<img src="media/image1.jpg" style="width:2.02569in;height:1.51944in"  
alt="BDlogo" />\*\*<span class="smallcaps">  
</span>\*\*  
  
\*\*<span class="smallcaps">PERFORMANCE OPTIMIZATION REPORT</span>\*\*  
  
\*\*<span class="smallcaps">FOR</span>\*\*  
  
\*\*<span class="smallcaps">COMBINED SYSTEM DEVELOPMENT SERVICES</span>\*\*  
  
\*\*<span class="smallcaps">FOR</span>\*\*  
  
\*\*<span class="smallcaps">LICENSING SELF-CERTIFICATION PORTAL</span>\*\*  
  
\*\*<span class="smallcaps">OF</span>\*\*  
  
\*\*<span class="smallcaps">BUILDINGS DEPARTMENT</span>\*\*  
  
\*\*Version 0.1\*\*  
  
\*\*Jan 2025\*\*  
  
? The Government of the Hong Kong Special Administrative Region  
  
The contents of this document remain the property of and may not be  
reproduced in whole or in part without the express permission of the  
Government of the HKSAR.  
  
<table>  
<colgroup>  
<col style="width: 28%" />  
<col style="width: 71%" />  
</colgroup>  
<thead>  
<tr class="header">  
<th colspan="2"><strong>Distribution</strong></th>  
</tr>  
<tr class="odd">  
<th>Copy No.</th>  
<th>Holder</th>  
</tr>  
<tr class="header">  
<th>1</th>  
<th><blockquote>  
<p>Buildings Department (BD)</p>  
</blockquote></th>  
</tr>  
<tr class="odd">  
<th>2</th>  
<th><blockquote>  
<p>Master Concept (Hong Kong) Limited (MC)</p>  
</blockquote></th>  
</tr>  
</thead>  
<tbody>  
</tbody>  
</table>  
  
| Amendment History | | | | |  
|---------|------------------------------|-----------|----------|-------------|  
| Change Number | Revision Description | Pages Affected | Revision/ Version Number | Date |  
| 1 | 1st draft | All | 0.1 | 16/01/2025 |  
| | | | | |  
| | | | | |  
| | | | | |  
| | | | | |  
  
\*\*TABLE OF CONTENTS\*\*  
  
[\*\*1. Introduction 5\*\*](#introduction)  
  
> [1.1 Goal of Performance Optimization  
> 6](#goal-of-performance-optimization)  
>  
> [1.1.1 Server Loading 6](#server-loading)  
>  
> [1.1.2 Bandwidth Usage 6](#bandwidth-usage)  
>  
> [1.1.3 Better user experience 6](#better-user-experience)  
>  
> [1.2 Performance Optimisation Actions  
> 7](#performance-optimisation-actions)  
>  
> [1.3 Storage Allocation 8](#storage-allocation)  
>  
> [1.4 Required Response Time 9](#required-response-time)  
>  
> [1.4.1 Online Transaction 9](#online-transaction)  
>  
> [1.4.2 Online Reports 10](#online-reports)  
  
[\*\*2. Critical online transition timing  
11\*\*](#critical-online-transition-timing)  
  
[\*\*3. Critical Batch Cycle Timing 12\*\*](#critical-batch-cycle-timing)  
  
[\*\*4. Optimization changes 13\*\*](#optimization-changes)  
  
> [4.1 Optimization Actions 14](#optimization-actions)  
>  
> [4.1.1 Create stored procedures 14](#create-stored-procedures)  
>  
> [4.1.2 Create clustered indexes 14](#create-clustered-indexes)  
>  
> [4.1.3 Database Indexing Optimization 15](#database-indexing-optimization)  
>  
> [4.1.4 Data Archiving Strategy 15](#data-archiving-strategy)  
>  
> [4.1.5 Review Data Types 15](#review-data-types)  
  
##  
  
## 1. Introduction  
  
This document outlines the performance optimization report for the Licensing Self-Certification Portal (LSCP) system developed for the Buildings Department (BD). The optimization focuses on improving system response time and efficiency to ensure a better user experience and system stability under load.  
  
This report is based on the analysis of the database schema named `bd`, last updated on 2025/3/4 ??10:10:39.  
  
\*\*Database Overview:\*\*  
  
- \*\*Database Name:\*\* bd  
- \*\*Database Size:\*\* 88.10 MB  
- \*\*Collections:\*\* 12  
- \*\*Total Documents:\*\* 1278983  
- \*\*Total Data Size:\*\* 371.24 MB  
  
The system comprises 12 collections, with `sysfilerefs` and `adrblkfilerefs` being the largest in terms of document count and data size, contributing significantly to the total data footprint.  
  
## 1.1 Goal of Performance Optimization  
  
The primary goal of performance optimization is to enhance the system's responsiveness for users. Achieving better response times involves considering several key areas in program implementation.  
  
### 1.1.1 Server Loading  
  
Server loading is a critical factor influencing system performance. Increased server load directly impacts response times. Server load is primarily affected by:  
  
1. \*\*Number of System Users:\*\* More concurrent users increase the demand on server resources.  
2. \*\*Efficiency of Programming Code:\*\* Inefficient code consumes more server resources, leading to higher load.  
  
Optimizing programming code to minimize server resource usage is crucial for reducing server load and improving response times, especially as the number of users grows.  
  
### 1.1.2 Bandwidth Usage  
  
Bandwidth capacity is another fixed system variable. High bandwidth usage can degrade response times. Bandwidth load is influenced by:  
  
1. \*\*Number of System Users:\*\* More users accessing the system simultaneously increase bandwidth consumption.  
2. \*\*Size of Transmitting Resource:\*\* Larger data transmissions consume more bandwidth.  
  
Reducing the size of transmitted data, such as optimizing page and image sizes, can significantly decrease bandwidth usage, allowing the system to serve more users efficiently within the available bandwidth.  
  
### 1.1.3 Better User Experience  
  
The LSCP system is intended for public use and represents the Buildings Department. System performance directly impacts the department's image. Poor performance, such as slow response times or system unavailability, can negatively affect user perception and departmental reputation. Therefore, ensuring optimal performance is paramount for a positive user experience and maintaining public trust.  
  
## 1.2 Performance Optimisation Actions  
  
To enhance system performance and user experience, the following general optimization measures should be considered:  
  
- \*\*Optimize Programming and Query Logic:\*\* Refine code and database queries to reduce server processing burden. This includes efficient algorithms, optimized database queries, and minimizing unnecessary computations.  
- \*\*Optimize Page and Image Size:\*\* Reduce the size of web pages and images to minimize bandwidth consumption. Techniques include image compression, lazy loading, and efficient front-end development practices.  
- \*\*Improve Resource Retrieval Speed with Indexing and Hashing:\*\* Implement appropriate indexing strategies in the database and utilize hashing techniques for faster data retrieval.  
- \*\*Cache Frequently Used Resources:\*\* Implement caching mechanisms to store and quickly retrieve frequently accessed data, reducing database load and improving response times.  
- \*\*Pre-generate Resource for Heavy Loading Features:\*\* For features that require significant server processing, pre-generate resources or use background processing to minimize real-time server load.  
- \*\*Reduce Waiting Time of Third-Party Services:\*\* Optimize interactions with external services to minimize latency and improve overall response time.  
- \*\*Archive Expired Records:\*\* Implement a data archiving strategy to move historical or inactive data to separate storage, keeping the active database size minimal and improving query performance on current data. This is particularly relevant for large collections like `sysfilerefs` and `adrblkfilerefs`.  
- \*\*Database Indexing Optimization:\*\* Focus on creating indexes on frequently queried fields, especially in large collections like `sysfilerefs` and `adrblkfilerefs`. Consider compound indexes for common query patterns and indexes on foreign key fields for efficient joins.  
- \*\*Review Data Types:\*\* Ensure optimal data types are used for all fields. For instance, using `objectId` or integer types for IDs instead of strings can improve performance and reduce storage.  
  
## 1.3 Storage Allocation  
  
The system data and textual data will be stored in the "Database" server within the Integrated system. Data is logically organized into filegroups within the Microsoft SQL Server database.  
  
| FileGroup | TableName | TableSize |  
|-----------|-----------|-----------|  
| \*To be determined based on SQL Server configuration\* | \*To be determined based on SQL Server configuration\* | \*To be determined based on SQL Server configuration\* |  
| | | |  
| | | |  
| | | |  
| | | |  
  
The filegroup growth size is configured to be below 256 MB to align with best practices for data file management. \*(Note: Specific FileGroup and TableName allocation needs to be populated based on the actual SQL Server database configuration.)\*  
  
## 1.4 Required Response Time  
  
The required response time is categorized into Online Transactions and Online Reports, reflecting different user interaction patterns and expectations. 95% of all online functions should meet the committed response times.  
  
The committed response time is contingent upon the network health of the site, defined by the following criteria for the testing environment:  
  
- \*\*Maximum Concurrent Users:\*\* 100  
- \*\*Minimal Bandwidth:\*\* 2Mb/s per testing machine  
- \*\*Maximum Network Round-Trip Latency:\*\* 200ms to the Integrated system  
- \*\*Remote Testing Site Mark-up:\*\* 50% increase to committed response time  
  
### 1.4.1 Online Transaction  
  
Online transactions, encompassing User Account and Form/Record Management Programs, are further classified by complexity:  
  
<span class="mark">\[RY Note: Needs user input/ refer to Load Test data given by user]</span>  
  
| Transaction Complexity | Number of Concurrent Users | | | |  
|---------------------|-----------------|-----------------|-----------------|--|  
| | 40 | 80 | 100 | |  
| Simple | \< 1 sec | \< 2 sec | \< 3 sec | |  
| Medium | \< 2 sec | \< 3 sec | \< 4 sec | |  
| Complex | \< 2.5 sec | \< 3.5 sec | \< 5 sec | |  
  
Online transactions are also categorized by type:  
  
| Online transactions | Description |  
|---------------------------------|---------------------------------------|  
| Online Update Transactions | Used to update records in LSCP, for example, create supervision plan |  
| Online Enquiry Transactions | Used to retrieve records from LSCP, for example, filter site monitoring records |  
| Full-text Search | Used to search for records with given key words, for example, search assigned TCP |  
  
### 1.4.2 Online Reports  
  
Online reports, such as XXXXXXXXXXX, are designed for internal BD use. Performance estimations are based on a concurrent user count of no more than 20.  
  
| On-line Reports | Committed Response Time |  
|-----------------|-------------------------|  
| \*To be defined based on report complexity and user requirements\* | \*To be defined based on report complexity and user requirements\* |  
| | |  
  
# 2. Critical Online Transition Timing  
  
The following matrix outlines programs with their complexity and transaction types. Offline module handling is asynchronous and its data sending logic is consistent with other APIs, thus it's covered under functional testing and not separately detailed in online transition timing.  
  
<table>  
<colgroup>  
<col style="width: 8%" />  
<col style="width: 8%" />  
<col style="width: 16%" />  
<col style="width: 13%" />  
<col style="width: 12%" />  
<col style="width: 9%" />  
<col style="width: 12%" />  
<col style="width: 18%" />  
</colgroup>  
<thead>  
<tr class="header">  
<th><strong>Module ID</strong></th>  
<th><strong>Program ID</strong></th>  
<th><strong>Program Name</strong></th>  
<th><blockquote>  
<p><strong>Complexity<br />  
(Simple / Medium / Complex)</strong></p>  
</blockquote></th>  
<th><blockquote>  
<p><strong>Online Transaction Program</strong></p>  
</blockquote></th>  
<th><blockquote>  
<p><strong>Report Program</strong></p>  
</blockquote></th>  
<th><strong>Type of transactions</strong></th>  
<th><blockquote>  
<p><strong>Mobile Apps</strong></p>  
</blockquote></th>  
</tr>  
<tr class="odd">  
<th colspan="6"><strong>USER ACCOUNT PROGRAM</strong></th>  
<th></th>  
<th></th>  
</tr>  
<tr class="header">  
<th></th>  
<th></th>  
<th></th>  
<th></th>  
<th></th>  
<th></th>  
<th></th>  
<th></th>  
</tr>  
<tr class="odd">  
<th></th>  
<th></th>  
<th></th>  
<th></th>  
<th></th>  
<th></th>  
<th></th>  
<th></th>  
</tr>  
<tr class="header">  
<th></th>  
<th></th>  
<th></th>  
<th></th>  
<th></th>  
<th></th>  
<th></th>  
<th></th>  
</tr>  
<tr class="odd">  
<th></th>  
<th></th>  
<th></th>  
<th></th>  
<th></th>  
<th></th>  
<th></th>  
<th></th>  
</tr>  
<tr class="header">  
<th></th>  
<th></th>  
<th></th>  
<th></th>  
<th></th>  
<th></th>  
<th></th>  
<th></th>  
</tr>  
<tr class="odd">  
<th></th>  
<th></th>  
<th></th>  
<th></th>  
<th></th>  
<th></th>  
<th></th>  
<th></th>  
</tr>  
</tbody>  
</table>  
  
\*(Note: This section requires population with specific Module IDs, Program IDs, Program Names, Complexity, Transaction Types, and Mobile App applicability for each program.)\*  
  
# 3. Critical Batch Cycle Timing  
  
The following table lists batch programs and their cycle timings. Optimization focus is primarily on the 'generate report' module due to its potential impact on user experience.  
  
| \*\*Program ID\*\* | \*\*Program Name\*\* | \*\*Online Web\*\* | \*\*Mobile\*\* | \*\*Update Batch\*\* | \*\*Require Optimization?\*\* | \*\*Cycle Timing\*\* |  
|----------------|----------------|---------|---------|---------|---------|--------|  
| \*\*BATCH PROGRAM\*\* | | | | | | |  
| \*To be defined\* | \*To be defined\* | \*To be defined\* | \*To be defined\* | \*To be defined\* | \*To be defined\* | \*To be defined\* |  
| | | | | | | |  
| | | | | | | |  
| | | | | | | |  
| | | | | | | |  
| | | | | | | |  
| | | | | | | |  
  
\*(Note: This section needs to be populated with specific Program IDs, Program Names, Online/Mobile applicability, Batch type, Optimization requirement, and Cycle Timing for each batch program.)\*  
  
# 4. Optimization changes  
  
Optimization efforts will primarily focus on program and report performance.  
  
## 4.1 Optimization Actions  
  
### 4.1.1 Create stored procedures  
  
Reports are implemented using stored procedures, which offer performance benefits due to pre-compilation and caching. Stored procedures are optimized for searching using primary keys.  
  
### 4.1.2 Create clustered indexes  
  
Primary key constraints automatically create unique clustered indexes (if none exist), which is beneficial for query performance. Primary keys are of BIGINT data type, suitable for clustered index keys. Unique constraints by default create unique non-clustered indexes.  
  
### 4.1.3 Database Indexing Optimization  
  
Beyond clustered indexes on primary keys, consider the following indexing strategies, particularly for large collections like `sysfilerefs` and `adrblkfilerefs`:  
  
- \*\*Non-clustered Indexes:\*\* Create non-clustered indexes on frequently queried fields that are not part of the primary key.  
- \*\*Compound Indexes:\*\* For queries that frequently filter or sort on multiple fields, create compound indexes to improve query performance.  
- \*\*Indexes on Foreign Keys:\*\* Ensure indexes are in place on foreign key fields (e.g., `application`, `submissionCase`, `user` in various collections) to optimize join operations.  
- \*\*Index Analysis:\*\* Regularly analyze query patterns and database performance to identify missing or inefficient indexes and adjust indexing strategies accordingly. Tools like database profiling and query analyzers should be used.  
  
\*\*Recommendations based on Database Analysis:\*\*  
  
- \*\*`sysfilerefs` and `adrblkfilerefs` Collections:\*\* These collections are the largest and should be prioritized for indexing optimization. Focus on indexing fields frequently used in search queries and filtering, such as `createdDt`, `sysFileRefId`, `adrBlkFileRefId`, `adrBlkId`, `display`, and `frefPref`.  
- \*\*`tasks` Collection:\*\* Consider indexing fields like `application`, `submissionCase`, `status`, `taskType`, `team`, and `user` as these appear to be frequently accessed based on field analysis.  
- \*\*`applications` Collection:\*\* Given the wide variety of fields, analyze common query patterns and consider indexing fields like `ApplicationNo`, `ApplicationType`, `assignedBS`, `assignedGR`, and `createdAt`.  
  
### 4.1.4 Data Archiving Strategy  
  
Implement a data archiving strategy for large collections like `sysfilerefs` and `adrblkfilerefs`. Archiving older, less frequently accessed data will:  
  
- Reduce the size of the active database, improving query performance.  
- Decrease storage costs for active data.  
- Maintain historical data for compliance and audit purposes in a separate archive.  
  
The archiving strategy should define criteria for data aging, archiving frequency, archive storage location, and procedures for data retrieval from the archive if needed.  
  
### 4.1.5 Review Data Types  
  
Conduct a review of data types used across all collections to ensure they are optimal for performance and storage efficiency.  
  
- \*\*ObjectIds vs. Strings for IDs:\*\* Verify that `objectId` is used consistently for document IDs and relationships where appropriate, instead of strings, for better indexing and join performance.  
- \*\*Numerical Data Types:\*\* Ensure appropriate numerical data types (integer, decimal, etc.) are used for numerical fields instead of strings to improve query performance and reduce storage.  
- \*\*Date Data Types:\*\* Confirm that date fields are stored as `date` data type for efficient date-based queries and indexing.  
  
<span class="mark">\<\< End of Document\>\></span>