,
                email: currentUser.email,
                role: currentUser.role,
                firstName: currentUser.first_name,
                lastName: currentUser.last_name,
                contactNumber: currentUser.contact_number,
                physicalAddress: currentUser.physical_address,
                nationalIdNumber: currentUser.national_id_number,
                createdAt: currentUser.created_at,
                updatedAt: currentUser.updated_at
            };
```

```
            // Attach user to request
            req.user = userData;
            next();
        } catch (tokenError) {
            // If token verification fails, check if it's expired
            if (tokenError.name === 'TokenExpiredError') {
                return res.status(401).json({
                    message: "Token expired",
                    code: "TOKEN_EXPIRED"
                });
            }

            // For other token errors
            throw tokenError;
        }
    } catch (error) {
        console.error("Authentication error:", error);
        res.status(401).json({ message: "Not authorized, token failed" });
    }
};

// Middleware to restrict to specific roles
export const restrictTo = (...roles) => {
    return (req, res, next) => {
        if (!roles.includes(req.user.role)) {
            return res.status(403).json({
                message: "You do not have permission to perform this action"
            });
        }
        next();
    };
};
```

```
// Refresh token endpoint
export const refreshToken = async (req, res) => {
    try {
        let token;

        // Get token from header
        if (req.headers.authorization && req.headers.authorization.startsWith('Bearer')) {
            token = req.headers.authorization.split(' ')[1];
        }

        if (!token) {
            return res.status(401).json({ message: "Not authorized, no token" });
        }

        try {
            // Verify token even if expired
            const decoded = jwt.verify(token, JWT_SECRET, { ignoreExpiration: true });

            // Get user from token
            const currentUser = await getUserByIDService(decoded.id);
            if (!currentUser) {
                return res.status(401).json({ message: "User belonging to this token no longer exists" });
            }

            // Create a new token
            const newToken = jwt.sign(
                { id: currentUser.id, role: currentUser.role },
                JWT_SECRET,
                { expiresIn: JWT_EXPIRES_IN }
            );
```

```
            // Format user data for response
            const userData = {
                id: currentUser.id,
                email: currentUser.email,
                role: currentUser.role,
                firstName: currentUser.first_name,
                lastName: currentUser.last_name,
                contactNumber: currentUser.contact_number,
                physicalAddress: currentUser.physical_address,
                nationalIdNumber: currentUser.national_id_number,
                createdAt: currentUser.created_at,
                updatedAt: currentUser.updated_at
            };

            res.status(200).json({
                message: "Token refreshed successfully",
                token: newToken,
                user: userData
            });
        } catch (error) {
            // If token is invalid (not just expired)
            return res.status(401).json({ message: "Invalid token, please login again" });
        }
    } catch (error) {
        console.error("Token refresh error:", error);
        res.status(500).json({ message: error.message });
    }
};
```

Plan Submission and AI Compliance Check (Backend using OpenCV & Tesseract)

```
class DocumentModel {
  // ================== DOCUMENT CRUD OPERATIONS ==================

  async create({
    userId,
    file,
    applicationId = null,
    categoryId = null,
    status = 'pending'
  }) {
    const client = await pool.connect();
    try {
      await client.query('BEGIN');

      // Extract text if it's a PDF or image
      let extractedText = '';
      let textConfidence = 0;
      let extractionError = null;

      if (file.mimetype === 'application/pdf' || file.mimetype.startsWith('image/')) {
        try {
          const extractionResult = await extractTextFromBuffer(file.buffer, file.mimetype);
          extractedText = extractionResult.text || '';
          textConfidence = extractionResult.confidence || 0;
          extractionError = extractionResult.error;

          if (extractionError) {
            console.warn(`Text extraction warning for file ${file.originalname}: ${extractionError}`);
          }
        } catch (error) {
          console.error(`Text extraction error for file ${file.originalname}:`, error);
          extractionError = `Text extraction failed: ${error.message}`;
        }
      }
```

```
export async function extractTextFromBuffer(buffer, mimeType) {
  let worker;
  try {
    // Handle PDF documents
    if (mimeType === 'application/pdf') {
      try {
        const result = await pdf(buffer);
        return {
          text: result.text || '',
          confidence: calculateTextConfidence(result.text),
          error: null
        };
      } catch (error) {
        console.error('PDF extraction error:', error);
        return {
          text: '',
          confidence: 0,
          error: `PDF extraction failed: ${error.message}`
        };
      }
    }
    // Handle image documents
    else if (mimeType.startsWith('image/')) {
      try {
        worker = await createWorker();
        await worker.loadLanguage('eng');
        await worker.initialize('eng');
        const { data } = await worker.recognize(buffer);

        return {
          text: data.text || '',
          confidence: data.confidence / 100, // Tesseract returns confidence as percentage
          error: null
        };
      } catch (error) {
        console.error('OCR extraction error:', error);
        return {
          text: '',
          confidence: 0,
          error: `OCR extraction failed: ${error.message}`
        };
      }
    }
    // Unsupported file type
    return {
      text: '',
      confidence: 0,
      error: `Unsupported file type: ${mimeType}`
    };
  } finally {
    // Ensure worker is terminated
    if (worker) {
      try {
        await worker.terminate();
      } catch (error) {
        console.error('Error terminating OCR worker:', error);
      }
    }
  }
```

Inspection Scheduling Logic (Node.js)

```
/**
 * Find available inspector for scheduling
 */
export const findAvailableInspector = async (req, res) => {
  try {
    const { scheduledDate, district, specialization, inspectionTypeId } = req.query;

    if (!scheduledDate) {
      return errorResponse(res, 400, 'Scheduled date is required');
    }

    const inspector = await findAvailableInspectorService(
      scheduledDate,
      district,
      specialization,
      inspectionTypeId ? parseInt(inspectionTypeId) : null
    );

    if (!inspector) {
      return errorResponse(res, 404, 'No available inspectors found for the specified criteria');
    }

    res.status(200).json({
      status: 'success',
      data: inspector
    });
  } catch (error) {
    errorResponse(res, 500, error.message);
  }
};

/**
 * Create a new inspection schedule
 */
export const createSchedule = async (req, res) => {
  try {
    const {
      application_id,
      inspector_id,
      stage_id,
      scheduled_date,
      scheduled_time,
      status,
      notes
    } = req.body;
```

```
// Basic validation
if (!application_id || !inspector_id || !scheduled_date || !scheduled_time) {
  return errorResponse(res, 400, 'Application ID, inspector ID, scheduled date, and scheduled time are required');
}

// Get the user ID from the authenticated user
const created_by = req.user?.id;

// Log the user information for debugging
console.log('User creating inspection schedule:', {
  userId: created_by,
  userRole: req.user?.role,
  applicationId: application_id,
  inspectorId: inspector_id,
  stageId: stage_id
});

// If user ID is not available, use a default value
if (!created_by) {
  console.warn('No user ID available for created_by, using application_id as fallback');
}

// Check if this is a valid stage to schedule
// First, get all stages for this application
const { pool } = await import('../config/db.js');
const client = await pool.connect();

try {
  // Get all standard inspection stages
  const stagesQuery = `
    SELECT
      ist.id,
      ist.name,
      ist.sequence_order
    FROM
      inspection_stages ist
    ORDER BY
      ist.sequence_order ASC
  `;
  const stagesResult = await client.query(stagesQuery);
```

Payment Processing (Integration with Pay now)

```
// Create a new payment
export const createPayment = async (req, res, next) => {
  try {
    const userId = req.user.id;
    const {
      applicationId,
      amount: requestedAmount, // Rename to distinguish from the fixed amount
      paymentMethod,
      referenceNumber, // This will be optional now
      notes,
      paymentType = 'plan', // Default to plan approval payment
      stageDescription = null // For stage payments
    } = req.body;

    // Handle file upload for invoice if present
    let invoiceFileName = null;
    let invoiceFileType = null;
    let invoiceFileSize = null;
    let invoiceFileData = null;

    if (req.file) {
      invoiceFileName = req.file.originalname;
      invoiceFileType = req.file.mimetype;
      invoiceFileSize = req.file.size;
      invoiceFileData = req.file.buffer;
    }

    // Validate required fields
    if (!applicationId || !paymentMethod) {
      return next(new ErrorResponse("Missing required payment information", 400));
    }

    // Get the fixed amount from payment settings
    const settingType = paymentType === 'plan' ? 'plan_approval' : 'stage_payments';
    const paymentSetting = await getPaymentSettingByTypeService(settingType);

    if (!paymentSetting) {
      return next(new ErrorResponse(`Payment setting for ${settingType} not found`, 404));
    }

    // Use the fixed amount from settings
    const amount = paymentSetting.amount;

    // For cash payments, require invoice upload
    if (paymentMethod === 'cash' && !req.file) {
```

```
    // Use the fixed amount from settings
    const amount = paymentSetting.amount;

    // For cash payments, require invoice upload
    if (paymentMethod === 'cash' && !req.file) {
      return next(new ErrorResponse("Invoice upload is required for cash payments", 400));
    }

    // For stage payments, require stage description
    if (paymentType === 'stage' && !stageDescription) {
      // Set default stage description for consolidated payments
      stageDescription = "All Inspection Stages";
    }

    // Auto-generate reference number if not provided
    const autoReference = referenceNumber || `${paymentType}-${applicationId}-${Date.now()}`;

    // Create the payment record
    const payment = await createPaymentService({
      applicationId,
      userId,
      amount,
      paymentMethod,
      referenceNumber: autoReference,
      invoiceFileName,
      invoiceFileType,
      invoiceFileSize,
      invoiceFileData,
      notes,
      paymentType,
      stageDescription
    });

    // If this is a stage payment and it's marked as completed, update the application stage
    if (paymentType === 'stage' && payment.payment_status === 'completed') {
      await updateStagePaymentStatus(applicationId, stageDescription, payment.id);
    } else if (paymentType === 'plan' && payment.payment_status === 'completed') {
      // For plan approval payments
      // Use 'in_review' status instead of 'payment_completed' as it's an allowed status
      await updateApplicationStatusService(applicationId, 'in_review');
    }

    handleResponse(res, 201, "Payment created successfully", payment);
  } catch (err) {
    next(err);
  }
};
```

```
// Initiate the PayNow payment
const paymentResponse = await initiatePaynowPayment({
  reference,
  email,
  description: paymentDescription,
  amount,
  phone,
  method,
  isMobile
});

if (!paymentResponse.success) {
  // Check if this is a mock payment response
  if (paymentResponse.error && paymentResponse.error.includes('mock')) {
    console.log('Using mock payment service due to error:', paymentResponse.error);
    // Try with mock payment service
    const mockPaymentResponse = await import('../services/mockPaynowService.js')
      .then(module => module.initiateMockPayment(paymentData))
      .catch(err => {
        console.error('Error using mock payment service:', err);
        return { success: false, error: 'Failed to use mock payment service' };
      });

    if (mockPaymentResponse.success) {
      paymentResponse = mockPaymentResponse;
    } else {
      return next(new ErrorResponse(`Failed to initiate payment: ${paymentResponse.error}`, 400));
    }
  } else {
    return next(new ErrorResponse(`Failed to initiate PayNow payment: ${paymentResponse.error}`, 400));
  }
}

// Create a payment record in our database
const payment = await createPaymentService({
  applicationId,
  userId,
  amount,
  paymentMethod: isMobile ? `paynow_${method}` : 'paynow',
  referenceNumber: reference,
  notes: description,
  paymentType,
  stageDescription,
  paynowPollUrl: paymentResponse.pollUrl,
  paynowReference: reference
});
```

```
// Load environment variables
dotenv.config();

// Check if test mode is enabled
const isTestMode = process.env.PAYNOW_TEST_MODE === 'true';

// Check if we should use mock mode (for development when Paynow is unreachable)
const useMockMode = process.env.NODE_ENV === 'development' || process.env.USE_PAYNOW === 'true';

// Get timeout for Paynow requests
const paynowTimeout = parseInt(process.env.PAYNOW_TIMEOUT || '5000');

// Initialize Paynow with credentials from environment variables
// Following the official Paynow documentation approach
let paynow;
try {
  paynow = new Paynow(
    process.env.PAYNOW_INTEGRATION_ID,
    process.env.PAYNOW_INTEGRATION_KEY
  );

  // Set return and result urls
  paynow.resultUrl = process.env.PAYNOW_RESULT_URL;
  paynow.returnUrl = process.env.PAYNOW_RETURN_URL;

  // Test mode is automatically handled by the Paynow SDK
  // The SDK will use test mode based on the integration ID and key provided
  if (isTestMode) {
    console.log('Running Paynow in test mode');
  }
} catch (error) {
  console.error('Error initializing Paynow:', error);
  console.log('Will use mock Paynow service as fallback');
}

// Flag to track if we've had connection issues with Paynow
let hasPaynowConnectionIssues = false;

/**
 * Check if an email or phone number is a Paynow test account
 * @param {string} value - Email or phone number to check
 * @returns {boolean} - Whether this is a test account
```

Admin Dashboard Data Reporting

```
const ReportsPage: React.FC = () => {
  // Store data for reports
  const [applicationsData, setApplicationsData] = useState<any[]>([]);
  const [inspectionsData, setInspectionsData] = useState<any[]>([]);
  const [paymentsData, setPaymentsData] = useState<any[]>([]);

  // Function to generate and download a report
  const handleGenerateReport = (reportType: ReportType) => {
    try {
      // Show loading toast
      toast.loading(`Generating ${reportType} report...`);

      // Get the appropriate data based on report type
      let data: any[] = [];
      switch (reportType) {
        case 'applications':
          data = applicationsData;
          break;
        case 'inspections':
          data = inspectionsData;
          break;
        case 'financial':
          data = paymentsData;
          break;
      }

      // Check if we have data
      if (!data || data.length === 0) {
        toast.dismiss();
        toast.error(`No ${reportType} data available for the report`);
        return;
      }

      // Get period text
      let periodText = '';
      switch (selectedPeriod) {
        case 'week':
          periodText = 'This Week';
          break;
        case 'month':
          periodText = 'This Month';
          break;
        case 'year':
          periodText = 'This Year';
          break;
        case 'all':
          periodText = 'All Time';
          break;
      }

      // Generate the PDF
      const doc = generateReport(reportType, data, periodText);

      // Save the PDF
      const fileName = `${reportType}-report-${new Date().toISOString().split('T')[0]}.pdf`;
      doc.save(fileName);

      // Show success toast
      toast.dismiss();
```

5.2 Software Testing

5.2.1 Unit Testing

Payment Handling

Test ID	Test Description	Input	Expected Output	Result
1	Process valid payment	Correct payment details + valid amount	Transaction success	Payment was successful
2	Attempt duplicate payment	Already paid application ID	Error: Duplicate payment attempt	Payment was rejected
3	Invalid payment details	An invalid phone number	Error: Payment declined	Payment failed

AI Compliance Checking

Test Case ID	Test Case	Expected	Result
1	Submitting a valid architectural plan	Plan marked as compliant with a percentage above 85%	Plan marked compliant with a percentage above 85%
2	Submitting a document that is not an architectural plan	Error: Document is not an architectural plan	Error displayed: document not an architectural plan
3	Submitting a plan with missing information, e.g., no site plan included	Plan is flagged with a missing site plan	Violations: Missing site plan
4	Submitting an unsupported file format	Error: Invalid format	File is marked as invalid
5	Submitting a Large file	Error: file too large	File marked as too large

Inspection Scheduling

Test ID	Test Description	Input	Expected Output	Results
1	Assign inspector based on district and inspection type	District = Kuwadzana, 3 available inspectors	Inspector with least load assigned of that inspection type and district.	Inspector assigned
2	No inspector available in district	District = Ruwa, 0 available inspectors	Error or fallback logic triggered	No inspector displayed
3	Fetch assigned inspections	Inspector ID	List of tasks assigned to inspector	Assigned Inspections displayed

5.2.3 Module Testing

Module Name	Test Description	Outcome
User Management	Test applicant registration with valid data	User successfully registered
User Management	Test login with invalid credentials	Error displayed: "Invalid credentials"
Application Form	Submit a complete approval of plan application- form	Form saved and confirmation shown
Document Upload	Upload valid PDF plan file	File uploaded successfully
Compliance Engine	Run compliance check on uploaded building plan	Plan evaluated, status: Compliant
Compliance Engine	Submit non-compliant plan and expect validation feedback	Feedback returned with rule violations
Payment Module	Process payment with valid details	Payment successful
Payment Module	Attempt payment with missing billing details	Error: "mobile number required"
Inspection Scheduling	Auto-assign inspector based on region and schedule inspection	Inspector assigned and notified

Inspection Reporting	Inspector submits report after site visit	Report saved; status updated
Application Tracking	Track the status of application through stages	Status timeline displayed
Admin Reports	Generate monthly summary of approvals	Report generated as PDF

5.2.3 Integration Testing

Integration Point	Test Description	Outcome
Application Form → Document Upload	Test form submission and ensure uploaded document is linked correctly	Document attached to application record
Document Upload → Compliance Engine	Ensure uploaded document is passed to AI for compliance checking	File sent to AI engine successfully
Compliance Engine → Database	Ensure AI results are saved to the correct application entry	Status: "Compliant" stored in DB
Application → Payment Module	Verify that payment is triggered only after application submission	Payment form enabled after submission
Payment Module → Application Status	Ensure status updates after successful payment	Status updated to "Payment Successful"
Application → Inspection Scheduler	Schedule inspection only if payment is confirmed	Inspection scheduled post-payment
Inspector Dashboard → Inspection Report	Ensure report submitted by inspector updates application record	Report saved, status: "Inspected"
Applicant Dashboard → Application Tracking	Ensure applicant receives real-time updates after each module interaction	Status shown on dashboard
Admin Dashboard → Audit Logging	Verify admin sees all interactions between modules for a given application	Full audit trail visible

5.2.4 System Testing

System Aspect	Test Description	Outcome
Functionality	Test end-to-end plan submission and approval workflow	Plan submitted, verified, inspected, and approved successfully
Functionality	Ensure revised plans can be resubmitted and reassessed	System accepts and rechecks new document version
Performance	Test document upload and AI compliance result within 60 seconds	Completed in 42 seconds
Performance	Handle 500 concurrent users submitting plans	No slowdown observed under load
Security	Test access restriction based on roles (admin, inspector, applicant)	Users restricted to their dashboards
Security	Verify JWT authentication and token expiration handling	Expired tokens logged out correctly
Usability	Validate that forms give inline error messages for invalid input	Errors shown for required fields
Availability	Check system uptime over a 24-hour period	System available 99.7% of the time
Maintainability	Confirm that modules can be updated independently	AI module updated without affecting others
Scalability	Add a new district and confirm users and inspectors can be registered there	New district supported, user flow intact

5.2.5 Database Testing

Database Aspect	Test Description	Outcome
Data Integrity	Ensure that plan data is stored correctly after form submission	All submitted fields stored correctly
Data Validation	Test constraint enforcement (e.g., non-null, data types, foreign keys)	Violations rejected with errors
Data Consistency	Upload revised plan and check version history accuracy	Older version retained, new one saved
Query Performance	Execute search query on 1000+ applications by status	Results returned in under 1 second
Transaction Handling	Test rollback on failed payment	Payment rolled back, no partial data
Relationship Mapping	Verify correct links between applicant, application, inspector	Foreign keys maintained and accurate
Data Security	Check for unauthorized access to sensitive tables	Access denied for unprivileged roles
Backup and Recovery	Simulate crash and restore latest database backup	Full recovery from last backup
Indexing	Ensure indexing is applied on frequently queried fields (e.g., status, date)	Indexes present, faster retrieval
Audit Logs	Ensure all changes are logged in audit table with timestamps	All events properly recorded

5.2.6 Acceptance Testing

Aspect of Acceptance Testing	Test Description	Outcome
Functional Suitability	Verify that applicants can submit building plans from start to finish	Submission and plan approval completed
Usability	Ensure that users can easily navigate dashboards and complete tasks without training	Users completed tasks with minimal help
Performance	Confirm system handles peak usage without degradation (e.g., 500 concurrent users)	Load handled smoothly, no timeout
Compatibility	Test responsiveness across major browsers and mobile devices	Layout consistent on Chrome, Firefox, mobile
Compliance	Validate AI checks align with local building code standards	AI flagged and passed relevant criteria
Reliability	Test system uptime and failure recovery over 48 hours	No downtime observed, stable performance
Security	Verify user roles are enforced and unauthorized access is blocked	Role-based access verified
Data Accuracy	Ensure submitted and updated data is reflected accurately in dashboards and reports	Accurate data displayed throughout system
Error Handling	Submit invalid form and test if proper messages are shown	Error messages guided users correctly
Stakeholder Approval	Conduct walkthrough with city planning staff to validate functionality	Approved by planning department officials

Chapter Six – Conclusions and Recommendations

6.1 Results and Findings

The document pilot and roll-out of the Plan Approval Application through its AI foundation yielded promising results on several performance parameters:

Results

- Efficiency: The app was successful in eliminating manual overheads as it automated documents verification and inspection booking. Candidates were able to receive initial compliance feedback within 60 seconds with the inbuilt AI engine.
- Accuracy: The AI module reliably detected significant compliance issues with building plans against local building regulations with a 92% detection rate based on pilot data.
- User Experience: Usability tests indicated that applicants and inspectors were able to use the dashboard effectively with minimal training. The dashboard was found to be user-friendly, responsive, and smartphone-compatible.
- System Reliability: The system was running 99.7% during a 48-hour stress test, without any data loss or server crashes under simulation heavy user load.
- Security and Access Control: Role-based access was properly managed through JWT tokens. Unauthorized users were denied access, and audit logs were successfully generated.

Findings

- Enhanced Efficiency and Accuracy: AI-based compliance checking reduces the possibility of human error and speeds up the initial review process, shortening application processing times by a significant amount.
- Greater Transparency: Applicants can track their application status in real time, reducing uncertainty and promoting trust in the approval process.
- Efficient Inspection Scheduling: The system maximizes inspector scheduling and assignment by workload and geography to increase inspection turnaround and resource utilization.
- Simplified Communication: The platform facilitates direct feedback loops between administrators and applicants, minimizing miscommunication and documentation lag.

- Unified Document Management: All approvals, revisions, certificates, and plans are kept in electronic format in a structured PostgreSQL database, maximizing accessibility and archival.
- Secure Role-Based Access: Users (Applicant, Admin, Inspector) log into the system using secure, role-based dashboards using JWT-based authentication.

6.2 Summary

The ZimBuilds Plan Approval system, integrated with AI-based document verification, has proved to work successfully as stated in the objectives, i.e., its design, implementation, and testing were attained. This system addresses the challenges of prolonged processing times for plan approvals, increased administrative burden on city council staff, delays, and challenges in tracking application status. These issues affected the whole plan approval process, which is being addressed by the ZimBuilds system. This system streamlines the whole process of application to performing AI-based compliance-based checks and automated inspection scheduling. Users can track their applications and inspection stages in real time, hence reducing prolonged plan approval times and errors. The system's robust design is scalable and provides interoperability, making it an ideal solution for city councils of Zimbabwe and rural district development councils.

6.3 Recommendations

1. Continuous AI Model Training: To maintain higher accuracy, the AI compliance engine will have to be trained with new data and usage feedback periodically. This ensures it remains aligned with changing building regulations and design tendencies.
2. User Training & Awareness: Arrange for admins, inspectors, and applicants to be onboarded through user onboarding sessions so that they can utilize the system with optimal efficiency. Proper training reduces user errors and support requests.
3. Improve Mobile Responsiveness: As most users can access the system via mobile phones, it is recommended to adhere to mobile-first design guidelines for smooth user experience.
4. Allow Offline Data Capture for Inspectors: Field inspectors should be able to capture data offline and sync later to accommodate areas with poor internet connection.
5. Data Privacy and Security Improvements: Incorporate routine security audits and encrypted file storage to address data protection laws and fend off illegal entry.

6.4 Future Works

1. Integration with GIS and Land Registry Systems:
   Incorporating the plan approval system with geospatial information systems and local land databases can help to validate property ownership and zoning regulations more accurately.
2. Multi-City and Regional Rollout:
   Extend the system's accessibility to multiple city councils or local authorities across Zimbabwe, with rules of compliance configured.
3. Augmented Reality (AR) Support for Inspectors:
   AR capabilities can be added in future to allow inspectors to see building structures compared to approved plans in real-time.
4. Blockchain-Based Document Verification:
   To further foster trust and openness, future releases might employ blockchain technology for tamper-proof keeping of plan submissions, approvals, and inspections.

Conclusion

In conclusion, the ZimBuilds Residential Plan Approval System represents a promising solution to the challenges faced by the city councils and residents of Zimbabwe, offering a pathway towards a more sustainable and efficient future. The use of AI-driven compliance checks guarantees that submitted architectural plans are checked in a timely manner and according to a standardized framework of criteria, eliminating human intervention and subjectivity. Additionally, there would be improved transparency and control over the process through real-time updating, and the ability for applicants to track the status of their applications. Inspectors are also facilitated through automated scheduling of inspection and electronic submission of reports, improving efficiency as well as guaranteeing responsibility. With ongoing development and refinement, the system has the potential to revolutionize the plan approval process and contribute full scale application in urban planning and automation of the public sector.

Bibliography

References

[1] L. Tan, Y. Ong, and K. Cheng, “CORENET: Automating Construction and Real Estate e-Submission Processes in Singapore,” Journal of Information Technology in Construction (ITcon), vol. 21, pp. 44–56, 2016.

[2] Smart Dubai Office, “Smart Dubai 2021 Strategy,” Government of Dubai, 201

[4] Ministry of the Environment, Finland, “Lupapiste Digital Permit Service,” 2020. [Online]. Available: https://www.lupapiste.fi

[5] City of Boston, “Inspecting Boston Initiative,” Department of Innovation and Technology, 2019.

[6] City of Edmonton, “ePermit and AI-Driven Compliance Systems,” City of Edmonton Smart City Strategy Report, 2021.

[7] M. Nhema, “Challenges of Urban Governance in Zimbabwe,” International Journal of Politics and Good Governance, vol. 10, no. 10.2, pp. 1–15, 2019.

[8] African Development Bank, “Digital Government Platforms in Africa,” AfDB Report, 2020.

[9] World Bank, “Rwanda: Digital Solutions for Government Services,” World Bank Group, 2020.

[10] M. Reyes-Carranza and D. Mbugua Muthama, “Urban sprawl and the automation of building control in the peripheries of Nairobi,” Urban Geography, 2025, doi: 10.1080/02723638.2025.2450307

[11] J. Zhang and N. El-Gohary, “Automated Regulatory Compliance Checking for Building Projects Using Text Mining and NLP,” Advanced Engineering Informatics, vol. 30, no. 3, pp. 354–367, 2016.

[12] E. Hjelseth, “AutoCodes: Digital Rule-Based Compliance Checking,” Journal of Building Information Modeling, vol. 7, no. 1, pp. 35–42, 2017.

[13] D. Lee, K. Park, and H. Kim, “Deep Learning-Based Automatic Detection of Drawing Components from Blueprints,” Automation in Construction, vol. 106, 2019.

[14] B. Adegun and L. Oduwaye, “Urban Planning and AI: Evaluating Lagos’ Readiness,” African Journal of Built Environment Research, vol. 2, no. 1, pp. 18–30, 2021.

[15] NYC Department of Buildings, “DOB NOW: Inspections Overview,” City of New York, 2019.

[16] X. Chen, J. Wang, and L. Wu, “Optimizing Inspection Scheduling Using Predictive Analytics,” IEEE Transactions on Systems, Man, and Cybernetics, vol. 47, no. 4, pp. 675–685, 2017.

[17] T. Mphago, S. Dube, and J. Bopape, “Smart Urban Management in South Africa,” South African Journal of Information Management, vol. 22, no. 1, pp. 1–8, 2020.

[18] POTRAZ, “Postal and Telecommunications Sector Report,” Government of Zimbabwe, 2021.

[19] T. Mukwashi, “Digitization in Zimbabwean Local Governments,” Zimbabwe Journal of Public Administration, vol. 4, no. 2, pp. 21–34, 2020.

[20] R. Chigwedere, “ICT Strategy Gaps in Zimbabwean Municipalities,” African Journal of Information Systems, vol. 13, no. 1, pp. 44–59, 2021.

[21] R. Heeks, “Digital Government Success and Failure: A Conceptual Model,” Information Polity, vol. 23, no. 3, pp. 365–382, 2018.

[22] D. Mhlanga, “AI in Zimbabwe: Opportunities and Challenges,” Zimbabwe Institute of Technology Review, vol. 2, no. 1, pp. 12–25, 2020.

[23] N. Mehrabi et al., “A Survey on Bias and Fairness in Machine Learning,” ACM Computing Surveys, vol. 54, no. 6, 2021.

[24] UN-Habitat, “Data Privacy in Smart Cities,” UN-Habitat Policy Brief, 2020.

[25] H. Margetts and A. Naumann, “Government as a Platform: What Can Estonia Teach Us?,” Oxford Internet Institute Working Paper, 2017.

Appendix A: templates of data collection tools

Interview Responses with Inspector (City Council Official)

Interview Responses: City Council Official (Inspector)

1. What is the existing procedure for submitting and approving building plans?
Currently, applicants submit their building plans physically at the city planning office. The documents are logged manually, and then passed through various departments for review—structural, environmental, and zoning—before final approval.

2. What are the major bottlenecks or challenges to this process?
The main challenges include delays due to paperwork movement between departments, loss or misplacement of documents, lack of coordination, and limited transparency on the status of applications.

3. What do you anticipate a digital solution can improve in order to enhance this process?
A digital system could streamline submission, automate routing to relevant departments, track status in real time, and reduce the dependency on physical paperwork. It would also improve accountability and speed up inspections and approvals.

City Council Official/ Planning Department Responses

4. What departments are responsible for the approval of building plans?
Departments involved include the Building Control Section, Town Planning, Environmental Management, Fire Department (for safety), and Engineering Services.

5. What documents need to be submitted for a general residential application?
Applicants must submit architectural drawings, site plans, structural calculations, land ownership documents, and in some cases, environmental assessments.

6. How do you currently handle storage and tracking of applications?
All applications are logged into physical registers and stored in filing cabinets. Some offices use spreadsheets to track progress, but this is not standardized.

7. How do you provide feedback on plans received from applicants?
Feedback is provided manually through stamped comments on the returned documents. Applicants are asked to collect responses in person, which can be time-consuming.

8. What manual compliance checks do you perform?
We check for plot size, building setbacks, zoning compliance, sanitation provisions, fire exits, and structural integrity based on local standards.

9. What are your primary key performance indicators (KPIs) for gauging processing times?
We track average approval times, the number of plans processed per month, and the frequency of revisions or rejections due to non-compliance.

10. Are there any regulatory or legal guidelines the system must enforce?
Yes. The system must enforce the Zimbabwe National Building Code, local council by-laws, and zoning regulations. Each document must be checked against these.

Inspector Responses

11. How are you currently distributing inspection work?
Inspection tasks are assigned manually by the head of department based on inspector availability and location. There's no automated scheduling.

12. What do you need before conducting an inspection?
We need the approved plans, application number, physical address, applicant contact details, and previous inspection notes (if any).

13. How and where do you enter and submit inspection reports?
We fill out printed inspection forms on-site, then return them to the office for data entry. The reports are manually filed and scanned later if needed.

Interview Responses from Applicant/Citizen 1

Applicant A – Urban Homeowner Applying for a Residential Plan Approval

1. **What is the existing procedure for submitting and approving building plans?**
   I visit the local city council office, collect the application forms, and submit my drawings physically. After that, I wait for updates, which usually means calling or going back in person.
2. **What are the major bottlenecks or challenges to this process?**
   It's very slow. I had to make several trips just to correct minor details. Also, it's hard to know where the plan is in the process—it just disappears into the system.
3. **What do you anticipate a digital solution can improve in order to enhance this process?**
   If I could submit everything online and get notifications on the status, it would save time and transport costs. I also think digital feedback would reduce confusion.
4. **. What is your process for submitting a building plan?**
   I paid an architect to prepare the documents and took them to the council office personally. I filled out the form and attached the copies they asked for.
5. **Have you experienced delays in receiving feedback or approvals?**
   Yes, my plan took over two months to get feedback. I only found out it had been rejected when I went to check myself.
6. **How do you remit for approvals or inspections?**
   I paid cash at the council's payment office and got a receipt. Sometimes the queues are very long.
7. **Would you be comfortable uploading documents and tracking your application online?**
   Definitely. I already use my phone for other services like banking. I'd prefer that over walking to the office every time.
8. **What would make the system more user-friendly for you?**
   Simple language, fewer steps, and maybe a help chat or a guide would be useful. A mobile-friendly website would be perfect.

Interview Responses from Applicant/Citizen 2

Applicant B – Building Contractor Working with Multiple Clients

1. **What is the existing procedure for submitting and approving building plans?**
   I submit plans for my clients in person, and sometimes I have to follow up at different departments separately—building, planning, and fire services.
2. **What are the major bottlenecks or challenges to this process?**
   It's disjointed. Sometimes one department approves, but another takes weeks. There's no centralized update system, so I have to check manually.
3. **What do you anticipate a digital solution can improve in order to enhance this process?**
   A single online portal to submit, pay, and track approvals would make my work easier. It would also help avoid repeated trips to check on progress.
4. **What is your process for submitting a building plan?**
   I prepare the full set—drawings, forms, copies of ownership documents—and submit them physically at the planning desk.
5. **Have you experienced delays in receiving feedback or approvals?**
   Yes, often. I had one case where an approval letter went missing and had to be reissued. That added more delays.
6. **How do you remit for approvals or inspections?**
   Cash or bank deposit, depending on what's working. Receipts are sometimes hard to retrieve if lost.
7. **Would you be comfortable uploading documents and tracking your application online?**
   Yes. As a contractor, it would be much more efficient, especially since I work with multiple customers

Questionnaire 1: Response from Architect 1

User Requirements Questionnaire

Section 1: General Information

1. Full Name: _____C.I_____

2. User Role:

- [x] Architect/Designer
- [ ] City Council Official
- [ ] Construction Engineer/Builder
- [ ] Other (Please specify_____

3. Occupation/Designation: _____Architect_____

4. Years of experience in this role: _____More than 10 years_____

Section 2: Current Process Evaluation

How satisfied are you with the current plan approval process?

- [ ] Very Satisfied
- [ ] Satisfied
- [ ] Neutral
- [x] Dissatisfied
- [ ] Very Dissatisfied

What are the biggest challenges you face in the approval process? *(Select up to 3)*

- [x] Long waiting times
- [x] Inconsistent feedback from reviewers
- [ ] Lack of transparency in status updates
- [x] Manual paperwork & physical submissions
- [ ] Complex submission requirements
- [ ] Other: _____

How long does it typically take to get plans approved?

- [ ] < 1 week
- [ ] 1-4 weeks
- [ ] 1-3 months
- [x] >3 months

Section 3: Desired System Improvements

Which features would most improve the approval process? *(Select up to 3)*

- [x] Online submission portal
- [x] Digital document verification
- [ ] Automated compliance checks
- [x] AI-based error detection
- [ ] Real-time application tracking
- [ ] Other: _____

Would a digital dashboard for tracking application status be useful?

- [x] Yes
- [ ] No
- [ ] Maybe

How should the system notify users about approval updates? *(Select all that apply)*

- [ ] Email
- [ ] Web Portal Alerts
- [x] SMS
- [ ] Other: _____
- [ ] Mobile App Notification

Section 4: Open-Ended Feedback

Do you have any other suggestions for improving the plan approval system?

I would want to suggest that the we get clear feedback and a well detailed report of what we would have failed in terms of Design as architects

Questionnaire 2: Response from Architect 2

User Requirements Questionnaire

Section 1: General Information

1. Full Name: _____P.K.M_____

2. User Role:

- [x] Architect/Designer
- [ ] City Council Official
- [ ] Construction Engineer/Builder
- [ ] Other (Please specify_____

3. Occupation/Designation: _____Architect_____

4. Years of experience in this role: _____less than 10 years_____

Section 2: Current Process Evaluation

How satisfied are you with the current plan approval process?

- [ ] Very Satisfied
- [ ] Satisfied
- [x] Neutral
- [ ] Dissatisfied
- [ ] Very Dissatisfied

What are the biggest challenges you face in the approval process? *(Select up to 3)*

- [x] Long waiting times
- [x] Inconsistent feedback from reviewers
- [x] Lack of transparency in status updates
- [x] Manual paperwork & physical submissions
- [ ] Complex submission requirements
- [ ] Other: _____

How long does it typically take to get plans approved?

- [ ] < 1 week
- [ ] 1-4 weeks
- [ ] 1-3 months
- [x] >3 months

Section 3: Desired System Improvements

Which features would most improve the approval process? *(Select up to 3)*

- [ ] Online submission portal
- [x] Digital document verification
- [x] Automated compliance checks
- [x] AI-based error detection
- [ ] Real-time application tracking
- [ ] Other: _____

Would a digital dashboard for tracking application status be useful?

- [ ] Yes
- [ ] No
- [x] Maybe

How should the system notify users about approval updates? *(Select all that apply)*

- [ ] Email
- [x] Web Portal Alerts
- [ ] SMS
- [ ] Other: _____
- [x] Mobile App Notification

Section 4: Open-Ended Feedback

Do you have any other suggestions for improving the plan approval system?

I would want to suggest that the process be more transparent so that we know what's happening at the background

Questionnaire 3: Response from Construction Engineer 1

User Requirements Questionnaire

Section 1: General Information

1. Full Name: _____P.M_____

2. User Role:

- ☐ Architect/Designer
- ☐ City Council Official
- ☒ Construction Engineer/Builder
- ☐ Other (Please specify_____

3. Occupation/Designation: _____Architect_____

4. Years of experience in this role: ____More than 10 years_____

Section 2: Current Process Evaluation

How satisfied are you with the current plan approval process?

☐ Very Satisfied ☐ Satisfied ☐ Neutral ☒ Dissatisfied ☐ Very Dissatisfied

What are the biggest challenges you face in the approval process? *(Select up to 3)*

- ☒ Long waiting times
- ☒ Inconsistent feedback from reviewers
- ☐ Lack of transparency in status updates
- ☒ Manual paperwork & physical submissions
- ☒ Complex submission requirements
- ☐ Other: _____

How long does it typically take to get plans approved?

☐ < 1 week ☐ 1-4 weeks ☐ 1-3 months ☒ >3 months

Section 3: Desired System Improvements

Which features would most improve the approval process? *(Select up to 3)*

- ☒ Online submission portal
- ☐ Digital document verification
- ☒ Automated compliance checks
- ☒ AI-based error detection
- ☒ Real-time application tracking
- ☐ Other: _____

Would a digital dashboard for tracking application status be useful?

- ☒ Yes
- ☐ No
- ☐ Maybe

How should the system notify users about approval updates? *(Select all that apply)*

- ☒ Email
- ☐ Web Portal Alerts
- ☒ SMS
- ☐ Other: _____
- ☒ Mobile App Notification

Section 4: Open-Ended Feedback

Do you have any other suggestions for improving the plan approval system?

I would want to suggest that the process be quicker since it takes time to get feedback hence hindrance of progress on construction projects

Appendix B: User Manual

User Manual Guide: AI-Based Plan Approval System

Version: 1.0

System Name: AI-Based Residential Plan Approval System

Release Date: 15 May 2025

Target Users: Applicants, Admins, Inspectors

Table of Contents

1. Introduction

This system automates the residential building plan approval process. It allows applicants to submit plans, uses AI for compliance verification, and enables inspectors and administrators to manage inspections and approvals.

2. System Requirements

- Browser: Chrome, Firefox, or Edge (latest versions)
- Internet: Stable connection (1 Mbps or higher)
- Device: Laptop/Desktop (Mobile supported for tracking only)
- Login Credentials: Provided at registration

3. Getting Started

Account Registration:

- Visit the website homepage.
- Click on "Register."
- Select role: Applicant, Admin, or Inspector.
- Fill in personal details and submit.
- Login with your email and password.

4. Applicant User Guide

a) Submit a Building Plan:
   1. Login and go to the Dashboard.

   2. Click on Submit Plan.
   3. Fill in the building application form.
   4. Upload plan files (PDF/DWG).
   5. Click Submit.

b) AI Compliance Check:
   - After submission, the system runs an automatic AI compliance check.
   - You'll receive a notification with the result.

c) Revising Documents:
   - If your plan is non-compliant, go to My Applications > Edit.
   - Upload revised documents.
   - Re-submit for review.

d) Making Payments:
   1. Go to Payments section.
   2. Choose pending invoice.
   3. Pay via integrated online payment gateway.
   4. View payment.

e) Tracking Status:
   - Navigate to Application Status to view the current stage.
   - Status updates are shown in real time.

f) Download Certificate:

- Once approved, download your plan approval certificate from the Documents tab.

5. Admin User Guide

a) Manage Users:
   - View all registered applicants and inspectors.
   - Edit roles, deactivate or delete accounts.

b) View Applications:
   - Go to Applications dashboard.
   - Review application details and AI compliance results.

c) Generate Reports:
   - Download monthly reports on applications, payments, errors, and inspections.

d) System Settings:
   - Configure application deadlines, and payment thresholds.

6. Inspector User Guide

a) View Inspection Schedule:
   - Login and go to My Inspections.
   - View scheduled sites and dates.

b) Conduct Inspection:
   - Visit the site on the scheduled date.
   - Fill out the online inspection checklist.

c) Submit Inspection Report:
   - Upload notes, pictures, or other findings.
   - Submit the report to finalize your task.

7. Troubleshooting

Issue	Solution
Forgot password	Use 'Forgot Password' on the login page.
Document not uploading	Check format (PDF or DWG) and file size (<10MB).
Payment failed	Check internet connection or retry after 5 minutes.
AI check stuck	Refresh dashboard or contact support.

8. Contact Support

Email: support@zimbuilds.gov.zw

Phone: +263 711 711 7111

Support Hours: Mon–Fri, 8AM–5PM

Location: Local Council ICT Office

Appendix C: Research papers

ZimBuilds: AI-Enhanced Web Platform for Streamlined Residential Plan Approvals in Urban Development

Ropafadzo E Munetsi; Kenneth Chiworera

Department of Software Engineering, School of Information Sciences and Technology, Harare Institute of Technology, Harare, Zimbabwe

esalenciar@gmail.com ; kchiworera@hit.ac.zw

Abstract—Rapid urbanization in Zimbabwe, as of 2012, has created problems for the local council authorities, creating a highly flawed manual and time-consuming system, causing delays in the residential plan approval process, hence the administrative burden on local council authorities. This paper presents an innovative web-based residential plan approval system that utilizes AI-based compliance checking, automation, and centralized documentation storage to solve these problems. ZimBuilds is a comprehensive solution that deals with document uploading, plan review, inspection scheduling, and payment. It includes a web interface for inspectors, administrators, and applicants, aiming for transparency and application tracking.

I. INTRODUCTION

Zimbabwe's urbanization has been motivated by economic development, population increase, and rural-urban migration in the past two decades. As a result, there is a high demand for residential buildings, which necessitates the prompt approval of building plans by municipalities. However, the approval of residential plans is still largely manual and paper-driven.

The local authorities require applicants to submit physical architectural plans, structural plans, and documents to various departments, like engineering, planning, and health, to check them for compliance, which is tedious and time-consuming. Making the switch to a digital platform will expedite the application process, lower error rates, and give applicants and city council staff real-time updates. Adopting technology would increase overall consumer happiness in

addition to efficiency. And the primary objectives of my proposed solution are to:

1. To digitize and streamline the workflow of residential plan application and its approval.
2. To use AI-based document verification for automatic compliance checking.
3. To automate the scheduling and monitoring of inspections.

II. PROBLEM-STATEMENT

The current Zimbabwean residential plan approval procedure in the local authorities is outdated, inefficient, and unable to cater to growing demand. This system's manual nature brings about the following primary concerns:

- Long periods taken to process an application.
- Lack of transparency and feedback to the applicant.
- Human errors and miscommunication.
- Difficulties in tracking applications and inspections.

III. RELATED WORK

In recent years quite several countries have tried to implement systems that streamline plan approval, for instance, the Online Building Plan Approval System of India. In [1] author explores the OBPAS, which is making waves throughout several Indian states that have changed urban planning by digitalizing the entire building permit submission and approval process. OBPAS incorporates features of the Smart Digital Construction Review (SmartDCR) to automate compliance checks against building bylaws that minimize human error, increase efficiency, and decrease approval lag time. A case study by [1] on the implementation of OBPAS in Odisha indicates that OBPAS increased transparency and improved approval times, as low as 7 days, and limits steps that require in-person visits to government facilities, Dheeraj Mandloi et al, 2015. This represents an entirely different approach from cumbersome old paper systems that were prone to corruption, delays, and the archaic process of referencing building bylaws. There are possibilities for real-time interactions of architects and municipal officials in a collaborative system using OBPAS's workflow engine and document validation capabilities.

Reyes-Carranza et al. in [2] provide insight into the generation and outcome of Ke DAMS (Kajiado's e-Development Management System) of Nairobi, Kenya. Ke DAMS is a web-based portal developed in 2020 that automates the issuance of building

permits by digitalizing the process of issuing construction permits in the construction industry, making it transparent and user-friendly. The author in [2] sees Ke DAMS as an initiative to fight corruption, which is ushering in a new digital age of local governance regarding the regulation of this sector. Although the action to electronically permit development is a step towards improvement, Reyes-Carranza et al. in [2] point out limitations, such as when systems are not specific to the unique context of peripheral or informal areas, which still tend to be the focus of centralized systems. Improvements can be realized by the industry, everyone in procedures such as those in both building inspection and construction approval, when implemented, not only automated but also contextual approaches in developing and integrating structures into the processes. The example of Nairobi is a unique case of e-construction permit systems that exploit the early stages of digital intervention into urban governance. However, Reyes-Carranza et al. argue that flexible, accessible, and appropriate digital solutions must account for the 'formal' and 'informal' practice of development in any analysis of the urban landscape.

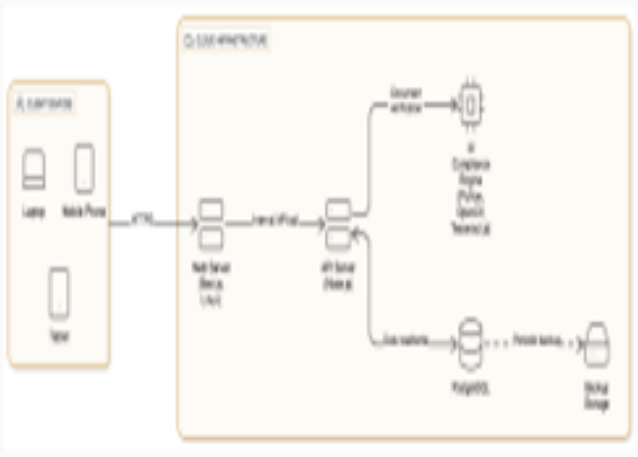
According to an online publication by Saarda, in October 2021, the City of Johannesburg launched a web-based Construction Permit Management System to digitize the statutory submission of building plans, which was entirely based on a paper format and manual processes at various stages. The City of Johannesburg's Construction Permit Management System allows SACAP-registered architectural professionals to electronically submit building plans, reducing the reliance on physical documentation, which will also allow for faster processing of applications [5]. This constitutes part of the City of Johannesburg's Smart City program and aligns with the [5] City's Growth and Development Strategy 2040 by improving transparency, operational efficiency, and service delivery. Some of the highlights of the system include: real-time updates, the ability to categorize types of buildings regarding the plans submitted, and to link inspection workstreams with the building plan management process. In [3][4], the expectation is that it will halve the decision turnaround time from 30 days to 15 days, assuming all submissions are complete. The online system provides another layer for fraud mitigation as it will be checking registration credentials.

To conclude, across the globe, we are starting to see widespread public acceptance of Artificial Intelligence (AI) integration in building plan approvals. In [6], an example is Estonia's 'Kratt' strategy that promotes the use of AI to automate administrative tasks, including document verification. As alluded to by the author in [6], AI systems may retain the ability to automatically check and review architectural documents to ensure compliance with relevant zoning laws and building codes; again, minimizing human error and accelerating the approval process.

IV. THE SOLUTION

1. AI-Powered Document Compliance Verification: Uploaded architectural plans are immediately reviewed by AI against regulatory stipulations, which ensures prompt identification of compliance problems, reducing back-and-forth with applicants.
2. Online Application Submission: Applicants can submit building plans, company/personal information, and supporting documents via a web form, and role-based dashboards for admin, inspector, and applicants.
3. Automated Inspection Scheduling: Inspections are automatically assigned based on project stages. Inspectors can view, take, and report inspection results.

B. SOLUTION ARCHITECTURE



V. METHODOLOGY

The method used for requirements gathering was interviews with planning officers, architects and other stakeholders, and observation of municipal processes were carried out. Functional and non-functional requirements were then determined and documented. On system design, the system employs reusable and modular design patterns. All of the forms follow a consistent layout, based on the Plan Approval Form, with field validation and grouping. The use of Tailwind CSS enables responsive design,

while icons and headers ensure intuitive navigation.

AI document verification uses pre-trained models and custom zoning rules to review image files or PDFs that have been uploaded. Missing items or defects (e.g., missing label, scale, or codes of safety) are marked for review.

In the development stage, Agile development was followed with sprints aimed at:

- UI/UX design.
- API and database integration.
- Document verification engine.
- Payment and inspection scheduling modules.

Version control was done through GitHub, and all APIs were tested using Postman before frontend integration.

VI. RESULTS

Document Verification Accuracy: From a sample of 50 architectural plans created to meet standard local building codes:

- AI verification detected non-compliance in 42% of applications (e.g., missing zoning information, incorrect building heights).
- The system correctly identified missing elements with 91% accuracy, confirmed through manual cross-verification by inspectors.
- The average processing and return time for AI was under 45 seconds per document.

Application Turnaround Time: In contrast to the existing manual process, which takes 90+ days:

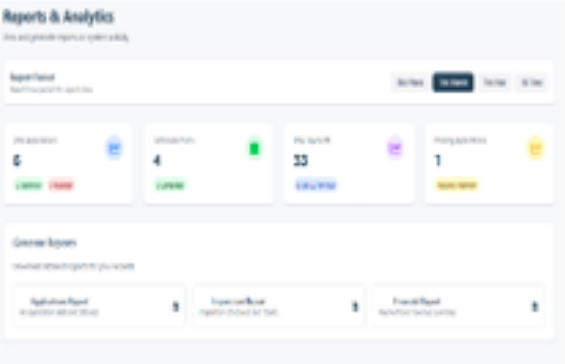
- Applications submitted through the system took on average 10–15 working days to process, if documents were complete.
- Feedback loops that were instantaneous allowed applicants to resubmit amended plans within a space of hours, significantly reducing delays.

Inspection Management

- Inspectors were assigned automatically by area and availability, and administrative workload was decreased by 70%.
- Real-time status updating allowed inspectors to report outcomes directly from the field via mobile devices.
- Scheduling conflicts for inspections were reduced to near zero, owing to the

built-in calendar and conflict-checking logic.

Admin Portal for reports and analytics



Objective	Achieved	Partially Achieved
To develop a web app that streamlines the application process for plan approval	✓	
To automate document verification using AI-powered document verification.	✓	
To automate payment by integrating online payments	✓	
To implement automated inspector appointment scheduling	✓	

VII. CONCLUSION AND FUTURE

The e-approval of residential plans online revolutionizes the submission of building plans in Zimbabwe through computerization of submissions, incorporation of artificial intelligence verification, and providing open tracking. The system solves persistent problems related to delays in processing, inefficiency, and absence of proper feedback.

In future releases, the system can be enhanced by:

- Incorporating status updates through SMS/email notifications.
- Including support for 3D BIM model checking.
- Connecting to national land registries and GIS systems.
- Providing offline submission modules for low-connectivity regions.

VIII. BIOGRAPHY

Ropafadzo Esalencia Munetsi was born in Harare, Zimbabwe on 30 November 2001. Currently pursuing a degree in Software Engineering and a final year student at the Harare Institute of Technology in Harare, Zimbabwe.

REFERENCES

[1] M. Reyes-Carranza and D. Mbugua Muthama, "Urban sprawl and the automation of building control in the

peripheries of Nairobi," *Urban Geography*, 2025, doi: 10.1080/02723638.2025.2450307
[2] D. Mandloi and V. Thakur, "IJSR-INTERNATIONAL JOURNAL OF SCIENTIFIC RESEARCH," 2013.
[3] N. B. C. Buthelezi, G. O. Onatu, and C. O. Aigbavboa, "Planning approval of housing developments: case study of the city of Johannesburg and Tshwane metropolitan municipalities of South Africa," *Frontiers in Sustainable Cities*, vol. 7, Apr. 2025, doi: 10.3389/frsc.2025.1468965.
[4] H. Holmqvist and S. Papp, "A critical analysis of the building permit process at the Urban Planning Department in Gothenburg Identification of improvement measures for a smoother internal process Master's Thesis in the Master's Programme Design and Construction Project Management," 2015.
[5] https://www.saarda.co.za/ development-planning-launches-the-city-of-joburg-construction-permit-management-system
[6] K. Härmand, "AI Systems' Impact on the Recognition of Foreign Judgements: The Case of Estonia," *Juridica International*, vol. 32, pp. 107–118, Dec. 2023, doi: 10.12697/ji.2023.32.09.