**BBM 354: SOFTWARE DEVELOPMENT**

**Prerequisite:**

**Units : 3**

**Hours : 42**

**Course Purpose**
Facilitate learners with a comprehensive understanding of software development principles, methodologies, and practices through theoretical instruction and hands-on practical experience.

**Course Objective**
The Course will facilitate learning about:
1. concepts and theories of software development principles, methodologies, and best practices
2. practical skills in software design, implementation, testing, and maintenance, utilizing industry-standard tools and techniques.
3. problem-solving abilities and critical thinking skills for analyzing complex software problems and developing innovative solutions.
4. effective communication, collaboration, and teamwork skills in development environments.

**Expected Learning Outcomes:**
By the end of the course, students will be able to:
1. Demonstrate a solid understanding of software development principles, methodologies, and best practices.
2. Design, implement, test, and maintain software solutions using appropriate programming languages, frameworks, and tools.
3. Analyze and solve complex software problems through systematic problem-solving approaches and algorithmic thinking.
4. Communicate effectively with team members, stakeholders,

**Course Content**
Introduction to Software Development, Software development lifecycle, Programming languages and paradigms, Software engineering principles, Programming Fundamentals; Basics of programming languages, Control structures and data types, Functions and modular programming, Object-Oriented Programming (OOP); Principles of OOP, Classes, objects, and inheritance, Polymorphism and encapsulation, Software Design and Architecture; Design principles and patterns, Architectural styles and design considerations, UML diagrams and design documentation, Software Development Tools and Environments; Integrated Development Environments (IDEs), Version control systems (e.g., Git), Collaboration tools and project management software, Testing and Quality Assurance; Testing strategies and methodologies, Unit testing, integration testing, and test-driven development, Quality assurance techniques and best practices, Software Maintenance and Refactoring; Code refactoring techniques, Code reviews and software maintenance practices, Legacy system migration and modernization, Software Project Management; Project planning, estimation, and scheduling, Agile methodologies (e.g., Scrum, Kanban), Team collaboration and communication, Ethical and Legal Considerations in Software

Development; Intellectual property rights and licensing, Ethical implications of software design and development, Privacy and security considerations, Emerging Trends in Software Development; Cloud computing and microservices architecture, Big data analytics and machine learning, Internet of Things (IoT) and embedded systems development.

**Instructional Methods**
Interactive lectures and Presentations, Hands-on Labs, and Coding Exercises, tutorials, group discussions, experiential learning, e-learning and case method.

**Instructional Materials and Equipment**
Textbooks, Journals, writing boards, LCD, computers with development tools and internet access and online resources

**Assessment**

| **Type** | **Weighting (%)** |
|---|---|
| Continuous Assessment Tests & Tasks | 40 |
| End of Semester Examination/Assessment | 60 |
| **TOTAL** | **100** |

**References:**

**Core Text**

1. Martin, R. C. (2019). *Clean Architecture: A Craftsman's Guide to Software Structure and Design*. Prentice Hall.
2. Burns, P. (2020). *Programming Rust: Fast, Safe Systems Development* (2nd ed.). O'Reilly Media.
3. Adzic, G., & Evans, D. (2020). *Impact Mapping: Making a Big Impact with Software Products and Projects*. Leanpub.


**Other recommended Texts**

1. Ford, N., Parsons, R., & Kua, P. (2020). *Building Evolutionary Architectures: Support Constant Change* (2nd ed.). O'Reilly Media.
2. Metz, S. (2019). *Practical Object-Oriented Design: An Agile Primer Using Ruby* (2nd ed.). Addison-Wesley Professional.
3. Martin, R. C. (2021). *Clean Code: A Handbook of Agile Software Craftsmanship* (2nd ed.). Prentice Hall


**Journals**

1. EEE Software. IEEE Computer Society. https://www.computer.org/csdl/magazine/so
2. Journal of Systems and Software. Elsevier. https://www.journals.elsevier.com/journal-of-systems-and-software
3. ACM Transactions on Software Engineering and Methodology. Association for Computing Machinery. https://dl.acm.org/journal/tosem
4. Software: Practice and Experience. (n.d.). Software: Practice and Experience. Wiley. Retrieved from https://onlinelibrary.wiley.com/journal/1097024x