Examiner’s Board Meeting

Taylor Letsoaka : Front-End
Nando Bingani : Back-End
Lesetsa Mafisa : Front-End
Ziphozonke Mbatha : Back-End
Nkavelo Nxumalo

August 2018

Contents

1 Introduction

1.1 Purpose

The purpose of this document is to present a detailed description of the Examiners' Board Meeting Web-Application. It will explain the purpose, functionality, features and the interfaces of the system. The constraints under which it must operate and how the system will react to external stimuli. This document is intended for both the stakeholders and the developers of the system and will be proposed to the Faculty of Science for its approval .This will give a user a whole feel of the Web-application, it's basic functionality and basic operations.It will also give a brief explanation of how we improved the security .

1.2 Problem Statement

The faculty of sciences has to meet up at every end of each academic year to discuss the progress of all the students in the faculty . The meeting is mainly focused on determining the progress of learners who are on the boarder line , learners that have failed by a very small margin . The meeting is attended by faculty members and representatives from various schools of the faculty . The meeting is lead by the Dean of schools and the Registrar , they look at each and every student at a time. They don't discuss a passed student , they normally let them pass without any discussions or voting as its always anonymous .They vote based on the information and merits given by the faculty school representatives of the student . The major problem is that the whole process is time consuming and very tedious since they use a pdf document to check each student's progress by order and then vote by show of hands . The document is too larger and thick since it contains all the records of the students in the faculty . The manual process has a huge room for human error, deliberate error and does not have a tight security.

1.3 Overview of the project

This software system is a Web-Application System for the Faculty of Science. This system is designed to maximize the productivity by providing tools to assist in automating the voting system review and publishing process, which would otherwise have to be performed manually. By maximizing the faculty registers work efficiency and production the system will meet the faculty's needs while remaining easy to understand and use. More specifically, this system is designed to allow the faculty to manage and communicate with a group of Administrators, Dean and Lecturers to vote on the website. The software will facilitate voting between Administrators, Dean and Lecturers using the web-application. This is a Java web application that will allow members of the Science Faculty at Wits University to approve the promotion, failure or exclusion status of each student. This application enables each student's marks to be viewed on any device used by the participants of the meeting .

1.4 Project objectives

- Replace the old paper system with an electronic system which keeps track of all the students and their information in the various schools of the faculty.
- This system is designed to maximize the productivity by providing tools to assist in automating the voting system
- Synchronization of the Web-Application with the mobile devices.
- Make the system secure and easy to operate by members of the committee .
- Voting must be secure and the outcome must only be affected by the voting results

1.5 Stakeholders

Student : A registered individual for a specific degree in the faculty.
Faculty : Obtain correct information about each student.
Academic Staff : Be able to vote on each student mark

2 Requirements Specifications

2.1 Glossary

1. **Database** - Collection of all the information monitored and used by the system.
2. **User** - A faculty member that is part of the administration or part of the mark review committee(The Dean, Registrar and Faculty Representatives).
3. **Software Requirements Specification** - A document that completely all of the functions of a proposed system and the constraint under which it must operate. For example this document.
4. **Faculty of Science** - A group of university scientific schools.
5. **School** - A department concerned with a specific discipline e.g School Of Mathematics.
6. **Member** - An employee of the faculty of Science who is part of the mark review committee e.g The Dean Of Faculty.
7. **Student** - A registered individual for a specific degree in the faculty.
8. **Vaadin** - An open-source platform for web application development.
9. **MySQL** - an open-source relational database management system.
10. **Dean Of Schools** - The head of a university faculty of Science.
11. **Registrar** - An official responsible for keeping a register or official records.
12. **End user** - Anyone who is using the Web-Application.
13. **PCD** - Permitted to proceed.
14. **MBR** - Exclusion waived-must return to the same year of study.
15. **RET** - Must return to complete requirements for year of study.
16. ******* - Result depends on outcome of Deferred, Supplementary, Additional Test.
17. **MRNM** - Minimum requirements not met.
18. **SDD** - Software Design Document.

2.2 References

IEEE Std 830-1998, IEEE Recommended Practice for Software Requirements Specifications (Revision of IEEE Std 830-1993), IEEE Computer Society, 1998.

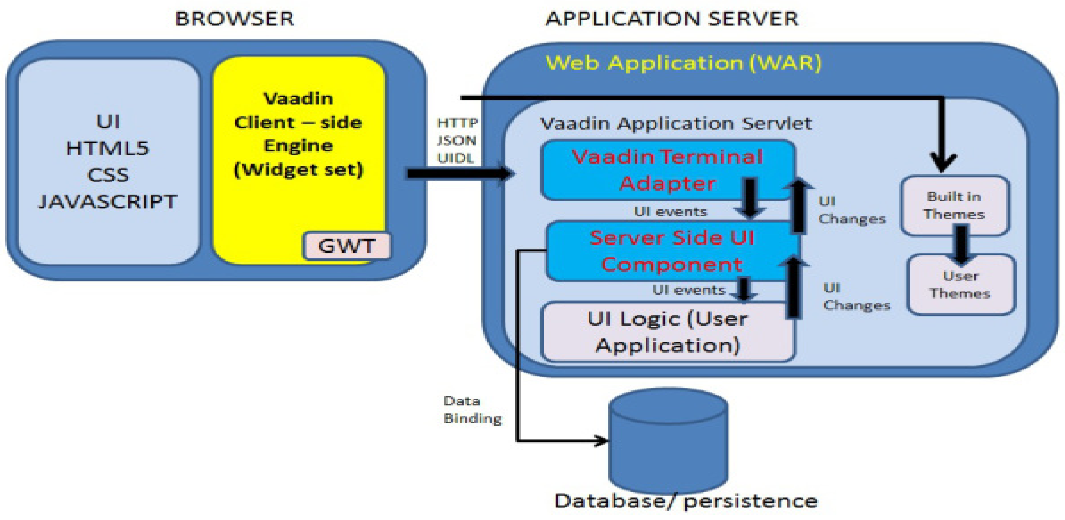
2.3 Overview

The forthcoming section i.e. the Overall Description, will give a more broad explanation of how the product is intended to work. It will delve into things like the product functions, design constraints, assumptions etc. The Specific Requirements section is a derivation of the preceding section written specifically for developers hence it uses a lot of technical jargon.

2.4 Product perspective

An implementation of a web based client-server site for a University Faculty.

2.4.1 System and Software Interfaces



2.4.2 User Interfaces

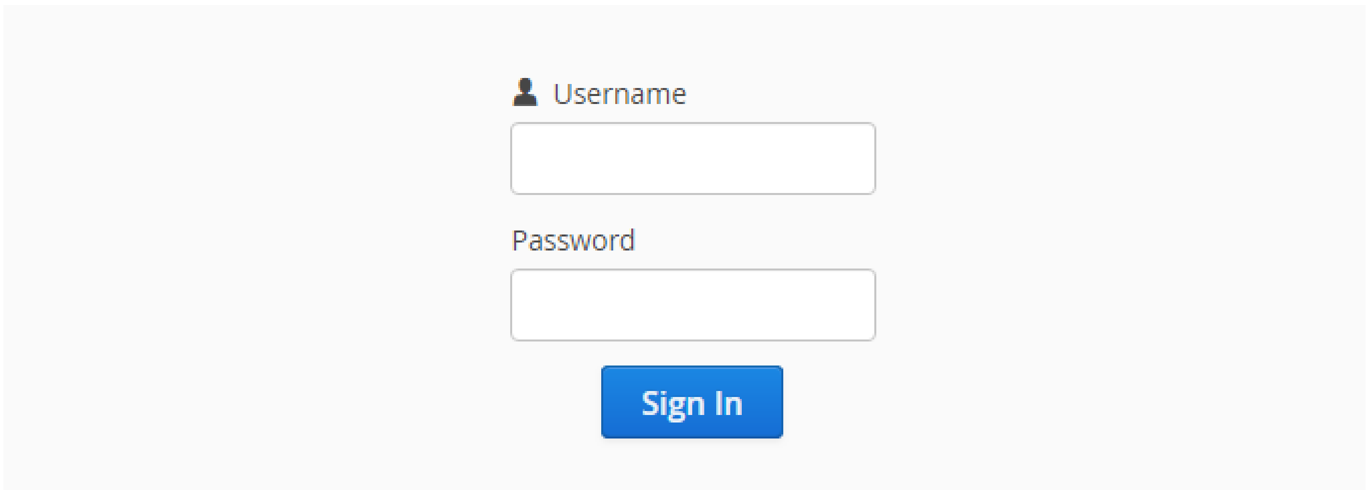
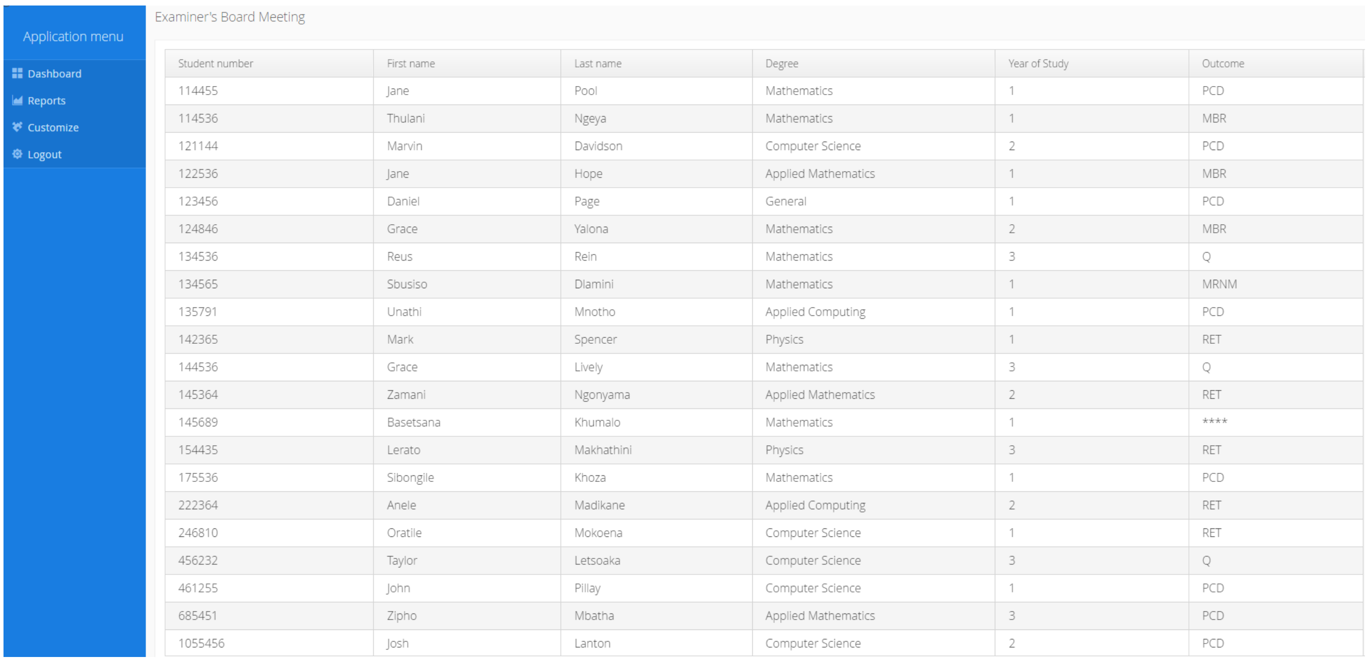


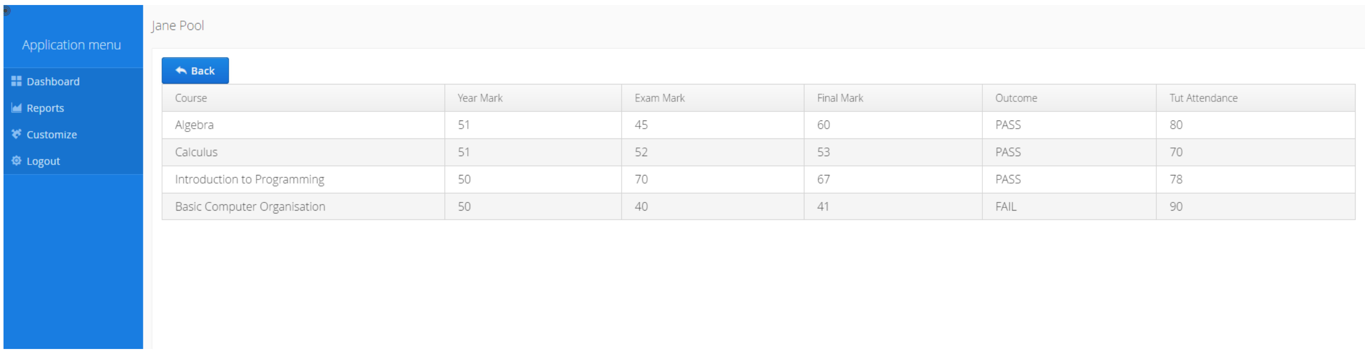
Figure 1: Login Page



Examiner's Board Meeting

Student number	First name	Last name	Degree	Year of Study	Outcome
114455	Jane	Pool	Mathematics	1	PCD
114536	Thulani	Ngeya	Mathematics	1	MBR
121144	Marvin	Davidson	Computer Science	2	PCD
122536	Jane	Hope	Applied Mathematics	1	MBR
123456	Daniel	Page	General	1	PCD
124846	Grace	Yalona	Mathematics	2	MBR
134536	Reus	Rein	Mathematics	3	Q
134565	Sbusiso	Dlamini	Mathematics	1	MRNM
135791	Unathi	Mnotho	Applied Computing	1	PCD
142365	Mark	Spencer	Physics	1	RET
144536	Grace	Lively	Mathematics	3	Q
145364	Zamani	Ngonyama	Applied Mathematics	2	RET
145689	Basetsana	Khumalo	Mathematics	1	****
154435	Lerato	Makhathini	Physics	3	RET
175536	Sibongile	Khoza	Mathematics	1	PCD
222364	Anele	Madlkane	Applied Computing	2	RET
246810	Oratile	Mokoena	Computer Science	1	RET
456232	Taylor	Letsoaka	Computer Science	3	Q
461255	John	Pillay	Computer Science	1	PCD
685451	Zipho	Mbatha	Applied Mathematics	3	PCD
1055456	Josh	Lanton	Computer Science	2	PCD

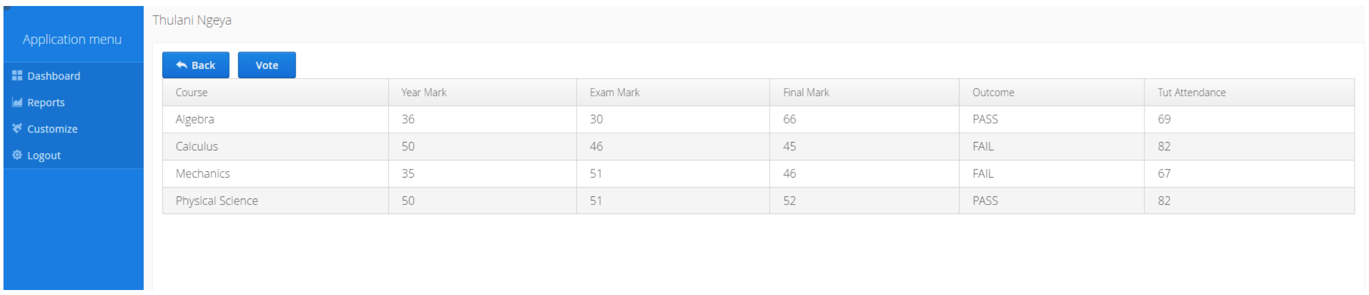
Figure 2: List of all students



Jane Pool

Course	Year Mark	Exam Mark	Final Mark	Outcome	Tut Attendance
Algebra	51	45	60	PASS	80
Calculus	51	52	53	PASS	70
Introduction to Programming	50	70	67	PASS	78
Basic Computer Organisation	50	40	41	FAIL	90

Figure 3: Marks of student not eligible for a vote since they passed overall



Thulani Ngeya

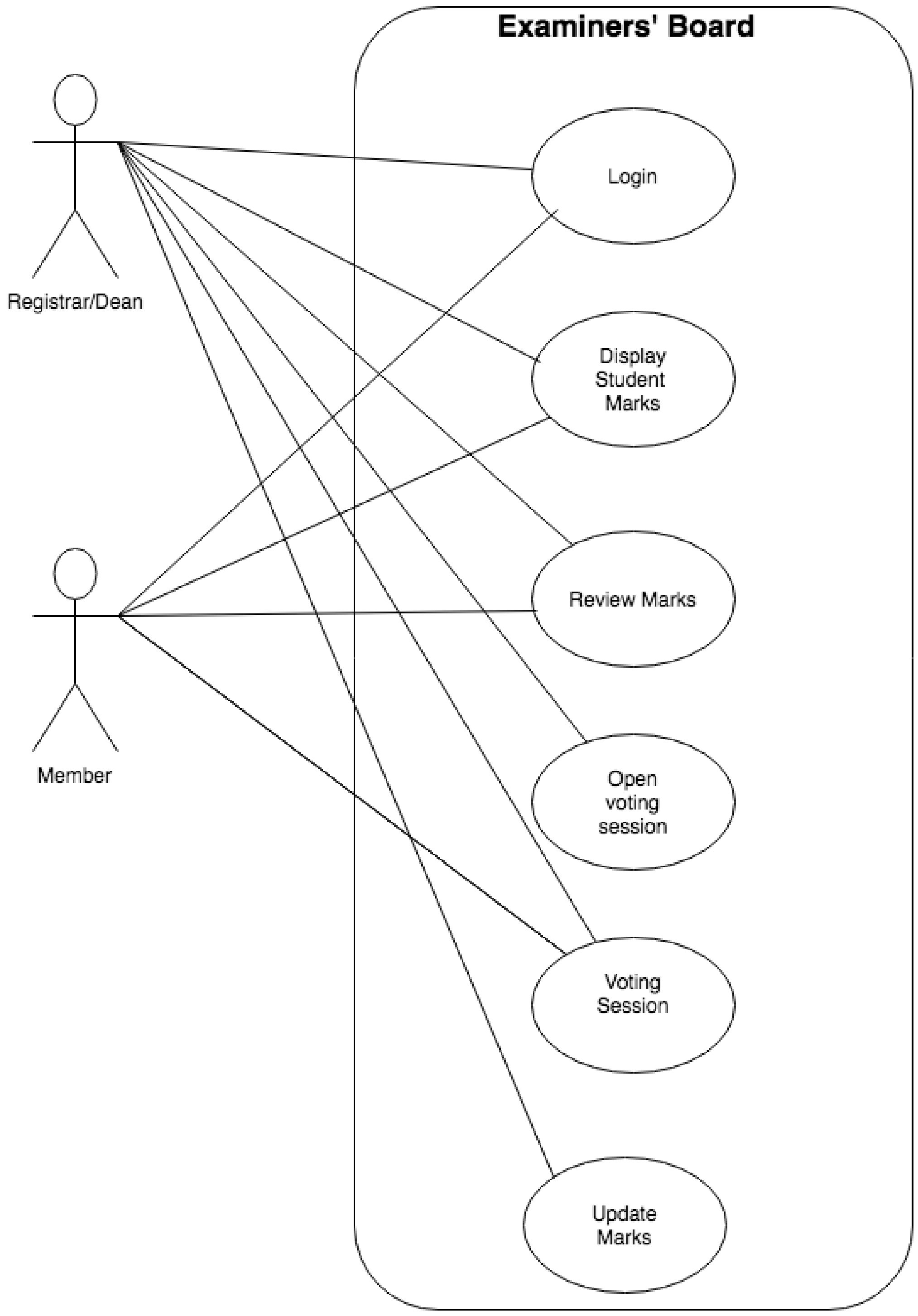
Course	Year Mark	Exam Mark	Final Mark	Outcome	Tut Attendance
Algebra	36	30	66	PASS	69
Calculus	50	46	45	FAIL	82
Mechanics	35	51	46	FAIL	67
Physical Science	50	51	52	PASS	82

Figure 4: Marks of student eligible for a vote

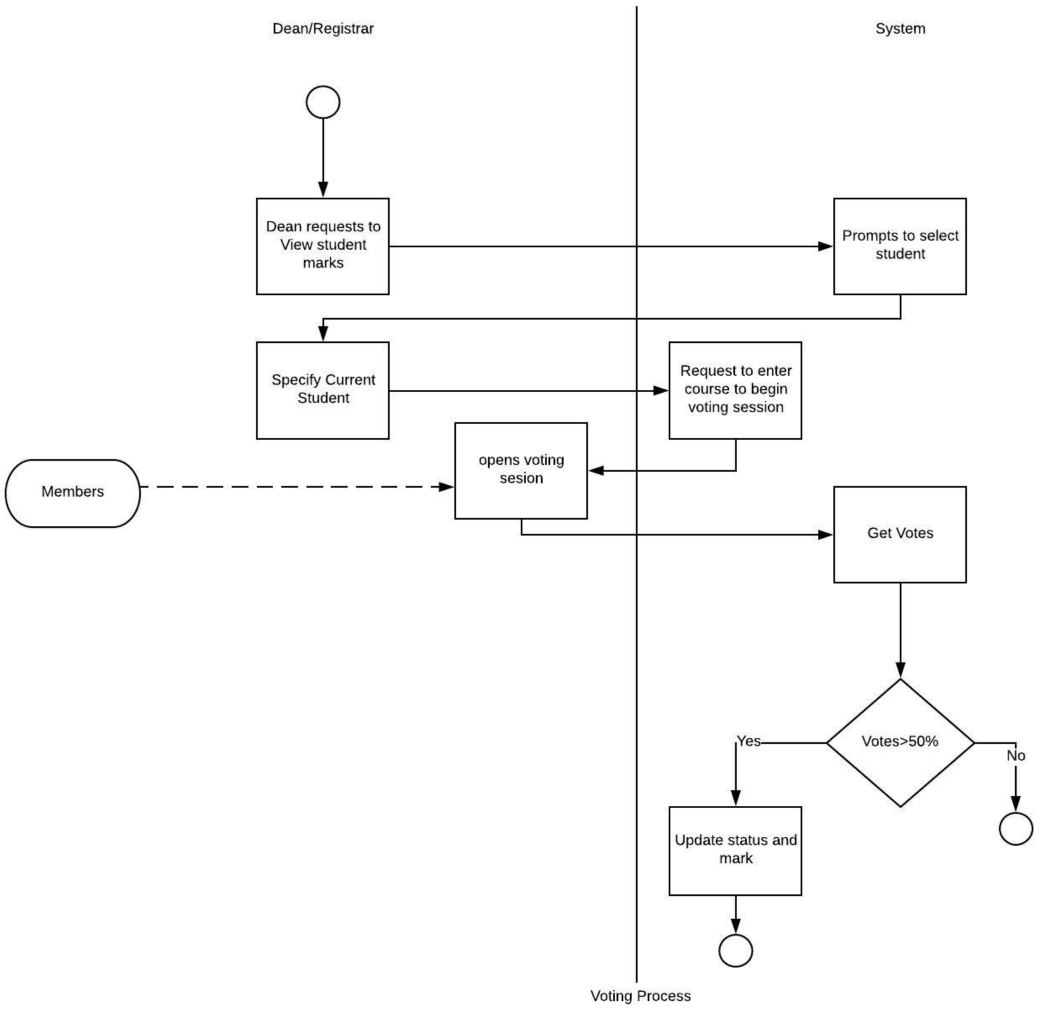
2.4.3 Core Use cases

- View all Students
- View Student marks
- Create Voting Session

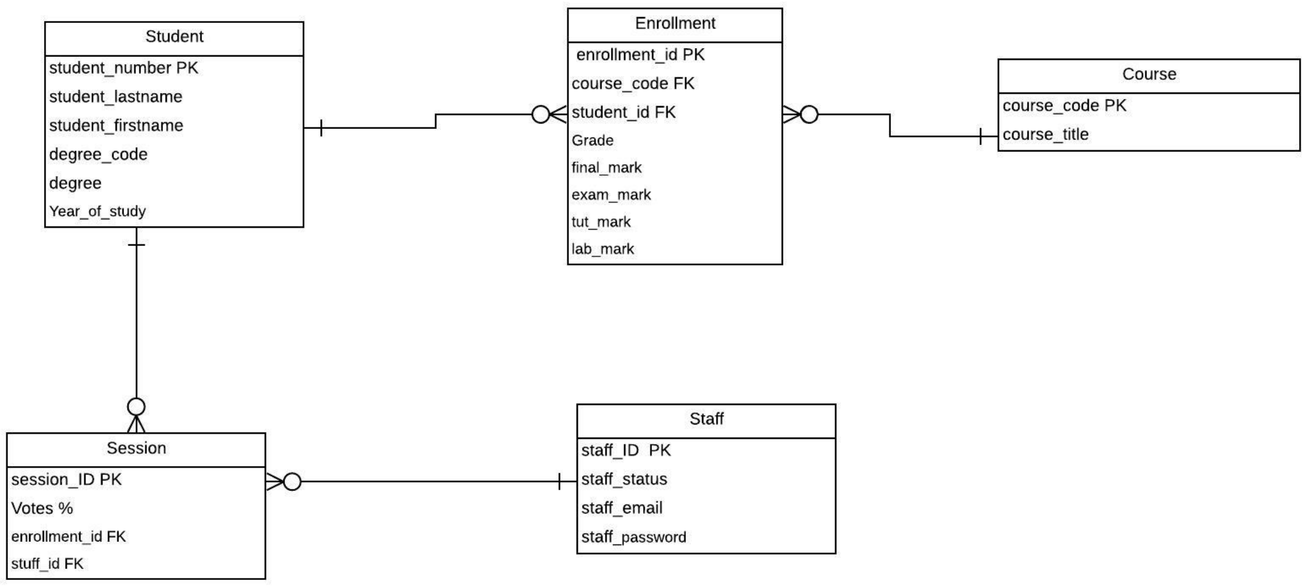
2.4.4 Use case model



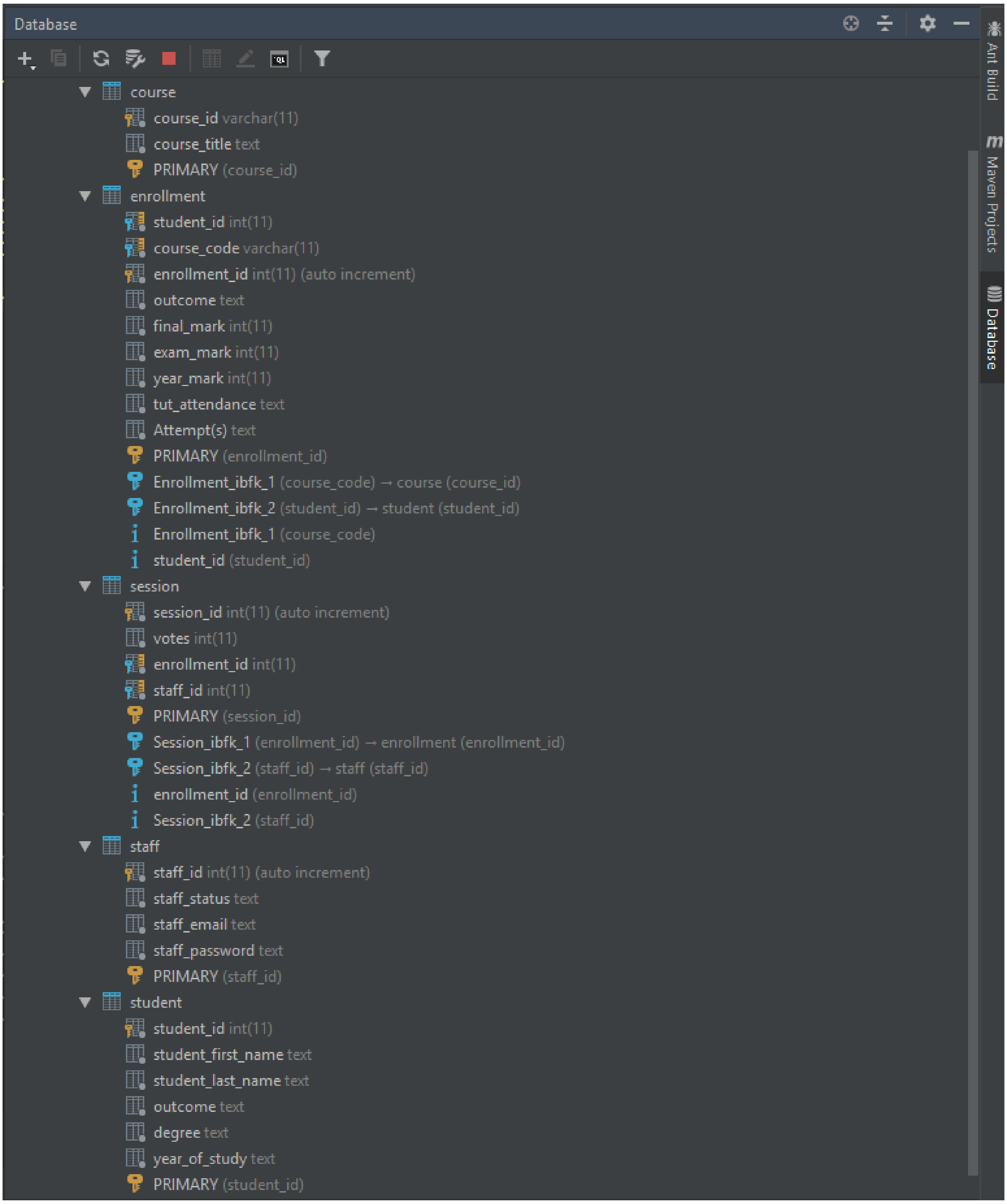
2.4.5 Process



2.4.6 Entity Relation Diagram



Entity Relationship Diagram: Examiner's board



```
course
    course_id varchar(11)
    course_title text
    PRIMARY (course_id)
enrollment
    student_id int(11)
    course_code varchar(11)
    enrollment_id int(11) (auto increment)
    outcome text
    final_mark int(11)
    exam_mark int(11)
    year_mark int(11)
    tut_attendance text
    Attempt(s) text
    PRIMARY (enrollment_id)
    Enrollment_ibfk_1 (course_code) → course (course_id)
    Enrollment_ibfk_2 (student_id) → student (student_id)
    Enrollment_ibfk_1 (course_code)
    student_id (student_id)
session
    session_id int(11) (auto increment)
    votes int(11)
    enrollment_id int(11)
    staff_id int(11)
    PRIMARY (session_id)
    Session_ibfk_1 (enrollment_id) → enrollment (enrollment_id)
    Session_ibfk_2 (staff_id) → staff (staff_id)
    enrollment_id (enrollment_id)
    Session_ibfk_2 (staff_id)
staff
    staff_id int(11) (auto increment)
    staff_status text
    staff_email text
    staff_password text
    PRIMARY (staff_id)
student
    student_id int(11)
    student_first_name text
    student_last_name text
    outcome text
    degree text
    year_of_study text
    PRIMARY (student_id)
```

Figure 5: Database Design

2.4.7 Memory Constraints

End user or Members of the committee need to have 500mb of free disk in their mobile devices.

2.4.8 Operations

- End user is able to login with their distinct login details if they are part of the committee.
- End user is able see the list and marks of all the students in the faculty.
- End user is able to vote for a student under scrutiny.

2.4.9 Site Adaptation Requirements

- If End user is using a mobile phone they need to be running on Android lollipop 5.1 and on an iOS with a safari version of 9.1.3
- If end user is using a laptop they should be running on windows 7(or above) , Ubuntu 14.04.3 LTS(or above) and iOS OS X 10.9 Mavericks(or above).

2.5 Product Functions

- Display a list of all registered students in the Science faculty.
- View all the courses in which each student is enrolled for, marks, class attendance, progress etc.
- Create a voting session for each individual course.
- Enable members to vote on progress outcome of each course.
- Count number of votes and decide on outcome.
- Update the student's status together with the mark.

2.6 User characteristics

- The system requires basic computer skills.
- Access is only limited to the members of the committee with their distinct login details.
- Members of the committee all need to have mobile devices in possession.
- It requires a working internet connection.

2.7 Constraints

The application will be web based and written in Java, JavaScript, HTML and CSS. Anyone willing to further develop it will have to know these technologies. And since it is web based, it will require internet access. The apps should be restricted to work on the Wits proxy only. Furthermore, the system should only allow members of the Science Faculty to access it.

2.8 Safety, Security and Reliability

Note that only committee members have access to the web-application with their distinct passwords created by them. Note also that the actual change of outcome will be done in the back round code just to avoid any form of human error or deliberate error. No one else has access to the database except for the registrar and since voting has no human influence, this makes the system very outcome very reliable and safe.

2.9 Assumptions and dependencies

The system assumes that the user has basic web-browser knowledge. The software is meant to be run locally on the clients designated hardware in which the design of the system is based. A working Java Run-time Environment version 1.8 and above is required.

3 Modules and Demonstration

3.1 Database.java

Databse()

```
 public Database() {

      try {
          Class.forName("com.mysql.jdbc.Driver");
          connection = DriverManager.getConnection("jdbc:mysql://db4free.net:3306/facultydatabase",
              "witsadmin", "witsadmin");
          statement = connection.createStatement();
      }

      catch(Exception exception) {
          System.out.println("Error:" + exception);
      }
  }

  }
}
```

getStudents()

```
    /*
     * Gets all the students in the database
     * @return an array list of students
     * @throws SQLException when it fails to execute a query
     */
     public ArrayList<Student> getStudents() {


        ArrayList<Student> students = new ArrayList<Student>();
        String query = "SELECT * FROM student";
        try{
            result = statement.executeQuery(query);

            while(result.next()) {

                String id = Integer.toString(result.getInt("student_id"));
                String firstName = result.getString("student_first_name");
                String lastName = result.getString("student_last_name");
                String degree = result.getString("degree");
                String yearOfStudy = result.getString("year_of_study");
                String outcome = result.getString("outcome");
                students.add(new Student(id, firstName, lastName, degree, yearOfStudy, outcome));

            }
        }
        catch (SQLException e){

        }

        return students;
    }
```

```
}
```

getCourseName()

```
    /**
     * Returns a name of a course corresponding to a given course id
     * @param course_id
     * @return course name given a course id
     * @throws SQLException
     */
    public String getCourseName(String course_id) throws SQLException {

        String courseName = null;
        String query = String.format("SELECT * FROM course WHERE course_id = '%s'", course_id);
        Statement statement = connection.createStatement();
        try {
            ResultSet result = statement.executeQuery(query);
            if(result.next()){
                courseName = result.getString("course_title");
            }
            return courseName;
        }
        catch (SQLException e) {
            return "No course for now";
        }

    }
```

getMarks(Student student)

```
    /**
     * Returns all the marks for a given student
     * @param student
     * @return an array list containing student Mark objects
     */
     public ArrayList<Marks> getMarks(Student student) {

        ArrayList<Marks> marks = new ArrayList<>();
        String query = String.format("SELECT * FROM enrollment WHERE student_id = '%s'",
            student.getId());

        try {

            result = statement.executeQuery(query);

            while(result.next()){
                String yearMark = result.getString("year_mark");
                String examMark = result.getString("exam_mark");
                String finalMark = result.getString("final_mark");
                String tutAttendance = result.getString("tut_attendance");
                String outcome = result.getString("outcome");
                String course_id = result.getString("course_code");
                String course = getCourseName(course_id);
                marks.add(new Marks(course, yearMark, examMark, finalMark, outcome, tutAttendance));
            }
            return marks;
        }
        catch (SQLException e){
            return marks;
        }
    }
}
```

validLogin(Student student)

```
    /**
     * Checks if the login details of a client are valid
     * @param username
     * @param password
     * @throws SQLException when we fail to execute a query
     * @return true if the username and password correspond to a record in the database, returns
         false otherwise
     */
     public boolean validLogin(String username, String password) {

        String query = String.format("SELECT * FROM staff WHERE staff_password = '%s' and staff_id =
            '%s'", password, username);
        try {
           preparedStatement = connection.prepareStatement(query);
           result = preparedStatement.executeQuery();
           if(result.next()){
              return true;
           }
           return false;
        }
        catch (SQLException e) {
           e.printStackTrace();
        }
        return false;
    }
}
```

3.2 MyUI.java

onCreate()

```
    /**
     * Displays the application's login page
     */
     private void onCreate() {

        loginPage.setResponsive(true);
        homePage.setResponsive(true);

        TextField usernameField = new TextField("Username");
        usernameField.setResponsive(true);
        usernameField.setIcon(VaadinIcons.USER);
        PasswordField passwordField = new PasswordField("Password");
        Button signInButton = new Button("Sign In");
        signInButton.setStyleName("primary");
        signInButton.setClickShortcut(ShortcutAction.KeyCode.ENTER);

        loginPage.addComponents(usernameField, passwordField, signInButton);
        loginPage.setComponentAlignment(usernameField, Alignment.TOP_CENTER);
        loginPage.setComponentAlignment(passwordField, Alignment.TOP_CENTER);
        loginPage.setComponentAlignment(signInButton, Alignment.TOP_CENTER);
        setContent(loginPage);

        homePage.logout.addClickListener(clickEvent -> {
           setContent(loginPage);
        });

        signInButton.addClickListener(clickEvent -> {

           String username = usernameField.getValue();
           String password = passwordField.getValue();

           if(!database.validLogin(username, password)){
              Notification.show("Invalid Login");
```

```
            }
            else{
                displayAllStudents();
            }
        });
    }

}
```

displayAllStudents()

```
    /**
    * Displays all the students in the database in a grid
    */
    private void displayAllStudents() {

        homePage.studentTable.removeAllComponents();
        setContent(homePage);
        studentList = database.getStudents();
        homePage.title.setValue("Examiner's Board Meeting");
        grid.removeAllColumns();

        grid.setResponsive(true);
        grid.setWidth("100%");
        grid.setHeight("100%");

        grid.setItems(studentList);

        grid.addColumn(Student::getId).setCaption("Student number");
        grid.addColumn(Student::getFirstName).setCaption("First name");
        grid.addColumn(Student::getlastName).setCaption("Last name");
        grid.addColumn(Student::getDegree).setCaption("Degree");
        grid.addColumn(Student::getYearOfStudy).setCaption("Year of Study");
        grid.addColumn(Student::getOutcome).setCaption("Outcome");
        homePage.studentTable.addComponent(grid);

        grid.addItemClickListener(itemClick -> {
            displayStudentsMarks(itemClick);
        });
    }

}
```

displayStudentsMarks()

```
/**
    * Displays a given student's in a grid when the student is clicked
    * @param itemClick
    */
   private void displayStudentsMarks(Grid.ItemClick<Student> itemClick) {

        homePage.studentTable.removeAllComponents();

        Student student = itemClick.getItem();
        ArrayList<Marks> marks = database.getMarks(student);

        marksContainer.removeAllColumns();
        marksContainer.setResponsive(true);
        marksContainer.setWidth("100%");

        if(marks.size() > 0){
            marksContainer.setHeightByRows(marks.size());
        }
        else {
            marksContainer.setHeightByRows(1);
        }
        marksContainer.setItems(marks);

        marksContainer.addColumn(Marks::getCourse).setCaption("Course");
        marksContainer.addColumn(Marks::getYearMark).setCaption("Year Mark");
        marksContainer.addColumn(Marks::getExamMark).setCaption("Exam Mark");
        marksContainer.addColumn(Marks::getFinalMark).setCaption("Final Mark");
        marksContainer.addColumn(Marks::getOutcome).setCaption("Outcome");
        marksContainer.addColumn(Marks::getTutAttendance).setCaption("Tut Attendance");

        Button voteButton = new Button("Vote");
        voteButton.setResponsive(true);
        voteButton.setStyleName("primary");

        Button backButton = new Button("Back");
        backButton.setIcon(VaadinIcons.ARROW_BACKWARD);
        backButton.setResponsive(true);
        backButton.setStyleName("primary");

        HorizontalLayout toolbar = new HorizontalLayout();
        toolbar.setResponsive(true);

        if(student.getOutcome().equals("PCD") || student.getOutcome().equals("Q") ||
            student.getOutcome().equals("***")){
            toolbar.addComponent(backButton);
        }
        else {
            toolbar.addComponents(backButton, voteButton);
        }

        homePage.studentTable.addComponents(toolbar, marksContainer);

        homePage.title.setValue(String.format("%s %s", student.getFirstName(), student.getlastName()));

        voteButton.addClickListener(clickEvent -> {
           voteSession(clickEvent, student);
        });

        backButton.addClickListener(clickEvent -> {
           displayAllStudents();
        });
    }


}
```

3.3 Marks.java

Marks(course, yearMark, examMark, finalMark, outcome, tutAttendance)

```
     /**
     * Instantiates a Mark object
     * @param course
     * @param yearMark
     * @param examMark
     * @param finalMark
     * @param outcome
     * @param tutAttendance
     */
    public Marks(String course, String yearMark, String examMark, String finalMark, String outcome,
        String tutAttendance){
        this.course = course;
        this.yearMark = yearMark;
        this.examMark = examMark;
        this.finalMark = finalMark;
        this.outcome = outcome;
        this.tutAttendance = tutAttendance;
    }
```

getCourse()

```
    /**
     * Retrieves the name of a course
     * @return course name
     */
    public String getCourse() {
        return course;
    }
```

getFinalMark()

```
   /**
     * Retrieves a student's final mark
     * @return final mark, weighted average of exam and year mark
     */
    public String getFinalMark() {
        return finalMark;
    }
```

getOutcome()

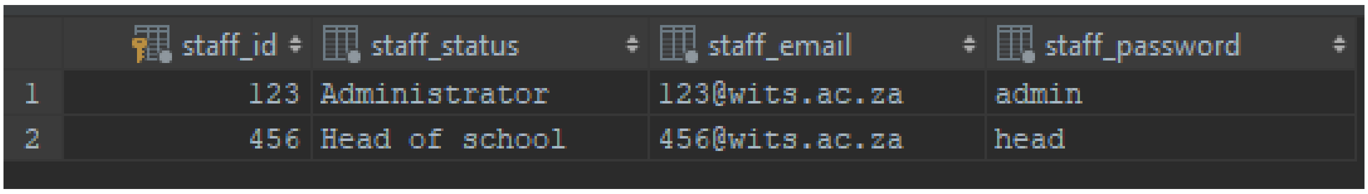
```
    /**
     * Retrieves a student's progress outcome for a given course
     * @return outcome, PASS if student's mark is 50 and above, FAIL otherwise
     */
    public String getOutcome() {
        return outcome;
    }
```

getTutAttendance()

```
  /**
     * Retrieves a student's tutorial attendance, to be used as voting factor
     * @return tutorial attendance as a percentage
     */
    public String getTutAttendance() {
        return tutAttendance;
    }
```

getYearMark()

```
    /**
     * Retrieves a student's year mark
     * @return year mark
     */
    public String getYearMark() {
        return yearMark;
    }
```



	staff_id	staff_status	staff_email	staff_password
1	123	Administrator	123@wits.ac.za	admin
2	456	Head of school	456@wits.ac.za	head

Figure 6: User Credentials

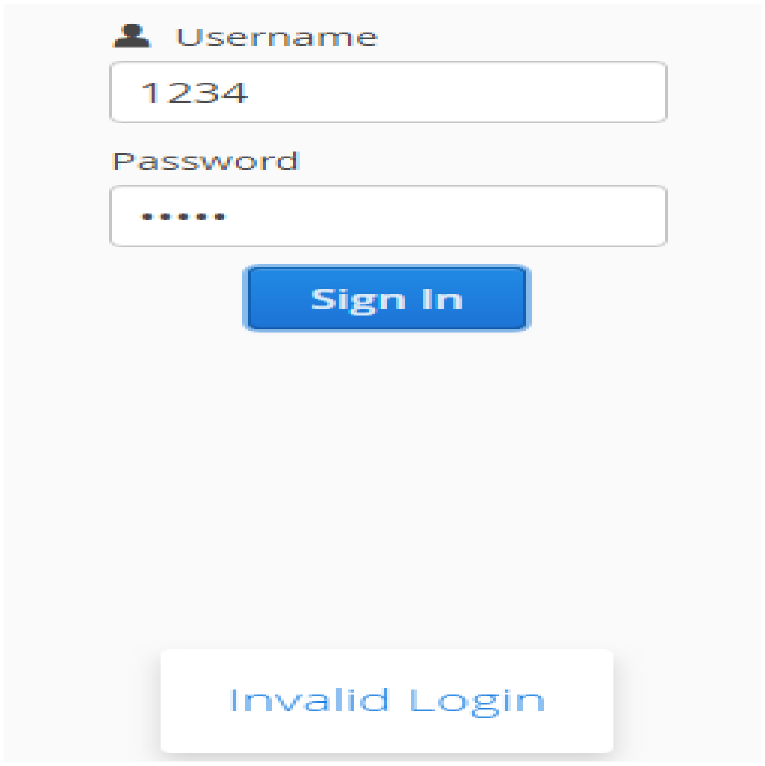


Figure 7: Failed Login

Demo 1

```
validLogin(123, "admin")
```

Will return true and the sign the user in successfully

Demo 2

```
validLogin(1234, "admin")
```

Will return false and display the screen in Figure 7

4 Sprint documentation

4.1 Purpose

This document Serves to test assumptions, address risks or deliver features. Ensures a focused Daily Scrum because the Development Team can use it to inspect their progress. Provides guidance to the Development Team on why it is building the Increment. Offers flexibility regarding the functionality implemented within the Sprint. Helps setting priorities when "the going gets tough".Fosters teamwork and team-building by jointly working towards a shared Sprint Goal. Supports the Product Owner in creating the product roadmap.Stimulates Product Backlog cohesion when planning a release.Can be used as an instrument for stakeholder management.Supports a focused Sprint Planning by crafting a shared Sprint Goal.Enables efficient decision-making

4.2 Sprint 1

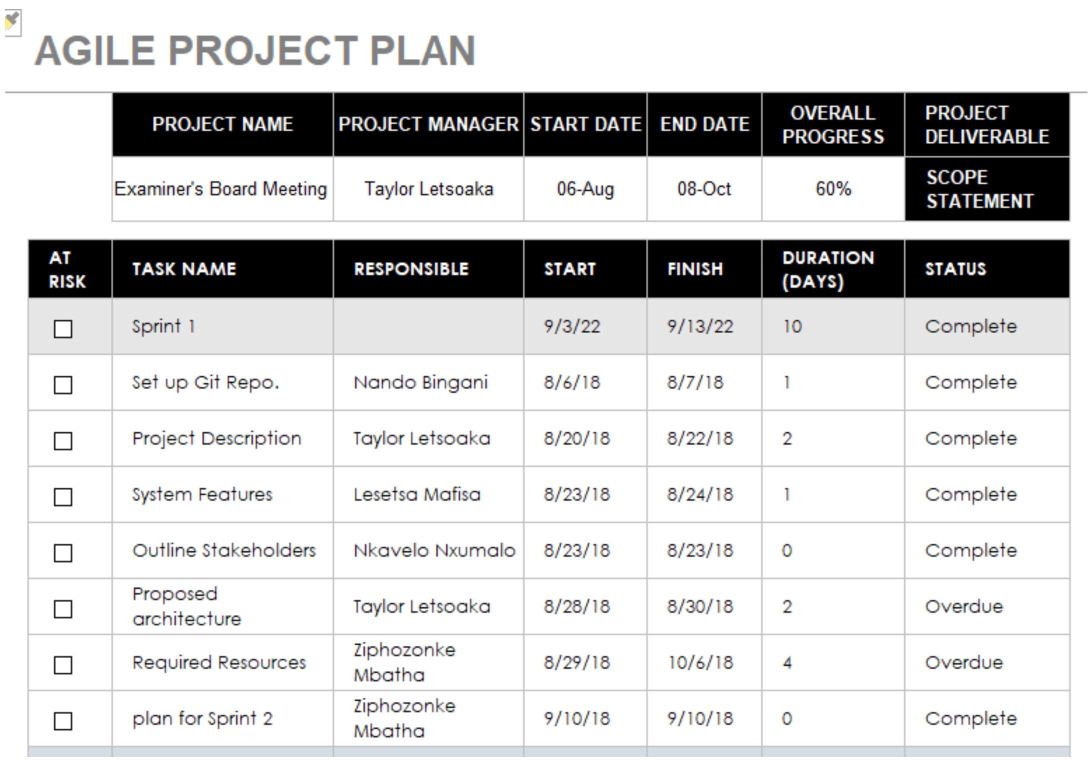
Aim of the Sprint : The main point of this phase is to establish a good understanding of the business case/problem and to then determine the scope and feasibility of the system that will address that business case/problem. team will need to submit SRS document for the required project. This sprint is responsible for setting up the working base station the git Repository and outline the plan for the next sprint.

Objectives the SRS Document

- A problem description of what is to be build
- Systems' features
- Perceived users and their roles
- Technologies and proposed architecture
- Some projected required resources

Plan for the next Sprint

Refine documents from the first sprint start of with the system implementation and team sign



AGILE PROJECT PLAN

PROJECT NAME	PROJECT MANAGER	START DATE	END DATE	OVERALL PROGRESS	PROJECT DELIVERABLE
Examiner's Board Meeting	Taylor Letsoaka	06-Aug	08-Oct	60%	SCOPE STATEMENT

AT RISK	TASK NAME	RESPONSIBLE	START	FINISH	DURATION (DAYS)	STATUS
☐	Sprint 1		9/3/22	9/13/22	10	Complete
☐	Set up Git Repo.	Nando Bingani	8/6/18	8/7/18	1	Complete
☐	Project Description	Taylor Letsoaka	8/20/18	8/22/18	2	Complete
☐	System Features	Lesetsa Mafisa	8/23/18	8/24/18	1	Complete
☐	Outline Stakeholders	Nkavelo Nxumalo	8/23/18	8/23/18	0	Complete
☐	Proposed architecture	Taylor Letsoaka	8/28/18	8/30/18	2	Overdue
☐	Required Resources	Ziphozonke Mbatha	8/29/18	10/6/18	4	Overdue
☐	plan for Sprint 2	Ziphozonke Mbatha	9/10/18	9/10/18	0	Complete

offs. 1.PNG
Sprint 1

4.3 Sprint 2

Aim of Sprint 2: The point of this phase is to elaborate on what was started during the sprint 1.Design and domain models need to be crafted and the use cases that are being fully dressed need to be implemented into a working version of the system. Consistency across the models, documentation, and system is an important element to consider during this iteration. showcase the functionality in the system and outlining the plan for the next sprint.

System Prototype and Implementation

The functionality of the system must be exhibited while the analysis models for that aspect of functionality. The initialization of the database, database server, designing the interface and implementing the basic modules.

Testing

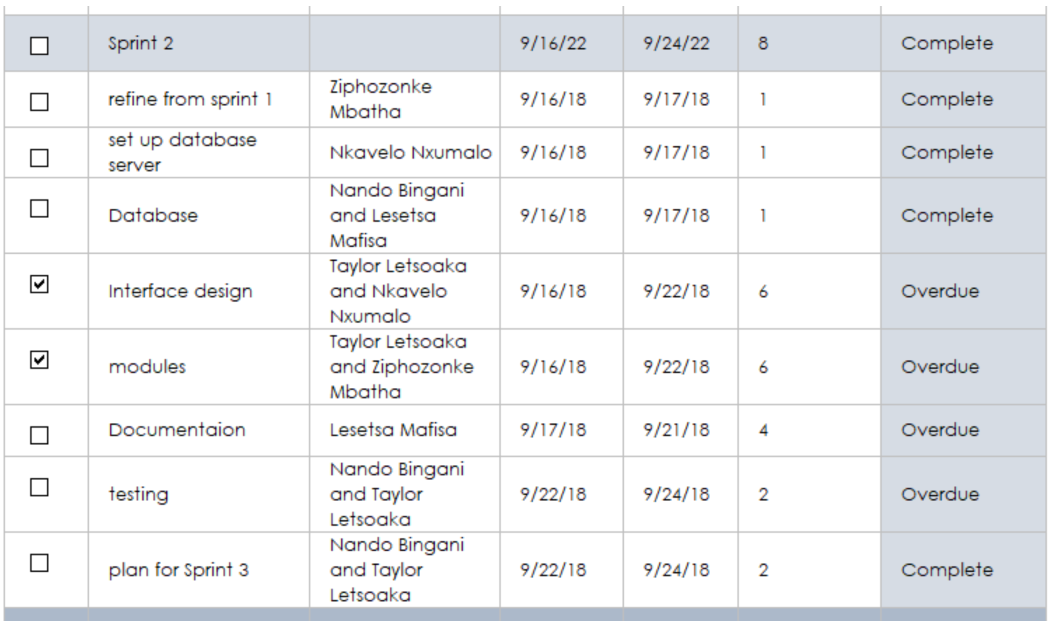
testing relevant modules, unit testing.()..interface testing etc

SRS Document

SRS Documents should be refined based off feedback from sprint 1 and continued requirements gathering and team sign offs.

Plan for the next Sprint

Fix errors from the first sprint, continue with the system implementation.



☐	Sprint 2		9/16/22	9/24/22	8	Complete
☐	refine from sprint 1	Ziphozonke Mbatha	9/16/18	9/17/18	1	Complete
☐	set up database server	Nkavelo Nxumalo	9/16/18	9/17/18	1	Complete
☐	Database	Nando Bingani and Lesetsa Mafisa	9/16/18	9/17/18	1	Complete
☑	Interface design	Taylor Letsoaka and Nkavelo Nxumalo	9/16/18	9/22/18	6	Overdue
☑	modules	Taylor Letsoaka and Ziphozonke Mbatha	9/16/18	9/22/18	6	Overdue
☐	Documentaion	Lesetsa Mafisa	9/17/18	9/21/18	4	Overdue
☐	testing	Nando Bingani and Taylor Letsoaka	9/22/18	9/24/18	2	Overdue
☐	plan for Sprint 3	Nando Bingani and Taylor Letsoaka	9/22/18	9/24/18	2	Complete

2.PNG
Sprint 2

4.4 Sprint 3

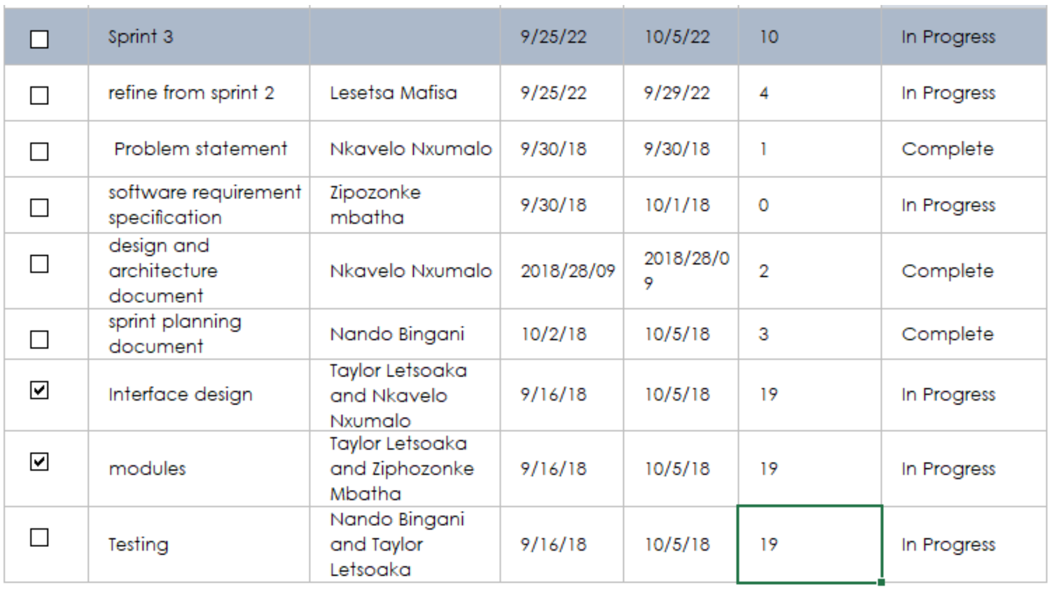
Aim of Sprint 3: Sprint 3 is the final aspect of the system development project. The completed systems will be handed in. Team is expected to create a professional looking project. This should include Documents covering: The problem statement/business problem how the solution addresses the business problem, system functionality,software requirement specication, design and architecture document sprint There should also be a focus on the underlying business process that the systems support and links between that process and the use cases of the system.

Final Testing

Final testing of the relevant modules, unit testing.()..interface testing etc

Final Documentation and System

Finalized SRS Documents should be produced based off feedback from sprint 2 and requirements gathering.The system will be assessed based on the successful implementation of all of the use cases and feature functionality, the design of the system, and the reporting functionality. The system should have validation implemented, appropriate navigation and error messages, and all use case, feature, and reporting functionality should be working. final team sign offs should be performed.



☐	Sprint 3		9/25/22	10/5/22	10	In Progress
☐	refine from sprint 2	Lesetsa Mafisa	9/25/22	9/29/22	4	In Progress
☐	Problem statement	Nkavelo Nxumalo	9/30/18	9/30/18	1	Complete
☐	software requirement specification	Zipozonke mbatha	9/30/18	10/1/18	0	In Progress
☐	design and architecture document	Nkavelo Nxumalo	2018/28/09	2018/28/0 9	2	Complete
☐	sprint planning document	Nando Bingani	10/2/18	10/5/18	3	Complete
☑	Interface design	Taylor Letsoaka and Nkavelo Nxumalo	9/16/18	10/5/18	19	In Progress
☑	modules	Taylor Letsoaka and Ziphozonke Mbatha	9/16/18	10/5/18	19	In Progress
☐	Testing	Nando Bingani and Taylor Letsoaka	9/16/18	10/5/18	19	In Progress

3.PNG

Sprint 3

4.5 Declaration

We declare that this Project Report was written and compiled by all group members in our own words, according to our own knowledge. We used references to aid in our understanding and research. We also declare that we have followed the rules which are against copying or falsifying data from other sources of information.