Final Year Projects – Computer Science Department, University of Moratuwa

#	Title	Description
1	Smart systems implementation with machine learning	**Problem**: System anomaly detection and root cause identification using machine learning, based on analysis of system statistics and other observable states. **Description**: Capital market systems are mission-critical real-time systems; anomalous behaviors in these systems can cause serious business impacts. Early identification of unexpected system behaviors can prevent serious system failures. Machine learning concepts such as outlier/novelty detection can be used to identify possible system failures and alert the system operators to take corrective/preventive actions. Also when a failure does happen, it's important to identify the origination point of the problem, so that appropriate recovery actions can be taken, without being misled by all the consequent cascading failures that occur afterward. This can be done by analyzing different failures patterns and building a failure model that can be applied to future failures. **Additional information:** This is a continuation of several final year projects that had been done previously. Several supervised and unsupervised machine learning methods have been investigated for anomaly detection, and a subsystem based root cause identification model (a theoretical model) has been explored for root cause identification. The new project is expected to improve on these and/or explore new avenues yielding better results.
2	Collusion detection for Market Surveillance	**Problem:** Market surveillance systems typically analyze the trading activities of different users and try to ascertain whether there is market manipulation. However, it is difficult to detect such manipulations when seemingly unrelated parties get together and collectively do it. The expectation is to detect such collusion between unrelated parties manipulating the market using machine learning techniques. **Description:** A market is a collection of different complex behavior. In a typical time window, it is possible to model the underline behavior (generator) of the market. If the market is not behaving as it is expected to behave (modeled behavior), an outlier behavior is generated and a system can be built to capture that deviation. In a market, a set of attributes/models can be used to characterize the behavior of a given time window. These attributes are also used to define different abusive scenarios (e.g., pump-and-dump). People don't manipulate individually, they work as groups to manipulate stocks. These manipulative groups buy/sell stocks with each other to manipulate the market. These not-normal trading rings should be detected and be used for market manipulation detection.

3	Settlement Optimisation using machine learning	**Problem**: Most of the previous decade in the capital markets space has been spent on optimizing the efficiency and speed of the order execution. As a result, they generate hundreds of millions of trades per day. The underlying order processing latencies are already measured in nanoseconds. Despite the fact that buy and sell orders can be rapidly processed, the settlement process, the actual delivery of the underlying security asset (to the buyer) and actual payment (to the seller) are still measured in days. **Description**: The settlement is a batch process where millions of settlement instructions are processed. Current heuristics-based optimization algorithm, tries to minimize settlement failures, given the constraint of limited stock and cash balances. The scope of the project is to complete the settlement process within the minimum possible time by optimizing the algorithm so that settlement failures are minimum.
4	Software code relationship analysis	**Problem**: Automated impact analysis of changing parts of a large codebase. **Description**: Software entropy is the tendency for software to become increasingly difficult and costly to maintain. One main contributing factor is the unforeseen impacts that propagate to other parts of the system when one area in the codebase in modified. The requirement is to automatically scan a large repository of software code and identify relationships between different parts of code for the purpose of identifying impact if a change is to be done on a certain section of the code. This will require, for example, when a function is modified, to identify callers and callees of that function and assess whether the change in the given function impacts the callers and callees. It should be possible to limit the depth of scanning to control the execution time (e.g. depth=1 would mean the scanning happens to immediate callers and callees, depth=2 to means scan extends to callers of the immediate callers and so on) **Other uses**: The information can also be used for other purposes such as determining the complexity of change and also to identify possible areas for code refactoring. **Possible Enhancements**: Correlate the identified impacts with the results from an automated code coverage tool to highlight the risk of impacted but untested code.

5	Application lifecycle management in the cloud	**Problem**: Current application orchestrator technologies available in the market, for managing application high availability generally works for the collection of stateless processes running as a load-balanced cluster. However, when it comes to stateful applications such as capital markets software, there are other clustering requirements such as primary/replica and also requirements involving a group of processes running as a collective application cluster. **Description:** Research on stateful application orchestration with high availability clusters with extensive load balancing capabilities that is suitable for low latency systems. Find a suitable approach for the above using and extending existing components available and/or by developing a new clustering technique.