MOBILE APPLICATION DEVELOPMENT

- COURSE CODE: CS3202
- PROGRAMME: BIT
- SEMESTER :2
- YEAR OF STUDY: 3

MOBILE APPLICATION DEVELOPMENT

COURSE OVERVIEW:

COURSE OVERVIEW

- This course provides an introduction to mobile application development, covering both Android and iOS platforms
- You will learn the fundamentals of mobile app development, including user interface design, programming languages, and development frameworks.

COURSE OBJECTIVES:

1. Understand the basics of mobile application development.
2. Learn how to design user interfaces for mobile applications.
3. Gain proficiency in programming languages and development frameworks for mobile platforms.
4. Develop practical skills through hands-on projects and assignments.
5. Explore best practices for testing, debugging, and deploying mobile applications.

COURSE OUTLINE:

- **Introduction to Mobile App Development**
  - Overview of mobile platforms (Android, iOS)
  - Evolution of mobile applications
  - Understanding mobile app architecture
- **User Interface Design**
  - Principles of mobile UI/UX design
  - Designing interfaces for different screen sizes and resolutions
  - Tools and frameworks for UI design (e.g., Sketch, Adobe XD)

CONT'D

- **Programming Languages and Frameworks**

  - Introduction to Java/Kotlin for Android development
  - Introduction to Swift for iOS development
  - Overview of cross-platform development frameworks (e.g., React Native, Flutter)

- **Building Your First Mobile App**

  - Setting up development environments (Android Studio, Xcode)
  - Creating a simple mobile application
  - Basic app functionalities (e.g., navigation, user input)

CONT'D

- **Working with Data**
  - Storing and retrieving data in mobile apps
  - Working with local databases (SQLite, Core Data)
  - Integrating remote APIs for data retrieval
- **Advanced Topics in Mobile Development**
  - Implementing advanced UI features (e.g., animations, custom views)
  - Handling device sensors and hardware features
  - Integrating location-based services

CONT'D

- **Testing and Debugging**
  - Strategies for testing mobile applications
  - Debugging techniques for identifying and fixing errors
  - Performance optimization for mobile apps
- **Deployment and Distribution**
  - App distribution platforms (Google Play Store, Apple App Store)
  - App signing and submission process
  - Strategies for app monetization and marketing

CONT'D

- **Project Work**
  - Individual or group projects where students apply their knowledge to develop a complete mobile application
  - Project presentations and peer feedback
- **Emerging Trends and Future Directions**
  - Overview of emerging technologies in mobile app development (e.g., AR/VR, wearable devices)
  - Discussion on the future of mobile applications

ASSESSMENT:

- Quizzes and exams to assess theoretical knowledge
- Coding assignments to evaluate practical skills
- Project evaluation based on creativity, functionality, and implementation

PREREQUISITES:

- Basic programming knowledge (preferably in Java, Swift, Python or a similar language)
- Familiarity with software development concepts

- The following is a breakdown of typical prerequisites for a mobile application development course:

PREREQUISITES CONT'D

- **Programming Fundamentals:**
- **Familiarity with Mobile Platforms:**
- **Web Technologies:**
- **Software Development Lifecycle (SDLC):**
- **Version Control Systems:**
- **Database Fundamentals:**

PREREQUISITES CONT'D

- **User Interface (UI) Design:**
- **Problem-Solving Skills:**
- **Communication and Collaboration:**
- **Motivation and Self-Learning:**
- **Recommended Resources:**

PROGRAMMING FUNDAMENTALS:

- Proficiency in a programming language such as Java, Kotlin, Swift, or Objective-C is essential.
- Understanding of basic programming concepts like variables, data types, control structures (loops, conditionals), functions, and object-oriented programming (OOP) principles.

FAMILIARITY WITH MOBILE PLATFORMS:

- Basic knowledge of the target mobile platforms (Android or iOS) and their respective development environments (Android Studio for Android, Xcode for iOS).
- Understanding of the mobile ecosystem, including app stores, device fragmentation, and platform-specific guidelines and best practices

WEB TECHNOLOGIES:

- Familiarity with web development technologies such as HTML, CSS, and JavaScript is beneficial, especially for hybrid or web-based mobile app development approaches.
- Knowledge of web APIs, AJAX, and RESTful services for integrating web-based functionalities into mobile applications.

SOFTWARE DEVELOPMENT LIFECYCLE (SDLC):

- Understanding of software development methodologies such as Agile, Scrum, or Kanban.
- Knowledge of the SDLC phases including requirements gathering, design, development, testing, deployment, and maintenance.

VERSION CONTROL SYSTEMS:

- This covers version control systems like Git for managing source code, collaboration, and tracking changes.
- Understanding of basic Git commands (commit, push, pull, merge) and workflows (branching, tagging).

DATABASE FUNDAMENTALS:

- Basic understanding of relational databases and SQL (Structured Query Language) for data storage and retrieval.
- Knowledge of database design principles, normalization, and basic CRUD operations (Create, Read, Update, Delete).

USER INTERFACE (UI) DESIGN:

- Familiarity with principles of user interface (UI) and user experience (UX) design.
- This can be done with the Knowledge of design tools such as Adobe XD, Sketch, or Figma for creating wireframes, mockups, and prototypes.

PROBLEM-SOLVING SKILLS:

- Strong problem-solving skills and analytical thinking are essential for identifying and resolving issues during development.
- Ability to break down complex problems into smaller, manageable tasks and systematically troubleshoot errors and bugs.

COMMUNICATION AND COLLABORATION:

- Effective communication skills for collaborating with team members, stakeholders, and clients.
- Ability to articulate ideas, provide feedback, and participate in discussions related to project requirements, progress, and decisions.

MOTIVATION AND SELF-LEARNING:

- Motivation to stay updated with the latest trends, technologies, and best practices in mobile app development.
- Willingness to engage in self-directed learning and explore new concepts, frameworks, and tools independently.

RECOMMENDED RESOURCES:

- Online tutorials and documentation for Android and iOS development
- Books on mobile app development
- Community forums and discussion groups for additional support

RECOMMENDED RESOURCES CONT'D

1. B'far, R. (2004). *Mobile computing principles: designing and developing mobile applications with UML and XML*. Cambridge University Press. Ebook - http://content.schweitzer-online.de/static/catalog_manager/live/media_files/representation/zd_std_orig__zd_schw_orig/002/209/645/9780521817332_foreword_pdf_1.pdf

2. Mukherjea, S. (Ed.). (2016). *Mobile Application Development, Usability, and Security*. IGI Global. https://books.google.co.tz/books?hl=en&lr=&id=JchNDQAAQBAJ&oi=fnd&pg=PR1&dq=%22MOBILE+APPLICATION+%22+book&ots=qKVoJrZr2u&sig=sovOLRlD8pHqrSPHjVzcTs34GF0&redir_esc=y#v=onepage&q=%22MOBILE%20APPLICATION%20%22%20book&f=false

3. Velu, V. K. (2016). *Mobile Application Penetration Testing*. Packt Publishing Ltd.

   - https://books.google.co.tz/books?hl=en&lr=&id=Y0XiCwAAQBAJ&oi=fnd&pg=PP1&dq=%22MOBILE+APPLICATION+%22+book&ots=ZnuFs866nU&sig=WTVFj2RPrAgzCQLHphAa_HLTa7k&redir_esc=y#v=onepage&q=%22MOBILE%20APPLICATION%20%22%20book&f=false

INTRODUCTION TO MOBILE APP DEVELOPMENT

- **Mobile platforms** refer to the operating systems on which mobile applications run. The two dominant platforms are Android and iOS.
- **Android:** Developed by Google, **Android** is an open-source platform based on the Linux kernel. It powers a wide range of devices, from smartphones to tablets, smartwatches, and even TVs.
- **iOS:** Developed by Apple, **iOS** is a privately owned platform for Apple devices, including iPhone, iPad, and iPod Touch.
- It's known for its sleek interface, security features, and tight integration with Apple's ecosystem.

EVOLUTION OF MOBILE APPLICATIONS

- **Mobile applications** have evolved significantly since the introduction of the first smartphones.
- **Early mobile apps** were simple, often limited to basic functions like calls, messaging, and email.
- **The advent of app stores** (e.g., Apple App Store, Google Play Store) revolutionized app distribution, enabling developers to reach a global audience.
- **Modern mobile apps** span a wide range of categories, from social networking and entertainment to productivity, gaming, and beyond.

UNDERSTANDING MOBILE APP ARCHITECTURE

- **Mobile app architecture** refers to the high-level structure and organization of an application's components.

KEY COMPONENTS OF MOBILE APP ARCHITECTURE

- **User Interface (UI):** This is the visual elements through which users interact with the app.
- **Business Logic:** This refers to underlying functionality that defines the app's behavior and processes data.
- **Data Storage:** This refers to the mechanisms for storing and retrieving data, such as databases or cloud services.
- **Networking:** This covers the communication with external servers or APIs to fetch data or perform actions.

MOBILE APPS ARCHITECTURAL PATTERNS

- Model-View-Controller (MVC)
- Model-View-Presenter (MVP)
- Model-View-ViewModel (MVVM)
- Clean Architecture

MODEL-VIEW-CONTROLLER (MVC)

- **MVC** is a software architectural pattern used primarily in user interface development.
- It divides an application into three interconnected components: Model, View, and Controller.
- Each component has distinct responsibilities and interacts with the others to achieve separation of concerns and maintainability.

(MVC) CONT'D

- **Model:**
- Represents the data and business logic of the application.
- Manages data manipulation, validation, and storage.
- Does not depend on the user interface (View) or the user's input (Controller).

(MVC) CONT'D

- **View:**
- Represents the presentation layer of the application.
- Displays the data to the user in a particular format (e.g., UI elements, screens, pages).
- Receives user input and sends it to the Controller for processing.

(MVC) CONT'D

- **Controller:**
- Acts as an intermediary between the Model and the View.
- Receives user input from the View and processes it (e.g., user actions, requests).
- Manipulates the Model based on the input received and updates the View accordingly.

MODEL-VIEW-PRESENTER (MVP)

MVP is a variation of the MVC pattern that emphasizes a clear separation between presentation logic and user interface components. By promoting modularity, testability, and reduced coupling, MVP enables efficient development and maintenance of user interfaces in various software applications.

MODEL-VIEW-VIEWMODEL (MVVM)

- **MVVM** is a variation of the MVC and MVP patterns that emphasizes data binding to establish communication between the View and the View Model.
- This is done by promoting separation of concerns, testability, and declarative UI development, MVVM enables efficient development and maintenance of user interfaces in modern app development frameworks.

CLEAN ARCHITECTURE

- **Clean Architecture** provides a complete approach to software design that prioritizes maintainability, testability, and adaptability.
- It defines clear boundaries between layers and enforcing the **Dependency Rule**.
- **Clean Architecture** enables the creation of robust, flexible, and scalable systems that can evolve over time to meet changing requirements and environments.

KEY CONCEPTS IN MOBILE DEVELOPMENT

- Responsive Design: Designing apps to adapt to various screen sizes and orientations, ensuring a consistent user experience across devices.
- Performance Optimization: Techniques for improving app performance, such as optimizing code, reducing memory usage, and minimizing network requests.
- Security: Implementing measures to protect user data and ensure secure communication between the app and external services.

KEY CONCEPTS IN MOBILE DEVELOPMENT CONT'D

- **Accessibility:** Designing apps to be usable by individuals with disabilities, ensuring inclusivity and compliance with accessibility standards.
- **Internationalization and Localization:** Adapting apps for different languages, regions, and cultural preferences to reach a global audience.

CHALLENGES AND OPPORTUNITIES

- Mobile app development presents various challenges, including **fragmentation across devices and platforms**, **compatibility issues,** and **evolving user expectations.**
- However, there are numerous opportunities for **innovation, creativity**, and **reaching a vast and diverse user base**.
- Keeping track of industry **trends**, **adopting best practices**, and **staying adaptable** are all crucial for success in mobile app development.

STEPS TO CREATE A DATABASE USING ALPHA ANYWHERE

1. **Launch Alpha Anywhere**:

   Open Alpha Anywhere on your computer.

2. **Create a New Project**:

   Click on "File" in the menu bar.

   Select "New Project" to create a new project.

3. **Choose Project Type**:

   Select the type of project you want to create. For a database, you might choose "Mobile or Web App with SQL database."

STEPS CONT'D

4. **Set Project Properties**:

   Configure project properties such as name, location, and description.

5. **Create a New SQL Database**:

   In the "Data" section of the Alpha Anywhere workspace, click on "SQL Database Control Panel."

   Click on "Create a new SQL database" or "Connect to an existing SQL database."

   Follow the prompts to set up your new SQL database. You may need to provide connection details such as server address, username, and password

STEPS CONT'D

6. **Design Tables**:

   Once the database is created, you can start designing tables.

   In the SQL Database Control Panel, select your database.

   Click on "Design Tables" to define the structure of your database.

   Add tables and define their fields (columns) along with their data types.

7. **Define Relationships (if necessary)**:

   If your database requires relationships between tables, you can define them in the table designer.

8. **Save Changes**:

   Make sure to save your changes regularly as you design your database.

STEPS CONT'D

9. **Test Database Connection**:

   Before proceeding further, it's a good idea to test the database connection to ensure everything is set up correctly.

10. **Implement Business Logic** (Optional):

    Depending on your project requirements, you may need to implement business logic such as triggers, stored procedures, or validation rules.

    Alpha Anywhere supports various methods for implementing business logic.

STEPS CONT'D

11. **Deploy the Database**:

    Once you've completed designing your database and implementing any necessary business logic, you can deploy it to your desired environment. This might involve deploying it to a server or integrating it into your application.

12. **Test the Database**:

    After deployment, thoroughly test your database to ensure that it functions as expected and meets your requirements.

- By following these steps, you should be able to create a database using Alpha Anywhere. Remember to consult the documentation or seek further assistance if you encounter any difficulties along the way.

CREATING GRID COMPONENT

- Creating a grid component in Alpha Anywhere allows you to display and manipulate data from a database or other data source.
- Below are the steps to create a grid component:

CREATING A GRID COMPONENT

1. **Launch Alpha Anywhere**:

   Open Alpha Anywhere on your computer.

2. **Open or Create a Project**:

   Either open an existing project or create a new one where you want to add the grid component.

3. **Navigate to the Component Palette**:

   In the Alpha Anywhere workspace, locate the Component Palette. It's usually on the left side of the screen.

4. **Drag and Drop the Grid Component**:

Find the Grid Component in the Component Palette.

Click and drag the Grid Component onto the design surface of your form or layout where you want the grid to appear.

CONFIGURING THE GRID PROPERTIES:

5. **Configure the Grid Properties**:
    1. With the grid component selected, you can configure its properties in the Property Grid, usually located on the right side of the screen.
    2. Set the data source for the grid. This could be a table from your SQL database or another data source.
    3. Define the columns you want to display in the grid. You can specify the data source fields to be shown and customize column headers, widths, and other properties.

DEFINING GRID BEHAVIOR:

Define Grid Behavior:

1. To Configure how users interact with the grid. You can set options such as sorting, filtering, grouping, and editing capabilities.
2. Specify any actions or events associated with the grid, such as what happens when a user clicks on a row or selects multiple rows.

GRID CONT'D

1. **Customize Appearance (Optional)**:
    1. If needed, you can customize the appearance of the grid, including colors, fonts, borders, and row highlighting.
2. **Save your Changes**:
    1. Make sure to save your project and form/layout after configuring the grid component.

STEPS TO ENTER DATA TO DATABASE USING A GRID COMPONENT IN ALPHA ANYWHERE

Create or Open a Project:

Open Alpha Anywhere and create a new project or open an existing one where you want to add the grid component for data entry.

Add a Grid Component:

Drag and drop a grid component onto your form or layout where you want to allow data entry.

-

DATA ENTRY USING GRID COMPONENT

Configure Grid Properties:

1. Select the grid component, then configure its properties in the Property Grid.
2. Set the data source of the grid to the table in your database where you want to enter data.
3. Define the columns to display in the grid, corresponding to the fields in your database table.
4. Enable editing capabilities for the grid, allowing users to add, modify, or delete records.

CONT'D

Set Validation Rules (Optional):

1. If necessary, define validation rules for the grid's columns to ensure data integrity.
2. You can specify required fields, data types, ranges, or custom validation scripts.

CONT'D

Configure Save Options:

1. Determine how changes made in the grid will be saved to the database.
2. Alpha Anywhere provides various options for saving data, including automatic saving, manual saving via buttons, or custom scripts.

Add Save Button (Optional):

1. If you choose manual saving, add buttons to your form or layout for users to initiate the saving process.
2. Configure the button's click event to trigger the saving action for the grid component.

TEST DATA ENTRY:

Test Data Entry:

1. Test the grid component to ensure that users can enter data into the database through it.
2. Enter new records, modify existing ones, and delete records if applicable.
3. Verify that validation rules, if any, are enforced correctly.

CREATING USER EXPERIENCES (UX) USING ALPHA ANYWHERE

- Alpha Anywhere is a low-code platform that allows you to build web and mobile applications quickly and easily. When creating user experiences (UX) using Alpha Anywhere, follow the following steps :

1. **Define Requirements**:

Before starting development, gather requirements from stakeholders to understand what the application needs to accomplish and who the target users are.

2. Design the UX:

Use wireframing and prototyping tools to create a visual representation of the user interface. Consider the flow of the application, user interactions, and the overall look and feel.

CONT'D

3. Start a New Project in Alpha Anywhere:

Open Alpha Anywhere and start a new project. You can choose from various templates or start from scratch.

4. Design the Layout:

Use Alpha Anywhere's drag-and-drop interface builder to design the layout of your application. Place components such as buttons, input fields, and navigation elements where you want them.

CONT'D

5. **Add Functionality**: Use Alpha Anywhere's built-in features and scripting capabilities to add functionality to your application. This may include data integration, form validation, and business logic.
6. **Optimize for Mobile**: If you're building a mobile application, make sure to optimize the UX for smaller screens. Alpha Anywhere provides tools for creating responsive designs that adapt to different device sizes.
7. **Test the UX**: Before deploying your application, thoroughly test the UX to ensure that it meets the requirements and provides a smooth user experience. This may involve usability testing, performance testing, and bug fixing.

CONT'D

8. **Iterate Based on Feedback**: Gather feedback from users and stakeholders, and use it to iterate on the UX. Make improvements based on user needs and preferences.
9. **Deploy the Application**: Once you're satisfied with the UX, deploy the application to your desired platform. Alpha Anywhere supports deployment to web servers, mobile app stores, and other environments.
10. **Monitor and Update**: Continuously monitor the application's performance and user feedback, and make updates as needed to maintain a positive UX over time.

ADD A LIST CONTROL TO THE PANEL:

- In the UX Builder, locate the panel where you want to add the list control.
- From the Controls pane, find the "List" control (or any other list-related control such as "List View").
- Drag and drop the list control onto the panel within the UX Builder.

ADDING A BETTER LOOKING LIST

- **Choose the Right Control**:

Alpha Anywhere offers different list controls with varying styles and features. Consider which type of list best suits your application's needs.

Options include List View, Grid View, and Detail View.

- **Customize Appearance**:
- Adjust the properties of the list control to customize its appearance. This includes settings for font size, font color, background color, borders, and padding.
- Use themes and styles to apply consistent visual styling across your application. Alpha Anywhere provides built-in themes and allows you to create custom styles.

CONT'D

- **Add Icons and Images**:
- Incorporate icons or images to enhance the visual presentation of the list items. You can display icons next to each item or include images within the list.
- Utilize Alpha Anywhere's image control to display images stored in your database or hosted externally.

CONT'D

- **Implement Responsive Design**:
- Ensure that your list adapts gracefully to different screen sizes and devices. Use responsive design techniques to make the list layout adjust automatically based on the screen size.
- Test your list on various devices and screen resolutions to ensure it remains readable and usable.

CONT'D

- **Apply Conditional Formatting**:
- Use conditional formatting to highlight specific list items based on predefined criteria.
- For example, you can change the background color of items that meet certain conditions or apply different styles to different types of items.

CONT'D

- **Enable Sorting and Filtering**:
- Implement sorting and filtering functionality to make it easier for users to navigate and find relevant information within the list. Alpha Anywhere provides built-in capabilities for sorting and filtering data.
- **Enhance Interaction**:
- Incorporate interactive features such as hover effects, tooltips, and animations to make the list more engaging and user-friendly.
- Implement click events or touch gestures to allow users to interact with list items and perform actions such as opening details or editing records.

CONT'D

1. **Optimize Performance**:
    1. Ensure that your list control performs efficiently, especially when dealing with large datasets. Use server-side pagination and lazy loading to fetch data incrementally and avoid overwhelming the user interface. Eg default record display list(10, 25, 50, 100 etc)
2. **Test and Iterate**:
    1. Test your enhanced list thoroughly to ensure it meets usability and performance standards. Solicit feedback from users and stakeholders, and iterate on the design based on their input.

SET UP YOUR LIST AND DETAILS PANEL:

- **Set up your list and details panel**:

- Create a list control and a details panel card in your UX. Make sure you have populated your list with data and configured your details panel card to display detailed information based on the selected list item.
- **Add a Selection Changed event to the List control**:
- In the UX Builder, select your list control.
- In the Properties pane, find the Events section.
- Locate the Selection Changed event and double-click it to add an event handler.

CONT'D

- **Create an Action to switch to the details panel card**:
  - Inside the Selection Changed event handler, click the "Add Action" button.
  - Choose the action type "Switch Panel Card" from the list of available actions.
- **Configure the Switch Panel Card action**:
  - In the action settings, select your details panel card as the target panel card to switch to.
  - If necessary, specify any transition effects or animation settings for the switch.

CONT'D

- **Bind the List Control's Selected Item to the Details Panel Card** (Optional):
- To ensure that the details panel card displays information corresponding to the selected list item, you may need to bind the selected item from the list to the details panel card.
- You can do this by setting the Data Source property of the details panel card to the same data source as the list control and configuring the filter or criteria to match the selected item.

CONT'D

- **Test your application**:
- Preview your application and test the functionality by clicking on different rows in the list. Verify that the details panel card updates accordingly and displays the details of the selected item.

ACTION JAVA SCRIPT CONTROL

1. **Add a Button or Control**:
    1. In your UX, add a button or any other control that will trigger the JavaScript action.
2. **Add an Event Handler**:
    1. Double-click on the button/control to open the properties panel.
    2. Find the event that you want to trigger the JavaScript action. Common events include "Click" for buttons.
    3. Double-click the event to create an event handler.
3. **Add a "Run JavaScript" Action**:
    1. Inside the event handler, click the "Add Action" button.
    2. Choose the action type "Run JavaScript" from the list of available actions.

MINUTES 46:50. CLASS 28 MINUTES