

SRI SIVASUBRAMANIYA NADAR COLLEGE OF ENGINEERING

(AN AUTONOMOUS INSTITUTION,
AFFILIATED TO ANNA UNIVERSITY)

Rajiv Gandhi Salai (OMR), Kalavakkam - 603 110.

LABORATORY RECORD

NAME : ...KARTHIK...D.....
Reg. No. : ...195001047.....
Dept. : ...CSE..... Sem. : ...VI..... Sec. : ...A.....

SRI SIVASUBRAMANIYA NADAR
COLLEGE OF ENGINEERING, CHENNAI
(AN AUTONOMOUS INSTITUTION, AFFILIATED TO ANNA UNIVERSITY)

BONAFIDE CERTIFICATE

Certified that this is the bonafide record of the practical work done in the

...VCS1617 - MINIPROJECT LAB..... Laboratory by

NameKARTHIK D.....

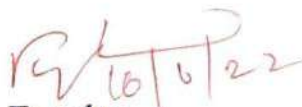
Register Number195001047.....

SemesterVI.....

BranchCSE.....

Sri Sivasubramaniya Nadar College of Engineering, Kalavakkam.

During the Academic year2022-23.....


Faculty


Head of the Department

Submitted for the.....Practical Examination held at SSNCE
on.....

Internal Examiner

External Examiner

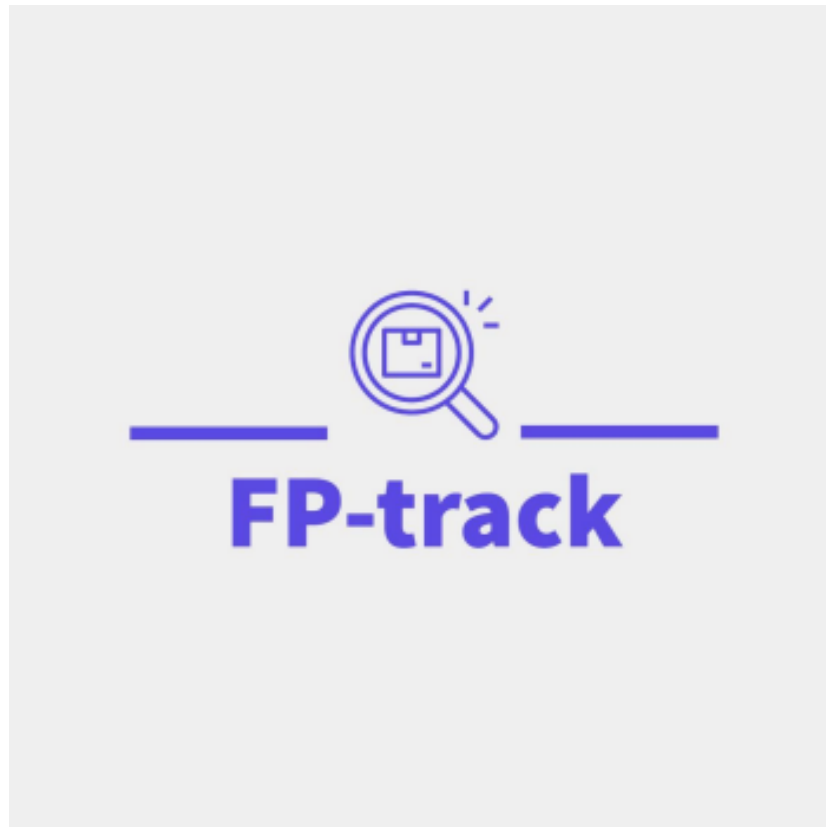
INDEX

Name : KARTHIK D Reg. No. 195001047

Sem : V.I. Sec : A

[illegible]

Funded Project and Resource Tracking System



Project Design and Specifications Document

Version 2.0

Revision History

Date	Version	Changelog	Specifications	Authors
March 13th, 2022	1.0	Initial	Problem Statement	Karthik D Karthik Raja Jagadish R Jeet Golecha
April 1st, 2022	1.1	Addendum	Requirements Specification	Karthik D Karthik Raja Jagadish R Jeet Golecha
April 10th, 2022	1.2	Addendum	Use Case Document	Karthik D Karthik Raja Jagadish R Jeet Golecha
April 12th, 2022	1.3	Addendum	Domain Model and Class Diagram	Karthik D Karthik Raja Jagadish R Jeet Golecha
April 18th, 2022	1.4	Addendum	Sequence Diagram	Karthik D Karthik Raja Jagadish R Jeet Golecha
April 28th, 2022	1.5	Addendum	State Machine Diagram	Karthik D Karthik Raja Jagadish R Jeet Golecha
May 10th, 2022	1.6	Addendum	Activity Diagram	Karthik D Karthik Raja Jagadish R Jeet Golecha
June 14th, 2022	2.0	Final	Final Document with Interfaces	Karthik D Karthik Raja Jagadish R Jeet Golecha

Table of Contents

Revision History	2
Table of Contents	3
Problem Statement	10
Proposed Solution	10
Updating Project Details on the Web-Application	10
Internal Projects	10
External Projects	11
Resource Management	11
Report Generation	11
Entity Description	11
User Stories	13
Open Access (Unauthenticated Guest User)	13
Admin	13
Applicant - Faculty/Student	13
Resource Manager	13
Software Requirements Specification	14
Introduction	14
Updating Project Details on the Web-Application	14
Internal Projects	14
External Projects	15
Resource Management	15
Report Generation	15
Entity Description	15
Purpose	16
Scope	16
Definitions, Acronyms, and Abbreviations	17
References	17

Overview	17
Overall Description	18
Product Perspective	18
Product Functions	19
User characteristics	20
Constraints	20
Assumptions and Dependencies	20
Specific Requirements	20
Functional	20
Updating Project Details on the Web-Application	21
Internal Projects Management	21
External Projects Management	21
Resource Management	21
Report Generation	22
Login Capabilities	22
Mobile Devices	22
Alerts	22
Non-Functional	22
Usability	22
Reliability	22
Availability	22
Accuracy	22
Access Reliability	22
Resource Utilization	23
Supportability	23
Internet Protocols	23
Information Security Requirement	23
Maintenance	23
Standards	23
Performance	23

Design Constraints	24
Software Language Used	24
Development Tools	24
Class Libraries	24
Online User Documentation and Help System Requirements	24
Interfaces	24
User Interfaces	24
Hardware Interfaces	24
Software Interfaces	24
Communications Interfaces	24
Licensing Requirements	25
Legal, Copyright, and Other Notices	25
Applicable Standards	25
Supporting Information	25
Use Case Diagram	26
Aim	26
UML Notations	27
Overview of the Complete System	28
Use Case Description	29
Scope	29
Level	29
Primary Actors	29
Stakeholders and Interests	29
Student/faculty applicant	29
Resource manager	29
Department	29
Principal/College	29
Student/Faculty	29
Funding Agencies	29
Preconditions	29

Postconditions	30
Main Scenario 1: Project Petition and Approval	30
Main Scenario 2: Request for Resources	30
Alternate Scenario 1: Student/Faculty Updates Progress Review	31
Alternate Scenario 2: Student/Faculty Records Project Completion	31
Alternate Scenario 3: Resource Manager Reclaim Resources	31
Alternate Scenario 4: Resource Manager Reports Faulty Components	32
Alternate Scenario 5: Guest Generates/Views Reports	32
Alternate Scenario 6: Guest Searches Projects	32
Technology and Data Variation list	32
Frequency of Occurrence	33
Open Issues	33
Domain and Class Diagram	34
Aim	34
UML Notations	34
Domain Model	34
Class Diagram	34
Identification of Classes	35
Conceptual Class Category List	35
Identification of Noun Phrases	35
Elimination of Noun Phrases	36
Final List of Classes	36
Categorization of Classes	37
Identification of Associations	38
Association Category list	38
Associations Identified and Categorization	38
Definition of Associations and their notations	39
Generalization	39
Aggregation	39
Composition	40

Multiplicity Based Associations	40
Domain Model Diagram	41
Class Diagram	41
Observations	41
Sequence Diagrams	43
Aim	43
UML Notations	43
Identified Scenarios	43
Main Scenario 1: Student/Faculty Registers Project	43
Main Scenario 2: Request for Resources	44
Alternate Scenario 1: Student/Faculty Updates Project Progress	44
Alternate Scenario 2: Student/Faculty Records Project Completion	44
Alternate Scenario 3: Resource Manager Reclaim Resources	44
Alternate Scenario 4: Resource Manager Reports Faulty Components	45
Alternate Scenario 5: Guest Generates/Views Reports	45
Alternate Scenario 6: Guest Searches for Projects	45
UML Sequence Diagrams	46
Login Process - Reusable Template	46
Main Scenario 1: Student/Faculty Registers Project	46
Main Scenario 2: Request for Resources	47
Alternate Scenario 1: Student/Faculty Updates Project Progress	47
Alternate Scenario 2: Student/Faculty Records Project Completion	48
Alternate Scenario 3: Resource Manager Reclaim Resources	48
Alternate Scenario 4: Resource Manager Reports Faulty Components	49
Alternate Scenario 5: Guest Searches and Generates Project Stats	49
Observations	50
State Diagram	50
Aim	51
UML Notations	51
Identification of States	52

State Machine Diagram	54
Workflow 1: Record Project Application	54
Workflow 2: Resource Request	54
Workflow 3: Project Milestones and Completion Updation	54
Workflow 4: Resource Reclamation	55
Workflow 5: Resource Fault Reporting	55
Workflow 6: Project Search and Report Generation	55
Observations	56
Activity Diagram	57
Aim	57
General Description	57
UML Notations	57
Identification of Activity Effector States	59
Activity Diagram	60
Workflow 1: Record Project Application	60
Workflow 2: Resource Allocation Request	61
Workflow 3: Project Milestones and Completion Updation	62
Workflow 4: Resource Management	62
Workflow 5: Project Search and Report Generation	63
Observations	63
Implementation Interfaces	64
End-User View	64
User Dashboard	64
Submit Proposals	64
View Proposals	65
View Projects - Overview	65
View Projects - Detailed	66
Approve/Reject Proposals [Admin]	68
Allocate Resources for Project [Resource Manager]	69
User Management [Admin]	69

Database View	69
Users	70
Proposals	70
Projects	71
Resource Groups	71
Resources	71
Resource Assignments	72
Source Code	73
Backend	73
Frontend	99
Test Cases	110
Aim	110
Identification of Testing Scenarios	110
Results	117

Problem Statement

Every year, several SSNites face issues in submitting and receiving timely response on their funded project proposals. There is no centralized system to apply, approve and track the progress of applications.

Furthermore, there is a need to track resources available with the department and repurpose them for upcoming projects with similar requirements, as opposed to granting funds to acquire redundant components.

Proposed Solution

We propose a centralized online system to manage internally and externally funded projects in the college. The following aspects will be handled by the system:

Updating Project Details on the Web-Application

The web-application can be used to update and track internal funding requests. In addition, it will serve as a repository to record the status of externally funded projects affiliated to the college.

Internal Projects

Prior to approving funds and allocating resources, the internal funding committee can find resources already available with the department, allocate them to the project and grant funds only for components that aren't already available with the department.

Faculty and students applying for internal funding must update details and status of their applications, detailing aspects about the funding authority, domain of work, budget requested, etc.

Approved projects can then be used by the applicants to record regular progress, directly visible to the research cell. The final outcomes of the projects will be added to the project in the end and archived into the database.

Intermediate and final outcomes of the projects must also be updated by the applicants, and will be archived after completion.

Upon project completion, the newly acquired resources are inventorized on the system and submitted to the department.

External Projects

Applicants applying for external funding must update details and status of their applications, detailing aspects about the funding authority, domain of work, budget requested, etc.

Intermediate and final outcomes of the projects must also be updated by the applicants, and will be archived after completion.

Resource Management

The system also maintains a repository of project resources already available with the system. This includes the availability status of the resources.

The department can choose to allocate these resources to newly approved applicants, whilst approving funds for other requirements alone. This ensures efficient usage and economic repurposing of the college's resources.

Faculty/Students, upon project completion, will deposit the newly acquired components with the department. These items will be inventoried on the system for reuse in future.

Report Generation

Summary statistics of applied and approved funding projects can be curated through the system based on multiple filtering criteria.

This can prove useful during research showcases and academic year progress presentations. Furthermore, it can be pivotal in auditing research and academic activities.

Entity Description

Project types can be one of the following types:

- Externally funded projects
 - Faculty projects
- Internally funded projects
 - Internally Funded Faculty Projects
 - Internally Funded Student Projects

Proposal/Applications can be set to one of the following statuses:

- Applied
- Shortlisted for Presentation
- Approved
- Rejected

Approved projects are maintained as a separate repository with the following possible statuses:

- Ongoing
- Completed

Regular updates and final outcomes are recorded along with the completed projects before archival. Outcomes will be represented as:

- Publications
- Link to publications project OR DBLP/GScholar page
- Patents
- Forwarded to External Funding

Resource repository to inventory components already available, categorized as:

- Resources currently under use
- Available resources
- Faulty resources

Report generation with filters, such as:

- Calendar period
- Project members
- Project domain
- Budget
- Status

Status updation interface will contain:

- Status upgrades
- Email status triggers

User Stories

Open Access (Unauthenticated Guest User)

- View statistics
- Generate reports and graphs
- View approved projects list

Admin

- User and Database administration
- Status updation of applicants

Applicant - Faculty/Student

- Updates information about funding applications on the portal
- Updates progress of projects on the web-application
- Records final outcome of projects

Resource Manager

- Updates status and inventories resources when they are surrendered
- Monitors resource inventory managed on the website

Software Requirements Specification

Introduction

Research and development projects are commonly characterized by funding applications to support the project at its multiple stages. Such funding agencies exist within the university and as part of governmental organizations and other venture and incubation organizations.

Every year, several SSNites face issues in submitting and receiving timely responses on their funded project proposals. There is no centralized system to apply, approve and track the progress of applications.

Furthermore, in the case of funding within the university department, there is a need to track resources available with the department and repurpose them for upcoming projects with similar requirements, as opposed to granting funds to acquire redundant components.

We propose a centralized online system to manage internally and externally funded projects in the college. The following aspects will be handled by the system:

Updating Project Details on the Web-Application

The web application can be used to update and track internal funding requests. In addition, it will serve as a repository to record the status of externally funded projects affiliated with the college.

Internal Projects

Before approving funds and allocating resources, the internal funding committee can find resources already available with the department, allocate them to the project, and grant funds only for components that aren't already available with the department.

Faculty and students applying for internal funding must update details and status of their applications, detailing aspects about the funding authority, domain of work, the budget requested, etc. Approved projects can then be used by the applicants to record regular progress, directly visible to the research cell.

The final outcomes of the projects will be added to the project in the end and archived into the database. Intermediate and final outcomes of the projects must also be updated by the applicants and will be archived after completion. Upon project completion, the newly acquired resources are inventoried on the system and submitted to the department.

External Projects

Applicants applying for external funding must update details and status of their applications, detailing aspects about the funding authority, domain of work, the budget requested, etc. Intermediate and final outcomes of the projects must also be updated by the applicants and will be archived after completion.

Resource Management

The system also maintains a repository of project resources already available with the system. This includes the availability status of the resources. The department can choose to allocate these resources to newly approved applicants, whilst approving funds for other requirements alone. This ensures efficient usage and economic repurposing of the college's resources.

Faculty/Students, upon project completion, will deposit the newly acquired components with the department. These items will be inventoried on the system for reuse in the future.

Report Generation

Summary statistics of applied and approved funding projects can be curated through the system based on multiple filtering criteria.

Entity Description

1. Project types can be one of:
 - Externally funded projects
 - Faculty projects
 - Internally funded projects
 - Internally Funded Faculty Projects
 - Internally Funded Student Projects
2. Proposal/Applications can be set to one of the following statuses:
 - Applied
 - Shortlisted for Presentation
 - Approved
 - Rejected
3. Approved projects are maintained as a separate repository with the following possible statuses:
 - Ongoing
 - Completed
4. Regular updates and outcomes are recorded along with the completed projects before archival.
5. Outcomes will be represented as:
 - Publications
 - Link to publications project OR DBLP/GScholar page
 - Patents
 - Forwarded to External Funding

6. Resource repository to inventory components already available, categorized as:
 - Resources currently underuse
 - Available resources
7. Report generation with filters, such as:
 - Calendar period
 - Project
 - Members
 - Project
 - Domain
 - Budget
 - Status
8. Status updation interface

To record and display the project updates through a centralized portal, that can be monitored by project mentors and the research cell as well.

Purpose

The purpose of the **Software Requirements Specification** (SRS) document is to describe the external behavior of the Funded Project Tracking System proposed here. This specification defines and describes the operations, interfaces, performance, and quality assurance requirements of the Funded Project Tracker. The document also describes the non-functional requirements such as the user interfaces. It also describes the design constraints that are to be considered when the system is to be designed, and other factors necessary to provide a complete and comprehensive description of the requirements for the software. The Software Requirements Specification (SRS) captures the complete software requirements for the system, or a portion of the system.

Scope

The Online Funded Project Tracker that is to be developed exposes a centralized online system for faculty and students to manage internally and externally funded projects in the college. The following aspects and features will be handled by the system:

- An *unauthenticated guest* user can:
 - view funding statistics
 - generate reports and graphs of applicant-wise stats
 - view approved projects list
- An *admin* user can:
 - perform user and database administration
 - update status of applicants

- An *applicant faculty/student* user can:
 - update information about funding applications on the web-application
 - update progress of projects on the web-application Records final outcome of projects
- A *resource manager* user can:
 - update status and inventories of resources when they are surrendered
 - monitor resource inventory stored on the web-application

The features that are described in this document are used in the future phases of the software development cycle. The features described here meet the needs of all the users. The success criteria for the system is based on the level up to which the features described in this document are implemented in the system.

Definitions, Acronyms, and Abbreviations

- SSN - Sri Sivasubramaniya Nadar College of Engineering
- CSE - Computer Science and Engineering
- IFSP - Internally Funded Student Project
- IFFP - Internal Funded Faculty Project
- FPT - Funded Project Tracker

References

This SRS document uses the following documents as references:

Problem Statement Specification: To specify the existing system, its pitfalls and the proposed solution along with entity descriptions and user stories are presented in the problem statement document stored here:
<https://www.github.com/karthik-d/Funded-Project-Tracker>

Current State of Funded Project Listings: SSN's official website features a basic tabular display of all funded project details. This listing is basic in its functionality and does not allow resource reallocation. This webpage can be found here:
<https://www.ssn.edu.in/college-of-engineering/it-newsletter-faculty-internal-funded-projects/>

Overview

This SRS will provide a detailed description of the FPT System. This document will provide the outline of the requirements, an overview of the characteristics and constraints of the system.

The next section of the SRS will provide the general factors that affect the product and its requirements. It provides the background for those requirements. The items such as product perspective, product function, user characteristics, constraints, assumptions and dependencies, and requirements subsets are described in this section.

The final section of SRS contains all the software requirements mentioned in section 2 in detail sufficient enough to enable designers to design the system to satisfy the requirements and testers to test if the system satisfies those requirements.

Overall Description

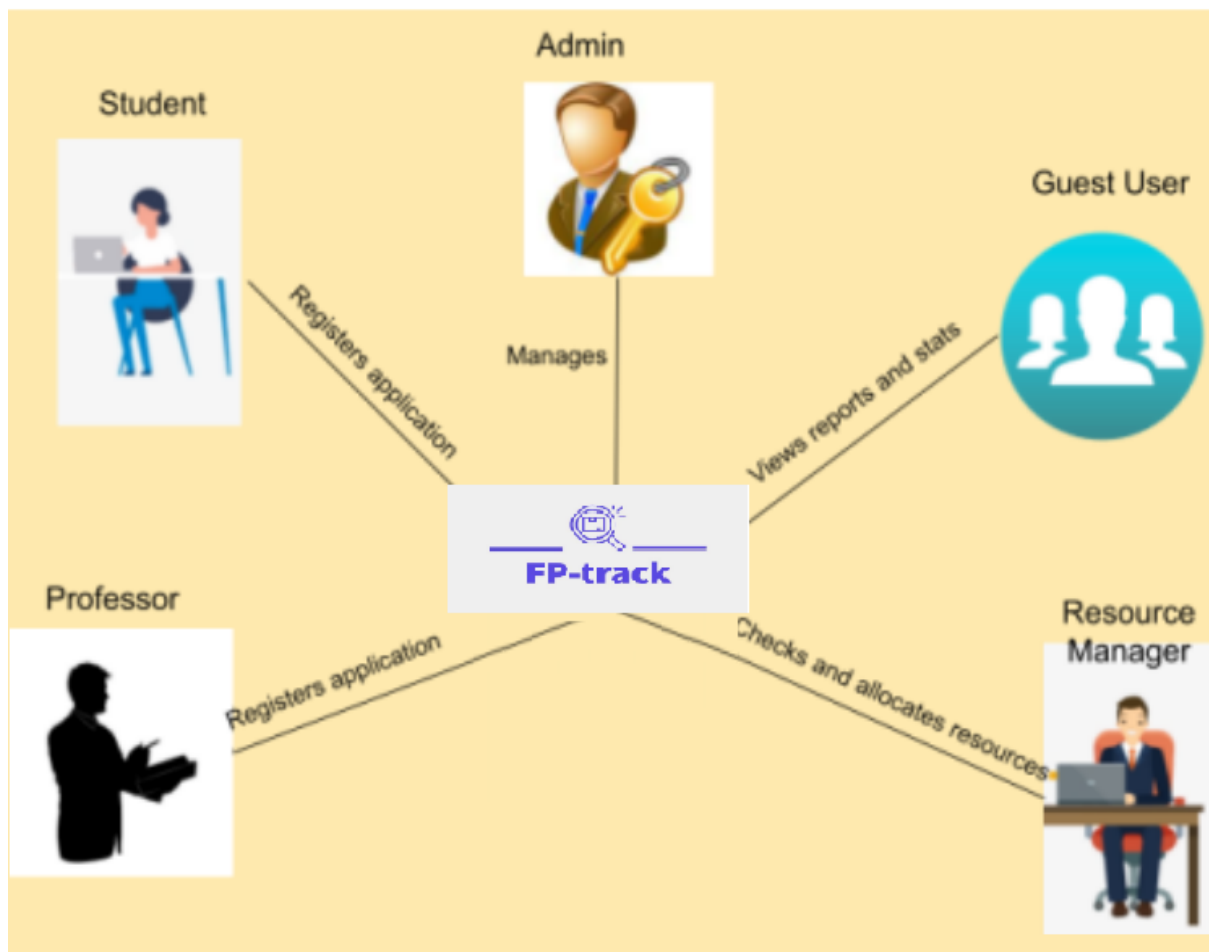
Product Perspective

The project tracking system is a package that will be used by students as well as faculty members to upload their project ideas and get the appropriate resources required for the project. The system developed will be used by the SSN College of Engineering to manage the Internally funded projects.

The product to be developed has interactions with the following users:

- Student applying for funding
- Faculty applying for funding
- Resource manager
- Admin
- Guest user

The complete overview of the system is as shown in the overview diagram below:



Product Functions

The Funded Project Tracking system provides online real-time information about the projects and resources for components available with the department and serves as a repository for project status and outcomes. The Product functions are more or less the same as described in the product perspective. The functions of the system include the system providing different types of services based on the type of users.

Provisions for the student/faculty to record their funding applications and project updates on the web-application and ultimately, link their outcomes on the centralized portal. Upon approval by the scrutiny team, the resource manager at the department can reallocate available resources and approve the budget for the remaining components alone.

Website visitors will be provided an interface to view all ongoing internally and externally funded project activities along with updates and outcomes that can be filtered by calendar period, project supervisor, members, domain and budget, and also generate customized reports.

The system uses the University's Google One membership to authenticate users into the system to secure access to the facility to the users.

User characteristics

The users of the system are students, faculty, and resource manager (who is also a delegated faculty) of the university along with administrators who maintain the system. All the users are assumed to have a basic working knowledge of computers and surfing through web pages.

The administrators of the system are expected to have more knowledge of the internals of the system and can rectify the small problems that may arise due to disk crashes, power failures, and other catastrophes to maintain the system.

The proper user interface, users manual, online help, and the guide to install and maintain the system must be sufficient to educate the users on how to use the system without any problems.

Constraints

- The information of all the faculty/students must be stored in a database that is accessible by the FPT system. This can be linked to the Google Organization account.
- The FPT system is connected to the university intranet and is running throughout the day.
- The users can access the FPT system from any computer that is connected to SSN's intra-network.
- The users must have their correct usernames and passwords to sign in to their organizational accounts hosted on Google One.

Assumptions and Dependencies

- The users have sufficient knowledge of computers.
- The students/faculty must update details of their funded project applications and status on the web-application
- The department's computer should have Internet connection and Internet server capabilities at least sufficient to connect to the local intra-network
- The project outcomes must be specified by the project stakeholders or the web administrator

Specific Requirements

This section describes in detail all the functional and non-functional requirements of the system.

Functional

Updating Project Details on the Web-Application

The web-application can be used to update and track internal funding requests. In addition, it will serve as a repository to record the status of externally funded projects affiliated to the college.

Internal Projects Management

Prior to approving funds and allocating resources, the internal funding committee can find resources already available with the department, allocate them to the project and grant funds only for components that aren't already available with the department.

Faculty and students applying for internal funding must update details and status of their applications, detailing aspects about the funding authority, domain of work, budget requested, etc.

Approved projects can then be used by the applicants to record regular progress, directly visible to the research cell. The final outcomes of the projects will be added to the project in the end and archived into the database.

Intermediate and final outcomes of the projects must also be updated by the applicants, and will be archived after completion.

Upon project completion, the newly acquired resources are inventorized on the system and submitted to the department.

External Projects Management

Applicants applying for external funding must update details and status of their applications, detailing aspects about the funding authority, domain of work, budget requested, etc.

Intermediate and final outcomes of the projects must also be updated by the applicants, and will be archived after completion.

Resource Management

The system also maintains a repository of project resources already available with the system. This includes the availability status of the resources.

The department can choose to allocate these resources to newly approved applicants, whilst approving funds for other requirements alone. This ensures efficient usage and economic repurposing of the college's resources.

Faculty/Students, upon project completion, will deposit the newly acquired components with the department. These items will be inventoried on the system for reuse in future.

Report Generation

Summary statistics of applied and approved funding projects can be curated through the system based on multiple filtering criteria.

This can prove useful during research showcases and academic year progress presentations. Furthermore, it can be pivotal in auditing research and academic activities.

Login Capabilities

The system shall provide the users (students and faculty) with login capabilities through Google OAuth

Mobile Devices

The Project Tracking System will also be supported on mobile devices such as cell phones.

Alerts

The system can alert the admin in case of any ambiguity or problem with the system

Non-Functional

Usability

- The system shall allow the users to access the system from the Internet using HTML or its derivative technologies. The system uses a web browser as an interface.
- Since all users are familiar with the general usage of browsers, no specific training is required.
- The system is user-friendly and self-explanatory.

Reliability

The system has to be very reliable due to the importance of data and the damages incorrect or incomplete data can do.

Availability

The system is available 100% for the user and is used 24 hrs a day and 365 days a year. The system shall be operational 24 hours a day and 7 days a week.

Accuracy

The accuracy of the system is limited by the accuracy of the speed at which the students and faculty use the system

Access Reliability

The system shall provide 100% access reliability.

Resource Utilization

The resource inventory is updated based on the active projects and the projects that have been completed.

Supportability

The system designers shall take into consideration the following supportability and technical limitations.

Internet Protocols

The system shall comply with the TCP/IP protocol standards and shall be designed accordingly.

Information Security Requirement

The system shall support the FPT information security requirements and use the same standard as the FPT information security requirements.

Maintenance

The maintenance of the system shall be done as per the maintenance contract.

Standards

The coding standards and naming conventions will be as per the Indian and US standards.

Performance

- **Response Time**

The information page should be able to be downloaded quickly and details must be updated in real time at regular intervals. The information is refreshed every two minutes. The access time for a mobile device should be less than a minute. The system shall respond to the member in not less than two seconds from the time of the request submission. The system shall be allowed to take more time when doing large processing jobs.

- **Throughput**

The number of transactions is directly dependent on the number of users, who may be the administrators, students, and also the faculty who use the portal

- **Capacity**

The system should at least handle 250 users at a time.

- **Resource Utilization**

The resources are modified according to the user requirements and also according to the requests of the users, if possible

Design Constraints

Software Language Used

The languages that shall be used for coding the FPT System are Node.js, MongoDB, HTML/CSS, JavaScript.

Development Tools

Will make use of the available Visual Studio Code and will be version controlled with Github.

Class Libraries

Will make use of the existing libraries available for Express.js, Mongoose, Tailwind CSS and Bootstrap.

Online User Documentation and Help System Requirements

Online help is provided for each of the features available with the Funded Project Tracking System. The nature of these systems is unique to application development as they combine aspects of programming (hyperlinks, etc) with aspects of technical writing (organization, presentation).

The User Manual describes the use of the system to librarians and Employees. It describes the use of the system on mobile systems. The user manual should be available as a hard copy and also as online help.

An installation document will be provided that includes the installation instructions and configuration guidelines, which is important to a full solution offering. Also, a readme file is typically included as a standard component. The Read Me includes a "What's New With This Release" section, and a discussion of compatibility issues with earlier releases. Most users also appreciate documentation defining any known bugs and workarounds in the readme file

Interfaces

User Interfaces

Will make use of the existing Web Browsers such as Google Chrome, Mozilla Firefox, Microsoft Edge or Safari. The user interface of the system shall be designed as shown in the user-interface prototypes.

Hardware Interfaces

The existing Local Area Network (LAN) will be used for collecting data from the users.

Software Interfaces

A firewall will be used with the server to prevent unauthorized access to the system.

Communications Interfaces

The Funded Project Tracking System will be connected to the Intranet services of the university.

Licensing Requirements

The usage is restricted to only Dept. of CSE at SSN College of Engineering, Chennai.

Legal, Copyright, and Other Notices

The FP Tracking system is a trademark of the CSE department at SSN College of Engineering and cannot be used without its consent.

Applicable Standards

The ISO/IEC 6592 guidelines for the documentation of computer-based application systems will be followed.

Supporting Information

The use-case storyboards or the user-interface prototypes are not available.

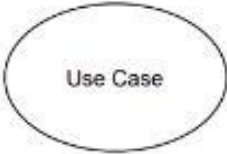

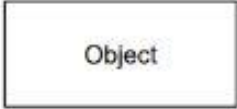

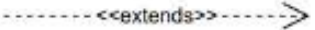


Use Case Diagram

Aim

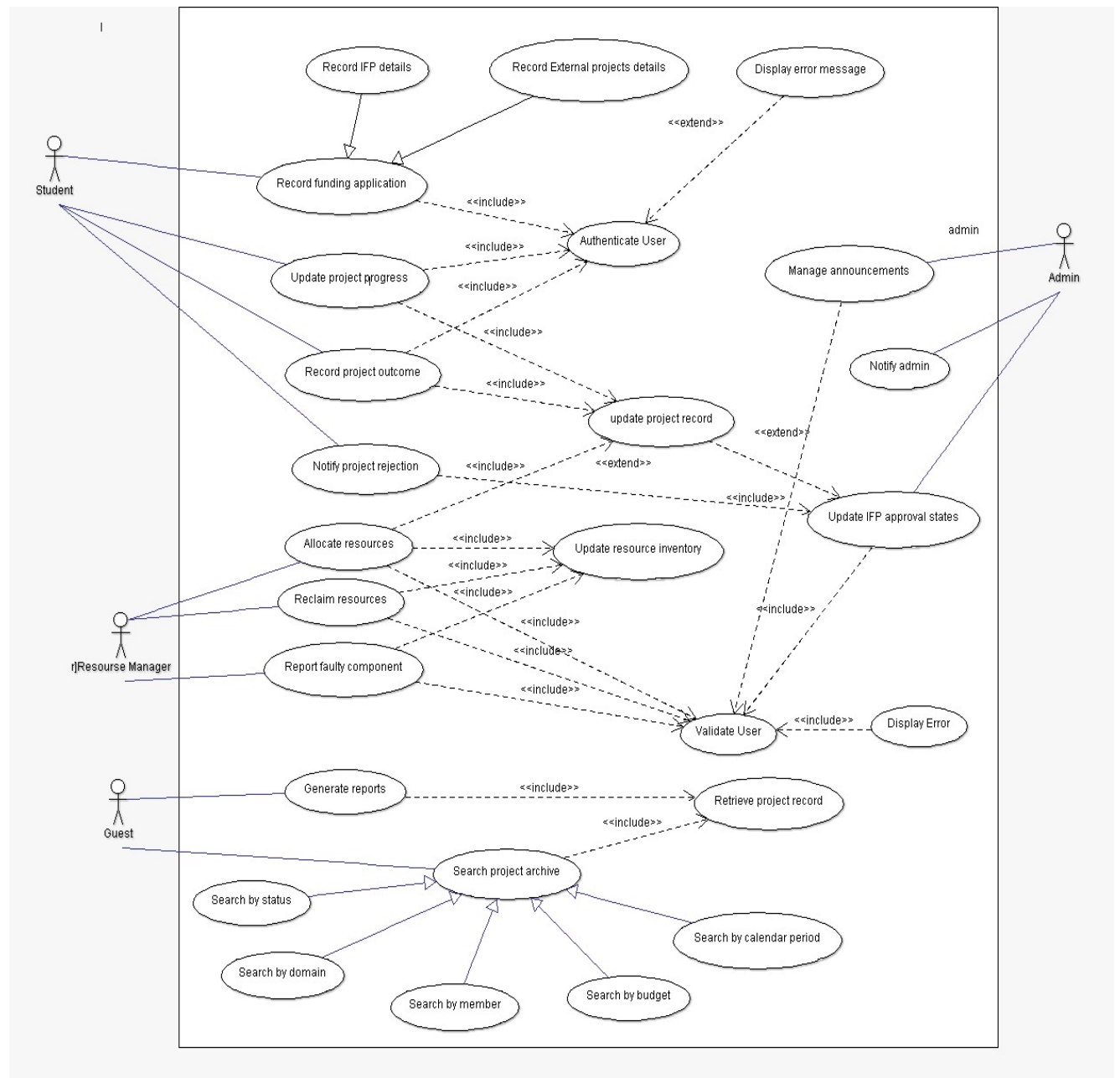
To create a UML use case model for the problem domain that includes:

- researching the business domain
- documenting UML notations
- identifying use cases
- identifying different types of actors
- identifying scenarios
- identifying sub-functions
- determining relationship between use cases
- determining relationship between use cases and actors
- creating use case diagrams adhering to UML standards for the identified scenarios
- expanding identified sub-functions by creating use case diagrams adhering to UML standards
- providing a fully dressed use case description for each scenario
- documenting observations on the use case model of the system

UML Notations

Figure	Notation	Explanation
	Use Case	Represents various functionality of the system
	Actor	Represents the various types of users of the system
	Object	Represents an object associated with the system
	Relationship	Represents "includes" relationship between use cases
	Relationship	Represents "extends" relationship between use cases
	Relationship	Represents generalization between actors and between use cases
	Relationship	Represents association between use cases

Overview of the Complete System



Use Case Description

Scope

This is a system used within the CSE department in SSN. It is a project management system for internal and external funding for projects.

Level

Overview of the Complete System

Primary Actors

- Student
- Faculty
- Resource Manager

Stakeholders and Interests

Student/faculty applicant

Needs a portal to upload their funding project proposals which are accepted by jury, get the desired resources to continue them and notified of their approval status

Resource manager

Wants a digital interface to interact with and maintain the inventory

Department

Wants to digitalize the manual procedural system, for efficient updation and maintenance of Funded project records.

Principal/College

Wants yearly updates of the Funded projects from each department

Student/Faculty

Expects to know existing projects to get novel ideas for their project

Funding Agencies

Wants to know the count of successful projects completed under their funding

Preconditions

The applicant and resource manager, admin are identified and authenticated. The inventory of previous project resources is maintained.

Postconditions

Project proposals are recorded in the system. The projects are reviewed by a jury. Resources are allocated. Projects are being started. Applicant details and their corresponding project details are saved, the overall yearly project report is generated.

Main Scenario 1: Project Petition and Approval

1. A student or faculty member logs into the system
2. System validates the log-in credentials
3. They create a proposal under the particular funding project title
4. They provide all the required details to the system such as
 - a. Team members
 - b. Problem statement
 - c. Mentor
 - d. Resource required and its corresponding budget
 - e. Documents shared with jury
5. They submit the above draft for further checking
6. The admin checks the validity of the submitted proposals
7. After successful completion of IFP presentations, and the admin gets the approved projects from the Management, the admin updates the application status for the submitted proposals.
8. The student/faculty gets notified of their project status.

Main Scenario 2: Request for Resources

1. Once the applicant gets notified of their project approval, they file their request for the resources for their project.
2. The resource manager logs into the system, and can view all the resource requests.
3. The Resource manager accesses the inventory, and allocates for the requested resources.

4. The admin and the applicant are notified about the allocations, and the corresponding data is updated in the project record by the system.
5. The applicant can get the approved resources after mapping his profile with the resource through barcode and biometric/id mapping and use them in their project.

Alternate Scenario 1: Student/Faculty Updates Progress Review

1. The applicants can regularly update their project progress through our system
2. The system validates the user by their log-in credentials
3. The system automatically updates the corresponding project record in the database
4. The System sends reminders if the project is at a stale point, and is not updated for a certain period of time.

Alternate Scenario 2: Student/Faculty Records Project Completion

1. The applicants log into the system and system validates them
2. They provide their project outcome or project completion status to the system for the given project after regular updates done earlier.
3. The system updates them to the corresponding project record automatically, and notifies the Resource manager, Admin.
4. This is used for reclaiming the resources and the official documentation of the project's completion status.

Alternate Scenario 3: Resource Manager Reclaim Resources

1. Once the resource manager is notified of the project completion or resource completion.
2. The resource manager gets back the allocated resources from the applicants and checks the conditions of the project.
3. Then the resource manager accesses the inventory and update the resources through our system
4. The system updates the project record automatically in the database
5. The admin updates the IFP status as completed
6. The project lead will then be notified for further formalities.

Alternate Scenario 4: Resource Manager Reports Faulty Components

1. Sometimes, a resource manager finds a faulty component in the inventory
2. Then he updates the inventory and the faultiness is reported to the admin
3. The admin gets notified about the faultiness of the corresponding resources mentioned by the resource manager
4. The admin can further proceed to place orders or bring in domain experts to handle the problem.

Alternate Scenario 5: Guest Generates/Views Reports

1. The guest can view or generate the reports of the projects in our management system
2. The reports consist all details entered by the applicants for their corresponding projects
3. The dates, subject, and other important information is detailed in the reports
4. This is used to generate yearly reports for the magazines , and other official documentation for the department.
5. These reports can be generated from the database , as and when required.

Alternate Scenario 6: Guest Searches Projects

1. The guest user can search their desired projects in our system
2. They can search based on various filters
 - a. Search by domain
 - b. Search by members
 - c. Search by budget
 - d. Search by status
 - e. Search by calendar period
3. The guest, or incoming project applicant uses this to extend their ideas and plan on the project ahead of time.
4. The admin, uses this to reduce project replications

Technology and Data Variation list

1. **OAuth** for using college credentials to login for registration and other project related activities.

2. **QR Code Labels and Scanners** for each item to assign item ID in the inventory that uniquely identifies a component.
3. **Unique IDs and Biometrics** to uniquely map resource seekers (project student/faculty) with the resources they are allocated.

Frequency of Occurrence

Varies based on project opening , and varies in the range of few hundred to less than a hundred requests per day.

Open Issues

- Digital copies of previous year funded projects may not be available to readily inventorize and create repositories on the system
- The system may require further functionality and verification from other in-charges for cross-department funded projects?

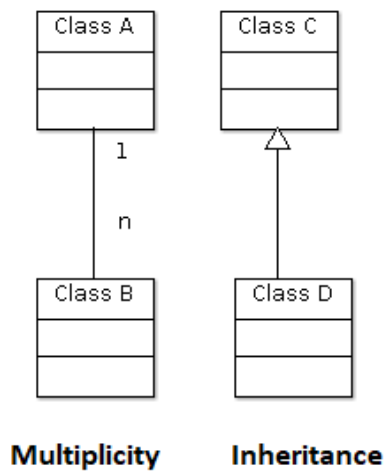
Domain and Class Diagram

Aim

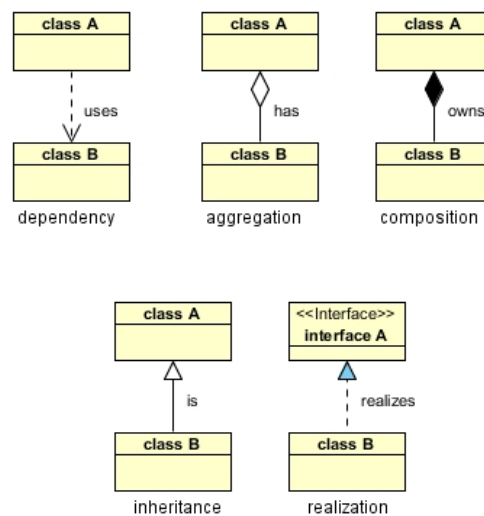
- To map the FPTrack System to a UML Domain Model
- To create a UML Class Diagram for the Agritech System

UML Notations

Domain Model



Class Diagram



Identification of Classes

Conceptual Class Category List

Conceptual Class Category	Description
Functional Unit	Non-tangible unit of functional purpose that may not be physically perceived but contributes to the development of an intellectual (and/or) perceptible system
Documents	Piece of printed matter that serves as a record
Records	An account of the data
Physical Tangible Objects	Objects that can be touched or felt
Roles of People	Set of behaviors among the people with respect to the problem statement
Place	A particular position, point, or area in space

Identification of Noun Phrases

- Login Device
- Resource Allocation Record
- Resource
- Software resources
- Hardware resources
- Project
- Internal Project
- Internal Faculty Project
- Internal Student Project
- External Project
- Funding Organization
- User
- Person Name
- Department

- Admin
- Student/Faculty
- Guest User
- Resource manager
- Storage Room
- Storage Room In-Charge
- QR Code Scanner
- QR Code Label
- ID Card

Elimination of Noun Phrases

Noun Phrase	Elimination Reasoning
Person name	Is an attribute
Department	Is an attribute
Storage Room In-Charge	Duplicate of Resource Manager, Role
ID Card	Irrelevant to the scenario
Login Device	Irrelevant to the scenario
Funding Organization	Out of System Scope, Is an attribute
QR Code Scanner	Out of System Scope, Scanned ID used as attribute
QR Code Label	Out of System Scope, Contained ID used as attribute

Final List of Classes

- Resource Allocation Record
- Resource
- Software Resources
- Hardware Resources
- Project
- Internal Project
- Internal Faculty Project
- Internal Student Project

- External Project
- User
- Admin
- Student/Faculty
- Guest User
- Resource manager
- Storage Room

Categorization of Classes

Class Name	Conceptual Class Category
Resource Allocation Record	Records, Documents
Resource	Functional Unit, Physical Tangible Object
Software Resource	Functional Unit
Hardware Resource	Physical Tangible Object
Project	Documents, Physical Tangible Object
Internal Project	Documents, Physical Tangible Object
Internal Student Project	Documents, Physical Tangible Object
External Student Project	Documents, Physical Tangible Object
External Project	Documents, Physical Tangible Object
User	Roles of People
Admin	Roles of People
Student	Roles of People
Faculty	Roles of People
Guest User	Roles of People
Resource Manager	Roles of People
Storage Room	Place

Identification of Associations

Association Category list

An **association** is a relationship between (instances of) classes that indicates some meaningful and interesting connection. These are used to show relationships that need to be remembered and preserved for some duration

Association Category	Description
B is a type of A	B inherits the properties of A and adds further functionality
A controls B	A controls and manages the operation of B and its contents
B describes A	B provides a detailed structured description of A
A is given to B	A is allocated to B for use and is expected to be released back
A belongs to B	B claims ownership over A

Associations Identified and Categorization

Source (A)	End (B)	Relationship	Category
Resource Manager	Faculty	Generalization	B is a type of A
User	Resource Manager	Generalization	B is a type of A
User	Admin	Generalization	B is a type of A
User	Guest User	Generalization	B is a type of A
User	Student	Generalization	B is a type of A
User	Faculty	Generalization	B is a type of A
Resource Manager	Storage Room	Aggregation	A controls B
Storage Room	Resource	Aggregation	A contains B
Resource	Software Resource	Generalization	B is a type of A
Resource	Hardware Resource	Generalization	B is a type of A
Software Resource	Software Resource Details	Composition	B describes A

Hardware Resource	Hardware Resource Details	Composition	B describes A
Resource Allocation Record	Resource	Aggregation	A describes B
Resource Allocation Record	Resource Manager	Aggregation	A describes B
Resource Allocation Record	Internal Project	Aggregation	A is given to B
Project Record	Internal Project	Generalization	B is a type of A
Project Record	External Project	Generalization	B is a type of A
Internal Project	Internal Student Project	Generalization	B is a type of A
Internal Project	Internal Faculty Project	Generalization	B is a type of A
Project Record	Student	Aggregation	A belongs to B
Project Record	Faculty	Aggregation	A belongs to B

Definition of Associations and their notations

Generalization

- Relationship that implements the concept of object orientation called inheritance. The generalization relationship occurs between two entities or objects, such that one entity is the parent, and the other one is the child.
- In UML modeling, a generalization relationship is a relationship that implements the concept of object orientation called inheritance
- The generalization relationship occurs between two entities or objects, such that one entity is the parent, and the other one is the child
- The child inherits the functionality of its parent and can access as well as update it
- Generalization relationship is utilized in class diagrams to specify that the child inherits actions, characteristics, and relationships from its parent

Aggregation

- Represents an associative relationship where the child can exist independently of the parent.
- An aggregation is a subset of association, which represents has a relationship
- It is more specific than association
- It defines a part-whole or part-of relationship
- In this kind of relationship, the child class can exist independently of its parent class

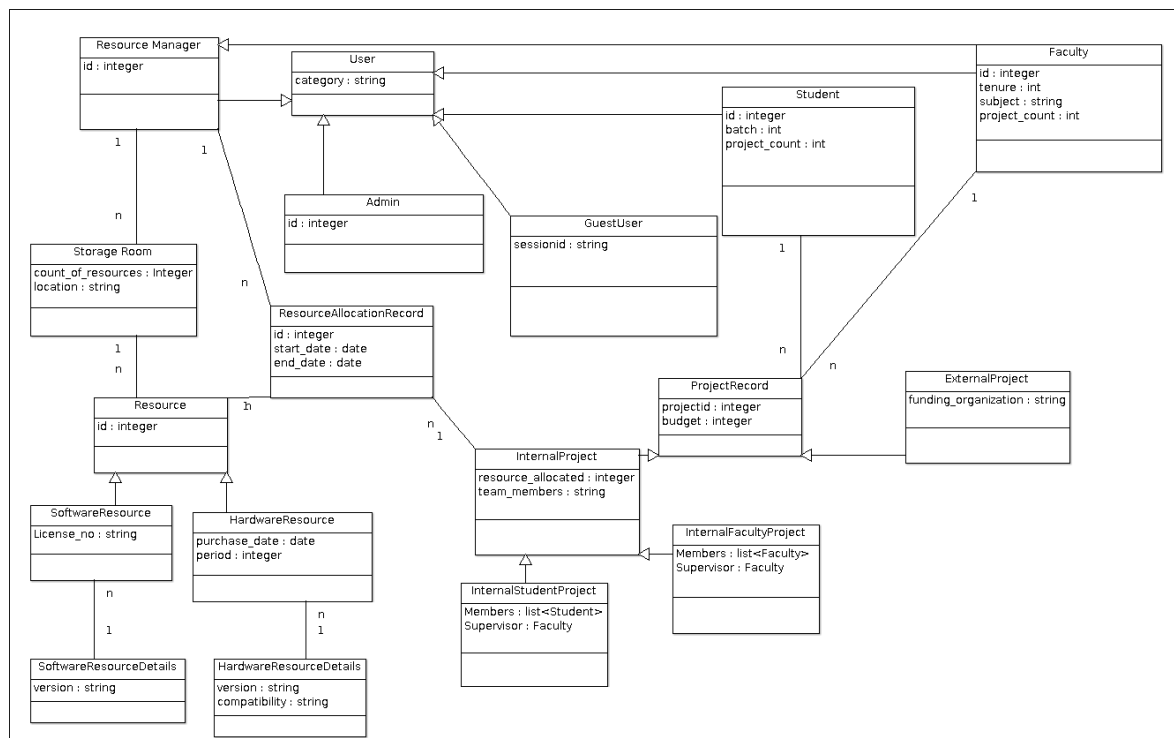
Composition

- Represents an associative relationship where one object is the owner of another object and they are dependent on one another.
- The composition is a subset of aggregation
- It portrays the dependency between the parent and its child, which means if one part is
- deleted, then the other part also gets discarded
- It represents a whole-part relationship

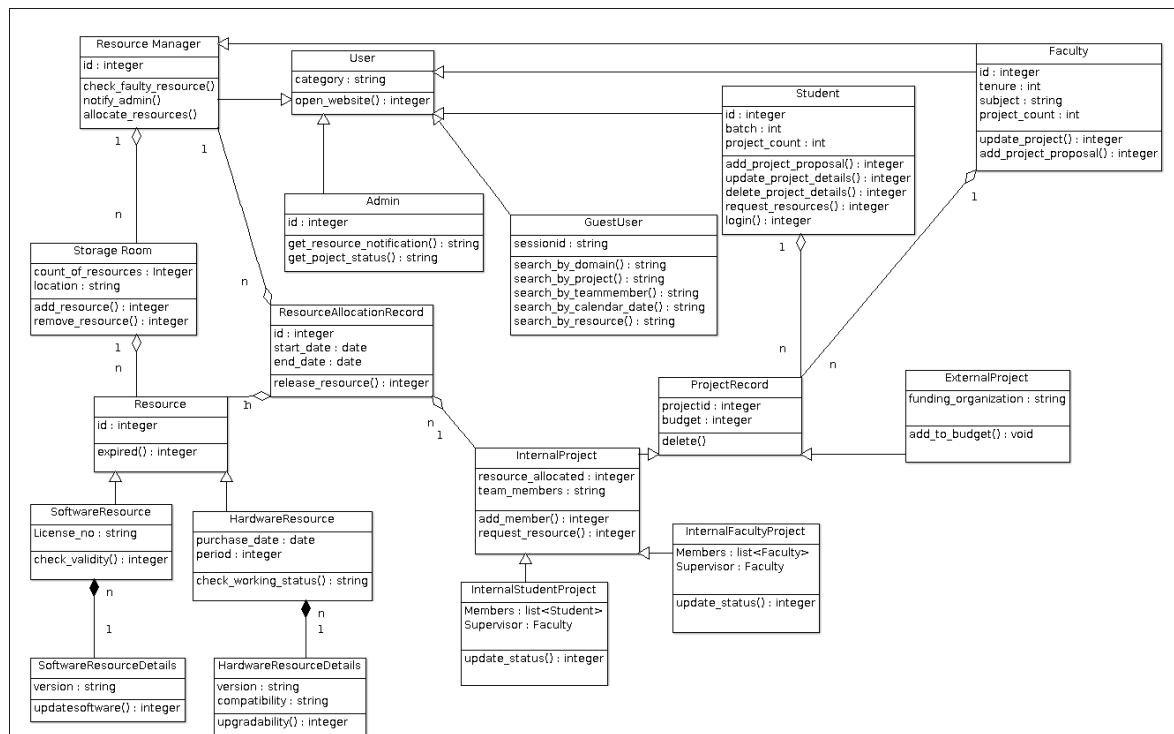
Multiplicity Based Associations

- Multiplicity based associations
- Multiplicity defines how many instances of a class A can be associated with one instance
- of a class B
- It defines a specific range of allowable instances of attributes
- In case if a range is not specified, one is considered as a default multiplicity

Domain Model Diagram



Class Diagram



Observations

While drafting the domain model and class diagram of this FP Tracking Software System, we observed the following:

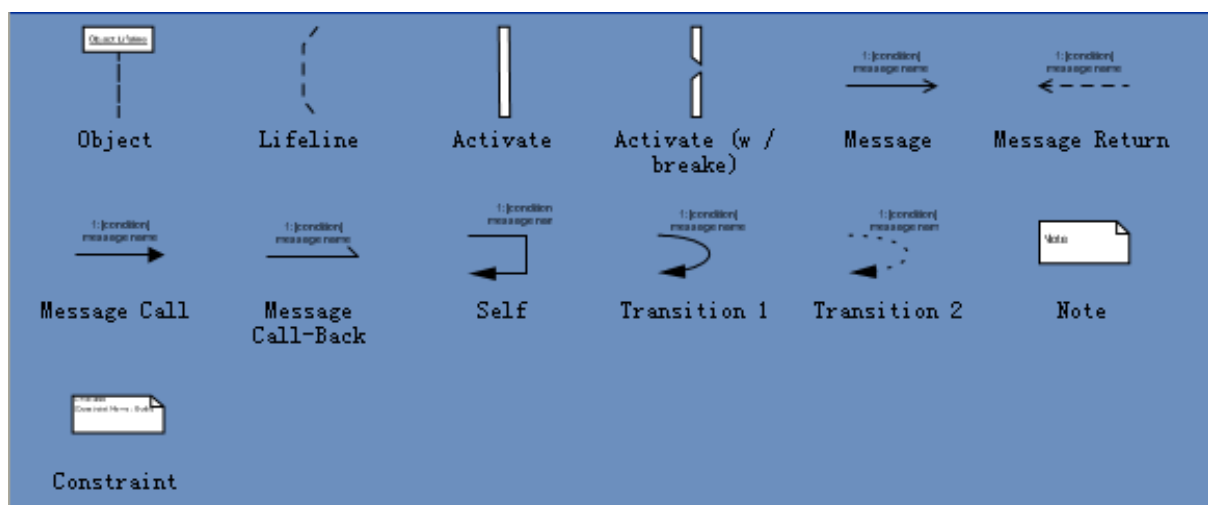
- Real world entities are seldom easily represented using abstract class descriptions
- Some real-world entities may require further complex organization of classes to describe them accurately
- Modeling the domain and the class diagrams brings us closer towards a concrete software implementation that is quickly extensible from the diagrams
- The domain model specification is a simplified representation to represent the object oriented design paradigm for any software system

Sequence Diagrams

Aim

- To define event sequences, which would have a desired outcome.
- The focus is more on the order in which messages occur than on the message per se.
- The majority of sequence diagrams designed here will communicate what messages are sent and the order in which they tend to occur

UML Notations



Identified Scenarios

Main Scenario 1: Student/Faculty Registers Project

- Student or faculty member logs into the system
- System validates the log-in credentials
- They register details of the applied projects on the website
- The admin checks the validity of the submitted proposals
- After successful completion of IFP presentations, and the admin gets the approved projects from the Management, the admin updates the application status for the submitted proposals.
- The student/faculty gets notified of their project status

Main Scenario 2: Request for Resources

- Once the applicant gets notified of their project approval, they file their request for the resources for their project.
- The resource manager logs into the system, and can view all the resource requests.
- The Resource manager accesses the inventory, and allocates for the requested resources.
- The admin and the applicant are notified about the allocations, and the corresponding data is updated in the project record by the system.
- The applicant can get the approved resources after mapping his profile with the resource through barcode and biometric/id mapping and use them in their project.

Alternate Scenario 1: Student/Faculty Updates Project Progress

- The applicants can regularly update their project progress through our system
- The system validates the user by their log-in credentials
- The system automatically updates the corresponding project record in the database
- The System sends reminders if the project is at a stale point, and is not updated for a certain period of time.

Alternate Scenario 2: Student/Faculty Records Project Completion

- The applicants log into the system and system validates them
- They provide their project outcome or project completion status to the system for the given project after regular updates done earlier.
- The system updates them to the corresponding project record automatically, and notifies the Resource manager, Admin.
- This is used for reclaiming the resources and the official documentation of the project's completion status.

Alternate Scenario 3: Resource Manager Reclaim Resources

- Once the resource manager is notified of the project completion or resource completion.

- The resource manager gets back the allocated resources from the applicants and checks the conditions of the project.
- Then the resource manager accesses the inventory and update the resources through our system
- The system updates the project record automatically in the database
- The admin updates the IFP status as completed
- The project lead will then be notified for further formalities.

Alternate Scenario 4: Resource Manager Reports Faulty Components

- Sometimes, a resource manager finds a faulty component in the inventory
- Then he updates the inventory and the faultiness is reported to the admin
- The admin gets notified about the faultiness of the corresponding resources mentioned by the resource manager
- The admin can further proceed to place orders or bring in domain experts to handle the problem.

Alternate Scenario 5: Guest Generates/Views Reports

- The guest can view or generate the reports of the projects in our management system
- The reports consist all details entered by the applicants for their corresponding projects
- The dates, subject, and other important information is detailed in the reports
- This is used to generate yearly reports for the magazines , and other official documentation for the department.
- These reports can be generated from the database , as and when required.

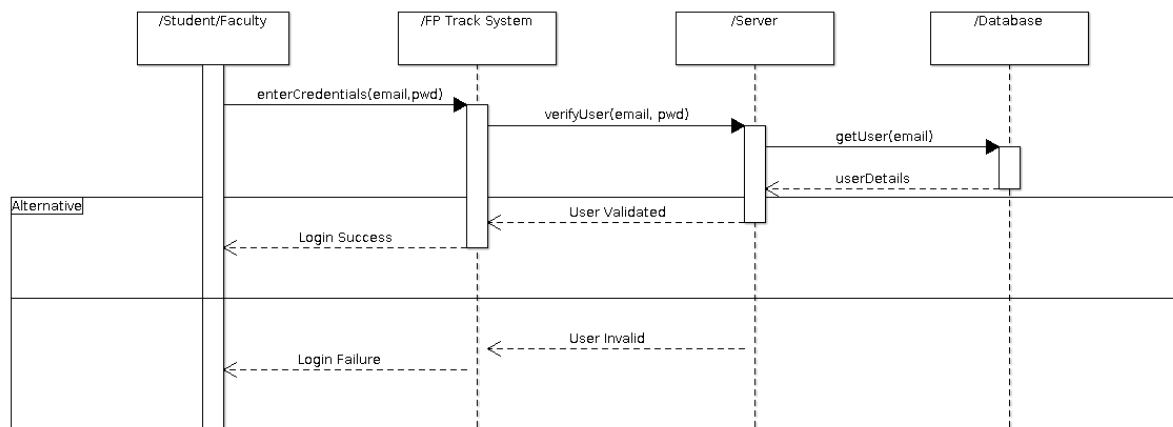
Alternate Scenario 6: Guest Searches for Projects

- The guest user can search their desired projects in our system
- They can search based on various filters
 - Search by domain
 - Search by members
 - Search by budget
 - Search by status

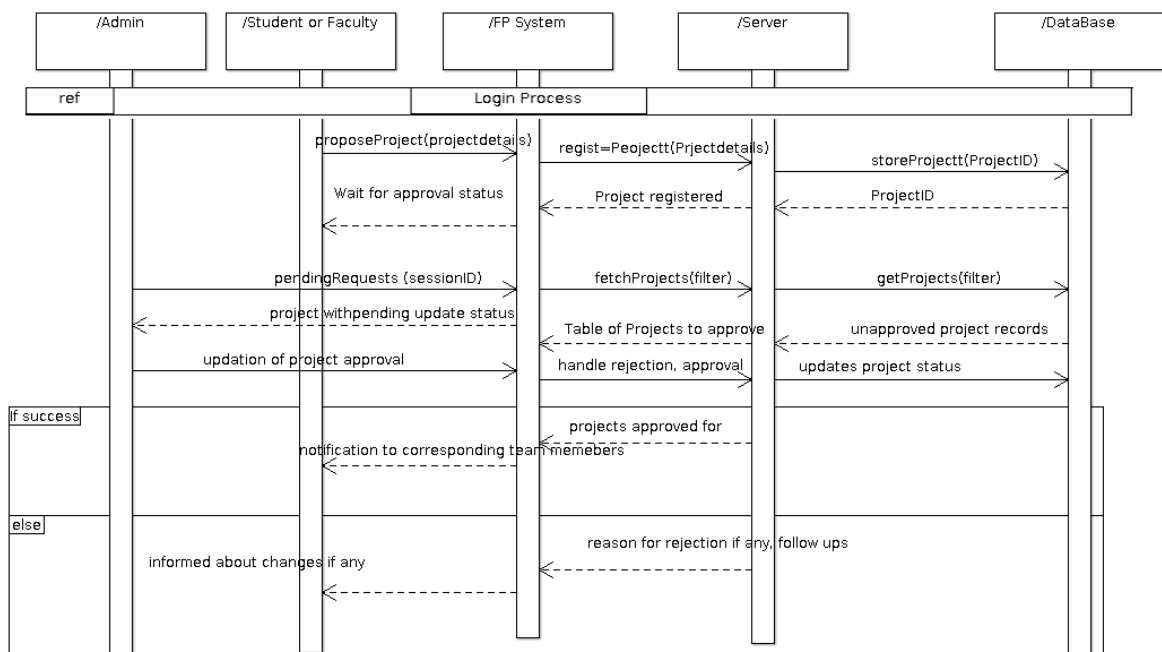
- Search by calendar period
- The guest, or incoming project applicant uses this to extend their ideas and plan on the project ahead of time.
- The admin, uses this to reduce project replications

UML Sequence Diagrams

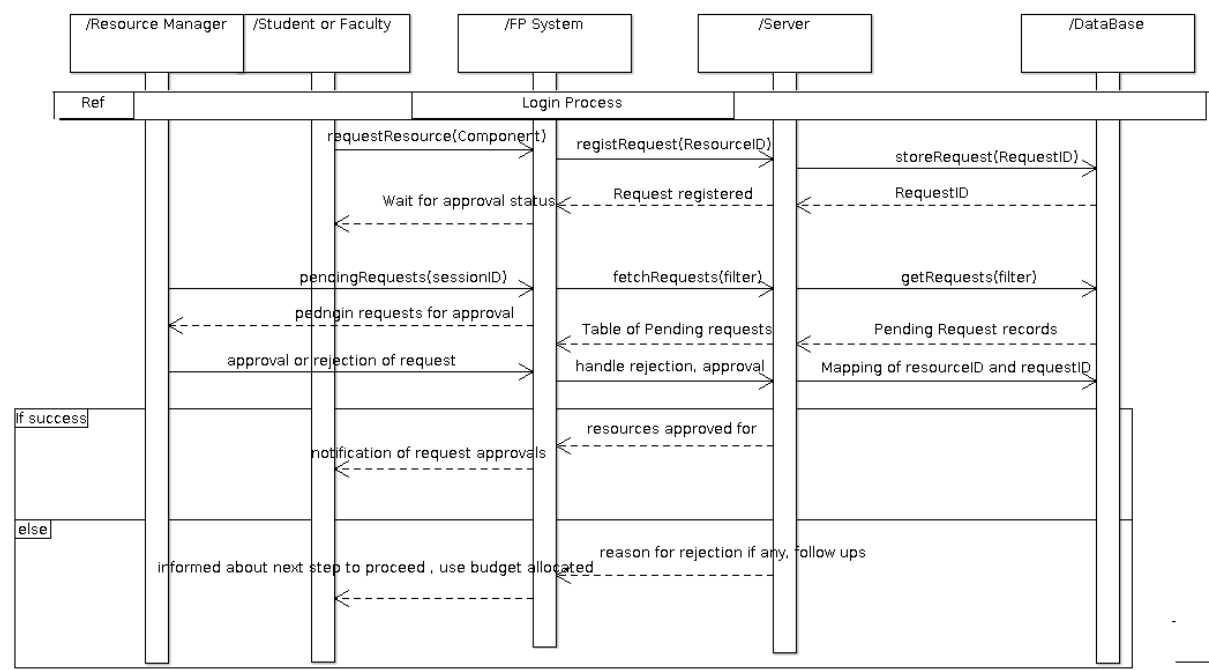
Login Process - Reusable Template



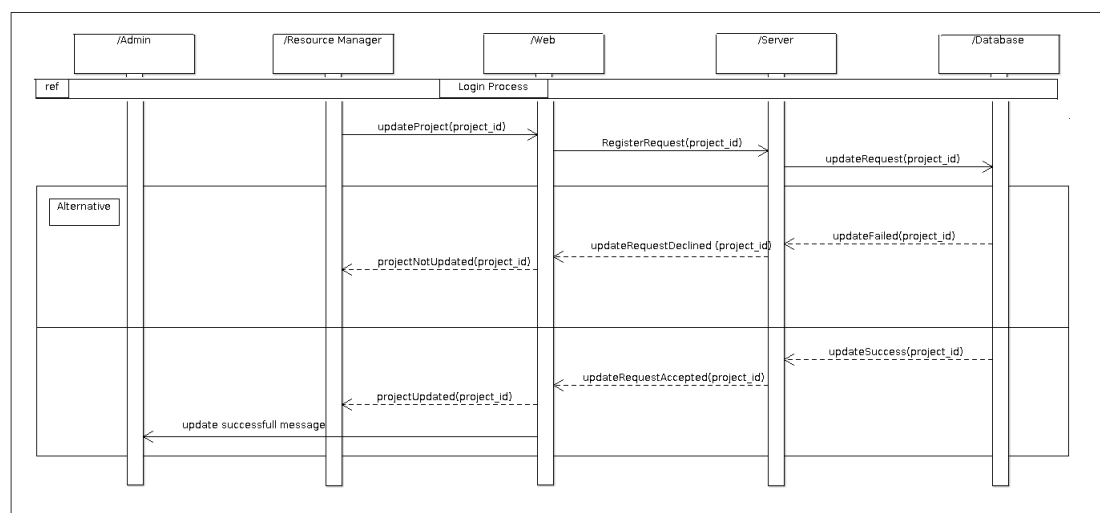
Main Scenario 1: Student/Faculty Registers Project



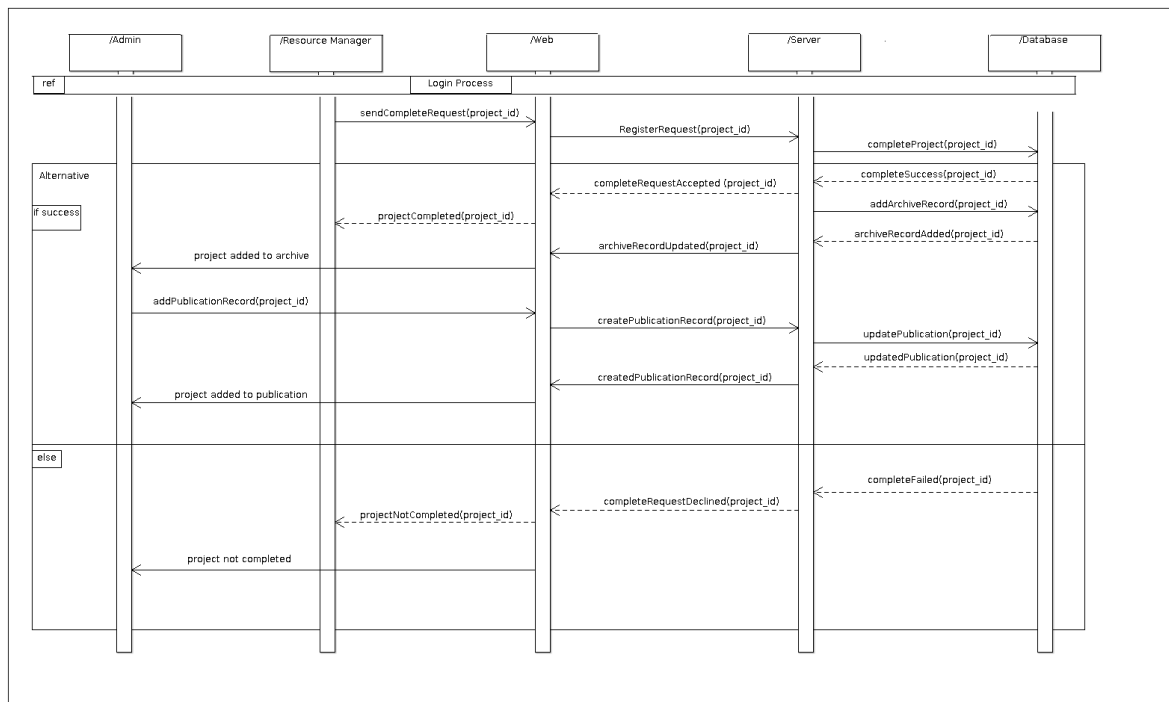
Main Scenario 2: Request for Resources



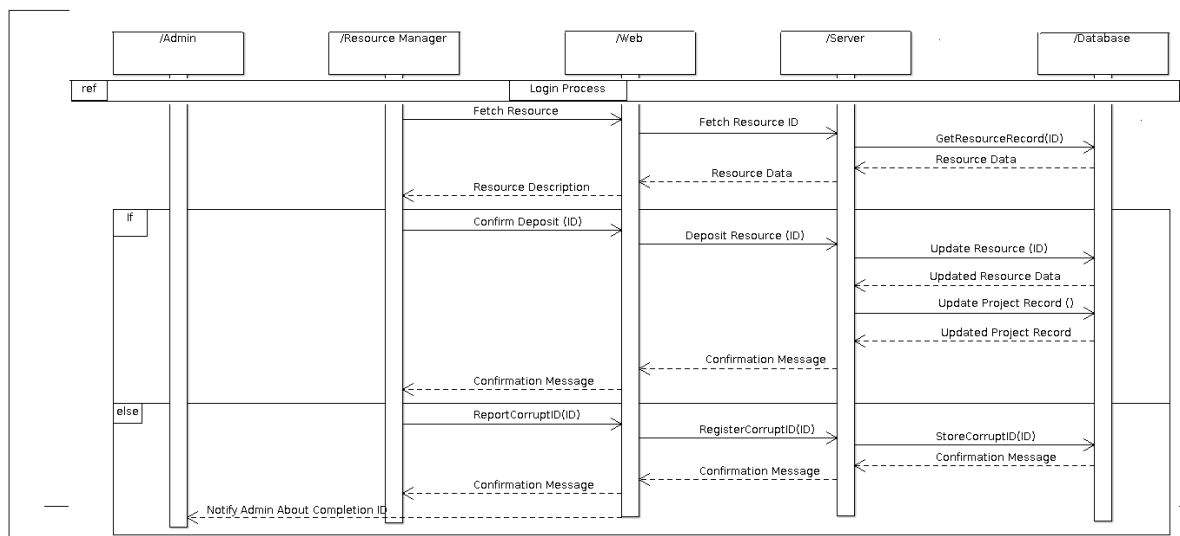
Alternate Scenario 1: Student/Faculty Updates Project Progress



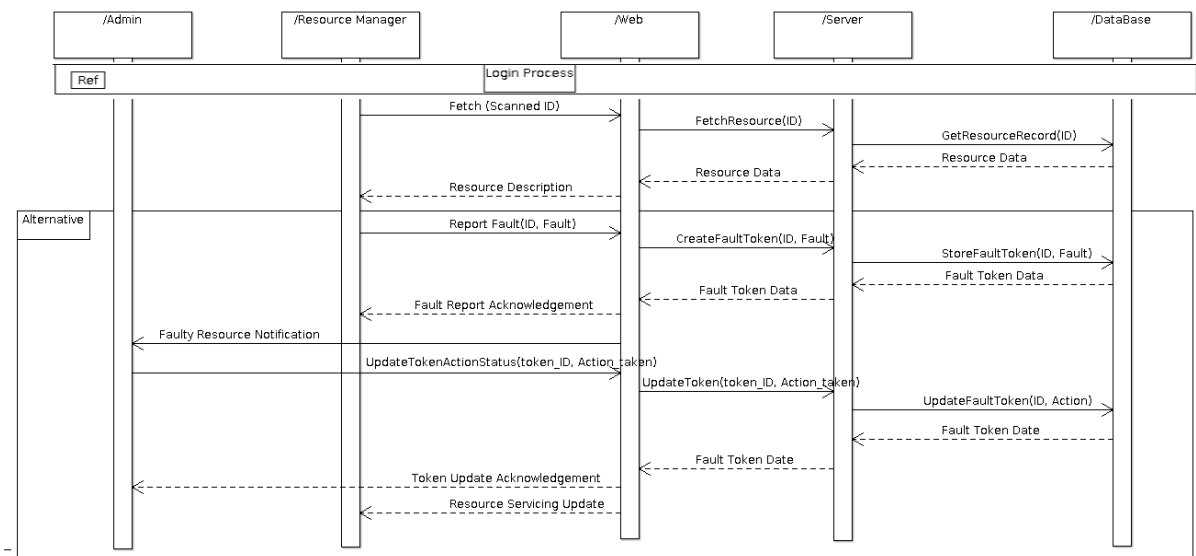
Alternate Scenario 2: Student/Faculty Records Project Completion



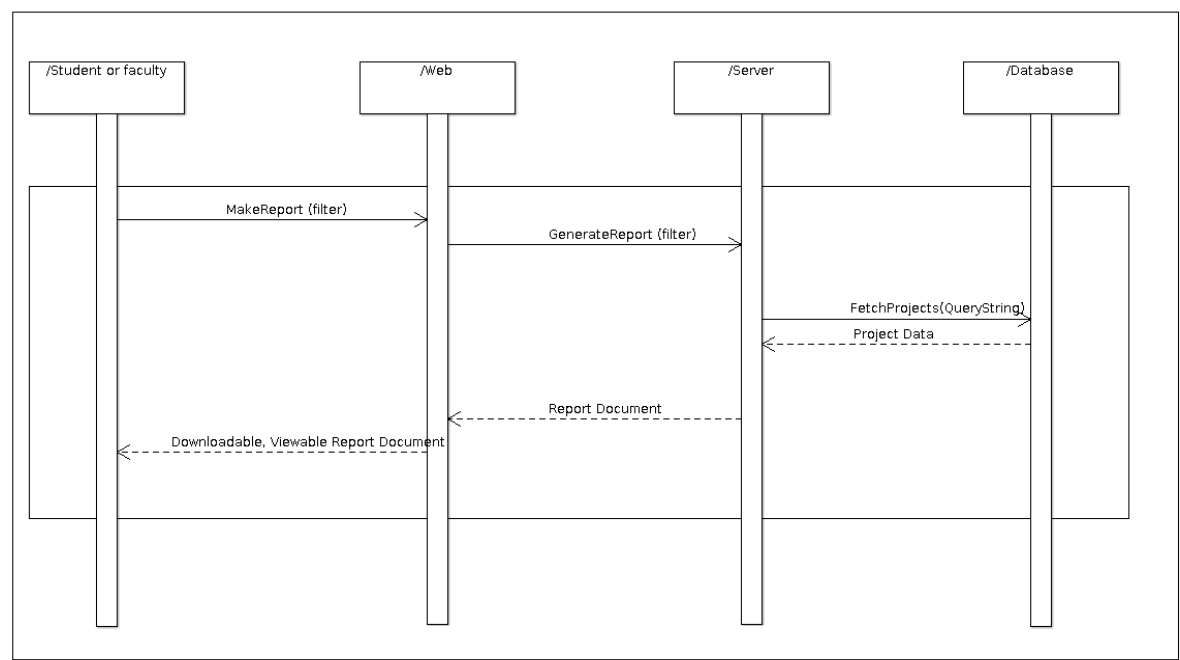
Alternate Scenario 3: Resource Manager Reclaim Resources



Alternate Scenario 4: Resource Manager Reports Faulty Components



Alternate Scenario 5: Guest Searches and Generates Project Stats



Observations

While identifying the scenarios and drafting the sequence diagrams for this FP Tracking Software System, we observed the following:

- Real world entities are seldom easily represented using abstract class descriptions
- Sequential interactions between multiple entities of the software system seamlessly represent the message passing at multiple levels
- Inter-level operations and their analysis gives a limpid picture the underlying implementation architecture
- Interaction diagrams can be readily converted and can serve as great assistance in the implementation of the final software system

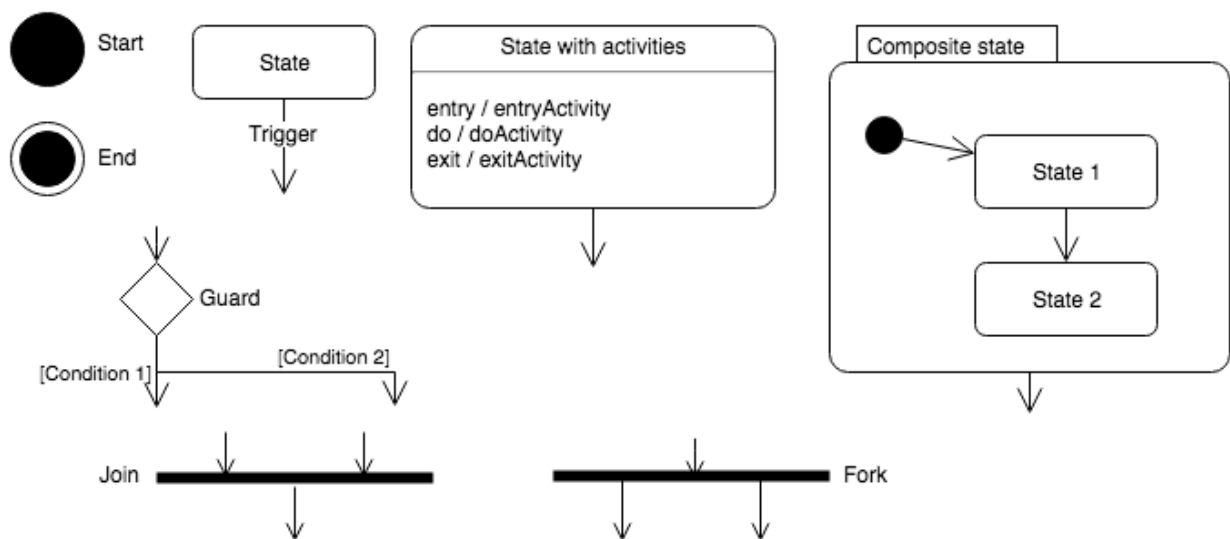
State Diagram

Aim

- To portray various changes in state of the class and not the processes or commands causing the changes.
- To give an abstract description of the behavior of a system.
- This behavior is analyzed and represented by a series of events that can occur in one or more possible states.

UML Notations

Basic UML state diagram notation



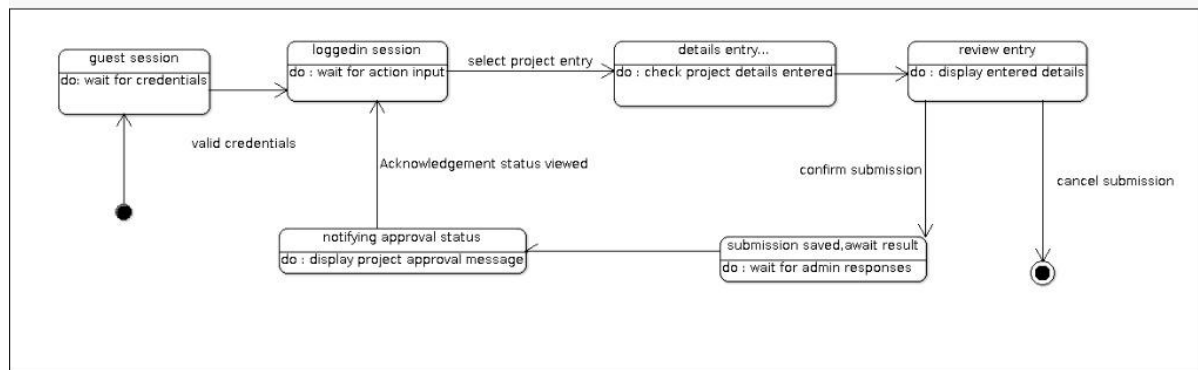
Identification of States

Generic State Template	Container Workflows	Description
Details/Field Entry	1, 2, 3, 5, 6	State that persistently performs data verification of entered fields and awaits completion of data entry in real-time
Data Entry Review	1, 2, 3, 5, 6	A review state that facilitates updation/cancellation of the details entered for a specific workflow and allows forwarding or backtracking to its immediate states
Guest Session	1, 2, 3, 4, 5	A dashboard display with response action triggers for a guest user of the website. Also provides the endpoint to transition to a logged-in session
Logged-In Session	1, 2, 3, 4, 5	A dashboard display with response action triggers for a registered user of the website — student, faculty, resource manager or admin
Waiting For User Action	1, 3	State that requires manual intervention to continue to one of the next states on the corresponding workflow
Dispatching Notifications	1, 3	State that prepares and dispatches email and/or dashboard based notification of artifact updates to the concerned stakeholders
Request / Complaint Registration	2, 3	State that compiles entered data and creates an associated entry in the database for further processing
Displaying Summaries	1, 2, 3, 4, 5, 6	State that displays a summary of success or failure for the particular workflow, at the end of all its elemental state transitions and activities
Update / Check Inventory	3, 4, 5	State that searches and interfaces the webpage with the inventory, almost always used only by the resource manager
Archiving Projects	2	State that converts and ongoing project record to a completed project archive, in the database

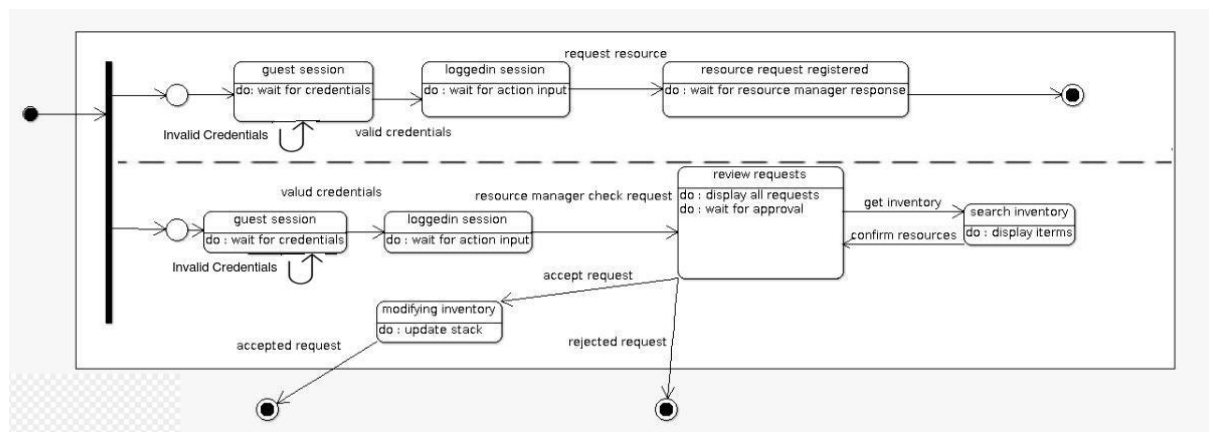
Retrieving Projects	2, 3, 6	State that interfaces the database to retrieve projects based on search filter criteria and forwards the data for display or report generation
Generating Reports	6	State that parses details of a display page and prepares a downloadable and printable document format of the data, typically for audit, summary and statistical analysis purposes

State Machine Diagram

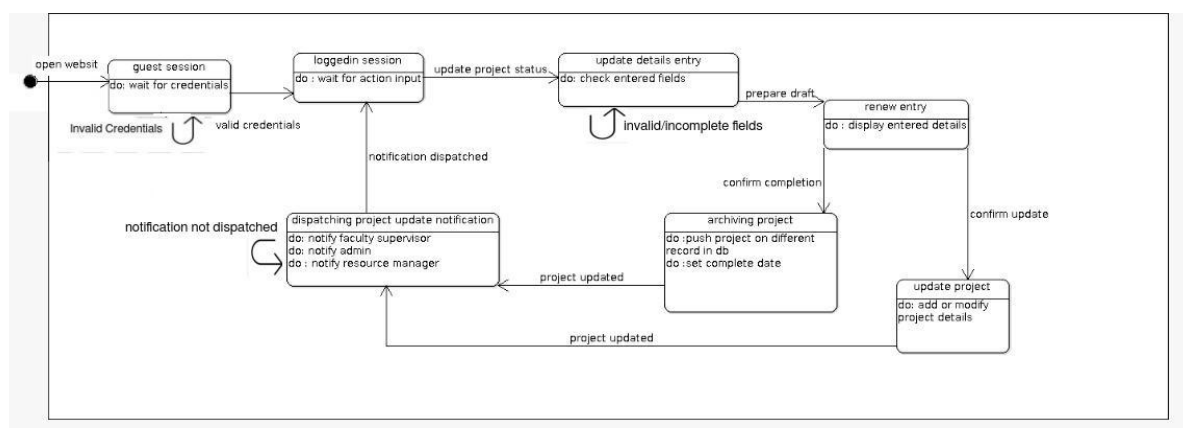
Workflow 1: Record Project Application



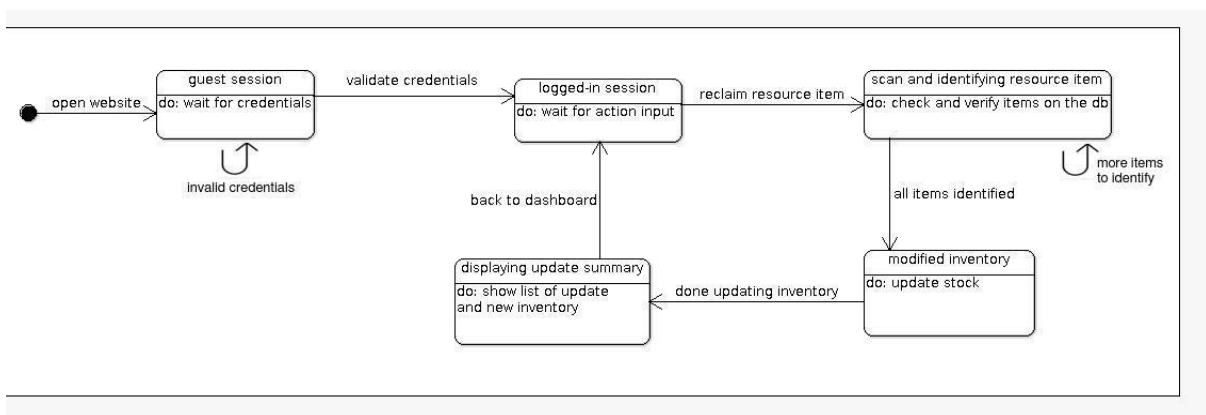
Workflow 2: Resource Request



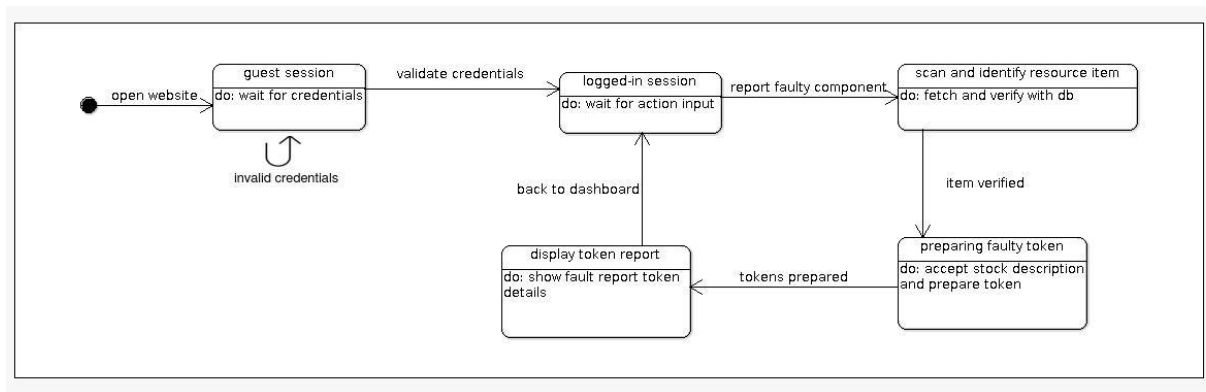
Workflow 3: Project Milestones and Completion Updation



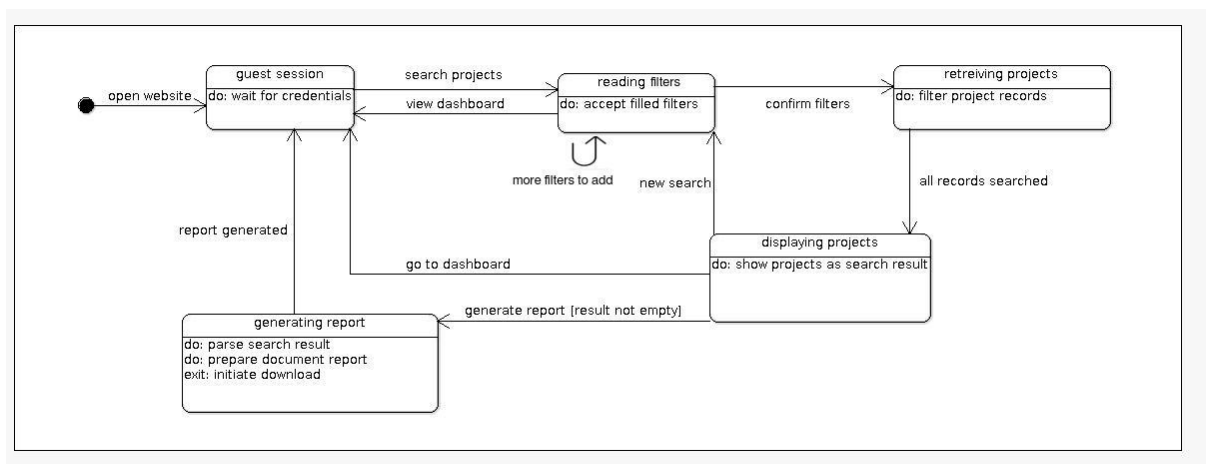
Workflow 4: Resource Reclamation



Workflow 5: Resource Fault Reporting



Workflow 6: Project Search and Report Generation



Observations

While identifying the states and drafting the state machine for this FP Tracking Software System, we observed the following:

- States of the software can be defined based on the different processing workflows that the system undergoes in response to each user action
- Several of the states, across multiple workflows, can be naturally grouped into generic comprehensible templates
- State transitions and pre-conditions for these transitions are well-represented and elicited during this documentation routine
- The state diagrams serve as a perfect resource for extending them to a software that is based on the controller-view architecture with resource-routing

Activity Diagram

Aim

- To capture the dynamic behavior of the system
- To show business and software processes as a progression of actions.
- To describe business processes and use cases as well as to document the implementation of system processes.

General Description

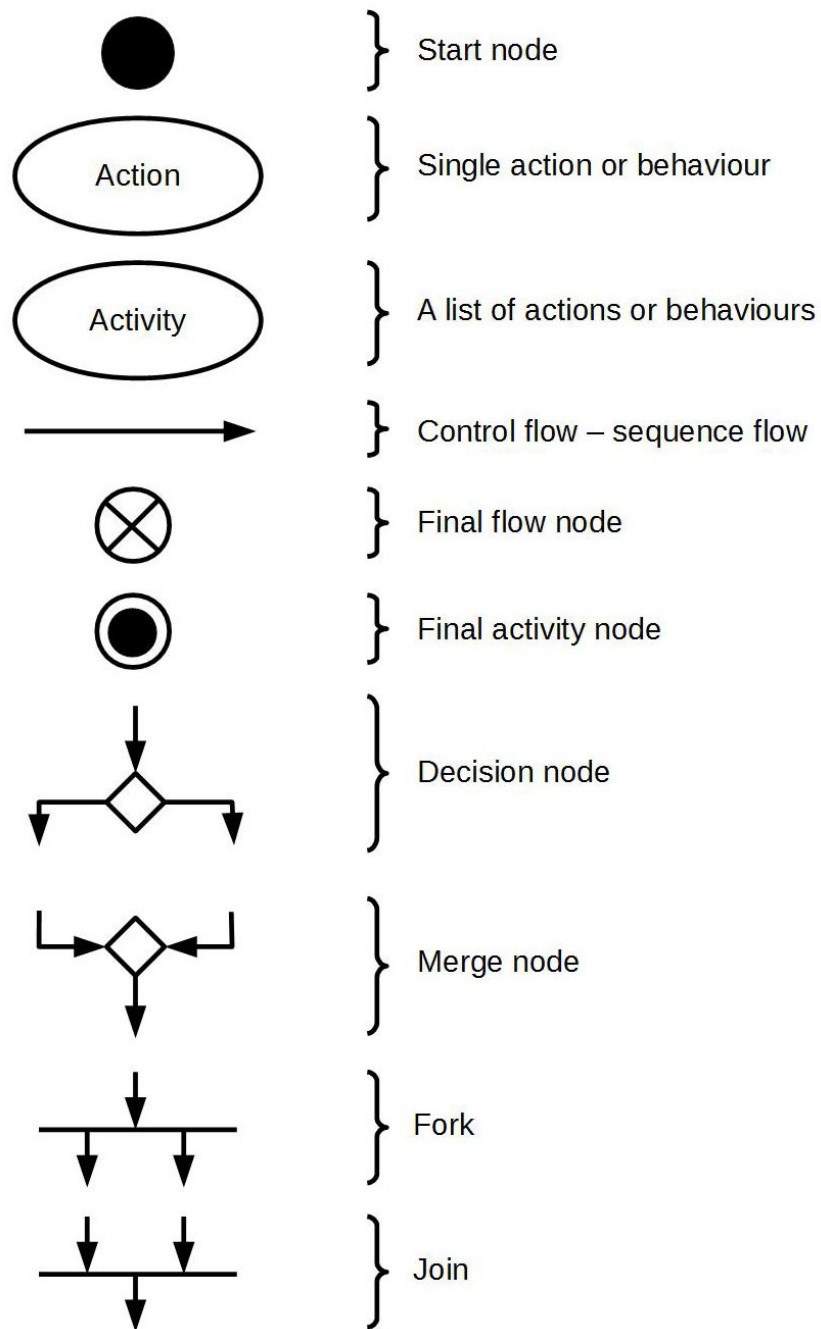
An activity diagram shows business and software processes as a progression of actions. These actions can be carried out by people, software components or computers. Activity diagrams are used to describe business processes and use cases as well as to document the implementation of system processes.

Even the most complex progressions can be visualized by activity diagrams. Sequential and peripheral workflows are depicted by control and object flows. Activity diagrams represent activities that are made up by a flow of actions.

Activity diagrams are ideal for describing the following processes:

- Use cases and the steps described in them,
- Business processes or workflows among users and systems,
- Software protocol, i.e. the permissible sequence of interactions between components
- Software algorithms

UML Notations



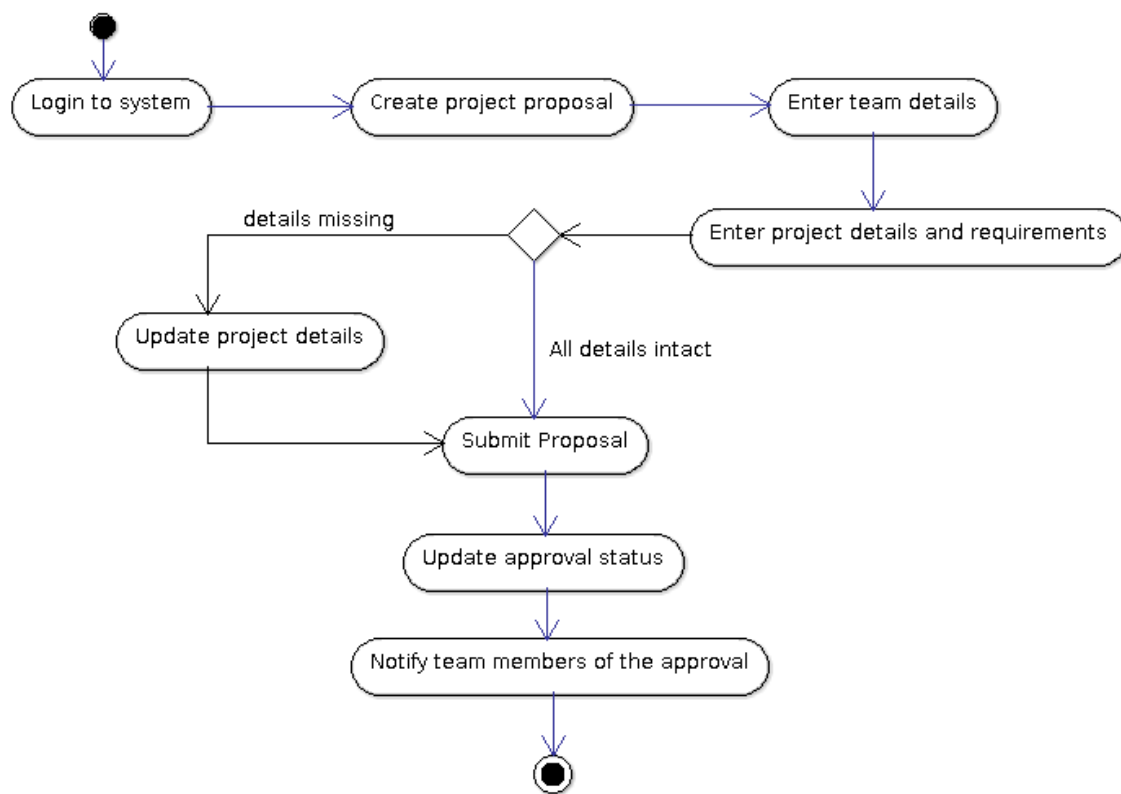
Identification of Activity Effector States

Generic State Template	Container Workflows	Description
Details/Field Entry	1, 2, 3, 5	State that persistently performs data verification of entered fields and awaits completion of data entry in real-time
Data Entry Review	1, 2, 3, 5	A review state that facilitates updation/cancellation of the details entered for a specific workflow and allows forwarding or backtracking to its immediate states
Guest Session	1, 2, 3, 4, 5	A dashboard display with response action triggers for a guest user of the website. Also provides the endpoint to transition to a logged-in session
Logged-In Session	1, 2, 3, 4, 5	A dashboard display with response action triggers for a registered user of the website — student, faculty, resource manager or admin
Waiting For User Action	1, 3	State that requires manual intervention to continue to one of the next states on the corresponding workflow
Dispatching Notifications	1, 3	State that prepares and dispatches email and/or dashboard based notification of artifact updates to the concerned stakeholders
Request / Complaint Registration	2, 3	State that compiles entered data and creates an associated entry in the database for further processing
Displaying Summaries	1, 2, 3, 4, 5	State that displays a summary of success or failure for the particular workflow, at the end of all its elemental state transitions and activities
Update / Check Inventory	3, 4, 5	State that searches and interfaces the webpage with the inventory, almost always used only by the resource manager
Archiving Projects	2	State that converts and ongoing project record to a completed project archive, in the database

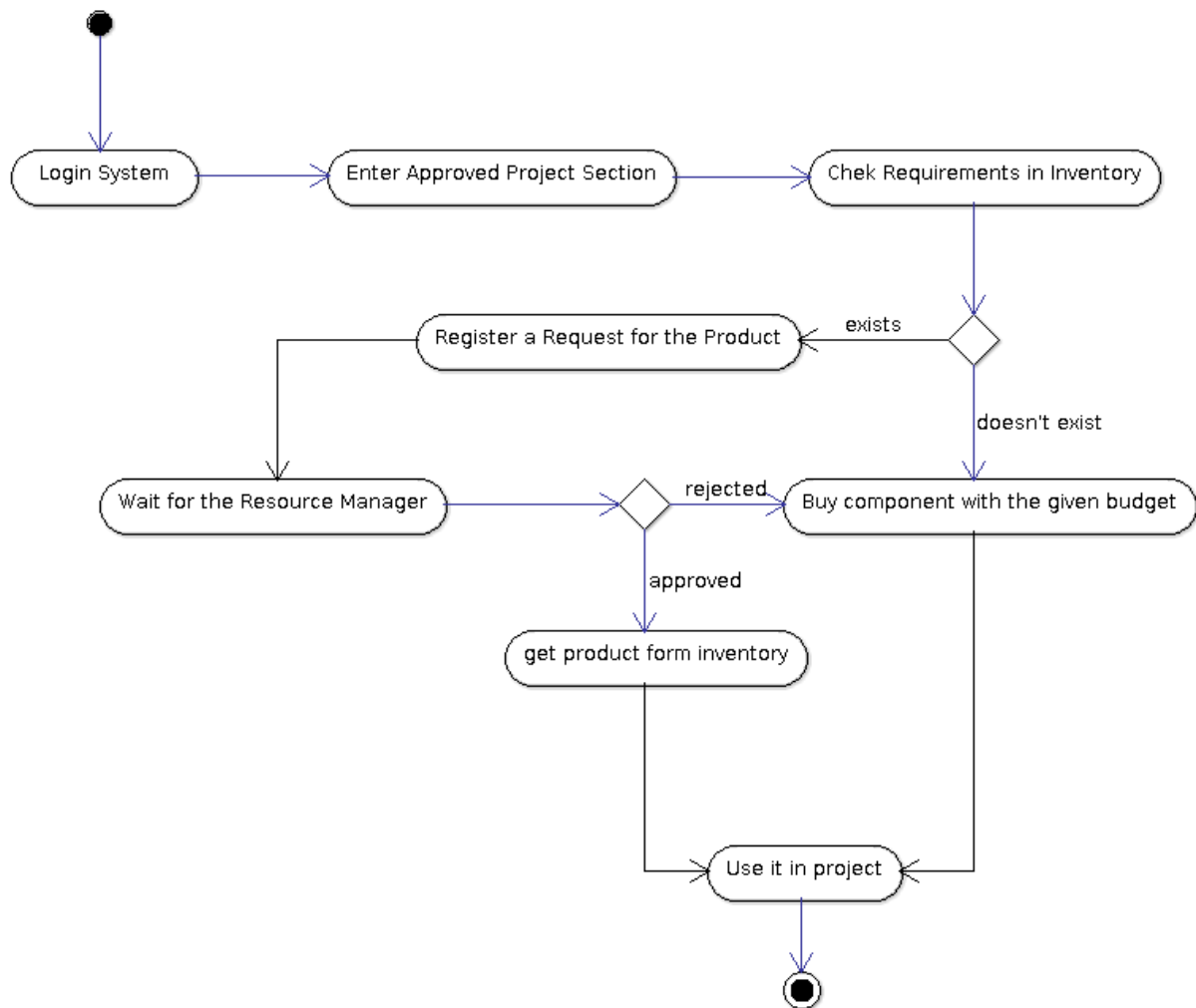
Retrieving Projects	2, 3	State that interfaces the database to retrieve projects based on search filter criteria and forwards the data for display or report generation
Generating Reports	5	State that parses details of a display page and prepares a downloadable and printable document format of the data, typically for audit, summary and statistical analysis purposes

Activity Diagram

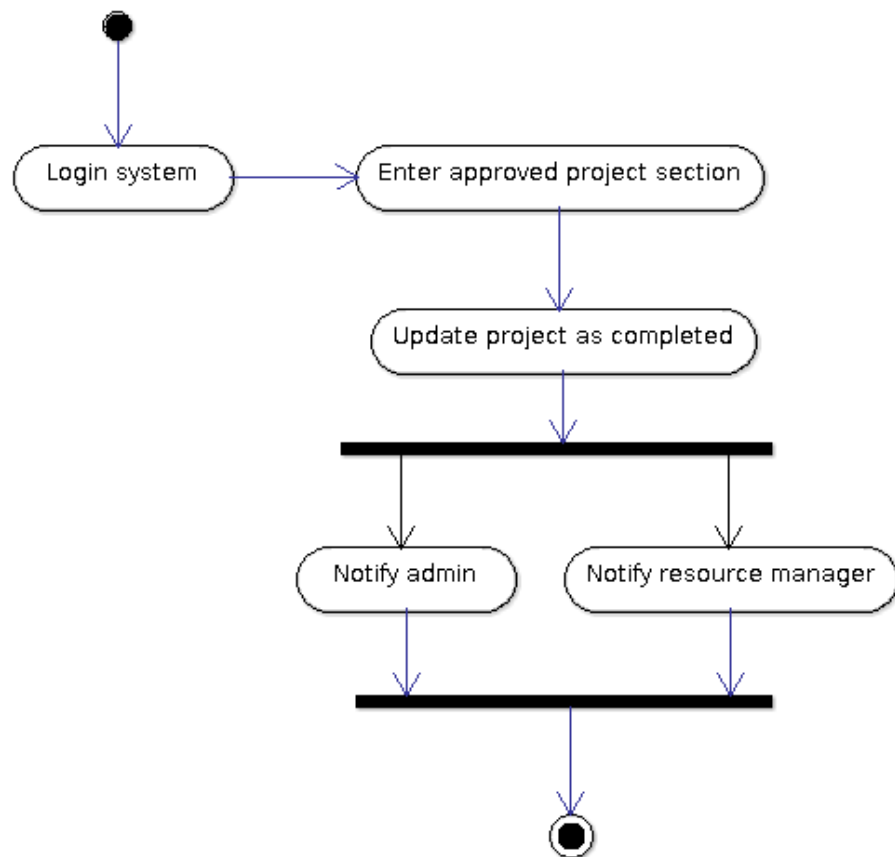
Workflow 1: Record Project Application



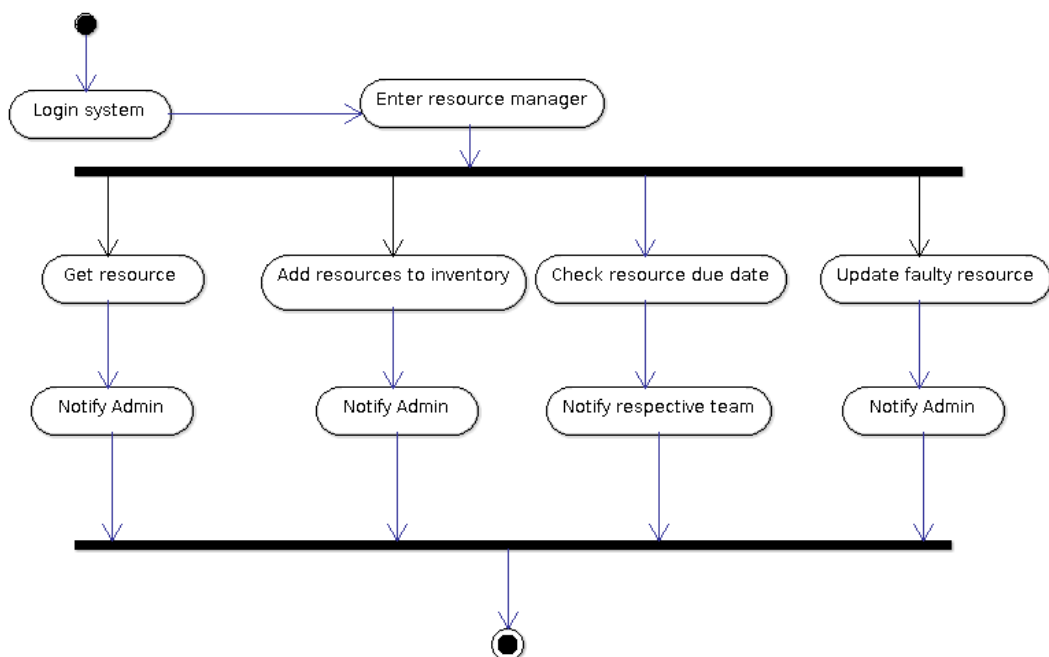
Workflow 2: Resource Allocation Request



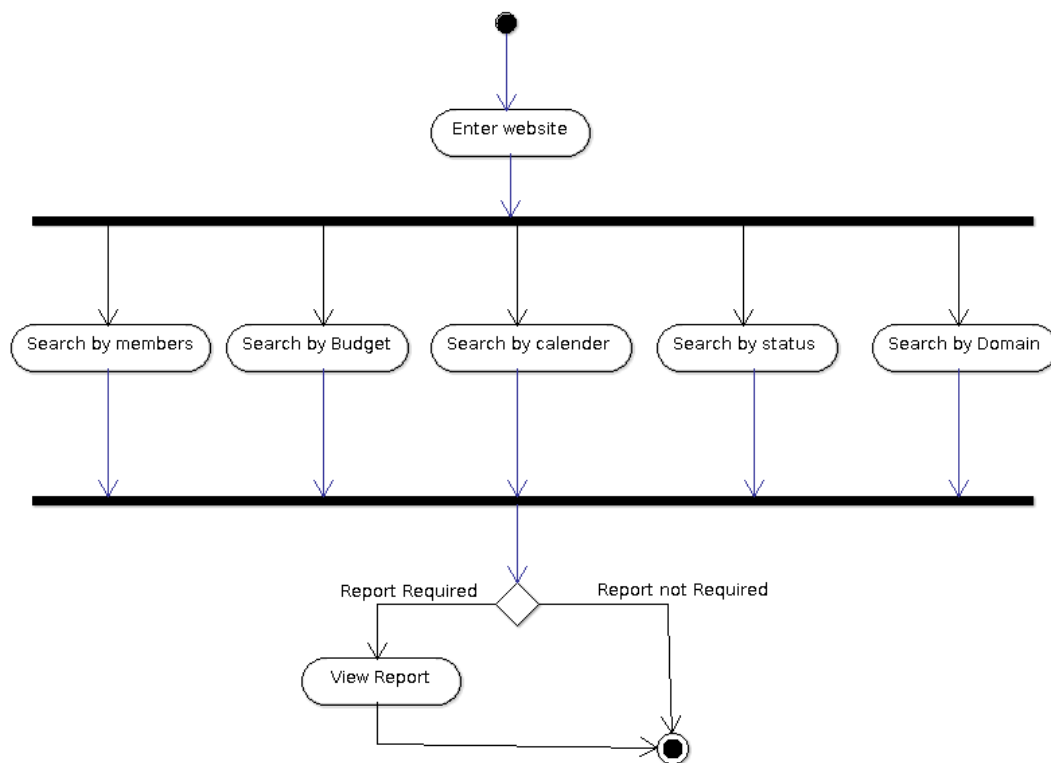
Workflow 3: Project Milestones and Completion Update



Workflow 4: Resource Management



Workflow 5: Project Search and Report Generation



Observations

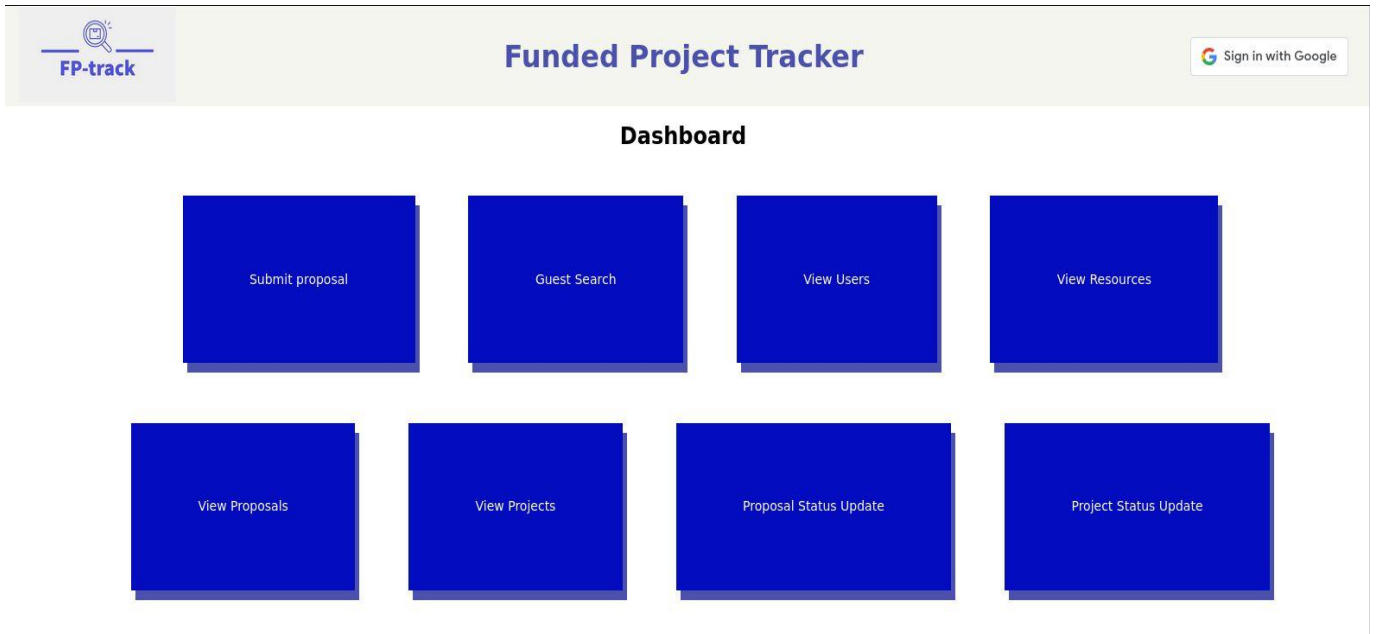
While identifying the states and drafting the activity diagram for this FP Tracking Software System, we observed the following:

- States of the software can be defined based on the different processing workflows that the system undergoes in response to each user action
- Several of the states, across multiple workflows, can be naturally grouped into generic comprehensible templates
- Functional workflows can be devised based on state transitions for each scenario to draft a concrete conceptual representation of its action sequence
- Specific user roles, and their parts in each workflow are well represented through activity diagrams
- The activity diagrams serve as a perfect resource for extending them to a software that is based on the controller-view architecture with resource-routing, by laying out a clear outline of each workflow, and consequently, the flow of control across modules of the software

Implementation Interfaces

End-User View

User Dashboard





Submit Proposals

The screenshot shows a 'Submit Proposals' form with a teal background. The form fields are arranged vertically: a text input for 'Leader', a text input for 'Members', a dropdown menu for 'Funding Type' currently set to 'Internal', a text input for 'Funding Agency', a file upload section for 'Proposal Document' with a 'Choose File' button and the text 'No file chosen', a text input for 'Budget', and a purple 'Submit' button at the bottom.



View Proposals

An overview of pending project proposals with filters by multiple fields

Funded Project Tracker		
		
<div><div>IOT DRIVEN SMART TRAINS</div><div>internalIDF Trust</div><div>Rs10000</div></div>	<div><div>IOT DRIVEN CLASSROOM</div><div>internalSSN Trust</div><div>Rs25000</div></div>	<div><div>NUMBER PLATE DETECTION</div><div>externalDST-SERB</div><div>Rs50000</div></div>
<div><div>TEST</div><div>internalSSN</div><div>Rs10000</div></div>	<div><div>KRTEST1</div><div>internalSSN</div><div>Rs20000</div></div>	<div><div>IOT DRIVEN TRAINS</div><div>internalSSN Trust</div><div>Rs40000</div></div>
<div><div>NUMBER PLATE DETECTION</div><div>externalDST-SERB</div><div>Rs50000</div></div>	<div><div>NUMBER PLATE DETECTION</div><div>externalDST-SERB</div><div>Rs50000</div></div>	<div><div>NUMBER PLATE DETECTION</div><div>externalDST-SERB</div><div>Rs50000</div></div>


View Projects - Overview

An overview of ongoing and completed projects with filters by multiple fields


Funded Project Tracker		
		
<div><input type="text" value="Enter Post Title"/> name ▼</div>		
<div><div>IoT Driven Smart Trains</div><div>lotMachine_learning</div></div>	<div><div>Number Plate Detection, Again</div><div>Machine_learning</div></div>	<div><div>IoT Driven Smart Trains</div><div>lotMachine_learning</div></div>
<div><div>IoT Driven Smart Trains</div><div>lotMachine_learning</div></div>	<div><div>IoT Driven Smart Trains</div><div>lotMachine_learning</div></div>	<div><div>IoT Driven Smart Trains</div><div>lotMachine_learning</div></div>
<div><div>IoT Driven Smart Trains</div><div>lotMachine_learning</div></div>	<div><div>IoT Driven Smart Trains</div><div>lotMachine_learning</div></div>	<div><div>IoT Driven Smart Trains</div><div>lotMachine_learning</div></div>

View Projects - Detailed

Expanded view of the Principal Investigator of the project



Funded Project Tracker



IoT Driven Smart Trains

Domain: iot machine_learning

Budget requested: Rs10000

Budget approved: Rs10000

Duration: 20 months

Proposal.pdf: [link](#)

Outcomes

Status

Principal Investigator

student

user

Yeates, Ben

SSN COLLGE OF ENGINEERING

ben@gmail.com

Access Rights


user

Members


Supervisors

Resource assigned(collaped)

Expanded view of Members of the project



Funded Project Tracker



IoT Driven Smart Trains

Domain: iot machine_learning

Budget requested: Rs10000

Budget approved: Rs10000

Duration: 20 months

Proposal.pdf: [link](#)

Outcomes

Status

Principal Investigator

Members

student

user

Xiao, Claire

SSN COLLGE OF ENGINEERING

claire@gmail.com

Access Rights

user

student

user

Woakes, Derek

SSN COLLGE OF ENGINEERING

derek@gmail.com


Access Rights

user

Supervisors

Resource assigned(collaped)

Expanded view of Status Updates of the project



Funded Project Tracker

Sign in with Google

IoT Driven Smart Trains

Domain: iot machine_learning

Budget requested: Rs10000

Budget approved: Rs10000

Duration: 20 months

Proposal.pdf: [link](#)

Outcomes

Status

1. Literature Survey Completed

2. Patent submitted Fully approved design patent made

Principal Investigator


Members

Supervisors

Resource assigned(collaped)

Resource assignable(collaped)

Expanded view of Supervisors of the project



Funded Project Tracker

Sign in with Google

IoT Driven Smart Trains

Domain: iot machine_learning

Budget requested: Rs10000

Budget approved: Rs10000

Duration: 20 months

Proposal.pdf: [link](#)

Outcomes

Status

Principal Investigator

Members

Supervisors

faculty

user

User, Admin

SSN COLLGE OF ENGINEERING

admin@admin.com


Access Rights

admin

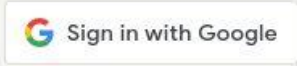
Resource assigned(collaped)

Approve/Reject Proposals [Admin]

Admin console to approve or reject proposal based on decision from the funding authority



Funded Project Tracker



funding_agency ▾

internal

KRTEST1

SSN

Rs20000

Accept

Reject

internal

IOT DRIVEN TRAINS

SSN Trust

Rs40000

Accept

Reject

external

NUMBER PLATE DETECTION

DST-SERB

Rs50000

Accept

Reject

external

NUMBER PLATE DETECTION

DST-SERB

Rs50000

Accept

Reject

external

NUMBER PLATE DETECTION

DST-SERB

Accept

Reject

external

NUMBER PLATE DETECTION


DST-SERB

Accept

Reject

Allocate Resources for Project [Resource Manager]

Interface for resource manager to allocate resources for a project from the inventorized database



Funded Project Tracker

Sign in with Google

IoT Driven Smart Trains

Domain: iot machine_learning

Budget requested: Rs10000

Budget approved: Rs10000

Duration: 20 months

Proposal.pdf: [link](#)

Outcomes

Status

Principal Investigator

Members

Supervisors

Resource assigned

Filter...

name

Resource assignable

Filter...

name

Arduino UNO

hardware

SSN COLLEGE OF ENGINEERING

Microprocessor board with GPIO pins. Version 3

Available quantity: 2

Requesting Quantity

+

-

 Request

Matlab

software

SSN COLLEGE OF ENGINEERING

Fully licensed Matlab V1

Available quantity: 3

Requesting Quantity

+

-

 Request

Raspberry Pi 3

hardware

SSN COLLEGE OF ENGINEERING

Microcontroller board without WiFi built-in

Available quantity: 0


Requesting Quantity

+

-

 Request

User Management [Admin]



Funded Project Tracker

Sign in with Google

faculty

user

User, Admin

SSN COLLEGE OF ENGINEERING

admin@admin.com

Access Rights

admin

student

user

Yeates, Ben

SSN COLLEGE OF ENGINEERING

ben@gmail.com

Access Rights

user

student

user

Xiao, Claire

SSN COLLEGE OF ENGINEERING

claire@gmail.com

Access Rights

user

student

user

Woakes, Derek

SSN COLLEGE OF ENGINEERING

derek@gmail.com

Access Rights

user

student

user

Vincent, Erin

SSN COLLEGE OF ENGINEERING

erin@gmail.com

Access Rights

user

faculty

user

Timothy, Greg

SSN COLLEGE OF ENGINEERING

greg@gmail.com

Access Rights

user

faculty

user

Yeates, Harry

SSN COLLEGE OF ENGINEERING

harry@gmail.com

Access Rights

resource_mgr

student

user

D, Karthik

SSN COLLEGE OF ENGINEERING

karthik19047@cse.ssn.edu.in

Access Rights

user admin resource_mgr

localhost:3000/api/user/628abd6a4bc531d7264a0aa2

Database View

Users

FPT-Cluster-J2K2

VERSION5.0.9

Overview

Real Time

Metrics

Collections

Search

Profiler

Performance Advisor

Online Archive

Cmd Line T

DATABASES: 1

COLLECTIONS: 9

+ Create Database

Q Search Namespaces

FPT_DB

changelog

migrations

project_updates

projects

proposals

resource_assignments

resource_groups

resources

users

FPT_DB.users

STORAGE SIZE: 36KB

TOTAL DOCUMENTS: 22

INDEXES TOTAL SIZE: 72KB

Find

Indexes

Schema Anti-Patterns

Aggregation

Search Indexes

FILTER

{ field: 'value' }

QUERY RESULTS: 1-20 OF MANY

_id: ObjectId("628abd604bc531d7264aa9b")

first_name: "Admin"

last_name: "User"

date_of_birth: 1999-09-09T00:00:00.000+00:00

email: "admin@admin.com"

role: "faculty"

access: Array

deleted_at: null

createdAt: 2022-05-22T22:47:04.532+00:00

Proposals

FPT-Cluster-J2K2

VERSION5.0.9

REGIONGCP Mumbai

Overview

Real Time

Metrics

Collections

Search

Profiler

Performance Advisor

Online Archive

Cmd Line Tools

DATABASES: 1

COLLECTIONS: 9

+ Create Database

Q Search Namespaces

FPT_DB

changelog

migrations

project_updates

projects

proposals

resource_assignments

resource_groups

resources

users

FPT_DB.proposals

STORAGE SIZE: 10.57MB

TOTAL DOCUMENTS: 23

INDEXES TOTAL SIZE: 36KB

Find

Indexes

Schema Anti-Patterns

Aggregation

Search Indexes

FILTER

{_id: ObjectId("628b5425099b1ba80b543da6")}

OPTIONS

Ag

_id: ObjectId("628b5425099b1ba80b543da6")

title: "IoT Driven Smart Trains"

decription: "blah blah blah..."

domains: Array

supervisors: Array

leader: ObjectId("628abd6a4bc531d7264aa9b")

members: Array

funding_type: "internal"

funding_agency: "IDF Trust"

pdf_document: Binary('JVBERi0xLjQKJcfsj6IKNSAwIG9iago8PC9hZWN0dGggNIAwIFIvRmlsdGdyI9G6bGF0ZURlY29kZT4+CnN0cmVhbQp4nMvav448d...', 0)

budget: 10000

approved_on: 2022-06-14T05:51:11.953+00:00

rejected_on: null

deleted_at: null

Projects

FPT-Cluster-J2K2

VERSION 5.0.9

OverviewReal TimeMetricsCollectionsSearchProfilerPerformance AdvisorOnline ArchiveCmd Line 1

DATABASES: 1 COLLECTIONS: 9

+ Create Database

Q Search Namespaces

FPT_DB

changelog

migrations

project_updates

projects

proposals

resource_assignments

resource_groups

resources

users

FPT_DB.projects

STORAGE SIZE: 36KB TOTAL DOCUMENTS: 11 INDEXES TOTAL SIZE: 36KB

FindIndexesSchema Anti-Patterns 0AggregationSearch Indexes ●

FILTER { field: 'value' }

>

_id: ObjectId("62a4e23619d791faaf76e19a")

proposal: ObjectId("62a4e28b19d791faaf76e195")

approved_budget: 12000

approved_duration: 18

completed_on: null

> status_updates: Array

> outcomes: Array

createdAt: 2022-06-11T18:43:02.687+00:00

updatedAt: 2022-06-13T16:54:45.470+00:00

__v: 0

Resource Groups

FPT-Cluster-J2K2

VERSION 5.0.9

OverviewReal TimeMetricsCollectionsSearchProfilerPerformance AdvisorOnline ArchiveCmd Line To

DATABASES: 1 COLLECTIONS: 9

+ Create Database

Q Search Namespaces

FPT_DB

changelog

migrations

project_updates

projects

proposals

resource_assignments

resource_groups

resources

users

FPT_DB.resource_groups

STORAGE SIZE: 36KB TOTAL DOCUMENTS: 3 INDEXES TOTAL SIZE: 36KB

FindIndexesSchema Anti-Patterns 0AggregationSearch Indexes ●

FILTER { field: 'value' }

QUERY RESULTS: 1-3 OF 3

_id: ObjectId("628abd6b4bc531d7264a0aa8")

name: "Arduino UNO"

description: "Microprocessor board with GPIO pins. Version 3"

kind: "hardware"

is_multi_assignable: false

deleted_at: null

createdAt: 2022-05-22T22:47:07.180+00:00

updatedAt: 2022-05-22T22:47:07.180+00:00

__v: 0

FPT-Cluster-J2K2

VERSION
5.0.9

Overview

Real Time

Metrics

Collections

Search

Profiler

Performance Advisor

Online Archive

Cmd Line T

DATABASES: 1COLLECTIONS: 9

+ Create Database

Search Namespaces

FPT_DB

changelog

migrations

project_updates

projects

proposals

resource_assignments

resource_groups

resources

users

FPT_DB.resources

STORAGE SIZE: 36KBTOTAL DOCUMENTS: 5INDEXES TOTAL SIZE: 72KB

Find

Indexes

Schema Anti-Patterns

Aggregation

Search Indexes

FILTER

{ field: 'value' }

```
_id: ObjectId("628abd6b4bc531d7264a9aac")
resource_group: ObjectId("628abd6b4bc531d7264a9aa8")
remarks: "Purchased through Amazon"
scan_code: "ABCD123456"
expiry: null
faulted_at: null
deleted_at: null
createdAt: 2022-05-22T22:47:07.389+00:00
updatedAt: 2022-05-22T22:47:07.389+00:00
__v: 0
```

FPT-Cluster-J2K2

VERSION
5.0.9

OverviewReal TimeMetricsCollectionsSearchProfilerPerformance AdvisorOnline ArchiveCmd Line T

DATABASES: 1COLLECTIONS: 9

+ Create Database

Q Search Namespaces

FPT_DB

changelogmigrationsproject_updatesprojectsproposalsresource_assignmentsresource_groupresourcesusers

FPT_DB.resource_assignments

STORAGE SIZE: 30KB TOTAL DOCUMENTS: 3 INDEXES TOTAL SIZE: 30KB

FindIndexesSchema Anti-PatternsAggregationSearch Indexes

FILTER { field: 'value' }

QUERY RESULTS: 1-3 OF 3

```
_id: ObjectId("62a729594264ac72715d9eef")  
resource: ObjectId("628abb0b4bc531d7264a0aac")  
assigned_to: ObjectId("62a4e23619d701faaf76e19a")  
assigned_by: ObjectId("628f1954f5b6a7772b0885b6")  
deleted_on: null  
createdAt: 2022-06-13T12:11:06.605+00:00  
updatedAt: 2022-06-13T12:11:06.605+00:00  
__v: 0
```

Source Code

Backend

controlller/api/auth.js

```
var express = require('express');
var mongoose = require('mongoose');
var { OAuth2Client } = require('google-auth-library');

var UserModel = require('../models/user');
var ErrorHandler = require('../helpers/error');

function getAuthUser(req, res, next) {
  // validate token
  var auth_client = new OAuth2Client({
    clientId: `${process.env.OAUTH_CLIENTID}`
  });
  auth_client.verifyIdToken({
    idToken: req.body.token,
    audience: `${process.env.OAUTH_CLIENTID}`
  })
  .then((ticket) => {
    var payload = ticket.getPayload();
    // retrieve user
    UserModel
    .onlyExisting()
    .getByEmail(payload.email)
    .then((user) => {
      if (user == null) {
        res.status(404).send({
          error_code: 901,
          message: "User must be created",
          url: "/api/user/",
          method: "POST"
        })
      }
      else {
        user.auth_token = req.body.token;
        res.status(200).send(user);
      }
    })
    .catch((error) => {
      res.status(400).send(
        ErrorHandler.construct_json_response(error)
      )
    });
  })
  .catch((error) => {
    res.status(400).send(
      ErrorHandler.construct_json_response(error)
    )
  })
}
```

```

    });
}

exports.getAuthUser = getAuthUser;

```

controller/api/user.js

```

var express = require('express');
var mongoose = require('mongoose');

var UserModel = require('../../models/user');
var ErrorHandler = require('../../helpers/error');

function create(req, res, next) {
  const user = new UserModel(req.body);
  user
    .save()
    .then((resource) => {
      res.status(201).send({
        id: resource._id,
        url: resource.url,
        message: "User created"
      })
    })
    .catch((error) => {
      res.status(400).send(
        ErrorHandler.construct_json_response(error)
      );
    });
};

function getAll(req, res, next) {
  // Extract filters
  const filterables = UserModel.getDirectFilterFields();
  const filters = Object.keys(req.query)
    .filter((field) => {
      return filterables.includes(field);
    })
    .reduce((obj, field) => {
      return Object.assign(
        obj,
        { [field]: req.query[field] }
      );
    }, {});

  // Get records
  UserModel
    .onlyExisting()
    .find(filters)

```

```

        .then((resources) => {
            res.status(200).send(resources);
        })
        .catch((error) => {
            res.status(400).send(
                ErrorHandler.construct_json_response(error)
            );
        });
    });
};

function getById(id, req, res, next) {
    if (mongoose.Types.ObjectId.isValid(id)) {
        UserModel
            .onlyExisting()
            .getById(id)
            .then((resource) => {
                res.status(200).send(resource);
            })
            .catch((error) => {
                res.status(400).send(
                    ErrorHandler.construct_json_response(error)
                );
            });
    }
    else {
        res.status(404).send({
            message: "User not found"
        });
    }
};

exports.create = create;
exports.getById = getById;
exports.getAll = getAll;

```

App.js

```

var createError = require("http-errors");
var express = require("express");
var path = require("path");
var cookieParser = require("cookie-parser");
var logger = require("morgan");
var dotenv = require("dotenv");

dotenv.config();

// Connect DB and Seed
var db = require("./db/connect");
//var _ = require('./init/seed_db');

var indexRouter = require("./routes/index");
var userRouter = require("./routes/user");
var proposalRouter = require("./routes/proposal");
var resourceGroupRouter = require("./routes/resource_group");

```

```

var resourceRouter = require("./routes/resource");
var projectRouter = require("./routes/project");

var resourceAssignmentRouter = require("./routes/resource_assignment");

var authRouter = require("./routes/auth");

var app = express();

// set request limit
app.use(express.json({ limit: "20mb" }));

// view engine setup
app.set("views", path.join(__dirname, "views"));
app.set("view engine", "pug");

app.use(logger("dev"));
app.use(express.json());
app.use(express.urlencoded({ extended: false }));
app.use(cookieParser());
app.use(express.static(path.join(__dirname, "public")));

/* Routing */

// Pre-functor for allow Cross Origin Requests
app.use((req, res, next) => {
  res.header("Access-Control-Allow-Origin", "*");
  res.header("Access-Control-Allow-Methods", "GET, PUT, POST, PATCH, DELETE");
  res.header(
    "Access-Control-Allow-Headers",
    "Origin, X-Requested-With, Content-Type, Accept"
  );
  return next();
});

// Normal routes
app.use("/", indexRouter);
app.use("/api/user", userRouter);
app.use("/api/proposal", proposalRouter);
app.use("/api/resource-group", resourceGroupRouter);
app.use("/api/resource", resourceRouter);
// <<<<<< HEAD
app.use("/api/resource-assignment", resourceAssignmentRouter);
// =====
// >>>>>> 8ef04d9992f6a449617d22334dbf60719305c3a5
app.use("/api/project", projectRouter);
app.use("/api/auth", authRouter);

// catch 404 and forward to error handler
app.use(function (req, res, next) {
  next(createError(404));
});

// error handler
app.use(function (err, req, res, next) {

```

```

// set locals, only providing error in development
res.locals.message = err.message;
res.locals.error = req.app.get("env") === "development" ? err : {};

// render the error page
res.status(err.status || 500);
res.render("error");
});

module.exports = app;

```

controller/api/proposal.js

```

var express = require("express");
var mongoose = require("mongoose");

var ProposalModel = require("../models/proposal");
var UserModel = require("../models/user");
var ErrorHandler = require("../helpers/error");
const { resource } = require("../app");

Promise.all([
  getUsersFromEmails(req.body.supervisors),
  getUsersFromEmails(req.body.members),
  getUsersFromEmails([req.body.leader]),
])
  .then(([supervisors, members, [leader]]) => {
    if (supervisors.length !== req.body.supervisors.length) {
      res.status(400).send({
        message: "One or more invalid emails for supervisors",
      });
    } else if (members.length !== req.body.members.length) {
      res.status(400).send({
        message: "One or more invalid emails for members",
      });
    } else if (leader === null) {
      res.status(400).send({
        message: "Invalid leader email",
      });
    } else {
      req.body.supervisors = supervisors;
      req.body.members = members;
      req.body.leader = leader;
      delete req.body.pdf_document;
      const proposal = new ProposalModel(req.body);
      proposal
        .save()
        .then((resource) => {
          // resource.pdf_document = resource.document_b64;
          res.status(201).send({
            id: resource._id,
            url: resource.url,
            message: "Proposal created",
          });
        });
    }
  });

```



```

    })
    .catch((error) => {
        res.status(400).send(ErrorHelper.construct_json_response(error));
    });
}
})
.catch((error) => {
    res.status(400).send(ErrorHelper.construct_json_response(error));
});

/*
function getUsersFromEmails(emailArr) {
    destnArr = [];
    emailArr.forEach(
        function (email, idx) {
            destnArr.push(
                UserModel
                    .onlyExisting()
                    .getByEmail(email)
            );
        }
    );
    return Promise.all(destnArr);
}

const proposal = new ProposalModel(req.body);
proposal
    .save()
    .then((resource => {
        res.status(201).send({
            id: resource._id,
            message: "Proposal created"
        })
    })))
    .catch((error) => {
        res.status(400).send(
            ErrorHelper.construct_json_response(error)
        );
    });
*/
}

function getAll(req, res, next) {
    ProposalModel.onlyExisting()
        .then((resources) => {
            // resources.forEach(function (rsrc, idx) {
            //     let go;
            //     // rsrc.pdf_document = rsrc.document_base64;
            // });
            // console.log(resources);

            resources = resources.map((obj) => {
                let temp = obj.toJSON();
                delete temp["pdf_document"];
                return temp;
            });
        });
}

```

```

    }); // README
    console.log(Object.keys(resources[0]));

    resources.map((obj) => {
      delete obj.pdf_document;
    });

    res.status(200).send(resources);
  })
  .catch((error) => {
    res.status(400).send(ErrorHelper.construct_json_response(error));
  });
}

function getByUser(user_id, req, res, next) {
  function getProposalsForRole(role_field, user_id) {
    return ProposalModel.onlyExisting().find({
      [role_field]: user_id,
    });
  }

  if (mongoose.Types.ObjectId.isValid(user_id)) {
    user_id = mongoose.Types.ObjectId(user_id);
    Promise.all([
      getProposalsForRole("supervisors", user_id),
      getProposalsForRole("members", user_id),
      getProposalsForRole("leader", user_id),
    ])
    .then(([supervisor_proposals, member_proposals, leader_proposals]) => {
      res.status(200).send({
        as_supervisor: supervisor_proposals,
        as_member: member_proposals,
        as_leader: leader_proposals,
      });
    })
    .catch((error) => {
      res.status(400).send(ErrorHelper.construct_json_response(error));
    });
  } else {
    res.status(404).send({
      message: "Proposal not found",
    });
  }
}

function getById(id, req, res, next) {
  if (mongoose.Types.ObjectId.isValid(id)) {
    ProposalModel.onlyExisting()
      .getById(id)
      .populate("leader")
      .populate("members")
      .populate("supervisors")
      .then((resource) => {
        if (resource.length == 0) {
          throw {

```

```

        error: "Proposal not found",
        message: "Could not find a proposal for that ID",
        code: 801,
    };
    }
    resource = resource[0];
    resource.pdf_document = resource.document_base64;
    res.status(200).send(resource);
  })
  .catch((error) => {
    res.status(400).send(ErrorHelper.construct_json_response(error));
  });
} else {
  res.status(404).send(
    ErrorHelper.construct_json_response({
      error: "Proposal not found",
      message: "Could not find a proposal for that ID",
      code: 801,
    })
  );
}
}
}

function rejectProposal(req, res, next) {
  // check if proposal is pending status
  if (mongoose.Types.ObjectId.isValid(req.body.id)) {
    proposal_id = mongoose.Types.ObjectId(req.body.id);
    new Promise((resolve, reject) => {
      ProposalModel.onlyExisting()
        .findById(proposal_id)
        .then(([proposal]) => {
          if (!proposal.isAwaitingDecision()) {
            reject({
              name: "Proposal not awaiting decision",
              message:
                "Proposal not awaiting decision. It has already been approved or
rejected",
              code: 951,
            });
          }
        })
      }
    } else {
      // update status of the proposal
      ProposalModel.updateOne(
        {
          _id: proposal_id,
        },
        {
          rejected_on: Date.now(),
          rejection_remarks: req.body.remarks,
        }
      ).then((result) => {
        resolve(result);
      });
    }
  });
}
});
})

```

```

        .then((updation_data) => {
        if (!updation_data.acknowledged) {
        throw {
            error: "Proposal could not be updated",
            message: "Error occurred when updating proposal status. Try later",
            code: 952,
        };
        }
        res.status(201).send({
        proposal_id: proposal_id,
        message: "Proposal marked as rejected",
        });
        })
        .catch((error) => {
        console.log(error);
        res.status(400).send(ErrorHelper.construct_json_response(error));
        });
    } else {
        res.status(404).send({
        message: "Proposal not found",
        });
    }
}

exports.create = create;
exports.getAll = getAll;
exports.getById = getById;
exports.getByUser = getByUser;
exports.rejectProposal = rejectProposal;

```

model/proposal.js

```

var mongoose = require("mongoose");
var Promise = require("promise");

var Schema = mongoose.Schema;

// Validations to consider
// - rejected_on and approved_on should NOT BOTH be Non-NULL!
// - setup a trigger/autorun to make on or the other null, if the other is
forcefully non-nulled
// - (add on...)
var ProposalSchema = new Schema(
{
    title: {
        type: String,
        required: true,
        maxLength: 100,
    },
    description: {
        type: String,
    },
    domains: {

```

```

    type: [
      {
        type: String,
        enum: [
          "machine_learning",
          "web_development",
          "iot",
          "computer_vision",
          "nlp",
          "cybersecurity",
        ],
      },
    ],
    required: true,
  },
  supervisors: {
    type: [
      {
        type: Schema.Types.ObjectId,
        ref: "user",
      },
    ],
    validate: {
      validator: function (given_proj) {
        var current_proposal = this;
        return new Promise(function (resolve, reject) {
          current_proposal.populate("supervisors").then((populated_data) => {
            var allFaculties = true;
            var supervisorsList = populated_data.supervisors;
            supervisorsList.forEach((supervisor) => {
              console.log(supervisor.role);
              if (supervisor.role !== "faculty") {
                allFaculties = false;
              }
            });
            return resolve(allFaculties);
          });
        });
      },
      message: (props) => `All supervisors must be faculty`,
    },
    required: true,
  },
  // the user type of leader determines if project is Student project or
  Faculty project
  leader: {
    type: Schema.Types.ObjectId,
    ref: "user",
    required: true,
  },
  members: {
    type: [
      {
        type: Schema.Types.ObjectId,
        ref: "user",

```

```

    },
  ],
  validate: {
    validator: function (given_proj) {
      var current_proposal = this;
      return new Promise(function (resolve, reject) {
        var memberList = current_proposal.members;
        var leaderEmail = current_proposal.leader.email;
        var existed = false;
        console.log(leaderEmail);
        memberList.forEach(function (member) {
          if (member.email == leaderEmail) {
            existed = true;
          }
        });

        if (existed) {
          return resolve(false);
        } else {
          return resolve(true);
        }
      });
    },
  },
  message: (props) =>
    `Duplicate entry for leader found in members list. Mention only once`,
  },
  required: true,
},
funding_type: {
  type: String,
  enum: ["internal", "external"],
  required: true,
},

funding_agency: {
  type: String,
  required: true,
},
// Accepts file sizes upto 16MB -- Indicate limit as 8MB on Frontend
// Transit as Base64 string on JSON
pdf_document: {
  type: Buffer,
  required: false,
},
budget: {
  type: Number,
  required: true,
},
approved_on: {
  type: Date,
  default: null,
},
rejected_on: {
  type: Date,
  default: null,
}

```

```

    validate: {
      validator: function (given_proj) {
        var current_proposal = this;
        return new Promise(function (resolve, reject) {
          var result = true;
          if (
            current_proposal.rejected_on != null &&
            current_proposal.approved_on != null
          ) {
            result = false;
          }
          // console.log(result)
          return resolve(result);
        });
      },
      message: (props) =>
        `Either rejected or approved date should be mentioned but not both!`,
    },
    message: (props) =>
      `Either rejected or approved date should be mentioned but not both!`,
    },
    rejection_remarks: {
      type: String,
      default: null,
    },
    deleted_at: {
      type: Date,
      default: null,
    },
  },
  {
    timestamps: {
      created_at: "created_at",
      modified_at: "modified_at",
    },
  }
});

// Chain <ModelName>.onlyExisting before any query to list only "non-deleted"
records
ProposalSchema.statics.onlyExisting = function () {
  return this.find().onlyExisting();
};

ProposalSchema.query.onlyExisting = function () {
  return this.find({
    deleted_at: null,
  });
};

// --

ProposalSchema.statics.Id = function (id) {
  return this.find().getById();
};

```

```

ProposalSchema.query.getById = function (id) {
  return this.find({
    _id: id,
  });
};

// --

ProposalSchema.methods.isApproved = function () {
  return this.rejected_on != null && this.approved_on == null;
};

ProposalSchema.methods.isRejected = function () {
  return this.rejected_on == null && this.approved_on != null;
};

ProposalSchema.methods.isAwaitingDecision = function () {
  return this.rejected_on == null && this.approved_on == null;
};

// --

// virtual for executive members
ProposalSchema.virtual("executive_members").get(function () {
  var all_members = [];
  all_members = all_members.concat([leader], members);
  return all_members;
});

// virtual for executive members
ProposalSchema.virtual("all_members").get(function () {
  var all_members = [];
  all_members = all_members.concat(executive_members, supervisors);
  return all_members;
});

ProposalSchema.virtual("document_base64").get(function () {
  console.log(this.pdf_document);
  if (this.pdf_document) {
    return Buffer.from(this.pdf_document).toString("base64");
  } else {
    return null;
  }
});

// virtual for user URL
ProposalSchema.virtual("url").get(function () {
  return "/api/proposal/" + this._id;
});

//Export model
module.exports = mongoose.model("proposal", ProposalSchema);

```


controller/api/project.js

```
var express = require("express");
var mongoose = require("mongoose");

var ProjectModel = require("../models/project");
var ProposalModel = require("../models/proposal");

var ErrorHandler = require("../helpers/error");
var Utils = require("../helpers/utils");

function create(req, res, next) {
  if (mongoose.Types.ObjectId.isValid(req.body.proposal)) {
    proposal_id = mongoose.Types.ObjectId(req.body.proposal);

    // check if corresponding proposal is pending status - update only if so
    new Promise((resolve, reject) => {
      ProposalModel.onlyExisting()
        .getById(proposal_id)
        .then((proposal) => {

          if (proposal.approved_on != null || proposal.rejected_on != null) {
            reject({
              name: "Proposal not awaiting decision",
              message:
                "Proposal not awaiting decision. It has already been approved or
rejected",
              code: 951,
            });
          } else {

            // update status of the proposal
            ProposalModel.updateOne(
              {
                _id: proposal_id,
              },
              {
                approved_on: Date.now(),
              }
            ).then((result) => {
              resolve(result);
            });
          }
        })
      ).then((updatation_meta) => {
        // create a new project
        if (!updatation_meta.acknowledged) {
          throw {
            name: "Proposal status could not be updated",
            message: "Error occurred when updating proposal status. Try later",
          };
        }
      });
    });
  }
}
```

```

        code: 952,
    };
}
const project = new ProjectModel(req.body);
project.save().then((resource) => {
    res.status(201).send({
        id: resource._id,
        url: resource._url,
        message: "Project created. Proposal updated",
    });
});
})
.catch((error) => {
    res.status(400).send(ErrorHelper.construct_json_response(error));
});
} else {
    res.status(404).send({
        message: "Proposal not found",
    });
}
}

```

```

function getAll(req, res, next) {
    ProjectModel.onlyExisting()
        .then((resources) => {
            Promise.all(
                resources.map((rsrc) => {
                    return rsrc.populate("proposal");
                })
            ).then((resources) => {
                res.status(200).send(resources);
            });
        })
        .catch((error) => {
            res.status(400).send(ErrorHelper.construct_json_response(error));
        });
}

```

```

function getByUser(user_id, req, res, next) {
    // Ashamed of the inelegance in this function!!
    function getProjectsForRole(role_field, user_id) {
        return ProjectModel.onlyExisting().then((with_proposal) => {
            return with_proposal.filter(function (project) {
                return project.proposal != null;
            });
        });
    }
}

```

```

if (mongoose.Types.ObjectId.isValid(user_id)) {
    user_id = mongoose.Types.ObjectId(user_id);
    Promise.all([
        getProjectsForRole("supervisors", user_id),
        getProjectsForRole("members", user_id),
        getProjectsForRole("leader", user_id),
    ])
}

```

```

        .then(([supervisor_projects, member_projects, leader_projects]) => {
            res.status(200).send({
                as_supervisor: supervisor_projects,
                as_member: member_projects,
                as_leader: leader_projects,
            });
        })
        .catch((error) => {
            res.status(400).send(ErrorHelper.construct_json_response(error));
        });
    } else {
        res.status(404).send({
            message: "User not found",
        });
    }
}

```

// Proposal field is explicitly populated

```

function getById(id, req, res, next) {
    if (mongoose.Types.ObjectId.isValid(id)) {
        ProjectModel.onlyExisting()
            .getById(id)
            .populate("proposal")
            .then((resource) => {
                res.status(200).send(resource);
            })
            .catch((error) => {
                res.status(400).send(ErrorHelper.construct_json_response(error));
            });
    } else {
        res.status(404).send({
            message: "Project not found",
        });
    }
}

```

```

function updateStatus(req, res, next) {
    // make update subdocument first
    if (mongoose.Types.ObjectId.isValid(req.body.id)) {
        project_id = mongoose.Types.ObjectId(req.body.id);
        outcome_obj = {};
        ["title", "description"].map((key) => {
            if (req.body.hasOwnProperty(key)) {
                Object.assign(outcome_obj, { [key]: req.body[key] });
            }
        });
    }
}

```

// check if last update was at least 2 days ago

```

new Promise((resolve, reject) => {
    ProjectModel.getById(project_id).then(([project]) => {
        let last_update = project.getMostRecentUpdate();
        if (
            last_update != null &&
            Utils.timeDelta_days(Date.now(), last_update.createdAt) < 0
        ) {

```

```

        reject({
          name: "Status update too frequent",
          message: "Previous status update was made less than 2 days ago",
          meta_info: {
            previous_update: last_update.createdAt,
          },
          code: 961,
        });
      }
      resolve(project);
    });
  })
  .then((project) => {
    // make update
    ProjectModel.onlyExisting()
      .updateOne(
        {
          _id: project_id,
        },
        {
          $addToSet: {
            status_updates: outcome_obj,
          },
        }
      )
      .then((updation_meta) => {
        if (!updation_meta.acknowledged) {
          throw {
            name: "Project update could not be written",
            message: "Error occurred when updating project. Try later",
            code: 952,
          };
        }
        res.status(204).send({
          id: project_id,
          message: "Project status updated",
          update_title: req.body.title,
        });
      });
  })
  .catch((error) => {
    res.status(400).send(ErrorHelper.construct_json_response(error));
  });
} else {
  res.status(404).send({
    message: "Project not found",
  });
}
}

function updateOutcome(req, res, next) {
  // make updation subdocument first
  if (mongoose.Types.ObjectId.isValid(req.body.id)) {
    project_id = mongoose.Types.ObjectId(req.body.id);
    outcome_obj = {};
  }
}

```

```

["title", "description", "kind", "reference"].map((key) => {
  if (req.body.hasOwnProperty(key)) {
    Object.assign(outcome_obj, { [key]: req.body[key] });
  }
});

// check if last outcome update was at least 2 days ago
new Promise((resolve, reject) => {
  ProjectModel.getById(project_id).then([project]) => {
    let last_outcome = project.getMostRecentOutcome();
    if (
      last_outcome != null &&
      Utils.timeDelta_days(Date.now(), last_outcome.createdAt) < 2
    ) {
      reject({
        name: "Outcome update too frequent",
        message: "Previous outcome update was made less than 2 days ago",
        meta_info: {
          previous_outcome: last_outcome.createdAt,
        },
        code: 961,
      });
    }
    resolve(project);
  });
})
.then((project) => {
  // make update
  ProjectModel.onlyExisting()
    .updateOne(
      {
        _id: project_id,
      },
      {
        $addToSet: {
          outcomes: outcome_obj,
        },
      },
    )
    .then((updation_meta) => {
      if (!updation_meta.acknowledged) {
        throw {
          name: "Project outcome update could not be written",
          message: "Error occurred when updating project. Try later",
          code: 952,
        };
      }
    })
    .then(() => {
      res.status(204).send({
        id: project_id,
        message: "Project outcomes updated",
        update_title: req.body.title,
      });
    });
})
.catch((error) => {

```

```

        res.status(400).send(ErrorHelper.construct_json_response(error));
    });
} else {
    res.status(404).send({
        message: "Project not found",
    });
}
}
}

```

```

exports.create = create;
exports.getById = getById;
exports.getAll = getAll;
exports.getByUser = getByUser;
exports.updateStatus = updateStatus;
exports.updateOutcome = updateOutcome;

```

models/project.js

```

/* A `proposal` becomes a `project` when it is approved */

```

```

var mongoose = require("mongoose");
var Promise = require("promise");

```

```

var Schema = mongoose.Schema;

```

```

// Child schema

```

```

var UpdatesSchema = new Schema(
{
    title: {
        type: String,
        required: true,
    },
    description: {
        type: String,
        required: false,
    },
},
{
    timestamps: {
        created_at: "created_at",
        modified_at: "modified_at",
    },
}
);

```

```

// Child schema

```

```

var OutcomesSchema = new Schema(
{
    title: {
        type: String,
        required: true,
    },
    description: {
        type: String,
    },
}
);

```

```

        required: false,
    },
    kind: {
        type: String,
        enum: ["research_paper", "patent", "incubation", "scaled", "other"],
        required: true,
    },
    // URL to publication, patent, etc.
    reference: {
        type: String,
        required: true,
    },
},
{
    timestamps: {
        created_at: "created_at",
        modified_at: "modified_at",
    },
}
);

```

```

// Valiations to consider:
// - Allow resources ONLY for Internal Projects
// - (add on...)

```

```

var ProjectSchema = new Schema(
{
    proposal: {
        type: Schema.Types.ObjectId,
        ref: "proposal",
        required: true,
    },
    status_updates: {
        type: [
            {
                type: UpdatesSchema,
            },
        ],
        default: () => [],
    },
    outcomes: {
        type: [
            {
                type: OutcomesSchema,
            },
        ],
        default: () => [],
    },
    approved_budget: {
        type: Number,
        required: true,
    },
    approved_duration: {
        type: Number, // in MONTHS
        required: true,
    },
},

```

```

    completed_on: {
      type: Date,
      default: null,
    },
  },
  {
    timestamps: {
      created_at: "created_at",
      modified_at: "modified_at",
    },
    toJSON: { virtuals: true },
    toObject: { virtuals: true },
  }
);

//--

ProjectSchema.statics.onlyExisting = function () {
  return this.find().onlyExisting();
};

ProjectSchema.query.onlyExisting = function () {
  return this.find({
    deleted_at: null,
  });
};

//--

ProjectSchema.statics.getById = function (id) {
  return this.find().getById(id);
};

ProjectSchema.query.getById = function (id) {
  return this.find({
    _id: id,
  });
};

//--

ProjectSchema.methods.getMostRecentUpdate = function () {
  if (this.status_updates.length > 0) {
    return this.status_updates.reduce((prev_update, current_update) => {
      return prev_update.created_at >= current_update.created_at
        ? prev_update
        : current_update;
    });
  } else {
    return null;
  }
};

//--

```



```

ProjectSchema.methods.getMostRecentOutcome = function () {
  if (this.outcomes.length > 0) {
    return this.outcomes.reduce((prev_outcome, current_outcome) => {
      return prev_outcome.created_at >= current_outcome.created_at
        ? prev_outcome
        : current_outcome;
    });
  } else {
    return null;
  }
};

//--

// virtual for URL
ProjectSchema.virtual("url").get(function () {
  return "/api/project/" + this._id;
});

```

```

module.exports = mongoose.model("project", ProjectSchema);

```

controllers/api/resource_assignment.js

```

var express = require("express");

var mongoose = require("mongoose");

var ResourceAssignmentModel = require("../models/resource_assignment");
var ResourceModel = require("../models/resource");

var ResourceHelper = require("../helpers/resource");
var ErrorHandler = require("../helpers/error");

var ResourceFilters = require("../helpers/filters/resource");
var Utils = require("../helpers/utls");

function create(req, res, next) {
  const rsrc = new ResourceAssignmentModel(req.body);
  rsrc
    .save()
    .then((resource) => {
      res.status(201).send({
        id: resource._id,
        url: resource.url,
        message: "Resource assignment made",
      });
    })
    .catch((error) => {
      res.status(400).send(ErrorHelper.construct_json_response(error));
    });
}

function getAll(req, res, next) {
  ResourceAssignmentModel.onlyExisting()
    .then((resources) => {

```

```

        res.status(200).send(resources);
    })
    .catch((error) => {
        res.status(400).send(ErrorHelper.construct_json_response(error));
    });
}

```

```

function getById(id, req, res, next) {
    if (mongoose.Types.ObjectId.isValid(id)) {
        ResourceAssignmentModel.onlyExisting()
            .getById(id)
            .then((resource) => {
                res.status(200).send(resource);
            })
            .catch((error) => {
                res.status(400).send(ErrorHelper.construct_json_response(error));
            });
    } else {
        res.status(404).send({
            message: "Resource assignment record not found",
        });
    }
}

```

```

function getByProject(id, req, res, next) {
    if (mongoose.Types.ObjectId.isValid(id)) {
        project_id = mongoose.Types.ObjectId(id);
        ResourceAssignmentModel.aggregate(
            [
                {
                    $match: {
                        deleted_on: null,
                        assigned_to: project_id,
                    },
                },
                {
                    $lookup: {
                        from: "resources",
                        localField: "resource",
                        foreignField: "_id",
                        as: "resource_data",
                    },
                },
                {
                    $group: {
                        _id: "$resource_data.resource_group",
                        data: {
                            $addToSet: "$$ROOT",
                        },
                        count: {
                            $sum: 1,
                        },
                    },
                },
                {
                    $project: {
                        data: "$data",
                        count: "$count",
                    },
                },
            ],
            (err, result) => {
                if (err) {
                    res.status(500).send({
                        message: "Internal server error",
                    });
                } else {
                    res.status(200).send(result);
                }
            }
        );
    }
}

```

```

    $replaceRoot: {
      newRoot: {
        $mergeObjects: [
          {
            assigned_qty: "$count",
          },
          { resource_records: "$data" },
        ],
      },
    },
  ],
  (error, results) => {
    if (error) {
      throw error;
    }
    return results;
  }
)
.then((grouped_assigns) => {
  console.log(grouped_assigns);
  res.status(200).send(grouped_assigns);
})
.catch((error) => {
  res.status(400).send(ErrorHelper.construct_json_response(error));
});
} else {
  res.staus(404).send(
    ErrorHelper.construct_json_response({
      error: "Project not found",
      message: "Could not find a project for that ID",
      code: 801,
    })
  );
}
}

function assignResourcesToProject(req, res, next) {
  // verify id values
  if (mongoose.Types.ObjectId.isValid(req.body.rsrc_mgr_id)) {
    rsrc_mgr_id = mongoose.Types.ObjectId(req.body.rsrc_mgr_id);
  } else {
    return void res.staus(404).send(
      ErrorHelper.construct_json_response({
        error: "Resource Manager not found",
        message: "Could not find a user for that ID",
        code: 801,
      })
    );
  }

  if (mongoose.Types.ObjectId.isValid(req.body.project_id)) {
    project_id = mongoose.Types.ObjectId(req.body.project_id);
  } else {
    return void res.staus(404).send(

```

```

        ErrorHandler.construct_json_response({
            error: "Project not found",
            message: "Could not find a project for that ID",
            code: 801,
        })
    );
}

if (mongoose.Types.ObjectId.isValid(req.body.rsrc_grp_id)) {
    rsrc_grp_id = mongoose.Types.ObjectId(req.body.rsrc_grp_id);
} else {
    return void res.status(404).send(
        ErrorHandler.construct_json_response({
            error: "Resource group not found",
            message: "Could not find a resource group for that ID",
            code: 801,
        })
    );
}

// get group's resources
ResourceModel.onlyExisting()
    .find({
        resource_group: rsrc_grp_id,
    })
    .then((group_resources) => {
        // get current assigned quantity
        ResourceHelper.get_resource_count_for_project(
            rsrc_grp_id,
            project_id
        ).then((count_obj) => {
            console.log(count_obj);
            if (count_obj == undefined) {
                count_obj = { count: 0 };
            }
            var qty_delta = req.body.qty - count_obj.count;

            if (qty_delta < 0) {
                // unallocate resources
                var qty_delta = -1 * qty_delta;
                var qty_to_modify =
                    qty_delta > count_obj.count ? count_obj.count : qty_delta;

                ResourceAssignmentModel.onlyExisting()
                    .find({
                        assigned_to: project_id,
                    })
                    .populate({
                        path: "resource",
                        match: {
                            resource_group: rsrc_grp_id,
                        },
                    })
                    .then((project_assigns) => {

```

```

// remove the `null` populations
project_grp_assigns = project_assigns.filter((rsrc) => {
  return Boolean(rsrc.resource);
});

var to_unassign = project_grp_assigns.slice(0, qty_to_modify);
Promise.all(
  to_unassign.map((rsrc_assign) => {
    return ResourceAssignmentModel.deleteOne({
      _id: rsrc_assign._id,
    });
  })
)
.then((deletion_status) => {
  deletion_status.map((status) => {
    if (status.deletedCount != 1) {
      throw {
        name: "Assignment could not be deleted",
        message:
          "Error occurred when deleting resource assignment. Try
later",
        code: 952,
      };
    }
  });
  res.status(201).send({
    total_qty: project_grp_assigns.length - qty_to_modify,
    deallocated_qty: qty_to_modify,
    project_id: project_id,
    resource_group_id: rsrc_grp_id,
    message: "Resource deallocations made",
  });
})
.catch((error) => {
  res
    .status(400)
    .send(ErrorHelper.construct_json_response(error));
});
});
} else if (qty_delta == 0) {
  res.status(200).send({
    total_qty: count_obj.count,
    assigned_qty: 0,
    message: "No resource assignments made",
  });
} else {
  // get resources that can be assigned
  Utils.applyAsyncFilters(group_resources, [
    ResourceFilters.not_assigned,
  ]).then((resources) => {
    var qty_to_modify =
      qty_delta > resources.length ? resources.length : qty_delta;

    // make assignments
    var to_assign = resources.slice(0, qty_to_modify);

```

```

        console.log("to_assign", qty_to_modify, qty_delta);
        Promise.all(
            to_assign.map((rsrc) => {
                return new ResourceAssignmentModel({
                    resource: rsrc._id,
                    assigned_to: project_id,
                    assigned_by: rsrc_mgr_id,
                }).save();
            })
        )
        .then((_) => {
            console.log(_);
            res.status(201).send({
                total_qty: qty_to_modify + count_obj.count,
                assigned_qty: qty_to_modify,
                project_id: project_id,
                resource_group_id: rsrc_grp_id,
                message: "Resource assignments made",
            });
        })
        .catch((error) => {
            res
                .status(400)
                .send(ErrorHelper.construct_json_response(error));
        });
    });
}

exports.create = create;
exports.getById = getById;
exports.getAll = getAll;
exports.getByProject = getByProject;
exports.assignResourcesToProject = assignResourcesToProject;

```

Frontend

FullProjectCard.js

```

import React, { useState } from "react";
import styles from "../components/styles/FullProjectCard.module.css";
import userStyles from "../components/styles/UserCard.module.css";
import useCollapse from "react-collapsed";
import Usercard from "../components/UserCard";
import ResourceAllocationcard from "../components/ResourceAllocationCard";
import Resourcecard from "../components/ResourceCard";
import useSWR from "swr";
import Popup from "reactjs-popup";
import { Document, Page } from "react-pdf";
import loadingGif from "../src/assets/loading.gif";

```

```

function Usercard_byid(props) {
  const fetcher = (url) => fetch(url).then((res) => res.json());
  const { data, error } = useSWR(
    "http://localhost:3000/api/user/" + props.props,
    fetcher
  );

  if (error) return <div>failed to load</div>;
  if (!data)
    return (
      <div
        style={{
          position: "relative",
          width: "175px",
          margin: "auto",
          transform: "translateY(110%)" /* or try 50% */,
        }}
      >
        <div>
          <img
            src={loadingGif.src}
            alt="wait until the page loads"
            height="100%"
          />
          <center>loading...</center>
        </div>
      </div>
    );

  return <Usercard {...data[0]} />;
}
function Idlist_usercard(props) {
  return props.props.map((obj) => {
    return <Usercard_byid props={obj} />;
  });
}
function Members(props) {
  const config = {
    duration: 200,
  };
  const { getCollapseProps, getToggleProps, isExpanded } = useCollapse(config);

  return (
    <div className="collapsible">
      <div className="header" {...getToggleProps()}>
        {isExpanded ? (
          <div className={styles.collapse}>Members</div>
        ) : (
          <div className={styles.expand}>Members</div>
        )}
      </div>
      <div {...getCollapseProps()}>
        <div style={{ display: "flex", "flex-direction": "row" }}>

```

```

        <Idlist_usercard props={props.props} />
      </div>
    </div>
  </div>
);
}
function Leader(props) {
  const config = {
    duration: 200,
  };
  const { getCollapseProps, getToggleProps, isExpanded } = useCollapse(config);

  return (
    <div className="collapsible">
      <div className="header" {...getToggleProps()}>
        {isExpanded ? (
          <div className={styles.collapse}> Principal Investigator</div>
        ) : (
          <div className={styles.expand}>Principal Investigator</div>
        )}
      </div>
      <div {...getCollapseProps()}>
        <div style={{ display: "flex", "flex-direction": "row" }}>
          <Idlist_usercard props={[[props.props]]} />
        </div>
      </div>
    </div>
  );
}

```

```

function Supervisors(props) {
  const config = {
    duration: 200,
  };
  const { getCollapseProps, getToggleProps, isExpanded } = useCollapse(config);

  return (
    <div className="collapsible">
      <div className="header" {...getToggleProps()}>
        <h1>
          {isExpanded ? (
            <div className={styles.collapse}>Supervisors</div>
          ) : (
            <div className={styles.expand}>Supervisors</div>
          )}
        </h1>
      </div>
      <div {...getCollapseProps()}>
        <div style={{ display: "flex", "flex-direction": "row" }}>
          <Idlist_usercard props={[[props.props]]} />
        </div>
      </div>
    </div>
  );
}

```



```

function Resource_temp(props) {

  const fetcher = (url) => fetch(url).then((res) => res.json());
  let temp = props.props.resource_records[0].resource_data[0].resource_group;

  const { data, error } = useSWR(
    "http://localhost:3000/api/resource-group/" + temp, // gives project id
    fetcher
  );

  if (error) return <div>failed to load</div>;
  if (!data) return <div>loading...</div>;

  let name = data.name;
  data[0].avl_qty = props.props.assigned_qty;

  if (String(name).toLowerCase().includes(props.props.query.toLowerCase()))
    return <Resourcecard {...data[0]} />;
}

function Resource(props) {
  const types = [{ key: "name", name: "name" }];
  const [query, setQuery] = useState("");
  const [type, setType] = useState(types[0]["key"]);
  const [counter, setCounter] = useState(0);
  const fetcher = (url) => fetch(url).then((res) => res.json());
  const { data, error } = useSWR(
    "http://localhost:3000/api/resource-assignment/project/" + props.props._id, //
    gives project id
    fetcher
  );
  const { getCollapseProps, getToggleProps, isExpanded } = useCollapse(config);
  if (error) return <div>failed to load</div>;
  if (!data) return <div>loading...</div>;

  const config = {
    duration: 200,
  };

  return (
    <div className="collapsible">
      <div className="header" {...getToggleProps()}>
        <h1>
          {isExpanded ? (
            <div className={styles.collapse}>Resource assigned</div>
          ) : (
            <div className={styles.expand}>Resource assigned(collaped)</div>
          )}
        </h1>
      </div>
      <div {...getCollapseProps()}>
        <div style={{ display: "flex", "flex-direction": "row" }}>
          <div style={{ margin: "auto" }}>
            <input

```

```

        onChange={(event) => setQuery(event.target.value)}
        placeholder="Enter Post Title"
      />
      <select
        id="search_key"
        onChange={(event) => setType(event.target.value)}
      >
        {types.map((obj) => {
          return <option value={obj.key}> {obj.name}</option>;
        })}
      </select>
    </div>
    {data.map((obj) => {
      obj.query = query;
      return <Resource_temp props={obj} />;
    })}
  </div>
</div>
);
}

function ResourceAllocation(props) {
  const types = [
    { key: "name", name: "name" },
    { key: "kind", name: "kind" },
  ];
  const [query, setQuery] = useState("");
  const [type, setType] = useState(types[0]["key"]);
  const [counter, setCounter] = useState(0);
  const fetcher = (url) => fetch(url).then((res) => res.json());
  const { data, error } = useSWR(
    "http://localhost:3000/api/resource-group",
    fetcher
  );
  const { getCollapseProps, getToggleProps, isExpanded } = useCollapse(config);
  if (error) return <div>failed to load</div>;
  if (!data) return <div>loading...</div>;

  const config = {
    duration: 200,
  };

  return (
    <div className="collapsible">
      <div className="header" {...getToggleProps()}>
        <h1>
          {isExpanded ? (
            <div className={styles.collapse}>Resource assignable</div>
          ) : (
            <div className={styles.expand}>Resource assignable(collaped)</div>
          )}
        </h1>
      </div>
      <div {...getCollapseProps()}>

```

```

<div style={{ display: "flex", "flex-direction": "row" }}>
  <div style={{ margin: "auto" }}>
    <input
      onChange={(event) => setQuery(event.target.value)}
      placeholder="Enter Post Title"
    />
    <select
      id="search_key"
      onChange={(event) => setType(event.target.value)}
    >
      {types.map((obj) => {
        return <option value={obj.key}> {obj.name}</option>;
      })}
    </select>
  </div>
  {data.map((obj) => {
    obj.project_id = props.props._id;
    let temp = obj[type];
    if (String(temp).toLowerCase().includes(query.toLowerCase()))
      return <ResourceAllocationcard {...obj} />;
  })}
</div>
</div>
);
}
function Outcomecard(props) {

  return (
    <h1>
      {props.props.title} {props.props.description}
    </h1>
  );
}
function Statuscard(props) {
  return (
    <div>
      <li>
        <h2>{props.props.title}</h2>
      </li>
      {props.props.description}
    </div>
  );
}

function ViewStatus(props) {
  const config = {
    duration: 200,
  };
  const [title, setTitle] = useState();
  const [desc, setDesc] = useState();
  const handleSubmit = (e) => {
    e.preventDefault();
    let parent = e.target.parentElement;

```

```

parent.children[1].value = "";
parent.children[2].innerHTML = "";
setTitle("");
setDesc("");
const axios = require("axios");

axios
  .patch("http://localhost:3000/api/project/update-status ", {
    id: props.props._id,
    title: title,
    description: desc,
  })
  .then(function (response) {

  })
  .catch((error) => {

  });
};
const { getCollapseProps, getToggleProps, isExpanded } = useCollapse(config);

return (
  <div className="collapsible">
    <div className="header" {...getToggleProps()}>
      <h1>
        {isExpanded ? (
          <div className={styles.collapse}>Status</div>
        ) : (
          <div className={styles.expand}>Status</div>
        )}
      </h1>
    </div>
    <div {...getCollapseProps()}>
      <div>
        <label>Title:</label>
        <input
          id="title"
          onChange={(event) => setTitle(event.target.value)}
          type="text"
        ></input>
        <label>Description:</label>
        <textarea
          onChange={(event) => setDesc(event.target.value)}
          id="desc"
        ></textarea>
        <button onClick={handleSubmit}>Submit</button>
      </div>
      <div style={{ display: "flex", "flex-direction": "row" }}>
        <ul>
          {props.props.status_updates.map((obj) => {
            return <Statuscard props={obj} />;
          })}
        </ul>
      </div>
    </div>
  )

```

```

        </div>
    </div>
);
}
function ViewOutcomes(props) {
    const config = {
        duration: 200,
    };
    const { getCollapseProps, getToggleProps, isExpanded } = useCollapse(config);

    return (
        <div className="collapsible">
            <div className="header" {...getToggleProps()}>
                <h1>
                    {isExpanded ? (
                        <div className={styles.collapse}>Outcomes</div>
                    ) : (
                        <div className={styles.expand}>Outcomes</div>
                    )}
                </h1>
            </div>
            <div {...getCollapseProps()}>
                <div style={{ display: "flex", "flex-direction": "row" }}>
                    {props.props.map((obj) => {
                        return <Outcomecard props={obj} />;
                    })}
                </div>
            </div>
        </div>
    );
}

```

```

export default function FullProject(props) {

```

```

    // convert array to base64 string (given pdf_document data)

```

```

function arrayBufferToBase64(buffer) {
    var binary = "";
    var bytes = new Uint8Array(buffer);
    var len = bytes.byteLength;
    for (var i = 0; i < len; i++) {
        binary += String.fromCharCode(bytes[i]);
    }
    return window.btoa(binary);
}
var res = arrayBufferToBase64(
    props.props.proposal.pdf_document != null
        ? props.props.proposal.pdf_document.data
        : "Nothing"
);

```

```

return (
    <div className={styles.main_container}>
        <h1 className={styles.title}>{props.props.proposal.title}</h1>
        <div className={styles.domain}>

```

```

    <span className={styles.domain_tag}>Domain:</span>{" "}
    {props.props.proposal.domains.map((obj) => {
      return <span className={styles.domain_sub}>{obj}</span>;
    })}
  </div>
  <div className={styles.table_container}>
    <div className={styles.sub_table}>
      Budget requested: Rs
      <span className={styles.budget}>{props.props.proposal.budget}</span>
    </div>
    <div className={styles.sub_table}>
      Budget approved: Rs
      <span className={styles.budget}>{props.props.approved_budget}</span>
    </div>
    <div className={styles.sub_table}>
      Duration: <span>{props.props.approved_duration} </span>months
    </div>
    <div className={styles.sub_table}>
      Proposal.pdf:
      <a
        style={{ color: "blue", "text-decoration": "underline" }}
        href={`data:application/pdf;base64,${res}`}
        download
      >
        {" "}
        link
      </a>
    </div>
  </div>
  <ViewOutcomes props={props.props.outcomes} />
  <ViewStatus props={props.props} />
  <Leader props={props.props.proposal.leader} />
  <Members props={props.props.proposal.members} />
  <Supervisors props={props.props.proposal.supervisors} />
  <Resource props={props.props} />
  <ResourceAllocation props={props.props} />
</div>
);
}

```

ResourceAllocationCard.js

```

import React, { useState } from "react";

import styles from "../styles/ProjectCard.module.css";
import { useRouter } from "next/router";

import useSWR from "swr";

export default function ResourceCard(props) {
  let [count, setCount] = useState(0);

  console.log("resource card", { props });

```

```

const setCount_modified = (counter) => {
  if (counter < 0 || counter > props.avl_qty) return;
  setCount(counter);
};
const router = useRouter();
const handlesubmit = (e) => {
  e.preventDefault();
  if (count >= 0) {
    const requestOptions = {
      method: "PUT",
      headers: { "Content-Type": "application/json" },
      body: JSON.stringify({
        rsrc_mgr_id: "628f1954f5b6a7772b9885b0",
        project_id: props.project_id,
        rsrc_grp_id: props._id,
        qty: count,
      }),
    };

    console.log(requestOptions);
    console.log(props);
    const axios = require("axios");

    axios
      .put("http://localhost:3000/api/resource-assignment", {
        rsrc_mgr_id: "628f1954f5b6a7772b9885b0",
        project_id: props.project_id,
        rsrc_grp_id: props._id,
        qty: count,
      })
      .then(function (response) {
        console.log("allocated:", response);
      })
      .catch((error) => {
        console.log(error.response);
      });
    console.log("called fetcher");
  }
};
return (
  <div className={styles.wrapper}>
    <div className={styles.card_container}>
      <span className={styles.pro}>{props.name}</span>
      <h3 className={styles.h3}>{props.kind}</h3>
      <h6 className={styles.h6}>SSN Collge of Engineering</h6>
      <p className={` ${styles.email} ${styles.p}`}>{props.description}</p>
      <p className={` ${styles.email} ${styles.p}`}>
        Available quantity: {props.avl_qty}
      </p>
    </div>
    Requesting Quantity :
    <button onClick={() => setCount_modified(count + 1)}>+</button>
    <span>{count}</span>
    <button onClick={() => setCount_modified(count - 1)}>-</button>
    <button onClick={handlesubmit}>Request</button>
  </div>
)

```

```
    </div>
    </div>
    </div>
  );
}
```


Test Cases

Aim

To develop test cases for the Funded Project Tracking and Resource Management system and try to improve the design based on the results of testing

Identification of Testing Scenarios

1. Check if user can be authenticated
2. Check if new user can register successfully
3. Check if new proposals can be submitted without errors
4. Check if proposal status can be modified suitably
5. Check if project can be created upon proposal approval
6. Check if proposal record persists after proposal rejection
7. Check if new project is created with all details
8. Ensure that new project creation references corresponding proposal
9. Check if resource groups can be created
10. Check if resources can be added to corresponding resource groups
11. Check if resources can be assigned to projects
12. Ensure that resources are assigned, respecting the inventory data
13. Ensure valid user rights for each operation
 - a. Resource can be managed only with resource manager rights
 - b. User records can be managed only with admin rights
 - c. Proposal approval status can be updated only with admin rights
 - d. Proposals can be submitted only with user access
 - e. Project updates and outcomes can be registered only by corresponding project members
14. Check if resources assigned to projects can be reclaimed successfully
15. Check if resource fault reporting works to raise tokens
16. Check whether resource fault tokens can be resolved by the admin successfully

TEST CASE ID	TEST SCENARIO	TESTING STEPS	TEST DATA	EXPECTED OUTCOME	ACTUAL OUTCOME	PASS OR FAIL
T01	User authentication	Enter login credentials like user id and password, submit the credentials	userid: jag password: 1	Invalid (incorrect password)	Invalid	Pass
T02	User authentication	Enter login credentials like user id and password, submit the credentials	userid: jag Password: 1234	Valid	Valid	Pass
T03	New user registration	OAuth verification	Google Token: <random-string>	Invalid token	Invalid token	Pass
T04	New user registration	OAuth verification	Google Token: <from-auth-server>	Valid token	Valid token	Pass
T05	Proposal Submission	Navigate to proposal submission, fill details and submit the form	Guide is NOT faculty	Invalid guide details	Invalid guide information	Pass
T06	Proposal Submission	Navigate to proposal submission, fill details and submit the form	category: Seed requested page: available	Proposal submitted	Proposal submitted	Pass

T07	Project leader not duplicated in members	Navigate to proposal submission, fill details submit form	Leader is entered in members	Invalid proposal form — duplicate leader entry	Invalid proposal form	Pass
T08	Project guide in proposal in faculty	Navigate to proposal submission, fill details submit form	Leader is NOT entered in members	Valid proposal form	Valid proposal form	Pass
T09	Resource Assignment for Project	Navigate to resource assignment page, assign resources	Resource assigned for external project	Invalid assignment	Invalid assignment	Pass
T10	Resource Assignment for Project	Navigate to resource assignment page, assign resources	Resource assigned for internal project	Valid assignment	Valid assignment	Pass
T11	Resource Assignment for Project	Navigate to resource assignment page, assign resources	Attempting assignment of more quantity that available	Insufficient resource quantity	Insufficient resource quantity	Pass
T12	Resource Assignment for Project	Navigate to resource assignment page, assign resources	Attempting void quantity assignment	Resources assigned	Resources assigned	Pass

T13	Resource Assignment for Project	Navigate to resource assignment page, assign resources	Attempting assignment of already assigned resources	Resource already assigned	Resource already assigned	Pass
T14	Resource Assignment for Project	Navigate to resource assignment page, assign resources	Attempting assignment of unassigned available assigned resources	Resource assigned	Resource assigned	Pass
T15	Project updates	Navigate to working projects page, submit new update	Record a project update within 2 days of previous update	Too frequent update	Too frequent update	Pass
T16	Project updates	Navigate to working projects page, submit new update	Record a project outcome after 2 days of previous update	Update submitted	Update submitted	Pass
T17	Project outcomes	Navigate to working projects, submit new outcome	Record a project update within 2 days of previous outcome	Too frequent project outcome update	Too frequent project outcome update	Pass
T18	Project outcomes	Navigate to working projects, submit new outcome	Record a project update after 2 days of previous outcome	Outcome update submitted	Outcome update submitted	Pass

T19	Resource Group creation	Navigate to resources inventory, add new resource group	Attempt to create a duplicate resource group	Resource group NOT created	Resource group NOT created	Pass
T20	Resource Group creation	Navigate to resources inventory, add new resource group	Attempt to create a new unique resource group	Resource group created	Resource group created	Pass
T21	User registration	Use the google OAuth service to register user	Attempt to use admin or resource manager role without access key	Registration unauthorized	Registration not permitted	Pass
T22	User registration	Use the google OAuth service to register user	Attempt to use admin or resource manager role with access key	Registration authorized	Registration permitted	Pass
T23	Resource allocation	Navigate to resource assignment, assign resources	Attempt to assign resources without resource manager access	Unauthorized operation	Unauthorized operation	Pass
T24	Resource allocation	Navigate to resource assignment, assign resources	Attempt to assign resources without resource manager access	Resources assigned	Resources assigned	Pass

T25	Project approval	Navigate to projects page, approve proposal	Attempt to approve project without admin rights	Unauthorized access	Unauthorized access	Pass
T26	Project approval	Navigate to projects page, approve proposal	Attempt to approve project with admin rights	Proposal approved. Project created	Proposal approved and project created	Pass
T27	Project approval	Navigate to projects page, approve proposal	Attempt to reject project without admin rights	Unauthorized access	Unauthorized access	Pass
T28	Project approval	Navigate to projects page, approve proposal	Attempt to approve project without admin rights	Proposal rejected	Proposal rejected	Pass
T29	Proposal submission	Navigate to proposal submission, submit proposal	Attempt submission with PDF size over 8MB	Proposal not submitted	Proposal not submitted	Pass
T30	Proposal submission	Navigate to proposal submission, submit proposal	Attempt submission with PDF within limit and with valid sentinel characters	Proposal submitted	Proposal submitted	Pass

T31	Proposal status updation	Navigate to proposals page, make project decision update	Attempt to approve and already approved/rejected proposal	Operation invalid	Operation invalid	Pass
T32	Proposal status updation	Navigate to proposals page, make project decision update	Attempt to reject an already approved/rejected proposal	Operation invalid	Operation invaliud	Pss
T33	Proposal status updation	Navigate to proposals page, make project decision update	Attempt to approve a proposal that has approval status pending	Proposal approved. Project created	Proposal approved, Project created	Pass
T34	Proposal status updation	Navigate to proposals page, make project decision update	Attempt to reject a proposal that has approval status pending	Proposal rejected	Proposal rejected	Pass
T35	Resource fault reporting	Navigate to resources page, resort fault	Attempt to mark a resource as faulty without resource manager access	Unauthorized access	Unauthorized access	Pass
T36	Resource fault reporting	Navigate to resources page, resort fault	Attempt to mark a resource as faulty with resource manager access	Faulty resource token created	Faulty resource token made	Pass

Results

Quality assurance testing looks for potential problems in a proposed design. A test plan offers a road map for testing activity; it should state test objectives and how to meet them. Developing test cases and a test plan will help handle two issues in software quality namely, validation (& user satisfaction) and verification (or quality assurance).