# **Student Help Desk Application Documentation**

# **IMT2023596 – Dheeraj Muppiri**

# **IMT2023083 – Ashok Reddy**

# **IMT2023120 – Sasank Lingareddy**

# **IMT2023588 – Taral Sanka**

# **IMT2023527 – Thrilochan Reddy**

# **IMT2023558 – Akul Anhith**

## **Overview**

The Student Help Desk is a comprehensive application developed using Java and C++ with JNI (Java Native Interface) to manage and track various student queries and issues across different domains.

## **System Architecture**

### **Components**

1. **Java Frontend (Main.java)**
   1. User interface and input handling
   2. JNI bridge to native C++ implementation
2. **C++ Backend (projwithjni.cpp)**
   1. Core logic for query management
   2. Data structures for storing and processing queries
   3. Admin-specific functionalities

## **User Types**

### **1. Student**

Students can submit queries in the following categories:

* WiFi Issues

Router not working

WiFi speed slow

* Hot Water Issues

Not getting hot water during timing

Geyser not working

* Water Filter Issues
* Room Problems

Switchboard not working

Lightbulb not working

Broken furniture

Water leakage

Other specific issues

* Other miscellaneous issues

### **2. Admin**

Admins are categorized into three types:

## **a. Data Center Admin**

* Manages WiFi-related queries
* Can view, set resolve dates, and resolve WiFi issues

## **b. Property Manager**

* Manages:

Hot Water Issues

Water Filter Issues

Room Problems

* Can view, set resolve dates, and resolve issues in these categories

## **c. SAC**

* Manages other miscellaneous queries
* Can view, set resolve dates, and resolve other issues

## **Data Storage Architecture**

### **Core Data Structures**

The application uses sophisticated nested data structures, primarily leveraging C++ Standard Template Library (STL) containers:

## **1. Problem Report Storage**

The problemReport class maintains multiple unordered\_map containers for different query types:

unordered\_map<string, vector<string>> query\_wifi;

unordered\_map<string, vector<string>> query\_hotwater;

unordered\_map<string, vector<string>> query\_waterfilter;

unordered\_map<string, unordered\_map<string, string>> query\_roomproblem;

unordered\_map<string, vector<string>> query\_other;

## **Storage Pattern**

* **Key**: Usually the roll number or room number
* **Value**:

For most queries (WiFi, Hot Water, Water Filter, Other):

[0] - Room Number

[1] - Problem Description

[2] - Alert Status

For Room Problems:

Nested map with problem types as keys:

{

"Switchboard not working": problem details,

"Lightbulb not working": problem details,

"Broken furniture": problem details,

...

"Alert": alert status

}

### **Resolution Date Tracking**

The admin class uses a custom-hashed unordered\_map to track resolution dates:

unordered\_map<vector<string>, string, VectorHasher> resDate;

* **Key**: Vector containing [roll number, query type]
* **Value**: Resolved date

## **Key Features**

### **Query Management**

* Queries are stored in different data structures based on their type
* Each query is associated with:

Roll number

Room number

Problem description

Optional resolution status

### **Date Tracking**

* Administrators can set resolution dates for queries
* System checks if resolution dates are past due
* Generates alerts for overdue issues

### **Alert System**

* Queries that exceed their expected resolution time are marked with "Alert!!"
* Helps track and prioritize pending issues

## **JNI Integration**

### **Native Methods**

1. processStudentQuery
   1. Processes student submissions
   2. Handles different query types
   3. Creates appropriate query objects
2. adminViewQuery
   1. Retrieves and displays queries based on admin type
   2. Filters queries specific to each admin's domain
3. adminResolveDate
   1. Sets or removes resolution dates
   2. Triggers date check and alert generation

## **Security**

* Simple password-based admin login
* Default password: "123"
* Minimal authentication mechanism

## **Input Handling Process**

### **Student Query Input Flow**

1. **Input Capture**
   1. Uses Java Scanner to collect user inputs
   2. Supports both numeric and string inputs
   3. Provides hierarchical menu-driven interface

### **Query Processing Steps**

1. User selects problem category
2. Provide specific problem details
3. Enter room number or relevant information
4. System creates corresponding query object
5. Query stored in appropriate data structure

## **Potential Improvements**

1. Implement more robust authentication
2. Add persistent storage (database integration)
3. Enhance alert and notification system
4. Implement more detailed reporting
5. Add user feedback mechanism

## **System Requirements**

* Java Runtime Environment
* C++ Compiler with JNI support
* Native library compilation for the specific platform

## **Limitations**

* In-memory data storage
* Basic authentication
* No data persistence between application runs

## **Workflow**

1. Launch application
2. Choose user type (Student/Admin)
3. Follow on-screen prompts
4. Submit or manage queries

Admins can set and track resolution status

**GITHUB Repository Link**  
  
https://github.com/ashokCh-dev/studentHelpDesk

## **Conclusion**

The Student Help Desk application provides a straightforward interface for students to report issues and for administrators to track and resolve them efficiently.

**UML Diagram**

