PROJECT REPORT

Group Name: **Venkataraman SundaramSShankara RamanS**

Member 1: Sai Rashwant Venkataraman Sundaram

Member 2: Swetha Shankara Raman

README:

Demonstration:

https://drive.google.com/file/d/1z7wz0eMnPJf23wp0Dye4uVpBfyrJ7zmn/view?usp=sharing

Project Overview

RentHive is a web application designed to facilitate property rental management, allowing owners to list properties and tenants to find suitable rentals.

Prerequisites

To build and run the RentHive project, ensure you have the following installed on your computer:

1. **Python 3.7 or higher**
    - Download: Python Download Page
    - Installation Directory: **C:\Python37** (or your preferred directory)
2. **MySQL Server**
    - Download: MySQL Community Server Download Page
    - Installation Directory: **C:\Program Files\MySQL\MySQL Server.**
3. **MySQL Workbench**
    - Download: MySQL Workbench Download Page
    - Installation Directory: **C:\Program Files\MySQL\MySQL Workbench X.X**
4. **Flask**

- Install via pip: pip install Flask

5. **MySQL Connector for Python**
    - Install via pip: pip install mysql-connector-python

Setting Up the Project:

1. Navigate to the project directory.
2. Install required libraries: install -r requirements.txt
3. Set up the MySQL database:
    - Create a new database named **renthive8**.
    - Run the SQL scripts provided in the **sql** directory to set up the tables and stored procedures.

Running the Application

1. Ensure the MySQL server is running.
2. Make sure to include your name of the root, password and database name in app.py
3. Run the application: python app.py
4. Access the application in your web browser at: http://127.0.0.1:5000/ (Also refer to the terminal for the link)

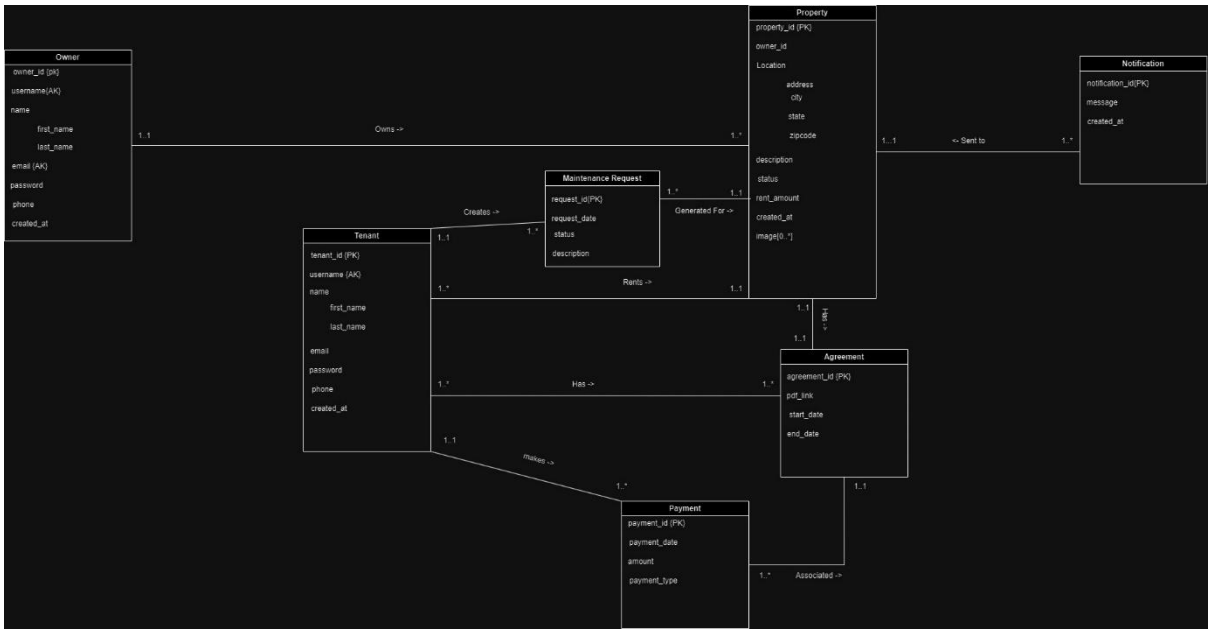
Technical Specifications

Software Used

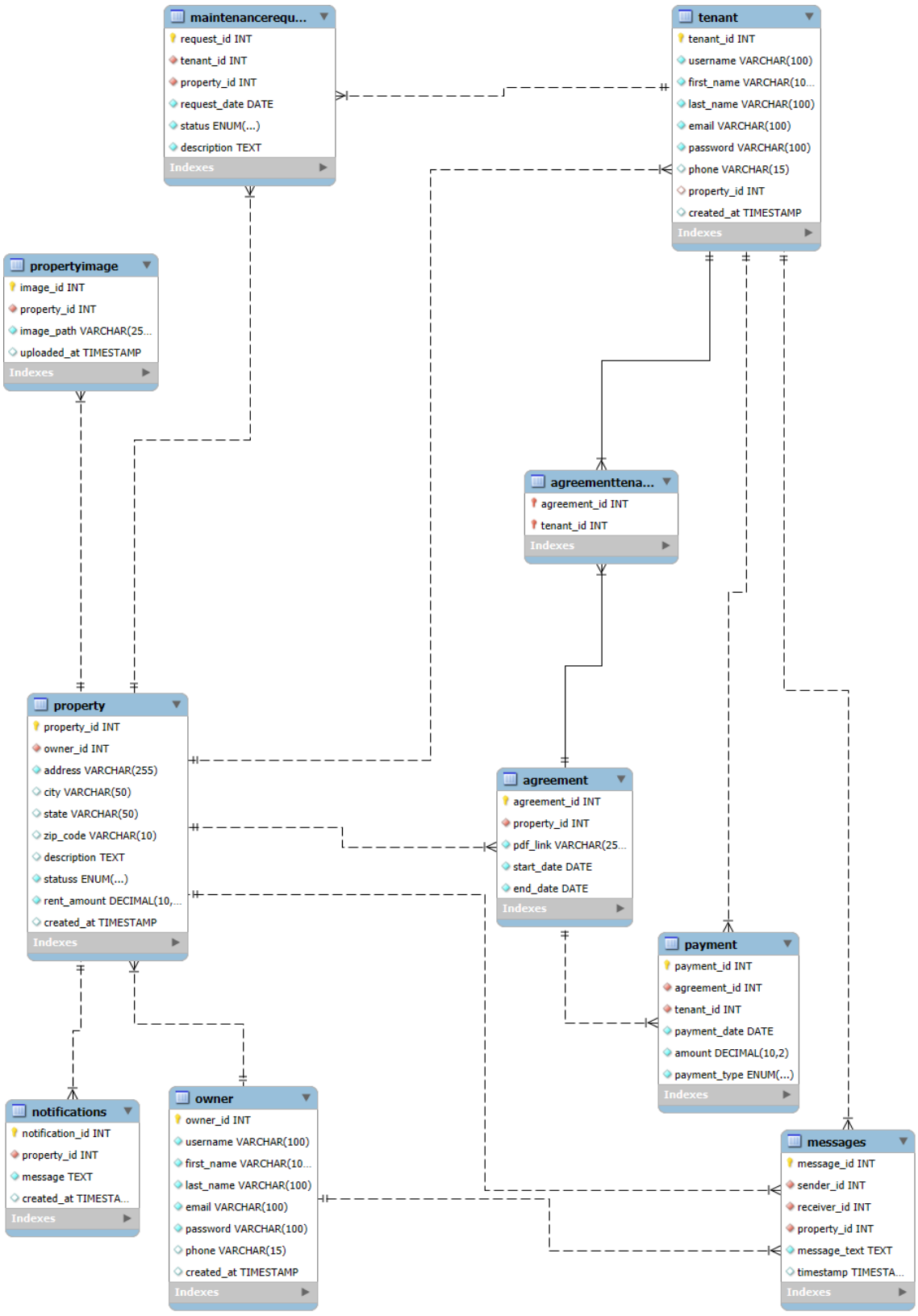
- **Host Language**: Python
- **Web Framework**: Flask
- **Database**: MySQL
- **Frontend Technologies**: HTML, CSS, JavaScript, Bootstrap
- **Database Management**: MySQL Workbench
- **Version Control**: Git

CONCEPTUAL DESIGN

UML Diagram



LOGICAL DESIGN



DATABASE SCHEMA

1. Owner Table

- Primary table for property owners
- Key fields: owner_id (PK), username, first_name, last_name, email, password, phone
- Tracks basic owner information and authentication details

2. Property Table

- Stores property listings
- Key fields: property_id (PK), owner_id (FK), address, city, state, zip_code, description, status (Available/Rented), rent_amount
- Links to Owner table through owner_id
- Tracks property availability and rental pricing

3. Tenant Table

- Stores tenant information
- Key fields: tenant_id (PK), username, first_name, last_name, email, password, phone, property_id (FK)
- Links to Property table through property_id
- Tracks which property a tenant is renting

4. PropertyImage Table

- Stores images for properties
- Key fields: image_id (PK), property_id (FK), image_path
- Links to Property table through property_id
- Allows multiple images per property

5. Agreement Table

- Stores lease agreements
- Key fields: agreement_id (PK), property_id (FK), pdf_link, start_date, end_date
- Links to Property table through property_id
- Tracks lease duration and documentation

6. AgreementTenant Table

- Junction table linking agreements to tenants
- Key fields: agreement_id (FK), tenant_id (FK)
- Enables many-to-many relationship between agreements and tenants. An owner can register many updated leases for the same tenant and for the same property. To remove relationship between that tenant and that property he needs to delete all the existing leases later.

7. Payment Table

- Tracks rent and deposit payments
- Key fields: payment_id (PK), agreement_id (FK), tenant_id (FK), payment_date, amount, payment_type (Deposit/Rent)
- Links to both Agreement and Tenant tables

8. MaintenanceRequest Table

- Manages property maintenance requests
- Key fields: request_id (PK), tenant_id (FK), property_id (FK), request_date, status (Open/In Progress/Completed), description
- Links to both Tenant and Property tables

9. Notifications Table

- Stores system notifications
- Key fields: notification_id (PK), property_id (FK), message, created_at
- Links to Property table

The schema includes numerous stored procedures for common operations like:

1. User registration and login
2. Property management (adding, updating, viewing)
3. Lease agreement management
4. Payment processing
5. Maintenance request handling
6. Notification management

Key Features:

1. Full user authentication for both owners and tenants
2. Property listing and management
3. Lease agreement tracking
4. Payment processing
5. Maintenance request system
6. Messaging and notification system
7. Image management for properties

The database uses proper foreign key constraints for referential integrity and includes timestamps for tracking creation dates of various entities. It's designed to handle the complete lifecycle of rental property management, from listing to tenant management and maintenance.

USER FLOW

1. User Registration

- **Objective**: Allow new users (owners or tenants) to create an account.
- **Flow**:
    - Users navigate to the "Sign Up" page on the website.
    - Users are required to provide their username, email address, password, and personal information (e.g., first name, last name, phone number).
    - There is an option to select their user type as either "Owner" or "Tenant."
    - After submitting the required details, the system verifies the data for validity and uniqueness (e.g., unique email and username), and their account is created.
- **Outcome**: Users are successfully registered and can now log in to access the application.

2. User Login

- **Objective**: Provide access to the user-specific dashboard.
- **Flow**:
    - Registered users can navigate to the "Login" page and enter their username and password.
    - The system checks the credentials:
        - If valid, users are redirected to their dashboard.
        - If invalid, an error message is displayed, and users are given the chance to retry.
- **Outcome**: Users gain access to their personalized dashboard based on their role (Owner or Tenant).

3. Property Management

- **Objective**: Manage property listings (add, edit, delete) and allow tenants to view available properties.
- **Owner Flow**:
    - **Add Property**: Owners can navigate to their dashboard and select the "Add Property" option.
        - Owners provide details about the property, including address, rent amount, description, and upload images of the property.
        - Upon submission, the property is added to the list of available properties.
    - **Edit Property**: Owners can edit details of properties they previously listed.
    - **Delete Property**: Owners can also remove properties they no longer want to rent out.
- **Tenant Flow**:
    - **View Available Properties**: Tenants can see all available properties, along with detailed information about each property, such as images, location, rent, and availability status.
- **Outcome**: Owners have full control over property listings, while tenants can browse and inquire about available properties.

4. Lease Agreements

- **Objective**: Facilitate the creation and management of lease agreements between owners and tenants.
- **Owner Flow**:
    - **Create Lease Agreement**: Owners can create a lease agreement once a tenant is ready to move forward.

        - Owners specify details like tenant information, property details, lease start and end dates, and agreement documents (e.g., PDF).
    - **Edit/Delete Lease Agreement**: Owners have the ability to update the lease agreement details or delete the agreement if necessary.
- **Tenant Flow**:
    - **View Lease Agreement**: Tenants can view their lease agreements, including all details and uploaded documents.
- **Outcome**: Lease agreements are created, stored, and accessible to tenants and owners for easy reference.

5. Maintenance Requests

- **Objective**: Facilitate maintenance requests from tenants and provide owners with a mechanism to track and manage them.
- **Tenant Flow**:
    - **Create Maintenance Request**: Tenants can create a maintenance request if any issues arise in the rented property.
        - Tenants specify the issue, location, and description, and the request is logged in the system.
    - **View Request Status**: Tenants can track the status of their maintenance requests (e.g., Open, In Progress, Completed).
- **Owner Flow**:
    - **View Maintenance Requests**: Owners can view all maintenance requests for their properties.
    - **Modify Request Status**: Owners can update the status of requests (e.g., mark it as In Progress or Completed).
- **Outcome**: Maintenance issues are effectively logged, managed, and tracked, ensuring tenant satisfaction and efficient handling by owners.

6. Payment Processing

- **Objective**: Provide a system for tenants to make payments and allow owners to track payment history.
- **Tenant Flow**:
    - **Make Payment**: Tenants can make rent payments through the application.
        - Tenants can enter the amount and payment type and confirm the transaction.
    - Payments are logged in the Payment table for tracking purposes.
- **Owner Flow**:
    - **View Payment History**: Owners can access a history of all payments made for their properties, including tenant details, payment date, and payment type (e.g., rent, deposit).
- **Outcome**: Payments are efficiently processed through the application

7. Notifications

- **Objective**: Allow owners to send important notifications to tenants.
- **Owner Flow**:
    - **Send Notifications**: Owners can send notifications to tenants regarding lease details, maintenance updates, payment reminders, or other important matters.
        - Notifications are logged and sent directly through the portal.
- **Tenant Flow**:
    - **View Notifications**: Tenants can view the notifications sent by the owners within their dashboard.
- **Outcome**: Notifications keep tenants informed about any updates, ensuring smooth communication between tenants and owners.

8. Homepage

- **Objective**: Provide an overview of all available properties to all visitors.
- **Flow**:
    - The homepage of the website displays a list of available properties, allowing visitors to view basic information about properties without logging in.
    - Visitors can click on a property to see more details, such as images, description, rent amount, and owner contact information.
    - If interested, users can choose contact the owner for more information.
- **Outcome**: The homepage serves as the entry point for new visitors and provides an overview of properties to attract potential tenants.

LESSONS LEARNED

Technical Expertise Gained

- Gained experience in database design and management using MySQL.
- Developed and implemented normalized database schemas to ensure data integrity and reduce redundancy including the creation of tables, relationships, and constraints.
- Designed and optimized stored procedures and functions to encapsulate business logic and facilitate complex queries, enhancing the efficiency of data retrieval and manipulation.
- Utilized Entity-Relationship (ER) diagrams well to visualize and plan database structures, ensuring a clear understanding of data relationships and dependencies.

- Improved proficiency in Python and Flask for web development.
- Enhanced skills in front-end technologies like HTML, CSS, and JavaScript.

INSIGHTS

1. Time Management Insights

- **Prioritization of Tasks:** Learned to prioritize tasks based on their impact on project milestones, allowing for more efficient use of time and resources. This included identifying critical path tasks that directly influenced project deadlines.
- **Setting Realistic Deadlines:** Developed the ability to set realistic deadlines by assessing the complexity of tasks and potential roadblocks, leading to more accurate project timelines and reduced stress.

2. Data Domain Insights

- **Understanding Data Relationships:** Gained a deeper understanding of data relationships and dependencies through the design of relational databases, which improved the ability to model complex data structures effectively.
- **Data Integrity and Quality:** Recognized the importance of data integrity and quality in database management, leading to the implementation of validation rules and constraints to ensure accurate data entry and retrieval.
- **Domain-Specific Knowledge:** Acquired domain-specific knowledge related to real estate and property management, enhancing the ability to design features and functionalities that meet user needs effectively.

3. Overall Project Insights

- **Collaboration and Communication**: Improved collaboration and communication skills by working in a team environment, learning to articulate ideas clearly and listen to feedback constructively.
- **Problem-Solving Skills**: Enhanced problem-solving skills through troubleshooting technical issues and debugging code, fostering a proactive approach to identifying and resolving challenges.
- **User -Centric Design**: Gained insights into user-centric design principles emphasizing the importance of user experience (UX) in web development which led to the creation of more intuitive and accessible interfaces.

Alternative Designs

- Considered using a NoSQL database for flexibility but opted for MySQL for structured data management.
- Explored different front-end frameworks (e.g., React) but decided to stick with Flask, Html, CSS and JavaScript for simplicity

FUTURE WORK

Planned Uses of the Database

- Expand the database to include user reviews and ratings for properties.

Potential Areas for Added Functionality

- Currently, visitors contact property owners via email regarding properties. If both parties agree to proceed, the visitor creates an account, and the owner creates the lease agreement using the tenant's email address. Going forward, we plan to integrate an in-platform messaging feature. This will ensure all communication between tenants and owners occurs directly through the portal, enhancing convenience and maintaining all records in one place.
- Develop a mobile application version of RentHive for better accessibility.