Senior Staff Software Engineer (Credit Reporting) - Domain Challenge

You are working at a company that creates software for rental property management. The system has to keep track of apartments within individual properties and tenants within those apartments. The system has to track rent payments and handle communications with tenants around rent.

Property managers own one or more properties with an address and a name. Apartments exist within a property and have unit numbers and one or more tenants. One or more of those tenants can be listed as primary lease-holders, while the rest are listed as secondary tenants. Tenants have the following information: first name, last name, date of birth, social security number.

Tenants may create payments to fulfill their rent obligations, which include a date and amount in USD dollars.

1. Draw an entity-relation diagram illustrating a database schema that could support this system, explaining the relationship between property managers, properties, tenants, payments, and apartments.
2. Implement a RESTful CRU API microservice that allows one to create, read, update properties, apartments, and tenants. You can use any language or framework that you prefer. Pay particular attention to code quality, including readability, and ease of development.
    a. The system should run locally and use a local SQLite database for testing.
    b. Each endpoint should be covered by unit tests. You may choose any testing framework you wish, but the tests must be isolated and not require any network connection (for example, may run on the Cloud without requiring database setup).
    c. Document your code and explain how to run the service alongside the tests.
    d. It is preferable to drive your development with Test-Driven-Development methodologies.
3. Create a web API that allows the creation of payments for a given tenant. Validate the payment to make sure that the amount is correct, and that the date of payment is a valid date in the past. The API should handle potential errors, and return the correct response in case of an error or a successful operation.
4. Create a web API that will return a historical list of payments per month for a specific tenant. This API should return a time series array, consisting of a payment (amount and timestamp) per month. The API should aggregate partial payments made in a single month: so if the tenant has two or more payments in a single month, these payments should be summed and the date of the payment will be the date of the oldest payment in the month.

5. In practice, there is another component of processing rent payments and managing accounts and leases. Let's say that we want our system to be able to create a report showing how much revenue a given property generated within a historical period, considering payments made by all tenants. Describe how you would extend the service to store rent payments history and make queries that allow you to pull revenue history from a given property or apartment over a given time window.

Requirements:

1. All the APIs must follow RESTful practices.
2. All the APIs must be covered by at least one unit test, aiming for full test coverage. The test environment should be isolated and self-contained.
3. Include a README file where anyone can view how to run the app locally, how to run the tests, and the endpoints available for use.

****Please submit your complete domain challenge to the following recipients:*

pradeep@esusu.org
ismail@esusu.org
abbyc@esusu.org