**Kingston University, BSc (Hons) (top-up)**

**Coursework Coversheet**

**Draft Coursework – Subject to Moderation**

**Part 1 - To Remain with the Assignment after Marking**

|  |  |
| --- | --- |
| **Student ID:** E142433 | **Student Name:** Minura Perera |
| **Module Code:** CI6125 | **Module Name:** Software Development Practice |
| **Assignment number:** 1 | **ESoft Module Leader:** |
| **Date set:** | **Date due:** 21st March 2025 |

**Guidelines for the Submission of Coursework**

1. Print this coversheet and securely attach both pages to your assignment. You can help us ensure work is marked more quickly by submitting at the specified location for your module. You are advised to keep a copy of every assignment.

2. Coursework deadlines are strictly enforced by the University.

3. You should not leave the handing in of work until the last minute. Once an assignment has been submitted it cannot be submitted again.

**Academic Misconduct**: **Plagiarism** and/or **collusion** constitute **academic misconduct** under the University's Academic Regulations. Examples of academic misconduct in coursework: making available your work to other students; presenting work produced in collaboration with other students as your own (unless an explicit assessment requirement); submitting work, taken from sources that are not properly referenced, as your own. By printing and submitting this coversheet with your coursework you are confirming that the work is your own.

|  |  |  |  |
| --- | --- | --- | --- |
| |  | | --- | | ESoft Office Use Only:  Date stamp: work received | | |  | | --- | |  | |

**Kingston University, BSc (Hons) (top-up)**

**Coursework Coversheet**

**Part 2 – Student Feedback**

|  |  |
| --- | --- |
| **Student ID:** | **Student Name:** |
| **Module Code:** | **Module Name:** |
| **Assignment number:** | **ESoft Module Leader:** |
| **Date set:** | **Date due:** |

|  |
| --- |
| Strengths (areas with well-developed answers) |

|  |
| --- |
| Weaknesses (areas with room for improvement) |

|  |
| --- |
| Additional Comments |

|  |  |  |
| --- | --- | --- |
|  |  |  |
| **ESoft Module Lecturer:** | **Provisional mark as %:** |  |
| **ESoft Module Marker:** | **Date marked:** |

**Table of Contents**

[1. System Introduction and Background 5](#_Toc193216010)

[1.1. System Introduction 5](#_Toc193216011)

[1.2. Problem Definition 6](#_Toc193216012)

[2. Software Requirement Specification 7](#_Toc193216013)

[2.1. Requirement Elicitation 7](#_Toc193216014)

[2.2. Functional and Non-Functional Requirements 8](#_Toc193216015)

[2.2.1. Functional Requirements 8](#_Toc193216016)

[2.2.2. Non-Functional Requirements 10](#_Toc193216017)

[3. Software Design 11](#_Toc193216018)

[3.1. Software Architecture Diagram (High Level Diagram) 11](#_Toc193216019)

[3.2. UML Diagrams 12](#_Toc193216020)

[3.2.1. Use Case Diagram 12](#_Toc193216021)

[3.2.2. Class Diagram 13](#_Toc193216022)

[3.2.3. Activity Diagrams 14](#_Toc193216023)

[3.2.4. Sequence Diagrams 15](#_Toc193216024)

[3.2.5. ER Diagram 16](#_Toc193216025)

[3.3. Wireframes 17](#_Toc193216026)

[4. Implementation 18](#_Toc193216027)

[4.1. Development Methodology 18](#_Toc193216028)

[4.1.1. What is RAD..? 18](#_Toc193216029)

[4.2. Justification of Technologies Used 20](#_Toc193216030)

[4.3. Development Framework 21](#_Toc193216031)

[4.4. User Interfaces and their related Codes 22](#_Toc193216032)

[4.5. User Manual 23](#_Toc193216033)

[5. Software Quality Assurance 24](#_Toc193216034)

[5.1. Test Plan 24](#_Toc193216035)

[5.2. Test Cases 25](#_Toc193216036)

[5.3. User Feedback 26](#_Toc193216037)

[6. Use of Collaborative Tools 27](#_Toc193216038)

[7. Conclusion 28](#_Toc193216039)

[8. Team Contribution 29](#_Toc193216040)

[References 30](#_Toc193216041)

[Appendix 31](#_Toc193216042)

# 1. System Introduction and Background

## System Introduction

Fostering a collaborative and productive academic culture in the dynamic setting of a university campus requires effective administration of events, resources, and participation. The Smart Campus Management System (SCMS) is a digital platform which was designed to improve staff and student engagement, maximize resource use, and streamline campus operations.

This system ensures smooth coordination between the students, lecturers and the administrators of the university by offering a centralized system for handling academic timetables, facility reservations, campus events, and communication channels. Through the use latest technology, SCMS makes it possible for process automation, and obtaining real-time updates, which lessens administrative workloads and improves the overall campus experience.

With tools related to resource allocation, event planning and attendance monitoring, SCMS seeks to make campus life more engaging and productive. By promoting involvement in university events, the system not only increases operational effectiveness but also cultivates a feeling of community. The university makes a major stride toward digital transformation by implementing this Smart Campus Management System, guaranteeing a more intelligent, connected, and future-ready academic ecosystem.

## Problem Definition

With the constant growth of the student population and the availability of diverse academic and extra-curricular activities, modern universities are facing a massive expansions in academic, administrative and social atmospheres. Due to the developing dynamic world, more new courses and degrees and being added to the university curriculums in order to provide the education necessary for sectors which have become recently popular (Ex: Artificial Intelligence) which results in more student enrollments. And with the growth of the student population more likely it is for organization and participation in extra-curricular activities and campus events to increase. Therefore the time schedules, resources, notices related to all these aspects should be managed by the campus administration.

But due to the vastness of the campus eco system, using traditional and manual systems for managing all these sectors is pointless since a limited number of administrators have to oversee the management of the whole campus, which would create operational bottlenecks in the campus administration. And since the young generation of the current society is familiar with digital technology it would be easier for them as well to manage their activities such as course registration, event participation, student-teacher communication and coursework submissions through a digital platform. The lecturers of the university will also benefit from a platform such as this, since it will allow them to manage all their classroom activities through one centralized point. So the Smart Campus Management System (SCMS) is being developed to address all these issues and provide a centralized hub which would streamline all the campus operations.

# 2. Software Requirement Specification

## 2.1. Requirement Elicitation

## 2.2. Functional and Non-Functional Requirements

### 2.2.1. Functional Requirements

The features or functions that the developers should integrate into the system in order for the end users to perform their required tasks can be defined as Functional Requirements. Basically they are the functionalities that a user expects or demand from a system. These requirements are derived from the user requirements gathered during the Requirement Elicitation process at the beginning of the development cycle (altexsoft, 2023; geeksforgeeks, 2025). In this scenario the users are the students, lecturers and the administrators within the university. Therefore given below are the Functional Requirements of this system categorized according to the specific user types:

**Students should be able to:**

* Register for courses
* Receive a confirmation message after registration
* Login to their account
* View their lecture schedule
* Register to campus events such as workshops
* Register to participate in extra-curricular activities
* Register to student councils
* Engage in group discussions with their lecturers and fellow students
* Receive and Download coursework material (lecture notes, assignments)
* Upload assignments
* Receive in-app notifications about campus events, announcements, lecture schedule changes etc.

**Lecturers should be able to:**

* Login to their accounts
* Schedule lectures and exams
* Change lecture and exam schedules if needed
* Reserve classrooms and labs
* View already reserved classrooms and labs to prevent overlapping in lecture schedules
* Upload coursework material
* Track student coursework submissions
* Mark student attendance
* Use group messaging to interact with students
* Send notifications to students regarding change in lecture schedules, assignment submissions or other important updates

**Administrators should be able to:**

* Login to their accounts
* Create accounts for students and lecturers
* Publish campus wide announcements and notices
* Allocate resources for campus activities

### 2.2.2. Non-Functional Requirements

Non-Functional Requirements refer to how the system should operate in terms of efficiency, security, reliability, usability, maintainability etc. These requirements are not mandatory but including them would guarantee a good overall user experience (altexsoft, 2023; geeksforgeeks, 2025). Given below are the Non-Functional Requirements of this system:

* The system should be responsive
* The system should be able to handle 2000 users simultaneously
* The pages of the system should load within 3 seconds
* The system should send a confirmation email to the students after they register to a course
* The system should protect user data such as personal details and passwords through encryption and prevent unauthorized access
* The system should have a user friendly design
* The system should be easy to modify and update in the future

# 3. Software Design

## 3.1. Software Architecture Diagram (High Level Diagram)

## 3.2. UML Diagrams

### 3.2.1. Use Case Diagram

### 3.2.2. Class Diagram

### 3.2.3. Activity Diagrams

### 3.2.4. Sequence Diagrams

### 3.2.5. ER Diagram

## 3.3. Wireframes

# 4. Implementation

## 4.1. Development Methodology

A software development methodology can be defined as framework or a set of steps followed by developers when developing software applications. The purpose of following these methodologies is to ensure that the development process is carried our smoothly (TatvaSoft, 2020). There are many development methodologies available such as Prototyping, Scrum, DevOps etc. But the methodology used for this project is RAD (Rapid Application Development).

### 4.1.1. What is RAD..?

RAD is a type of agile development methodology which provides high quality results within a quicker period of time when compared with other methodologies. The main purpose of this methodology is to accelerate the entire development process and deliver a functional system within a short timeline. Given below are the steps within this process as a graphical representation.

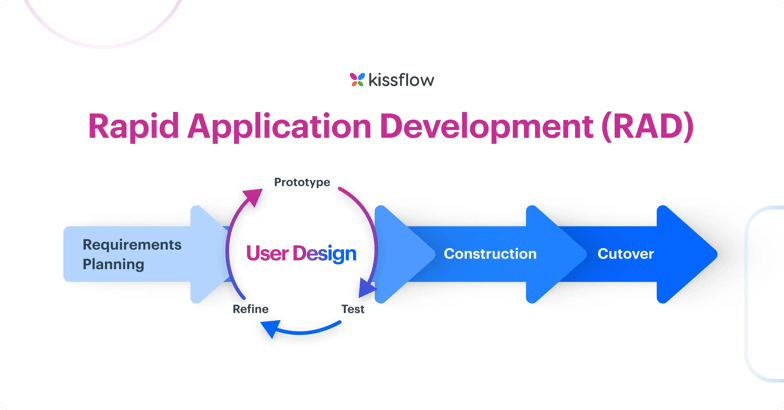


Figure 1: RAD Methodology (kissflow, 2025)

* **Requirement Planning:** Also known as the requirement elicitation process where the user requirements are gathered and analysed. Here the clients provide their ideas regarding what the system should be like which enables to developers to gain an overall idea define the requirements in a manner which satisfies the client’s vision.
* **Prototype:** In this stage, the developers create a prototype including the key features and functionalities. And then it is presented to the client or end users to obtain feedback. Afterwards the prototype is refined according to the obtained feedback to suit the client’s needs.
* **Construction:** Here the developers convert the working model of the application into a fully functional system.
* **Cutover:** This is the stage where the testing phase occurs and the final updates are made. Then the system is deployed into the production environment.

Justification for using RAD Methodology:

* **Time Constraints:** Since this system has to be developed within a limited period of time, RAD’s rapid iterations and modular approach helps deliver the system components quickly
* **Balanced Approach:** Instead of solely focusing on UX/UI feedback, RAD allows both the design and the functional components of the system to evolve in parallel.
* **Flexibility:** Changes can be integrated into the system without any complications since this methodology does not follow a rigid plan which is suitable for the evolving needs of the clients.

## 4.2. Justification of Technologies Used

## 4.3. Development Framework

## 4.4. User Interfaces and their related Codes

## 4.5. User Manual

# 5. Software Quality Assurance

## 5.1. Test Plan

## 5.2. Test Cases

## 5.3. User Feedback

# 6. Use of Collaborative Tools

# 7. Conclusion

# 8. Team Contribution

# References

# Appendix