**Course Title:** Advance Statistics **Credit Hours:** 3 (3+0)

**Course Description:** This course delves into advanced statistical methods essential for data analysis, modeling, and inference in the context of Artificial Intelligence and Data Science. Topics include multivariate analysis, generalized linear models, methods for clustered and panel data, dimensionality reduction, and modern statistical applications.

**Learning Objectives:** Upon completion of this course, students will be able to:

* Apply multivariate statistical techniques to complex datasets.
* Understand and implement generalized linear models for various data types.
* Analyze clustered and panel data using mixed models.
* Utilize principal component analysis for dimensionality reduction and feature extraction.
* Apply statistical methods to modern AI use cases such as text and image analysis.
* Gain proficiency in using statistical software (e.g., R) for advanced analysis.

**Course Contents:**

1. Introduction to the R statistical software: Data manipulation, visualization, basic statistical tests.
2. Review of Linear Regression: Assumptions, inference, model diagnostics.
3. Multivariate Linear Regression: Estimation, inference, interpretation of multiple predictors.
4. Analysis of Categorical Data: Contingency tables, chi-square tests.
5. Generalized Linear Models (GLMs): Introduction, link functions, exponential families.
6. Logistic Regression: Modeling binary outcomes, interpretation, model evaluation.
7. Poisson Regression: Modeling count data, applications.
8. Analysis of Clustered and Panel Data: Introduction to hierarchical data.
9. Mixed Models: Random effects, fixed effects, model fitting and interpretation.
10. Principal Component Analysis (PCA): Theory, computation, interpretation, applications in dimensionality reduction.
11. Introduction to Factor Analysis and other dimensionality reduction techniques.
12. Modern Use Cases: Statistical methods in text mining (e.g., topic modeling foundations).
13. Modern Use Cases: Statistical methods in image processing (e.g., feature analysis).
14. Model Selection and Validation Techniques.

**Course Title:** Agile Software Development **Credit Hours:** 3 (2+1)

**Course Description:** This course provides a comprehensive understanding of Agile principles, methodologies, and practices for software development. Students will learn various Agile frameworks, techniques for requirements management, planning, estimation, development practices, and team collaboration, with practical application in lab sessions.

**Learning Objectives:** Upon completion of this course, students will be able to:

* Understand and articulate the core values and principles of the Agile Manifesto.
* Compare and contrast different Agile methodologies (Scrum, Kanban, XP, Lean).
* Apply techniques for gathering, writing, and prioritizing Agile requirements using user stories.
* Perform iterative planning and estimation in Agile projects.
* Implement key Agile development practices like TDD, continuous integration, and pair programming.
* Utilize Agile project management tools and metrics.
* Foster effective communication and collaboration within Agile teams.
* Understand strategies for scaling Agile in larger organizations.
* Address common challenges in Agile adoption and transformation.

**Course Contents:**

1. Introduction to Agile Software Development: Manifesto, values, principles, history, comparison with Waterfall.
2. Agile Methodologies and Frameworks: Deep dive into Scrum (roles, ceremonies, artifacts).
3. Agile Methodologies and Frameworks: Kanban (principles, boards, flow).
4. Agile Methodologies and Frameworks: Extreme Programming (XP) practices. Lean principles.
5. Agile Requirements and User Stories: Techniques for gathering and refining requirements.
6. Writing Effective User Stories and Acceptance Criteria. Prioritizing the product backlog.
7. Agile Planning and Estimation: Iterative and release planning.
8. Agile Estimation Techniques: Story points, planning poker. Tracking progress.
9. Agile Development Practices: Test-Driven Development (TDD) and Behavior-Driven Development (BDD).
10. Continuous Integration and Automated Testing. Pair Programming.
11. Agile Project Management and Metrics: Tools and techniques for managing Agile projects.
12. Measuring Team Performance: Velocity, burndown charts, etc. Addressing risks.
13. Agile Team Collaboration and Communication: Effective meetings, cross-functional teams.
14. Scaling Agile: Strategies for large projects (SAFe, LeSS basics). Agile transformation challenges.

**Suggested Labs/Activities:**

* Setting up an Agile project board (e.g., using Trello, Jira).
* Writing and refining user stories for a sample project.
* Practicing planning poker for story estimation.
* Implementing TDD for a small feature.
* Setting up a basic continuous integration pipeline.
* Conducting a mock sprint planning, daily stand-up, review, and retrospective.
* Using collaborative tools for team communication and documentation.

**Course Title:** Agent Based Modeling **Credit Hours:** 3 (3+0)

**Course Description:** This course introduces the principles and practices of Agent-Based Modeling (ABM) as a simulation technique for understanding complex systems. Students will learn how to design, implement, and analyze models composed of autonomous interacting agents to study emergent phenomena in various domains.

**Learning Objectives:** Upon completion of this course, students will be able to:

* Understand the core concepts of agent-based modeling and its applications.
* Design agent-based models for simulating complex systems.
* Implement ABMs using appropriate software tools or programming languages.
* Analyze the results of ABM simulations and interpret emergent behaviors.
* Evaluate the strengths and limitations of ABM compared to other modeling techniques.

**Course Contents:**

1. Introduction to Complex Systems and Simulation.
2. Overview of Modeling Paradigms: Equation-based vs. Agent-based modeling.
3. Fundamentals of Agent-Based Modeling: Agents, environment, interactions, emergence.
4. Designing ABMs: Defining agent rules, behaviors, and state.
5. Choosing ABM Platforms and Tools (e.g., NetLogo, Mesa, Repast).
6. Implementing Simple ABMs (e.g., Schelling's segregation model, predator-prey).
7. Verification and Validation of ABMs.
8. Analyzing ABM Output: Simulation runs, parameter sweeps, statistical analysis.
9. Emergent Phenomena and how to identify them.
10. Applications of ABM in Social Sciences (e.g., diffusion of innovation).
11. Applications of ABM in Ecological Systems (e.g., spread of disease).
12. Applications of ABM in Economic Systems (e.g., market dynamics).
13. Advanced Agent Architectures and Learning Agents.
14. Challenges and Future Directions in ABM.

**Course Title:** Data Mining **Credit Hours:** 3 (3+0)

**Course Description:** This course covers the fundamental concepts, techniques, and applications of data mining. Students will learn methods for pattern discovery, clustering, classification, and association rule mining, along with their practical implementation and evaluation.

**Learning Objectives:** Upon completion of this course, students will be able to:

* Understand the data mining process and its key tasks.
* Apply techniques for pattern discovery, including frequent itemsets and association rules.
* Implement and evaluate different clustering algorithms.
* Apply various classification methods to build predictive models.
* Understand sequential pattern mining and graph pattern mining.
* Explore applications of data mining in different domains.

**Course Contents:**

1. Course Orientation and Introduction to Data Mining: Concepts, tasks, process.
2. Data Preprocessing: Cleaning, integration, transformation, reduction.
3. Pattern Discovery Overview: Basic concepts, frequent patterns.
4. Efficient Pattern Mining Methods: Apriori algorithm, FP-growth.
5. Pattern Evaluation: Interestingness measures.
6. Mining Diverse Frequent Patterns.
7. Sequential Pattern Mining: Concepts and algorithms.
8. Pattern Mining Applications: Spatiotemporal and trajectory patterns.
9. Constraint-Based Mining.
10. Graph Pattern Mining.
11. Pattern-Based Classification.
12. Cluster Analysis Overview: Introduction, types of clustering.
13. Similarity and Dissimilarity Measures for Cluster Analysis.
14. Partitioning-Based Clustering Methods: K-Means, K-Medoids.
15. Hierarchical Clustering Methods: Agglomerative and divisive approaches.
16. Density-Based Clustering Methods: DBSCAN. Grid-Based Methods.
17. Methods for Clustering Validation: Internal and external indices.
18. Data Mining Applications: Text mining, image processing, other domains.

**Course Title:** Data-driven Decision Making **Credit Hours:** 3 (2+1)

**Course Description:** This course focuses on the principles and practices of using data to inform and improve decision-making processes. Students will learn how to identify relevant data, apply analytical techniques, interpret results, and communicate findings to support effective decisions in various contexts.

**Learning Objectives:** Upon completion of this course, students will be able to:

* Understand the importance of data in modern decision-making.
* Identify key performance indicators (KPIs) and relevant data sources.
* Apply basic statistical and analytical techniques to data.
* Utilize data visualization to communicate insights.
* Understand the role of algorithms and models in decision support.
* Critically evaluate data-driven insights and potential biases.
* Apply data-driven approaches to decision problems in different domains.

**Course Contents:**

1. Introduction to Data-Driven Decision Making (DDDM): What is DDDM? Importance and benefits.
2. Key Performance Indicators (KPIs): Definition, selection, measurement.
3. Understanding Data: Types of data, data sources, data quality.
4. Data Identification and Application: The financial perspective.
5. Data Identification and Application: The customer perspective.
6. Data Identification and Application: Marketing and sales perspective.
7. Data Identification and Application: Operational processes and supply chain perspective.
8. Data Identification and Application: Employee perspective.
9. Data Identification and Application: Corporate and social responsibility perspective.
10. Introduction to Algorithms and their role in decision making.
11. Basic Data Analysis Techniques: Descriptive statistics.
12. Introduction to Regression and Correlation.
13. Data Visualization: Principles and tools (e.g., Excel, basic charting).
14. Case Studies and Applications of DDDM.

**Suggested Labs/Activities:**

* Identifying KPIs for a given scenario.
* Collecting and cleaning a small dataset.
* Performing basic statistical analysis on a dataset using software (e.g., Excel, Python with Pandas).
* Creating various types of data visualizations.
* Analyzing case studies of data-driven decisions.
* Working through a data-driven decision-making workbook or case study project.

**Course Title:** Edge AI **Credit Hours:** 3 (2+1)

**Course Description:** This course explores the concepts, techniques, and applications of deploying Artificial Intelligence models on edge devices. Students will learn the differences between cloud and edge AI, the challenges and advantages of edge deployment, and the tools and workflows involved in developing and managing Edge AI systems.

**Learning Objectives:** Upon completion of this course, students will be able to:

* Understand the concepts of edge computing and its relationship with cloud computing.
* Explain the advantages and disadvantages of processing data on edge devices.
* Differentiate between AI, Machine Learning, and Deep Learning.
* Articulate the difference between model training and inference.
* Understand the benefits of running AI inference on edge devices.
* Choose appropriate edge computing devices for specific AI applications.
* Understand the lifecycle of an Edge AI project.
* Identify the principles and practices of Edge MLOps.
* Explore case studies of real-world Edge AI implementations.

**Course Contents:**

1. Introduction to Edge Computing: Definition, characteristics, comparison with cloud computing.
2. Advantages and Disadvantages of Edge Processing.
3. Introduction to the Internet of Things (IoT).
4. Review of Machine Learning Fundamentals: Categories of ML problems.
5. Introduction to Edge AI: Definition, training vs. inference.
6. Traditional Cloud-Based AI Inference Workflow.
7. Benefits of Running AI Algorithms on Edge Devices: Latency, privacy, bandwidth.
8. Examples of Edge AI Systems and Applications.
9. Business Implications of Edge AI Growth.
10. Choosing an Edge AI Device: Factors to consider, types of devices (microcontrollers, embedded systems).
11. Edge AI Lifecycle: Use case identification, constraints.
12. Edge AI Pipeline: Data collection, analysis, feature engineering, training, testing.
13. Edge AI Pipeline: Model deployment and monitoring.
14. Introduction to Edge MLOps: Principles (version control, automation, governance).
15. Edge MLOps vs. Cloud MLOps. Model drift (data drift, concept drift).
16. Edge AI Platforms and Tools (e.g., Edge Impulse, TensorFlow Lite, PyTorch Mobile).
17. Case Studies of Edge AI Implementations (e.g., industrial automation, smart cities, healthcare).

**Suggested Labs/Activities:**

* Setting up a development environment for an edge device (e.g., Raspberry Pi, microcontroller board).
* Deploying a pre-trained simple ML model (e.g., image classification) to an edge device.
* Collecting data on an edge device.
* Performing basic inference on collected data.
* Using an Edge AI platform (e.g., Edge Impulse) to build and deploy a small model.
* Exploring model optimization techniques for edge deployment (e.g., quantization).

**Course Title:** Expert Systems **Credit Hours:** 3 (3+0)

**Course Description:** This course introduces the principles, architecture, and development of expert systems, a classic area of Artificial Intelligence. Students will learn about knowledge representation, inference mechanisms, reasoning under uncertainty, and tools for building rule-based and knowledge-based systems.

**Learning Objectives:** Upon completion of this course, students will be able to:

* Understand the concept and architecture of expert systems.
* Represent knowledge using rules, frames, and semantic networks.
* Understand and apply different methods of inference (forward and backward chaining).
* Handle uncertainty in expert systems using techniques like certainty factors and Bayesian reasoning.
* Explore fuzzy logic and fuzzy inference systems.
* Gain practical experience with expert system shells or programming languages (e.g., CLIPS, Jess).

**Course Contents:**

1. Introduction to Expert Systems: Definition, components, history, applications.
2. Knowledge Representation: Rules (IF-THEN), semantic networks, frames.
3. Knowledge Engineering: Acquiring and modeling knowledge.
4. Methods of Inference: Forward Chaining.
5. Methods of Inference: Backward Chaining.
6. Reasoning Under Uncertainty: Introduction to uncertainty.
7. Bayesian Reasoning and Bayesian Networks (Introduction).
8. Inexact Reasoning: Certainty Factors.
9. Fuzzy Sets: Basic concepts, membership functions.
10. Fuzzy Rules and Fuzzy Reasoning.
11. Fuzzy Inference Systems (e.g., Mamdani, Sugeno).
12. Programming in Expert Systems: Introduction to CLIPS (C Language Integrated Production System).
13. CLIPS: Pattern Matching, Modular Design, Execution Control.
14. Other Expert System Tools and Frameworks (e.g., Jess, Protege basics).

**Course Title:** Fundamental of Data Science **Credit Hours:** 3 (2+1)

**Course Description:** This course provides a foundational introduction to the field of Data Science, covering essential concepts, tools, and techniques for data collection, cleaning, analysis, visualization, and basic machine learning. Students will gain practical experience with data science workflows and tools.

**Learning Objectives:** Upon completion of this course, students will be able to:

* Understand the scope and key activities of Data Science.
* Perform basic data exploration, cleaning, and preprocessing.
* Apply fundamental techniques for data visualization.
* Implement basic machine learning algorithms for classification and regression.
* Understand unsupervised learning techniques like clustering.
* Gain an introduction to text analytics and recommender systems.
* Explore concepts of big data engineering and streaming data.
* Utilize data science tools and platforms (e.g., R, Python, Azure ML Studio basics).

**Course Contents:**

1. Introduction to Big Data, Data Science, and Predictive Analytics.
2. Introduction to Data Science Workflow.
3. Introduction to R Programming or Python for Data Science (basics).
4. Fundamentals of Data Mining (overview).
5. Data Exploration, Visualization, and Feature Engineering.
6. Machine Learning Fundamentals: Introduction to predictive modeling.
7. Classification Algorithms: Decision Trees, Logistic Regression, Naïve Bayes.
8. Model Evaluation and Cross Validation for Classification.
9. Regression Algorithms: Linear Regression, Regularized Regression Models.
10. Model Evaluation for Regression.
11. Unsupervised Learning: K-Means Clustering.
12. Introduction to Text Analytics.
13. Introduction to Recommender Systems: Content-Based and Collaborative Filtering (basics).
14. Evaluation of Recommendation Systems (basic metrics).
15. Introduction to Ensemble Methods (Bootstrapping, Bagging, Boosting).
16. Operationalizing Machine Learning Models: Metrics, tuning parameters.
17. Introduction to Big Data Engineering: Concepts of large-scale systems.
18. Handling Real-Time and Streaming Data (basic concepts).

**Suggested Labs/Activities:**

* Setting up R or Python environment for data science.
* Loading, cleaning, and exploring a dataset.
* Creating visualizations using libraries (e.g., ggplot2 in R, Matplotlib/Seaborn in Python).
* Implementing a simple classification algorithm from scratch or using a library.
* Implementing a simple regression algorithm.
* Applying K-Means clustering to a dataset.
* Working with text data (e.g., TF-IDF).
* Using a data science platform like Azure ML Studio for building and deploying a simple model.

**Course Title:** Fuzzy Systems **Credit Hours:** 3 (3+0)

**Course Description:** This course provides an introduction to the theory and applications of fuzzy sets and fuzzy logic. Students will learn about fuzzy set theory, operations, relations, fuzzy logic, fuzzy inference systems, and explore applications in various domains, including control and expert systems.

**Learning Objectives:** Upon completion of this course, students will be able to:

* Understand the basic concepts of fuzzy sets (Type-1 and Type-2).
* Perform operations on fuzzy sets.
* Understand fuzzy relations and compositions.
* Apply fuzzy logic for reasoning.
* Design and implement fuzzy inference systems.
* Explore applications of fuzzy systems in control and expert systems.
* Gain an introduction to related concepts like Intuitionistic Fuzzy Sets and Soft Sets.

**Course Contents:**

1. Introduction to Crisp Sets vs. Fuzzy Sets.
2. Type-1 Fuzzy Sets: Membership functions, support, core, height, normality.
3. Type-1 Fuzzy Set Operations: Union, Intersection, Complementation.
4. Properties of Fuzzy Set Operations.
5. Distances and Similarity Measures between Fuzzy Sets.
6. Fuzzy Relations and Composition.
7. Fuzzy Numbers and Fuzzy Functions.
8. Introduction to Fuzzy Logic: Linguistic variables, fuzzy propositions.
9. Fuzzy Rules and Fuzzy Reasoning.
10. Fuzzy Inference Systems: Mamdani and Sugeno models.
11. Defuzzification Methods.
12. Introduction to Fuzzy Control Systems.
13. Introduction to Fuzzy Expert Systems.
14. Introduction to Type-2 Fuzzy Sets (basic concepts).
15. Introduction to Intuitionistic Fuzzy Sets (IFSs).
16. Introduction to Soft Sets and Set Operations.
17. Applications of Fuzzy Systems in various fields (e.g., pattern recognition, decision making).

**Course Title:** Game Development **Credit Hours:** 3 (2+1)

**Course Description:** This course provides a practical introduction to the principles and practices of game development, focusing on the game development lifecycle, core game mechanics, and using a popular game engine like Unity. Students will gain hands-on experience in building 2D and 3D games.

**Learning Objectives:** Upon completion of this course, students will be able to:

* Understand the game development lifecycle and the game loop.
* Implement basic game components like sprites, animations, and input handling.
* Utilize a game engine (e.g., Unity) for game creation.
* Implement basic game physics and collision detection.
* Work with 2D and 3D graphics, transformations, scenes, and cameras.
* Incorporate assets, lighting, and particle systems into games.
* Optimize games for different platforms.
* Understand basic game monetization strategies.
* Create and animate 3D characters.

**Course Contents:**

1. Introduction to the Game Development Lifecycle and the Game Loop. History of game development.
2. Basic Components of a Game: Sprites, animations, coordinate systems. Handling input for 2D games.
3. Introduction to Unity Engine: Interface, scenes, game objects, components.
4. Introduction to Game Physics: Concepts (velocity, acceleration, collision, momentum).
5. Introduction to 3D Game Systems and Coordinates. Objects and their transformations (translation, rotation, scaling).
6. 2D and 3D Vectors and their use in games.
7. Scenes and Cameras: Perspective and different game angles (birds eye view, etc.).
8. Object Collision and Triggering Events.
9. Environments and Drawing the World.
10. Particle Systems (fire, smoke, water, etc.).
11. Handling Lights and Shading.
12. Using Terrains, Mountains, Landscapes, and Texturing. Placing objects.
13. Acquiring and Importing Assets from online sources (e.g., Unity Asset Store).
14. Materials, Bounciness, Elasticity, Joints (chains, ropes, hinges).
15. Game Optimization Techniques for performance.
16. Publishing Games on different stores.
17. Monetization Models: Advertisements, in-app purchases.
18. Building Humanoids for 3D games: Character design, texturing, rigging.
19. Creating Animations and using Humanoids in games.

**Suggested Labs/Activities:**

* Setting up Unity and creating a new project.
* Creating a simple 2D game with sprites and basic movement.
* Implementing collision detection in a 2D game.
* Creating a basic 3D scene with objects and transformations.
* Implementing simple 3D physics.
* Adding lighting and particle effects to a scene.
* Importing and using external assets.
* Creating a simple animated 3D character.
* Building and deploying a game to a target platform (e.g., PC build, Android build).

**Course Title:** Game Engine Development **Credit Hours:** 3 (2+1)

**Course Description:** This course explores the fundamental concepts and techniques involved in developing game engines. Students will gain insights into the architecture of game engines, real-time rendering pipelines, physics simulation, and other core systems that power modern games.

**Learning Objectives:** Upon completion of this course, students will be able to:

* Understand the history and evolution of game engines.
* Identify the common systems and architecture of game engines.
* Understand principles of real-time rendering, including scene graphs, lighting, and shading.
* Implement basic physics concepts and integrate them into an engine.
* Explore different techniques for indoor and outdoor rendering.
* Gain an overview of character animation techniques.
* Understand the role of modern physics engines.

**Course Contents:**

1. History of Game Development and Game Engines.
2. Overview of Game Engine Architecture: Core systems (rendering, physics, audio, input).
3. Programming for Game Engines: Language choices, performance considerations.
4. Real-time Rendering: The scene-graph model.
5. Indoor Real-time Rendering Techniques (e.g., BSP, portal rendering).
6. Outdoor Real-time Rendering Techniques (e.g., ROAM, Geomipmapping).
7. Character Animation: Explicit and Implicit methods.
8. Shading: Lighting models, Non-Photorealistic Rendering (NPR).
9. Shadows and Full-screen Effects. High Dynamic Range (HDR) rendering.
10. Physically Based Rendering (PBR) concepts.
11. Physics Engine Fundamentals: Basic physical concepts, properties of bodies (mass, center of mass, moment of inertia).
12. Newton's Laws and their application in games.
13. Kinematics and Kinetics for particles and rigid bodies.
14. Collision Detection and Resolution. Conservation of Momentum.
15. Integrating Physics into a Game Engine.
16. Overview of Current Physics Engines (e.g., PhysX, Bullet).
17. Game Engine Development Pitfalls and Best Practices.
18. Introduction to specific game engines (e.g., Unity 3D, Unreal Engine - architecture overview).

**Suggested Labs/Activities:**

* Setting up a basic rendering window using a graphics API (e.g., OpenGL, Vulkan - basic setup).
* Implementing a simple 2D physics simulation (e.g., bouncing ball).
* Creating a basic scene graph structure.
* Implementing simple lighting or shading.
* Exploring the architecture of an open-source game engine.
* Implementing basic collision detection between simple shapes.

**Course Title:** Generative AI **Credit Hours:** 3 (2+1)

**Course Description:** This course provides an in-depth exploration of generative models in Artificial Intelligence, covering fundamental concepts, key architectures like VAEs, GANs, and Transformers, and their applications in generating various types of data, including text and images. Ethical considerations and recent advancements will also be discussed.

**Learning Objectives:** Upon completion of this course, students will be able to:

* Understand the core concepts and applications of generative models.
* Explain the principles behind Autoencoders and Variational Autoencoders (VAEs).
* Understand the architecture and training of Generative Adversarial Networks (GANs).
* Apply Transformer models for sequence generation tasks.
* Understand the concepts of Large Language Models (LLMs).
* Evaluate the performance of generative models using appropriate metrics.
* Explore techniques for controlling and guiding generative models (e.g., prompt programming).
* Discuss ethical considerations related to generative AI.
* Identify recent research trends and future directions in generative AI.

**Course Contents:**

1. Introduction to Generative AI: What are generative models? Applications.
2. Probability Theory and Deep Learning Foundations for Generative AI.
3. Normalizing Flow Models (Introduction).
4. Autoencoders and Variational Autoencoders (VAEs): Architecture, training, latent space.
5. Autoregressive Models for sequence generation.
6. Generative Adversarial Networks (GANs): Introduction, Generator and Discriminator.
7. Training GANs and addressing stability issues.
8. Conditional GANs and Progressive GANs.
9. Transformers: Architecture (Self-Attention, Positional Encoding).
10. Text Generation and Language Modeling using Transformers.
11. Large Language Models (LLMs): Concepts, pre-training, fine-tuning (overview).
12. Evaluation Metrics for Generative Models (e.g., Inception Score, FID, Perplexity).
13. Optimization Methods for Generative Models.
14. Data Augmentation using Generative Models.
15. Prompt Programming and Neural Text Decoding Techniques.
16. Ethical Considerations in Generative AI: Bias, fairness, privacy.
17. Addressing Ethical Challenges.
18. Advances in Generative AI: Review of recent research and emerging trends.

**Suggested Labs/Activities:**

* Implementing a simple Autoencoder or VAE using a deep learning framework (e.g., TensorFlow, PyTorch).
* Implementing a basic GAN for generating simple data (e.g., MNIST digits).
* Experimenting with a pre-trained Transformer model for text generation.
* Exploring prompt engineering techniques using a publicly available LLM API.
* Implementing conditional generation with a GAN or VAE.
* Evaluating the output of generative models using relevant metrics.

**Course Title:** HCI & Computer Graphics **Credit Hours:** 3 (2+1)

**Course Description:** This course explores the interdisciplinary fields of Human-Computer Interaction (HCI) and Computer Graphics. Students will learn principles of designing effective user interfaces, usability testing, and fundamental concepts in 2D and 3D computer graphics, including rendering, animation, and visual design.

**Learning Objectives:** Upon completion of this course, students will be able to:

* Understand basic principles and guidelines of Human-Computer Interaction.
* Apply user-centered design principles.
* Design effective user interfaces and create prototypes.
* Understand principles of visual design for user interfaces.
* Understand fundamental concepts in 2D and 3D computer graphics.
* Learn about rendering algorithms, lighting, and shading techniques.
* Gain an introduction to animation, Virtual Reality, and Augmented Reality.
* Conduct basic usability testing and evaluate user experience.

**Course Contents:**

1. Introduction to Human-Computer Interaction (HCI): Definition, importance, history.
2. Basic Principles and Guidelines of HCI (e.g., usability, learnability, efficiency).
3. User-Centered Design Process: Understanding users, requirements gathering.
4. Usability Testing and Evaluation Methods.
5. Designing Effective User Interfaces: Principles and guidelines.
6. User Interface Prototyping: Low-fidelity and high-fidelity prototypes.
7. Prototyping using Wireframes and Mockups.
8. Designing for Accessibility and Mobile Devices.
9. Visual Design Principles for User Interfaces: Layout, typography, color, iconography.
10. Introduction to Computer Graphics: Overview, applications.
11. 2D Graphics Concepts: Coordinate systems, transformations.
12. 3D Graphics Concepts: Coordinate systems, transformations (translation, rotation, scaling).
13. Introduction to 3D Modeling and Scene Representation.
14. Rendering Pipeline: Overview.
15. Lighting and Shading Techniques (e.g., Phong, Gouraud).
16. Introduction to Rendering Algorithms (e.g., Ray Tracing basics).
17. Introduction to Animation Techniques.
18. Introduction to Virtual Reality (VR) and Augmented Reality (AR).
19. Usability Testing and Evaluation (revisited with practical focus).
20. User Feedback and User Experience (UX) Metrics.

**Suggested Labs/Activities:**

* Analyzing the usability of existing interfaces.
* Creating wireframes and mockups for a simple application interface.
* Developing a simple interactive prototype using a prototyping tool.
* Implementing basic 2D graphics drawing using a programming library.
* Implementing basic 3D transformations.
* Creating a simple 3D scene.
* Conducting a small-scale usability test.

**Course Title:** High Performance Computing **Credit Hours:** 3 (2+1)

**Course Description:** This course introduces the fundamental concepts and techniques of High Performance Computing (HPC). Students will learn about parallel computing architectures, programming models for shared and distributed memory systems, GPU programming, performance analysis, and applications of HPC in scientific computing and AI.

**Learning Objectives:** Upon completion of this course, students will be able to:

* Understand fundamental concepts in High Performance Computing.
* Differentiate between shared memory and distributed memory architectures.
* Write parallel programs using OpenMP for shared memory systems.
* Write parallel programs using MPI for distributed memory systems.
* Gain an introduction to GPU programming.
* Analyze the performance of parallel programs.
* Understand concepts of parallel decomposition and high-performance I/O.
* Explore typical scientific and AI applications of HPC.

**Course Contents:**

1. Fundamental Concepts in High Performance Computing: Definition, motivation, metrics (FLOPS, speedup, efficiency).
2. Parallel Computing Architectures: Shared memory, distributed memory, clusters, supercomputers.
3. Parallel Programming Models: Shared memory vs. Message Passing.
4. Shared Memory Programming: Introduction to OpenMP.
5. OpenMP Directives for parallel regions, work sharing, and synchronization.
6. Message Passing Programming: Introduction to MPI (Message Passing Interface).
7. MPI: Basic communication primitives (send, receive).
8. MPI: Collective communication operations (broadcast, reduce).
9. Parallel Decomposition Techniques: Domain decomposition, task parallelism.
10. Performance Measurement and Analysis: Profiling, Amdahl's Law, Gustafson's Law.
11. Introduction to GPU Programming (e.g., CUDA or OpenCL basics).
12. High Performance I/O.
13. High Performance Networking.
14. High Performance Computing Systems: Architectures and interconnects.
15. Typical Scientific Applications of HPC (e.g., simulations).
16. Applications of HPC in Artificial Intelligence (e.g., training deep learning models).

**Suggested Labs/Activities:**

* Setting up a parallel programming environment (e.g., OpenMP and MPI).
* Writing and running simple OpenMP programs.
* Writing and running simple MPI programs.
* Implementing a parallel version of a simple algorithm (e.g., matrix multiplication, sorting).
* Measuring the speedup and efficiency of parallel programs.
* Exploring basic GPU programming examples.

**Course Title:** Information Retrieval **Credit Hours:** 3 (2+1)

**Course Description:** This course covers the fundamental concepts and techniques of Information Retrieval (IR), the field concerned with searching for information in documents and databases. Students will learn about indexing, querying, scoring, evaluation, and modern IR techniques.

**Learning Objectives:** Upon completion of this course, students will be able to:

* Understand the basic concepts of Information Retrieval.
* Learn about inverted indices and Boolean queries.
* Understand techniques for text processing and index construction.
* Apply scoring and ranking methods, including the vector space model.
* Evaluate the performance of IR systems using standard metrics.
* Explore advanced topics like query expansion and retrieval models.
* Gain an introduction to classification and clustering in IR.

**Course Contents:**

1. Introduction to Information Retrieval: Definition, goals, components.
2. Basic IR Model: Document collection, queries.
3. Inverted Indices: Structure and creation.
4. Boolean Queries and Boolean Retrieval Model.
5. The Term Vocabulary and Postings Lists.
6. Text Encoding and Tokenization.
7. Index Construction: Hardware and software considerations.
8. Index Compression Techniques.
9. Dictionaries and Tolerant Retrieval: Wild-card queries, permuterm indices, n-gram indices.
10. Spelling Correction and Synonyms.
11. Scoring, Term Weighting, and the Vector Space Model.
12. TF-IDF weighting. Cosine Similarity.
13. Computing Scores in a Complete Search System.
14. Evaluation of IR Systems: Metrics (Precision, Recall, F1-Score, Mean Average Precision).
15. Relevance Feedback and Query Expansion.
16. Introduction to Probabilistic IR Models.
17. Link Analysis (e.g., PageRank basics).
18. Introduction to Classification and Clustering in IR (K Nearest Neighbors, LSI).

**Suggested Labs/Activities:**

* Building a simple inverted index for a small document collection.
* Implementing Boolean queries over the index.
* Calculating TF-IDF scores for documents.
* Implementing the vector space model for scoring.
* Calculating Precision and Recall for a set of search results.
* Experimenting with a publicly available IR toolkit.

**Course Title:** Introduction to Autonomous Robotics **Credit Hours:** 3 (2+1)

**Course Description:** This course provides an introduction to the fundamental concepts and techniques for designing and programming autonomous robots. Topics include robot kinematics, sensors, perception, localization, mapping, path planning, and control, with practical exercises using robot simulators or hardware.

**Learning Objectives:** Upon completion of this course, students will be able to:

* Understand the basic components and types of autonomous robots.
* Learn about robot kinematics and inverse kinematics.
* Understand different types of sensors used in robotics.
* Gain an introduction to robot perception and computer vision for robotics.
* Learn techniques for robot localization and mapping (SLAM basics).
* Understand algorithms for path planning and navigation.
* Explore basic robot control strategies.
* Gain practical experience with robot programming using simulators or hardware.

**Course Contents:**

1. Introduction to Robotics: Definition, history, types of robots.
2. Autonomous Robotics: Concept and challenges.
3. Robot Components: Actuators, sensors, manipulators.
4. Robot Kinematics: Forward and Inverse Kinematics (basics).
5. Sensors for Robotics: Proprioceptive and Exteroceptive sensors (e.g., encoders, cameras, LiDAR).
6. Robot Perception: Processing sensor data.
7. Computer Vision for Robotics (basics).
8. Robot Localization: Estimating robot's position.
9. Mapping: Building representations of the environment.
10. Simultaneous Localization and Mapping (SLAM) - Introduction.
11. Path Planning: Algorithms for finding paths (e.g., A\*, Dijkstra's - review/application).
12. Navigation: Following planned paths, avoiding obstacles.
13. Robot Control: Basic control loops.
14. Introduction to Robot Operating System (ROS) or other robotics frameworks.
15. Robot Programming: Using simulators (e.g., Gazebo, CoppeliaSim).
16. Introduction to Robot Learning (basics).
17. Human-Robot Interaction (Introduction).
18. Ethical considerations in Autonomous Robotics.

**Suggested Labs/Activities:**

* Setting up a robot simulator and basic environment.
* Programming basic robot movement.
* Using simulated sensors to collect data.
* Implementing a simple obstacle avoidance behavior.
* Implementing a basic path planning algorithm in the simulator.
* Controlling a simulated robot arm (basic kinematics).
* (If hardware is available) Programming a simple task on a physical robot.

**Course Title:** Knowledge Based Systems **Credit Hours:** 3 (3+0)

**Course Description:** This course explores the principles and techniques for building knowledge-based systems, which are computer programs that use a knowledge base to solve problems. It covers knowledge representation formalisms, inference mechanisms, and the architecture of such systems, building upon concepts from Expert Systems.

**Learning Objectives:** Upon completion of this course, students will be able to:

* Understand the concept and architecture of knowledge-based systems.
* Represent knowledge using various formalisms (logic, rules, frames, ontologies).
* Understand and apply different inference mechanisms.
* Differentiate between data, information, and knowledge.
* Explore techniques for knowledge acquisition and engineering.
* Understand the role of ontologies in knowledge-based systems.
* Gain an introduction to semantic web technologies.

**Course Contents:**

1. Introduction to Knowledge-Based Systems (KBS): Definition, components, relationship to AI.
2. Data, Information, and Knowledge.
3. Knowledge Representation: Logic (Propositional and First-Order Logic - review/application).
4. Knowledge Representation: Rules and Rule-Based Systems.
5. Knowledge Representation: Frames and Semantic Networks.
6. Knowledge Representation: Ontologies (Introduction).
7. Inference Mechanisms: Deduction, Abduction, Induction (overview).
8. Inference in Rule-Based Systems (Forward and Backward Chaining - revisited).
9. Inference in Logic-Based Systems (Resolution - introduction).
10. Knowledge Acquisition Techniques.
11. Knowledge Engineering Process.
12. Architecture of Knowledge-Based Systems: Knowledge base, inference engine, user interface.
13. Handling Uncertainty in KBS (revisited, potentially more formal methods).
14. Introduction to Description Logics.
15. Introduction to Semantic Web Technologies (RDF, SPARQL basics).
16. Building Simple Knowledge-Based Systems.
17. Applications of KBS in various domains.

**Course Title:** Large Language Model (LLM) **Credit Hours:** 3 (2+1)

**Course Description:** This course provides a comprehensive exploration of Large Language Models (LLMs), covering their underlying architectures (especially Transformers), pre-training and fine-tuning techniques, applications in natural language processing, prompt engineering, and ethical considerations.

**Learning Objectives:** Upon completion of this course, students will be able to:

* Understand the fundamentals of Natural Language Processing (NLP).
* Explain the concepts of statistical and neural language models.
* Understand the Transformer architecture in detail.
* Differentiate between different pre-training strategies for LLMs.
* Apply techniques for fine-tuning and adapting LLMs for specific tasks.
* Utilize prompt-based learning and advanced prompting techniques.
* Understand methods for aligning LLMs with human preferences (e.g., RLHF).
* Explore techniques for retrieval-augmented generation.
* Gain an introduction to knowledge graphs and their relationship with LLMs.
* Discuss ethical considerations, bias, and toxicity in LLMs.

**Course Contents:**

1. Course Introduction and Introduction to NLP: NLP Pipeline, Applications.
2. Introduction to Statistical Language Models: N-grams, Smoothing, Evaluation.
3. Introduction to Deep Learning for NLP: Perceptron, ANN, CNN, RNN (review).
4. Introduction to PyTorch or TensorFlow for NLP tasks.
5. Word Representation: Word2Vec, fastText, GloVe.
6. Tokenization Strategies.
7. Neural Language Models: LSTM, GRU.
8. Sequence-to-Sequence Models: Encoder-Decoder architecture.
9. Decoding Strategies: Greedy Search, Beam Search, Nucleus Sampling, Temperature Sampling.
10. Attention Mechanism in Sequence-to-Sequence Models.
11. Introduction to Transformers: Self and Multi-Head Attention, Positional Encoding, Layer Normalization.
12. Implementation of Transformers using deep learning frameworks.
13. Pre-Training Strategies: ELMo, BERT (Encoder-only).
14. Pre-Training Strategies: Encoder-decoder and Decoder-only Models.
15. Introduction to HuggingFace Transformers Library.
16. Fine-tuning LLMs for downstream tasks.
17. Instruction Tuning and Prompt-based Learning.
18. Advanced Prompting Techniques and Prompt Sensitivity.
19. Alignment of Language Models with Human Feedback (RLHF).
20. Retrieval-Augmented Generation (RAG): Concepts and techniques.
21. Open-book Question Answering.
22. Knowledge Graphs (KGs): Representation, completion, alignment.
23. Relationship between KGs and LLMs.
24. Parameter-efficient Adaptation (Prompt Tuning, Prefix Tuning, LoRA).
25. Interpretability Techniques for LLMs.
26. Overview of recently popular LLMs (e.g., GPT, Llama, Claude, Gemini).
27. Ethical NLP: Bias and Toxicity in LLMs.
28. Conclusion and Future Directions.

**Suggested Labs/Activities:**

* Using a deep learning framework to implement basic NLP tasks (e.g., text classification).
* Experimenting with different tokenization strategies.
* Implementing a simple Attention mechanism.
* Using the HuggingFace library to load and experiment with pre-trained Transformer models.
* Fine-tuning a pre-trained LLM on a small dataset.
* Experimenting with different prompting techniques using an LLM API.
* Exploring techniques for evaluating LLM outputs.
* Implementing a basic RAG system.

**Course Title:** MLOps **Credit Hours:** 3 (2+1)

**Course Description:** This course introduces the principles and practices of MLOps (Machine Learning Operations), focusing on the deployment, monitoring, and management of machine learning models in production environments. Topics include version control, automation, testing, deployment strategies, and monitoring model performance.

**Learning Objectives:** Upon completion of this course, students will be able to:

* Understand the MLOps lifecycle and its importance.
* Apply version control to machine learning code, data, and models.
* Automate various stages of the ML pipeline.
* Implement testing strategies for ML models and code.
* Understand different model deployment strategies.
* Monitor model performance in production and detect model drift.
* Utilize tools and platforms for MLOps.
* Understand the differences between cloud-based and edge MLOps.

**Course Contents:**

1. Introduction to MLOps: Definition, motivation, lifecycle.
2. MLOps Principles: Version Control, Automation, Governance.
3. Version Control for ML Projects: Code, data, models (using Git and platforms like GitHub/GitLab).
4. Linux System and Bash Scripting for MLOps workflows.
5. Automation of the ML Pipeline: Data ingestion, training, evaluation.
6. Continuous Integration (CI) for ML code.
7. Testing ML Models: Unit tests, integration tests, data validation tests.
8. Setting up unit tests with Pytest.
9. Introduction to integration tests.
10. Continuous Delivery/Deployment (CD) for ML models.
11. Model Deployment Strategies: REST APIs, containerization (Docker basics).
12. Introduction to Workflows and Automation with tools like GitHub Actions or Jenkins.
13. Monitoring ML Models in Production: Performance metrics, logging.
14. Detecting Model Drift: Data drift and concept drift.
15. Retraining and Updating Models.
16. MLOps Platforms and Tools (e.g., MLflow, Kubeflow, AWS SageMaker - overview).
17. Edge MLOps: Challenges and differences from cloud MLOps.
18. Governance and Compliance in MLOps.

**Suggested Labs/Activities:**

* Setting up a Git repository for an ML project and practicing version control.
* Writing and running unit tests for ML code.
* Creating a simple Bash script for automating a task.
* Setting up a basic CI pipeline using GitHub Actions or a similar tool.
* Containerizing a simple ML model using Docker.
* Deploying a containerized model locally.
* Implementing basic monitoring for a deployed model.
* Exploring an MLOps platform through tutorials.

**Course Title:** Recommender Systems **Credit Hours:** 3 (2+1)

**Course Description:** This course provides an introduction to the principles, techniques, and evaluation of recommender systems. Students will learn about different recommendation approaches, including content-based, collaborative filtering, and hybrid methods, and explore advanced topics and ethical considerations.

**Learning Objectives:** Upon completion of this course, students will be able to:

* Understand the fundamental concepts and applications of recommender systems.
* Explain and implement content-based filtering techniques.
* Explain and implement collaborative filtering techniques (user-based and item-based).
* Understand matrix factorization techniques for recommendation.
* Evaluate recommender systems using appropriate metrics.
* Explore concepts of exploration vs. exploitation.
* Gain an introduction to context-aware and multi-objective recommendation.
* Discuss ethical considerations in recommender systems.

**Course Contents:**

1. Introduction to Recommender Systems: Problem definition, applications, types.
2. Data for Recommender Systems: User-item interactions, content data.
3. Content-Based Filtering: Representing items and users, similarity measures.
4. Collaborative Filtering: User-based collaborative filtering.
5. Collaborative Filtering: Item-based collaborative filtering.
6. Matrix Factorization Techniques: Singular Value Decomposition (SVD) basics.
7. Matrix Factorization Techniques: Funk SVD, Alternating Least Squares (ALS).
8. Evaluation Methodology and Metrics: Offline evaluation (Precision, Recall, F1-Score, RMSE, MAE).
9. Evaluation Methodology and Metrics: Online evaluation (A/B testing basics).
10. Implicit Feedback and its handling in recommender systems.
11. Cold Start Problem: Solutions for new users and new items.
12. Exploration vs. Exploitation Trade-off.
13. Introduction to Hybrid Recommender Systems.
14. Introduction to Deep Learning for Recommender Systems.
15. Personalization and Context-Aware Recommendation (Introduction).
16. Introduction to Natural Language Processing for Recommender Systems (e.g., using text data).
17. Multi-objective Recommendation (Introduction).
18. Ethical Considerations in Recommender Systems: Bias, fairness, transparency, privacy.

**Suggested Labs/Activities:**

* Implementing a simple content-based recommender system.
* Implementing a simple user-based or item-based collaborative filtering system.
* Applying matrix factorization to a recommendation dataset.
* Implementing evaluation metrics for recommender systems.
* Experimenting with a recommendation library (e.g., Surprise, LightFM).
* Analyzing a real-world recommendation dataset.

**Course Title:** Speech Processing **Credit Hours:** 3 (2+1)

**Course Description:** This course provides an introduction to the field of speech processing, covering the fundamentals of speech production and perception, digital signal processing techniques for speech analysis, and key applications such as speech recognition and speech synthesis.

**Learning Objectives:** Upon completion of this course, students will be able to:

* Understand the basics of speech production and perception.
* Apply digital signal processing techniques for speech analysis.
* Extract relevant features from speech signals.
* Understand the principles of Automatic Speech Recognition (ASR).
* Understand the principles of Speech Synthesis (Text-to-Speech).
* Explore other applications of speech processing.
* Gain practical experience with speech processing tools and libraries.

**Course Contents:**

1. Introduction to Speech Processing: What is speech? Applications.
2. Fundamentals of Speech Production: Anatomy and physiology.
3. Fundamentals of Speech Perception: Auditory system basics.
4. Digital Representation of Speech Signals: Sampling, quantization.
5. Basic Digital Signal Processing for Speech: Filtering, Fourier Analysis.
6. Time-Domain Speech Analysis: Pitch detection, energy.
7. Frequency-Domain Speech Analysis: Spectrograms.
8. Linear Predictive Coding (LPC) for speech analysis.
9. Feature Extraction for Speech Processing: MFCCs, Perceptual Linear Prediction (PLP).
10. Introduction to Automatic Speech Recognition (ASR): Components (acoustic model, language model).
11. Hidden Markov Models (HMMs) for ASR (Introduction).
12. Deep Learning for ASR (Introduction).
13. Introduction to Speech Synthesis (Text-to-Speech - TTS): Concatenative vs. Parametric.
14. Deep Learning for TTS (Introduction).
15. Speaker Recognition and Verification (Introduction).
16. Emotion Recognition from Speech (Introduction).
17. Speech Enhancement and Noise Reduction (Introduction).
18. Speech Processing Libraries and Tools (e.g., Librosa, Praat, Kaldi - overview).

**Suggested Labs/Activities:**

* Recording and visualizing speech signals.
* Implementing basic signal processing operations on speech (e.g., filtering).
* Extracting features like MFCCs from speech.
* Using a speech processing library to perform basic tasks (e.g., pitch detection).
* Experimenting with a pre-trained ASR model or API.
* Experimenting with a pre-trained TTS model or API.

**Course Title:** Swarm Intelligence **Credit Hours:** 3 (2+1)

**Course Description:** This course explores the fascinating field of Swarm Intelligence, which studies the collective behavior of decentralized, self-organized systems, often inspired by nature (e.g., ant colonies, bird flocks). Students will learn about agent-based modeling, behavioral and computational swarm intelligence algorithms, and their applications in optimization and problem-solving.

**Learning Objectives:** Upon completion of this course, students will be able to:

* Understand the principles of agent-based modeling and complex systems.
* Explain concepts of behavioral swarm intelligence (e.g., flocking).
* Understand and apply computational swarm intelligence algorithms for optimization (e.g., PSO, ACO).
* Explore other swarm-inspired algorithms (e.g., Bees Algorithm, Bats Algorithm).
* Apply swarm intelligence algorithms to solve real-world problems.
* Differentiate between behavioral and computational swarm intelligence.

**Course Contents:**

1. Introduction to Swarm Intelligence: Definition, inspiration from nature, properties (decentralization, self-organization, emergence).
2. Agent-Based Modeling: Review of concepts, bottom-up modeling.
3. System Theory and Complex Systems. Multi-agent systems.
4. Behavioral Swarm Intelligence: Modeling collective behaviors.
5. Flocking Behavior: The Boids model.
6. Applications of Flocking Behavior (e.g., simulations, robotics).
7. Computational Swarm Intelligence (CSI): Optimization theory overview.
8. Multi-objective Optimization (Introduction).
9. Particle Swarm Optimization (PSO): Algorithm, variations, applications.
10. Ant Colony Optimization (ACO): Inspiration from ants, algorithms for optimization problems (e.g., Traveling Salesperson Problem).
11. Bees Colony Algorithm (BCO).
12. Bats Algorithm.
13. Other Swarm-Inspired Algorithms (brief overview).
14. Selected Applications of Swarm Intelligence: Multi-robot path planning.
15. Selected Applications of Swarm Intelligence: Task scheduling.
16. Selected Applications of Swarm Intelligence: Data clustering, feature selection.
17. Implementing Swarm Intelligence Algorithms.
18. Challenges and Future Directions in Swarm Intelligence.

**Suggested Labs/Activities:**

* Implementing a simple agent-based model (e.g., a basic flocking simulation).
* Implementing the Particle Swarm Optimization (PSO) algorithm.
* Implementing the Ant Colony Optimization (ACO) algorithm for a simple problem.
* Applying a swarm intelligence algorithm to a test function optimization problem.
* Exploring a library for swarm intelligence algorithms.

**Course Title:** Vulnerability Assessment & Reverse Engineering **Credit Hours:** 3 (2+1)

**Course Description:** This course provides an introduction to the principles and techniques of vulnerability assessment and reverse engineering, focusing on software and system security. Students will learn how to identify security weaknesses, analyze compiled code, and understand malware analysis techniques.

**Learning Objectives:** Upon completion of this course, students will be able to:

* Understand the basics of computer architecture and assembly language (Intel Assembly).
* Learn the process and algorithms used in disassembly.
* Identify common software vulnerabilities (e.g., integer errors).
* Perform basic source code and binary analysis.
* Understand techniques for analyzing stripped binaries.
* Utilize debugging tools for security analysis.
* Understand modern exploit mitigation techniques (ASLR, DEP, CFG).
* Gain an introduction to Return-Oriented Programming (ROP) and Structured Exception Handling (SEH).
* Understand basic OS internals relevant to security.
* Gain an introduction to malware analysis.
* Explore fuzzing techniques for vulnerability discovery.

**Course Contents:**

1. Introduction to Computer Security and Vulnerability Assessment.
2. Introduction to Intel Assembly Language.
3. Computer Architecture Basics relevant to security.
4. The Disassembly Process and Algorithms.
5. Static Binary Analysis Techniques.
6. Dynamic Binary Analysis Techniques.
7. Analyzing Stripped Binaries.
8. Debugging Demystified: Using debuggers for analysis.
9. Identifying Common Software Vulnerabilities: Buffer overflows, format string bugs.
10. Integer Errors and their exploitation.
11. Source Code Analysis for vulnerabilities.
12. Introduction to Exploit Mitigation Techniques: ASLR (Address Space Layout Randomization).
13. Exploit Mitigation Techniques: DEP (Data Execution Prevention), CFG (Control Flow Guard).
14. Introduction to Return-Oriented Programming (ROP).
15. Introduction to Structured Exception Handling (SEH) exploitation.
16. Basic OS Internals relevant to security (e.g., memory management, process execution).
17. Introduction to Malware Analysis: Types of malware, basic analysis techniques.
18. Introduction to Fuzzing for vulnerability discovery.
19. Advanced Reverse Engineering Topics (overview).

**Suggested Labs/Activities:**

* Setting up a reverse engineering environment (e.g., disassembler, debugger).
* Analyzing simple assembly code snippets.
* Using a debugger to step through a program.
* Identifying a simple vulnerability in a small C program.
* Analyzing a simple binary file.
* Experimenting with basic fuzzing tools.
* (Ethical considerations are paramount; all labs should be conducted in isolated, controlled environments).