OWASP Project Report on "TWERLINGHALL.COM"

Secure Software Design and Programming
SE 1003

Project Name
TWERLINGHALL.COM

Team Members:

Shanto Rahman	BSSE 0302
Md. Mostafijur Rahman	BSSE 0303
S M Sofiqul Islam	BSSE 0323

Submitted to:
Asif Imran
Lecturer, IIT, DU

Institute of Information Technology

University of Dhaka

Submission Date: 10th June, 2015

Letter of Transmittal

We declare that this OWASP project report on “**TWIRLINGHALL.COM**” is submitted to Institute of Information Technology, University of Dhaka to fulfill the requirements for the Masters of Information Technology (BSSE) in 1st Semester. This documentation work has done under the guidance of **Asif Imran**. Though this project is mainly developed for Institute of Information Technology, this is certified that we have completed this documentation under MS-1003, Secure Software Design and Programming course and it has not been submitted elsewhere for the requirement of any degree or for any other purposes.

Members of the Project	Signature and Date
1. Shanto Rahman	_______________
2. Md. Mostafijur Rahman	_______________
3. S M Sofiqul Islam	_______________

Acknowledgement

We would like to express our deepest gratitude to **Asif Imran**, Lecturer, Institute of Information Technology, University of Dhaka for his kind support and inspiration which has immensely strengthened our confidence during our project security report development process.

We owe our profound sense of gratitude to Dr. **(Mrs.) Nilufar Nahar**, Provost, Kabi Sufia Kamal Hall, University of Dhaka for providing us valuable information and project development opportunity for Kabi Sufia Kamal Hall.

We are also very much grateful to hall officials for their great support by providing required information and data.

Abstract

With the advancement of information technology, internet, software and website offers gratification to people by minimizing their work load such as huge computation, huge paper work and other things. The Twirling Hall.com is such kind of Web portal which provides an automated platform for the hall office manager and student to manage their work. The purpose of creating this system is to improve the efficiency of the work of the officers and to reduce the hassle of the students. The system enables the hall management officers to store all the information about students in a more synchronized way. As a result, there is no chance of losing any information of any student. On the other hand, student will be much more benefited. In a nutshell we can say, it increases the productivity of work and reduces the time and hassle of both the students and the officers. The project consists of many modules and each of them has its own features and functions. In our project, we will define two kinds of role those are admin and student. Many features will be included here, such as ID card generation, testimonial, online seat allocation, virtual noticeboard and others. But, main challenge of this web project is to provide proper security against different kinds of vulnerability as providing security in every stage of SDLC may increase the time, cost, and expertise demand. However, we have fixed security requirements for this project on the basis of OWASP security guidelines. This report presents the functional requirement analysis as well as security requirement analysis with proper testing and mitigation plan on the basis of OWASP security guideline. This project will be well enough to meet user expectation providing a more secure and user friendly web interface with all expected functionalities.

List of Figures

List of Tables

Chapter 1: Introduction

This chapter briefly discuss on the overview of this research. It contains seven sections. The first section is background study; follow by the problem statement. Next are the business requirement document which contains business problem statement, business scope, objective or goal of the project, assumptions, functional and non-functional requirements. Domain analysis, feasibility study is also discussed in next sections and lastly is the document organization which briefly describes the structure of this document.

1.1 Project Background

The TwirlingHall.com is mainly a web portal to manage all the information of the students' of any institution. Almost every institution offers the hall facility to the students. The students come from different areas of the country to achieve their education. Most of these students live in the hall of their institution. With the increase in education rate, the numbers of students in the halls are also increasing. The hall authority is facing problem to manage these students' information manually. The traditional work style in any hall office is to do all the works manually. For this reason, they sometimes loose information about students', complain information about different things of the hall by the students' etc. This causes problem both for the students and the hall authority. Students have to go to the office several times for a simple work like for getting the ID card, boarding card, testimonial, scholarship etc. The hall authority gives notice on the noticeboard but as many of the students do not check the noticeboard regularly, they often miss important notices. An automated system to keep track of all the works of the hall office can reduce these hassles for both the students and the hall authority. With this system, the officers can manage the students' information more efficiently than the manual process. The students' can also get rid of some painful activities like standing in a long line to update their information. They can also view the notices of the scholarship or other activities in the system.

1.2 Project Scheduling

Table 1.1: Project Schedule

Task	**Time**
Project Proposal	01.07.2014-16.07.2014
Requirement gathering and specification	17.07.2014-31.07.2014
Design and Development	01.08.2014-30.09.2014
Testing and enhancement	01.10.2014-31.10.2014
Project submission	01.11.2014

1.3 Problem Statement

University of Dhaka is the largest public university in Bangladesh with more than 30,000 students. All students must be attached with halls. The main aim to develop this software is to manage the operations performed in our hall.

The basic information about our hall is as follows

- ❖ There are about 56 departments with a total strength of 26000 students. From the beginning of the admission information of all these students are managed by their attached hall.
- ❖ Now, there are 26 halls available.

The managing services required are

- ➢ Information management: To maintain the following details
  a. Students' previous needed information.
  b. Students' current education status.
  c. Department and institutions detail.
- ➢ Other services:
  a. Hall gives ID card to each student.
  b. Students collect their admit card, testimonial and boarding card from attached hall.

  Now, hall officers give all discussed services manually. On failure or delay of any service can cause destructive to both students and halls authority.

So, here is the need of this proposed system "TiwrlingHall.com", a website that for managing hall works.

The purpose of the TwirlingHall.com is mainly to provide automation of hall officer's work. The categories of users provided are:

Admin: He can read or write the information about any students and can update, delete or create other users' accounts.

Student: They can collect their information.

With this website, the manual work is converted into automated online application. For example, using these site students can download their ID card, Boarding card by giving some input.

1.4 Organization of Document

This document consists of nine (9) chapters. Chapter 1 will discuss on introduction to system. The purpose of this chapter is to introduce about the system that will be develop. This chapter contain several part namely introduction, problem statement and document organization.

Chapter 2 contains software development plan document which consist of process modeling, resource and scheduling.

Chapter 3 contains business requirement document, business case, domain analysis, and feasibility study.

Chapter 4 is about software requirement specification and analysis. This chapter explains about the requirement gathering/ specification and analysis process which includes user scenario, quality function development (QFD), use case and diagram, activity diagram, swimelane diagram, state diagram, data flow diagram, UML class diagram and CRC, data modeling including E-R diagram and database schema.

Chapter 5 is about software design which includes architectural and user interface design of the system.

Chapter 6 is about risk management plan. This chapter consists of identification of risks, probability and impact of those risks using risk register (RR) and risk matrix (RM). It also discusses the plan to mitigate those risks.

Chapter 7 presents security requirement analysis. This chapter describes the security requirements with extensive analysis and proper testing on the basis of OWASP. Moreover, top ten security vulnerability, their testing process for this project and mitigation plan are described in this chapter.

Chapter 8 is about implementation and testing. This chapter contains the implementation process which includes implementation tools, technologies and methodologies. This chapter also includes test plan including black box and white box testing and test reports conducted on developed system.

Chapter 9 discusses about user manual which consists how to interact with different modules of the systems. It explains the user manual dividing into two part, such as administrative and user (student and global) responsive interface.

Chapter 10 is about conclusion. This chapter will briefly summarize and conclude the proposed project.

Chapter 2: Software Development Plan

This chapter presents the software development plan (SDP) for “twirlinghall.com”. This SDP describes the organization and procedures to be used by the project development team in performing software development for hall management system. This plan also includes resource planning, work break down structure, and activity network etc. This plan is intended to be used by the sponsor, developer as well as engineers.

.

2.1 Overview of Required Work

Development activities in this plan are subject to the following requirements.

2.1.1 System and Software Development Requirements and Constraints

a. The Twirlinghall.com will implement the software requirements identified in its System Requirements Specification (SRS).

b. The Twirlinghall.com will be written in the php programming language.

c. The hardware and software are expected to be delivered to the development site on time.

2.1.2 Project Documentation Requirements and Constraints

Project documents will be developed in accordance with software product descriptions in IEEE format, Software Development.

2.1.3 Project Position in the System Life Cycle

Twirlinghall.com is at the beginning of its life cycle. The software development model selected for Twirlinghall.com is once-through (waterfall) strategy as possibility of requirements change is low.

2.1.4 Selected Program/Acquisition Strategy

The twirlinghall.com is a new development. Software development will be accomplished and managed by hall personnel, with selected activities contracted out using existing contractual agreements.

2.1.5 Project Schedules and Resources

Project schedules start from 15th July, 2014 and will be end at 30th November, 2014. Resources will be handover to development team on time.

2.1.6 Other Requirements and Constraints

None.

2.2 General Software Development Activities Plan

2.2.1 Software Development Process

The TwirlingHall.com software development is tailored from the guidelines of IEEE/EIA 12207. Major project development phases and activities related to these standards are shown in Table 2.1.

Table 2.1: Major NISBS Project Activities related to IEEE/EIA 12207

TwirlingHall.com Activity	**IEEE/EIA 12207.0 Activity**
Project Planning and oversight	Process implementation Management Process
Phase 1: Software Requirements	Software requirements analysis
Phase 2: Software Design	Software arch. Design Software detailed design
Phase 3: Software unit development, test, and Integration	Software coding and testing Software integration
Phase 4: System Qualification Test and Delivery	Software qualification testing System integration System qualification testing
Phase 5: Support of installation and use	Software installation Software acceptance support
Software Configuration Management	Configuration management process
Software Quality Assurance	Quality assurance process
Project reviews	Joint review process

The **TwirlingHall.com** team will develop the system in accordance with processes defined in context of the software engineering process model presented above. The software development process is to construct an overall software architecture that will develop software in a single build upon the chosen architecture. The process will integrate reusable software from existing sources with newly-developed software. Software design and coding will be performed using an object oriented design approach and generating class and object process interaction diagrams. Artifacts and evidence of results of software development activities will be deposited in Software

Development Files (SDFs) These artifacts, along with pertinent project references will be deposited and maintained in a Software Development Library (SDL) and made available to support management reviews, metrics calculations, quality audits, product evaluations, and preparation of product deliverables.

2.2.2 General Plans for Software Development

TwirlingHall.com development will follow the guidelines in IEEE/EIA 12207. The development approach will apply selected Level 2 and Level 3 software engineering processes in accordance with the Software Engineering Institute (SEI) Capability Maturity Model (CMM) and product evaluation procedures recommended by SSC San Diego’s Software Engineering Process Office (SEPO). The project team has tailored these standards, practices, and processes to TwirlingHall.com development.

2.2.2.1 Software Development Methods

The TwirlingHall.com development will apply the following general methods:

a. Phase 1: Software Requirements. The project will follow the defined processes documented in Chapter 3 & 4 to conduct software requirements analysis and develop the Software Requirements Description (SRD) and preliminary User Documentation Description (UDD). Software requirements will be expressed in language that addresses a single performance objective per statement and promotes measurable verification.

b. Phase 2: Software Design. The software architecture will consist of reusable software components and components to be developed. Software requirements will be allocated to one or more components of that architecture. The project will follow the defined processes documented in Chapter 4 & 5 to conduct object-oriented architectural and detailed software design of new software and to capture the design, and reengineer if necessary the software to be reused. Emphasis will be placed on good software engineering principles such as information hiding and encapsulation, providing a complete description of processing, and the definition of all software and hardware component interfaces to facilitate software integration and provide a basis for future growth. The project will only design and develop software to meet requirements that cannot be satisfied by reusable software. New software will be based on principles of object-oriented design and exploit object-oriented features associated with the selected high-level language and development environment. New software design will be defined at top-level and detailed design stages of development in the SDD. Software design will promote ease of future growth, specifically new application interfaces. Top-level design will be expressed in a graphical form. Detailed design will also be expressed in a graphical form depicting classes, relationships, operations, and attributes.

c. Phase 3: Software Unit Development, Test, and Integration. The project will develop new code (or modify reused code), unit test, integrate, and document software following the processes in Chapter 6. While reused code will not be expected to conform to a single coding standard, changed source code must be supplemented with sufficient new comments and standard code headers to meet commenting provisions of the coding standard and to promote understandability.

d. Phase 4: System Qualification Test and Delivery. The project will conduct Qualification Testing according to Qualification Test Plans and Procedures, and document results in a Test Report. After successful Software Usability Review (SUR), the software will be ready for installation at www.

e. Phase 5: Support Installation and Use. The TwirlingHall.com Project team will provide support to software installation, acceptance testing, and user operation. More detail of this phase will be added to future updates of this SDP.

2.2.2.2 Standards for Software Products

TwirlingHall.com development will follow the standards and guidelines listed in Referenced Documents. These documents impose standards that are applicable to software requirements, design, coding, testing, and data. TwirlingHall.com documentation will comply with applicable directions contained in the documents listed in Table 2.2.

Table 2.2: TwirlingHall.com Documentation Guidelines

TwirlingHall.com Document	**Primary Guidelines**
Software Development Plan (SDP)	IEEE: Software Development Plan
Business Requirements Description (BRD)	12207.1 6.21: Business requirements description
User Requirements Description (URD)	12207.1 6.30: User documentation description
Software Requirements Specification (SRS)	12207.1 6.22: Software requirements description
Software Design Description (SDD)	12207.1 6.16: Software design description
Software Interface Design (SID)	12207.1 6.19: Software interface design
Integration Test Plan/Procedures	12207.1 6.27/28: Test or validation plan/procedures
Integration Test Report	12207.1 6.29: Test or validation results report
Source Code Record (SCR)	12207.1 6.24: Source code record
Qualification Test Plan/Procedures	12207.1 6.27/28: Test or validation plan/procedures
Qualification Test Report	12207.1 6.29: Test or validation results report

2.2.2.3 Reusable Software Products

This section identifies and describes the planning associated with software reuse during development of the TwirlingHall.com project and provisions to promote future reuse of newly developed software. TwirlingHall.com components that must be developed will be designed and documented to facilitate their reuse within similar architectures.

2.2.2.4 Handling of Critical Requirements

Compatibility of interfaces with other systems is important in successful development of the TwirlingHall.com software. These programs will be continually monitored by the Software Project Manager to identify, track, and evaluate potential risks.

2.2.2.5 Computer Hardware Resource Utilization

The Software Project Manager will establish and maintain a detailed schedule for computer hardware resource utilization that identifies anticipated users, purposes, and scheduled time to support analysis, software design, coding, integration, testing, and documentation. It will address sharing of resources by multiple users and workarounds to resolve conflicts and equipment downtime. The Software Project Manager will coordinate resource needs with development, integration, and test groups.

2.3 Detailed Software Development Activities Plan

The overall life cycle to be used for TwirlingHall.com development is shown in Figure 2.3. A modified once-through (waterfall) strategy with five major phases and supporting activities will be used. Major documents and management reviews are also shown in the figure. Detailed plans for individual activities are contained in the following subsections.

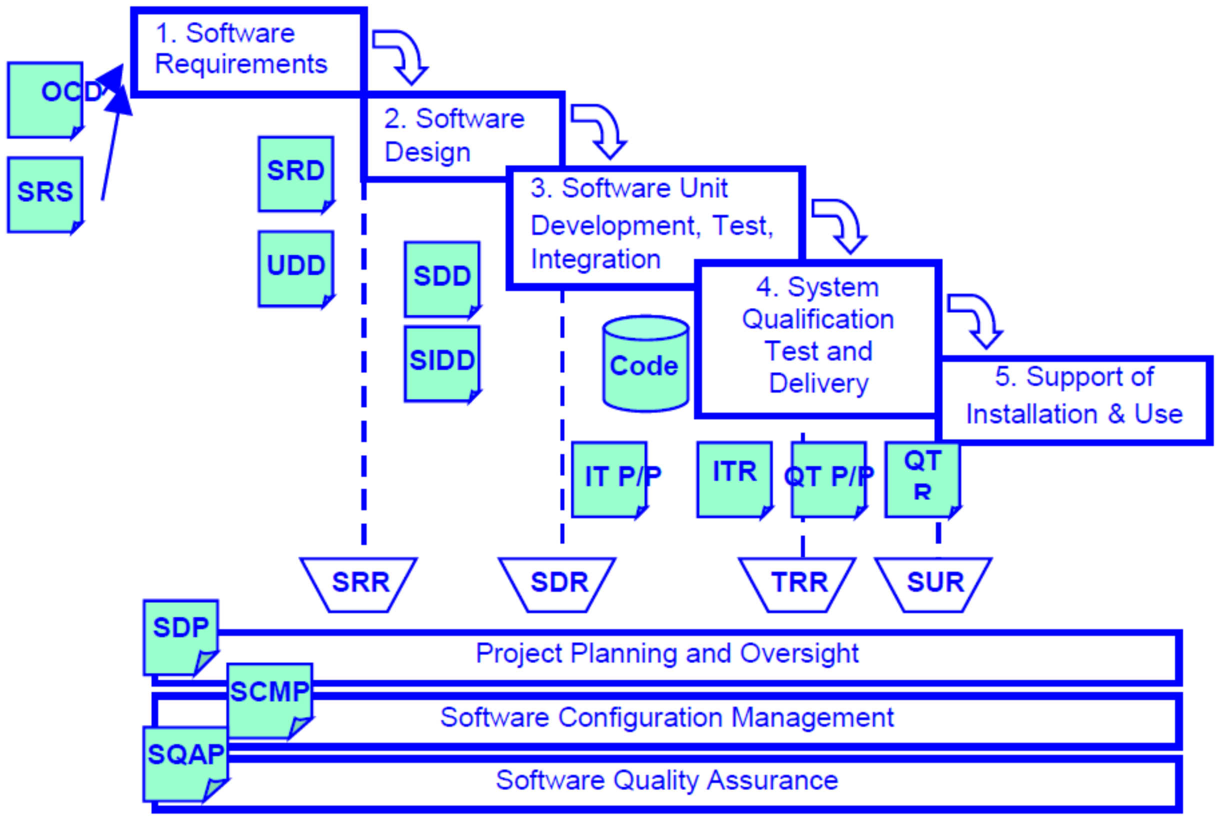

1. Software Requirements
OCD
SRS
2. Software Design
SRD
UDD
3. Software Unit Development, Test, Integration
SDD
SIDD
Code
4. System Qualification Test and Delivery
5. Support of Installation & Use
IT P/P
ITR
QT P/P
QT R
SRR
SDR
TRR
SUR
SDP
Project Planning and Oversight
SCMP
Software Configuration Management
SQAP
Software Quality Assurance

Figure 2.1: TwirlingHall.com software life cycle

2.3.1 Project Planning and Oversight

Good software project management principles shall be followed throughout the NISBS Formatter life cycle. Software development activities shall be planned in advance and monitored carefully throughout the life cycle.

RESPONSIBILITY: the Software Project Manager is responsible for preparing and executing this Software Development Plan.

ENTRANCE CRITERIA: Approved Operational Concept Document (OCD), Systems Requirements Specification (SRS), and project approval received from customer.

INPUTS: OCD, SRS, project approval

TASKS:

1. Clarify the requirements

   - Baseline the requirements documents (OCD and SRS) and control changes
   - Select and tailor standards, processes, and policies to be used

2. Identify the processes

   - Identify the life cycle to be used on the project
   - Establish processes for each phase of the life cycle

3. Document the plans

   - Document software plans in this Software Development Plan. Participation by appropriate parties in the SDP Peer Review and acceptance signatures on the title page demonstrates agreements to commitments in this SDP.
   - Identify work products
   - Establish schedules for the timely completion of tasks
   - Estimate the size of software work products
   - Estimate the project effort, costs, and resources
   - Identify project risks
   - Establish the project organization

4. Track the progress

- Monitor execution of planned activities
- Analyze status and take action to modify processes, change resources, adjust schedules, or amend plans to satisfy requirements. Obtain agreements from affected groups.
- Conduct Management Reviews and Status Reviews to determine status of ongoing operations
- Review status of all activities periodically with SSC SD Senior Management

5. Control the products

- Establish a software configuration management activity to control baselined work products
- Establish a software quality assurance activity to monitor project products and processes

6. Cultivate teamwork

- Assemble and manage the required project staff and resources
- Provide training needed by the project staff
- Include contractors in technical discussions

7. Apply appropriate technology

- Consider reuse during all life-cycle phases
- Strive to continually improve project processes

OUTPUTS: this Software Development Plan, operational TwirlingHall.com system

EXIT CRITERIA: Satisfactory execution of this Software Development Plan

2.3.2 PHASE 1: Software Requirements

The TwirlingHall.com Team shall establish and document software requirements, including the quality characteristics, specifications, as described below.

RESPONSIBILITY: Software development team, System engineering team

ENTRANCE CRITERIA: Approved Operational Concept Document (OCD) and Systems Requirements Specification (SRS) received from customer.

INPUTS: OCD and SRS

TASKS:

1. Document the software requirements in Software Requirements Description (SRD), including functional and capability specifications, interfaces, safety specifications, security specifications, human-factors engineering, data definition and database requirements, installation and acceptance requirements, user documentation, user operation and maintenance requirements.

2. Evaluate the requirements against criteria: traceability to system requirements, external consistency with system requirements, internal consistency, testability, flexibility of software design, operation, and maintenance.

3. Develop a preliminary User Documentation Description (UDD).

4. Conduct Peer Reviews on SRD and UDD.

5. Conduct Software Requirements Review (SRR).

OUTPUTS: Approved SRD and UDD

EXIT CRITERIA: Successful completion of SRR checklist

2.3.3 PHASE 2: Software Design

The TwirlingHall.com Team shall transform the requirements for the software item into an architecture that describes its top-level structure and identifies the software components, and then develop a detailed design for each software component.

RESPONSIBILITY: Software Development Team

ENTRANCE CRITERIA: Successful completion of SRR checklist

INPUTS: Approved SRD and UDD

TASKS:

1. Transform requirements into a top-level architecture

2. Document top-level design for interfaces

3. Document top-level design for the database

4. Develop the design for each component and document it in the Software Design Description (SDD).

5. Develop the design for interfaces and document it in the System Interface Design Description (SIDD)

6. Document the design for database

7. Evaluate architecture against criteria: traceability to requirements, external consistency with requirements, internal consistency between software components, appropriateness of design methods and standards, feasibility of design, operation, and maintenance.

8. Update the UDD as necessary based on the software design.

9. Conduct Peer Reviews on SDD, SIDD, and updated URD

10. Conduct Software Design Review (SDR)

OUTPUTS: SDD, SIDD, updated UDD

EXIT CRITERIA: Successful completion of SDR checklist

2.3.4 PHASE 3: Software Unit Development, Test, and Integration

The TwirlingHall.com Team shall transform the software design into executable code, conduct unit testing of each unit, and conduct integration testing of all units.

RESPONSIBILITY: Software Development Team, Testing Team

ENTRANCE CRITERIA: Approved SRD and URD for Integration Test Plans/Procedures; successful completion of SDR checklist to begin Coding and Unit Testing

INPUTS: Approved SDD and SIDD

TASKS:

1. Develop each software unit and database and document in Software Development Files

2. Develop unit test procedures and data for testing each software unit

4. Evaluate code and test results considering this criteria: traceability to requirements, external consistency with requirements, internal consistency, test coverage of units, appropriateness of coding methods and standards used, feasibility of integration, testing, operation, and maintenance.
6. Document integration plans and procedures in the Integration Test Plan/Procedures (ITP/P)
7. Conduct integration tests and document results in the Integration Test Report (ITR)
8. Evaluate plans and tests against criteria: traceability to requirements, external consistency with requirements, internal consistency, test coverage of requirements, appropriateness of test standards and methods used, conformance to expected results, feasibility of software qualification testing, operation, and maintenance.
10. Conduct Test Readiness Review (TRR)
OUTPUTS: Integrated, executable code, ITP/P; ITR
EXIT CRITERIA: Successful completion of TRR checklist

2.3.5 PHASE 4: System Qualification Test and Delivery

The TwirlingHall.com Team shall conduct qualification testing for each software item, integrate software configuration items with hardware configuration items, and conduct system qualification testing.
RESPONSIBILITY: Testing Team
ENTRANCE CRITERIA: Approved SRD, UDD for Qualification Test Plans/Procedures; successful completion of TRR checklist to begin Qualification Testing
INPUTS: Code package that has passed integration testing, SRD, UDD
TASKS:
1. Document system qualification plans and procedures in the Qualification Test Plan/Procedures (QTP/P
2. Integrate the software with hardware configuration items, manual operations, and other systems as necessary, into the TwirlingHall.com system.
3. Conduct system qualification tests and document results in the Qualification Test Report (QTR)
4. Evaluate test plans and tests against criteria: traceability to requirements, external consistency with requirements, internal consistency, test coverage of requirements, appropriateness of test

standards and methods used, conformance to expected results, feasibility of system qualification testing, operation, and maintenance.

5. Conduct Peer Reviews on QTP/P and QTR

6. Conduct Software Usability Review (SUR)

7. Prepare product for installation

OUTPUTS: Integrated, executable system; QTP/P; QTR

EXIT CRITERIA: Successful completion of SUR

2.3.6 PHASE 5: Support of Installation and Use

The TwirlingHall.com Team shall install the completed system at the designated user sites and conduct lifecycle support as necessary.

RESPONSIBILITY: TwirlingHall.com Project Manager, supported by members of the software development team and testing team

ENTRANCE CRITERIA: Successful completion of SUR checklist

INPUTS: Integrated, executable system, URD

TASKS:

1. Support installation of the TwirlingHall.com.

2. Support acceptance testing as needed.

3. Provide life-cycle support through implementation of Engineering Change Requests.

5. Assist with retirement or replacement of the system as needed.

OUTPUTS: Executable system in place through retirement

EXIT CRITERIA: Successful retirement/replacement of system

2.3.7 Software Configuration Management

The TwirlingHall.com Project will apply administrative and technical procedures throughout the software life cycle to identify, define, and baseline software items; control modifications and releases of the items; record and report the status of the items and modification requests; ensure the completeness, consistency, and correctness of the items; and control storage, handling, and delivery of the items. A Software Configuration Control Board (SCCB), led by the Sponsor, will oversee configuration management activities. The Configuration Management team will develop a TwirlingHall.com SCM Plan with details of the following activities.

2.3.7.1 Configuration Identification

The CM Team will assign unique identifiers to TwirlingHall.com program code and related documentation that specify the configuration item and its version.

2.3.7.2 Configuration Control

Software products will be baselined under CM control when they have completed the Peer Review. Change requests to baselined products will be identified and recorded by the CM Team. The SCCB will evaluate and approve or disapprove changes. An audit trail shall be maintained by the CM team to trace each modification to baselined products, and its reason and authorization.

2.3.7.3 Configuration Status Accounting

The CM Team will prepare records and status reports as needed to show the status and history of controlled items.

2.3.7.4 Configuration Audits

The CM Team will evaluate the functional completeness of the software items against their requirements and the physical completeness of the software item (whether their design and code reflect an up-to-date technical description).

2.3.7.5 Release Management and Delivery

The release and delivery of TwirlingHall.com software products and documentation will be formally controlled. Master copies of code and documentation will be maintained for the life of the TwirlingHall.com system.

2.3.8 Software Quality Assurance

The TwirlingHall.com Project will apply a quality assurance process to provide adequate assurance that the software products and processes in the project life cycle conform to their specified requirements and adhere to their established plans. The Quality Assurance Team will prepare and execute a TwirlingHall.com SQA Plan containing quality standards, methodologies, procedures, tools, resources, schedules, and responsibilities for performing the following activities.

2.3.9 Project Reviews

The purpose of peer reviews and management reviews is to provide management with tracking and oversight of the progress of software development undertaken by the TwirlingHall.com Project and fulfillment of requirements. Timely technical and management reviews at the appropriate level of detail facilitate information reporting and interchange that tracks progress against plans, identify and resolve action items, and verify appropriate expenditure of assigned resources.

2.3.9.1 Peer Reviews

Peer Reviews (Joint Technical Reviews) will be held on work products using guidelines in the SSC SD Peer Review Process, Version 1.0, 22 June 1998, based on Table 2.3.

Table 2.3: Peer Review Process

TwirlingHall.com Work Product	**Type of Peer Review**
Software Development Plan (SDP)	Formal Inspection
Software Requirements Description (SRD)	Formal Inspection
Preliminary User Documentation Description (UDD)	Formal Inspection
Software Design Description (SDD)	Formal Inspection
Software Interface Design Description (SIDD)	Formal Inspection
Final User Documentation Description (UDD)	Formal Inspection
Code of Software Units	Walkthrough
Integration Test Plans/Procedures	Walkthrough
Integration Test Report	Technical review
Source Code Record (SCR)	Technical review
Qualification Test Plan/Procedures	Walkthrough
Qualification Test report	Technical review

2.3.9.2 Management Reviews

The following reviews will be held using guidelines in Annex G of IEEE/EIA 12207.2 and IEEE Std. 1028. Attendees shall include Suppliers, the TwirlingHall.com Project Manager, the Hardware Manager, the Software Project Manager, and others designated by the NISBS Project Manager. The focus will be on completion of review checklists derived from the Software Management for Executives Guidebook, and metrics collected in accordance to the Project Measurement Plan.

- System Requirements Review (SRR)
- Software Design Review (SDR)
- Test Readiness Review (TRR)
- Software Usability Review (SUR)

2.3.9.3 Status Reviews

Project Status Reviews among all project members will be held at least biweekly. Monthly status meetings with suppliers will be held by the TwirlingHall.com Project Manager and other designated individuals. The focus of these informal reviews will be the status of current processes and completion status of Management Review checklists.

2.4 Schedules and Activity Network

2.4.1 Schedules

TwirlingHall.com Formatter development activities are shown in Table 2.4.

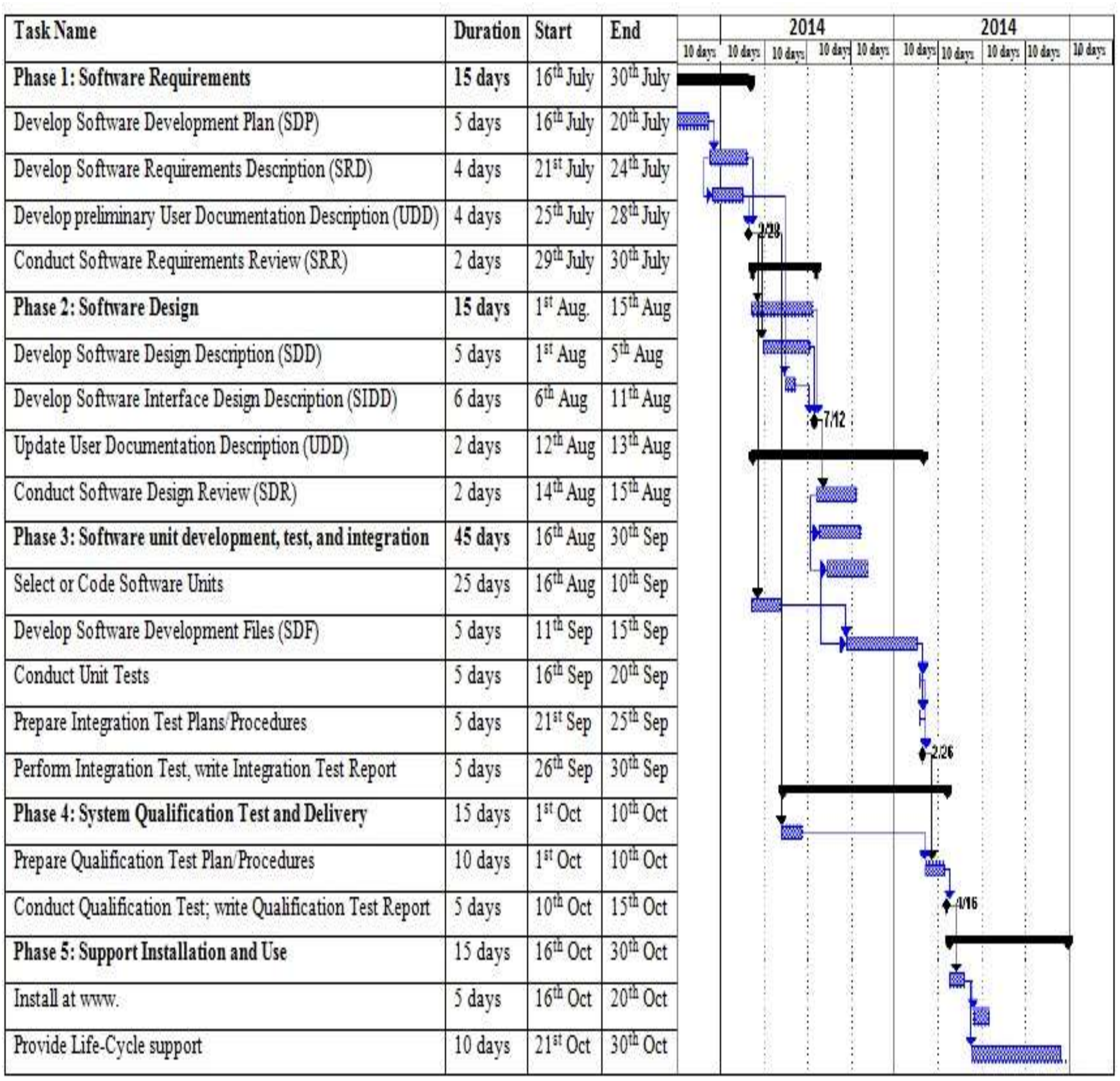
Table 2.4: Project Timeline

Task Name	**Duration**	**Start**	**End**
Phase 1: Software Requirements	**15 days**	16th July	30th July
Develop Software Development Plan (SDP)	5 days	16th July	20th July
Develop Software Requirements Description (SRD)	4 days	21st July	24th July
Develop preliminary User Documentation Description (UDD)	4 days	25th July	28th July
Conduct Software Requirements Review (SRR)	2 days	29th July	30th July
Phase 2: Software Design	**15 days**	1st Aug.	15th Aug

Develop Software Design Description (SDD)	5 days	1st Aug	5th Aug
Develop Software Interface Design Description (SIDD)	6 days	6th Aug	11th Aug
Update User Documentation Description (UDD)	2 days	12th Aug	13th Aug
Conduct Software Design Review (SDR)	2 days	14th Aug	15th Aug
Phase 3: Software unit development, test, and integration	**45 days**	16th Aug	30th Sep
Select or Code Software Units	25 days	16th Aug	10th Sep
Develop Software Development Files (SDF)	5 days	11th Sep	15th Sep
Conduct Unit Tests	5 days	16th Sep	20th Sep
Prepare Integration Test Plans/Procedures	5 days	21st Sep	25th Sep
Perform Integration Test, write Integration Test Report	5 days	26th Sep	30th Sep
Phase 4: System Qualification Test and Delivery	15 days	1st Oct	10th Oct
Prepare Qualification Test Plan/Procedures	10 days	1st Oct	10th Oct
Conduct Qualification Test; write Qualification Test Report	5 days	10th Oct	15th Oct
Phase 5: Support Installation and Use	15 days	16th Oct	30th Oct
Install at www.	5 days	16th Oct	20th Oct
Provide Life-Cycle support	10 days	21st Oct	30th Oct

2.4.2 Milestone Chart

A Schedule and Milestone chart is contained in Figure 2.2



Task Name	Duration	Start	End
Phase 1: Software Requirements	**15 days**	16th July	30th July
Develop Software Development Plan (SDP)	5 days	16th July	20th July
Develop Software Requirements Description (SRD)	4 days	21st July	24th July
Develop preliminary User Documentation Description (UDD)	4 days	25th July	28th July
Conduct Software Requirements Review (SRR)	2 days	29th July	30th July
Phase 2: Software Design	**15 days**	1st Aug.	15th Aug
Develop Software Design Description (SDD)	5 days	1st Aug	5th Aug
Develop Software Interface Design Description (SIDD)	6 days	6th Aug	11th Aug
Update User Documentation Description (UDD)	2 days	12th Aug	13th Aug
Conduct Software Design Review (SDR)	2 days	14th Aug	15th Aug
Phase 3: Software unit development, test, and integration	**45 days**	16th Aug	30th Sep
Select or Code Software Units	25 days	16th Aug	10th Sep
Develop Software Development Files (SDF)	5 days	11th Sep	15th Sep
Conduct Unit Tests	5 days	16th Sep	20th Sep
Prepare Integration Test Plans/Procedures	5 days	21st Sep	25th Sep
Perform Integration Test, write Integration Test Report	5 days	26th Sep	30th Sep
Phase 4: System Qualification Test and Delivery	15 days	1st Oct	10th Oct
Prepare Qualification Test Plan/Procedures	10 days	1st Oct	10th Oct
Conduct Qualification Test; write Qualification Test Report	5 days	10th Oct	15th Oct
Phase 5: Support Installation and Use	15 days	16th Oct	30th Oct
Install at www.	5 days	16th Oct	20th Oct
Provide Life-Cycle support	10 days	21st Oct	30th Oct

Figure 2.2: Milestone Chart

2.5 Project Team Organization and Resources

The TwirlingHall.com team will be responsible for the software requirements analysis, design, coding, integration and testing of the delivered NISBS Formatter software.

2.5.1 Project Organization

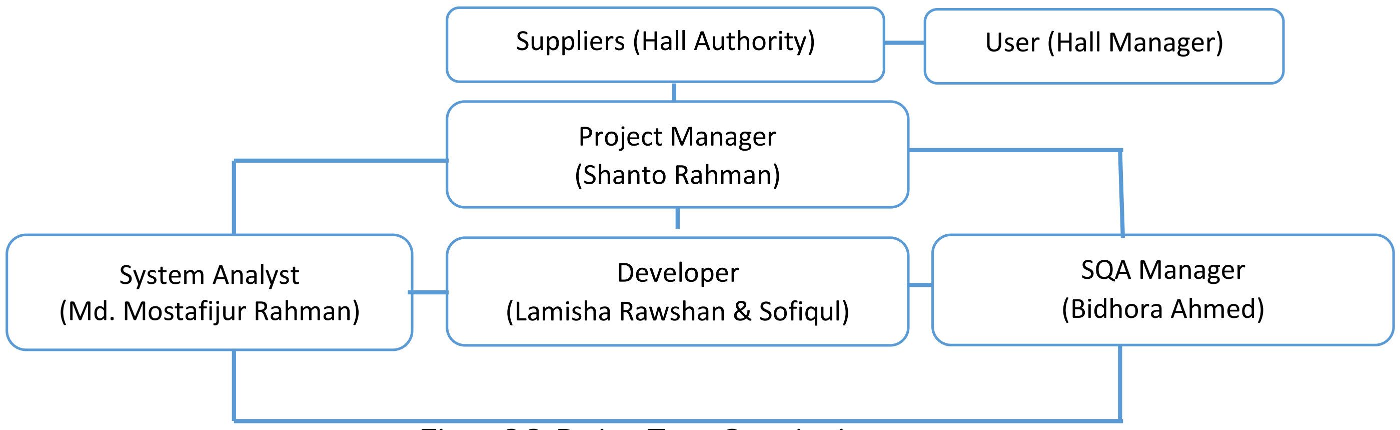

Suppliers (Hall Authority)
User (Hall Manager)
Project Manager (Shanto Rahman)
System Analyst (Md. Mostafijur Rahman)
Developer (Lamisha Rawshan & Sofiqul)
SQA Manager (Bidhora Ahmed)

Figure 2.3: Project Team Organization

Table 2.5 Project Roles and Responsibilities

Roles	**Responsibilities**
Suppliers	▪ Oversee project execution ▪ Provide approved OCD and SRS ▪ Chair SCCB
User	▪ Review OCD, SRS, UDD ▪ Conduct Qualification Testing ▪ Accept system
Project manager	▪ Oversee software development activities ▪ Manage development ▪ Develop and maintain SDP
Development Team	▪ Prepare SRD, SDD, SIDD, SDFs ▪ Code, integrate, and unit test the software ▪ Support testing and delivery
Testing team	▪ Prepare Test Plans/Procedures and Test Reports ▪ Conduct Integration testing ▪ Support Installation and Qualification testing

Chapter 3: Business Perspective

This chapter is about software business perspective. This chapter explains about the domain analysis including goal, scope, and market size, competitive product etc. This chapter also describes feasibility study including technical, social, and economical feasibility. Furthermore, business requirement document and business case are described in this chapter.

3.1 Domain Analysis

This website provides individual account for both residential and non-residential student of a university hall. It also gives necessary interfaces to add new students, store students basic information, track students' current education status.

3.1.1 General Knowledge about the Domain

TwirlingHall.com is a website that will be available to everyone. External part of the website is visible to everyone. Only students and admins have access to internal part. Admin can manage and store student information and create new event. Student has individual account so they can collect their information by using this system.

3.1.2 Scopes

The TwirlingHall.com software has the ability to display the details of the students in all departments. It enables a hall officer to add new students, modify students' basic information, search student by categories and track students' current education status. It enables students to update their basic information.

Benefits:

- Saves time.
- Reduces the manual work.
- Makes the searching work easier.

3.1.3 Potential Clients and User

Potential users are people who will interact with the system. In this system potential users are:

- Developers, who are responsible to develop the website.
- System analyst, who analyzes and design the system.
- DBA, who design and maintain database.
- Quality assurance staff, who checks the quality of the system.
- System admins, who update system information.
- Students, who collect information from the website.

3.1.4 System Environment

TwirlingHall.com is a hall management website. So, client needs a web browser and internet to access the system. They can use different platform and different browser in their computer to use this software.

3.1.5 Tasks and Procedures Currently Performed

Currently there is no software for maintain hall office work. So, officers do all their work manually. As a result there exists delay and fault in hall maintenance. So, this solution will be helpful for both hall officers and students to maintain their work.

3.1.6 Competing Software

This solution is a new idea for managing hall. There exists no system which manages hall works and help students to find their needed material like TwirlingHall.com.

3.2 Feasibility Study

Feasible study is an important tool for making right decision about a new project. The purpose of a feasibility study is to determine if a business opportunity is possible, real, and sustainable. A feasibility study permits its stockholders' to take a realistic look at both the positive and negative aspects of the proposed idea.

3.2.1 Product

The project requires a website to be developed for collecting and storing students' information of a hall. Students can gather their needed card and notice from this system.

3.2.2 Technical Feasibility

Technical feasibility is the assessment of the hardware and software needed for a project. Mainly it concentrates on achieving an understanding of the existing technical resource of project team. So, we can say that TiwirlingHall.com is technically feasible project for our team. Because there will not be much difficulty in getting resource needed for developing the project.

3.2.3 Social Feasibility

Social feasibility considers whether the proposed system would prove acceptable to the people who would be affected by the system. Hall officers and students will be affected by TwirlingHall.com. Here, hall officers will play the role of admins. So, they need training to learn how to use this system to maintain their work.

3.2.4 Economic Feasibility

Economic feasibility considers the cost/benefits of the proposed system. Project can be delivered within budget. So, this project is economically feasible.

3.2.5 Market Research

Market research indicates that this project will be beneficial for its stakeholder.

3.2.6 Alternative Solution

- **Alternative 1**

An alternative to TwirlingHall.com will be a website portal that will contain history, contact information of a student hall. All users will access this website and it will not contain any information about student.

- **Alternative 2**

Another alternative of TwirlingHall.com will be an offline database system that will be only used by hall officers to maintain their works in hall office. So, outside people and students will not have any access to this website.

3.3 Business Requirement Document (BRD)

3.3.1 Business Problem Statement

Nowadays, the rate of education learner has increased day by day. As a result, the number educational institution has increased significantly. Along with this, the numbers of hostel are also increasing. Although technology has reached in the entire sector more or less, it doesn't touch in hall management life cycle. As a result, a huge amount of peoples suffers a long run. Their daily tasks have been completed in a manual fashion which is time consuming and hampers huge amount of productivity of both students and the officials' people. Besides, any kind of information cannot be easily gained such as what number of students are currently stayed in the hall, how much seats are vacant, who are not legal student, who are capable for achieving hall scholarship etc. Moreover, many tasks are manually performed such as ID card generation, testimonial creation, managing manual noticeboard. On the other hand, if we consider the student scenario, they also have to face several difficulties such as if one student's reading lite is cut off, they had to go hall office and complaining about the issue and finally that writings has ignored by the hall office due to cut off the page. Along with this, students are involved in a large number manual task which hinders their productivity. If there would be an online solution based automated tool from which management team of the hall could easily know the current status of the student and complete the manual task without any manual process rather by only one single click they got the desired output. Considering this scenario, the need of an online based automated system has been rose which provides the several kinds of information of students and with related issues.

3.3.2 Business Scope

With the increment of education system, number of hostels is increased. But no one hostel has any automated tool by which they can mitigate their daily manual task. So here is a scope of creating a web based portal. The authority might be significantly benefitted via this system.

3.3.3 Business Objectives

- Provide an automated system which provides student information.
- Id card, testimonial creation will be automatically generated.
- Reduce hassle of both students and hall managing committee.
- Gain popularity and trust of people.
- Get benefitted financially with the increment of number of hostel

3.3.4 Impact Statement

- The hall managers will be able to know about their legal students.
- Online Seat Allocation
- Manager can easily know what seats are vacant.
- Easy to give hall scholarship
- Easy to find a student who occurs crime.
- Virtual noticeboard can reach the objectives of hall managing committees to the students
- Id card and testimonial can be easily created.
- Students need not to stand up a line.
- Reduces frequent going hassle to hall office.
- The popularity of the system will rise rapidly.
- It will save time, effort and money of customers.

3.3.5 Assumptions

- IT staffs need to be hired or trained to support the new system.
- The project will be completed within the budget.
- It would take around one year to reach break eve.
- The number of clients per day to this site will increase gradually.

3.3.6 Functional Requirements

- The non-authorized users can only see the external outlook of the system.
- The authorized admin users get the basic functionalities of admin panel, which are described above.
- It represents the details information of the student.
- Student can edit or update their information.
- Any kinds of search options which are frequently needed are also aggregated.
- It generates price list of all shops for selected products.

3.3.7 Non Functional Requirements

- The system will be reliable and scalable.
- The response time for generating the student information or id card or any kind of operation will be very small.
- The backup of the data will be available in case of system crash.
- The system will be user friendly.
- The security of the system will be satisfactory.

3.4 Business Case

3.4.1 Expenditure

3.4.1.1 Capital Expenditure

Activities	**Amount in Taka**
Domain name registration (8 domains)	(25000*8) = 200000
Acquiring hosting space	20000

Machine Procurement	100000
Total	**320000**

3.4.1.2 Operational Expenditure

3.4.1.2.1 Employee Cost

Cost of 1 man hour 1500 taka

Weekly hours per worker 36

Half yearly hour per worker 6×36 = 216

Half yearly cost per worker = 1500×216 = 324000 taka

Total Developers 5

Total cost = 5×324000 = **1620000**

3.4.1.2.2 Overall Operational Expenditure

Areas	**Amount in Taka**
Employee cost	1620000
Training	10000
Other	20000
Total	**1650000**

Total Expenditure

Areas	**Amount in Taka**
Capital Expenditure	320000
Overall Operational Expenditure	1650000
Total	**1970000**

3.4.2 Expected Profit

Table 1 show expected profit after 5 years.

Table 3.1: Expected Profit after 5 Years

Year	**Capital (Tk.)**	**Operational (Tk.)**	**Total (Tk.)**	**Earn (Tk.)**	**Profit (Tk.)**
1	320000	1650000	1970000	70000	-1900000
2	0	650600	650600	400000	-250600
3	0	348200	348200	340000	-8200
4	0	204000	204000	1354000	1150000
5	0	53200	53200	5757300	5704100

3.4.3 Break-even Analysis

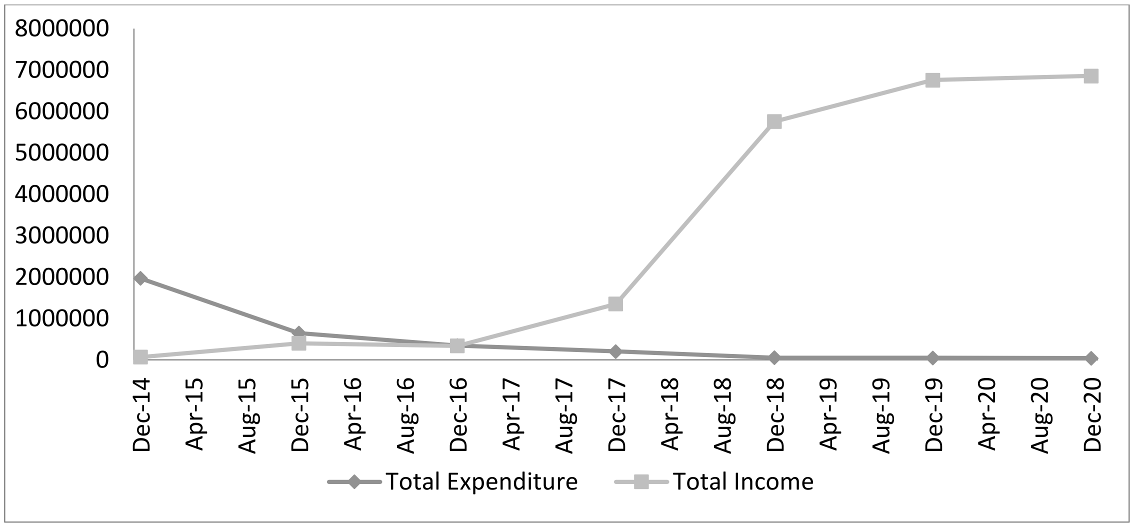


Figure 3.1: Break-even analysis

Chapter 4: Software Requirement Specification

This chapter is about software requirement specification and analysis. This chapter explains about the requirement gathering/ specification and analysis process which includes user scenario, quality function development (QFD), use case and diagram, activity diagram, swimelane diagram, state diagram, data flow diagram, UML class diagram and CRC, data modeling including E-R diagram and database schema.

4.1 Introduction

In this part of the document the software requirements specification are described briefly. First the purpose, scope and features of the product are narrated. Following this comes the use case scenario, use case diagram, activity diagram, swimlane diagram, data modeling, class responsibility collaborator, data flow diagram, state diagram and sequence diagram respectively.

4.1.1 Purpose

TwirlingHall.com aims to reduce the hassle of managing students' information manually by the hall authority. In today's global world, people are getting more conscious about their career. The education rate in Bangladesh is gradually increasing. That means more people are coming forward for the higher education. Maximum of these students come to another city leaving their home for the higher education. So they live in the halls of their respective institution. The hall authority has to manage this huge number of students' information manually. They face a lot of problem in doing this work. Sometimes they lost the file of a student which causes a great problem for both the hall authority and the student. Again in case of updating student information, the student has to stand in the line for a long time. If there is an automated system which can help to manage all these students information, it will be obviously helpful to the hall authority. As far as our knowledge, there is no such system for managing students' information. That's why, the step to build such an automated system has been taken.

This Software Requirements Specification (SRS) describes the functional and nonfunctional requirements for the website TwirlingHall.com. The requirements were developed specifically for the Kabi Sufia Kamal Hall, University of Dhaka but are believed to be suitable for the information managing process of all the halls of any institution.

4.1.2 Product Scope & Features

The TwirlingHall.com facilitates the information managing process of the tenants of hall of an institution. The scope of the system is as follows.

- Registering user accounts
- Managing students' information, statuses and privileges
- Providing facilities to generate students' ID card, boarding card and testimonial
- Tracking hall's deposit, withdraw and expense
- Providing virtual notice board

4.1.3 Intended Audience

This SRS is intended for officers of Kabi Sufia Kamal Hall, University of Dhaka. As such, it aims for a high level of readability for a non-technical audience, while providing enough specificity to be useful to a software developer.

It is assumed that when software development occurs, it will be in a highly collaborative and iterative environment in which end-users have multiple opportunities to review prototypes and refine the user interface and software functionality.

4.1.4 Document Conventions

The SRS includes requirements, process flowcharts, and use cases. Use cases are included for some of the most frequently performed activities. They are intended to supplement the requirements and highlight activities that offer a great potential for increased efficiency and ease of use. Again, they should be considered to be contextual rather than prescriptive.

4.1.5 Overview

Here the starting phase of our project is described. Mainly this section contains the ice breaking stage of the product.

4.1.5.1 Inception

Inception is the first phase of requirements engineering. In this phase the ice breaking is done. That is it helps to understand the basic part of the problem, scope of the project and the nature of the problem to be solved and the effectiveness of preliminary communication and collaboration between the other stakeholder and the software team. For establishing the ground work we have considered the following phases of inception.

i. Identifying the stakeholders
ii. Recognizing multiple viewpoints
iii. Working toward collaboration
iv. Asking to first questions

4.1.5.1.1 Identifying the Stakeholders

Stakeholders are the persons who will be affected by the system. Stakeholders include the people of the organization who are sponsoring for the project and also the end user. It is very important to identify the stakeholders correctly. For this we have asked the following question to the officers of the Sufia Kamal Hall, University of Dhaka.

i. Who is sponsoring the project?
ii. Who are the users of the project outcome?
iii. Who will provide the resources of the project?

4.1.5.1.2 Recognizing Multiple Viewpoints

We have tried to recognize multiple viewpoints like as students' viewpoint, officers' viewpoint, and sponsors' viewpoint.

Student’s viewpoint:

- Easy to use
- Attractive

Officer’s viewpoint:

- Easy to use
- Security
- Update of the info
- Hassle free

Sponsor’s viewpoint:

- Less expensive
- More features
- No manual input
- High security

4.1.5.1.3 Working towards collaboration

Different stakeholder may have different opinions about the set of requirements. For a successful system to result collaboration among stakeholders and the software team is needed. We have merged our requirements with the stakeholders. At first we have identified the common and conflicting area. Then we have analyzed the requirements on the basis of the stakeholders’ priority.

4.1.5.1.4 Asking the first questions

The first set of questions is context free which focuses on the customer and the other stakeholders, the overall project goals and benefit. These set helps to identify the stakeholders and the benefit of successful implementation of the project. The next set of questions enables to gain a better understanding of the problem and allows the customer to voice his or her perceptions about the solution.

4.1.5.2 Elicitation

Elicitation is a task that helps the customer to define what is required. To complete the elicitation step we face many problems like problems of scope, problems of volatility and problems of understanding. However, this is not an easy task. To help overcome these problems, we have worked with the Eliciting requirements activity in an organized and systematic manner.

4.1.5.2.1 Collaborating Requirements Gathering

i. Meeting with officers
ii. Meeting with students
iii. Meeting with teachers

4.1.5.2.2 Quality Function Deployment

Quality function deployment (QFD) is a quality management technique that translates the needs of the customer into technical requirements for software. QFD "concentrates on maximizing customer satisfaction from the software engineering process". QFD identifies two types of requirements.

4.1.5.2.2.1 Normal Requirements

HallREQ-101 **Priority:** 3

Name: Admin Login System

Description: Support for Admin login accounts; access to modules is granted by use of "roles" or "privileges" that allow Admin access to as many modules as needed.

HallREQ-102 **Priority:** 3

Name: Uploading Employee Information from Excel to Database

Description: Admin can load the excel file into the Database provided that the excel file contains current student information.

HallREQ-103 **Priority:** 3

Name: Automatic Id Card generation depends on student type such as resident, non-resident etc. Along with this, testimonial generation.

Description: Id card will be generated dynamically based on the student type.

HallREQ-104 **Priority:** 3

Name: Different kinds of Search options.
Description: Search options are needed so that in any time it is easily possible to get any kinds of students.

HallREQ-105 **Priority:** 3

Name: Student Authentication.
Description: Students also have to be a valid profile page by which they can easily get all the notice or other thins.

HallREQ-106 **Priority:** 3

Name: Automatic Id Card generation depends on student type such as resident, non-resident etc. Along with this, testimonial generation.

Description: Id card will be generated dynamically based on the student type.

HallREQ-107 **Priority:** 3

Name: Virtual Noticeboard.

Description: In any time admin can send any notice to the students. After sending a notice all students will got a notification

HallREQ-108 **Priority:** 3

Name: Several kinds of student permission will be set by admin such as student info update permission or other things.

Description: Not all the times students are capable for updating their information rather admin will decide when a student will capable for updating their info.

HallREQ-109 **Priority:** 3

Name: Online Seat Allocation.

Description: Seat allocation will be occurred via online.

HallREQ-110 **Priority:** 3

Name: Student Activate an deactivate options
Description: Admin will be able to activate or deactivate a student so that it can be easily traced who are the legal students or not

HallREQ-111 **Priority:** 3

Name: Several kinds of Reports need to be generated.

Description: Reports are needed to generate based on the student information.

4.1.5.2.2.2 Expected Requirements

This project is for Hall used to manage hall activities. So, for each role change password options are expected. Along with this a nice profile is also expected to the customers.

4.1.5.2.2.3 Exciting Requirements

Now, this project is for a specific hall like Kabi Sufia Kamal hall and Surjasen hall to manage their student activities. So, we have to strictly follow the requirements provided by the hall. If we are able to provide some features those excite the hall Stakeholders, we will include those requirements later in the documentation.

4.2 Overall Description

The following sections contain an overall idea of the product. We have described here the product perspective, features, user classes and the non-functional requirements of the product.

4.2.1 Product Perspective

TwirlingHall.com is a system that performs the functions of managing students' information by the hall authority including students' profile, generation of Id card, boarding card, testimonial of the students', searching facility, virtual noticeboard and keeping track of different accounting activities like as deposit, expense and withdraw of the hall.

4.2.2 Product Features

TwirlingHall.com provides online managing information facilities to the officers. The functions of the system include the system providing different type of services based on the type of users [Officer/Student].

- Creation of students' account, allocation of students; room number, generation of ID card, boarding card and testimonial.
- Students' information can be searched in the system in a variety of ways such as by the departments, by room numbers, by blood group, by session year.
- Students can apply for different things such as sit allocation, scholarship from the hall authority, change of utensils of room, ID card, testimonial etc.
- System sends a notification to the students in case of a new post in the noticeboard.
- Students can update their contact information in the system.
- Admin can use accounting module in terms of recording withdraw, expense and deposit of the hall.

4.2.3 User Classes and Characteristics

The TwirlingHall.com is a student management system for managing students' information faster, easier, and more convenient than manual work. Consequently, the application will have little or no learning curve, and the user interface will be as intuitive as possible. Thus, technical expertise should not be an issue. Instead, anticipated users can be defined by how they will use the product in a particular situation.

4.2.4 Non Functional Requirements

The non-functional requirements include different safety and security requirements, software quality attributes, different constraints like as the hardware and software constraints etc.

4.2.4.1 Safety Requirements

The database may get crashed at any certain time due to virus or operating system failure. Therefore, it is required to take the database backup.

4.2.4.2 Security Requirements

We are going to develop a secured website for the institute .There are different categories of users namely students, administrators etc. Depending upon the category of user the access rights are decided. It means if the user is an administrator then he can be able to modify the data, delete, append etc., all other users only have the rights to retrieve the information about database.

4.2.4.3 Software Quality Attributes

The Quality of the system is maintained in such a way so that it can be very user friendly to all the users of the system.

4.2.4.4 Hardware Constraints

The system requires a database in order to store persistent data. The database should have backup capabilities.

4.2.4.5 Software Constraints

The development of the system will be constrained by the availability of required software such as database and development tools. The availability of these tools will be governed by.

4.2.4.6 Design and Implementation Constraints

The information of all the users must be stored in a database that is accessible by the TwirlingHall.com. The institute information security system must be compatible with the Internet applications. The website is connected to the institute computer and is running all 24 hours a day. The users access the website from any computer that has Internet browsing capabilities and an Internet connection. The users must have their correct usernames and passwords to enter into the TwirlingHall.com.

4.3 User Documentation

4.3.1 Benefits of this SRS

The benefits of the SRS are as follows.

a. Force the use to consider specific requirements carefully.
b. Enhances communication between the customer and System developers.
c. Provides a firm foundation for the system design phase.
d. Enable planning of validation, verification and acceptance procedures.
e. Enable project planning, for example estimates the cost and time, resource scheduling.
f. Useable during maintenance phase.

4.4 Use Case Scenario

The TwirlingHall.com is a student management web portal for managing students' or tenants information of different halls of any institution. This system helps to diminish the aggravations of storing and manipulating students' information manually.
The system has two types of users – Admin and general user or student. Admin will maintain and coordinate all the activities, and users can register for their intended account and get related information through this portal.

The access of Admin user is restricted through an authentication system. There will be a separate logging interface for the admin, where he/she will fill up some required fields (for example: E-mail and password). Initially, default e-mail and password will be provided to the admin user.
After logging in, the system will give acquiescence to the admin for doing some predefined maneuver such as creating students' accounts, allocating students' room number, generating ID cards, generating the testimonial, posting new notice in the noticeboard etc.

Admin can create students' profile in two different ways. The profile for the newly admitted students can be created through an excel sheet in which information are collected from the registered office. These are then stored in the database. Another way is to create the profile manually. The system provides this preference for creating the profile of the students' who are already the residents or attached to the hall. The students' profile who have completed their study and left the hall will be inactivated by the admin. They can add any new department, room, faculty or institution in the system. There is an option to create new events in the system so that the students can easily know about these and participate in the events.

Students have the option for editing their profile. They can update their current status such as studying year, room number, contact information, etc. They can apply for different stuffs here such as scholarship from the hall authority, change of light or other things of room, ID card, testimonial etc. also. The system will send a notification to the students in case of a new post in the noticeboard. They can view the notice from here.

Students' information can be searched in the system in a variety of ways. One way is to search the students' by their name. The other ways are to search them by the departments, by room number, by blood group, by session year.

There is an accounting module in the system. This module includes deposit, withdraw and the expense of the hall. Total deposit of the hall is to be entered in the deposit module. Similarly admin can input the total amount of withdraw and expense of the hall in withdraw and expense module respectively. At the end of a month or year, the sum of the deposit, withdraw and expense can be seen form the system.

4.4.1 Use case

Table 4.1: Use Case

Level 0	Level 1	Level 2	Actor
TwirlingHall.com	Authentication	Sign up Sign in Sign out Change Password	Admin Student
	Student Creation	Creating Student Update Student Information Deactivate student Register Student	Admin
	Boarding Card generation	Generate Boarding Card	Admin
	ID Card Generation	Generate ID Card	Admin
	Testimonial Generation	Generate Testimonial	Admin
	Search Student	Searching	Admin Student
	Event Creation	Create Event	Admin Student
	Upload Excel	Upload Excel File	Admin
	Notice Board	Create Notice	Admin

1. Use Case for Authentication

Use case	Authentication
Use case no	1.1
Primary actor	Admin, Student
Description	Allowing user to log in to the system
Precondition	Appropriate user ID & password must be obtained
Trigger	Admin decides to authenticate

2. Use Case for Student Creation

Use case	Student Creation
Use case no	1.2
Primary actor	Admin
Description	Create new student into the system
Precondition	Appropriate user ID & password must be obtained
Trigger	Admin decides to create new student

3. Use Case for Boarding Card Generation

Use case	Boarding card generation
Use case no	1.3
Primary actor	Admin
Description	Create Boarding card for hall student
Precondition	Appropriate user ID & password must be obtained
Trigger	Admin decides to boarding card generation

4. Use Case for ID Card Generation

Use case	ID card generation
Use case no	1.4
Primary actor	Admin
Description	Create ID card for hall student
Precondition	Appropriate user ID & password must be obtained
Trigger	Admin decides to ID card generation

5. Use Case for Testimonial Generation

Use case	Testimonial generation
Use case no	1.5
Primary actor	Admin
Description	Create Testimonial card for hall student
Precondition	Appropriate user ID & password must be obtained
Trigger	Admin decides to testimonial generation

6. Use Case for Search Student

Use case	Search student
Use case no	1.6
Primary actor	Admin
Description	Searching student into the system
Precondition	Appropriate user ID & password must be obtained
Trigger	Admin decides to searching student

7. Use Case for Event Creation

Use case	Event Creation
Use case no	1.7
Primary actor	Admin
Description	Generate new event into the system
Precondition	Appropriate user ID & password must be obtained
Trigger	Admin decides to generate new event

8. Use Case for Upload Excel

Use case	Upload excel file
Use case no	1.8
Primary actor	Admin
Description	Upload excel file into the system
Precondition	Appropriate user ID & password must be obtained
Trigger	Admin decides to upload excel file

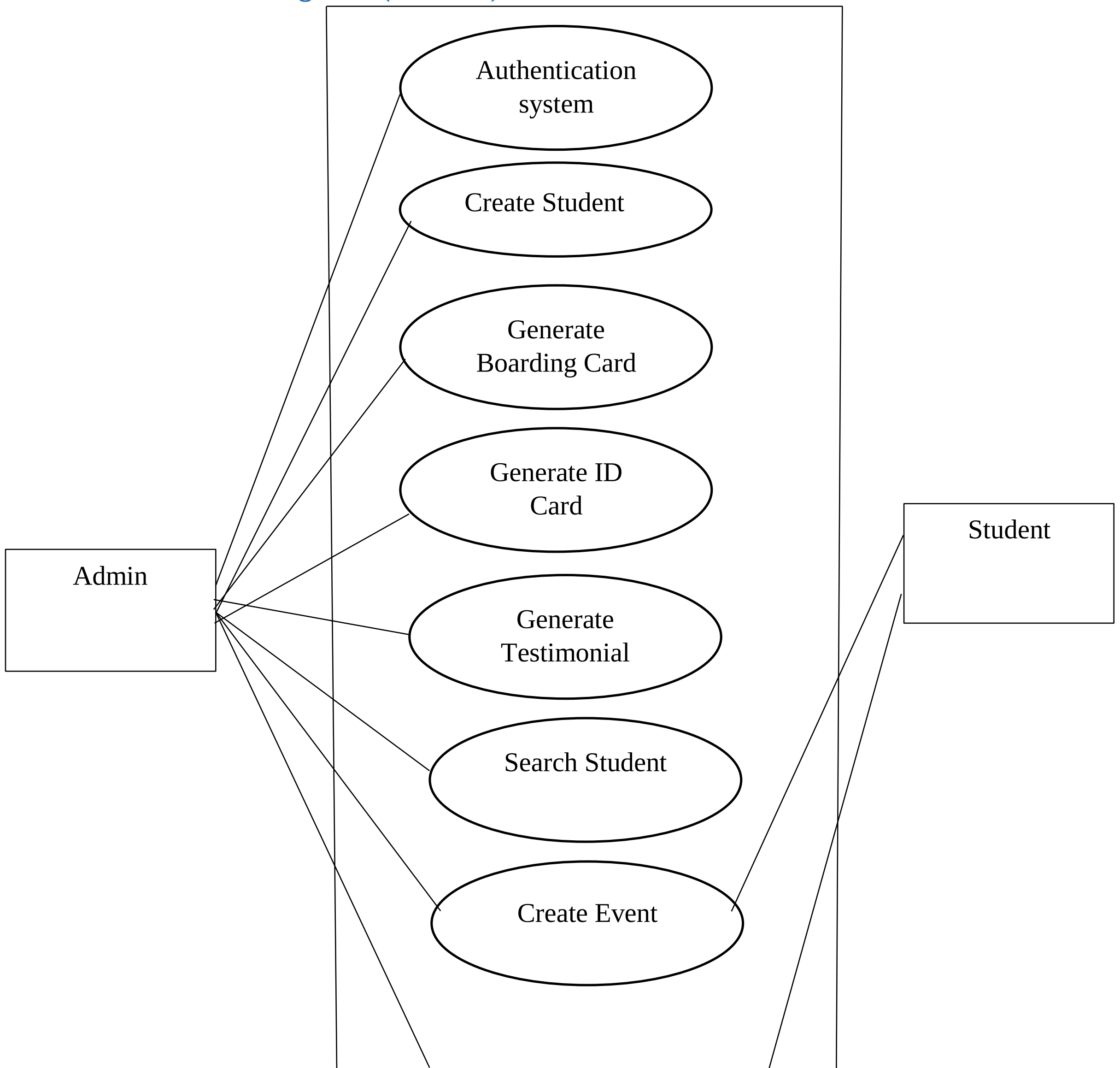
9. Use Case for Notice Board

Use case	Notice board
Use case no	1.9
Primary actor	Admin
Description	Create new notice &submit into the notice board
Precondition	Appropriate user ID & password must be obtained
Trigger	Admin decides to create new notice

4.4.2 Use case diagram

Here is use case diagram for the whole system. This diagram shows the overview of the integration of the user and the system:

4.4.2.1 Use Case Diagram (Level 0)



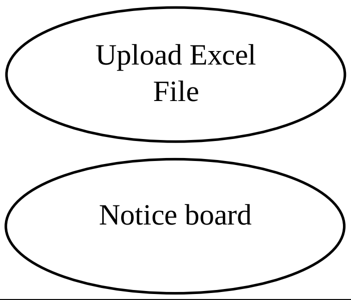


FIGURE 4.1: USE CASE DIAGRAM LEVEL 0

4.4.2.2 Use Case Diagram (Level 1.1)

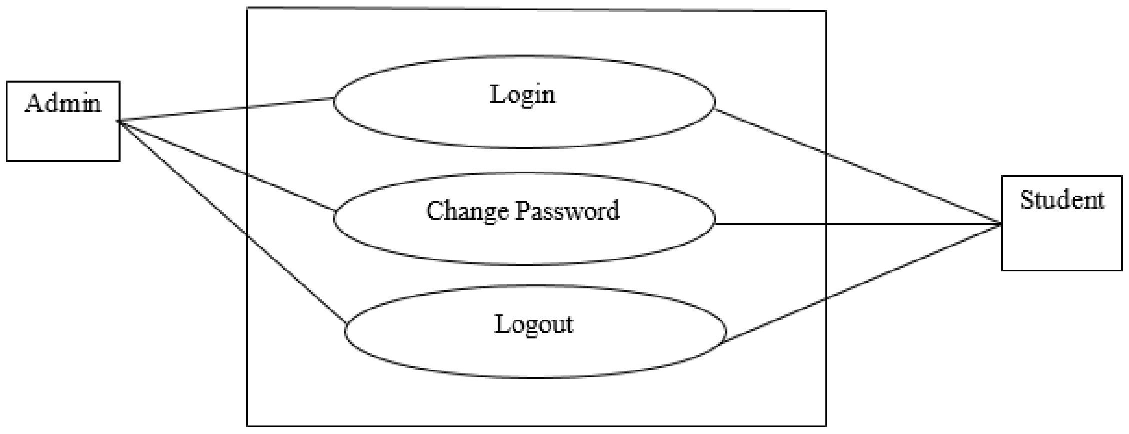


Figure 4.2: Use Case Diagram Level 1.1

4.4.2.3 Use Case Diagram (Level 1.2)

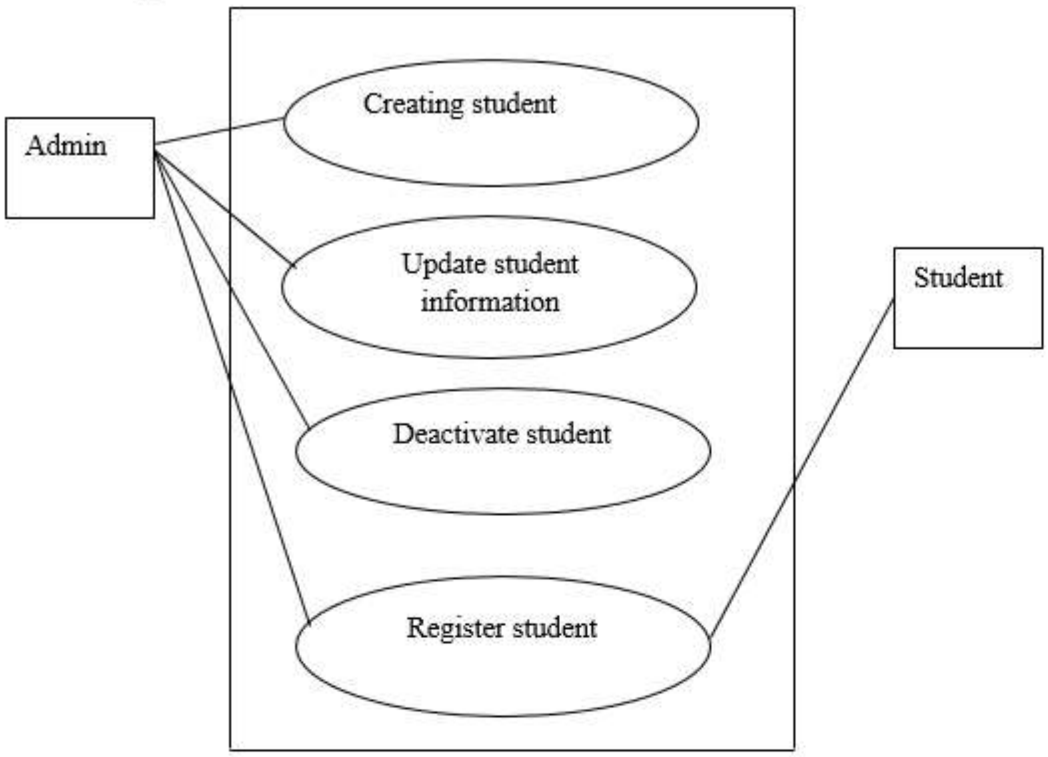


Figure 4.3: Use Case Diagram Level 1.2

4.4.2.4 Use Case Diagram (Level 1.3)

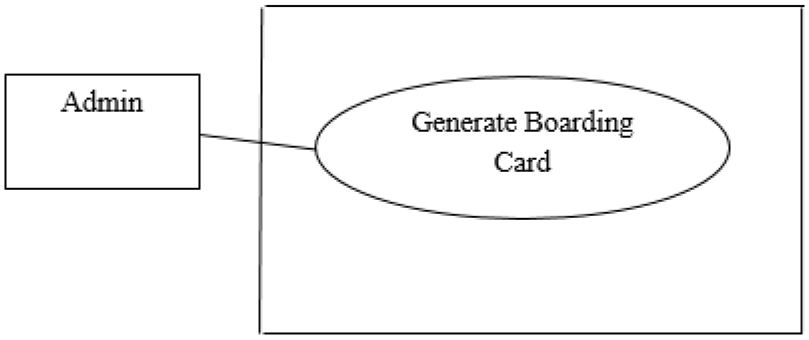


Figure 4.4: Use Case Diagram Level 1.3

4.4.2.5 Use Case Diagram (Level 1.4)

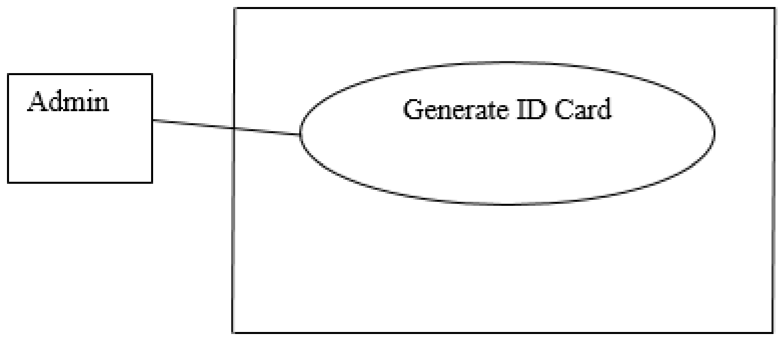


Figure 4.5: Use Case Diagram Level 1.4

4.4.2.6 Use Case Diagram (Level 1.5)

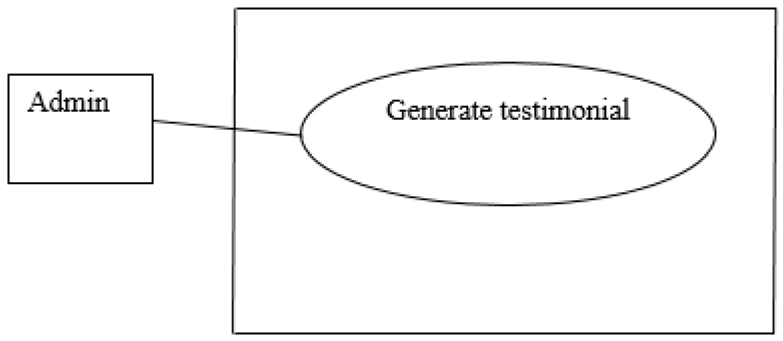


Figure 4.6 Use Case Diagram Level 1.5

4.4.2.7 Use Case Diagram (Level 1.6)

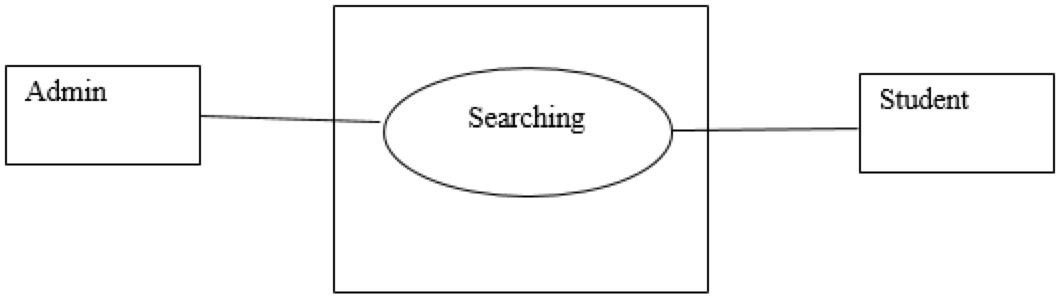


Figure 4.7: Use Case Diagram Level 1.6

4.4.2.8 Use Case Diagram (Level 1.7)

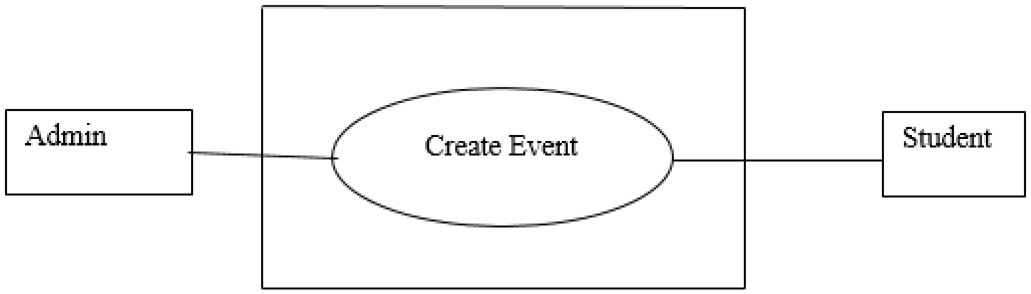


Figure 4.8: Use Case Diagram Level 1.7

4.4.2.9 Use Case Diagram (Level 1.8)

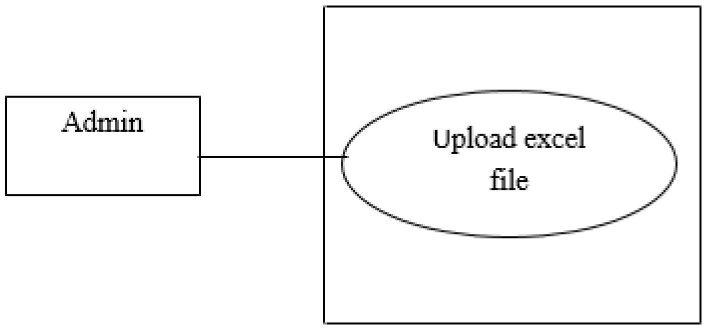


Figure 4.9 Use Case Diagram Level 1.8

4.4.2.10 Use Case Diagram (Level 1.9)

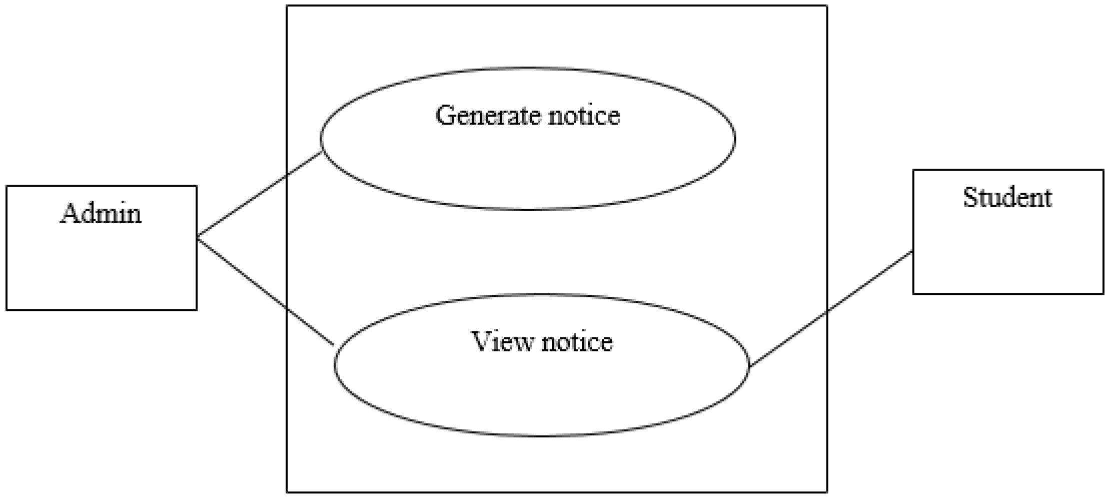


Figure 4.10 Use Case Diagram Level 1.9

4.4.3 Activity Diagram

The activity diagram for all the use cases are as follows.

4.4.3.1 Activity diagram for Authentication

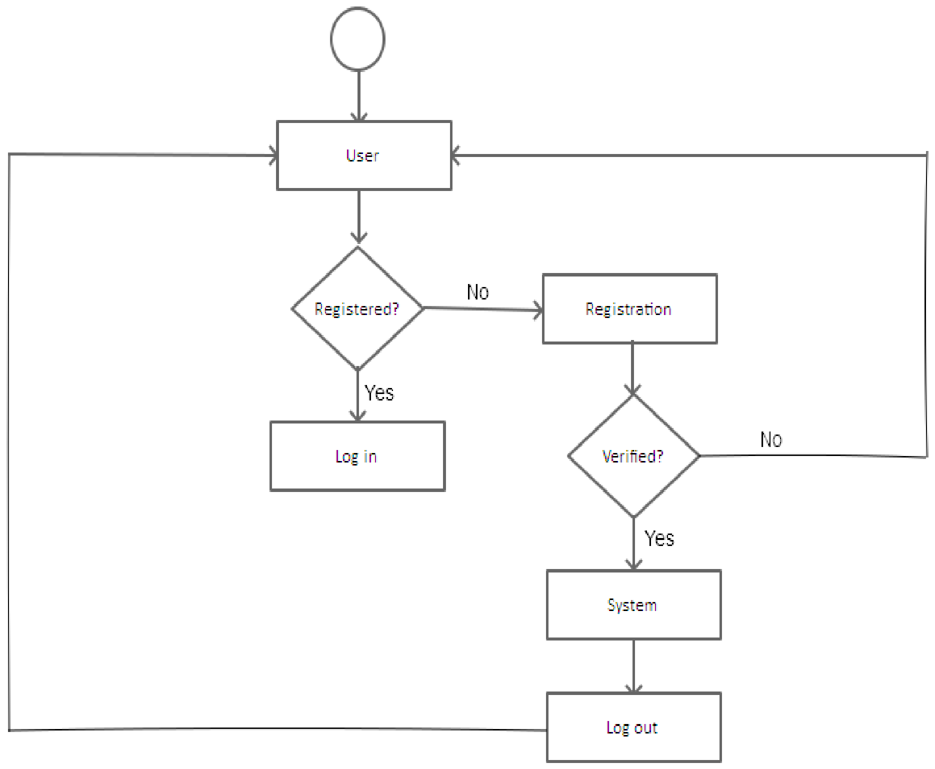


Figure 4.11: Activity Diagram for Authentication

4.4.3.2 Activity Diagram for Student Creation

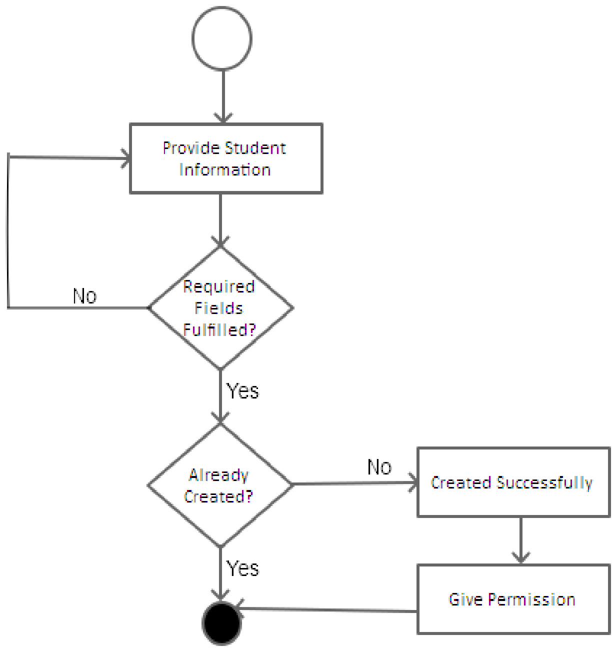


Figure 4.12: Activity Diagram for Student Creation

4.4.3.3 Activity Diagram for Boarding Card Generation

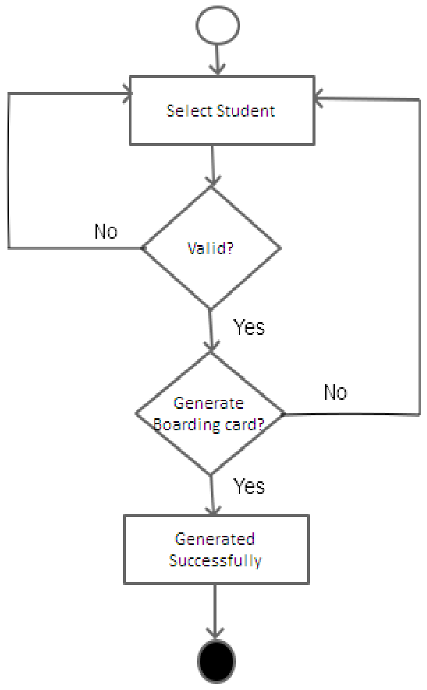


Figure 4.13 Activity Diagram for Boarding Card Generation

4.4.3.4 Activity Diagram for ID Card Generation

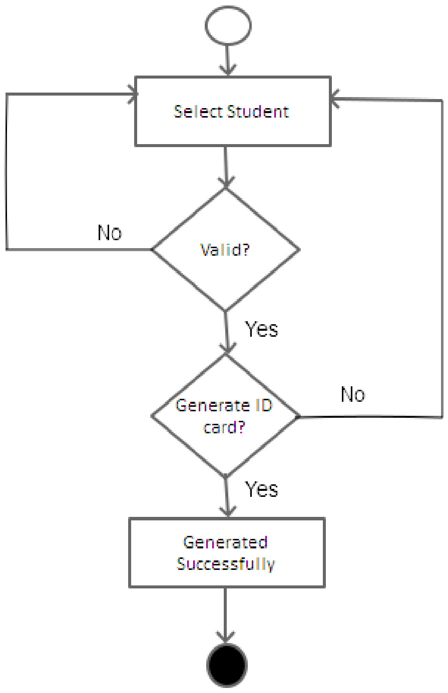


Figure 4.14 Activity Diagram for ID Card Generation

4.4.3.5 Activity Diagram for Testimonial Generation

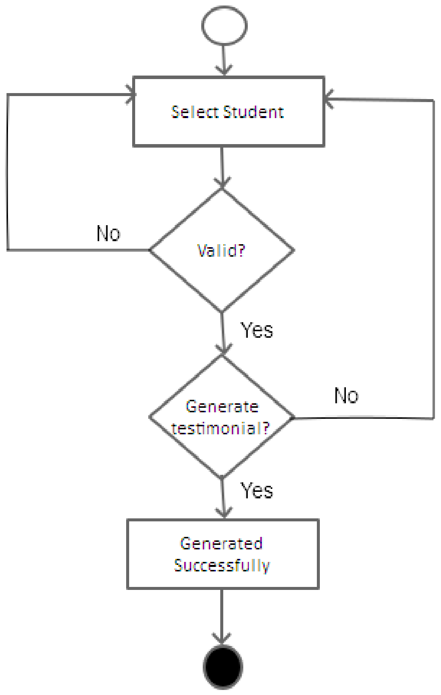


Figure 4.15: Activity Diagram for Testimonial Generation

4.4.3.6 Activity Diagram for Search Student

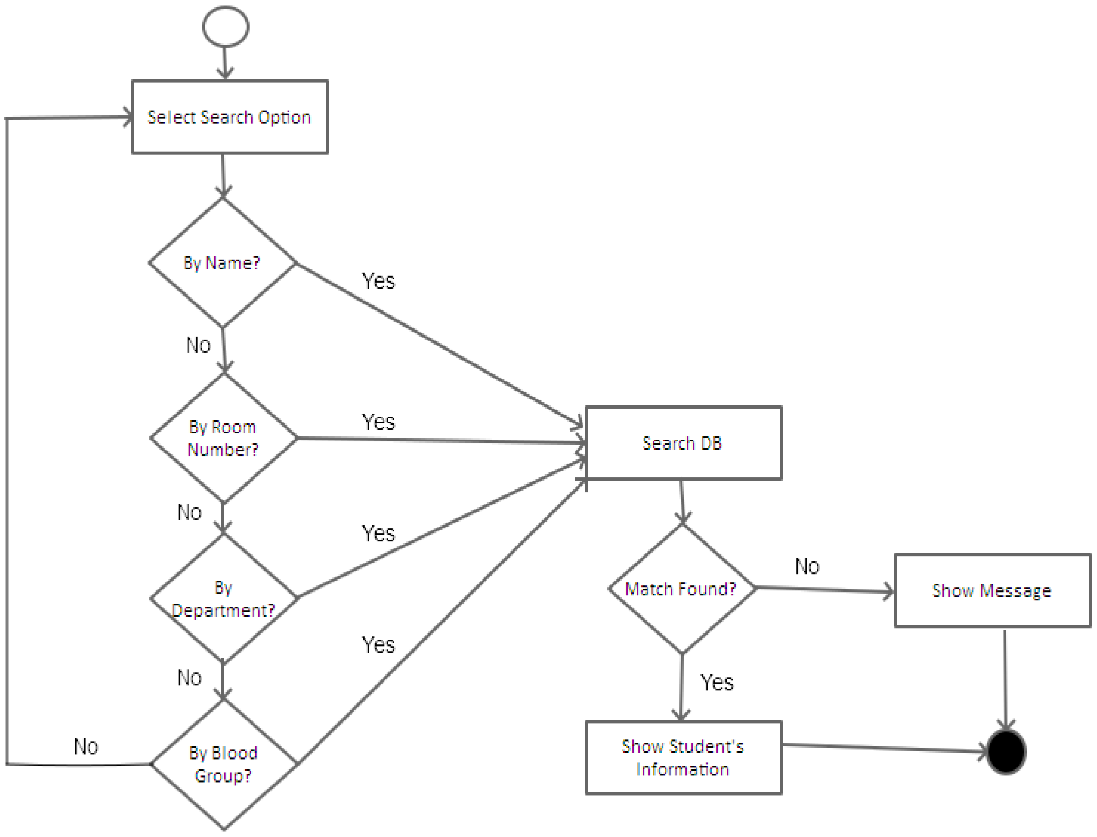


Figure 4.16 Activity Diagram for Search Student

4.4.3.7 Activity Diagram for Event Creation

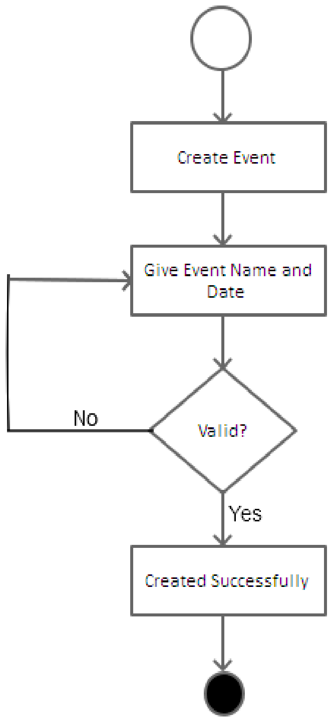


Figure 4.17: Activity Diagram for Event Creation

4.4.3.8 Activity Diagram for Upload Excel

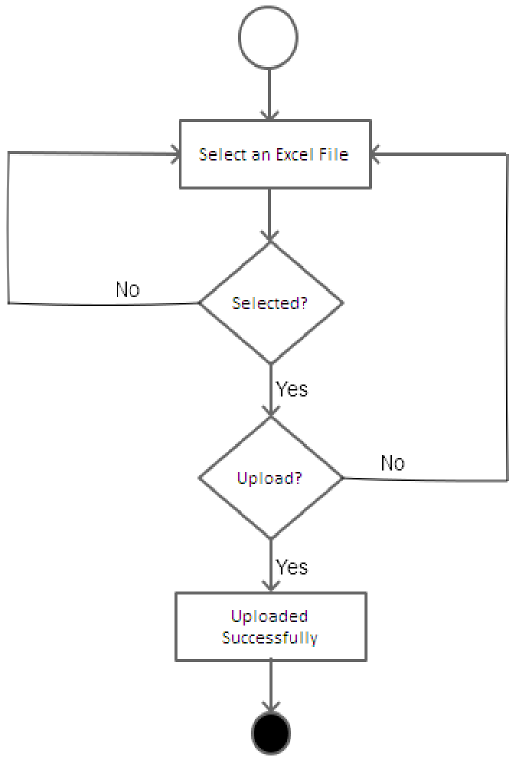


Figure 4.18: Activity Diagram for Upload Excel

4.4.3.9 Activity Diagram for Notice Board

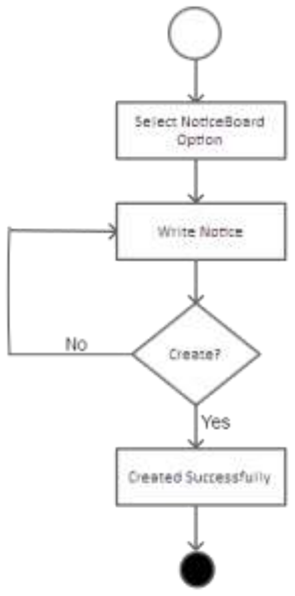


Figure 4.19: Activity Diagram for Notice Board

4.4.4 Swimlane Diagram

Here are the swimlane diagrams for the usecases.

4.4.4.1 Swimlane Diagram for Authentication

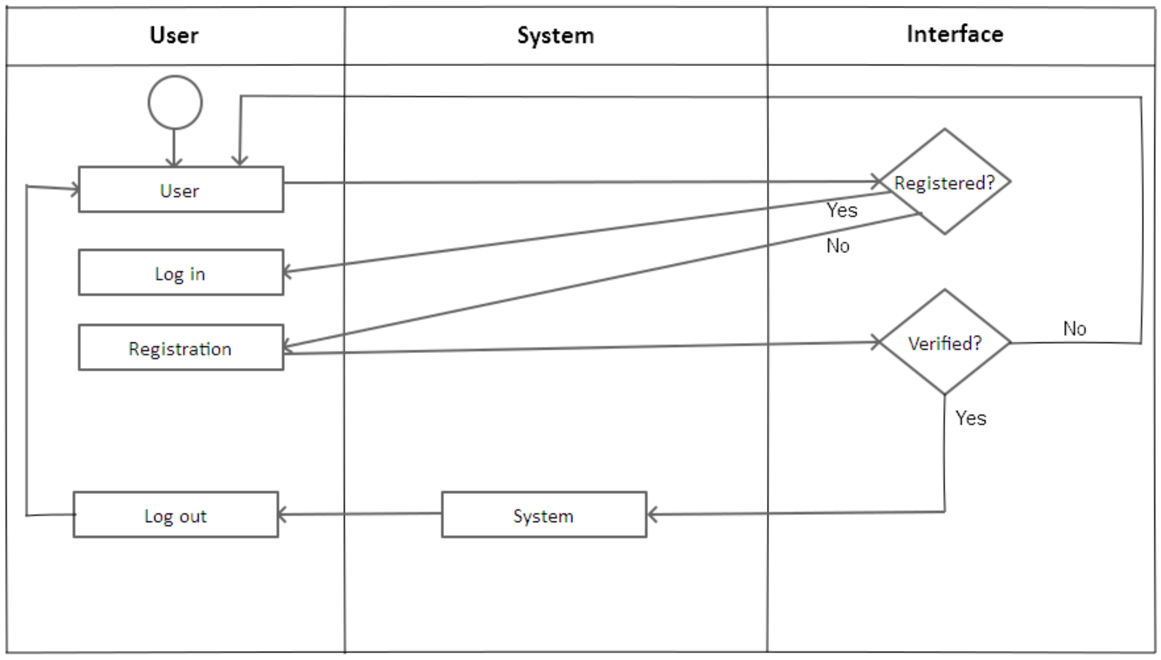


Figure 4.20 Swimlane Diagram for Authentication

4.4.4.2 Swimlane Diagram for Student Creation

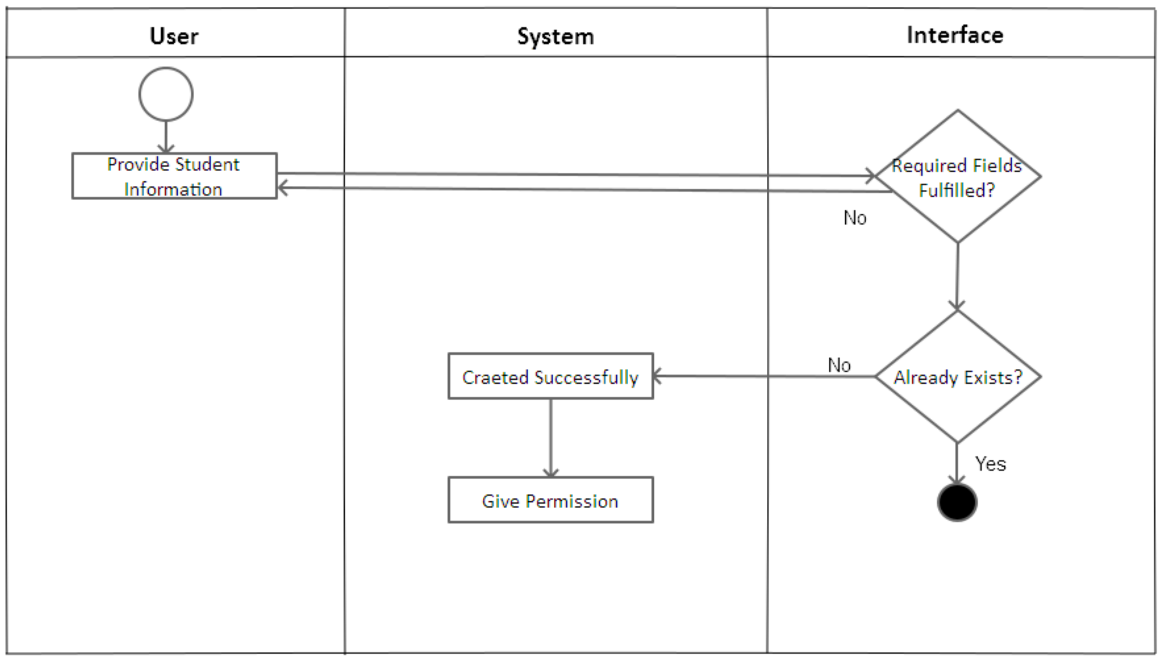


Figure 4.21 Swimlane Diagram for Student Creation

4.4.4.3 Swimlane Diagram for Boarding Card Generation

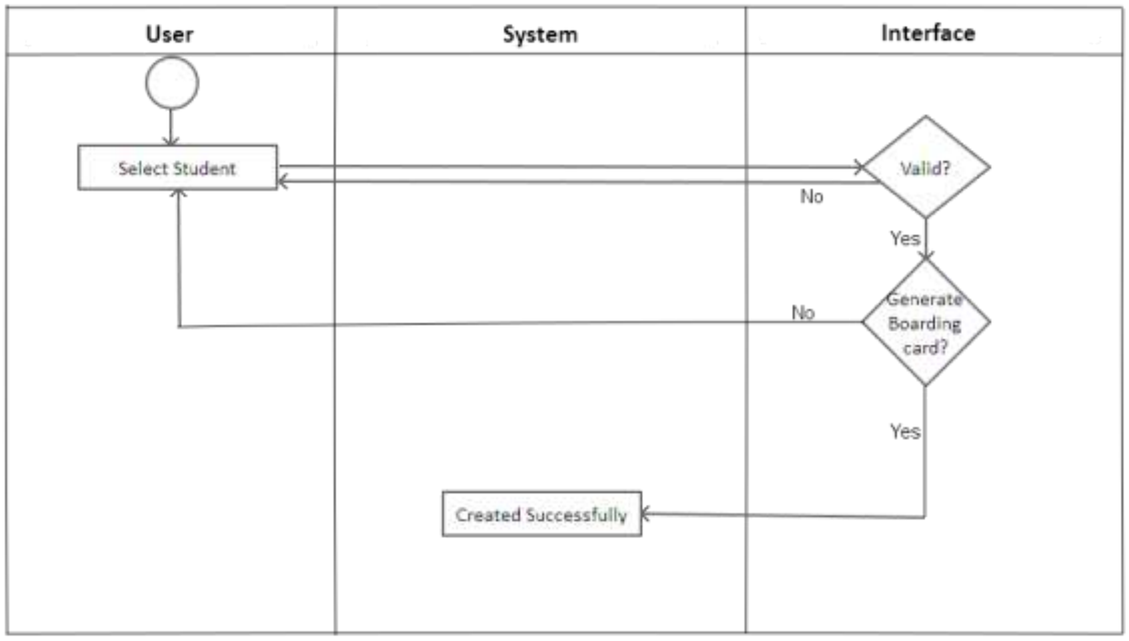


Figure 4.22 Swimlane Diagram for Boarding Card Generation

4.4.4.4 Swimlane Diagram for ID Card Generation

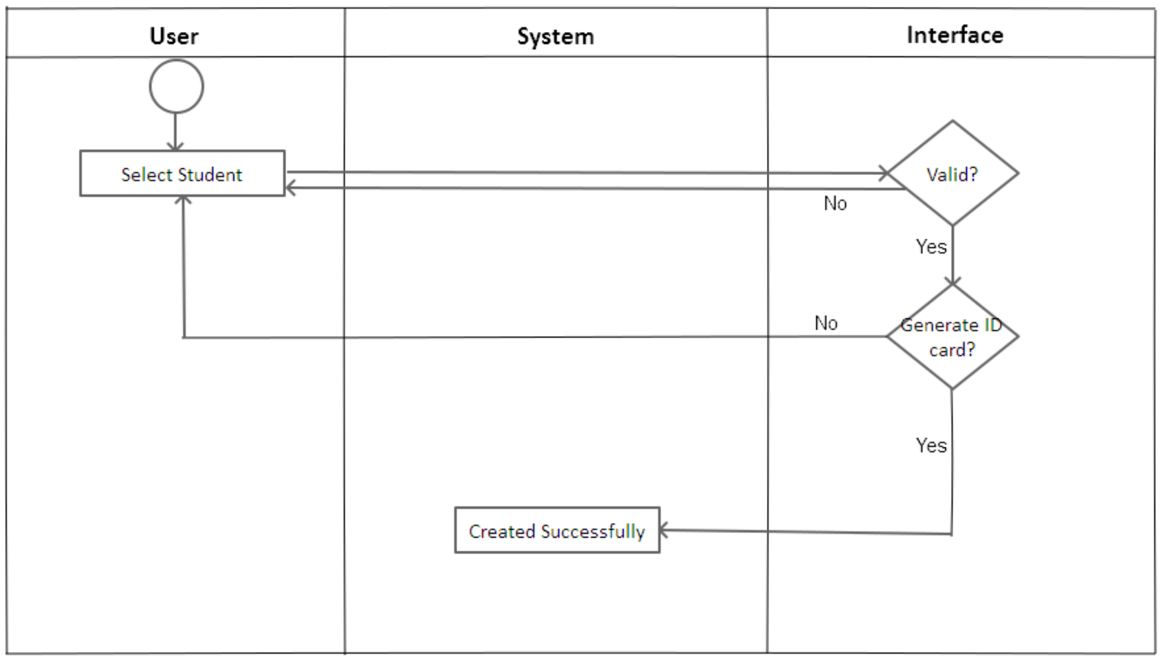


Figure 4.23: Swimlane Diagram for ID Card Generation

4.4.4.5 Swimlane Diagram for Testimonial Generation

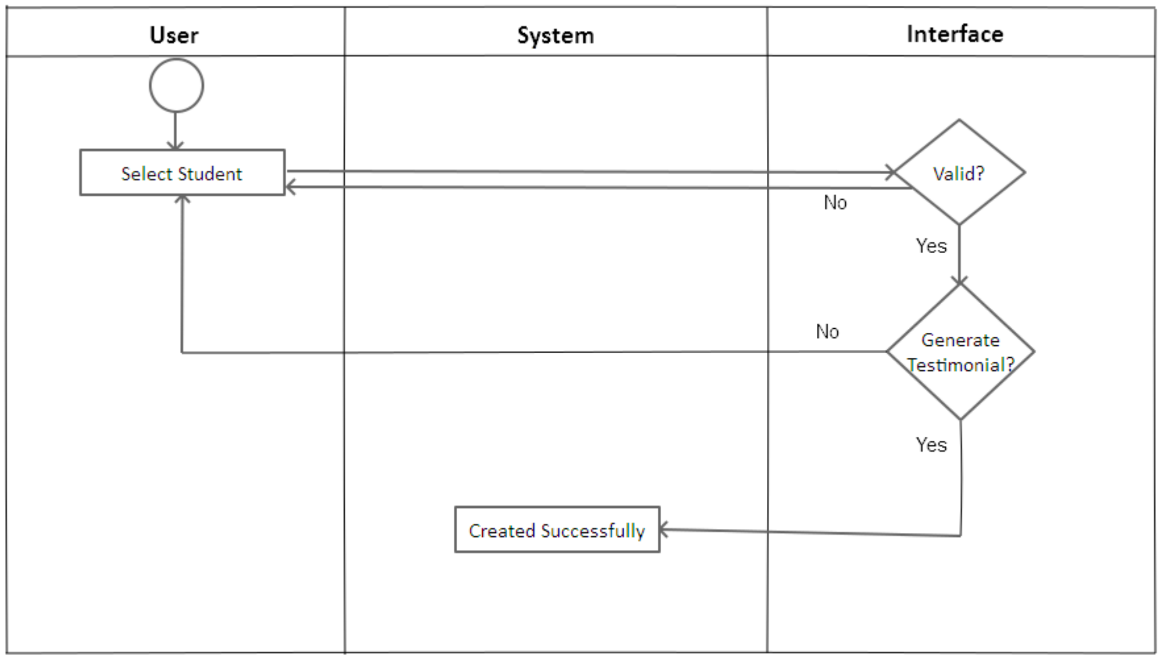


Figure 4.24: Swimlane Diagram for Testimonial Generation

4.4.4.6 Swimlane Diagram for Search Student

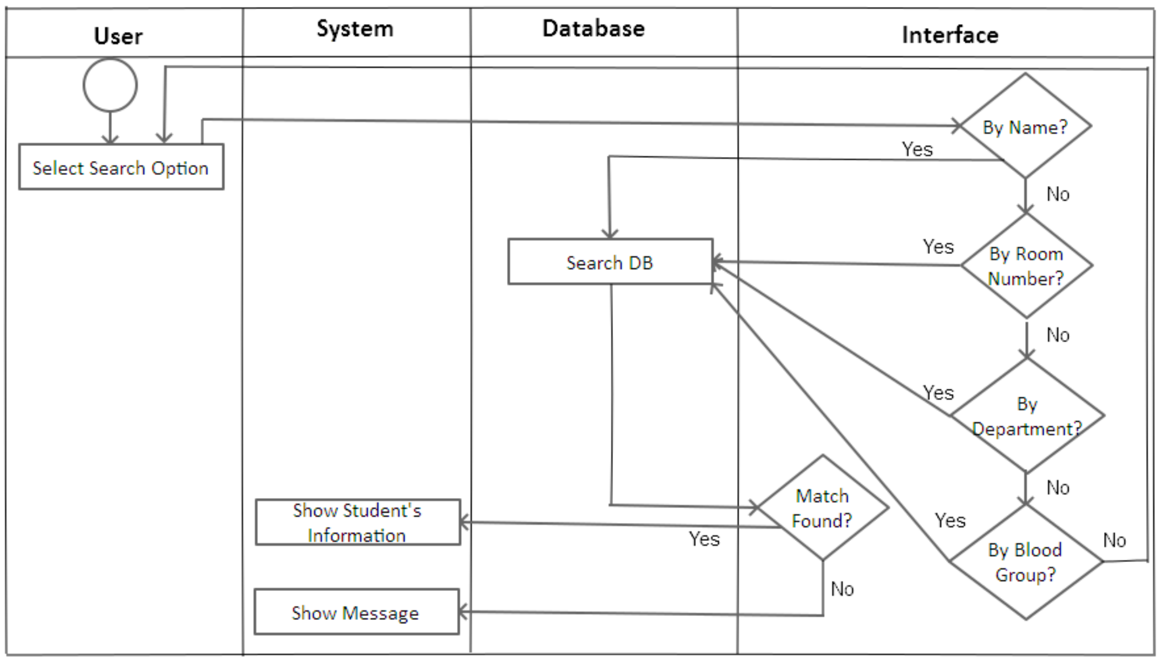


Figure 4.25: Swimlane Diagram for Search Student

4.4.4.7 Swimlane Diagram for Event Creation

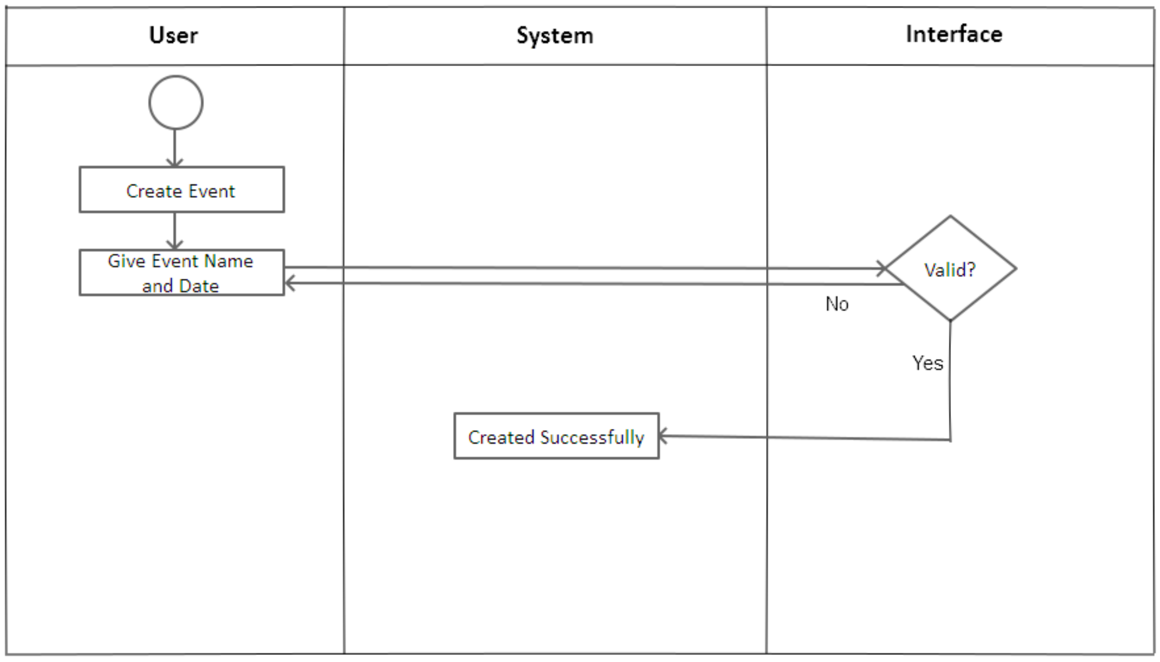


Figure 4.26: Swimlane Diagram for Event Creation

4.4.4.8 Swimlane Diagram for Upload Excel

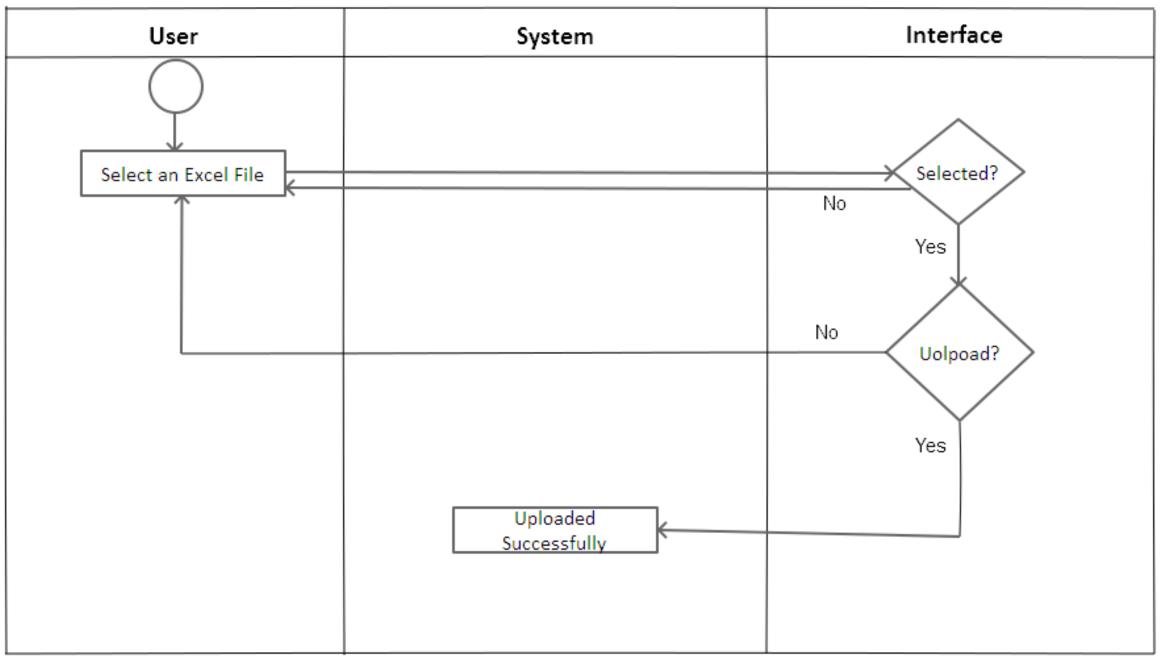


Figure 4.27: Swimlane Diagram for Upload Excel

4.4.4.9 Swimlane Diagram for NoticeBoard

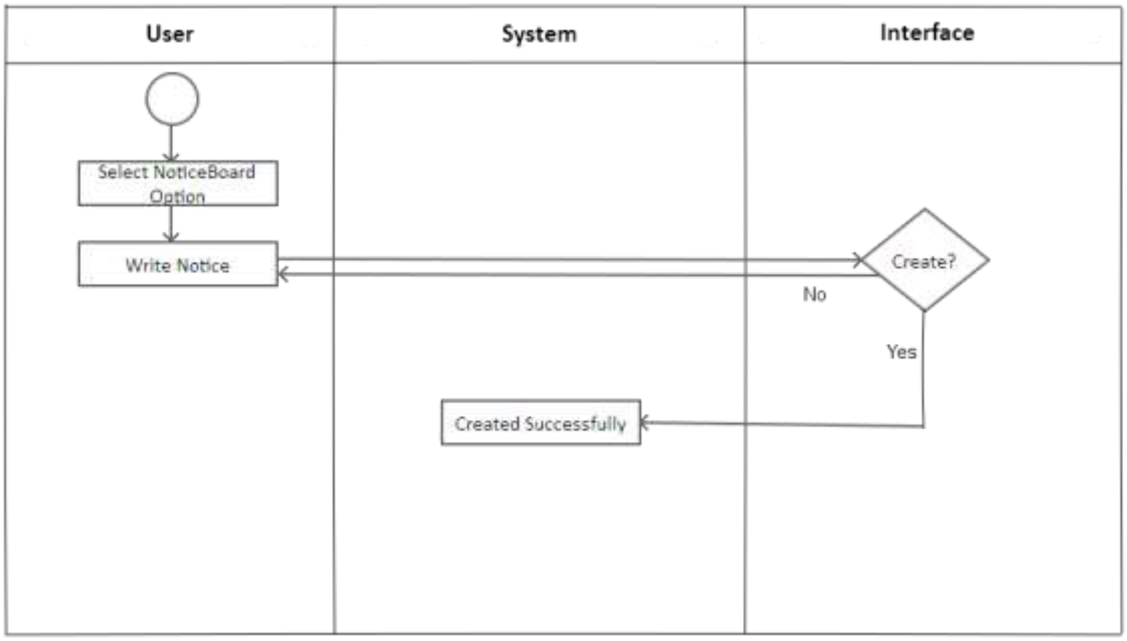


Figure 4.28: Swimlane Diagram for NoticeBoard

4.4.5 Data Modeling

Entities of our database are presented below.

1. admin
2. students_basic_info
3. students_address
4. students_legal_gardian
5. students_local_gardian
6. students_hall_info
7. students_previous_edu_info
8. students_current_edu_info
9. hall_accounting_info
10. faculty_members
11. events
12. category
13. sub_category
14. pictures
15. notice

4.4.5.1 E-R Diagram

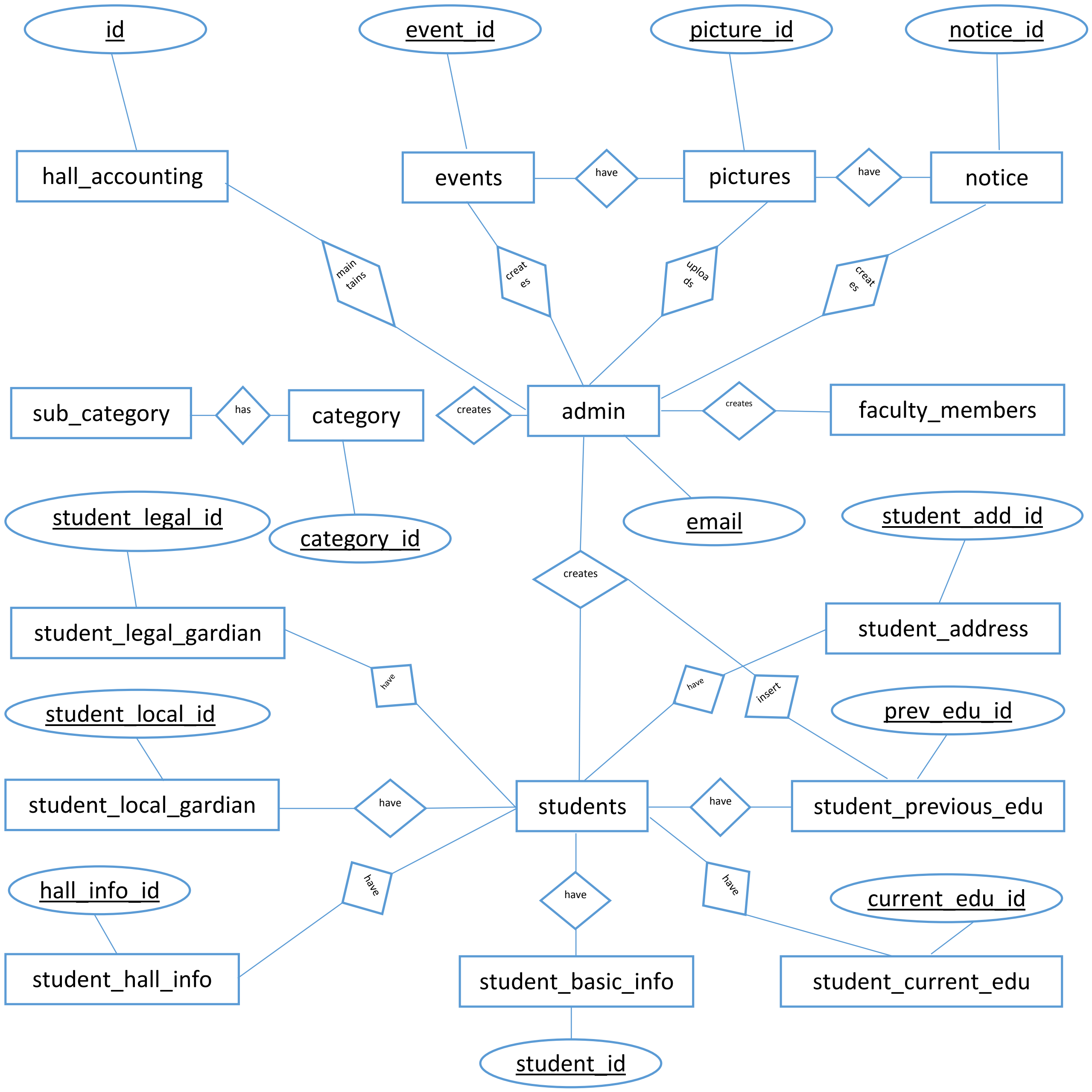


Figure 4.29: E-R diagram

4.4.5.2 Database Schema

students_basic_info	
Attribute Name	Type
student_id(PK)	int(10)
student_full_name	varchar(30)
father_name	varchar(30)
mother_name	varchar(30)
birth_date	date
blood_group	varchar(5)
nick_name	varchar(30)
student_contact_no	varchar(30)
student_email_address	varchar(30)
parents_contact_no	varchar(30)
student_religion	varchar(30)
student_nationality	varchar(30)
national_id	varchar(30)
student_profile_pic	varchar(30)

students_address	
Attribute Name	Type
student_add_id(PK)	int(10)
student_id(FK)	int(10)
current_district	varchar(30)
current_upazilla	varchar(30)
current_ward	varchar(30)
current_postcode	varchar(30)
current_village	varchar(30)
current_home	varchar(30)
permanent_district	varchar(30)
permanent _upazilla	varchar(30)
permanent _ward	varchar(30)
permanent _postcode	varchar(30)
permanent _village	varchar(30)
permanent _home	varchar(30)

students_legal_gardian	
Attribute Name	Type
legal_gardian_id(PK)	int(10)
student_id(FK)	int(10)
legal_gardian_name	varchar(30)
relationship_with_student	varchar(30)
legal_district	varchar(30)
legal_upazilla	varchar(30)
legal_ward	varchar(30)
legal_postcode	varchar(30)
legal_village	varchar(30)
legal_home	varchar(30)
legal_contact_no	varchar(30)
legal_occupation	varchar(30)
legal_monthly_income	number(30)

students_local_gardian	
Attribute Name	Type
local_gardian_id(PK)	int(10)
student_id(FK)	int(10)
local _gardian_name	varchar(30)
relationship_with_student	varchar(30)
local _district	varchar(30)
local _upazilla	varchar(30)
local _ward	varchar(30)
local _postcode	varchar(30)
local _village	varchar(30)
local _home	varchar(30)
local _contact_no	varchar(30)
local _occupation	varchar(30)
local _monthly_income	number(30)

students_hall_info	
Attribute Name	Type
hall_info_id(PK)	int(10)
student_id(FK)	int(10)
admission_session	varchar(30)
hall_name	varchar(30)
time_of_boarding	date
residence_type	varchar(30)
current_residence_permit_date	date
room_no	varchar(10)
residence_card	varchar(5)
library_card	varchar(5)
study_break	varchar(5)
study_break_year	varchar(10)
extra_curricular_activities	varchar(1000)

students_previous_edu_info	
Attribute Name	Type
prev_edu_id(PK)	int(10)
student_id(FK)	int(10)
hsc_roll	varchar(30)
hsc_board	varchar(30)
hsc_pass_year	varchar(30)
hsc_gpa	number(10)
gpa_without_fourth	number(10)
ssc_roll	varchar(30)
ssc_board	varchar(30)
ssc_pass_year	varchar(30)
ssc_gpa	number(10)
ssc_gpa_without_fourth	number(10)

students_current_edu_info	
Attribute Name	Type
current_edu_id(PK)	int(10)
student_id(FK)	int(10)
current_session	varchar(10)
current_year	varchar(10)
faculty	varchar(50)
department	varchar(50)
institute	varchar(50)
semester	varchar(10)
semester_cgpa	number(10)
total_cgpa	number(10)
status	varchar(10)

Admin	
Attribute Name	Type
Name	varchar(30)
email(PK)	varchar(30)
password	varchar(30)

hall_accounting_info	
Attribute Name	Type
id(PK)	int(10)
accounting_year	varchar(10)
initial_budget	number(10)
event_name	varchar(30)
event_cost	number(10)

faculty_members	
Attribute Name	Type
faculty_id	int(10)
faculty_members_name	varchar(30)
faculty_members_designation	varchar(50)

Events	
Attribute Name	Type
event_id(PK)	int(10)
event_title	varchar(30)
event_date	date
event_time	time(6)
event_details	varchar(1000)
pic_id(FK)	int(10)
event_creator_name	varchar(30)
event_creator_user_id(FK)	int(10)
event_status	varchar(30)

Category	
Attribute Name	Type
category_id(PK)	int(10)
category_name	varchar(100)

sub_category	
Attribute Name	Type
sub_cat_id(PK)	int(10)
sub_category_name	varchar(100)
category_id(FK)	int(10)

Picture	
Attribute Name	Type
picture_id(pk)	int(10)
pic_title	varchar(30)
pic_url	varchar(30)
pic_type	varchar(30)
pic_detail	varchar(1000)
pic_author	varchar(30)
pic_status	varchar(30)

Notice	
Attribute Name	Type
notice_id	int(10)
notice_title	varchar(30)
notice_detail	varchar(1000)
notice_date	date
notice_time	time(6)
notice_type	varchar(30)
notice_file	varchar(30)
notice_status	varchar(30)

4.4.6 UML Class Diagram

Admin	
Attributes	Methods
1. User Name	1. LoginSystem()
2. Password	2. CreatingStudentsAccounts()
3. Phone Number	3.AllocatingRoomNumber()
4. E-mail	4.GeneratingIdCards()
	5.GeneratingTestimonial()
	6.PostingNewNotice()
	7.UpdateStudentInformation()
	8. CreateNewEvents()
	9. CreateStudentProfile()
	10. Inactive student account()
	Creating students' accounts, Allocating students' room number, Generating ID cards, Generating the testimonial, Posting new notice in the noticeboard, Authentication etc.

General User or Student	
Attributes	Methods
1. User Name 2. Password 3. Phone Number 4. E-mail	1. Register() 2. UpdateProfile() 3. ApplyForDifferentStuffs()
	Database.

Student Information	
Attributes	Methods
1) Student Name 2) Student Department 3) Date of Birth 4) Gender 5) Contact Information 6) Student ID Number 7) Student Room Number 8) Student Studying Year 9) E-Mail	1. InsertValue() 2. UpdateValue()
	Sending Notification, Database.

Notification	
Attributes	Methods
1) E-Mail 2) Content 3) Attached Files	1. SendMail()
	Selected Student, Database.

Student Creation	
Attributes	Methods
1) Student Name 2) Student Department 3) Date of Birth 4) Gender 5) Student ID number 6) Student Room Number	1. CreateUser()
	Admin, Database.

Database	
Attributes	Methods
1) User Name 2) Password 2) Student Department 3) Date of Birth 4) Gender 5) Contact Information 6) Student ID Number 7) Student Room Number	1. Add() 2. Update() 3. Deactivate()
	Admin, User creation, Selected Student, Student Information, Authentication.

Authentication System	
Attributes	Methods
1) User Name 2) Password	1. Login() 2.SearchingStudentInformation() 3. Logout()
	Admin.

Accounting Module	
Attributes	Methods
1) User Name 2) Password 3)Date 4)Money 5)Purpose	1. Deposit() 2. Withdraw() 3. Expense()
	Admin.

4.4.7 Class Responsibility Collaboration (CRC) Model

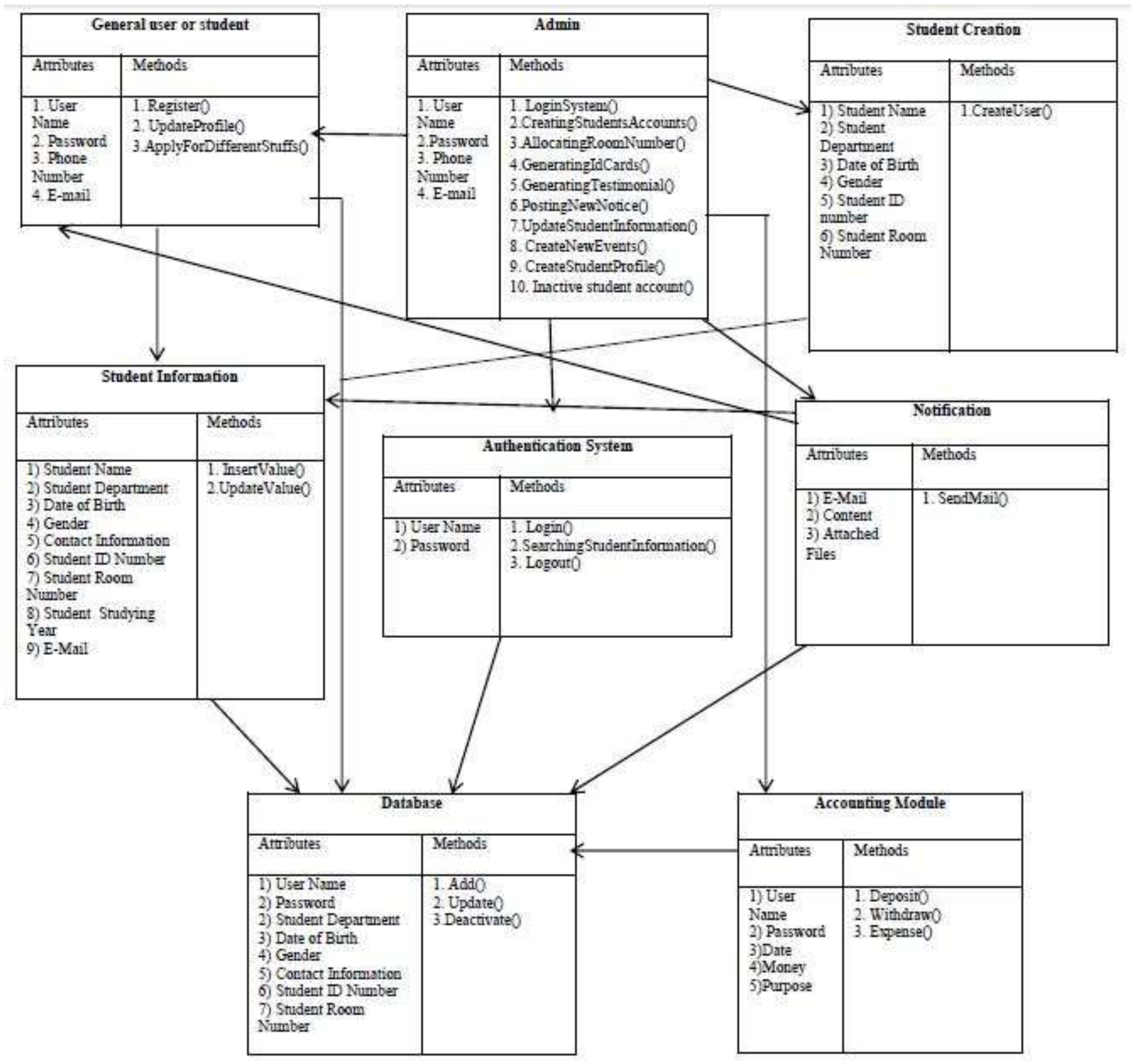


Figure 4.30: CRC diagram

4.4.8 Data Flow Diagram

Here is the DFD for this SRS.

4.4.8.1 DFD Level 0

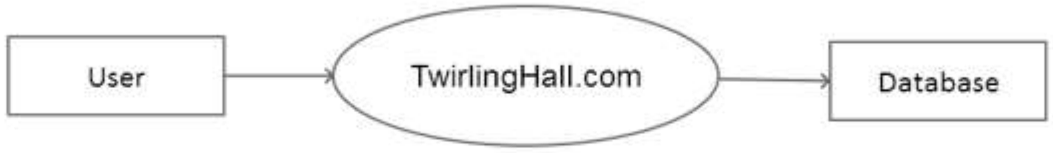


Figure 4.31 DFD Level 0

4.4.8.2 DFD Level 1

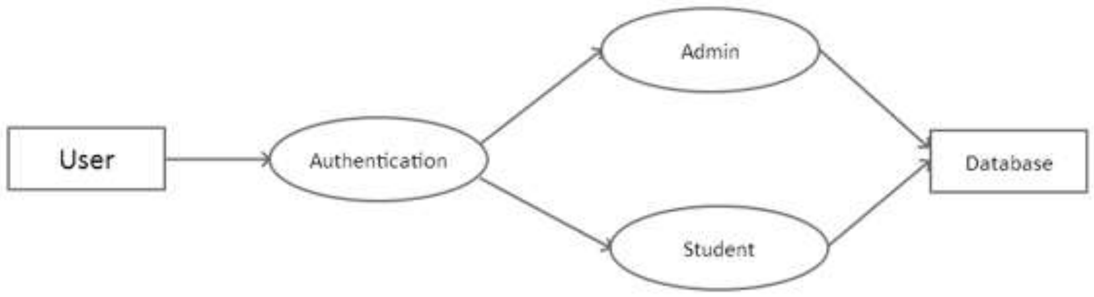


Figure 4.32: DFD Level 1

4.4.8.3 DFD Level 2.1

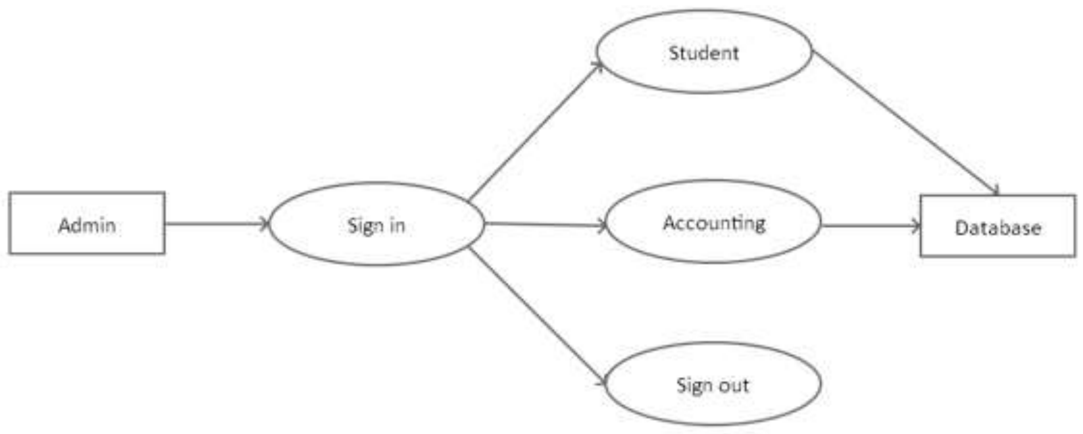


Figure 4.33 DFD Level 2.1

4.4.8.4 DFD Level 2.2

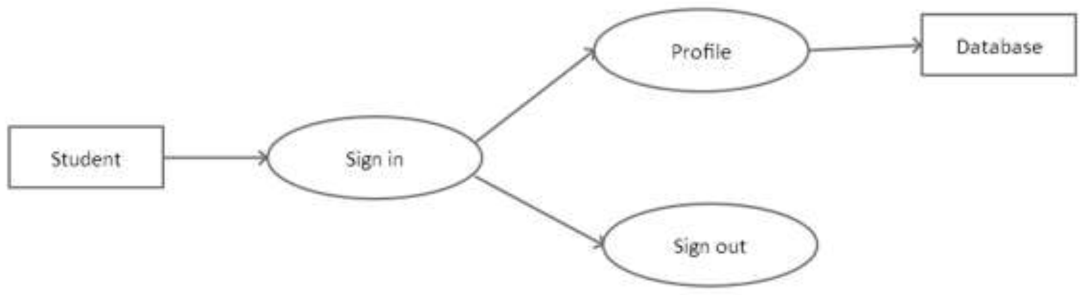


Figure 4.34 DFD Level 2.2

4.4.8.5 DFD Level 3.1

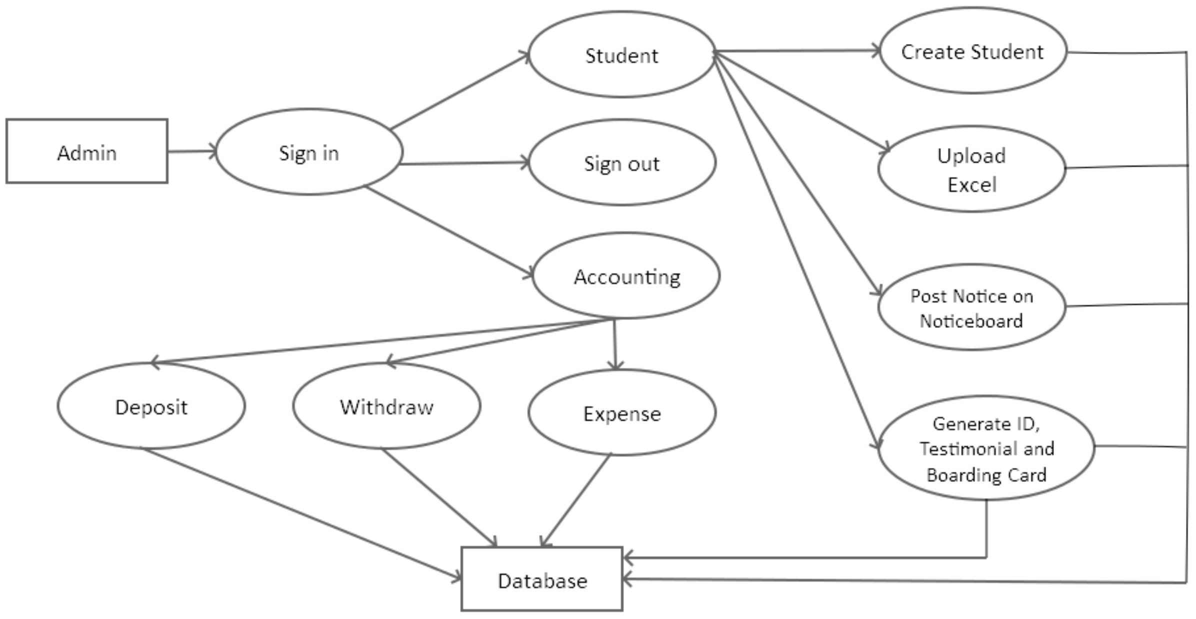


Figure 4.35 DFD Level 3.1

4.4.8.6 DFD Level 3.2

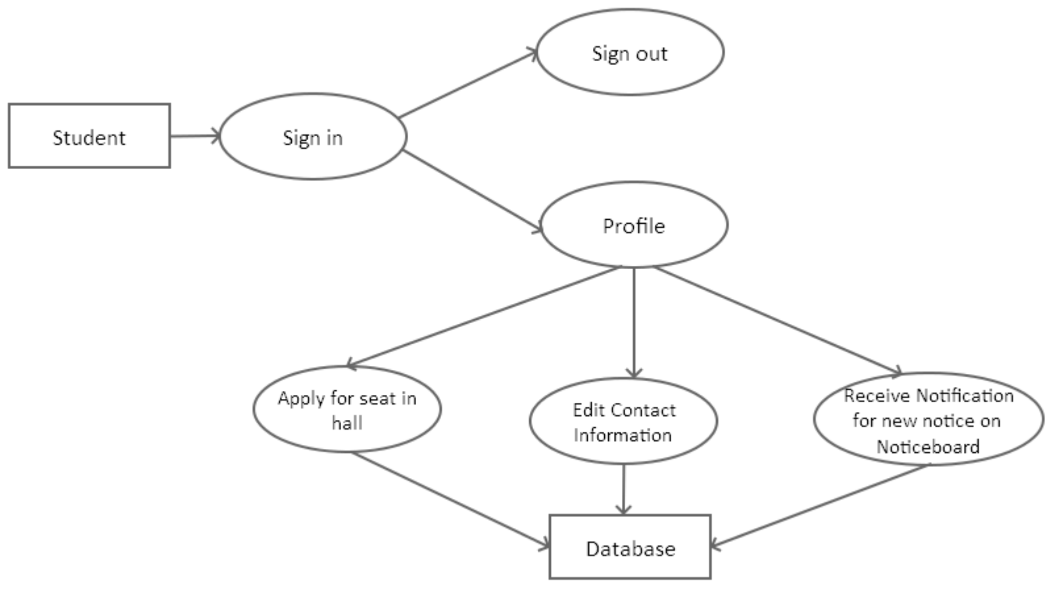


Figure 4.36 DFD Level 3.2

4.4.9 State Diagram

In the following the state diagrams for this SRS are given.

4.4.9.1 State Diagram for Authentication

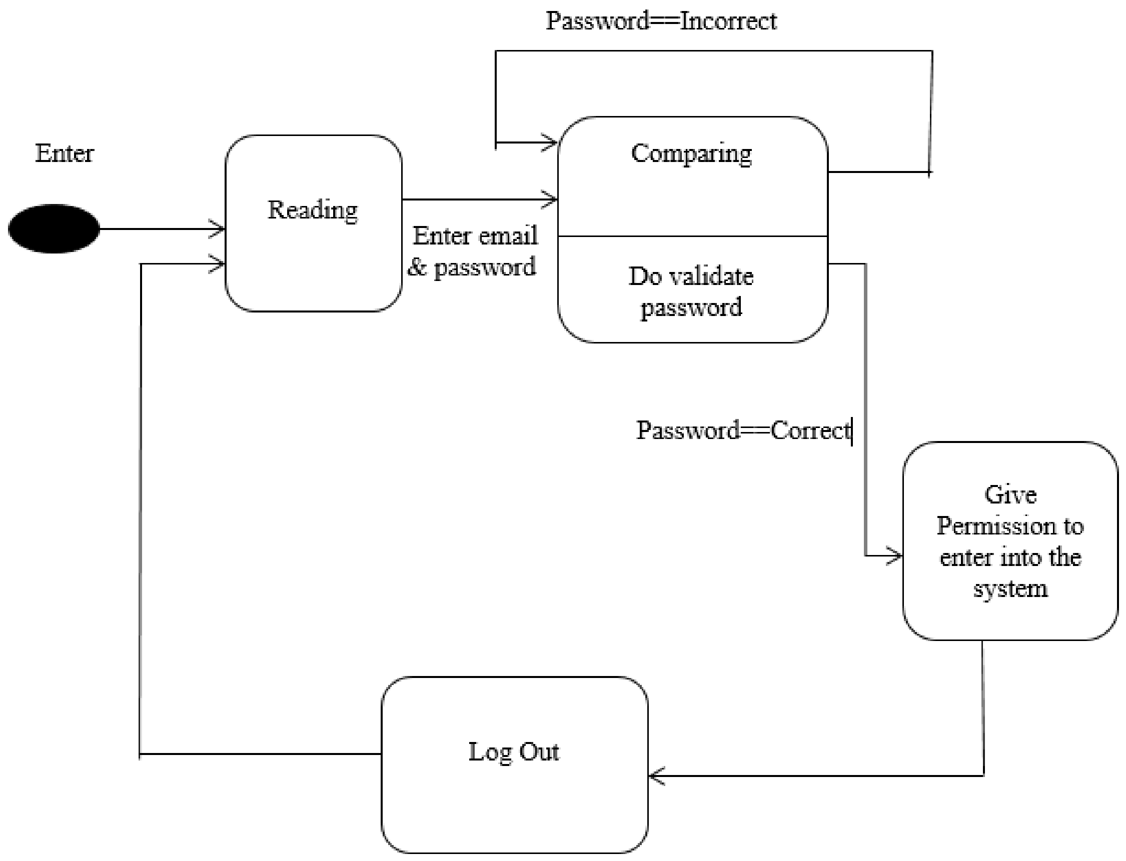


Figure 4.37: State Diagram for Authentication

4.4.9.2 State Diagram for Student Creation

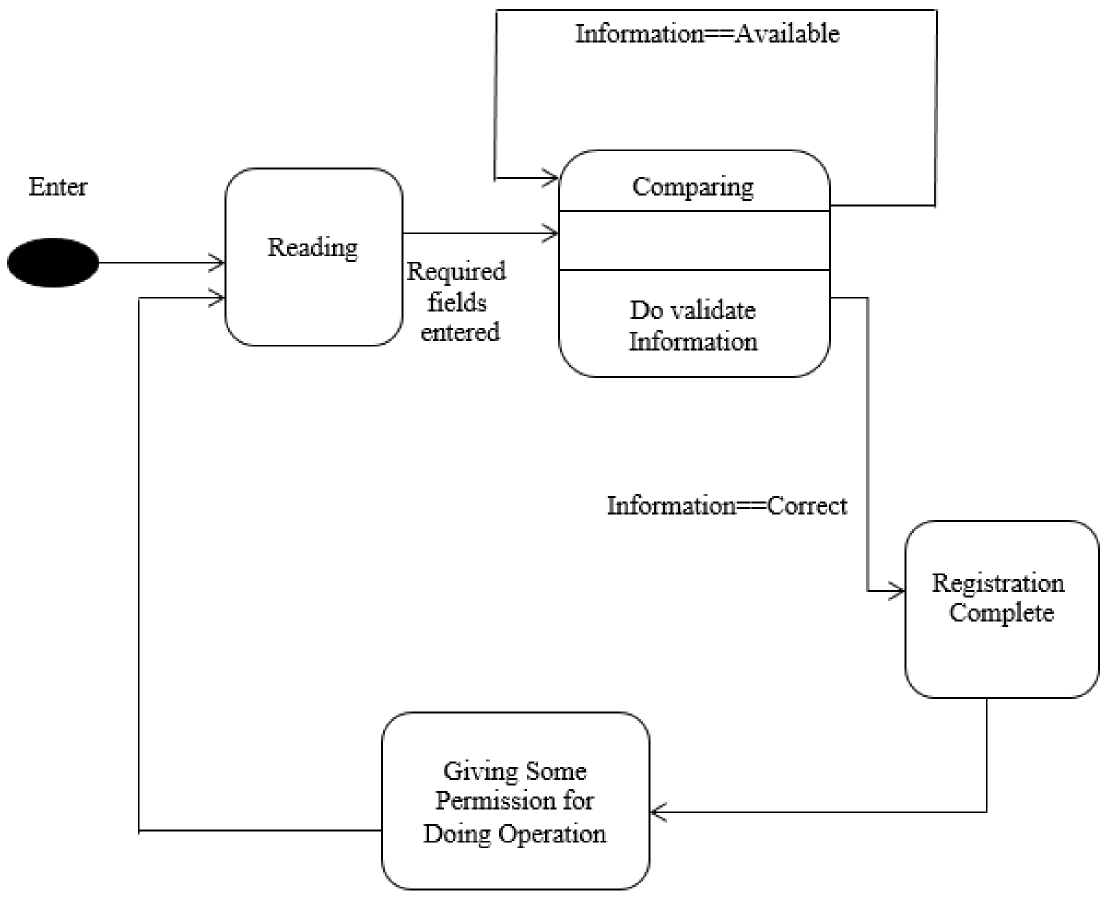


Figure 4.38: State Diagram for Student Creation

4.4.9.3 State Diagram for Boarding Card Generation

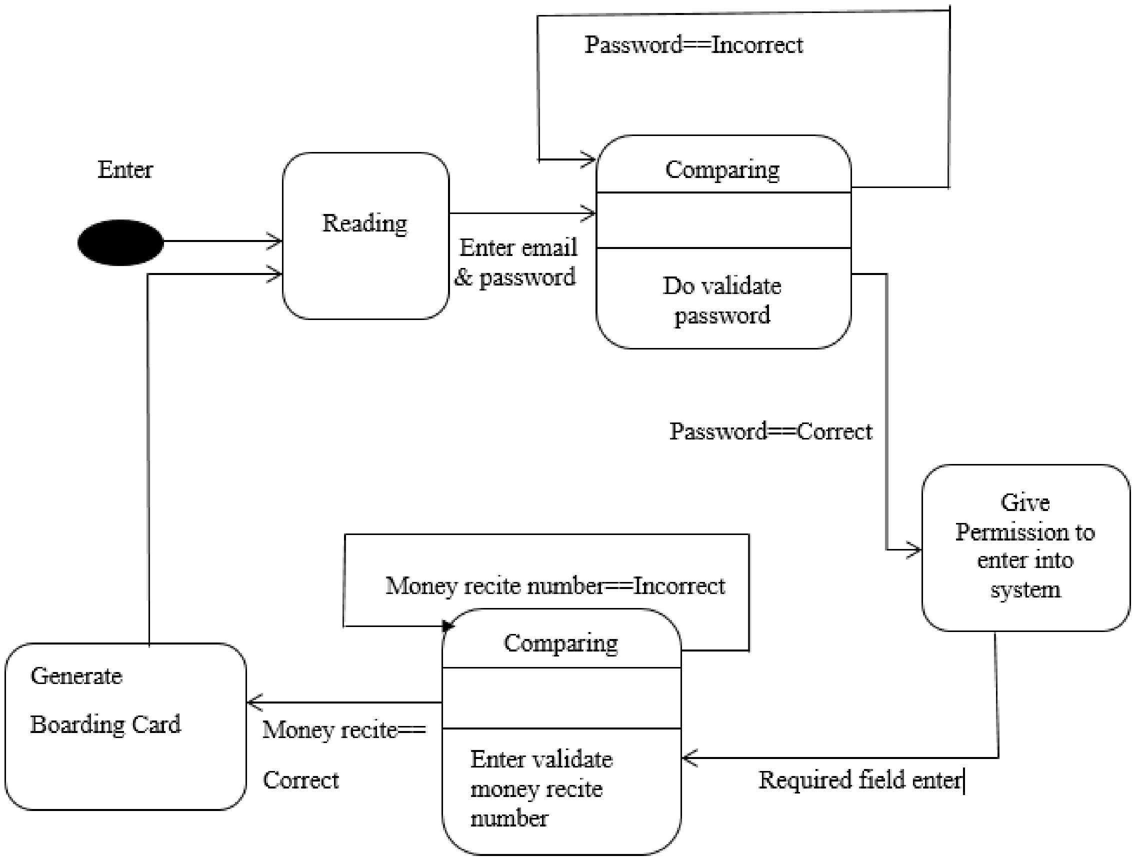


Figure 4.39: State Diagram for Boarding Card Generation

4.4.9.4 State Diagram for ID Card Generation

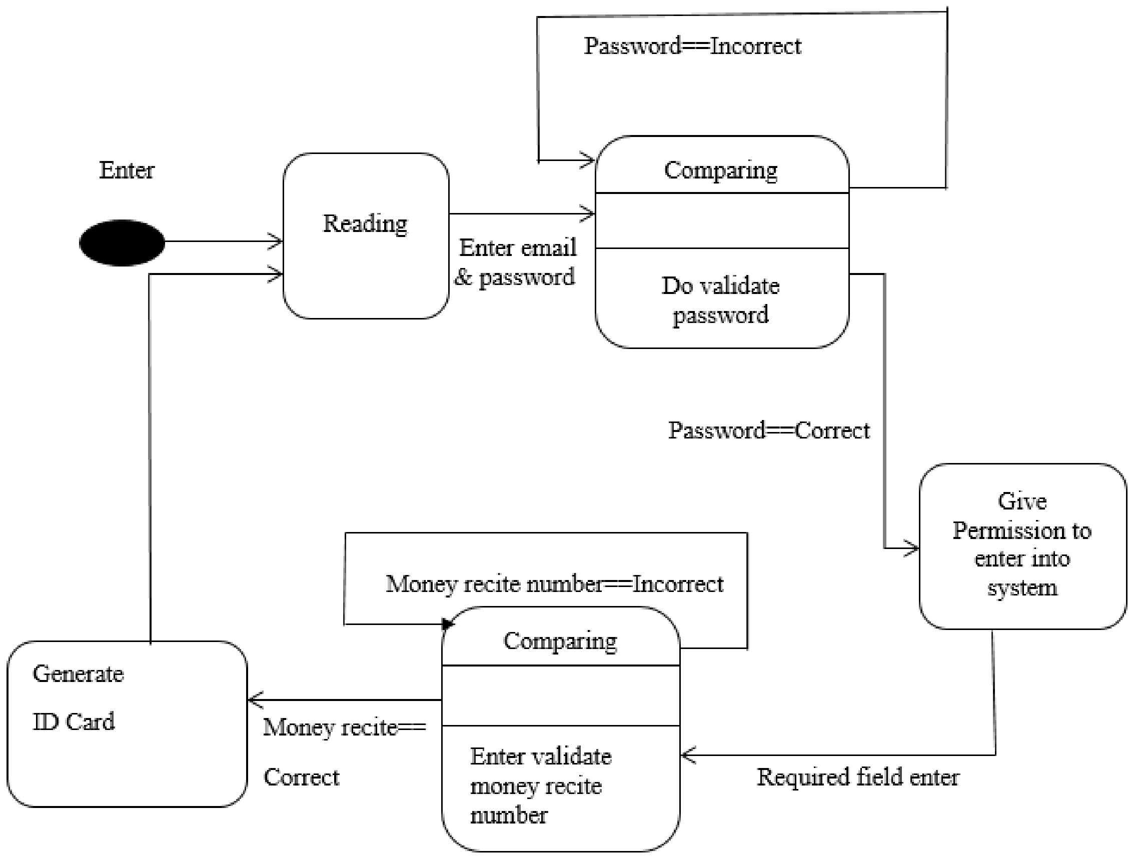


Figure 4.40: State Diagram for ID Card Generation

4.4.9.5 State Diagram for Testimonial Generation

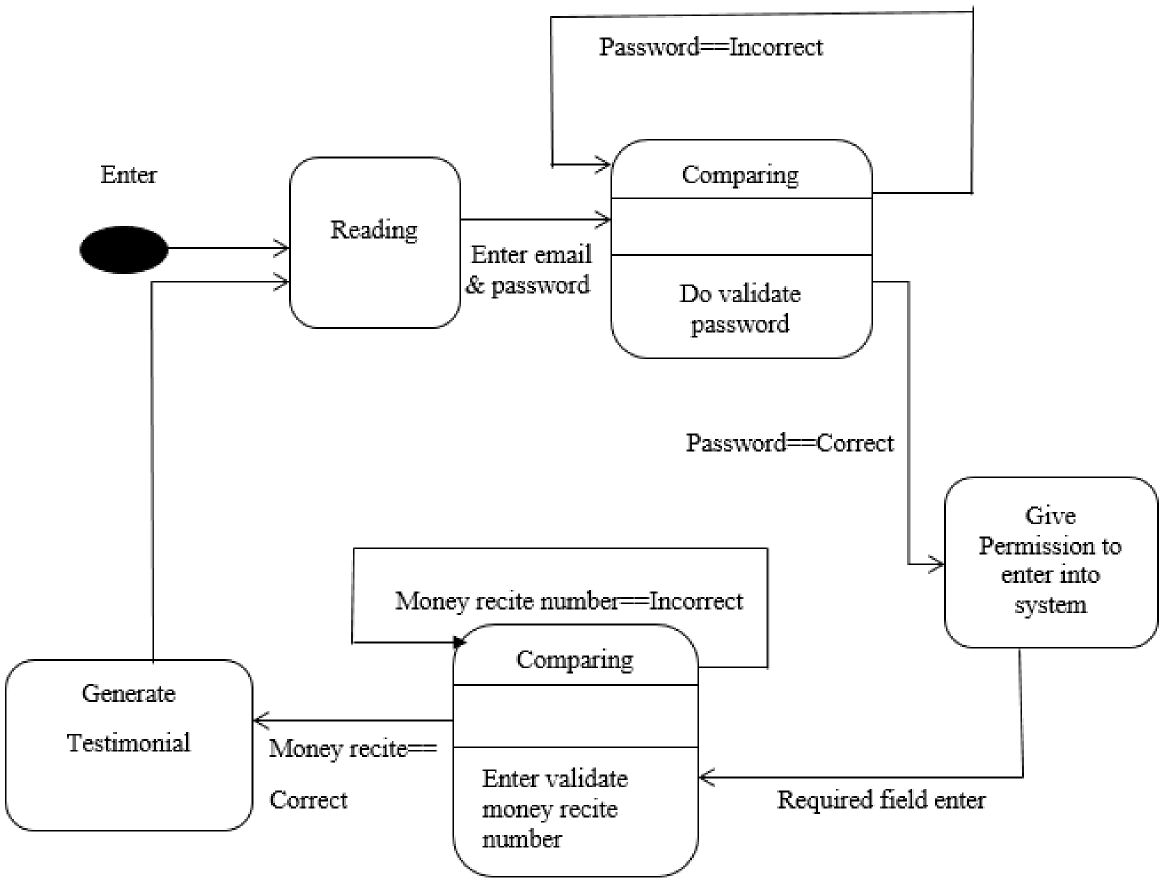


Figure 4.41: State Diagram for Testimonial Generation

4.4.9.6 State Diagram for Search Student

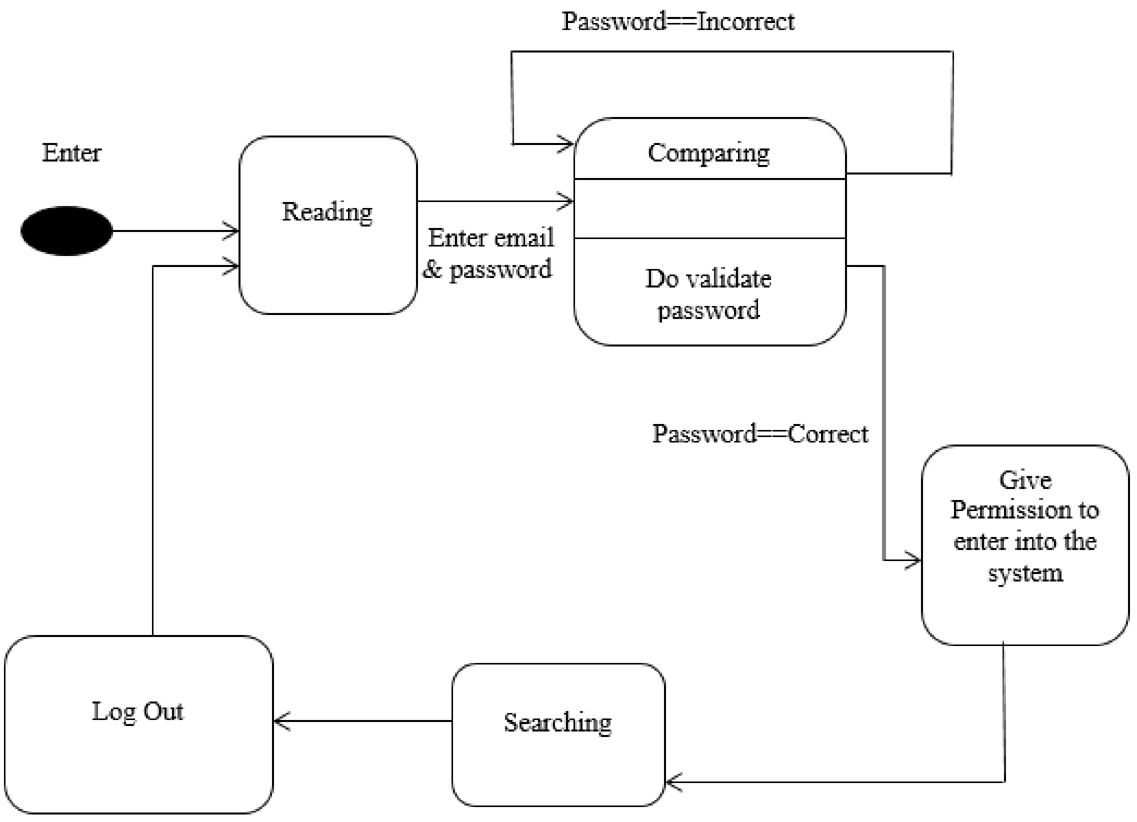


Figure 4.42: State Diagram for Search Student

4.4.9.7 State Diagram for Event Creation

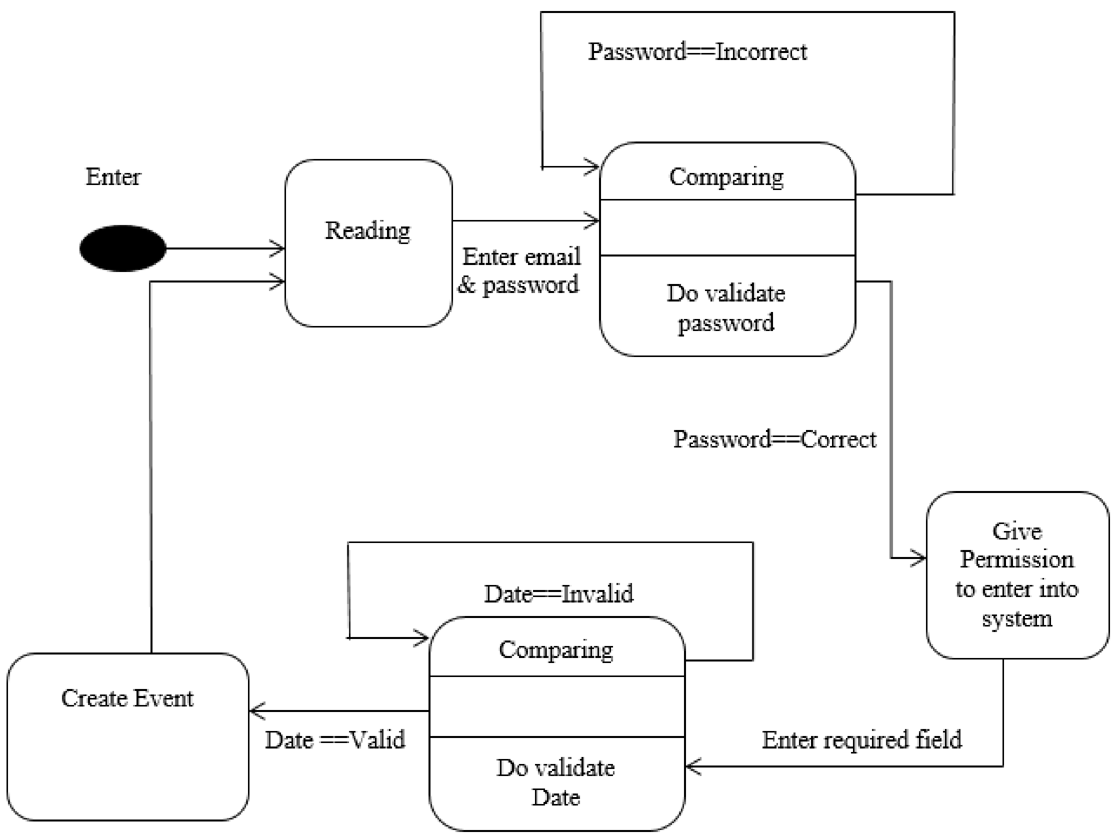


Figure 4.43: State Diagram for Event Creation

4.4.9.8 State Diagram for Upload Excel

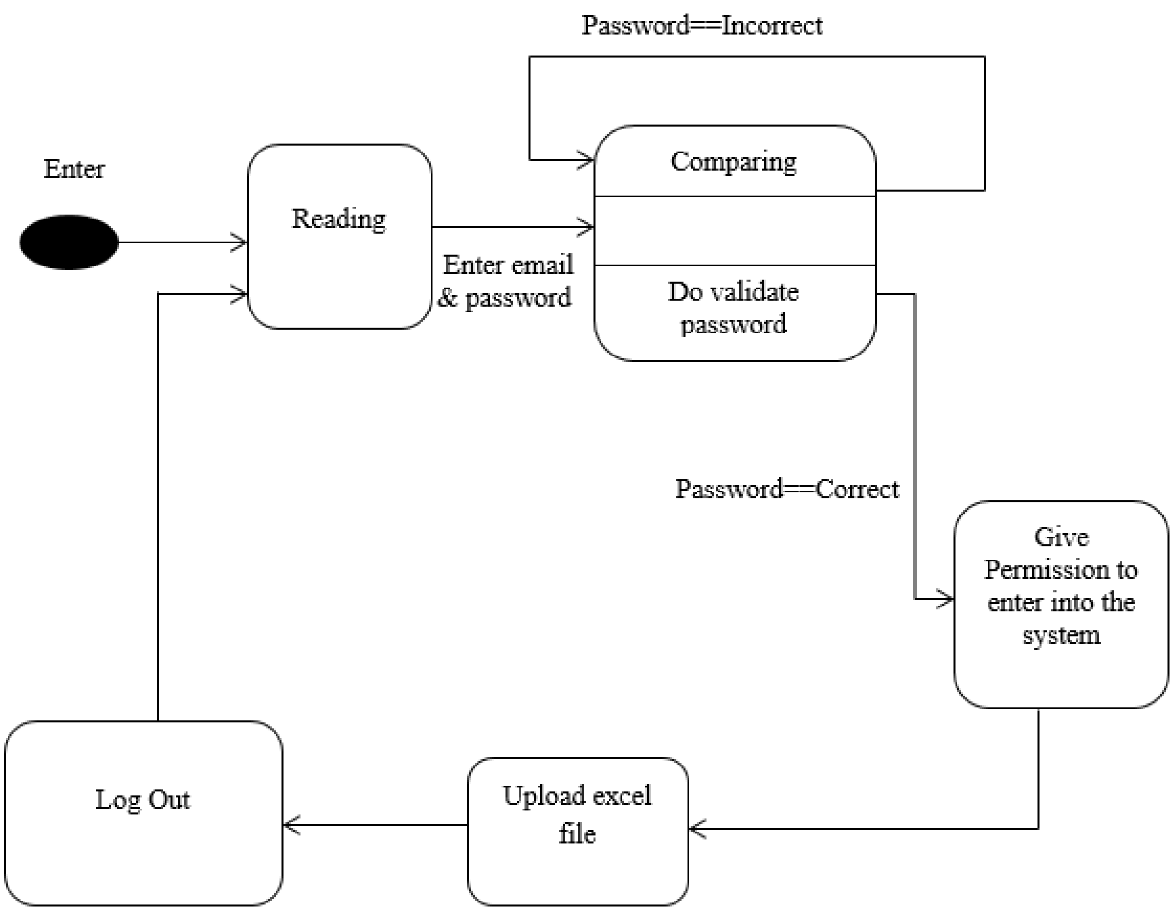


Figure 4.44: State Diagram for Upload Excel

4.4.9.9 State Diagram for NoticeBoard

4.4.9.9.1 State Diagram for Notice Creation by Admin

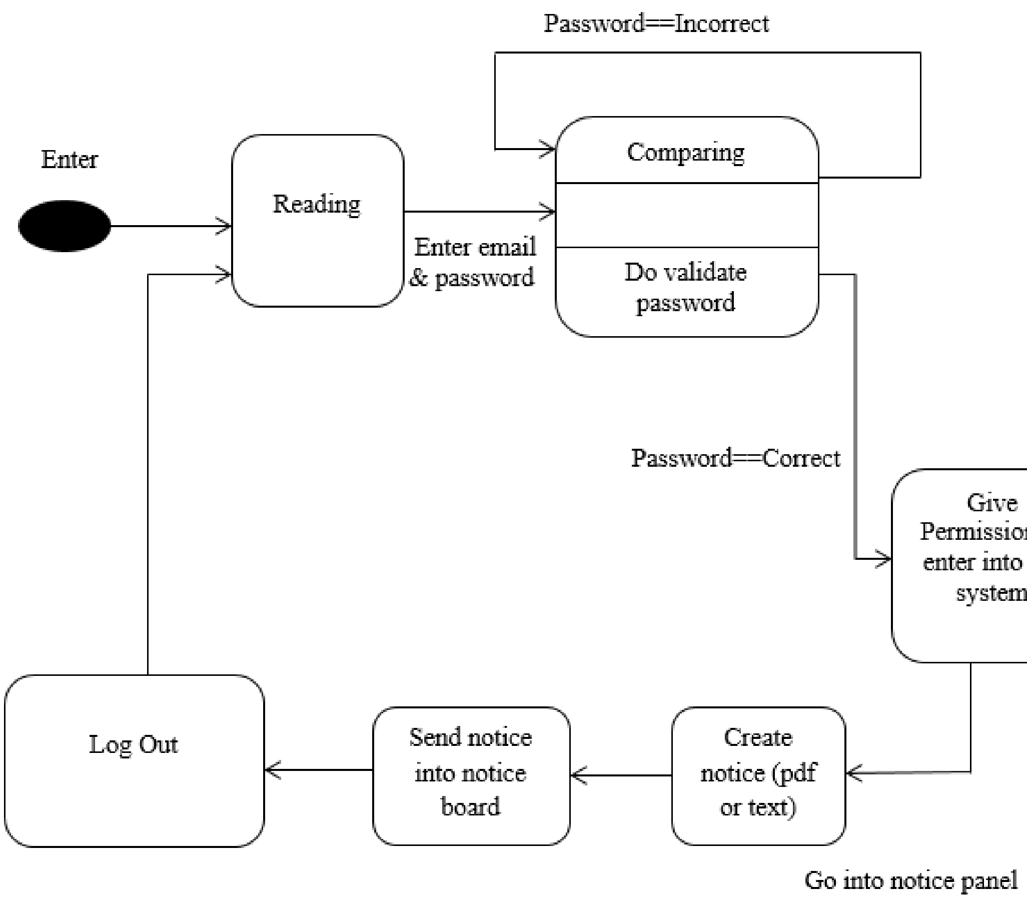


Figure 4.45: State Diagram for Notice Creation by Admin

4.4.9.9.2 State Diagram for Viewing Notice by Student

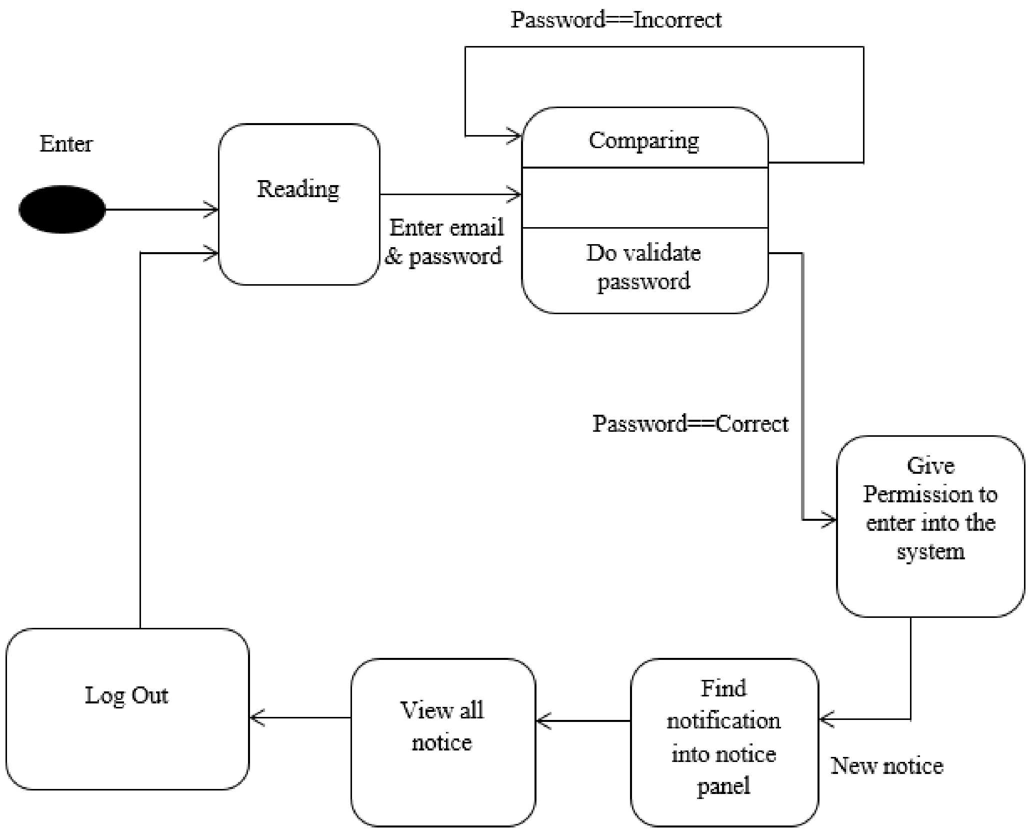


Figure 4.46: State Diagram for View Notice by Student

4.4.10 Sequence Diagram

In the following the sequence diagrams for this SRS are given.

4.4.10.1 Sequence Diagram for Authentication

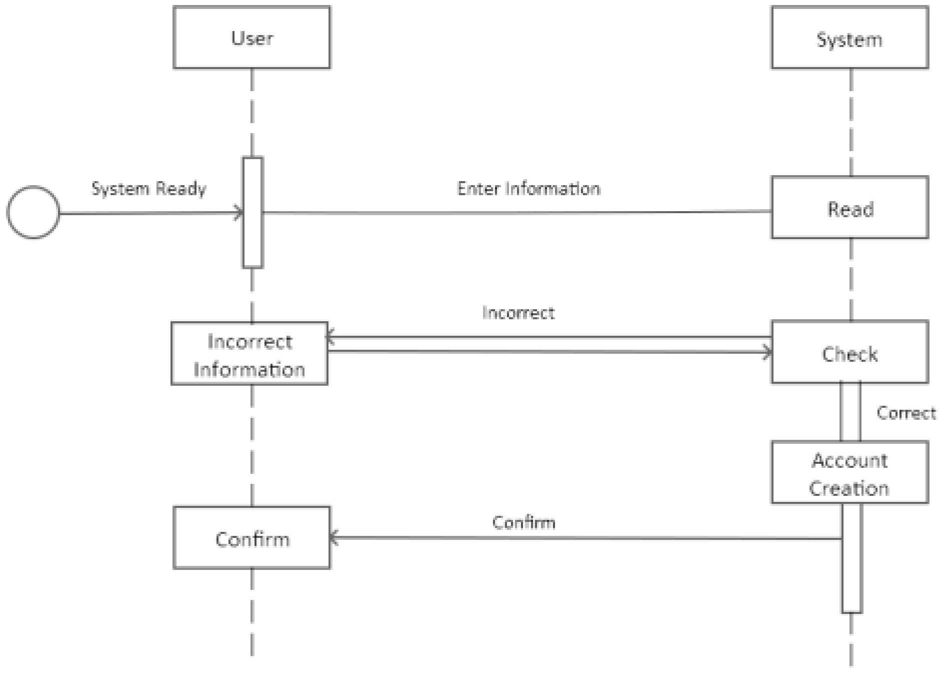


Figure 4.47: Sequence Diagram for Authentication

4.4.10.2 Sequence Diagram for Student Creation

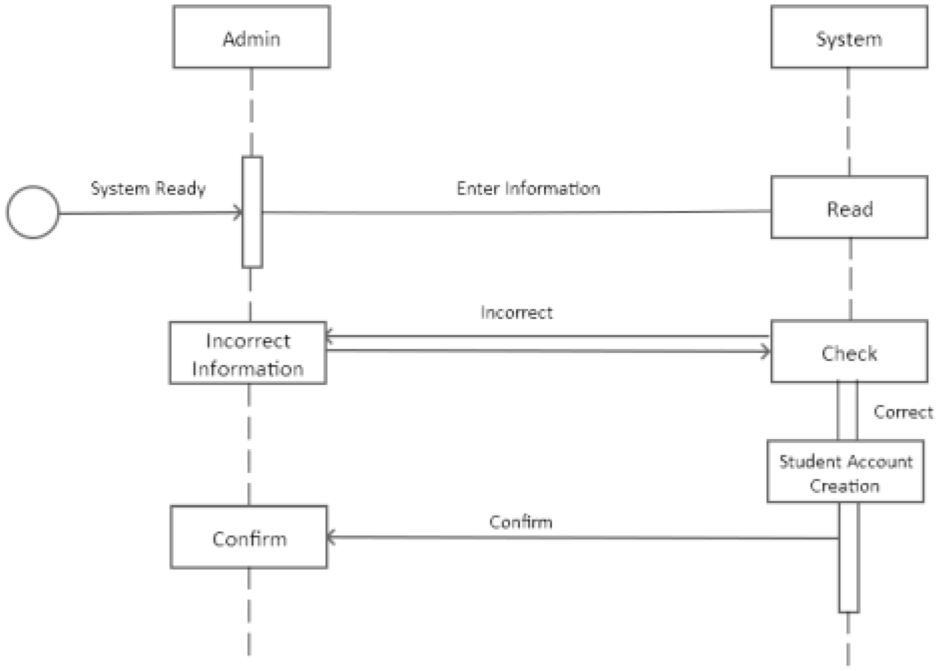


Figure 4.48: Sequence Diagram for Student Creation

4.4.10.3 Sequence Diagram for Boarding Card Generation

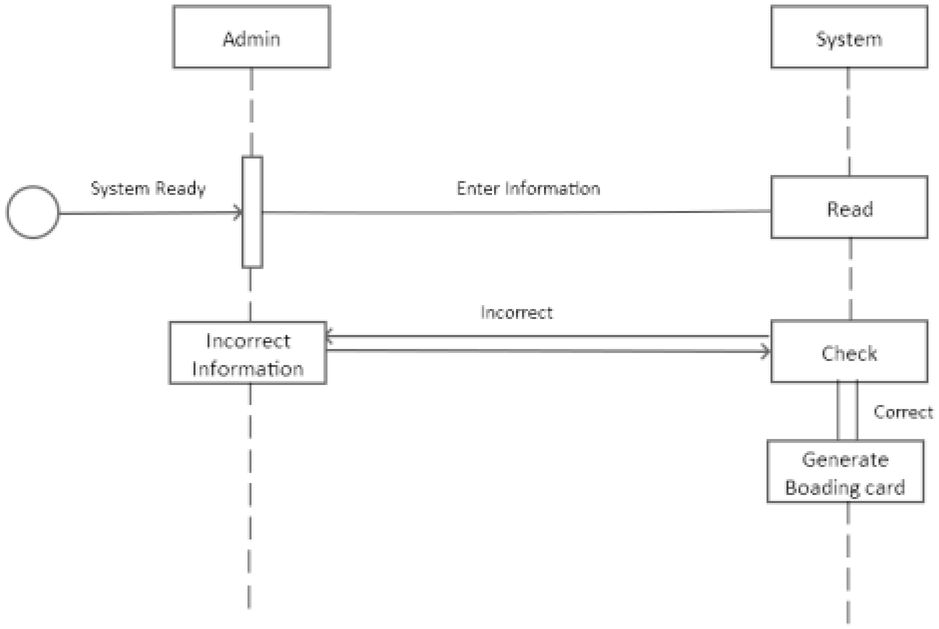


Figure 4.49: Sequence Diagram for Boarding Card Generation

4.4.10.4 Sequence Diagram for ID Card Generation

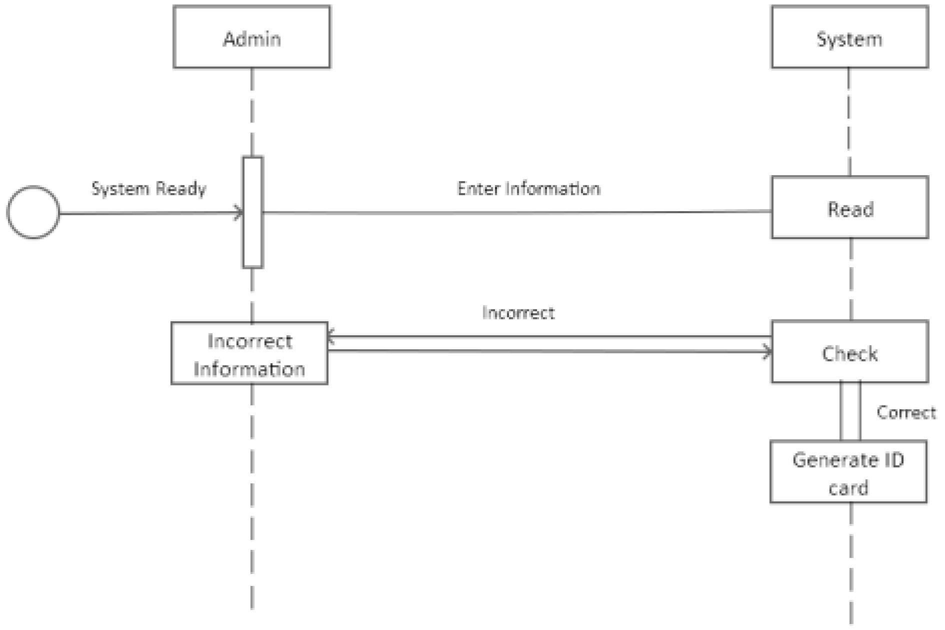


Figure 4.50: Sequence Diagram for ID Card generation

4.4.10.5 Sequence Diagram for Testimonial Generation

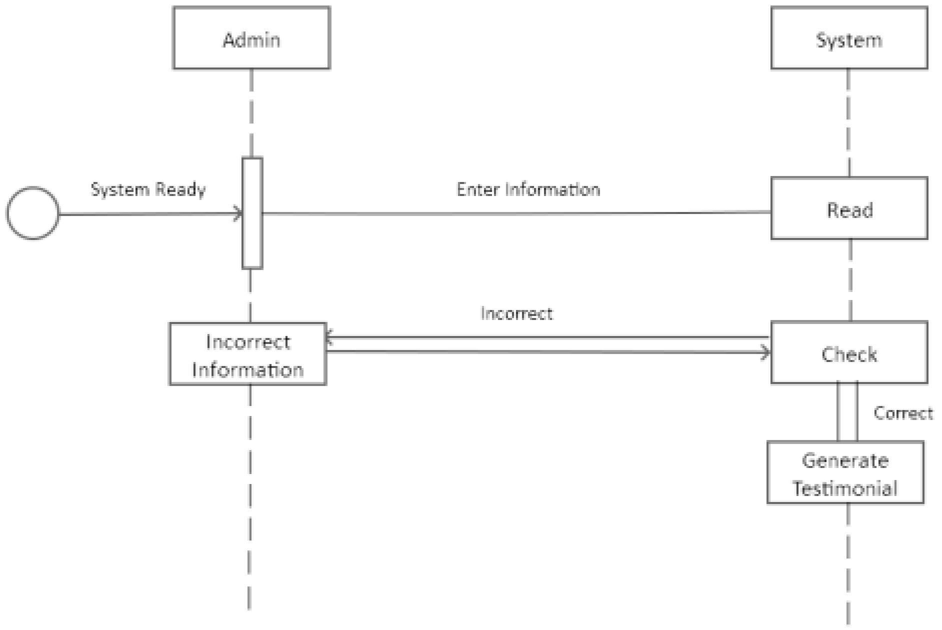


Figure 4.51: Sequence Diagram for Testimonial Generation

4.4.10.6 Sequence Diagram for Search Student

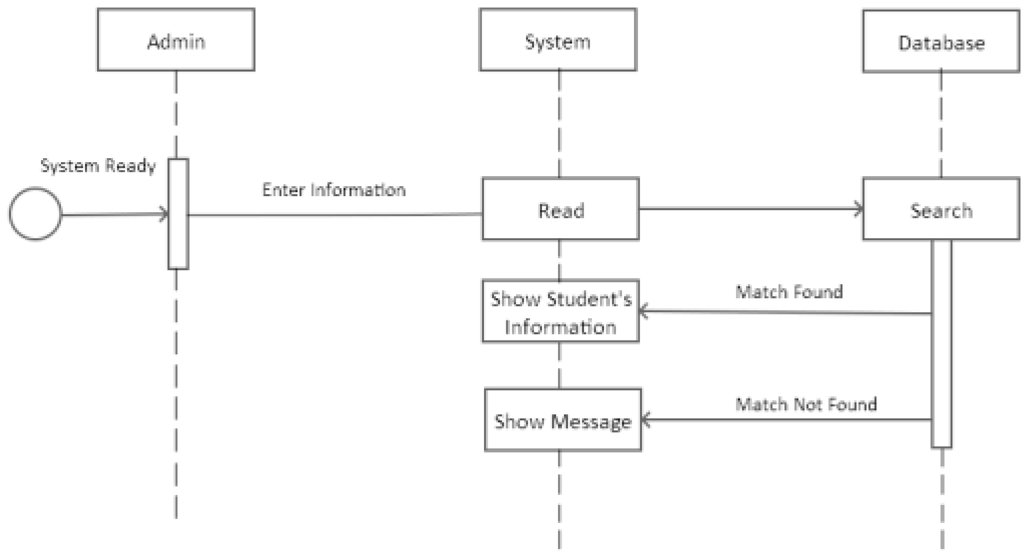


Figure 4.52: Sequence Diagram for Search Student

4.4.10.7 Sequence Diagram for Event Creation

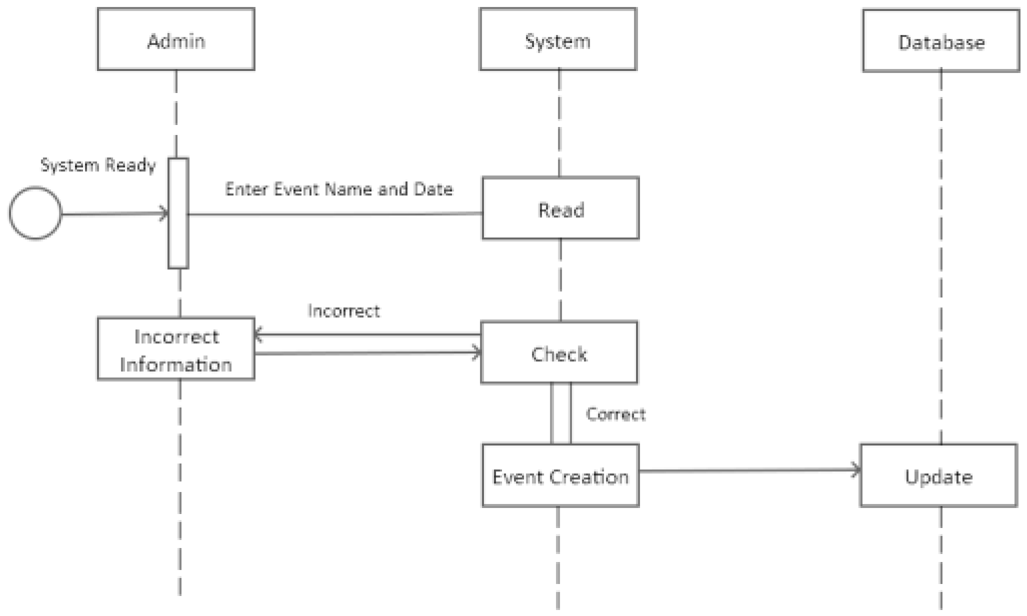


Figure 4.53: Sequence Diagram for Event Creation

4.4.10.8 Sequence Diagram for Upload Excel

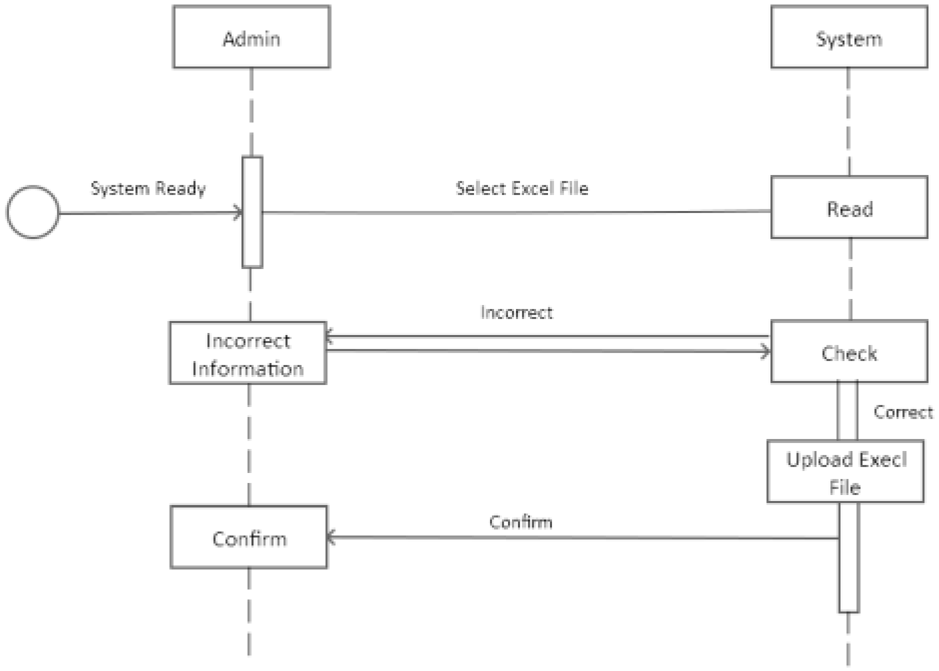


Figure 4.54: Sequence Diagram for Upload Excel

4.4.10.9 Sequence Diagram for Notice Board

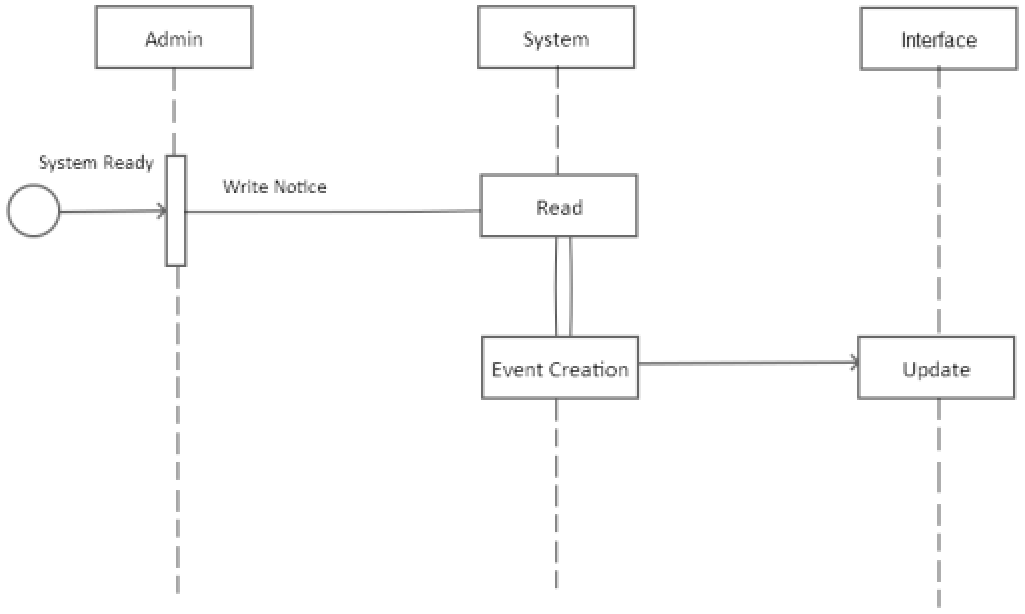


Figure 4.55: Sequence Diagram for NoticeBoard

4.5 Conclusion

Because of using the methodology of SRS our practical understanding regarding software engineering has been enhanced. We have tried our level best in order that fruitful outcome can be expected. We hope this SRS document will be cooperative for customers to understand the website for managing students' information of the hall.

Chapter 5: Software Architectural Design

Chapter presents the software architectural design where component level design and user interface design is performed.

5.1 Software Design Process

Software design is an important part of the software design life cycle. It usually involves problem solving and planning a software solution. This includes both low-level component and algorithm design and high-level, architecture design.

5.1.1 Software Design

Software design is what almost every engineer wants to do. It is the place where creativity rules, where stakeholder requirements, business needs, and technical considerations all come together in the formulation of a product or system. Design creates a representation or model of the software, but unlike the requirements model that focuses on describing required data, function, and behavior, the design model provides detail about software architecture, data structures, interfaces, and components that are necessary to implement the system.

5.1.2 Who does it

Each of the design tasks are conducted by the software engineers. They first avail through the requirements of the customer. Based on that requirement they design the software.

5.1.3 Importance

Software design allows the engineers to model the system or product that is to be built. This model can be assessed for quality and improved before code is generated, tests are conducted, and end users become involved in large numbers. Software design is the place where software quality is established.

5.1.4 Steps of the Software Design

Design depicts the software in number of different ways. The steps for the software design are as follows.

- The architecture of the system or product must be represented.

- The interface that connects the software to end users, to other systems and devices, and to its own constituent components are modeled.
- The software components that are used to construct the system are designed.

Each of the above stated views represents a different design action but all of them must conform to a set of basic design concepts that guide software design work.

5.1.5 Assessment of Correctness

The design model is assessed by the software team in an effort to determine whether it contains errors, inconsistencies or omissions; whether better alternatives exist and whether the model can be implemented within the constraints, schedule and cost that have been established.

5. 2 Discussion on Design Actions

The steps of the software design has already been described in the previous section. These steps has to be followed in consecutive manner. The design has to be done by following the sequence below.

1. Architectural design
2. Component level design
3. User interface design

5. 2.1 Architectural Design

The architecture of a system is a comprehensive framework that describes its form and structure, its components and how they fit together. The architectural design is first in sequence in the software design process.

5.2.1.1 Definition

The software architecture of a program or computing system is the structure or structures of the system, which comprise software components, the externally visible properties of those components, and the relationships among them. Software architecture must model the structure of a system and the manner in which the data and the procedural components collaborate with one another. It is a representation that enables the software engineers to do the following.

- analyze the effectiveness of the design in meeting its stated requirements
- consider architectural alternatives at a stage when making design changes is still relatively easy
- reduce the risk associated with the construction of the software

5.2.1.2Who does it

A software engineer can design both data and architecture. But still the job is allocated to specialists when large and complex systems are to be built. A database or data warehouse designer creates the data architecture for a system. The system architect selects an appropriate architectural style from the requirements derived during software requirements analysis.

5.2.1.3 Importance

There are three key reasons that specifies the importance of software architecture. These are

- Representations of software architecture are an enabler for communication between all stakeholders interested in the development of a computer-based system.
- The architecture highlights early design decisions that will have a profound impact on all software engineering work that follows and as important, on the ultimate success of the system as an operational entity.
- Architecture constitutes a relatively small, intellectually graspable model of how the system is structured and how its components work together.

5.2.1.4 Steps of Architectural Design

Architectural design begins with data design and then proceeds to the derivation of one or more representations of the architectural structure of the system. Alternative architectural styles or patterns are analyzed to derive the structure that is best suited to customer requirements and quality attributes. Once an alternative has been selected, the architecture is elaborated using an architectural design method.

5.2.1.5 Assessment of Correctness

At each stage, software design work products should be reviewed for clarity, correctness, completeness, and consistency with requirements and with one another.

5.2. 2 Component-Level Design

The component level design is started only after the first iteration of the architectural level has been completed. At this stage, the overall data and program structure of the software has been established. The intent of this level is to translate the design model into operational software.

5.2.2.1 Definition

The architectural design level defines a complete set of software components. But the level of abstraction of the internal data structure and processing details of each component are relatively high. Component-level design defines the data structures, algorithms, interface characteristics, and communication mechanisms allocated to each software component.

5.2.2.2 Who does it

The component level is designed by the software engineers.

5.2.2.3 Importance

The engineers should determine beforehand whether the software would work before they build it. The component-level design represents the software in a way that allows them to review the details of the design for correctness and consistency with other design representations that is the data, architectural, and interface designs. It provides a means for assessing whether data structures, interfaces, and algorithms will work.

5.2.2.4 Steps of Component-Level Design

The foundation for the component level design is formed by the design representations of the data, architecture and interfaces. The class definition or processing narrative for each component is translated into a detailed design that makes use of diagrammatic or text-based forms that specify internal data structures, local interface detail, and processing logic. Design notation encompasses UML diagrams and supplementary forms. Procedural design is specified using a set of structured programming constructs.

5.2.2.5 Assessment of Correctness

To assure the correctness of the design, a design review is conducted. The design is examined to determine whether data structures, interfaces, processing sequences, and logical conditions are correct and will produce the appropriate data or control transformation allocated to the component during earlier design steps.

5.2.3 User Interface Design

User interface design is the design applications such as websites, mobile communication devices, software etc. with the focus on the user's experience and interaction. The goal of user interface design is to make the user's interaction as simple and efficient as possible, in terms of accomplishing user goals. Good user interface design facilitates finishing the task at hand without drawing unnecessary attention to itself. Graphic design may be utilized to support its usability. The design process must balance technical functionality and visual elements to create a system that is not only operational but also usable and adaptable to changing user needs.

5.2.3.1 Definition

User interface design creates an effective communication medium between a human and a computer. It focuses on anticipating what users might need to do and ensuring that the interface has elements that are easy to access, understand, and use to facilitate those actions. Following a set of interface design principles, design identifies interface objects and actions and then creates a screen layout that forms the basis for a user interface prototype.

5.2.3.2 Who does it

The user interface design is done by a software engineer by applying an iterative process that draws a predefined design principles.

5.2.3.3 Importance

If software is difficult to use, if it forces the users into mistakes, or if it frustrates their efforts to accomplish their goals, they won't like it, regardless of the computational power it exhibits, the content it delivers, or the functionality it offers. The interface has to be right because it molds a user's perception of the software.

5.2.3.4 Steps of User Interface Design

User interface design begins with the identification of user, task, and environmental requirements. Once user tasks have been identified, user scenarios are created and analyzed to define a set of interface objects and actions. These form the basis for the creation of screen layout that depicts graphical design and placement of icons, definition of descriptive screen text, specification and titling for windows, and specification of major and minor menu items. Tools are used to prototype and ultimately implement the design model, and the result is evaluated for quality.

5.2.3.5 Assessment of Correctness

An interface prototype is test driven by the users and feedback from the test drive is used for the next iterative modification of the prototype.

5.3 Design Process for TwirlingHall.com

5.3.1 Architectural Design

The architectural design of the TwirlingHall.com is done following the structured design. This design is characterized as data flow oriented design method because it provides a convenient transition from a dataflow diagram to software architecture. At first the transaction flow or transform flow is identified from the dataflow diagram. Based on that the architectural design is conducted.

- **Transform flow:**

  In transform flow the data flows in sequential manner and follows one straight line.

- **Transaction flow:**

  In this flow data flow has a single transaction node that triggers other data flow.

The architectural design for TwirlingHall.com is given below.

Architectural design level 0

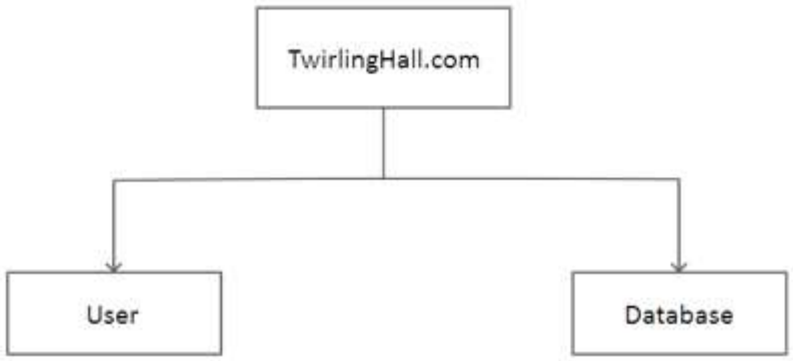


Figure 5.1: Architectural Design level 0

Here, user is the input controller because user give input and database show output. So, database is the output controller. TwirlingHall.com process user input and saved in database.

Architecture level 1

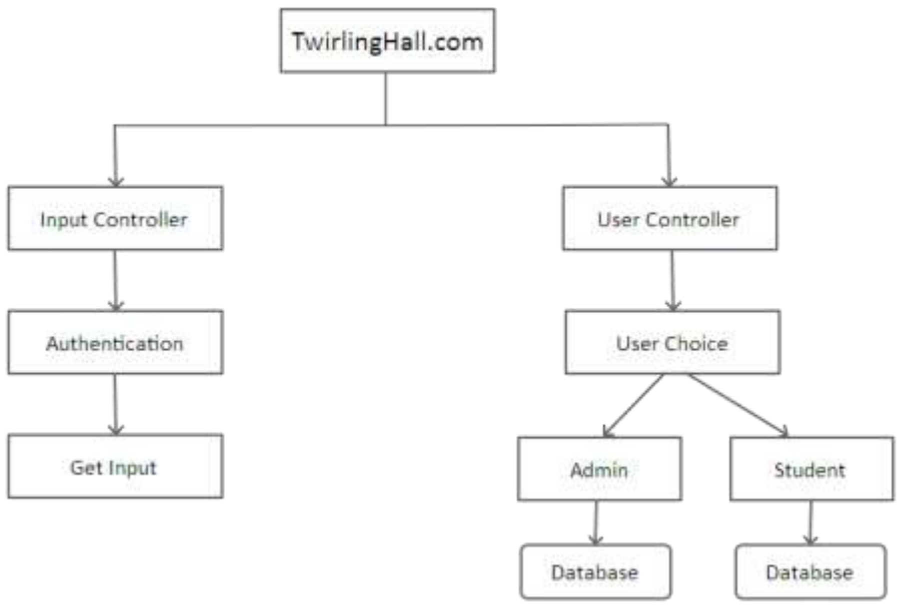


Figure 5.2 Architectural Design Level 1

Input controller get input from authentication. Authentication is a transaction point. User controller process user input. After user choice user type is saved in database.

Architecture level 2.1

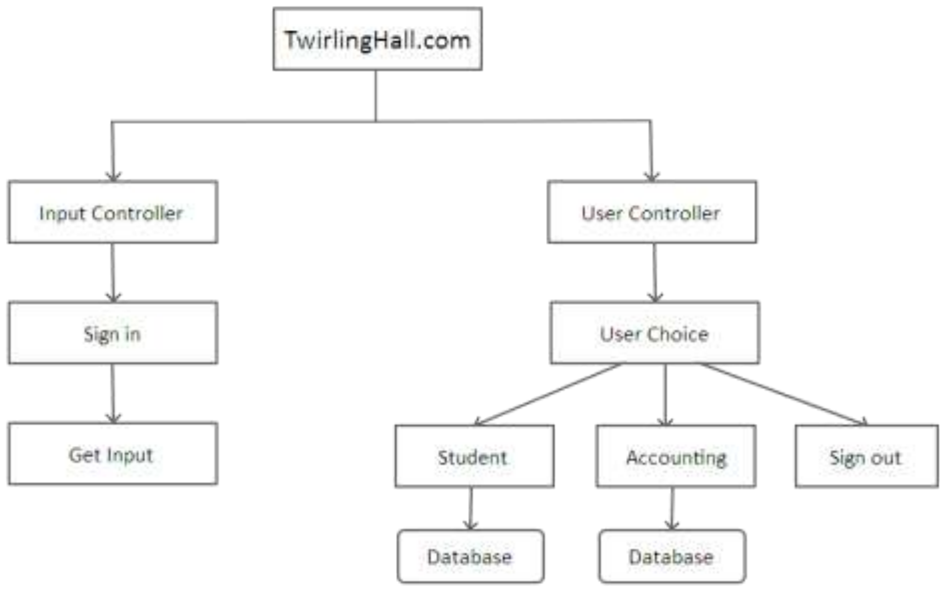


Figure 5.3 Architectural Design Level 2.1

Here, input controller gets input when admin signs in. After sign in there is a transaction point. After choice user controller processes input and save in database.

Architectural design 2.2

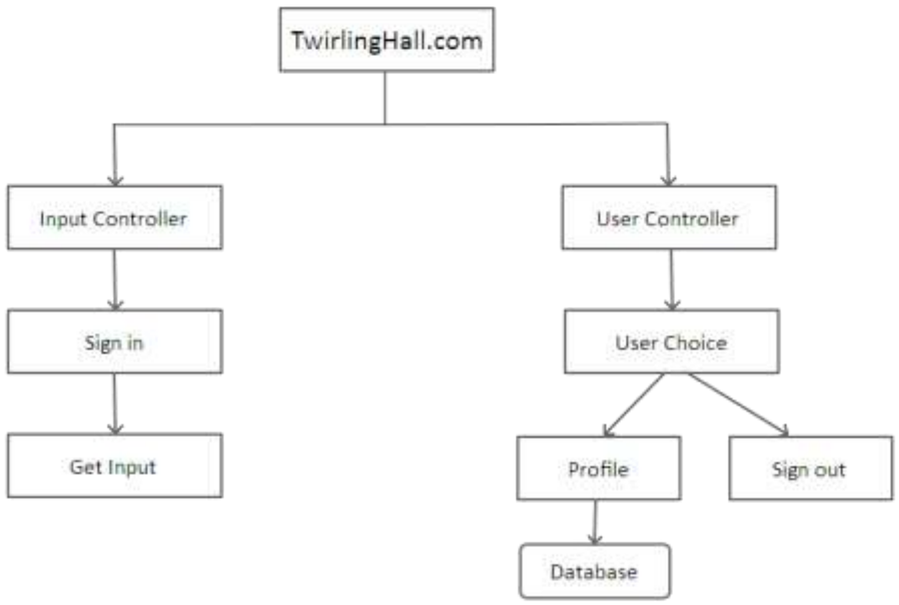


Figure 5.4: Architectural Design Level 2.2

Here, input controller gets input when a student signs in. After sign in there is a transaction point. After choice user controller processes input and save in database.

Architectural design 3.1

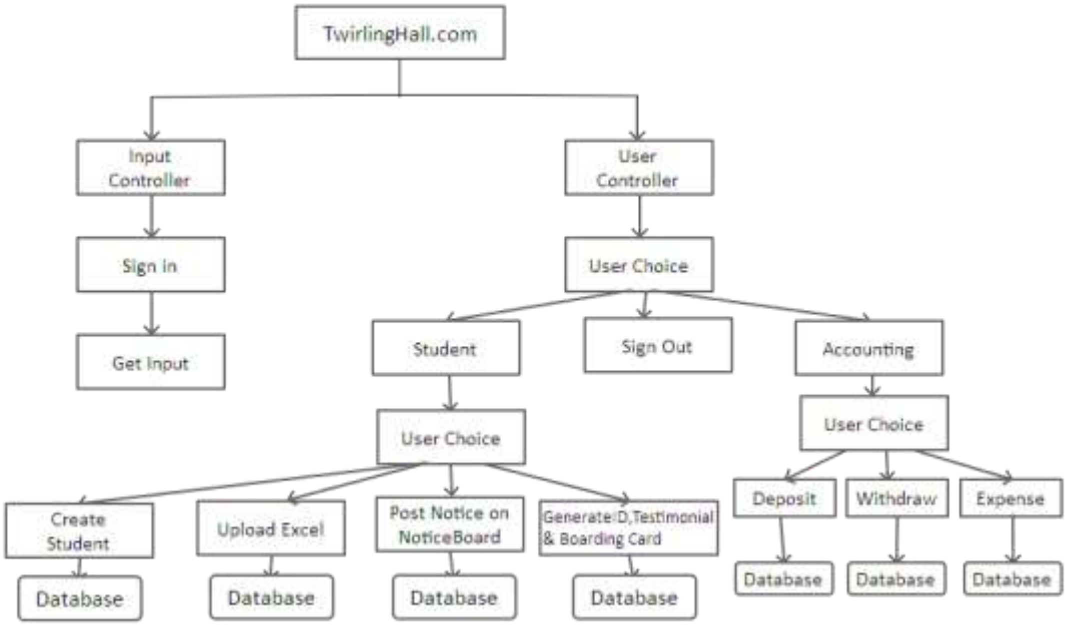


Figure 5.5 Architectural Design Level 3.1

Here, input controller gets input when admin signs in. After sign in there is a transaction point. After choosing the option there are another transaction points. One is in student and another is in accounting section. After choice user controller processes input and save in database.

Architectural Design 3.2

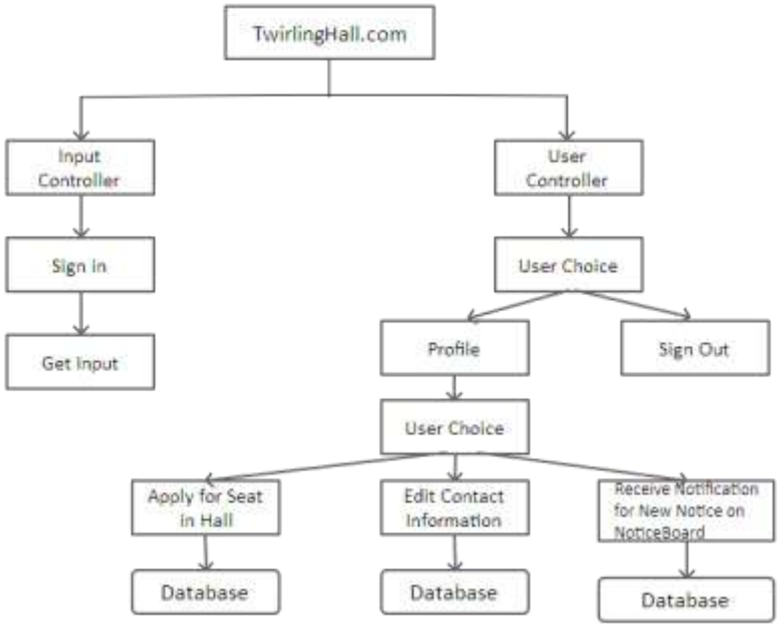


Figure 5.6 Architectural Design Level 3.2

Here, input controller gets input when students sign in. After sign in there is a transaction point. After choosing the option there is another transaction point which is the profile section. After entering profile section there is another transaction point. After choice user controller processes input and save in database.

5.3.2 User Interface Design

The user interface design for web application TwirlingHall.com is given below.

5.3.2.1 Rough sketch of TwirlingHall.com interface

First we develop rough sketch for our web site interface.These are some example.

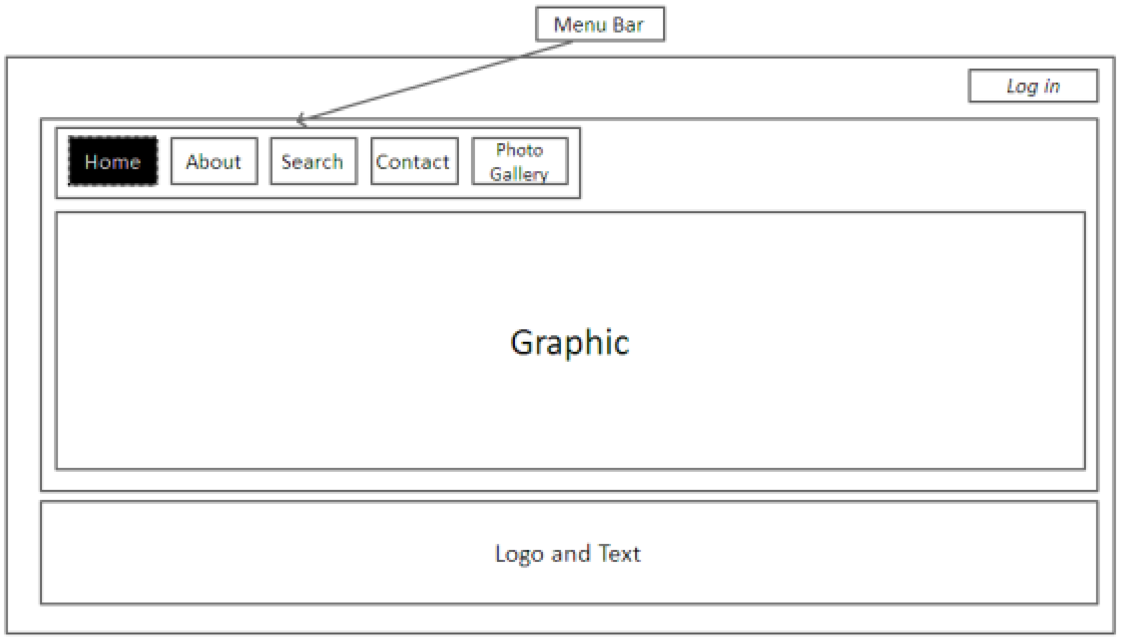


Figure 5.7: Rough sketch for Home Page

- **Mapping user objectives into interface actions of home**

The user objectives and the interface are mapped below.

1. Home: This menu item enables a user to see home page of TwirlingHall.com.

2. About: This menu item enables a user to see about page of TwirlingHall.com. In about page a user can find basic information about TwirlingHall.com.

3. Search: This menu item enables a user to search students in TwirlingHall.com.

4. Contact: This menu item enables a user to see contact page of TwirlingHall.com. In contact page a user can find contact information of the hall.

5. Log in: This link button enables a user to log in the website using username and password.

➢ Design Description:

Menu bar is the heart of a dynamic website. When anyone at first enters in a web site this is a first thing that a user searches. Menu bar contain menu items which represent what a web site can serve. For this reason we place it at the top of the pages.
Log in button is a link button. This button is separately highlighted. Because registered users may not have interest on visiting external part of the site. They need to visit their profile. So, they need login button.

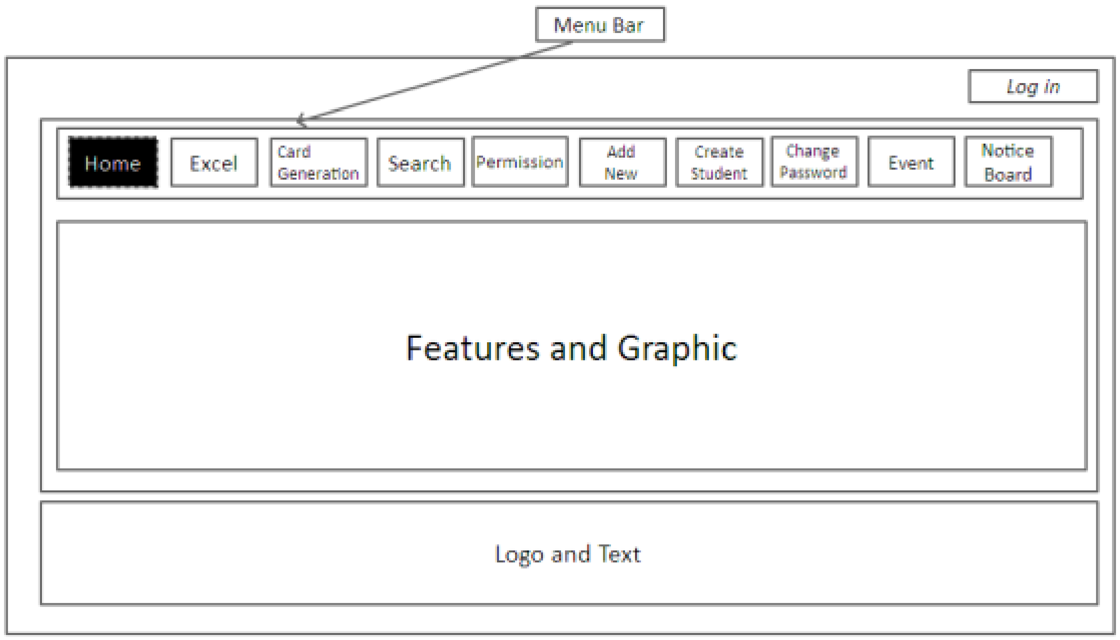


Figure 5.8: Rough sketch for Admin Profile

➢ Mapping user objectives into interface actions of student profile:

The user objectives are mapped into the interface actions of profile below.

1. Home: This menu item enables a user to see home page of TwirlingHall.com.
2. Excel: This menu item enables the admin to upload excel file to the system.
3. Card Generation: Admin can generate ID card, boarding card and testimonial of the students in this item.
4. Search: Admin can search students of the hall in this item by different means such as by name, by department, by room number, by blood group etc.

5. Permission: Through the help of this item, admins can give permissions to the students.
6. Add New: In this menu item admin can add new faculty, department, session, room-number and institute.
7. Create Students: Admin can create new students' profile with the help of this item.
8. Change Password: The password of the profile of admins can be change in this menu item.
9. Event: New events can be created by the admin through this menu item.

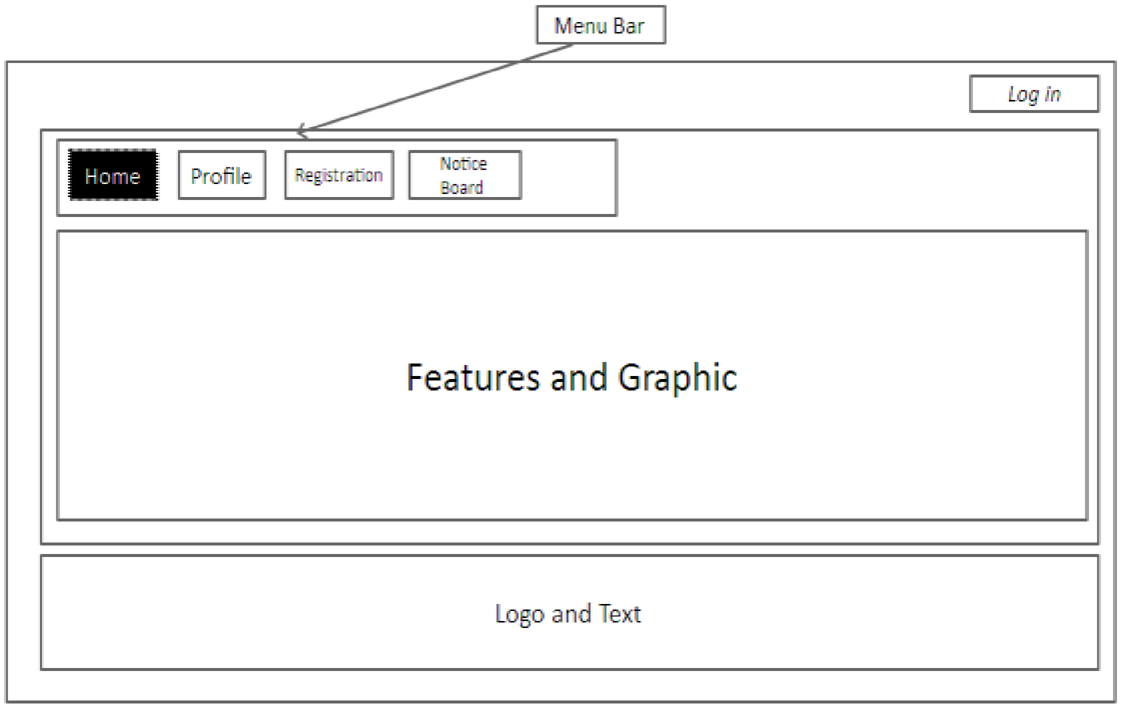


Figure 5.9: Rough sketch for Student Profile

➢ **Mapping user objectives into interface actions of teacher profile:**
The user objectives are mapped into the interface actions of teacher profile is as below.

1. Home: This menu item enables students to see home page of TwirlingHall.com.
2. Profile: This link button enables a student to update her information.

Chapter 6: Project Risk Management

This chapter is about risk management plan. This chapter consists of identification of risks, probability and impact of those risks using risk register (RR) and risk matrix (RM). It also discusses the plan to mitigate those risks.

6.1 Introduction

In software engineering, risk may create various future harms that could be possible on the software due to some minor or non-noticeable mistakes in software development project or process. Software projects have a high probability of failure so effective software development means dealing with risks adequately. In order to, this we want to clearly identify all the risk which may occur.

Software risks could be classified as internal and external. Those risks that come from risk factors within the organization are called internal risks whereas the external risks come from out of the organization and are difficult to control. Internal risks are project risks, process risks, and product risks. External risks are generally business with the vendor, technical risks, customers' satisfaction, political stability and so on.

Some of most important risks in software engineering project are categorized as software requirement risks, software cost risks, software scheduling risk, software quality risks, and software business risks. These risks are explained detail below (Hoodat, H. & Rashidi).

We create a risk register in where different kinds of risks may accumulate in our system. After getting a risk register we will create a risk matrix.

6.2 Software Risk Identification

Identifying risks is an absolutely essential activity for all software projects. A risk is an event which is unpredictable and which has negative consequences. There are several techniques for risk identification. We follow the technique described below.

At a very high level of granularity, we take each of the four components of our project (time, cost, quality, features) and ask ourselves, what could happen to make any of these four components go wrong. For example, "In our project, what events would make me go over budget?" Pros: easy, good way to get started. Cons: too high level to catch all risks.

Software development, given the intangible nature and uniqueness of software, is inherently difficult to estimate and schedule. As we have limited time the scheduling is also a risk for our project.

Requirement inflation that means as the project progresses more and more features that were not identified at the beginning of the project emerge that threaten estimates and timelines.

Employee turnover that means one of our project member's illness or absence in any reason may hamper our progress of development.

Many other risks may be appeared in project development and maintenance process. All the probable risks are presented in risk register at section 5.3.

6.3 Software Risk Register

A Risk Register is a Risk Management tool commonly used in Project Management and organizational risk assessments. It acts as a central repository for all risks identified by the project or organization and, for each risk, includes information such as risk probability, impact, counter-measures, risk owner and so on.

Table 6.1: Software risk register

Id	Risk	Probability	Impact	Exposure
1	Lack of analysis for change of requirements	Low	High	Medium
2	Change of requirements	Low	Medium	Low
3	Poor definition of requirements	Medium	High	High
4	Invalid requirements	Low	Medium	Low
5	Lack of good estimation in projects	Medium	High	High
6	Human errors	Medium	Medium	Medium

7	Lack of testing	Medium	Medium	Medium
8	Inadequate budget	High	High	High
9	Inadequate knowledge about tools and techniques	Medium	Medium	Medium
10	Lack of employment of project manager experience	Low	Medium	Low
11	Lack of enough skill of users	Medium	Low	Low
12	Infrastructure failures such as Server crash, hardware	High	Medium	High
13	Political risks	High	High	High

6.4 Software Risk Matrix

Table 6.2: Risk Matrix

Severity

	L	**M**	**H**
L	**LL**	**ML** **11**	**HL**
M	**LM** **2,4,10**	**MM** **6,7,9**	**HM** **12**
H	**LH** **1**	**MH** **5**	**HH** **8,13**

Probability of occurrenc

6.5 Risk Mitigation Plan

Id	Risk	Mitigation plan
1	Lack of analysis for change of requirements	1. This project follows agile process. Before every big change clients examine this system and give their comments. As a result changing requirements affect less to development.
2	Change of requirements	1. This project follows agile process. Before every big change clients examine this system and give their comments. As a result changing requirements affect less to development.
3	Poor definition of requirements	1. Project manager showed paper pencil prototype for better understanding of clients.
4	Invalid requirements	1. All requirements are tested by quality assurance staff so there is less possibility for invalid requirement.
5	Lack of good estimation in projects	1. Working progress of all members is evaluated weakly. 2. Clients are notified after completion of each single sub module. So, they can give advice in case of bad estimation of any criteria of project.
6	Human errors	1. System will give message before any important operation (example-delete operation, update operation). 2. System will also be checked by the different hall staff so there is less possibility of human error.
7	Lack of testing	1. Developers unit test report will be checked by both PM and QA staff. 2. Requirements with high priority will be checked deeply first.

8	Inadequate budget	1. Important features will be developed first. So, system will not become wastage to the clients if development will stop for inadequate budget.
9	Inadequate knowledge about tools and techniques	1. Project will use well known tools and technologies. So, helpful resources will be available.
10	Lack of employment of project manager experience	1. PM will gather knowledge of project management from helpful resources.
11	Lack of enough skill of users	1. A well written user guide will be provided with the system. 2. Project team members will help users to interact with the system.
12	Infrastructure failures such as Server crash, hardware	1. Backup of database will be kept every day. 2. Use of distributed database.
13	Political risks	1. Hall authority will take concious step in order to mitigate hall internal political issues.

6.6 Conclusion

In the above sections, software risk management, risks classification in this project, are clearly described. If risk management process is in place for each and every software development process then future problems could be minimized or completely eradicated. Hence, understanding various factors under risk management process and focusing on risk management strategies explained above could help in building risk free products in future.

Chapter 7: Security Requirements Engineering

This chapter presents security requirement analysis. This chapter describes the security requirements with extensive analysis and proper testing on the basis of OWASP. Moreover, top ten security vulnerability, their testing process for this project and mitigation plan are described in this chapter.

7.1 Introduction

Open Web Application Security Project (OWASP) is a worldwide not-for-profit charitable organization which focused on improving the security of software. The mission of this organization is to make software more secure by providing security issues and this is done to concern worldwide individuals and organizations about true software security risks.

7.2 The OWASP Top Ten

OWASP Top Ten is a list of 10 most dangerous current Web application security flaws, along with effective methods of dealing with those flaws. Varieties of security experts from around the world are involved who share their knowledge of vulnerabilities, threats, attacks and countermeasures.

7.2.1 Unvalidated Input

Information taken from web requests is not validated before being used by a web application. Attackers can use these kinds of flaws to attack backend components through a web application. The requests may come from different location such as, URL, query string, form fields, hidden fields, cookies, and headers.

Fig 7.1: Unvalidated user login page

7.2.2 Broken access control

Access control defines the restrictions on what authenticated users are allowed to not properly enforce. Attackers can exploit these flaws to access other users' accounts, view sensitive files, or use unauthorized functions. The problem occurs because many applications provide authentication, but don’t handle access control well due to having a complex access control policy

7.2.3 Broken authentication and session management

In this case, account credentials and session tokens are not properly protected. Attackers can take passwords, keys, session cookies, or other tokens can defeat authentication restrictions and assume other users' identities. To resolve this problems account and session management policies should be defined.

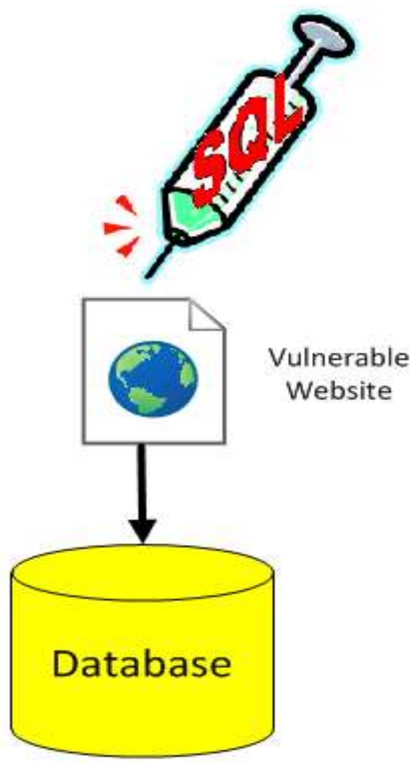


Fig 7.2: Breaks authentication

7.2.4 Cross site scripting (XSS) flaws

Cross-site scripting (XSS) is a type of computer security vulnerability which enables attackers to inject client side script into Web pages viewed by other users. The vulnerabilities may be used by attackers to bypass access controls.

Fig 7.3: Cross site scripting

7.2.5 Buffer overflows

Web application components that do not properly validate input can be crashed and, in some cases, used to take control of a process. The vulnerable languages for buffer overflow is c, c++.

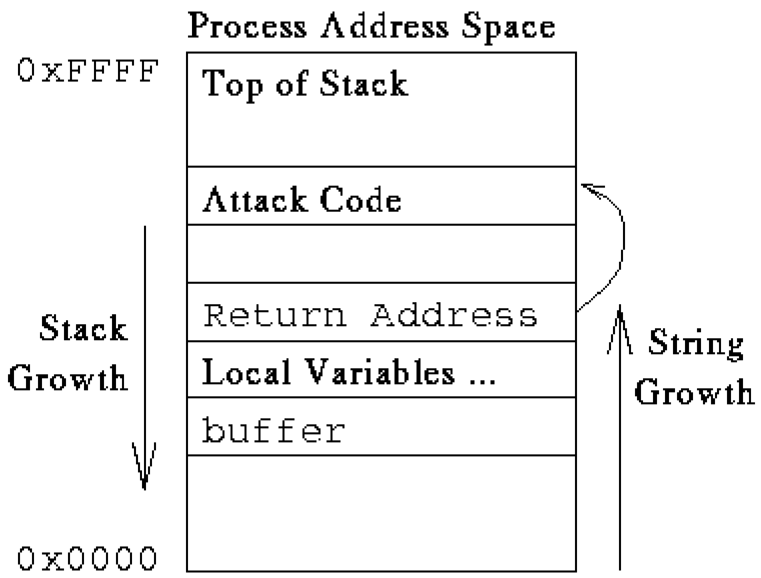


Fig 7.4: Buffer overflow

7.2.6 Injection flaws

Web applications pass parameters during the access of external systems. If an attacker provides malicious commands in these parameters, the external system may execute those commands on behalf of the web application which is not desirable. Malicious code may be sent via HTTP request.

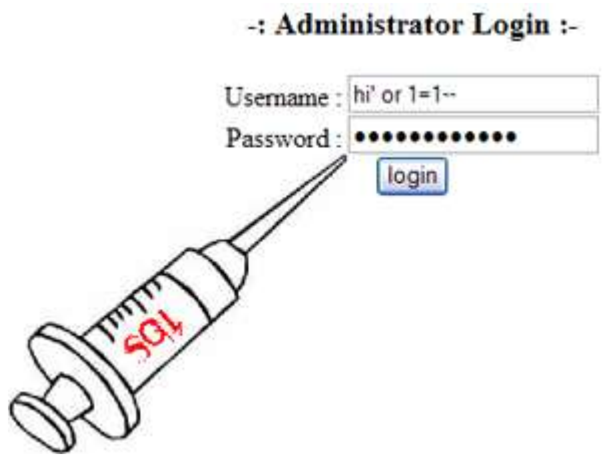


Fig 7.5: Sql injection

7.2.7 Improper error handling

Error conditions that occur during normal operation are not handled properly. If an attacker can cause errors to occur that the web application does not handle, they can gain detailed system information, deny service, cause security mechanisms to fail, or crash the server. Out of memory, Database failure those are the common failure incurred by the attacker. To mitigate this problem, keeping log is one of the choices.

7.2.8 Insecure storage

Web applications frequently use cryptographic functions to protect information and credentials. Sometime, developers think that a new invented algorithm will perform better than existing but there exist lot of vulnerabilities. Besides this, the password should not store in plain text, rather it should be kept in hashed.

7.2.9 Denial of service (DoS)

If the Web application faced DOS attack, then legitimate users can no longer access or use the application. Attackers continuously send request to the web application and due to respond corresponding request, the web application become busy and thus the entire application may fail. It is much more difficult to protect the website from DOS attack. Distributed DOS is one kind of DoS attack where many zombie PC will be created and continuous requests are sent to the server.

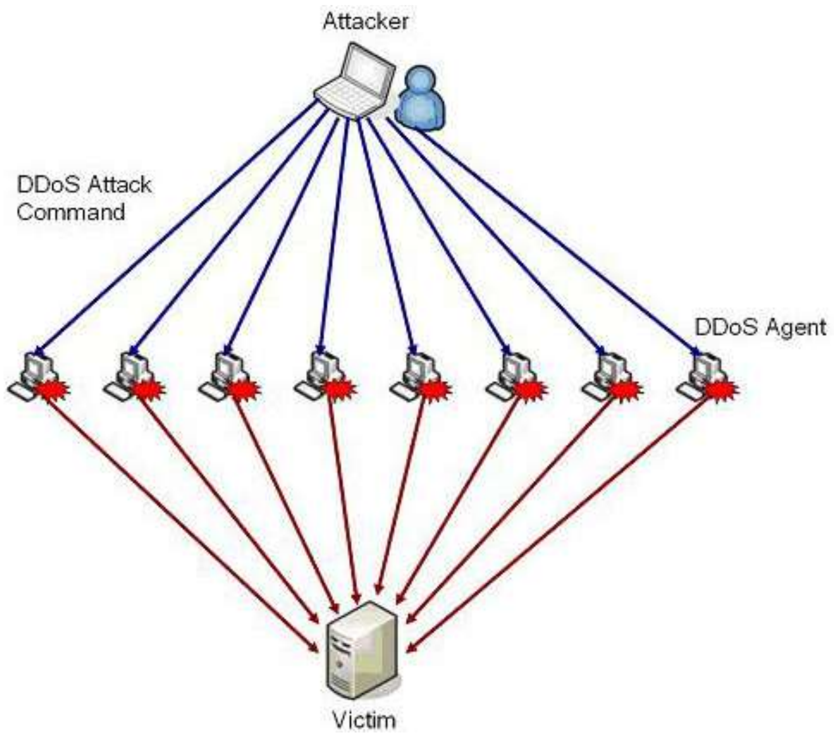


Fig 7.6: DDoS attack

7.2.10 Insecure configuration management

Servers have many configuration options that affect security and are not secure out of the box. To secure the server unnecessary default, backup, libraries, informative error messages should not be kept. To test the security of our developed "Kabi Sufia Kamal Hall" project, we need to define the use case and misuse case of the project.

7.3 Security Use & Misuse Case

Misuse Case is a business process modeling tool used in the software development industry to test the software from attacker points of view. The term *Misuse Case* is derived from and is the inverse of use case [4]. The main purpose of generating the misuse case is to communicate potential risks to stakeholder. Besides this, if it is possible to determine the probable attack in advance, then prevention can also be taken into account.

To test the security of our developed "Kabi Sufia Kamal Hall" project, we need to define the use case and misuse case of the project.

7.3.1 Security use and misuse case (Level 1)

Fig 7.7 represents some of the modules of our project which are more probable to fell in the attack. This figure provides an overview of relationship between user and attacker.

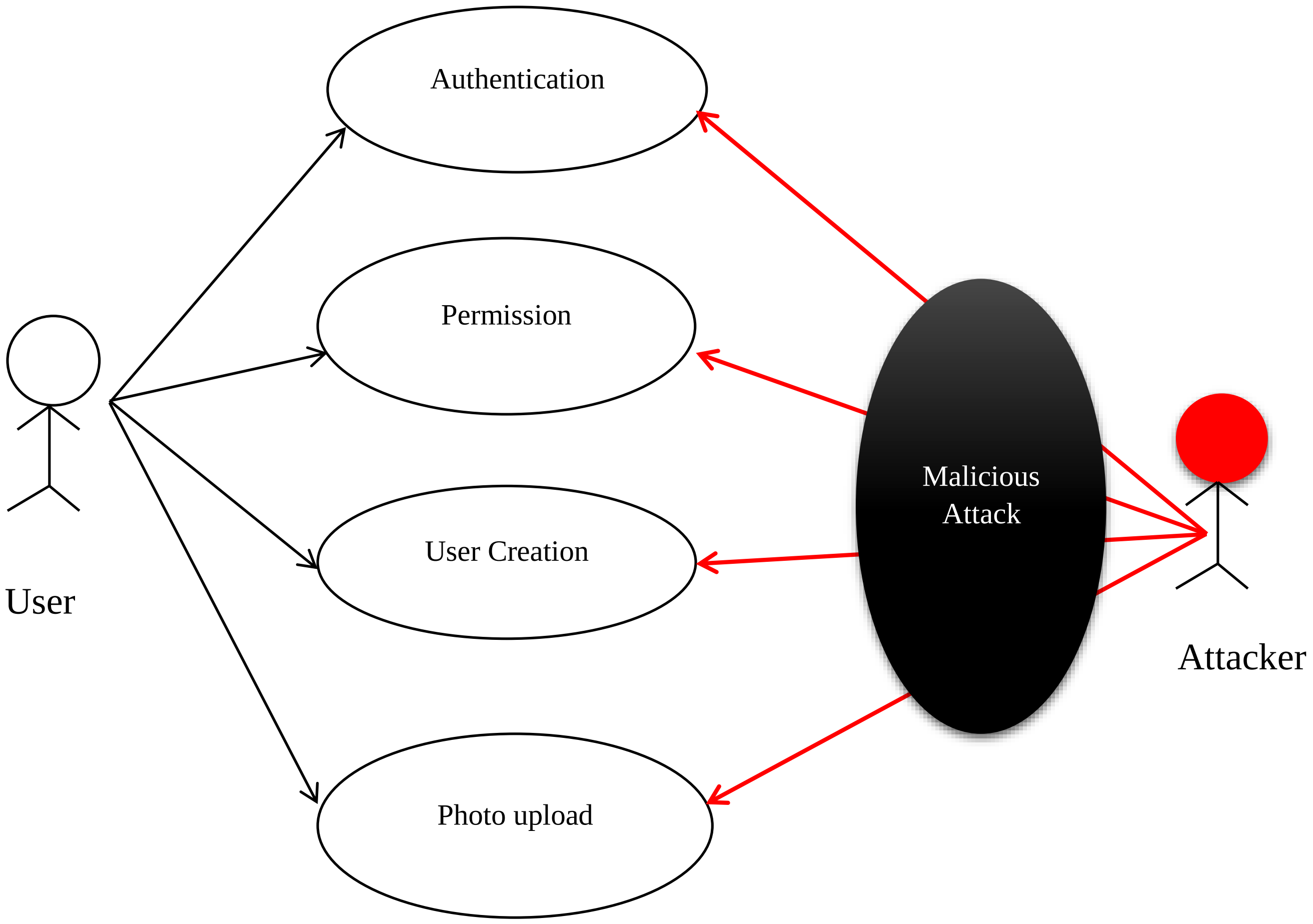


Fig. 7.7: Security use and misuse case (Level 1)

7.3.2 Authentication security use and misuse case (Level 2)

Fig 7.8 is a granular level of Fig 7.7 and it represents how attacker can attack in our system. From the figure, it is clear that attacker may attack the dictionary attack into the system by getting the length of password. Attacker can guess password by deriving information from generic error message. Moreover, using bruit force attack authentication can be bypassed.

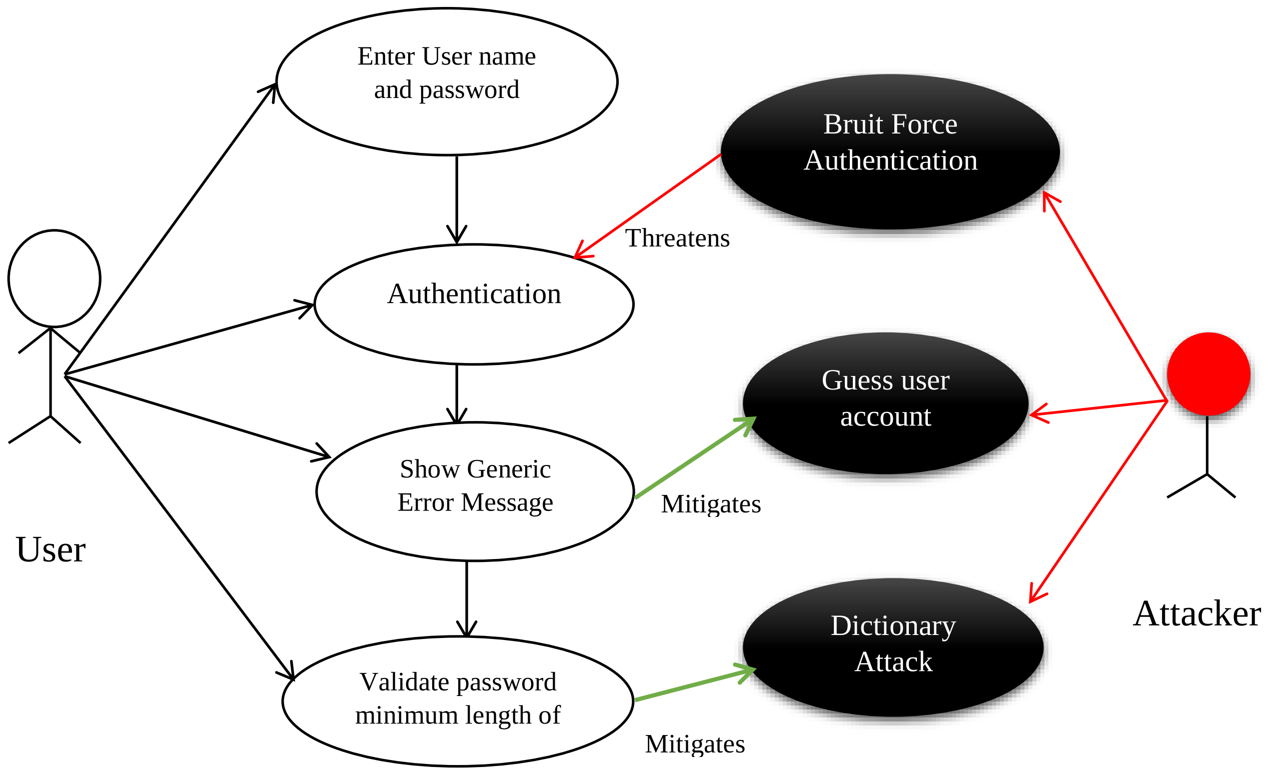

Enter User name and password
Authentication
Show Generic Error Message
Validate password minimum length of
User
Bruit Force Authentication
Threatens
Guess user account
Mitigates
Dictionary Attack
Mitigates
Attacker

Fig. 7.8: Authentication security use and misuse case (Level 2)

7.3.3 User creation security (SQL Injection) (Level 2)

Fig 7.3 is a granular level of Fig 7.1 and it represents that how attacker can create a new user into the system. Through SQL injection the attacker can insert a user name and corresponding information as unauthorized users.

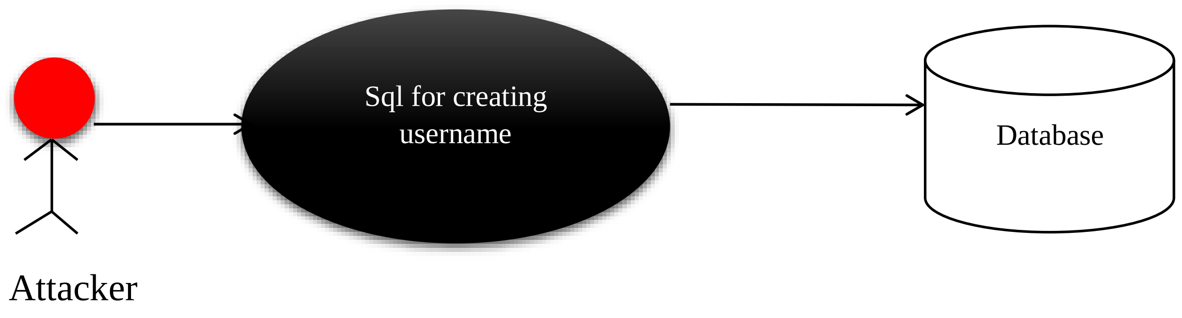


Fig. 7.9: User creation security (SQL Injection) use and misuse

7.3.4 User creation security (Interception with SQL Injection) (Level 2)

Fig 7.10 represents that a valid admin tries to create another user or admin into the system. But the attacker intercepts the message and understands the user creation pattern by analyzing the traversed packets. Then the attacker changes that user name and password as he wishes. If this is happened, when a legal user will try to enter into the system using his or her previously created account, he/she cannot enter into the system whereas the attacker can easily access into the system using his or her provided user name and password.

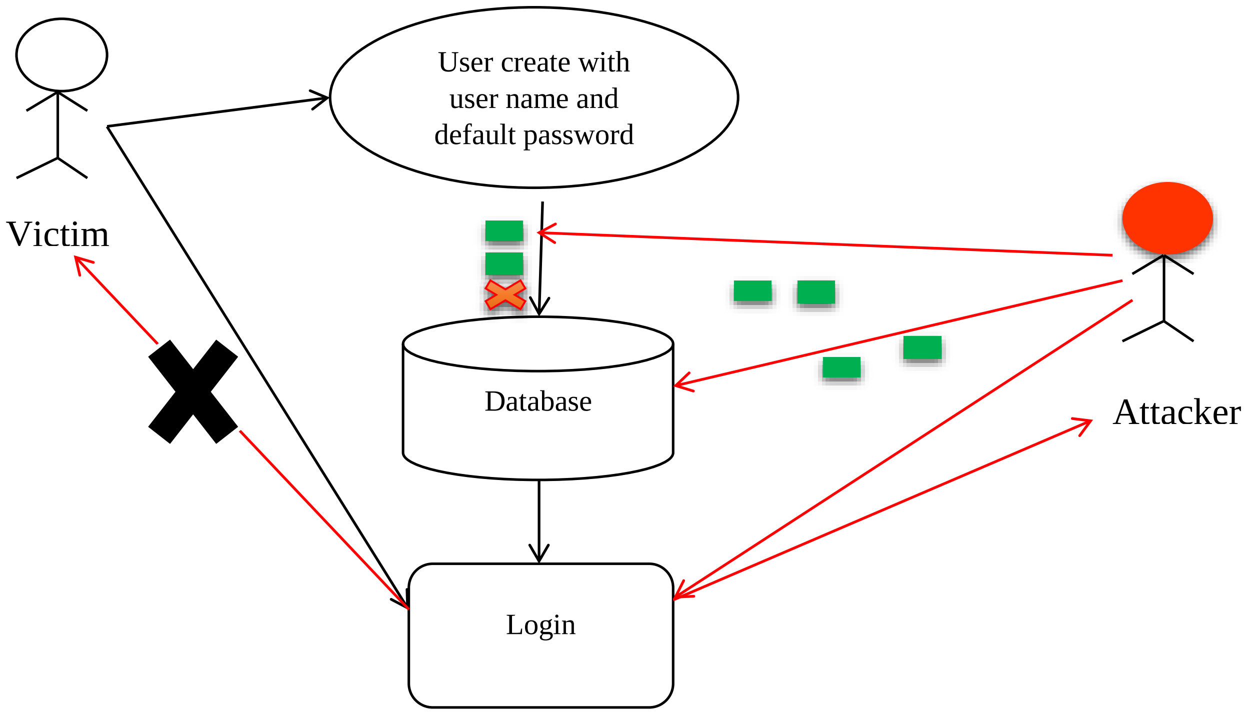


Fig 7.10: User creation security (Interception with SQL Injection) use and misuse case (Level 2)

7.3.5 Photo upload security (Cross site scripting) (Level 2)

Fig. 7.11 represents a misuse case of photo upload scenario. If a malicious user gets permission to the system then he or she can upload new photo in our system where a link is embedded behind the image and that link is harmful for a user. When a validate user or student wants to see the photo and if he or she clicks on the photo then he or she will be redirected into the malicious website.

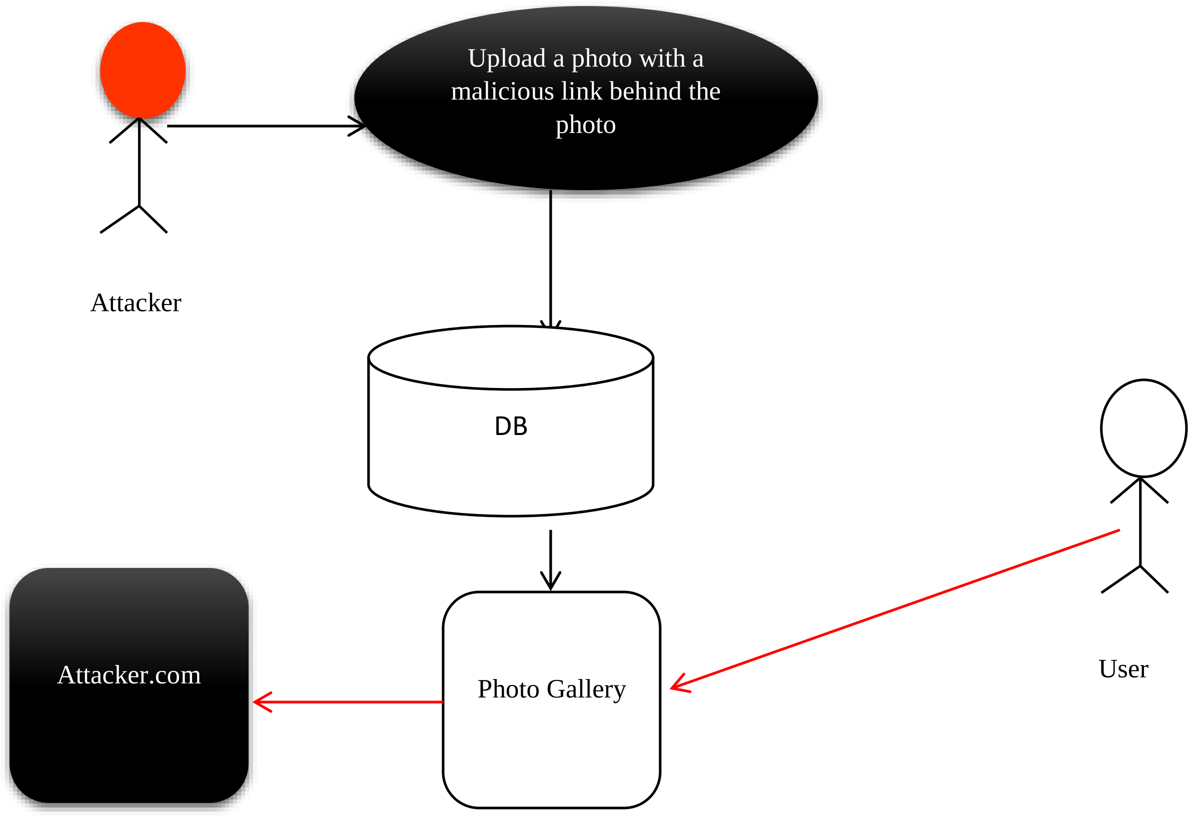


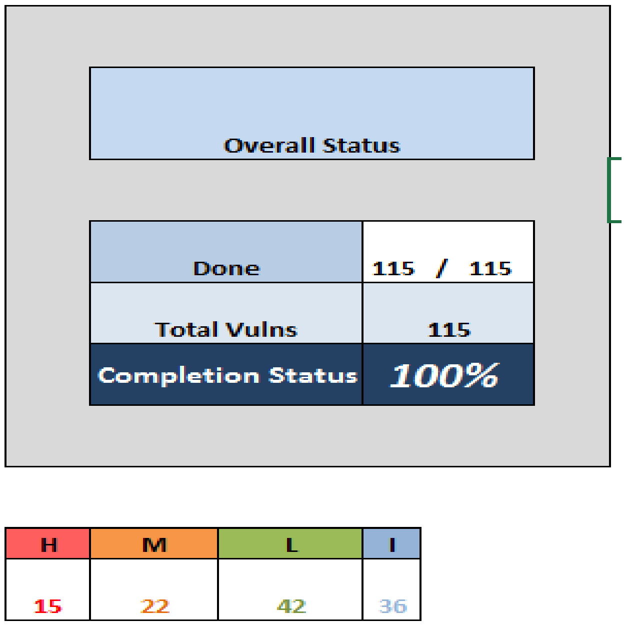
Fig. 7.11: Photo upload security (Cross site scripting) use and misuse case (Level 2)

7.4 Security Checklist of “Kabi Sufia Kamal Hall” Project

Test Name	Ref. Number	Status	Risk
Checking the caches of major search engines for publicly accessible sites - Search Engine Discovery	SKIG-001	Done	**L**
Checking Webpage Comments and Metadata for Information Leakage - Information Leakage	SKIG-002	Done	*I*
Checking Web Application Framework - Application Framework	SKIG-003	Done	**L**
Identifying technologies used – Technology	SKIG-004	Done	**H**
Identifying application entry points - Application entry points	SKIG-005	Done	**L**
Identifying client-side code and user roles - Client-side code and user roles	SKIG-006	Done	**M**
Identifying all hostnames and ports - Hostnames and ports	SKIG-007	Done	**M**
Checking for commonly used application and administrative URLs - commonly used application and administrative URLs	SKCM-001	Done	*I*
Checking HTTP methods supported and Cross Site Tracing (XST) - Cross Site Tracing	SKCM-002	Done	*I*
Infrastructure Configuration Management Testing - Infrastructure Configuration management weakness	SKCM-003	Done	*I*
Application Configuration Management Testing - Application Configuration management weakness	SKCM-004	Done	*I*
Testing for File Extensions Handling - File extensions handling	SKCM-005	Done	**M**
Old, backup and unreferenced files - Old, backup and unreferenced files	SKCM-006	Done	**M**
Check SSL Version, Algorithms, Key length - SSL, algorithm weakness	*SKST-001*	Done	**H**
Check for Digital Certificate Validity - Duration, signature	*SKST-002*	Done	**H**
Check credentials and login form only delivered over HTTPS - Credentials and login form	*SKST-003*	Done	**L**
Testing for user enumeration - User enumeration	*SKAT-001*	Done	*I*
Testing for Guessable (Dictionary) User Account - Guessable user account	*SKAT-002*	Done	**M**
Testing for brute force protection - Credentials brute forcing	*SKAT-003*	Done	**M**
Testing for Credentials Transported over an Encrypted Channel - Credentials Transported over an Encrypted Channel	*SKAT-004*	Done	**H**
Testing password quality rules - Password combination of upper, lower case, numerical and special character	*SKAT-005*	Done	**H**
Testing CAPTCHA - Captcha implementation	*SKAT-006*	Done	**L**
Testing Email address as a User ID - Validate and verify Email address	*SKAT-007*	Done	**H**
Testing remember me functionality - Vulnerable remember password	*SKAT-008*	Done	**M**
Testing for autocomplete on password forms/input - Autocomplete password and user id field	*SKAT-009*	Done	**H**
Testing for bypassing authentication schema - Bypassing authentication schema	*SKAT-010*	Done	*I*
Testing password reset or recovery - Weak Password Recovery Mechanism	*SKAT-011*	Done	**H**
Testing password change process - Weak password change process	*SKAT-012*	Done	**M**
Testing multi factor authentication - Weak multi factor authentication	*SKAT-013*	Done	*I*
Testing logout functionality presence - Logout function not properly implemented	*SKAT-014*	Done	**L**
Testing for cache management on HTTP - browser cache weakness	*SKAT-015*	Done	**L**
Testing for default logins - Default login	*SKAT-016*	Done	**L**
Testing for user-accessible authentication history - user-accessible authentication history	*SKAT-017*	Done	**L**
Testing for Weak security question/answer - Weak security question/answer	*SKAT-018*	Done	**H**
Testing for out-of channel notification of account lockouts and successful password changes - Out-of channel notification	*SKAT-019*	Done	*I*
Testing for Race Conditions - Race Conditions vulnerability	*SKAT-020*	Done	*I*

Test for consistent authentication across applications with shared authentication schema / SSO and alternative channels - Consistent authentication	*SKAT-021*	Done	*I*
Testing session cookie duration - Expire and Max-Age	*SKSM-001*	Done	**L**
Testing session cookie scope (path and domain)	*SKSM-002*	Done	*I*
Testing session tokens for cookie flags - httpOnly and secure or no time validity	*SKSM-003*	Done	**H**
Testing how session management is handled in the application (eg, tokens in cookies, token in URL)	*SKSM-004*	Done	**L**
Testing for Session Management Schema - Bypassing Session Management Schema, Weak Session Token	*SKSM-005*	Done	**M**
Testing session termination after a maximum lifetime - Session auto termination	*SKSM-006*	Done	**L**
Testing session termination after logout - vulnerable session	*SKSM-007*	Done	**L**
Testing session termination after relative timeout - Session relative timeout termination	*SKSM-008*	Done	**M**
Testing to see if users can have multiple simultaneous sessions	*SKSM-009*	Done	**M**
Testing session cookies for randomness - random session cookies	*SKSM-010*	Done	**L**
Testing for Session Fixation - Session Fixation	*SKSM-011*	Done	**L**
Testing for Exposed Session Variables - Exposed sensitive session variables	*SKSM-012*	Done	*I*
Testing for session puzzling - session can be puzzled	*SKSM-013*	Done	*I*
Testing new session tokens are issued on login, role change and logout	*SKSM-014*	Done	**L**
Testing for consistent session management across applications with shared session management	*SKSM-015*	Done	**L**
Testing for CSRF – CSRF	*SKSM-016*	Done	**L**
Testing for Insecure Direct Object References	*SKAZ-001*	Done	**M**
Testing for missing authorization	*SKAZ-002*	Done	**L**
Testing for Path Traversal - Path Traversal	*SKAZ-003*	Done	*I*
Testing for bypassing authorization schema - Bypassing authorization schema	*SKAZ-004*	Done	*I*
Testing for Privilege Escalation - Privilege Escalation	*SKAZ-005*	Done	**H**
Testing for access between two users at the same privilege level - Horizontal access control	*SKAZ-006*	Done	**M**
Testing for feature misuse - Feature misuse	*SKBL-001*	Done	**L**
Testing for Business Logic - Bypassable business logic	*SKBL-002*	Done	*I*
Testing for lack of non-repudiation – Repudiation	*SKBL-003*	Done	*I*
Testing for trust relationships - Trust relationship	*SKBL-004*	Done	*I*
Testing for integrity of data - Data integrity	*SKBL-005*	Done	**H**
Testing segregation of duties - Segregation of duties	*SKBL-006*	Done	*I*
Testing defenses against application misuse - Application misuse	*SKBL-007*	Done	**L**
Testing Upload of Unexpected File Types - Upload of Unexpected File Types	*SKBL-008*	Done	**L**
SQL Injection - SQL Injection	*SKDV-001*	Done	*I*
LDAP Injection - LDAP Injection	*SKDV-002*	Done	*I*
ORM Injection - ORM Injection	*SKDV-003*	Done	*I*
XML Injection - XML Injection	*SKDV-004*	Done	*I*
SSI Injection - SSI Injection	*SKDV-005*	Done	*I*
XPath Injection - XPath Injection	*SKDV-006*	Done	*I*
Testing for Reflected Cross Site Scripting - Reflected XSS	*SKDV-007*	Done	**L**
Code Injection - Code Injection	*SKDV-008*	Done	**L**
Testing for Stored Cross Site Scripting - Stored XSS	*SKDV-009*	Done	**L**

Testing for DOM based Cross Site Scripting - DOM XSS	*SKDV-010*	Done	**L**
Testing for Cross Site Flashing - Cross Site Flashing	*SKDV-011*	Done	*I*
IMAP/SMTP Injection - IMAP/SMTP Injection	*SKDV-012*	Done	*I*
Testing buffer overflow - Buffer overflow	*SKDV-013*	Done	**L**
Testing for Command Injection - Command Injection	*SKDV-014*	Done	**L**
Testing for Format String - String formatting	*SKDV-015*	Done	**L**
Testing for Open Redirection - Open redirection	*SKDV-016*	Done	**M**
Testing for NoSQL injection - NoSQL injection	*SKDV-017*	Done	*I*
Testing for NULL/Invalid Session Cookie - Vulnerable session	*SKDV-018*	Done	**L**
Testing for XQuery Injection - Xquery injection	*SKDV-019*	Done	*I*
Testing Incubated vulnerability - Incubated vulnerability	*SKDV-020*	Done	*I*
Test for Expression Language Injection - Expression language injection	*SKDV-021*	Done	**L**
Testing for HTML Injection - HTML injection	*SKDV-022*	Done	**L**
Test for Local File Inclusion - Local File Inclusion	*SKDV-023*	Done	**L**
Test for Remote File Inclusion - Remote File Inclusion	*SKDV-024*	Done	**L**
Compare client-side and server-side validation rules - Compare client-side and server-side validation	*SKDV-025*	Done	**M**
Testing for HTTP parameter pollution - HTTP parameter pollution	*SKDV-026*	Done	*I*
Testing for Mass Assignment - Mass assignment	*SKDV-027*	Done	*I*
Testing for anti-automation - Anti-automation	*SKDS-001*	Done	*I*
Testing for SQL Wildcard Attacks - SQL Wildcard vulnerability	*SKDS-002*	Done	**L**
Locking Customer Accounts - Locking Customer Accounts	*SKDS-003*	Done	**M**
Testing for DoS Buffer Overflows - Buffer Overflows	*SKDS-004*	Done	**L**
Testing for account lockout - Weak account lockout	*SKDS-005*	Done	**L**
User Input as a Loop Counter - User Input as a Loop Counter	*SKDS-006*	Done	**L**
Writing User Provided Data to Disk - Writing User Provided Data to Disk	*SKDS-007*	Done	*I*
Storing too Much Data in Session - Storing too Much Data in Session	*SKDS-008*	Done	**H**
Test for HTTP protocol DoS - HTTP protocol	*SKDS-009*	Done	**M**
Check if data which should be encrypted is not - Data protection	*SKCR-001*	Done	**L**
Testing for wrong algorithms usage depending on context - Wrong algorithm	*SKCR-002*	Done	*I*
Testing for weak algorithms usage - Weak algorithm	*SKCR-003*	Done	**M**
Testing for proper use of salting - Salt usage	*SKCR-004*	Done	**M**
Testing for randomness functions - Randomness Function	*SKCR-005*	Done	**M**
Testing acceptable file types are whitelisted - Acceptable files	*SKRF-001*	Done	**L**
Testing file size limits, upload frequency and total file counts are defined and are enforced - File size, upload frequency	*SKRF-002*	Done	**L**
Testing all file uploads have Anti-Virus scanning in-place - Anti-Virus scanning of file	*SKRF-003*	Done	**H**
Testing uploaded files are not directly accessible within the web root - File directory access	*SKRF-004*	Done	**L**
Testing for Insecure Cryptographic Storage - Insecure Cryptographic Storage	*SKRC-002*	Done	**M**
Testing insufficient transport layer protection - Insufficient Transport Layer Protection	*SKRC-003*	Done	**H**
Checking for Error Codes and Stack Trace	*SKEH-001*	Done	**M**



Overall Status	
Done	115 / 115
Total Vulns	115
Completion Status	*100%*

H	M	L	I
15	22	42	36

Fig. 7.12: Vulnerability Statistics

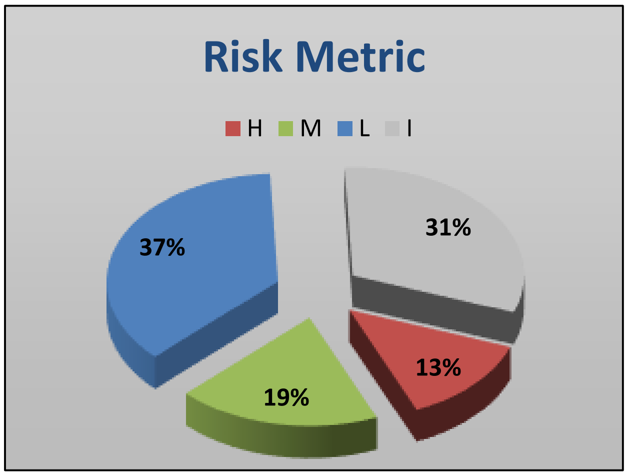


Fig. 7.13: Risk Metric

7.5 Security Testing Summary

Category	**Ref. Number**	**Test Name**	**Vulnerability**	**Tests**	**Tools**
Information Gathering	SKIG-001	Checking the caches of major search engines for publicly accessible sites	N.A.	*Search google with various google dorks using MANUAL INSPECTIONS & REVIEWS*	*Goolag scanner, Google Hacking db (Johny), Goolge, Kartoo*
	SKIG-002	Checking webpage comments and metadata for information leakage	Information Disclosure	*Search webpage comments and metadata for information leakage through MANUAL INSPECTIONS & REVIEWS*	
	SKIG-003	Checking web application framework	N.A.	*Identify framework and check for vulnerabilities through MANUAL INSPECTIONS & REVIEWS*	
	SKIG-004	Identifying technologies used	N.A.	*Identify the used technology through MANUAL INSPECTIONS & REVIEWS*	
	SKIG-005	Identifying application entry points	N.A.	*Google for subdomain and ports discovery, Network Tools or through MANUAL INSPECTIONS & REVIEWS*	
	SKIG-006	Identifying client-side code and user roles	N.A.	*MANUAL INSPECTIONS & REVIEWS*	
	SKIG-007	Identifying all host names and ports	N.A.	*MANUAL INSPECTIONS & REVIEWS*	
Configuration Management Testing	SKCM-001	Checking for commonly used application and administrative URLs	N.A.	*Threat Modeling*	
	SKCM-002	Checking HTTP methods supported and Cross Site Tracing (XST)	XSS attack	*Threat Modeling*	
	SKCM-003	Infrastructure Configuration Management Testing - Infrastructure Configuration management weakness	Infrastructure Confi guration manageme nt weakness	*Understand the infrastructure elements interactions, Admin tools review, Ports used, Version check.*	
	SKCM-004	Application Configuration Management Testing	Application Config uration management weakness	*Only enable server modules, Handle Server errors (40*,50*),Minimal Privilege, Software Logging, Overload Handling against DOS (Logs purging check), Log review*	
	SKCM-005	Testing for File Extensions Handling	File extensions han dling	*Spidering, Googling, Crawling, Manual Inspection*	*Curl, wget, web mirroring tool, Nessus, Nikto*
	SKCM-006	Old, backup and unreferenced files - Old, backup and unreferenced files	Old, backup and unr eferenced files	*Check for On-the-fly backup files created, Check comments, Check JS source code, Random guessing of filename, Directory Listing, Search cached files*	*HTTrack,Wikto/Nikto , Goolag, Spike Proxy*
Secure Transmission	SKST-001	Check SSL Version, Algorithms, Key length	SSL weakness		
	SKST-002	Check for Digital Certificate Validity	Invalid Certificate		
	SKST-003	Check credentials and login form only delivered over HTTPS	Credentials did not deliver over HTTPS		

Authentication Testing	SKAT-001	Testing for user enumeration - User enumeration	User enumeration	*Generic login error statement check, return codes/parameter values,Page Titles,Recovery msg, Userid guessing,*	*Webscarab*
	SKAT-002	Testing for Guessable (Dictionary) User Account - Guessable user account	Guessable user acco unt	*Default username and passwords check, App name as userid,name of app contacts,another account userid/email, js source,parameters,comment s,username /password generation,password policy check,source code - harcoded pass check, Config files check*	*Brutus, THC Hydra, Burp Intruder, Cain & Abel*
	SKAT-003	Brute Force Testing - Credentials Brute forcing	Credentials Brute fo rcing	*Dictionary, Search, Rule-Based (pswd masks) Bruteforce attacks*	*Brutus, THC Hydra, Burp Intruder, Cain & Abel, John the Ripper, OPHCRACK, Rainbow Tables*
	SKAT-004	Credentials transport over an encrypted channel - Credentials transport over an encrypted channel	Credentials transpor t over an encrypted channel	*Check referrer whether its HTTP or HTTPS, Check the method used*	*Wireshark, Proxy*
	SKAT-005	Testing password quality rules - Password combination of upper, lower case, neumerical and special character	Weak password	Penetration Testing	
	SKAT-006	Testing CAPTCHA - Captcha implementation	Weak or no CAPTCHA	*CAPTCHA Image Complexity, Set of possible answers,Analysing the return encrypted Captcha code, identify the parameters, Reuse the session id of known CAPTCHA, Send old CAPTCHA value with old ID,Send old decoded CAPTCHA value with old session id*	*CAPTCHA Decoders -PWNtcha,The Captcha Breaker, Captcha Decoder, Online Captcha Decoder.*
	SKAT-007	Testing Email address as a User ID - Validate and verify Email address	Invalid Email	Penetration Testing	
	SKAT-008	Testing remember me functionality - Vulnerable remember password	Vulnerable remember password	Penetration Testing	
	SKAT-009	Testing for autocomplete on password forms/input - Autocomplete password and user id field	Autocomplete password and user id field	Penetration Testing	
	SKAT-010	Testing for bypassing authentication schema - Bypassing authentication schema	Bypassing authentic ation schema	*Forward Browsing, Param Modification,Session ID Predication (Session Hijacking), SQL Injection*	*Webscarab*
	SKAT-011	Testing password reset or recovery - Vulnerable Password Recovery Mechanism	Weak password res et	*Secret qns asked?,strength of secret qns,no of qns,no of password reset attempts,whether new password is emailed to primary emailid check. Should not cache the passwords (remember me), Passwords stored in permanent coookies should be hashed. Autocomplete Off enabled.*	

	SKAT-012	Testing password change process - Weak password change process	Weak password change	*Secret qns asked? Strength of secret qns,no of qns,no of password reset attempts,whether new password is emailed to primary emailid check. Should not cache the passwords (remember me), Passwords stored in permanent coookies should be hashed. Autocomplete Off enabled.*	
	SKAT-013	Testing Multiple Factors Authentication - Weak Multiple Factors Authentication	Weak Multiple Fact ors Authentication		
	SKAT-014	Testing logout functionality presence - Logout function not properly implemented	Logout function not properly implemented	*Penetration Testing*	
	SKAT-015	Testing for cache management on HTTP - browser cache weakness	browser cache weak ness	*HTTP.Session.invalidate()-Java, Java.Session.abandon()-.Net implemented. Press back button/reload check,check presense of logout btns in all page, User browser closed instead of session invalidate check,insert Set-Cookie check, Time out interval, Timeout not by client check,Modify the session expiration time at clientside, Check META Cache-Controlin HTML,*	
	SKAT-016	Testing for default logins - Default login	weak login		
	SKAT-017	Testing for user-accessible authentication history	user accessible authentication history		
	SKAT-018	Testing for Weak security question/answer - Weak security question/answer	weak security question and answer		
	SKAT-019	Testing for out-of channel notification of account lockouts and successful password changes	out-of channel notification		
	SKAT-020	Testing for Race Conditions - Race Conditions vulnerability	Race Conditions vulnerability	*Make multiple simultaneous requests while observing the outcome for u nexpected behavior, Manual Code Review*	
	SKAT-021	Test for consistent authentication across applications with shared authentication schema / SSO and alternative channels - Consistent authentication	Inconsistent authentication		
Session Management	SKSM-001	Testing session cookie duration - Expire and Max-Age			
	SKSM-002	Testing session cookie scope (path and domain)			

	SKSM-003	Testing session tokens for cookie flags - httpOnly and secure or no time validity	Cookies are set not 'HTTP Only', 'Secu re', and no time vali dity	*";secure", HTTPOnly - Always set, "; domain=app.mysite.com", "; path=/myapp/", expires-Future Value => inspect for sensitive data*	*Webscarab,BurpProx y,Paros, TamperIE/Data*
	SKSM-004	Testing how session management is handled in the application (eg, tokens in cookies, token in URL)			
	SKSM-005	Testing for Session Management Schema - Bypassing Session Management Schema, Weak Session Token	Bypassing Session Management Schema, Weak Session Token		
	SKSM-006	Testing session termination after a maximum lifetime - Session auto termination	Session auto termination missing		
	SKSM-007	Testing session termination after logout - vulnerable session			
	SKSM-008	Testing session termination after relative timeout - Session relative timeout termination	Session relative timeout termination off		
	SKSM-009	Testing to see if users can have multiple simultaneous sessions	users can have multiple simultaneous sessions		
	SKSM-010	Testing session cookies for randomness - random session cookies	session cookies are not random		
	SKSM-011	Testing for Session Fixation - Session Fixation	Session Fixation		*Webscarab*
	SKSM-012	Testing for Exposed Session Variables - Exposed sensitive session variables	Exposed sensitive s ession variables	*Encryption & Reuse of Sessi on Tokens vulnerabilities, Proxies & Caching vulnerab ilities,TGET & POST vulner abilities, Transport vulnerab ilities*	
	SKSM-013	Testing for session puzzling - session can be puzzled	N.A.		
	SKSM-014	Testing new session tokens are issued on login, role change and logout - New session tokens are issued on login, role change and logout	Weak new session		
	SKSM-015	Testing for consistent session management across applications with shared session management	Session Fixation		*Webscarab*
	SKSM-016	Testing for CSRF - CSRF	CSRF	*URL Analysis and authentication requirements.*	
Authorization Testing	SKAZ-001	Testing for Insecure Direct Object References			
	SKAZ-002	Testing for missing authorization			

	SKAZ-003	Testing for Path Traversal - Path Traversal	Path Traversal	*a) Input vector enumeration b) Testing Techniques dot-dot-slash attack (../), directory traversal,directory climbing, or backtracking*	*Grep, Nikto, Burp Suite, Paros, Webscarab*
	SKAZ-004	Testing for bypassing authorization schema - Bypassing authorization schema	Bypassing authoriza tion schema	*Access a resource without authentication/after logout, Forceful Browsing*	
	SKAZ-005	Testing for Privilege Escalation - Privilege Escalation	Privilege Escalation	*Testing for role/privilege ma nipulatio - Manipulate the values of hidden variables , analyse the error messages etc*	*Proxy Tools*
	SKAZ-006	Testing for access between two users at the same privilege level - Horizontal access control			
Business logic testing	SKBL-001	Testing for feature misuse - Feature misuse	Feature misuse		
	SKBL-002	Testing for Business Logic - Bypassable business logic	Bypassable business logic	**Understanding the applicat ion *Creating raw data for desi gning logical tests (Workflo ws, ACLs) *Designing the logical tests *Standard prerequisites *Execution of logical tests*	*Automated tools fails*
	SKBL-003	Testing for lack of non-repudiation - Repudiation	Repudiation		
	SKBL-004	Testing for trust relationships - Trust relationship	Untrusted relationship		
	SKBL-005	Testing for integrity of data - Data integrity	loss of data integrity		
	SKBL-006	Testing segregation of duties - Segregation of duties	Segregation of duties		
	SKBL-007	Testing defenses against application misuse - Application misuse	Application misuse		
	SKBL-008	Testing Upload of Unexpected File Types - Upload of Unexpected File Types	Unexpected File Types		
Data Validation Testing	SKDV-001	SQL Injection - SQL Injection	SQL Injection	***Tests*** *1.Heuristic Analysis(' , : , --) 2.Construct SQL Injection Vectors 3.Analyse Error Messages*	*OWASP SQLiX SQL Power Injector sqlbftools sqlmap SqlDumper sqlninja*

	SKDV-002	LDAP Injection - LDAP Injection	LDAP Injection	*Ability to • Access unauthorized content • Evade Application restrictions • Gather unauthorized information • Add or modify Objects inside LDAP tree structure.*	*Softerra LDAP Browser*
	SKDV-003	ORM Injection - ORM Injection	ORM Injection	*Black box testing for ORM Injection vulnerabilities is identical to SQL Injection testing*	
	SKDV-004	XML Injection - XML Injection	XML Injection	*Check with XML Meta Characters ', " , <>, <!--/-->, &, <![CDATA[/]]>,*	
	SKDV-005	SSI Injection - SSI Injection	SSI Injection	** Presense of .shtml extension * Check for these characters < ! # = / . " - > and [a-zA-Z0-9] * include String = <!--#include virtual="/etc/passwd" -->*	*Burp Suit, WebScarab, Paros*
	SKDV-006	XPath Injection - XPath Injection	XPath Injection	** Check for XML error enumeration by supplying a single quote (') * Username: ' or '1' = '1 Password: ' or '1' = '1*	
	SKDV-007	Testing for Reflected Cross Site Scripting - Reflected XSS	Reflected XSS	*1. Detect input vectors. 2. Analyze each input vector to detect potential vulnerabi lities 3. Replace the vector used to identify XSS with the vector which can exploit the vulnerability.*	*CAL9000, Rsnake XSSdb, XSSMe firefox addon, XSS proxy, WebScarab, Rat proxy, Burp Proxy*
	SKDV-008	Code Injection - Code Injection	Code Injection	*Enter commands in the input field*	
	SKDV-009	Testing for Stored Cross Site Scripting - Stored XSS	Stored XSS	*1.Input Forms 2.Analyze HTML code 3.Leverage Stored XSS with BeEF 4.File Upload*	*CAL9000, Hackvertor, XSSProxy, BeEF, WebScarab*
	SKDV-010	Testing for DOM based Cross Site Scripting - DOM XSS	DOM XSS	*Test for the user inputs obtained from client-side JavaScript objects*	*Automated tools fails*
	SKDV-011	Testing for Cross Site Flashing - Cross Site Flashing	Cross Site Flashing	*1.Decompile 2.Undefined Variables 3.Unsafe methods 4.Include malicious SWF*	*SWFIntruder, Flare, Flasm*

	SKDV-012	IMAP/SMTP Injection - IMAP/SMTP Injection	IMAP/SMTP Injection	• *Exploitation of vulnerabilities in the IMAP/SMTP protocol* • *Application restrictions evasion* • *Anti-automation process evasion* • *Information leaks* • *Relay/SPAM* *The standard attack patterns are:* • *Identifying vulnerable parameters* • *Understanding the data flow and deployment structure of the client* • *IMAP/SMTP command injection*	
	SKDV-013	Buffer overflow - Buffer overflow	Buffer overflow		*OllyDbg, Spike, Brute Force Binary Tester (BFB), Metasploit. RATS, Flawfinder and ITS4 are available for analyzing C-style languages*
	SKDV-014	Testing for Command Injection - Command Injection	Command Injection	*Understand the application platform, OS, folder structure, relative path and execute those*	*Webscarab*
	SKDV-015	Testing for Format String - String Formatting	String Formatting	*Penetration Testing*	*ITS4*
	SKDV-016	Testing for Open Redirection - Open redirection	Open redirection	*Penetration Testing*	*DOMinator*
	SKDV-017	Testing for NoSQL injection - NoSQL injection	NoSQL injection		
	SKDV-018	Testing for NULL/Invalid Session Cookie - Vulnerable session	Vulnerable session	*Penetration Testing and code review*	*ITS4*
	SKDV-019	Testing for XQuery Injection - Xquery injection	Xquery injection		
	SKDV-020	Incubated vulnerability - Incubated vulnerability	Incubated vulnerability	*File Upload, Stored XSS , SQL/XPATH Injection, Manage server files via server misconfigs*	*XSS-proxy, Paros, Burp, Metasploit*
	SKDV-021	Test for Expression Language Injection - Expression language injection	Expression language injection		
	SKDV-022	Testing for HTML Injection - HTML injection	HTML injection		
	SKDV-023	Test for Local File Inclusion - Local File Inclusion	Local File Inclusion		
	SKDV-024	Test for Remote File Inclusion - Remote File Inclusion	Remote File Inclusion		

	SKDV-025	Compare client-side and server-side validation rules - Compare client-side and server-side validation	Different client-side and server-side validation	*Penetration Testing Code review*	*OllyDbg*
	SKDV-026	Testing for HTTP parameter pollution - HTTP parameter pollution	HTTP parameter pollution		
	SKDV-027	Testing for Mass Assignment - Mass assignment	Mass assignment		
	SKDV-028	Testing for HTTP Splitting/Smuggling - HTTP Splitting, Smuggling	HTTP Splitting, Smuggling	param=foobar%0d%0aContent-Length:%200%0d%0a%0d%0aHTTP/1.1%20200%20OK%0d%0aContent-Type:%20text/html%0d%0aContent-Length:%2035%0d%0a%0d%0a<html>Sorry,%20System%20Down</html>	
Denial of Service Testing	SKDS-001	Testing for anti-automation - Anti-automation	Anti-automation		
	SKDS-002	Testing for SQL Wildcard Attacks - SQL Wildcard vulnerability	SQL Wildcard vulnerability	'%_[^!_%/%a?F%_D)_(F%)_%([)({}%){()}£$&N%_)$*£()$*R"_)][%](%[x])%a][$*"£$-9]_%' '%64_[^!_%65/%aa?F%64_D)_(F%64)_%36([)({}%33){()}£$&N%55_)$*£()$*R"_)][%55](%66[x])%ba][$*"£$-9]_%54' bypasses modsecurity [r/a)_ _(r/b)_ _(r-d)_%n[^n]y[^j]l[^k]d[^l]h[^z]t[^k]b[^q]t[^q][^n]!%%_[aaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaa[! -z]@$!_%	
	SKDS-003	Locking Customer Accounts - Locking Customer Accounts	Locking Customer Accounts - Locking Customer Accounts		*Wrong Attempts Valid Username enumeration - Login Page, New User Registration Page, Password Reset Page*
	SKDS-004	Testing for DoS Buffer Overflows - Buffer Overflows	Testing for DoS Buffer Overflows - Buffer Overflows	*If you have received a response (or a lack of) that makes you believe that the overflow has occurred, attempt to make another request to the server and see if it still responds.*	*Submit large inputs and check how the server responds*
	SKDS-005	Test for account lockout - Account auto-lockout		*1. Evaluate the account lockout mechanism's ability to mitigate brute force password guessing. 2. Evaluate the unlock mechanism's resistance to unauthorized account unlocking.*	*WebScarab*
	SKDS-006	User Input as a Loop Counter - User Input as a Loop Counter	User Input as a Loop Counter		

	SkDS-007	Writing User Provided Data to Disk - Writing User Provided Data to Disk	Writing User Provid ed Data to Disk	*1. The tester submits an extremely long value to the server in the request, and the application logs the value directly without having validated that it conforms to what was expected.* *2. The application may have data validation to verify the submitted value being well formed and of proper length, but then still log the failed value (for auditing or error tracking purposes) into an application log.*	
	SKDS-008	Storing too Much Data in Session - Storing too Much Data in Session	Storing too Much D ata in Session	*The developer may have chosen to cache the records in the session instead of returning to the database for the next block of data. If this is suspected, create a script to automate the creation of many new sessions with the server and run the request that is suspected of caching the data within the session for each one. Let the script run for a while, and then observe the responsiveness of the application for new sessions. It may be possible that a Virtual Machine (VM) or even the server itself will begin to run out of memory because of this attack.*	
	SKDS-009	Test for HTTP protocol DoS - HTTP protocol			
	SKDS-010	Failure to Release Resources - Failure to Release Resources	Failure to Release R esources	*• An application locks a file for writing, and then an exception occurs but does not explicitly close and unlock the file* *• Memory leaking in languages where the developer is responsible for memory management such as C & C++. In the case where an error causes normal logic flow to be circumvented, the allocated memory may not be removed and may be left in such a state that the garbage collector does not know it should be reclaimed* *• Use of DB connection objects where the objects are not being freed if an exception is thrown. A number of such repeated requests can cause the application to consume all the DB connections, as the code will still hold the open DB object, never releasing the resource.*	

Cryptography Testing	CR-001	Check if data which should be encrypted is not - Data protection		*Black Box testing*	
	SKCR-002	Testing for wrong algorithms usage depending on context - Wrong algorithm			
	SKCR-003	Testing for weak algorithms usage - Weak algorithm			
	SKCR-004	Testing for proper use of salting - Salt usage	Broken Confidentiality	*Code Review*	
	SKCR-005	Testing for randomness functions - Randomness Function			
Risky Functionality - File Uploads Testing	SKRF-001	Testing acceptable file types are whitelisted - Acceptable files			
	SKRF-002	Testing file size limits, upload frequency and total file counts are defined and are enforced - File size, upload frequency	Broken file upload	*Code review, Black box testing, Upload files of different sizes with different frequency and test for limit.*	
	SKRF-003	Testing all file uploads have Anti-Virus scanning in-place - Anti-Virus scanning of file			
	SKCR-004	Testing uploaded files are not directly accessible within the web root - File directory access		*Black box testing, upload files in data directory and try to directly access this directory.*	
Error Handling Testing	EH-001	Checking for Error Codes and Stack Trace			

7.6 Security Testing

7.6.1 Information Gathering

7.6.1.1 Conduct search engine discovery

Summary

For each search engine discovery, there exist many direct and indirect methods. Direct methods relate to searching the associated content from caches whereas indirect methods relate to the sensitive design, configuration information by searching forums, newsgroups, and tendering websites.

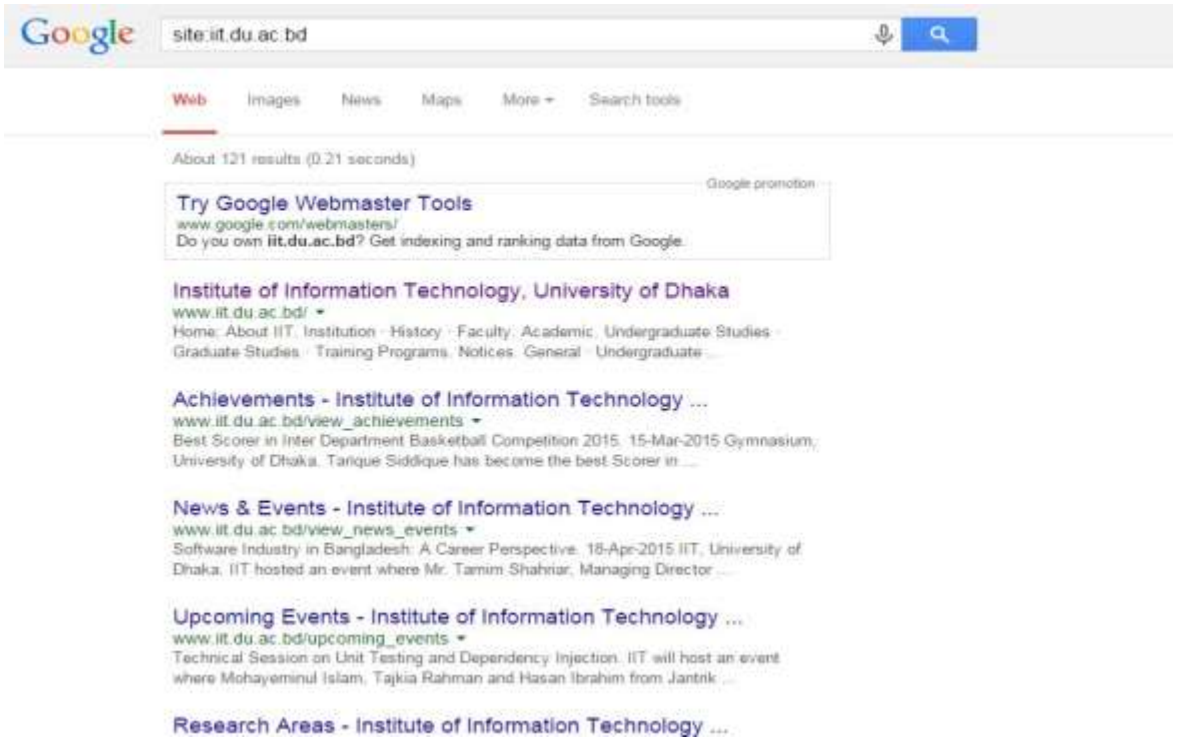
Test Objectives

To understand what sensitive design and configuration information of the application is exposed both directly (on the organization's website) or indirectly (on a third party website). Besides this, to find out the page and this will help attacker to execute malicious task.

How to Test

Use a search engine to search for:

- Archived posts and emails by administrators and other key staff
- Log on procedures and username formats
- Usernames and passwords
- Error message content



About 121 results (0.21 seconds)

Google promotion
Try Google Webmaster Tools
www.google.com/webmasters/
Do you own **iit.du.ac.bd**? Get indexing and ranking data from Google.

Institute of Information Technology, University of Dhaka
www.iit.du.ac.bd/
Home · About IIT · Institution · History · Faculty · Academic · Undergraduate Studies · Graduate Studies · Training Programs · Notices · General · Undergraduate ...

Achievements - Institute of Information Technology ...
www.iit.du.ac.bd/view_achievements
Best Scorer in Inter Department Basketball Competition 2015. 15-Mar-2015 Gymnasium, University of Dhaka. Tarique Siddique has become the best Scorer in ...

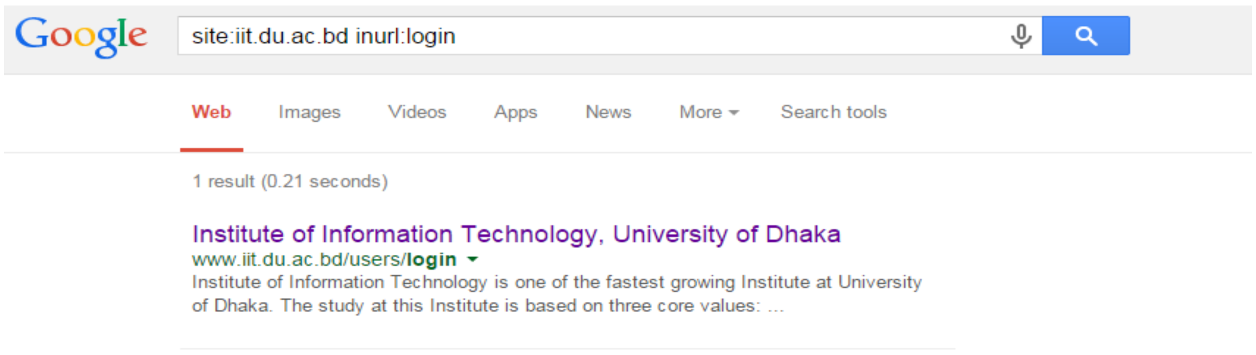
News & Events - Institute of Information Technology ...
www.iit.du.ac.bd/view_news_events
Software Industry in Bangladesh: A Career Perspective. 18-Apr-2015 IIT, University of Dhaka. IIT hosted an event where Mr. Tamim Shahriar, Managing Director ...

Upcoming Events - Institute of Information Technology ...
www.iit.du.ac.bd/upcoming_events
Technical Session on Unit Testing and Dependency Injection. IIT will host an event where Mohayeminul Islam, Tajkia Rahman and Hasan Ibrahim from Jantrik ...

Research Areas - Institute of Information Technology ...

Fig. 7.14: All pages which are publicly available for a web site

From the Fig. 7.14, it is clear that for IIT website the pages named home page, achievements, news & events, etc. all are available. By this the attacker can enter into any page as he wishes and can perform any malicious activity.



site:iit.du.ac.bd inurl:login

1 result (0.21 seconds)

Institute of Information Technology, University of Dhaka
www.iit.du.ac.bd/users/**login**
Institute of Information Technology is one of the fastest growing Institute at University of Dhaka. The study at this Institute is based on three core values: ...

Fig. 7.15: Only login page of IIT

Tools

- ➢ Google Hacker[5]
- ➢ FoundStone SiteDigger[6]

7.6.1.2 Fingerprint Web Server

Summary

Web server fingerprinting is critical for the penetration tester. By knowing the version and type of a running web server allows testers to determine known vulnerabilities and the appropriate exploits to use during testing.

Test Objectives

To find out the version and type of a running web server and to determine known vulnerabilities and the appropriate exploits to use during testing.

How to Test

- Black Box testing
- HTTP header field ordering

Tools

- ➢ Httprint[7]
- ➢ httprecon [8]

7.6.1.3 Enumerate Applications on Webserver

Summary

The test for web application vulnerabilities is to find out which particular applications are hosted on a web server. Different applications have different known vulnerabilities and known attack strategies and that can be easily exploited in order to exploit data.

Test Objectives

Enumerate the applications within scope that exist on a web server.

How to Test

- Black Box Testing
- Gray Box Testing

Tools

- ➢ Search engines [9]

7.6.1.4 Review webpage comments and metadata for information leakage

Summary

There exist a very common technique for programmers to include detailed comments and metadata on their source code. However, comments and metadata included into the HTML code might reveal internal information that should not be available to potential attackers. Using this valuable information attacker can easily guess the website status and he can easily exploit the vulnerabilities.

Test Objectives

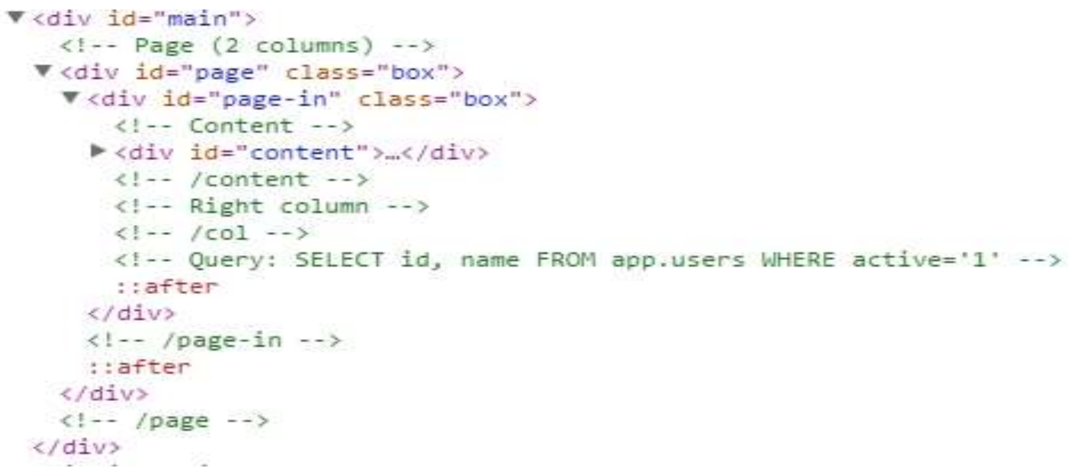
Review webpage comments and metadata to better understand the application and to find any information leakage.

How to Test

- Black Box Testing

Tools

- ➢ Browser "view source" function



```
▼<div id="main">
   <!-- Page (2 columns) -->
 ▼<div id="page" class="box">
   ▼<div id="page-in" class="box">
       <!-- Content -->
     ▶<div id="content">…</div>
       <!-- /content -->
       <!-- Right column -->
       <!-- /col -->
       <!-- Query: SELECT id, name FROM app.users WHERE active='1' -->
       ::after
     </div>
     <!-- /page-in -->
     ::after
   </div>
   <!-- /page -->
 </div>
```

Fig. 7.16: Source code of a module of “Sufia Kamal Hall” website

7.6.1.5 Identify application entry points

Summary

Enumerating the application and its attack surface is a key requirement before any thorough testing can be undertaken, as it allows the tester to identify likely areas of weakness.

Test Objectives

Understand how requests are formed and typical responses from the application.

How to Test

- Black Box Testing
- Gray Box Testing

Tools

- ➢ OWASP: WebScarab

7.6.1.6 Identifying technologies used-Technology

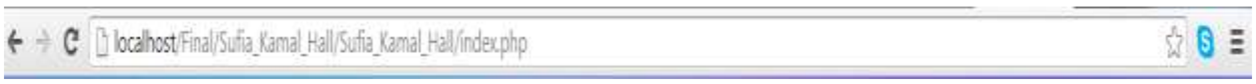
If it is possible to find out which technology is used then for that technology specific vulnerabilities can be easily exploited.

Test Objectives

To attack the system as soon as possible by exploiting the known vulnerabilities.

How to Test

By observing the url link.



localhost/Final/Sufia_Kamal_Hall/Sufia_Kamal_Hall/index.php

Fig. 7.18: Technology Identification

7.6.1.7 Map execution paths through application

Before performing security testing, understanding the structure of the application is important. Without a thorough understanding of the layout of the application, it is unlucky that it will be tested thoroughly.

Test Objectives

To understand the principal workflows of the target system.

How to Test

- Black Box Testing
- Gray/White Box testing

Tools

- Spreadsheet software
- Diagramming software

7.6.2 Configuration Management Testing

7.6.2.1 Test Network/Infrastructure Configuration

The complexity of interconnected and heterogeneous web server infrastructure includes hundreds of web applications and makes them configuration more manageable and review a fundamental step in testing and deploying every single application. It takes only a single vulnerability to undermine the security of the entire infrastructure.

Test Objectives

This test case can map the infrastructure supporting the application and understand how it affects the security of the application.

How to Test

- Known Server Vulnerabilities
- Administrative tools

7.6.2.2 Test Application Platform Configuration

Proper configuration of the single elements that make up application architecture is important for preventing mistakes that might compromise the security of the architecture.

How to Test

- ➢ Black Box Testing
- ➢ Gray Box Testing

7.6.2.3 Test File Extensions Handling for Sensitive Information

File extensions are commonly used in web servers to easily determine which technologies, languages and plugins needs to be used to fulfill the web request. This behavior is consistent with RFCs and Web Standards.

How to Test

- Gray Box testing

7.6.2.4 Review Old, Backup and Unreferenced Files for Sensitive Information

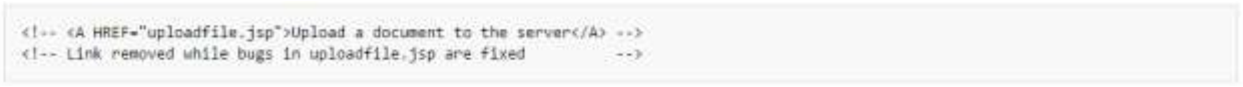
When most of the files are directly handled by the server itself, it is not uncommon to find unreferenced or forgotten files that can be used to obtain important information about the infrastructure or the credentials.

How to Test

- Black Box Testing
- Inference from the naming scheme used for published content
- Blind guessing
- Other clues in published content

Tools

- Vulnerability assessment tools



```
<!-- <A HREF="uploadfile.jsp">Upload a document to the server</A> -->
<!-- Link removed while bugs in uploadfile.jsp are fixed          -->
```

Fig. 7.19: Review old, backup and unreferenced files for sensitive information

7.6.2.5 Test HTTP Methods

HTTP offers a number of methods that can be used to perform actions on the web server. Many of these methods are designed to aid developers in deploying and testing HTTP applications. The Hypertext Transfer Protocol (HTTP) allows several other methods such as:

- HEAD
- GET
- POST
- PUT
- DELETE

- TRACE
- OPTIONS
- CONNECT

How to Test

- Discover the Supported Methods

- Testing for arbitrary HTTP methods

Tools

- NetCat [10]
- cURL[11]

7.6.2.6 Test HTTP Strict Transport Security

The HTTP Strict Transport Security (HSTS) header is a mechanism by which the web sites have to communicate to the web browsers. All traffic must be exchanged with a given domain must always be sent over https, this will help to protect the information from being passed over the unencrypted requests.

How to Test

Testing can be done by checking the existence of the HSTS header in the server's response.

7.6.3 Identity Management Testing

7.6.3.1 Test Role Definitions

It is common in modern enterprises to define system roles to manage users and authorization to system resources. Most of the cases, it is expected that at least two roles exist, administrators and regular users.

Test objectives

Validate the system roles defined within the application sufficiently define and separate each system and business role to manage appropriate access to system functionality and information.

How to test

Either with or without the help of the system developers or administrators, develop an role versus permission matrix where access control list and capability list will be stayed.

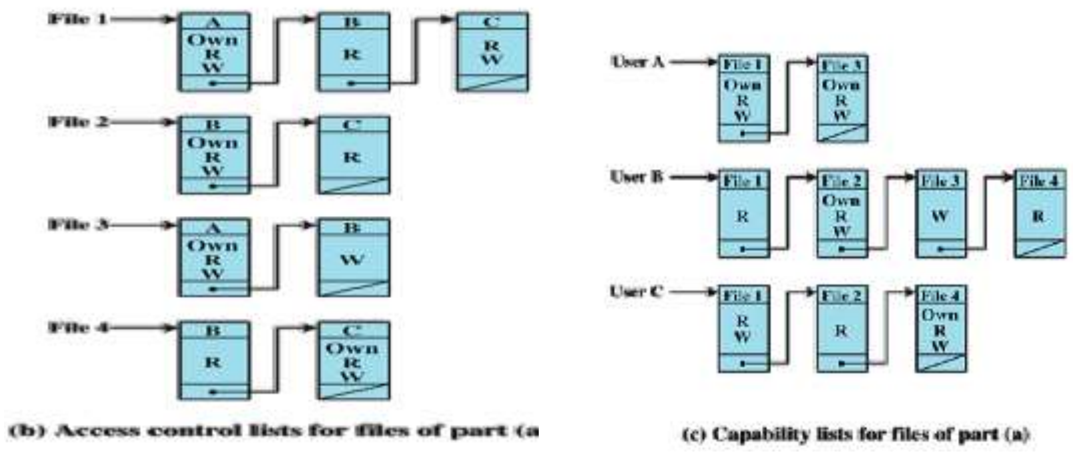


Fig. 7.20: Access control

Tools

While the most accurate approach to completing this test is to conduct it manually.

7.6.3.2 Test User Registration Process

Summary

Some websites offer a user registration process that automates the provisioning of system access to users.

Test objectives

1. Verify that the identity requirements for user registration are aligned with business and security requirements.

2. Validate the registration process.

How to test

Verify that the identity requirements for user registration are aligned with business and security requirements:

1. Can anyone register for access?
2. Are registrations vetted by a human prior to provisioning, or are they automatically granted if the criteria are met?
3. Can the same person or identity register multiple times?
4. Can users register for different roles or permissions?
5. What proof of identity is required for a registration to be successful?
6. Are registered identities verified?

Tools

A HTTP proxy can be a useful tool to test this control.

7.6.3.3 Test Account Provisioning Process

Summary

The provisioning of accounts presents an opportunity for an attacker to create a valid account without application of the proper identification and authorization process.

Test objectives

Verify which accounts may provision other accounts and of what type.

How to test

Determine which roles are able to provision users.

- Is there any verification, vetting and authorization of provisioning requests?

- Is there any verification, vetting and authorization of de-provisioning requests?
- Can an administrator or other user provision accounts with privileges greater than their own?

7.6.3.4 Testing for Account Enumeration and Guessable User Account

This test verifies if it is possible to collect a set of valid usernames by interacting with the authentication mechanism of the application. These types of test will help during brute force testing, in which the tester verifies if, given a valid username, it is possible to find the corresponding password.

How to Test

- Black box testing
- Gray Box testing

Tools

- WebScarab[12]
- CURL[13]

7.6.3.5 Testing for weak or unenforced username policy

User account names are often highly structured and valid strong account names can not easily be guessed. On the other hand, easy or commonly used user names can easily guessable.

Test objectives

Check whether an account name structure renders the application vulnerable to account enumeration. Determine whether the consistent application's error messages permit account enumeration.

How to test

- Determine the structure of account names.
- Evaluate the application's response to valid and invalid account names.

- Use different responses to valid and invalid account names to enumerate valid account names.
- Use account name dictionaries to enumerate valid account names.

7.6.4 Authorization Testing

7.6.4.1 Testing Directory traversal/file include

Many web applications use files as part of their daily operation. Input validation methods that have not been well designed or deployed, an attacker can use that validation methods to exploit the system in order to read or write files that are not intended to be accessible. In particular situations, it could be possible to execute arbitrary code or system commands for exploiting the vulnerabilities.

How to Test

- Black Box testing
- Gray Box testing

7.6.5 Authentication Testing

7.6.5.1 Testing for Credentials Transported over an Encrypted Channel

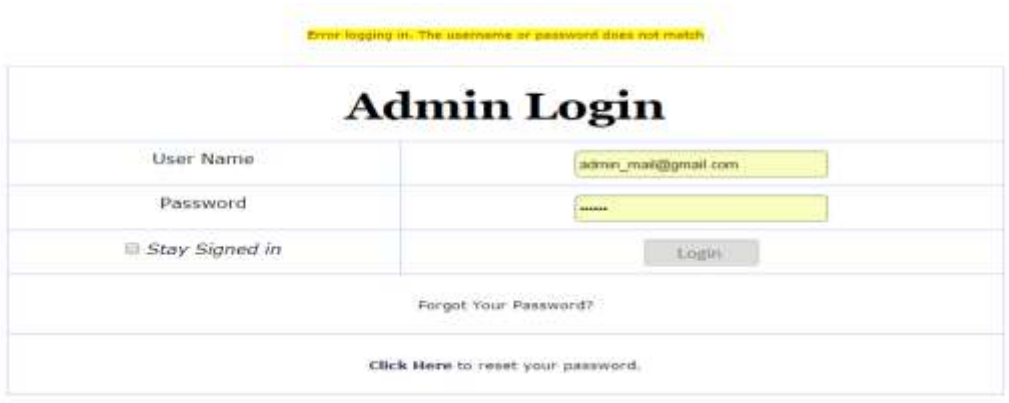
Testing for credentials transport means verifying that the user's authentication data are transferred via an encrypted channel to avoid being intercepted by malicious users.

How to Test

- Black Box testing
- Gray Box testing

Tools

- WebScarab


Error logging in. The username or password does not match

Admin Login

User Name	admin_mail@gmail.com
Password	•••••••
Stay Signed in	Login

Forgot Your Password?

Click Here to reset your password.

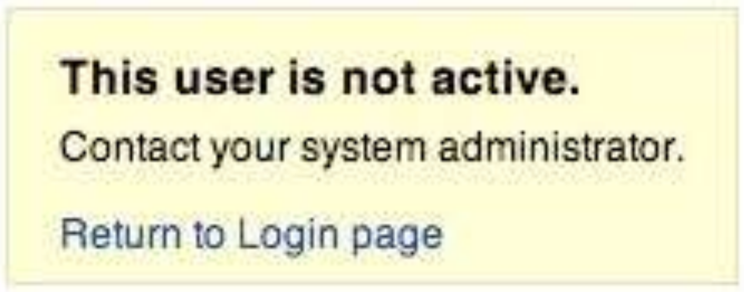
Fig. 7.21: Authentication testing

7.6.5.2 Testing for default credentials

Nowadays web applications can use popular open source or commercial software that can be installed on servers with minimal configuration or customization by the server administrator.

How to Test

- Testing for default credentials of common applications
- Testing for default password of new accounts
- Gray Box testing


This user is not active.

Contact your system administrator.

Return to Login page

Fig. 7.22: Testing for default credentials

7.6.5.3 Testing for Weak password policy

The most prevalent and most easily administered authentication mechanism is a static password. The password represents the keys to the kingdom, but is often subverted by users in the name of usability

Test objectives

Check the resistance of the application against brute force password guessing using available password dictionaries by evaluating the length, complexity, reuse and aging requirements of passwords.

How to Test

1. What characters are permitted and forbidden for use within a password?
2. How often can a user change their password? How quickly can a user change their password after a previous change?

Remediation

To mitigate the risk of easily guessed passwords facilitating unauthorized access there are two solutions: introduce additional authentication controls.

7.6.5.4 Testing for Weak lock out mechanism

Account lockout mechanisms are used to mitigate brute force password guessing attacks. Accounts are typically locked after 3 to 4 unsuccessful login attempts and can only be unlocked after a predetermined period of time, via a self-service unlock mechanism, or intervention by an administrator

Test objectives

1. Evaluate the account lockout mechanism's ability to mitigate brute force password guessing.
2. Evaluate the unlock mechanism's resistance to unauthorized account unlocking.

How to Test

To test the strength of lockout mechanisms, you will need access an account that you are willing to lock. If there exist only one account with which one can log on to the web application, he can perform this test at the end of his test plan.

7.6.5.5 Testing for Bypassing Authentication Schema

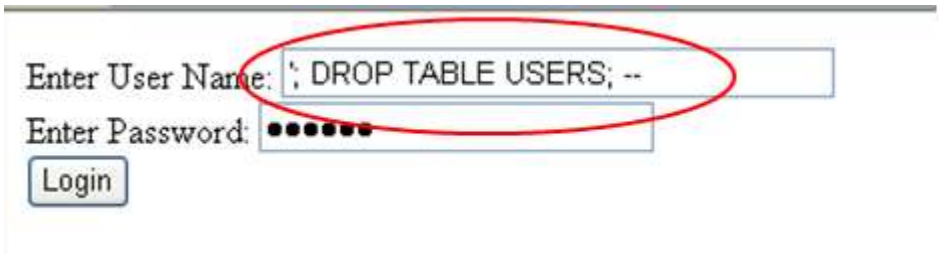
While most applications require authentication to gain access to private information not every authentication method is able to provide adequate security

How to Test

- Black Box testing

There are several methods of bypassing the authentication schema that is used by a web application:

- Session ID prediction
- SQL injection



Enter User Name: '; DROP TABLE USERS; --

Enter Password: ●●●●●●

Fig. 7.23: Testing for Bypassing Authentication Schema

7.6.5.6 Testing for Vulnerable Remember Password

Sometimes browsers will ask a user if the users wish to remember the password that user just entered. The browser will then store the password if the user agrees to store the password, and later the user can use their saved password when they wants to enter into the website without not giving the password.

How to Test

- Look for passwords being stored in a cookie.
- Examine the cookies stored by the application.
- Verify that the credentials are not stored in clear text, but are hashed.

Remediation

Ensure that no credentials are stored in clear text or are easily retrievable in encoded or encrypted forms in cookies.

7.6.5.7 Testing for Browser cache weakness

In this phase, the tester checks that the application correctly instructs the browser not to remember sensitive data.

How to Test

- ➢ Browser History
- ➢ Browser Cache

Tools

- Firefox add-on CacheViewer2

7.6.5.8 Testing for Weak security question/answer

This mechanism often named "secret" questions and answers. Security questions and answers are used to recover forgotten passwords.

How to Test

- ➢ Testing for weak pre-generated questions
- ➢ Testing for weak self-generated question
- ➢ Testing for brute-forcible answers

7.6.5.9 Testing for weak password change or reset functionalities

The password change and reset function of an application is a self-service password change or reset mechanism for users. This self-service mechanism allows users to quickly change or reset their password without an administrator intervening.

How to Test

For both password change and password reset it is important to check:

1. If users, other than administrators, can change or reset passwords for accounts other than their own.
2. If users can manipulate or subvert the password change or reset process to change or reset the password of another user or administrator.
3. If the password change or reset process is vulnerable to CSRF.

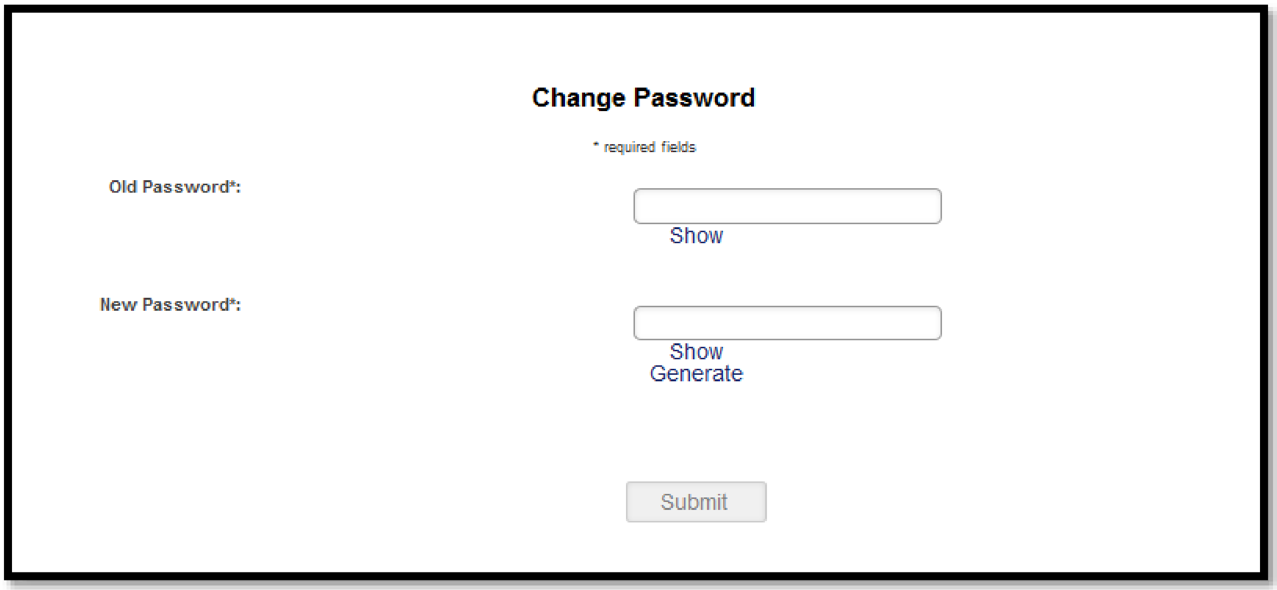


Fig. 7.24: Testing for weak password change or reset functionalities

7.6.5.10 Testing for Stack Traces

Stack traces are not vulnerabilities by themselves, but they often express information in where attackers are interested. Attackers attempt to generate these stack traces to find out what actually happen.

How to Test

- Black Box testing
- Gray Box Testing

Tools

- ZAP Proxy

7.6.6 Input Validation Testing

7.6.6.1 Testing for Cross Site Scripting

Cross-site Scripting (XSS) occur when an attacker injects browser executable code within a single HTTP response. This impacts when users open a maliciously crafted link or third-party web page. The attack string is included as part of the crafted URI or HTTP parameters, improperly processed by the application, and returned to the victim.

One of the primary difficulties in preventing XSS vulnerabilities is proper character encoding. In some cases, the web server or the web application could not be filtering some encodings of characters, so, for example, the web application might filter out "<script>", but might not filter %3cscript%3e which simply includes another encoding of tags.

How to test

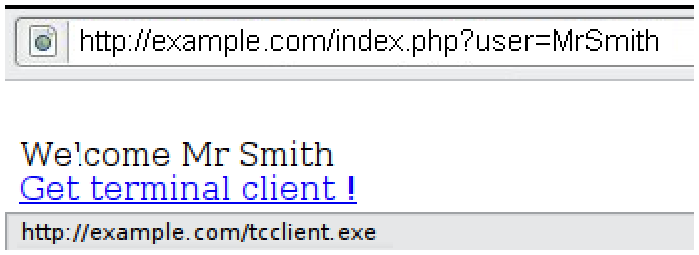
A black-box test will include at least three phases:

1. Detect input vectors. For each web page, the tester must determine all the web application's user-defined variables and how to input them.
2. Detect input vectors. For each web page, the tester must determine all the web application's user-defined variables and how to input them.

- For example: <script>alert(123)</script>

1. For each test input attempted in the previous phase, the tester will analyze the result and determine if it represents a vulnerability that has a realistic impact on the web application's security.

For example, consider a site that has a welcome notice "Welcome %username% " and a download link.



http://example.com/index.php?user=MrSmith

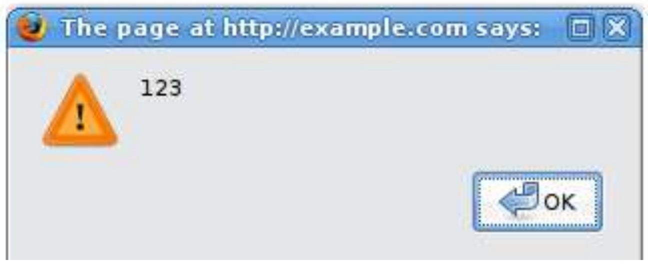
We!come Mr Smith
Get terminal client !

http://example.com/tcclient.exe

Fig. 7.24: Testing for Cross Site Scripting

Let's try to click on the following link and see what happens:

```
http://example.com/index.php?user=<script>alert(123)</script>
```



The page at http://example.com says:

123

Fig. 7.25: Testing for Cross Site Scripting (result)

7.6.6.2 Testing for SQL Injection

SQL injection attack consists of insertion or "injection" of a SQL query via the data input from the client (browser) to the web application. A successful SQL injection attack can read sensitive data from the database, modify database data and execute administration operations on the database.

For example: Select email from login where name=”N” or 1=1

How to Test

When authentication is performed using a web form, chances are that the user credentials are checked against a database that contains all usernames and passwords.

The string submitted by the user could be used in a SQL query that extracts all relevant records from a database.

7.6.6.3 Testing for Buffer overflow

How to Test

- Testing for heap overflow vulnerability
    - Heap is a memory segment that is used for storing dynamically allocated data and global variables. Each chunk of memory in heap consists of boundary tags that contain memory management information.
    - When a heap-based buffer is overflowed the control information in these tags is overwritten. When the heap management routine frees the buffer, a memory address overwrite takes place leading to an access violation.

    How to test heap overflow

    - Black Box testing
    - Gray Box testing

- Testing for stack overflow vulnerability
    - Stack overflows occur when variable size data is copied into fixed length buffers located on the program stack without any bounds checking. Vulnerabilities of this class are generally considered to be of high severity since their exploitation would mostly permit arbitrary code execution or Denial of Service.

    How to test stack overflow

    - Black Box testing
    - Gray Box testing

Chapter 8: Implementation and Testing

This chapter is about implementation and testing. This chapter contains the implementation process which includes implementation tools, technologies and methodologies. This chapter also includes test plan including black box and white box testing and test reports conducted on developed system.

8.1 Tools and Technology

We have used many technologies for the developing process of our website TwirlingHall.com. These technologies are named below.

- PHP
- Xampp
- Javascript
- jQuery
- Cascading Stylesheet (CSS)
- Bootstrap

8.2 Testing

Software testing is an investigation conducted to provide stakeholders with information about the quality of the product or service under test. We have tried our level best to test the website through the help of our knowledge in this field. The following is the testing section done by us. First we have prepared a test plan for testing the website. Then we have prepared the test result of the website.

8.2.1 Test Plan

The main purpose of test plan is to describe the approach, resources and schedule of the testing part of the software life cycle. Its intended audience is the project manager, development team and testing team. This document can also be shared with the end user to get the input and approval for the test activities.

8.2.1.1 Objectives

Objective of this test plan is to provide a framework for manager and test team to test the website TwirlingHall.com in an effective way. This document will provide-

- Subsystems of the website that should be tested.
- Describes overall test strategy that is followed by developers, test stuff, project manager.
- Environmental needs for maintaining test strategy of the website.
- Schedule that is needed to deliver product timely.
- Critical or risk issues of website that needed extra effort from both tester and developers.
- Deliverable list of the test activities.
- Identify people who are responsible for testing work.

8.2.1.2 Scope

This test plan describes unit test, performance test, security test, integration test, user interface test, acceptance test, regression test. Both local PCs and remote PCs will be tested. Subsystems that are related to registered user have high priority. Conversion testing is not considered as part of this project since this is the first version.

Test Plan Identifier

TwirlingHall.com-version_1

Reference Document

Software Requirements and Specification of TwirlingHall.com

8.2.1.3 Test Items

The listing below identifies test items of the website. In this website there are mainly two modules- external module and internal module. These modules have sub modules. Test items are categorized according to these modules. Here, ‘EX’ and ‘I’ in test number represent external and internal module respectively.

Test Number	Test Items
EX-T01	Verifying contents of home page
EX-T02	Adding and removing items from ‘About’ page
EX-T03	Verifying search operation
EX-T04	Updating ‘Contact information’
EX-T05	Verifying functions of ‘Google map’
I-T01	Adding new student in Authentication sub module
I-T02	Information entry in Login page
I-T03	Profile updating and password changing
I-T04	Testimonial, ID card, Boarding card generation
I-T05	Uploading excel
I-T06	Searching student
I-T07	Verifying permission
I-T08	Adding faculty, department, session, room number, institution
I-T09	Verifying event

Table 8.1 Test Items

8.2.1.4 Software Risk Issues

Below is a listing of some critical areas of Twirling hall.com website.

- Internal module has sub modules (admin panel, student sub module etc.) that take inputs from input fields. In order to avoid ambiguous input each field needs input validation.
- Twirling hall.com has both authorized and unauthorized user. Security must be ensured to limit unauthorized access.
- Full website behavior should be checked when a new module will be added.
- If any module crushes then other related modules must behave normally.
- In heavy load, website should behave same for all users.

- Twirling hall.com is a large website as the span time is short testing the whole website is little bit risky.
- Quality assurance staff may not available.

8.2.1.5 Features to be Tested

Following list represents those features that have been identified as target for testing. This list represents what features will be tested with the level of security as high (H), medium (M) and low (L).

i. Graphics, texts of home page (H)
ii. Items of about page (L)
iii. External search option (L)
iv. Contact information package (H)
v. Google map (H)
vi. Authentication sub module (H)
vii. Profile of internal users (H)
viii. Testimonial, ID card and Boarding card generation (H)
ix. Excel upload (H)
x. Providing Permission (H)
xi. Adding item (H)
xii. Event creation (M)

8.2.1.7 Approach

The total approach for the test process of TwirlingHall.com is narrated as follows. The testers have to follow the strategy to achieve a better result.

8.2.1.7.1 Testing Level

- **Unit Test**

Unit test will be performed by developers in each module when they are developing. This test will be executed using source code. When all test cases give positive results only then unit test will be accepted.

- **Performance Test**

Performance testing will be done by test stuff. Response time of all pages, availability, portability, and scalability of the website will be tested. To test performance of this website tester will do load test and stress test. Result of this performance test will be approved by project manager.

- **User Interface Test**

Test stuff will do user interface testing. Goal of this testing is to provide users an interface that will contain enough information and help users to interact easily. Tester will verify interaction between pages, tab keys and mouse movements and menus, size, position of web content.

- **Integration Test**

Integration testing will be performed after unit test. This testing is extension of unit test. In this website this testing will be done by test stuff and project manager. Developers will be informed about integration test result. Here, test stuff will follow bottom-up test strategy because this strategy facilitates efficient error detection.

- **Security Test**

The test stuff is responsible for performing the security test. The main purpose of this test is to ensure the security of the information provided by the user. The tester will check whether the database is accessible to the outsider and they can give input to the database. The test result will be judged by the project manager.

- **Acceptance test**

The acceptance test is done by the test stuff. The output of the website to the requirements of the user will be matched. If the level of incorrectness falls between the boundaries of fault tolerance, the test case will be accepted. The user will judge the test result to match the boundaries of the acceptance.

- **Regression Test**

This test is done to check whether the change in one module of the website causes a change in other module. The test stuff checks this by doing small changes in some module or by fixing problem of one module. The result is checked by the project manager.

8.2.1.7.2 Test Tools

The tools to perform the test are listed as follows.

i. For performing component test and integration test selenium will be used.
ii. User interface testing will be done through IE tester.
iii. Microsoft visual studio 2012 can be used to perform the load test and performance test.
iv. Regression test can be done using selenium.
v. Bugzilla can be used to track the bugs in security testing.
vi. For management of the testing process, Microsoft word and Microsoft excel can be used.

8.2.1.7.3 Meeting

The test team stuff has to meet the project leader once in every week to discuss and evaluate the test progress of the website. The test team stuff will meet developers once in every two weeks. Additional meeting can also be arranged when needed.

8.2.1.8 Item Pass/Fail Criteria

This section describes pass/fail criteria for each of the features to be tested. Given pass/fail criteria are based on main three test types described in this test plan.

- **Pass/ Fail criteria of unit test**

Unit test is done by developer for each individual sub module. Every test case of unit test is only passed when they satisfy all requirements of specification. That is, in unit test 100% completeness is needed for all test cases.

- **Pass/ Fail criteria of integration test**

Integration testing is done after unit test. So, when developers' individual sub modules will be integrated, this test will start. In this website, integration test is considered to be passed only when authentication systems will work correctly, admin sub module, student sub module will give expected output according to requirements. Here, all errors of internal module must be fixed. If any test case gives result that program cannot store authentication information, excel, student information, event information, new items in appropriate database, it will fail.

- **Pass/ Fail criteria of acceptance test**

In this test tester will match user requirements with program outputs. Here, in internal module 5% test case incompleteness can be tolerated. Test cases related to external module must pass.

8.2.1.9 Suspension and resumption Criteria

This section describes suspension criteria for test cases in the test plan. Here, we also give conditions when testing will be resumed.

- **Unit Test**

When developers complete their module they will start unit test. At first they will test methods of underlying classes. Then they will start for testing relationship between classes. When any test reports failure, test will be suspended. After analyzing and resolving failed test case, test activity will be resumed.

- **Integration Testing**

This testing is done by integrating modules of unit testing. This testing will stop when each module cannot perform their task after integrating and relationship between modules is not established that is some data can’t pass within modules. After resolving error and establishing expected relationship between modules testing will restart.

- **Security Testing**

Security testing will be suspended if major test cases (test cases related to limit unauthorized access) fail to perform desire output. When failed test cases behave normally then security test will be resumed.

- **Performance Testing**

Performance test will stop when

1. Website response time is greater than .1 second if no feedback is necessary, 5 second if user feedback is needed.
2. Website is not available due to crushing of any sub module.
3. Database failed to handle many request at a time.
4. Fail to give same service for different user.

Testing will restart when these problems will be fixed by developers.

- **User Interface Testing**

In user interface testing, tester will test interface of the website. If following criteria will fail, testing will be suspended.

1. Levels, texts, images, spelling of texts, font size are not correct.
2. Links, buttons, menu items are not working properly.
3. Error messages, popup boxes are not working properly.
4. Lack of information of current activity.

After fixing discussed things testing will restart.

- **Acceptance Testing**

When program fails any test case related to normal requirement and expected requirement of user then acceptance test will be suspended. When failed test cases are recovered then testing will restart.

8.2.1.10 Test Deliverables

The test deliverables for this system are listed as follows.

i. Test plan for unit test
ii. Test plan for integration test
iii. Test plan for acceptance test
iv. Test environment
v. Test defect report
vi. Test result

8.2.1.11 Environmental Needs

The following elements are needed to perform and support the overall testing process for the website.

i. Access to the development part of the website
ii. Access to the database for giving the input for testing
iii. Access to the recovery system of the website
iv. Tester computer should have sql database and php
v. All modern web browser installed in the tester’s pc
vi. Presence of needed platform

8.2.1.12 Responsibilities

Table 8.2 Responsibilities for Testing the System

Task	Project supervisor	Test Stuff	End user
Testing external module		✓	
Testing internal modules		✓	
Creating test report		✓	
Reviewing test report	✓		
Testing and creating report for the integrated system		✓	
Reviewing integrated system test report	✓		
Creating the acceptance test report		✓	
Reviewing the acceptance test report	✓		✓
Changing control	✓	✓	

8.2.1.13 Schedule

The time allocated for the testing activities are defined in the project plan. The test should be done by the defined person within the time period for the following tasks.

Table 8.3: Schedule for Testing the System

Task	Responsible personnel	Time
Checking the requirement document	Test Stuff	2 days
Development of master test plan	Test Stuff	5 days
Developing the integration test plan	Test Stuff	2 days
Reviewing the integration test plan	Test Stuff	1 day
Developing the acceptance test plan	Test Stuff	2 days
Reviewing the integration test plan	Test Stuff	1 day
Testing the external module	Test Stuff	2 days
Testing the internal module	Test Stuff	5 days

8.2.2 Test Result

The system is tested as documented in the test plan. The tests are executed for several times for correct result. This section of the document depicts the test result of the website TwirlingHall.com.

8.2.2.1 Unit Test

Equivalence Class Partitioning of Authentication

Input/Output	Valid Class	Invalid Class
Email address	V1-All characters V2-Symbol V3-numbers	I1--- <17
Password	V4-All characters V5-Symbol V6-numbers	

Table 8.4: Equivalence Class Partitioning of Authentication

ECP Test Cases for Authentication

ECP test	Email address	Password	Expected result
1	V1^V2	V4	Account created
2	V1^V2^V3	V5	Account created
3	V1^V2^V3	V4^V6	Account created
4	I1	V4	Error
5	I1	V5	Error
6	I1	V6	Error
7	V1^V2	V5	Error
8	V1^V2	V6	Error

Table 8.5 ECP Test Cases of Authentication

All Pair Testing for Authentication

Test case	Email address	Password
Tc1	V1	V4
Tc2	V1	V5
Tc3	V1	V6
Tc4	V2	V4
Tc5	V2	V5
Tc6	V2	V6
Tc7	V3	V4
Tc8	V3	V5
Tc9	V3	V6
Tc10	I1	V4
Tc11	I1	V5
Tc12	I1	V6

Table 8.6 All Pair Testing for Authentication

Equivalence Class Partitioning of Student Creation

Input/Output	Valid Class	Invalid Class
HSC Roll	V1- Number with length 6	I1- All number less than length 6 I2-All number greater than length 6 I3-All characters and symbols
HSC Passing year	V2—1999-2014	I4-- <1999 I5-- >2014 I6-- All characters and symbol I7- All numbers less than or greater than length 4

HSC Board	V3- All characters	I8- Symbols I9-Numbers
SSC Roll	V4- Number with length 6	I10- All number less than length 6 I11-All number greater than length 6 I12-All characters and symbols

Table 8.7 Equivalence Class Partitioning of Student Creation

ECP Test Cases of Student Creation

ECP test	HSC roll	HSC passing year	HSC board	SSC roll	Expected result
1	V1	V2	V3	V4	Student created
2	V1	I4	I8	I10	Error
3	V1	I4^I6	I8	I11	Error
4	V1	I4^I7	I8	I12	Error
5	I1	V2	I9	I10^I12	Error
6	I2	V2	I8	I11^I12	Error
7	I3	V2	I8^I9	I10^I12	Error
8	I1^I3	I5^I6	I9	I10	Error
9	V1	I7	I8	I12	Error
10	V1	V2	V3	I12	Error

Table 8.8 ECP Test Cases of Student Creation

All Pair Testing for Student Creation

Test cases	HSC roll	HSC passing year	HSC board	SSC roll
Tc1	V1	V2	V3	V4
Tc2	V1	I1	V3	V4
Tc3	V1	I2	V3	V4
Tc4	V1	I3	V3	V4
Tc5	V1	V2	I4	V4
Tc6	V1	V2	I5	V4

Tc7	V1	V2	I6	V4
Tc8	V1	V2	I7	V4
Tc9	V1	V2	V3	I8
Tc10	V1	V2	V3	I9

Table 8.9 All Pair Testing for Student Creation

Equivalence Class Partitioning of Search Student

Input/Output	Valid class	Invalid class
Search by	V1-All characters V2-All symbols	I1-All numbers I2--< length 1
Select	V3-All characters V4-All symbols V5-All numbers	I3--< length 1

Table 8.10 Equivalence Class Partitioning of Search Student

ECP Test Cases of Search Student

ECP test	Search by	Select	Expected result
1	V1	V3	Student info
2	V1^V2	V3	Student info
3	V1	V4	Student info
4	V1	V5	Student info
5	V1	I3	Error
6	V1^V2	I3	Error
7	I1	I3	Error
8	I2	I3	Error
9	I2	V3^V4^V5	Error
10	I2	V3^V4	Error

Table 8.11 ECP Test Cases of Search Student

All Pair Testing for Search Student

Test cases	Search by	Select
Tc1	V1	V3
Tc2	V1	V4
Tc3	V1	V5
Tc4	V2	V3
Tc5	V2	V4
Tc6	V2	V5

Tc7	V1	I1
Tc8	V1	I2
Tc9	V2	I1
Tc10	V2	I2
Tc11	V3	I1
Tc12	V3	I2

Table 8.12 All Pair Testing for Search Student

Equivalence Class Partitioning of ID Card Generation

Input/Output	Valid Class	Invalid Class
HSC Roll	V1- Number with length 6	I1- All number less than length 6 I2-All number greater than length 6 I3-All characters and symbols
HSC Passing year	V2—1999-2014	I4-- <1999 I5-- >2014 I6-- All characters and symbol I7- All numbers less than or greater than length 4
HSC Board	V3- All characters	I8- Symbols I9-Numbers
SSC Roll	V4- Number with length 6	I10- All number less than length 6 I11-All number greater than length 6 I12-All characters and symbols
Class roll no	V5- All characters with length less than 11 V6- All numbers with length less than 11 V7- All symbols with length less than 11	I13--< length 1
Current session	V8—1999-2014	I14-- <1999 I15-- >2014

		I16-- All characters and symbol
Card no	V9- All characters with length less than 11 V10- All numbers with length less than 11 V11- All symbols with length less than 11	I17--< length 1

Table 8.13 Equivalence Class Partitioning of ID Card Generation

ECP Test Cases of ID Card Generation

ECP test	HSC roll	HSC passing year	HSC board	SSC roll	Class roll no	Current session	Card no	Expected result
1	V1	V2	V3	V4	V5	V8	V9	Student created
2	V1	I4	I8	I10	I13	I14	I17	Error
3	V1	I4^I6	I8	I11	V6	V8	I17	Error
4	V1	I4^I7	I8	I12	V7	I16^I16	V9^V10	Error
5	I1	V2	I9	I10^I12	V6^V7	I16	V11	Error
6	I2	V2	I8	I11^I12	V5	V8	V11	Error
7	I3	V2	I8^I9	I10^I12	V7	I16	V10	Error
8	I1^I3	I5^I6	I9	I10	I13	I14	V9	Error
9	V1	I7	I8	I12	V7	V8	I17	Error
10	V1	V2	V3	I12	V6	I14	V9	Error

Table 8.14 ECP Test Cases of ID Card Generation

All Pair Testing for ID Card Generation

Test cases	HSC roll	HSC passing year	HSC board	SSC roll	Class roll no	Current session	Card no
Tc1	V1	V2	V3	V4	V5	V8	V9
Tc2	V1	I1	V3	V4	V5	V8	V9
Tc3	V1	I2	V3	V4	V5	V8	V9
Tc4	V1	I3	V3	V4	V5	V8	V9
Tc5	V1	V2	I4	V4	V5	V8	V9
Tc6	V1	V2	I5	V4	V5	V8	V9

Tc7	V1	V2	I6	V4	V5	V8	V9
Tc8	V1	V2	I7	V4	V5	V8	V9
Tc9	V1	V2	V3	I8	V5	V8	V9
Tc10	V1	V2	V3	I9	V5	V8	V9
Tc11	V1	V2	V3	V4	V6	V8	V9
Tc12	V1	V2	V3	V4	V7	V8	V9
Tc13	V1	V2	V3	V4	I13	V8	V9
Tc14	V1	V2	V3	V4	V5	I14	V9
Tc15	V1	V2	V3	V4	V5	I15	V9
Tc16	V1	V2	V3	V4	V5	V16	V9
Tc17	V1	V2	V3	V4	V5	V8	V10
Tc18	V1	V2	V3	V4	V5	V8	V11
Tc19	V1	V2	V3	V4	V5	V8	I17

Table 8.15 All Pair Testing for ID Card Generation

8.2.2.2 User Interface Testing

The goal of user interface testing is to ensure the interface provide users an interface containing enough information and easy to interact. It also ensures that the objects of the interface performs their role properly.

- **Test Objective**
  i. Verification of the interaction between pages, tab keys and mouse movements.
  ii. Verification of menus, size, position different of window objects.

- **Test Technique**
  Performing test and modify them for each window of the website to verify the objectives mentioned.

- **Completion Criteria**
  Each window successfully verified to remain consistent with benchmark version or within acceptable standard.

8.2.2.3 Security Testing

The main purpose of this test is to ensure the security of the information provided by the user. The security testing is to be done for application security and system security.

Application security ensures that the access to the data of the system is secured that means the users are restricted to specific functions available to them. All the users are not allowed to all the functions as per to security of the system. For example, the students do not have rights to change their information other than the contact information because all other information needs some clarification from the hall authority.

The system security ensures that only those users granted access to the system are capable of accessing the applications and only through the appropriate gateways.

- **Test Objective**
  1. Verification of the access of the data to the users.
  2. Verification of the access to the system by the user.

- **Test Technique**
  1. Making a list of the users and data that are accessed by them.
  2. Perform and modify tests for each type of user and verify them.

- **Completion Criteria**

  The users have only access to the data available for them and the system functions properly in this case.

8.2.2.4 Regression Testing

The main goal of this test is to ensure that the change in module does not cause change in another module. As the website is developed following the agile process, any time the requirement can change. When this new requirement is fulfilled, this should not change the functions of the other modules. This is ensured through this test.

- **Test Objective**

  Verification of proper functions of all the modules of the system.

- **Test Technique**
  1. To run test for verifying the proper functionality of user access to the system and data, interaction between pages, tabs etc.
  2. The test case is validate for the proper functioning of the modules.

- **Completion Criteria**
  1. All the functions of all the modules are verified.
  2. All the defects have been properly addressed.

8.2.2.5 System Testing

The system testing ensures the proper data acceptance, processing and retrieval and the appropriate implementation of the business rules. This type of testing is done through the black box testing. The black box testing is the testing where the test is run by interacting with the system through the interface that is the GUI and analyzing the output.

- **Test Objective**
  1. Ensure proper data acceptance, processing, retrieval of the system.
  2. Ensure the proper functionalities of the system.

- **Test Technique**

Run test case for each use case of the system with the valid and invalid data to verify the proper functionalities of the system.

- **Completion Criteria**
  i. The system runs perfectly for the valid input
  ii. System shows message for invalid inputs

8.2.2.6 Data and Database Integrity Testing

The databases and the database processes should be tested as separate systems. These systems should be tested without the applications (as the interface to the data). Additional research into the DBMS needs to be performed to identify the tools / techniques that may exist to support the testing identified below.

- **Test Objective**

  Ensure Database access methods and processes function properly and without data corruption.

- **Test Technique**
  i. Invoke each database access method and process, seeding each with valid and invalid data.
  ii. Inspect the database to ensure the data has been populated as intended, all database events occurred properly, or review the returned data to ensure that the correct data was retrieved.

- **Completion Criteria**

  All database access methods and processes function as designed and without any data corruption.

8.2.2.7 Performance Testing

The performance testing is done to verify the response time of all pages, availability, portability and scalability of the website. This testing is executed several times using different load on the system. The initial test is done with a normal load which is expected to be the load on the system. The further test is done using the peak load.

- **Test Objective**

  Verify system response time in case of

  i. Normal Load
  ii. Peak Load

- **Test Technique**

  i. Use test scripts for system test.
  ii. Modify the scripts to add extra load on the system for different tests.
  iii. Test should be done with one machine and repeated with multiple machines.

- **Completion Criteria**

  Successful completion of the test for both single machine and multiple machine within expected amount of time.

8.2.2.8 Load Testing

Load testing measures the system-under-test to varying workloads to evaluate the system's ability to continue to function properly under different workloads. The goal of load testing is to determine and ensure that the system functions properly beyond the expected maximum workload. Additionally, load testing evaluates the performance characteristics (response times, transaction rates, and other time sensitive issues).

- **Test Objective**

  Verify System Response time for designated transactions or business cases under varying workload conditions.

- **Test Technique**

  i. Use test scripts for system test.
  ii. Modify the scripts to add extra load on the system for different tests.

- **Completion Criteria**

  Successful completion of the test for both single machine and multiple machine within expected amount of time.

8.2.2.9 Acceptance Testing

The acceptance test is carried out by the end user with the help of test stuff and project manager. This test is done after the successful completion of the system test. That is the system can only enter the process of this test after all the major errors in the system are corrected. Through this test, the end users are given an opportunity to verify whether the developed system meets all the requirements of them correctly. Alongside this, it gives an idea to the project team whether the system is ready for the deployment. The test strategy for the acceptance test is as follows.

i. A risk assessment will be made for the system.

ii. The test cases with high risk has to be re tested.

iii. Decision will be made upon the difference in the result of original result and the re tested result of the high risk test cases. If the difference is greater, the team members will take appropriate actions to minimize the difference.

iv. The end user will test the system to ensure that the system has successfully achieved all the requirements of them. If the users feel any negative issues, they will meet with the team members to minimize the negative impact.

Chapter 9: User Manual

This chapter discusses about user manual which consists how to interact with different modules of the systems. It explains the user manual dividing into two part, such as administrative and user (student and global) responsive interface.

9.1 Website Privileges

Website privileges according to user types are discussed below.

- **Administrator**

Administrators have access to all website functions and features. They can add new student and admin.

- **Student**

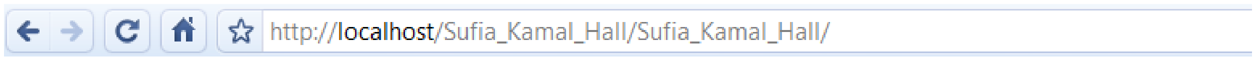
Students have right to update their registration form and see external side of the website.

- **Other users**

All other users except administrators and students can see home, about, search and contact page.

Accessing the TwirlingHall.com Website

By writing **http://localhost/Sufia_Kamal_Hall/Sufia_Kamal_Hall/** in user's browser URL bar and pressing enter to access the website, user can see the "Home page" of the website.


http://localhost/Sufia_Kamal_Hall/Sufia_Kamal_Hall/

9.2 Home Page

All users can see the home page of the website. Home page contains four menu items and the login button.

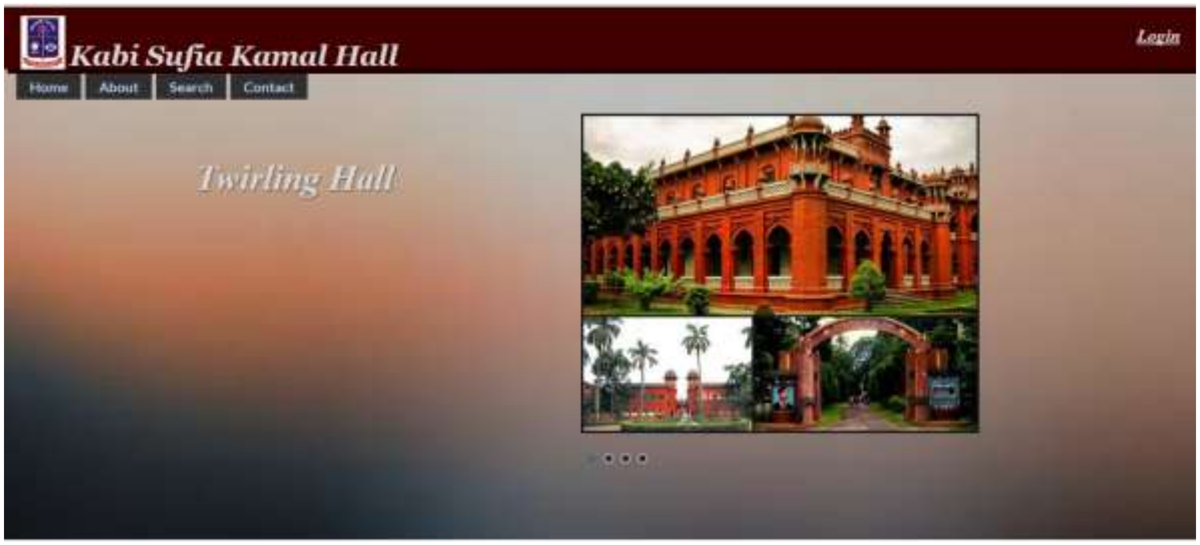
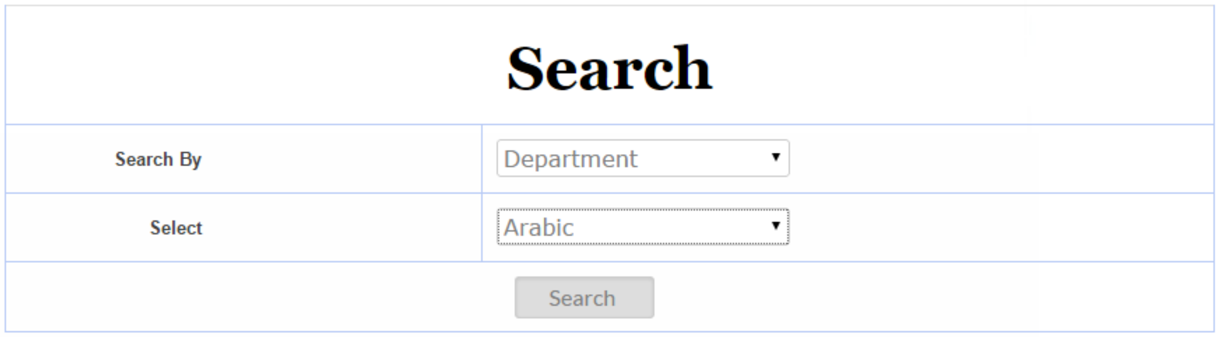


Figure 9.1: Home page of TwirlingHall.com

The user can go to the about, search and contact section from here. The about section depicts the details of the website. The contact section provides the contact information of the hall with the google map. The Search page provides an interface to search student information by different categories. Users have to fill “Search By” and “Select” input field to see a student’s information.



Search

Search By	Department
Select	Arabic

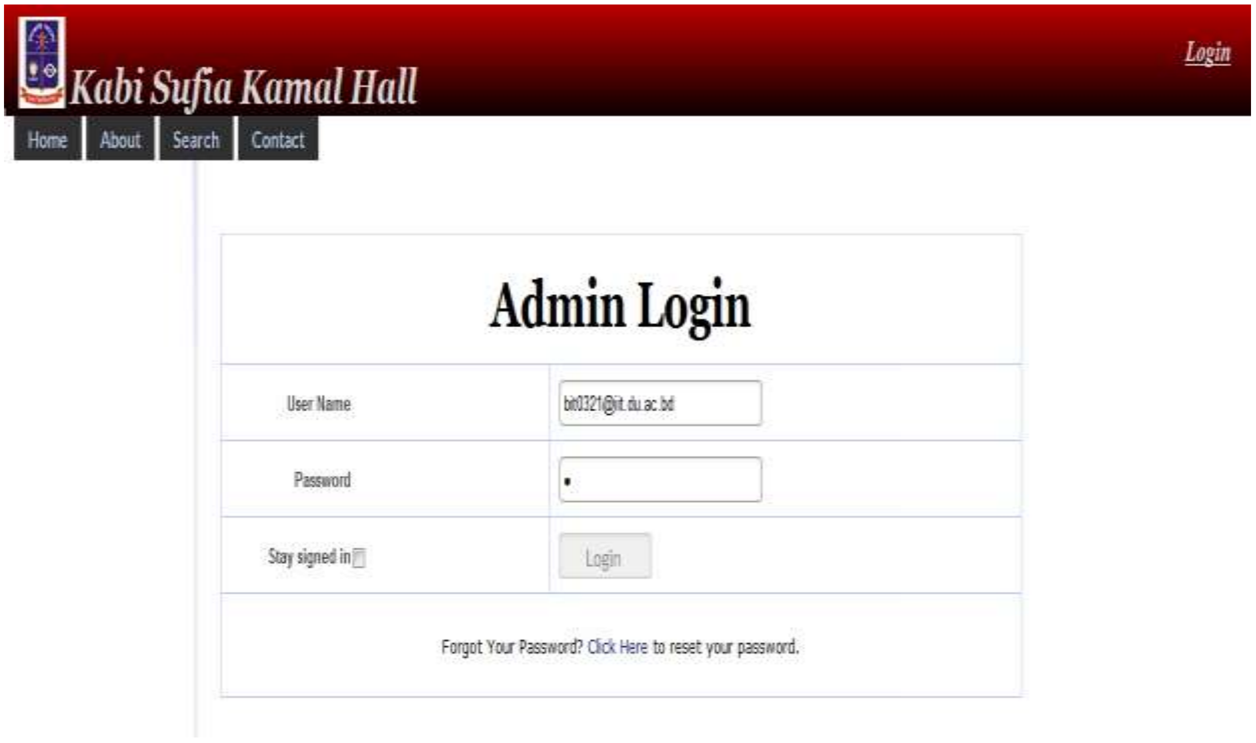
Figure 9.2 Searching Student by Category

9.3 Website Internal Access

TwirlingHall.com is a secure web based application so each user must have an account to access the internal part of the website.

9.3.1 Admin Login

Administrators automatically have login access using the email and password that was specified from the beginning of development.



Admin Login

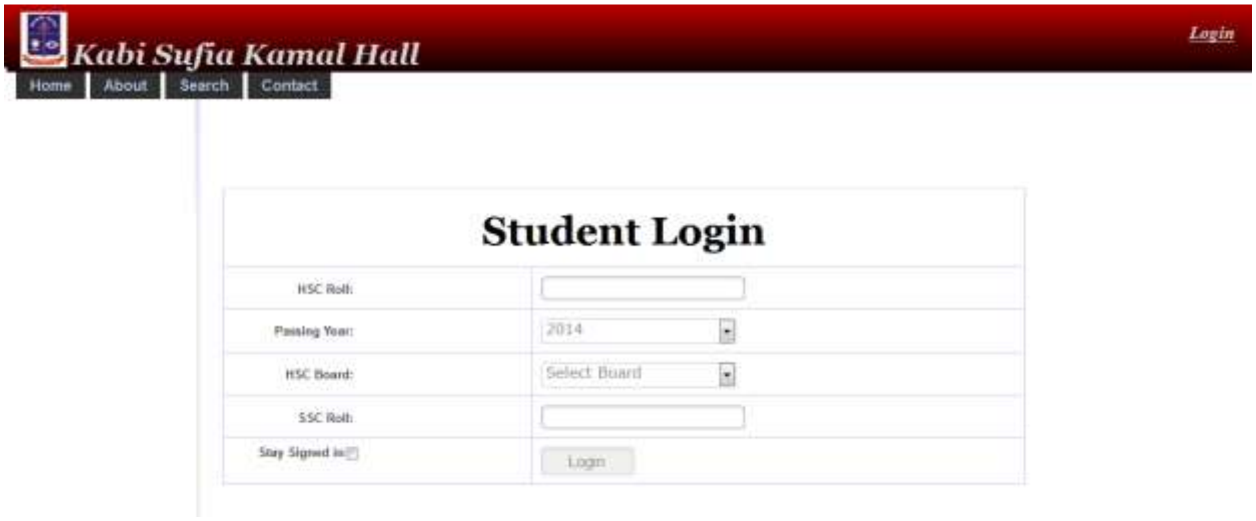
User Name	bit0321@iit.du.ac.bd
Password	•
Stay signed in	Login

Forgot Your Password? Click Here to reset your password.

Figure 9.3 Admin Login Interface

9.3.2 Student Login

In order to login as a student user has to fill HSC roll, SSC roll, HSC passing year and HSC board input field with valid information.



Student Login

HSC Roll:	
Passing Year:	2014
HSC Board:	Select Board
SSC Roll:	
Stay Signed in: ☐	Login

Figure 9.4 Student Login Interface

9.3.3 Student Profile

After logged in as a student a user can see home, profile and registration page. This page contains basic info of a student. Students can edit their info by clicking “Edit Profile” button. They can save edited info by clicking “Save” button. In the registration page, a student can edit their image and fill up their registration form.

9.3.4 Admin Profile

After logged in as an admin a user can see following menu items.

9.3.4.1 Excel

Here, admin can upload student information through excel file. After clicking browse button user can see a window for choosing excel file. Then, user will press upload button to store student information in database.

9.3.4.2 Card

This menu contains three submenus: ID card, boarding card and testimonial.

9.3.4.2.1 ID Card Generation

ID card generation need a student's HSC roll, SSC roll, HSC passing year, HSC board, class roll no, current session and ID card no. ID card will generate in two steps. Pressing "Create front page" button will create the front page and "create back page" button will create the back page of ID card.

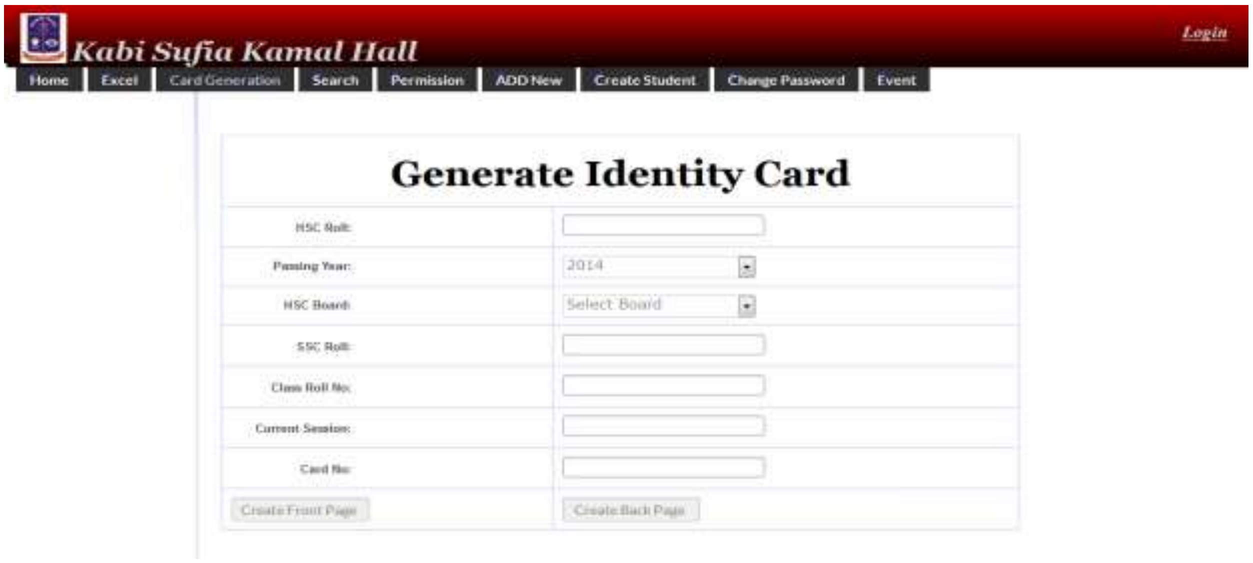
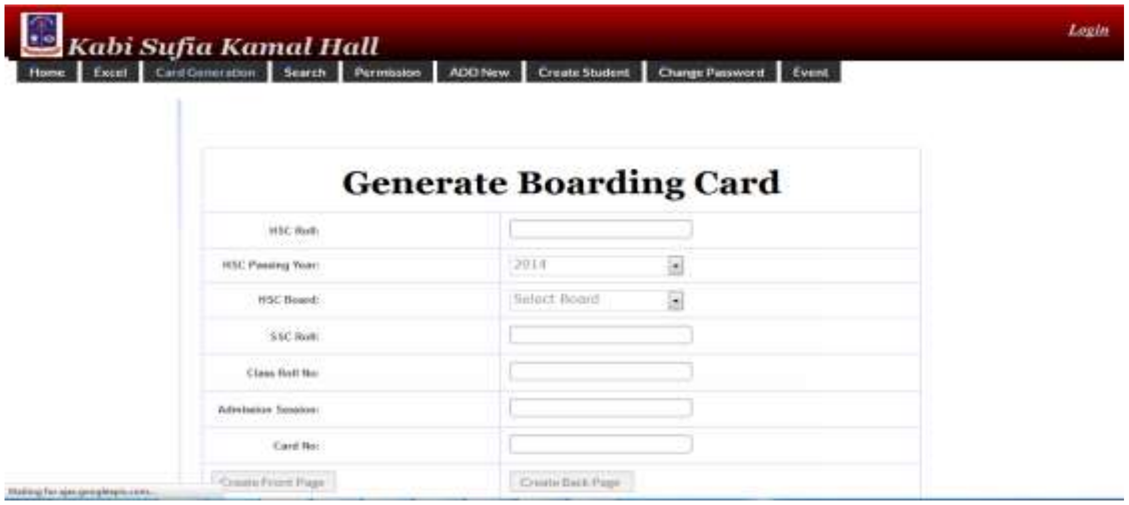


Figure 9.5 Interface of ID Card Generation

9.3.4.2.2 Boarding Card Generation

Boarding card follows the same way of ID card generation. In input field boarding card need admission session and boarding card no instead of current session and ID card no.



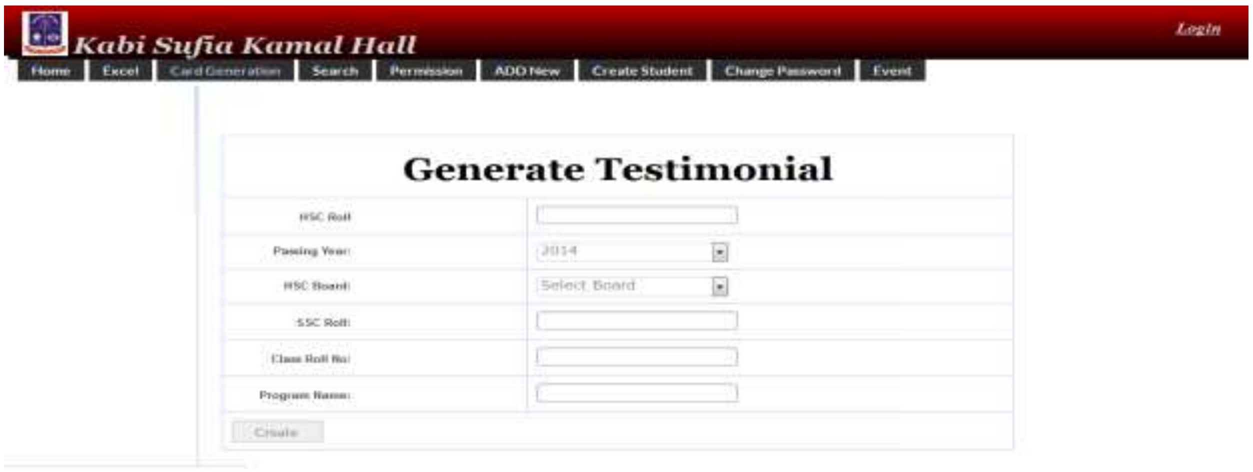
Generate Boarding Card

HSC Roll:	
HSC Passing Year:	2014
HSC Board:	Select Board
SSC Roll:	
Class Roll No:	
Admission Session:	
Card No:	
Create Front Page	Create Back Page

Figure 9.6 Interface of Boarding Card Generation

9.3.4.2.3 Testimonial Generation

Testimonial generation needs HSC roll, SSC roll, HSC passing year, HSC board, class roll no and program name. Pressing “create” button will create testimonial in pdf format.



Generate Testimonial

HSC Roll	
Passing Year:	2014
HSC Board:	Select Board
SSC Roll:	
Class Roll No:	
Program Name:	
Create	

Figure 9.7 Interface of Testimonial Generation

9.3.4.3 Search

This page provides an interface to search student information by category. At first user will choose category from “search by” drop down list then sub category will be chosen from the “select” drop down list.

9.3.4.4 Permission

In permission page admin can give permission to student for editing student info and student SIF form. User will select permission type and change status for permission.

9.3.4.5 Add New Item

This page provides the interface for adding new faculty, department, session, room-number and institute. Admin will select the category and write name of the new item. After pressing the "add item" user can see a message.

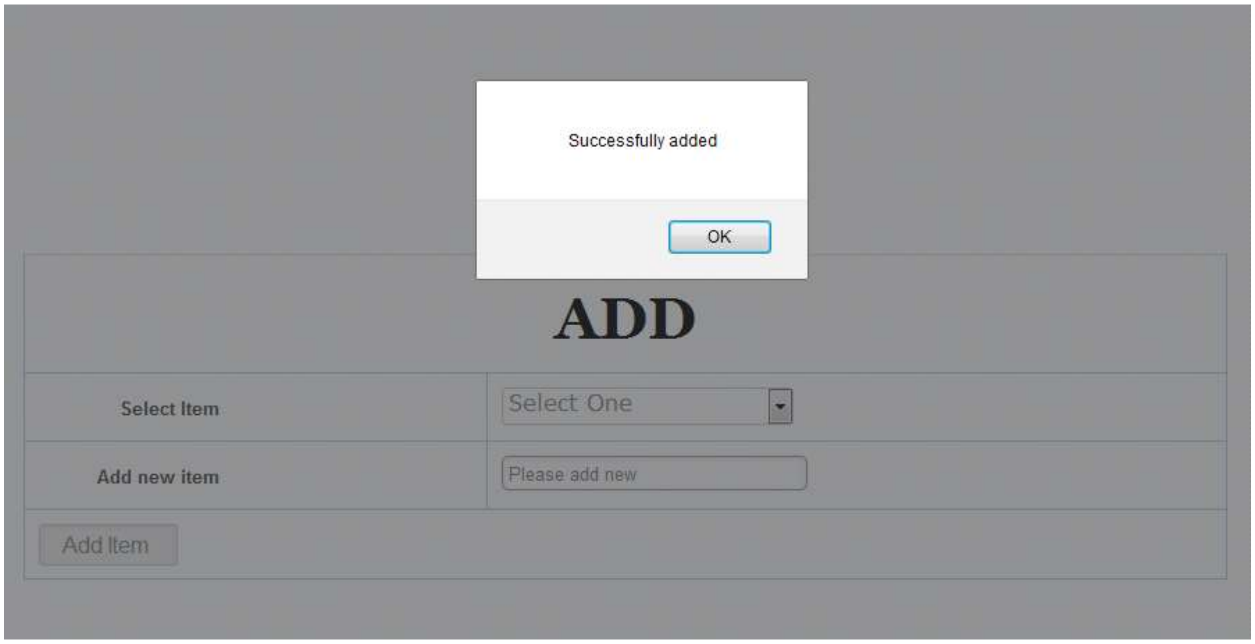

Successfully added

Figure 9.8 Adding a New Item

9.3.4.6 Create Student and Admin

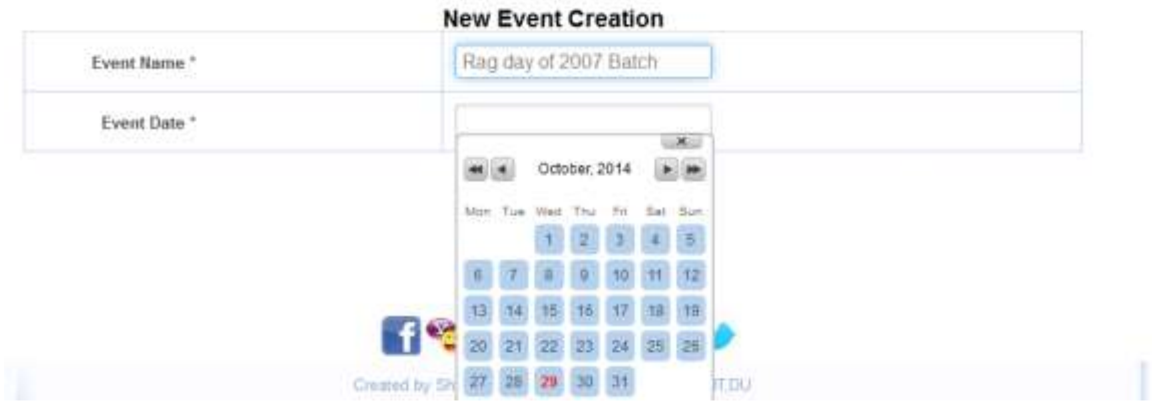
Here, an admin can create new student and admin. For creating new admin email address and password is needed. New student creation need student's HSC roll, SSC roll, HSC passing year and HSC board.

9.3.4.7 Change Password

Users can change their existing password from this interface by giving old password and new password in textbox.

9.3.4.8 Event

Creating new event needs to fill up event name and event date textbox.



New Event Creation

Event Name *	Rag day of 2007 Batch
Event Date *	

Figure 9.9 Creating a New Event Interface

Chapter 10: Conclusion

This chapter is about conclusion. This chapter will briefly summarize and conclude the proposed project.

The project is developed using PHP and MySQL, based on the requirement specification of the user and the analysis of the existing system, with flexibility for future enhancement.

The expanded functionality of today's software requires an appropriate approach towards software development. This hall management software is designed for people who want to manage various activities in the hall. For the past few years, the numbers of educational institutions are increasing rapidly.

Thereby the numbers of halls are also increasing for the accommodation of the students studying in this institution. And hence there is a lot of strain on the person who are running the hall and software's are not usually used in this context. This particular project deals with the problems on managing a hall and avoids the problems which occur when carried manually.

Identification of the drawbacks of the existing system leads to the designing of computerized system that will be compatible to the existing system with the system which is more users friendly and more GUI oriented.

The security test is performed based on OWASP security testing guideline which helps to mitigate common security vulnerabilities in this project. This report presents the extensive security testing procedure, result on "Sufia Kamal Hall" project as well as functional requirement testing.

References

[1] Roger S. Pressman, Software Engineering: A Practitioner's Approach, 7th International edition

[2] http://www.sqa.org.uk/e-learning/SDM02CD/page_12.htm

[3] http://www.brighthubpm.com/project-planning/63692-project-feasibility-study-samples/

[4] Sindre and Opdahl (2001). "Capturing Security Requirements through Misuse Cases"

[5] http://yehg.net/lab/pr0js/files.php/googlehacker.zip

[6] http://www.mcafee.com/uk/downloads/free-tools/sitedigger.aspx

[7] http://net-square.com/httprint.html

[8] http://www.computec.ch/projekte/httprecon/

[9] http://www.insecure.org

[10] http://nc110.sourceforge.net

[11] http://curl.haxx.se/

[12] http://curl.haxx.se/

[13] http://www.aboutsecurity.net

[14] https://www.owasp.org/index.php/OWASP_Guide_Project