CSC3170 Course Project Description

This is the description file for the course project of CSC3170, 2022 Fall, CUHK(SZ). It's designed to be **semi-open**, so that students can have fun with the project with their own sense of creativity and imagination with considerable degree of freedom, while not put at a loss by those requirements with a scope that is too large and vague.

So glad to tell you that, at this stage, you have finished quite a part of our course named after "Database System" (actually it'd be better to use the plural form here), and would be able to understand and start to implement most of our project. It contains four options (with two branches & three levels of difficulty in Option 1) so that students with different background and familiarity to the course can make the most suitable choice, either making **application with** a database system or having a first taste in **implementation of** it.

Note that the description file could be still gradually updated, so please pay attention to the announcement on Blackboard (BB). Hopefully the update would be incremental (but not contradictory) to the older version. For the presentation requirement and the report format, we may release other files (or update those contents in this file) when the time is proper.

Scoring & Proportion Settings

> **Note**: For convenience of description, the percentage here we mention are all of the total mark of the course, rather than of the project (unless explicitly stated to be the percentage of other ones). The grading strategy and option choice are for a group (or say, a team), i.e., such decision should be carried out by all members as a whole (rather than by a single member diversely). Hence, when we use the personal pronoun "you" (sometimes "I" when the author write in your role), we mean to refer to the members in your whole group. For some concepts mentioned in this section, you will see that they might be recalled with the details in the following sections.

- Percentage the project accounted for the total mark of the course: 15%
- Proportion Settings of different options & branches:

Project Selection	**Presentation**	**Group Type**	**Pre-allocated, α**	**Presentation, β**	**Report, γ**
Option 1, Simplified	No	Standard	0%	0%	15%
Option 1, Normal	Yes	Standard	2%	8%	5%
Option 1, Enhanced	Yes	Super	4%	6%	5%
Option 2	Yes	Standard	3%	7%	5%
Option 3	Yes	Flexible	5%	5%	5%
Option 4	Yes	Flexible	10%	5%	0%

- In Option 1, we actually provided two branches (although not listed here, and we have similar strategy for Option 2). These two branches make no difference in grading settings, except in the case that Option 1, enhanced version is picked, where the functionalities of both branches need to be implemented. The details will be introduced later in the following sections.
- Suppose the ratio of grading got in the **Presentation** part is x, $0 \leq x \leq 1$, and that in the **Report** part is y, $0 \leq y \leq 1$. Also denotes the boolean variable $i = \mathbb{1}_{\text{The project is complete with enough work}}$, i.e. $i = 1$ means that the project is at least complete (according to the requirement of the option picked) with acceptable work carried out by the team members, and contrariwise $i = 0$, which is designed to decide whether the **Pre-allocated** marks can be granted. Then, we should have the following scoring formula for the overall grade of the project g:

$$g = \alpha i + \beta x + \gamma y$$

- **All options & branches with pre-allocated points are of the presentation part**, and in such step, all groups should provide **real-time program execution** (but it's allowed to deploy your service on your PC linked with wired network cable to the campus LAN or some server you hold in WAN; you may even utilize the GitHub Action and register your own server, if you have any, as a self-hosted GitHub Action Runner), and the functionality will be the key factor in deciding the completeness of the project, as well as whether the pre-allocated points can be granted. The conclusion will be carried out by the course instructor, but the feedback of students present will also be considered -- that is to say, if the completeness of some project is commonly denied, the pre-allocated points can not be granted.
- **The pre-allocation strategy is designed for anti-involution.** Here we didn't set the bonus points as, to some extent, it's quite the same as changing the number of total marks. The option that is of more pre-allocated points should be more challenging and require more workload to finish. Although actually students are striking for the remaining points despite the pre-allocated ones, with such design, we can ensure that a project that is more challenging should always get higher grades in the case that it's of the same degree of completion compared with the other ones. There isn't closed form expression for why the number for pre-allocation is set as you

can see, but basically it depends on the workload and the difficulty (the higher the larger). You have the chance to make a choice, and you can just pick the one you think that is most beneficial.

- **For the report, we don't want to overemphasize it if you have implemented your project well.** Only for the case that you pick Option 1 and choose not to attend the presentation, we will be rigorous in the report pattern and pages -- basically you need to cover those contents of the presentation in your report if your team choose not to attend the presentation, and our expectation and requirement for your report will be a lot stricter than those cases with some presentation. Note that this is still easier than other options, since students need to spare lots of efforts for the presentation. For groups that did carry out a proper presentation, you just need to make a brief summary, and put some supplementary materials (in the appendix section), which we will generously give fair marks to, compared with those long version of report without presentation. Normally if you want to achieve a higher grade, you are suggested to pick those options and branches with the presentation part, but we will give you a chance to downgrade to non-presentation version using the requirement of Option 1 (for Option 1, normal or Option 2). For the most challenging option (Option 4), the report is even not required. For details, you may check the following sections.
- **In all, more challenging option means more possible for a high grade, but is of more potential workload.** You may choose the most suitable option considering its difficulty. Note that you don't have to pick either the easiest or the toughest option -- just the one you feel the best with is okay.
- **The condition for completeness (whether the pre-allocated points can be got)**: This will be decided by the course instructor, but also students with doubts for other group's completeness during the presentation can make an argument. The completeness condition is also briefly talked about in the description for each options, but we will not be too rigid with that.

Team-up & (Presentation) Timing Settings

- Suggested number of team members in a group:

Group Type	Suggested # of members
Standard	6
Super	8-10
Flexible	6-10

- Here we emphasize again that for Option 1 & 2, students need to form a group with exactly 6 students, except that in Option 1 the group chooses to implement the requirement of both branches -- in this case, it's required to formed a "Super Group" with at least 8 (but not exceeding 10) students, so that each one's workload will not be to too heavy.
- For Option 3 & 4, these are advanced options and the grouping choice is more flexible. You may form a group from 6 to 10 students as long as you find it suitable with the highest probability that your group will performs with the highest efficiency.
- We emphasize again that it's not the case that you pick the more challenging option or form a larger group, you'd be bound to get a higher grade. We can only say that in such a case you're provided with more chances to strike for more marks, but at the same time it'd be more risky as the expected workload is higher.
- The time budget for presentation will be proportional to (actually the same as) the number of members, that is to say, if you got 6 members in the team, you'd got 6 minutes for presentation, and for the case with 10 members, the time budget should be 10 minutes.
- For the workload considering the team size, we didn't expect it to be proportional or (affine) linear to the number of members, as the efficiency of the group members can not be simply regarded as summable. However, we would still expect that a team with more members will be more productive, which is also reflected in the presentation -- the extra time budget should be treasurable for larger group, and they should show more contents or explain more in their highlights of the project, and hence grip the chance for getting a higher mark.
- In very few cases, some standard group can be slightly more than (or less than) the suggested number. For example, we may manually merge some students that are "too shy to find a group" to some existed group, but it doesn't mean that such students will have a easier job. The time budget of such a group will also be extended as shown above, and we emphasized that behaviors like **"taking a lift" will not be acceptable** -- the project will be submitted via GitHub Classroom, and the contribution can be clearly shown in the commit records.

Intra-Group Contribution & Grading

- We expect that the members in a group will negotiate in the division of work, and that the grades will be equally distributed within a group by default.
- We didn't expect that the work within a group is divided of fully equality, as everyone has his or her own specialty and it would be fine as long as consistency in decision is reached in a group. We suggested that members of a group could be kind to each other and consider the project as a good chance for cooperation.
- However, if there is some extreme imbalance of workload and some student find that he or she has finished too much work and feels not treated well, it's applicable to email us for further investigation. It's not required to reach agreement by the absolute majority, and

it's even allowed for in-person application for the investigation. The **commit records** on GitHub is an important factor for work imbalance judgement, so you should have a good behavior in Git. The contribution is not based on the number of times of commit or the line of codes added or deleted, but the constructive submission, which is important to the functionality of the project. For a very rigorous way of writing commit messages, you may refer to conventionalcommits.org for details, and there is also some existed linter, commitlint, for such a standard (but this is not required as sometimes it's too rigorous; the key thing is to clearly and shortly describe your contribution in each submission).

- The "imbalance weight of grading based on contribution" should be provided by the student(s) who urge for the investigation mentioned above, and after reviewing, the result will be sent to the members of the group. It should include the member ID and the preferred corresponding weight. The weight might be reset, and the marks will be re-divided according to the weight.
- In the finalized report, every group should attach the actual contribution (rather than the initial work division), but this is just for backup, not for "imbalance weight of grading" -- the grades are still divided equally by default. Only if a formal application for workload invigilation and grades re-division is received will we consider adjusting the grading weight for members in the group.
- We really hope that there will not be any such application for "imbalance work invigilation", and that members within their groups can cooperate just as friends.

Timeline & Important Submissions

For the project, basically you have **around three to four weeks for substantive design and implementation**, although the actual period that you're involved with it could be as long as six weeks. In this section, we show the suggested timeline and plan, as well as the milestones for important submissions.

For the suggested plan, it's a rough and common plan almost regardless of the actual option and branch you pick, **but when you carry out your own plan after picking some option and branch, it'd better be detailed and specified**. For example, if you pick Branch 1 in Option 1, you also need to decide when to settle down the Restful API (if you adopt front-end & back-end separation and want to follow some rules on the interface, recommended to describe it with swagger), and when to implement different layers like the data persistence layer and the application layer (if you use hierarchical program design), etc. This is not required to be submitted, but could be helpful for your progress control.

You don't have to stick strictly to the timeline and plan recommended here, but you **need to be careful on the milestones**, or say time-points, shown in the following table. For details of the way and contents for important submissions, we will provide more information. For those weeks near the end of the semester, you will need to spare quite a lot of efforts on the presentation (if required in your option) and the final exam, so that you need to take your time and have a rational pace when moving the progress of the project forward.

- Suggested Timeline & Plan
  (Here "Week 1" refers to the week that the project description file is released)

Time Period	**Major Work**
Erenow	◦ To Learn Prerequisites ◦ To Find Potential Teammates
Week 1	◦ Option & Branch Selection ◦ Project Requirement Analysis ◦ Team-up & Preliminary Work Division ◦ Initial Framework & ER/Schema Design (if any)
Week 2	◦ Module Decomposition & Further Work Division ◦ Implementation of Backbone & Core Parts ◦ Extensive Functionalities Selection & Design ◦ **Important Submission 1**
Week 3	◦ Major Functionalities Implementation ◦ Test Cases Design & Partial Integration ◦ Trials in Further Implementation ◦ Phased Work Summary ◦ Initial Ver. Report (if required)
Week 4	◦ Extensive Functionalities Implementation ◦ System Test & Integration & Deployment ◦ Slides Edit & Preparation for Presentation (if required) ◦ **Important Submission 2**

Time Period	Major Work
Week 5	◦ Presentation Rehearsal (if required) ◦ Report Revision & Polish (if required) ◦ **Important Submission 3**
Week 6	◦ Materials Organization & Finalization ◦ **Important Submission 4**

- Milestones for Important Submissions:

Milestone	Submission	Corresponding Week
November 23, 5:00 p.m.	◦ Team member list & Leader Info ◦ Option & Branch Choice	Middle of Week 2
December 5, 5:00 p.m.	◦ Project Abstract ◦ Progress Report ◦ Background Specification (Opt. 2 Only) ◦ Downgrade Application (Opt. 1, normal/Opt. 2, if any)	Start of Week 4
December 12, 5:00 p.m.	◦ Finalized Slides (if required) ◦ Videos (if any) ◦ Presentation Time Slot Allocation (if required)	Start of Week 5
December 25, 11:00 a.m.	◦ Finalized Report (if required) ◦ Actual Contribution (Attached in the Report)	End of Week 6

Specially, for students picking Option 2, the *Background Specification Document* submitted in the second important submission will be reviewed together with other materials by the teaching staff, and (in usual case) if it's passed, students can move ahead normally. However, if they didn't pass the review, they will need to make an option downgrade. For Option 1 with normal difficulty, if the materials like the *Progress Report* didn't pass the checking, they will also encounter the option downgrade. Such downgrade can be also applied intuitively, if the students feel that they can not handle everything well.

For the *Project Abstract*, the option picked should be explicitly clarified, and further required contents will be introduced in the following sections, diversely depends on the option you pick. For the *Background Specification*, it's only required for Option 2, and you should imitate Option 1 and show the background and settings of your project.

For option downgrade, if the group originally pick Option 2, it can choose either the Option 1, normal version or simplified (i.e. non-presentation) version. You should also provide your choice of branch in this case. Note that the requirement is switch to that of Option 1, and those efforts will appear to be in vain. For Option 1 in normal version, the only potential downgrade is to pick the non-presentation mode, and the choice of branch can not be changed. For Option 1 in super version or other options, there is even no chance for downgrading, and hence **you need to be careful when you pick some challenging option**. These details will be mentioned in the section for each options.

Options & Branches

In this section, we shall show you the four options that you can select based on your knowledge, skill set and own relish. The first two options emphasize the application with database systems and is of **elementary level**, while the last two options focus on the implementation of database systems, which is of **advanced level**. We may read the option description carefully, and pick your choice with the instruction provided.

Option 1

I don't want to read other choices: This is a option that is rather specified in the background and the settings, but it doesn't apply strong limits in functionalities, and students can still run their imagination for specification in **function points** (for Branch 1) or in **analytical targets** (for Branch 2). It should be suitable for students who didn't know those concepts of database systems before the enrollment in this course. It's also the only option that provides two branches and three levels/versions of difficulty.

Normal Version

Background & Settings

Suppose by the year of 2035, semiconductor companies in China have made a significant breakthrough, where most chips (Integrated Circuits, ICs) with mainstream process can be produced at an amazing speed. Except for some specialized computational task that are processed on quantum computers, most tasks are still finished on general computing units, while the size of such chips has turned to a extremely tiny scale, and could be embedded to almost every common object with the communication unit for the Internet of Things (IOT). The demand for different chipsets has reached an unprecedentedly highest level, and the IC manufacture system becomes so automated (even regarded from the demand side).

The manufacture of chips has turned to a sub-contraction mode, and there are many plants that can produce quite a few types of chips, and the design of chips is largely simplified via unitization and standardized development interfaces. With the help of smart contract, high-speed internet and cloud manufacturing systems, chip consumers (with ability of basic circuit design) can send their request of chip manufacturing and even make real-time scheduling over the machines of those plants. Below are the major concepts and corresponding settings:

- **Consumer**: The one (either a real person or a company) who request some plants with some packages.
- **Package**: A package refers to the bundle of (i.e. one or more) chips that a consumer requires to finish; each package has an overall time and expense budget.
- **Plant**: A plant holds one to many machines. For simplification, these machines are of the same machine type.
- **Machine Type**: We say that a machine type is determined, if and only if for any operation, the feasibility, time, and expense given this machine to process is determined.
- **Machine**: A machine might be able to process different types of operations, but it can be only process one operation at the same time.
- **Chip Type**: To produce a chip of some specific chip type, a few operations must be processed in some partial order, i.e. an operation precedent to the other must be processed earlier. If some chip type is given, the number, precedency and operation type of its contained operations should be determined, and vice versa. For simplification, the precedent constraint can be considered as sequential, i.e. one operation can have only one precedent, and there isn't repeated operation types when processing one chip. Sometimes "Chip Type" has an alias called "Chip Model".
- **Chip**: An actual integrated circuit to be produced. Every chip can be of only one chip type. A chip can be either processed by one plant or multiple plants, but it usually (not always) requires the cooperation of different machines.
- **Operation Type**: We say that an operation type is determined, if and only if for any machine, the feasibility, time, and expense to process it is determined. Typical operation type would be like "design-import", "etch", "bond", "drill", "test" etc.
- **Operation**: For an operation, there could be multiple plants that hold the ability to process it, but in a package, it can be assigned to only one plant. One operation can be processed by one machine at the same time.
- **Processing Record**: This involves the operation that is processed and the machine that is used to process it (not violating the plant assignment). There should be the start-time, end-time and the expense. These properties might be of "planned version" and "actual version".

Suppose that you are working for an organization which provides platform for online circuit manufacture orders, and the major functionalities of such platform might be:

- Register the package information that is released by some consumer
- Allow the consumer to appoint some plant for some package manually
- The assignment and the start-time of some operation with some machine could be further set under the constraint of plant appointment
- Once some operation is successfully finished, the processing record in end-time and expense could be written back.
- The production information, like manufacture capacity of some plant, or the demand changes of some consumer within some period of time can be calculated.

These are only a small part of the potential functionalities, and you are just suggested to but not required to utilize such functionality settings. You may draw other kinds of functionalities as long as it's rational. The functionalities will be further described as **function points** in Branch 1, and **analytical targets** in Branch 2, which you should show in the abstract and recall in the presentation (or in the report for the case without presentation).

Here we provide the extensive settings, so that you may introduce some extensive functionalities; you can set the details if some description is not detailed enough. Note that it's almost impossible to consider all of these with given time for this project, so you may just consider those few parts that you are interested in:

- **Sub-contraction-Share Constraint**: For some operation type, there might be quota (by number or by expense) for different plants to get the appointment.
- **Geometrical Constraint**: Here we add the concept of *location*. The consumer and plants have the info of (one) location, and those deal for a consumer and a plant that are too far away from each other can not be made.
- **Complex Plant**: Some plant might hold multiple machine types and can be more flexible in chip processing.

- **Centralized Banking System**: *(This is highly worthed a trial for Branch 1)* Although some blockchain technology is adopted, the payment is still handled by the centralized banking system. Here we add the concept of *bank account* and *plant owner*. A plant can belong to only one plant owner, but one plant owner could have multiple plants. Both the consumer and the plant owner both has (for simplification, only one) bank account. Here you may utilize this setting for testing transactional control, failure recovery that is introduced in the second half of this course. You can also test two independent database system as the banking system is physically different from our platform. In branch 1, you may add some function point of cross-system interaction in payment: a consumer user may scan the QR-code provided by the platform for payment pointing to the service of the bank with encrypted arguments, and via redirection, the webpage can be directed to some "success/failure payment page" after the bank has processed the request.
- **Directed Acyclic Package Settings**: An operation might have multiple precedent operations, and the whole package can be described as a directed acyclic graph. This is the real case especially consider that there is a operation type called "assembly". You may find some way to speed up the reading and writing of such structure.
- **Set-up Cost for Switching & Transportation**: When a machine is to process an operation of the same type as the formerly processed one, it will not have Set-up cost in time and expense, but does contrariwise. If one chip is processed in multiple plants, it additionally require transportation cost considering the distance of their location. The chips might be transported in batch.
- **Uncertainty**: The processing time and the expense etc. might be fuzzy; the chip production requirement in one package might not be released from the very beginning, and could arrives randomly; the machine of some plant might be of some probability to break down.
- **Time-variant**: The settings like *Sub-contraction-Share Constraint*, *processing time & expense* might be time-variant even in the stochastic case, e.g. the parameter of some distribution is depended on the time (might be time periods like hour/day/week/month).

Course-relevant Jobs

1. Analyze and specify the requirements of the organization. Here the major concepts are already given, and you need to specify the functionalities based on reasonable assumption.
2. Identify the relevant entities, attributes, and relationships together with any constraints and properties. Note that it's not explicitly pointed out whether some concept is an attribute, an entity, or a relationship, and you need to make reasonable assumption by yourselves. Also note that some attribute might be derived and might not be necessary to persist.
3. Produce an E-R diagram for the database. This should act as the guidance for your schema design, and should be included in the presentation (or the report for non-presentation version).
4. Convert the E-R diagrams to relational schemas (clearly indicating the primary keys, foreign keys, functional and/or multivalued dependencies, as well as justifying that your designs are in good, normalized form; de-normalization is allowed but you need to show your reasons and concerns).
5. Populate the schemas with a reasonable amount of data. Here in this option, your don't use realistic data, but just generate them via some random number generator with your pre-assumed distributional settings. For the randomness of naming, like for operation type, you may name it after a typical one with random characters and numbers (e.g. "etch-MK28").
6. Produce sample SQL queries on these relations that are used for practical daily operations and activities, mainly based on the functionalities you've specified. You may further make such query files as "template", so that the host language (e.g. Python) can change the arguments and make them "dynamic queries". It's not recommended to use ORM framework, as you might omit the details of SQL queries and transactional control.

Branch 1

I want to wrap things up and make it a web application: It's recommended that at least two in your group knows the technology stack for web application implementation. It's a branch of "web application with database systems". Here you're viewing the functionalities you specified from a perspective of users, which are reflected as the function point.

For each **function point**, the user will interact with your application and finish some concrete task. For example, we may describe a function point as "login as a consumer user to config and release a package". You may adjust the granularity of function point so that your description is of proper length. Your application should of good human-computer interaction.

For the whole application, you should provide a complete scenario, where different users (e.g. consumers & plant owners) can go through a series of function points and achieve some goals. You should also emphasize how the database system help, like helping you to persist data, to decompose the complexity, or to reach consistency for some constraints.

To make your interaction more favorable, you may provide visualization for some function points. For example, you may utilize echarts to draw dynamic Gantt-Chart (Chip Perspective or Machine Perspective) when you are showing that there is some update to the processing record.

Branch 2

I don't like writing web pages but I'd like to do some analysis: It's recommended that at least two in your group masters statistics well and are familiar with randomness generation. It's a branch of "simulation based analysis". When we talk about "analysis", we are talking about some KPIs that we concern, and we care about "what is gonna influence it" and "how the influence is like". More specifically, we expect these things that make the influence to some metrics concerned as "decision variable" if these are controllable by the decision maker. When it turns to finding a way to "change the decision variables and make the KPIs optimal", this becomes "optimization" (you're

allowed to do that, namely "simulation based optimization", but it seems to be too far again). If we got some function to determine what decision to be made given the current state, it's called the "policy". Here we call those KPIs as **analytical targets**.

In this topic, the example KPIs can be the make-span (i.e. total processing time) and the total expense of a package, and the decision variable should be "to assign which operation to which plant/machine at what time". For the *processing record*, you may assume that the time is discretized and the time unit is 1 second. You can apply "time-indexed simulation", i.e. when every unit time is elapsed, you check whether any thing happen and write some corresponding data. For example, when an operation is ready to be assigned, you may choose the machine that can process it, and it's called the "Shortest Processing Time First (SPTF)" policy.

You may assume that the processing time or even the expense is stochastic, and with some heuristic policies (i.e. dispatch rules), how the analytical targets of make-span and total expense will be influenced. You may also set other kinds of analytical targets and do the analysis. The analysis speed in this branch is highly valued, and you should show the efforts you've made for that, like which data fields of the relational schemas should be indexed or hashed, and explain the reason. You may also utilize some built-in function of DBMS for speeding up the analysis procedure, as many DBMS dialects with extensive supports has became turing-complete.

As we have real-time execution requirement if you attend the presentation, you should ensure that some of your analytical job can be done on-site, like being calculated instantly and plotted with tool-kits like matplotlib.

Requirement for the Abstract

In the abstract, you should specify the settings you adopted, point out the branch you pick, and list the function points or analytical targets.

Requirement for the Report

Don't worry about the report, since we don't emphasize it as long as you've made a good presentation. The report is for you to fill-in some supplementary materials or additional functionalities you implemented after the presentation, and should not have any strict requirement in format. It's recommended to use markdown for writing, so that you can just focus on the contents rather than the pattern and save your time (but still you need to export it as a "pdf" file for submission).

Condition for Completeness

As long as students can cover the major functionalities (function points/analytical targets) and the program is runnable, it could be considered as **complete**.

Simplified Version

I want to lie-flat: We are afraid that at most you can adopt "pseudo-lie-flat". Here we can just provide you a chance for a non-presentation project. It's said to be "simplified" mainly because that you don't have to spare that much efforts for real-time execution of your program, and for a high quality presentation. You'd have more time for writing the report, but the requirement is a lot higher than that for the normal version: you need to cover all parts that should have been covered in the presentation, and your writing should be well-formatted with fluent English. It's recommended to use LaTex if you'd submit the report only and do not attend the presentation. You still need to make some trade off as you are "placing all eggs in a basket" considering the grading schema. We would rather say it's harder to get high marks in this version, as we favor those trying to show their work in real-time.

Enhanced Version

I am an adult -- I want both branches: Glad to see that you'd be a challenger, but not only kids need to make a choice. It's not allowed for a standard group to pick both branches, as the human resource is apparently not enough. If a group want to accept both branches, they need to form a super group of 8-10 students, and those things in the analysis part (Branch 2) should also appear in the web application part (Branch 1), i.e. there should be some real-time analysis entrance on the web-page, and you should provide some "click-a-button-to-show" procedure. You are provided more pre-allocated marks, but please be aware that the workload will be larger.

Option 2

I'd like to Do-it-yourself (DIY): This option is suitable for those students who already had or want to have an idea for the topic of the project other than that shown in Option 1. It's a good chance for designing some project that can be utilized in the future when you are seeking some jobs like data scientist or data analyst (you may search the Internet to see what kinds of project is favored).

Suppose you are engaged as a database designer for an organization to support its operations. You have first to choose and determine the exact nature of the organization. You may pick one that you have encountered in real-life or one that you are interested to work for after graduation (e.g., university, sports association, retail, and manufacturing). You have to make certain assumptions concerning the organization’s operations, as well as the underlying entities, attributes and relationships. You should state and justify these assumptions clearly, and these assumptions should be reasonable and realistic. Your project must include the following, quite the same as those mentioned in Option 1:

1. Analyze and specify the requirements of the organization. The background and settings are described in the *Background Specification Document*, and you should stick to those concepts and may recall (better reshow) them in your presentation or report.

2. Identify the relevant entities, attributes, and relationships together with any constraints and properties. Some attribute might be derived and might not be necessary to persist.
3. Produce an E-R diagram for the database. This should act as the guidance for your schema design, and should be included in the presentation or the report.
4. Convert the E-R diagrams to relational schemas (clearly indicating the primary keys, foreign keys, functional and/or multivalued dependencies, as well as justifying that your designs are in good, normalized form; de-normalization is allowed but you need to show your reasons and concerns).
5. Populate the schemas with a reasonable amount of data. You can either use realistic data or use random number generators to generate that, but you need to make sure the rationality of the data.
6. Produce sample SQL queries on these relations that are used for practical daily operations and activities, mainly based on the functionalities you've specified. You may further make such query files as "template", so that the host language (e.g. Python) can change the arguments and make them "dynamic queries". It's not recommended to use ORM framework, as you might omit the details of SQL queries and transactional control.
7. You can **EITHER** wrap your program as a web application with some function points considering the same standard mentioned in Option 1, Branch 1, **OR**:
8. Carry out some data analysis work with some analytical targets, not necessarily using simulation as long as you can find proper and realistic data, but still considering the standard of Option 1, Branch 2.

We want to emphasize that it doesn't mean that if you pick Option 2, you will get more marks than Option 1. The additional pre-allocated marks is for your additional submission of *Background Specification Document*, but the originality and novelty of the "story" you tell will also be taken into account, so that it's possible for the case where the pre-allocated marks doesn't cover your shortage of proposal. If your topic is not attractive, it's very possible that you can not get a high grade.

Condition for Completeness

Similar to that of the Option 1, as long as students can cover the major functionalities (function points/analytical targets) and the program is runnable, it could be considered as **complete**.

Option 3

I don't like to Create, Read, Update and Delete (CRUD): In this Option, you may have a first taste in implementation of a database management system (DBMS). We utilize an excellent project design of a course (not necessarily database courses) from UCB; but the point why it provides less pre-allocated marks is that this project will only implement a limited part of the DBMS. Note that for Option 3 and Option 4, they are considered as advanced options, and the teaching staff might not reply to some very detailed technical problems. You will need to go through most things by yourselves, and visit Google and Stackoverflow etc. quite frequently. **If you refer to some existed codes, you need to explicitly mark it, and show what's your own contribution.** For Option 3, it's recommended that some of your group mates should be familiar with static programming language (it's written in Java, taught in CSC1003; also helpful if you've learned C++, taught in CSC3002).

UCB CS61B, Fall 2014: Data Structures and Advanced Programming, Project #1

Project Description

> (**Copied from the project page**) This project involves writing a miniature relational database management system (DBMS) that stores tables of data, where a table consists of some number of labeled columns of information. Our system will include a very simple query language for extracting information from these tables. For the purposes of this project, we will deal only with very small databases, and therefore will not consider speed and efficiency at all.

Resource Links

- Project Description
- Project Check List
- Phase 1 Getting Started Video
- Phase 2 Getting Started Video
- Video Slides
- Project Codes

You need to finish those contents mentioned in **Your Task** described in the PDF file, and it'd be considered as **complete** (though some tiny errors are allowed when you execute your program during the presentation).

Optional/Additional Work

To make your work attractive, you may think about interesting things to do by yourself. Here are some examples:

- **Python-based Implementation**: Although Python is not a static language, it's a **strongly typed language**, which makes it possible to transfer from those Java codes to Python ones. You may use the typing library for type hinting (but remember to switch the Python version that is compatible to the version you are using when you are visiting this typing api webpage).
- **Application with Your Implementation**: You may write some small-scale application with the DBMS you just implemented. For example, you can try to "re-implement" those work in Assignment 2 -- there would be some expression in your queries that needed to be changed, as your DBMS is not "as powerful as MySQL", but it's okay as long as you'd make it work.

Things you can do are not limited above. You can run your imagination to somewhere you want to go.

Option 4

I am a true challenger: In this option, you will be able to try implementing more parts of (although still not all of) DBMS, and we utilize the materials from CMU and MIT: you can pick either of the following choice, but we note that for the CMU version, it's written in C++, while for the MIT version, it's written in Java. If your teammate has attended some contest like ICPC, this would help for handling those corner cases when you dive deep into the implementation.

MIT 6.830, Fall 2010: Database Systems

Project Description

It's consisted of five labs, including:

- Lab 1: SimpleDB
- Lab 2: SimpleDB operators
- Lab 3: SimpleDB transactions
- Lab 4: Query optimization
- Lab 5: Rollback and recovery

Condition for Completeness

You don't have to do all of these -- just pick those you are interesting to, and we are generous in grading for challengers (but the labs could have precedent constraint). If you finished three labs (okay even if runnable with tiny problems), it can be regarded as **complete** (you need to find your own ways to avoid the precedent issue when some parts are not implemented, like providing some stubs, if you want to skip some labs).

Resource Links

- Course Website

CMU 15-445/645, Fall 2021: Database Systems

Project Description

It's consisted of four labs, including:

- Lab 1: Buffer Pool Manager
- Lab 2: Hash Index
- Lab 3: Query Execution
- Lab 4: Concurrency Control

Condition for Completeness

Again, you don't have to do all of these -- pick things you like (but the labs could have precedent constraint). If you finished three labs (okay even if runnable with tiny problems), it can be regarded as **complete** (you need to find your own ways to avoid the precedent issue when some parts are not implemented, like providing some stubs, if you want to skip some labs).

Resource Links

- Project Website

Optional/Additional Work

You are **NOT** recommended to do more things over the scope in this option. The jobs are already hard, but if you still want to make some "improvement", you can do so as you wish.