**School Management System (SMS)**

**App**

**Module 3 – Payment Management**



**CMPS 350 Project Phase 1 – WebApp UI Design and Implementation**

|  |  |
| --- | --- |
| **Group Id:** | G12 |
| **Group Members:** | Huda  Alanoud  Batoul |

**Grading Rubric - In the Functionality column please specify either: *Working (completed x%)*, *Not Working (completed x%)* or *Not done*.**

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
|  | Criteria | % | Functionality\* | Quality of the implementation | Grade |
| 1) Design the app Web UI and navigation (10%).  2) Implement the app Web UI and navigation using HTML, CSS and JavaScript (30%)  3) Implement the client-side data access repositories to read/write the app data from/to IndexedDB (40%) | | **80%** | Working (100%) |  | **62** |
|  | Login | 5 | Working (100%) | Good but authenticate method poorly done as it reads ALL users from DB. It should only return the user matching the username (and the password).  After login it redirects to index.html but page not found! Then has to rerun the app. | 4.5 |
|  | Setup and maintain the free structure | ~~10~~ | ~~Working (100%)~~ | Not done. Screen below of a static page could NOT be found in the submission. | 0 |
|  | Get pending payments | 9 | Working (100%) | Good work.  Must should studentId as it is an important identifier. | 9 |
|  | Make payment | 9 | Working (100%) | - Storing parent\_name, parent\_email,  student\_name and student\_grade with the payment is unneeded data duplication. StudentId is enough. Then you can get the student details from the students collection.  You could keep it simple by only allowing to make full payment of pending fees. | 8 |
|  | Get payments history | 8 | Working (100%) | Ok | 8 |
|  | Pending payments report | 8 | Working (100%) | Ok | 8 |
|  | Received payments report | 8 | Working (100%) | OK | 8 |
|  | Request bus service | 8 | Working (100%) | Should allow selecting a student if they have rejected bus requests.  Unnecessary data duplication, "student\_name", "parent\_name", "parent\_email", "student\_grade" NOT needed. StudentId is enough then you can get the student details from other collections.  Also, it could be nice to allow the parent to select either one-way to 2-ways service:   * Home to school * School to home * Two ways | 6 |
|  | Approve / reject bus service request | 9 | Working (100%) | Accept/Reject enough no need for add comment. When Accept/Reject is saved then any comment entered will also be saved. | 8 |
|  | Get status of bus service request | 8 | Working (100%) | Ok | 8 |
|  | Generate due payments | 9 | Working (100%) | Wrong implementation due to misunderstood of the requirements.  The requirement is that the principal can select a semester such as Fall 2021 and a due date set by default **today's date + 7 days**, then the system should generate a due payment for each student and add it to the payments object store. The due payment should include the amount each student need to pay for *Tuition Fee* & *Transport Fee* (if they use the bus service) based on the school fee structure and the student's grade level. | 3 |
|  | Overall App design and navigation  Code Quality | 9 | Working (100%) And Kept simple. | * Avoid inline style. Put all styles in external css files. * Everywhere change ‘principle’ to ‘principal’. * Everywhere change duoDate to dueDate. Duo should be due. * All data filtering should be done in the DB. NOT read all then filter on the client. * It will be easier and more modular to have multiple pages instead of putting all in index.html. * Too much code duplication. Same functionally should be coded once and reused by any user. * Must use IndexedDB instead of LocalStorage. * Avoid hardcoding such as payment methods and fee types. * See further comments added to the testing screenshots. | 7 |
| **4) Application Design:** Entities Class Diagram and the Repositories Class diagram. | | **8** | Completed | Entities ok but repositories poorly done. See comments added to the diagram. | 6 |
| 5) Create test data JSON files for your module entities | | 6 | Working (100%) | - Make separate students.json to init students. Students should be in their own collection as they are a core entity heavily used in everywhere in the app.  - Storing parent\_name, parent\_email,  student\_name and student\_grade with the payment and bus request is unneeded data duplication. StudentId is enough. Then you can get the student details from the students collection. | 5 |
| **6) Testing documentation** using screen shots illustrating the testing results.  - Discussion of the project contribution of each team member. | | 6 |  | Excellent | 6 |
| **Total** | | 100 |  | Very good work but code duplication, did not use IndexedDB and some issues discussed above. | 79 |
| Copying and/or plagiarism or not being able to explain or answer questions about the implementation | | -  100 |  |  |  |

**\* Possible grading for functionality** - ***Working*** (get 60% of the assigned grade), ***Not*** ***working*** (lose 40% of assigned grade and ***Not done*** (get 0). The remaining grade is assigned to the quality of the implementation.

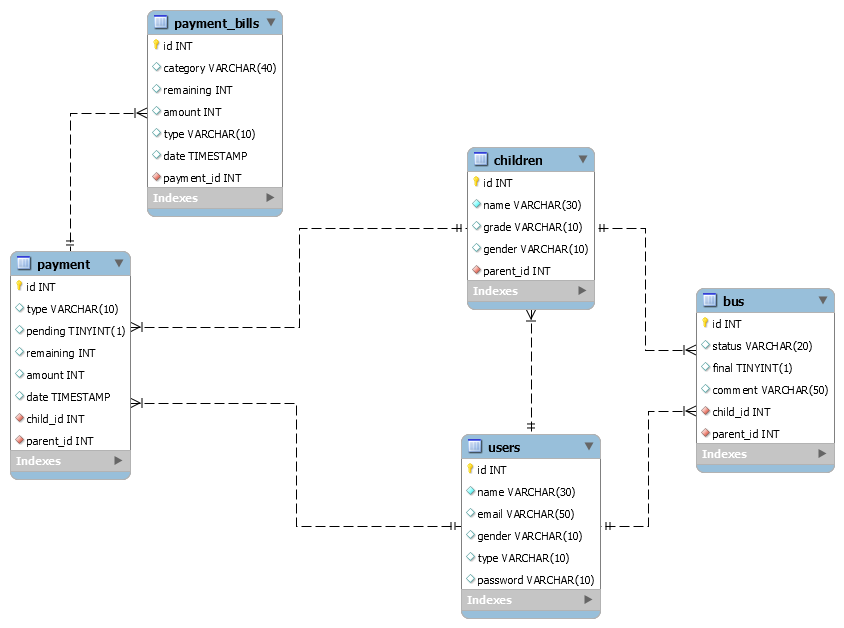
In case your implementation is not working then 40% of the grade will be lost and the remaining 60% will be determined based on of the code quality and how close your solution to the working implementation.

Solution quality also includes meaningful naming of identifiers (according to Android naming conventions), no redundant code, simple and efficient design, clean implementation without unnecessary files/code, use of comments where necessary, proper code formatting and indentation.

**Marks will be reduced** for **code duplication**, poor/inefficient coding practices, poor naming of identifiers, unclean/untidy submission, and unnecessary complex/poor user interface design.

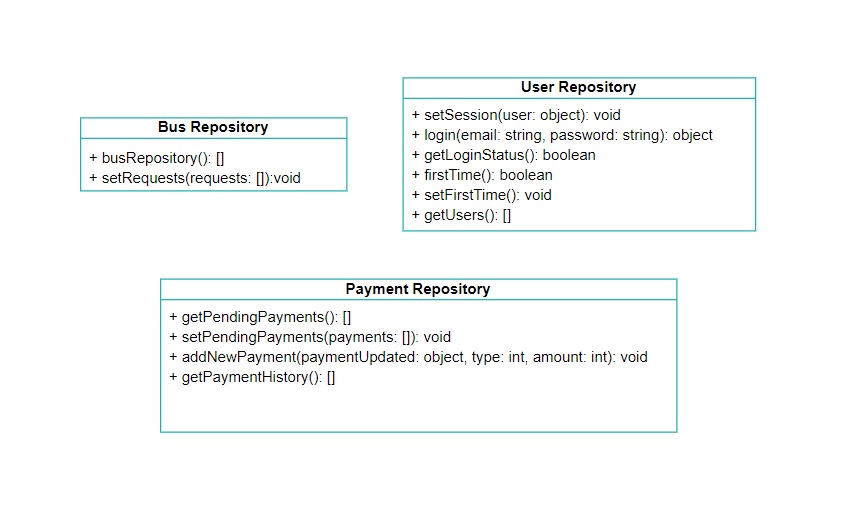
# Application Design

# Entities Class Diagram

2



# Repositories Class Diagram





# App Testing

# Login

# Setup and maintain the free structure

# Table Description automatically generated

# Get pending payments



# Make payment



# Get payments history

# Pending payments report

# Received payments report



# Request bus service



# Approve / reject bus service request



# Get status of bus service request

# Generate due payments

# Summary of team member contributions

During the start of the project, we started by setting up the authentication system, so we set the user repository and its functionalities, after that we started designing the login page and its js functionalities. After finishing with the authentication part we started working on the principal functionalities that are common with the parent, so we worked on pending payments, and bus service for those parts we designed a repository for each one. Using the same procedure we won't one undertook the designing part, and then added the functional part. Being done from the pending payments and bus service for the principal, we headed on to the parent and completed his functionalities as a whole using the same structure as above. Being completed with the parent, we started working on the report system for the rest of principle functionalities. We had the structure setup ( same as principal pending payments ) so we added and improved the functionalities by adding total and filtering options for the rest of functionalities. After being done with that, the code was complex to read being in a single file, so we changed and put each functionality in its own file.  
After that we added the data loading mechanism from json files. Due to miss understanding, the duo payments generator functionality was incorrect, so we had to update it and along the way we found and fixed lots of bugs that we found.  
In addition to all this, all the work done, was made with the whole team present...