IJIRCCE

e-ISSN: 2320-9801 | p-ISSN: 2320-9798

INTERNATIONAL JOURNAL OF INNOVATIVE RESEARCH

IN COMPUTER & COMMUNICATION ENGINEERING

Volume 10, Issue 4, April 2022

INTERNATIONAL STANDARD SERIAL NUMBER INDIA

Impact Factor: 8.165

International Journal of Innovative Research in Computer and Communication Engineering

| e-ISSN: 2320-9801, p-ISSN: 2320-9798| www.ijircce.com | |Impact Factor: 8.165 |

|| Volume 10, Issue 4, April 2022 ||

| DOI: 10.15680/IJIRCCE.2022.1004005 |

Astrophilia : A Full Stack Android App

Mr. Jugal Gala, Mr. Nimit Bhinde, Mr. Punit Dama, Prof. Mrs. Swati Gajbhiye

Department of Information Technology, Shah & Anchor Kutchhi Engineering College, Mumbai, India

Department of Information Technology, Shah & Anchor Kutchhi Engineering College, Mumbai, India

Department of Information Technology, Shah & Anchor Kutchhi Engineering College, Mumbai, India

Professor, Department of Information Technology, Shah & Anchor Kutchhi Engineering College, Mumbai, India

ABSTRACT: GUIs lie at the crux of this HCI with smartphones. Engineering and designing a GUI requires one to draw from supporting knowledge on both the human and computer side. On the machine side, techniques in computer graphics, OSes, programming languages, and development environments are relevant. On the human side, communication theory, graphic and industrial design disciplines, linguistics, social sciences, cognitive psychology, and human performance are relevant. Rich GUIs open doors to 3D graphics which convey high fidelity information easily. Recent developments in Computer Vision and Augmented Reality can also enable an interactive experience of a real-world environment enhanced by computer-generated perceptual information, which can engage young learners to a greater degree and blur the information boundary between the physical and real world. We shall design and engineer a companion Android application for a physical Lego–like rocket building educational game for kids aged 10–12 years. The app will house a parts list with their subcategories and details based on each kit purchased, the list of rocket building missions to accomplish based on different constraints, the child's completed builds and a news feed periodically updated over the internet; all dynamic. Provisions to add a future augmented reality module will be made. Interactive learning is just what the doctor ordered.

KEYWORDS: component, formatting, style, styling, insert.

I. INTRODUCTION

The Problem Statement Design and engineer a companion Android application for a physical Lego-like rocket building educational game for kids aged 10-12 years. Some of the specific requirements include the following:

Design and deploy a suitable, scalable and economical back-end software infrastructure. Use containerization technologies and topical industry trends to achieve the same, with proper provisions for self hosting in product prototyping and development phase to save costs.
Have seamless cohesion between the application and the Android platform and ecosystem, both for the developer and the end user. The app should have dynamic news feed, user-specific content and a strong code-base for future expansion. 3D model and exploded view. The software licensing model should be profitable for the business moving down the line.
Design and engineer a non-standard front-end GUI for the target demographic. The GUI must be attractive and be rendered natively without the use of unnecessary widgets and toolkits. The app should house the 3D models of the rockets and their parts with an interactive exploded view.
Implement a polished solution using modern techniques in 2D and 3D Computer Graphics. The correct 2D and 3D transformation techniques should be used wherever applicable. The platform specific graphics pipelines must be used. Appropriate solid modelling techniques should be used for 3D models as well. Techniques used must be well documented for future developers to be able to pick up quickly.

II. REVIEW OF LITERATURE

Most programmers know how to make a GUI by dragging and dropping buttons onto the screen or hard coding the interface into their program. However, most programmers overlook the fact that a programming language does not automatically come with a GUI library. When a language is first created, programmers must make a GUI system from scratch. A current day example is OpenGL which does not have a build in GUI. All programmers have are the mouse inputs, keyboard inputs, and the ability to draw images onto the screen. This means that the programmers need to create everything from the components, such as buttons, to the event handler system. There are 2 main methods of programming a GUI system. The methods are retained mode and immediate mode. They both have advantages and disadvantages and have specific circumstances where

| e-ISSN: 2320-9801, p-ISSN: 2320-9798| www.ijircce.com | |Impact Factor: 8.165 |

|| Volume 10, Issue 4, April 2022 ||

| DOI: 10.15680/IJIRCCE.2022.1004005 |

you would use each.[1]
OpenGL is the most widely available graphics programming library, and is used for almost every discipline of computer graphics: research, scientific visualization, entertainment and visual effects, computer-aided design, interactive gaming, and many more. To creating applications using OpenGL, a beginning OpenGL programmer would go from the basics of what's required for OpenGL operation, through geometric modelling and transformations, topics such as lighting, depth buffering, alpha blending, and texture mapping, to advanced topics such as the stencil and accumulation buffers, and display lists.[2]
In the last years triangle meshes have become increasingly popular and are nowadays intensively used in many different areas of computer graphics and geometry processing. In classical CAGD irregular triangle meshes developed into a valuable alternative to traditional spline surfaces, since their conceptual simplicity allows for more flexible and highly efficient processing.[3]
A rotation in R3 about an axis through the origin can be represented by a 3 × 3 orthogonal matrix with determinant 1. However, the matrix representation seems redundant because only four of its nine elements are independent. Also the geometric interpretation of such a matrix is not clear until we carry out several steps of calculation to extract the rotation axis and angle. Furthermore, to compose two rotations, we need to compute the product of the two corresponding matrices, which requires twenty-seven multiplications and eighteen additions. Quaternions are very efficient for analysing situations where rotations in R3 are involved. A quaternion is a 4-tuple, which is a more concise representation than a rotation matrix. Its geometric meaning is also more obvious as the rotation axis and angle can be trivially recovered. Quaternion algebra allows us to easily compose rotations. This is because quaternion composition takes merely sixteen multiplications and twelve additions. [4]

III. COMPARATIVE ANALYSIS

A. The Back-End Dilemma

Every IT solution needs a scalable and flexible back end infrastructure which is tailor made for specific needs. We have listed some of the many design choices we'd need to make.
What kind of a database do we need? Traditional relational databases like MySQL sure work well for tasks like authentication and user login, but object based databases like MongoDB are a much better fit for say stuff like a dynamic news feed.
What kind of a software delivery model do we use and how will we incorporate it in our back-end? The waterfall model is a tried and tested software model used by countless software giants like Redhat, Oracle, etc. but a more modern model like Agile which treats methods of continuous codebase integration and delivery as first class citizen, say using Circle CI/CD, is more suited to a rapidly growing startup. Case on point: Spotify, Office 365 and even modern day Microsoft windows uses CI/CD do make new features available to developers on the insider channel.
What infrastructure do you use? We can not just blindly follow the trend of basing our entire back end infrastructure off Amazon Web Services or Microsoft Azure and dig ourself into a hole of monthly recurring costs even before the business venture is even profitable. We have chosen to exhibit financial prudence as a startup and design a flexible back end using containers which can run exactly the same way on local machines in the prototyping and development phase and continue fostering a culture of self-hosting and using a hybrid clouds to save money and have better control of IT services.

B. The Android OS Architecture and Ecosystem.

We believe that the first step to making an android app is understanding the platform very well so you can leverage it to our advantage.
Android OS is Linux based. But it is very crucial to note that developing software for it is NOTHING like developing software for traditional GNU/Linux systems like Ubuntu and Redhat Enterprise Linux which are so widely used in the corporate world. Your C/C++ libraries are different, you have the Bionic C library on Android, you have a bunch of hardware abstraction layers on android purely because of how diverse the platform is, there are a lot of API levels and we have to target them based on our target demographic, our app pretty much runs in a sandbox throughout its execution life so we will have to use the native hardware interface it provides for performance critical tasks like 3D modelling.
What programming language should we choose? Considering Oracle just lost a lawsuit against Google on their use of JAVA APIs on the Android Platform, it implies Google can continue using Java and natively supporting it on the platform and Google has even committed to the same. It clearly wouldn't be wise to stretch the already tight timelines we will be working on just to learn another programming, even thought the availability of Kotlin developers will improve in the future. Thus Java is the clear choice here.

C. Front-end GUI Design and Engineering

This application will have a very specific design language. In light of this, we feel it is important to choose our technology correctly because going back in time and changing the technology used is not an option after we'd have spent a couple 100s of hours coding it out.

| e-ISSN: 2320-9801, p-ISSN: 2320-9798| www.ijircce.com | |Impact Factor: 8.165 |

|| Volume 10, Issue 4, April 2022 ||

| DOI: 10.15680/IJIRCCE.2022.1004005 |

The application will have 3D models in it with an exploded view which a child can interact with to explore further. This criteria alone means we will have to use lower level graphics APIs. The application aspires to have a video game component in it one day. All the more reason to leverage these graphics APIs extensively. While we are at it we might as well render the entire 2D GUI using OpenGL and use the flexibility it offers to our advantage. We must keep in mind that this app is for children. Adults like consistency, using material design would have been a better choice there. Children like new colourful fancy attractive and interactive interfaces. We thus will be designing and implementing the entire UI from scratch using OpenGL on the Android Canvas. The Java NDK will allow us to have some more lower level access to the hardware.

GUIs can be categorized in 2 ways, based on the way they are rendered. One is Immediate mode, in which every single element are constantly computed and the frame buffer is updated accordingly and the other is retained mode in which very specific parts of the GUI are updated based on the previous events. The former is more useful for highly interactive applications whereas the latter is the GUI we use when we are using a software like a word processor.

Imperative UI This is the most well-known worldview. It includes having a different model/model of the application's UI. This configuration centers around the how instead of the what. A genuine model is XML formats in Android. We plan the gadgets and parts which are then delivered for the client to see and cooperate with.

Declarative UI This example is an arising pattern that permits the engineers to plan the UI in view of the information gotten. This then again centers around the what. This plan worldview utilizes one programming language to make an complete app.

We depicted a system to carry out UIs in a high level, revelatory way. Our methodology depends on isolating the primary, utilitarian, and format parts of a UI . We demonstrated the way that the highlights accessible in useful logic dialects can be taken advantage of to give suitable determinations of these issues. The various leveled design of UIs can be effectively determined as term structures. The related usefulness can be indicated by joining occasion overseers (i.e., capacities) to the components of these term structures. The associations of occasion overseers to the singular gadgets of the UI can be depicted by logic.

D. Modern Techniques in 2D and 3D Computer Graphics

Our first stepping stone would be to understand how non euclidean geometries work. And by that I don't mean just learning how to compute a couple of standard problems, I mean really understanding them so we can leverage them for our task.

We are all aware of tradition euclidean geometry, that's what we generally tend to use, in which the interior angles of triangles add up to 180 degrees, and the shortest distance between two points is always a straight line and two parallel lines never cross. Now as mother nature would have it, the geometry we live in can not exclusively be described by this Euclid's methods. Think about a pair of parallel train tracks, if you were to observe, they get narrower as you go further and converge into a point at the horizon, i.e. infinity. We hope this example is sufficient to illustrate our point that Euclid's geometry has its limitations, and one needs to use non-euclidean geometry when implementing computer graphics, more specifically affine and projective geometry Computer animations require heavy use of Complex numbers and Matrix transformations. However they suffer from the problem of a gimbal lock in 3D space in which an axis of rotation is lost at specific edge cases. This problem is overcome by Hamilton's quaternion number system and it's rotation properties.

How to model 3D objects in 2D space and how to use lighting effects to make them look 3D? The most common methods of 3D modelling are Polygon Mesh Modelling and Non-uniform rational B-spline modelling. While the latter model offers more accuracy, it is also more difficult to implement for complex objects. The rendering of these object models is also a challenge, because we need to use lighting effects to give the illusion of a 3D object on a 2D plane of the smartphone display. The most commonly used method for this is called rasterization which uses projective geometry to project objects on a 2D plane without any advanced optics. The more modern and complex method is ray-tracing which offers higher fidelity results but is more complex to implement.

There are numerous assortments of documents supporting 3D designs, for instance, Wavefront .obj files and .x DirectX documents. Each document type by and large will in general have its own interesting information structure. Each document organization can be gotten to through their particular applications, for example, DirectX records, and Quake.

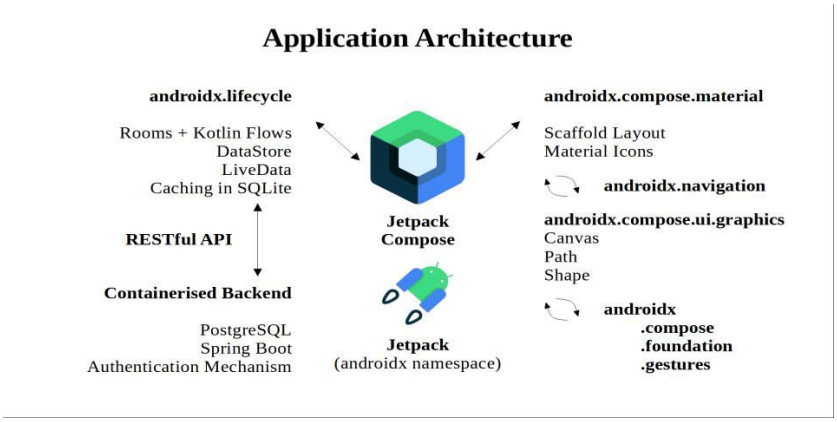
International Journal of Innovative Research in Computer and Communication Engineering

| e-ISSN: 2320-9801, p-ISSN: 2320-9798| www.ijircce.com | |Impact Factor: 8.165 |

|| Volume 10, Issue 4, April 2022 ||

| DOI: 10.15680/IJIRCCE.2022.1004005 |

IV. IMPLEMENTATION



At the heart of this project is Jetpack Compose. Jetpack Compose is Android's modern toolkit for building native UI. It simplifies and accelerates UI development on Android bringing your apps to life with less code, powerful tools, and intuitive Kotlin APIs. It makes building Android UI faster and easier.

A. Containerised Backend

Podman is a daemonless container engine for developing, managing, and running OCI Containers on your Linux System. Containers can either be run as root or in rootless mode. PostgreSQL is a permissively licensed, free and open-source relational database management system emphasizing extensibility and SQL compliance. Ktor is a framework to easily build connected applications – web applications, HTTP services, mobile and browser applications. Modern connected applications need to be asynchronous to provide the best experience to users, and Kotlin coroutines provide awesome facilities to do it in an easy and straightforward way. Ktor provides the Authentication feature to handle authentication and authorization. Typical usage scenarios include logging in users, granting access to specific resources, and securely transmitting information between parties.

B. RESTful API

Instead of exposing our SQL server to the internet andwhipping our own encryption to protect our data, we will use a REST API. Representational state transfer (REST) is a software architectural style that was created to guide the design and development of the architecture for the World Wide Web. REST defines a set of constraints for how the architecture of an Internet-scale distributed hypermedia system, such as the Web, should behave. The REST architectural style emphasises the scalability of interactions between components, uniform interfaces, independent deployment of components, and the creation of a layered architecture to facilitate caching components to reduce userperceived latency, enforce security, and encapsulate legacy systems. The goal of REST is to increase performance, scalability, simplicity, modifiability, visibility, portability, and reliability. This is achieved through following REST principles such as a client–server architecture, statelessness, cacheability, use of a layered system, support for code on demand, and using a uniform interface. These principles must be followed for the system to be classified as RESTful

C. Database Cache using SQLite[5]

We want our app to work offline. Thus we will use a local database cache to ensure the same. Using lifecycle aware components allows us to produce better organized and lighter weight code which is easier to maintain. The Room persistence library provides an abstraction layer over SQLite to allow for more robust database access while harnessing the full power of SQLite. A Kotlin coroutine is a concurrency design pattern that you can use on Android to simplify code that executes asynchronously.
On Android, coroutines help to manage longrunning tasks that might otherwise block the main thread and cause your app to become unresponsive. In Kotlin coroutines, a flow is a type that can emit multiple values sequentially, as opposed to suspend functions that return only a single value. Flows are built on top of coroutines and can provide multiple values. A flow is conceptually a stream of data that can be computed asynchronously. The emitted values must be of the same type. Jetpack DataStore is a data storage solution that allows you to store key-value pairs or typed objects with protocol buffers. DataStore uses Kotlin coroutines and Flow to store data asynchronously, consistently, and transactionally. DataStore is ideal for small , simple datasets and does not support partial updates or referential integrity. In such situations, using Rooms would be

| e-ISSN: 2320-9801, p-ISSN: 2320-9798| www.ijircce.com | |Impact Factor: 8.165 |

|| Volume 10, Issue 4, April 2022 ||

| DOI: 10.15680/IJIRCCE.2022.1004005 |

overkill.
LiveData is an observable data holder class. Unlike a regular observable, LiveData is lifecycle-aware, meaning it respects the lifecycle of other app components, such as activities, fragments, or services. This awareness ensures LiveData only updates app component observers that are in an active lifecycle state. The advantages of using LiveData are that we can ensure our UI matches our data state, no memory leaks, no crashes due to stopped activities and no manual lifecycle handling with always up to date data.

D. User Interface

Jetpack Compose is a modern declarative UI Toolkit for Android. Compose makes it easier to write and maintain your app UI by providing a declarative API that allows you to render your app UI without imperatively mutating frontend views. Historically, an Android view hierarchy has been representable as a tree of UI widgets. As the state of the app changes because of things like user interactions, the UI hierarchy needs to be updated to display the current data. The most common way of updating the UI is to walk the tree using functions like findViewById(), and change nodes by calling methods like button.setText(String), container.addChild(View), or img.setImageBitmap(Bitmap). These methods change the internal state of the widget.
Manipulating views manually increases the likelihood of errors. If a piece of data is rendered in multiple places, it's easy to forget to update one of the views that shows it. It's also easy to create illegal states, when two updates conflict in an unexpected way. For example, an update might try to set a value of a node that was just removed from the UI. In general, the software maintenance complexity grows with the number of views that require updating.
Over the last several years, the entire industry has started shifting to a declarative UI model, which greatly simplifies the engineering associated with building and updating user interfaces. The technique works by conceptually regenerating the entire screen from scratch, then applying only the necessary changes. This approach avoids the complexity of manually updating a stateful view hierarchy. Compose is a declarative UI framework.
One challenge with regenerating the entire screen is that it is potentially expensive, in terms of time, computing power, and battery usage. To mitigate this cost, Compose intelligently chooses which parts of the UI need to be redrawn at any given time. This does have some implications for how you design your UI components.

E. Navigation, Gestures and Graphics[5]

Navigation refers to the interactions that allow users to navigate across, into, and back out from the different pieces of content within your app. Android Jetpack's Navigation component helps you implement navigation, from simple button clicks to more complex patterns, such as app bars and the navigation drawer. The Navigation component also ensures a consistent and predictable user experience by adhering to an established set of principles.
Android provides a variety of APIs to help you create and detect gestures. Although an app should not depend on touch gestures for basic behaviors (since the gestures may not be available to all users in all contexts), adding touch-based interaction to an app can greatly increase its usefulness and appeal. To provide users with a consistent, intuitive experience, our app follows the accepted Android conventions for touch gestures.
Using android's graphics library, we can make drawables and use hardware acceleration to render render advanced graphics.

V. BACKEND METHODOLOGY

A. Spring Boot

Spring Boot is an open-source Java-based structure used to make a miniature Help. It is created by Significant Group and is utilized to construct independent and creation prepared spring applications. Miniature Help is a design that permits the engineers to freely create and convey administrations. Each help running has its own cycle and this accomplishes the lightweight model to help business applications. Spring Boot gives a decent stage to Java engineers to foster an independent and creation grade spring application that you can just run. You can begin with least arrangements without the requirement for a whole Spring design arrangement.
Benefits: Spring Boot offers the accompanying benefits to its designers –
1. Straightforward and foster spring applications.
2. Increments efficiency.
3. Lessens the improvement time.

Objectives: Spring Boot is planned with the accompanying objectives –
1. To stay away from complex XML arrangement in Spring.
2. To foster a creation prepared Spring applications in a simpler manner.
3. To diminish the advancement time and run the application freely.
4. Offer a more straightforward approach to beginning with the application.

| e-ISSN: 2320-9801, p-ISSN: 2320-9798| www.ijircce.com | |Impact Factor: 8.165 |

|| Volume 10, Issue 4, April 2022 ||

| DOI: 10.15680/IJIRCCE.2022.1004005 |

B. Maven

Maven is a form computerization device utilized essentially for Java projects. Maven can likewise be utilized to assemble and oversee projects written in C#, Ruby, Scala, and different dialects. The Maven venture is facilitated by the Apache Programming Establishment, where it was previously important for the Jakarta Undertaking. Maven tends to two parts of building programming: how programming is constructed, and its conditions. Dissimilar to prior instruments like Apache Subterranean insect, it involves shows for the form method. Just exemptions should be determined. A XML document depicts the product project being constructed, its conditions on other outside modules and parts, the form request, indexes, and required modules. It accompanies pre-characterized focuses for playing out specific clear cut errands like arrangement of code and its bundling. Maven progressively downloads Java libraries and Maven modules from at least one vaults like the Maven 2 Focal Storehouse, and stores them in a nearby reserve. This nearby store of downloaded ancient rarities can likewise be refreshed with antiquities made by neighborhood projects. Public vaults can likewise be refreshed. Maven is constructed utilizing a module based engineering that permits it to utilize any application controllable through standard information. A C/C++ local module is kept up with for Maven 2. Elective advancements like Gradle and sbt as fabricate instruments don't depend on XML, however keep the key ideas Maven presented. With Apache Ivy, a devoted reliance chief was created too that additionally upholds Maven repositories.[4]. Apache Maven has support for reproducible forms.
What is Maven: Objective
Maven's motivation is to give engineers:
1. A complete, viable, reusable, and basic model for projects.
2. A bunch of instruments and modules that can interface with the definitive model.

What is Maven: Elements
Maven is stacked with numerous significant and valuable highlights, which goes far towards making sense of its notoriety. Here are a portion of Maven's more vital highlights:
1. A gigantic, persistently developing store of client libraries.
2. The capacity to set up projects effectively, utilizing best practices.

Reliance the board, highlighting programmed refreshing
1. In reverse viable with past forms.
2. Solid mistake and respectability revealing.
3. Programmed parent forming.
4. Guarantees steady use across all activities.
5. It's extensible, and you can without much of a stretch compose modules utilizing prearranging dialects or Java.

Benefits of Maven:
1. Deals with every one of the cycles, like structure, documentation, delivering, and appropriation in project the executives.
2. Improves on the course of task building.
3. Expands the exhibition of the venture and the structure interaction.
4. The undertaking of downloading Container records and different conditions is done consequently.
5. Gives simple admittance to all the expected data.
6. Makes it simple for the engineer to assemble an undertaking in various conditions without stressing over the conditions, processes, and so forth.
7. In Maven, it's not difficult to add new conditions by composing the reliance code in the pom document.

Alternately, Maven has a couple of downsides:
1. Maven requires establishment in the functioning framework and the Maven module for the IDE.
2. Assuming the Maven code for a current reliance is inaccessible, you can't add that reliance utilizing Maven itself.

C. JSON Web Token

What is JSON Web Token?
1. JSON Web Token (JWT) is an open norm (RFC 7519) that characterizes a conservative and independent way for safely sending data between parties as a JSON object. This data can be checked and trusted on the grounds that it is carefully marked. JWTs can be marked utilizing confidential (with the HMAC calculation) or a public/private key pair utilizing RSA or ECDSA.

2. Despite the fact that JWTs can be encoded to likewise give mystery between parties, we will zero in on marked tokens. Marked tokens can check the honesty of the cases held inside it, while encoded tokens conceal those cases from different gatherings. At the point when tokens are marked utilizing public/private key matches, the mark additionally confirms that

| e-ISSN: 2320-9801, p-ISSN: 2320-9798| www.ijircce.com | |Impact Factor: 8.165 |

|| Volume 10, Issue 4, April 2022 ||

| DOI: 10.15680/IJIRCCE.2022.1004005 |

main the party holding the private key is the one that marked it.

The accompanying outline shows how a JWT is acquired and used to get to APIs or assets:

Illustration 2: JWT Authorization

The application or client demands approval to the approval server. This is performed through one of the different approval streams. For instance, a commonplace OpenID Associate consistent web application will go through the/oauth/approve endpoint utilizing the approval code stream. When the approval is conceded, the approval server returns an entrance token to the application. The application utilizes the entrance token to get to a safeguarded asset (like a Programming interface).
Do take note of that with marked tokens, all the data held inside the token is presented to clients or different gatherings, despite the fact that they can't transform it. This implies you shouldn't put restricted intel inside the token.

D. PostgreSQL

PostgreSQL, otherwise called Postgres, is a free and open-source social data set administration framework stressing extensibility and SQL consistence. It was initially named POSTGRES, alluding to its starting points as a replacement to the Ingres information base created at the College of California, Berkeley.
PostgreSQL highlights exchanges with Atomicity, Consistency, Seclusion, Toughness (Corrosive) properties, naturally updatable perspectives, appeared sees, triggers, unfamiliar keys, and put away systems. It is intended to deal with a scope of responsibilities, from single machines to information stockrooms or Web administrations with numerous simultaneous clients. It is the default data set for macOS Server and is additionally accessible for Windows, Linux, FreeBSD, and OpenBSD.

VI. SUMMARY

1. We will build an android application with a sturdy and potent back-end according to the client's needs. The app will have dynamic content which changes on the basis of the user characteristics and in-session behaviour, bringing personalised experience to the user upon login. We aim to deliver an efficient code-base that is modular and can easily be expanded to include features like augmented reality in the future. We will be rendering the GUI directly using OpenGL and avoid using any toolkits/widgets or frameworks. The application is built natively i.e., specifically for one platform (ANDROID), we will take full advantage of all the platform specific hardware, software and development pipelines. The app will be following Google Play's platform policies and will be licensed using the permissive 3-Clause BSD license. The back-end infrastructure will be containerized and will be OCI compliant so it can quickly be deployed using any compliant platform. Infrastructure decisions will be based on topical criteria.

2. Finally the most important thing to note is that we will be using Modern Techniques in 2D and 3D Computer Graphics at the crux of solving our problems in the front end, as discussed in Chapter 3: Comparative Analysis.

3. We will also be following a top-down approach while building our solution. Thus, we have already designed basic application mock-ups in an application design software named InVision Studio for our reference as shown in the Appendix

REFERENCES

1. Feldmeier, "GUI Programming", University of Wisconsin Platteville, 2013. [Online]. Available: http://people.uwplatt.edu/~yangq/CSSE411/csse41 1-materials/f13/feldmeiera-gui %20_programming.doc
2. Angel, D. Shreiner, and V. Shreiner, "An Interactive Introduction to OpenGL Programming," in ACM SIGGRAPH 2007 Courses, San Diego, California, 2007, pp. 1-124, doi: 10.1145/1281500.1281596.
3. M. Botsch et al., "Geometric Modeling Based on Polygonal Meshes," in ACM SIGGRAPH 2007 Courses, San Diego, California, 2007, pp. 1–es, doi: 10.1145/1281500.1281640.
4. Y.-B. Jia, "Quaternions and rotations", Com S 477/577 Notes, pp. 1-11, 2008.
5. Google, "Documentation for App Developers", n.d. [Online]. Available: https://developer. android.com/doc

INNO SPACE
SJIF Scientific Journal Impact Factor

Impact Factor: 8.165

doi cross ref

ISSN INTERNATIONAL STANDARD SERIAL NUMBER INDIA

निस्केयर NISCAIR

INTERNATIONAL JOURNAL OF INNOVATIVE RESEARCH

IN COMPUTER & COMMUNICATION ENGINEERING

9940 572 462 6381 907 438 ijircce@gmail.com

www.ijircce.com

Scan to save the contact details