**Software Requirements and Design Document**

**for**

**FYP CONNECT**

**Prepared by**

1. **22i-0773 Shayan**
2. **22i-0781 Ryan**
3. **22i-0789 Zubair**

**Company name : Lucid Softwares**

**Date : 26 nov 2024**

**Table of Contents**

**Table of Contents** [**ii**](https://docs.google.com/document/d/1mLt0a8jgmy2HxIJDwFwciODV8BesKKUk/edit#heading=h.2bn6wsx)

**1.** **Introduction** [**1**](https://docs.google.com/document/d/1mLt0a8jgmy2HxIJDwFwciODV8BesKKUk/edit#heading=h.qsh70q)

1.1 Purpose [1](https://docs.google.com/document/d/1mLt0a8jgmy2HxIJDwFwciODV8BesKKUk/edit#heading=h.3as4poj)

1.2 Product Scope [1](https://docs.google.com/document/d/1mLt0a8jgmy2HxIJDwFwciODV8BesKKUk/edit#heading=h.1pxezwc)

1.3 Title 1

1.4 Objectives 1

1.5 Problem Statement 1

**2.** **Overall Description** [**1**](https://docs.google.com/document/d/1mLt0a8jgmy2HxIJDwFwciODV8BesKKUk/edit#heading=h.49x2ik5)

2.1 Product Perspective [1](https://docs.google.com/document/d/1mLt0a8jgmy2HxIJDwFwciODV8BesKKUk/edit#heading=h.2p2csry)

2.2 Product Functions [2](https://docs.google.com/document/d/1mLt0a8jgmy2HxIJDwFwciODV8BesKKUk/edit#heading=h.147n2zr)

2.3 List of Use Cases [2](https://docs.google.com/document/d/1mLt0a8jgmy2HxIJDwFwciODV8BesKKUk/edit#heading=h.3o7alnk)

2.4 Extended Use Cases 2

2.5 Use Case Diagram 2

**3.** **Other Nonfunctional Requirements** [**2**](https://docs.google.com/document/d/1mLt0a8jgmy2HxIJDwFwciODV8BesKKUk/edit#heading=h.23ckvvd)

3.1 Performance Requirements 2

3.2 Safety Requirements [2](https://docs.google.com/document/d/1mLt0a8jgmy2HxIJDwFwciODV8BesKKUk/edit#heading=h.ihv636)

3.3 Security Requirements [2](https://docs.google.com/document/d/1mLt0a8jgmy2HxIJDwFwciODV8BesKKUk/edit#heading=h.32hioqz)

3.4 Software Quality Attributes [2](https://docs.google.com/document/d/1mLt0a8jgmy2HxIJDwFwciODV8BesKKUk/edit#heading=h.1hmsyys)

3.5 Business Rules [3](https://docs.google.com/document/d/1mLt0a8jgmy2HxIJDwFwciODV8BesKKUk/edit#heading=h.41mghml)

3.6 Operating Environment [3](https://docs.google.com/document/d/1mLt0a8jgmy2HxIJDwFwciODV8BesKKUk/edit#heading=h.2grqrue)

3.7 User Interfaces 3

**4.** **Domain Model 3**

**5.** **System Sequence Diagram 3**

**6.** **Sequence Diagram 3**

**7.** **Class Diagram 4**

**8.** **Package Diagram** [**4**](https://docs.google.com/document/d/1mLt0a8jgmy2HxIJDwFwciODV8BesKKUk/edit#heading=h.vx1227)

**9.** **Deployment Diagram** [**4**](https://docs.google.com/document/d/1mLt0a8jgmy2HxIJDwFwciODV8BesKKUk/edit#heading=h.3fwokq0)

# **Introduction**

## **Purpose**

This document outlines the software requirements for the Final Year Project (FYP) Management Portal, designed specifically for FAST University. The portal aims to simplify the management of undergraduate FYPs by automating administrative processes, improving communication, and offering tools to track project progress.

In this SRS document, we detail the portal’s key features, catering to the needs of students, supervisors, and administrators. The goal is to create a centralized platform that manages every stage of the FYP journey, from submitting proposals to final evaluations.

## **Product Scope**

The FYP Management Portal is a desktop application built with JavaFX, designed to tackle the inefficiencies in managing Final Year Projects (FYPs) at FAST University. With JavaFX, the portal offers an engaging and responsive interface that enhances the user experience while ensuring solid performance.

**Purpose and Goals:**

* **Simplify Administrative Work**: Automates tasks like scheduling evaluations, tracking deadlines, and managing project documentation in an easy-to-use desktop application.
* **Enable Effective Communication**: Creates a smooth communication flow between students, supervisors, and administrators in a user-friendly setting.
* **Centralized Progress Monitoring**: Provides tools to keep track of milestones, approvals, and deadlines, making sure projects stay on track.

**Relevant Benefits:**

* **Rich User Experience**: Utilizes JavaFX to offer a modern and dynamic interface with easy navigation.
* **Streamlined Processes**: Consolidates everything into one platform, removing the need for multiple communication channels or tools.
* **Desktop-Focused Solution**: A standalone application designed for offline use, perfectly suited to FAST University’s academic environment.

By leveraging JavaFX, the FYP Management Portal strikes a balance between high performance, ease of use, and scalability, catering to the needs of students, supervisors, and administrators while supporting FAST University’s goal of integrating technology into academic workflows.

## **Title**

| **FYP Connect** |
| --- |
| Simplifying Final year project management! |

## **Objectives**

* **Simplify Project Management**: Provide a single platform to manage every stage of Final Year Projects, from initial proposals to final evaluations.
* **Improve Communication:** Make it easier for students, supervisors, and admins to stay connected and share updates.
* **Track Progress:** Help keep projects on track with tools to monitor milestones and deadlines.
* **Boost Accountability:** Keep clear records of progress and communication to ensure everyone stays accountable.
* **Increase Efficiency:** Reduce delays and eliminate manual work with streamlined, organized workflows.

## **Problem Statement**

Managing Final Year Projects (FYPs) can be a complex and time-consuming task. Things like submitting proposals, scheduling evaluations, and tracking deadlines are often done manually through emails and documents, which can lead to confusion, missed deadlines, and added work for students, supervisors, and administrators.

This project aims to address these challenges by developing a straightforward desktop application using JavaFX. The platform will bring all FYP management into one place, automate repetitive tasks, and make it easier to track progress and communicate. This will save time, minimize mistakes, and ensure a smoother process for everyone involved.

# **Overall Description**

## **Product Perspective**

The FYP Connect application is a fresh, standalone solution built to manage Final Year Projects at academic institutions such as FAST University. It is not an upgrade or replacement for any current system, but rather aims to fill the gap in project management by providing a centralized, automated, and easy-to-use platform.

While the application operates independently, it can integrate with existing systems like university email or databases to improve communication and data storage. Below is a high-level diagram that shows how the system’s components interact with each other.

1) Have to add images of user interface, faculty panel , student panel etc

## **Product Functions**The main features of the FYP Connect application include:

## **Proposal Management:** Students can submit project proposals, and supervisors can review and approve them.

## **Progress Tracking:** Supervisors and admins can monitor milestones, deadlines, and project updates.

## **Document Handling:** A centralized space for uploading and accessing project documents.

* **Admin Controls:** Admins can manage users and system settings*.*

## **Communication Platform:** A seamless way for students, supervisors, and admins to stay in touch and share updates.

## 

## **List of Use Cases**

1. Create Groups
2. Submit Deliverables
3. Group Request
4. Join Groups
5. Search Functionality
6. Feedback Sharing
7. Mentorship Request
8. Resource Sharing
9. Supervise Projects
10. Set Timelines/Deadlines
11. Evaluate/Grade Students

## **Extended Use Cases**

## **Use Case Diagram**

# **Other Nonfunctional Requirements**

## **Performance Requirements**

***Response Time****: The desktop application should load pages and respond to user actions (e.g., submitting deliverables, loading feedback) within* ***3 seconds*** *for 95% of interactions. This ensures a fast and responsive user experience.*

***System Resource Usage****: The application should consume minimal CPU and memory resources to run efficiently, even on average systems, ensuring that it can function smoothly without causing system slowdowns.*

***Offline Functionality****: The application should be capable of functioning* ***offline*** *for most tasks, such as project submission and feedback management, and sync data when the connection is restored.*

***Scalability****: The system should handle up to* ***500 users*** *for local data storage (e.g., in case of large FYP batches) without performance degradation, ensuring smooth operation during peak usage times.*

***Real-Time Updates****: Any changes (e.g., project status or feedback updates) should be updated locally within* ***5 seconds*** *to ensure users are working with the latest data.*

## **Safety Requirements**

***Data Integrity****: The application should include automatic save and recovery mechanisms to prevent data loss during unexpected shutdowns. For example, it should save progress regularly and allow for recovery when the application restarts.*

***Input Validation****: The application should ensure that user inputs, such as project details or dates, follow proper formats to prevent incorrect data entry or system crashes. Any erroneous input should prompt the user with an appropriate error message.*

## **Security Requirements**

* ***User Authentication****: All users (students, supervisors, admins) must log in securely using a* ***username and password  
    
  Role-Based Access: Access to different features should be restricted based on user roles. For example, students should only be able to see and modify their own projects, while supervisors can view and grade projects assigned to them. Admins should have access to all user data and system settings.  
    
  Privacy: The system should adhere to privacy standards and ensure that personal data, like student information and project grades, is not accessible to unauthorized users. For example, only the assigned supervisor should be able to view and grade a student’s project.***

## **Software Quality Attributes**

***Usability****: The application should be intuitive and easy to use for all user roles (students, supervisors, admins). User tasks such as project submission, feedback sharing, and grade evaluation should require no more than* ***3 steps*** *to complete. The system should provide clear guidance and error messages when users make mistakes.*

***Reliability****: The application should be reliable, with a goal of* ***99.9% uptime****. It should not crash during routine operations, and error recovery mechanisms should be in place to prevent loss of data during unexpected failures.*

***Maintainability****: The codebase should be modular and well-documented to allow for easy maintenance and updates. Maintenance tasks, such as adding new features or fixing bugs, should be able to be completed within* ***2 weeks*** *for minor changes and* ***4 weeks*** *for significant updates.*

***Performance****: The application should handle a growing user base without compromising performance.* ***Response times*** *for data retrieval and interaction should remain under* ***3 seconds*** *even as the number of projects or users increases.*

***Adaptability****: The software should be adaptable for future features, such as support for more complex project structures or integration with additional external services (e.g., email notifications).*

***Testability****: The software should be testable, with unit tests covering at least* ***80%*** *of the codebase to ensure that new changes do not break existing functionality.*

***Portability****: The application should be portable across different operating systems, including* ***Windows, macOS****, and* ***Linux****, with minimal changes to codebase or configuration.*

***Interoperability****: The application should be capable of integrating with other systems used within the university, such as* ***email clients*** *for communication or* ***local databases*** *for user management.*

***Scalability****: The system should scale well to handle future increases in the number of projects and users, with minimal adjustments to the underlying infrastructure.*

## **Business Rules**

*<****Student Role****: Students can submit project proposals, request mentorship, join groups, submit deliverables, and track project progress. They cannot access or modify any other student's project or feedback.*

***Supervisor Role****: Supervisors can review project proposals, approve or reject them, supervise project work, provide feedback, grade projects, and set timelines for deliverables. Supervisors can only see and grade projects assigned to them.*

***Admin Role****: Admins can manage all users (students, supervisors), set global timelines, handle system settings, and view all projects and feedback. Admins have full access to the system and can make global changes, such as adding new users or modifying system configurations.*

***Access Control****: A student can only view and manage their own projects, while supervisors can access the projects assigned to them. Admins have access to all projects and user data.*

***Feedback and Evaluation****: Supervisors must provide feedback for every student submission, and the grades must be finalized before the project is marked as completed. Admins may review feedback and grades but cannot alter them.*

## **Operating Environment**

***Hardware Platform****:*

* *Minimum:* ***Intel i3*** *processor or equivalent,* ***4GB RAM****,* ***500MB free disk space***
* *Recommended:* ***Intel i5*** *or higher,* ***8GB RAM****,* ***1GB free disk space***

***Operating System****:*

* ***Windows 10*** *or later,* ***macOS 10.12*** *or later,* ***Ubuntu 18.04*** *or later*

***Software Components****:*

* ***Java Development Kit (JDK 8 or higher)***
* ***JavaFX*** *for the graphical user interface*
* *Local database (e.g.,* ***SQLite*** *or* ***H2*** *for lightweight local storage)*
* ***Git*** *for version control and team collaboration*

***External Systems/Integrations****:*

* *Integration with* ***university email systems*** *for notifications*
* *Possible integration with external document storage systems (e.g.,* ***Google Drive****,* ***Dropbox****) for handling large project files and resources*

***Network Requirements****:*

* ***Local Area Network (LAN)*** *for university campus deployment*
* *Internet connection for syncing with external resources, such as backups or email notifications*

## **User Interfaces**

*<Describe the logical characteristics of each interface between the software product and the users. This may include sample screen images, any GUI standards or product family style guides that are to be followed, screen layout constraints, standard buttons and functions (e.g., help) that will appear on every screen, keyboard shortcuts, error message display standards, and so on. Define the software components for which a user interface is needed. Details of the user interface design should be documented in a separate user interface specification.>*

# **Domain Model**

# **System Sequence Diagram**

# **Sequence Diagram**

# **Class Diagram**

# **Component Diagram**

# **Package Diagram**

# **Deployment Diagram**