Click to verify

Skip to content Sign up now and upgrade anytime. All of our plans include unlimited app installs, unlimited app users and sessions and enterprise-grade encryption. Save 30% with Yearly plans! Prices shown in: EUR () USD ($) Get, reproduce, and fix bugs faster than ever at any stage of development. FREE 100K Daily Log Lines 24 hour Log Retention 2,000 Devices Unlimited Team Members Unlimited Apps Summary Email Daily For startups caring about the quality of their apps. From 49 /month 1M Daily Log Lines 7-day Log Retention Unlimited Devices All Free Plan Features Crash Reporting User Feedback Collection GDPR DPA Basic Permissions For agencies and product companies. From 89 /month 1M Daily Log Lines 30-day Log Retention All Basic Plan Features Fine-Grained Permissions Log Achiving to S3 Bucket API Access Webhooks Elastic Search Integration For companies with enterprise-grade requirements. From 399 /month Custom Volume 30-Day Log Retention All Pro Plan features Priority Support, including legal and compliance 99.95% uptime SLA on dashboard SSO with SAML Purchase Order Billing Payment via bank transfer And more... It is possible to purchase Bugfender to use on your own servers, see the On-Premises pricing page. If you have any special requirements, we'll be happy to help create the perfect package for you. This could include: Unlimited storage, apps, and log lines Corporate private cloud, or installation on a dedicated server On-premises license and support Geographic location of your servers Custom retention or backup policies Custom encryption needs Custom API and webhooks Custom terms and conditions, premium support, and Service Level Agreements Integration with your (non-SAML) Single Sign-On system: OIDC, LDAP, CAS or similar If you're interested in a custom plan or have any questions, get in touch. You can record up to 100,000 log lines every day which are retained for 24 hours. We accept all major credit cards including Visa, Mastercard, Discover, and American Express. For Enterprise plans, we can also accept payments via PayPal or wire transfer - just get in touch. Contact Us It is possible to purchase Bugfender to use on your own servers, see the On-Premises pricing page. Learn More Looking for a pdf with Bugfender's info and pricing? Download one here. Download Premium plans have an SLA of 99.95% monthly uptime on the provided service and 8 business hours initial response time on support (Monday to Friday, 9 to 5 UTC). In the event of violation, compensation is provided in form of a discount applicable on the next purchase for 24 times the time in violation. Other plans do not have an SLA, but we make an effort to respond to all support tickets within 24 hours. Feel free to reach out for custom support plans. Contact Us Don't wait for crashes to happen. Use Bugfender to get constant insights from every single user device anywhere in the world, so you can see the bug coming and fix it first. No credit card required Free forever 5 minutes setup Core Data is an essential framework in iOS development that allows developers to manage and persist data efficiently in their apps. Whether you're creating a simple note-taking app or a more complex social media platform, understanding Core Data can help you manage, save, and retrieve your apps data seamlessly. In this article, well explore what Core Data is, why its valuable, and when to use it in your iOS projects. Core Data is an object graph and persistence framework developed by Apple. It helps developers organize and store data locally on an iPhone or iPad. At its core, Core Data allows you to define a data model, create entities (similar to tables in a database), and manage their relationships. It can also manage the lifecycle of these entities and provide powerful tools for fetching, updating, and deleting data. There are several reasons why Core Data is a preferred choice for local data management: Data Persistence: Core Data allows you to store data on the device, so it remains available even after the app is closed. Data Relationships: You can model relationships between objects, like users and posts in a social app. Efficient Data Access: Core Data is optimized for performance and can efficiently fetch and manage large datasets. Change Tracking: Core Data tracks changes in your data, making it easier to manage updates. Data Versioning: Core Data supports data migrations, making it possible to evolve your data model as your app grows. Core Data is not always the best solution for every app. It's most useful in situations where you need to manage complex data relationships, persist large amounts of data, or use advanced querying capabilities. However, if your app just needs to store small pieces of data, like user preferences, UserDefaults or even saving to files might be simpler alternatives. Some situations where Core Data is a good fit: Managing large datasets (e.g., a to-do list app, social media app). Handling complex relationships between data (e.g., a user has multiple posts, posts have multiple comments). Needing to query data efficiently with filtering and sorting. Persisting data over multiple app launches or across multiple users. At a high level, Core Data works by creating a data model (which is a blueprint of the data you want to store) and managing instances of these models (referred to as managed objects). These managed objects are instances of your data entities, and Core Data automatically takes care of storing, fetching, and updating them. Lets break down the basic components of Core Data: Data Model: This is a schema that defines your entities (similar to tables in a database), their attributes, and relationships. NSManagedObject: This is a class that represents a single instance of an entity. For example, if you have a "Task" entity, each task you create in your app will be an instance of NSManagedObject. NSManagedObjectContext: Think of this as a workspace where you perform operations like saving, deleting, or fetching data. Persistent Store: This is where your data is physically saved on the device. It can be a SQLite database, XML file, or binary file. NSPersistentContainer: This is a helper class that sets up the Core Data stack and provides you with a ready-to-use managed object context. Lets go through a basic example where we create a "Task" entity and store it in Core Data. Well then fetch and display the tasks saved in the app. To enable Core Data in your project: When creating a new project in Xcode, select the "Use Core Data" checkbox. Xcode automatically generates a data model file (with a .xcdatamodeld extension) and sets up the Core Data stack for you. Next, we define a simple "Task" entity with a single attribute called "name" in the Data Model Editor. You can open the .xcdatamodeld file, add a new entity called "Task," and give it an attribute called "name" of type String. Heres how you can add a task to Core Data: Now lets retrieve all the tasks weve saved: Core Data is a powerful tool for persisting and managing data in iOS apps. By understanding the basics of how it works, you can start using it in your projects to store and retrieve data efficiently. In this series, we will delve deeper into the Core Data stack, relationships, performance optimization, and more advanced concepts. In this article, we covered the basics of what Core Data is and why its useful for managing and persisting data in iOS applications. We also walked through a simple example of setting up Core Data and performing basic CRUD operations. In the next article, we'll dive deeper into setting up Core Data in your projects and the structure of the Core Data stack. In the previous article, Mastering Relationships in Core Data: Fundamentals we explored the basic concepts and principles of relationships in Core Data. Building on that foundation, this article aims to share practical experience and techniques for handling relationships in Core Data. The goal is to assist developers in more effectively utilizing the relational features of the Core Data framework, thereby enhancing development flexibility and efficiency.This article is intended for readers who already have some knowledge and practical experience with Core Data relationships, providing an advanced understanding and application perspective, rather than offering a comprehensive tutorial.OptionalWhen defining entity attributes in the Xcode model editor, developers should differentiate between the Optional option in the editor and the Optional type in Swift, as they are not the same. In Core Data, the Optional option means that the corresponding SQLite field can accept NULL values. In contrast, the Optional type in Swift is a language-level feature that indicates a variable can be nil. In Core Data models, the use of these two types of Optional depends on the specific scenario and the developers needs, and they do not necessarily correspond directly.In Core Data, if an attribute of a model is marked as Optional, it can be defined as Non-Optional in the corresponding Swift code. This approach offers more flexibility, allowing developers to decide whether to use Swifts Optional type in the code based on the actual application context.For more detailed information on the Optional values in Core Data, please read Ask Apple 2022 Q&A Related on Core Data (Part 2).For instance, consider Item and Tag, two entities with a One-to-One relationship. When using Core Data with CloudKit, these relationships must be marked as Optional in the model editor. However, in practical application, if these two entity instances are always related to each other, meaning their relationship always has a value, they can be adjusted to be non-optional in the Swift code. The benefit of this adjustment is more convenient access to these properties in the code, eliminating the need for frequent unwrapping. The default code generated by Core Data is as follows: extension Item { @NSManaged public var timestamp: Date? @NSManaged public var tag: Tag? // Optional}extension Tag { @NSManaged public var name: String? @NSManaged public var item: Item? // Optional}However, you can adjust them to be non-optional based on the actual situation: extension Item { @NSManaged public var timestamp: Date @NSManaged public var tag: Tag // Non-Optional}extension Tag { @NSManaged public var name: String // None-Optional @NSManaged public var item: Item // Non-Optional}This allows for more convenient data retrieval in the code, provided that developers ensure that the properties have been assigned values before they are accessed: Text(item.tag.name)Swiftifying Core Data Collection TypesWhen dealing with to-Many relationships in Core Data, especially those involving ordered relationships, adjusting their representation in Swift code can offer significant benefits.For instance, consider changing tag to an ordered to-Many relationship tags: The default code generated by Core Data is as follows: extension Item { @NSManaged public var timestamp: Date? @NSManaged public var tags: NSOrderedSet?}To enhance readability and usability of the code, we can consider converting the NSOrderedSet? type to Array. This adjustment not only reduces the need for unwrapping but also aligns the tags property more closely with Swift language conventions, such as using subscripting and iterators. extension Item { @NSManaged public var timestamp: Date? @NSManaged public var tags: Array}After this adjustment, we can more conveniently manipulate these data in Swift, for example (as Array conforms to the RandomAccessCollection protocol): ForEach(item.tags){ tag in Text(tag.name ?? "")}However, its worth noting that converting a non-ordered to-Many relationship to an Array type may not always be the best choice. This is mainly due to the intrinsic characteristics of non-ordered collections and their management in Core Data. In Core Data, non-ordered relationships are typically represented as NSSet, intuitively reflecting the unordered nature and uniqueness of the elements in the collection. Converting this to an Array type might cause a loss of these key characteristics at face value. Therefore, for non-ordered relationships, using Swifts Set type is often a more appropriate choice.For example, for the tags attribute of the Item entity, if it is a non-ordered, optional to-Many relationship, it can be represented in Swift as follows: extension Item { @NSManaged public var timestamp: Date? @NSManaged public var tags: Set}This approach maintains the unordered nature and uniqueness of the collection while aligning the code more closely with Swift usage habits, enhancing its readability.CountWhen dealing with to-Many relationships in Core Data, its often necessary to obtain the count of associated objects. While directly using the .count property is a common method, developers can also consider using a derived attribute (Derived Attribute) for a more efficient way to obtain this count.For example, in the situation shown below, we have created a derived attribute named count for the TodoList entity. This allows developers to simply access todolist.count to directly obtain the number of items objects associated with the TodoList. This method makes retrieving the count of associated objects both intuitive and efficient. Compared to directly calling the .count property of a relationship, using a derived attribute for counting is generally more efficient. This is because derived attributes employ a different counting mechanismthey calculate and save the count value when data is written, and use this pre-calculated value when data is read. This mechanism is particularly suited to scenarios where read operations significantly outnumber write operations.However, an important limitation of derived attributes is that they can only count data that has been persisted. This means if there are data that have not yet been saved to persistent storage, i.e., in a transient state, these data will not be included in the count by the derived attribute. Therefore, when using derived attributes, developers need to be mindful of this limitation and ensure that their data handling logic takes this counting method into consideration.For a deeper understanding of how to use derived attributes, its recommended to read How to use Derived and Transient Properties in Core Data, which provides a detailed introduction to the application of derived attributes.Managing Non-Ordered to-Many RelationshipsIn many practical application scenarios, to-Many relationships are often non-ordered. This is especially evident when using Core Data with CloudKit, as it does not support ordered relationships.When data is directly retrieved through relationship properties, as shown in the example code below, Core Data cannot guarantee the order of the returned data: let tags = Array(items.tags)In most cases, Core Data uses a SQLite database for data storage at the backend. In the database, unless a specific sort order is explicitly defined, the retrieval order of records is indeterminate.Therefore, to ensure consistency when fetching non-ordered to-Many data, it is advised not to rely solely on direct use of relationship properties. Instead, create an NSFetchRequest that includes predicates and sort criteria to perform the query, as shown below: func fetchTagsBy(item:Item) -> [Tag] { let request = NSFetchRequest(entityName: "Tag") request.predicate = NSPredicate(format: "item = %@", item) request.sortDescriptors = [NSSortDescriptor(keyPath: \Tag.name, ascending: true)] return (try? viewContext.fetch(request)) ?? []}In SwiftUI development, its recommended to encapsulate the interface displaying to-Many data into a separate view and fetch data using @FetchRequest. This approach not only ensures the stability of the data retrieval order but also promptly responds to data changes, making view updates more efficient: struct TagsList: View { @FetchRequest var tags: FetchedResults init(item: Item) { let request = NSFetchRequest(entityName: "Tag") request.predicate = NSPredicate(format: "item = %@", item.objectID) // Using NSManagedObject and NSManagedObjectID generates the same SQL commands request.sortDescriptors = [NSSortDescriptor(keyPath: \Tag.name, ascending: true)] _tags = FetchRequest(fetchRequest: request) } var body: some View { List(tags) { tag in TagDetail(tag: tag) } }}struct TagDetail: View { @ObservedObject var tag: Tag var body: some View { Text(tag.name) }} In our previous article, we discussed how relationships can enhance query efficiency and expand querying capabilities in certain scenarios. Subqueries in Core Data are a prime example of this in action.A subquery is an efficient querying technique within the Core Data framework, allowing developers to perform more complex queries on an existing set of results. This is particularly useful when dealing with complex data models, especially when filtering based on attributes of related objects.The basic format of a subquery is as follows: SUBQUERY(collection, $x, condition)collection refers to the set to be queried, typically a to-many relationship property.$x is a variable representing each element in the set (the name can be arbitrarily set).condition is the criterion applied to each element in the collection.For example, suppose we want to retrieve all Item instances that have at least one Tag with a name starting with A. The following NSPredicate expression can be used: NSPredicate(format: "SUBQUERY(tags, $tag, $tag.name BEGINSWITH 'A').@count > 0")The corresponding operation using Swifts higher-order functions in memory would be: let result = items.filter { item in item.tags.contains { tag in tag.name.hasPrefix("A") }}Subqueries are executed directly at the SQLite level, meaning they are more efficient both in terms of performance and memory usage compared to filtering in memory. Additionally, it is recommended to perform all filtering and sorting operations at the SQLite level, using well-designed predicates and sorting conditions. This approach not only improves data processing efficiency but also helps reduce the memory load on the application, especially when dealing with large data sets.Whats NextIn this article, we have explored a series of techniques for applying relationships in Core Data within real-world development scenarios. Indeed, once developers grasp the fundamental theories and internal mechanisms of relationships, they can continuously summarize and discover methods and experiences that are more suitable for their own projects.The upcoming article will concentrate on SwiftData, the successor framework to Core Data. We will explore the changes in how SwiftData manages data relationships and critically assess the applicability of these changes. Particular attention will be given to how potential performance issues in relational operations can be effectively avoided in its initial version. Share copy and redistribute the material in any medium or format for any purpose, even commercially. Adapt remix, transform, and build upon the material for any purpose, even commercially. The licensor cannot revoke these freedoms as long as you follow the license terms. Attribution You must give appropriate credit , provide a link to the license, and indicate if changes were made . You may do so in any reasonable manner, but not in any way that suggests the licensor endorses you or your use. ShareAlike If you remix, transform, or build upon the material, you must distribute your contributions under the same license as the original. No additional restrictions You may not apply legal terms or technological measures that legally restrict others from doing anything the license permits. You do not have to comply with the license for elements of the material in the public domain or where your use is permitted by an applicable exception or limitation . No warranties are given. The license may not give you all of the permissions necessary for your intended use. For example, other rights such as publicity, privacy, or moral rights may limit how you use the material. Was this page useful? Let us know! 1 2 3 4 5 Average rating: 4.0/5 Programs to support computer infrastructureUtility software is a program specifically designed to help manage and tune system or application software. It is used to support the computer infrastructure - in contrast to application software, which is aimed at directly performing tasks that benefit ordinary users. However, utilities often form part of the application systems. For example, a batch job may run user-written code to update a database and may then include a step that runs a utility to back up the database, or a job may run a utility to compress a disk before copying files.Although a basic set of utility programs is usually distributed with an operating system (OS), and this first party utility software is often considered part of the operating system, users often install replacements or additional utilities.[1][2] Those utilities may provide additional facilities to carry out tasks that are beyond the capabilities of the operating system.Many utilities that might affect the entire computer system require the user to have elevated privileges, while others that operate only on the user's data do not.[3]Anti-virus utilities scan for computer viruses and block or remove them.Clipboard managers expand the clipboard functionality of an operating system.Computer access control software grants or denies requests for access to system resources.Debuggers typically permit the examination and modification of data and program instructions in memory and on disk.Diagnostic programs determine and report the operational status of computer hardware and software. Memory testers are one example.Network utilities analyze the computer's network connectivity, configure network settings, check data transfer or log events.Package managers are used to configure, install or keep up to date other software on a computer.Registry cleaners clean and optimize the Windows Registry by removing old registry keys that are no longer in use.System monitors monitor resources and performance in a computer system.System profilers provide detailed information about installed software and hardware.Backup software makes copies of all information stored on a disk and restores either the entire disk (aka Disk cloning) in an event of disk failure or selected files that are accidentally deleted or corrupted. Undeletion utilities are sometimes more convenient.Disk checkers scan an operating hard drive and check for logical (filesystem) or physical errors.Disk compression utilities transparently compress/uncompress the contents of a disk, increasing the capacity of the disk.Disk defragmenters detect computer files whose contents are scattered across several locations on the hard disk and collect the fragments into one contiguous area.Disk formatters prepare a data storage device such as a hard disk, solid-state drive, floppy disk or USB flash drive for initial use. These are often used to permanently erase an entire device.Disk partition editors divide an individual drive into multiple logical drives, each with its own file system which can be mounted by the operating system and treated as an individual drive.Disk space analyzers provide a visualization of disk space usage by getting the size for each folder (including sub folders) and files in folder or drive. showing the distribution of the used space.Tape initializers write a label to a magnetic tape or other magnetic medium. Initializers for DECtape format the tape into blocks.Archivers output a stream or a single file when provided with a directory or a set of files. Archive suites may include compression and encryption capabilities. Some archive utilities have a separate un-archive utility for the reverse operation. One nearly universal type of archive file format is the zip file.Cryptographic utilities encrypt and decrypt streams and files.Data compression utilities output a shorter stream or a smaller file when provided with a stream or file.Data conversion utilities transform data from a source file to some other format, such as from a text file to a PDF document.Data recovery utilities are used to rescue good data from corrupted files.Data synchronization utilities establish consistency among data from a source to a target data storage and vice versa. There are several branches of this type of utility:File synchronization utilities maintain consistency between two sources. They may be used to create redundancy or backup copies but are also used to help users carry their digital music, photos and video in their mobile devices.Revision control utilities can recreate a coherent structure where multiple users simultaneously modify the same file.Disk cleaners find files that are unnecessary to computer operation, or take up considerable amounts of space.File comparison utilities provide a standalone capability to detect differences between files.File managers provide a convenient method of performing routine data management, email recovery and management tasks, such as deleting, renaming, cataloging, uncataloging, moving, copying, merging, setting write protection status, setting file access permissions, generating and modifying folders and data sets.Data generators (e.g. IEBDG) create a file of test data according to specified patterns.Hex (or octal) editors directly modify the text or data of a file without regard to file format. These files can be data or programs.HTML checkers validate HTML code and check links.Installation or setup utilities are used to initialize or configure programs, usually applications programs, for use in a specific computer environment. There are also Uninstallers.Patching utilities perform alterations of files, especially object programs when program source is unavailable.Screensavers prevent phosphor burn-in on CRT and plasma computer monitors.Sort/Merge programs arrange records (lines) of a file into a specified sequence.Standalone macro recorders permit use of keyboard macros in programs that do not natively support such a feature.List of DOS commandsList of macOS built-in appsSupport programs for OS/360 and successorsList of Unix commandsList of KDE utilitiesBatch fileShell scriptSystem software^ "Non-Opec (advertisement)". Computerworld. Sep 3, 1979. Retrieved May 20, 2019.^ Mendelson, Edward (June 8, 1999). "Fix What Ails Your PC". PC Magazine. Retrieved May 20, 2019.^ "Linux ifconfig command". Computer Hope. Retrieved May 20, 2019.Retrieved from " Were going to investigate the basic strategies to debug an application created with Unity, from logs to breakpoints, during the development stage. Once an app is in production we can switch to using Bugfender, and well explain this too. To illustrate this tutorial we have created a custom app called The Bugfender Game, a variation of the popular Flappy Bird. Unity is a cross-platform game engine that gives users the ability to create games and experiences in 2D, 3D, VR and AR. Due to its simplicity and its all-in-one development environment it has been the most popular game engine for several years. Almost every single indie game developer either started building games using Unity, or is still using it. On one side, Unity has a very intuitive design that makes it easy to use. On the other side, the choice of C# as the main language keeps things easy for programmers, especially when compared to other game engines like Unreal that require a high level of C++ knowledge. All of these features make Unity a compelling choice for small teams as well as for people starting to make games. Add to that a great community, a huge asset store, and tons of tutorials and online courses and its easy to see why Unity has become and remains the most widely used game engine on the market. Unity offers a Personal License free of charge for people with annual revenue of less than $100,000. We dont expect that you can earn that amount of money with this tutorial (although wed be delighted to be proven wrong!) so go ahead and download Unity from the official website if you havent done so yet. Download Unity from the official webpage and follow the steps to install Unity Hub so you can manage both your projects and Unity versions. Check out this repository using: $git clone Open Unity and use the option Open Project to access the previously downloaded repo. We have used Unity 2019.4.0, but you can use any version older than 2018. The project is what we have called The Bugfender Game. It is a clone of the popular Flappy Bird. You can press the play button and use the left click to move the main character. In the lower left corner in the previous screenshot, you can see the Project panel where you can find the game classes. Double click in the GameController class and Unity will open Microsoft Visual Studio, which is the IDE we will use to write code and is installed by default on your computer, along with Unity. In Unity, like almost every other environment, we can use logs and breakpoints to debug our code. To print a log in the console we use Debug.Log in any part of our code. Debug.Log("This is a log in Unity"); Actually, we can differentiate between five different types of logs: Debug.Log: the default log level. We can use this to register the execution of a concrete line of code or to log the value of a variable.Debug.LogFormat: similar to the previous log type, this one allows us to introduce formatted text.Debug.LogWarning: this is the warning log level. We can use it if for example our game reaches a certain point in the execution that was unexpected but not so worrying that we should call it an error. (Maybe some frames where dropped?)Debug.LogError and Debug.LogException: these are the error and exception log levels that we should use to register serious issues, like the main character of the game suddenly falling out of the scene. In general, it is good practice using as much logs as possible, so when something unexpected happens we can go to the console and quickly revisit what exactly happened in the last few minutes. If you use the different log levels then, when you have several different logs, the Console Panel will display the logs with a different icon, which will help you understand the code execution. Moreover, you can use the buttons in the top right corner to filter the log levels that are important for you. You can find more information about the Unity Logger via the official documents. As you might know, a breakpoint is a mark in your code that will pause the execution of the game at the exact moment the code reaches the relevant point. If you are not familiar with the use of breakpoints, we wrote two articles studying the concept in depth. Those articles were written for iOS and Android but the concepts are pretty much the same: In Visual Studio, we can place a breakpoint in the left-hand bar of the window, close to the line number in which we want to interrupt the execution. Once we set the breakpoint, we will need to click the button Attach to Unity on the toolbar, which enables the debugger. Now, if everything worked, Visual Studio will switch to an orange outline meaning that we are in debugging mode. Finally, we use our game until the execution of the program reaches the part of the code which contains the breakpoint. The execution will be immediately paused when this point is reached. Where the execution has been interrupted, we can check that the Attach to Unity button now says Continue. This is the button we must press to resume the game execution. But before resuming the execution lets take a look at the lower panels. In the left-hand panel we can now see the variables that are in the scope of the breakpoint and even check its value. In the right-hand panel we can see the stack trace of the functions that brought us to this breakpoint. Finally, instead of continuing, we can use the step by step and the step over buttons (the arrows to the right of the Continue button in the screen). This button allow us to move only one line forward or to the next function without resuming the execution. We cannot add more code in Visual Studio until we finish the debugging session. That can be done simply by pressing the Stop button. You can find more Information about the Unity debugger in the official docs. Logs and breakpoints are perfect to debug our game in our computer. However, when the game is sent to production and users start to complain, this might be not enough. Most of the users have no technical skills, so we will often receive bug reports like it doesnt work or it crashes. Our recommended strategy is to contact those users and ask them to fill out a bug report template, or just ask them a few questions that might look obvious to you as a game developer but that are not obvious to the users: What were you doing when the game stopped working?What do you mean by stopped working? Does it mean you cant control the game anymore? Does it mean the app was suddenly closed?Could you describe a list of the steps by which the app reaches this unstable state where you cant play anymore? In an ideal world, we would visit the home of our users to see the problem with our own eyes and if necessary, we would execute the Unity debugger on their computers to find the exact problem with the code. We cant do that but we can use Bugfender, which is going to behave in a very similar way. Bugfender is a remote logger that allows you to revisit the console logs for a specific device. The only two things we need to do is add Bugfender to the Unity project and replace the calls to Debug.Log() with Bugfender.Log(). Then, when we receive a bug report from a user we can just go to the Bugfender console and check exactly what happened. Installing the Bugfender SDK in your project will take you just two minutes. Download and import this assets package into your project.Drag the package to Unity to import it. 3. Drag the Bugfender package from the Assets\Bugfender\Prefabs to the Hierarchy panel If you dont have a Bugfender account you can get one for free here.Go back to Unity, select the Bugfender GameObject in the hierarchy panel and set the Bugfender app key in the Inspector panel. Even if your game has several scenes, you only need to add Bugfender once, as we did in the last step. 6. Now you can use Bugfender.Log() to send logs to Bugfender. Run the game and ensure the code reaches the spot where you placed the log line. Log into the Bugfender console and after a few seconds a new device should show up in the devices list. 8. If you click on the device then you will be able to see the console logs for the device that youve sent to Bugfender using Bugfender.log(). Congratulations! Now youve completed the circle and you can debug your games locally, both while youre developing them and later on in production, when your users are already playing them. Thats all for this tutorial, but remember that weve only scratched the surface of Bugfenders capabilities. Bugfender can also help you to detect crashes, get user feedback, provide live customer service and create automated actions really, the list is endless. We suggest you visit our blog from time to time, because we regularly publish engineering-related content. If you would like to read more Unity-related tutorials or you have any specific doubts, feel free to reach us at , through our customer service channels or via Github. We have also an extensive section of documents on our website that offer plenty more useful information about Bugfender. Happy debugging! Swift Core Data, a powerful framework, streamlines the process of managing persistent storage in iOS and macOS applications. In this article, we will explore the fundamentals of Swift Core Data and demonstrate how to effectively utilize it in your projects.What is Core Data ?Core Data is a framework that provides an object-oriented approach to managing and persisting data in your application. Integrated seamlessly with Xcode, Core Data supports various data models and uses the SQLite database as its default persistent store, with additional support for XML and binary files.How to use it ?By integrating Core Data into your app, you can efficiently manage and store data, improving the user experience and enabling offline functionality. Lets walk through the process of integrating Core Data into your Swift app, step by step.Step 1: Set Up Core Data in Xcode:To begin, open your Xcode project and follow these steps:1. Select your project in the Project Navigator.2. Navigate to the Signing & Capabilities tab.3. Click the + button and search for Core Data in the list.4. Add the Core Data capability to your project.Step 2: Create a Data Model:1. Right-click on your projects folder in the Project Navigator and choose New File.2. Select Data Model under the Core Data category.3. Name your data model (e.g., MyAppModel) and click Create.Step 3: Design Your Data Model:1. In the newly created data model file, youll see an empty canvas.2. Click the + button to add entities and define their attributes.3. Add attributes such as name, image, or any other relevant data for your app.4. Establish relationships between entities, if necessary.Step 4: Generate Managed Object Subclasses:1. Select your data model file.2. Go to Editor -> Create NSManagedObject Subclass3. Select the entities you want to generate subclasses for.4. Click Next and choose the destination folder for the generated files.5. Click Create to generate the managed object subclasses.Lets begin by examining the addCoreData function within the CoreDataService class:static func addCoreData(name: String, image: URL) { let context = (UIApplication.shared.delegate as! AppDelegate).persistentContainer.viewContext let newData = MyData(context: context) newData.name = name newData.image = image do { try context.save() } catch { print("error-Saving data") }}This function allows you to add new data objects to the persistent store. It creates a new instance of the MyData entity within the managed object context. You can customize the properties of the new object, such as name and image. Once you save the context, the changes are persisted, making the data available for retrieval later.To retrieve data from the persistent store, we can use the fetchCoreData function:static func fetchCoreData(onSuccess: @escaping ([MyData]?) -> Void) { let context = (UIApplication.shared.delegate as! AppDelegate).persistentContainer.viewContext do { let items = try context.fetch(MyData.fetchRequest()) as? [MyData] onSuccess(items) } catch { print("error-Fetching data") }}This function fetches data objects of the MyData entity from the persistent store. It retrieves the managed object context from the application delegate and performs a fetch request. The fetched items are then passed to the onSuccess closure, allowing you to handle and process the data accordingly.When removing specific data objects from the persistent store, you can use the deleteCoreData function:static func deleteCoreData(indexPath: Int, items: [MyData]) { let context = (UIApplication.shared.delegate as! AppDelegate).persistentContainer.viewContext let dataToRemove = items[indexPath] context.delete(dataToRemove) do { try context.save() } catch { print("error-Deleting data") }}This function deletes the selected data object, referenced by the indexPath parameter, from the persistent store. It fetches the managed object context and uses it to remove the specified object. By saving the context, the deletion is committed and the changes are reflected in the persistent store.Conclusion:Swift Core Data provides a powerful and flexible solution for managing persistent storage in your iOS applications. In this article, we explored the basics of Core Data and learned how to perform essential operations such as adding, fetching, and deleting data objects. By utilizing the provided code snippets and understanding their purpose, you can harness the capabilities of Core Data to build robust and data-driven applications. Core Data is a framework that you use to manage the model layer objects in your application. It provides generalized and automated solutions to common tasks associated with object life cycle and object graph management, including persistence. Core Data typically decreases by 50 to 70 percent the amount of code you write to support the model layer. This is primarily due to the following built-in features that you do not have to implement, test, or optimize: Change tracking and built-in management of undo and redo beyond basic text editing. Maintenance of change propagation, including maintaining the consistency of relationships among objects. Lazy loading of objects, partially materialized futures (faulting), and copy-on-write data sharing to reduce overhead. Automatic validation of property values. Managed objects extend the standard key-value coding validation methods to ensure that individual values lie within acceptable ranges, so that combinations of values make sense. Schema migration tools that simplify schema changes and allow you to perform efficient in-place schema migration. Optional integration with the applications controller layer to support user interface synchronization. Grouping, filtering, and organizing data in memory and in the user interface. Automatic support for storing objects in external data repositories. Sophisticated query compilation. Instead of writing SQL, you can create complex queries by associating an NSPredicate object with a fetch request. Version tracking and optimistic locking to support automatic multiwriter conflict resolution. Effective integration with the macOS and iOS tool chains. Note This document uses an employees database-style example for expediency and clarity. It represents a rich but easily understood problem domain. However, the Core Data framework is not restricted to database-style applications, nor is there an expectation of client-server behavior. The framework is equally as useful as the basis of a vector graphics application such as Sketch or a presentation application such as Keynote. Creating a Managed Object Model Copyright 2018 Apple Inc. All rights reserved. Terms of Use | Privacy Policy | Updated: 2017-03-27Advanced Data Management for iOS ApplicationsiOS applications often require users to store personal data, settings, and other information. Using the right data management system for such data directly impacts your apps success. Core Data, provided by Apple, is a framework for data modeling and management on iOS and macOS. Simply put, Core Data is a powerful tool that allows you to manage, store, query, and retrieve your data.In this article, we will cover the basics of Core Data and provide a detailed guide with code examples to show you how to use it effectively.Photo has taken from OceanoBe websiteWhat is Core Data?Core Data is a framework used to manage database operations. It provides an abstraction layer over relational databases like SQLite, handling database operations for your app. However, Core Data is not just a database tool; its a comprehensive model that also optimizes in-memory data management, handles data relationships, and processes your data efficiently.Advantages of Core DataDatabase abstraction: You dont have to deal with SQL queries. Core Data handles them for you.Powerful data relationship management: Core Data makes it easy to manage relationships between your data (such as one-to-many, many-to-many). Share copy and redistribute the material in any medium or format for any purpose, even commercially. Adapt remix, transform, and build upon the material for any purpose, even commercially. The licensor cannot revoke these freedoms as long as you follow the license terms. Attribution You must give appropriate credit , provide a link to the license, and indicate if changes were made . You may do so in any reasonable manner, but not in any way that suggests the licensor endorses you or your use. ShareAlike If you remix, transform, or build upon the material, you must distribute your contributions under the same license as the original. No additional restrictions You may not apply legal terms or technological measures that legally restrict others from doing anything the license permits. You do not have to comply with the license for elements of the material in the public domain or where your use is permitted by an applicable exception or limitation . No warranties are given. The license may not give you all of the permissions necessary for your intended use. For example, other rights such as publicity, privacy, or moral rights may limit how you use the material. ,the free encyclopedia that anyone can edit.117,937 active editors 7,000,660 articles in EnglishAndrea Navagero (14831529) was a Venetian diplomat and writer. He entered the Great Council of Venice at the age of twenty, five years younger than was normal at the time. He edited manuscripts at the Aldine Press, garnering a reputation as a scholar and a highly skilled writer. In 1515, he was appointed the official historian of the Republic of Venice as well as the caretaker of a library containing the collection of the scholar Bessarion. Navagero was named the Venetian ambassador to Spain in 1523 and navigated the volatile diplomatic climate caused by the conflict between CharlesV of Spain and FrancisI of France. By the time Navagero arrived back in Venice in 1528, he had grown disillusioned with politics and wished to return to editing manuscripts and cultivating his prized gardens. Much to his dismay, he was appointed ambassador to France in January 1529. After traveling to meet with FrancisI, he fell ill and died that May. (Fullarticle...)Recently featured: Nosy KombaMcDonnell Douglas Phantom in UK serviceTransportation during the 2024 Summer Olympics and ParalympicsArchiveBy emailMore featured articlesAboutEngraving of the Great Pyramid of Giza... that the 1572 Eight Wonders of the World (engraving pictured) is the source of the modern list of classical Seven Wonders of the World?... that Hedwig Tam gained 20 pounds to play a postpartum mother in Montages of a Modern Motherhood?... that the Alfonsine Ordinances punished Jews and Muslims with enslavement if they disguised their identity with the intention of "sinning with Christian women"?... that even though he had never seen a field hockey game, Willy Miranda became a high school coach and went on to win over 450 games across a 42-year tenure?... that a false viral rumour claimed 42 people committed suicide after their homoerotic fan art was included in the film Crazy About One Direction?... that an Arizona TV station put a satellite dish in a vacant swimming pool?... that 42 years after Jilly Cooper's How to Stay Married was first published, she described it as "terribly politically incorrect"?... that wrestler Kurt Howell won all 108 of his matches in high school?... that the second-place candidate in the 2018 Taipei mayoral election lost by just 0.23%, demanded a recount, and ended up losing by even more?ArchiveStart a new articleNominate an articlenosato Daikinosato Daiki (pictured) becomes sumo's 75th yokozuna.In association football, Liverpool win the Premier League title.In motor racing, lex Palou wins the Indianapolis 500.In basketball, the EuroLeague concludes with Fenerbahe winning the Final Four Playoff.Ongoing: Gaza warM23 campaignRussian invasion of UkrainetimelineSudanese civil wartimelineRecent deaths: Phil RobertsonMary K. GaillardPeter DavidAlan YentobGerry ConnollySebastio SalgadoNominate an articleMay 29: Feast day of Saint PaulVI (Catholicism)Headline in the New York Times1233 MongolJin War: The Mongols entered and began looting Kaifeng, the capital of the Jin dynasty of China, after a 13-month siege.1416 A squadron of the Venetian navy captured many Ottoman ships at the Battle of Gallipoli, confirming Venetian naval superiority in the Aegean Sea for the next few decades.1913 During the premiere of the ballet Le Sacre du printemps by Igor Stravinsky at the Thtre des Champs-lyses in Paris, the avant-garde nature of the music and choreography caused a near-riot in the audience (report pictured).1999 Charlotte Perrelli, representing Sweden, won the Eurovision Song Contest, the first edition not to feature an orchestra or live accompaniment.2011 Residents of Portland, Oregon, held a rally called Hands Across Hawthorne in response to an attack against a gay couple holding hands while crossing the Hawthorne Bridge.Benedetto Pistrucci (b.1783)G.K. Chesterton (b.1874)Hubert Opperman (b.1904)Uro Drenovi (d.1944)More anniversaries: May 28May 29May 30ArchiveBy emailList of days of the yearAboutThe Australian white ibis (Threskiornis molucca) is a wading bird of the ibis family, Threskiornithidae. It is widespread across much of Australia, and has a predominantly white plumage with a bare, black head, long downcurved bill, and black legs. While it is closely related to the African sacred ibis, the Australian white ibis is a native Australian bird. Due to its increasing presence in the urban environment and its habit of rummaging in garbage, the species has acquired a variety of colloquial names such as "tip turkey" and "bin chicken". This Australian white ibis was photographed at the Royal Botanic Garden, Sydney.Photograph credit: Charles J. SharpRecently featured: Hell Gate BridgeAnemonoides blandaBluespotted ribbontail rayArchiveMore featured picturesCommunity portal The central hub for editors, with resources, links, tasks, and announcements.Village pump Forum for discussions about Wikipedia itself, including policies and technical issues.Site news Sources of news about Wikipedia and the broader Wikimedia movement.Teahouse Ask basic questions about using or editing Wikipedia.Help desk Ask questions about using or editing Wikipedia.Reference desk Ask research questions about encyclopedic topics.Content portals A unique way to navigate the encyclopedia.Wikipedia is written by volunteer editors and hosted by the Wikimedia Foundation, a non-profit organization that also hosts a range of other volunteer projects: CommonsFree media repository MediaWikiWiki software development Meta-WikiWikimedia project coordination WikibooksFree textbooks and manuals WikidataFree knowledge base WikinewsFree-content news WikiquoteCollection of quotations WikisourceFree-content library WikispeciesDirectory of species WikiversityFree learning tools WikivoyageFree travel guide WiktionaryDictionary and thesaurusThis Wikipedia is written in English. Many other Wikipedias are available; some of the largest are listed below. 1,000,000+ articles DeutschEspaolFranaisItalianoNederlandsPolskiPortugusSvenskaTing Vit 250,000+ articles Bahasa IndonesiaBahasa MelayuBn-lm-gCataletinaDanskEestiEsperantoEuskaraMagyarNorsk bokmlRomnSimple EnglishSloveninaSrpskiSrpskohrvatskiSuomiTrkeOzbekcha 50,000+ articles AsturianuAzrbaycancaBosanskiFryskGaeilgeGalegoHrvatskiKurdLatvieuLietuviNorsk nynorskShqipSlovenina Retrieved from " 2Calendar yearYearsMillennium2ndmillenniumCenturies12thcentury13thcentury 14thcenturyDecades1210s1220s1230s 1240s1250sYears1230123112321233 123412351236vte1233 by topicLeadersPolitical entitiesState leadersReligious leadersBirth and death categoriesBirths DeathsEstablishments and disestablishments categoriesEstablishments DisestablishmentsArt and literature1233 in poetryvte1233 in various calendarsGregorian calendar1233MCCXXXIIIAb urbe condita1986Armenian calendar682 Assyrian calendar5983Balinese saka calendar11541155Bengali calendar639640Berber calendar2183English Regnal year17Hen.318Hen.3Buddhist calendar1777Burmese calendar595Byzantine calendar67416742Chinese calendar (WaterDragon)3930 or 3723to (WaterSnake)3931 or 3724Coptic calendar949950Discordian calendar2399Ethiopian calendar12251226Hebrew calendar49934994Hindu calendars- Vikram Samvat12891290- Shaka Samvat11541155- Kali Yuga43334334Holocene calendar11233Igbo calendar233234Iranian calendar611612Islamic calendar630631Japanese calendarJei 2 / Tenpuku 1()Javanese calendar11421143Julian calendar1233MCCXXXIIIKorean calendar3566Minguo calendar679 before ROC679Nanakshahi calendar235Thai solar calendar17751776Tibetan calendar(male Water-Dragon)1359 or 978 or 206to(female Water-Snake)1360 or 979 or 207 Henry I of Cyprus receives a messageYear 1233 (MCCXXXIII) was a common year starting on Saturday of the Julian calendar.War of the Lombards: Lombard forces at Kyrenia surrender to John of Beirut, after a 10-month siege. The defenders, with their personal belongings, are allowed to retire to Tyre. Captured prisoners are exchanged for those held by Richard Filangieri, commander of the Lombards, at Tyre. Cyprus is wholly restored under the rule of the 16-year-old King Henry I ("the Fat"). His vassals are rewarded, and loans that they have made are repaid.[1]August 20 Oath of Bereg: King Andrew II of Hungary vows to the Holy See that he will not employ Jews and Muslims to administer royal revenues, which causes diplomatic complaints and ecclesiastical censures.[2]Winter Reconquista: King Ferdinand III of Castile ("the Saint") conquers the cities of Trujillo and beda. The Castilian army besieges the city of Peniscola. Ferdinand forces Ibn Hud, ruler of the Taifa of Zaragoza, to sign a truce.[3]August Richard Marshal, 3rd Earl of Pembroke, signs an alliance with Llywelyn the Great, to join forces to revolt against King Henry III. Richard is faced by demands from royal bailiffs in September where the garrison of Usk Castle is forced to surrender.November Henry III's army camped at Grosmont Castle is attacked in the night, by a force of Welsh and English rebels. Several of Henry's supporters are captured, and the castle is returned to Hubert de Burgh, one of the rebels.May 29 MongolJin War: The Mongol army led by gedei Khan captures Kaifeng, capital of the Jin dynasty ('Great Jin'), after the 13-month Siege of Kaifeng (1232). The Mongols plunder the city, while Emperor Aizong of Jin flees for the town of Caizhou. Meanwhile, gedei departs and leaves the final conquest to his favoured general, Subutai.December Siege of Caizhou: The Mongols under gedei Khan besiege Caizhou and ally themselves with the Chinese Song dynasty to eliminate the Jin Dynasty.Gendt receives its city rights from Otto II ("the Lame"), count of Guelders (modern Netherlands).Pope Gregory IX establishes the Papal Inquisition, to regularize the persecution of heresy.June/July Ibn Manzur, Arab lexicographer and writer (d. 1312)August 15 Philip Benizi de Damiani, Italian religious leader (d. 1285)October Al-Nawawi, Syrian scholar, jurist and writer (d. 1277)Adelaide of Burgundy, duchess of Brabant (d. 1273)Choe Ui, Korean military leader and dictator (d. 1258)Ibn al-Quff, Ayyubid physician and surgeon (d. 1286)Sancho of Castile, archbishop of Toledo (d. 1261)January 6 Matilda of Chester, Countess of Huntingdon (or Maud), English noblewoman (b. 1171)January 18 Yang (or Gongsheng), Chinese empress (b. 1162)February 12 Ermengarde de Beaumont, queen of ScotlandMarch 1 Thomas I (or Tommaso), count of Savoy (b. 1178)May Simon of Joinville, French nobleman and knight (b. 1175)June Yolanda de Courtenay, queen consort of HungaryJuly 8 Konoe Motomichi, Japanese nobleman (b. 1160)July 26 Wilbrand of Oldenburg, prince-bishop of UtrechtJuly 27 Ferdinand (or Ferrand), count of Flanders (b. 1188)July 29 Savari de Maulon, French nobleman (b. 1181)July 30 Konrad von Marburg, German priest (b. 1180)October 8 Ugo Canefri, Italian health worker (b. 1148)October 22 Fujiwara no Shunshi, Japanese empress consort (b. 1209)November 22 Helena, duchess of Brunswick-LneburgNovember 27 Shi Miyuan, Chinese politician (b. 1164)Ibn al-Athir, Seljuk historian and biographer (b. 1160)Bertran de Born lo Filhs, French troubadour (b. 1179)Bohemond IV ("the One-Eyed"), prince of Antioch (b. 1175)Gkbri ("Blue-Wolf"), Ayyubid general and ruler (b. 1154)Guilln Prez de Guzmn, Spanish nobleman (b. 1180)John Apokaukos, Byzantine bishop and theologianMathilde of Angoulme, French noblewoman (b. 1181)Sayf al-Din al-Amidi, Ayyubid scholar and jurist (b. 1156)William Comyn, Scoto-Norman nobleman (b. 1163)^ Steven Runciman (1952). A History of The Crusades. Vol III: The Kingdom of Acre, pp. 169170. ISBN978-0-241-29877-0.^ Berend, Nora (2001). At the Gate of Christendom: Jews, Muslims and "Pagans" in Medieval Hungary, c. 1000-c.1300. Cambridge University Press. p.158. ISBN978-0-521-02720-5.^ Lourie, Elena (2004). Jews, Muslims, and Christians in and around the Crown of Aragon: essays in honour of Professor Elena Lourie. Brill. p.270. ISBN90-04-12951-0.[permanent dead link]Retrieved from " 3One hundred years, from 1101 to 1200See also: Renaissance of the 12th centuryMillennia2ndmillenniumCenturies11thcentury12thcentury13thcenturyTimelines11thcentury12thcentury13thcenturyState leaders11thcentury12thcentury13thcenturyDecades1100s1110s1120s1130s1140s1150s1160s1170s1180s1190sCategories:Births Deaths Establishments DisestablishmentsvteEastern Hemisphere at the beginning of the 12th centuryThe 12th century is the period from 1101 to 1200 in accordance with the Julian calendar. In the history of European culture, this period is considered part of the High Middle Ages and overlaps with what is often called the "'Golden Age' of the Cistercians". The Golden Age of Islam experienced significant development, particularly in Islamic Spain.In Song dynasty China, an invasion by Jurchens caused a political schism of north and south. The Khmer Empire of Cambodia flourished during this century, while the Fatimids of Egypt were overtaken by the Ayyubid dynasty. Following the expansions of the Ghaznavids and Ghurid Empire, the Muslim conquests in the Indian subcontinent took place at the end of the century.The Ghurid Empire converted to Islam from Buddhism.1101: In July, the Treaty of Alton is signed between Henry I of England and his older brother Robert, Duke of Normandy in which Robert agrees to recognize Henry as king of England in exchange for a yearly stipend and other concessions. The agreement temporarily ends a crisis in the succession of the Anglo-Norman kings.11011103: David the Builder takes over Kakheti and Hereti (now parts of Georgia).1102: King Coloman unites Hungary and Croatia under the Hungarian Crown.1102: Muslims conquer Seoro de Valencia.11031104: A church council is convened by King David the Builder in Urbnisi to reorganize the Georgian Orthodox Church.1104: In the Battle of Ertsukhi, King David the Builder defeats an army of Seljuks.1104: King Jayawarsa of Kadiri (on Java) ascends to the throne.[citation needed]1106: Battle of Tinchebray.11071111: Sigurd I of Norway becomes the first Norwegian king to embark on a crusade to the Holy Land. He fights in Lisbon and on various Mediterranean isles and helps the King of Jerusalem to take Sidon from the Muslims.1108: By the Treaty of Devol, signed in September, Bohemond I of Antioch has to submit to the Byzantine Empire, becoming the vassal of Alexius I.1109: On June 10, Bertrand of Toulouse captures the County of Tripoli (northern Lebanon/western Syria).1109: In the Battle of Nako, Boleslaus III Wrymouth defeats the Pomeranians and re-establishes Polish access to the sea.1109: On August 24, in the Battle of Hundsfeld, Boleslaus III Wrymouth defeats Emperor Henry V of Germany and stops German expansion eastward.1111: On April 14, during Henry V's first expedition to Rome, he is crowned Holy Roman Emperor.1113: Paramavishnulok is crowned as King Suryavarman II in Cambodia. He expands the Khmer Empire and builds Angkor Wat during the first half of the century. He establishes diplomatic relations with China.1115: The Georgian army occupies Rustavi in the war with the Muslims.1115: In Java, King Kamesvara of Kadiri ascends to the throne. Janggala ceases to exist and comes under Kadiri domination, highly possible under royal marriage. During his reign, Mpu Dharmaja writes Kakawin Smaradahana, a eulogy for the king which become the inspiration for the Panji cycle tales, which spread across Southeast Asia.[1]1116: The Byzantine army defeats the Turks at Philomelion.1116: Death of doa Jimena Daz, governor of Valencia from 1099 to 1102.c. 1119: The Knights Templar are founded to protect Christian pilgrims in Jerusalem.A Black and White Photo of the 12th century Cuenca Cathedral (built from 1182 to 1270) in Cuenca, Spain1120: On January 16, the Council of Nablus, a council of ecclesiastic and secular lords in the crusader Kingdom of Jerusalem, establishes the first written laws for the kingdom.1120: On November 25, William Adelin, the only legitimate son of King Henry I of England, drowns in the White Ship Disaster, leading to a succession crisis which will bring down the Norman monarchy of England.1121: On August 12, in the Battle of Didgori, the greatest military victory in Georgian history, King David the Builder with 45,000 Georgians, 15,000 Kipchak auxiliaries, 500 Alan mercenaries and 100 French Crusaders defeats a much larger Seljuk-led Muslim coalition army.1121: On December 25, St. Norbert and 29 companions make their solemn vows in Premontre, France, establishing the Premonstratensian Order.1122: The Battle of Beroia (Modern-day Stara Zagora, Bulgaria) results in the disappearance of the Pechenegs Turkish tribe as an independent force.1122: On September 23, the Concordat of Worms (Pactum Calixtinum) is drawn up between Emperor Henry V and Pope Calixtus II bringing an end to the first phase of the power struggle between the papacy and the Holy Roman Empire.1122: King David the Builder captures Tbilisi and declares it the capital city of Georgia, ending 400 years of Arab rule.1123: The Jurchen dynasty of China forces Koryo (now Korea) to recognize their suzerainty.1124: In April or May, David I is crowned King of the Scots.1125: On June 11, in the Battle of Azaz, the Crusader states, led by King Baldwin II of Jerusalem, defeat the Seljuk Turks.1125: In November, the Jurchens of the Jin dynasty declare war on the Song dynasty, beginning the JinSong wars.1125: Lothair of Supplinburg, duke of Saxony, is elected Holy Roman Emperor instead of the nearest heir, Frederick of Swabia, beginning the great struggle between Guelphs and Ghibellines.1127: The Northern Song dynasty loses power over northern China to the Jin dynasty.1128: On June 24, the Kingdom of Portugal gains independence from the Kingdom of Len at the Battle of So Mamede; (recognised by Len in 1143).The temple complex of Angkor Wat, built during the reign of Suryavarman II in Cambodia of the Khmer Era.11301180: 50-year drought in what is now the American Southwest.11301138: Papal schism, Pope Innocent II vs. Antipope Anacletus II.1130: On March 26, Sigurd I of Norway dies. A golden era of 95 years comes

to an end for Norway as civil wars between the members of Harald Fairhair's family line rage for the remainder of the century.1130: On Christmas Day, Roger II is crowned King of Sicily, the royal title being bestowed on him by Antipope Anacletus II.1132: The Southern Song dynasty establishes China's first permanent standing navy, although China had a long naval history prior. The main admiral's office is at the port of Dinghai.11321183: the Chinese navy increases from a mere 3,000 to 52,000 marine soldiers stationed in 20 different squadrons. During this time, hundreds of treadmill-operated paddle wheel craft are assembled for the navy to fight the Jin dynasty in the north.1135: King Jayabaya of Kadiri ascends to the throne.[2]11351154: The Anarchy takes place, during a period of civil war in England.1136: Suger begins rebuilding the abbey church at St Denis north of Paris, which is regarded as the first major Gothic building.1137: On July 22, the future King Louis VII of France marries Eleanor, the Duchess of Aquitaine.1138: On October 11, the 1138 Aleppo earthquake devastates much of northern Syria.1139: in April, the Second Lateran Council ends the papal schism.1139: On July 5, in the Treaty of Mignano, Pope Innocent II confirms Roger II as King of Sicily, Duke of Apulia, and Prince of Capua and invests him with his titles.1139: On July 25, the Portuguese defeat the Almoravids led by Ali ibn Yusuf in the Battle of Ourique; Prince Afonso Henriques is acclaimed King of Portugal by his soldiers.Averroes in a 14th-century painting by Andrea di Bonaiuto11401150: Collapse of the Ancestral Puebloan culture at Chaco Canyon (modern-day New Mexico).1141: The Treaty of Shaoxing ends the conflict between the Jin dynasty and Southern Song dynasty, legally establishing the boundaries of the two countries and forcing the Song dynasty to renounce all claims to its former territories north of the Huai River. The treaty reduces the Southern Song into a quasi-tributary state of the Jurchen Jin dynasty.1143: Manuel I Komnenos is crowned as Byzantine emperor after the death of John II Komnenos.1143: Afonso Henriques is proclaimed King of Portugal by the cortes.1143: The Treaty of Zamora recognizes Portuguese independence from the Kingdom of Len. Portugal also recognizes the suzerainty of the pope.1144: On December 24, Edessa falls to the Atabeg Zengi.11451148: The Second Crusade is launched in response to the fall of the County of Edessa.1147: On October 25, the four-month-long Siege of Lisbon successfully brings the city under definitive Portuguese control, expelling the Moorish overlords.1147: A new Berber dynasty, the Almohads, led by Emir Abd al-Mu'min, takes North Africa from the Almoravides and soon invades the Iberian Peninsula. The Almohads began as a religious movement to rid Islam of impurities.1147: The Wendish Crusade against the Polabian Slavs (or "Wends") in what is now northern and eastern Germany.1150: Ramon Berenguer IV, Count of Barcelona marries Petronilla, the Queen of Aragon.1151: The Treaty of Tudiln is signed by Alfonso VII of Len and Raymond Berengar IV, Count of Barcelona, recognizing the Aragonese conquests south of the Jcar and the right to expand in and annex the Kingdom of Murcia.1153: The Treaty of Wallingford, ends the civil war between Empress Matilda and her cousin King Stephen of England fought over the English crown. Stephen acknowledges Matilda's son Henry of Anjou as heir.1153: The First Treaty of Constance is signed between Emperor Frederick I and Pope Eugene III, by the terms of which, the emperor is to prevent any action by Manuel I Comnenus to reestablish the Byzantine Empire on Italian soil and to assist the pope against his enemies in revolt in Rome.1154: the Moroccan-born Muslim geographer Muhammad al-Idrisi publishes his Geography.1154: On December 27, Henry II is crowned King of England at Westminster Abbey.1155: Pope Adrian IV grants overlordship of Ireland to Henry II of England in the bull Laudabiliter.1156: On June 18, the Treaty of Benevento is entered into by Pope Adrian IV and the Norman Kingdom of Sicily. After years of turbulent relations, the popes finally settles down to peace with the Hauteville kings. The kingship of William I is recognized over all Sicily, Apulia, Calabria, Campania, and Capua. The tribute to the pope of 600 schifati agreed upon by Roger II in 1139 at Mignano is affirmed and another 400 shift is added for the new lands.1158: The Treaty of Sahagn ends the war between Castile and Len.The Liuhe Pagoda of Hangzhou, China, 11651161: the Song dynasty Chinese navy, employing gunpowder bombs launched from trebuchets, defeats the enormous Jin dynasty navy in the East China Sea in the Battle of Tangdao and on the Yangtze River in the Battle of Caishi.1161: Kilij Arslan II, Sultan of Rum, makes peace with the Byzantine Empire, recognizing the emperor's primacy.1161: In the siege of Ani, troops from the Kingdom of Georgia take control over the city, only to have it sold for the second time to the Shaddadids, a Kurdish dynasty.1162: Genghis Khan, the founder of the Mongol Empire, is born as Temjin in present-day Mongolia.1163: The Norwegian Law of Succession takes effect.11651182: Tensions and disputes between the Pagan Empire and the Kingdom of Polonnaruwa causes the Sinhalese under Parakramabahu the Great to raid Burma.1168: King Valdemar I of Denmark conquers Arkona on the Island of Rgen, the strongest pagan fortress and temple in northern Europe.1169: Political disputes within the Pandya Empire sparks the decade-long Pandyan Civil War.1169: On May 1, the Norman invasion of Ireland begins. Richard fitzGilbert de Clare ('Strongbow') allies with the exiled Irish chief, Dermot MacMurrough, to help him recover his kingdom of Leinster.The defense of the Carroccio during the battle of Legnano (1176) by Amos Cassioli (18321891)1170: The Treaty of Sahagn is signed by Alfonso VIII of Castile and Alfonso II of Aragon. Based on the terms of the accord, Alfonso VIII agrees to provide Alfonso II with three hostages, to be used as tribute payments owed by Ibn Mardan of Valencia and Murcia.1170: On December 29, Thomas Becket is murdered in Canterbury Cathedral.1171: Saladin deposes the last Fatimid Caliph Al-'id and establishes the Ayyubid dynasty.1171: On November 11, Henry II of England lands in Ireland to assert his claim as Lord of Ireland.1172: The Pandyan city of Madurai is sacked by the Sinhalese army due to an attempt to drive off the rival throne claimant, Kulasekara Pandyan.1173: Sinhalese king Parakramabahu the Great gains a decisive victory by invading the Chola Empire as an ally of the Pandyas in the Pandyan Civil War.1174: On July 12, William I of Scotland is captured by the English in the Battle of Alnwick. He accepts the feudal overlordship of the English crown and pays ceremonial allegiance at York.1175: Hnen Shnin (Genk) founds the Jdo sh (Pure Land) sect of Buddhism.1175: The Treaty of Windsor is signed by King Henry II of England and the High King of Ireland, Ruaidr Ua Conchobair.1176: On May 29, Frederick Barbarossa's forces are defeated in the Battle of Legnano by the Lombard League which results in the emperor's acknowledgment of the pope's sovereignty over the Papal States and Alexander acknowledging the emperor's overlordship of the imperial Church.1176: On September 17, The Battle of Myriokephalon (Myriocephalum; Turkish: Miryakefalon Sava) is fought between the Byzantine Empire and the Seljuk Turks in Phrygia. It is a serious reversal for the Byzantine forces and will be the final, unsuccessful, effort by the Byzantines to recover the interior of Anatolia from the Seljuk Turks.1177: The Treaty or Peace of Venice is signed by the papacy and its allies, and Frederick I, Holy Roman Emperor. The Norman Kingdom of Sicily also participates in negotiations and the treaty thereby determines the political course of all of Italy for the next several years.1178: Chinese writer Zhou Qufei, a Guangzhou customs officer, writes of an island far west in the Indian Ocean (possibly Madagascar), from where people with skin "as black as lacquer" and with frizzy hair were captured and purchased as slaves by Arab merchants.1179: The Treaty of Cazola (Cazorla) is signed by Alfonso II of Aragon and Alfonso VIII of Castile, dividing Andalusia into separate zones of conquest for the two kingdoms, so that the work of the Reconquista would not be stymied by internecine feuding.1180: The Portuguese Navy defeats a Muslim fleet off the coast of Cape Espichel.11801185: the Genpei War in Japan.1181: Parakramabahu the Great conducts a large-scale raid on Burma, after a ship transporting a Sinhalese princess to the Khmer Empire is attacked by Burmese naval fleets.1182: Religious reformations of Theravada Buddhism in Pagan Burma under the patronage of Narapatisithu are continued with the end of the Polonnaruwa-Pagan War.1182: Revolt of the people of Constantinople against the Latins, whom they massacre, proclaiming Andronicus I Comnenus as co-emperor.1183: On January 25, the final Peace of Constance between Frederick Barbarossa, the pope and the Lombard towns is signed, confirming the Peace of Venice of 1177.1183: On September 24, Andronicus I Comnenus has his nephew Alexius II Comnenus strangled.1184: On March 24, Queen Tamar, King of Georgia, accedes to the throne as sole ruler after reigning with her father, George III, for six years.1184: Diet of Pentecost organised by Emperor Frederick I in Mainz.1185: The Uprising of Asen and Peter against the Byzantine Empire leads to the restoration of the Bulgarian Empire.1185: Andronicus I Comnenus is deposed and, on September 12, executed as a result of the Norman massacre of the Greeks of Thessalonika.1185: The cathedral school (Katedralskolan) in Lund, Sweden, is founded. The school is the oldest in northern Europe and one of the oldest in all of Europe.1185: Beginning in this year the Kamakura shogunate deprives the emperor of Japan of political power.1186: On January 27, the future Holy Roman Emperor Henry VI marries Constance of Sicily, the heiress to the Sicilian throne.1187: On July 4, in the Battle of Hattin, Saladin defeats the king of Jerusalem.1187: In August, the Swedish royal and commercial center Sigtuna is attacked by raiders from Karelia, Couronia, and/or Estonia.[3]1188: The Riah were introduced into the Habt and south of Tetouan by the Almohad caliph, Abu Yusuf Yaqub al-Mansur, and Jochem and Acem were introduced in Tamesna.[4]1189: On September 3, Richard I is crowned King of England at Westminster.1189: On November 11, William II of Sicily dies and is succeeded by his illegitimate cousin Tancred, Count of Lecce instead of Constance.11891192: The Third Crusade is an attempt by European leaders to wrest the Holy Land from Saladin.Richard I of England, or Richard the Lionheart.1190: On June 10, Emperor Frederick Barbarossa drowns in the River Salef, leaving the Crusader army under the command of the rivals Philip II of France and Richard I of England, which ultimately leads to the dissolution of the army.1191: Holy Roman Emperor Henry VI attacked the Kingdom of Sicily from May to August but fails and withdrawn, with Empress Constance captured (released 1192).1191: On September 7, Saladin is defeated by Richard I of England at the Battle of Arsuf.1192: In April, Isabella I begins her reign as Christian Queen of the Kingdom of Jerusalem1192: In the Battle of Jaffa, King Richard the Lionheart defeats Saladin.1192: In June, the Treaty of Ramla is signed by Saladin and Richard Lionheart. Under the terms of the agreement, Jerusalem will remain under Muslim control. However, the city will be open to Christian pilgrims. The Latin Kingdom is reduced to a coastal strip that extends from Tyre to Jaffa.1192: Minamoto no Yoritomo is appointed Sei-i Taishgun, "barbarian-subduing great general", shgun for short, the first military dictator to bear this title.1192: Sultan Shahbuddin Muhammad Ghori establishes the first Muslim empire in India for 14 years (11921206) by defeating Prithviraj Chauhan.1193: Nalanda, the great Indian Buddhist educational centre, is destroyed.1194: Emperor Henry VI conquers the Kingdom of Sicily.1195: On June 16, the struggle of Shamqori. Georgian forces annihilate the army of Abu Baqar.1198: The brethren of the Crusader hospital in Acre are raised to a military order of knights, the Teutonic Knights, formally known as the Order of the Knights of the Hospital of St. Mary of the Teutons in Jerusalem.1199: Pope Innocent III writes to Kaloyan, inviting him to unite the Bulgarian Church with the Roman Catholic Church.1200: Construction begins on the Grand Village of the Natchez near Natchez, Mississippi. This ceremonial center for the Natchez people is occupied and built until the early 17th century.[5]Eastern Hemisphere at the end of the 12th centuryChina is under the Northern Song dynasty. Early in the century, Zhang Zeduan paints Along the River During the Qingming Festival. It will later end up in the Palace Museum, Beijing.In southeast Asia, there is conflict between the Khmer Empire and the Champa. Angkor Wat is built under the Hindu king Suryavarman II. By the end of the century, the Buddhist Jayavarman VII becomes the ruler.Japan is in its Heian period. The Chj-jinbutsu-giga is made and attributed to Toba Sj. It ends up at the Kzan-ji, Kyoto.In Oceania, the Tui Tonga Empire expands to a much greater area.Europe undergoes the Renaissance of the 12th century. The blast furnace for the smelting of cast iron is imported from China, appearing around Lapphyttan, Sweden, as early as 1150.Alexander Neckam is the first European to document the mariner's compass, first documented by Shen Kuo during the previous century.Christian humanism becomes a self-conscious philosophical tendency in Europe. Christianity is also introduced to Estonia, Finland, and Karelia.The first medieval universities are founded. Pierre Abelard teaches.Middle English begins to develop, and literacy begins to spread outside the Church throughout Europe.[6] In addition, churchmen are increasingly willing to take on secular roles. By the end of the century, at least a third of England's bishops also act as royal judges in secular matters.[7]The Ars antiqua period in the history of the medieval music of Western Europe begins.The earliest recorded miracle play is performed in Dunstable, England.Gothic architecture and trouvre music begin in France.During the middle of the century, the Cappella Palatina is built in Palermo, Sicily, and the Madrid Skylitzes manuscript illustrates the Synopsis of Histories by John Skylitzes.Fire and plague insurance first become available in Iceland, and the first documented outbreaks of influenza there happens.The medieval state of Serbia is formed by Stefan Nemanja and then continued by the Nemanji dynasty.By the end of the century, both the Capetian dynasty and the House of Anjou are relying primarily on mercenaries in their militaries. Paid soldiers are available year-round, unlike knights who expected certain periods off to maintain their manor lifestyles.[8]In India, Hoysala architecture reaches its peak.In the Middle East, the icon of Theotokos of Vladimir is painted probably in Constantinople. Everything but the faces will later be retouched, and the icon will go to the Tretyakov Gallery of Moscow.The Georgian poet Shota Rustaveli composes his epic poem The Knight in the Panther's Skin.Shahab al-Din Suhrawardi founds his "school of illumination".In North Africa, the kasbah of Marrakesh is built, including the city gate Bab Agnaou and the Koutoubia mosque.In sub-Saharan Africa, Kente cloth is first woven.In France, the first piedfort coins were minted.The city of Tula burns down, marking the end of the Toltec EmpireIn West Africa the Ife Empire is established.See also: Timeline of historic inventions 12th century1104: The Venice Arsenal of Venice, Italy, is founded. It employed some 16,000 people for the mass production of sailing ships in large assembly lines, hundreds of years before the Industrial Revolution.1106: Finished building of Gelati.1107: The Chinese engineer Wu Deren combines the mechanical compass vehicle of the south-pointing chariot with the distance-measuring odometer device.1111: The Chinese Donglin Academy is founded.1165: The Liuhe Pagoda of Hangzhou, China, is built.1170: The Roman Catholic notion of Purgatory is defined.[9]1185: First record of windmills.Wikimedia Commons has media related to 12th century.^ Soekmono, R, Drs., Pengantar Sejarah Kebudayaan Indonesia 2, 2nd ed. Penerbit Kanisius, Yogyakarta, 1973, 5th reprint edition in 1988 p.57^ Britannica, T. Editors of Encyclopaedia (1998, July 20). Kairi. Encyclopedia Britannica.^ Enn Tarvel (2007). Sigtuna hukkumine. Archived 2017-10-11 at the Wayback Machine Haridus, 2007 (7-8), p 3841^ Notice sur les Arabes hilaliens. Ismal Hamet. p.248.^ Francine Weiss and Mark R. Barnes (May 3, 1989). "National Register of Historic Places Registration: Grand Village of the Natchez Site / Fatherland Plantation Site (22-Ad-501)" (pdf). National Park Service. and Accompanying 3 photos, from 1989.(680KB)^ Warren 1961, p.129.^ Warren 1961, p.159.^ Warren 1961, p.60-61.^ Le Goff, Jacques (1986). The Birth of Purgatory. Chicago: University of Chicago Press. ISBN0226470822.Warren, Wilfred Lewis (1961). King John. University of California Press. p.362. ISBN9780520036437. {{cite book}}: ISBN / Date incompatibility (help)Retrieved from " 4The following pages link to 12th century External tools(link counttransclusion countsorted list) See help page for transcluding these entriesShowing 50 items.View (previous 50 | next 50) (20 | 50 | 100 | 250 | 500)Antisemitism in Christianity (links | edit)Catharism (links | edit)List of decades, centuries, and millennia (links | edit)Dialect (links | edit)House of Habsburg (links | edit)House of Hohenzollern (links | edit)History of Mali (links | edit)Maimonides (links | edit)Passport (links | edit)Sumba (links | edit)Taoism (links | edit)Weregild (links | edit)Zanzibar (links | edit)20th century (links | edit)1040 (links | edit)15th century (links | edit)16th century (links | edit)17th century (links | edit)18th century (links | edit)14th century (links | edit)1st century (links | edit)13th century (links | edit)4th century (links | edit)11th century (links | edit)1282 (links | edit)7th century (links | edit)10th century (links | edit)9th century (links | edit)8th century (links | edit)6th century (links | edit)5th century (links | edit)3rd century (links | edit)2nd century (links | edit)4th century BC (links | edit)1st century BC (links | edit)2nd century BC (links | edit)3rd century BC (links | edit)5th century BC (links | edit)6th century BC (links | edit)21st century BC (links | edit)11th century BC (links | edit)1000s (decade) (links | edit)1040s (links | edit)1299 (links | edit)1154 (links | edit)1163 (links | edit)1160s (links | edit)1141 (links | edit)1135 (links | edit)1204 (links | edit)View (previous 50 | next 50) (20 | 50 | 100 | 250 | 500)Retrieved from " WhatLinksHere/12th_century" Introduced in 2023, SwiftData is the latest addition to the range of database framework options in Swift, Apples primary programming language for iOS. Built on top of Core Data, two levels above SQLite, its great for simplifying our persistent stores and it allows us to use declarative code, which is a really useful time-saver. However, for all its flexible functionality, SwiftData framework presents certain challenges for us devs, particularly if were migrating from other database tools. So here, were going to walk you through the basics of SwiftData, so you can jump-start your learning and feel confident whenever you use this amazing framework that simplifies data persistence and makes app development easier without the deep knowledge required to master Core Data. Well, there are lots! But here some of the benefits that will really impact your day-to-day work. You can use SwiftData with minimal code, enabling you divert your creative energies to other aspects of your sprint. You can write the entire model layer (essentially, the business logic bit) of your app with just this one framework. There are no external file formats to worry about. Its got SwiftUI integration baked in. In fact, its pretty much Swift-native. The syntax is easy to get to grips with. So, basically, itll save you a lot of time and remove a lot of the pain-points youll find with other frameworks. Which means you can build clean, robust and reliable apps more quickly. In this article were going to look at how to build a SwiftData model database for a simplified version of a social media Post. For the purpose of this demonstration, well need to persist (preserve data once the program has stopped) with just three objects: form our basic schema: A Post that will have: A title An author A list of likes . A User that will: Have a username Follow other Users . Like objects that: Have a date Are associated with a User. Before SwiftData can persist the objects in our database, well need to create well need to create each object as the model class shown below:: struct Post { var author: User var likes: [Like] var title: String}struct User { var username: String var follows: [User]}struct Like { var user: User var date: Date} Weve created the objects of our schema as Structs, but to make them SwiftData-compliant theyll need to be considered SwiftData Objects. This means making them @Model objects, which is done by simply adding @Model , as you can see below: @Model struct Post { var author: User var likes: [Like] var title: String} @Model struct User { var username: String var follows: [User]}@Model struct Like { var user: User var date: Date} After weve added @Model the compiler will return the following error: @Model requires an initializer be provided for 'Post'@Model requires an initializer be provided for 'User'@Model requires an initializer be provided for 'Like' Dont worry, this is expected behavior. @Model requires all structs have both implicit and explicit initializers associated with them, and we can solve the issue by adding an explicit initializer to our model type: @Model struct Post { var author: User var likes: [Like] var title: String init(author: User, likes: [Like], title: String) { self.author = author self.likes = likes self.title = title }}@Model struct User { var username: String var follows: [User] init(username: String, follows: [User] = []) { self.username = usernameself.follows = follows }}@Model struct Like { var user: User var date: Date init(user: User, date: Date = Date.now) { self.user = user self.date = date }} The @Model we have used is one of the schema macros provided by SwiftData framework, these schema macros add functionality to our model classes without requiring the developer to code complex strategies. Now the SwiftData model classes are ready, its time to prepare our app to store them by making the app a ModelContainer. To do this, simply go to the root of the app and add the line below: @mainstruct SwiftDataApp: App { var body: some Scene { WindowGroup { ContentView() } .modelContainer (for: [Post.self, User.self, Like.self]) }} Great! Now the app is a SwiftData modelContainer making our app a model container, which means it can store SwiftData objects and its a persistent store. As the Swift Macro provides a convenient user interface, we dont really need **to know what the @Model macro adds to our types. However, we can easily check by right-clicking the Macro and choosing expand Macro. Now our data models are configured, lets see how can we use them to store, read and delete objects. Having set up the modelContainer, we can access the modelContext in any view we choose. In this example, well create a view with a button for adding users to our database: import SwiftUIimport SwiftDatastruct ContentView: View {@Environment(\.modelContext) private var modelContext var body: some View { VStack { Button { modelContext.insert(RandomGenerator.user()) } label: { Text("Add user") } } .padding() }}#Preview { ContentView()}class RandomGenerator { static func user() -> User { let usernames = ["Quantum", "Giraffe", "Mystic", "Penguin", "Nebula", "Phoenix", "Zenith", "Whisper", "Radiant", "Tiger", "Lunar", "Cascade", "Celestial", "Breeze", "Ephemeral", "Dragon", "Cosmic", "Echo", "Enigma", "Sparrow"] let index1 = Int.random(in: 0...usernames.count - 1) let index2 = Int.random(in: 0...usernames.count - 1) return User(username: "\(usernames[index1])\(usernames[index2])") }} The most important elements here are the connection to our SwiftData Context, and the insertion of the object into this context. @Environment(\.modelContext) private var modelContext...modelContext.insert(RandomGenerator.user()) We can check the button is working as it should by querying the data, which well cover next. Another of the great things about SwiftData is that we can see all users in our database simply by adding a List to our SwiftUI app view. To do this, we simply modify our ContentView as follows: import SwiftUIimport SwiftDatastruct ContentView: View { @Environment(\.modelContext) private var modelContext @Query var users: [User] var body: some View { VStack { Button { modelContext.insert(RandomGenerator.user()) } label: { Text("Add user") } List(users) { user in Text(user.username) } } .padding() }} By adding just four simple lines of code, were now able to query all users from our data model and get a full breakdown of them. So weve seen how to write and read data from our database. Now, lets take the next natural step and look at how to delete. Well need to make a small change to how we display our List to make it easier to add the onDelete modifier. Again, however, this is simple. We just need to change: List(users) { user in Text(user.username)} to: List { ForEach(users, id: \.self) { user in Text(user.username) }} This will not change the overall functionality of our UI or app, but we can check everything still works by simply running the app. Next well add the onDelete modifier, after which our full List should look like below: List {ForEach(users, id: \.self) { user in Text(user.username) } .onDelete(perform: { offsets in modelContext.delete(users[offsets.first ?? 0]) })} Now we can view, add and delete users in our app: Having mastered the creation of basic SwiftData operations using a User struct , we can now consider more complex examples such as bulk add/delete, search, and sort operations, using all our structures. If we want to make data available to demonstrate, well first need to change our RandomGenerator class, so we can generate more complete and complex sets of data. The new RandomGenerator class well be using for these examples is as follows: class RandomGenerator { static func user(thatFollows follows: [User] = []) -> User { let usernames = ["Quantum", "Giraffe", "Mystic", "Penguin", "Nebula", "Phoenix", "Zenith", "Whisper", "Radiant", "Tiger", "Lunar", "Cascade", "Celestial", "Breeze", "Ephemeral", "Dragon", "Cosmic", "Echo", "Enigma", "Sparrow"] let index1 = Int.random(in: 0...usernames.count - 1) let index2 = Int.random(in: 0...usernames.count - 1) return User(username: "\(usernames[index1])\(usernames[index2])", follows: follows) } static func posts() -> [Post] { var oneHundredUsers: [User] = [] for _ in 1...100 { let otherUsers = Int.random(in: 0...100) var follows: [User] = [] for _ in 0...otherUsers { follows.append(RandomGenerator.user()) } oneHundredUsers.append(RandomGenerator.user(thatFollows: follows)) } let blogpostTitleArray = ["Resilience", "Serendipity", "Thrive", "Wanderlust", "Illuminate", "Empower", "Unleash", "Harmonize", "Catalyst", "Flourish", "Zenith", "Enchant", "Pinnacle", "Catalyst", "Odyssey", "Quench", "Jubilant", "Synergy", "Revitalize", "Traverse"] var fiveHundredPosts: [Post] = [] for _ in 1...500 { let index1 = Int.random(in: 0...blogpostTitleArray.count - 1) let index2 = Int.random(in: 0...blogpostTitleArray.count - 1) fiveHundredPosts.append(Post(author: oneHundredUsers[Int.random(in: 0...oneHundredUsers.count - 1)], likes: RandomGenerator .randomLikes(from: oneHundredUsers), title: "\(blogpostTitleArray[index1]) \(blogpostTitleArray[index2])")) } return fiveHundredPosts } static func randomLikes(from users: [User]) -> [Like] { let amountOfLikes = Int.random(in: 0...users.count - 1) var likes: [Like] = [] for likeIndex in 0...amountOfLikes { likes.append(Like(user: users[likeIndex])) } return likes }} Now, rather than generating single users, we can generate multiple users, each with a list of other users they follow. In addition we can create as many as 500 posts at a time, each with authors, titles and a (random) number of likes. So, weve used our RandomGenerator to generate 500 posts and we now want to add them to our database. We could just add them to our database individually, as we saw earlier: let posts = RandomGenerator.posts()for post in posts { modelContext.insert(post)} Or we can make this process more efficient by creating a transaction object, and simply saving the items to the database after the entire transaction is set. We do this as follows: let posts = RandomGenerator.posts() do { try modelContext.transaction { for post in posts { modelContext.insert(post) } do { try modelContext.save() } catch { // Handle your error here } } } catch { // Handle your error here } While the first method took 12 seconds to complete, the second only took nine seconds. That may seem like a marginal gain, but every little advantage helps when were sprinting towards a target. Deleting follows the exact same process. We just need to replace: modelContext.insert(... with: modelContext.delete(... Our base View, which shows us the Post, author and number of likes, will look like the one below: struct ContentView: View { @Environment(\.modelContext) private var modelContext @Query var posts: [Post] var body: some View { VStack { Button { let posts = RandomGenerator.posts() do { try modelContext.transaction { for post in posts { modelContext.insert(post) } do { try modelContext.save() } catch { // Handle error } } } catch { // Handle error } } label: { Text("Add posts") } List(posts) { post in VStack { Text(post.title).font(.system(size: 24)) HStack { Text(post.author.username) Spacer() Text(post.likes.count.description + " likes") } } } } .padding() }} Now lets look at some examples of queries to sort the data, which will be crucial as our app evolves and we need to extract specific information. To sort by author/user, we can use a SortDescriptor and well need to change our Query as follows: @Query (sort: [SortDescriptor(\Post.author.username)]) var posts: [Post] Now the Posts will be grouped by author/user when they are displayed. Lets say we only wanted to see posts created by the user called CascadeMystic. In this case, we would use a Predicate, as below: @Query (filter: #Predicate { post in post.author.username == "CascadeMystic" })var posts: [Post] Just FYI, we use SortDescriptors to sort by a specific keypath, and Predicates for filtering the objects we want to fetch. To use SwiftData in your app, you need to remember the following two crucial steps: Make the Structs/Classes you want to store conform to @Model. Make your app a Container of your @Models with .modelContainer (for: [MyModel.self]). Once thats done, you can use SwiftData in any View by adding a model context to it:@Environment(\.modelContext) private var modelContext So now, you can read your saved types with a query: @Query var myTypeArray: [MyModel] Add new elements with an insert: modelContext.insert(myElement) And delete them: modelContext.delete(myElement) The more you practice and play around with this, the more youll see the benefits of SwiftData when adding and using local database capabilities in your apps. You can really get creative here, so let your imagination roam!

What is core data. Core data meaning. Core data concepts. Describe core data concepts.