Project: BuildingManager

By: Emmanuel Teferra, Garrett Wood, Adebayo (Hamid) Afolabi, Chenxi (Charles) Mu

Project: BuildingManager

Emmanuel Teferra - Team Lead
Garrett Wood - Contributor
Adebayo “Hamid” Afolabi - Contributor
Chenxi “Charles” Mu - Contributor

CIS453 Software Specification
Professor Edmund Yu
October 14th, 2022

Table of Contents

1. Introduction

Product

Despite being frequently packed with staff, students, residents, and employees, commercial and industrial residential buildings like hotels, dormitories, and corporate offices rarely have a structured plan for dealing with common building issues, maintenance requests, ITS problems, and more. With colleges accepting more students, hotels getting larger and larger, and offices hiring more employees, a new approach to industrial building oversight must be developed to cut through miscommunication and enhance efficiency. For that, we have BuildingManager.

BuildingManager is an interactive application that can be used by employers, residents, and employees alike to request and track maintenance tickets, identify technology or building issues, and much, much more. BuildingManager does this through a complicated web of “scores”; employing dozens of preset tiers that are then ranked and compiled based on data received from the user. After processing the most recent user data, BuildingManager creates its “tasklist”; a collated set of maintenance tickets, filtered by urgency, importance, ease of fixing, and more.

Despite being geared for utility and efficiency, the team behind BuildingManager is aware of the constant stress facilities workers are under to deliver quickly and perfectly. For that reason, BuildingManager will implement numerous provisions in the name of preventing overwork, exhaustion, or injury. These features include but are not limited to:

1. Response messages after tickets are filed informing residents on how to prevent the solved mistakes from occurring
2. Personal employee profiles detailing injuries/weaknesses
3. Mandated break times between filling out tickets when the same employee is working nonstop

Product Scope

The purpose of this brief is to articulate the methods, procedures, and requirements governing the BuildingManager app, a scheduling software used to track and regulate repairs in industrial buildings. This document will contain information on required user inputs, program constraints, technical details, and how BuildingManager operates. While the SRS is publicly available to all, the target audience of this presentation are users, software developers, programmers, and technical advisors.

2. Glossary

Resident - term used to designate any one outside user submitting a report, data, or repair request to the BuildingManager program. User must be apart of designated school or organization specified by client before interacting with software

Client - Outside user who is employing BuildingManager; they are responsible for reframing program parameters and merging BuildingManager with their maintenance systems

Schedule - the output of the BuildingManger program, the term denotes a processed tasklist created by the program that is then sent to designated locations (i.e. maintenance, ITS, DPS)

Break Case - this is a term used by BuildingManager to oversee all urgent building emergencies, injuries, and crisis; viewable by staff, client, and software developers alone

3. User Requirements

a. Functional user requirements

R1: Users shall select their college which for now is Syracuse

R2:User shall log in with their school info which includes their net Id and password for easy and secure verification

R3:Users will choose their exact location within the college, whether it’s in one of the dorm buildings stating the specific floor and room number or a specific street and apartment/house address

R4: Users will select whether it’s a emergency or pick from the drop down menu what type of problem it is

R5: Users that select emergency will then choose whether or not there were any injuries

b. Non functional user requirements

R1: Based on the type of problem, the system should be able to separate it into ITS, maintenance or DPS

R2: System should also be able to rate the urgency of most inquiries

R3: There should be a process of giving the user a chance to give a review on their experience of the program

4. System Requirements

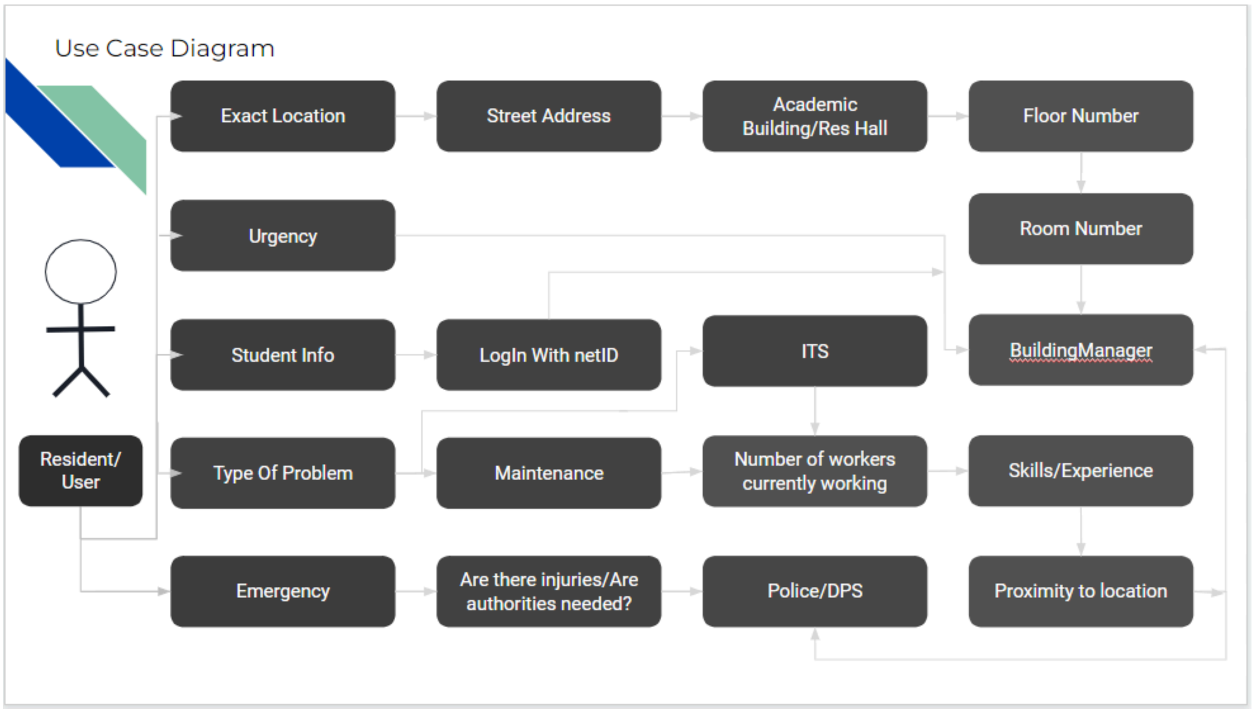
a. Functional System Requirements

1. Database that keeps students' address and profile, ITs and maintenance staff information, and common issues and emergencies.
2. Users shall be able to connect with their netID.
3. A dropdown menu of streets and resident halls for users to select their address.
4. Students can enter their floor and room number, or exact location.
5. ITs and maintenance staff can select their operating time, office location, and their capabilities/skills.
6. System will acquire GPS to have staff and students' location.
7. A menu for users to select the type to issue.
8. If it is an emergency, the system will give several contact information: DPS/police.
9. For non-emergency issues, users can select common problems from a dropdown box provided by the system or enter their own problem in a textbox.

b. Non-Functional System Requirements

1. After selecting a problem, users will check their contact information and location.
2. Students can choose a period of time to fix their problem.
3. System will rank the problems based on urgency.
4. ITs and maintenance staff can choose their priority such as distance and skills.
5. Most frequent problems will have records for users.

5. System Diagrams



Urgency of Issue	
Actor	Student
Description	A numerical scale with word descriptors allowing those who report issues to gauge the severity of an issue
Data	A numerical value (that is double checked by maintenance/ITS) rating the urgency of the issue at hand

Stimulus	Compares given urgency with predetermined value assigned to that category of emergency (usually assigned by employee/client)
Response	Confirmation statement is returned to user asking them to confirm their chosen ranking

Student Information	
Actor	Student
Description	After students log in with their netID, their name and contact information are provided from the university for students to verify them. If the student currently lives on campus, their exact address will be loaded automatically, otherwise the student needs to enter the address in the textbox.
Data	A set of data with the user's name, contact information and also exact address.
Stimulus	Users will be presented with the option to login and connect with their netID. After successful login, they will be prompted to verify their information.
Response	The app will redirect students to the problem selecting page.

Type of Problem	
Actor	Students
Description	A drop down menu will be presented to the user with a collection of the most common maintenance issues (wifi down, printer not working, microwave not working, plumbing issue, graffiti, etc.). Helps reduce unknown input cases from users.
Data	A drop-down menu will collect a choice from the user amidst a predetermined list of common residential/college issues.

Stimulus	Once the resident’s choice is collected from the drop down, it is assigned to a pre-selected numerical value that BuildingManager will use to score the severity of that issue.
Response	The program will inform the user that their problem has been recorded and sent to maintenance for review and solving.

Location	
Actor	Students
Description	A drop-down box showing all the dorms and school housing and an option labeled other which will prompt students that live elsewhere to put their specific location
Data	Residence halls, box to select specific floor and room number. An option to click other that gives a textbox to write your location along with an option for apartment number and floor plus zip code.
Stimulus	If the location is already within the database, then we can confirm it is an accurate location. If written inside the other textbox, our system would check withing Google's gps, that it is an actual location.
Response	Shows user that Location has been received and is confirmed as accurate.

Emergency	
Actor	Student
Description	If Emergency is clicked, a drop down should show for students to select the option of whether there were any injuries sustained and if authorities are needed.
Data	Just two options to choose which if it’s not the first , then they can click the other option which is No.

Stimulus	After receiving the response, if the first choice was selected then that immediately be put directly to the emergency choices which usually consist of mostly DPS.
Response	Shows user that we understand it’s an emergency and we'll bring a help service right away.

BuildingManager	
Actor	System Administrators and ITS employees
Description	After collecting all the required data from the user, BuildingManager then compares the given data against a predetermined score list and ranks the emergencies based on their score; i.e. the order in which the requests should be filled out
Data	Numerically, BuildingManager will output a list with maintenance tickets ordered from highest score to lowest score, with highest being “attend to now or first” (depending on the review and opinion of the receivers of the schedules/tickets) and lowest being “non-threatening / easy , quick fix”
Stimulus	If/When the user feels as if though they are finished or if they have no more information to put (aside from integral data like location, student info, etc.), they may select a “Finish Request/Report” button at the bottom that will let the program know it has finished its interaction with the user and can now begin processing the inputted ticket
Response	The program - after finishing its interaction with the user - would print a text statement with a summary of their inputted information (severity ranking, location, name, student info), a thank you message, and a list of SU emergency numbers to follow up with if needed.

6. Activity Diagrams

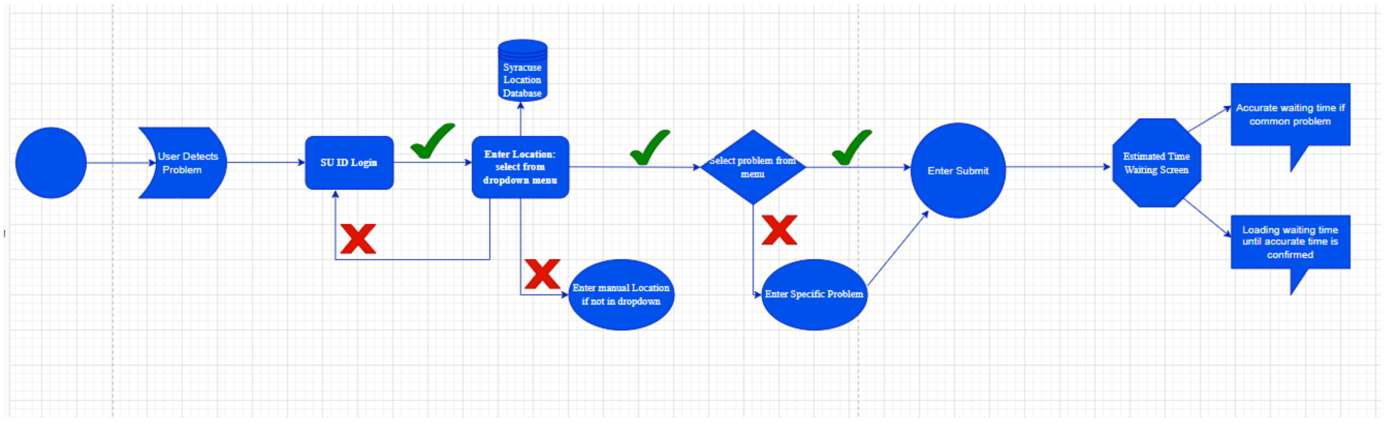

User Detects Problem
SU ID Login
Syracuse Location Database
Enter Location: select from dropdown menu
Enter manual Location if not in dropdown
Select problem from menu
Enter Specific Problem
Enter Submit
Estimated Time Waiting Screen
Accurate waiting time if common problem
Loading waiting time until accurate time is confirmed

Figure: User Activity Diagram

- This diagram shows the process a user from Syracuse University goes through when applying through the system to report a problem through our service. From the time, they recognize a problem has occurred to them getting online and going through the process of verifying themselves through the SU ID login all the way till submitting the full form.

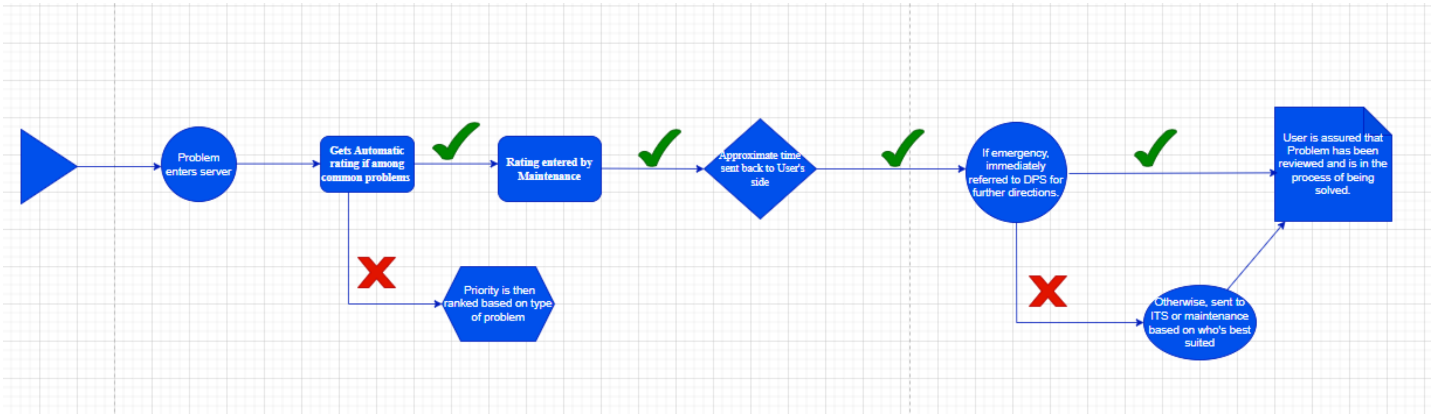
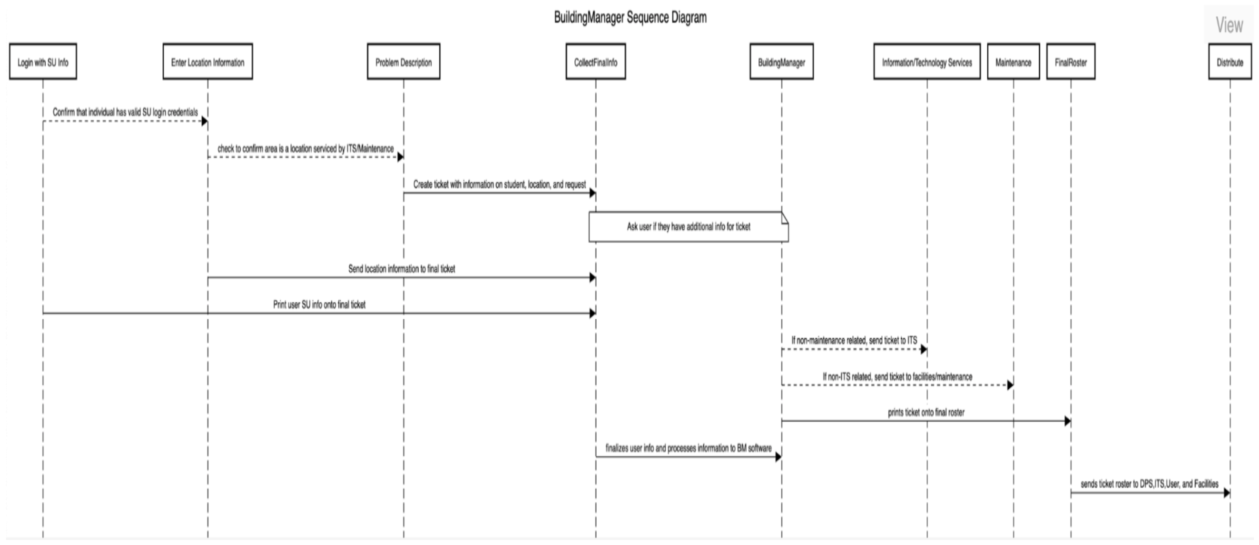

Problem enters server
Gets Automatic rating if among common problems
Rating entered by Maintenance
Approximate time sent back to User's side
If emergency, immediately referred to DPS for further directions.
User is assured that Problem has been reviewed and is in the process of being solved.
Priority is then ranked based on type of problem
Otherwise, sent to ITS or maintenance based on who's best suited

Figure: Worker Activity Diagram

- This diagram shows the behind the scenes of what one of the employees see on their side of the screen from analyzing the problems that aren't automated by the system and rating it based on the severity of the problem and delivering a proper message to the user screen based on how long the problem will take.

7. Sequence Diagrams


BuildingManager Sequence Diagram
View
Login with SU Info
Enter Location Information
Problem Description
CollectFinalInfo
BuildingManager
Information/Technology Services
Maintenance
FinalRoster
Distribute
Confirm that individual has valid SU login credentials
check to confirm area is a location serviced by ITS/Maintenance
Create ticket with information on student, location, and request
Ask user if they have additional info for ticket
Send location information to final ticket
Print user SU info onto final ticket
If non-maintenance related, send ticket to ITS
If non-ITS related, send ticket to facilities/maintenance
prints ticket onto final roster
finalizes user info and processes information to BM software
sends ticket roster to DPS,ITS,User, and Facilities

Sequences

Login with SU Info: In the first sequence, the client will login with their Syracuse University-provided credentials to verify that they are an individual permitted to submit a maintenance request. This not only allows the program to build a user profile for each ticket in case repair teams seek to follow up, but it also acts as a screen to avoid false or unauthorized maintenance requests from being accepted.

Enter location information: In the following sequence, the program seeks to collect exact user location data through a combination of text boxes and drop down menus. This allows maintenance teams to efficiently locate and deploy repair teams to affected areas without having to search through student info or decipher hurried, garbled phone calls which can lead to miscommunications and disorganization.

Problem Description: In the problem reporting sequence, clients are provided with a character-capped blank text box to be used to describe the issue at hand to receiving ITS/Maintenance staff.

CollectFinalInfo: After verifying that the individual is a valid user, collecting their location, and problem description, the software will then ask the user for any final pieces of information they wish to be reported to responding repair teams, as well as if they can be contacted for follow-ups. With the collection of all required information, the retrieved data is then collated onto a repair ticket containing student info, problem location, problem description, presence of injuries (if any) which is then sent to the BuildingManager algorithm.

BuildingManager: After receiving a completed ticket, the BuildingManager program will assign it a score based on real-time environment data provided by maintenance/ITS staff (# of specialists working, location of staffers, skillset of each employee, presence and count of injuries (if any), threat to life) throughout the day. After assigning a score to the ticket, BuildingManager keeps a running tier list of each ticket - dictated by its assigned score - that is updated after a problem is fixed, removed, or an additional ticket is submitted.

Information/Technology Services: After receiving the tiered list from BuildingManager, the system will perform an "ITS check" on the Schedule*(*see* Glossary) and inspect the current problem's (the ticket at top of the stack with the most points) repair requirements. If the problem requires an ITS fix, it is marked with an ITS flag and an ITS worker is then selected (based on skill set and proximity to the issue) and written onto the ticket.

Maintenance: After receiving the tiered list from BuildingManager, the system will perform a "Maintenance check" on the Schedule*(*see* Glossary) and inspect the current problem's (the ticket at the top of the stack with the most points) repair requirements. If the problem requires a maintenance fix, it is marked with a M flag and a maintenance worker is then selected (based on skill set and proximity to the issue) and written onto the ticket.

FinalRoster: After performing its final checks with maintenance and ITS, the software then creates a live document and visual for the staff containing ranked ticket info, estimated problem solution time, and an order in which tickets are to be fulfilled.

Distribute: The program then communicates the live results of the Schedule to a set of pre-specified contacts input by the staff before run time. In our prototype, the "Distribute" list contains contact information for ITS, users, maintenance, and Public Safety (which handles all emergency cases and receives a record of all tickets submitted to BuildingManager).

8. State Diagrams

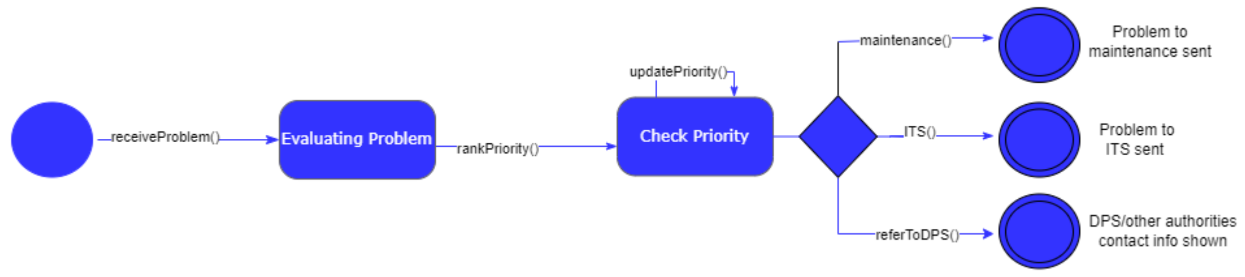


Figure: Submitted Problems State Diagram

Submitted Problems State Diagram: Each State represents a different level in the process that a problem goes through when received by BuildingManager. Each action corresponds to what should be taking place for that current action. The problem will be evaluated for what priority it should be first, and then that problem's priority will be checked and ranked. If a new problem comes in that should be higher, the list will be updated. After that, it will see where the problem should be sent to and it will send the problem to the correct professionals.

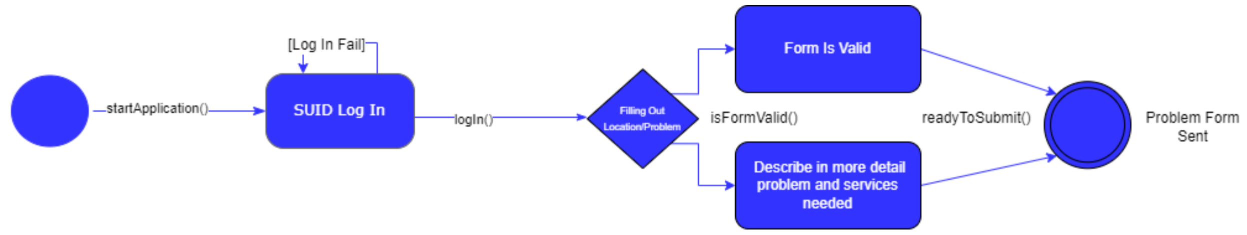


Figure: User/Application State Diagram

User/Application State Diagram: In this state diagram, the process of a user logging in and filling out the problem form is shown. The majority of the time will be spent in the filling out form portion where the user will be asked questions about locations and type of problem that it is. The process will see if the form is valid or needs any additional information. If the form is valid, the problem will be sent. If it needs more information, the user will be prompted and then send the problem.

9. Class Diagrams

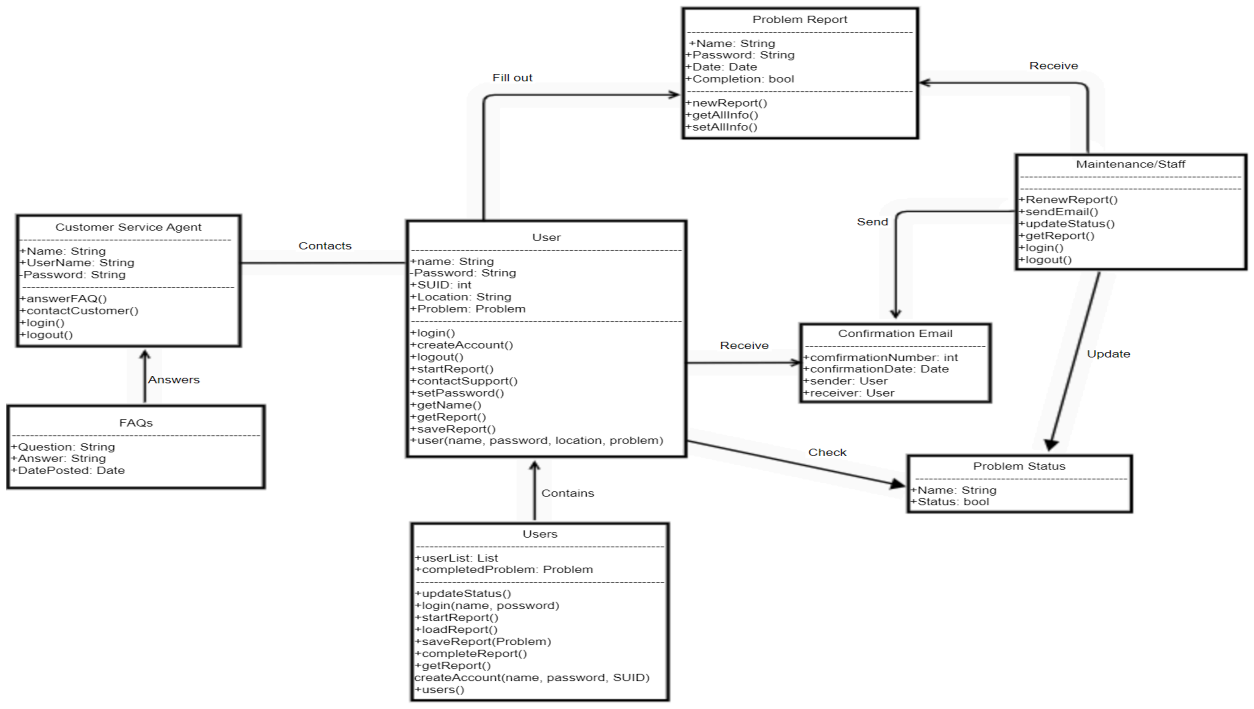


Figure: Class Diagram

Maintenance/Staff: This class will be the role of maintenance and staff and will gain access to the problems reported, sending confirmation email, update status, and renew reports.

<u>Methods</u>

- **RenewReport()**: will be called when the maintenance grants the user a renewed report.
- **sendEmail()**: will be called when the user completes their report.
- **updateStatus()**: will be called when the user first submits their report, or when they have been granted a renewed report or when the problem is fixed or closed.
- **getReport()**: will be called whenever the maintenance needs to get the problem from the user.
- **login()**: will be called when the maintenance needs to login to their account.
- **logout()**: will be called when the maintenance clicks the logout button.

User: This class will be for the user who will be sending in their problem report, they will also be able to contact customer service, look at FAQs, and check their problem status.

Methods

- **createAccount()**: will be called when the user clicks the create account button.
- **startReport()**: will give the user a new report.
- **getReport()**: will be called whenever the user needs to get the problem from the maintenance.
- **contactSupport()**: will be called when the user and agent communicate for help.
- **getName()**: returns the name of the user when trying to get info for the system.
- **saveReport()**: will save the data in the report so the user can resume it later.
- **setPassword()**: sets the password.
- **login()**: will be called when the user needs to login to their account.
- **logout()**: will be called when the user clicks the logout button.

Users: This class will be a list of users with some extra features that will apply to multiple users at once, such as getting rid of completed applications, logging in users and updating status.

Methods

- **updateStatus()**: will update the status of the report of the user in the list with the new status.
- **startReport()**: will pull up a fresh report that the user will fill out when they want to report a problem.
- **loadReport()**: will pull up the user's report when they go to complete an already saved report.
- **saveReport(Problem)**: this will save the current information that was input to the user's report with the report parameter.
- **completeReport()**: when the user goes to fill out the report this will be called to allow them to do so.
- **getReport()**: the maintenance will call this to the completed application from the user in the list of users.
- **login(name, password)**: will be called when the user needs to login and will authenticate it for the user in the list.
- **createAccount(name, password, SUID)**: create an account with the information given in the parameter.

Customer Service Agent: This class will be the customer service agent who will be able to talk to the user to help them in their report process and answer any FAQs on the service application.

Methods

- **answerFAQ()**: will be called when the customer service agent wants to answer a FAQ and will get them to put in the answer for the question.
- **contactCustomer()**: will be called when the user and agent communicate for help.
- **login()**: will be called when the customer service agent needs to login to their account.
- **logout()**: will be called when the customer service agent clicks the logout button.

Problem Report: This class will be the user problem report and will get all the information for the user and set it in the problem class for the specific user.

Methods

- **newReport()**: will be called when the user first starts their report.
- **getAllInfo()**: will be called when setting all the information for the users' data, getting all the information for the setAllInfo() method, may also be used when the maintenance checks the data from the report.
- **setAllInfo()**: when the user submits their report, it will set the data they input to their user object.

Confirmation Email: This class will be the email sent to the user and will contain data that will be in the generic format sent to each user when they complete their report.

FAQs: This class will be the FAQs that the customer service agent answers and the user can look at.

Problem Status: This will be a class that updates when the user completes their report and then whether they received a renewed report.

10. System Architecture Diagram

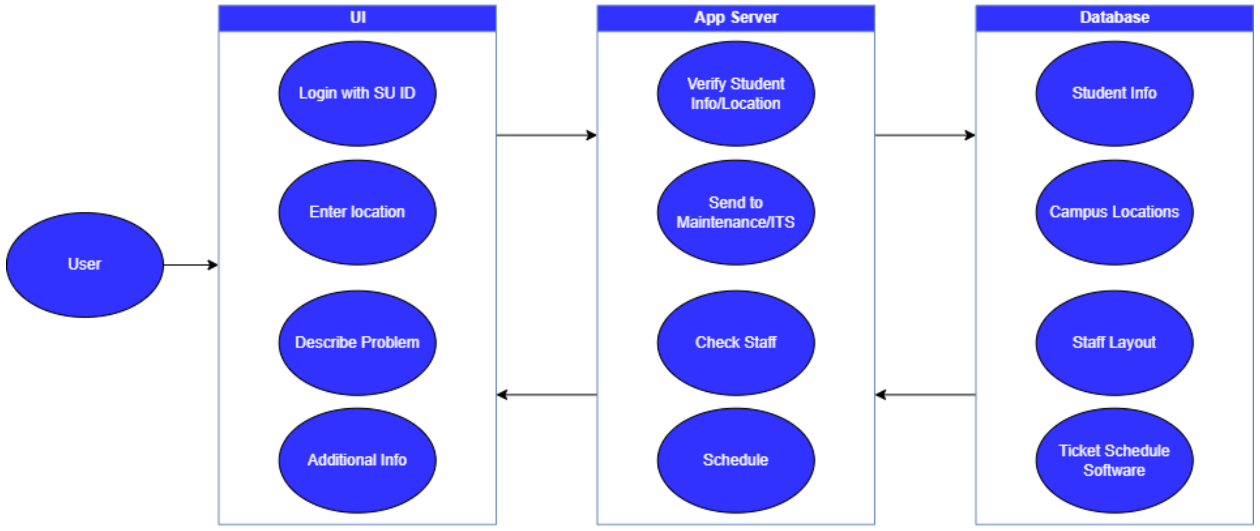


Figure: System Architecture Diagram

User Interface (UI)

- Enter SU login
    - User prompt to collect Syracuse login information from client
- Enter location data
    - Drop down box allowing client to select from a predetermined list of campus locations when submitting maintenance request
- Describe maintenance request
    - Blank-text box for user to describe current issues at hand
- Add additional info to ticket (if applicable)
    - Additional text box for user to include any further information about issue for responding employees

Application Server

- Verify Student Info/Location
    - Through SU, the software will use an individual's NETID and password to ensure that all requests come from valid, current students/employees and their current location.
- Send Ticket To ITS/Maintenance
    - Send the ticket from the student to either ITS or Maintenance depending on the type of flag the ticket has.
- Check Staff
    - Tells BuildingManager who is on staff and what their skillset and proximity

is to the problem

- Schedule
  - Creates the schedule based on urgency to determine which staff goes where first.

<u>Database</u>

- Student Info
  - Through SU, the software will use an individual's NETID and password to ensure that all requests come from valid, current students/employees
- Campus Locations
  - Through SU, the software will use the University's Database of SU owned buildings.
- Maintenance Staff
  - Through access to the maintenance/Facilities database, BuildingManager will track how building staff on hand, where on campus (or in the building) they are, and what skill sets they hold
- Information/Technology Services Staff
  - Through access to the ITS staff database, BuildingManager will track how much ITS staff is on hand, where on campus they are, and what skill sets they hold
- Ticket Scheduler Software
  - The Back-End software that will create the schedule for ITS/Maintenance to use.

11. Bibliography

❖ "Here to Help." *Information Technology Services Information Technology Services ITS Comments*, https://its.syr.edu/.

❖ "Fixit: SU's Housing and Food Services Maintenance Zone." *SU News*, 18 Nov. 2016, https://news.syr.edu/blog/2010/09/09/fixit-sus-housing-and-food-services-maintenance-zone/.