<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 Database Indexing 14](#database-indexing)  
>  
> [4.1.2 Query Optimization 14](#query-optimization)  
>  
> [4.1.3 Data Archiving and Management 15](#data-archiving-and-management)  
>  
> [4.2 Database Analysis and Recommendations 16](#database-analysis-and-recommendations)  
  
  
## 1. Introduction <a name="introduction"></a>  
  
The performance optimization of the system could be classified into  
optimization of Online Transaction. This report outlines the performance optimization strategies for the Licensing Self-Certification Portal, focusing on database aspects based on the provided database schema analysis.  
  
## 1.1 Goal of Performance Optimization <a name="goal-of-performance-optimization"></a>  
  
The main goal of performance optimization is to improve the response  
time of the system to users. In order to achieve better response time,  
the program implementation should take the following areas into  
consideration.  
  
### 1.1.1 Server Loading <a name="server-loading"></a>  
  
The Capacity of Server Loading is a fixed variable of a system. An  
increase in server loading would increase the response time of the  
system. Server loading is affected by  
  
1. the number of system users; and  
  
2. the efficiency of the programming code.  
  
When the number of system users increases, the server loading increases  
and hence there is less resource for running programming code for each  
system user. The response of the program would be slower as there is  
less server resource for the program to utilize.  
  
On the other hand, if the programming code could use fewer server  
resources, there is less server loading and hence the server could  
respond to users quicker and serve more users simultaneously.  
  
Therefore, response time could be reduced if the programming code could  
be optimized to use fewer server resources.  
  
### 1.1.2 Bandwidth Usage <a name="bandwidth-usage"></a>  
  
Capacity of Bandwidth Usage is a fixed variable of a system. An increase  
in bandwidth usage would decrease the response time of the system.  
Bandwidth loading is affected by  
  
1. the number of system users; and  
  
2. the size of transmitting resource.  
  
When the number of system users increases, the bandwidth usage increases  
and hence there is less bandwidth resource for transmitting data to each  
system user. The response time would be slower as there is less  
bandwidth for each user to utilize.  
  
On the other hand, if the size of transmitting data is smaller, there is  
less bandwidth usage and hence the fixed network bandwidth could serve  
more users simultaneously.  
  
Therefore, response time could be reduced if the size of the  
transmitting resource could be optimized to use less server resources.  
  
### 1.1.3 Better User Experience <a name="better-user-experience"></a>  
  
This application will be used by the public, in order words, it  
represents the department. If it performs well, the department can take  
credit from it, the image of the department may be affected if this  
application performs bad or causes other problems, for example very slow  
or no response.  
  
## 1.2 Performance Optimisation Actions <a name="performance-optimisation-actions"></a>  
  
To serve more users simultaneously with better response time, the system  
and the programs should be optimized to use the server loading and  
bandwidth resource more efficiently. The followings are the possible  
measures that could be taken generally:  
  
- Optimize the programming and query logic to reduce server loading  
 burden.  
- Optimize the page size and image size to reduce bandwidth burden.  
- Improve resource retrieval speed by indexing and hashing.  
- Cache frequently used resources.  
- Pre-generate resource that required heavy instant server loading.  
- Reduce waiting time of third-party services.  
- Archive expired records to keep the size of active datastore minimal.  
  
## 1.3 Storage Allocation <a name="storage-allocation"></a>  
  
The storage of the database data will be stored as following:  
  
i\. System data ? Store in the ?Database? server in the Integrated  
system.  
  
ii\. Textual data ? Store in the ?Database? server in the Integrated  
system.  
  
All the required system and textual data will be logically stored in collections within the database system. The following table provides an overview of collection sizes.  
  
| Collection Name | Document Count | Size (MB) |  
|-------------------|----------------|-----------|  
| sysfilerefs | 601808 | 204.70 |  
| adrblkfilerefs | 566948 | 154.89 |  
| bsblocks | 98397 | 6.40 |  
| oauthtokens | 3019 | 2.29 |  
| cases | 451 | 1.17 |  
| tasks | 5523 | 0.99 |  
| applications | 381 | 0.36 |  
| notifications | 1837 | 0.24 |  
| attachments | 370 | 0.13 |  
| users | 116 | 0.04 |  
| eminutes | 133 | 0.03 |  
| submissions | 0 | 0.00 |  
  
The storage configuration should be reviewed to ensure optimal performance, especially for the largest collections (`sysfilerefs` and `adrblkfilerefs`). Consider factors like storage engine configuration, compression, and sharding strategy if necessary for scalability.  
  
## 1.4 Required Response Time <a name="required-response-time"></a>  
  
The required response time is defined according to 2 pre-defined  
categories. They are Online Transaction and Online Report. All of the  
programs and reports that require to interact and provide immediate  
response to users are categorized. The categorized programs and reports  
should respond to the user within required response time. The required response time should be defined according to the complexity of the programs. The required response time and the complexity will be  
discussed in the coming section.  
  
For those procedures or reports that could not able to optimize to have  
immediate response, some part of the program that take a lot of  
processing time will be swapped to batch job.  
  
95% of all online functions should meet the committed response time.  
  
The committed response time is affected by the network status of the  
site. The environment of testing site should meet an agreed network  
health level. The following criteria define the agreed network health  
level.  
  
- Maximum number of Concurrent users is 100.  
- Minimal bandwidth is 2Mb/s per testing machine.  
- Maximum network round-trip latency (ping) to the Integrated system  
 is 200ms.  
- Remote testing site will have \*\*50% mark-up\*\* time to the committed  
 response time.  
  
### 1.4.1 Online Transaction <a name="online-transaction"></a>  
  
The programs in the following category are classified as online  
transaction:  
  
- User Account Program  
- Form and Record Management Program  
  
Online transactions can be classified into the following groups:  
  
<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 can also be classified as follows:  
  
| 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 <a name="online-reports"></a>  
  
The programs in the following category are classified as online reports:  
  
- XXXXXXXXXXX  
  
As the report is a standalone internal system for BD, the measurement  
would only estimate when the concurrent users are not more than 20.  
  
| On-line Reports | Committed Response Time |  
|-----------------|-------------------------|  
| | |  
| | |  
  
# 2. Critical Online Transition Timing <a name="critical-online-transition-timing"></a>  
  
The following matrix is the list of programs with the complexity and  
transaction type being marked.  
  
Offline module helps users save information in local devices when  
Internet connectivity is not available. Once Internet connectivity is  
back, it will submit the data and store it in the database. The major  
difference between offline modules with other modules is that the  
offline module is asynchronous. It will not submit immediately but will  
wait until the Internet is available. For the asynchronous handling  
logic, we included it in the functional test. On the other hand, the  
data sending logic is no different with other API, thus we would not  
have a separate section for offline module in critical 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 table requires input on specific modules and programs, their complexity, transaction types, and mobile app involvement to be populated effectively. This information is application-specific and should be filled in based on system design and functional specifications.\*  
  
# 3. Critical Batch Cycle Timing <a name="critical-batch-cycle-timing"></a>  
  
The below are the list of batch programs. The cycle timing of the batch  
is identified in the tables below.  
They are different from modules in the last section whereas they will  
run a thousand times everyday but the modules in this section probably  
run once or twice daily. Moreover, most of the modules described in this  
section are scheduled jobs running in backend, normally, the fluctuation  
of performance in these items would not affect the end user. Except, the  
generate report module, has a higher impact, thus it is the only item  
required to optimize.  
  
| \*\*Program ID\*\* | \*\*Program Name\*\* | \*\*Online Web\*\* | \*\*Mobile\*\* | \*\*Update Batch\*\* | \*\*Require Optimization?\*\* | \*\*Cycle Timing\*\* |  
|----------------|----------------|---------|---------|---------|---------|--------|  
| \*\*BATCH PROGRAM\*\* | | | | | | |  
| | | | | | | |  
| | | | | | | |  
| | | | | | | |  
| | | | | | | |  
| | | | | | | |  
| | | | | | | |  
| | | | | | | |  
  
> \*Note: This table requires input on specific batch programs, their type, optimization needs, and cycle timing. This information is application-specific and should be filled based on system design and operational requirements.\*  
  
# 4. Optimization changes <a name="optimization-changes"></a>  
  
The optimization will \*\*<u>only</u>\*\* focus the performance on programs  
and reports, with a particular emphasis on database performance based on the schema analysis.  
  
## 4.1 Optimization Actions <a name="optimization-actions"></a>  
  
Based on the database schema analysis, the following optimization actions are recommended:  
  
### 4.1.1 Database Indexing <a name="database-indexing"></a>  
  
Appropriate indexing is crucial for query performance. Based on the field analysis, consider the following:  
  
- \*\*Identify Query Patterns:\*\* Analyze application queries to understand frequently accessed fields and common query patterns for each collection.  
- \*\*Index Frequently Queried Fields:\*\* Create indexes on fields that are frequently used in `where` clauses, `sort` operations, and joins (if applicable).  
 - \*\*Potential Index Candidates:\*\*  
 - `tasks`: `application`, `submissionCase`, `taskType`, `user`, `status`, `createdAt`  
 - `applications`: `ApplicationNo`, `ApplicationType`, `assignedBS`, `assignedGR`, `createdAt`  
 - `notifications`: `task`, `user`, `notificationType`, `createdAt`  
 - `cases`: `application`, `assignedBS`, `assignedGR`, `Category`, `ReceivedDate`, `TargetReplyDate`, `createdAt`  
 - `sysfilerefs`: `sysFileRefId`, `createdDt`, `lastModifiedDt`, `frefPref`, `frefSeq`, `frefYr`  
 - `adrblkfilerefs`: `adrBlkFileRefId`, `adrBlkId`, `sysFileRefId`, `createdDt`, `lastModifiedDt`  
 - `eminutes`: `submissionCase`, `eminuteId`, `from`, `to`, `createdAt`  
 - `attachments`: `application`, `submissionCase`, `sysFileRefId`, `createdAt`, `receivedDate`, `type`, `subType`  
- \*\*Compound Indexes:\*\* For queries that filter or sort on multiple fields, consider creating compound indexes to improve efficiency.  
- \*\*Index Type Selection:\*\* Choose appropriate index types (e.g., B-tree, text, geospatial) based on the query requirements and data types.  
- \*\*Index Size Management:\*\* Monitor index sizes, especially for large collections, as excessive indexing can impact write performance and storage.  
  
### 4.1.2 Query Optimization <a name="query-optimization"></a>  
  
Efficient query design is essential for minimizing database load and improving response times.  
  
- \*\*Query Analysis:\*\* Utilize database profiling tools and query explain plans to identify slow-performing queries.  
- \*\*Optimize Query Selectivity:\*\* Ensure queries are selective and retrieve only the necessary data. Use specific filters and avoid fetching entire collections when possible.  
- \*\*Projection:\*\* Use projection to return only required fields in queries to reduce data transfer overhead.  
- \*\*Avoid `skip` and `limit` for Pagination in Large Datasets:\*\* For pagination in large collections, consider using cursor-based pagination or other techniques to avoid performance issues associated with `skip`.  
- \*\*Efficient Aggregation:\*\* Optimize aggregation pipelines for complex data processing. Use indexes to support aggregation operations where applicable.  
- \*\*Review Data Modeling:\*\* Evaluate if the data model is optimized for common query patterns. Consider embedding vs. referencing relationships based on access patterns.  
  
### 4.1.3 Data Archiving and Management <a name="data-archiving-and-management"></a>  
  
Managing data growth, especially in large collections like `sysfilerefs` and `adrblkfilerefs`, is crucial for long-term performance.  
  
- \*\*Archiving Strategy:\*\* Implement a data archiving strategy to move older, less frequently accessed data from active collections to archive storage. This can significantly reduce the size of active datasets and improve query performance. Consider archiving data based on date fields like `createdDt`, `lastModifiedDt`, or `createdAt`.  
- \*\*Data Retention Policies:\*\* Define data retention policies to manage the lifecycle of data and ensure compliance requirements are met.  
- \*\*Regular Data Cleanup:\*\* Periodically review and clean up unnecessary or redundant data to maintain database efficiency.  
- \*\*Collection Optimization:\*\* Utilize database-specific commands to optimize collection storage and defragment data files if needed.  
  
## 4.2 Database Analysis and Recommendations <a name="database-analysis-and-recommendations"></a>  
  
Based on the "Database Analysis: bd" report, the following observations and recommendations are made:  
  
- \*\*Large Collections:\*\* The `sysfilerefs` and `adrblkfilerefs` collections are significantly larger than others in terms of both document count and size. These collections should be prioritized for performance optimization efforts, including indexing, query optimization, and data archiving strategies.  
- \*\*Collection Sizes:\*\* Review the size and growth rate of all collections regularly. Monitor collections like `bsblocks`, `oauthtokens`, `cases`, and `tasks` to proactively address potential performance bottlenecks as data volume increases.  
- \*\*Empty Collection:\*\* The `submissions` collection is currently empty. Verify if this is expected. If submissions are anticipated, ensure the application logic and database schema are correctly configured for submission data.  
- \*\*Field Type Variety:\*\* Some fields exhibit multiple data types (e.g., `tasks.user`, `eminutes.from`, `eminutes.to`, `applications.assignedBS`). While flexible, this can sometimes indicate potential schema design issues or require careful index design to ensure efficient querying. Review the data model and application logic to ensure data type consistency where appropriate.  
- \*\*Average Document Sizes:\*\* Collections like `cases` and `applications` have relatively larger average document sizes. Analyze the structure of documents in these collections to identify potential areas for schema optimization or data compression if applicable.  
- \*\*Indexing Strategy:\*\* Develop a comprehensive indexing strategy based on application query patterns, focusing on the potential index candidates identified in section 4.1.1. Regularly review and adjust indexes as application usage evolves.  
- \*\*Performance Monitoring:\*\* Implement database performance monitoring tools to track key metrics like query execution time, index usage, and resource utilization. Establish performance baselines and proactively identify and address performance regressions.  
  
  
By implementing these optimization actions and continuously monitoring database performance, the Licensing Self-Certification Portal can achieve improved response times, enhanced user experience, and efficient resource utilization.  
  
<span class="mark">\<\< End of Document\>\></span>