# Introduction

The project shows the complete process of development of computerized course management system of Woodland University College (WUC) which was previously managed in a clerically based system. Scrum agile method will be used for development of the system.

## 1.1 Project Background

This project mainly focuses to fulfill the requirements described in the project description and other problem findings during the interview session. It mainly includes management system for student record, staff record, course record, module management, assignment management, attendance records, personal tutor management, timetable management, diary management, and report management.

## 1.2 Project Aims and Objectives

The aim of this project is to discuss and investigate the possibility of developing and implementing a computerized course management system for computing course provided by Woodlands University College, as per the directions and requirements of the stakeholders involved and the requirements developed from different other existing system like this one. The system should later be adaptive to scale it for all the courses provided by the college. To conquer this aim, following objectives can be set for this project:

* 1. Study the given documents including brief.
  2. Recognize and collect necessary information of problem domains to understand requirements.
  3. Form plans for elicitation to form requirements specifications.
  4. Conduct meetings with stakeholders to validate the requirements.
  5. Create system designs documents to fulfill the needs from user logs and requirements.
  6. Create prototypes according to requirements with regular validation from clients.
  7. Create Testing plans to test the system.
  8. Confirm and deliver the system to the clients.

## 1.3 Project Development Methodology

The project is conducted through Scrum Agile method. This is a principle of software development where clients and users are also involved in developing the system throughout the development. Jira is used for Project management and GitHub is used to implement the version control of the system. The system is created in Object Oriented Method which implies all the major areas of software or system as classes with different objects. Node JS with express libraries is used for creating the backend of the system whereas HTML, CSS and JavaScript is used for frontend. MongoDB is used to create databases used to store records.

# 2. Requirements Engineering

Requirements Engineering is the process of formulating the requirements of the system. The requirements can be functional (features of the system or software) and non-functional (speed, reliability). This is formed by different elicitation activities like interviews with clients and references from other system software.

## 2.1 Elicitation Activities

## 2.2.1.1 Existing Business Operation

Currently all the activities are performed on a clerical-based method. All the information is categorized and stored in registers. The current working process of paper-based system are listed below with a flowchart describing its basic working mechanism in clerical-based system.

2.2.1.1.1 Student Life-cycle

The student life cycle starts as soon as the student gets admission in a particular course and life cycle ends when the student is graduated. All the information like student’s detail and course details are stored manually in written form in a register. Current basic working mechanism is demonstrated in flow-chart:

New student gets admission in the college

Store the student details and course details in register

2.2.1.1.2 Personal Tutorial Life-cycle

Personal tutor life cycle starts when a student decides to have a personal tutor for additional help in his/her course and available tutor agrees for the activity. Above information are explained with the help of flowchart:

Student enquires for the available tutor for specific subject

If tutor is available and agrees

NO

YES

Information of tutor and student is recorded in a register

2.2.1.1.3 Staff Life-cycle

Staff life cycle starts with the permission of college to join as a staff in the college and ends when the staff leaves his/her job. All the working mechanism of staff life cycle is very similar to student life cycle.

2.2.1.1.4 Course Life-cycle

Course life cycle starts when a course leader finalizes a course and students start to enrol in it. A course may contain one or multiple modules. Analysing the complexity of a modules, one or multiple tutors may be assigned. It is further shown in the form of flowchart:

The course leader creates a course format

Course details are stored in a register

If students enroll in the course

NO

YES

Store student and course details in register

Assign module tutors and store their information in register

2.2.1.1.5 Module Life-cycle

Module life cycle starts after the beginning of course life cycle as a course may contain single or multiple modules and its life cycle may end if a course leader decides to remove course or a particular module. Module life cycle mainly includes the activity of storing information of students enrolled and module tutors assigned for each module. All of these activities are already shown in flowchart of course life cycle.

2.2.1.1.6 Assignment Life-cycle

Assignment is created by a module tutor and students of a particular module are informed. Assignment information are stored in register based on students who are taking the module of course. Its is further shown in flowchart:

A module tutor creates assignment and recorded in register

Students of a particular modules are assigned and recorded in register

2.2.1.1.7 Attendance Life-cycle

Attendance is maintained on two level in the organization. One level is at the staff level whereas another level is at student level. Both the records are managed in a register. The staff attendance record is only for the day counting of works, but the student attendance shows the class joined by student in each module. The basic working mechanism is similar, so only student attendance process is shown in the flowchart:

Each module tutor records students attendance for specific module in register

2.2.1.1.7 Timetable Life-cycle

Timetable is created by a course leader and the life cycle of timetable starts which may be change on different circumstances. As timetable is for all the module tutors their agreement is also required it is further explained in flowchart:

Course leader creates a timetable and informs all module tutors

NO

If all module tutors agree

YES

Timetable is recorded in register

2.2.1.2 Summary of existing limitations requiring resolution and existing strengths

As we have already listed the main working mechanism of current clerically based system, we can clearly see the difficulties in performing record maintenance task. It is not only about difficulties in maintaining records but the time-consuming process for searching any kind of information. There is huge chance of losing information because register is not a reliable source for storing data for long period of time. The solution to these problems can be a development of a computerized based system that can be reliable and maintained for a long period of time.

### 2.2.2 Functional Requirements

Functional Requirements are the features and facilities that a system needs to deliver. They are generally active requirements of the system that requires an input and gives a certain output (directly or indirectly). The following are the lists of functional requirements for records management system and Student portal for Woodland University College:

**Records Management System**

1. A user with proper access level should be able to add any data, and edit, delete, or archive it with one click of a button.
2. A user should be able to view the data (according to needs) with one click.
3. A user should be able to re-enlist the archived data.
4. Only Hashed password should be stored instead of actual password literals.

**Students Portal**

Requirements for student portal is separated for different types of users.

Common:

1. Users should log in with proper credentials.
2. All users should be able to manage their schedule through a diary management system.
3. Different pages should be shown for different kinds of users (admin, tutor, module leader and student).
4. Users should be able to change their passwords.

Admin:

1. Admin should be able to create global announcements.
2. Admin should be able to admit a student to a certain module.
3. Admin should be able to create, edit and delete records.

Module Leaders:

1. Module leader should be able to create announcement for a certain module.

Tutors:

1. The teachers should be able to create announcements for the students they teach.
2. The teachers should be able to upload resources and assignment public to the students they teach.
3. The teachers should be able to view the submitted documents and grade the assignment.

Students:

1. Students need to be able to submit a document or a file as assignment submission.
2. Grades should be visible to the students in their modules.
3. Students should be able to access the announcements and reading materials from the teachers.

### 2.2.2 Non-Functional Requirements

Non-functional requirements are the passive features requirements or performance requirements of a system. Some of these can be superficial attributes and others can be well-defined properties of the system needs. These requirements are listed below.

2.2.3.1) **Record Management System**   
   
 2.2.3.1.1) Speed: The system’s server is hosted by the university, and its speed is fast enough to record the data of the student. The network speed of the R.M.S (Record Management System) is 100 Mbps. The information can be interchanged in the system within some seconds. The system can interchange approximately 100MB of data each second. The system response to data with a high speed as the network in this system has high-level network. One can regain its data by searching within some seconds.  
   
  
 2.2.3.1.2) Capacity: As the number of students who take admission in the given different intakes is approximately around 5000. The data of each student and staff needs to be up to date in daily basis. The system has its own server which has a capacity of 1000TB. All the biodata and day to day accomplishment, activities of student and staff are recorded in R.M.S(Record Management System).  
  
 2.2.3.1.3) Reliability: The university needs to run its own server in order keep all it’s all record. The server we used for this system can run/operate around 200 hours per week. The system’s server can also be operated though there is a system failure as it’s designed in such a way. i.e., medium fault-tolerance. The backup system’s server of the university is totally operated by the university itself by the IT department. Whether the hardware part or software part is damaged we can get the information of the student and staff through the backup system server or by manually repairing the part of the server.  
  
 2.2.3.1.4) Usability: The system is easy enough to use by all the student and staff of the university. It can operate its information among the person who want to use the system /operator within an hour. The person who wants to operate the system can store the at least 100 data per hour.   
  
 2.2.3.1.5) Accessibility: The system is only accessible among operator who is given the permission to access the system by the university main server. Main server whose operations are execute by the specified person of the IT department or the principal of the university itself. Other operators than the one who isn’t granted the permission cannot get the accessible to the system. Also, the system is accessible to all the devices unless the id number or designed number for the operator isn’t registered in the main server.   
  
 2.2.3.2) **Student Records/ Information Portal:**

2.2.3.2.1) Speed: The student’s detail from the top to bottom are kept or recorded in the server. The information portal for the student is quite quick enough. It helps student to get info about the upcoming events, their day to day to assignments and the change in schedule and all. The portal/page is designed in such a way that the information is update twice/ thrice a week as per the need. The system loads the info at 1 second time as per the student needs.  
  
 2.2.3.2.2) Capacity: The information portal of the students doesn’t need very huge amount of space per student. Around 5GB memory space/capacity is assigned to the student for their personal details to fill-up along with their photo. Also, some spaces are assigned to the student in order update their personal info.  
  
 2.2.3.2.3) Reliability: To keep the record of student this server can perform or give satisfactory result. The capacity, speed of the server is as good as it needs to be. On the other hand, the backup server is also well managed. If there is one error or any kind of hardware damage, we can resolve the just by changing that particular part. We can, totally rely on this system because we can retrieve data through backup.  
  
 2.2.3.2.4) Usability: The information portal is easy enough for a student at this university to run. The page is easy enough to use for a student to run the page, store the data and access to the data in the page.  
  
 2.2.3.2.5) Accessibility: The information portal of students is accessible only to the students at the university. The university provides username and password to log into the portal so the data of the student is safe and easy to access for the genuine user or student of the student as they can get info about their upcoming events, results, and plan on likewise.

**Commercial Constraints**  
  
  
Table

Description automatically generated

# 3. System Analysis and Design

This component of the report contains the design documentation and system analysis. The system's functional requirements are obtained using object-oriented analysis (OOA) methodologies. Based on their function and characteristics, classes are recognized and categorized. Mongo DB is used to store records and data.

## 3.1 Preliminary Design Stages

Preliminary design stages consists of textual analysis, even analysis and a class-responsibility-collaborator diagrams to validate the requirements specification.

### 3.1.1 Textual Analysis

The textual analysis of the system is illustrated in this sub-section of the report. The major candidate classes and their responsibilities are recorded in tabular form according to requirement specification constructed above.

|  |  |
| --- | --- |
| **Candidate Classes** | **Responsibilities** |
| Users | Add,  Edit,  Delete,  Login,  Logout |
| Students | Add,  Edit,  Delete,  Assign course,  Assign personal tutor,  View courses,  View assignment,  View announcements |
| Modules | Add,  Edit,  Delete,  Assign course |
| Assignment | Create,  Edit,  Delete,  Assign course |
| Assignment\_students | Create,  Edit,  Delete,  Assign students,  Assign assignment |
| Courses | Add,  Edit,  Delete |
| Modules\_students | Add,  Edit,  Delete,  Assign student,  Assign modules |
| Teachers | Add,  Edit,  Delete,  Assign course,  Announce |
| Logs | Add,  Edit,  Delete,  Assign student,  Add schedule,  Calculate time remaining |
| Announcement | Add,  Delete,  Edit |
| Submission | Add,  Delete,  Assign student,  Assign Assignment,  Grade |

### 3.1.2 Significant Event Analysis

All the major activities within the system are listed below with its performers and their attributes involved in the event:

|  |  |  |
| --- | --- | --- |
| **Events** | **Performers** | **Attributes** |
| Log In | Users,  Stutents,  Teacher | Uid,  Password,  userRole |
| Log out | Users,  Stutents,  Teacher | Uid,  userRole |
| Create\_student | Users,  Admins,  Student | userRole,  user\_name,  user\_id,  user\_address,  user\_email,  user\_password\_hash,  user\_role,  user\_phone,  student\_module,  personal\_tutor\_id,  student\_level |
| Create\_teacher | Teachers,  Users | userRole,  user\_name,  user\_id,  user\_address,  user\_email,  user\_password\_hash,  user\_role,  user\_phone,  module, |
| Edit\_user | Users,  Admin | userRole,  user\_name,  user\_id,  user\_address,  user\_email,  user\_password\_hash,  user\_role,  user\_phone |
| create\_modules | Users,  Teacher,  Modules,  Course\_ID | User\_role,  Module\_Id,  Module\_name,  Module\_duration,  Module\_credit,  Course\_Id |
| Edit\_modules | Users,  Teacher,  Modules,  Course\_ID | User\_role,  Module\_Id,  Module\_name,  Module\_duration,  Module\_credit,  Course\_Id |
| Delete\_record | Users,  Teachers,  Modules,  Students,  Assignment,  Submission,  Attendance,  logs | User\_id,  User\_role,  Module\_id,  Students\_id,  Assignment\_id,  Submission\_id,  Attendance\_id  Log\_id |

### 3.1.3 Class Responsibility Collaborators

Class Responsibility Collaborators Table:

|  |
| --- |
| **Class Name:** Users |
| * **Responsibilities** * Add * Delete * Edit * Login * Logout |
| * **Collaborators** * Teachers * Students |
|  |

|  |
| --- |
| **Class Name:** Teachers |
| * **Responsibilities** * Add * Delete * Edit * Assign Modules |
| * **Collaborators** * Users * Assignment. * Announcement |
|  |

|  |
| --- |
| **Class Name:** Students |
| * **Responsibilities** * Add * Delete * Edit * RequestPersonalTutor |
| * **Collaborators** * Users * Teachers * Personal\_tutor |
|  |
| **Class Name:** Modules |
| * **Responsibilities** * Add * Delete * Edit * Assign course * Assign teacher |
| * **Collaborators** * Course * Users * teachers |
|  |
| **Class Name:** Modules\_students |
| * **Responsibilities** * Add * Delete * Edit * Assign\_student * Assign\_module |
| * **Collaborators** * Courses * Users * teachers |
|  |
| **Class Name:** Courses |
| * **Responsibilities** * Add * Delete * Edit |
| * **Collaborators** * Modules * Users * Teachers |
|  |

|  |
| --- |
| **Class Name:** Logs |
| * **Responsibilities** * Add * Delete * Edit * Calculate Remaining Time |
| * **Collaborators** * users |
|  |

|  |
| --- |
| **Class Name:** Assignment |
| * **Responsibilities** * Add * Delete * Edit * Announce |
| * **Collaborators** * Course * Users * teachers |
|  |

|  |
| --- |
| **Class Name:** Submission |
| * **Responsibilities** * Add * Delete * Edit * AddGrade |
| * **Collaborators** * Course * Users * Students * teachers |
|  |

|  |
| --- |
| **Class Name:** Announcement |
| * **Responsibilities** * Add * Delete * Edit * Announce |
| * **Collaborators** * Course * Users * Teachers * Assignment |
|  |

## Detailed Static System Designs

# 4. System Interface Designs

Note :- Write in this format… Only add lines in your section Do not change or touch other sections and push. Heading 2 should be inside heading 1 and heading 3 should be inside heading 2.