**IT&C 350-001: Database Principles + Applictn**

Milestone 1: Project definition and requirements.

Due: Jan 20, 2024 at 11:59pm

|  |  |
| --- | --- |
|  |  |
| **Details** | Rationale  The first step in designing or re-designing a database is understanding and clearly articulating the purpose and requirements of the application and database. There are different types of requirements based on functionality, expected changes, scale, etc. This assignment will help you create an initial project definition, list of requirements, and basic front-end user interface that will allow you to move forward with the design. It will inevitably change, based on the recommended iterative design process, but it is important to create a solid first pass so you can get started in the right direction. You will update this document throughout the semester, and it will ultimately become part of your final project submission. This portion of the project will help you prepare for your Capstone project, which also requires teams to define a project, and develop requirements, and prototypes, albeit in more detail for a larger project.  Working as a team  With the exception of very small projects, data modeling, design, and development activities are typically performed by teams of analysts and programmers who have different knowledge and skills. By working in teams, more work can be done in a shorter period of time, and the expertise of individual team members can be utilized in task assignments.  This will be true with student teams as well. The teams should be able to accomplish more work in a shorter time by sharing the workload and benefiting from each other’s strengths. However, since the primary goal of these assignments is learning about systems analysis and design (not developing an actual information system), each team member should participate in all of the analytical activities that the team undertakes. For example, a team may agree that one member has the best understanding of ERD and may want to assign full responsibility for ERDs to that person. However, other team members would not improve their understanding of ERDs in this case. As an alternative, the team could work together initially on the ERDs; the "team expert" could draft the ERDs; and the whole team could do a "walk-through" of the ERDs.  Team members do not have to meet together all the time to do all things. You should use all the facilities of the college, including electronic communications, to help you coordinate work. A limited amount of time will be reserved in class for short team meetings and coordination. However, most teamwork will have to be done outside of the class.  Occasionally, a project team finds that one member is not pulling his or her weight. When this situation occurs in organizations, the team member's performance evaluation is affected. This will be the case in class as well. At the end of a case assignment, each team member will evaluate all team member’s contributions to the effort (in confidence). Dr. Tay will also be keeping up with each team’s activities and issues during the semester. Based on this information, an individual’s grade on a case assignment or overall case grade could be lower that the team’s grade if there has been serious “freeloading.”  Forming teams  Students will self-select their teams before the first team assignment begins. Each student should consider other team members’ willingness to meet outside of class and potential schedule conflicts before committing to a group. Once the teams are formed, members will work together on all of the case assignments. Teams should have 3 to 5 members.  Related Course Learning Outcomes   * Conceptual and Data Modeling: Apply conceptual modeling concepts using standard techniques such as Entity Relationship and Universal Modeling Language diagrams and tools. Use manual and automated techniques for mapping from conceptual to data models (e.g., ER to Relational). * Use conceptual and data models to design and implement distributed database access applications applying current languages, system technologies, and tools.   Procedures   * Develop a project definition statement, which should be a concise statement (e.g., under 30 words) that summarizes the goal of the system you are designing. For example, “Create a sports pickup game app that allows users to find, create, rank, and take attendance at informal sports games." * After selecting a project, you and your team members will identify the stakeholders of the project (e.g., those who will use it; those who will create or maintain it; others impacted by it). You will then talk to them to understand what they want out of the system. Ask questions like: “What data would you want to access? What tasks would you want to complete? What features would you expect to see?” We do not have time in this class to do a full user experience design process, so just focus on the most critical needs that they identify. * Use the information you have gathered to create high-level Requirements for the system. A later assignment will create requirements for the database. For this assignment, focus on the following types of requirements:   1. Functional Requirements describe the functionality of the product – i.e., what users can do with it. Think of the core tasks that each user of the system needs to accomplish. Then think of all of the sub-tasks involved in the bigger task.  You may want to consider the following structure for stating your functional requirements: **<user> can <perform action> under <conditions>**. The “under <conditions>” part is optional, depending on the nature of the requirement. To reduce repetition, you can organize them by the user as the examples provided in the sample report shows. Nested bullets can be used to help show what sub-requirements are related to parent requirements as shown in the sample report. Feel free to modify the structure to meet your project, so long as you achieve clarity.   2. Non-Functional Requirements describe how a system should behave and often relate to quality attributes of the system. In the context of our project, you should consider requirements related to security, portability, maintainability, reliability, performance, availability, flexibility, usability, and scalability. You don’t need one in each category, and some categories will have multiple, but you should consider all of them. These don’t need to follow the strict formula outlined for functional requirements. For example, a usability requirement could be: *The website will have a responsive design so it works on mobile phones and desktops*. A scalability requirement could be: *The database must support 3,000 concurrent users*. * Create a low-fidelity prototype (e.g., paper prototype) of the major front-end interface pages or views (if creating a web or mobile app). These should include enough detail that you will know what database "views" (and ultimately REST API endpoints) you will need to create to support the app. For this class, the look and feel of the prototype are not a priority; however, functionality is critical. You may want to look at the [React-Admin](https://marmelab.com/react-admin/Readme.html) documentation for examples of their interface elements that we will introduce later in the semester. For now, just focus on what will be on each page and what functionality you'll want on each page (e.g., filtering, sorting). * Use the FinalReportTemplate.docx and submit the information. This same template will be updated and added to throughout the semester. Just ignore sections that aren't part of this assignment. Make sure you save a copy where your entire team can access and modify your updated document (e.g., on BYU Box). |

**IT&C 350-001: Database Principles + Applictn**

Milestone 2: ER diagram, Schema Diagram, and Business Rules

Due: Feb 10, 2024 at 11:59pm

|  |  |
| --- | --- |
|  |  |
| **Details** | Rationale  Once the app requirements for a project are gathered, it is important to create requirements for the database that will support the app. This requires creating a conceptual model of the data that fleshes out the details of all of data fields that will be stored, information about the data fields (e.g., data types and other metadata), the relationships between fields, and business rules that determine if data is valid or makes sense in the context of the app. This assignment will help you present database requirements and business logic in three complementary ways: 1) an Entity Relationship (or ER) Model visualization, 2) a Relational Schema visualization showing the logical model of the SQL database, 3) a list of business rules that are not captured in those other forms. These will form the foundation for the database you will build in the next group assignment.  Related Course Learning Outcomes   * Conceptual and Data Modeling: Apply conceptual modeling concepts using standard techniques such as Entity Relationship and Universal Modeling Language diagrams and tools. Use manual and automated techniques for mapping from conceptual to data models (e.g., ER to Relational). * Use conceptual and data models to design and implement distributed database access applications applying current languages, system technologies, and tools.   Procedures   * CREATE ER DIAGRAM   + Using your App Requirements and front-end prototype, generate a list of all of the fields you will need to support your requirements. What data will you need to collect that will be shown to the user? What data do you need to collect that may not be shown to the user (e.g., timestamps for later analysis; IP addresses…)? Separate them into **core data fields** and **derived fields**(e.g., calculated and/or composite fields), which will not be part of your core fields, but should be listed separately. Show them to your potential users or database designers (e.g., classmates) and get feedback from others who may help you think of fields you haven’t yet thought of. These may be other classmates, the TAs, or potential users.   + Identify the key entities (which will become tables) and associated attributes (which will become fields). Entities should typically have a single focal subject that may be a noun (e.g., Users, Classes, Places, Objects) or a verb (e.g., Purchases, Transactions, Events). Attributes should typically be assigned to just one entity, though foreign keys may be used in a secondary table (e.g., a userid in a Purchases table).   + Review each field in each table and consider refining or removing composite and multipart fields. Make sure you don’t have unnecessary duplication of fields. Make sure the field names are meaningful and consistently applied throughout.   + Identify a primary key for each table, as well as candidate keys. Also label foreign keys when appropriate.   + Identify the types of relationships between the entities (e.g., tables), and their attributes. And give them names. Identify their cardinality. Identify if they are binary, unary, etc.   + Create an Entity Relational Model using the tool found here: <https://erdplus.com/> Make sure you create a user account so that your progress isn’t lost. You will be updating this in the future. This will capture the entities, associated attributes, and their relationships to other entities. Make sure you include all the details for each field and entity (e.g., check the appropriate boxes to indicate if a field is unique, multivalued, optional, composite, derived, etc.; and if each entity is regular, associative, or weak).   + Add the **ER diagram** to the Final Report write-up. * CREATE RELATIONAL SCHEMA DIAGRAM   + Start by converting your ER diagram into a Relational Schema diagram using <https://erdplus.com/>. This will give you a starting point, but may help you realize problems with your original ER diagram that you'll want to address, before re-converting it again. If you categorized all of your keys, relationship types, etc. properly, then this conversion should get you very close to what you'll need.   + Look at the editor and make sure you change the "data type" for each field to be an appropriate one. This is important to get right, since this tool will automatically generate the SQL queries that will be used to build your database. Notice you can also reorder the fields in a table.   + Add the **Relational Schema diagram** to the Final Report write-up. * CREATE BUSINESS RULES   + Consider what constraints you should place on users or your database fields. What data or actions should be allowed or denied? Under what circumstances? Go through each low-fidelity mockup screen, as well as each field in each table and each relationship to make sure you consider all key constraints. These will become business rules. Consider the following types of constraints:     1. Who can perform which actions? (e.g., who can delete; who can add to…)     2. Degree of participation (e.g., is there a minimum or maximum number of relationships?)     3. Is there a default value? What should it be?     4. What limitations should there be for certain data fields? Length requirements?     5. What values should never occur (e.g., creating a new pickup game before the current time)?   + Add the **business rules** you come up with to the Final Report section titled Business Rules. You may want to organize your rules (which can be bullet points) into sub-sections (e.g., field rules; relational rules; role-specific rules; application rules). Create a list of business rules that should include a clear and specific description of the rule. An example field rule would be: A username should be at least 8 characters long and only include letters and numbers. An example relational rule would be: A class must have at least 5 students in it. An example of a role-specific rule might be: Only administrators can delete other user accounts. An example of an application rule (enforced in code, not the logical database design) would be: No credit check should be performed on return visitors. (or) The total money spent should not exceed the money available.   + Update your ER diagram and relational schema diagram to reflect any of the rules that can be reflected in them, but were not previously (e.g., required versus optional values; degree of participation…). Not all of your rules can be enforced in the database schema itself; some will be through validation code that you will write or stored procedures that are triggered. This will be done later. Update your Field Name descriptions from your prior assignment to reflect any field-level business rules that were not included originally (e.g., in the Data Type, Key? Optional? Restrictions, and Notes columns).   + Submit the updated FinalReportTemplate.docx, including all sections from the prior assignment after updating them to reflect any changes you have made. It is meant to be a living document that is iteratively improved throughout the semester.   See [Appendix C: Design Guidelines in *Database Design for Mere Mortals third edition*](http://etutorials.org/SQL/Database+design+for+mere+mortals/Part+IV+Appendixes/Appendix+C.+Design+Guidelines/), for guidelines to review for each field, primary key, foreign key, tables, field descriptions, table descriptions, field names, |

**IT&C 350-001: Database Principles + Applictn**

Milestone 3: Update Schema and ERD Diagrams, start Git repo, and Build Database with dummy data

Due: Mar 2, 2024 at 11:59pm

|  |  |
| --- | --- |
|  |  |
| **Details** | Rationale  The purpose of this part of the project is to set up your initial database. We will use PostgreSQL Server, a popular open-source database server that is flexible enough that it should meet the needs of a wide variety of projects. You will also be required to set up a GitHub project as a team and use it to collaborate throughout the semester on the project.  Related Course Learning Outcomes   * Use conceptual and data models to design and implement distributed database access applications applying current languages, system technologies, and tools. * Integrate, deploy, and demonstrate operational applications on database system infrastructure. Includes use of DDL, DML, programming interfaces, and UI implementation. [first part of this outcome] * Learn to collaborate on a project using a content repository.   Procedures   * Download and install the most recent version of MySQL Server on the local machine of at least one group member. * Export the SQL schema queries that will build the database structure from the [https://erdplus.com/ (Links to an external site.)](https://erdplus.com/) website related to your existing design. * Make sure primary keys in your entities use SERIAL as column types so that they're auto-generated and auto-incremented. (This doesn't apply to entities that have primary composite keys) * Make sure primary keys are composite for many-to-many relationships. * Make any updates you need to enforce your business rules using CHECKs, foreign key constraints, etc. * Populate the data with dummy data, so you can test queries and visualize the dataset. * Create an account on GitHub if you don't have one. Create a new repository and have all team members join it and make a change to demonstrate that they can contribute. * Migrations:   + Create at least one INSERT script (e.g., to populate the database with dummy data) and one DELETE script (e.g., to depopulate the same data). These are called data migrations.   + In the future, each change you make will be done via ALTER and ROLLBACK submissions. Later in the semester when you realize you have to modify the database structure, create ALTER scripts (e.g., to add columns) and a corresponding ROLLBACK script. Make sure you name them starting with numbers so you can keep track of the order. These are called schema migrations. * Document the information about connecting to GitHub in your Final Report Template. * Save the initial schema SQL files that build your database to the GitHub repo. This will serve as a backup. Make sure you do not share any confidential information, such as authentication information, publicly on GitHub. * Document the information about connecting to GitHub in your Final Report Template. * Document information about all of the software you have used and how to connect to your database, run the schema script, etc. * Please add the instructor and TAs to your github account if it is private.     Demo files for INSERT scripts and table creation can be found at <https://github.com/OrangeySnicket/350DemoFiles>    Submit your Project Documentation, updated with new diagrams and SQL code, for points. |

**IT&C 350-001: Database Principles + Applictn**

Milestone 4: Build Database Views, Forms, Reports

Due: Mar 23, 2024 at 11:59pm

|  |  |
| --- | --- |
|  |  |
| **Details** | Rationale  The purpose of this part of the project is to build the views that correspond with your API endpoints. This requires you to think about not just how to store the data, but how to access it from the system.  Related Course Learning Outcomes   * Use conceptual and data models to design and implement distributed database access applications applying current languages, system technologies and tools. * Integrate, deploy, and demonstrate operational applications on database system infrastructure. Includes use of DDL, DML, programming interfaces and UI implementation.   Procedures   * Define what data will be needed for each of your application pages (relying on your low fidelity prototype pages). These will be sent via a REST API that will be built in the next lab. * CREATE VIEW statements that generate the data associated with the REST API endpoints, i.e., Forms for updating key tables, Forms for entering and maintain * Save the VIEW statements to your GitHub repo. * Document them in your Final Report Template and update any prior information based on any changes you have made to the database. |

**IT&C 350-001: Database Principles + Applictn**

Milestone 5: Build REST API and Front end

Due: Apr 13, 2024 at 11:59pm

|  |  |  |
| --- | --- | --- |
|  |  |  |
| **Details** |  | Rationale  The purpose of this part of the project is to translate your database Views into a REST API. You will do this using a tool that helps automate this process. You will also make sure to use best practices for securing your API.  Related Course Learning Outcomes   * Use conceptual and data models to design and implement distributed database access applications applying current languages, system technologies, and tools. * Integrate, deploy, and demonstrate operational applications on database system infrastructure. Includes use of DDL, DML, programming interfaces, and UI implementation. [first part of this outcome] * Learn to collaborate on a project using a content repository.   Procedures   * Follow the instructions in the <https://blog.dreamfactory.com/create-a-mysql-rest-api-in-minutes-using-dreamfactory/> * This portion of the project will be more flexible than other projects. You can use whatever front-end you want to use whether that is a website, mobile app, or other interface, so long as it meets your vision for the system. Don't worry about the look and feel as much as the core functionality. * You can use tools like VueJs that were introduced in IT&C 210 (or the example VueJS project shown below) or [React-Admin](https://marmelab.com/react-admin/Readme.html), a tool designed to provide an administrative front-end to REST APIs, other technologies that you may be familiar with from prior classes. * Make sure you use GitHub while working on your code. All students on a team should be making changes. * Document your code in the Final Report Template. * Document everything in the Final Report Template. Make sure to accurately explain the security of the system (e.g., how it handles SQL injection).   Sample Reference Project   * A sample VueJS project is available here: <https://github.com/aatishnn/it350_sample/tree/master/frontend> . * This utilizes the same database example that we used for the REST API tutorials. * Watch these videos to see a demo of the front and some walkthrough of the code: * If you use this code, we strongly encourage you to use the instructions provided in the README.md file to set up an instance of this demo database and the front end. Try it out before modifying this for your own project.   You do not have to use any specific technology for the front end. We suggest Vue or React but you are free to implement UI using anything. It doesn't even have to be a web app. It could be something like a mobile application. |

**IT&C 350-001: Database Principles + Applictn**

Milestone 6: Final Report (with all updates)

Due: Apr 13, 2024 at 11:59pm

|  |  |
| --- | --- |
|  |  |
| **Details** | Rationale  The purpose of the final report is to improve your technical writing skills, which are essential to success in your career.  Related Course Learning Outcomes   * Define and apply the concepts and terminology of Information Management. * Improve technical writing, particularly documentation.   Procedures  The final report should include all of the sections from your prior group assignments and be based on the Final Report Template you have been using. The content should be mostly finished since you have been updating it throughout the semester. For this assignment, you will focus on improving the written quality of your work. It should be the genre of a professional documentation write-up. Act as if this is a real paper documenting what you have done so that others could pick it up and take over the project if needed. It should include a cover page, and table of contents, and then use sub-headings as appropriate for the various sections and subsections. Make sure it is up-to-date and reflects what was accomplished. If you have things that you were not able to implement, then put them down as "next steps" for implementation. Make sure your database schemas match up with what was implemented.  Below are some tips on writing based on years of grading student technical writing. You will be graded based on these:   1. All figures and images need to be labeled and captioned. When referring to them in your writing, use the label (e.g., “Figure 1” or “UML 2”). The caption should briefly inform the reader of what the image is. 2. Do not refer to your work as a “Lab Write-up,” or as a lab in general. Give your work/project a name, and use that. Write this as though you were writing documentation for a professional situation, not a class. 3. Do not write in the first person. I.e., do not use the pronouns *I* or *we*, or any of their corresponding constituents: *me, my, our, us.*This is not a narrative document. Your purpose is not to tell your story of how you did the lab; your purpose is to describe and explain the work itself. For example, don’t say “I installed the server but it didn’t work so I had to fix the network parameters.”; Instead say “The network parameters in the init.d file are set to “open” so the server installation process can succeed.” The more you avoid narration, the easier it will be to also avoid writing in the first person. 4. Use headings for your sections. It seems obvious, but make sure the headings are visually distinct – large font, bold text. Headings are like chapter titles in a book; they tell your reader where he/she is in the document. Use sub-headers as appropriate. 5. I know Microsoft Word can be a huge annoyance when it comes to formatting, but do your best to keep the document visually organized and tidy. When it comes to formal writing, presentation is just as important as the content of the document itself. 6. Be clear and accurate    * Avoid vague terms like “it” when stating what “it” is would be clearer; be specific and detailed.    * Make sure to introduce terminology that may not be obvious to your audience.    * Make sure your statements are not ambiguous (e.g., “clicking on the ASP.NET ‘submit’ calls the C# ImageFilter function, which applies the filter chosen by…” is less vague and potentially ambiguous than “clicking the button runs the filter”).    * Make sure your statements are accurate (e.g., does the code or the database check for the business rule?)  * Use correct style, grammar, and spelling   + Avoid grammatical and spelling errors (e.g., changing tenses within a section; plural/singular agreement problems such as “the [singular]*user* did this, and then [the plural] *they* did that”)   + Code snippets should be in a different font than the explanatory text. Use a monospace font (e.g., Courier New) that clearly differentiates between otherwise hard-to distinguish letters: l, 1, I, (l, 1, I). |

**IT&C 350 - Group Evaluation**

**This assessment is due on the last day of the semester.**

**YOUR NAME: \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_**

Please give your assessment of each team member's participation in and contribution to the team's efforts in the following areas. Include yourself in this evaluation. Your evaluation is strictly confidential. I will use all evaluations by team members, along with my own observations, to determine individual grades.

For questions 1-4, use the following scale on the chart below.

**[1]** Took major responsibility for this aspect of the work

**[2]** Made a fair contribution to this aspect of the work

**[3]** Did not contribute to this aspect of the work

For question 5, if one person did not do his or her fair share, and you think they deserve a lower grade, write “lower.” If you think that, overall, everyone's contribution was about equal, put "same grade" for each name. **Include yourself in the evaluation**. These are recommendations that give me some insight into the team's work. I will make the final grade determination.

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
|  | **Team member:** | **Team member:** | **Team member:** | **Team member:** | **Team member:** |
| 1. To what degree did each team member participate in and contribute to project management tasks? |  |  |  |  |  |
| 2. To what degree did each team member participate in and contribute to applying the analytical methods in this case? |  |  |  |  |  |
| 3. To what degree did each team member participate in preparing the written assignments? |  |  |  |  |  |
| 4. To what degree did each team member participate in and contribute to MySQL and applications? |  |  |  |  |  |
| 5. Considering each team member’s participation in and contribution to the team's effort, should he or she get a higher, lower, or the same grade as the team for this assignment? |  |  |  |  |  |

**Comments:**