Software Engineering

PROJECT FINAL REPORT

Submitted by 21f1005173 and 21f1005606

Introduction

The aim of the project is to create a working prototype of an online support ticketing system for the students of IITM , where the students are able to ask their queries and get responses in a timely manner. This report outlines our attempt to create the same.

Detailed report on work done from Milestone 1 through Milestone 5

Milestone 1 : Identify User Requirements

There are three types of users here , The primary users are the SupportStaff/Lecturers for each subject, Students, secondary users would be the ones with admin privileges and tertiary users would be the developers. The user stories are written based on the SMART guidelines, which are S - Specific, M - Measurable, A - Achievable, R - Relevant, T - Time Bound. Below are the user stories we came up with.

1. As a student, I want to be able to create support ticket so that my queries are addressed
2. As a student, I want to edit or delete the created support ticket so that my query is correctly conveyed or solution has been found without support intervention
3. As a student, I want to upvote existing ticket so that duplication of ticket is resolved and staff can prioritize issue faced by major students
4. As a student, I want to see status of created ticket so that I can learn the solution addressed by the support staff
5. As a student, I want to be notified regarding the status and resolution of the created ticket so that I can avoid logging into the system to see if query has been addressed
6. As a support staff, I want to be able to see list of support tickets sorted by date so that I can respond the ticket which is created earliest
7. As a support staff, I want to be able to update the status to open/resolve so that student knows the progress regarding the queries raised
8. As a support staff, I want to sort the support tickets based on +1 votes so that highest priority of the ticket can be addressed
9. As a support staff, I want to be able to add comment/suggestion to the created ticket so that student is notified regarding the inputs further required
10. As a support staff, I want the ability to notify admin regarding the queries raised so that common queries can be added to FAQ
11. As an admin, I want to be able to see total count of support tickets raised so that I can monitor number of tickets raised per month

12. As an admin, I want to be able to see list of tickets suggested for FAQ updation so that I can add the tickets to FAQ
13. As an admin, I want to filter the ticket status to open/closed so that list of filtered tickets is shown

Out of the above we were able to implement everything except a notification system and sorted view for support staff and admin.

Milestone 2 : User Interfaces

A storyboard script was made and a wireframe was made and updated on the github project page, highlighting a short scenario of how a student could use this app to get response from a support staff. The wireframe was made using figma and is attached below

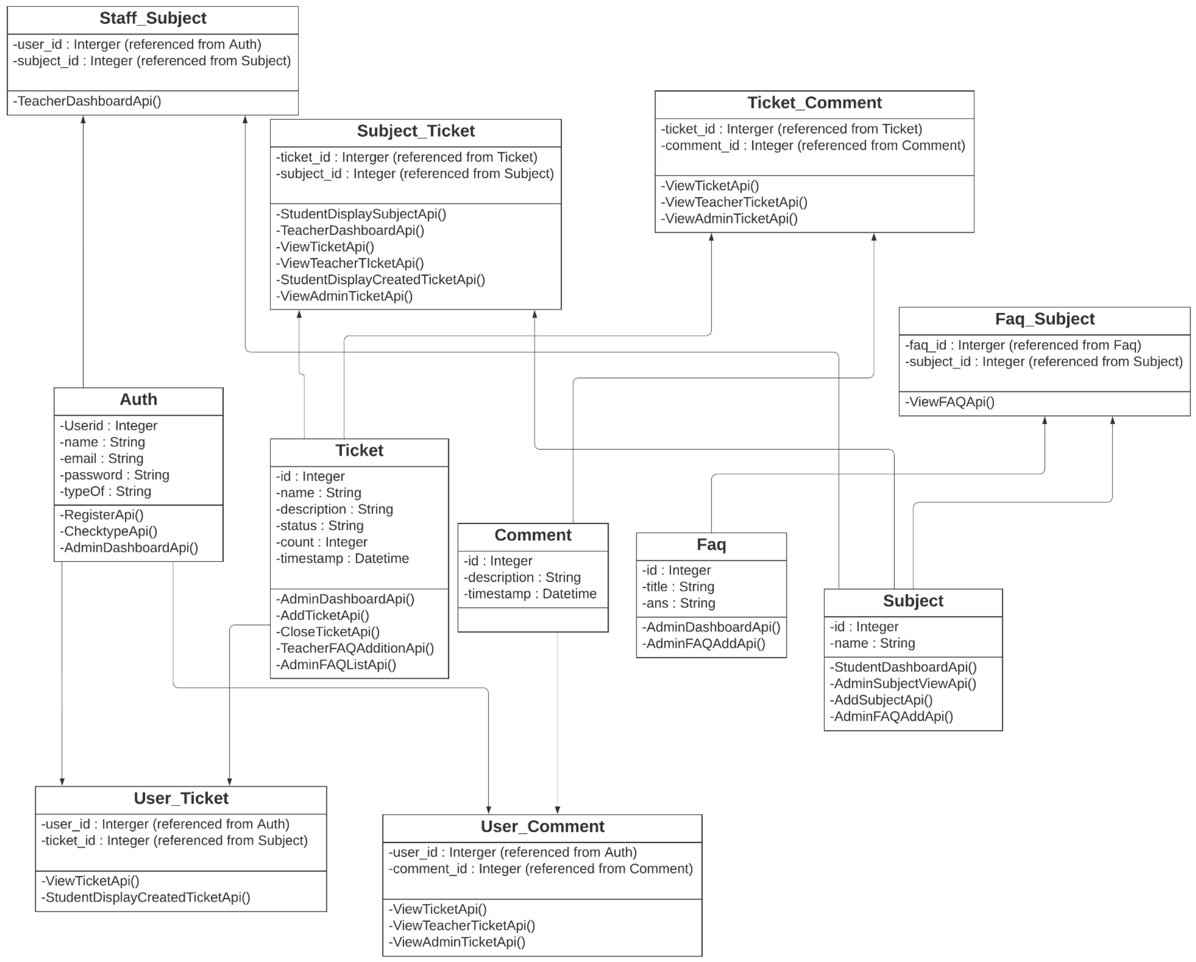
https://drive.google.com/file/d/1H5DTEmffdX6cq88y1NidYvBdgSuyLJxT/view?usp=sharing

Milestone 3- Scheduling and Design

Component Identification : Based on the software guidelines and principles ,critical components identified are:: 1. Student component 2. Staff component 3. Ticket management component .Through the student component, a student is able to register/login to the system and can raise queries by creating tickets or can view the ticket created by fellow students. The created ticket is then resolved by the concerned faculty and updated. All the tickets are maintained using the ticket management system.Solved queries are then added to FAQ which reduces duplication of tickets.

Project Scheduling : We chose to use Jira for Project Scheduling. The gantt charts were prepared with time bound tasks which are discussed with fellow team members which are shown in milestone 3 submission

UML Diagram is as below



Milestone 4 - API Endpoints

Api endpoints were made using swagger.io editor and the yaml file has been linked below
https://drive.google.com/file/d/1FNuis6ekWuDMEkWkHPZ7-qt_b-2FWS28/view?usp=sharing

Milestone 5 - Test Cases

[api.add_resource(RegisterAPI, "/api/register"),
username : test, email : test@email.com, password : test, TypeOf : Lecturer,
Expected Output : Successfully Registered Lecturer,
Actual Output - Successfully Registered Lecturer,
Result- Success]

[api.add_resource(CheckTypeAPI, "/api/checktype"),
current_user = {11 , Mark, Student },
Expected Output : {'username': Mark , 'typeOf': Student},
Actual Output - {'username': Mark , 'typeOf': Student},

Result- Success]

[api.add_resource(StudentDashboardAPI,"/api/student/dashboard"),
current_user = {11 , Mark, Student },
Expected Output : {username: 'mark', message: 'Subjects Present', subject_list: Array(3)},
Actual Output : {username: 'mark', message: 'Subjects Present', subject_list: Array(3)}, Result-Success]

[api.add_resource(StudentDisplaySubjectAPI,"/api/student/displaySubject/"),
id=1, current_user = {11 , Mark, Student },
Expected Output : {username: 'mark', subject_list: {…}, tickets_list: Array(1), isTicket: true, message: 'Subject Present'},
Actual Output : {username: 'mark', subject_list: {…}, tickets_list: Array(1), isTicket: true, message: 'Subject Present'},
Result-Success]

[api.add_resource(AddTicketAPI,"/api/student/createTicket/"),
name = Testing, description = Testing description field,
Expected Output : Ticket created successfully,
Actual Output : Ticket created successfully,
Result-Success]

[api.add_resource(StudentDisplayCreatedTicketAPI,"/api/student/viewCreatedTicket"),
current_user = {10 , Lark, Student }
Expected Output : {username: 'lark', tickets_list: Array(1), isTicket: true},
Actual Output : {username: 'lark', tickets_list: Array(1), isTicket: true},
Result-Success]

[api.add_resource(ViewFAQAPI,"/api/student/viewFAQ/<int:id>"),
id=1, current_user = {10 , Lark, Student },
Expected Output : {username: 'lark', message: 'FAQ Present', faq_list: Array(1)},
Actual Output : {username: 'lark', message: 'FAQ Present', faq_list: Array(1)},
Result-Success]

[api.add_resource(CloseTicketAPI,"/api/closeTicket/<int:ticket_id>"),
ticket_id=6, current_user = {10 , Lark, Student },
Expected Output : {Event: 'Event Updated successfully'},
Actual Output : {Event: 'Event Updated successfully'},
Result-Success]

[api.add_resource(ViewTicketAPI,"/api/student/viewTicket//"),
subject_id=1,ticket_id=3, current_user = {11 , Mark, Student },
Expected Output : {username: 'mark', subject_list: {…}, message: 'Subject Present', ticket_list: {…}, enableComment: true, alreadyLiked:false,comments:null},

Actual Output : {username: 'mark', subject_list: {…}, message: 'Subject Present', ticket_list: {…}, enableComment: true, alreadyLiked:false,comments:null},
Result-Success]

[api.add_resource(TeacherDashboardAPI,"/api/teacher/dashboard"),
current_user = {15 , Manu, Lecturer },
Expected Output : {username: 'manu', subject_list: Array(1), tickets_list: Array(2), isTicket: true, message: 'Subject Present'},
Actual Output : {username: 'manu', subject_list: Array(1), tickets_list: Array(2), isTicket: true, message: 'Subject Present'},
Result-Success]

[api.add_resource(ViewTeacherTicketAPI,"/api/teacher/viewTicket//"),
subject_id=1,ticket_id=3, current_user = {15 , Manu, Lecturer },
Expected Output : {username: 'manu', subject_list: {…}, message: 'Subject Present', ticket_list: {…}, enableComment: true,comments:null},
Actual Output : {username: 'manu', subject_list: {…}, message: 'Subject Present', ticket_list: {…}, enableComment: true,comments:null},
Result-Success]

[api.add_resource(TeacherFAQAdditionAPI,"/api/teacher/faqAddition"),
ticket_id=6, current_user = {11 , Babu, Lecturer },
Expected Output : {Event: 'FAQ Event added successfully'}
Actual Output : {},
Result-Success]

[api.add_resource(AdminDashboardAPI,"/api/admin/dashboard"),
current_user = {1, admin, admin},
Expected Output : {username: 'admin', message: 'Subjects Present', subject_list_len: 3},
Actual Output : {username: 'admin', message: 'Subjects Present', subject_list_len: 3},
Result-Success]

[api.add_resource(AdminSubjectViewAPI,"/api/admin/subject"),
current_user = {1, admin, admin},
Expected Output : {username: 'admin', message: 'Subjects Present', subject_list: Array(3)},
Actual Output : {username: 'admin', message: 'Subjects Present', subject_list: Array(3)},
Result-Success]

[api.add_resource(AddSubjectAPI,"/api/addsubject"),
subject_name = DBMS, current_user = {1, admin, admin},
Expected Output : Subject added successfully,
Actual Output : Subject added successfully,
Result-Success]

[api.add_resource(AdminFAQListAPI,"/api/admin/listFAQ"),

current_user = {1, admin, admin},
Expected Output : {username: 'admin', message: 'FAQ Present', faq_list_dict: Array(5)},
Actual Output : {username: 'admin', message: 'FAQ Present', faq_list_dict: Array(5)},
Result-Success]

[api.add_resource(ViewAdminTicketAPI,"/api/admin/viewTicket/<int:ticket_id>"),
ticket_id=1, current_user = {1, admin, admin},
Expected Output : {username: 'admin', message: 'Subject Present', ticket_list: Array(1), comments: Array(2)},
Actual Output : {username: 'admin', message: 'Subject Present', ticket_list: Array(1), comments: Array(2)},
Result-Success]

[api.add_resource(AdminFAQAddAPI,"/api/admin/createFAQ"),
faq_title=testing, faq_ans=test faq, subject_id =1, current_user = {1, admin, admin},
Expected Output : {Event: 'FAQ added successfully'}
Actual Output : {Event: 'FAQ added successfully'}
Result-Success]

Implementation details of project

Technologies and tools used :

1. Figma was used to create the wireframe
2. Jira was used for project scheduling
3. Vue.js is used for the frontend
4. Flask is used for the backend
5. Sqlite is used for the database via flask_sqlalchemy
6. flask_restful is used for apis
7. flask_security is used for authentication
8. axios is used for making api calls

Application Hosting :

Application is hosted on a local machine.

Instructions to run Application :

The application was programmed on a windows machine and the steps to run it on a windows machine is as follows.
Step 1 : Create a Virtual Environment
1. Create folder env
2. python -m venv ./env
3. Execute Scripts\activate.bat

Step 2 : Installing Requirements

1. pip install -r requirements.txt

Step 3 : Execute

1. python main.py
2. open 127.0.0.1:8080 and login/register an account to start using the application
3. To login as admin , use email : admin@gmail.com and password : admin

Demo Link

https://drive.google.com/file/d/1eW9zpAflAhO-2I6PbEnfLhlvyBHBszrt/view?usp=sharing