Programme Curriculum

Participants need to take at least 12 courses towards coursework, and complete one Project/ Dissertation. The coursework requirement for the programme would consist of a set of core courses and electives. Core courses are compulsory for all participants, while electives can be chosen based on individual learning preferences.

First Semester

- Software Architectures
- Cloud Computing
- Agile Software Processes
- Software Product Management

Second Semester

- Software Quality Assurance and Testing
- Elective 1
- Elective 2
- Elective 3

Third Semester

- Elective 4
- Elective 5
- Elective 6
- Elective 7

Fourth Semester

- Dissertation

General Pool of Electives

- Data Structures & Algorithms
- Data visualization and Interpretation
- Artificial and Computational Intelligence
- Blockchain Technologies & Systems
- Cyber Security
- Data Warehousing
- Applied Machine Learning
- Secure Software Engineering
- Middleware Technologies
- Advanced topics in Software Engineering

Pool of Electives :
Full Stack Engineering (FSE)

- Cross Platform Application Development
- Introduction to DevOps#
- Scalable Services#
- Software Testing Methodologies
- Full-stack Application Development#
- Database Design & Applications
- API-based Products
- User Experience Design
- Design of Conversational Experiences
- API-driven Cloud Native Solutions
- Open Source Software Engineering
- Object Oriented Analysis and Design

Pool of Electives :
Software Product Management

- Product Discovery and Requirements Engineering#
- Product Strategy and Planning#
- Communication, Estimation and Negotiation#
- Product Analytics
- API-based Products
- User Experience Design
- Marketing
- Software Project Management
- Open Source Software Engineering

For any specialization, 4 courses (including mandatory electives marked #) are to be selected for that specialization.

For graduating without a specialization, at least 3 courses from those marked # (from either specialization) to be selected.

Note: Student can also obtain the degree without any specialization.

 # indicates mandatory elective for this specialization

Choice of Electives is made available to enrolled students at the beginning of each semester. A limited selection of Electives will be

offered at the discretion of the Institute.

Coures Wise Syllabus

Software Architectures

Systems engineering and software architectures; Hatley°Pirbhai architectural template; architecture flow diagrams; requirements engineering and software architecture; architectural design processes; design post-processing; real- VII-68 time architectures; architectural design patterns; software architecture and maintenance management; object oriented architectures; client-server architectures; forward engineering for object oriented and client-server architectures; emerging software architectures.

Cloud Computing

Concurrency and distributed computing, message passing over the network, connectivity and failure models, local vs remote connectivity, distributed resource modeling, distributed data models; replication & consistency; virtualization; CPU virtualization, memory and storage virtualization, virtualized networks, computing over WAN and Internet; computing on the cloud, computing models, service models and service contracts, programming on the cloud; Cloud infrastructure, LAN vs Wan issue, resource scaling and resource provisions, performance models, scalability, performance measurement and enhancement techniques; cloud applications and infrastructure services.

Agile Software Processes

Introduction to Agile; Basics of Agile Software Development approaches; Principles of Agile; Agile Methodologies; Release Planning; Roles and Artifacts in Agile; Agile Requirements; Iteration Planning and Ceremonies; Executing a Sprint; Agile Metrics; Agile Testing and Maintenance; Agile Pitfalls; Ensuring Agile Success.

Software Product Management

Identifying customer needs. Defining value proposition. Specifying and validating MVP. Building products through agile and scrum. Metrics, measurement and improvements. Software product lifecycle management; analytical evaluation techniques; quality systems.

Software Quality Assurance and Testing

Quality assurance, management and testing; SQA process and activities; Quality planning, metrics and QMS; Team structure & organization of SQA; Quality control tools, Six-Sigma methodology; Types of quality; Software testing as a tool for improving quality; Planning, management & control of testing; Testing in different domains; Test Automation – Strategy, Process and Architecture.

Data Structures & Algorithms

Introduction to Abstract Data Types, Data structures and Algorithms; Analysis of Algorithms – Time and Space Complexity, Complexity Notation, Solving Recurrence Relations.; Divide-and-Conquer as a Design Technique; Recursion – Design of Recursive Functions / Procedures, Tail Recursion, Conversion of Recursive Functions to Iterative Form. Linear data structures – Lists, Access Restricted Lists (Stacks and Queues) – Implementation using Arrays and Linked Lists; Searching and Order Queries. Sorting – Sorting Algorithms (Online vs. Offline, In-memory vs. External, In°space vs. Out-of-space, QuickSort and Randomization). Unordered Collections: Hashtables (Separate Chaining vs. Open Addressing, Probing, Rehashing). Binary Trees – Tree Traversals. Partially Ordered Collections: Search Trees and Height Balanced Search Trees, Heaps and Priority Queues. Algorithm Design: Greedy Algorithms and Dynamic Programming. Graphs and Graph Algorithms: Representation schemes, Problems on Directed Graphs (Reachability and Strong Connectivity, Traversals, Transitive Closure. Directed Acyclic Graphs - Topological Sorting), Problems on Weighted Graphs (Shortest Paths. Spanning Trees). Introduction to Complexity Classes (P and NP) and NP-completeness. NP°Hard problems. Designing Algorithms for Hard Problems – Back tracking, Branch-and-Bound, and Approximation Algorithms.

Data visualization and Interpretation

Visualization as a Discovery tool, Visualization skills for the masses, The Visualization methodology, Visualization design objectives, Exploratory vs. explanatory analysis, Understanding the context for data presentations, 3 minute story, Effective Visuals, Gestalt principles of visual perception, Visual Ordering, Decluttering, Story Telling, Visualization Design; Taxonomy of Data Visualization Methods: Exploring Tableau, Dashboard and Stories, Bullet graphs, Pareto charts, Custom background images; Dashboard : Dashboard categorization and typical data, Characteristics of a Well-Designed Dashboard, Key Goals in the Visual Design Process; Power of Visual Perception: Visually Encoding Data for Rapid Perception, Applying the Principles of Visual Perception to Dashboard Design.

Artificial and Computational Intelligence

Agents and environments, Task Environments, Working of agents; Uninformed Search Algorithms: Informed Search. Local Search Algorithms & Optimization Problems: Genetic Algorithm; Searching with Non-Deterministic Actions, Partial Information and Online search agents, Game Playing, Constraint Satisfaction Problem, Knowledge Representation using Logics: TT-Entail for inference from truth table, Proof by resolution, Forward Chaining and Backward Chaining, Inference in FOL, Unification & Lifting, Forward chaining, Backward Chaining, Resolution; Probabilistic Representation and Reasoning : Inference using full joint distribution, Representation of Conditional Independence using BN, Reinforcement Learning; Difference between crisp and fuzzy logic, shapes of membership function, Fuzzification and defuzzification, fuzzy logic reasoning; Decision making with fuzzy information, Fuzzy Classification; Connectionist Models: Introduction to Neural Networks, Hopfield Networks, Perceptron Learning, Backpropagation & Competitive Learning, Applications of Neural Net: Speech, Vision, Traveling Salesman; Genetic Algorithms - Chromosomes, fitness functions, and selection mechanisms, Genetic algorithms: crossover and mutation, Genetic programming.

Blockchain Technologies & Systems

Highly successful decentralized blockchain-based systems, such as Bitcoin, have immense potential to revolutionize financial, information, and other infrastructures. This course aims to provide a broad overview of the essential concepts involved in blockchain technology in order to lay down the foundation necessary for developing applications. This course also covers the technical aspects of consistency and consensus in distributed algorithms, public distributed ledgers, public-key cryptography and cryptographic properties, cryptocurrencies, and smart contracts. The course aims to develop expertise among students to build these systems, interact with them, and to design and build secure distributed applications.

Cyber Security

Cyber Security principles; Security architectures; Security threats, attacks and vulnerabilities; CIA Triad, Cyber Security Policies, Models and Mechanisms; Types of Cyber Attacks; Security Risk Management; Malware; Ransomware; Implementing Cyber Security Solutions.

Data Warehousing

Introduction, evolution of data warehousing; decision support systems; goals, benefit, and challenges of data warehousing; architecture; data warehouse

information flows; software and hardware requirements; approaches to data warehouse design; creating and maintaining a data warehouse; Online Analytical Processing (OLAP) and multi-dimensional data, multi-dimensional modeling; view materialization; data marts; data warehouse metadata; data mining.

Applied Machine Learning

Need for machine learning. Prediction and classification methods. Use cases in application domains. Interpretation of results. Limitations of various techniques. End to end Machine learning - data collection, data preparation, model selection.

Secure Software Engineering

Best practices for designing secure systems, software engineering principles for designing secure systems, criteria for designing secure systems; analysis of system properties and verification of program correctness; use of formal methods and verification for security; tools for verification of security properties; techniques for software protection (such as code obfuscation, tamper-proofing and watermarking) and their limitations; analysis of software based attacks (and defenses), timing attacks and leakage of information, and type safety.

Middleware Technologies

Evolution of Middleware Technologies: Transaction Processing, Remote Procedure Calls, Message-Oriented°Middleware, Object Request Brokers, Web services and REST; Forms of Middleware: Enterprise Middleware, Web Middleware, and Cloud / Services Middleware; Middleware Elements: communication protocols, middleware protocols, data representation, server process control, naming and directory services, security, system management; Select case studies such as MS .NET, J2EE. Service Oriented Architecture: Loosely Coupled Systems, Business processes, Tiers, Architectural Choices; Resiliency in Middleware: resiliency techniques, hardware failures, communication failures, software failures; Performance and scalability in Middleware; Security in Middleware; Implementation Aspects: business process implementation, enterprise integration, web and database middleware (e.g. NoSQL middleware) change management. Case studies of Enterprise application architecture (EAI) - Eg. Tibco, Websphere.

Advanced Topics in Software Engineering

Recent and emerging topics in software engineering will be discussed in detail with the help of latest publications, software product information and industry practice.

Cross Platform Application Development

Cross-platform applications development involves creation of software applications that are compatible with multiple platforms or software environments. This can be achieved through various development frameworks like Ionic, React Native, Adobe PhoneGap, Xamarin etc. This course aims to equip students with the expertise to design and develop web and mobile based applications that can operate in varied environments and platforms. Additionally, it also aims to develop the understanding of the role and importance of API management in such applications. The course involves hands-on exposure to full stack development of cross°platform applications using some of the existing development frameworks.

Introduction to DevOps

Continual Service - continuous integration and continuous delivery; Scaling: automating infrastructure and infrastructure°as-code; DevOps and Cloud: platform-as-a service and DevOps, use of virtual machines and containers for deployment, Micro-services; application lifecycle management: deployment pipeline and application deployment, continuous deployment pipeline; stack management - life cycle of stack and events, resource and event monitoring, auto healing; Security: security of deployment pipeline, policy-as-code.

Scalable Services

Software principles related to scalability. Architectures for Scaling. Microservices - design, service discovery, load balancing, API management. Deployment - container configurations and orchestrations, automated deployments of microservices, integration with CI/CD pipelines. Performance: Scaling and load balancing with containers and microservices, Ensuring QoS and SLAs

Software Testing Methodologies

Software testing techniques and tools; software testing life cycle and its management; specification-based testing; code-based testing; model-based testing; integration testing; system testing; object oriented testing; regression testing; user acceptance testing; Automated testing at different levels (unit, integration, and system) – scripting and testing tools; test case minimization, prioritization & optimization.

Full-Stack Application Development

Evolution in web app architectures: Client Server - 2 tier, 3/n tiered, Layered; Distributed - SOA, Web Services, Microservices, Cloud (IaaS/PaaS/FaaS); Modern application landscape; Web applications: Typical structure of end-to-end

application; Application components-Frontend / Backend / API / Database / Services; Web Browsers - Client WebAPIs - Browsers APIs for storage, audio, video; Web Assembly; Responsive web; Web Servers; Load balancers; Application servers; API gateways; ORM; DNS; HTTP/S: HTTP headers, HTTP messages, HTTP request methods; Caching; Modern application architectures and Tech-Stacks: Microservices, Serverless; Application development Stacks-Conventional; Modern JavaScript stacks; (Full-Stack) Application Development: Languages (client/server side), Frameworks; Platforms, Deployment (on-prem/cloud); Databases(RDBMS/NoSQL); Interactions(method calls, APIs/REST, messaging); MEAN/MERN as exemplar frameworks.

Database Design & Applications

DBMS architecture; Data models: Network model, Hierarchical model and Relational model; Database design & optimization; Query processing & Query optimization; Transaction Processing; Concurrency control; Recovery; Security & protection; Introduction to Object Oriented data model & Multimedia Databases.

API-based Products

API-based Products: Case for digital business/transformation, API-based Product mindset; API users; API types, API paradigms; API ecosystem; API life cycle; Principles and elements of API design; Collaborative API design process; API Design-First approach; API standards and documentation; API architectural stylesand implementation: REST, gRPC, GraphQL etc., API design and implementation; Async APIs; API design practices: Design patterns and anti-patterns; Scaling, API security, change management/versioning, API publish / release; maintenance/deprecation; API testing strategies: Acceptance testing, Automations, Contract testing; Tools; Developer Experience: DevRel and DX; Developer ecosystem strategy; Developer resources; API product management: API strategy, API economy, API revenue models; Metrics for APIbased Products; API management platforms; API lifecycle management; API analytics.

User Experience Design

UX principles; UX roles and responsibilities; UX design Frameworks; UX strategy; UX design process; UX research: Generative research, Evaluative research, Qualitative and Quantitative research; Usability studies; Observation techniques and feedback methodologies; Synthesis of results- Deriving actionable strategy from the observations; Empathizing with users and user perspective; Interaction design: User stories; User journey maps; Information architecture; UX writing;

Visual design: Wireframes; Prototypes, Storyboards; Design systems; Design for accessibility; Internationalization and Localization; UX evaluation- Testing and Validation; Design for Conversational UI- Chatbots, Personal/Voice Assistants; Proximity-based UI.

Design of Conversational Experiences

Cognitive virtual assistant (CVA): Use-cases; Classification of conversational AI platforms; Architecture of Conversational Platform; Deployment and Pricing models; Platform landscape; Designing Bots: Bot Architecture; Bot Anatomy; Design process overview; Branding, Personality, and Human Involvement; Conversation; Rich interactions; Engagement methods; Use case definition and exploration; Conversation scripting; Context and Memory; User testing; Designing Voice User Interfaces(VUI): Conversational Voice User Interface(VUI); VUI Designer; VUI design principles; Designing effective process and dialogue; Personas, Avatars, Actors; Speech recognition technology; Advanced VUI Design; User testing; Development: Building and deploying conversational AI assistants (voice assistants & chatbots) using cloud native / open source platforms such as Google Dialogflow, RASA or MS Bot framework; Bot Discovery and installation; Monetization; Analytics and Continuous improvement; Trends: SuperBot Platforms; Multiplatform Bots; Identity consolidation; Voice-enabled Devices – Smart Homes and Smart Cars as example environments.

API-driven Cloud Native Solutions

Analyze, Design, Develop and Deploy cloud native applications in innovative areas such as Artificial Intelligence/Machine Learning (AI/ML), IoT, Data Analytics etc.; Build an end to end complex application; Extensive usage of well-known PaaS/APIs; Demonstration of compliance with relevant, industry adapted best practices; Deployment using modern strategies; Presentation of the milestones and outcomes in appropriate forms; Periodic review of progress of the project by faculty.

Open Source Software Engineering

Understanding Open Source Software: The Cathedral and the Bazaar, Teams and Hierarchies, Processes, Licensing, and Business Models; Code walking and analysis of Open Source Products: Analyze the code of a few select products – extract data flow and control flow from an input-output perspective or a high level events perspective; Tools and Technologies in the Open Source Community: Understand and use select open source tools for development, code analysis and transformation, and deployment; Modify a module in an Open Source Product: Identify a functional and/or a critical missing feature / or a potential improvement.

Product Discovery and Requirements Engineering

Finding and defining product opportunities; Market research vs Product discovery; Product discovery techniques - framing, assessing opportunities, planning, ideation, prototyping, testing transformation; Product discovery approaches, frameworks and tools; User Research and experiments; Creation of discovery hypothesis, measurement for product-market fit, MVP specification; OKRs, KPIs definition; Requirements specification and validation; Agile requirements: User personas, Stories, Epics, Themes; Acceptance tests, Product Backlog, Story Maps, Wireframes, Storyboards, Prototypes; Requirements estimation and prioritization techniques; Managing use case evolution.

Product Strategy and Planning

Market scanning and analysis, Strategy Formulation, Strategy Implementation and Control, Strategic Issues in Managing Technology & Innovation, Creating and validating Opportunity Hypothesis; Product vision and product strategy, Principles of Product Vision and Strategy Product objectives; Product Roadmap: Planning, prioritization, development and communication of Roadmap, Roadmap changes; Product Lines and Portfolios. Portfolio Roadmaps; Evaluating product portfolio, OKRs, KPIs; New Products: Proposal, Development, and Launch; Outsourcing, Licensing models, Economics of Software: Development Cost (Buying/Licensing, Outsourcing/Building), Deployment Cost (Packaging and Distribution, Hosting, Scaling, and Usage), Maintenance Cost; Costing vs. pricing (B2B vs. B2C, Services, Usage models); Product release planning: Product evangelism; GTM and Sales strategy; Product positioning and branding.

Communication, Estimation, and Negotiation

Product Manger's Roles and Responsibilities: Understand organizational culture, product stakeholders and their perspectives, Effective product descriptions and presentations, Product news and crisis communication; The cross-functional nature of product manager's work: Strategic and tactical communication, Working with engineering teams, Working with design teams, Working with customerfacing stakeholders, Working with customers; Networking skills; Product-to-Project Translation – Resource (human resources, tools/technologies, and time) estimation and prioritization. Negotiation: Strategic negotiation with leadership; Tactical negotiation with internal (i.e. design/engineering) teams and customerfacing stakeholders.

Product Analytics

Concepts, approaches and process for data Analytics; Product Analytics scope; Qualitative and Quantitative Analysis techniques; Product vs Marketing analytics; Analytics in product lifecycle; Product analytics design; Product phases, goals and metrics; Analytical frameworks; Direct/Indirect, Exploratory, Descriptive/Statistical, Predictive product analytics; Product Analytics tools and platforms; Web and Mobile analytics; Other relevant analysis techniques.

Marketing

Definition and scope, fundamentals of consumer behaviour, competitive behaviour, demand estimation, new product introduction, channels of distribution, advertising and other sales promotion, positioning, marketing regulation, market research, basics of industrial marketing.