HSTU Online Academic Transcript Distribution System

B.Sc. Level 3 Semester I PROJECT REPORT
Course Code: CSE 305 Course Title: Software Engineering

Submitted By

Amit Hasan Sikder(ID:2102043)

Submitted To

Pankaj Bhowmik
Lecturer, Department of CSE

Department of Computer Science and Engineering
Hajee Mohammad Danesh Science and Technology University,
Dinajpur-5200

Contents

Figures

Tables

Introduction

In today's fast-paced digital world, efficient academic services are crucial to a university's operations. The need for digitalization of academic transcripts is increasingly becoming essential for universities. This project involves designing and implementing an online system to automate and streamline the distribution of academic transcripts at Hajee Mohammad Danesh Science and Technology University (HSTU). The system is developed using the Agile Software Development Life Cycle (SDLC) to ensure flexibility, transparency, and continuous improvement during the development process.

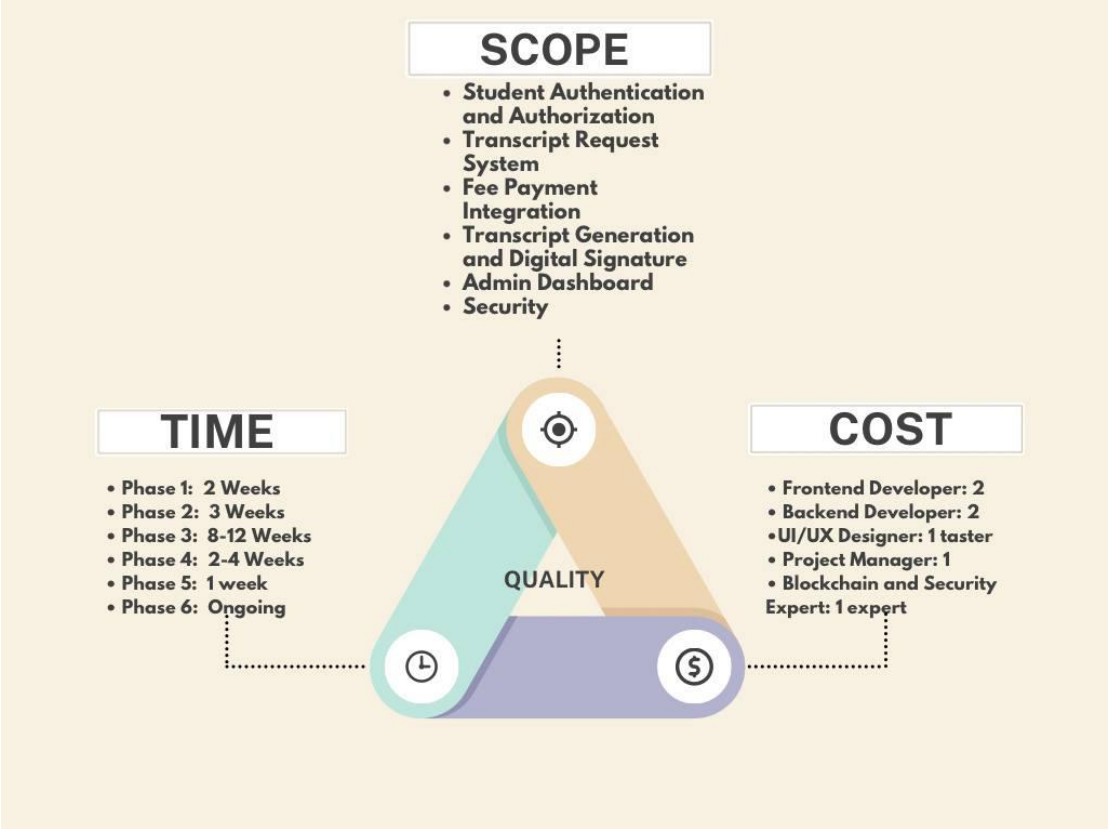

Fig 1.1 Project Management Triangle

Project Vision and Planning

2.1 Requirement Engineering

2.1.1. Feasibility Study

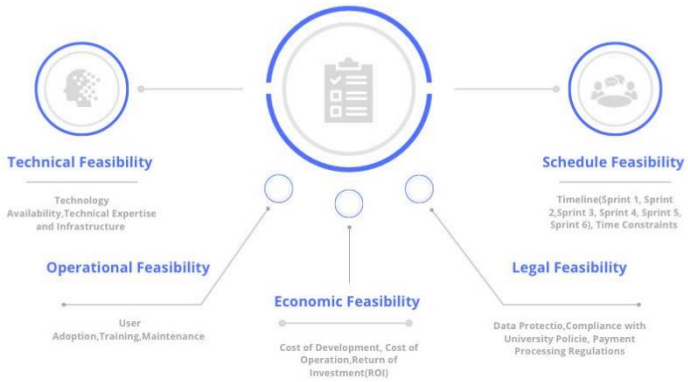

Fig 2.1 All Types of Feasibility Study

2.1.2. Product Backlog Creation: Gather requirements from stakeholders (students, faculty, administrative staff). Key features might include:

- Student authentication and authorization.
- Online request for academic transcripts.
- Fee payment integration (if applicable).
- Automated generation and digital signature on transcripts.
- Notifications for students (via email/SMS --Check students Account).
- Administrator control panel for managing requests.

2.1.3. Requirement Specification

- **User Requirements and System Requirements**

2.1.3.1. User Requirements

2.1.3.1(a) Student Requirements:

- **Student Authentication and Authorization**: Students must be able to securely log in using a university-issued ID or credentials.
- **Transcript Request**: Students should be able to easily request their academic transcripts online.
- **Payment Processing**: Students should be able to pay transcript-related fees online, if applicable.
- **Notifications**: Students should receive updates via email or SMS for their transcript requests' status (e.g., approval, processing, or rejection).
- **Transcript Receipt**: Students should be able to download or receive a digital copy of their transcript once approved.

2.1.3.1(b). Admin Requirements:

- **Request Management**: Admins must be able to view and manage transcript requests.
- **Approval/Denial**: Admins must have the ability to approve or deny transcript requests.
- **System Control**: Admins need access to a control panel to manage and oversee all operations.
- **Transcript Generation**: Admins should be able to generate transcripts, apply a digital signature, and send them to students.

2.1.3.1(c). Additional Stakeholder Requirements:

- **Faculty Requirements**: Faculty may need access to student records for academic purposes.
- **Administrative Requirements**: Administrative staff should be able to maintain the system and generate reports on student requests.

2.1.3.2. System Requirements

2.1.3.2 (a) Functional Requirements:

- **Student Login System**: A secure login system for students, integrated with the university's existing authentication system (e.g., OAuth, JWT).
- **Transcript Request Form**: A form where students can request transcripts, including fields for personal details, graduation year, and the number of copies needed.
- **Admin Dashboard**: A system dashboard that allows admin users to manage requests, approve or deny transcript requests, and view student history.
- **Transcript Generation**: Automatic generation of digital transcripts based on student records, including the application of digital signatures.
- **Notifications System**: Integration with email and SMS services to notify students of the status of their requests.
- **Payment Integration**: If applicable, an integrated payment gateway for processing transcript request fees.

2.1.3.2(b) Non-Functional Requirements:

Security: The system must ensure data security with encryption for student login, transcript data, and payment information. Blockchain may be used for authentication and securing transcript data.

Scalability: The system must be scalable to handle multiple requests from students without delays or performance issues.

Performance: The system should load efficiently even under high demand, and process multiple transcript requests simultaneously.

Reliability: The system should have high availability, with a robust backup system to ensure no data is lost.

Compliance: The system should comply with university data protection policies and international standards for student privacy (e.g., GDPR).

Usability: The system should have an intuitive user interface for both students and admins to ensure ease of use.

2.1.4. Requirements Elicitation

2.1.4.1. Requirement Elicitation Technique Apply(Survey/Questionnaire)

Survey(Close-Ended) Link:

https://docs.google.com/forms/d/e/1FAIpQLScOocRsYNOBFUCrXKw5-0LncaTc8EF4_HUF_1I7ihcfhxUFyg/viewform

Survey Result:

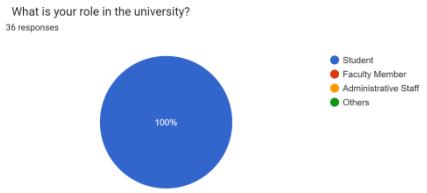

Fig.2.2 User role in University

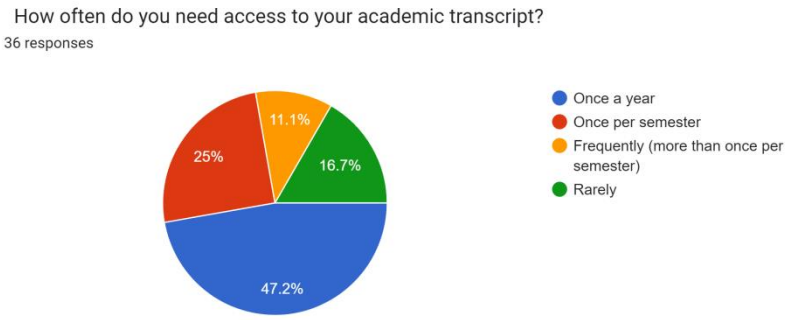

Fig 2.3 Question-1

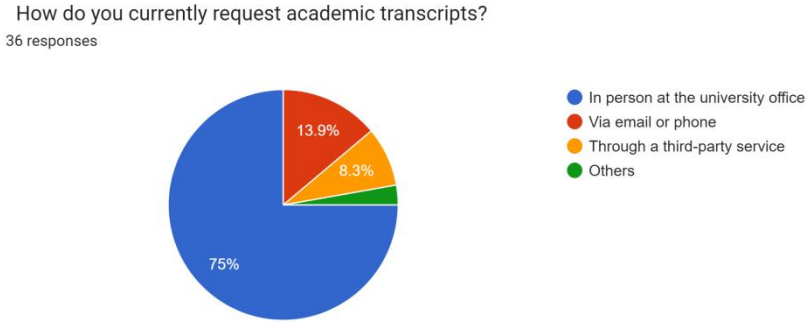

Fig 2.4 Question-2

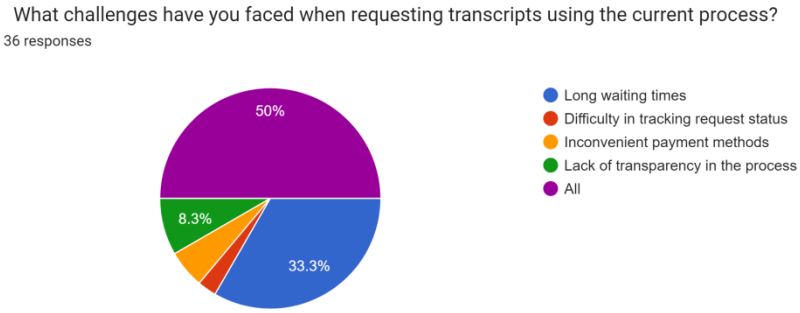

Fig 2.5 Question-3

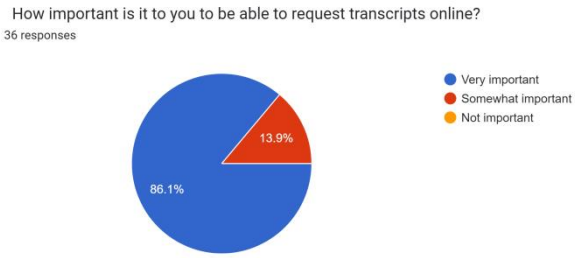

Fig 2.6 Question-4

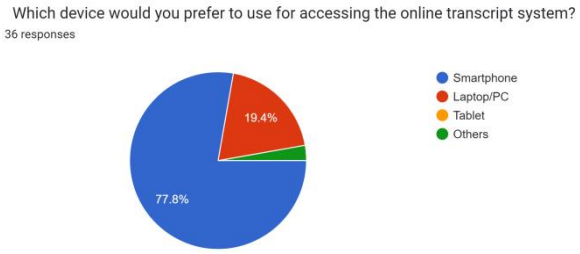

Fig 2.7 Question-5

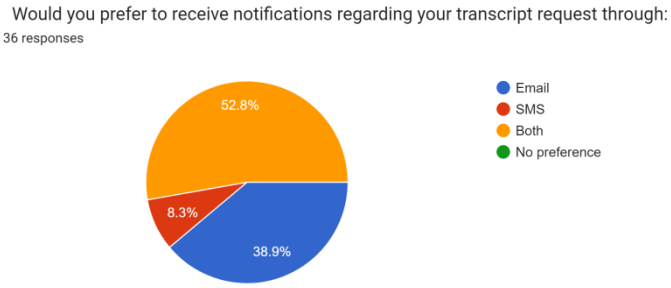

Fig 2.8 Question-6

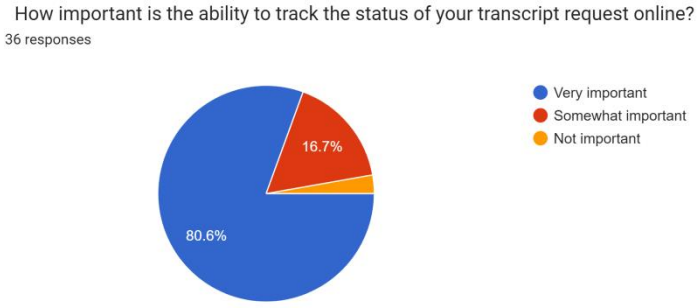

Fig 2.9 Question-7

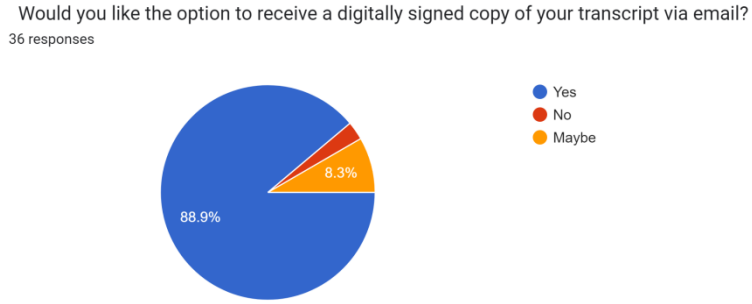

Fig 2.10 Question-8

What features can make the **Online Academic Transcript Distribution System** more useful for Students?

- Yes
- Digitization
- Add Student picture features for verification
- Devitalized system through website and time saving.
- Time should be 1 day

2.1.5. User Stories: Break down the features into smaller user stories.

For example:

- As a student, I want to securely request my transcript online.
- As an admin, I want to approve transcript requests.
- As a user, I want to pay online for transcript processing.

Initial Sprint Planning: Prioritize the most important stories for the first sprint.

Design and Architecture

System Architecture: Outline the high-level system architecture. This includes the back-end (database for storing student information, transcript data, and request statuses) and front-end (student and admin interfaces).

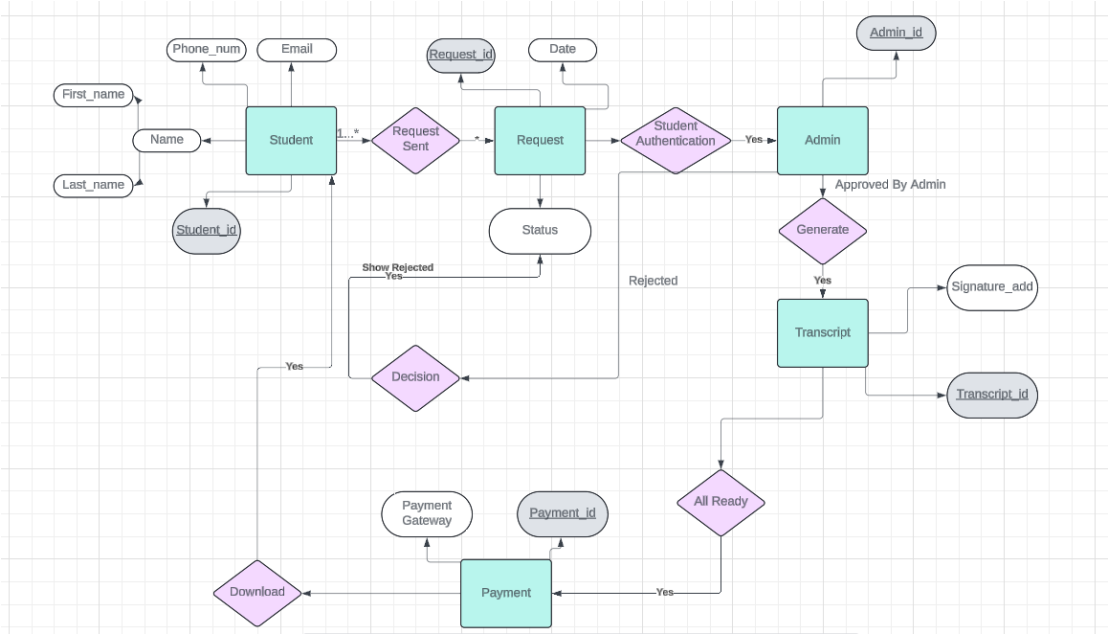

Fig 3.1 ER Diagram for System

3.1. Technology Stack:

- **Frontend:** React.js, Vue.js.
- **Backend:** PHP Laravel for managing server-side logic.
- **Database:** MySQL for student records and transcript data.
- Integration with payment gateway (if applicable).
- For Security,authentication use Block chain.

3.2. Mockups and Wireframes:

Design simple wireframes of how the user interface will look for both students and admins.

Development (Iterations/Sprints)

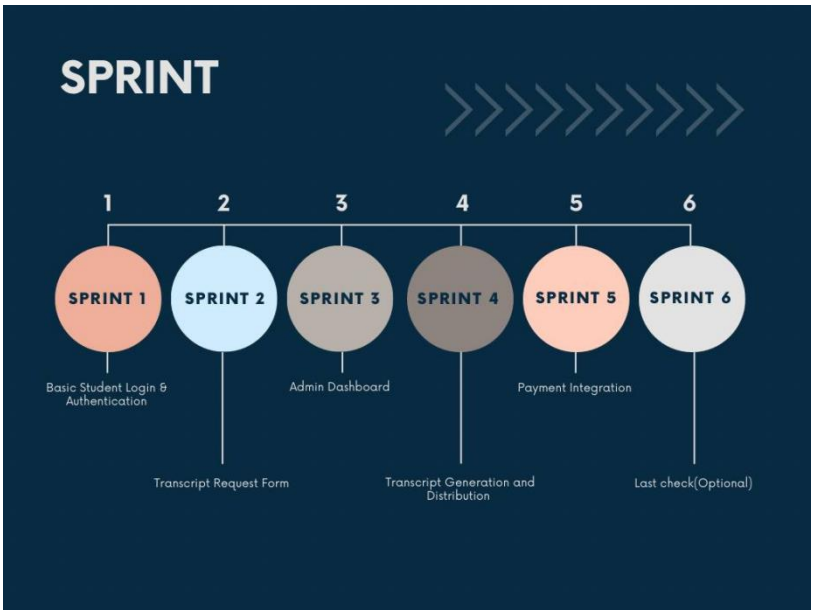

Fig 4.1 Sprint Serial

Agile focuses on delivering small, functional parts of the system in each sprint. For example:

4.1. Sprint 1: Basic Student Login & Authentication:

- Implement the student login feature using a secure authentication method (e.g., OAuth, JWT).

4.2. Sprint 2: Transcript Request Form:

- Build the interface where students can request their transcripts.
- Validate data and ensure a secure submission process.

4.3. Sprint 3: Admin Dashboard:

- Create an admin panel for viewing and approving/rejecting requests.

4.4. Sprint 4: Transcript Generation and Distribution:

- Implement logic to automatically generate the transcript and send a digital copy to the student.

4.5. Sprint 5: Payment Integration (if applicable):

- Add functionality for fee payment using a local or international payment gateway.

Each sprint includes regular meetings (daily stand-ups), sprint reviews, and retrospectives to continuously improve the process.

4.6. Sprint 6: Last check(Optional):

- Check all sprint's work accuracy

Testing and Feedback

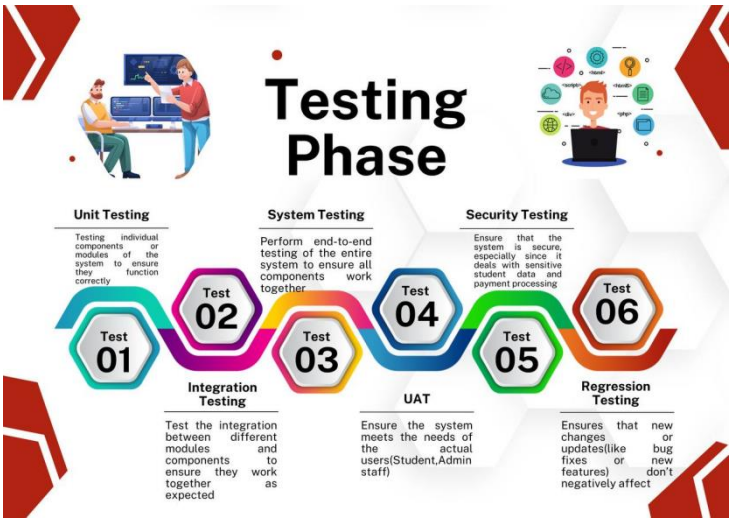

Fig 5.1 Testing Phase

5.1. Unit Testing:

Activities:

- Test **student authentication and login** (e.g., validation of credentials, password strength).
- Test the **transcript request form** for correct data submission and form validation.
- Test the **admin dashboard** for proper request management (e.g., approving/rejecting requests).
- Test the **payment gateway integration** (if applicable) for smooth transaction processing.

Tools:

- Use a testing framework such as **PHPUnit** for backend testing (Laravel).
- Use **Jest** or **Mocha** for testing frontend components (React.js/Vue.js).

5.2. Integration Testing:

Activities:

- Ensure the **student login** and **request submission** processes work seamlessly together.
- Verify the connection between **payment processing** and **request approval** (successful payments trigger the request status update).
- Test the flow from **admin approval** to **transcript generation** and email/SMS notifications.

Tools:

- Use **Postman** to test API endpoints and their integration with the frontend and backend.

5.3. System Testing:

Activities:

- Simulate full workflows like students requesting transcripts, admins approving them, and students receiving notifications.
- Test system load and performance (e.g., how the system handles multiple simultaneous requests).

Tools:

- Use **Selenium** or **Cypress** for automated system testing.
- **Apache JMeter** for load testing.

5.4. User Acceptance Testing (UAT):

Activities:

- Organize test cases for both students and admins based on their real-world scenarios.
- Have a group of selected students test the system by logging in, making requests, paying (if applicable), and receiving their transcripts.

- Have admin staff use the dashboard to approve/reject requests and manage system operations.

Process:

- Feedback is gathered, and necessary adjustments are made before the final release.

5.5. Security Testing:

Activities:

- **Authentication Testing:** Ensure student logins are secure, with proper encryption of passwords and user data.
- **Payment Gateway Security:** Verify that payment transactions are encrypted and safe from threats like **man-in-the-middle** attacks.
- **Data Protection:** Test blockchain integration (if used) for safeguarding transcript data.
- Ensure **SSL certificates** are correctly implemented to secure data transmission.

Tools:

- Use **OWASP ZAP** (Zed Attack Proxy) or **Burp Suite** for vulnerability testing.

5.6. Regression Testing:

Activities:

- Re-run all previous tests after changes to the system (e.g., adding new features like a payment gateway or admin management tools).
- Automate regression tests to quickly identify if something breaks after updates.

Tools:

- Use **Selenium** or **Cypress** for automating regression tests.

Deployment

- Deploy the system to a server or cloud platform (e.g., AWS, Azure, or HSTU's local servers).
- Ensure proper SSL **(Secure Sockets Layer)** certificates and security measures are in place for student data privacy.
- **Automation:** Automate repetitive and error-prone tasks to streamline the deployment process and reduce the likelihood of human errors.
- **Rollback Plan:** Develop a rollback plan in case issues arise during deployment, enabling a quick return to the previous stable state.
- **Environment Parity:** Ensure consistency between development, test, and production environments to minimize unexpected behavior due to environmental differences.
- **Communication:** Maintain clear communication channels between development, operations, and other stakeholders. This ensures everyone is aware of the deployment status and any potential issues.
- **Testing:** Rigorously test the software before deployment, including regression testing to ensure that existing functionality remains unaffected.
- **Version Control:** Use version control systems to manage and track changes to the software, providing a clear history and allowing for easy collaboration among team members.

Maintenance and Continuous Improvement

- Monitor the system's performance and resolve any bugs or issues.
- Continue to release updates and enhancements based on feedback.

7.1 Time and Space Complexity(Approximately):

Table 7.1 Complexity

Time Complexity	**O(n),** where n is the number of students or transcript requests.
Space Complexity	**O(n),** where n represents the number of students, transcript requests, or block chain entries.

Key Benefits of Using Agile for This Project:

- Iterative Approach: You can deliver and test functional pieces of the system regularly.
- Flexibility: Changes can be made quickly based on user feedback or evolving requirements.
- Transparency: Stakeholders are involved at every stage, ensuring that the system meets their needs.

Conclusion

The development of an online academic transcript distribution system for HSTU using the Agile SDLC offers a solution that is both efficient and user-friendly. The iterative nature of Agile ensures continuous improvement and adaptability to evolving needs. By automating the process, the system will save time for both students and administrative staff, while ensuring the accuracy and security of academic records.

This project demonstrates how modern software engineering methodologies, combined with cutting-edge technology, can greatly enhance university services.

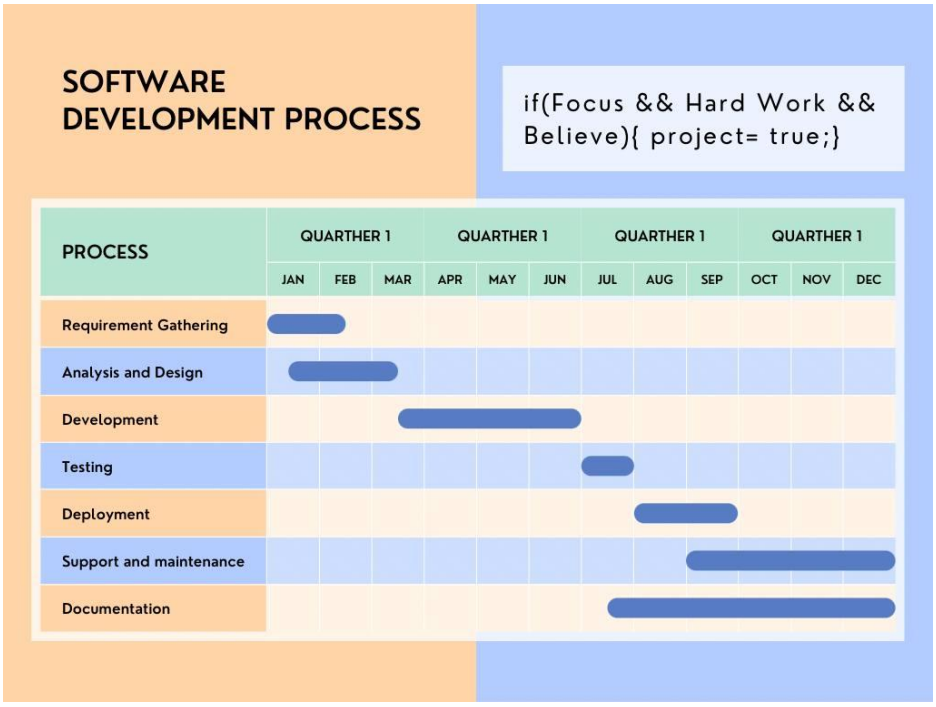

Fig 9.1 Suggested Software Development

References:

- **Book: Software Engineering, 9th Edition, Lan Sommerville**
- **Greeks For Greeks website: https://www.geeksforgeeks.org/software-engineering-requirements-engineering-process/**
- **Designing Website: https://lucid.app/documents#/home?folder_id=recent**
- **Canva Designing App**