Software Requirements Specification

for

UWIDormFinder System

Version 1.2

Prepared by

Rashene Dillon	**620161588**	**rashene.dillon@mymona.uwi.edu**
Damion Henry	**620071079**	**damion.henry02@mymona.uwi.edu**
Shantay Kellyman	**620157798**	**shantay.kellyman@mymona.uwi.edu**
Shari Oliver	**620156656**	**shari.oliver02@mymona.uwi.edu**
Samantha Samuels	**620154205**	**samantha.samuels02@mymona.uwi.edu**
Tahmia Vincent	**620156659**	**tahmia.vincent@mymona.uwi.edu**

Course Instructor:	**Dr.Ricardo Anderson**
Course:	**COMP2140 – Introduction to Software Engineering**
Studio Facilitator:	**Mr. Aldon Milton**
Date:	**November 28, 2024**

TABLE OF CONTENTS

1 Overall Description

1.1 Product Context and Need ---------- 4

1.2 Product Functionality ---------- 4

1.3 Stakeholders and Users Characteristics ---------- 5

1.4 Operating Environment ---------- 5

1.5 Design and Implementation Constraints ---------- 5-6

1.6 Assumptions and Dependencies ---------- 6

2 Specific Requirements

2.1 External Interface Requirements ---------- 7

2.1.1 Hardware Interfaces ---------- 7

2.1.2 Software Interfaces ---------- 7

2.1.3 Communications Interfaces ---------- 7

2.2 Functional Requirements ---------- 8 - 13

2.3 Behaviour Requirements ---------- 14

2.3.1 Use Case View ---------- 15 - 18

3 Other Non-functional Requirements

3.1 Performance Requirements ---------- 19

3.2 Safety and Security Requirements ---------- 19

3.3 Software Quality Attributes ---------- 19 - 20

Appendix

Appendix A - Interview Questions ---------- 21

Transcript of Interview ---------- 21 - 23

Appendix B - Survey Questions ---------- 24

Summary of Survey ---------- 24 -28

Revisions

Version	Primary Author(s)	Description of Version	Date Completed
Version 1.1	Rashene Dillon Damion Henry Shantay Kellyman Shari Oliver Tahmia Vincent Samantha Samuels	This is the first version of the document and there are no updates available.	02/11/2024
Version 1.2	Rashene Dillon Damion Henry Shantay Kellyman Shari Oliver Tahmia Vincent Samantha Samuels	This is the second version of the document.	28/11/2024

1 Overall Description

1.1 Product Context and Need

The Irvine Hall at the University of the West Indies is dedicated to fostering a dynamic and supportive community for students residing on campus. Currently, the hall lacks a personalized application to facilitate dorm room bookings, which has resulted in an inefficient/inaccurate process for both students and administration. Key challenges include the absence of application status tracking and a lack of detailed room descriptions or visuals to help students make informed decisions. Additionally, staff have limited access to information such as students' disabilities, making it difficult to assign appropriate accommodation. The system also lacks real-time room availability and notifications, as a result administrators are unable to effectively manage capacity and provide information such as "limited rooms available" or "fully booked." Addressing these issues is essential to create a more efficient and user-friendly housing system for UWI Mona students interested in living at Irvine Hall. The purpose of the UWIDormFinder system is designed to enhance the overall student experience at Irvine Hall, reflecting the hall's commitment to providing a supportive and well-organized living environment. By implementing this solution, the hall will simplify the administrative workload and foster a stronger sense of community among residents.

1.2 Product Functionality

- The system shall enable students to search for available dormitory rooms.
- The system shall allow students to book a dorm room and make updates to their booking information as needed.
- The system shall be able to track student applications to assist housing staff in approving or denying requests.
- The system shall allow students to be assigned to a dorm room based on their preference and priority.
- The system shall notify students of any major changes.
- The system shall log any relevant changes.

1.3 Stakeholders and Users Characteristics

The key stakeholders of this system are the Students (the room occupants), the University Administration/Housing Management team and the Mona Information Technology Services (MITS). The primary (most important) stakeholder of the system are the Students followed by the University Administration/Housing Management team. The secondary (least important) stakeholder is the Mona Information Technology Services (MITS).

The students are the end users who will apply for dorm rooms and track the status of their applications. They are technically proficient and expect a smooth user-friendly interface and timely notifications. The University Administration/Housing Management Team oversees the room allocation process and sets room policies. They are the decision-makers, non-technical users, requiring clear data and reports to make informed decisions. The Mona Information Technology Services (MITS) supports the deployment and maintenance of the system, ensuring system availability and resolving technical issues. They are highly technical users who need access to system logs, performance metrics, and security controls.

1.4 Operating Environment

The UWIDormFinder application for Irvine Hall is a web-based application designed to operate on any platform that can run a modern web browser, this includes Windows, Linus, macOS. As long as the device can run a compatible browser, such as Google Chrome (version 90+), Mozilla Firefox (version 85+), or Safari (version 14+), users will be able to access and utilize the application without additional software installation. The application is intended for use on Laptops and Desktops with minimum hardware requirement as follows:

- **Windows:** 1GHz processor, 4GB RAM, and 64GB of storage.
- **Linux:** 2GHz processor, 4GB RAM, and 80GB of storage.
- **macOS:** 1.1GHz processor, 8GB RAM, and 256GB of storage

1.5 Design and Implementation Constraints

- The project is developed with HyperText Markup Language (HTML), Cascading Style Sheets (CSS), JavaScript, Bootstrap 5, SQLite and SQLAlchemy and Flask 3.1.0

(Python 3) using the Model-View-Controller (MVC) design pattern and Visual Studio Code as Integrated Development Environment (IDE).

- The device that will run the software on the client's side, should have a substantial amount of storage space and memory space. The most ideal space for storage is 256GB and up, and the most ideal space for memory would be 4GB and up.
- The system shall be available every weekday between 8 a.m and 9 p.m.
- The system must have access to database software to store students' inputted data.

1.6 Assumptions and Dependencies

- It is assumed that the system will be used by a significant number of students on a daily basis.
- It is assumed that they use the keyboard and mouse effectively.
- It is assumed that the user is using either a laptop or desktop.
- It is assumed that they are connected to a stable internet.

2 Specific Requirements

2.1 External Interface Requirements

2.1.1 Hardware Interfaces

The software is designed to be used for laptops and/or desktops. It requires a device with a single or dual processor, at least 4GB of RAM, and a minimum download speed of 10 Mbps. The device used by employees to access the system must have a minimum of 256GB of storage to accommodate the generated database files. As the student clientele grows, additional storage capacity may be necessary.

2.1.2 Software Interfaces

The UWIDormFinder software will automatically generate and manage dorm assignment records and databases without the need for external applications. The software is designed to be compatible with Linux, Windows and MacOS operating systems, ensuring accessibility for all users across different platforms. Since Python 3 will be the primary language for developing UWIDormFinder, the application will utilize key Python 3 frameworks and databases such as Flask 3.1.0 for web framework support, SQLite and SQLAlchemy for database management, and additional libraries for user authentication and data handling. This setup allows the system to efficiently manage dorm assignments, student preferences, and other essential functionalities.

2.1.3 Communications Interfaces

The UWIDormFinder software will leverage standard HTTP and HTTPS protocols for client-server communication, with HTTPS providing secure data exchange between the clients and the Flask-based backend. The system will support email notifications for key actions, such as booking confirmations, and application status updates. Email communications will comply with SMTP standards, with content formatted in HTML for user readability. The system will use SQLite and SQLAlchemy databases to manage and store application data locally within the server environment, facilitating smooth communication between the application and the database for storing user data, booking records, and application tracking information.

2.2 Functional Requirements

Requirement #: 1 - Search Room Availability
Use case: Search Room Availability
Rationale: Students may have specific preferences and requirements for dorm rooms. They need to know what rooms are available that match their preferences, especially if some preferred options are not available.
User Requirement: The system shall allow students to search for available dorm rooms.
System Requirements:

1.1 Upon successful validation of the student's login credentials, the system shall allow access to the room search feature.
1.2 The system shall allow students to filter room search results based on criteria such as room type, dormitory building, and preferred floor.
1.3 The system shall update the availability status to ensure accurate room listings are presented to the student.

Acceptance Criteria:

1. The user is able to successfully log in and access the room search feature 100% of the time after providing valid credentials.
2. The system displays accurate room availability based on the student's search preferences 100% of the time.
3. The search filters (room type, dormitory building, and preferred floor) return relevant results matching the criteria 100% of the time.

Relates to/Dependencies: None
Priority: High **Team Owner:** Samantha Samuels

Requirement #: 2 - Book Dorm Room
Use Case: Book Room
Rationale: To facilitate the housing process for students at Irvine Hall, it is essential to provide an efficient and user-friendly interface for students to book dorm rooms. A student successfully completes their application by finishing the booking process.
User Requirement: The system shall enable students to submit their booking application.
System Requirements:

2.1 The system shall provide a user interface for students to submit a room application based on their preferences and availability.
2.2 The system shall guide students through the booking process, ensuring that all required fields are filled out before submission.

2.3 The system shall send a confirmation of the booking to the student via email, including details of the booked room.

2.4 The system shall log each booking application for tracking and auditing purposes.

Acceptance Criteria:

1. 100% of booking applications are processed successfully and saved in the system.
2. The system must send confirmation notifications to students upon successful booking.

Relates to/Dependencies: Search Room Availability, Send Status Update

Priority: High **Team Owner:** Tahmia Vincent

Requirement #: 3 - View Student Room Application

Use case: View Application

Rationale: All students and/housing admin may have room application information that they need to review the submitted application/s.

User Requirement: The system shall allow an authorized student and/or housing admin to review the information related to a student's room application/s.

System Requirements:

3.1 Upon successful validation of the student's and/or housing admin's login credentials, the system shall allow access to the view application feature.

3.2 On the view application page, the system shall allow students access to view their application/s.

Acceptance Criteria:

1. 100% of the application/s details can be viewed by the authorized student and housing admin.

Relates to/Dependencies: None

Priority: Low **Team Owner:** Rashene Dillon

Requirement #: 4 - Edit Student Room Application

Use case: Edit Application

Rationale: All students may have room application information that needs to be added based on changes in preferences or otherwise outdated and needs to be updated with more recent information.

User Requirement: The system shall allow an authorized student to edit the information related to a student's room application.

System Requirements:

4.1 Upon successful validation of the student's login credentials, the system shall allow access to the view application feature.

4.2 On the view application page, the system shall allow students access to edit their applications.

4.3 The system shall allow users to confirm and save all changes.

Acceptance Criteria:

1. 100% of the application details added or updated must be saved and reflected in the system.

Relates to/Dependencies: View Application

Priority: High **Team Owner:** Rashene Dillon

Requirement # :5 - Track Application Status

Use Case: Track Application Status

Rationale: Students need to know the status of their dormitory application to make necessary plans. The system should provide updates to keep students informed.

User Requirement: The system shall allow students to track the current status of their dormitory application and receive updates as the application progresses through different stages.

System Requirements:

5.1 Upon successful validation of the student's login credentials, the system shall allow access to the track application status feature.

5.2 The system shall display the current status of the application (e.g., Pending, Application Approved/Declined, Payment Under Review, Room Booked).

5.3 The system shall update the application status automatically based on actions taken by the housing admin and student.

5.4 The system shall notify the student of any change in status via the send status updates

5.5 The system shall allow the Housing Office Staff to add comments on the status updates viewable by the student.

Acceptance Criteria:

1. 100% of the time students are able to view the application status.
2. Notifications are sent to students whenever the application status changes.
3. All status updates are logged and accessible by the student.

Relates to/Dependencies: Send Status Updates, Upload Payment Receipt

Priority: High **Team Owner:** Shantay Kellyman

Requirement #: 6- Send Student Status Updates

Use case: Send Status Updates

Rationale: Students need timely notifications regarding updates, or changes to their booking process to ensure they are informed and prepared for their stay.

User Requirement: The system shall send notifications to students regarding their application status.

S**ystem Requirements:**

6.1 The system shall generate notifications for the student's application status update.
6.2 The system shall send notifications to the student via email.
6.3 The system shall include essential details in notifications (e.g. Pending, Application Approved/Declined, Payment Under Review, Room Booked).
6.4 The system shall allow administrators to customize notification templates.
6.5 The system shall log all notifications sent for auditing purposes.

Acceptance Criteria:

1. 100% of notifications sent must be delivered to students' registered email
2. All notifications must include accurate and essential details.

Relates to/Dependencies: Book Room, Track Application Status, Assign Room, Upload Payment Receipt
Priority: High **Team Owner:** Damion Henry

Requirement #: 7 - Upload Payment Receipt
Use Case: Upload Payment Receipt
Rationale: Each student who has made payment towards their dorm fees needs to provide proof of payment to confirm that their financial obligations are met in order to secure their door room.
User Requirement: The system shall allow students to upload a digital copy of their payment receipt as proof of payment of dorm fees.
System Requirements:

7.1 Upon approved application, the system shall allow access to the payment receipt upload feature.
7.2 The system shall accept various file formats for the receipt upload, such as PDF, JPEG, and PNG, to accommodate different types of documents.
7.3 The system shall securely store the uploaded receipt and link it to the student's profile for administrative review.

Acceptance Criteria:

1. The user is able to log in and access the payment receipt upload feature 100% of the time after application approval.
2. The system successfully accepts and verifies receipt uploads in supported formats 100% of the time.
3. The uploaded receipt is stored securely and linked to the student's profile for easy access by administrative staff.

Relates to/Dependencies: Track Application Status, Send Status Update
Priority: High **Team Owner:** Samantha Samuels

Requirement #: 8- Assign Dorm Rooms
Use Case: Assign Room

Rationale: Each student who applies to the Irvine Hall requires a designated dorm room upon enrollment, and dorm room assignments must be managed efficiently to avoid overbooking and ensure fair allocation of available space.
User Requirement: The system shall allow authorized administrators to approve the assigned dorm rooms to students and the student should be able to confirm assignment of the room.
System Requirements:

8.1 The system shall verify room availability before assigning the room to a student.
8.2 The system shall allow staff to select a room and approve the assignment for a student.
8.3 The system shall log the assignment for tracking purposes.
8.4 The system shall provide a confirmation to the housing staff and a notification to the student upon successful assignment.

Acceptance Criteria:

1. 100% of the room assignments and modifications are saved and reflected on the system.
2. All modifications, including changes to room assignments, are logged for each session.

Relates to/Dependencies: Book Room , Track Application Status, Upload Payment Receipt, Send Status Update
Priority: High **Team Owner:** Shari Oliver

Requirement #: 9 - Maintain System Log
Use Case: Maintain System Log
Rationale: To ensure accountability and transparency in the dorm assignment process, the system must maintain a comprehensive log of all key activities, including booking dorm rooms, updating application statuses, editing bookings, and assigning rooms. This will help in tracking changes, auditing processes, and resolving any disputes that may arise.
User Requirement: The system shall automatically log all relevant activities (such as booking dorm rooms, updating application statuses, editing bookings, and assigning rooms) performed by authorized users to maintain an accurate record of actions taken within the system.
System Requirements:

9.1 The system shall automatically generate a log entry for every booking of a dorm room, including details of the user who made the booking, timestamp, and room assigned.
9.2 The system shall generate a log entry for each change in application status, documenting the previous status, new status, user making the change, and timestamp.
9.3 The system shall log every modification made to existing bookings, capturing the details of the original booking and the updated information.
9.4 The system shall create a log entry for each room assignment, including the room assigned, the student assigned to, and the user who performed the action.

9.5 The system shall ensure that all log entries are secure and accessible only to authorized personnel for auditing purposes.

Acceptance Criteria:

1. 100% of activities related to booking, application status changes, booking modifications, and room assignments are logged and saved in the system.
2. All log entries contain complete and accurate information, including user identification, timestamps, and details of the actions performed.

Relates to/Dependencies: Track Application Status, Edit Booking, Book Dorm Room, Assign Dorm Rooms

Priority: High **Team Owner:** Damion Henry

Requirement #: 10 - Create Admin Account

Use Case: Create Admin Account

Rationale: An admin account is important for managing and overseeing room bookings processes which ensure smooth operation.

User Requirement: The system shall allow an IT user to create an admin account.

System Requirements:

10.1 Upon successful validation of the IT's login credentials, the system shall allow access to the create admin account feature.

10.2 On the create admin account page, the system shall allow IT to input admin user details and create a account

10.2 All admin user account shall have a initial password of "password"

Acceptance Criteria:

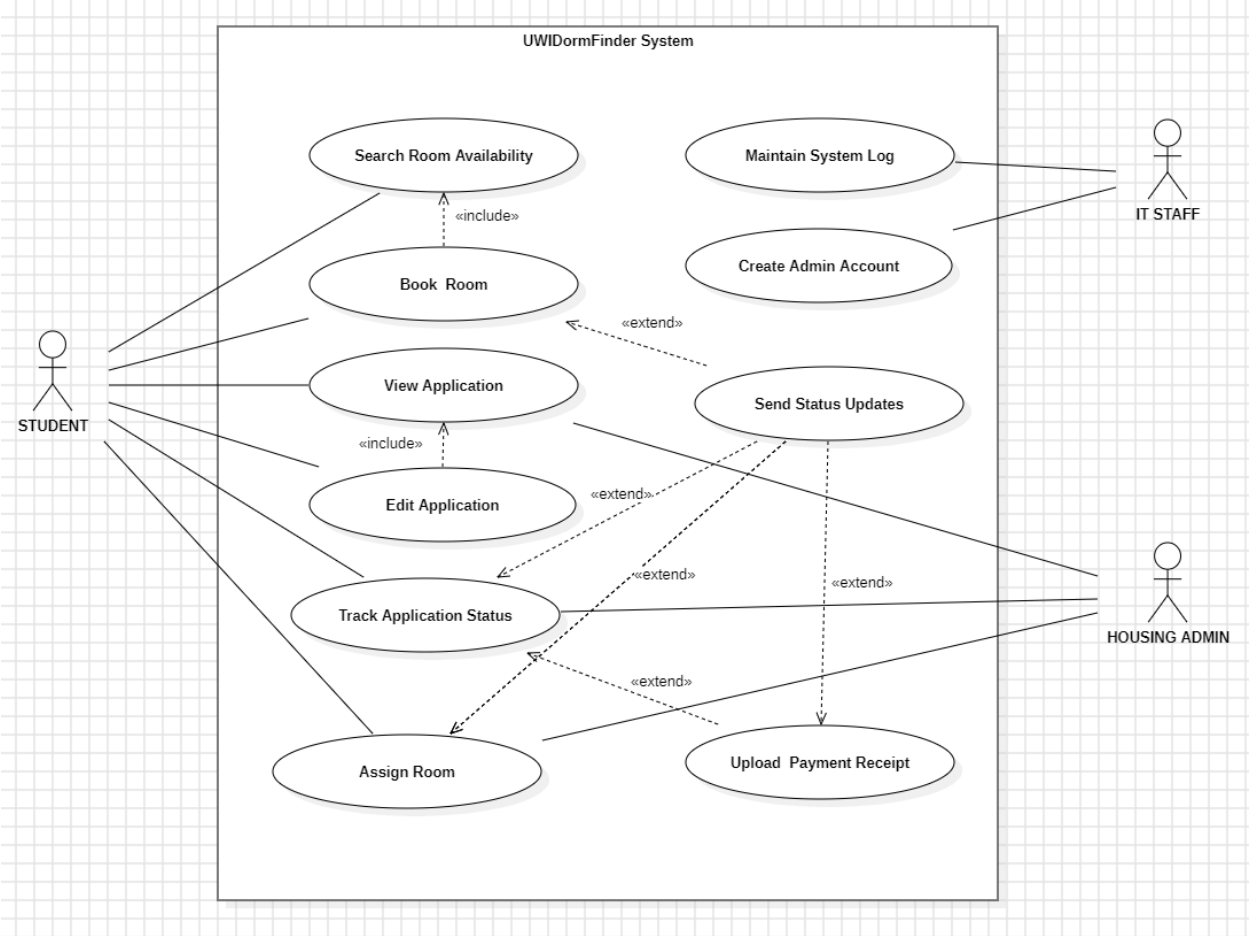
1. 100% of the user created shall be logged into the system.

Relates to/Dependencies: None

Priority: High **Team Owner:** Damion Henry

2.3 Behavior Requirements

2.3.1 Use Case View



Case Narratives

Use Case 1: Search Room Availability	
Description	*A student should be able to search for available dorm rooms.*
Precondition	*Students have successfully logged on to the system and can select their desired search criteria.*
Basic Flow	*After successfully logging into the system, the student is taken to the room search feature and can view the details of each available room.*
Post Condition	*Show details of the hall rooms based on filtered criteria.*

Use Case 2: Book Room	
Description	*Students should be able to book dorm rooms by searching and selecting available room options.*
Precondition	*Student has successfully logged into the system.*
Basic Flow	*The student accesses the booking interface after successfully searching for a room, upon applying for a room the system logs the record and sends a confirmation email with room details.*
Post Condition	*The booking transaction is successfully processed and saved in the system.*

Use Case 3: View Application	
Description	*A student and/or housing admin should be able to view a student's application.*
Precondition	*Student and/or housing admin have successfully logged on to the system and can select the view application feature..*
Basic Flow	*After successfully logging into the system, the student is taken to the view application feature and can view the details of a student's application/s.*

Post Condition	*Show the application/s made by a student.*

Use Case 4: Edit Application	
Description	*A student should be able to make changes to the details of a dorm room booking or correct outdated information*
Precondition	*When the student is in the view application feature, they should be able to view the existing application/s.*
Basic Flow	*Upon viewing the application/s, the student will indicate that they want to modify booking information. Next, the system will allow the student to edit their application details.*
Post Condition	*The changes should be saved and reflected in the system so it can be visible to students and housing staff.*

Use Case 5: Track Application Status	
Description	*Students should be able to monitor the status of their dormitory application .*
Precondition	*The student must sign up to submit a dormitory application through the system. The student has successfully logged on to the system.*
Basic Flow	*After logging on to the system, the student navigates to the application status section. Then the system displays the current status of the dormitory application (e.g., Pending, Application Approved/Declined, Payment Under Review, Room Booked). The system automatically updates the application status based on actions taken by the Housing Office Staff. If there is a change in the application status, the system generates a status update.*
Post Condition	*The student can view their application status and access all logged status updates and comments from Housing Office Staff.*

Use Case 6: Send Status Update	
Description	*The system should send timely notifications to students about updates or changes.*
Precondition	*The student must be registered in the system using their school email and username.*
Basic Flow	*Students receive timely updates about their dorm room status while verifying their registration*
Post Condition	*When a student makes any updates or changes regarding the dorm room information, the system will send a notification to the student's registered email.* *The notification has been logged in the system for future auditing and reference.*

Use Case 7: Upload Payment Receipt	
Description	*Students should be able to upload a copy of their payment receipt as proof of payment for dorm fees to confirm their financial obligations*
Precondition	*The student has successfully logged on to the system.*
Basic Flow	*After logging on to the system,The student accesses the payment receipt upload feature on the track application page. Then, the student selects a file in an accepted format (PDF, JPEG, PNG), and uploads it. The system then verifies the file, and provides confirmation of a successful upload.*
Post Condition	*The payment receipt is successfully uploaded, verified, and securely stored, linked to the student's profile for housing staff review.*

Use Case 8: Assign Dorm Room	
Description	*The housing office staff shall assign a dorm room to a student based on their booking request and application status.*
Precondition	*The student has successfully submitted an application and payment receipt.* *The dorm room availability must be updated in the system.*
Basic Flow	*The housing office staff logs into the UWIDormFinder application.*

	The staff accesses the list of pending student applications. *The staff reviews a student's application and verifies the payment status.* *The staff assigns a suitable available dorm room to the student.* *The system updates the application status to "Assigned."* *A notification is sent to the student, confirming their dorm room assignment.*
Post Condition	*The student's application status is updated to reflect the room assignment.* *The dorm room is marked as "occupied" in the system.* *The student receives a confirmation of the assignment.*

Use Case 9: Maintain System Log	
Description	*This system should be able to log all key activities related to dorm assignments, including bookings, status changes, modifications, and room assignments.*
Precondition	*The system is initialized and running.* *The user is authenticated and authorized to access the system.*
Basic Flow	*The system logs all key activities related to dorm assignments, including bookings, status changes, modifications, and room assignments. This ensures a complete record of all actions taken within the system.*
Post Condition	*All actions performed by authorized users shall be logged and securely stored. The log entries shall be accessible only to authorized personnel for auditing and review purposes.*

Use Case 10: Create Admin Account	
Description	*The IT staff creates admin account to ensure comprehensive activity for dorm room bookings.*
Precondition	*The system is initialized and running.* *The user is authenticated and authorized to access the system.*
Basic Flow	*The system logs all key activities related to dorm assignments, including bookings, status changes, modifications, and room assignments. This ensures a complete record of all actions taken within the system.*
Post Condition	*All actions performed by authorized users shall be logged and securely stored. The log entries shall be accessible only to authorized personnel for auditing and review purposes.*

3 Other Non-Functional Requirements

3.1 Performance Requirements

- The system shall process and display search results for room availability within 10 seconds 95% of the time to ensure a smooth user experience.
- The system must be scalable to handle peak periods, such as the start of the academic semester, with minimal latency.
- The system shall update room availability and assignment records in real time, with a maximum data processing delay of 10 seconds.

3.2 Safety and Security Requirements

- The system shall include basic error-handling mechanisms to alert users and administrators if an issue occurs, such as failed room assignment or login errors.
- The system shall require user authentication for students and housing staff to access the application. A basic username and password system will be implemented and encrypted before being stored.
- Different levels of access shall be defined (e.g., student, housing staff) to ensure only authorized users can modify room assignments or view certain data.
- The system shall log important actions (e.g., room assignments, account changes) to provide basic auditing capabilities i.e. to log all data for future reference by administrators.

3.3 Software Quality Attributes

Usability, Portability and Reliability

- The system shall be easy for students and housing staff to use, with a clear and intuitive user interface by keeping students informed about the stats of their room applications.
- The software shall run on multiple platforms, including Linux, Windows, and macOS, when accessed via a laptop or desktop.
- The system shall operate consistently under expected conditions and handle errors gracefully.

Implementation Plan:

- Use familiar design patterns and layouts to minimize the learning curve, ensuring users can perform core tasks (e.g., viewing room availability or assigning rooms) with minimal training.
- Test the system on various OS environments during the development phase to confirm compatibility.
- Include basic error handling mechanisms to alert users when issues arise (e.g., failed database connections or login errors).
- Implement basic testing procedures, such as unit tests, to ensure that core functionalities (e.g., room search, assignments) work as expected.

Appendix

A survey/ questionnaire (for the students) and an interview (for the housing/administrative staff) were done to elicit the requirements for the UWIDormFinder system. The survey was conducted on October 22nd, 2024 via Google Forms, and distributed via WhatsApp to approximately 20 students residing in the UWI dorms. Additionally, the interview was conducted on October 29th, 2024 from 1:09 pm to 4:51 pm at the the Irvine Hall with Mrs. Simone Williams, SSDM of Irvine Hall.

Appendix A- Interview Questions

University Administration and Housing/Dormitory Management Team

1. Are there any existing systems or tools that you are using for room allocation or booking management?
2. What criteria does the university use to allocate rooms to students (e.g., year of study, course, financial status, preferences)?
3. Are there any special considerations for students with disabilities, international students, or other specific groups?
4. What student data do you need to collect for room allocation (e.g., personal details, room preferences, roommate requests)?
5. Are there any other specific privacy or data protection concerns that need to be addressed? google form to excel…
6. How should room availability be displayed to students (e.g., by dorm type, price, amenities)?
7. What kind of administrative control would you need over the system (e.g., manually overriding room assignments, approving/denying bookings)?
8. Should the dorm booking system integrate with other university systems (e.g., student records, payment systems, etc.)?

Transcript of Interview

University Administration and Housing/Dormitory Management Team

1. Are there any existing systems or tools that you are using for room allocation or booking management?

   To reside in Irvine, individuals must apply through the University of the West Indies (UWI), as the selection process is managed by the Student Services and Development Manager (SSDM). They are responsible for determining which students are granted accommodation in Irvine and assigning them to specific rooms.

2. What criteria does the university use to allocate rooms to students (e.g., year of study, course, financial status, preferences)?

   For room allocation, only basic student data is collected, focusing on essential information such as ID number, age, name, and place of origin. The data collection process is minimal and does not extend to co-curricular activities or more detailed personal information.

3. Are there any special considerations for students with disabilities, international students, or other specific groups?

   Irvine Hall is known to be the “disability hall” as they offer dedicated rooms for students with disabilities, which were initially established for Old Irvine Hall and continue to be a priority. These disability-friendly rooms are designed to meet accessibility needs, with features such as larger space, private bathrooms, lowered fixtures, and support rails, making them suitable for wheelchair users and others with physical disabilities. Additionally, the hall accommodates non-physical disabilities, such as autism or Tourette’s syndrome. To ensure inclusivity, Irvine Hall aims to match the number of disability-adapted rooms to the demand, creating as many as needed to serve students with disabilities.

4. What student data do you need to collect for room allocation (e.g., personal details, room preferences, roommate requests)?

For room allocation, only basic student data is collected, focusing on essential information such as ID number, age, name, and place of origin. Notably, information regarding disabilities is not part of the standard application process, which has sometimes led to situations where administrators were unaware of a student's disability upon their arrival. While there is acknowledgment that including disability information could improve the support provided to students, it has yet to be implemented.

5. How should room availability be displayed to students (e.g., by dorm type, price, amenities)?

   Yes, I would like room availability to be displayed by dorm type, price, and amenities. Additionally, I agree that it would be a good idea to identify the disability friendly rooms

6. What kind of administrative control would you need over the system (e.g., manually overriding room assignments, approving/denying bookings)?

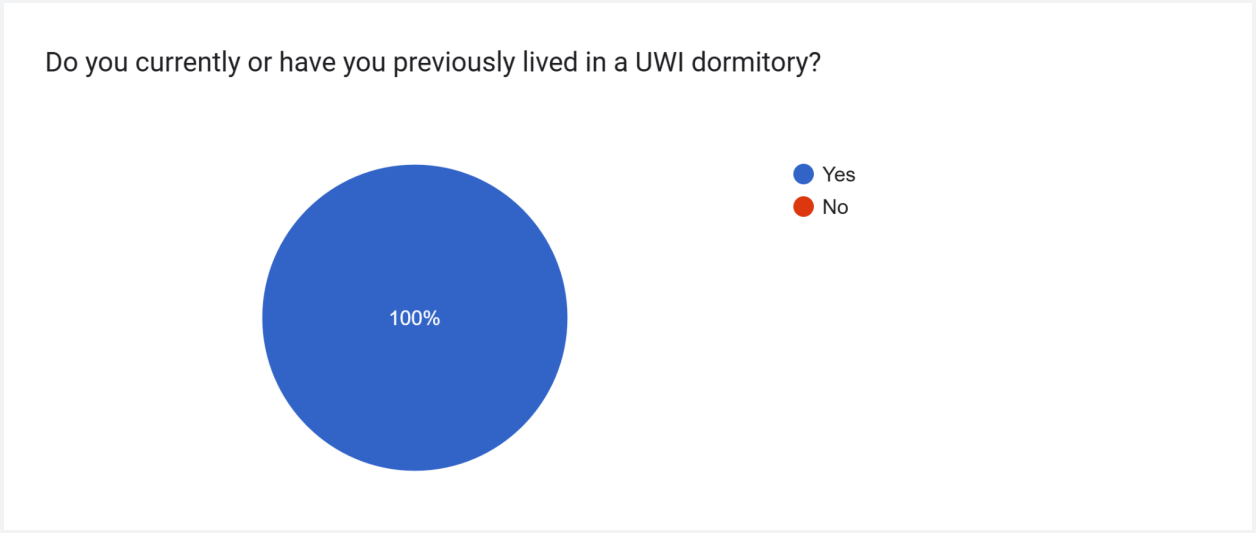
   Yes, It would be nice to simply click approve or deny bookings based on student application status. After that, we would send the offer letter with a deadline for payment, and then you could decide whether to accept or decline the offer.

Appendix B- Survey Questions

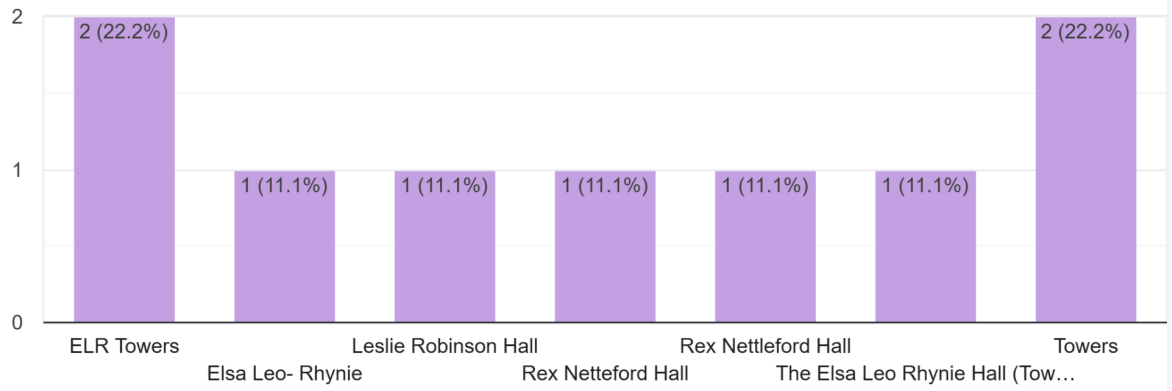
Students

1. Do you currently or have you previously lived in a UWI dormitory? If yes, which dormitory ?
2. How long have you been living in UWI dorms?
3. How would you rate your overall experience with the current dorm room booking process? 1 - Very Dissatisfied, 2- Dissatisfied, 3 - Neutral, 4 - Satisfied, 5 - Very Satisfied
4. How do you currently find and book dorm rooms at UWI? What challenges do you face with the current process?
5. Did you feel the online advertisement was deceiving? If yes, what could be a better approach?
6. On a scale of 1 to 5, how much would you prefer a user-friendly software solution for booking dorm rooms? 1 - Not at all preferred, 2 - Slightly preferred, 3 - Moderately preferred, 4 - Very preferred, 5 - Extremely preferred
7. Would you like the option to view pictures and descriptions of the dorms before booking?
8. How important is it for you to see real-time availability of rooms? 1 - Not Important, 2 -Somewhat Important , 3 -Important , 4 - Very Important
9. What filters or criteria would you like to use when searching for a dorm room?9.What personal information do you think should be part of your profile?
10. How would you like to receive booking confirmations?
11. Provide specific details regarding any additional features or functionalities you would like to see in the software.

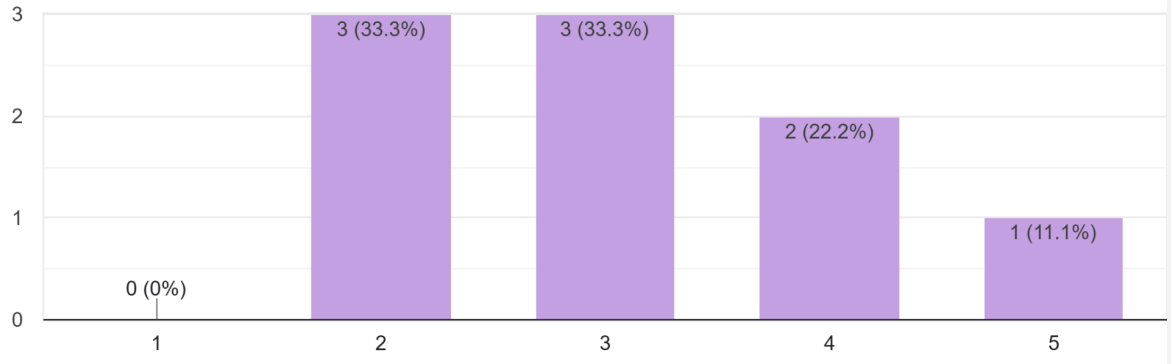
Summary of Survey Responses



If yes, which dormitory ?



How would you rate your overall experience with the current dorm room booking process? 1 - Very Dissatisfied 2- Dissatisfied 3 - Neutral 4 - Satisfied 5 - Very Satisfied



Did you feel the online advertisement was deceiving? If yes, what could be a better approach?

No

They could give a better description of the layout of the dorm

No not really

List of current Amenities not once's they will implement in the future

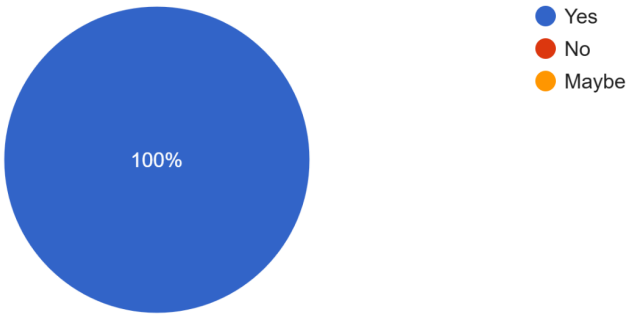
In a sense, yes. To me a better approach would be to update and utilize current technology. Most of the photos are from years ago. A more up-to-date system, would make the process way easier and efficient. To relay my personal experience, based on the images displayed , the rooms seemed very small and in reality it's way more spacious. An updated system could easily change the selection process for students.

Yes

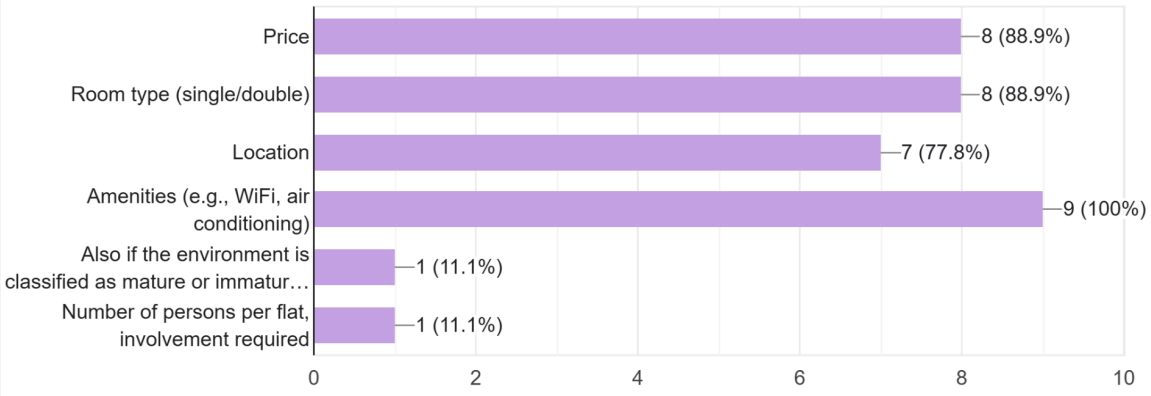
There is literally nothing to go off when choosing a hall apart from searching their IG profiles high and low. Some content you can find elsewhere are also outdated.

Slighty...it would help to know what exactly will be provided on hall and how many persons will be sharing the amenities

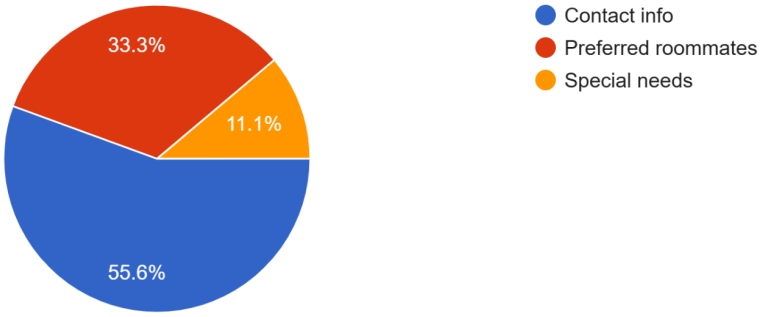
Would you like the option to view pictures and descriptions of the dorms before booking?



What filters or criteria would you like to use when searching for a dorm room?



What personal information do you think should be part of your profile?



As a first year CS student, I had this project idea in mind after having a terrible uwi application experience as well as applying for residence. I think students should be able to apply to as many halls as they want and get their decisions and then choose from those. Students having the ability to see how many rooms are left during the application period may persuade them to act promptly. We should have at our disposal all that we need to know about a hall: what the rooms, kitchen, toilets, laundry areas are like so we can make an informed decision. Should be required by every hall. Being able to pay for hall fees through the platform would be desirable. Also, students should be able to file complaints and create tickets for issues to be addressed probably in their room or floor or wherever else. Some form of live chat/messaging too.

Provide specific details regarding any additional features or functionalities you would like to see in the software

A view where the user can move around the room to get a good feel of the place from different angles

Rent payment updates

NA

In my opinion a slot should be allocated to, check or view your: Accommodation, status.

For example, "Your application is under review...your application is now being finalized" etc...

A likability scale, this would be where you can see your chance of getting a room on specific Halls of residence, based on time frame, space availability and your overall application. The persons who review applications could complete this for each applicant. This saves time on both ends and also allows applicants to venture off into Halls of residence that are more geared towards them. And of course make the preparations process more easier and gives the applicant a sense of confirmation.

Time it takes to travel from different faculties to hall

Software Design Specification

for

UWIDormFinder System

Version 1.2

Prepared by

Rashene Dillon	**620161588**	**rashene.dillon@mymona.uwi.edu**
Damion Henry	**620071079**	**damion.henry02@mymona.uwi.edu**
Shantay Kellyman	**620157798**	**shantay.kellyman@mymona.uwi.edu**
Shari Oliver	**620156656**	**shari.oliver02@mymona.uwi.edu**
Samantha Samuels	**620154205**	**samantha.samuels03@mymona.uwi.edu**
Tahmia Vincent	**620156659**	**tahmia.vincent@mymona.uwi.edu**

Course Instructor:	**Dr.Ricardo Anderson**
Course:	***COMP2140 – Introduction to Software Engineering***
Studio Facilitator:	**Mr. Aldane Milton**
Date:	***November 28, 2024***

TABLE OF CONTENTS

1.0 Project Overview …… 3

2.0 Architectural Design

2.1 General Constraints …… 4

2.2Alternative Considered …… 4 - 5

2.3 System Architecture Diagram ……..…… 5

2.3.1 Architectural Description …… 6-7

2.3.2 Architectural Justification …… 7

3.0 Architecture Decomposition

3.1 Architecture Decomposition …… 8-9

3.2 Structural Design …… 10

3.3 Design Notes ……..….. 11 - 12

1.0 Project Overview

The Irvine Hall at the University of the West Indies (UWI), Mona, is dedicated to fostering a dynamic and supportive community for students residing on campus. However, the hall currently faces challenges due to an inefficient dorm room booking system. The key issues include the lack of application status tracking and limited room descriptions and visuals to assist students in making informed decisions. Moreover, staff have limited access to student information, such as students' disabilities, which makes it difficult to assign appropriate accommodation. The system also lacks real-time room availability updates which hinders effective capacity management. Consequently, the proposed UWIDormFinder system aims to resolve these problems by offering a user-friendly, efficient solution that enhances the student experience and simplifies administrative tasks. The primary users are the students, who are technically proficient and expect seamless, timely notifications. The other primary users are the University Administration/Housing Management team, who are responsible for overseeing room allocations and setting housing policies, and will utilise the system for reports and decision-making. Mona Information Technology Services (MITS) are the secondary users and will handle the system's deployment, maintenance, and security to ensure its reliability.

The UWIDormFinder system is designed to streamline administrative processes, improve communication, and foster a more organized and supportive community at Irvine Hall. The system will allow students to search for available dorm rooms, track the status of their application, upload proof of payment in the form of digital receipts and receive notifications about room assignments and updates. The students and housing staff will be able to edit room booking details, update student assignments, and view students' proof of payment in the form of digital receipts. The housing staff will also have the ability to assign rooms to students and make changes when necessary.Mona Information Technology Services (MITS) will manage the system logs, ensuring an accurate record of all actions performed within the system.

To ensure a smooth user experience, the system will allow the processing of room availability searches and displaying of results within 10 seconds 95% of the time. It must also be scalable to handle peak periods, such as the start of the academic semester, with minimal latency. The system will be able to real-time updates for room availability and assignment records, with a maximum data processing delay of 10 seconds.

2.0 Architectural Design

2.1 General Constraints

Several global limitations and constraints influence the design of the UWIDormFinder program, affecting its functionality, performance, and cross-platform accessibility. First, the hardware and software environments present compatibility constraints that influence the system's architecture. Because the application is designed to run on a wide range of devices, including Windows, Linux, and macOS, it must support several browser versions (e.g., Chrome 90+, Firefox 85+, and Safari 14+) and be responsive to different screen sizes on desktops and laptops. The minimum hardware requirements for each platform are 4GB of RAM and 256GB of storage. These parameters affect the application's performance since it must give a smooth user experience on devices with restricted processing power, memory, and storage.

Network requirements are also important, as the application is web-based and requires persistent internet connectivity. Users in places with limited or unstable networks may encounter delays in loading sites or refreshing data, particularly if bandwidth-intensive features such as real-time updates are enabled. To address this, the application is designed to reduce data transmission by utilizing lightweight assets and efficient caching algorithms, balancing performance and usability even in slower network situations.

Furthermore, interface requirements necessitate a user-friendly and accessible design that adheres to the MVT design pattern, ensuring separation of concerns and allowing for maintainability. Given that different user roles—such as students, housing administrators, and IT staff—interact with the system, the application includes multiple interfaces tailored to these roles, with security measures to restrict access accordingly. Data privacy and security standards are also critical, requiring the use of encrypted connections, secure authentication, and controlled data storage in SQLite. These constraints together ensure that UWIDormFinder is accessible, performant, and secure across diverse user environments.

2.2 Alternatives Considered

In evaluating architectural patterns for UWIDormFinder, two alternatives were considered: the **Layered Architecture Pattern** and the **Repository Pattern**, but each presented limitations compared to the chosen Model-View-Template (MVT) pattern, which aligns well with Flask’s design principles and the project’s requirements.

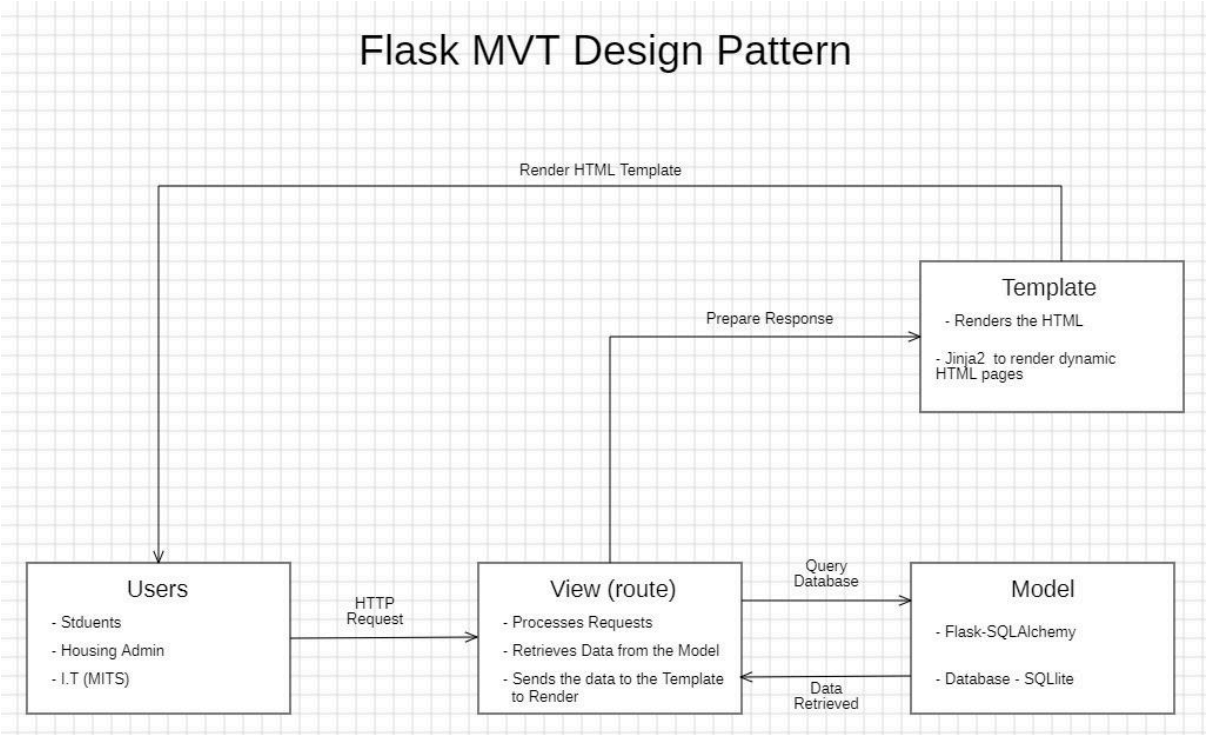
The **Layered Architecture Pattern** separates concerns and improves maintainability by dividing the program into layers, which typically include a presentation layer, business logic layer, and data access layer. This design is commonly employed in complicated systems that require rigorous separation and controlled interaction between levels. However, it was found less appropriate for UWIDormFinder since MVT better supports the application's emphasis on rapid development and simplicity. With Flask’s built-in capability, MVT easily integrates views and templates, making it more effective for producing web pages and managing data without the need for additional inter-layer interactions. Implementing a layered design may

increase unneeded complexity by increasing overhead for data transfer and communication across layers, which is overkill for a dorm management application.

The **Repository Pattern** abstracts database interactions, separating the business logic from the data access layer. This can be useful in applications with various data sources or sophisticated queries. However, because UWIDormFinder employs Flask's Object-Relational Mapping (ORM) technology, the Repository Pattern would introduce redundancy. Flask's ORM already includes a sophisticated data management layer that handles database queries and updates automatically and integrates neatly with the MVT structure. Adding a repository layer would complicate the architecture without providing significant benefits, as it would duplicate most of the functionality already provided by Flask's ORM. This makes the MVT pattern, which natively supports Flask's ORM, a more efficient and natural fit for UWIDormFinder.

2.3 System Architecture Diagram

The Diagram Showing the Design Pattern for the UWIDormFinder System and the Relationship Between the Components



2.3.1 Architectural Description

In the Flask Model-View-Template (MVT) design pattern for UWIDormFinder, each component Model, View, and Template plays a distinct role in ensuring both functional and non-functional requirements are met effectively.

Model

The Model component manages data exchanges and provides a direct interface to the database. In Flask, the ORM (Object-Relational Mapper) allows the Model to communicate with the database by mapping classes directly to database tables. Models for UWIDormFinder will represent critical entities such as students, dorm rooms, bookings, and user responsibilities (for example, Housing Administrators and IT personnel). Each model provides fields and relationships, which Flask transforms into related tables and columns in Microsoft SQLite, allowing for efficient and structured data storage. This design choice addresses functional needs by providing reliable data management and retrieval. Flask’'s ORM meets non-functional requirements like maintainability by abstracting SQLite syntax and reducing database complexity, making changes easier and code more readable.

View

The View component handles HTTP requests from the user, interacting with the Model to retrieve or change data before routing the results to the appropriate Template for rendering. Views in UWIDormFinder will manage actions such as room searches, booking requests, application status tracking, and role-specific functionality. For example, when a student requests to see available dorm rooms, the view collects the data from the Model, arranges it, and passes it to the Template for presentation in a user-friendly fashion. Views are useful for receiving user inputs, processing them using business logic, and controlling data flow. By isolating functionality from display, the View component improves system efficiency and scalability while also addressing non-functional requirements by handling interactions efficiently and lowering reaction times.

Template

The Template component renders HTML pages given to the user's browser. Each Template specifies the structure and styling of a page, maintaining uniformity across devices and screen sizes. Templates in UWIDormFinder are used to display pages like the room booking form, application status page, and admin dashboards. Flask Templates utilize a templating language to dynamically insert data acquired from the Model via the View, ensuring that each page is tailored to the user's requirements. Templates help to address non-functional needs such as usability and accessibility by providing clean, well-structured, and responsive pages that allow users to traverse and interact with the system smoothly.

Interaction Between Components

These components work together to ensure UWIDormFinder meets functional requirements, such as enabling students to search for rooms and submit applications, and non-functional requirements, like performance and security. For example, when a student searches for

available rooms, the request is handled by the View, which communicates with the Model to retrieve data, organizes it, and passes it to the Template. The Template then renders an HTML page displaying the results to the student in a structured and user-friendly manner. This interaction loop, facilitated by Flask's MVT architecture, reduces development complexity, speeds up response times, and allows for efficient data management.

2.3.2 Architecture Justification

The Model-View-Template (MVT) architecture is ideally suited for the UWIDormFinder system due to its alignment with Flask's core framework and its ability to support rapid development, modularity, and maintainability. Compared to other architectural patterns, MVT directly integrates with Flask's built-in functionality, allowing efficient data handling, rendering, and user interaction while reducing code complexity and development time.

Alignment with Flask Framework

Flask natively supports MVT, which contains a sophisticated Object-Relational Mapper (ORM) and templating engine for efficient data management and front-end rendering. Using MVT, UWIDormFinder can easily handle database interactions within the Model component, leveraging Flask's ORM. This built-in functionality eliminates the need for a separate data access layer, which would increase complexity without yielding significant benefits. As a result, the system acquires a direct and efficient means to manage data across entities such as students, dorm rooms, and booking requests, making MVT an ideal fit.

Modularity and Separation of Concerns

By dividing the system into three separate parts Model, View, and Template each with a clear function, the MVT architecture improves modularity. The Template creates user-friendly HTML pages, the View handles business logic and request-response cycles, and the Model controls database interactions. Because one component (such as the database schema in the Model) can be updated with little effect on other components, this division of responsibilities enhances maintainability. Additionally, this modularity facilitates scalability, enabling the implementation of future features or expansions (such report creation or new user roles) without requiring significant architectural modifications.

Efficient Development and Performance

MVT is made for online applications that prioritize flexibility, quick prototyping, and effective data management. This architecture enhances system responsiveness and user experience for UWIDormFinder by facilitating effective data retrieval, processing, and presentation in a simple flow. For instance, the View can manage particular user requests (such as looking for rooms) and promptly provide the required information to Templates for rendering. Response times are shortened and system performance is guaranteed, satisfying non-functional objectives like responsiveness and speed thanks to this optimized approach. MVT also makes templating and reusing code easier, which speeds up development and enhances system consistency.

3.0 Architecture Decomposition

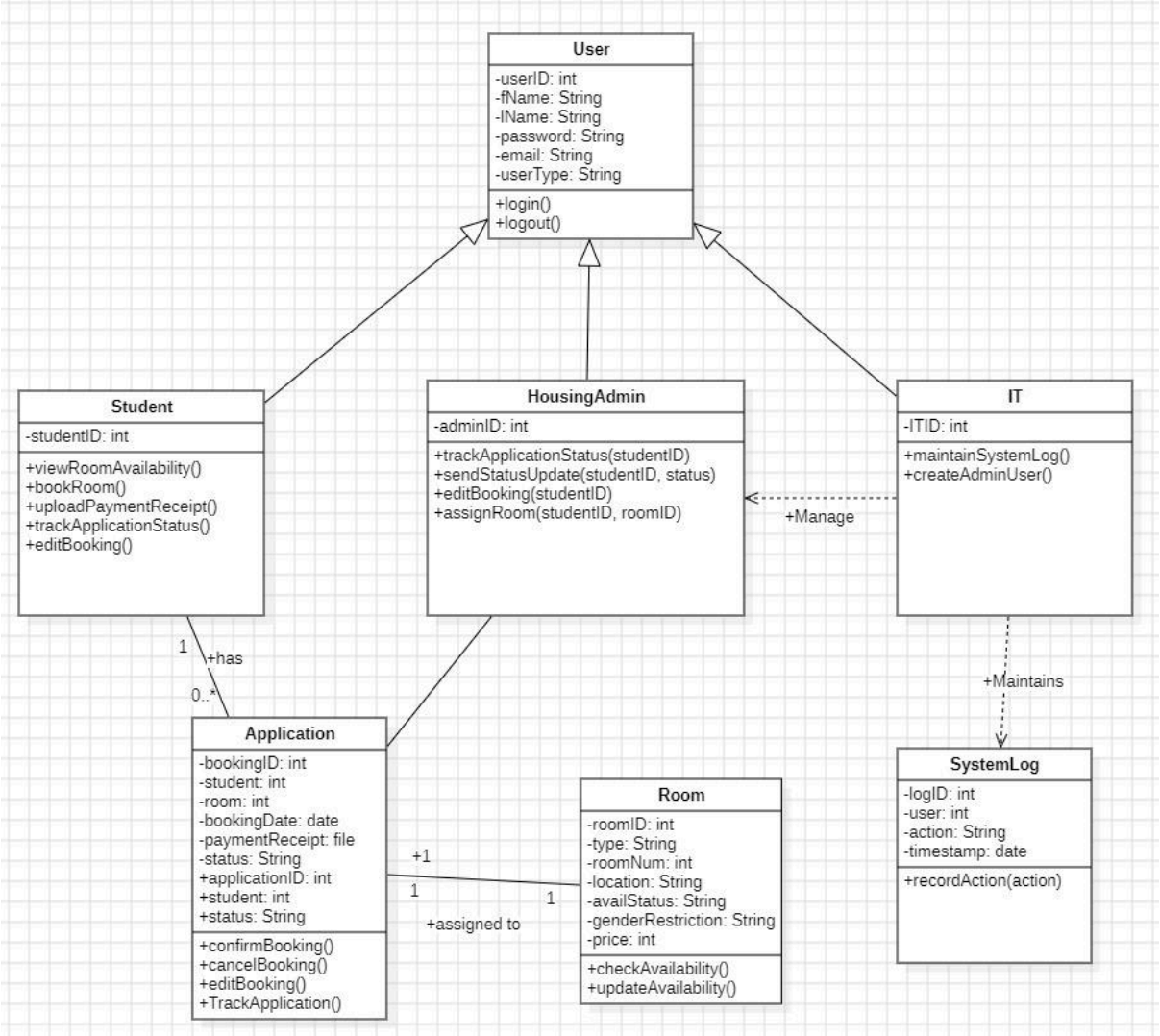
3.1 Component Decomposition – Classes

Requirement ID	**Architecture Component**	**Class**	**Description**
Req. 1	User	Student	This allows the student to search for available rooms
Req. 2	User	Student	This allows the student to book a dorm room.
Req. 3	User	Student HousingAdmin	This allows the student to view dorm booking information This allows the admin to view dorm booking information
Req. 4	User	Student HousingAdmin	This allows the student to edit dorm booking information This allows the admin to edit dorm booking information
Req. 5	User	Student HousingAdmin	This allows the student to track their application This allows the admin to track a student application
Req. 6	User	Student HousingAdmin	Allows admin to send application status updates.

Req. 7	User	Student	This allows the student to upload a payment receipt when booking a dorm room.
Req. 8	User	Housing Admin	This assigns a student to a room.
Req. 9	User	IT	This allows the IT (MITS) to maintain the system log
Req. 10	User	IT	This allows IT (MITS) to create an admin account.

3.2 Structural Design – <Class Diagram / Structure Char not both>

The Diagram Showing the Class Diagram for the UWIDormFinder System and the Relationship Between the Classes



3.2.1 Design Notes

The UWIDORMFINDER class diagram shown above illustrates the relationships and responsibilities of various classes and modules, focusing on the interactions between different user roles (Student, Housing Admin, and IT) and the core functionality of the system (Booking, Room, Application, and SystemLog management). Below is an explanation of the diagram, covering relationships, assumptions, and design constraints:

Relationships Between Classes/Modules

1. **User Inheritance**
    - **Student**, **HousingAdmin**, and **IT** classes inherit from the **User** superclass, which contains common attributes such as userID, fName, lName, password, email, and userType, along with methods like login() and logout().
    - This inheritance relationship is essential for sharing common user attributes and methods across all user types, reducing redundancy and allowing polymorphism when handling users in the system.
2. **Composition and Association**
    - Students have an association with Booking, as indicated by the "0..*" multiplicity, meaning a student can have zero or more bookings. The Student class also has methods like bookRoom(), uploadPaymentReceipt(), and editBooking() to interact with the booking system.
    - **HousingAdmin** manages **Application** objects, with a "1" multiplicity, signifying that each **Application** is linked to a single **HousingAdmin**. This relationship supports Housing Admin functionalities such as tracking application status and assigning rooms.
    - **IT** has an association with **SystemLog**, allowing IT personnel to maintain the system logs. The **SystemLog** module logs user actions, enabling audit and tracking capabilities for the system.
3. **Aggregation**
    - Booking aggregates Room instances, as seen in the "0..*" multiplicity from Booking to Room, representing that a booking can be associated with zero or one room. Room has attributes like roomID, type, location, and availability, supporting availability checks and room assignment.
    - **Application** is also linked to **Student**, where each student can have one application. The **Application** class includes methods to update application status, making it easier for both students and Housing Admin to track application progress.

Assumptions

- **Unique User Roles:** The system assumes that users fall into one of three roles: Student, Housing Admin, or IT. These roles determine access privileges, such as the ability to manage applications, book rooms, or maintain logs.
- **One Active Application Per Student**: Each student has one active application at a time, simplifying tracking and status updates.
- **Room Constraints**: Each room has specific attributes (e.g., genderRestriction), assuming that room assignment may depend on additional criteria such as gender.

Constraints

- **System Security and Authorization**: Given that different users have varying levels of access, the design requires a secure login system with role-based permissions to ensure that only authorized users access certain data and functions.
- **Data Consistency and Referential Integrity**: The relationships between **Student**, **Booking**, **Application**, and **Room** require data consistency. For instance, a booking should not exist without a corresponding room, and an application must be associated with a valid student.
- **Log Maintenance for Audits**: **SystemLog** was included as a dedicated component for IT to track actions. This assumes the system requires consistent logging for audit and debugging, impacting performance and data storage needs.

Why Certain Relationships Were Specified

- **Inheritance for User Roles:** This inheritance hierarchy provides a flexible way to manage user roles, avoiding duplicate attributes for each role and centralizing login functionality.
- **Booking and Room Aggregation**: This design choice reflects real-world dorm booking systems, where bookings are assigned rooms based on availability and user preferences, simplifying room allocation.
- **Application Management by Housing Admin**: This relationship allows Housing Admin to effectively track and update application statuses, a necessary feature for managing dorm applications and ensuring students are aware of their application progress.

Overall, the design reflects the primary requirements of the UWIDORMFINDER system while incorporating essential constraints and assumptions to maintain functionality, security, and data integrity.

Requirement #: 1 - Search Room Availability

Test Case ID	Test Case Description	Test Input (Accept)	Test Input (Reject)
TC01	**Search without selecting any filter but submitting the form**	None selected (should show all available rooms)	No Rejection
TC02	**The Appropriate results should display**	Room Type= Single, Dormitory Building = Building 3, Preferred Floor = 2nd Floor	No Rejection

Requirement #2 - Book Dorm Room

Test Case ID	Test Case Description	Test Input (Accept)	Test Input (Reject)
TC03	**Valid application submission**	Complete form with valid student ID, first name, last name, email, telephone, gender, education level, program type, reason for applying, co-curricular activities, and agreement.	Missing any required field or invalid data (e.g., empty fields or invalid email).
TC04	**Booking a room that the student has already booked**	Room was already booked by the student.	Room ID already exists in the Application table for this student (i.e., student cannot book the same room twice).
TC05	**Student ID mismatch**	Student login ID is different from, input ID.	Student CANNOT apply for another student
TC05	**Missing first name**	First Name = "John"	First Name = "" (empty first name).

Test Case ID	Test Case Description	Test Input (Accept)	Test Input (Reject)
TC06	**Missing last name**	Last Name = "Doe"	Last Name = "" (empty last name).
TC07	**Missing email address**	Email = "student@example.com"	Email = "" (empty email).
TC08	**Invalid email format**	Email = "student@example.com"	Email = "student.com" (invalid email format).
TC09	**Missing telephone number**	Telephone = "1234567890"	Telephone = "" (empty telephone number).
TC10	**Missing gender**	Gender = "Male"	Gender = "" (empty gender).
TC11	**Missing education level**	Education Level = "Undergraduate"	Education Level = "" (empty education level).
TC12	**Missing program type**	Program Type = "Full-time"	Program Type = "" (empty program type).
TC13	**Missing reason for applying**	Reason for applying = "Want to live on campus"	Reason for applying = "" (empty reason for applying).
TC14	**Agreement not accepted**	Agreement = True (checkbox checked)	Agreement = False (checkbox not checked).

Requirement #: 3 - View Student Room Application

Test Case ID	Test Case Description	Test Input (Accept)	Test Input (Reject)
TC15	**Only successfully login students should be able to access this page**	Login with a registered student account	Try to login with a unregistered student

Requirement #: 4 - Edit Student Room Application

Test Case ID	Test Case Description	Test Input (Accept)	Test Input (Reject)
TC16	**Only successfully login students should be able to access this page**	Login with a registered student account	Try to login with a unregistered student
TC17	**Missing email address**	Email = "student@example.com"	Email = "" (empty email).
TC18	**Invalid email format**	Email = "student@example.com"	Email = "student.com" (invalid email format).
TC19	**Missing telephone number**	Telephone = "1234567890"	Telephone = "" (empty telephone number).
TC20	**Missing gender**	Gender = "Male"	Gender = "" (empty gender).
TC21	**Missing education level**	Education Level = "Undergraduate"	Education Level = "" (empty education level).
TC22	**Missing program type**	Program Type = "Full-time"	Program Type = "" (empty program type).
TC23	**Missing reason for applying**	Reason for applying = "Want to live on campus"	Reason for applying = "" (empty reason for applying).

Test Case ID	Test Case Description	Test Input (Accept)	Test Input (Reject)
TC24	**Agreement not accepted**	Agreement = True (checkbox checked)	Agreement = False (checkbox not checked).

Requirement # :5 - Track Application Status

Test Case ID	Test Case Description	Test Input (Accept)	Test Input (Reject)
TC25	**Only successfully login students should be able to access this page**	Login with a registered student account	Try to login with a unregistered student
TC26	**Track Application**	A student should have a booking in progress in order the track an application	The students don't have a booking in progress

Requirement #: 6- Send Student Status Updates

Test Case ID	Test Case Description	Test Input (Accept)	Test Input (Reject)
TC27	**Send update when a application is submitted for a room**	Send pending update once the application is submitted	Application was not submitted
TC28	**Send update when a application is approved/declined**	Admin must have approved or decline application	Admin did not approved or decline application

Requirement #: 7 - Upload Payment Receipt

Test Case ID	Test Case Description	Test Input (Accept)	Test Input (Reject)
TC29	**Upload Payment Receipt**	Application is in Approval Status (Admin approved application)	Application still in Pending status (Admin did not approve application)
TC30	**Upload correct file format**	Students upload a PDF, JPG or PNG file	Students try to upload a file format that is not PDF, JPG or PNG

Requirement #: 8- Assign Dorm Rooms

Test Case ID	Test Case Description	Test Input (Accept)	Test Input (Reject)
TC32	**Assign a Room**	Admin user assigns a room to a student	Admin user did not assigned a room to the student

Requirement #: 9 - Maintain System Log

Test Case ID	Test Case Description	Test Input (Accept)	Test Input (Reject)
TC33	**Access System log**	Only IT user is able to access System Logs	User is not an IT user

Requirement #: 10 - Create Admin Account

Test Case ID	Test Case Description	Test Input (Accept)	Test Input (Reject)
TC34	**Accessing Create Admin Account**	Only IT user is able to access the Create Admin Account page	User is not an IT user
TC35	**Missing Staff ID**	Staff ID = 12345678	Staff ID = “” (empty staff ID)
TC36	**Existing Staff ID**	No staff have the same ID	There is a Staff with the same ID number
TC37	**Staff ID is not a number**	Staff ID is of the form XXXXXXXX Where XXXXXXXX is and integer	Staff ID is not of the form XXXXXXXX where XXXXXXXX is a integer
TC38	**Missing first name**	First Name = "John"	First Name = "" (empty first name).
TC39	**Missing last name**	Last Name = "Doe"	Last Name = "" (empty last name).
TC40	**Missing email address**	Email = "student@example.com"	Email = "" (empty email).
TC41	**Invalid email format**	Email = "student@example.com"	Email = "student.com" (invalid email format).
TC41	**Missing telephone number**	Telephone = "1234567890"	Telephone = "" (empty telephone number).
TC42	**Missing gender**	Gender = "Male"	Gender = "" (empty gender).

UWIDormFinder Project Libre

Our structured plan for this entire project has been outlined below.

Project Conceptualization

Date: September 9th - 19th
Time: 10 days
Description: Initial brainstorming and idea validation. This phase includes discussions on project feasibility, stakeholder engagement, and high-level goals.

Project Proposal

Date: September 19th - 27th
Time: 8 days
Description: Formalizing the project idea into a proposal document, outlining objectives, scope, timeline, and initial resource requirements.

Establishing Functional Requirements

Date: October 17th - 24th
Time: 7 days
Description: Gathering detailed functional requirements through stakeholder consultation and team workshops. Deliverable: Functional Requirements Document.

UWIDormFinder Survey

Date: October 22nd, 2024
Time: 1 day
Description: Conducting a survey among students to collect feedback on essential features and usability expectations for the system.

Interview with Irvine Hall's SSDM

Date: October 25th, 2024
Time: 1 day

Description: Meeting with the Student Services and Development Manager (SSDM) to gather insights on dorm allocation challenges and system expectations.

System Requirements Specification (SRS) Document

Date: November 2nd, 2024
Time: 1 day
Description: Delivering the finalized SRS document based on gathered requirements and stakeholder feedback.

Software Design Presentation #3

Date: November 7th, 2024
Time: 1 day
Description: Presenting the software design to stakeholders for review and feedback.

Coding of the UWIDormFinder System Begins

Date: November 2nd, 2024
Time: Continuous until completion
Description: Initial development phase of the UWIDormFinder system based on the finalized SRS and Software Design Specifications.

Test Case Plan Development

Date: November 21st - 28th (in progress)
Time: 7 days
Description: Developing comprehensive test cases to ensure all functionalities meet user expectations and specifications.

Software Requirements Specification Document Version 1.2

Date: November 28th, 2024
Time: 1 day
Description: Delivering an updated version of the SRS document based on new insights and iterative feedback.

Software Design Specifications Document Version 1.2

Date: November 28th, 2024
Time: 1 day
Description: Delivering an updated version of the Software Design Specifications Document to address feedback and changes during the development phase.

Final Project Presentation

Date: November 29th, 2024
Time: 4:20 PM
Description: Presenting the final project to stakeholders, showcasing the system's features, functionalities, and user interface.

Our Additional Project Management Steps

These steps were the intermediate ones, implied but stated just for transparency with sir.

1. **Project Kickoff Meeting**
   Date: September 19th, 2024
   Time: 1 day
   Description: Hosting a kickoff meeting with the project team to establish roles, responsibilities, and expectations.

2. **Progress Review Meetings**
   Frequency: Weekly (before every tutorial session)
   Description: Regular meetings to track progress, address risks, and align deliverables with the project plan.

3. **Risk Assessment and Mitigation Plan**
   Date: October 24th, 2024
   Time: 1 day
   Description: Identifying potential risks and documenting mitigation strategies whilst developing the SDS document.

4. **Internal Testing Phase**
   Date: November 24th - 28th, 2024
   Time: 4 days
   Description: Conducting internal testing of the system using predefined test cases.

5. **Final Documentation Compilation (with SDS + SRS + Test Plan + Appendix stuff)**
   Date: November 28th - 29th, 2024
   Time: 1 day
   Description: Compiling all project-related documents, including the SRS, Design Specifications, and Test Plan, for submission.

GitHub Like

https://github.com/IAMCHEEZTRIX/uwidormfinder.git