Name and Surname: Achuma Madubula

Student Number: ST10455989

Module: Programming 2B Part 1

Lecture: Mr. Mohale

Due Date: 09 September

1. Documentation

1.1 Review of System

Designed to simplify the process of monitoring and approving work completed by staff on several projects and contracts, the Contract Monthly Claim System is an intranet web application. Replacing a manual, paper-based or spreadsheet-based claiming method will help to minimize errors, increase openness, and speed the approval process.

Core features are:

Employees recording their monthly time spent on several contracts.

Managers looking over, accepting, or rejecting these time claims.

The arrangement keeping a safe and verifiable log of all transactions.

1.2 Assumptions and Constraints in Core Design These assumptions and limitations have been set up to direct the creation of the prototype:

A. User Roles and Authentication:

Assumption: There will be two main user roles with various levels of authority in the system:

Employee: Create, modify, delete (if not submitted), and send their own monthly claims. Can view the status of their past claims.

Manager: Inheritance of all employee skills Furthermore, can see approved or declined claims made by staff members they supervise.

Constraint: Any activity requires a user to be registered and logged in into the system. User registration won't be handled by the first prototype; rather, users will be pre-provisioned in the database.

B: Claim Lifecycle and Business Rules

Assumption 1 (Claim Structure): One "Claim" covers all of an employee's work for one calendar month (e.g., April 2024). This claim will include several "Claim Items," which each describe the hours spent on a particular contract throughout the month.

* Assumption 2 (Status Workflow): A claim will go across a predetermined status lifespan:
* Draft: The employee is still working on it. It is available to be freely changed.
* Submitted: The employee has sent it for approval after finishing it. The employee is unable to edit it anymore.
* Approved: The manager has accepted the claim. Locked now, it is added to the official documentation
* Rejected: The management has sent the claim back to the employee for adjustments. The employee can revise and re submit since the status returns to Draft.
* An employee can only make one claim each month; this is a limitation. If one already exists (in any status), they cannot make another claim for April 2024.

C. Data Structure and Relationships:

* Assumption: The data model will need connections between claims, contracts, and users. One manager, who is also an employee, runs an employee. One claim belongs to one employee and can be approved by one manager. One contract is connected to several items in a claim.

* Constraint: For the prototype, historical data integrity will be given first priority. This implies that any past claims that utilized a Contract labelled IsActive = false in the future should still be linked to it. It should not be removed from the database.

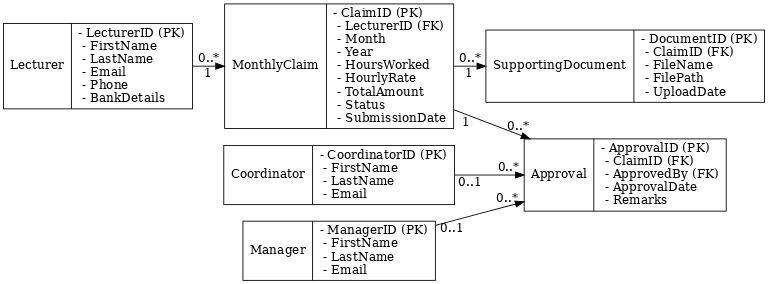
D. Technical Restrictions

* The ASP.NET Core MVC framework will be used to create the application since it offers a perfect separation of concerns (Model-View-Controller) that matches the demands of the project.
* To guarantee a neat, responsive, and professional-looking user interface prototype, the frontend will use Bootstrap styling.

* The non-functional prototype means:

* Buttons and links will not carry out actual functions; for instance, the "Submit" button will not alter the claim's status in a database.
* All hard-coded, placeholder information shown on displays will be data.
* To show the planned user flow, navigational between pages will be practical.

Step 2: UML Diagram



3. Project Plan

1. **Overview of the project**

* Project Title: Development of a Prototype for a Contract Monthly Claim System
* The goal is to create a non-functional system prototype that includes a clickable user interface mock-up, project plan, and UML design.

Important Deliverables:

1. Diagram of the UML Class
2. Project Schedule
3. ASP.NET Core MVC User Interface Prototype (Non-functional)

* One developer (you) makes up the team.
* Duration: Three Weeks

#### **2. Work Breakdown Structure (Tasks)**

This breaks the project into manageable phases and tasks.

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Phase | Task ID | Task Name | Description | Dependencies | Estimated Effort (Hours) |
| **Phase 1:** Planning & Design | 1.1 | Requirements Analysis | Define system scope, user roles, and assumptions. | None | 3 |
|  | 1.2 | Database Design | Create a detailed UML Class Diagram. | 1.1 | 4 |
|  | 1.3 | Project Planning | Develop this project plan and timeline. | 1.1 | 2 |
|  | 1.4 | UI Wireframing | Sketch the layout and flow of key pages. | 1.1 | 3 |
| **Phase 2:** Environment Setup | 2.1 | Tool Setup | Install and configure .NET Core SDK, IDE (Visual Studio), and Git. | None | 2 |
|  | 2.2 | Project Creation | Create a new [ASP.NET](https://asp.net/) Core MVC project solution. | 2.1 | 1 |
| **Phase 3:** UI Prototype Development | 3.1 | Implement Layout | Create a consistent layout (e.g., using Bootstrap). | 2.2 | 3 |
|  | 3.2 | Build Authentication Views | Create Login and Registration page mockups. | 3.1 | 2 |
|  | 3.3 | Build Dashboard Views | Create role-specific dashboards (Employee & Manager). | 3.1, 1.4 | 4 |
|  | 3.4 | Build Claim CRUD Views | Create pages to Create, View, and List claims. | 3.1, 1.4 | 5 |
|  | 3.5 | Build Approval Views | Create the page for managers to approve/reject claims. | 3.1, 1.4 | 3 |
| **Phase 4:** Review & Submission | 4.1 | Internal Review | Test the UI flow and check all requirements are met. | All Phase 3 tasks | 2 |
|  | 4.2 | Final Documentation | Compile all deliverables into a final report. | All tasks | 3 |
|  | **Total Estimated Effort** |  |  |  | **37 Hours** |

3. Gantt chart or project timeline

This displays the tasks' schedule for the next three weeks. This is simple to draw in Google Sheets or Excel.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Task** | **Week 1** | **Week 2** | **Week 3** | **Status** |
| 1.1 Requirements Analysis | █████ |  |  | Complete |
| 1.2 UML Diagram | █████ |  |  | Complete |
| 1.3 Project Plan | ███ |  |  | In Progress |
| 1.4 UI Wireframes | █████ |  |  | Complete |
| 2.1 Environment Setup |  | ██ |  | Not Started |
| 2.2 Project Creation |  | █ |  | Not Started |
| 3.1 Implement Layout |  | ███ |  | Not Started |
| 3.2 Auth Views |  | ██ |  | Not Started |
| 3.3 Dashboard Views |  | ████ |  | Not Started |
| 3.4 Claim Views |  |  | █████ | Not Started |
| 3.5 Approval Views |  |  | ███ | Not Started |
| 4.1 Internal Review |  |  | ██ | Not Started |
| 4.2 Final Documentation |  |  | ███ | Not Started |
| \*Legend: █ = 1 day of work\* |  |  |  |  |
|  |  |  |  |  |

4. Important Requirements

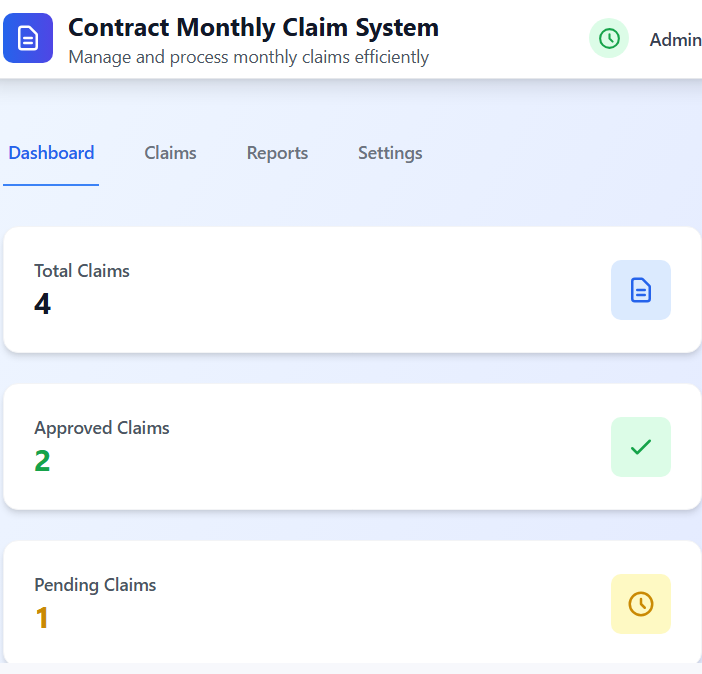
* Because the UI must represent the data model, the completion of the UML Diagram (1.2) is a prerequisite for the UI Prototype (3.3, 3.4, 3.5).
* The completion of Project Setup (2.2) is a prerequisite for all Phase 3 tasks.
* All other tasks must be finished before beginning Final Documentation (4.2).

#### **5. Risk Assessment**

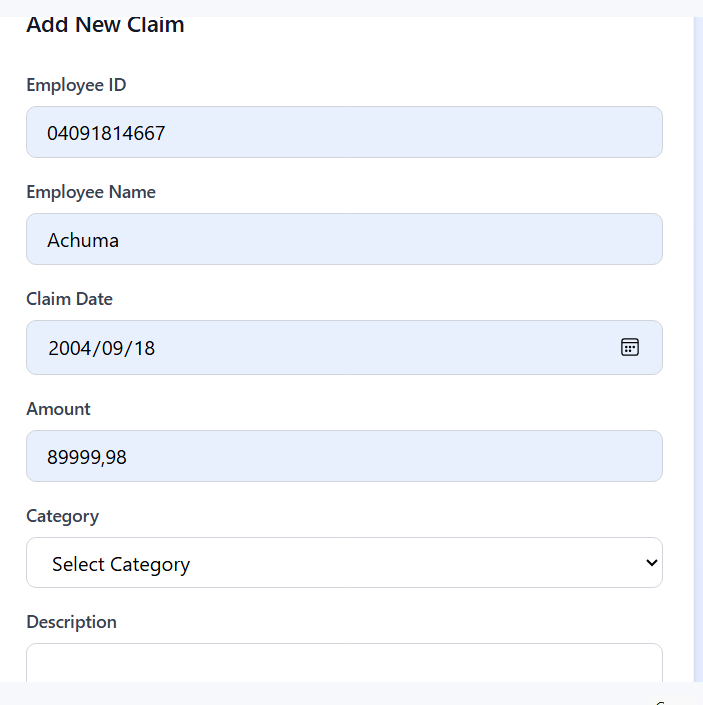
|  |  |  |  |
| --- | --- | --- | --- |
| Risk | Probability | Impact | Mitigation Strategy |
| **Scope Creep** | Medium | High | Stick to the core requirements for the prototype. Define "nice-to-have" features for a future version. |
| **Time Management** | High | High | Break tasks into small, manageable chunks. Use this project plan to track progress. |
| **Technical Challenges** | Medium | Medium | Focus on creating a non-functional UI first. If stuck, use online resources (documentation, tutorials) for [ASP.NET](https://asp.net/) Core. |

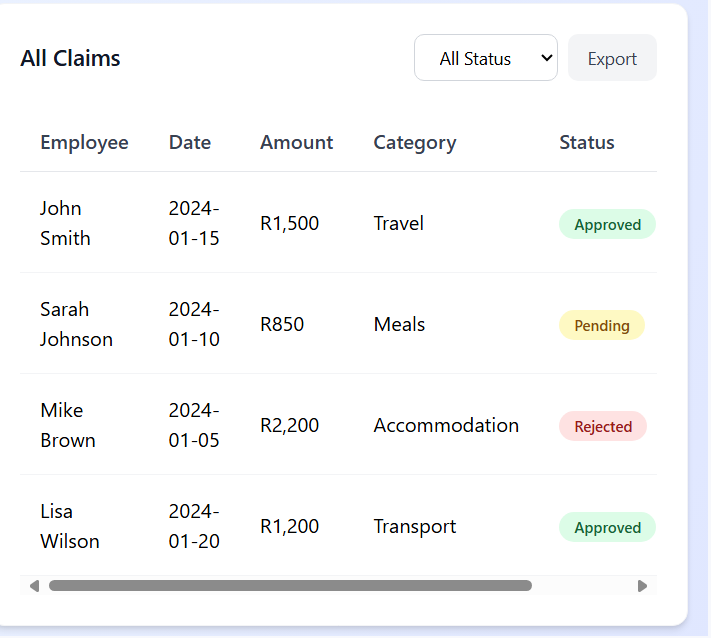
One developer is supposed to finish the project plan in three weeks. To guarantee that the design (Phase 1) is finished before any development (Phase 3) starts, the work is divided into four consecutive phases. This reduces the possibility of rework. The project plan itself, the UML diagram, and the UI mock-ups are the three main prototypes that will be delivered during the estimated 37-hour effort.

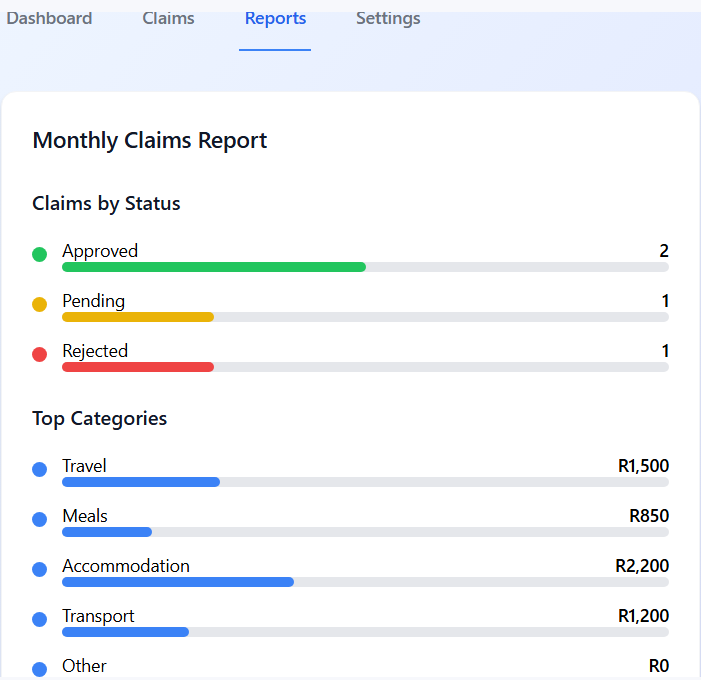
1. GUI Prototype

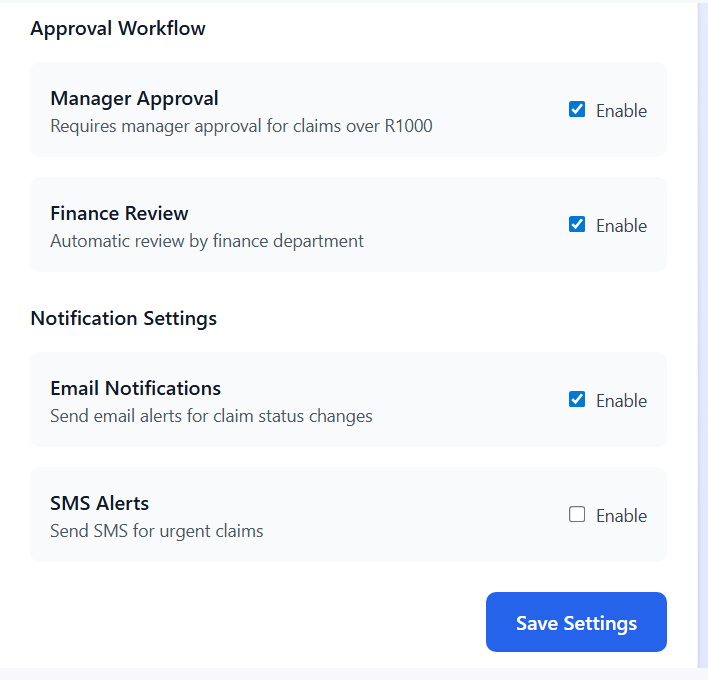


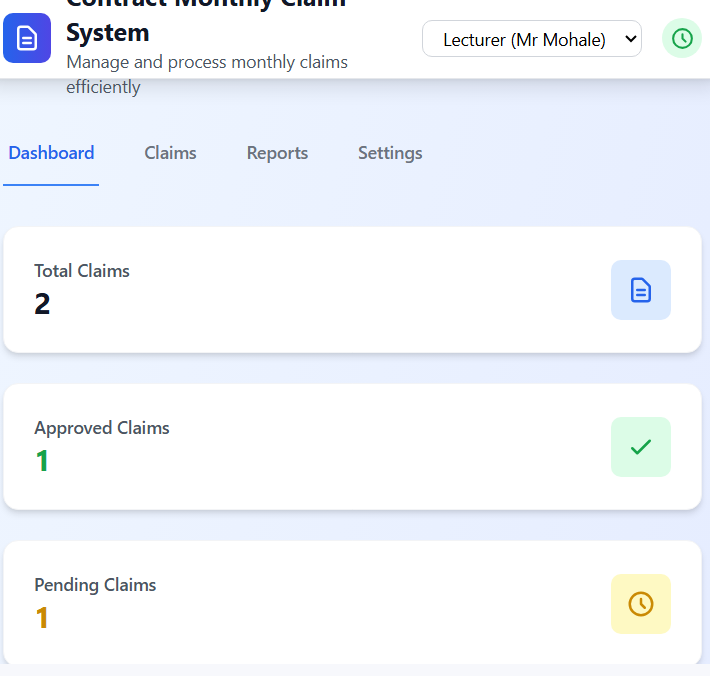


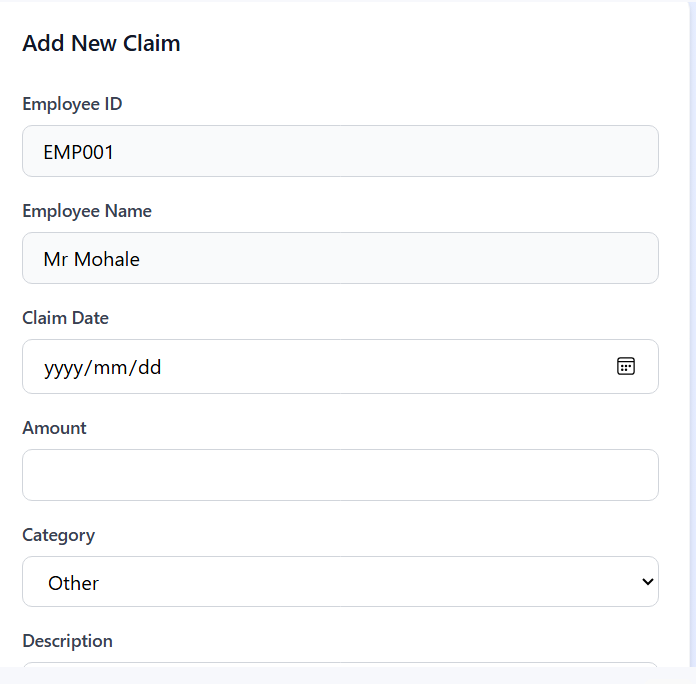
Claims Page 

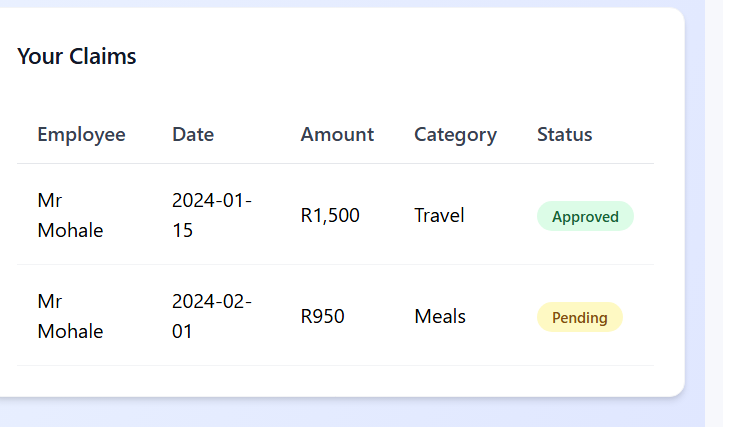


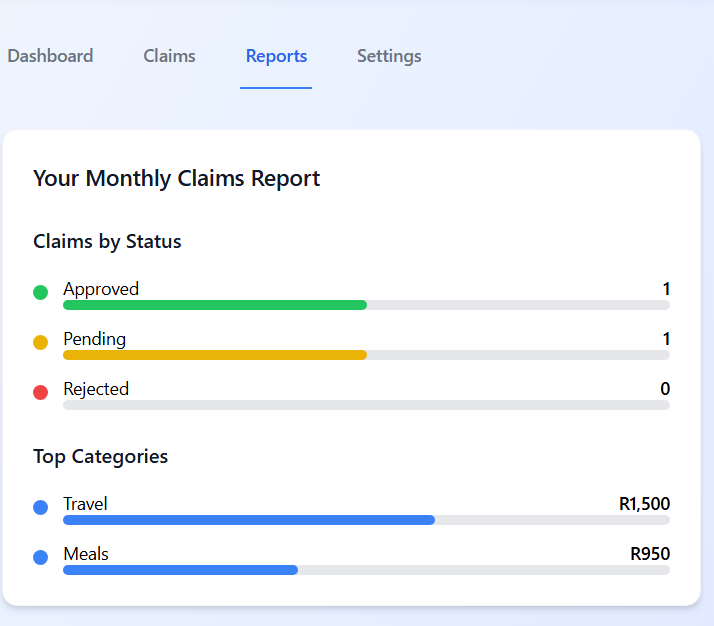


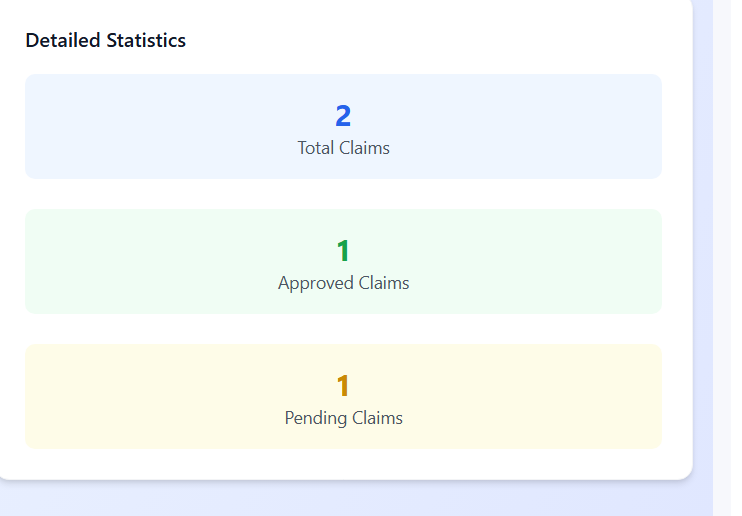
System Settings Page

Lecture View

Lecture claim page 



Report



REFERENCES

**1. For UML & Software Design Concepts:**  
Fowler, M. (2004) *UML distilled: a brief guide to the standard object modeling language*. 3rd edn. Boston: Addison-Wesley.

Larman, C. (2005) *Applying UML and patterns: an introduction to object-oriented analysis and design and iterative development*. 3rd edn. Upper Saddle River, N.J.: Prentice Hall PTR.

**2. For Project Management & Planning:**  
Project Management Institute (2017) *A guide to the project management body of knowledge (PMBOK guide)*. 6th edn. Newtown Square, PA: Project Management Institute.

**3. For** [**ASP.NET**](https://asp.net/) **Core MVC Development:**  
Microsoft (2024) *Overview of* [*ASP.NET*](https://asp.net/) *Core MVC*. Available at: <https://learn.microsoft.com/en-us/aspnet/core/mvc/overview?view=aspnetcore-8.0> (Accessed: 22 May 2024).

**4. For Bootstrap Framework (UI Styling):**  
Otto, M. and Thornton, J. (2024) *Bootstrap*. Available at: <https://getbootstrap.com/> (Accessed: 22 May 2024).**5. For the PlantUML Tool (Diagram Generation):**  
PlantUML (2024) *PlantUML*. Available at: <https://plantuml.com/> (Accessed: 22 May 2024).

+

ambler, S.W., and Sadalage, P.J. (2006) *Refactoring databases: evolutionary database design*. Boston: Addison-Wesley.

Beck, K. (2000) *Extreme programming explained: embrace change*. Reading, Mass.: Addison-Wesley.

Gamma, E., Helm, R., Johnson, R. and Vlissides, J. (1994) *Design patterns: elements of reusable object-oriented software*. Reading, Mass.: Addison-Wesley.

Jacobson, I., Booch, G. and Rumbaugh, J. (1999) *The unified software development process*. Reading, Mass.: Addison-Wesley.

McConnell, S. (2004) *Code complete*. 2nd edn. Redmond, Wash.: Microsoft Press.