Authors:

Ayodeji Olagoke

Precious Oziwo

Martin Radoychev

Threading in C#3
Start document

ROB LOVES

VERSION 1.1

Table of Contents

Reason

The document aims to help readers understand the key aspects of our "Threading in C#" submission.

Application Description

The Book Reading App is a platform for the publication and reading of books. Users of the app are categorized into three main roles: Writer, Admin/Editor, and Reader. Each role has specific privileges and interactions within the system for example a Writer can publish books, with permission of admin while Reader can read only. However, Writer and Editor are extensions of a Reader. An API is also provided along with the code for the login system.

Features

- 3 user privileges
- All can read and leave comment on page.
- Reader can request permission to become writer but they need a pitch
- Writer makes request to publish book with pitch
- Editor can drop by and leave comments on writer's work.
- We would like to implement thread pool, locking, semaphore, mutex, TPL and an Asynchronous IO
- Seamless UI interaction
- Version control on GitHub
- Code Conventions according to c# .NET standard. (See references)
- Installation manual for debugging
- MVVM architecture
- C# .NET Maui

MOSCOW ANALYSIS

Functionality	Must have	Should have	Could have	Won't have
Uploading and retrieving books	✓			
Request permission to upgrade role via pitch	✓			
Implement at least 4 instances of threading	✓			
Responsive UI controls	✓			
Users can leave comments on individual books				✓
Editor can accept request to publish	✓			
Version control on GitHub	✓			

Code conventions	✓			
Installation manual for debugging	✓			
MVVM architecture	✓			
C# .NET Maui	✓			

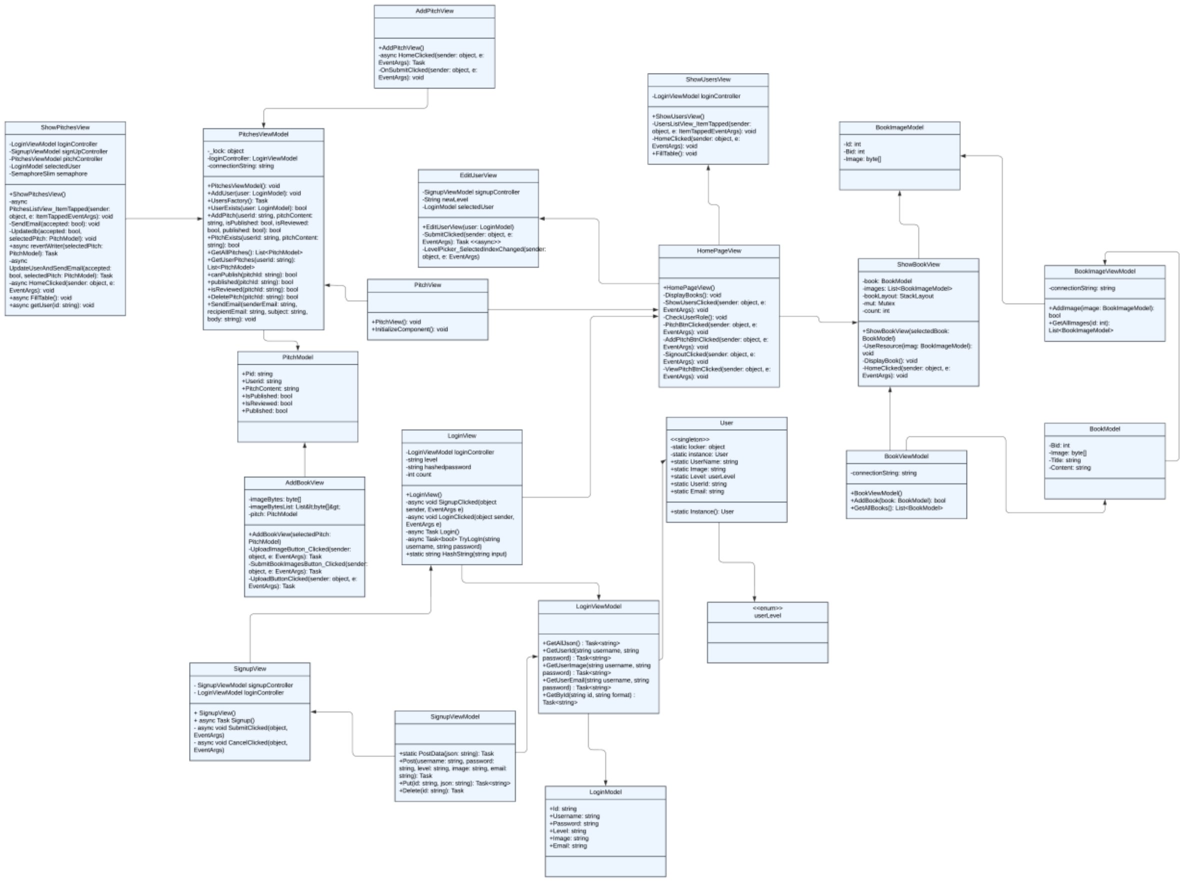
Threading techniques

- TPL (Task Parallel Library/Async & Await)
- Semaphore (SemaphoreSlim)
- Asynchronous I/O
- Locking
- Mutex

Definition of done

The project is done when the conditions stated in the MOSCOW analysis are reached.

Class diagram


AddPitchView
+AddPitchView()
-async HomeClicked(sender: object, e: EventArgs): Task
-OnSubmitClicked(sender: object, e: EventArgs): void
ShowPitchesView
-LoginViewModel loginController
-SignupViewModel signUpController
-PitchesViewModel pitchController
-LoginModel selectedUser
-SemaphoreSlim semaphore
+ShowPitchesView()
-async PitchesListView_ItemTapped(sender: object, e: ItemTappedEventArgs): void
-SendEmail(accepted: bool): void
-Updatedb(accepted: bool, selectedPitch: PitchModel): void
+async revertWriter(selectedPitch: PitchModel): Task
-async UpdateUserAndSendEmail(accepted: bool, selectedPitch: PitchModel): Task
-async HomeClicked(sender: object, e: EventArgs): void
+async FillTable(): void
+async getUser(id: string): void
PitchesViewModel
-_lock: object
-loginController: LoginViewModel
-connectionString: string
+PitchesViewModel(): void
+AddUser(user: LoginModel): void
+UsersFactory(): Task
+UserExists(user: LoginModel): bool
+AddPitch(userId: string, pitchContent: string, isPublished: bool, isReviewed: bool, published: bool): bool
+PitchExists(userId: string, pitchContent: string): bool
+GetAllPitches(): List<PitchModel>
+GetUserPitches(userId: string): List<PitchModel>
+canPublish(pitchId: string): bool
+published(pitchId: string): bool
+isReviewed(pitchId: string): bool
+DeletePitch(pitchId: string): bool
+SendEmail(senderEmail: string, recipientEmail: string, subject: string, body: string): void
ShowUsersView
-LoginViewModel loginController
+ShowUsersView()
-UsersListView_ItemTapped(sender: object, e: ItemTappedEventArgs): void
-HomeClicked(sender: object, e: EventArgs): void
+FillTable(): void
BookImageModel
-Id: int
-Bid: int
-Image: byte[]
EditUserView
-SignupViewModel signupController
-String newLevel
-LoginModel selectedUser
+EditUserView(user: LoginModel)
-SubmitClicked(sender: object, e: EventArgs): Task <<async>>
-LevelPicker_SelectedIndexChanged(sender: object, e: EventArgs)
HomePageView
+HomePageView()
-DisplayBooks(): void
-ShowUsersClicked(sender: object, e: EventArgs): void
-CheckUserRole(): void
-PitchBtnClicked(sender: object, e: EventArgs): void
-AddPitchBtnClicked(sender: object, e: EventArgs): void
-SignoutClicked(sender: object, e: EventArgs): void
-ViewPitchBtnClicked(sender: object, e: EventArgs): void
ShowBookView
-book: BookModel
-images: List<BookImageModel>
-bookLayout: StackLayout
-mut: Mutex
-count: int
+ShowBookView(selectedBook: BookModel)
-UseResource(imag: BookImageModel): void
-DisplayBook(): void
-HomeClicked(sender: object, e: EventArgs): void
BookImageViewModel
-connectionString: string
+AddImage(image: BookImageModel): bool
+GetAllImages(id: int): List<BookImageModel>
PitchView
+PitchView(): void
+InitializeComponent(): void
PitchModel
+Pid: string
+UserId: string
+PitchContent: string
+IsPublished: bool
+IsReviewed: bool
+Published: bool
LoginView
-LoginViewModel loginController
-string level
-string hashedpassword
-int count
+LoginView()
-async void SignupClicked(object sender, EventArgs e)
-async void LoginClicked(object sender, EventArgs e)
-async Task Login()
-async Task<bool> TryLogin(string username, string password)
+static string HashString(string input)
User
<<singleton>>
-static locker: object
-static instance: User
+static UserName: string
+static Image: string
+static Level: userLevel
+static UserId: string
+static Email: string
+static Instance(): User
BookViewModel
-connectionString: string
+BookViewModel()
+AddBook(book: BookModel): bool
+GetAllBooks(): List<BookModel>
BookModel
-Bid: int
-Image: byte[]
-Title: string
-Content: string
AddBookView
-imageBytes: byte[]
-imageBytesList: List<byte[]>
-pitch: PitchModel
+AddBookView(selectedPitch: PitchModel)
-UploadImageButton_Clicked(sender: object, e: EventArgs): Task
-SubmitBookImagesButton_Clicked(sender: object, e: EventArgs): Task
-UploadButtonClicked(sender: object, e: EventArgs): Task
LoginViewModel
+GetAllJson() : Task<string>
+GetUserId(string username, string password) : Task<string>
+GetUserImage(string username, string password) : Task<string>
+GetUserEmail(string username, string password) : Task<string>
+GetById(string id, string format) : Task<string>
<<enum>>
userLevel
SignupView
- SignupViewModel signupController
- LoginViewModel loginController
+ SignupView()
+ async Task Signup()
- async void SubmitClicked(object, EventArgs)
- async void CancelClicked(object, EventArgs)
SignupViewModel
+static PostData(json: string): Task
+Post(username: string, password: string, level: string, image: string, email: string): Task
+Put(id: string, json: string): Task<string>
+Delete(id: string): Task
LoginModel
+Id: string
+Username: string
+Password: string
+Level: string
+Image: string
+Email: string

References

BillWagner. (2023, December 15). *Identifier names - rules and conventions - C#*. Microsoft Learn. https://learn.microsoft.com/en-us/dotnet/csharp/fundamentals/coding-style/identifier-names