Milestone 1: Project Proposal

Authors: Oleksandr Anyshchenko, Ziqi Gu, Jiaqi Wang, Ryan Opande
Date due: October 05, 11:59pm

About the Project: Inventory Management and Checkout System

This project will deliver a web-based application, Inventory Management System, that lets admins create an inventory, organized by categories, staff process loans, and borrowers browse and borrow items. The backend is a relational database (MariaDB/MySQL) modeling users, category hierarchies, items, physical assets, storage locations, and loan transactions.

The application will allow for various workflows:

- Admins manage catalogs (categories/items/assets/locations) and users (including bans/activation and soft-deleting items via an "active" flag)
- Staff review and process loans (approve/decline, check-out/in).
- Borrowers search, add assets to a cart (up to a configurable limit), and submit loan requests.

Implementation outline:

- Backend: RESTful services (any modern framework or custom implementation) encapsulating business rules (availability, renewals, overdue detection).
- Database: MariaDB/MySQL enforcing keys, constraints, and transactional integrity.
- Frontend: Responsive UI for browsing, cart, and loan processing.

Business Logic, Requirement Specification, Assuptions

In the Asset Lending System, the database manages **users**, **category hierarchies**, **items**, physical **assets**, **storage locations**, **loans**, and **loan details**.

- Each **User** is uniquely identified by *u_id* and has attributes *u_fname*, *u_lname*, *u_email*, *u_role*, and *u_active*. Admins can activate/ban users; banned users cannot access the system.
- Each **Category** is uniquely identified by *cat_id*, with *cat_name* and optional *cat_parent_id* to form a tree. A category can have zero or many child categories; a child has exactly one parent. Since a category can have many child categories, the **categories_children** table explicitly records parent–child pairs to facilitate hierarchy.
- Each **Item** is uniquely identified by *it_id* and includes *it_name*, *it_description*, *it_max_time_out*, *it_active*, and *it_renewable*, and a reference to the category it belongs to through *cat_id*. Admins may "shallow delete" by setting *it_active* = false.
- Each **Asset** is uniquely identified by *a_id* and includes *a_status*, *a_condition*, *loc_id* (nullable), and *it_id* referencing the catalog item it instantiates. An item can have multiple assets; each asset belongs to exactly one item. A location can store multiple assets; an asset is stored in exactly one location or virtually (when loc_id is NULL). (At least one asset should exist per defined location.)
- Each **Loan** is uniquely identified by *l_id* and records *u_id* (borrower), *l_status*, *l_checked_out_at*, *l_due_at*, optional *l_checked_in_at*, and *l_notes*. A user can have zero or many loans; each loan belongs to one user.
- A **Loan** may include multiple assets, and an **Asset** can appear in multiple loans over time. Loan_Details (l_id,a_id) is the associative entity linking loans and assets (many-to-many).

Rules:

- Checkout: An asset may be checked out only if *a_status* = 'Available'. On approval/checkout, create a loans record and corresponding loan_details rows; set each asset's a_status = 'CheckedOut'.
- Check-in: When the loan is returned, set the loan's *l_checked_in_at*, and set the asset's *a_status* = 'Available'.
- Overdue logic: An asset (within a loan) is overdue if current time is later than loans.l_due_at and its loan_details.checked_in_at is NULL.
- Attributes such as *l_due_at*, *l_checked_out_at,* and other time-related attributes will have the domain type TIMESTAMP, which will allow for the recording of the time of day and date, allowing same-day loan check-out and check-in.
- Since items might have a different *it_max_time_out*, the loan's *l_due_at* will be determined by the minimum *l_due_at* of all the assets in the loan.

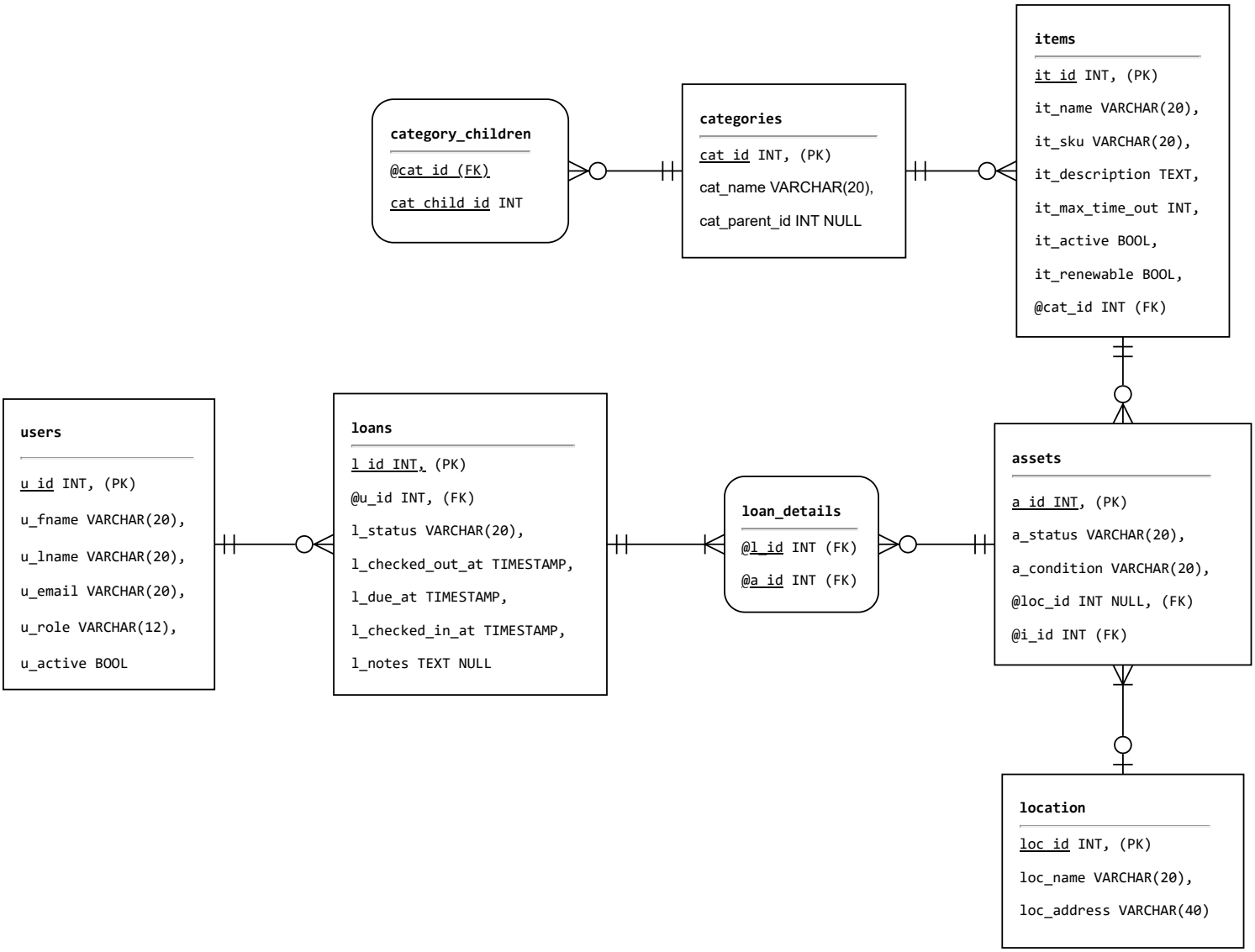
Roles & Permissions:

- Admins: create/edit categories, items, assets, locations; manage users (ban/unban); edit any record.
- Staff: process loans (approve/decline; check-out/in).
- Borrowers: browse, add assets to a cart (up to a configurable limit), request loans, and return assets.
- Cart behavior: Borrowers can stage multiple assets in a UI cart up to a system-configured maximum before submitting a loan request.

This specification ensures normalized relationships, enforces referential integrity with clear foreign keys, and encodes domain rules (availability, overdue, role permissions) in both schema constraints and application logic.

Assumptions: The system runs on a properly configured MariaDB/MySQL instance with enforced PK/FK and uniqueness constraints; the category tree is well-formed and items attach only to leaf categories. Role-based access control is enforced exactly as specified (admins manage catalogs/users, staff process loans, borrowers request/return), and banned/inactive users cannot authenticate. Server clocks are synchronized and all time fields use TIMESTAMP so overdue detection is reliable; a loan's due_at is computed from the minimum allowed duration across its assets (it_max_time_out). Asset lifecycle is transactional: checkout only when a_status='Available', checkout/check-in atomically update loans and loan_details (per-asset check-ins for overdue logic), and the cart capacity is a configurable limit.

Entity Relationship Diagram (ERD/ERM)



Relational Model / Schema Statements

```
users (
      u_id INT,
      u_fname VARCHAR(20),
      u_lname VARCHAR(20),
      u_email VARCHAR(20),
      u_role VARCHAR(12),
      u_active BOOL
);

categories (
      cat_id INT,
      cat_name VARCHAR(20),
      cat_parent_id INT NULL,
);

categories_children (
      @cat_id,
      cat_child_id INT
);
```

```
items (
      it_id INT,
      it_name VARCHAR(20),
      it_sku VARCHAR(20),
      it_description TEXT,
      it_max_time_out INT,
      it_active BOOL,
      it_renewable BOOL,
      @cat_id INT
);

locations (
      loc_id INT,
      loc_name VARCHAR(20),
      loc_address VARCHAR(40)
);
```

```
assets (
      a_id INT,
      a_status VARCHAR(20),
      a_condition VARCHAR(20),
      @loc_id INT,
      @it_id INT
);

loans (
      l_id INT,
      @u_id INT,
      l_status VARCHAR(20),
      l_checked_out_at TIMESTAMP,
      l_due_at TIMESTAMP,
      l_checked_in_at TIMESTAMP,
      l_notes TEXT NULL
);

loan_details (
      @l_id INT,
      @a_id INT
);
```