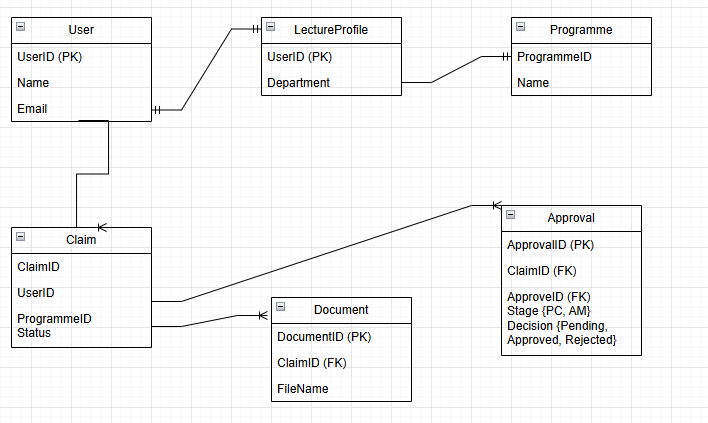
1. **Contract Monthly Claim System Prototype**

**Documentation**

The Contract Monthly Claim System is designed to streamline the process of submitting, verifying, and approving monthly claims for lectures. The prototype models a monthly claims workflow for part-time lecturers. The GUI provides role-specific views: lecturers can submit and track claims, while coordinators and managers can verify, approve, or reject them. The system will provide a user-friendly interface for lectures to submit claims, upload supporting documents, and track the status for their claims. Key pages include a dashboard, claim submission, verification, and tracking, designed to be user-friendly, consistent, and transparent. Program Coordinators and Academic Managers will be able to easily verify and approve claims, ensuring a transparent and efficient process. While authentication and file handling are simulated, the design is scalable for future backend integration within a controlled environment. The system will be web-based, with a responsive design to accommodate different screen sizes and devices. We keep users in a single User table with a role enumeration to simplify authentication for the prototype. Constraints: no payment processing in Part 1, and the GUI is a non-functional mock built in .NET Core (WPF or MVC). The GUI is designed to be scalable for future integration with backend logic. File uploads are stimulated; no actual file handling is implemented. The system will incorporate a robust security mechanism to ensure data integrity and confidentiality. The system will operate within controlled environment with predefined roles and responsibilities. The system will work on the following Part of the assignment (Vlok, 2015).

1. **UML Diagram**

The UML class diagram below represents the data requirements of the Contract Monthly Claim System

**** (Date, 2019)

1. **Project Plan**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Phase** | **Tasks** | **Timeline** | **Dependencies** | **Deliverables** |
| **1. Project Initialization** | Define scope, assemble team, gather requirements, initial documentation | 2 days | None | Project charter, requirements document |
| **2. Research & Design** | Conduct research, create wireframes, stakeholder review | 3 days | Completion of Phase 1 | Design mockups, research report |
| **3. Development Setup** | Setup environment, define architecture | 4 days | Approved design specifications | Development environment, code repository |
| **4. Prototype Development** | Implement core features, iterative testing, documentation | 10 days | Completion of setup | Working prototype with core features |
| **5. Testing & Feedback** | UAT, gather feedback, fix bugs | 5 days | Completed prototype | Tested prototype, feedback report |
| **6. Final Adjustments & Docs** | Final enhancements, prepare documentation, plan future steps | 7 days | Feedback incorporation | Final prototype, documentation |
| **7. Project Closure** | Review, archive, transition to next phase | 5 days | Final adjustments | Closure report, archived files |

(Zuckerberg, 2015)

**4.** <https://github.com/AvuyileNdongeni/ContractMonthlyClaim.git>

# References

Date, C., 2019. *An Introduction to Database Systems.* 8th ed. Boston: Pearson Education.

Vlok, C., 2015. *Solutions for all.* Cape Town: University Press.

Zuckerberg, M., 2015. *Artificial Intelligence.* [Online]   
Available at: http://www.ai.meta.com  
[Accessed 11 December 2015].