4.1  
Для реалізації календаря з логуванням користувача та можливістю додавання подій можна використовувати підхід **Layered architecture**. Цей підхід полягає в тому, щоб розділити проект на окремі рівні, де кожен рівень відповідає за свої функції.

На першому рівні буде інтерфейс користувача, який взаємодіє з додатком. На другому рівні буде рівень логіки додатку, де будуть перевірятися дані користувача та виконуватися необхідні операції з базою даних. На третьому рівні буде база даних, де будуть зберігатися дані про користувачів та їх події.  
Цей підхід є достатньо простим та зрозумілим, тому його можна легко пояснити. Крім того, він дозволяє забезпечити досить високий рівень безпеки, оскільки дані користувачів будуть зберігатися у базі даних на третьому рівні, що зменшує ризик їх втрати або підтримки відповідної конфіденційності.

Загальне поняття про різні підходи:

1. Layered architecture: архітектура проекту, яка розділяє додаток на логічні рівні (presentation, business, data) і забезпечує взаємодію між ними через інтерфейси.
2. Hexagonal architecture: архітектура, яка розділяє додаток на внутрішні та зовнішні шари та забезпечує ізольованість бізнес-логіки від залежностей зовнішніх факторів.
3. Onion architecture: архітектура, яка розділяє додаток на кілька логічних рівнів, що знаходяться один в середині іншого, забезпечуючи легкість розширення та тестування.
4. Clean architecture: архітектура, яка розділяє додаток на внутрішні та зовнішні шари і забезпечує чистоту коду та його тестовість шляхом використання принципів SOLID та інших підходів.

4.2  
Layered architecture - це архітектурний підхід, при якому програмне забезпечення розбивається на окремі шари (логічні групи) з певними відповідностями інтерфейсів та функцій.

ASP .NET Core MVC - це платформа розробки веб-додатків, яка підтримує багато різних архітектурних підходів, включаючи Layered architecture.

Отже, побудова базових архітектурних структур в рамках вибраного архітектурного підходу (Layered architecture) та платформи ASP .NET Core MVC означає, що розробник буде створювати шарову архітектуру для свого веб-додатку з використанням ASP .NET Core MVC. Це дозволить додатку бути більш організованим, розширюваним та обслуговуваним у майбутньому.  
Побудова базової архітектурної структури для вибраного архітектурного підходу (Layered architecture) та платформи ASP .NET Core MVC. Основними компонентами такої архітектури є:

Presentation Layer: цей рівень відповідає за відображення інформації користувачеві та обробку його запитів. Для платформи ASP .NET Core MVC це можуть бути контролери та розташований в них код, який відповідає за обробку запитів користувача.

Application Layer: цей рівень містить бізнес-логіку та сервіси, які реалізовують операції з даними та інші бізнес-процеси. Для платформи ASP .NET Core MVC це можуть бути сервіси, які виконують різні бізнес-операції, наприклад, додавання нової події до календаря або видалення події з календаря.

Domain Layer: цей рівень відповідає за доменну модель, яка описує об'єкти та взаємодії між ними в додатку. Для платформи ASP .NET Core MVC це може бути модель даних, яка описує об'єкти, що використовуються в додатку (події, користувачі, тощо).

Infrastructure Layer: цей рівень відповідає за реалізацію засобів доступу до даних, зовнішніх сервісів та інших залежностей. Для платформи ASP .NET Core MVC це можуть бути різні репозиторії, сервіси доступу до даних, які виконують операції з базою даних або іншими зовнішніми сервісами.

Окрім того, важливо визначити, які сервіси та функціональності будуть включені в кожен з рівнів архітектури та як вони будуть взаємодіяти між собою. Наприклад, в рівні Presentation можуть бути сервіси для обробки запитів користувача та відображення даних на екрані, в рівні Business Logic - сервіси для обробки бізнес-логіки, а в рівні Data Access - сервіси для збереження та отримання даних з бази даних.

Крім того, важливо забезпечити розширюваність та гнучкість архітектури, щоб забезпечити можливість змінювати функціональність проекту в майбутньому без необхідності повного переписування коду.

У платформі ASP .NET Core MVC для реалізації Layered architecture можна використовувати такі елементи, як контролери (Controllers), сервіси (Services), репозиторії (Repositories), моделі (Models) та інші. Контролери відповідають за приймання та обробку запитів від користувача, сервіси - за виконання бізнес-логіки, репозиторії - за збереження та отримання даних з бази даних, а моделі - за представлення даних та їх передачу між рівнями архітектури.

4.3   
Ми напевно оберемо PostgreSQL для баз даних. Оскільки він має готові рішення для забезпечення безпеки, резервного копіювання, масштабованості та інших вимог, які можуть знадобитися для веб-додатків.

Код для бази даних для календаря з логуванням користувачів та можливістю додавати події для авторизованих користувачів. Ось схема бази даних на PostgreSQL:  
  
CREATE TABLE users (

id SERIAL PRIMARY KEY,

username VARCHAR(50) NOT NULL UNIQUE,

password VARCHAR(100) NOT NULL

);

CREATE TABLE events (

id SERIAL PRIMARY KEY,

title VARCHAR(255) NOT NULL,

date DATE NOT NULL,

user\_id INTEGER NOT NULL REFERENCES users(id) ON DELETE CASCADE

);

Тут створюються дві таблиці: "users" та "events". Таблиця "users" містить інформацію про користувачів, які мають доступ до системи. Таблиця "events" містить інформацію про події, які додані користувачами. Кожен запис у таблиці "events" посилається на відповідний запис у таблиці "users", щоб забезпечити відношення один до багатьох між користувачами та подіями.

В таблиці "users" містяться наступні поля:   
"id": унікальний ідентифікатор користувача, автоматично згенерований PostgreSQL.  
"username": логін користувача, обов'язкове поле, не повинно повторюватися із вже зареєстрованими користувачами.  
"password": пароль користувача, обов'язкове поле.

У таблиці "events" містяться наступні поля:  
"id": унікальний ідентифікатор події, автоматично згенерований PostgreSQL.  
"user\_id": ідентифікатор користувача, що додав подію. Забезпечує відношення один до багатьох між користувачами та подіями.  
"title": назва події, обов'язкове поле.  
"start\_date": дата та час початку події, обов'язкове поле.  
"end\_date": дата та час закінчення події, обов'язкове поле.  
"is\_public": ознака, що вказує, чи можуть інші користувачі бачити цю подію.

4.4  
Ми напевно оберемо StyleCop - це аналізатор, який перевіряє дотримання кодувальних стандартів в .NET-проектах. Він допомагає забезпечити стиль кодування, що допомагає підтримувати код в чистоті та зрозумілості.